O‘ZBEKISTON RESPUBLIKASI OLIY VA O‘RTA MAXSUS TA’LIM VAZIRLIGI

TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI

RAXMANOV QURBON SODIKOVICH

KASIMOVA SHOISTA TASHEVNA

C++ TILIDA DASTURLASH

o‘quv qo‘llanma

Ta’lim sohasi:	300000 - Komputer texnologiyalari va informatika
Ta’lim yonalishlari:	5330500 – Komputer injiniringi (“Komputer injiniringi”, “IT-servis”, “Axborot xavfsizligi”, “Multimediya texnologiyalari”)
	5330600 – Dasturiy injiniring
Ta’lim sohasi:	330000 – Aloqa va axborotlashtirish, telekommunikatsiya texnologiyalari
Ta’lim yonalishlari:	5350100 - Telekommunikatsiya taxnologiyalari ("Telekommunikatsiya", "Teleradiouzatish", "Mobil tizimlar")
	5350200 – Televizion texnologiyalar ("Audiovizual texnologiyalar", "Telestudiya tizimlari va dasturlari")
	5350300 – AKT sohasida iqtisodiyot va menejment
	5350400 - AKT sohasida kasb ta’limi
	5350500 – Pochta aloqasi texnologiyalari
	5350600 – Axborotlashtirish va kutubxonashunoslik

Toshkent -2017

Mualliflar: Raxmanov Q.S., Kasimova Sh.T. “C++ tilida dasturlash” o‘quv qo‘llanma // TATU. Toshkent, 2017. 424 b.

Insoniyat yaqin kunlargacha yetarli axborotga ega emasligidan aziyat chekkan bo‘lsa, hozirgi kunda axborotning ko‘pligidan qiynalmoqda. Zamonaviy axborot vositalari orqali olinadigan ma’lumotlar ko‘pligidan, ba’zan ular tushunib olinmayotir. Ana shunda insonga informatika yordamga kelmoqda.

Hozirgi vaqtda, dasturlash bo‘yicha fanlarning o‘qitilishida asosiy e’tibor talabalarda axborot almashish madaniyatini shakllantirishga, kompyuterning apparat va dasturiy ta’minotini o‘rganishga qaratilishi bilan birga, dasturlash asoslarini o‘rganishga ham qaratilgan.

Shunga ko‘ra, hozirgi vaqtda zamonaviy dasturlash tillari bo‘yicha rus va o‘zbek tillarida o‘quv adabiyotlarini yaratish dolzarb masaladir. Chunki, bunday adabiyotlar etishmasligi, oily o‘quv yurtlarida, o‘quv jarayoniga sezilarli ta’sir o‘tkazmoqda. Shu bilan birga, mavjud rus tilidagi o‘quv adabiyotlari “C++ tilida dasturlash” fani bo‘yicha tasdiqlangan namunaviy o‘quv rejasiga mos kelmaydi. “C++ tilida dasturlash” o‘quv qo‘llanmasi dasturlash asoslarini o‘rgatish bilan birgalikda, ob’ektga yo‘naltirilgan dastulash ko‘nikmalarini hosil qilishga ham qaratilgan.

Qo‘llanma oliy o‘quv yurtlari professor-o‘qituvchilari va talabalari uchun mo‘ljallangan.

Taqrizchilar:

O‘zbekiston Respublikasi Prezidenti huzuridagi Davlat boshqaruvi akademiyasi “Boshqaruvda axborot-kommunikasiya texnologiyalari” kafedrasi mudiri t.f.d., prof. X.N.Zayniddinov.

TATU “Axborot texnologiyalari” kafedrasi professori, t.f.d. M.S.Yakubov

Qo‘llanma Toshkent axborot texnologiyalari universiteti ilmiy kengashi tomonidan tasdiqlangan.

MUNDARIJA

K I R I SH

C++ tili paydo bo‘lgandan beri, o‘qitish, ilmiy izlanish, tijorat dasturiy ta’minotlarini yaratish bo‘yicha amaliy til sifatida tobora ko‘proq e’tiborga ega bo‘lmoqda. Mazkur davr mobaynida C++ tili yangi imkoniyatlar bilan boyib bormoqda, ular yangi kiritilgan funksiyalar(shablonlar va istisnolarga ishlov berish) hisobiga amalga oshirildi.

Dasturlashni yangi o‘rganayotganlarning oldida ko‘pincha: “Qanday boshlasam ekan? Endi nima qilsam ekan?” – degan savollar ko‘ndalang bo‘ladi. Albatta, dasturlash kabi murakkab jarayonni ko‘rsatmalar ko‘rinishidagi qo‘llanma sifatida rasmiylashtirish mumkin emas. Sunga qaramay, dasturlashni qadamma-qadam tasvirlovchi qo‘llanma hamma vaqt ham foydali bo‘ladi. Albatta, talabalar turli dasturlarni yoza olish imkoniyatiga ega bo‘lishi kerak, ammo bunda ular yaxshi ko‘rsatgichlarga ega bo‘la olishlari bo‘ycha ishonch zarur bo‘ladi.

Mazkur o‘quv qo‘llanma C++ tiliga kirish sifatida talqin etilishi mumkin. Ushbu o‘quv qo‘llanma C++ tilida dasturlash bilan birgalikda, o‘z imkoniyatlaridan to‘g‘ri foydalanishni hamda ob’yektga yo‘naltirilgan dasturlash asoslarini ham o‘rgatadi.

Qo‘llanma yaratilishi jarayonida mualliflar o‘z oldilariga uchta maqsadni qo‘ygan edilar. Birinchidan, dasturlashga kirish ortiqcha tavsiflardan holi ravishda bajarilganki, dasturlashni dastlab o‘rganuvchilar qisqa vaqt ichida bunga muvaffaq bo‘ladilar.

Ikkinchidan, qo‘llanma stilini etalon stil bilan birlashtirilishiga harakat qilinganki, har bir bobda keltirilgan ma’lumotlar qisqa va lo‘nda holda keltirilishiga harakat qilingan.

Shuni qayd etish lozimki, har bir bob oxirida keltirilgan misollar, ushbu bobda talqin etilgan nazariy bilimlarni amaliyotda qo‘llash imkoniyatini yaratadi.

Ushbu qollanma C++ tillariga kirish sifatida qaralib, asosiy e’tibor samarali o‘qishga va o‘rganishga qaratilgan. Qo‘llanma turli toifadagi talabalarning qiziqishlarini qamrab olgan bo‘lib, dasturlash bilan bevosita shug‘ullanmoqchi

bo‘lgan va umuman dasturlash tajribasiga ega bo‘lmagan talabalar, tadqiqotchilar uchun foydali boladi.

O‘ylaymizki, qo‘llanma oliy o‘quv yurtlari talabalari, o‘rta maxsus, kasb-hunar ta’limi va akademik litsey o‘quvchilari uchun ahamiyatli bo‘ladi.

I BOB. AXBOROTLAR TIZIMI. MA’LUMOTLARNING TUZILISHI VA ULARNING JAMG‘ARMASI

1.1 Jamiyatni axborotlashtirish

Informatika yangi fan bo‘lishiga qaramasdan uning dastlabki paydo bo‘lish sarchashmasi yetarli darajada qadimiydir.

Informatikani tarixi to‘g‘risidagi masalani qarayotganda, o‘sha vaqtda informatikani fan sifatida qaralmaganini hisobga ogan holda biz axborot almashuvidagi birinchi belgilar va hodisalarga asoslanamiz [5].

Misol. Chexiyada hisob ishlarida (bo‘ri suyagi va tishlari) birinchi ish eramizdan oldingi 30000 yillarda ishlatilgan.

Axborot almashuvida eng muhim va birinchi farq albatta nutq hisoblangan, keyinroq esa birinchi belgili tizimlar (rasmlar, tasvirlar, musiqa, grafika, raqs, udumlar va h.k.). So‘ngra yozuv paydo bo‘ldi: dastlab u daraxt va h.k. iborat rasmli, iyeroglifli shaklda bo‘lgan.

Misol. Qadimgi Misrda eramizdan oldingi 3000 yillarda toshda yozuvlar, iyeroglifli yozuvlar paydo bo‘lgan. So‘ngra papirusda iyeratik (iyeroglif bo‘lmagan) yozuvlar paydo bo‘lgan.

Bronza asri bizga ideogrammalarni, ya’ni qaytariluvchi tushunchalardan iborat tasvirlarni berdi. Bu ideogrammalar eramizgacha bo‘lgan IV asrda rasmli, iyeroglifli yozuvga aylandi.

Turli tizimlar, hisoblar va hisoblash mexanizasiyasi (ma’lumki bular dastlabki avtomatlashtirishdir) rivojlanib bordi.

Misol. Qadimgi Vavilonda eramizgacha bo‘lgan 8000 yillarda turli hisoblash etalonlari (toshli sharlar, konuslar, silindrlar va h.k.) ishlatilgan. U yerda eramizdan oldingi 1800 yillarda o‘n oltilik sanoq tizimini ishlatish boshlangan. Qadimgi rimliklar qo‘l barmoqlari (bu sanoq tizimini barcha simvollarini qo‘l barmoqlari orqali tasvirlash mumkin) orqali eroglif sanoq tizimiga asos soldilar. Barmoqlar

orqali hisoblash yetarlicha uzoq vaqt davom etdi va u bizga butun jahonda ishlatiladigan o‘nlik sanoq tizimini yaratilishiga asos bo‘lib xizmat qildi.

Toshdagi rasmlardan (piktogrammalar) yupqa taxtachalardagi va loydan qilingan sopol plastinkalardan (mixxat) rasmlarga, mixxatdan – bo‘g‘inli (vavilonchasiga) yozuvga, vavilon yozuvidan – grek yozuviga, grek va lotin yozuvidan – asosiy g‘arb yozuvi tizimi yoki punktasion yozuvga o‘tish amalga oshirilgan.

Lotin va grek yozuvi asosida turli bilim sohalari bo‘lmish – matematika, fizika, medisina, ximiya va hokazo sohalar uchun terminologik tizim yaratilgan va yaratilmoqda. Hozirgi kunda turli bilimlarni formallashtirish asosi bo‘lgan matematik (algebraik) til rivojlanmoqda. Matematika belgilar (simvolika) va tillarni tarqalishi umum tabiiy fanlarni rivojlanishiga olib keldi, chunki turli hodisalarni bayon etish va tadqiq qilish uchun adekvat va qulay apparat paydo bo‘ldi.

Misol. Differensiallash, integrallash kabi simvollar paydo bo‘ldi va ularni fizika, ximiya, geologiya va boshqa fanlarda qo‘llash “qurol” sifatida ishlatilmoqda.

Xaritalar, chizmalar, piramidalar, saroylar, mexanizmlar, fraktallar va boshqa ko‘rinishdagi axborotlarni vizuallashtirish tizimi mukamallashtirilmoqda. Misol, piramida ko‘rinishidagi qo‘rg‘onlarni shturm qilish. Yetarlicha murakkab, qadimgi suv uzatish tizimi hozirgi kunda ham ishlab kelmoqda.

Papirusning paydo bo‘lishi bilan axborot hajmi kengaydi (kattalashdi), axborot yangi xossasi bo‘lgan – siqiluvchanligi dolzarb bo‘lib qolmoqda.

Qog‘ozning paydo bo‘lishi axborotni effektiv tashuvchisi bo‘lgan kitob paydo bo‘lishiga olib keldi, chop etuvchi stanokni (Guttenberg) ixtiro etilishi axborotni (axborot almashinuvini yangi xossasi) nusxalashtirishga (tiraj), ko‘paytirishga olib keldi. Yetarli darajada adekvat bo‘lgan ommabop axborot kommunikasiya vositalari (instrumentariyalari) paydo bo‘ldi. Virtual fikrlash (misol uchun, ma’lum rassomlar rasmlarida) elementlari rivojlanmoqda.

Axborotni tarqalishi universitetlar, pochta, kutubxona paydo boʻlishi va rivojlanishiga oʻz hissasini qoʻshmoqda bular esa oʻz navbatida jamiyatda axborot, bilim madaniyatini yigʻish uchun markazlarni paydo boʻlishiga olib keldi.

Misol. Axborotlarni saralashda markazlashgan omborlar paydo boʻladi, masalan Xitoy davlatlarining saroylarida 20 mingga yaqin loydan qilingan sopol taxtalar (tablichka) saqlanmoqda.

Axborotlarni ommaviy nusxalashtirish roʻy bermoqda, kasbiy bilim va axborot texnologiyalarining rivojlanishi keng quloch yoymoqda. Axborotlar va bilimlarni parallel qayta ishlash mexanizimlari paydo boʻldi.

Misol. Axborot xossalarini oʻzgarishi ishlab chiqarish va kommunikasion munosabatlarda oʻz izini qoldirmoqda, masalan, mehnatni taqsimlash (fazo boʻyicha) sodir boʻlmoqda, savdo, kemasozlikka va turli tillarni oʻrganishga zarurat tugʻilmoqda.

Fotografiya, telegraf, telefon, radio, kinomatografiya, televideniya, kompyuterlar, kompyuter tarmoqlari, mobil telefonlarni paydo boʻlishi va ularni yanada rivojlanishi axborot tizimlari va texnologiyalarini ommaviy ravishda tarqalishi va oʻsishiga olib keldi.

Fan sohasida tilli tizimlar sodir boʻlmoqda. Kimyoviy formulalar tili, fizik qonunlar tili, genetik aloqalar tili va boshqalar.

Kompyuterlarni paydo boʻlishi kasbiy bilim dasturlarini saqlash, avtomatlashtirish va foydalanishga imkon berdi: maʼlumotlar bazasi, bilimlar bazasi, ekspert tizimlari va boshqalar paydo boʻldi.

Misol. Shaxsiy kompyuterlar birinchi navbatda bilimlarni avtoformallashtirish vositasi va stimulyatori boʻlib qolmoqda va ular EHMni (faqat dasturchilar) ishlatishi faqat mutaxassislar emas, balki umum foydalanuvchilar uchun ham ishlatilishiga olib kelmoqda.

Yigirmanchi asrning oxirida axborot krizisi, “axborot portlashi” degan tushunchalar paydo boʻldiki, bu tushunchalar ilmiy-texnik nashrlar hajmini juda tezkor oʻsishi oqibatida paydo boʻldi. Axborotlarni qabul qilish, qayta ishlash, umum potokdan va h.k.dan kerakli axborotlarni ajratib olishda katta qiyinchiliklar

paydo bo‘ldi. Bu sharoitda yagona va ruxsatli jahon axborot fazosida, informatikani usul texnologiyalarini rivojlanishida, axborotlarni dolzarblashtirishda metodologiyasi sifatida informatikani rivojlantirish, tayanch texnologiyalar tizimi hosil qilish va jamiyat, fan, texnologiyada informatikani o‘rnini belgilash zarur bo‘lib qoldi.

Hozirgi vaqtda jamiyatni axborotlashtirish rivojlanib bormoqda. Bu vaqtda jamiyatda axborot hajmini kuchlidan o‘sishi, ularni amaliyotda qo‘llanishini tezlashishi, axborotlarni dolzarbligi, haqqoniyligi, turg‘unligi bo‘yicha talablarni ko‘payishiga olib keldi. XXI asrni haqli ravishda “axborotlashtirilgan jamiyat” deb atash mumkinki, bunda yagona va ruxsat etilgan jahon axborot fazosi (maydoni) da ishlab chiqarish kuchlari va ishlab chiqarish munosabatlari ham insoniy shaxs va jamiyat ham doimiy ravishda o‘sib boradi.

Matematika, fizika, boshqarish, texnika, lingvistika, harbiy va boshqa fanlar rivojlanish mahsulida informatikani fan sifatida paydo bo‘lishi axborotlarni yig‘ish, qayta ishlash, uzatish va ishlatishga asoslanadi.

Informatika – fundamental ilmiy va ta’lim sohasi bo‘lib, u hodisalar va tizimlarni chuqur o‘rganish uchun kuchli formal apparatga ega va ularni amaliy instruksiyasi hamda fanlararo aloqasi kuchayishi evaziga muhandislik, foydalanuvchi traktovkasi, prosedurali dasturlash bilan chegaralanib qolmaydi.

Informatika intuitiv (o‘zining tushunchalarida, ta’riflarida, maqsadlarida) bosqichdan allaqachon o‘tib bo‘ldi. Hozirgi kunda u matematika yoki fizika (yoki boshqa fanlar) kabi yetarli darajada “nazariylashdi” va to‘laqonli fundamental tabiiy ilmiy fanga aylandi.

Misol. Informatikani ta’lim distsiplinasi (fan) qatoriga qo‘shish davrida bu fan ko‘proq dasturiy va foydalanuvchi yondoshuvi sifatida ishlatiladi. O‘sha vaqtda informatika prosedurali dasturlash va EHM da masalalarni yechish bilan qanoatlanib qolmadi. Informatika tabiiy ravishda maktab va oliy ta’limda o‘qitilmoqda.

Agar informatika uni qo‘llash va qo‘llanishi nuqtai nazaridan qaralsa, uni texnik, jamiyatni texnologik muhim, ta’minlash vositasi, masalan jamiyatni kommunikasion ehtiyoji deb qarash mumkin.

Agar informatika bilimlar uzatish nuqtai nazaridan qaralsa, u holda uni tabiiy va jamiyatni umummadaniy muhit va vositasi sifatida qarash mumkin. Bu ikkala yondoshuvda o‘zaro bog‘lanish mavjud.

Birinchi yondoshuvni absolyutlashtirish turli texnokrat anglashish va hayolparastlikka olib keladi.

Ikkinchi yondoshuvni absolyutlashtirish esa, ortiqcha formalizm va idiallashtirishga olib kelishi mumkin.

Endi mazkur kurs doirasida informatikani ishchi ta’rifini keltiramiz. Bu ta’rif bo‘lmagan, aniq va formal (bunday ta’rifni berish mumkin emas) bo‘lmagan ta’rif hisoblanmaydi hamda shu bilan birga uni yetarli darajada qabul qilish mumkin.

Ta’rif. Informatika bu – axborot jarayonlari, modellar, algoritmlar va algoritmlash, dasturlar va dasturlash, turlicha algoritmlarni ijro etuvchilar va amalga oshiradigan va ularni bilishda, tabiatda va jamiyatda foydalanish (qo‘llash) haqidagi umumfoydalanishli metodologik fan.

“Informatika” termin (I’informatique) fransuz olimlari tomonidan kiritilgan bo‘lib turli avtomatik vositalar yordamida axborotlarni qayta ishlash (dastlab bu ilmiy-texnik, kutubxona tavsifining axboroti sifatida qaralgan).

Ko‘pincha mamlakatlarda informatika terminini o‘rniga “Computer science” (kompyuterli fan, kompyuter haqidagi fan, to‘g‘rirog‘i kompyuterlar yordamida axborotlarni almashtirish haqidagi fan) termini ishlatiladi.

Informatika o‘ta murakkab, ko‘pqirrali, dinamik bo‘lgani uchun uning predmetini, aniq ta’riflash mumkin emas.

Informatikani uchta asosiy bo‘lagini e’tirof etish mumkin. Nazariy, amaliy va texnik. Ta’kidlab o‘tish lozimki, informatikaning u yoki bu qismlar bo‘yicha fan sifatida va insoniyat faoliyat sifatida bo‘lish qaralayotgan muammo resurslari, masalalar va maqsadlar bilan bog‘langan va u ko‘pincha shartli bo‘ladi.

Nazariy informatika (brown ware, “aqliy” ta’minot) axborot muharriri nazariy muammolarni o‘rganadi.

Amaliy informatika esa (software, “ixcham” dasturiy ta’minot) axborot muhitining amaliy masalalarni o‘rganadi.

Texnik informatika (hardware, “og‘ir” apparat ta’minot) esa axborot muhitining texnik muammolarini o‘rganadi.

Misol. Bankni kredit tavakkalini bashorat qilishni matematik modelini qurish bu nazariy informatika va iqtisodiyotning masalasidir. Bu model bo‘yicha algoritmni bashoratini qurish esa – nazariy informatika masalasi hisoblanadi.

Tavakkalni bashorati uchun kompyuter dasturlarini yaratish esa amaliy informatika masalasidir.

Ko‘pincha soha informatikasi haqida (ayniqsa amaliy informatika sohasida) gap ketganda, masalan, medisina informatikasi, fizika informatikasi, kompyuterli fizika va h.k. tushuniladi.

Misol. Kimyoviy, tibbiy, fizik informatikani predmetlarini aniqlaymiz. Kimyoviy informatikani kimyoviy muhitidagi jarayon va tizimlarni, kimyoviy axborot muhitida boshqarish muammolarini o‘rganadi.

Medisina informatikasi medisina axborot tizimlarida axborot jarayonlari va ularni boshqarish muammolarini o‘rganadi. Fizik informatika (ba’zida kompyuterli fizika deb interpretasiya qilinadi) ochiq fizik tizimlarda axborot jarayonlari, o‘z-o‘zini sodir etish masalalari va tartibni o‘rganadi.

Ixtiyoriy axborot sohasiga, mazkur sohadan tashqari ijtimoiy-huquq, ekologo-iqtisodiy, gumanitar ta’lim va falsafiy aspektlarni o‘z ichiga oladi.

“Informatika” fanini predmet sohasi bu axborot tizimlari, modellar, tillar va ularni yoritish, ularni dolzarblashtirish texnologiyasidir.

Axborot jarayonlari tirik organizmlarda ham, texnik qurilmalarda ham, jamiyatning turli institutlarida, individual va jamiyat anglashida ham sodir bo‘ladi.

Informatika ham matematika kabi turli boshqa fanlar muammolarini yoritish va tadqiq qilish uchun fandir. U o‘zining g‘oyalari, usullari, texnologiyalari

yordamida turli fanlar muammolarini tadqiq qilishda fanlararo aloqalarni mustahkamlashga yo‘l ochadi va uni rivojiga o‘zini hissasini qo‘shadi.

Informatikaning fundamentalligi nafaqat matematika va boshqa tabiiy fanlar formal usullar va vositalarni keng va chuqur qo‘llash bilimi cheklanib qolmasdan, balki, jamiyat bilimini ishlab chiqishda uni olgan natijalari umumiyligi va fundamentalligi, ularni universal metodologik yo‘naltirilgani bilan ham tavsiflanadi.

Xususan informatikani dunyoqarashdagi ro‘li shundan iboratki, u bizni o‘rab turgan dunyodagi, masalan, yashirin, sirtda yotmaydigan tizimlari ham tashqi ham ichki aloqalarni bayon etish va tadqiq etishda, hodisalarni ma’nosini anglashga (ayniqsa, axborotlikda) yordam beradi.

Xususan, informatikani tarbiyaviy ahamiyati shundan iboratki, u tadqiqiy, ijodiy, ishdagi algoritmik yo‘l, tirishqoqlik, chidam va mehnatsevarlik, ehtiyotkorlik, mulohazada mantiqiylik va qat’iylik, muammo ma’nosiga ta’sir etmaydigan ikkinchi darajalilarni inkor etish va asosiylarni ajratib olish mahoratini rivojlantirishlilik, yangi masalalarni qo‘yish va tadqiq qilish, turli masalalarni yechishda axborot texnologiyalarni qo‘llashlik va boshqalardan tashkil topadi.

Xususan, informatikani madaniy ro‘li shundan iboratki, u informatikani funksiyalarga mos ravishda tabiiy ravishda axborot va kompyuter madaniyati kasbiy mahoratni oshirishda va umumiy madaniyatni (fikrlash, xulq, tinglash) oshirishdan iboratdir.

Informatika bilimlarni axborot-mantiqiy tasvirlashning o‘ziga xos madaniyat va san’atidir.

Xususan, informatikani estetik roli shundan iboratki, tadqiq qilinayotgan muammoning tarqoq elementlari va aloqalarini estetik xossaga (chiroyli, xushro‘y, rang, forma, proporsiya, simmetriya, gormoniya, bo‘laklarni yaxlitligi, huzur qilish va h.k.) ega bo‘lgan yagona butun kompozisiyaga keltirish, shuningdek jarayon, hodisalarni estetik qabul qilishni o‘stirish maqsadida ularni qismlarini butunlikka keltirishdan iborat.

Informatika tufayli fanlar tili rivojlanadi, ularni oʻzaro boyligi amalga oshadi. Fanlarni oʻzi ham rivojlanadi.

Informatikani oʻzi ham buning natijasida yangi gʻoyalar va ilovalar bilan boyiydi va bilimlarni olish, saqlash va qoʻllash jarayoni industriyallashadi.

Informatika anʼanaviy, tabiiy-ilmiy sohalar (fizika, biologiya va h.k.) da ham, gumanitar, lingvistik, psixologiya, sosiologiya va h.k. ham keng qoʻllaniladi.

Informatika 60-yillarda Fransiyada elektron hisoblash mashinalari yordamida axborotni qayta ishlash bilan shugʻullanuvchi sohani ifodalovchi atama sifatida yuzaga keldi. Informatika atamasi lotincha informatic soʻzidan kelib chiqqan boʻlib, tushuntirish, xabar qilish bayon etish maʼnolarini anglatadi. Fransuzcha informatique (informatika) soʻzi axborot avtomatikasi yoki axborotni avtomatik qayta ishlash maʼnosini anglatadi. Ingliz tilida soʻzlashuvchi mamlakatlarda bu atamaga Computer science (kompyuter texnikasi haqidagi fan) sinonimi mos keladi.

Oʻzbekistan Respublikasi informatika va hisoblash texnikasi yoʻnalishida jahon darajasidagi ilmiy maktablar yaratganligi, ularda tadqiqotlar muvaffaqiyatli olib borilayotganligi bilan shartli ravishda faxrlana oladi. *"Matematika fanining ehtimollar nazariyasi va matematik statistika, differensial tenglamalar va matematik fizika, funksional tahlil sohasidagi yutuqlari respublikadan ancha uzoqda ham mashhur*" deb yozadi Oʻzbekiston Respublikasi Prizedenti I.A.Karimov [1].

Informatikaning inson faoliyatining mustaqil sohasi sifatida ajralib chiqishi birinchi navbatda kompyuter texnikasining rivojlanishi bilan bogʻliq. Bunda asosiy xizmat mikroprosessor texnikasiga toʻgʻri keladi, uning paydo boʻlishi 70-yillar oʻrtalarida ikkinchi elektron inqilobni boshlab berdi [3].

Shu davrdan boshlab hisoblash mashinalarining element negizini integral chizma va mikroprosessorlar tashkil etdi. Informatika atamasi nafaqat kompyuter texnikasi yutuqlarini aks ettirish va foydalanish, balki axborotni uzatish va qayta ishlash jarayonlari bilan ham bogʻlanadi.

Informatika axborotni qayta ishlash, ularni qo‘llash va ijtimoiy amaliyotning turli sohalariga ta’sirini EHM tizimlariga asoslangan holda ishlab chiqish, loyihalash, yaratish, baholash, ishlashning turli jihatlarini o‘rganuvchi kompleks ilmiy va muhandislik fani sohasidir.

Informatika bu jihatdan axborot modellarini qurishning umumiy metodologik tamoyillarini ishlab chiqishga yo‘naltirilgan. Shu bois axborot uslublari ob’yekt, hodisa, jarayon va hokazolarni axborot modellari yordamida bayon etish imkoniyatiga ega.

Informatikaning vazifalari, imkoniyatlari, vosita va uslublari ko‘p qirrali bo‘lib, uning ko‘plab tushunchalari mavjud. Ularni umumlashtirib quyidagicha talqinni tavsiya etadilar.

Informatika va kibernetika tushunchalarida ko‘pincha chalkashliklar uchrab turadi. Ularning o‘xshashligi va farqini tushuntirishga harakat qilamiz.

N.Vinner tomonidan kibernetikaga kiritilgan asosiy fikr inson faoliyatining turli sohalarida murakkab dinamik tizimlarni boshqarish nazariyasini ishlab chiqish bilan bog‘liq. Kibernetika kompyuterlar mavjudligi yoki yo‘qligidan qat’iy nazar mavjuddir.

Informatika yangi axborotni ancha keng, kibernetika kabi turli Ob’yektlarni boshqarish vazifalarini amaliy hal etmay, o‘zgartirish va barpo etish jarayonlarini o‘rganadi. Shu bois informatika haqida kibernetikadan ancha keng fan sohasi, degan tasavvur hosil bo‘lishi mumkin. Biroq, boshqa jihatdan informatika kompyuter texnikasi bilan bog‘liq bo‘lmagan muammolar yechimi bilan ifodalanmaydi, bu shubhasiz, uning umumlashtiruvchi xususiyatini cheklaydi.

Informatika kompyuter texnikasi rivojlanishi tufayli yuzaga keldi, unga asoslanadi va usiz mavjud bo‘la olmaydi. Kibernetika kompyuter texnikasining barcha yutuqdaridan unumli foydalansa ham, lekin Ob’yektlarni boshqarishning turli modellarini yaratgan holda o‘zicha rivojlanaveradi. Kibernetika va informatika tashkiliy jihatdan bir-biriga juda o‘xshash bo‘lsa ham, lekin:

• informatika — axborot va uni qayta ishlovchi texnikaviy, dasturiy vositalari xususiyatlariga asoslanishi;

• kibernetika esa — Ob’yektlar modellarining konsepsiyalarini ishlab chiqish va ko‘rishda xususan axborotlardan keng foydalanishi jihatidan farqlanadi.

Informatika keng ma’noda insoniyat faoliyatining barcha sohalarida asosan kompyuterlar va telekommunikasiya aloqa vositalari yordamida axborotni qayta ishlashi bilan borliq fan, texnika va ishlab chiqarishning xilma-xil tarmoqlari birligini o‘zida namoyon etadi.

Informatikani tor ma’noda o‘zaro aloqador uch qism — *texnik vositalar (hardware), dasturiy vositalar (software) va algoritmli vositalar (brainware)* sifatida tasavvur etish mumkin.

O‘z navbatida informatikani ham umuman, qismlari bo‘yicha turli jihatlardan: xalq xo‘jaligi tarmog‘i, fundamental fan, amaliy fan sohasi sifatida ko‘rib chiqish mumkin.

Informatika *xalq xo‘jaligi tarmog‘i* sifatida kompyuter texnikasi, dasturiy mahsulotlarni ishlab chiqarish va axborotni qayta ishlash zamonaviy texnologiyasini ishlab chiqish bilan shug‘ullanadigan xo‘jalik yuritishning turli shakllaridagi korxonalarning bir turda jamlanishidan iborat bo‘ladi. Informatikaning ishlab chiqarish tarmog‘i sifatidagi o‘ziga xosligi va ahamiyati shundaki, xalq xo‘jaligining boshqa tarmoqlari mehnat samaradorligi ko‘p jihatdan unga bog‘liqdir. Bundan tashqari, bu tarmoqlar me’yorida rivojlanishi uchun informatikaning o‘zida mehnat samaradorligi ancha yuqori sur’atlarda o‘sib borishi lozim, chunki hozirgi davrda jamiyatda axborot ko‘proq so‘nggi iste’mol predmeti sifatida namoyon bo‘lmoqda: odamlarga dunyoda ro‘y berayotgan voqyealar, ularning kasbiy faoliyatiga doir predmet va hodisalar, fan va jamiyatning rivojlanishi haqida axborot zarur. Mehnat samaradorligining bundan keyingi o‘sishi va farovonlik darajasini ko‘tarish, katta hajmdagi multimedia axborotini (matn, grafika, videotasvir, tovush, animasiya) qabul qilish va ishlashga yangi intellektual vositalar va “inson mashina” interfeyslaridan foydalanish asosidagina erishish mumkin. Informatikada mehnat unumdorligi oshishi sur’atlari yetarli bo‘lmasa, butun xalq xo‘jaligida mehnat samaradorligi o‘sishining anchagina

kamayishi ro‘y berishi mumkin. Hozir dunyodagi barcha ish joylarining 50% ga yaqini axborotni qayta ishlash vositalari bilan ta’minlangan.

Informatika fundamental fan sifatida kompyuter axborot tizimlari negizida istalgan Ob’yektlar bilan boshqaruv jarayonlarini axborot jihatidan ta’minlashni barpo etish metodologiyasini ishlab chiqish bilan shug‘ullanadi. Shunday fikr ham mavjudki, fanning asosiy vazifalaridan biri - axborot tizimlari nima, ular qanday o‘rinni egallaydi, qanday tuzilmaga ega bo‘lishi lozim, qanday ishlaydi, uning uchun qanday qonuniyatlar xos ekanligini aniqlashdir. Yevropada informatika sohasida quyidagi asosiy ilmiy yo‘nalishlarni ajratib ko‘rsatish mumkin: *tarmoq tuzilmasini ishlab chiqish, kompyuterli integrasiyalashgan jarayonni ishlab chiqarish, iqtisodiy va tibbiy informatika, ijtimoiy sug‘urta va atrof muhit informatikasi, professional axborot tizimlari.*

Informatikada fundamental tadqiqotlar maqsadi istalgan axborot tizimlari haqida umumlashtirilgan axborotni olish, ularning qurilishi va ishlashining umumiy qonuniyatlarini aniqlashdir.

Informatika *amaliy fan* sohasi sifatida quyidagilar bilan shug‘ullanadi:

a) axborot jarayonlaridagi qonuniyatlarni o‘rganish (axborotlarni yig‘ish, qayta ishlash, tarqatish);

b) inson faoliyatining turli sohalarida kommunikasion - axborot modellarini yaratish;

c) aniq bir sohalarda axborot tizimi va texnologiyalarini ishlab chiqish va ularning hayotiy bosqichini, ularni ishlab chiqarish, ishlashni va hokazolarni loyihalash, ishlab chiqish bosqichlari uchun tavsiyalar tayyorlash.

Informatikaning bosh vazifasi axborotni yangilash, uslub va vositalarni ishlab chiqish va axborotni qayta ishlashning texnologik jarayonlarini tashkil etish, ulardan foydalanishni ishlab chiqishdir.

Informatikaning asosiy vazifalari quyidagilarni o‘z ichiga oladi:

- istalgan xususiyatdagi axborot jarayonlarini tadqiq etish;
- axborot jarayonlarini tadqiq etishdan olingan natijalar negizida axborotni qayta ishlaydigan axborot tizimini ishlab chiqish va yangi texnologiyani yaratish;

- jamiyat hayotining barcha sohalarida kompyuter texnikasi va texnologiyasidan samarali foydalanishning ilmiy va muxandislik muammolarini yaratish, tadbiq etish va ta’minlashni hal etish.

Informatika o‘z-o‘zicha mavjud bo‘lmay, balki boshqa sohalardagi muammolarni hal etish uchun yangi axborot texnika va texnologiyalarini yaratishga qaratilgan kompleks ilmiy-texnik sohadir. U boshqa sohalar, hatto jarayonlar va hodisalar noformallashuvi tufayli miqdoriy uslublarni qo‘llash mumkin emas deb hisoblanadigan sohalarga ham tadqiqot uslub va vositalarini taqdim etadi. Informatikada kompyuter texnikasi sharofati tufayli amaliy ro‘yobga chiqishi mumkin bo‘lgan matematik modellash uslublarining hal qilinishini alohida ajratib ko‘rsatish lozim. Axborot texnologiyalari rivojlanishining zamonaviy jaxon darajasi shundaki, respublikada jaxon axborot makonining infratuzilmalari va milliy axborot - hisoblash tarmog‘i integrasiyasiga mos keluvchi milliy tizimni yaratish iqtisodiyot, boshqarish, fan va ta’lim samaradorligining muhim omili bo‘lmoqda. Bu muammolar ancha murakkab va ayni paytda respublikamiz uchun dolzarbdir. Hozirda olib borilayotgan iqtisodiy, tuzilmaviy va boshqa o‘zgarishlarni amalga oshirish natijalari respublikada axborotlashtirish bilan borliq muammolarning qanday va qaysi muddatlarda hal etishga ham bog‘liqdir.

1956 yilda akademik M.T.O‘rozboyev tashabbusi bilan O‘zbekiston Fanlar Akademiyasi tarkibida V.I.Romanovskiy nomli Matematika instituti qoshida Hisoblash texnikasi bo‘limi ochilib, unga V.K.Qobulov rahbar etib tayinlanadi va 1958 yilda Respublikamizda ilk bor "Ural-1" toifasidagi EHM o‘rnatiladi.

1966 yilda Markaziy Osiyo mintaqasida O‘zbekiston Respublikasi Fanlar Akademiyasining hisoblash markaziga ega bo‘lgan Kibernetika instituti, 1978 yilda esa uning asosida Kibernetika ilmiy-ishlab chiqarish birlashmasi tashkil etildi. Hozir birlashma tarkibiga quyidagilar kiradi: Kibernetika instituti, Tizimli tadqiqotlar ilmiy-tadqiqot instituti, “Algoritm” xo‘jalik hisobidagi ilmiy-tadqiqot instituti, avtomatlashtirish va hisoblash texnikasi bo‘yicha ihtisoslashtirilgan loyiha-konstruktorlik byurosi, “Biokibernetika” ilmiy-tadqiqot markazi va birlashma tajriba-eksperemental zavodining sanoat-kichik korxonasi.

Xalq xo‘jaligidagi turli vazifalarni hal etishda algoritmlashtirish nazariyasini rivojlantirgan akademik V.K.Qobulov boshchiligidagi birlashmaning yetakchi olimlari O‘zbekistonda kibernetikaning tarkib topishi va rivojlanishiga ulkan hissa qo‘shdilar.

Timsollarni tekshirib bilish va sun’iy intellekt nazariyalar bo‘yicha katta maktab yaratgan akademik M.M.Komilov, matematik modellash va hisoblash eksperimenti, matematik va fizika murakkab vazifalarini hal etishning miqdoriy-tahliliy usullari bo‘yicha muxbir a’zolar F.B.Abutaliyev, B.A.Bondarenko, T.Bo‘riyev, axborotni qayta ishlash bo‘yicha - akademik D.A.Abdullayev, o‘z fanining turli yo‘nalishlari bo‘yicha ulkan ilmiy maktablar o‘zagini yaratgan professorlar T.A.Valiyev, Z.T.Odilova, O.M.Nabiyev, Sh.A.Nazirov, D.N.Ahmedova, R.S.Sadullayeva, Z.M.Solixov, F.T.Odilova, N.A.Mo‘minov va boshqalarning katta xizmatlarini ta’kidlash lozim.

O‘zbekiston Respublikasi mustaqillikka erishgach, birlashma olimlari tomonidan fundamental va amaliy ilmiy yo‘nalishlar belgilandi, O‘zR FA Hay’ati tomonidan Respublikada axborot kommunikatsiya texnologiyalarini rivojlantirish konsepsiyasi ishlab chiqildi va tasdiqlandi hamda takomillashtirilib borilmoqda.

Birlashma olimlarining asosiy vazifasi bozor munosabatlarini hisobga olgan va zamonaviy axborot texnologayalari, tizim va tarmoqlarini qo‘llagan, shuningdek ularni dasturiy ta’minlagan holda ishlab chiqarish, ijtimoiy sohani boshqarish, iqtisodiyotning yirik xalq, xo‘jalik vazifalarini hal etish nazariyasining yangi asoslarini ishlab chiqish va rivojlantirishdir.

Akademik V.K.Qobulov tashabbusi bilan Toshkent davlat iqtisodiyot universiteti (oldingi Toshkent xalq xo‘jaligi instituti) qoshida Iqtisodiy kibernetika fakulteti ochildi. Ushbu fakultet 30 yil mobaynida mamlakatimiz xalq xo‘jaligi uchun kibernetika va informatika sohalari bo‘yicha ko‘plab yuqori malakali mutaxassislarni tayyorlab kelmoqda. Ushbu fakultet qoshida matematik modellar asosida xalq xo‘jaligi muammolarini hal qilish, iqtisodiyotda axborotlar tizimlaridan unumli va oqilona foydalanish, zamonaviy kompyuter texnologiyalarini hayotga keng tadbiq qilish sohalari bo‘yicha akademik

S.S.G‘ulomovning, iqtisodiy kibernetika yo‘nalishi bo‘yicha professor T.Sh.Shodiyevning maktablarini tilga olish joizdir.

Davlat tomonidan tartibga solishning muhimligi va respublikada axborotlashtirish jarayonini tezlashtirish zaruriyatini hisobga olib, O‘zbekiston Respublikasi Vazirlar Mahkamasining 1992 yil 8 dekabr qarori bilan Fan va texnika Davlat Qo‘mitasi (FTDQ qoshida Axborotlashtirish bo‘yicha bosh boshqarma (Bosh axbor) tuzildi. Mazkur qarorda belgilab berilgan asosiy vazifa va faoliyat yo‘nalishlari doirasida O‘zR FTDQ, tashabbusi bilan axborotlashtirish jarayonini rivojlantirishga yo‘naltirilgan bir qator qonunlar qabul qilindi.

Axborotlashirish (1993 yil, may), EHM va ma’lumotlar bazasi uchun dasturlarni huquqiy muhofazalash haqidagi (1994 yil, may) qonunlar shular jumlasidandir.

O‘zR FTDQning Davlat patent idorasida 1995 yil sentyabridan EHM va ma’lumotlar bazasi uchun dasturlarni huquqiy muhofazalash bo‘yicha Agentlik ishlab turibdi. Bu idora dasturiy mahsulotlar, shuningdek to‘liq yoki qisman mulkiy huquqlarni berish shartnomalarini rasmiy ro‘yxatdan o‘tkazadi.

1994 yil dekabrda O‘zbekiston Respublikasi Vazirlar Mahkamasi O‘zbekiston Respublikasining axborotlashtirish konsepsiyasini qabul qildi. Ushbu Konsepsiyaning asosiy maqsadi va unda qo‘yilgan masalalar quyidagilardan iboratdir:

- milliy axborot-hisoblash tarmog‘ini yaratish;
- axborotlarga tovar sifatida yondashishning iqtisodiy, huquqiy va me’yoriy xujjatlarini yuritish;
- axborotlarni qayta ishlashda jahon standartlariga rioya qilish;
- informatika industriyasini yaratish va rivojlantirish;
- axborotlar texnologiyasi sohasidagi fundamental tadqiqotlarni rag‘batlantirish va qo‘llab-quvvatlash;
- informatika vositalaridan foydalanuvchilarni tayyorlash tizimini muvofiqlashtirish.

Konsepsiyaning asosiy qoidalari hisobga olingan “O‘zbekiston Respublikasining axborotlashtirish dasturi” ishlab chiqildi, u uch maqsadli dasturni o‘z ichiga oladi:

a) milliy axborot — hisoblash tarmogi:

b) EHMni matematik va dasturiy ta’minlash;

v) shaxsiy kompyuter.

Axborot texnologiyalarini rivojlantirishning olti ustuvor yo‘nalishi quyidagilardan iborat:

1. Davlat statistika tizimi, kredit, moliya va bank tizimlari.

2. Elektron ma’lumotlar bazasi.

3. Fan-texnika axboroti (FTA) tarmog‘i.

4. Ta’lim, kadrlar tayyorlash va qayta tayyorlash, ijtimoiy muhofaza va sog‘liqni saqlash sohalari axborot tizimlari.

5. Ma’lumotlarni uzatish va aloqa tizimlari.

6. Favqulotda holatlarning oldini olish va xabar berishning axborot tizimlari.

Mazkur dasturda vazirlik va mahkamalar axborot tarmoqlari, Milliy axborot-hisoblash tarmog‘ini yaratish, kompyuterlar va hisoblash texnikasi vositalarini ishlab chiqarishni tashkil etish, yangi axborot texnologiyalari sohasida kadrlar tayyorlashni takomillashtirish, xujjatlashtirishning me’yoriy-uslubiy va huquqiy tizimini yaratish va boshqalar joy olgan.

Maqsadli dasturlar va ustuvor izlanishlar kiritilgan ko‘pgina axborot tizimlari loyihalash va amalga oshirish bosqichida turibdi. Bunday tizimlarga soliq organlari, Vazirlar Mahkamasi, Markaziy bank, Tashqi iqtisodiy faoliyat milliy banki, Tashqi ishlar vazirligi, Makroiqtisodiyot va statistika vazirligi, Davlat mulk qo‘mitasi, Tashqi iqtisodiy aloqalar vazirligi va boshqalarning kompyuter tizimlarini kiritish mumkin. Bir qator yirik loyihalar, jumladan Tashqi iqtisodiy faoliyatni axborot bilan ta’minlashning yagona avtomatlashtirilgan davlat tizimi, Fan-texnika axborotining respublika tarmog‘i, Aholi bandligi xizmatining kompyuter tizimi, Ichki ishlar organlarining yagona axborot tizimi, Adliya vazirligining axborot tizimi va boshqalar ishlab chiqilmoqda.

Milliy axborot hisoblash tarmog‘i davlat aloqa tizimi negizida ishlaydigan va yagona o‘rnatilgan qoidalarga rioya qilish asosida qurilgan davlat va idoraviy xususiyatga ega axborot hisoblash tarmoqlari mujassamlashganligini o‘zida namoyon etuvchi ochiq, tizim sifatida yaratilishi lozim.

Yuqorida ko‘rsatilgan muammolarni bir necha bosqichda hal etish ko‘zda tutilmoqda. *Birinchi bosqichda* milliy tarmoqlar o‘zagini tashkil etuvchi asosiy davlat muassasalari axborot tizimi va tarmoqlarini yaratish ko‘zda tutilgan. Bu tizimlarga asosiy talab: yuqori malakali ekspertizadan o‘tgan loyixalar mavjudligi; yuqori sifatli lisenziyali dasturiy vositalar va zamonaviy kompyuter texnikasi mavjudligi; ochiq tizimlardagi o‘zaro amaldagi xalqaro standart va qoidalardan foydalanilishi.

Ikkinchi bosqichda milliy tarmoqlar o‘zagini davlat tashkilot va muassalarida yaratilgan axborot tizimlari integrasiyasi amalga oshiriladi.

Uchinchi bosqichda tarmoqda boshqa idoralar, konsernlar, assosiasiyalar, ITI, KB, ilmiy markazlar va hokazolar ulanadi.

O‘zbekiston axborot texnologiyalarini tadbiq etish va rivojlantirish uchun ko‘plab intellektual imkoniyat va axborot zaxiralariga ega. Fanlar akademiyasi, oliy va o‘rta maxsus o‘quv yurtlari, ishlab chiqarish korxonalari va firmalarda kompyuter texnikasi, aloqa, dasturiy va axborot ta’minoti, axborot tizimlari bo‘yicha malakali kadrlar ishlamoqda.

“Algoritm”, “Zenit”, “Foton”, “Signal” kabi va boshqa ishlab chiqarish korxonalari respublika sanoat potensialini tashkil etadi, lekin ularning ishlab chiqarish quvvatidan to‘liq foydalanilmayapti.

Respublikadagi kompyuter parki ahvolini tahlil etish shuni ko‘rsatadiki, shaxsiy kompyuterlarni sotish hajmi 1994 yilda 40 ming, 1995 yilda 55 ming, 1996 yilda 70 mingga yetgan, axborot vositalariga ehtiyoj kelgusida ham doimo o‘sib boradi, chunki:

Birinchidan, maktab, kollej, oliy va o‘rta maxsus o‘quv yurtlarini kompyuterlash zarur. Bugun maktablar orasida ayrimlarigina hisoblash texnikasining zamonaviy vositalariga ega, hatto ulardagi jihozlanish darajasini ham

yevropacha namunadagi maktablar bilan qiyoslab boʻlmaydi, zero respublikada aholining yarmidan koʻpi 16 yoshgacha boʻlgan farzandlarimizdir.

Ikkinchidan, axborot tizimi va texnologiyalarini tadbiq etish iqtisodiyot, fan, ta'lim va boshqarishning barcha sohalarida davom etayapti. Axborot vosita va tizimlari, tarmoqlari miqyosining tahlili shuni koʻrsatmoqdaki, kelgusida elektron pullar, bilimlar, xizmatlar va boshqalar ommaviy qoʻllanadi. Jahon axborot makoni bilan integrasiyadagi avtomatlashgan axborot muhiti insoniyat jamlagan bilimlardan, ular axborot muhitiga yetib kelgan vaqt va joydan qatiy nazar, maksimal foydalanish imkonini yaratadi.

Uchinchidan, kompyuter parkini har 5-7 yilda ma'naviy va jismoniy eskirishi tufayli yangilab turish lozimdir.

Yuqorida koʻrsatilgan dalillar hisoblash texnikasi vositalariga ehtiyoj oʻsib borayotganligini koʻrsatadi.

Axborot nima?

Axborot haqida aniq bir ta'rif yoʻq. Lekin unga quyidagicha tushuncha berish mumkin: bizni oʻrab turgan dunyo, tabiat va jamiyatdagi voqyealar, hodisalar va jarayonlar haqidagi xabarlarning toʻplami Axborotdir.

Informatika fani nimani oʻrganadi?

Informatika fani axborotlarning xususiyatlarini, axborotlarning berilish usullarini oʻrganuvchi va axborotlarni toʻplash, jamlash, saqlash, qayta ishlash, texnik va dasturiy vositalar yordamida uzatish qonun, qoidalarini oʻrganadi.

Informatikaning texnik asosi koʻp fanlar bilan, misol uchun, fizika, ximiya, elektronika va radiotexnika fanlari bilan bogʻlangan.

Axborot texnologiyasi nima?

Informatikaning oʻzagini aniq texnik va dasturiy vositalar majmuasi deb qaraluvchi axborot texnologiyasi tashkil etadi. Bu texnologiya yordamida hayotimizdagi va faoliyatimizdagi axborotlarni qayta ishlashda bajariladigan koʻp ishlarni amalga oshiramiz.

Umuman, texnologiya deganda ma'lum bir maqsadga erishish uchun amalga oshiriladigan jarayonlar tizimidan iborat boʻlgan yaratuvchilik faoliyatini

tushuniladi (texno- lotincha so‘z- xunar, san’at, logi- fan).

Texnologiyani tashkil etuvchi tizim jarayonlarida va ular orasidagi axborot almashinuvini tashkil etishda kompyuterlardan foydalanish mazkur texnologiyaning samaradorligini oshiradi. Bu texnologiyaga kompyuterni qo‘llash uchun mazkur texnologiyada axborotlar almashinuvini, jarayonlar tizimini boshqarishning axborot ta’minotini tahlil etish zarur. Shularga qarab bu texnologiyaning unga mos axborotli modeli tuziladi. Ushbu modelda texnologiyaning bajarilishi jarayonida yuz beradigan hamma axborot almashinuvlari, axborotlar ustida bajariladigan amallar, axborot manbalari va jarayonda qatnashadiganlarning imkoniyatlari to‘liq aks ettiriladi. Bu modelning algoritmini tuzib shu asosda texnologiyani bajaruvchi dastur tuziladi. Texnologiyaning ana shu dasturli variantini yangi axborot texnologiyasi deb yuritiladi.

Shunday qilib, axborotlarni qabul qilish, qayta ishlash va yangi axborotni yaratish bilan shug‘ullanuvchi texnologiyalarni kompyuter asosida joriy etish yangi axborot texnologiyalarini vujudga keltiradi.

Shunday qilib, YaAT deganda qandaydir yaratish faoliyatini amalga oshiruvchi kompyuter va unda joriy etilgan dasturiy ta’minot nazarda tutiladi.

Axborot texnologiyasida markaziy o‘rinda kompyuter turadi. U axborotlarni qayta ishlash uchun texnik vosita bo‘ladi.

Har bir o‘qimishli inson asosiy fanlarning asosini bilganidek axborot texnologiyasining asosini ham bilishi kerak.

Zamonaviy mutaxassis quyidagi narsalardan xabardor bo‘lishi va amaliy ishlarda ulardan foydalanish yo‘llarini bilishlari kerak:

- Dasturiy va texnik vositalarning asosiy ishlash tamoyillari va kompyuter tizimlarida berilganlarni tashkil qilish yo‘llarini;
- Shaxsiy kompyuterda ishlashni;
- Zamonaviy axborot texnologiyasining asosiy elementlarini, matnli xujjatlarni qayta ishlashni, ma’lumotlar jamg‘armasi bilan ishlashning tamoyillarini;

- Hozirgi jamiyatda turli sohalarda kompyuterni qo‘llash yo‘nalishlarini.

Zamonaviy kompyuterlarning yaratilishi axborotlarni qayta ishlash va axborotlarga qarashni tubdan o‘zgartirib yubordi.

Axborot texnologiyasi bu umuminsoniy madaniyatning ajralmas bir qismidir. Yuqorida axborot texnologiyasiga berilgan ta’rifdan ko‘rinib turibdiki, u ikki qismdan - texnik asbob-uskuna ta’minoti va dasturiy ta’minotdan iborat.

Texnik asbob-uskuna ta’minoti kompyuter va unga tegishli bo‘lgan qo‘shimcha asbob-uskunalardan tashkil topadi. Hozirgi kompyuterlarning qo‘shimcha qurilmalari juda ko‘pki, ular yordamida axborotlarni kompyuter xotirasiga kiritish va xotiradagi axborotlarni tashqi har xil ma’lumot tashuvchilarga chiqarish ishlari qulay va oson bajariladi.

Bularga har xil asbob - uskunalar, misol uchun, skanerlar, “sichqoncha”, planshet, CD-ROM, grafopostroitellar, ovoz berish qurilmalari, musiqa platalari va boshqalar kiradi.

Hozirgi zamon kompyuterlari maxalliy, regional va jaxon tarmoqlari tizimi bilan ishlay oladi. Bular uchun maxsus tarmoq va kommunikasion asbob-uskunalar (tarmoq platalari, modemlar, adapterlar faksmodem va h.z.) kerak bo‘ladi.

CD-ROM egiluvchan va qattiq disk-vinchesterga o‘xshash ma’lumot tashuvchi kompakt-disk bo‘lib, katta hajmga ega. Uning hajmi 650 Mbayt bo‘lib, o‘rtacha vinchesterning hajmiga tengdir. Uning disketlardan farqi shuki, ma’lumotlar unga uni ishlab chiqaradigan korxonada yozib qo‘yiladi. Undan faqat ma’lumotlarni o‘qish mumkin.

Xuddi shunga o‘xshash multimedia haqida gapiradigan bo‘lsak (tarjimasi ko‘p muhitli ma’lumot tashuvchilar) u maxsus texnologiya - texnik va dasturiy vositalardan iborat bo‘lib, ovoz, harakatlar, xatto videofilmlarni ko‘rsatishi mumkin.

Kompyuter tarmoqlaridan biri bu elektron (E-mail) pochtadir. Unga ulangan har bir foydalanuvchi uchun maxsus elektron manzil beriladi. Xuddi oddiy pochtaga o‘xshash shu tarmoqdagi kompyuterlar bir biriga ma’lumotlarni yuboraverishadi.

Kompyuter texnologiyasida hozirgi paytda yangi yo‘nalish har xil kommunikasiya vositalari yordamida butun dunyo global tarmog‘i - INTERNET tizimining yaratilishidir. Bu butun dunyo axborot tizimi bo‘lib, ko‘p davlatlar shu tizimga ulangan.

INTERNET tarmog‘iga

1981 yilda 213 ta,

1983 yilda 562 ta,

1986 yilda 5089 ta,

1992 yilda 727000 ta,

1995 yilda 20-40 million,

2015 yilda **3.175 mlrd.** fuqaro internetga ulangan [17].

Mamlakatimiz rivojlangan davlatlar qatoridan mustahkam o‘rin egallashi uchun zamonaviy axborot (kompyuter) texnologiyalarini hayotimizning barcha jabhalariga keng joriy etish zarur. Buning uchun, birinchidan, zamonaviy axborot texnologiyalarini rivojlantirish, davlat muassasalari va xo‘jalik subyektlari, muassasa va tashkilotlar, xususiy shaxslar uchun axborot xizmatini yo‘lga qo‘yish. Ikkinchidan, ilm, fan, ta’lim, texnika, iqtisodiyot, ijtimoiy, xalq xo‘jaligi va uni boshqarish sohalarida axborot tizimlarini shakllantirish. Uchinchidan, respublikaning jaxon axborot tizimlari va xalqaro tarmoqlarga ulanishini ta’minlash kerak.

Muhokama savollari

1. Informatika fanining vujudga kelishi tarixi va hozirgi kundagi ro‘li.

2. O‘zbekiston Respublikasi Prezidenti Islom Karimovning O‘zbekistonning 2002-2010 yillari “Kompyuterlashtirish va axborot kommunikasiya texnologiyalarini rivojlantirish dasturi”ning ushbu fanni o‘qitishdagi ro‘li va ahamiyati.

3. Axborot tushunchasi.

4. Informatika fani va uning o‘rganish obyekti.

5. Axborot texnologiyasi tushunchasi.

6. Axborotlar tizimi. Ma’lumotlarning tuzilishi va ularning jamg‘armasi.

7. Axborot, uni tasvirlash va o‘lchash.

8. Informatikaning asosiy tushunchalari- alifbo, so‘z, axborot, ma’lumot.

9. Axborot turi va xususiyatlari, axborot o‘lchami me’yorlari (Xartli va Shennon bo‘yicha), uning qiymati.

10. Axborotlarni olish, uzatish, saqlash, to‘plash va qayta ishlash.

11. Axborotlarni o‘lchash.

12. Axborotlarni kodlash.

13. Axborotlarni kompyuter xotirasida tasvirlash.

14. Axborotni kodlash va shifrlash.

Nazorat savollari

1. Informatika tushunchasi.

2. Informatikaning vujudga kelishi va rivojlanishi.

3. Informatika rivojlanishining asosiy bosqichlari.

4. Informatikaning bugungi kundagi ahamiyati.

5. Zamonaviy informatika tushunchasi.

6. Axborot texnologiyalari tushunchasi.

7. O‘zbekiston Respublikasi Vazirlar Mahkamasi tomonidan tasdiqlangan “O‘zbekiston Respublikasining axborotlashtirish konsepsiyasi”.

8. INTERNET tarmog‘i.

9. Kompyuter -informatikaning texnik asosi.

10. Informatika va kompyuterlar.

11. Kompyuterning rivojlanish tarixi.

12. Kompyuterning asosiy va yordamchi qurilmalari.

13. Prosessor, shinalar, kiritish va chiqarish qurilmalari.

14. Axborotni kompyuterda qabul qilish va qayta ishlash.

1.2 Axborotlarni o‘lchash. Axborotni kodlash. Axborotni komputer xotirasida tasvirlanishi

Axborot tushunchasi eng murakkab tushuncha hisoblanadi va odatda informatikaning kirish kurslarida boshlang‘ich bazaviy tushuncha sifatida intuitiv ravishda, soddalik bilan qabul qilinadi. Ko‘p hollarda bu tushuncha “xabar” tushunchasi bilan noto‘g‘ri ravishda tenglashtiriladi.

Turli predmet soxalarida “axborot” tushunchasi turlicha talqin qilinadi.

Masalan, axborot deb quyidagilarni tushunish mumkin:

- Abstraksiya, ko‘rilayotgan tizimning abstrakt modeli (matematikada);
- Boshqarish uchun signallar, ko‘rib chiqilayotgan tizimning moslamalari (kibernetikada);
- Ko‘rib chiqilayotgan tizimning xaos o‘lchovi (termodinamikada);
- Ko‘rib chiqilayotgan tizimda tanlash ehtimoli (ehtimollar nazariyasida);
- Ko‘rib chiqilayotgan tizimda ko‘p shakllilik o‘lchovi (biologiyada) va b.;

Informatikaning bu fundamental tushunchasini “alifbo” tushunchasi asosida ko‘rib chiqamiz (“alifboli”, formal yondoshuv). Alifboga formal ta’rif beramiz.

Alifbo, bu - ular uchun konkatenasiya (simvolni simvolga qo‘shish, qo‘shib yozish va simvollar zanjiri) amali aniqlangan turli belgi, simvollarning chekli to‘plamidir; uning yordamida bu alifboda simvollar va so‘zlarni bog‘lash uchun belgilangan qoidalar bo‘yicha so‘zlarni (belgilar zanjiri) va so‘z birikmalarini (so‘zlar zanjiri) hosil qilish mumkin.

X alifboning ixtiyoriy x ($x \in X$) elementi harf yoki belgi deb ataladi. Belgi tushunchasi y orqali belgilanayotgan (ma’nosi) bilan chambarchas bog‘liqdir, ular birga elementlar juftligi (x,y) sifatida qaraladi, bunda, x-belgining o‘zi, y esa shu belgi orqali belgilangan ma’noning o‘zi.

Misol. Alifbolarga misol: o‘nta raqamdan iborat to‘plam, rus tili belgilaridan iborat to‘plam, nuqta va Morze alifbosidagi chiziqcha va boshqalar.

Alifbo harflarining chekli to‘plami alifbodagi (yoki alifbodan) so‘z deyiladi. X alifbodan biror r so‘zning $|p|$ uzunligi deb undagi harflar soniga aytiladi.

Nol o‘lchovga ega bo‘lgan so‘z bo‘sh so‘z deyiladi.

X alifbosidagi turli so‘zlarning to‘plamini *S*(*X*) orqali belgilaymiz va alifboning lug‘at zaxirasi deb ataymiz.

Chekli alifbodan farqli ravishda lug‘at zaxirasi cheksiz bo‘lishi mumkin.

Berilgan alifbodan olingan so‘zlar xabarni bildiradi.

Misol. Kiril alifbosidagi so‘zlar - “Informatika”, “into”, “iiii”, “i”. O‘nli raqamlar alifbosidagi va arifmetik amallar belgilarining so‘zlari - “1256”, “23+78”, “35-6+89”, “4”. Morze alfbosidagi so‘zlar - “.”,”..-,” “---”.

Alifboda harflarning ketma-ketlik tartibi aniqlangan bo‘lishi kerak (“oldingi element-keyingi element” turidagi tartib), ya’ni ixtiyoriy alifbo $X=\{x_1, x_{2,...}, x_n\}$ alifbosi tartiblangan ko‘rinishga ega.

Shunday qilib, alifbo leksikografik (alifboli) tartiblash masalasini yechish imkoniyatini, ya’ni bu alifbodan (alifbo simvollari bo‘yicha) so‘zlarning alifboda aniqlangan tartib bilan mos ravishda joylashish masalasini yechishi lozimdir.

Axborot, bu - bizning bilimlarimizni aks ettiradigan, uzatadigan va ko‘paytiradigan tartiblangan xabarlar ketma-ketligidir. Axborot xabarlar aniq bir turdagi signallar, belgilarning turli shakllari yordamida aktuallashadi.

Axborot - boshlang‘ich manbaga yoki qabul qiluvchiga nisbatan uch turda bo‘lishi mumkin: kiruvchi, chiquvchi va ichki.

Axborot - oxirgi natijaga nisbatan boshlang‘ich, oraliq va natijaviy bo‘lishi mumkin.

Axborot - uning o‘zgaruvchanligiga qarab o‘zgarmas, o‘zgaruvchi va aralash bo‘ladi.

Axborot - undan foydalanish davri bo‘yicha birlamchi va ikkilamchi bo‘ladi.

Axborot - to‘liqligi bo‘yicha keragidan ortiqcha, yetarli va yetishmaydigan bo‘ladi.

Axborot - foydalanish huquqiga ko‘ra ochiq va yopiq bo‘lishi mumkin.

Axborotni sinflashning boshqa turlari ham mavjuddir.

Misol. Falsafiy jihatdan axborot dunyoqarash bo‘yicha, estetik, diniy, ilmiy, xo‘jalik, texnik, iqtisodiy, texnologik axborotlarga bo‘linadi.

Axborotning asosiy xossalari quyidagilardan iborat:

- ✓ to‘liqlik;
- ✓ aktuallik;
- ✓ adekvatlik;
- ✓ tushunarlilik;
- ✓ ishonchlilik;
- ✓ ommaviylik;
- ✓ turg‘unlik;
- ✓ qiymati.

Axborot - xabarning mazmuni, xabar esa axborotning shakli.

Ixtiyoriy axborotlar baytlarda, kilobaytlarda, megabaytlarda, gegabaytlarda, terabaytlarda, petabaytlarda va ekzobaytlarda o‘lchanadi, masalan, kompyuterda nollar va birlar yordamida kodlanadi, EHMlarda bitlarda yoziladi va amalga oshiriladi.

Axborotning o‘lchov birliklari orasidagi asosiy munosabatlarni keltiramiz:

1 bit (binary digit-ikkilik son) = 0 yoki 1,

1 bayt 8 bit,

1 kilobayt (1Kb) = 2^{13} bit,

1 megabayt (1Mb) = 2^{23} bit,

1 gegabayt (1Gb) = 2^{33} bit,

1 terabayt (1Tb) = 2^{43} bit,

1 petabayt (1Pb) = 2^{53} bit,

1 ekzobayt (1Eb) = 2^{63} bit.

Misol. Agar quyidagi munosabatlar to‘g‘ri bo‘lsa, *x* va y noma’lumlarni toping:

128^{y}(K) = 32^{x}(bit);

2^{x}(M) = 2^{y}(bayt).

Axborotning o‘lchov birliklarini tenglashtiramiz:

2^{7y}(K) = 2^{7y+13}(bit);

2^{x}(M) = 2^{x+20}(bayt).

Tenglamaga qo‘yib va axborotning o‘lchovini tashlab yuborib, quyidagini hosil qilamiz:

$2^{7y+13} = 2^{5x}$;

$2^{x+20} = 2^{y}$.

Ikki algebraik tenglamalardan iborat tizimni hosil qilamiz:

$$\begin{cases} 7y + 13 = 5x \\ x + 20 = y \end{cases}.$$

Yoki, bu tizimni yechib, x = -76.5, y = -56.5 yechimlarga ega bo‘lamiz.

Axborotni o‘lchash uchun turli yondoshishlar va usullardan, masalan, R.Xartli va K.Shennon usullari bo‘yicha axborot o‘lchamini topishdan foydalaniladi.

Axborot miqdori - baholanayotgan tizimdagi ko‘p shakllilikni (tarkiblilik, aniqlik, holatlarni tanlash va h.k) adekvat tavsiflovchi son.

Axborotning miqdori ko‘pincha bitlarda o‘lchanadi, ba’zida bunday baholash bit ulushlarda ham ifodalanadi (chunki xabarlarni o‘lchash emas balki kodlash haqida fikr yuritmoqdamiz).

Axborot miqdorini baholash kriteriysi, bu - axborotning o‘lchami deyiladi. Odatda u hodisalar to‘plamida aniqlangan va additiv bo‘lgan biror manfiy bo‘lmagan funksiya orqali beriladi, ya’ni, o‘lcham – hodisalar (to‘plamlar) chekli birlashmalarining o‘lchami har bir hodisa o‘lchamlarining yig‘indisiga teng.

Axborotning turli o‘lchamlarini ko‘rib chiqamiz:

R.Xartli o‘lchamini olamiz. Faraz qilaylik, S tizimining N ta holati ma’lum bo‘lsin (tizimning ketma-ket bo‘lgan turli, imkoniyati teng bo‘lgan N ta tajribasi). Tizimning har bir holatini ikkilik kodlar bilan kodlashda d kodning uzunligini, barcha turli kombinasiyalari soni N dan kichik bo‘lmaydigan qilib tanlashimiz lozim bo‘ladi:

$$2^{d} \geq N$$

Bu tengsizlikni logarifmlab, quyidagini hosil qilamiz:

$$d \geq \log_2 N$$

Bu tengsizlikning eng kichik yechimi yoki tizim holatlarining to‘plam ko‘p shaklliligi o‘lchami R.Xartli formulasi orqali beriladi:

$$N \geq \log_2 N(bit)$$

Misol. To‘rtta mumkin bo‘lgan holatdan tizim holatini aniqlash uchun, ya’ni tizim haqida biror axborotni olish uchun 2 ta savol berish lozim. Birinchi savol, masalan: “Holat nomeri 2 dan kattami?”. Javobni (“ha”, “yo‘q”) bilganimizdan so‘ng, tizim haqida yig‘indi axborotni 1 bitga kattalashtiramiz ($I=\log_2 2$). Shundan so‘ng, aniqlik kirituvchi yana bir savol beramiz, masalan “ha” javobini olganimizda: “Holat nomeri 3 ga tengmi?”. Demak, axborot miqdori 2 bitga teng ($I=\log_2 4$). Agar tizim n ta turli holatlarga ega bo‘lsa, axborotning maksimal miqdori $I=\log_2 N$ ga teng.

Agar $X=\{x_1, x_{2,...} x_n\}$ to‘plamida ixtiyoriy elementni izlaydigan bo‘lsak, uni (R.Xartli usuli bo‘yicha) topish uchun $I=\log_2 N$ (birlik) dan kam bo‘lmagan axborotga ega bo‘lishimiz kerak.

N kamaytirish N tizimining holatlar ko‘p shaklliligini kamaytirishni bildiradi. N orttirish N tizimining holatlar ko‘p shaklliligini orttirishni bildiradi.

Xartli o‘lchami ideal, abstrakt tizimlargagina qo‘llanilishi mumkin, chunki real tizimlarda tizim holatlari har xil amalga oshiriladi (teng bo‘lmagan ehtimolda).

Bunday tizimlar uchun ularga mos keluvchi K.Shennon o‘lchamidan foydalaniladi. Shennon o‘lchami axborotni uning ma’nosiga e’tibor bermasdan baholaydi:

$$I = -\sum p_i \log_2 p_i,$$

n-tizim holatlarining soni; p_i - tizimni i-holatga o‘tish ehtimoli, barcha p_i lar yig‘indisi 1 ga teng bo‘lishi lozim.

Ko‘rib chiqilayotgan tizimning barcha holatlari teng imkoniyatli, teng ehtimolli bo‘lsa, ya’ni p_i =1/n bo‘lgan holatda Shennon formulasidan (xususiy holat sifatida) Xartli formulasini keltirib chiqarishimiz mumkin:

$I=\log_2 2$

Misol. 10 ta katakdan iborat tizimda nuqtaning joylashish o‘rni ma’lum bo‘lsa, masalan, agar, nuqta ikkinchi katakda joylashgan bo‘lsin, ya’ni

p_i =0, i=1,2,3,4,...,10, p_2=1,

U holda nolga teng bo‘lgan axborot miqdoriga ega bo‘lamiz:

$I=\log_2 1=0$.

Miqdorni quyidagicha belgilaymiz: $f_i = -n\log_2 p_i$,

U holda Shennon formulasidan I axborot miqdorini f_i miqdorlarining o‘rta arifmetigi deb qabul qilamiz, ya’ni f_i miqdorini axborot uzatayotgan ixtiyoriy xabar (so‘z)da bu belgining uchrashi ehtimoli p_i miqdorda va i indeksli alifbo belgisining axborot mazmuni deb izohlash mumkin.

Termodinamikada Bolsman koeffisiyenti deb ataluvchi miqdor

$k = 1{,}38*10^{-16}$(erg/grad)

va termodinamik tizimda xaos o‘lchami yoki entropiya uchun quyidagi

$$S = -k\sum_{i=1}^{n} p_i \ln p_i$$

Ifoda (Bolsman formulasi) bizga ma’lumdir.

I va S uchun ifodalarni taqqoslab, I miqdorni tizimda (tizim haqida) axborot yetarli bo‘lmaganligi munosabati bilan entropiya deb tushunishimiz mumkin.

Entropiya va axborot o‘rtasidagi asosiy funksional munosabat quyidagi ko‘rinishga ega:

$$I+S(\log_2 e)/k = \text{const}$$

Bu formuladan muhim natijalar kelib chiqadi:

1. Shennon o‘lchamining ortishi tizim entropiyasining kamayishini (tartibning ortishini) bildiradi;

2. Shennon o‘lchamining kamayishi tizim entropiyasining ortishini (tartibning kamayishini) bildiradi;

Axborot ma’nosini e’tiborga olmasligi Shennon formulasining ijobiy tomonidir. Shuningdek, Xartli formulasidan farqli ravishda xolatlarning turli-tumanligini hisobga oladi, bu esa uni amaliy hisoblashlarda foydalanish imkoniyatini beradi.

Tizimning turli holatlarini bir xil ehtimolda aniqlay olmaganligi Shennon formulasining asosiy kamchiligidir.

Axborotni hosil qilish usullarini uchta katta guruhlarga bo‘lish mumkin:

1. Empirik usullar yoki empirik ma’lumotlarni olish usullari;

2. Nazariy usullar yoki turli nazariyani qurish usullari;

3. Empirik-nazariy usullar yoki Ob’yekt, jarayon, hodisa haqidagi empirik ma’lumotlar asosida nazariyani qurish usullari;

Empirik usullarni qisqacha tavsiflaymiz.

1. Kuzatish, bu - Ob’yekt, jarayon, hodisa haqidagi birlamchi axborotni yig‘ish.

2. Taqqoslash, bu - farqini va umumiyligini topish va ularning o‘zaro nisbatini aniqlash.

3. O‘lchash, bu - Ob’yekt, jarayon, hodisalarning yangi biror xossalarini aniqlash maqsadida ularni ko‘rib chiqish, o‘zgartirish.

Ularni amalga oshirishning klassik shakllaridan tashqari, oxirgi vaqtda so‘rov, intervyu, testlash usullaridan foydalanilmoqda.

Empirik-nazariy usullarni qisqacha tavsiflaymiz.

1. Abstraklash, bu - tadqiq qilinayotgan Ob’yekt, jarayon, hodisalarning muhim bo‘lgan xossalarini, tomonlarini ajratish, hamda ahamiyatsiz va ikkinchi darajali tomonlarini e’tiborga olmaslik;

2. Tahlil - yaxlitlikni ular orasidagi bog‘liqlikni aniqlash maqsadida qismlarga bo‘laklash.

3. Dekompozisiya - bir butunni atrof muhiti bilan bog‘liqlikni yo‘qotmagan holda qismlarga bo‘laklash.

4. Sintez - ular orasidagi bog‘liqlikni aniqlash maqsadida qismlarni bir butun qilib birlashtirish.

5. Kompozisiya - atrof muhiti bilan bog‘liqlikni aniqlash maqsadida qismlarni bir butun qilib birlashtirish.

6. Induksiya - qismlar bo‘yicha bilimlar haqida yaxlit bilimga ega bo‘lish.

7. Deduksiya - yaxlit bilimlar bo‘yicha qismlar haqida bilimga ega bo‘lish.

8. Evristikalar, evristik proseduralardan foydalanish - bu, qismlar bo‘yicha bilimlardan va kuzatishlar, tajriba, intuisiya, oldindan ko‘ra bilish bo‘yicha yaxlitligi haqida bilimlarni olish.

9. Modellashtirish (oddiy modellashtirish), asboblardan foydalanish - model yoki asboblar yordamida qismlari haqida va yaxlitligi haqida bilimlarni olish.

10. Tarixiy usul, bu - real mavjud boʻlgan yoki boʻlishi mumkin boʻlgan, tarixidan oldingi davrdan foydalanib bilimlarni izlash.

11. Mantiqiy usul - qismlarni, bogʻlanishlarni yoki tafakkur elementlarini qayta tiklash yoʻli bilan bilimlarni izlash.

12. Maketlash - maket boʻyicha, qismlarni soddalashgan, lekin yaxlit koʻrinishda taqdim etib axborotni olish.

13. Aktuallashtirish - yaxlit yoki uning qismlarini statistik holatdan dinamik holatga oʻtkazish yordamida axborotni hosil qilish.

14. Vizuallashtirish - Obʼyekt, jarayon, hodisalarning holatlarini koʻrgazmali yoki vizual taqdim etish yordamida axborotni hosil qilish.

Nazariy - empirik usullarni amalga oshirishning klassik shakllaridan tashqari, koʻpincha monitoring (kuzatish tizimi va holatlar tahlili), ishga aloqador boʻlgan oʻyinlar va vaziyatlar, ekspert baholash, imitasiya (taqlid qilish) va boshqa shakllar.

Nazariy usullarni qisqacha tavsiflaymiz.

1. Abstraktdan aniqlikka koʻtarilish, bu - ongda, fikrlashda abstrakt koʻrinishlar asosida yaxlitlik va uning qismlari haqida bilimlarni hosil qilish.

2. Ideallashtirish - taffakkurda real borliqda mavjud boʻlmagan yaxlitlik va uning qismlarini gavdalantirish yoʻli orqali yaxlitlik va uning qismlari haqida bilimlarni hosil qilish.

3. Formallashtirish - yaxlitlik va uning qismlari haqidagi bilimlarni sunʼiy yaratilgan tillar (formal tavsiflash, taqdim etish) yordamida hosil qilish.

4. Aksiomalashtirish - yaxlitlik va uning qismlari haqidagi bilimlarni baʼzi bir aksiomalar va undan (va oldingi olingan fikrlardan) yangi aniq fikr hosil qilinadigan qoidalari yordamida hosil qilish.

5. Virtuallashtirish - yaxlitlik va uning qismlari haqidagi bilimlarni sunʼiy muhit, vaziyat yordamida hosil qilish.

Misol. Mamlakat, region yoki yirik soha doirasida ishlab chiqarishni rejalashtirish va boshqarish modelini qurish uchun quyidagi muammolarni yechish lozim boʻladi:

1. Tarkibiy bogʻlanishlarni, qabul qilish va boshqarish darajalarini, resurslarni aniqlash: bu holda koʻpincha kuzatish, taqqoslash, oʻlchash, tajriba, tahlil va sintez, deduksiya va induksiya, evristik, tarixiy va mantiqiy usullar, maketlash va h.k dan foydalaniladi.

2. Gipoteza, maqsadlar, rejalashtirishning mumkin boʻlgan muammolarini aniqlash; bunda eng koʻp ishlatiladigan usullar - kuzatish, taqqoslash, tajriba, abstraklash, tahlil, sintez, deduksiya va induksiya, evristik, tarixiy, mantiqiy usullar va h.k.

3. Empirik modellarni konstruksiyalash; bunda eng koʻp ishlatiladigan usullar - abstraktlash, tahlil, sintez, induksiya, deduksiya, formallashtirish, ideallashtirish va h.k.

4. Rejalashtirish muammosi yechimini va turli variantlar xatoliklarini, rejalashtirish direktivlarini izlash, optimal yechimni izlash; bunda eng koʻp ishlatiladigan usullar - oʻlchash, taqqoslash, tajriba, tahlil, sintez, induksiya, deduksiya, aktuallashtirish, maketlash, vizuallashtirish, virtuallashtirish va h.k.

Tizimni boshqarish masalasining mohiyati-qimmatli axborotni “shov-shuv” (axborot tizimi uchun baʼzida zararli boʻlgan, foydasiz) axborotdan ajratish va shu tizim mavjud boʻlishi va rivojlanishiga imkon yaratadigan axborotni belgilash.

Axborot tizimi, bu - elementlari, tarkibi, maqsadi, resurslari axborot darajasida koʻrib chiqiladigan tizimdir (biroq, tabiiyki boshqa koʻrib chiqish darajalari ham mavjud).

Axborot muhiti - ushbu tizimlarda aktuallashtiriladigan axborot bilan birga, oʻzaro taʼsirda boʻlgan axborot tizimlar muhitidir (tizim va uning atrofi).

Munosabatlar va bogʻlanishlarni oʻrnatish, ularni formal vositalar bilan, tillar orqali tavsiflash, shu tavsiflarga mos keluvchi modellar va metodlarni algoritmlarni ishlab chiqish, shu model va usullarni qoʻllaydigan va inson faoliyati

doirasidagi, ta’lim sohasidagi fan sifatida informatikaning asosiy masalasini tashkil qiladi.

Informatikani turli predmet sohalarida, jamiyatda, tabiatda, borliqni anglashda kechadigan axborot jarayonlarning o‘zgarmaydigan mohiyatlarini (invariantlarini) o‘rganadigan fan sifatida izohlash mumkin.

Hayotda hox yosh, hox katta yoshli har bir insonning yashashi-yu, har bir qiladigan ishi axborotlarni qayta ishlash bilan chambarchas bog‘lanib ketgan. Hamma qilinadigan ishlar axborotlarni qayta ishlash bilan bo‘ladi.

Axborotning o‘zi nima degan savolning tug‘ilishi tabiiydir. Bu savolga biror aniq to‘liq javob berish qiyin. Fanda axborot birlamchi tushunchadir. Inson u bilan bog‘liq bo‘lgan axborot manbai, axborotni tashuvchi, axborotni uzatuvchi, axborotni qabul qiluvchi tushunchalarni yaxshi biladi.

Axborot tushunchasining o‘ziga xos xususiyati shundaki, u hamma sohada: gumanitar va tabiiy fanlarmi, falsafa, tibbiyotmi, inson va hayvonot psixologiyasimi, sosiologiya, sa’nat, texnika, iqtisod va h.z kundalik hayotda ishlatiladi. Shu ma’noda ham axborotga to‘liq ta’rif berish qiyin, chunki har bir sohaning axborotni qayta ishlashi va unga yondashishi o‘ziga xosdir.

Umumiy holda axborotga bizni o‘rab turgan dunyo, tabiat va jamiyatda sodir bo‘layotgan va bo‘lgan voqyealar hamda hodisalar haqidagi xabarlar majmuasi deb qarash mumkin.

Inson axborotni kamida uch xil yo‘l bilan qayta ishlaydi va o‘z munosabatini bildiradi.

1) Fiziologik - jismoniy yo‘l bilan qayta ishlash. Bu holda xabarlarni inson organizmi orqali qabul qilib, shunga qarab axborotga o‘z munosabatini bildiradi. Misol uchun, issiq choynakdan ehtiyot bo‘ladi, achchiqni sezib ichmaydi, olovni ko‘rib uning oldiga bormaydi va h.z.

2) Ongli ravishda qayta ishlash.

3) Aql-idrok va fikrlash asosida qayta ishlash. Bu ikki holda axborotlarni qayta ishlash juda ham murakkab bo‘lib, u odamning yoshi, hayotiy tajribasi, mutaxassisligi, xatto xulqi va xarakteriga, unga qanchalik zarur va h.z.larga

bog‘liq. Mashina dvigatelining notekis ishlashiga xaydovchi darrov e’tibor bersa, xaydovchi bo‘lmagan odam e’tibor bermaydi.

Axborotning hajmi to‘g‘risida gapiriladigan bo‘lsa, kompyuter nuqtai nazaridan olganda bitta “bit”dan (“ha” yoki “yo‘q”, 1 yoki 0 ni ifodalaydi) iborat. Odam uchun axborot hajmi odamning bilimi, tajribasi, axborotning unga qanchalik kerakligiga qarab aniqlanadi.

Misol uchun, mashina dvigatelining ishlamay qolishi mashina egasi uchun katta hajmdagi axborot bo‘lsa, begona odamga kichik hajmdagi axborot bo‘ladi. Sportchining yutig‘i siz uchun kichik hajmdgi axborot bo‘lsa, sportchining oilasi uchun juda katta axborot bo‘ladi.

Informatika fanida masala bu nuqtai nazardan qaralmaydi, aksincha hamma axborotlarni chekli belgilar majmuasi yordamida tasvir qilinib, bu belgilar orqali har xil axborotlar Ob’yektini hosil qilinadi.

Shu paytgacha insonga noma’lum bo‘lgan biror yangi bilimni egallash ilm bo‘lsa, shu egallagan bilim asosida qandaydir yangi mahsulot (u hox moddiy, hox ruhiy ozuqa beradigan mahsulot bo‘lsin) yaratish texnologiya bo‘ladi. Ilm texnologiyani rivojlantirishga olib kelsa, texnologiya ham ilmni rivojlanishiga ta’sir qiladi. Hayotimizning juda ko‘p sohalarida va inson faoliyatida axborotlarni qayta ishlash bilan bog‘lik har xil amallarni bajaradigan aniq texnik va dasturiy vositalar majmuasi axborot texnologiyasi deyiladi.

Informatika (axborot) fani axborot xususiyatlarini, axborotlarni tasvir etish usullarini, axborotlarni to‘plash, jamlash, uzatish, saqlash va qayta ishlash qonun va qoidalarini o‘rganadi.

Informatika fanining o‘zagini axborot texnologiyasi tashkil etadi. Unda axborotlarni qayta ishlashda asosiy o‘rinni texnik vosita sifatida kompyuter (hisoblovchi) egallaydi.

Informatika fanining nazariy asosini axborot nazariyasi, algortmlar nazariyasi, matematik mantiq, formal grammatika kabi fundamental fanlar hamda informatikaning o‘ziga xos kompyuter arxitekturasi, operasion tizim (tezkor tizim), dasturlash nazariyasi kabi bo‘limlari tashkil qiladi. Uning moddiy texnik asosini

esa fizika, ximiya fanlari, ayniqsa fizikaning radio texnika va elektronika bo‘limlari tashkil qiladi.

Zamonaviy kompyuterlarning yaratilishi axborotlarni qayta ishlash va axborotlarga qarashni tubdan o‘zgartirib yubordi.

Axborot texnologiyasi bu umuminsoniy madaniyatning ajralmas bir qismidir. Yuqorida axborot texnologiyasiga berilgan ta’rifdan ko‘rinib turibdiki, u ikki qismdan:

- texnik ta’minot;

- dasturiy ta’minot.

Texnik asbob - uskuna ta’minoti kompyuter va unga tegishli bo‘lgan qo‘shimcha asbob - uskunalardan iborat. Hozirgi kompyuterlarning qurilmalari juda ko‘pki, ular yordamida axborotlarni kompyuter xotirasiga kiritish va xotiradagi axborotlarni tashqi har xil ma’lumot tashuvchilarga chiqarish ishlari qulay va oson bajariladi.

Bularga har xil asbob - uskunalar, misol uchun, skanerlar, “sichqoncha”, planshet, CD-ROM, ovoz berish qurilmalari, musiqa platalari va h.z.lar kiradi.

Hozirgi zamon kompyuterlari maxalliy, regional va jaxon tarmoqlari tizimi bilan ishlay oladi. Bular uchun maxsus tarmoq va kommunikasion asbob - uskunalar (tarmoq platalari, modemlar, adapterlar va h.z.) kerak bo‘ladi.

Axborot texnologiyasining ikkinchi asosiy qismi - dasturiy ta’minot qismidir. Har qanday takomillashgan, tez ishlaydigan kompyuter bo‘lmasin dasturiy ta’minoti bo‘lmasa, u temirdir, chunki kompyuterning ishlashi faqat dasturlar bilan bajariladi.

Dasturiy ta’minotni ikki guruhga ajratish mumkin:

- tizimli dasturiy ta’minot,

- amaliy dasturiy ta’minot.

Tizimli dasturiy ta’minotning asosiy vazifasi asbob - uskuna vositalarining ishlarini boshqarish, uning imkoniyatlaridan to‘liq foydalanishni, odamning kompyuter bilan dialog olib borishini va amaliy dasturlarni ishga tushirishni

bajarishdir. U kompyuterdan foydalanuvchilarni va amaliy dasturlarni kompyuter qurilmalari bilan muloqotda bo‘lishning qulay usullari bilan taminlaydi.

Amaliy dasturiy ta’minot dasturlari kompyuterlarda amaliy masalalarni yechishni taminlaydi.

Hozirgi paytda hamma sohalarda bunday amaliy dasturlar juda ko‘p, minglab hisoblanadi. Ularni bajaradigan ishlariga qarab quyidagi guruhlarga ajratish mumkin:

- matn muharrirlari;
- nashriyot tizimlari;
- jadval ma’lumotlarni qayta ishlash;
- axborot jamg‘armalarini qayta ishlash.

Buning ichiga xalq ta’limi ishlarini boshqarish, xalq ta’limi muassasalari ishlarini boshqarish va fanlarni o‘qitish, sinov nazorat ishlarini bajaradigan amaliy dasturlar ham kiradi.

1) va 2) guruh dasturlarining qiladigan ishlari matnni taxrir qilish, nashriyot ishlarini bajarishdir.

Jadval ma’lumotlarni qayta ishlash dasturlar to‘plami berilgan ifodalar bo‘yicha har xil ma’lumotlar jadvallarini hisoblash, har xil diagramma, grafiklar chizish, har xil jadval ko‘rinishida berilgan ma’lumotlarni qayta ishlash ishlarini bajaradi. Keng ommalashgan jadval protsessorlar Lotus 1-2-3, SuperCalc, Microsoft Excel, Karat_m va hokazolardir.

Axborot jamg‘armalarini qayta ishlash dasturlari katta hajmdagi axborotlar to‘plamlarini boshqarish ishlarini bajarishga mo‘ljallangan. Eng sodda axborot jamg‘armalari bu bir o‘lchamli jadvallar bo‘lib, ular ustida kiritish, tuzatish, qidirish, saralash va har xil hisobotlar tayyorlash ishlarini bajarishdan iboratdir. Hayotda esa ancha murakkab bir necha o‘lchamli jadval axborotlar bilan ishlashga to‘g‘ri keladi. Bunday hollarda kiritiladigan va chiqariladigan ma’lumotlarni foydalanuvchi qulay holda olish uchun maxsus axborot jamg‘armalari tuziladi. Bunday jamg‘armalarni qayta ishlash uchun maxsus dastur to‘plamlari tuzilgan. Misol uchun, DBase, FoxPro, Clepper, Paradox, RBaselar. Shunday dasturlar

to‘plamlari va texnik hamda dasturiy ta’minot vositalari yordamida juda katta hajmdagi axborotlarni tez qayta ishlab kerakli natijalarni o‘z vaqtida olish mumkin.

Shularning hammasini birgalikda zamonaviy yoki yangi axborot texnologiyasi deb yuritilishi ham shundan kelib chiqqan.

Muhokama savollari

1. Axborot tushunchasi.
2. Axborotning o‘lchov birliklari.
3. Xartli usuli bilan axborotni o‘lchash.
4. Axborotni o‘lchashda Shennon formulasidan foydalanish.
5. Empirik modellarni konstruksiyalash.
6. Dasturiy ta’minot va uning turlari.

Nazorat savollari

1. Turli predmet sohalarida “axborot” tushunchasining turlicha talqini.
2. Axborotning asosiy xossalari.
3. Xabarning o‘lchov birliklari.
4. Empirik-nazariy usullar.
5. Amaliy va tizimli dasturiy ta’minot.

1.3 Axborotni kodlash. Komputerning asosiy qurilmalari

Jamiyatning shiddat bilan rivojlanishi bosqichida xoxlagan ko‘rinishdagi faoliyat qanday ma’lumotga ega ekanligiga va shu ma’lumotga ega emasligi bilan baholanadi. Bu faoliyat qanchalik chuqur amalga oshib borsa, ma’lumotlarni himoyalash shunchalik zarur. Bir so‘z bilan aytganda, ma’lumotlarni qayta tiklash jarayoni ularni himoyalash jarayonining rivojlanishiga olib keldi.

Jamiyatning muhim masalalaridan biri bu - xabarlarni kodlash va ma’lumotlarni shifrlashdan iborat [2].

Ma’lumotlarni ochish va himoyalash muammolari bilan kriptologiya fani shug‘ullanadi (kriptos- maxfiy, logos-fan).

Kriptologiya ikkita asosiy yo‘nalishga ega : kriptografiya va kriptotahlil. Bu yo‘nalishlar bir-biriga zid.

Kriptografiya ma’lumotlarni matematik metodlar orqali qayta ishlash, kriptoanaliz esa ma’lumotlarni kalitsiz ochish (rasshifrovka)dan iborat. «Kriptografiya termini ikkita grek so‘zidan olingan bo‘lib, kriptos va grofeyn – yozmoq ma’nosiga ega. Shunga ko‘ra bu maxfiy yozuv, ya’ni ma’lumotlarni, xabarlarni qayta kodlab boshqalarga tushunarsiz holatga keltirishdir.

Kodlash va shifrlashga oid asosiy tushunchalarni keltirib o‘tamiz.

Kod- bu bir qoidaga asosan X ko‘plikka tegishli bo‘lgan qiymatlarning Y ga tegishli qiymatlarga mutanosib bo‘lishiga aytiladi. Agar har bir X qiymatga kodlash orqali Y qiymat mos kelsa, unda bu kodlash deyiladi. Agar har bir Y uchun ma’lum qoidaga asosan X qiymat topilsa, bu dekodlash deyiladi. Kodlash – bu X to‘plamga tegishli bo‘lgan harflar (so‘zlarni), Y to‘plamdagisiga o‘zgartirish. Ma’lumotlarni EHM da qayta ishlash uchun barcha simvollar baytlarda kodlanadi.

Misol: Agar har bir rangni 2 bit deb olsak, unda $2^2 = 4$ ta rangni , 3 bit bilan $2^3=8$ rangni, 8 bit (bayt) - 256 rangni kodlash mumkin. Klaviaturadagi barcha simvollarni kodlashda baytlar yetarlidir [4].

Biz yuborayotgan xabarni ochiq xabar desak, aniqki, bu qaysidir alifbo orqali aniqlangan bo‘ladi.

Shifrlangan xabar boshqa alifbo yordamida tuzilgan bo‘lishi mumkin. Buni biz yopiq ma’lumot deb ataymiz. Ochiq xabarni yopiq xabarga aylantirish bu shifrlashdir.

Agar A- ochiq xabar, B-yopiq xabar (shifr), f-shifrlash qoidasi bo‘lsa, bunda f(A)=B dir. Shifrlash qoidalari shunday tanlanishi kerakki, bunda shifrlangan ma’lumotni qayta shifrlash mumkin bo‘lsin. Bir turdagi qoidalar sinflarga birlashtiriladi , sinflar orasidan biror – bir parametr tanlanadiki, bu parametr barcha qoidalarni tahlil qiladi . Bunday parametr ”shifr kaliti” deb ataladi . Qoidaga ko‘ra kalit maxfiy va u faqat shifrlangan ma’lumotni o‘qishi mumkin bo‘lgan odamga

beriladi. Kodlashda esa maxfiy kalit bo‘lmaydi, chunki kodlashdan maqsad ma’lumotni siqilgan va ixcham holatga keltirishdan iborat.

Agar K-kalit unda f(K(A))=B deb yozish mumkin. Har bir K-kalit uchun, keltirib chiqarilgan f(K) va K larning o‘zaro bir-biri bilan bog‘liqligi, mosligi shifr deyiladi.

Shifrlar ikkita katta guruhga bo‘linadi: o‘rin almashtirish shifrlari va almashtirish shifrlari.

O‘rin almashtirish shifrlari ma’lumotda keltirilayotgan simvollar ketma - ketligini almashtiradi.

Almashtirish shifri esa har bir simvolni boshqa simvolga almashtiradi.

Shifrning buzilishiga qarshi tura olishiga ishonchlilik deyiladi. Qayta shifrlashda xabar haqida barcha kalitdan boshqa ma’lumotlarga ega bo‘lish mumkin, ya’ni shifrlashdagi ishonchli kalitning maxfiyligi bilan ajratiladi.

Ochiq kriptografiyada, qaysiki shifrlashga turli kalitlar ishlatiladi, kalitga barcha ega bo‘lishi mumkin. Ammo kalitlar soni yuz tirillionga yaqin bo‘ladi.

Masalan: shifrlashga eng yaxshi misol bo‘la oluvchi – 1977 yilda Amerika Qo‘shma shtatlari milliy standartlash byurosi tomonidan shifrlashning algoritmi ishlab chiqildi. Tekshiruvlar shuni ko‘rsatadiki, kriptotahlilda kalitlarni to‘liq tanlab olishdan yaxshiroq metod topilganicha yo‘q. 1991 yilning iyulida yangi kriptoalgoritmlar yo‘lga qo‘yiladi (standart GOST - 28147-89), bu esa ishonchlilikni yanada oshiradi.

Kriptografik tizim – X qayta tashkil qilingan turkum. Bu turkumga kiruvchilar, K simvol bilan belgilanadi; K parametri esa kalit. K kalitlar ko‘pligi – bu K kalit qabul qilishi mumkin bo‘lgan belgilar. Asosan kalit ketma-ket kelgan alifbo harflarini qabul qiladi.

Ochiq matn asosan ixtiyoriy uzunlikka ega bo‘ladi. Agar matn katta bo‘lsa va u shifrator orqali butunligicha qayta ishlay olinmasa, unda aniq bloklarga bo‘lib, alohida shifrlanadi. Bunday kriptotizimlar blokli shifrlash deyiladi.

Kriptotizimlar simmetrik , ochiq kalitli va elektron imzolarga bo‘linadi.

Simmetrik kriptotizimlarda shifrlash va qayta shifrlashga ham bitta kalitning o‘zi ishlatiladi.

Ochiq kalitli tizimlarda esa 2 ta kalit ishlatiladi ochiq va yopiq. Ular matematik bir - biri bilan bog‘liq. Xabarni ochiq kalit orqali har bir hoxlovchi shifrlashi mumkin. Qayta shifrlash kaliti esa faqat xabarni qabul qiluvchida bo‘ladi.

Elektron imzo (EI) deb kriptografik qayta ishlangan va matnga birlashtirilgan, matn qabul qiluvchi tomonidan EIning haqiqiyligini tekshira olishiga aytiladi. EIga ikkita talab qo‘yiladi: birinchidan EIning haqiqiyligini osonlik bilan tekshira olish va ikkinchidan EIni rasmiylashtirilishining murakkabligi.

Kriptografiya kriptotizimlar (simmetrik, ochiq kalitli elektron imzo) dan tashqari kalitlarni boshqarishni ham o‘rganadi.

Kalitlarni boshqarish tizimi – bu shunday ma’lumotlar tizimiki, bundan maqsad foydalanuvchilar orasida kalitlarni tuzish va taqsimlashdan iborat. Tizimning xavfsizligi bo‘yicha administratorning an’anaviy vazifalaridan biri, bu kalitli, maxfiy ma’lumotlarni yaratishdir.

Kalitni biz kerakli kattalikdagi massiv statik ravishda bir-biriga bog‘liq bo‘lmagan holda tanlov asosida ikkilik ko‘rinishdagi {0,1} elementlardan tashkil etadi.

Misol: “Elektron ruletka” tamoyillariga ko‘ra topiladigan kalit dasturi. Foydalanuvchilar soni juda ko‘p bo‘lganida, tasodifiy sonlarni yetkazib beruvchi datchiklardan foydalinadi. Parolni ham almashtirish zarur.

Masalan: Morris virusi tizimga kirib olish uchun parolni o‘zida so‘ng bir necha yuz prosedura orqali o‘zgartirganiga o‘xshatib qo‘yadi.

Parolni xabar xavfsizligi uchun javobgar tizim administratori asosiy tamoyiliga ko‘ra tarqatish zarur. Shifrlash jarayonida kalit to‘liq foydalanish uchun, turli elementlar bilan ko‘p preseduralar orqali kodlash zarur. Asosiy sikllar turli turli elementlarni ko‘p marta takrorlanuvchidan iborat.

Masalan: Bank tizimidagi mijozlar va bank orasidagi kalitlar almashinuvi magnit tashuvchilar orqali amalga oshiriladi. Bunda kalit ochiq tarzda kompyuter tizimi orqali yuboriladi.

Kalitning maxfiy kaliti serverda saqlanadi va kirish (dostup) uchun yopiqdir, barcha elektron imzoli amallarni bajarish uchun mijoz kompyuteriga dastur o‘rnatiladi. Bu dastur bank tomonidan taqdim qilinib mijoz uchun kerakli bo‘lgan barcha ma’lumotlar – ochiq holda; yopiq kalit, login, parol va boshqalar asosan alohida diskda yoki maxsus kompyuterga ulanuvchi joyda saqlanadi.

Barcha kriptotizimlar Krixgoff prinsipiga asosan tuziladi: shifrlangan xabarlar kalitning maxfiyligi bilan belgilanadi. Bu degani, agar shifrlash algoritmi kriptoanalitikka ma’lum bo‘lsa ham, agar unda kalit bo‘lmasa ma’lumotni ochish imkoniyatiga ega emas.

Barcha klassik shifrlar shu tamoyilga asoslanib quyidagi tarzda tuzilganki, uni ochish uchun eng samarali yo‘li, barcha kalitlarni to‘liq tekshirish orqali ya’ni barcha kalitlarning imkoniyatlarini ko‘rish, bilan belgilanadi. Aniqki, barcha shifrlarning ochishga chidamliligi kalitlar to‘plamining kattaligiga bog‘liq.

Masalan: Rossiya shifrlarida asoson 256- bitli kalit ishlatiladi, kalit sig‘imi 2^{256} , hoxlagan kompyuter yordamida kalitning to‘liq tasodifiy yo‘l bilan qidirish uchun 100 yilgacha vaqt kerak bo‘ladi.

Rossiya kriptoalgoritmi aniq va qayta shifrlash chidamli ravishda tuzilgan. Axborot tizimning xavfsizligi – axborotning ximoyalanganligi kompyuter tizimining ichki va tashqi xavflardan, ya’ni axborot resurslarining ximoyalanganligiga bog‘liq va bu tizimning butunligi va evolyutsiyasini ta’minlaydi. Informasiyani himoyalashga elektron hujjatlar, dasturiy ta’minot, tuzilmalar va ma’lumotlar ombori va boshqalar kiradi.

Kompyuter tizimlarining xavfsizligi, himoyalash tizimlarining turli sinflariga asoslanadi:

1. Eng kam himoyalash (D)
2. Foydalanuvchining ixtiyoriga qarab (S)
3. Majburiy himoyalash (V)

4. Kafolatlangan himoya (A)

Bu sinflar yana alohida sinflarga bo‘linadi, biz bularni chuqur ko‘rib o‘tmaymiz. Kompyuter tizimidagi ma’lumotlar almashinuviga ta’sir etuvchi asosiy manbalar bu kompyuter viruslaridir.

Masalan: ko‘p marotaba o‘z kodini 2000 yilda tarqatgan Internetdagi virus dasturi xat ochilishidan avval (I love you –ya tebya lyublyu) sarlovhasi bilan chiqqan. U o‘z kodini adres orqali yelpig‘ich usulida tarqatgan. Ma’lumki, Internetdagi adres kitobida foydalanuvchilarning o‘nlab va yuzlab adreslari mavjud.

Kompyuter tizimlariga ta’sir etuvchi asosiy vositalar bu kompyuter viruslaridir, mantiqiy buzg‘unchilarni ma’lumotlar almashinuviga tatbiq qilish. Kompyuter viruslari bu maxsus yozilgan dastur bo‘lib, u kimlardir tomonidan yomon maqsadlarda tuzilib, dasturdan dasturga o‘tish qobiliyatiga ega. Virus – xuddi inson tanasiga kirib qolgan infeksiya kabi butun organizm bo‘ylab harakatlanadi.

Viruslarning asosiy turlari:

1. Yuklanuvchi- kompyuterning yuklanish vaqtida ishga tushib diskdagi fayl, ma’lumot va boot-sektorni zararlaydi.

2. Apparat qismga ta’sir etuvchi.

3. Dasturlarni (exe-fayllar) zararlovchi.

4. Polimorf – ko‘p shaklli viruslar.

5. Makroviruslar- hujjatlarni zararlovchi.

6. Ko‘p maqsadli viruslar.

Eng xavfli viruslar bu kompyuter tizimlariga ta’sir etuvchilardir, bular tarmoq ishini ishdan chiqarishi mumkin . Kompyuter tarmoqlariga virus quyidagi yo‘llar bilan kirishi mumkin.

1. Tashqi tashuvchilar orqali (fayl nusxalari, fleshlar).

2. Elektron pochta.

3. Internet.

Kompyuter viruslari bilan kurashishda turli yo‘llar mavjud.

Antiviruslarni tanlashda quyidagi tamoyillarga asoslanish zarur.

1. Antivirus oddiy va tushunarli bo‘lishi kerak.

2. Antivirus yangi viruslarini topa olishi kerak.

3. Lisenziyaga ega bo‘lishi va baza yangilanib turilishi kerak.

Kompyuter (Computer) inglizchadan o‘zbekchaga o‘girilganda "hisoblovchi" ma’nosini bildiradi, yani hisoblash ishlarini bajaruvchi qurilmadir.

Inson ongli faoliyati davomida o‘zining jismoniy ishlarini bajaradigan har xil texnik qurilmalar, asbob- uskunalar yaratishga harakat kilgan. Shu jumladan, aqliy faoliyatning ishlarini ham bajaradigan, asosan hisob - kitob ishlarini bajaradigan texnik qurilmalar yaratishga harakat qilgan. Eng qadimgi hisob asboblardan biri schyotdir.

1642 yili fransuz matematik va fizik olimi B.Paskal qo‘shish va ayirish amallarini bajaradigan mexanik mashina yaratgan.

1673 yilda nemis olimi Vilgelm Leybnis to‘rt arifmetik amallarni bajaradigan mexanik arifmometrni yaratdi.

XIX asrda arifmometr juda keng tarqaldi. Uni hatto eng murakkab hisob ishlariga ham qo‘llay boshlandi. Bunday hollarda oldin qilinadigan ishlarning bajarilish ketma - ketlik qo‘llanmasi to‘liq yozilib olinar va shu asosda ish bajarilar edi. Odatda bu qo‘llanmani bajariladigan ishning dasturi deyiladi.

1834 yili angliyalik olim Ch. Bebbidj analitik mashina ixtiro qildi. Bu mashina dastur asosida ishlaydigan birinchi hisoblash mashinasining loyihasi edi.

Keyinchalik 1883 yili Ch.Bebbidj hozirgi zamondagi hisoblash mashinasining g‘oyasini va loyihasini yaratdi, ammo o‘sha davr texnika darajasi bunday mashinalarni yaratish imkonini bermas edi. Uning fikriga ko‘ra bu qurilma hamma hisoblash ishlarini odamning ishtirokisiz o‘zi avtomatik ravishda bajarishi kerak edi. Buning uchun u qurilma hisoblash ishining dasturini tushunib, shu dastur asosida hamma ishlarni bajara olishi kerak edi.

Sonli hisoblash mashinasi tuzilishining asosiy prinsiplarini amerikalik matematik Djon fon Neyman, G.Goldsteyn va A.Berks ishlab chiqdilar. Ularning va Ch.Bebbidj g‘oyasi bo‘yicha hisoblash mashinalarining ishlashi ikki prinsipga

asoslanishi kerak:

- Masalani odamning ishtirokisiz yechish dastur asosida olib borilishi;
- Masalani yechish uchun kerak bo‘ladigan hamma boshlang‘ich va oraliq ma’lumotlar hamda masalani yechish dasturlarini saqlab turishi.

Buning uchun yaratilajak hisoblash mashinasi quyidagi qurilmalardan iborat bo‘lishi lozim edi:

- boshlang‘ich ma’lumotlarni, oraliq qiymatlarni hamda masalani yechish dasturini saqlab turadigan qurilma. Hozirda bunday qurilma xotira deb yuritiladi;
- ish bajaradigan qurilma. Odatda u arifmetik - mantiqiy qurilma deyiladi;
- dastur bo‘yicha ish bajaradigan va qurilmalarning ishlashini odamning ishtirokisiz boshqarib boradigan qurilma. U boshqarish qurilmasi deyiladi. Hozirgi paytda arifmetik - mantiqiy qurilma va boshqarish qurilmasini birgalikda prosessor yoki markaziy prosessor deb yuritiladi;
- boshlang‘ich ma’lumotlarni va ishlash dasturini xotiraga kiritadigan va ish natijasini tashqariga chiqarib beradigan qurilma. U kiritish va chiqarish qurilmasi deb yuritiladi.

XX asr boshlariga kelib angliyalik olim A.Tyuring va amerikalik olim E. Post hisoblash mashinasining nazariy asosini yaratgandan keyin hisoblash mashinasi asri boshlandi.

1930 yili amerikalik olim X. Atanasov va K. Berrilar elektron xotira, qo‘shish va ayirish qurilmalaridan iborat elektron hisoblash mashinasini yaratdilar.

1937 yili amerikalik olim X. Atanasov hisoblash mashinasi sanoq tizimi uchun ikkilik sanoq tizimini ishlatish g‘oyasini berdi va bu yo‘nalishda bir necha patentlar ham oldi.

1941 yilda olmon injeneri K.Suze Ch.Bebbidj g‘oyasi bo‘yicha birinchi hisoblash mashinasini yaratdi.

1943 yilda Ch.Bebbidj g‘oyasi bo‘yicha amerikalik G.Ayken elektro-mexanik relelar yordamida "MARK-1" nomli analitik hisoblash mashinasini yaratdi.1943 yildan boshlab Amerikada bir guruh mutaxassislar shu g‘oya bo‘yicha relelar o‘rniga elektr lampalardan foydalanib hisoblash mashinasini yaratishga kirishishdi.

Ularning yaratgan mashinalari "MARK-1" mashinasidan ming martacha tez ishlar edi.

Shundan keyin dunyoda XX asr 50-yillariga kelib Amerika, Angliya, Germaniya va sobiq SSSRda birinchi elektron hisoblash mashinalari yaratila boshlandi.

1945 yili Germaniyada K.Suze tomonidan “S-4”, 1949-51 yillarda sobiq SSSR da S.Lebedev rahbarligida “MESM” va 1950 yili Angliyada “AKE” kompyuteri yaratildi.

Hisoblash mashinasi - kompyuterlarning ishlash prinsipini umumiy holda tushunarli va sodda qilib bergan olim mashxur Djon fon Neymandir. Bu prinsipni odatda fon Neyman prinsipi deb ham yuritiladi. U ikki prinsipdan iborat:

- hamma kerakli ma’lumotlarni va masalani yechish dasturlarini yagona xotirada saqlab turish;
- kompyuterni dastur yordamida boshqarish.

Hozirgi zamon kompyuterlarining tuzilishi boshqacharoq bo‘lib, arifmetik-mantiqiy qurilma bilan boshqarish qurilmalari birgalikda markaziy prosessor qurilmasi deb yuritiladi. Ishlash prinsipida ham farqi bo‘lib, bir nechta prosessorlar bilan bir vaqtda bir qancha ma’lumotlarni parallel qayta ishlash mumkin. Dastur bilan ishlash davomida zarur bo‘lganda uning ishini to‘xtatib turib, zarur ishni bajarib, yana oldingi dastur ishini davom ettirish mumkin.

Kompyuter avlodlari

Kompyuterlar o‘zining elementlar bazalari bo‘yicha avlodlarga ajratilgan.

I avlod (1945 - 1956 yillar) kompyuterlari elementlar bazalari elektron lampalar ekanligi bilan xarakterlanadi. Bu avlod mashinalari katta zallarni egallagani holda, yuzlab kilovatt elektr energiya sarf qilar va tonnalab og‘irlikka ega hamda sekundiga 1-2 ming amal bajarar, xotirasining hajmi 1-2 ming so‘zni(ma’lumotni) saqlashga qodir edi. Bu avlod mashinalariga “Ural-1”, “Ural-2”, “BESM-1”, “BESM-2”, “M-1”, “M-2”, “M-20” kabi mashinalarni misol qilib keltirish mumkin.

II avlod (1957 - 1968 yillar) kompyuterlari elementlar bazalari

tranzistorlardan iborat edi, tezkorligi sekundiga 10-20 ming amal bajarish, xotirasining hajmi 4-8 ming so‘zni saqlashga kodir edi. Ikkinchi avlod kompyuterlari hisoblash ishidan ishlab chiqarish jarayonlarini boshqarish, iqtisodiy masalalarni yechish, harflar bilan ishlay olish «qobiliyati»ga ham ega bo‘ldi. Bu avlod mashinalariga “BESM-3”, “BESM-4”, “Ural-16”, “Minsk-22”, “IBM-608”, “BESM-6” misol qilib keltirish mumkin.

III avlod (1969 - 1980 yillar) kompyuterlarining elementlar bazalari integral sxemalardan iborat bo‘lib, tezkorligi sekundiga 10 mingdan boshlab, shu avlodning eng oxirgi mashinalari 2-2.5 million amal bajarishgacha yetdi. Xotirasining hajmi ham 8-10 ming baytdan(bu avlod xotira o‘lchami xalqaro o‘lcham baytlarda beriladigan bo‘lgan) 8 million baytlargacha yetdi. Bu avlod mashinalariga ES (yagona seriya) kompyuterlari - “ES-1010”, “ES-1020”, “ES-1030”, “ES-1035”, “ES-1050”, “ES-1060”, “ES-66” larni misol qilib ko‘rsatish mumkin.

IV avlod (1981 - 1990 yillar) kompyuterlarining elementlar bazalari katta integral sxemalar (KIS)dan iborat.Ularning tezkorligi sekundiga 6,5 million amal bajarishgacha yetdi, xotirasinig hajmi 64M baytgacha kengaydi. Bu avlod mashinalariga Super EHMlar, “Elbrus”, “1-KB”, “IBM PC” kabi kompyuterlarni ko‘rsatish mumkin.

V avlod (1990 yillardan boshlangan) kompyuterlarining elementlar bazalarini o‘ta katta integral sxemalar (O‘KIS) tashkil qiladi. Bu avlod kompyuterlari hozirgi zamonda keng qo‘llaniladi. Bu avlod kompyuterlari elektron va yorug‘lik nurlari energiyasidan foydalanishga, tuzilishi esa lazer texnikasiga, nurlanuvchi diodlarga asoslangan. Amal bajarish tezligi sekundiga 1 milliardgacha, xotirasinig hajmi 10 millliondan 3-4 milliard (Gbayt) baytgacha kengaydi.

Shaxsiy kompyuterlarining yaratilishi texnikada revolyusion harakterga ega bo‘ldiki, ular ommabob hisoblash mashinalariga aylanib qoldi.

Hozirgi paytda ishlab chiqarish va kundalik hayotda dunyoda 100 millionlab shaxsiy kompyuterlar ishlatilmoqda.

Kompyuter – inglizcha so‘z bo‘lib, hisoblovchi deganidir. Uning asosiy vazifasi turli ma’lumotlarni qayta ishlashdan iboratdir. Kompyuterlarning turli xillari mavjud – raqamli, analogli (uzluksiz), maxsuslashtirilgan va h.k. Raqamli (sifrovoy) kompyuterlardan foydalanish o‘zining osonligi, bajariladigan amallarning universalligi va aniqliligi kabi ko‘rsatgichlari bilan ajralib turadi.

Hozirda rivojlangan mamlakatlarda kompyuterlarning 5 guruhi keng qo‘llaniladi:

1. Super kompyuterlar – yadro sinovlarini va qurollarini modellashtirishda qo‘llaniladi. Ular sekundiga 10 trl. operatsiya bajaradi.

2. Katta kompyuterlar (Manfreame Computer) - fan va texnikaning turli sohalariga oid masalalarni yechishga mo‘ljallangan. Uning tezligi va xotirasi super kompyuterlarnikidan pastroq. Masalan: AQShda CRAY, Yaponiyada – M1800, Rossiyada ES-9000.

3. Mini kompyuterlar – asosan xarbiy maqsadlarda ishlatiladi. Hajmi va tezligi jihatidan katta kompyuterlardan biroz pastroq. Shaxsiy kompyuterlarga o‘xshash.

4. Shaxsiy kompyuterlar – jamiyatimizning turli sohalarida ishlatiladi. Hozirgi kunda PENTIUM toifasidagi prosessorli kompyuterlar keng tarqalgan. Ularning tezligi 750; 1000; 3000 MGS, operativ xotirasi 64; 128; 256 MB, …, 1 GB gacha, doimiy xotirasi 40; 80; 120 GB ... 1 TB gacha bo‘ladi. IBM, Compaq, Hewleft, Toshiba, Apple, Siemens va h.k. firmalaridan chiqqan kompyuterlar oq yasalgan deyiladi. Malayziya, Xitoy, Tayland kabi mamlakatlarda ishlab chiqarilgan kompyuterlar sariq yasalgan deyiladi.

5. Noutbuk kompyuterlar (Notebook)- hajmi ancha kichik bo‘lgani bilan bajaradigan amallarining soni va xotirasi jihatidan shaxsiy kompyuterlar bilan tenglashmoqda.

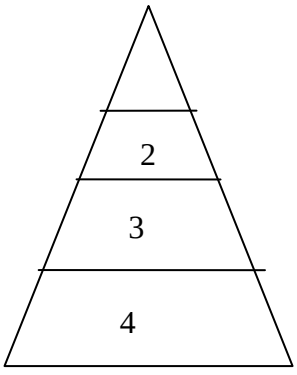
XX asrning 40-yillarida 3ta yirik davlatlarda elektron hisoblash mashinalarining (EHM) yaratish ustida ish boshlangan. Bular: SSSR, AQSh va Angliya. 1-EHM 1943 yili Angliyada ingliz matematigi va injeneri Alan Tyuring tomonidan yaratilgan. Lekin unchalik shuxrat qozona olmadi. 1945 yil AQShda

fizik va matematik fon Neyman tomonidan 1-arifmetik EHM yaratilgan (ENIAC). 1946 yili SSSRda akademik Lebedev boshchiligida EHM yaratilgan. Hozirgi shaxsiy kompyuterlarning arxitekturasi fon Neyman prinsiplari asosiga tayanadi. Ular quyidagilar:

1. Arifmetik – logik (mantiqiy) qurilmaning mavjudligi (ALU);
2. Boshqaruv qurilmasining mavjudligi (ustroystvo upravleniye UU);
3. Xotiraning mavjudligi;
4. Axborotlarni kiritish - chiqarish qurilmalarining mavjudligi.

Arifmetik-logik qurilma va boshqaruv qurilmasi birlashib, markaziy prosessorni tashkil etadi (MP), va u kompyuterning asosiy qurilmasi deb hisoblanadi. MP tizimni shinalar orqali tekshirib va boshqarib turadi. Shinalar 2 xil bo‘ladi: adress shinasi va ma’lumotlar shinasi. Adress shinasi xotira yacheykasini yoki kiritish - chiqarish portlarini tanlaydi. Ma’lumotlar shinasi esa MPga kattaliklarni uzatish va undan olish vazifasini bajaradi.

Kompyuterning tez ishlashi uchun xotirani bir necha darajaga bo‘lish zarur: juda tez ishlaydigan va sekinroq ishlaydigan.



1- MP registrlari

2- Kesh xotira

3- Operativ xotira

4- Doimiy xotira

Registrlar – juda tez ishlaydigan, lekin kam hajmga ega bo‘lgan xotiradir (1ta registrning hajmi 8, 16, 32 va h.k. bit bo‘lishi mumkin). Registrlarga qarab kompyuterlar ham farqlanadi. Agar programmist assembler tilida ishlayotgan bo‘lsa, registrlar ishini boshqarishi mumkin.

Kesh xotira – tez ishlaydigan xotira satriga kirib, bir necha yuz Kbdan bir necha MB bo‘ladi (Masalan: 256 KB-4MB). Uning vazifasi operativ xotirada

yaqinda ishlatilgan ba’zi axborotlarni, komandalarni vaqtinchalik saqlab turish. Aynan qaysi axborotni saqlash kerakligini tizimning o‘zi hal qiladi. Dasturchi bu jarayonni boshqara olmaydi.

Operativ xotira – RAM (Read / Write Memory – yozish/o‘qish uchun xotira) hajmi kichik, tezligi katta bo‘ladi. RAM-Random Access Memory

Doimiy xotira – ROM (Read Only Memory – o‘qish uchun xotira) katta hajmga ega, tezligi esa pastroq bo‘ladi.

Kompyuterlar avval tuzilgan dasturlar asosida ishlaydi. O‘z navbatida masalani yechish uchun biror dasturlash tilida yozilgan buyruqlar ketma-ketligini tarjimon dasturlar yordamida kompyuter tiliga o‘tkaziladi. Kompyuter tili 0 va 1 lardan tashkil topgan ma’lum qoidalar asosida yozilgan ketma-ketlikdan iborat. Dastur avval kompyuter xotirasiga kiritiladi va buyruqlar bajarila boshlaydi. Boshqaruv qurilmasi amallar ustidan nazorat o‘rnatadi. Amallarni prosessor bajaradi. Odatda kompyuter 2 qismdan tashkil topgan deyiladi: Hardware - qattiq qism, ya’ni texnik vosita - kompyuter , Software – yumshoq qism, ya’ni dasturiy ta’minot.

EHMning xotira yacheykasi 16, 32, 64 ta 2lik razryadlardan iborat bo‘ladi. Bu 2lik razryad bit deb ataladi va u axborotning eng kichik birligi hisoblanadi. 8 bit – 1 bayt bo‘ladi. Bayt nomeri adress deyiladi. 2 bayt 1 mashina so‘zi bo‘ladi.

Muhokama savollari

1. Kodlash va shifrlashga oid asosiy tushunchalar
2. Kriptografik tizim tushunchasi
3. Ma’lumotlar himoyasi va antivirus himoyasi
4. Kompyuterning yaratilish tarixi, avlodi.
5. Fon-Neyman arxitekturasi.
6. Kompyuterlarning klassifikasiyasi.
7. Kompyuterlarning ishlash asoslari, prinsiplari.

Nazorat savollari

1. Ma’lumotlarni ochish va himoyalash muammolari
2. Kodlash va shifrlashga oid asosiy tushunchalar
3. Viruslarning asosiy turlari
4. Kompyuter viruslari bilan kurashish
5. Kompyuterlarning yaratilish tarixi

1.4 Komputerda masala yechish bosqichlari. Sanoq tizimlari

Informatikada masalani yechish tushunchasi deganda, axborotlarni qayta ishlab, natijani oldindan belgilangan ma’lum bir ko‘rinishga olib kelish tushuniladi. Informatika fani 3 ta tarkibiy qismdan iborat:

1. Masalani to‘g‘ri yechib olish uchun kerakli bilim va mahorat (algoritm va usul);

2. EHMda mavjud bo‘lgan tarjimon dasturlardan foydalanib, masalaning dasturini tuzish;

3. Dasturi tuzilgan masalani EHMda yechish va natijani olish.

Masalani kompyuterda yechish bir necha bosqichlarni o‘z ichiga oladi:

1. Masalaning qo‘yilishi va maqsadi;
2. Masalaning matematik ifodasi;
3. Masalani yechish uchun kerakli usulni aniqlash;
4. Yechish algoritmini tuzish;
5. Algoritmga asosan biror algoritmik tilda dastur tuzish;
6. Dasturni kompyuterga kiritish;
7. Dasturda yuzaga kelgan xatoliklarni tuzatish;
8. Olingan natijani izohlash, tahlil qilish.

1-3 bosqichlar mutaxassisning malakasi va bilimiga bog‘liqdir. Bu ish ustida mutaxassislar bir necha oy yoki yillab izlanishlari mumkin. Buning uchun kerakli ma’lumotlar va ular orasidagi bog‘lanishlar aniq ifodalangan bo‘lishi zarur.

Har qanday hisoblash mashinalarining arifmetik asosi bo‘lgan sanoq tizimlari bilan tanishamiz. Shu kungacha matematika darslarida turli-tuman hisoblarni o‘nta raqamdan, ya’ni 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 raqamlaridan foydalanib bajarib keldik. Shuning uchun ham bu o‘nlik sanoq tizimi deb ataladi. Sanoq tizimida sonlarni yozish uchun qo‘llaniladigan raqamlar soni ushbu sanoq tizimining asosi deb yuritiladi. Kundalik hayotimizda qo‘llanib kelinayotgan sanoq tizimining asosi 10 ga teng.

Sanoq tizimlari, ularning tuzilishi va ulardagi amallar haqidagi tushunchalarni ko‘rib chiqamiz.

R simvolli alifbo X ning to‘g‘ri yozilishi va shu alifbodagi sonlarni shu *r* simvollar orqali qayta ishlovi sanoq tizimi deyiladi. Tizimdagi *r* ga asoslangan X soni $(x)_r$ yoki x_r deb belgilanadi.

Har bir sanoq tizimi – bu, kodlash va dekodlash opreratsiyasini bajaradigan sonli qiymatlarni (to‘plamlarni) kodlashdir, ya’ni bunda ixtiyoriy sonli qiymatga kodli ko‘rinishni topish va ixtiyoriy kodli yozishga unga mos ravishda sonli qiymatni tiklash kerak bo‘ladi.

Hamma sanoq tizimi bir umumiy prinsipga asosan tuziladi: tizimning asosi *r*-kattaligi aniqlanadi, *x* ixtiyoriy soni esa *r* - og‘irlik darajasining kombinatsiyasi ko‘rinishida 0 dan n-chi darajaga quyidagicha o‘tadi:

$$(x)_{10}=x_n p^n+x_{n-1}p^{n-1}+\ldots+x_1p^1+x_0p^0.$$

Informatikada eng ko‘p ishlatiladigani bu 10 lik sanoq tizimidan tashqari, - bu: 1) ikkilik X={0,1}; 2) sakkizlik X={0,1,2,3,4,5,6,7}; 3) o‘n oltilik, X={0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}, bu yerdagi simvollar A, B, C, D, E, F o‘nlik sanoq tizimidagi 10, 11, 12, 13, 14, 15 ni bildiradi.

Misol. $1102_2=1*2^3+1*2^2+0*2^1+1*2^0=8+4+1=13_{10}$,

$157_8=1*8^2+5*8^1+7*8^0=64+40+7=111_{10}$,

$A6F_{16}=10*256+6*16+15*1=2671_{10.}$.

Ko‘pincha, sanoq tizimidagi hamma sonlar (yoki alifbo simvollari) yozuvdagi joylashgan joyiga bog‘liq bo‘ladi. Bunday sanoq tizimi pozitsion deyiladi; aks holda nopozitsion deyiladi.

Misol. Nopozitsion tizim – qadimgi rim alifbosini sonlardagi ifodasi, joylashgan joyidan qat'iy nazar quyidagi ko'rinishga ega bo'ladi, X={I(1), V(5), X(10), L(50), C(100), D(500), M(1000)}. Rim sonlarining ko'rinishi (qavsda – oddiy o'nlik ekvivalentlari): III(3), IV(4), V(5), VI(6), IX(9), XI(11), DCL(650). Bu tizimdagi yozuv ikki yoqlama konkatenatsiya orqali hosil qilinadi, bunda o'ng taraflama konkatenatsiya qo'shilib borish orqali, chap taraflama konkatenatsiya olib tashlash orqalidir (masalan, IV va VI). Arifmetik operatsiyalarni bajarish joylashuviga qaramaydi (masalan, XIV+IV=XVIII).

O'nlik sanoq tizimidagi kasr sonlarga o'tish quyidagi formula orqali amalga oshiriladi.

Misol. $110{,}001_2 = 1x2^2 + 1x2^1 + 0x2^0 + 0x2^{-1} + 0x2^{-2} + 1x2^{-3} = 6{,}125_{10}$;

$A, B_{16} = Ax16^0 + Bx16^{-1} = 10x1 + 11x0{,}0625 = 10{,}6875_{10}.$

O'nlik sanoq tizimidan r-sanoq tizimiga o'tish yo'llari:

1. X sonining butun qismini ajratib olamiz, avval $[X]_{10}$ ni butun qismini *r* -ga bo'lamiz, so'ngra kasr qismini *r*-ga bo'lamiz, to shu sondan kichik son qolguncha va hosil bo'lgan qoldiqlari *r*-sanoq tizimini hosil qiladi.

2. $[X]_{10}$ sanoq tizimidagi sonning kasr qismi (mantissasi) ko'paytiriladi, so'ngra hosil bo'lgan *r*-dagi sonlar ko'paytiriladi, toki 0 ga teng bo'lgan mantissa hosil bo'lguncha, yoki kerakli $[X]_r$ sonlarining keraklisi hosil bo'lguncha. $[X]_r$ ning ko'rinishi birinchi ko'paytirishdan to oxirgi ko'paytirishga hosil bo'lgan sonlarning butun qismidan hosil qilinadi;

3. Natija quyidagi ko'rinishga ega bo'ladi $(X)_r=[X]_p, \{X\}_p$.

Misol. Topilsin: $12{,}8=?_2$. Yechimi:

1. Butun qismini olib o'tamiz:$12_{10}=1102_2$;

2. Kasr qismini olib o'tamiz: 0,8*2=1,6; 0,6*2=1,2; 0,2*2=0,4;

0,4*2=0,8; $0{,}8_{10}=0{,}1100110_2$;

3. Natija: $12{,}8=1100{,}1100110011_2$.

Misol. Topilsin $29{,}25_{10}=?_8$. Natija quyidagi ko'rinishga ega: 1) $29_{10}=35_8$;

2) $0{,}25_{10}=0{,}2_8$; 3)$29{,}25_{10}=35{,}2_8$.

Misol. Topilsin $79{,}26_{10}=?_{16}$. Natijasi: 1) $79_{10}=4F_{16}$; 2) $0{,}26_{10=0{,}40}16_;$

3) $79,26_{10}=4F,4_{16}$. Kasr qismini olib o‘tganimizda verguldan keyingi ikkita son olinadi va qolgani taxminiy butunlab olinadi, chunki aniq olib o‘tishni iloji bo‘lmaydi. 2-likka 8-likka teskari yo‘l bilan, 2-likdan 16-likka teskari yo‘l bilan, 8-likdan 16-likka teskari yo‘l bilan va yana orqaga qaytiladi. Quyida keltirilgan jadval ishlatiladi:

Tizim asosi			
10	2	8	16
0	0	000	0000
1	1	001	0001
2	-	010	0010
3	-	011	0011
4	-	100	0100
5	-	101	0101
6	-	110	0110
7	-	111	0111
8	-	-	1000
9	-	-	1001
10	-	-	1010
11	-	-	1011
12	-	-	1100
13	-	-	1101
14	-	-	1110
15	-	-	1111

8-lik tizimiga o‘tganda sonlar 3 bitga guruhlanadi, 16-likka o‘tganda esa 4 bitga guruhlanadi. Agar kerak bo‘lsa, 0 qo‘shish mumkin (chapdan butun qismiga va o‘ngdan mantissaga) yoki tashlab yuborish kerak bo‘ladi.

Misol. Bir sanoq tizimidan ikkinchisiga o‘tishni ko‘rib chiqamiz:

1. 2-likdan 8-likka o‘tish:

$101,10111_2=\underline{101}, \underline{101}\ \underline{110}_2=5,56_8$;

$5_8,\ 5_8\ \ 6_8$

2. 8-likdan 2-likka o‘tish:

$6,24_8=\underline{6,}\ \ \underline{2}\ \ \underline{4}_8=110,0101_2$;

$110_2, 010_2\ \ 100_2$

3. 2-likdan 16-likka o‘tish:

$101,10111_2=\underline{0101},\ \underline{1011}\ \underline{1000}_2=5,B8_{16}$;

5_{16}, $11(B)_{16}$ 8_{16}

4. 16-likdan 2-likka o‘tish:

$1A,F3_{16}=\underline{1}\ \underline{A}\ ,\ \underline{F}\ \underline{3}_{16}=11010,11110011_2$.

0001_2 1010_2 1111_2 0011_2

2-lik sanoq tizimida qo‘shish

$0+0=0,0+1=1,1+0=1,\ 1+1=2_{10}=10_2$ (1 yuqori razryadga o‘tadi)

2-likda ayirish

0-0=0, 1-0=1, 1-1=0, 0-1=10-1=1 (1 yuqori razryaddan olinadi)

2-likda ko‘paytirish quyidagicha

0*0=0,0*1=0, 1*0=0, 1*1=1.

2-likda bo‘lish

0:0=aniqmas, 1:0=aniqmas, 0:1=0, 1:1=1.

Tizimdagi sonning *r* – ga asoslangan qaytish kodi, shu tizimdagi hosil bo‘lgan sonning almashtirilganiga, razryaddagi har bir simvol tizimdagi sonning maksimal to‘ldirilishiga aytiladi. (ya’ni *r*-1).

Qo‘shimcha kod = qaytish kodi + quyi razryaddagi bir.

Misol.

1. 10011=ikkilik soni,

01100= ikkilikdagi qaytish kodi,

01101= ikkilikdagi qo‘shimcha kod;

2. 457=sakkizlik soni,

321=yordamchi kod;

3. A9=o‘n oltilik soni;

57=yordamchi kod.

Yordamchi kod yordamida ayirish: Razryaddagi ayirmadan yordamchi kodni topish va ayiriluvchiga qo‘shish. Ayirma natijasi hosil bo‘lgan yig‘indi bo‘ladi, yuqori razryaddan tashqari.

Misol. To‘g‘ridan to‘g‘ri ayirish orqali va qo‘shish orqali amalni bajaramiz (yordamchi kod orqali):

110110_2 $+110110_2$

10101_2 1011_2 --------yordamchi kod

100001_2 ---------ayirma 1100001_2--------yig‘indi tashlab yuboriladigan son

Matematikada butun sonlar va ularning analogi *n*-razryadli arifmetikada ayniyatlidir (son qiymati bo‘yicha) keltirilgan razryad bo‘yicha. Bunda insonning fikrlash darajasi va *n*-razryadli arifmetik sohasi hisobga olinadi:

1. Cheksiz va hisoblanuvchan butun sonlarning ko‘rinishi [-N;+N] kesmada keltiriladi, bunda N - arifmetikada keltirilgan maksimal son (bu yerdagi ko‘p nuqta n-ga teng bo‘lgan umumiy sonlar birligi): $N=(111\ldots1)_2$;

2. Cheksiz va hisoblanuvchan aniq sonlarning $(-\infty;+\infty)$ da sonlar o‘qida bir tekis va zich joylashuvi, *n*-razryadli arifmetikada notekis zichlikda joylashuvi (kichik sonlar orqali siqish);

3. Ko‘pchilik aniq sonlarda 0 o‘z sohasida ko‘p sonlarni tashkil etadi, *n*-razryadli arifmetikada esa 0 ajratilgan holda keltirilgan. Oddiy arifmetika nuqtai nazaridan, masalan, (-1;1) chegarasida juda zich joylashgan nuqtalar bor va har qanday tekislikda shu nuqtalar ichidan bir nuqta mavjud bo‘ladi. Bunday arifmetikani regulyar arifmetika deyiladi. Mashina arifmetikasi quyidagi xususiyatlarga ega. U regulyar emas – nuqta intervalida 0 ga yaqinlashganda qo‘shilishib ketadi; bundan tashqari bu intervalda nuqta x “ajratilgan” holda keladi – agar uning ixtiyoriy sohasida (x-a; x+a), bu yerda a-mashina 0 dan oshmaydigan sondir, bu intervalda boshqa nuqtalar yo‘q (x dan tashqari). Ehtimollar nazariyasi tarafidan qaraganda, sonlarni zichligini taqsimlash regulyar va noregulyar arifmetikada turlicha, shuningdek, butun va haqiqiy sonlarni zichligini taqsimlash ham arifmetikada xuddi shu. Ko‘pchilik haqiqiy sonlar mashina arifmetikasida (arifmetika razryadida aniqlangan) ratsional sonlar to‘plami sifatida beriladi. Bu

to‘plamlarning boshqa xususiyatlari ham bor (masalan, operatsiyalar bilan bog‘langan), lekin yuqorida keltirilganlari – asosiylaridir.

Sonlarni odddiy va mashina arifmetikasida (*n*-razryadli) berilishini farqi, kompyuterning “matematik” imkoniyatlarini chegaralaydi, shuningdek, “kompyuterning” imkoniyatlari matematikadagi matematik usul, algoritmlarni kompyuterda ifodalanishini chegaralaydi.

Shuni nazardan qochirmaslik kerakki, nazariy matematikadagi aniqlik tushunchasi – mavhumdir, amaliy matematikada ham, u yo‘q joyda ham, hayoliy aniqlik yuzaga kelishi mumkin, - agar ko‘rib chiqilayotgan hisoblash resurslaridagi xatoliklarni kerakli qiymatlarini chegarasidagi kelishuvlari yo‘q bo‘lsa, masalan, ishlash va vaqtga nisbatan aytiladigan bo‘lsa, bu boshqarish strategiyasi xatoliklardan holi emas. Bu diapazondagi *r* ga asoslangan *n*-razryadli sanoq tizimi $\left|(x)_p\right| \le p^n - 1$ chegarasida yotadi, kasr sonlarni keltirish uchun bu diapazon yanada kamayadi, negaki, razryadning bir qismini mantissa orqali ifodalash kerak bo‘ladi. Xuddi shu usulda “sezmaslik sohasi” atalmish arifmetikadagi *n*-razryadli sonlar formasi keltiriladi.

1937 yilda Konradom Suze bu diapazonni kengaytirish uchun, arifmetikada keltirilgan ikkilik sonlarni, shuningdek keltirilgan sonlarni aniqligini oshirish uchun, keltirilgan sonlar oquvchan, normal holatga keltirilgan formada keltirilgan, *x* soni quyidagicha keltirilgan: $x = m * p^k$, bunda m-mantissa soni, *k*-butun son tartibi,

$$p^{-1} \le |m| < 1.$$

Ikkita son berilgan bo‘lsin, $x = m * p^k$, va $y = n * p^l (k > l)$. U holda operatsiya bajarilish natijasini tekshirish mumkin:

$X+y=(m+n* p^{l-k}) * p^k$,

$x-y=(m-n* p^{l-k}) * p^k$,

$x*y=(m*n)* p^{k+l}$,

$x/y=(m/n)* p^{k-l}$.

Agar *n*-razryaddan, sonlar ko‘rinishi sifatida olinadigan, m-ikkilik razryadini mantissa ostiga olish, *k* – tartib osti, bir razryad – son belgisi ostiga va bir razryad – tartib belgisi ostiga o‘tsa (masalan, 0-musbat va 1- manfiy), u holda oquvchi vergulli sonlar formasida keltirilganlar diapazoni birdaniga ko‘payadi (m+k+2=n):

$$-(0.111...1)_2 * (10)_2^{+(111..1)_2} \le x \le +(0.111...1)_2 * (10)_2^{+(111..1)_2}$$

(ko‘p nuqta *k* birligini bildiradi).

Quyi chegaradagi musbat sonlardan kichik sonlar va yuqori chegaradagi manfiy sonlardan katta sonlar 0 ga teng deb olinadi va bir biridan farqlanmaydi. Yuqori chegaradagi musbat sonlardan katta sonlar musbat cheksizlikka tenglashtiriladi (quyi chegaradagi manfiy sonlardan kichigi – manfiy cheksizlikka tenglashtiriladi). Shuning uchun ikki xil kattalikdagi chegaralangan razryadli arifmetik sonlarni taqqoslash aniq bo‘lmagan natijalarga olib keladi, shuningdek matematik nuqtai nazardan qaraganda shu tizimdagi bir biriga teng ikkita sonlar uchun ham. Bunday tasavvur EHM da saqlashni qulayligi, sonni o‘zini saqlash emas, balki uni belgisini, mantissasini, tartibini va tartib belgisini, va sonlar operatsiyasi Ob’yekt operatsiyasi bilan solishtiriladi. Bunday Ob’yekt operatsiyasi oddiy: belgilarni solishtirish, kattalashtirish, tartibni kichiklashtirish, matissalarni ko‘paytirish, normaga keltirish, ya’ni xulosa qilib aytganda, oddiy siljitish, tenglashtirish operatsiyalarini taqqoslash razryadlarini keltiriladi.

Misol. Sanoq sitemasidagi 3-asosli eng katta va eng kichik 5-razryadli butun sonni hisoblab chiqarish kerak. N-razryadli eng katta butun sonni *r*-asosli sanoq tizimida quyidagicha yozish mumkin:

$$x_p^{\max} = \sum_{i=0}^{n-1}(p-1)p^i = \sum_{i=0}^{n-1} p^{i+1} - \sum_{i=0}^{n-1} p^i = p^n - 1$$

Eng kichik *n*-razryadli butun son bu tizimda:

$$x_{\min} = -x^{\max} = 1 - p^n.$$

Xuddi shuningdek, 3 ga asoslangan sanoq tizimida va 5 razryadli sonlarda quyidagi sonlar diapazonini keltiramiz:

$$-242 = 1 - 3^5 \le (x)_4 \le 3^5 - 1 = 242.$$

Formulalarni yanada ixcham koʻrinishga keltirish mumkin. Masalan, ikkilik tizimi uchun

$$\underbrace{a = 111...1 = 2^n - 1, b = 1 - 2^n}_{n}$$

Sakkizlik sanoq tizimida esa bu sonlar

$$\underbrace{a = 777...7 = 8^n - 1, b = 1 - 8^n}_{n}$$

Bu xildagi sanoq tizimi formasidagi “qulaysizliklar” “xavfli” vaziyatlarni yuzaga keltirishi mumkin:

1. Agar son yetarlicha oz boʻlsa, masalan, *a*=0.12E+00, u holda a kiritilgan eng kichik intervalda ixtiyoriy sonda keltirilishi , 0.120000001 yoki 0.199999999 deb, bu holda ham tengsizlikka taqqoslash mumkin emas (haqiqiy sonlarni qoʻzgʻaluvchi vergul formasi bilan taxminiy solishtirish xavflidir);

2. Operatsiyalarni bajarish tartibi natijaga ta’sir etishi mumkin, masalan, 4-razryadli arifmetikada fiksirlangan vergul bilan 20.0000+0.0001=20.0001, lekin bunda 0.2000E+02+0.1000E-05=0.2000E+02;

3. Katta sonlarni qoʻshishda (koʻpaytirishda) “tartibni oshib ketishi” vaziyati yuzaga kelishi mumkin yoki kichik sonlarni qoʻshishda (koʻpaytirishda) “tartibni yoʻqolishi” vaziyati yuzaga kelishi mumkin, 0.6000E+39*0.1200E+64=0.9999E+99 (yoki aniqlanmagan), shuningdek 0.6000E-35*0.0200E-65=0.9999E-99 (yoki aniqlanmagan), aniq razryadli oʻnlik arifmetikaga mos holatda;

4. Suriluvchi vergulli sonlarni qoʻshishda (barcha operatsiyalar qoʻshish orqali amalga oshiriladi) navbatdagi mantissalarni qoʻshish uchun tartibni tenglashtirish yuzaga keladi, darajalarni tenglashtirishda esa kichik razryadlarni yoʻqotish vujudga keladi, masalan, bunday vaziyat bir “katta sonni” ikkinchi “eng kichik songa” qoʻshishda yuzaga keladi.

Sonlar tizimini bir formaga keltirishni koʻp usullari mavjud (koʻpincha sun’iy).

Misol. Faktorial sanoq tizimida butun sonlar faktoriallarning chiziqli kombinatsiyasi orqali yoziladi, masalan,

2457=2*3!+4*2!+5*1!+7*0!(0!=1,n!=1*2*3*...*n). Bu tizim (shartli) pozitsiondir. Negaki 0!=1!=1, *n*-razryadli soni ikkita eng kichigi uchun $x=x_n x_{n-1}...x_2 x_1 (n)>1$) bu soni faktoriallarga taqsimlasak $x_2 *1!+x_1 *0!= x_1 *1!+x_2 *0!$, shuning uchun ham bu razryadlar og‘irligi pozitsiyaga bog‘liq emas.(shuning uchun *n*>1 da bu son nopozitsion shartli ravishda hisoblanadi). Faktorial sanoq tizimidan o‘nlik sanoq tizimiga o‘tish formulasi:

$$x=x_n x_{n-1}...x_1 = x_n *(n-1)!+x_{n-1} *(n-2)!+...+x_1 *0!$$

Sanoq tizimining rivojlanish tarixi ancha qiziqarlidir. Bir xil faktlarnigina keltirib o‘tamiz. Ilgari hisob kitob qo‘l barmoqlari yordamida amalga oshirilgan (oldin beshtalikda, so‘ngra o‘ntalikda). Bir xil mamlakatlarda 12 ga (masalan, Buyuk Britaniyada – 12 shilling) va 20 ga asoslangan (masalan, Fransiya – “quatre-vingts” yoki “to‘rt-yigirma” ya’ni 80; qadimgi adiglar analogli ravishda hisob bajarishgan: ya’ni “yigirma-uch” - 60) hisob kitob saqlanib qolgan va shunga o‘xshashdir.

Umuman olganda, sanoq tizimlarini ikki, pozitsion va pozitsion bo‘lmagan turga ajratish mumkin. Agar biror sanoq tizimida raqamlar qiymati tushish joyiga (pozitsiyasiga) qarab belgilansa, u holda bunday tizim pozitsion sanoq tizimi, aks holda pozitsion bo‘lmagan sanoq tizimi deyiladi.

Masalan Rim sanoq tizimidan boshqa sanoq tizimlarining hammasi pozitsion, rim sanoq tizimi esa pozitsion bo‘lmagan sanoq tizimiga misol bo‘ladi. Haqiqatdan, Rim sanoq tizimida o‘ttiz besh quyidagicha yoziladi.

X X X V
| | | |_____ qiymati besh
| | |_____ qiymati o‘n
| |_____ qiymati o‘n
|_____ qiymati o‘n

Bu yerda foydalangan X raqami uchta bo‘lishiga qaramasdan, hammasi ham o‘n qiymatga teng, ya’ni raqamning qiymati uning turish o‘rniga bog‘liq emas.

O‘nlik sanoq tizimida yozilgan 222 sonini olaylik:

2 2 2

| | |_____ikki birlik

| |_____ikki o‘nlik

|_____ikki yuzlik

Bu sonda keltirilgan uchta ikkining qiymatlari turlichadir, ya’ni o‘ngdan birinchisi ikki birlikni, ikkinchisi ikki o‘nlikni, uchinchisi ikki yuzlikni ifodalaydi. Shuning uchun u o‘nlik sanoq tizimi pozitsion tizimdir. Huddi shunday ikkilik, sakkizlik, o‘n oltilik va boshqa sanoq tizimlari ham pozitsion tizimgan misol bo‘la oladi.

Asosi o‘ndan katta bo‘lgan sanoq tizimlarida qaysi raqamlar ishlatiladi degan savol tug‘iladi. Masalan, o‘n oltilik sanoq tizimida biz bilgan o‘nta raqam yetarli emas, shuning uchun yana 6 raqami kerak bo‘ladi. bular ham o‘n oltilik sanoq tizimida bitta raqamlar uchun A B C D E F belgilarni kiritsak, o‘n oltita raqamga ega bo‘lamiz: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, va F. “16” soni esa 10 ko‘rinishda yoziladi.

Sizga ma’lumki, o‘nlik tizimning asosi bir-u nol (10) ko‘rinishda yozilgan edi. O‘n oltilik sanoq tizimining asosini har bir-u nol (10) ko‘rinishda yozdik.

Bundan keyin turli sanoq tizimlarida yozilgan sonlarni bir-biridan farq qilish uchun mos sonlarning indeksida tizim asosini ko‘rsatamiz. Masalan, 27***;2,37*;79A**;11* yozilgan bo‘lsa, bu sonlar mos ravishda o‘nlik, sakkizlik, o‘n oltilik, va ikkilik sanoq tizimining sonlarini ifodalaydi.

Har qanday asosli sanoq tizimida qisqa yozuvda berilgan sonlarni asos darajalari bo‘yicha yoyib yozish mumkin. Masalan, 451 soni 4x10*+5x10*+1x10 kabi yozish mumkin. Shu kabi quyidagilar ham o‘rinli:

Shunday qilib, soni biror asosli pozitsion sanoq tizimida ifodalash uchun bu soning o‘sha tizim asosining darajalari bo‘yicha yoyilmasining yig‘indisi shaklida yozish yetarlidir. Demak, biror sanoq tizimining asos darajalari bo‘yicha yoyilmasini mos darajalarga ko‘tarib, so‘ng qo‘shib chiqilsa, hosil bo‘lgan son o‘nlik sanoq tizimiga o‘tib qolar ekan.

Sonlarni bir sanoq tizimidan boshqasiga o‘tkazish. Ma’lumki, kompyuterda hisoblash ishlari ikkilik sanoq tizimida bajariladi va zarur bo‘lsa, natija o‘nlik sanoq tizimida olinishi mumkin.

Hozirgi zamon kompyuterlarida o‘n oltilik sanoq tizimi keng qo‘llaniladi. Endi bir sanoq tizimidan boshqasiga o‘tish bilan tanishaylik, ya’ni ixtiyoriy asosli, masalan *r* asosli sanoq tizimidan ixtiyoriy *k* asosli sanoq tizimiga o‘tishni qanday amalga oshirish kerak, degan savolga javob beramiz.

Odatda, r asosli sanoq tizimidan *k* asosli sanoq tizimiga o‘tish uchun oraliqda o‘nlik sanoq tizimidan foydalaniladi. Bunga sabab biz doim o‘nlik sanoq tizimida ishlab keldik va unga ko‘nikib ketganmiz. Shuning uchun dastlabki *r* asosli sanoq tizimidan o‘n asosli sanoq tizimiga, so‘ngra esa o‘n asosli sanoq tizimidan k asosli sanoq tizimiga o‘tiladi.

Avval berilgan *r* asosli sanoq tizimidan o‘n asosli sanoq tizimiga o‘tishga doir misollar keltiramiz.

1-misol. 101 sonini o‘nlik sanoq tizimiga o‘tkazing;

101 = 2*2 + 0*2 + 1*2 = 1*4 + 0*2 + 1*1 = 4+1 = 5 .

Demak, 101 = 5 .

2-misol. A9 sonini o‘nlik sanoq tizimiga o‘tkazing;

A9 = A*16 + 9*16 = 10*16 + 9*1 = 160 + 9 = 169.

Demak, A9 = 169 .

Endi berilgan o‘nlik sanoq tizimidagi butun sonni boshqa biror asosli sanoq tizimiga o‘tkazishni ko‘ramiz. Buning uchun ketma-ket bo‘lish usulidan foydalaniladi.

O‘nlik sanoq tizimidagi aralash sonlarning kasr qismini ham boshqa asosli sanoq tizimiga o‘tkazish mumkin. Buning uchun sonning kasr qismi sanoq tizimining asosiga ketma-ket ko‘paytiriladi.

Yangi sanoq tizimidagi kasr ko‘paytirish natijasida hosil bo‘lgan butun qismdagi raqamlar ketma-ketligi bilan ifodalanadi. Ketma-ket ko‘paytirish jarayoni kasr qismi nolga teng bo‘lguncha davom ettiriladi.

Ko‘rib o‘tilgan misollarda kasr qism nollarga aylangan. Lekin har qanday sonlarda ham osongina nollar chiqavermaydi. Hosil bo‘layotgan kasr son cheksiz davriy kasrlardan iborat bo‘lib qolishi ham mumkin. Bunday hollarda ko‘paytirish birinchi holda bir davr hosil bo‘lguncha, ikkinchi holda zarur bo‘lgan aniqlikka erishguncha davom ettiriladi.

Shunday qilib, o‘nlik sanoq tizimida berilgan aralash sonlarni biror asosli sanoq tizimiga o‘tkazish uchun dastlab ketma-ket bo‘lish bilan butun qismini, so‘ngra ketma-ket ko‘paytirish bilan kasr qismini o‘tkazib, javobni har ikkalasidan foydalanib yozilar ekan.

Umuman r sonli sanoq tizimidan k asosli sanoq tizimiga o‘tish uchun avval r asosli sanoq tizimidan o‘n asosli sanoq tizimiga, so‘ngra undan k asosli sanoq tizimiga o‘tish qoidasidan foydalanilar ekan, ya’ni bunda birinchi bosqichda asos darajalari bo‘yicha yoyib chiqish, keyingisida ketma-ket bo‘lish (butun sonda) yoki ketma-ket ko‘paytirish (kasr sonda) usullaridan foydalaniladi.

O‘n oltilik sanoq tizimidan ikkilik sanoq tizimiga va aksincha oson o‘tish mumkin. Buning uchun tetrada (to‘rtta ikkilik sanoq tizimidagi raqam) lardan foydalanilsa bo‘ladi. To‘rtta nol yoki bir raqam yordamida 0 dan 15 gacha sonni yozish mumkin. Shuning uchun ham ikkilik va o‘n oltilik sanoq tizimi orasidagi bog‘lanishni amalga oshirish mumkin. Haqiqatan, to‘rtta raqamdan iborat ikkilik sanoq tizimidagi eng katta son

1111 = 1*2 + 1*2 + 1*2 + 1*2 = 8 + 4 + 2 + 1 = 15 kabi bo‘ladi.

Misol. Tetrada yordamida 101010, 1101 sonini o‘n oltilik sanoq tizimiga o‘tkazing.

Yechish: Buning uchun verguldan boshlab butun qismini chapga qarab, kasr qismini o‘ngga qarab to‘rttadan xonalarga ajratib chiqamiz, yetmagan qismini nollar bilan to‘ldiramiz va so‘ngra to‘rtliklarga mos o‘n oltilik raqamlarni qo‘yib chiqamiz, ya’ni

0010 1010 1101

---- ----, ----- = 3A,D

3 A D

Axborotlarni raqamlar orqali ifodalash. Axborotning miqdori. Axborotni ma’lum qoida, qonun va belgilar asosida qayta ifodalash kodlash deb ataladi.

Qadimda kodlash maxfiy yozuv uchun foydalanilgan. Rim imperatori Yuliy Sezar begonalar davlat ahamiyatiga ega ma’lumotlarni o‘qiy olmasliklari uchun shartli belgilardan foydalangan. Uning shartli belgisi bo‘yicha alifbo aniq sondagi harfga o‘ngga yoki chapga surilar edi.

Masalan, biri o‘zgarmagan, ikkinchisi bir harfga chapga surilgan ikki qator o‘zbekcha harflarni yozaylik:

ABVGDEYOJZIYKLMNOPRSTUFXSCHSH’EYUYAUKGXBGDEYOJZI YKLMNOPRSTUFXSCHSH’EYUYAUKGXA

U holda bunday usul bilan “PAXTA” so‘zi “RBSUB” ko‘rinishda maxfiylashtirilishi mumkin.

Huddi shunga o‘xshash kodlashning boshqa usulini ko‘rish mumkin. Masalan, alifbo harflarni mos raqamlarga mos qo‘yib kodlash mumkin: “a” harfini 1, “b” harfini 2, “v” harfini 3 va h.k. Shu kabi davom etib, “x” harfini 35 soni bilan kodlaylik. U holda, bunday bunday kodlarda “Paxta” so‘zini 17; 1; 23; 20; 1; kabi raqamlar ketma – ketligida yozish mumkin, bu usul eng sodda kodlashdir.

Eski telegrafda, masalan axborot Morze alifbosi bilan, yani nuqta va tirelar ketma-ketligi ko‘rinishida kodlashtirishlar va uzatilar edi. Masalan Morze alifbosida STOR so‘zi kabi yozilishi mumkin. Kompyuter ixtiyoriy harfni «tanishi» uchun uning xotirasida harflar har xil usulda yozilgan bo‘lishi mumkin. Kompyuter ixtiyoriy harfni» tanishi» uchun uning xotirasida harflar har xil usulda yozilgan bo‘lishi kerak.

Shuning uchun uning qo‘lingizdagi darslikdagi matn harflarini kompyuter tanishi uchun uning xotirasida harf va belgilarning taxminan 2 ming xil ko‘rinishlarini saqlash kerak. Bu juda mushkul ish. Bu jarayonni soddalashtirish uchun barcha harflarni 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 raqamlari bilan almashtirish mumkin. Shu tarzda tinish belgilarini ham raqamlar orqali kodlash imkoniyati bo‘ladi. Masalan, nuqtani 36, vergulni 37 bilan va h.k

Tabiiyki, mashina raqamlari emas, balki raqamlarni ifodalovchi signallarni farq qiladi. Xullas, kodlash murakkab tushunchani hammasi bo‘lib signalning ikki qiymati bilan (magnitlangan yoki magnitlanmagan, manbaga ulangan yoki ulanmagan ,yuqori yoki past kuchlanishi va h.k) ifodalashdir. Bu holatning birinchisini 0 raqami bilan, ikkinchisini esa 1 raqami bilan begilash qabul qilingan bo‘lib, axborotni ikkilikda kodlash nomini olgan. Bunda har bir murakkab tushuncha, ikkilik belgilari ketma-ketligida ifodalanadi. Shunday qilib, quyidagilar bajariladi:

- o‘nlik raqamlarini ikkilikda (binarli) kodlash (I K):

- alifbo belgilarini ikkilikda kodlash (axborot almashishning alifboli standart kodi-AASK).

Kodlar ikki: tekis va tekis bo‘lmagan turda bo‘lishi mumkin. Tekis tur ikkilik belgilariga ega. Tekis bo‘lmagan kodga Morze alifbosi misol bo‘la oladi, chunki unda har bir harf va raqamga uzun va qisqa signallarning ikkilik ketma-ketligi mos keladi. Masalan, E harfiga birgina nuqta mos kelsa, R harfi uchun to‘rtta mos keladi.

Hisoblash texnikasida odatda tekis kodlarda foydalaniladi. Shular qatoriga axborotlarni kiritish va chiqarish uchun EHMda foydaniladigan axborot almashinish kodi AAK-8; ikkilik axborot almashinish kodi - IAAK va boshqalarni kiritish mumkin. Ko‘pgina zamonaviy kompyuterlarda har bir belgiga 8 bitlik (1 bayt) ketma-ketlik mos qo‘yiladi. 8 ta 0 va birlardan tashkil topgan turli ketma-ketliklar jami 2*=256 ta bo‘lib, ular 256 xil turli belgilarni kodlash, masalan, lotin, rus alifbosining katta va kichik harflari, raqamlar, tinish belgilari, va boshqa belgilarini kodlash imkonini beradi. Bayt va belgilarning mosligi, yani har bir kodga mos belgi jadvalda ko‘rsatiladi. MDH davlatlarida keng tarqalgan harf raqamli kodlashning AAK-8 (8-xonalik) jadvalni keltiramiz.

O‘zbek alifbosi harflarining kodlari lotin alifbosi harflarinikidan farq qiladi. Masalan, o‘zbekcha katta “I” harfi 11 101 001, “L” harfi 11 101 100 kodlariga ega.

Nol va birlar ketma-ketligi bilan grafik axborotlarni ham kodlash mumkin. Gazetadagi rasmga diqqat bilan razm solsangiz, u mayda nuqtalardan tashkil

topganligini koʻrasiz. Turli pligrafiya uskunalarida bu nuqtalarining zichligi turlicha boʻladi. Maslan, “Toshkent oqshomi” gazetasidagi rasm “Xalq taʼlimi” jurnalidagi rasmga qaraganda aniqroqdir. Koʻpchilik gazetalardagi rasmlarda bir santimetrlik uzunlikda 24 ta nuqta boʻladi, yani 10x10 sm li rasm tahminan 60 ming nuqtadan iborat. Agar bular faqat oq va qora nuqtalardan iborat boʻlsa, u holda ularning har birini bir bit bilan kodlasa boʻladi. Agar nuqtalar har xil boʻlsa, u holda bitta nuqtaga bir bit yetarli boʻlmaydi. Ikki bit bilan nuqtaning 4 xil rangini: 00-oq, 01-och kul rang, 10- toʻq kul rang, 11-qora rangni kodlash mumkin. Uch bit sakkiz xil rangni, 4 bit 16 xil rangni kodlash imkonini beradi va h.k.

Shuningdek, ovozni ham kodlash mumkin. Musiqaga yozilgan notalar ovozni kodlashning turlaridan biridir. Nota belgilariga raqamlarni mos keltirib, ovozni bitlar orqali ifodalash mumkin.

Kompyuter axborotni faqat kodlashtirilgan koʻrinishda qayta ishlaydi. Unga kiritilgan maʼlumotlar xotira qurilmasiga joylashtiriladi. Shuni eslatish zarurki, axborotning mazmuni, uni uzatuvchilarning turi va tashuvchilaridan maʼlumotlarni oʻkiydigan kiritish qurilmasining koʻrinishiga bogʻlik boʻlmagan holda, ular xotiraga nol va birlar ketma-ketligida yozilishi maʼlum.

Yuqorida eslatilganidek, mashina xotirasi katakchalarga ajratilgan. Katakchada aniq uzunlikdagi har qanday ikkilik raqamlari ketma-ketligi saqlanishi mumkin. Bu ketma-ketlik mashina soʻzi deb ataladi. Mashina soʻzining uzunligi kompyuterning tuzilishi bilan aniqlanadi. Masalan, kompyuterning xotira katakchasi 24 ikkilik belgidan tashkil topgan mashina soʻzini saqlashi mumkin.

Sonlar, belgilar va koʻrsatmalar ham mashina soʻzi yordamida ifodalanib, qoʻllanilishi mumkin. Kompyuter uchun har bir son maxsus koʻrinishdagi soʻzdir. Masalan, haqiqiy sonni tasvirlash uchun uning ishorasi,butun qismi va kasr qismlari koʻrsatilishi kerak, Shuning uchun EHM dastlab mashina soʻzining yuqorida sanab oʻtilgan xarakteristikalarini aniqlaydi, soʻngra nol va birlar ketma-ketligini aniqlab beradi.

Kompyuterlarda sonlarni tasvirlashning ikki: qoʻzgʻalmas va qoʻzgʻaluvchi vergulli usulidan foydalaniladi.

Sonlarni qo‘zg‘almas vergulli tasvirlash. Sonni qo‘zg‘almas vergulli tasvirlash uchun kompyuter xotirasining katakchasi ishora va raqamlarga mo‘ljallangan xonalarga ajratiladi. Katakchaning xonalari odatda, chapdan o‘ngga tomon raqamlar bilan tartiblanadi. Katakchaning bitta xonasiga sonning bir xonasi mos keladi. Bunday sonning butun va kasr qismini ajratadigan vergulning o‘rni oldindan belgilanadi.

Qo‘zg‘almas vergulli tasvirlangan sonlar ustida amallar juda sodda bajariladi, chunki vergulning o‘rni o‘zgarmaydi. Shuning uchun xonalardagi raqamlarni mos ravishda qo‘shib qo‘yish yetarli. Bu usulning kamchiligi, ishlatiladigan sonlarning chegaralanganligidadir. Haqiqatan EHM 24 xonali xotira katakchasiga ega bo‘lib vergul 10-xonadan keyin qo‘yiladigan bo‘lsa, u holda xotira katakchasidagi absolyut qiymati bo‘yicha eng katta son.

Sonlarni qo‘zg‘aluvchi vergulli tasvirlash. Q asosli sanoq tizimidagi ixtiyoriy a soni (a==0) bu usulda quyidagicha tasvirlanadi: a = M*Q , bu yerda M - a sonining mantissasi deyiladi va u musbat tug‘ri kasrdan iborat, *r* - *a* sonining tartibi deyiladi va u butun son. Q - sanoq tizimining asosi.

Bu usulda tasvirlashning kamchilliklari sonni tasvirlashda belgilar sonining ko‘payib ketishi va shu bilan mos holda arifmetik amallarni bajarish jarayonining murakkablashib yuborishidan iborat.

Xotira katakchasida sonlarni tasvirlashning ikki usulini ko‘rib chiqdik. Shuni aytish kerakki, kompyuterda faqat sonlarni emas, balki turli belgilarni ham tasvirlab ishlatish mumkin. Bunday belgilar ham xotira katakchalarida ularda mos ikkilik kodlari orqali tasvirlanadi.

Kompyuterda axborotlarning ko‘rinishi

Yuqorida aytilganidek, kompyuter faqat sonli ko‘rinishdagi ma’lumotlarni qayta ishlay oladi. Har qanday boshqa ma’lumotlarni (tasvir, tovush, harf, raqam, har xil tinish belgilari va h.z.) kompyuterda qayta ishlash uchun ularni sonli ko‘rinishda tasvirlash kerak.

Kompyuterda sanoq tizimi uchun ikkilik sanoq tizimi olingan. Shuning uchun har qanday ma’lumotni kompyuterda qayta ishlanadigan bo‘lsa, uning ikkilik

sanoq tizimida koʻrinishi bilan ish olib ba “bit” deb yuritiladi) olinadi. 8 ta “bit” ni bir bayt deb nomlangan.

Bitta belgi bir baytda ifodalanadi. Hozirgi paytda kompyuterda ishlatiladigan belgilar soni 256 ta. Bularga 10 ta arab raqamlari, 26 ta lotin katta va kichik harflari, katta va kichik rus harflari, har xil arifmetik va tinish belgilari hamda maxsus belgilar kiradi.

1024 baytni bir kilbayt (1K bayt);

1024 K baytni bir megobayt (1M bayt);

1024 M baytni bir gigobayt (1G bayt) deb ataladi

Muhokama savollari

1. Sanoq tizimiga oid asosiy tushunchalar
2. Bir sanoq tizimidan boshqasiga oʻtish
3. Turli sanoq tizimidagi sonlar ustida amallar
4. Kompyuterda axborotlarning koʻrinishi

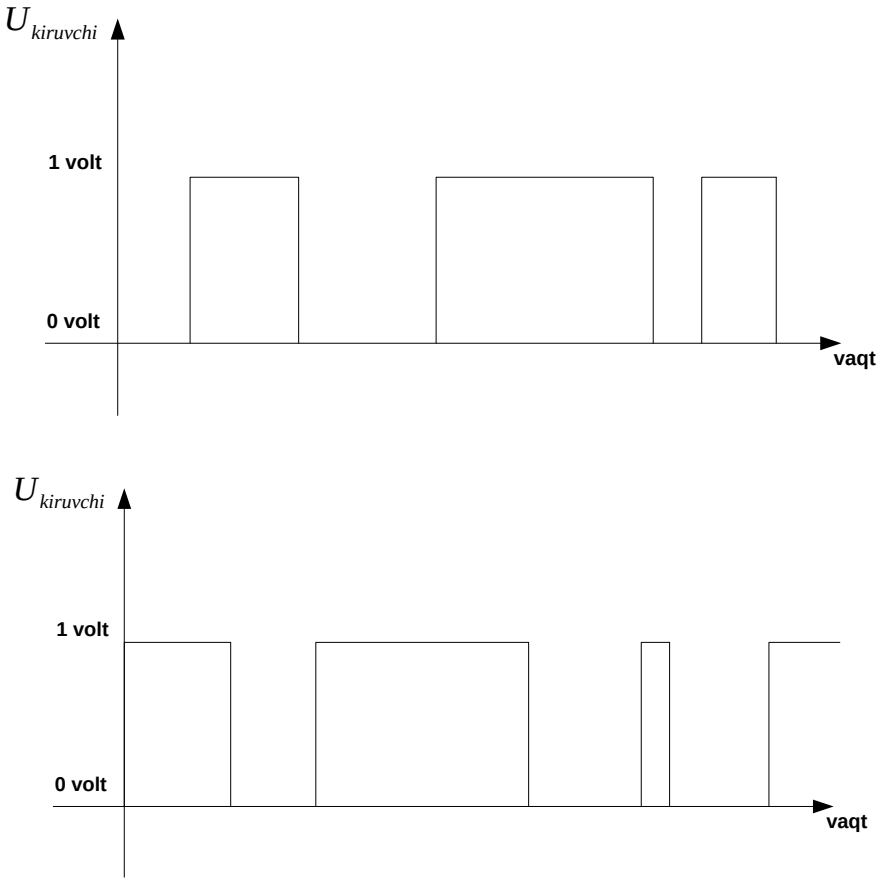
Nazorat savollari

1. Mashina soʻzi nima?
2. Sonlarni tasvirlashning qanday usullari mavjud?
3. Sanoq sistemasining asosi nima?
4. Ixtiyoriy asosli sanoq sistemasidagi sonni yoyib hisoblansa, qanday asosli sanoq sistemalaridagi son hosil boʻladi?
5. Ixtiyoriy sanoq sistemasida berilgan butun son oʻnlik sanoq sistemasiga qanday oʻtkaziladi?
6. Ixtiyoriy sanoq sistemasida berilgan kasr son oʻnlik sanoq sistemasiga qanday oʻtkaziladi?

1.5 Mulohaza va predikatlar algebrasining asosiy tushunchalari

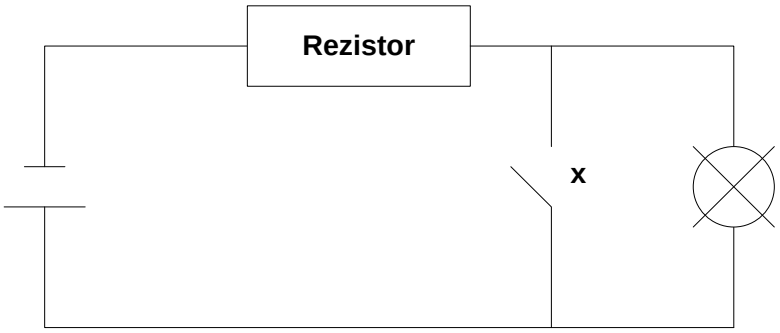
Ixtiyoriy kompyuter – murakkab texnik tuzilma. Lekin ushbu murakkab tuzilma ham, tabiatdagidek va texnikadagidek oddiy elementlardan tashkil topgan. Ixtiyoriy kompyuter, to‘g‘rirog‘i ixtiyoriy uning elektron Mantiqiy bloki o‘nlab va yuzlab ventillardan (Mantiqiy qurilmalar, Mantiqiy sxemalar bazasi), sxemalar va modullarda algebra ventillarini qonun qoidalarini birlashtirib (aksiomalar) tashkil qilinadi.

Mantiqiy ventil (keyinchalik - ventil) – qaysidir ma’noda EHM elektron tugunlarining atomi [3]. U signallarning yo‘lini ochish yoki yopish tamoyili asosida ishlaydi. Uning ishlash tamoyilini quyidagicha ta’riflash mumkin: agar, masalan “0” yoki “yolg‘on” ya’ni ushbu qurilmaga 0 voltli kuchlanish kirishi, va chiqishda 1 yoki «rost», ya’ni 1 voltli kuchlanish olinadi. Faraz qilaylik, agar invertorga kirishda kuchlanish 1 volt (rost) bo‘lsa, invertordan chiqishda 0 volt, ya’ni “yolg‘on” bo‘ladi (1.1-rasm *a*, *b* chizmaga qarang).



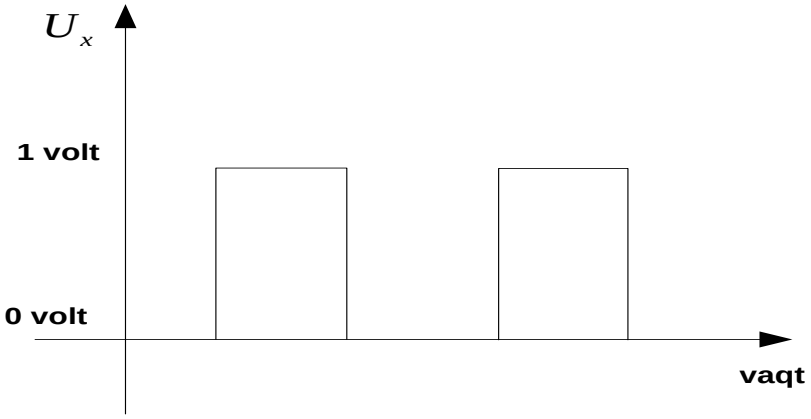
1.1-rasm. Invertorning ishlash tamoyili.

Inkor funksiyasini shartli ravishda lampochka zanjirida elektr sxemali bog‘lanish yordamida bir biriga tenglashtirish mumkin, (1.2-rasm), qachonki yopiq zanjir 1 (“rost”) yoki *x*=1, zanjirni uzishda esa 0 (“yolg‘on”) yoki *x*=0.

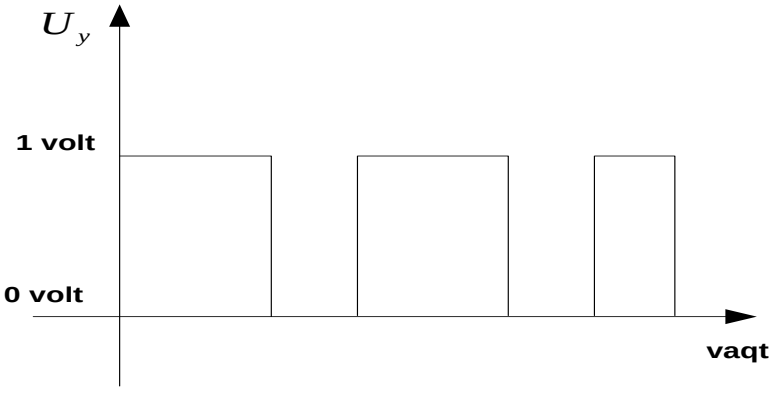


1.2-rasm. Invertor sxemasining elektr analogi

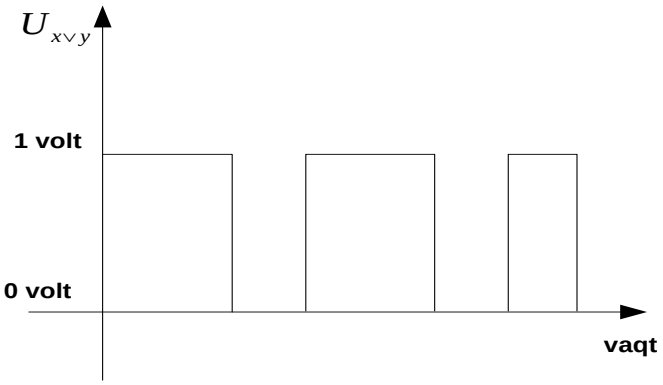
Diz’yunksiya x V y diz’yunktor deb ataluvchi Mantiqiy qurilmani (ventil) yaratadi. (1.3-rasm *a, b, c*):



1.3-rasm. *a*.

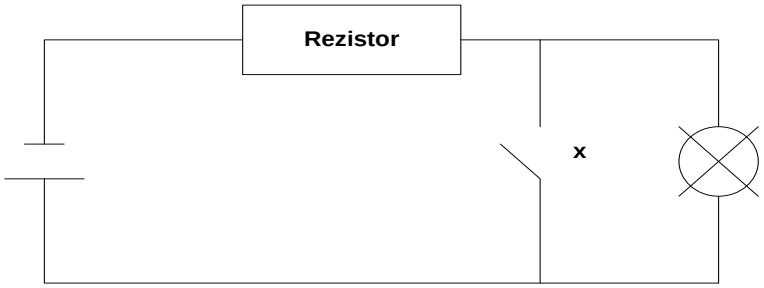


1.3-rasm. *b*.



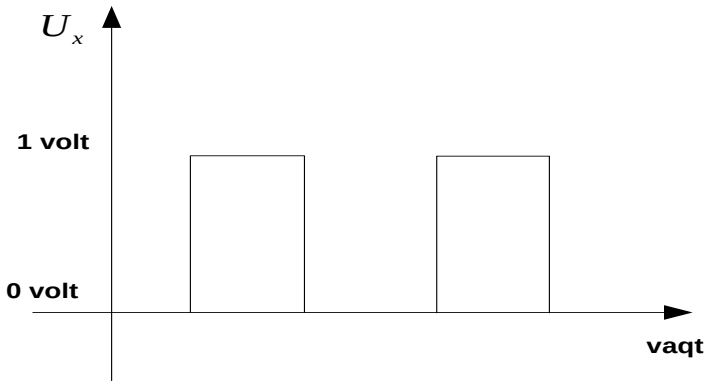
1.3-rasm. *c*. Diz’yunktorning ishlash tamoyili

Diz’yunktor shartli ravishda sxematik elektr zanjir ko‘rinishida bo‘ladi (1.4-rasm).

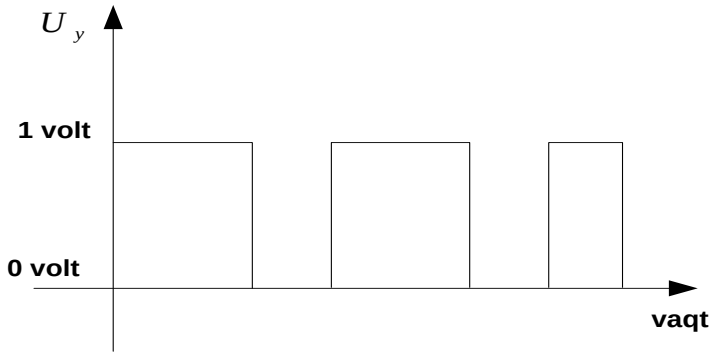


1.4-rasm. Diz’yunktor sxemasining elektr analogi

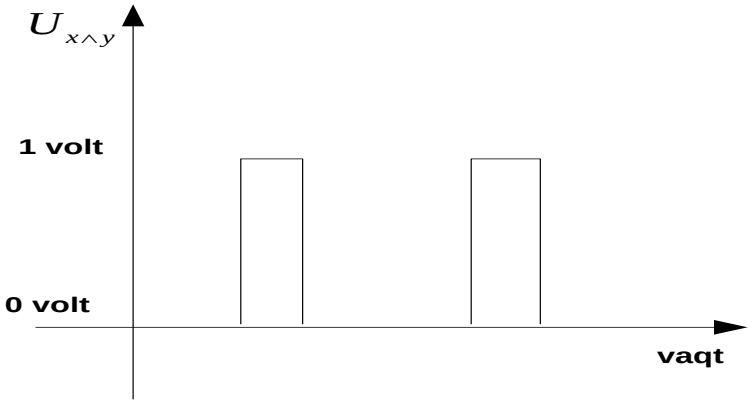
Kon’yunksiya x Λ y kon’yunktor deb ataluvchi Mantiqiy qurilmani (ventil) yaratadi. (1.5-rasm *a*, *b*, *c*):



1.5-rasm. *a*.

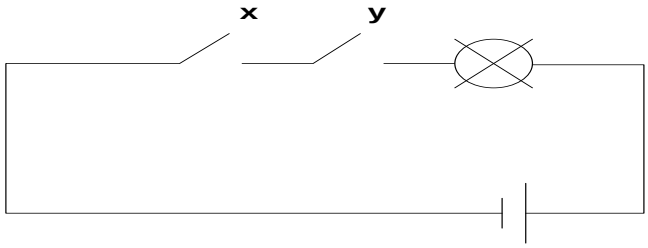


1.5-rasm. *b*.



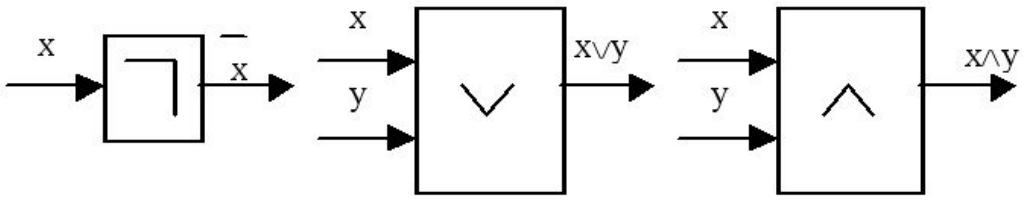
1.5-rasm *c*. Kon’yunktorning ishlash tamoyili

Kon’yunktorni sxematik elektr zanjirini shartli ravishda tasvirdash mumkin (1.6-rasm).



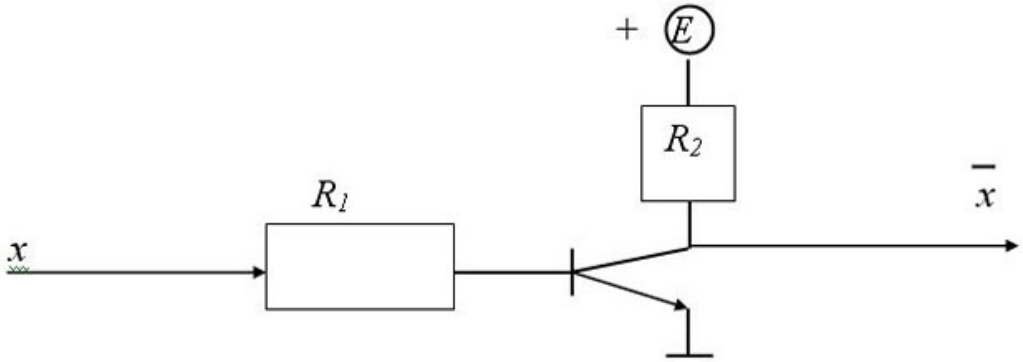
1.6-rasm. Kon’yunktor sxemasining elektr analogi

Sxematik invertor, har xil qurilmalarning Mantiqiy sxemalarida diz’yunktor va kon’yuktorlarni keyingi shaklda ifodalash mumkin (1.7-rasm *a*, *b*, *c*). Bundan tashqari ushbu belgilashlarning boshqacha ko‘rinishlari ham mavjud.

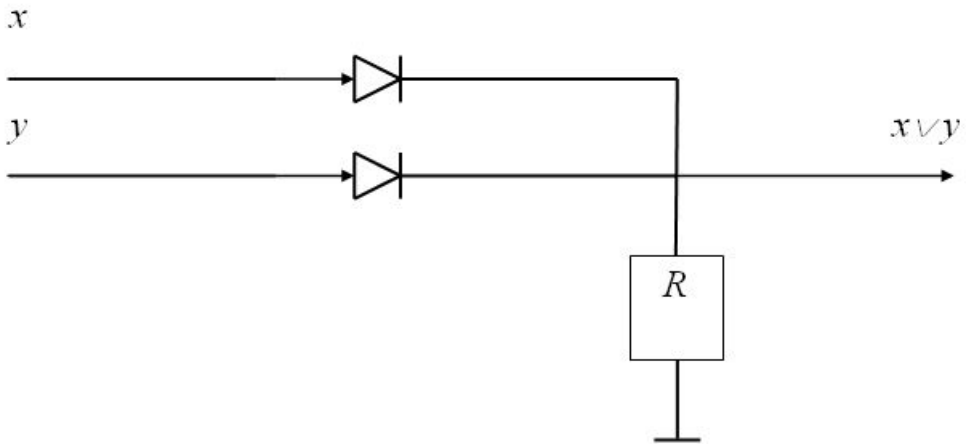


1.7-rasm *a*, *b*, *c*. Ventilning shartli belgilanishlari

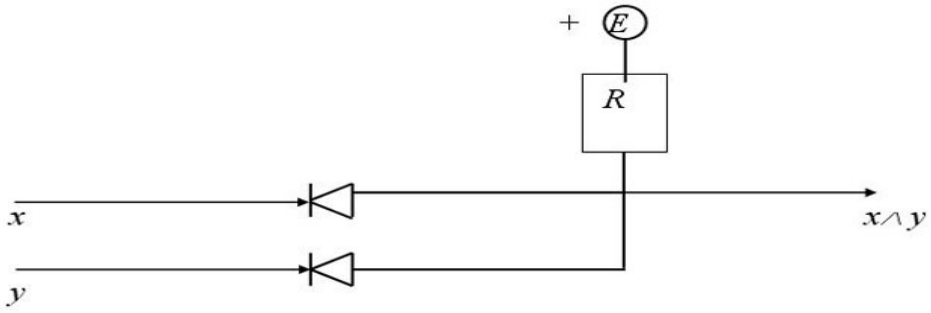
Masalan. Tranzistor sxemalari, Mantiqiy sxemaga to‘g‘ri keladigan ¬ (invertor), V (diz’yunktor), Λ (kon’yunktor) keyingi ko‘rinishga ega (1.7-rasm *a*, *b*, *c*):



1.8-rasm *a*. Invertor



1.8-rasm *b*. Diz’yunktor



1.8-rasm *c*. Kon’yunktor

Yuqorida ko‘rsatilgan Mantiqiy elementlar bazasidan, EHM ning murakkab mantiqiy sxemalari yig‘iladi, masalan, summatorlar, shifratorlar, deshifratorlar va hakoza. Katta (BIS) va juda katta (SBIS) integral sxemalar o‘zining tarkibi (kremniy krisstalida bir qancha kvadrat santimetirga ega sohada) o‘n minglab ventillardan iborat.

Muhokama savollari

1. Mantiqiy amallar tushunchasi.
2. Invertorning ishlash tamoyili.
3. Diz’yunktorning ishlash tamoyili.
4. Kon’yunktorning ishlash tamoyili.

Nazorat savollari

1. Elektron mantiqiy blok
2. Mantiqiy ventil – EHM elektron tugunlarining atomi
3. Inkor funksiyasi
4. Kon’yunktorning sxematik elektr zanjiri

1.6 Mantiq algebrasining asosiy tushunchalari va funksiyalari

Informatika – belgili (alfavitli) tizimlarni o‘rganadi. Mulohaza tushunchasi matematikadagi boshlang‘ich tushunchalardan biri hisoblanadi.

F amal *n* ta o‘rinli deyildi, agar u *n* ta operandni bir-biri bilan bog‘lasa. A algebraning amallar yig‘indisi uning signaturasi, algebra elementlarining yig‘indisi esa – algebra tashuvchisi deyiladi.

Mulohaza deb, - rost yoki yolg‘onligi haqida gapirish mumkin bo‘lgan har qanday darak gapga aytiladi. Bu ikki mumkin bo‘lgan mulohazalar qiymati “rost” va “yolg‘on”, “true” va “false” yoki “1” va “0” deb belgilanadi.

“1” yoki “0” qiymat qabul qiluvchi o‘zgaruvchi – mantiqiy o‘zgaruvchi yoki Bul o‘zgaruvchi deyiladi.

Masalan:

1. Moskva – AQSh poytaxti.
2. Moskva shahrining yashovchisi.
3. 5-7+9.
4. 5-9+28=4.
5. Qishning 5 chi haftasi juda sovuq bo‘ldi.
6. Yo‘lbarslar Antarktikada yashaydilar.

Mulohazalar bir qiymatli rost yoki bir qiymatli yog‘lon bo‘lishi kerak, shuning uchun 1), 4), 6) mulohaza hisoblanadi.

Mulohaza bo‘lmagan gaplarga misol qaraymiz.

Shunday gaplar ham borki, bu gaplarning tarkibida qandaydir Ob’yektlarning umumiy nomlarida noma’lumlar ishtirok etadi. Masalan, “*x* son *y* songa bo‘linadi”, “*x* rasional son”, “*sinx*+*cosx*=0”, “$sinx \le \frac{1}{2}$”, “*x* Norin tumanining markazi”, “*x*+2=81”, “*x* qiz *y* qizning dugonasi”, “*z* bugun 60 kg paxta terdi”. Aniqki, bu gaplarni mulohaza deya olmaymiz, chunki ularning rost yoki yolg‘onligini aniqlashda, gap tarkibidagi noma’lumlar xalaqit beradi. Ammo, bu gaplar tarkibidagi noma’lumlarni biror to‘plamning elementlari bilan almashtirsak, ular rost yoki yolg‘on mulohazalarga aylanishi mumkin. Masalan, birinchi misolda *x* va *y* o‘zgaruvchilarga ma’lum bir butun sonlarni, ikkinchi misolda *x* o‘rniga rasional sonlarni, uchinchi misolda *x* o‘rniga haqiqiy sonlarni, to‘rtinchi misolda ham haqiqiy sonlarni, beshinchi misolda *x* o‘rniga tuman markazlari nomlarini, oltinchi misolda *x* o‘rniga natural sonlarni, yettinchi misolda *x* va *y* lar o‘rniga qizlar nomlarini, sakkizinchi misolda *z* o‘rniga odamlarni nomlarini qo‘yilganda rost yoki yolg‘on mulohazalarni hosil qilamiz.

$x, y \in X$ mantiqiy o‘zgaruvchilar to‘plami ular ustida berilgan amallar bilan: $\overline{x}$ - inkor yoki inversiya, $x \vee y$ - mantiqiy qo‘shish yoki diz’yunksiya, x∧y –

mantiqiy ko‘paytirish yoki kon’yunksiya predikatlar algebrasi deyiladi, agar ular quyidagi aksiomalarni qanoatlantirsa:

1. Qo‘sh inkor tengkuchliligi: $\overline{\overline{x}} = x$,
2. Operandlarning o‘rin almashtirish aksiomasi (diz’yunksiya va kon’yunksiyaga nisbatan): $x \wedge y = y \wedge x, x \vee y = y \vee x$,
3. Diz’yunksiya va kon’yunksiya amallarining o‘rin almashtirish aksiomasi (operandlarga nisbatan):

$(x \wedge y) \wedge z = x \wedge (y \wedge z), (x \vee y) \vee z = x \vee (y \vee z)$,

4. Bir xil operandlar aksiomasi: $x \wedge x = x, x \vee x = x$,
5. $x \wedge (x \vee y) = x, x \vee (x \wedge y) = x$,
6. Amallar ketma ketligi aksiomasi (diz’yunksiya kon’yunksiyaga nisbatan va teskarisi): $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z), x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$,
7. De Morgan aksiomasi (binar amalni operandlarga o‘tkazish): $\overline{x \wedge y} = \overline{x} \vee \overline{y}, \overline{x \vee y} = \overline{x} \wedge \overline{y}$,
8. Beqarorlik aksiomasi (o‘zaro teskari yig‘indi va ko‘paytuvchilarga nisbatan): $x \wedge (y \vee \overline{y}) = x, x \vee (y \wedge \overline{y}) = x$,
9. “1, rost, true” va “0, yolg‘on, false” ning mavjudlik aksiomasi: $\overline{0} = 1, \overline{1} = 0, \overline{x} \vee x = 1, \overline{x} \wedge x = 0$.

Bu aksiomalardan keyin quyidagi munosabatlar o‘rinli:

$$x \wedge 1 = x,$$
$$x \vee 0 = x,$$
$$x \vee 1 = 1,$$
$$x \wedge 0 = 0,$$
$$\overline{x} \vee x = 1,$$
$$\overline{x} \wedge x = 0.$$

Predikatlar algebrasining uchta bazaviy amali, ularning qiymatlar jadvali bilan aniqlanadi.

x	y	$\overline{x}$	$x \wedge y$	$x \vee y$
0	0	1	0	0
0	1	1	0	1

1	0	0	0	1
1	1	0	1	1

Ba’zi bir mantiqiy funksiyalar uchun shu ko‘rinishdagi jadval mantiqiy jadval deyiladi.

Masalan: $z = \overline{x \wedge (\overline{x} \wedge y)}$ funksiya uchun rostlik jadvalini tuzamiz. Bu jadval quyidagicha bo‘ladi:

x	y	$\overline{x}$	$\overline{x} \wedge y$	$z = x \wedge (\overline{x} \wedge y)$	$z = \overline{x \wedge (\overline{x} \wedge y)}$
0	0	1	0	0	1
0	1	1	1	0	1
1	0	0	0	0	1
1	1	0	0	0	1

Bundan kelib chiqadiki funksiyamiz teng kuchli rost ekan. Buni quyidagi aksioma orqali tekshiramiz:

$z = \overline{x \wedge (\overline{x} \wedge y)} = \overline{x} \vee \overline{\overline{x} \wedge y} = \overline{x} \vee x \vee \overline{y} = 1 \vee \overline{y} = 1$.

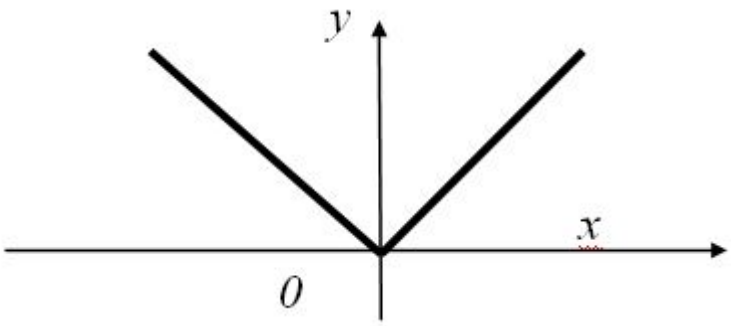
Misol: $w = x \wedge y \wedge z$ uch o‘rinli mantiqiy funksiya uchun rostlik jadvalini tuzamiz. Bu quyidagicha ko‘rinishga ega bo‘ladi:

x	y	z	$y \wedge z$	w
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
1	0	0	0	0
0	1	1	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Misol: $p(x,y)$ = “x ning absolyut qiymati u ning absolyut qiymatiga teng” ikki o‘rinli predikatning rostlik grafik to‘plamini yasaymiz, agar argument $x, y \in [0;1]$ o‘zgarish sohasi quyidagicha bo‘lganda. Ushbu tenglikka ega bo‘lamiz

$$E(p) = \{(x,y) : |x| = |y|\} = \{(x,y) : (x = y) \vee (x = -y)\} =$$
$$= \{(x,y) : x = y\} \cup \{(x,y) : x = -y\} = E(p1) \cup E(p2).$$

$p1(x,y)$ va $p2(x,y)$ predikatning ma’nosi aniq. To‘plam $y=|x|$ funksiya grafigi orqali ko‘rinadi, 1.9-rasm:



1.9-rasm. $y=|x|$ funksiyaning grafigi

Uchta bazaviy amaldan tashqari ular yordamida, keyingi asosiy amallarni kiritish mumkin (ularni bazaviy bo‘lmagan amal deb aytsa bo‘ladi):

1. implikasiya: $(x \to y) \equiv (\overline{x} \vee y)$,
2. ekvivalensiya: $(x \leftrightarrow y) \equiv (x \wedge y \vee \overline{x} \wedge \overline{y})$.

Implikasiya va ekvivalensiya amallari ko‘p qo‘llaniladigan amal bo‘lishiga qaramasdan, bazaviy bo‘lmagan amal hisoblanadi, nima uchun deganda ular yuqorida keltirilgan uchta bazaviy amal orqali aniqlanadi. Mantiqiy formulalarda amallar ketma-ketligi o‘rnatilgan, masalan: qavslar, inkor, kon’yunksiya, diz’yunksiya.

Hamma vaqt rost bo‘ladigan formulalar taftalogiya deyiladi.

Mantiqiy funksiyalar ekvivalent deyiladi, agar ularning rostlik jadvali, ya’ni qiymatlar sohasi bilan aniqlanish sohalari, va yana shu to‘plamda mumkin bo‘lgan barcha qiymatlar funksiya qiymatlari ustma – ust tushsa. Agar bu o‘xshashlik sohaning qaysi qismida sodir bo‘lsa, u holda formula shu sohada ekvivalent deyiladi.

Mantiqiy ifodalarni soddalashtirish masalasi, uni sodda ekvivalent (o‘zgaruvchilar soni, amal va operandlar) ko‘rinishga olib kelish demakdir.

Eng sodda ko‘rinish hamma vaqt “1” yoki “0” bo‘ladigan funksiya orqali olinadi.

Misol Soddalashtiramiz:

$$f(x,y) = (x \wedge y) \vee (\overline{x} \wedge y) \vee (x \wedge \overline{y}) = ((x \wedge y) \vee (\overline{x} \wedge y)) \vee (x \wedge \overline{y}) =$$
$$= (\text{аксиома дистрибутивности}) =$$
$$= ((x \vee \overline{x}) \wedge y) \vee (x \wedge \overline{y}) =$$
$$= (\text{аксиома нейтральности}) =$$
$$= y \vee (x \wedge \overline{y}) =$$
$$= (\text{аксиома дистрибутивности}) =$$
$$= (y \vee x) \wedge (y \vee \overline{y}) = (y \vee x) \wedge 1 = y \vee x = x \vee y$$

Ikkita mantiqiy ifodaning tengligini ko‘rsatish masalasi, ularning barcha o‘zgaruvchilarining qiymatini ma’lum bir sohada ekvivalent funksiyalarini o‘rnatishdan iborat.

Misol. $\overline{x \vee y} \vee \overline{x} \wedge y \wedge \overline{x} = \overline{x}$ mantiqiy ifodaning tengligini isbotlaymiz. Predikatlar algebrasidagi aksiomalardan foydalanib, quyidagi tenglikni hosil qilamiz: $\overline{x \vee y} \vee \overline{x} \wedge y \wedge \overline{x} = \overline{x} \wedge \overline{y} \vee \overline{x} \wedge y \wedge \overline{x} = \overline{x} \wedge \overline{y} \vee \overline{x} = \overline{x}$.

Tenglikning chap qismi o‘ng qismiga keltirilgan, ya’ni Ushbu tenglik isbotlandi.

Shu turdagi masalalar predikatlar algebrasi aksiomasining keyingi usullari yordamida yechiladi:

- tenglikning o‘ng qismi chap qismiga keltiriladi,
- tenglikning chap qismi o‘ng qismiga keltiriladi,
- tenglikning ikkala qismi ham uchunchi ifodaga keltiriladi.

Mantiqiy funksiyalar – infologik (informasion-mantiqiy) masalalarni aniq yechishga yordam beradi.

Informasion – mantiqiy (infologik) masala - shunday masalaki bunda informasion va mantiqiy bog‘liqlik o‘rnatilib ularda yetarlicha mantiqiy sabab-izohlar keltirib chiqarish zarur.

Shu turdagi masalalar har xil sohalarda uchraydi, yaxshi formallashmagan va strukturalashmagan bo‘ladi. Bu ish qanchalik to‘g‘ri qilinsa – qaralayotgan masala ham aniq va to‘g‘ri yechiladi.

Mulohazalar va predikatlar algebrasi qoidasi, inson fikrini yakunlashi, fikrlashi, isbotlashiga to‘g‘ri keladi.

Misol. Mulohazalar algebrasining kon’yunksiya, diz’yunksiya, inkor amallari – inson fikrini “va”, “yoki”, “yo‘q” so‘zlarining analogi hisoblanadi. EHM ga fikrlaydigan harakterdagi masalani yuklash uchun, bu qoidalarni qat’iy formallashtirish zarur. Bu mantiqiy algebrani hosil qilishga yordam beradi.

Fandagi isbotlash metodlarini o‘rganuvchi ba’zi bir mantiqiy aksiomalarni olib kelamiz.

1. uchinchisini olib tashlash aksiomasi: yoki mulohaza o‘rinli, yoki uning teskarisi,

2. ziddiyatlik aksiomasi: mulohaza va uning teskarisi bir vaqtning o‘zida bajarilmaydi,

3. ikkilik inkor aksiomasi: qaysidir tasdiqning ikki karrali inkori shu inkorga teng kuchli,

4. tenglik aksiomasi: har qanday mulohaza o‘z - o‘ziga tengdir.

Agar *x* va *y* mulohazalar bir biri bilan $x \Longrightarrow y$ munosabat orqali bog‘liq bo‘lsa, u holda *y*-mulohaza *x*-mulohazadan kelib chiqadi (yoki y *x* ning natijasi); agar X to‘plamning *x*-mulohazasi, Y to‘plamning *y*-mulohazasini o‘z ichiga olsa, u holda *x*-mulohaza – shart, *y*-mulohaza – yakunlash, $x \Longrightarrow y$ munosabat esa xulosa deyiladi.

Formal matematik tasdiqlarning (teorema) isboti – shartni yakunlashga olib keluvchi korrekt xulosalar ketma-ketligiga aytiladi.

Mantiqiy algebra teoremalarni isbotlashga yordam beradi (isbotlashning umumiy yondashuvi va metodi).

Teoremaning isbotiga umumiy yondashuv, mantiqiy algebralarni formallashtirish yordamida qarshi, teskari va qarama-qarshi teoremalar amalga oshiriladi.

Muhokama savollari

1. Mulohaza tushunchasi
2. Mantiqiy ko‘paytirish yoki kon’yunksiya predikatlar algebrasi
3. Predikatlar algebrasining uchta bazaviy amali
4. Implikasiya va ekvivalensiya amallari
5. Mantiqiy ifodalarni soddalashtirish
6. Mulohazalar va predikatlar algebrasi qoidasi
7. Mantiqiy aksiomalar

Nazorat savollari:

1. Mantiqiy amallar.
2. Kon’yunksiya predikatlar algebrasi.
3. Mantiqiy aksiomalar.
4. Predikatlar algebrasining uchta bazaviy amali.
5. Rostlik jadvalini tuzish.
6. Taftalogiya tushunchasi.

1.7 Algoritmlash asoslari

Algoritm — ijrochi uchun ma’lum bir masalani yechishga qaratilgan ko‘rsatmalarning aniq ketma-ketligi.

“Algoritm” so‘zi O‘rta Osiyolik buyuk matematik Al-Xorazmiyning nomi bilan bog‘liq. Al-Xorazmiyning nomini lotincha ifodasi — Algorithm. Algoritm — informatika va matematikaning asosiy tushunchalaridan hisoblanadi.

“Algoritm ijrochisi”.

Algoritm ijrochisi — algoritmda ko‘rsatilgan buyruqlarni bajara oladigan abstrakt yoki real (texnik, biologik yoki biotexnik) tizim.

Algoritmlarga xos xususiyatlar:

- oddiy harakatlar;
- buyruqlar tizimi.

Buyruqlar tizimi. Har bir ijrochi faqatgina ushbu ijrochi tushunadigan buyruqlarni (ya’ni, ijrochi bajaradigan buyruqlar ro‘yxatiga mansublarni) bajara oladi.

Ijrochi buyruqlarni bajarish jarayonida oddiy harakatlarni bajaradi.

Odatda ijrochiga algoritmning maqsadi ma’lum bo‘lmaydi. Shuning uchun ijrochi “nimaga” va “nima uchun” degan savollarni bermaydi.

Informatikada algoritmlarning universal ijrochisi bo‘lib kompyuterlar hisoblanadi.

Algoritmning xossalari

Algoritmlarning asosiy xossalari quyidagilardan iborat:

Tushunarlilik. Algoritm ijrochisi buyruqlar ketma-ketligini qanday bajarishni aniq bilishi kerak.

Diskpyetlik. Algopitm ijrochisi masalani yechish jarayonini alohida va sodda qadamlar ketma-ketligini bajarish deb tushunishi kerak.

Aniqlik. Algoritmning har biri qoidasi, undagi amallar va buyruqlar bir ma’noli bo‘lishi kerak. Shu xossaga asosan algoritm ijrochisi buyruqlar ketma-ketligini mexanik bajarish imkoniyatiga ega bo‘ladi.

Natijaviylik. Bu xossaning mazmuni shundan iboratki, har qanday algoritmning ijrosi oxir-oqibat ma’lum bir yechimga kelishi kerak.

Ommaviylik. Masalani yechish algoritmi umumiy hollar uchun yaratiladi, ya’ni faqatgina boshlang‘ich qiymatlari bilan farqlanuvchi bir turdagi masalalar sinfi uchun tuziladi. Bunda boshlang‘ich qiymatlar algoritmning qiymatlar qabul qilishi mumkin bo‘lgan sohadan olinadi.

Algoritmlarni tasvirlash usullari

Amaliyotda algoritmlarni tasvirlashning keng tarqalgan usullari quyidagilar:

- so‘zlar yordamida (og‘zaki nutqda ishlatiladigan so‘zlar yordamida, tabiiy tilda);
- grafik usulda (grafik simvollar yordamida);
- dastur ko‘rinishida (dasturlash tillariga oid xizmatchi so‘zlar, operator va funksiyalar yordamida);

- formulalar yordamida (matematik formulalardan foydalangan holda, analitik koʻrinishda);
- makrotildan foydalangan holda (dasturlovchi va EHMga tushunarli boʻlgan makrokomandalar yordamida);
- jadval koʻrinishida (mantiqiy algebra elementlaridan foydalangan holda).

Algoritmlarni soʻzlar yordamida tasvirlash.

Algoritmlarni *soʻzlar yordamida* tasvirlashda bajariladigan buyruqlar va koʻrsatmalar ketma-ket ogʻzaki nutqda ishlatiladigan soʻzlar orqali yoziladi.

Masalan, Ikki sonning eng katta umumiy boʻluvchisini (EKUB) topish algoritmi quyidagicha yozilishi mumkin:

1. Ikkita sonni kiriting;
2. Agarda bu sonlar teng boʻlsa, u holda ulardan birini javob sifatida oling va ishni toʻxtating, aks holda esa davom ettiring;
3. Ikkita son ichida kattasini aniqlang;
4. Katta va kichik sonlarning ayirmasini katta son bilan almashtiring;
5. Algoritmni 2-qadamdan boshlab qaytaring.

Keltirilgan algoritmni har qanday natural sonlarning EKUBini topish uchun ishlatish mumkin.

Algoritmlarni soʻzlar yordamida tasvirlashning bir qancha kamchiligi mavjud boʻlib, aksariyat hollarda algoritmlarni tasvirlashda bu usuldan foydalanilmaydi.

Algoritmlarni grafik usulda tasvirlash.

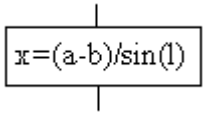
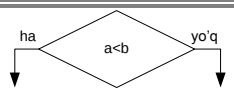
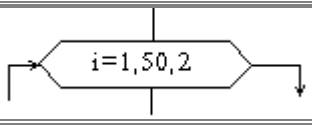
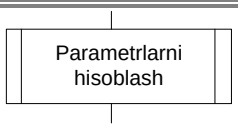
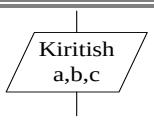
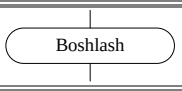
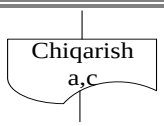
Algoritmlarni grafik usulda tasvirlashda har bir amal bir yoki bir nechta harakatni ifodalovchi oʻzaro bogʻliq funksional bloklar ketma-ketligi orqali tasvirlanadi.

Algoritmning bunday tasvirlash usuli algoritm sxemasi yoki blok-sxema deb ataladi.

Blok-sxemada har bir harakat turi (boshlangʻich qiymatlarni kiritish, ifodalar qiymatlarini hisoblash, shartlarni tekshirish, amallarni takrorlashni boshqarish, qayta ishlashni tugatish va h.k.) maʼlum bir geometrik figura orqali ifodalanadi.

Blokli belgilar (geometrik figuralar) chiziqlar orqali bog‘lanadi (bunda qaysi amal oldin, qaysinisi keyin bajarilishi ko‘rsatiladi).

1.1-jadval blok-sxemada ishlatiladigan bloklarni aks etadi.

Amallarni belgilanishi	Izoh
	Oddiy harakat
	Shart tekshirish
	Sikl (takrorlanish) boshi
	Yordamchi algoritmga murojaat
	Ma’lumotlarni kiritish va chiqarishning umumiy ko‘rinishi
	Algoritmning boshi va oxiri
	Natijani bosmaga chiqarish

1.1-jadval. Blok-shemada keltirilgan bloklar vazifalari

- “Oddiy harakat” belgisi orqali formulalar, hisob-kitob, o‘zlashtirish amallari ifodalaniladi. Bir nechta amallarni alohida yoki bitta belgi orqali ifodalash mumkin.
- “Shart tekshirish” bloki orqali amallar bajarilish yo‘nalishi shart bajarilishi asosida ko‘rsatiladi. Bunday blokning har birida savol, shart yoki munosabat ko‘rsatiladi.
- “Sikl” bloki amallarni takrorlash uchun ishlatiladi. Blok ichida siklning boshi va oxirini ko‘rsatuvchi parametr (i), parametrning o‘zgarish qadami ko‘rsatiladi.

- “Yordamchi algoritmga murojaat” bloki alohida va mustaqil ishlovchi qism dastur va yordamchi algoritmlarga murojaatni bildiradi.

Muhokama savollari

1. Algoritm tushunchasi.
2. Algoritm ijrochisi.
3. Algoritmning xossalari.
4. Algoritmni tasvirlash usullari va turlari.
5. Algoritm tuzilishi

Nazorat savollari

1. Algoritm deganda nimani tushunasiz?
2. Algoritm va algoritm ijrochisining vazifalari nimalardan iborat?
3. Algoritm qanday xossalarga ega?
4. Algoritm tasvirlash usullaridan qaysilari keng tarqalgan?
5. Quyidagi algoritm bajarilishi natijasida qanday natija olinadi:

1) a=3, b=4
2) a=a+b
3) b=a-b
4) a=a-b
5) nat a, b

6. Quyidagi amallar bajarilganidan so‘ng *x* va *u* o‘zgaruvchilarning qiymatlari qanday bo‘ladi:

a) t:=x
x:=y
y:=t

b) toki x<20
x:=x+1
y:=y+3
tam

1.8 Algoritm va bajaruvchi-inson va chekli avtomat. Algoritm, uning xossalari, berilish usullari

Bajaruvchi – bu biror bir biologik, texnik yoki aralash tuzilma bo‘lib (buyruq yoki dastur), biror operasion tizimda (mumkin bo‘lgan “uskunalar” va “buyruqlar” majmuasi) ma’lum algoritmlar sinfini bajaradi.

Algorimlarning eng ko‘p qo‘llaniladigan turi – bu inson va avtomat (kompyuter).

Inson algoritmlarni bajaruvchi sifatida bajariluvchi (muskullar, harakat, ko‘rish, hid bilish va boshq.) va boshqaruvchi (nerv, neyron) bajariluvchi qism tuzilmalar to‘plamini olish mumkin.

Nerv tizimi nerv tolalari mavjud bo‘lgan teri, ko‘z, quloq va boshqa organlar nerv markaziga uzatiladi va unda qayta ishlanib, mos javob qaytariladi. *Nerv tizimi* – nerv tolalari va neyronlar o‘zaro ta’siri majmuasi hisoblanadi. Insonda – bular juda ko‘p miqdorda bo‘ladi.

Misol. Fiziologlarning baholashicha insonning old miyasida tahminan 50 mlrd *neyron* mavjud. *Neyronlar*, sekin ishlagani bilan (sekundiga yuzlab vazifani bajaradi), o‘zaro aloqasining samarasi va murakkab neyro tuzilmali aloqani (klasterlar) amalga oshirishi natijasida murakkab fikrlashni amalga oshiradi va qaror qabul qiladi.

Misol. Inson uchun sodda masalalardan biri, bu “ob-havoga mos kiyinish” masalasi. Bunda ko‘z va quloq orqali olingan axborotni qayta ishlash va holatni “neyronli” baholash natijasida qaror qabul qilinadi. Kompyuter bu masalani yechishi ancha murakkab. Boshqa tomondan qaraganda, insonning hisoblash resursi kompyuter imkoniyatlaridan anchagina chegaralangan bo‘lib, kompyuterning aniq va yaxshi tuzilmali masalalarni yechishda imkoniyati (ya’ni tezkor va aniq) bir necha marotaba yuqori hisoblanadi.

Neyronlar – bu inson miyasining ma’lum qismlarida paydo bo‘ladigan (nerv impulslarini shifrdan chiqarish natijasida hosil bo‘lgan) ma’lumotlarni uzatish uchun hizmat qiladi.

Inson xotirasiga koʻrish, eshitish, hid bilish va boshqa organlaridan maʼlumot kelib tushadi. Soʻng bu maʼlumotlar tezkor xotiraga (ong) oʻtkaziladi. Soʻngra bu maʼlumotlar doimiy xotiraga koʻchiriladi (“formalar”, “Obʼyektlar va obrazlar”, “Obʼyektni topish va identifikasiyalash qoidalari va proseduralari”, “maʼlumotni tanlash qoidalari”, “hayot tajribasi” kabi boʻlimlarga joylashtiriladi).

Misol. Inson koʻrayotgan maʼlum bir kompyuter insonning doimiy xotirasidagi “kompyuter” boʻlimidagi termin bilan moslanadi. Bu boʻlimdan shu Obʼyektga mos kodlar oʻzaro bogʻlanib kompyuterning obrazini yaratadi.

Tirik organizmda maʼlumotni uzatish, saqlash va uni qayta ishlash biokimyoviy jarayon boʻlib, axborot molekulyar tizim signali va ularning turli konsentrasiyali moddalar, katalizlarning kimyoviy jarayonlarining hisobiga hosil boʻladigan koʻrinishi hisoblanadi. Harakatlarning potensiallarini markaziy nerv tizimi orqali boshqariladigan nerv tolalari oʻtkazadi (elektr signallari). Shu bilan bir qatorda DNK dan RNK ga, RNK dan oqsilga oʻtkaziladigan axborotdan ham foydalaniladi.

Bajaruvchining ikkinchi asosiy turi – bu *avtomatlar, (*yaʼni maʼlum bir vaqt oralig‘ida inson ishtirokisiz ishlaydigan) avtomatik qurilmalar boʻlib, chekli qadamlarda maʼlumotlarni kiritish, chiqarish kabi ishlarni bajaradi.

Ixtiyoriy avtomat biror boʻsh boʻlmagan, bajarilish ketma – ketligini aniqlovchi boshqaruv avtomatidan va avtomatning ishini boshqaruvchi tizimli avtomatdan iborat algoritmlar ketma – ketligini amalga oshiradi.

Misol. Misol uchun gazli suv bilan savdo qiluvchi avtomatni olish mumkin. Uning ishlash jarayonini 1.10-rasmda keltirilgan grafda koʻrish mumkin. Quyidagi shartli belgilarni kiritamiz:

$X = 1, 3, \Gamma, \oslash$ – kirish toʻplami,

Y = {V, S, O} – chiqish toʻplami,

S = {s_0 , s_1 , s_2 , s_3} – holat toʻplami,

1 – kirish signali “1 soʻm tashlang”,

3 – kirish signali “3 soʻm tashlang”,

G – kirish signali “qiyshiq tanga tashlang”,

$\varnothing$ – kirish signali “tanga tashlanmadi”,

V – chiqish signali “siropsiz gazli suv berish”,

S – chiqish signali “siropli gazli suv berish”,

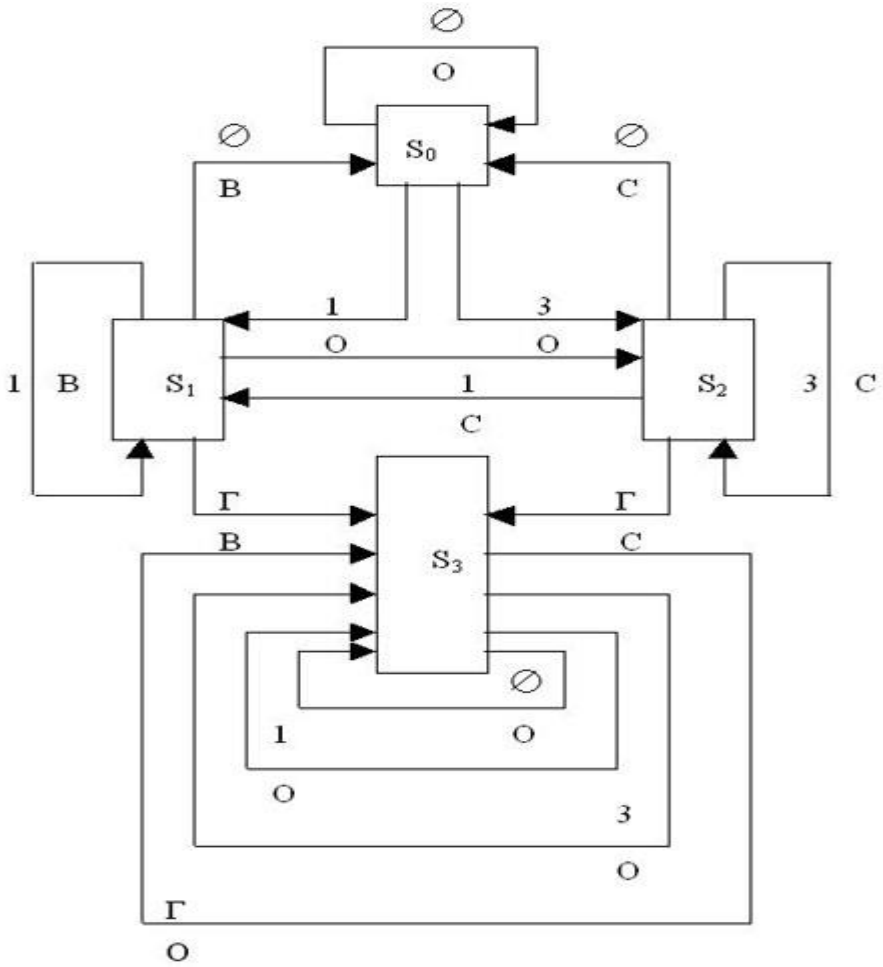
O – chiqish signali “gazli suv berilmasin”,

s_0 – birinchi holat – “boshlang‘ich holat”,

s_1 – ikkinchi holat – “1 so‘mni qayta ishlash”,

s_2 – uchinchi holat – “3 so‘mni qayta ishlash”,

s_3 – to‘rtinchi holat – “buzilish holati”.



1.10-rasm. Gazli suv savdosini amalga oshiruvchi avtomat grafigi

Avtomat ish faoliyati vaqtning diskret momentida amalga oshadi, ya’ni t = 0, 1, 2, ..., T. *Avtomat* holatining o‘zgarishi $s_t \in S = \{s_0, s_1, ..., s_n\}$, ya’ni bir holatdan s_t boshqa holatga s_{t+1} o‘tishi chiqish signali y_t dan oldin, yoki undan keyin amalga oshishi mumkin. Shu qatorda, avtomatlar ikki turga bo‘linadi – Mila

avtomati va Mur avtomati. Bu avtomatlar o‘zaro funksional qonuni bilan ajralib turadi.

Mila avtomati funksional qonuni:

$$s_{t+1} = \varphi(s_t, x_{t+1}),$$
$$y_{t+1} = f(s_t, x_{t+1}).$$

Mur avtomati funksional qonuni:

$$s_{t+1} = \varphi(s_t, x_{t+1}),$$
$$y_{t+1} = f(s_{t+1}, x_{t+1}).$$

Mur avtomatida chiqish funksiyasi *f* chiqish signaliga bog‘liq bo‘lmagan holda aniqlanadi, balki avtomatning ichki holatidan kelib chiqib, u ham o‘z navbatida kirish signali orqali aniqlanadi.

Misol. *Mur avtomatiga* misol qilib gazli suv savdosini amalga oshiruvchi avtomatni olish mumkin. *Mila avtomatiga* misol sifatida esa quyidagilarni olamiz: X = {x_1, x_2} , U = {u_1, u_2, u_3} , S = {s_0, s_1, s_2, s_3, s_4, s_5} , φ o‘tish funksiyasi va *f* chiqish funksiyalarini quyidagi jadvallar yordamida beramiz:

φ – o‘tish funksiyasi

s(t – 1)	S_1	s_1	s_2	s_3	s_3	s_4	s_5	
x(t)	X_1	x_2	x_1	x_2	x_1	x_2	x_2	x_1
s(t)	S_2	s_3	s_4	s_2	s_4	s_3	s_5	s_5

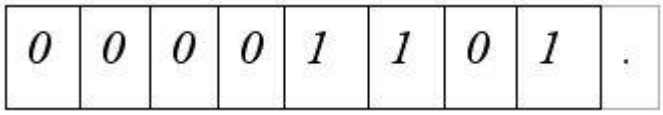
f – chiqish funksiyasi

s(t – 1)	S_1	s_1	s_2	s_2	s_3	s_3	s_4	s_5
x(t)	X_1	x_2	x_1	x_2	x_1	x_2	x_2	x_1
y(t)	U_2	u_3	y_1	y_1	y_3	u_2	u_3	y_2

Kompyuterni mutanosib ishlaydigan avtomatlar sifatida qarash mumkin. Bu tuzilmani atroflicha ko‘ramiz.

Kompyuter xotirasi – xotiraning yacheykalar ketma – ketligi bo‘lib, unda bitlar ketma – ketligini yozish yoki o‘qish mumkin.

Misol. 13_{10} sonini sakkiz razryadli yacheykaga (eng katta bit 0 yoki 1 raqamidan iborat bo‘lib, 1 – son manfiyligini va 0 – sonning musbatligini aniqlaydi) butun sonlar formatida yozish kerak. $13_{10} = 1101_2$ ekanligini bilib, quyidagi ko‘rinishga ega bo‘lamiz:



Mos ravishda kompyuter xotirasida haqiqiy sonlar, yoki alohida (butun qism alohida, kasr qism alohida), yoki maxsus, ya’ni normallashgan (faqat kasr qism va uning tartib raqami ikkining darajasi ko‘rinishida ifodalanadi va uning hisobi berilgan son ifodasiga teng bo‘ladi) ko‘rinishda ifodalanadi.

Misol. Agar son 5,25 bo‘lsa, u holda ikkilik sanoq tizimida – 101,01 ga teng bo‘ladi, va normallashgan ko‘rinishda: 0,10101 va tartib raqami ikkilikda 101 ga teng.

Buyruqlar, sonlar kabi bitli ko‘rinishda maxsus elektron qurilmada, ya’ni registrlarda saqlanadi.

Registr – bu elektron qurilma bo‘lib, xotira yacheykasi kabi ma’lum uzunlikka ega bitlar ketma – ketligini saqlaydi va uzatadi. *Registrlar* ancha qimmat va nozik qurilmalardan iborat bo‘lib, kompyuterning asosiy xotirasiga singari katta emas.

Misol. 512 megabaytga ega kompyuterning registr xotirasi 64 kilobaytga teng.

Har bir buyruqqa mos operatsiya moslanadi va bu operatsiyaning kodi qayta shifrlanadi. So‘ng ustida amal bajarilishi lozim bo‘lgan operand va sonlar ajratib olinadi. Bu operandlar ustida amallar bajarilgandan so‘ng natija xotiraning mos yacheykasiga joylashtiriladi.

Operativ xotiradan boshqa kompyuter katta hajmga ega, lekin yozish va o‘qish amali uchun katta vaqt sarflaydigan tashqi saqlash xotirasiga (TSX) ham ega (VZU). Tashqi xotira tashqi axborot tashuvchilar (magnit va optik disklar) bilan harakterlanadi.

Djon fon Neyman bir qator tamoyillarni taklif qildi:

1. xotira bir xil turdagi yacheykalardan iborat;

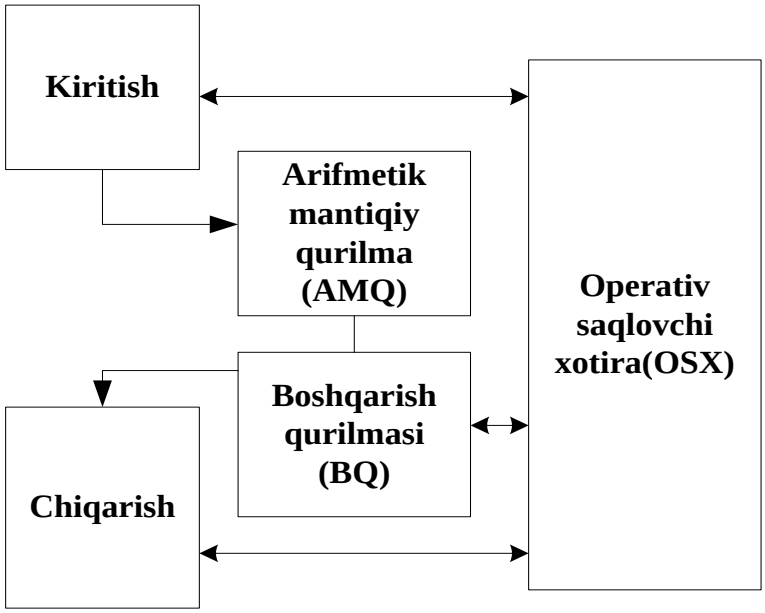
2. dastur buyruqlar ketma – ketligidan iborat;

3. dasturni saqlash va uning berilganlarini qayta ishlash bitli ko‘rinishda bo‘ladi;

4. buyruqlar ketma – ket bajariladi va buyruqlarga mos qiymatlar olinadi;

5. prosessor – yagona va xotiraga markazlashgan boshqaruvni amalga oshiradi.

Fon Neyman arxitekturasiga asoslangan EHM tuzilmasi 8.2. rasmda keltirilgan.



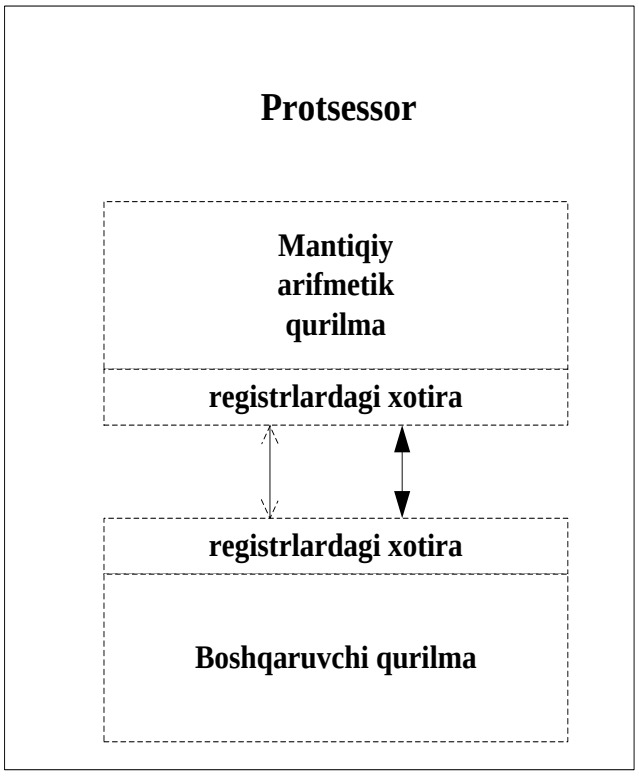
1.11-rasm. Fon Neyman arxitekturasiga asoslangan EHM tuzilmasi

Arifmetik – mantiqiy qurilma (AMQ) arifmetik va mantiqiy amallarni bajaradi.

Misol. Arifmetik – mantiqiy qurilma buyruqlari – sodda: «ikki sonni taqqoslash», «sonni uzatish», «diz’yunksiyasini hisoblash» va boshq.

Boshqarish qurilmalari (BQ) EHM ishini tashkil qiladi. Hususan, xotiradan keyingi buyruqni olish, buyruqni qayta shifrlaydi, buyruqni qayta shifrlash uchun xotiradan operandalarni olish va qayta shifrlangan amallarni amalga oshirish uchun AMQ ga uzatadi, olingan natijani xotiraga saqlash uchun uzatadi. Shu bilan bir qatorda BQ ish jarayonining yaxshi borayotganini yoki avariya holatini aniqlaydi.

AMQ va BQ ning birgalikdagi ishi, axborotli – boshqaruv yo‘nalishi kompyuterning ish jarayoni deb ataladi, 1.12-rasmda bu jarayon tuzilmasi keltirilgan (qalin chiziq – axborotli, boshqasi – boshqaruv).



1.12-rasm. Prosessor tuzilmasi

Kompyuter bilan ma’lumot almashinuvi kiritish va chiqarish qurilmalari orqali amalgan oshiriladi.

Misol. Kiritish qurilmalariga, misol uchun klaviatura, sichqoncha kiradi. Chiqarish qurilmalari esa — displey, printer, plotter.

Eng keng tarqalgan kompyuter turi – shaxsiy kompyuter. Shaxsiy kompyuterlar bir qator talablarga javob berishi kerak, masalan, kam harajat, kichik hajm, kam elektr energiya harajati, yuqori ishonchli, uning tarkibiy qismining mutanosib ishlashi, moslashuvchanlik va boshq.

Shaxsiy kompyuterlar yadrosi – bu tizimli plata bo‘lib, unda mikroprosessor, mikroprosessor xotira, mikroprosessorning boshqa qurilmalar bilan birikkan va aloqasini o‘rnatib beradigan interfeysli tizimi, taktli impulslar generatori, tizimli plataga biriktirilgan qurilmalar (sxemalar) boshqaruvi, tezkor saqlash xotirasi (TSX) va doimiy saqlash xotirasi (DSX) mikrosxemalari va boshqalar.

Shaxsiy kompyuterlarni boshqa asosiy qurilmalariga quyidagilar ham kiradi:

1. egiluvchan magnit disklar diskovodi; qattik magnit diskdar diskovodi;
2. CD-ROM (kompakt-disklarni o‘qish uchun qurilma) yoki CD-RW (o‘qish va yozish);
3. monitor (displey);
4. videokarta (videoadapter) tizimli blok va monitor aloqasini o‘rnatadi;
5. klaviatura;
6. printer;
7. skaner;
8. plotter;
9. digitayzer;
10. manipulyator – sichqoncha yoki manipulyator-trekbol;
11. ovoz kartasi (adapter);
12. ovoz kolonkalari;
13. modem va boshqa qurilmalar.

Kompyuter klassifikasiyasi uning tezkorligi, ishlash jarayoni va boshqalar. Belgilarga qarab ajratiladi. Bunga misol sifatida quyidagilarni aytib o‘tamiz

1. Superkompyuterlar – Dunyodagi eng murakkab masalalarni yechishda qo‘llaniladi. Masalan, koinotni tekshirish, yadro fizikada, geologiyada va boshqa sohalarda.
2. Universal kompyuterlar, murakkab va katta miqdordagi masalalarni yechish uchun mo‘ljallangan.
3. Shaxsiy kompyuterlar, yakka foydalanuvchining foydalanishi uchun, ya’ni murakkab bo‘lmagan, katta bo‘lmagan, va shu bilan bir qatorda katta va murakkab masalalarni yechish uchun xizmat qiladi.

Misol. Super hisoblash markazi davlat miqyosidagi masalalarni yechish uchun, misol uchun, himoya muammolari, koinot muammolari, ob – havo ma’lumotlarini olish, qayta ishlash, makroiqtisodiy ma’lumotlarni qayta ishlash uchun xizmat qiladi. Markazda ShEHMlar (markaz ishchilarining ish o‘rinlarida)

va universal kompyuterlardan ham, qo‘shimcha muammolarni yechish uchun foydalaniladi.

Kompyuterlardan foydalanishda ma’lum bir qator sanitar – gigiyena qoidalariga ham amal qilish lozim, chunki kompyuter inson sog‘ligiga ta’sir etuvchi faktorlari mavjud: nurlanish (infraqizil, rentgen nurlari, elektromagnit nurlari); to‘lqin va shovqin; elektrostatik maydon; monitorning chastotali ultratovushi va boshqalar.

Kompyuter zallarida kompyuterlardan foydalanishda quyidagi oddiy sanitar – gigiyena qoidalariga rioya qilish lozim:

- Kompyuterda bir momentning o‘zida 4 soatdan ortiq ishlamaslik (har bir soat oralig‘ida 10 daqiqa dam olish va intensiv ishlashdan so‘ng 2 soat dam olish);
- Ko‘z va monitor oralig‘i 0,6 metrdan kam bo‘lmasligi lozim;
- Displey foni sariq, yashil, kulrang yoki havorang ustunligi;
- Havo harorati 15-25 gradus (Selsiy bo‘yicha) bo‘lishi lozim;
- Havo namligi 45-75% bo‘lishi lozim;
- Ish stoli kamida 0,3x1,0 m bo‘sh bo‘lishi;
- Ekran maydoni kamida 17 dyum bo‘lishi;
- Kadr almashinuvi chastotasi kamida 70 Gs bo‘lishi;
- Kadrlar chastotasi kamida 75 Gs bo‘lishi lozim;
- Standart xavfsizlik, masalan MPR-II.

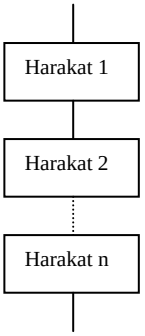
Professional masalalarni yechishda eng minimal bilim va kompyuter yordamida turli axtorot tizimlari va tarmoqlari orqali olingan ma’lumotlar bilan to‘ldirish, shu bilan bir qatorda kompyuter yordamida turli maishiy muammolarni hal etish – *kompyuter savodxonligi* deb ataladi.

Algoritm tuzilishi va turlari

Har kanday algoritmning mantiqiy tuzilishi uchta asosiy elementlar orqali ifoda qilinishi mumkin:

ketma-ketlik(chiziqli) , tarmoqlanish, takrorlanish (sikl)

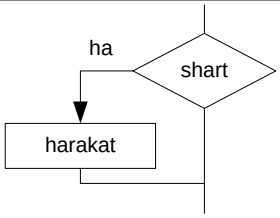
1. Chiziqli algoritm tuzilmasi ketma-ket bajariladigan buyruqlar tizimidan iborat bo‘ladi:

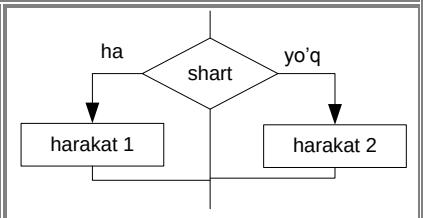
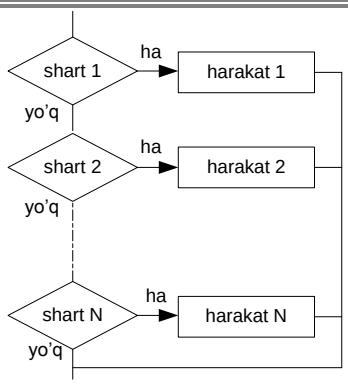
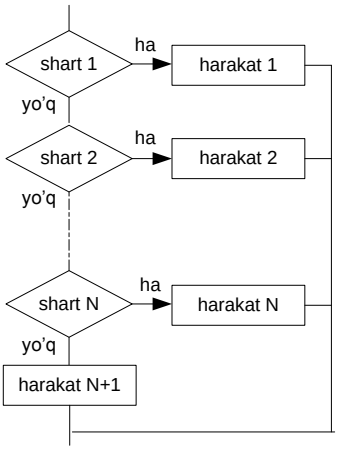
Harakatlar	*Blok-sxema*
harakat 1 harakat 2 . . . . . . harakat n	

2. Tarmoqlanish. Bu tuzilma shart bajarilishi natijasiga qarab (ha yoki yo‘q) algoritmni bajarish yo‘nalishini belgilaydi.

Tarmoqlanish tuzilmasi to‘rtta ko‘rinishda bo‘lishi mumkin:

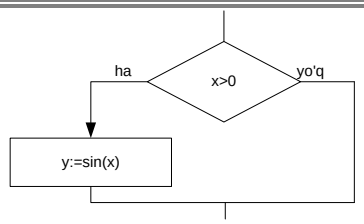
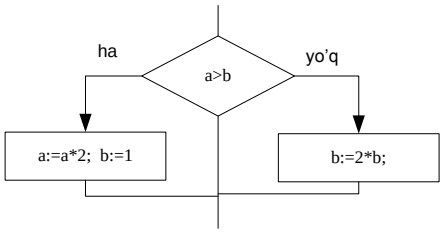
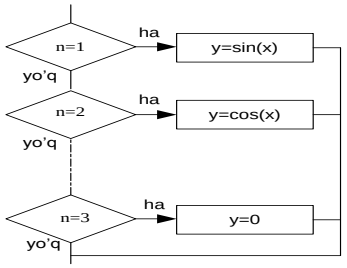
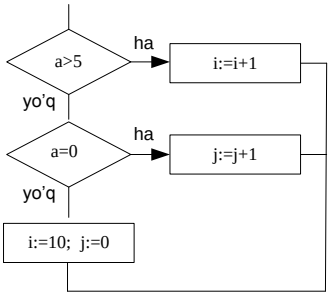
- agar-u holda;
- agar-u holda-aks holda;
- shartlar ketma-ketligi agar-u holda;
- shartlar ketma-ketligi agar-u holda-aks holda.

Algoritmik til	Blok-sxema
1. agar-u holda	
agar shart u holda harakat tamom	

2. agar-u holda-aks holda	
Agar shart u holda harakat 1 aks holda harakat 2 tamom	
3. tanlash	
Tanlash agar shart 1: harakat 1 agar shart 2: harakat 2 . . . . . . agar shart N: harakat N tamom	
4. tanlash-aks holda	
Tanlash Agar shart 1: harakat 1 Agar shart 2: harakat 2 . . . . . . agar shart N: harakat N aks xolda harakat N+1 tamom	

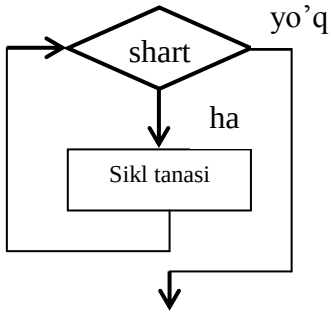
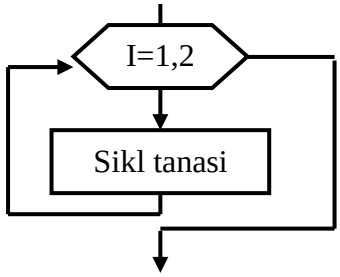
Agar buyrug‘iga misollar

Algoritmni so‘zlar yordamidagi ifodasi	Blok-sxema ko‘rinishidagi ifodasi

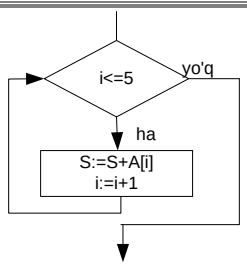
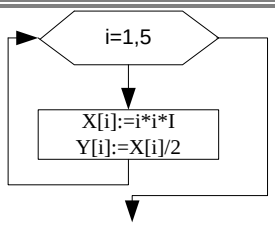
Agar x > 0 u holda y := sin(x) tamom	
Agar a > b u holda a := 2*a; b := 1 aks holda b := 2*b tamom	
Tanlash Agar n = 1: y := sin(x) Agar n = 2: y := cos(x) Agar n = 3: y := 0 Tamom	
Tanlash agar a > 5: i := i+1 agar a = 0: j := j+1 aks xolda i := 10; j:=0 tamom	

3. Sikl tuzilmasi buyruq, ko‘rsatma va amallarni ko‘p marotaba bajarilishini ta’minlaydi. Takrorlashni ta’minlashning asosiy turlari ushbu jadvalda ko‘rsatilgan:

Og‘zaki so‘zlar orqali	Blok-sxema tilida
Toki sikl turi Toki so‘zidan keyin keluvchi shart bajarilgunga qadar sikl tanasida ko‘rsatilgan buyruqlar bajariladi.	

Sikl boshi toki shart Sikl tanasi (buyruklar ketma-ketligi) Sikl oxiri	
Uchun sikl turi Sikl o‘zgaruvchisi (sikl parametri) barcha qiymatlarni qabul qilgunga qadar sikl tanasida ko‘rsatilgan buyruqlar bajariladi.	
Sikl boshi i uchun 1 dan 2 gacha Sikl tanasi (buyruqlar ketma-ketligi) Sikl oxiri	

Toki va *uchun* buyruqlariga misollar

So‘zlar orqali	Blok-sxema orqali
Sikl boshi toki i <= 5 S := S+A[i] i := i+1 sikl oxiri	
Sikl boshi i uchun 1 da 5 gacha X[i] := i*i*i Y[i] := X[i]/2 sikl oxiri	

Muhokama savollari

1. Bajaruvchi tushunchasi.
2. Avtomat tushunchasi.
3. Algoritm turlari.
4. Algoritm tuzilishi

Nazorat savollari:

1. Algoritm tushunchasi.
2. Algoritm turlari.
3. Fon Neyman arxitekturasi.
4. Toki va uchun buyruqlari.
5. Chiziqli algoritm.
6. Tarmoqlanish algoritmi
7. Takrorlanuvchi algoritm.

1.9 Dasturiy va texnik ta’minot. Operatsion tizim. Rasmiy tillar imlosi

Hozirgi zamon kompyuterlari asosan shaxsiy kompyuterlar bo‘lib, ularning texnik tuzilishini 3 qismga bo‘lish mumkin:

- Tizim bloki;
- Klaviatura;
- Displey;
- Printer.

Kompyuterning asosiy qismi bu tizim blokidir. U o‘z ichiga:

- kompyuter ishini boshqaruvchi hamma elektron sxemalarni (bular mikroprosessor, tezkor xotira, qurilmalarning kontrollerlari va shinalardan iborat);

- kompyuterning elektron sxemalariga berilayotgan o‘zgaruvchi tokni past kuchlanishli o‘zgarmas tokka o‘tkazuvchi ta’minlash blokini;

- ma’lumotlarni disketlarga yozuvchi va disketlardagi ma’lumotlarni xotiraga kirituvchi diskovodlarni;

- ma’lumotlarni qattiq diskga yozuvchi yoki undan o‘quvchi disk(vinchester)ni oladi.

Klaviatura ma’lumotlarni kompyuter xotirasiga kiritish uchun ishlatiladi.

Displey ma’lumotlarni kiritayotganda ekranda ko‘rish va xotiradagi ma’lumotlarni ekranda ko‘rish uchun ishlatiladi.

Printer matnli va grafikli ma’lumotlarni qog‘ozga chiqarish uchun ishlatiladi. Bu qurilmalardan tashqari kompyuterga yana har xil kiritish va chiqarish qurilmalarni ulash mumkin. Bularga quyidagi qurilmalar kiradi:

«Sichqoncha»- ma’lumotlarni kiritish ishlarini yengillashtiradigan qurilma.

Modem - telefon tarmog‘i orqali boshqa kompyuterlar bilan o‘zaro ma’lumotlarni bir-biriga uzatish qurilmasi.

Faks- modem telefaks ishlarini bajaradigan qurilma.

Skaner qog‘ozdagi matnli va grafikli ma’lumotlarni xotiraga kirituvchi qurilma. Uning ishlash prinsipi xuddi kserokopiyaga o‘xshagan bo‘ladi.

Plotter- rasmlarni qog‘ozga chiqarish qurilmasi.

Audioplata - kompyuterda ovoz chiqarish qurilmasi.

Kompakt diskdan ma’lumotlarni o‘quvchi qurilma. Hozirgi vaqtda egiluvchi disklardan boshqa katta hajmli (650M bayt) maxsus disklar ham yaratilgan. Bu qurilma shunday disklardan ma’lumotlarni o‘qish uchun ishlatiladi.

Trekbol - xuddi «sichqoncha» ishini bajaradigan qurilma.

Tarmoq adapteri - mahalliy tarmoqqa ulovchi qurilma.

Grafik planshet- chizmalarni mashina xotirasiga kirituvchi qurilma.

Kompyuterning dasturiy ta’minoti. Axborot texnologiyasining ikkinchi asosiy qismi - dasturiy ta’minot qismidir. Har qanday takomillashgan, tez ishlaydigan kompyuter bo‘lmasin dasturiy ta’minoti bo‘lmasa, u temirdir, chunki kompyuterning ishlashi faqat dasturlar bilan bajariladi.

Dasturiy ta’minotni ikki guruhga ajratish mumkin:

- tizimli dasturiy ta’minot;

- amaliy dasturiy ta’minot;

Tizimli dasturiy ta’minotning asosiy vazifasi asbob - uskuna vositalarining ishlarini boshqarish, uning imkoniyatlaridan to‘liq foydalanishni, odamning kompyuter bilan dialog olib borishini va amaliy dasturlarni ishga tushirishni bajarishdir. U kompyuterdan foydalanuvchilarni va amaliy dasturlarni kompyuter qurilmalari bilan muloqotda bo‘lishning qulay usullari bilan taminlaydi.

Amaliy dasturiy ta’minot dasturlari kompyuterlarda amaliy masalalarni yechishni taminlaydi.

Hozirgi paytda hamma sohalarda bunday amaliy dasturlar juda ko‘p, minglab hisoblanadi. Ularni bajaradigan ishlariga qarab quyidagi guruhlarga ajratish mumkin:

- matn muxarrirlari;
- nashriyot tizimlari;
- jadval ma’lumotlarni qayta ishlash tizimlari;
- ma’lumotlar jamg‘armalarini boshqarish tizimlari.

Buning ichiga xalq ta’limi ishlarini boshqarish, xalq ta’limi muassasalari ishlarini boshqarish va fanlarni o‘qitish, sinov nazorat ishlarini bajaradigan amaliy dasturlar ham kiradi.

Dastlabki ikki guruh dasturlarining qiladigan ishlari matnni taxrir qilish, nashriyot ishlarini bajarishdir.

Jadval ma’lumotlarni qayta ishlash dasturlar to‘plami berilgan ifodalar bo‘yicha har xil ma’lumotlar jadvallarini hisoblash, har xil diagramma, grafiklar chizish, har xil jadval ko‘rinishida berilgan ma’lumotlarni qayta ishlash ishlarini bajaradi. Keng ommalashgan jadval prosessorlar Lotus 1-2-3, SuperCalc, Microsoft Excel va hokazolardir.

Axborot jamg‘armalarini qayta ishlash dasturlari katta hajmdagi axborotlar to‘plamlarini boshqarish ishlarini bajarishga mo‘ljallangan.

Eng sodda axborot jamg‘armalari bu bir o‘lchamli jadvallar bo‘lib, ular ustida kiritish, tuzatish, qidirish, saralash va har xil hisobotlar tayyorlash ishlarini bajarishdan iboratdir.

Hayotda esa ancha murakkab bir necha o‘lchamli jadval axborotlar bilan ishlashga to‘g‘ri keladi. Bunday hollarda kiritiladigan va chiqariladigan ma’lumotlarni foydalanuvchi qulay holda olish uchun maxsus axborot jamg‘armalari tuziladi. Bunday jamg‘armalarni qayta ishlash uchun maxsus dastur to‘plamlari tuzilgan. Misol qilib DBase, Fox Pro, Clepper, Paradox, Rbaselar dastularini keltirish mumkin.

Shunday dasturlar to‘plamlari va texnik hamda dasturiy ta’minot vositalari yordamida juda katta hajmdagi axborotlarni tez qayta ishlab kerakli natijalarni o‘z vaqtida olish mumkin.

OS – bu foydalanuvchi va kompyuterning o‘zaro ta’sirini va kompyuter ishini boshqarishni tashkil etuvchi dasturlar yig‘indisidir. OS asosan quyidagi tarkibiy qismlardan iborat bo‘ladi:

1. Boshqaruvchi dastur – OSning bu qismi kompyuter bilan chambarchas bog‘lik bo‘lib, uning asosiy vazifasi kompyuterning ish faoliyatini boshqarishdan iborat.

2. Buyruqlar interpretatori – foydalanuvchi bilan kompyuter o‘rtasida muloqotni o‘rnatadi, uning buyruqlarini qabul qiladi, izohlaydi va bajarilishini ta’minlaydi.

3. Fayl tizimi – OS tarkibiga kiruvchi dasturiy vositalar yig‘indisidan iborat bo‘lib, ma’lumotlarni kiritish - chiqarish amallarining bajarilishini ta’minlaydi.

4. Dasturlashtirish tizimi – OSning bu tarkibiy qismi algoritmik tillarni tarjima qilish vazifasini bajaradigan vositalardan iborat. ShKlarda beysik, paskal, SI kabi bir qancha tarjimon dasturlar bo‘ladi.

5. Texnik xizmat ko‘rsatuvchi dastur – magnit disklarni, disk yurituvchi qurilmalarning ish faoliyatini nazorat qiladi. Dastur va ma’lumotlarni printerlardan chiqaradi.

Yana OTning tarkibiy qismlaridan tashqari kompyuterning ichki va tashqi dasturiy vositalari ham mavjuddir.

Ichki dasturiy vositalariga OS asosida ishlovchi MSDOS, NC (Norton Kommander) va WINDOWSlar mavjud.

Tashqi dasturiy vositalari turli maqsadlarga moʻljallangan koʻplab amaliy dasturlarni oʻz ichiga oladi. Bular orasida Microsoft Office tarkibiga kiruvchi dasturlardan Word, Excel, PowerPoint, Access kabilarni yaxshi bilamiz.

MS DOS quyidagi modullardan tashkil topadi:

1. BIOS (Basics input-output system)- kiritish - chiqarish tayanch tizimi. BIOS doimiy xotirada joylashgan boʻladi, ya'ni u kompyuterga yozilgan holda korxonadan chiqariladi. Kompyuter yoqilishi bilan xotirani, qurilmalar ishini maxsus testlar yordamida tekshiradi. Oxirida OSning yuklovchisini chaqiradi. OSning yuklovchisi disketning yoki vinchesterning 1-sektoriga yozilgan boʻladi.

2. IO.SYS orqali barcha kiritish-chiqarish amallari tashqi qurilmalar yordamida bajariladi.

3. MS DOS.SYS dasturi magnitli disk qurilmalarini yoqib-oʻchirish, oʻqish perosini kerakli joyga keltirish, fayllarni hosil qilish imkoniyatini beradi.

4. COMMAND.COM MS DOS ning buyruqli prosessori deb yuritiladi. U ichki buyruqlarni oʻzidan, tashqi buyruqlarni esa disklardan qidiradi. Buyruqlarni topgach, xotiraga yuklaydi va unga boshqaruvni topshiradi.

5. Qurilmalar drayverlari - MS DOS ning kiritish - chiqarish tizimlarini toʻldiradi va yangi qurilmalarga xizmat koʻrsatadigan yoki bor qurilmalardan foydalanadigan maxsus dasturdir. Drayverlar kompyuter xotirasiga OSning yuklanishida yuklanadi. Ularning nomlari CONFIG.SYS faylida koʻrsatiladi, bu esa MS DOS buyruqlarini faylga tegmasdan bajarish imkoniyatini beradi.

Kompyuterda axborotlar fayl koʻrinishida saqlanadi. Fayl – bu ma'lum bir axborot birligi boʻlib, biror nom bilan ataladigan soha. Fayllar 2 xil boʻladi: matnli va dasturiy. Matnli fayllarni ekranda koʻrish va uni taxrir qilish mumkin. Dasturiy fayllarni esa taxrir qilib boʻlmaydi. Fayl nomi 2 qismdan iborat: uning nomi va uning kengaytmasi. Fayl nomi ixtiyoriy harf va sonlardan iborat boʻlib, umumiy soni 256tadan oshmasligi kerak. Faylning kengaytmasi nuqtadan keyin yoziladi va unga qarab faylning turini aniqlab olish mumkin:

.pas - Pascal tilidagi dastur fayli

.bas – Beysik tilidagi dastur fayli

.cpp – C++ tilidagi dastur fayli

.bak – faylning ehtiyot qilingan nusxasi

.doc – Wordda hosil qilingan matnli fayl

.xls – Excelda hosil qilingan fayl

.ppt – PowerPointda hosil qilingan fayl

.mdb – Accessda hosil qilingan fayl

.rar – arxivlangan fayllar

.jpg – rasmli fayl

.avi – musiqali fayl va h.k.

Dasturiy mahsulotlarning o‘ziga xos xususiyatlari va ularning tasniflanishi haqida so‘z yuritishdan oldin, dasturlash, dastur, dasturiy ta’minot va dasturiy mahsulot tushunchalariga to‘xtalib o‘tish zarur.

Ma’lumki, kompyuterning faqatgina texnik ta’minoti, ya’ni qurilmalarning o‘zidan foydalangan holda har qanday amalni bajarib bo‘lmaydi. Qo‘yilgan vazifani va uni qanday bajarish kerakligini kompyuterga «tushuntirish» *dasturlash* (*programming*) jarayonining vazifasidir. Dasturlash qo‘yilgan talablar tahlili, ishlab chiqarish va amaliyotga qo‘llashning barcha bosqichlarni o‘zida mujassamlashtirgan dasturni yaratish jarayonini ifodalaydi.

Biror hisoblash tizimida bir butun qilib taqdim qilinishi mumkin bo‘lgan va ushbu tizimning faoliyatini boshqarish uchun ishlatiladigan operatorlar to‘plami *dastur* deb ataladi [2]. Yoki, boshqacha aytganda, dastur — biror bir vazifani hal qilish, bajarish uchun maxsus tartiblangan buyruqlar, yo‘riqnomalar majmuidir [3, 7]. Dastur tushunchasi ancha keng ma’noga ega bo‘lib, u nafaqat raqamli, balki dastlabki kod ko‘rinishidagi matn bo‘lishi ham mumkin.

Dasturiy ta’minot tushunchasiga ko‘plab adabiyotlarda [2,4-6] deyarli bir xil ta’rif berilgan. Ularni umumlashtirgan holda quyidagi ta’rifni keltirish mumkin.

Dasturiy ta’minot — elektron-hisoblash mashinalarida ishlatish uchun mo‘ljallangan, ishlab chiqarish jarayonida sinovlardan muvaffaqiyatli o‘tgan, belgilangan sifat ko‘rsatgichlariga ega bo‘lgan va undan samarali foydalanish uchun kerak bo‘lgan qo‘llanmalarlar bilan ta’minlangan dastur yoki dasturlar

to‘plamidir [3]. Ayrim adabiyotlarda [4] dasturiy ta’minotga ta’rif berishda ularni faqat shaxsiy kompyuterlar bilan bog‘langanligini ko‘rish mumkin. Bunday cheklanish albatta noto‘g‘ri chunki, bugungi kunda dasturiy ta’minot nafaqat shaxsiy kompyuterlarda deyarli har qanday raqamli qurilmalarda mavjud. Ular qatorida mobil telefonlar, planshetlar, turli pleyerlar, o‘yin konsollari, turli zamonaviy maishiy texnikalarni ko‘rsatish mumkin. Dasturiy ta’minot haqida gapirilganda ko‘p hollarda *dasturiy vosita* va *dasturiy mahsulot* atamalarining aralash ishlatilganligini kuzatish mumkin. Darhaqiqat, dasturiy mahsulot va dasturiy vosita tushunchalari bir narsani ifodalasada, ularning qaysi birini ishlatish odatda gapirilayotgan Ob’yektga qanday nuqtai nazar bilan yondashilayotganligiga bog‘liqdir. Agar dastur yoki dasturlar to‘plamiga biror vazifani bajarish vositasi sifatida qaralayotgan bo‘lsa, dasturiy vosita, biror ishlab chiqarish jarayonining mahsuli sifatida qaralayotgan bo‘lsa dasturiy mahsulot atamasini qo‘llash maqsadga muvofiq.

Dasturiy mahsulot tushunchasiga turli adabiyotlarda turlicha ta’riflar berilganligini ko‘rish mumkin. Quyida ushbu ta’riflardan bir nechtasi keltirilgan:

– dasturiy mahsulot — bu, sanoat mahsuloti sifatida realizatsiya uchun tayyorlangan, ommaviy talabning belgilangan vazifalarini yechish uchun mo‘ljallangan o‘zaro bog‘liq dasturlar kompleksidir [5];

– dasturiy mahsulot deganda belgilangan funksiyalarni bajarish uchun mo‘ljallangan o‘zaro bog‘liq dasturlar to‘plami (dasturlar paketi) tushuniladi;

– dasturiy mahsulot (software product): foydalanuvchiga yetkazib berish, uzatish, sotish uchun mo‘ljallangan dasturiy vositalar;

– dasturiy mahsulot (software product): kompyuter dasturlari to‘plami hamda ular bilan bog‘liq bo‘lgan hujjatlar va ma’lumotlar;

Bizning fikrimizcha, *dasturiy mahsulot* — sotish yoki foydalanish uchun taqdim qilishga mo‘ljallangan dastur yoki dasturlar to‘plami hamda u bilan taqdim qilinishi mumkin bo‘lgan qo‘llanma, hujjat va ma’lumotlardir.

Dasturiy mahsulot bir tomondan umumiy iqtisodiy qonunlarga bo‘ysunsada, boshqa tomondan bir qator o‘ziga xos xususiyatlarga egadir. Uning o‘ziga xos tomonlari quyidagilarda namoyon bo‘ladi :

– dasturiy mahsulot sotilayotganda, odatda mahsulotning o‘zi emas, balki, uni ishlatish uchun litsenziya sotilayotgan bo‘ladi;

– mahsulotning o‘zi asosan material ko‘rinishga ega bo‘lmasdan, uning fizik tarkibiy qismi (ma’lumot tashuvchi, o‘ram, qog‘ozga chop etilgan qo‘llanmalar va hokazolar)ning qiymati ma’lumotning qiymatidan ancha past bo‘ladi;

– dasturiy mahsulotlar sanoatida mahsulotning har bir qo‘shimcha birligini ishlab chiqarish deyarli hech qanday xarajat talab qilmaydi va xarajatlar nolga intiladi. Bu xususiyat ayniqsa zamonaviy ma’lumotlarni uzatish tarmoqlarining rivojlanishi bilan yaqqol ko‘zga tashlanmoqda. Bugungi kunda hech qanday fizik ko‘rinishga ega bo‘lmagan juda katta miqdordagi mahsulotlar mavjud bo‘lib, xaridorlar bunday mahsulotlarni onlayn rejimda Internet global tarmog‘ida topishadi, tanlashadi va sotib olishadi;

– ayrim dasturiy mahsulotlar turlari uchun mahsulotning qiymati unga bog‘liq bo‘lgan xizmatlar: tatbiq qilish, konsalting, qo‘llab quvvatlash va hokazolarga qaraganda ancha past bo‘lishi mumkin. Oxirgi vaqtlarda katta kompaniyalarning dasturiy mahsulot uchun millionlab dollar xarajat qilib turib, uni foydalanuvchilar uchun tekin (yoki ayrim cheklovlar bilan) tarqatishi tez-tez uchrab turibdi;

– dasturiy mahsulotlar fizik jihatdan eskirmagan holda, ularning ma’naviy eskirishi jarayoni juda tez yuz beradi.

Dasturiy mahsulotlarning o‘ziga xos tomonlari sabab ayrim klassik iqtisodiy qonunlarni ular uchun qo‘llab bo‘lmaydi yoki ular cheklangan holda amal qiladi. Bunday holatning ko‘plab sabablari qatorida quyidagilarni alohida ajratib ko‘rsatish mumkin [3]:

– qo‘shimcha mahsulotni ishlab chiqarish xarajatlari nolga teng emas va u ishlab chiqarish hajmining oshishi bilan kamayib boradi degan farazga asoslangan

talab va taklifning klassik egri chiziqlarini dasturiy mahsulotlar uchun qo‘llab bo‘lmaydi. Yuqorida aytib o‘tilganidek, dasturiy mahsulotlar sanoatida mahsulotni qo‘shimcha ishlab chiqarish deyarli hech qanday xarajat talab qilmaydi va xarajatlar nolga intiladi;

– dasturiy mahsulotlar bozoridagi holat juda tez o‘zgaradi — yangi texnologiyalar yuzaga keladi, yaqindagina eng mashhur bo‘lib turganlari esa aktual bo‘lmay qoladi;

– mahsulot uchun potensial bozorni prognozlash mumkin bo‘lsada, unga bo‘lgan talabni prognozlash ancha muammoli hisoblanadi. Bu odatda potensial xaridorlar uchun mahsulotning (ayniqsa u yangi texnologiya bo‘lsa) qanchalik ahamiyatga ega bo‘lishini taxmin qilishning qiyinligida o‘z aksini topadi;

– Alohida olingan mahsulot uchun narx egiluvchanligini oldindan taxmin qilishning deyarli iloji yo‘q. Bundan kelib chiqadiki, narxlar talab va taklif egri chiziqlariga asoslangan bo‘la olmaydi.

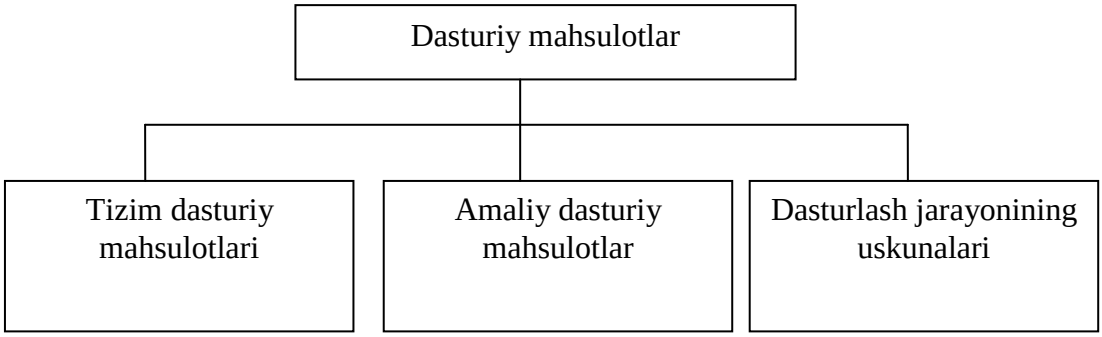
Dasturiy mahsulotlar uchun mijoz odatda aniq bo‘lmasdan, u qandaydir bir vazifani yechishga mo‘ljallangan bo‘ladi. Bunday holatda dasturiy mahsulotlar reklama qilinadi va odatiy ishlab chiqarish mahsulotlari kabi sotiladi. Ammo, dasturiy mahsulotlar oldindan aniq bo‘lgan buyurtmachi uchun, u tomonidan qo‘yilgan maxsus texnik vazifalar asosida ham ishlab chiqarilishi mumkin. Bunday dasturiy mahsulotlar odatda reklama qilinmaydi.

Dasturiy mahsulot odatda yuqori malakali mutaxassislar ishining natijasi bo‘lib intellektual mehnat mahsuloti sifatida qonunchilik tomonidan himoya qilinadi. Zamonaviy dasturiy mahsulotlarni ko‘pchiligi ishlab chiqaruvchi tomonidan kuzatib boriladi. Dasturiy mahsulotning kuzatib borilishi deganda uning to‘g‘ri ishlashini ta’minlab borish, yangi, mukammallashtirilgan versiyalariga o‘tkazish, o‘zgarishlar kiritish, dasturni ishlatish davomida topilgan xatolarni bartaraf qilish bilan bog‘liq ishlar tushuniladi.

Dasturiy mahsulotlarning turli xususiyatlariga ko‘ra tasniflash mumkin. Dasturiy mahsulotning ishlatilish sohasi yoki funksional mo‘ljallanganlik

alomatiga koʻra tasniflash eng koʻp uchraydi. Unga koʻra dasturiy mahsulotlar quyidagi uchta sinfga ajratiladi (1.13-rasm):

- tizim dasturiy mahsulotlari;
- amaliy dasturiy mahsulotlar;
- dasturlash jarayonining uskunalari.



113-rasm. Dasturiy mahsulotlarning ishlatilish sohasiga koʻra tasniflanishi

Tizim dasturiy mahsulotlari kompyuter texnik qurilmalarining ish jarayonini tashkillashtirish va amaliy dasturlar ishlashi uchun platforma vazifasini oʻtovchi dasturlardir. Tizim dasturiy mahsulotlari sinfiga quyidagi guruhlarni kiritish mumkin:

1) Operatsion tizimlar. Hisoblash va qayta ishlash jarayonlarini tashkillashtirish, resurslarni (tezkor xotira, prosessor, disk) taqsimlash, foydalanuvchining dasturlarini ishga tushirish, foydalanuvchining kompyuter bilan muloqotini ta'minlash (interfeys) kabi vazifalarni bajaradi. Operatsion tizimlarga misol qilib MS DOS, Windows, UNIX, Linux, Mac OSX, iOS, Android kabi tizimlarni koʻrsatish mumkin;

2) Operatsion qobiqlar — Foydalanuvchining operatsion tizim buruqlaridan foydalanishini osonlashtirish uchun moʻljallangan maxsus dasturlar. Operatsion qobiqlar matnli va grafikali foydalanuvchi interfeysiga ega boʻladi. Bunday guruhdagi dasturlarga misol qilib Norton Commander, Midnight Commander, FAR Manager, Total Commander kabi dasturlarni koʻrsatish mumkin.

3) Drayverlar — Qurilmalarni boshqaruvchi, yangi tashqi qurilmalarni ulash va bor qurilmalar (video karta, klaviatura, sichqoncha, modem va hokazolar)dan nostandart foydalanish imkoniyatini beruvchi maxsus dasturlar.

4) Utilitalar — tizimga xizmat koʻrsatish jarayonida biror maxsus vazifani bajarish uchun yordamchi dasturlar (antivirus dasturlari, arxivatorlar, diagnostika dasturlari va hokazolar).

Amaliy dasturiy mahsulotlar amaliy vazifalarni bajarish uchun moʻljallangan boʻlib, foydalanuvchining kerakli vazifalarni kompyuterda bajarishini ta'minlaydi. Amaliy dasturiy mahsulotlar foydalanuvchidan axborot texnologiyasi boʻyicha chuqur bilimlarni talab qilmagan holda uning kasbiy faoliyatidagi vazifalarini bajarish uchun yordamchi boʻladi. Bu sinfga ishlatilish sohasiga koʻra ajratiladigan juda koʻp guruhlar kiradi:

– matnlar (hujjatlar) tayyorlash uchun dasturlar (Microsoft Word, OpenOffice.org Writer);

– jadvalli ma'lumotlarni qayta ishlashga moʻljallangan dasturlar (Microsoft Excel, OpenOffice.org Calc);

– ma'lumot bazalarini boshqarish tizimlari (Microsoft Access, MS SQL, Oracle, MySQL);

– taqdimotlar tayyorlash uchun dasturlar (Microsoft Power Point, OpenOffice.org Impress);

– matematik dasturlar (Matlab, Maple, Mathematica);

– CASE texnologiyalar (Vantage Team Builder, Rational Rose, Microsoft Project);

– nashriyotlar uchun tizimlar (Page Maker, Adobe InDesign);

– bugʻalteriya dasturlari (1C Buxgalteriya, BEM);

– huquqiy axborot bazalari (NORMA, BEM Info);

– bank tizimlari;

– grafika, animatsion va video filmlar yaratish uchun dasturlar;

– avtomatlashtirilgan loyihalash tizimlari (SAPR, Kompas, AutoCad);

– statistik ma’lumotlar analizi uchun dasturlar (Statbraph, Statistica);

– matnlarni aniqlash dasturlari (ABBYY FineReader, Cunieform);

– tarjimon va lug‘at dasturlar (ABBYY Lingvo, PROMT, SPELLS Lug‘at);

– imlo va grammatikani tekshirish uchun dastrurlar;

– kompyuter o‘yinlari, ta’lim dasturlari, spravochniklar va hokazolar.

Amaliy dasturiy mahsulotlar sinfi tarkibidagi yuqorida keltirilgan guruhlardan eng asosiylaridan bo‘lib, bu ro‘yxatni yana uzoq davom ettirish mumkin. Amaliy dasturiy mahsulotlarning har xil turlarining yagona inegratsion paketga birlashish holatlari ham juda ko‘p uchraydi (Microsoft Office, Adobe Creative Suit, OpenOffice.org). Bundan tashqari alohida dasturiy mahsulotlar tarkibida boshqa turdagi dasturlarning bo‘lishi ham ko‘p uchraydigan holat. Masalan, deyarli barcha matnlar (hujjatlar) tayyorlash uchun mo‘ljallangan dasturlar tarkibida imloni tekshirish komponentasi mavjud bo‘ladi.

Dasturlash jarayonining uskunalari yangi dasturiy mahsulot yaratish va uni tuzatish jarayonida foydalaniladigan dasturiy mahsulotlar bo‘lib, ularning quyidagi asosiy turlarini ko‘rsatish mumkin:

– foydalanuvchi interfeysini yaratish vositalari;

– dastur matnini yozish, tahrirlash va versiyalarni boshqarish vositalari;

– standart dasturiy kutubxonalar;

– kompilyator va interpretatorlar;

– dasturni sinash va tuzatish vositalari.

Dasturiy mahsulotlarning ishlatilish sohasiga ko‘ra tasniflanishining batafsil sxemasi 2-ilovada ko‘rsatilgan.

Yuqorida keltirilgan va asosan ko‘pgina darsliklarda [1,4,5,6,7] uchraydigan tasniflashda bir qancha kamchilikni ko‘rsatish mumkin:

1) Operatsion qobiqlarning o‘rniga fayl boshqaruvchilari atamasining qo‘llanilishi maqsadga muvofiq;

2) Tizim dasturiy mahsulotlarning bir qancha turlari, xususan tarmoqlar bilan bog‘liq bo‘lgan dasturlar, server dasturlari e’tiborga olinmagan;

3) Amaliy dasturlar guruhlarida ko‘rsatilgan ma’lumotlar bazalarini boshqarish tizimlari va CASE texnologiyalarning hamma turlarini ham to‘liqligicha amaliy dasturiy mahsulotlar deb ko‘rsatish to‘g‘ri emas. Ular odatda murakkab tizimlar bo‘lib, ko‘proq tizim dasturiy mahsulotlari sinfiga xos bo‘lishadi. Tizim dasturiy mahsulotlaridagi utilitalar esa ko‘proq amaliy dastur sifatida ishlatiladi.

Muallif fikricha, dasturiy mahsulotlarni tasniflashda NAPCS (North American Product Classification System) tasniflash tizimini asos qilib olish maqsadga muvofiqdir. Ushbu tizim AQSh, Kanada va Meksika davlatlari statistika qo‘mitalarining qo‘shma tashabbusi bilan ishlab chiqilgan bo‘lib, o‘zida san’atning 12 ta sektorida ishlab chiqariladigan mahsulotlar ro‘yxatini mujassamlashtirgan.

NAPCS tasniflash tizimiga ko‘ra dasturiy mahsulotlar ikki sinfga bo‘linadi: tizim dasturiy mahsulotlari (System Software) va amaliy dasturiy mahsulotlar (Application Software). Dasturlash jarayonining uskunalari tizim dasturiy mahsulotlari tarkibiga kiritilgan.

Muhokama savollari

1. Zamonaviy kompyuterlarning arxitekturasi.
2. Kompyuterning texnik ta’minoti.
3. Kompyuterning dasturiy ta’minoti.
4. Operasion tizim.
5. Fayl va kataloglar tushunchasi .
6. Dasturiy mahsulot va uning xususiyatlari.

Nazorat savollari

1. Kompyuterning dasturiy ta’minoti
2. Dasturiy ta’minotning tarkibi.
3. Faylni nomlash.
4. Operasion sistema qismlari?
5. Fayl tushunchasi?

6. Katalog tushunchasi?

7. Kompyuterning fayllar tizimi.

1.10 Ob’yektlar, jarayonlar va ko‘rinishlarni modellashtirishga kirish. Modellashtirishning asosiy tushunchalari

Model va modellashtirish bu universal tushunchalar bo‘lib, Ob’yekt, jarayon va hodisalarni (model va modellashtirish orqali) hamda ixtiyoriy kasbiy sohani anglashning eng zo‘r metodlarining atributlaridan biridir.

Model va modellashtirish mazkur modellar va modellashtirish natijalari qayerda qo‘llanilishidan qat’iy nazar fanlararo muammolarni yechish ustida ishlayotgan har xil soha mutaxassislarini birlashtiradi.

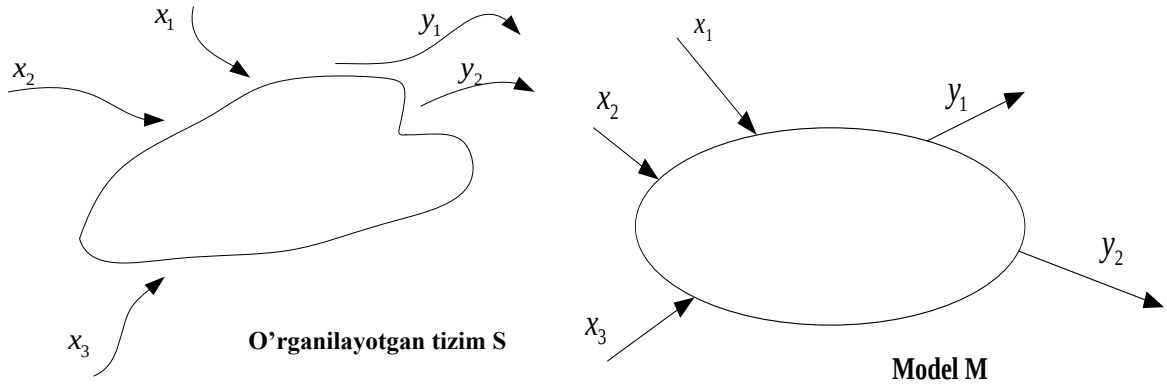
Model bu asl nusxani qandaydir yo‘l orqali tasvirlash va ta’riflashdan iborat bo‘lib, asl nusxa hodisa bo‘yicha ma’lum bir gipoteza va aniq bir taxminlarga ko‘ra uni a’lo darajada o‘rganish, tadqiq qilish, uning xossalarini tavsiflash uchun asl nusxani almashtirishga imkon beradi.

Misol: h balandlikdan tashlangan va t vaqt ichida erkin tushayotgan erkin jism uchun $h = \frac{gt^2}{2}$ munosabatni yozish mumkin. Bu jismni erkin tushish masofasini fizik – matematik modeli. Ushbu modelni qurish uchun quyidagi gepotezalar qabul qilingan: 1)tushish jarayoni vakuumda sodir bo‘ladi (havoni qarshilik kaeffisenti nolga teng); 2) shamol yo‘q; 3) jismni massasi o‘zgarmas; 4) jism ixtiyoriy nuqtada g tezlanish bilan harakat qiladi.

“Model” so‘zi (lotincha “madelium” so‘zidan olingan bo‘lib) “o‘lchov”, “usul”, “biror narsaga o‘xshash” ma’nosini anglatadi.

Modellashtirish muammosi uchta bir-biriga bog‘liq masalalardan tashkil topadi: yangi model qurish; modelni o‘rganish (o‘rganish metodini ishlab chiqish); modelni qo‘llash (amaliyotga yoki nazariy jihatdan).

S tizimni X kiruvchi va Y chiquvchi signallari bilan M modelni qurish sxemasi 13.1 – chizmada keltirilgan.



1.14-rasm. Model qurish sxemasi

Agar M ga kirishga kelayotgan signallar X dan va chiqayotganda hosil bo‘lgan signallar Y dan bo‘lsa, u holda f qoida tizimning funksionallangan modellaridir.

Agar modelni tasvirlashda parametrlari orasida vaqt parametri qatnashmasa, bu – statik model deyiladi.

Agar modelni tasvirlashda parametrlari orasida vaqt parametri aniq ajralib tursa, bu – dinamik model deyiladi.

Agar model vaqtning barcha oralig‘ida asli tasvirlangan bo‘lsa, bu uzluksiz model deyiladi.

Agar modelda jami kirayotgan har bir parametrlari bar xil ma’noli chiquvchi parametrga ajratishga imkon bersa, bu – determinallashgan model, aks holda determinallashmagan deyiladi.

Agar modelda taqdim etilayotgan tizim funksional bo‘lsa (masalan, tenglama), bu – funksional model deyiladi.

Agar model qandaydir to‘plam, uning elementlari va ular orasidagi munosabatlardan iborat bo‘lsa, bu – nazariy to‘plamli model deyiladi.

Agar model pridekatlar, mantiq funksiyalari va munosabatlardan iborat bo‘lsa, bu – mantiqiy model deyiladi.

Agar model model osti tarkibli elemetnlari va ular orasidagi mantiqiy munosabatlari to‘g‘risidagi axborotlardan iborat bo‘lsa, bu – mantiqiy axborot model deyiladi.

Agar model elemetnlari (o‘yinning Ob’yekt va subektlari) orasida qandaydir o‘yinli vaziyatlarni amalga oshirsa, bu – o‘yinli model deyiladi.

Agar modelning funksional ta’riflanishi qandaydir algoritm yoki algoritmlar majmualari bilan tasvirlangan bo‘lsa, bu – algoritmli model deyiladi.

Agar model graflar yoki graflar orasidagi munosabatlardan iborat bo‘lsa, bu – grafli model deyiladi.

Agar model iyerarxik (daraxtsimon) tuzilishdan iborat bo‘lsa, bu – iyerarxik (daraxtsimon) model deyiladi.

Agar model qandaydir lingvistik Ob’yektlardan iborat bo‘lsa, bu – tilli yoki lingvistik model deyiladi. Ko‘p hollarda sintaksis model ham deyiladi.

Agar model, modellashtirilayotgan tizimda munosabatlar va aloqalarni vizuallashtirish imkoniyatini bersa (xususan dinamikada), bu – vizual model deyiladi.

Agar model material jihatdan asli nusxasiga ega bo‘lsa, bu – aslidan olingan model deyiladi.

Agar model geometrik shakllar va ular orasidagi munosabatlardan iborat bo‘lsa, bu – geometrik model deyiladi.

Agar modelning Ob’yektini bir nechta yoki barcha parametrlarini variasiyalash, sinash va o‘rganish orqali qurilgan bo‘lsa, bu – imitatsiya (yasama) model deyiladi.

Misol: $F = am$ modeli – og‘ma tekislik bo‘ylab jism harakatining statistik modeli.

Nyuton qonunining dinamik modeli: $F(t) = a(t)m(t)$ yoki, yanada aniqroq aytganda $F(t) = s''(t)m(t)$. Agar faqat $t = 0.1,0.2,\ldots,1\,(c)$ qaralsa, u holda $S_t = \frac{gt^2}{2}$ model yoki $S_0 = 0,\ S_1 = \frac{0.01g}{2},\ S_2 = \frac{0.04g}{2},\ \ldots,S_{10} = \frac{g}{2}$ sonli ketma-ketlik erkin

tushayotgan jism harakatining diskret modeliga hizmat qiladi. $S = \frac{gt^2}{2}$, $0<t<10$ model (0;10) oraliqda uzluksizdir.

$a_1x_1 + a_2x_2 = S$ model ikki xil 1 va 2 mahsulot ishlab chiqaruvchi korxonaning ekonomik tizimi modeli, x_1 va x_2 mos ravishda mahsulot miqdorlari, a_1 va a_2 lar mos ravishda mahsulot narxlari, S - korxonada ishlab chiqarilgan mahsulotning umumiy qiymati. Bu modelni imitatsion model sifatida qo‘llab S ning qiymatini ishlab chiqarilgan tovarning hajm mohiyatiga bog‘lab aniqlab olish mumkin. Yuqorida fizikaviy –determinasiyalashgan model keltirilgan.

Agar $S = \frac{gt^2}{2}$, $0<t<10$ modelda biz tasodifiy parametrni, ya’ni jism erkin tushayotganda shamolni birdan kuchayish kuchi p ni hisobga olsak, masalan, $S(p) = \frac{g(p)t^2}{2}$, $0<t<10$, u holda biz tushishning (endi erkin bo‘lmagan) stoxastik modelini hosil qilamiz. Bu shuning bilan birga funksional model hamdir.

X={Baxtiyor, Ilhom, Mustafoyev, Kamolov, Zilola, Ziyoda, Mansur, Rayxon} to‘plami uchun, Y: “Baxtiyor – Zilolani turmush o‘rtog‘i”, “Ziyoda – Ilhomning xotini”, “Rayxona – Baxtiyor va Zilolaning qizlari”, “Mansur – Ziyoda va Ilhomning o‘g‘li”. U holda X va Y to‘plamlar ikki oilaning nazariy to‘plamlar modelini tashkil qiladi.

$z = \bar{x} \wedge y \vee x \wedge \bar{y}$, $p = x \wedge y$ ko‘rinishdagi ikki mantiqiy funksiya majmui mantiqiy modelni tashkil etadi.

1 – o‘yinchi insofli soliq inspektori bo‘la qolsin, a 2 – o‘yinchi noinsof soliq to‘lovchi. “O‘yin” ketyapti: bir tomondan soliqdan bosh tortish va boshqa tomondan berkitilgan soliqlarni fosh qilish. O‘yinchilar i va $j\ (i, j \le a)$ natural sonlarni tanlaydilar, birinchi o‘yinchi ikkinchi o‘yinchini soliq to‘lamaganligini to‘lanmagan faktlar orqali bilib qolishi va yana vaqtinchalik foydasini soliqdan berkitishini ham bilib qolishi, ikkinchi o‘yinchiga mos ravishda shtraf to‘lattirish. A matrisaning har bir elementi $a_{ij} = |i - j|$ qoida bilan topiladi.O‘yin modeli ushbu matrisani va chetga chiqish, qo‘lga tushurish strategiyalarini tavsiflab beradi.

Hisoblashni algoritmik modeli cheksiz kamayuvchi sonli qatorlarning yig‘indisining, qatorning chekli yig‘indisidan to uning biror berilgan aniq darajasi uchun hisoblash algoritmi bo‘lib xizmat qiladi.

To‘g‘ri yozish qoidasi – tilli, strukturali (tuzulishli) model.

Globus – yer sharining asl geografik modeli.

Uy maketi qurilayotgan uyning asl geometrik modeli. Aylanaga ichki chizilgan ko‘pburchak aylananing kompyuter ekranidagi vizual geometrik modelni beradi.

Model tiplari va uning tabiatiga emas, balki aloqalarga, uning tizim osti va elementlari munosabatlariga bog‘liq.

Misol: Yuqumli kasalliklar epidemiyasi dinamikasini, radioaktiv elementlarning parchalanishi, ishlab chiqarish korxonalarda buyumlarni ishlab chiqarish va boshqalarning jarayonlari har xil bo‘lsa ham, matematik tavsiflanishi bir xil bo‘ladi.

Ixtiyoriy modelning asosiy xossalari:

- aniq bir maqsadga qaratilganligi;
- cheklilik;
- soddalashgan;
- yaqinlashmoqlik;
- aynan bir xillik (adekvantlik);
- axborotlik;
- to‘liqlik (mukammallik);
- yopiqlik va boshqalar.

Modellashtirish tizimining hayotiy sikli (davri):

- axborotni yig‘ish;
- loyihalashtirish;
- tuzilish;
- o‘rganish;
- baholash;

- ko‘rinishini o‘zgartirish (modifikasiya).

Modellashtirish fani: modellashtirish jarayoni (tizimlar, modellar) va yuqori darajada o‘rganilgan, hamda foydali tavsiflangan, formallangan bosqichlarga (qism tizimlar, qism modellar) bo‘linadi.

Matematik va komyuterli modellashtirishlarni har xil sohalarda qo‘llanilishiga misol keltiramiz:

- energetika: atomli reaktorlarni boshqarish, termoatom jarayonini modellashtirish, energetik jarayonlarni oldindan aytib berish, energoresurslarni boshqarish va h.k.;
- iqtisod: modellashtirishda sotsial iqtisodiy va iqtisodiy jarayonlarni oldindan aytib berish, banklararo hisob-kitoblar, avtomatlashtirish ishlari va h.k.;
- kosmonavtikada: kosmik kemalarni uchishini boshqarish va trayektoriyasini hisob-kitoblari, sun’iy yo‘ldosh axborotlarini qayta ishlash va h.k.;
- tibbiyot: modellashtirish, epidemiyani oldindan aytib berish, yuqumli kasalliklar jarayoni, davolanish jarayonini boshqarish, kasallik diagnostikasi va davolanishning yuqori bosqichlarini ishlab chiqish va h.k.;
- ishlab chiqarish: texnik va texnologik jarayonlarni va tizimlarni boshqarish, resurslar, rejalashtirishlar, ishlab chiqarishni yuksak jarayonga olib chiqishni oldindan aytib berish va h.k.;
- ekologiya: ekologik tizimni ifloslanishini modellashtirish, ekologik tizimda sababli – natija aloqalarini oldindan aytish va h.k.;
- ta’lim: tartib – intizomlar orasidagi aloqa tizimlarini modellashtirish strategik va taktik o‘qitish va h.k.;
- harbiy ishlar: harbiy mojarolarni oldindan aytib berishda va modellashtirish, jangovor vazifalarda, qo‘shinlarni boshqarishda, armiyani ta’minlashda va h.k.;
- siyosat: siyosiy vaziyatlarni oldindan aytib berish va modellashtirish va h.k.;

- sotsialogiya, jamoatchilik ilmi: ijtimoiy guruh va jarayonlarni hodisani oldindan aytish va modellashtirish, jamoatchilik harakati va ta’siri va h.k.;
- OAV: xabarlarni u yoki bu guruh kishilariga ta’sir etishi effektini oldindan aytib berish va modellashtirish, ijtimoiy qatlam va h.k.;
- turizm: turistlar oqimini oldindan aytib berish va modellashtirish, turizm infrastrukturasini rivojlanishi va h.k.;
- loyihalash: turli tizimlarni loyihalash, modellashtirish, eng yaxshi loyihalar ishlab chiqish, loyihalash jarayonini avtomatik boshqarish va h.k.;

Murakkab jarayonlarni zamonaviy modellashtirish kompyutersiz va kompyuterli modellashtirishsiz yaratish mumkin emas.

Kompyuterli modellashtirish – kompyuter yordamida va ixtiyoriy axborotlardan foydalanib, kompyuterda singari tayanch bilimlarini namoyish etish va qaysi birlaridandir EHM da aktivlashtirish mumkin.

Kompyuterli modellashtirish turi – hisoblash tajribasi, eksperiment o‘tkazuvchi kompyuter yoki kompyuter texnologiyalari yordamida tizim yoki jarayon ustida tadqiqotni amalga oshirish.

Hisoblash tajribasi yangi Ob’yektiv qonuniyatlarga asoslangan, gipotezalarni sinaydi, xabarlarni vizuallashtiradi va h.k..

Kompyuterli modellashtirish boshidan oxirigacha quyidagi bosqichlarni bosib o‘tadi.

1. Masalani qo‘yilishi.
2. Modeloldi tahlil.
3. Masalaning tahlili.
4. Modelni o‘rganish.
5. Dasturlash, dasturni loyihalash.
6. Testlash va sozlash.
7. Modellashtirishni baholash.
8. Hujjatlashtirish.
9. Kuzatib borish.
10. Modelni foydalanish(qo‘llash).

Misol: Baliqchilikning ko‘payishi ko‘rib chiqamiz, ularning ichidan shu vaqt ichida ayrim miqdorda ushlansin (olib tashlansin). Bunday tizimning dinamikasi quyidagi model bo‘yicha aniqlandi: $x_{i+1} = x_i + ax_i - kx_i$, $x_0 = c$, bu yerda k - ushlash koeffisiyent, bitta ushlangan baliq narxi b so‘m. Modellashtirishning maqsadi - berilgan ushlash kvotasidan foydani bashoratlash. Bunday model uchun imitatsion hisoblash tajribalarini o‘tkazish mumkin va keyin quyidagicha modifikasiyalash mumkin.

Tajriba 1. Berilgan a,c parametrlar uchun k parametrlarni o‘zgartirib, uning o‘lmasdan (qirilib ketmasdan) ko‘payishining eng katta qiymatini aniqlash kerak.

Tajriba 2. Berilgan c,k parametrlar uchun a parametrlarni o‘zgartirib, uning o‘lishi (qirilib ketishi) ko‘payishining eng katta qiymatini aniqlash kerak.

Modifikatsiya 1. Ko‘payishning tabiiy o‘lish (masalan, yemish yetishmasligi) koffisiyenti b ga teng: $x_{i+1} = x_i + ax_i - (k+b)x_i$, $x_0 = c$.

Modifikatsiya 2. k koffisiyentni x ga bog‘liqligini hisobga olamiz (masalan, $k = dx$): $x_{i+1} = x_i + ax_i - dx_i^2$, $x_0 = c$.

Matematik modellashtirish va hisoblash tajribalari to‘g‘risida asosiy tushunchalar. Hozirgi paytda ilmiy - tadqiqotlarning yangi uslubiyati - matematik modellashtirish va hisoblash tajribasiga asos solinmoqda. Bu uslubiyatning mazmuni shundan iboratki, unda joriy Ob’yekt o‘zining matematik modeliga almashtiriladi, hamda matematik modellar zamonaviy hisoblash vositalari yordamida o‘rganiladi. Matematik modellashtirish uslubiyati tez sur’atlar bilan rivojlanib, katta texnik tizimlarni ishlab chiqish va ularni boshqarishdan boshlab, murakkab iqtisodiy va ijtimoiy jarayonlarni tahlil qiluvchi sohalarni ham qamrab olmoqda.

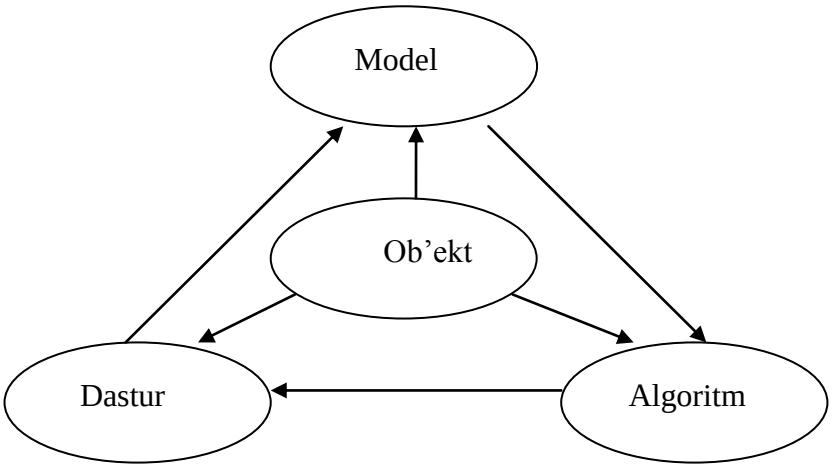
Matematik usullardan keng foydalanish nazariy tadqiqotlarning umumiy darajasini oshirishga va ularni tajribaviy tadqiqotlar bilan chambarchas aloqada olib borishga imkon beradi. Matematik modellashtirishga nazariya hamda tajribaning ko‘plab yutuqlarini o‘zida mujassamlashtirgan anglash, qurish, loyihalashtirishning yangi usuli sifatida qarash mumkin. Ob’yektning o‘zi bilan

emas, uning modeli bilan ishlash uning mavjud holatlardagi xatti-harakatini tez va sarf - harajatlarsiz o‘rganishga imkon beradi. Ayni paytda Ob’yektlarning modellari ustida o‘tkazilgan hisoblash (kompyuter, imitatsiyaviy) tajribalari zamonaviy hisoblash usullarining quvvati va informatikaning texnik vositalariga tayanib, Ob’yektlarni nazariy yondashuvga qaraganda to‘laroq va chuqurroq o‘rganiladi.

Texnik, ekologik, iqtisodiy jihatdan hamda zamonaviy fan tomonidan o‘rganiladigan boshqa tizimlar oddiy nazariy usullar orqali (zaruriy to‘liqlik hamda aniqlikda) o‘rganila olmaydi. Ular ustida olib boriladigan to‘g‘ridan-to‘g‘ri tadqiqot uzoq muddatli, qimmat, ko‘pincha xavfli bo‘ladi. Hisoblash tajribasi tadqiqotni tezroq va arzonroq o‘tkazishga imkon beradi. Matematik modellashtirish ilmiy-texnik taraqqiyotning muhim asoslaridan biridir. Rivojlangan mamlakatlarda bu uslubiyatdan foydalanmasdan birorta yirik masshtabli texnologik, ekologik yoki iqtisodiy loyihani ishlab chiqib bo‘lmaydi.

Matematik modellashtirish uslubiyatining paydo bo‘lishi va takomillashuvi XX asrning 40-yillari oxiri hamda 50-yillarning boshiga to‘g‘ri kelib, bunga ikkita omil sabab bo‘lgan. Kompyuterlarning paydo bo‘lishi birinchi, lekin asosiy bo‘lmagan omil bo‘lib xizmat qildi. Chunki ularning paydo bo‘lishi tadqiqotchilarni hajman katta bo‘lgan hisoblash ishidan ozod etdi. Ikkinchi, muhimroq omil Sobiq Ittifoq va AQShning raketa-yadroviy qobiqni yaratish bo‘yicha milliy dasturlarni bajarish bo‘yicha ijtimoiy buyurtmasi bo‘ldi. Bunday murakkab ilmiy-texnik muammolarni hisoblash vositalaridan foydalanmasdan turib, an’anaviy usullarda hal etib bo‘lmasdi. Yadroviy portlashlar, raketa va sun’iy yo‘ldoshlarning uchirilishi, avvalo, kompyuterlarda loyihalashtirildi, so‘ngra esa amaliyotga tadbiq etildi.

Matematik modellashtirishning asosini *«model-algoritm-dastur»* (1.15-rasm) uchligi tashkil etadi. O‘rganiladigan jarayonlarning matematik modellari murakkab bo‘lib o‘z ichiga chiziqli bo‘lmagan funksional-differensial tenglamalar tizimini qamrab oladi. Matematik model yadrosini xususiy hosilali tenglamalar tashkil etadi.



1.15-rasm. Matematik modellashtirishning intellektual yadrosi

Hisoblash tajribasining birinchi bosqichida Ob’yektning muhim xususiyatlari - uning tarkibiy xususiyatlariga xos bo‘lgan qonunlar matematik ko‘rinishda aks etadi. Matematik model (uning asosiy qismlari) Ob’yekt to‘g‘risida joriy ma’lumotlarni bilish uchun amaliy matematikaning an’anaviy analitik vositalari yordamida o‘rganiladi.

Ikkinchi bosqich modelni kompyuterda ishlab chiqish uchun hisoblash algoritmini tanlash (yoki ishlab chiqish) bilan bog‘liq. Qidirilayotgan kattaliklarni mavjud hisoblash texnikasida berilgan aniqlikda olish lozim. Hisoblash algoritmlari modelning, bevosita Ob’yektning asosiy xususiyatlarini cheklamasligi, yechilayotgan masalalarning va hisoblash vositalarining xususiyatlariga moslashishi kerak. Matematik modellar asosi matematik fizikaning xususiy hosilali tenglamalarining chegaraviy masalalarini yechishning sonli usullaridan tashkil topgan hisoblash matematikasi yordamida o‘rganiladi.

Uchinchi bosqichda model va algoritmni kompyuterda ishlatish uchun dasturiy vosita yaratiladi. Dasturiy mahsulot matematik modellashtirishning matematik modellar qatoridan foydalanish, hisoblashning ko‘p variantliligi bilan bog‘liq muhim xususiyatini nazarda tutishi kerak. Buning natijasida Ob’yektga mo‘ljallangan dasturlash asosida ishlab chiqariladigan amaliy dasturlarning majmui va paketlaridan keng foydalaniladi.

Matematik modellashtirish omili hisoblash tajribasining hamma asosiy qatlamlarini chuqur tahlil etishni ta’minlab beradi. «Model-algoritm-dastur» uchligiga tayanib, tadqiqotchi qo‘liga mukammal moslashuvchan va arzon vositani oladi va u avvaliga nazoratdan o‘tkaziladi. Bundan keyin o‘rganilayotgan Ob’yektning zaruriy sifatli hamda sonli xususiyatlari, tavsiflarini olish uchun matematik modellar keng qamrovda tahlil etiladi.

Hisoblash tajribasi o‘z tabiatiga ko‘ra sohalararo xarakterga ega. Zamonaviy ilmiy-texnik ishlab chiqarishda matematik modellashtirishning sintez ahamiyatini haddan tashqari ortiqcha baholab bo‘lmaydi. Umumiy tadqiqotlarda amaliy sohada, amaliy va hisoblash matematikasi, amaliy va tizimli dasturiy ta’minot bo‘yicha mutaxassislar ishtirok etadi. Hisoblash tajribasi - chiziqli bo‘lmagan matematik modellarni sifatli tahlil etishdan boshlab, to zamonaviy dasturlash tillarigacha bo‘lgan turli xil usul va yondashuvlarga tayanib o‘tkaziladi. Modellashtirish u yoki bu ko‘rinishda ijodiy faoliyatlarining deyarli barchasida ishtirok etadi. Matematik modellashtirish aniq bilimlar doirasini hamda ratsional usullarning ilovalar maydonini kengaytiradi. U asosiy tushunchalar va farazlarni aniq shakllantirish, qo‘llanilayotgan modellarning adekvatligini tahlil etishga, hisoblash algoritmlarining aniqligini nazorat qilishga, hisob ma’lumotlarini sifatli qayta ishlash va tahlil qilishga asoslanadi.

Zamonaviy bosqichda hayotiy ta’minlanganlik muammosini hal etish matematik modellashtirish va hisoblash tajribasidan keng foydalanishga asoslanadi. Hisoblash vositalari (kompyuterlar va sonli usullar) odatda tabiiy fandagi tadqiqotlarda, avvalo fizika hamda mexanikada yaxshi tasvirlangan. Kimyo va biologiyani, tuproq haqidagi fanlarni, ijtimoiy fanlarni faol matematikalashtirish jarayoni olib boriladi. Muhandislik va texnologiyada matematik modellashtirishni qo‘llashning sezilarli yutuqlariga erishildi. Matematik modellarning kompyuter vositasida o‘rganilishi uchiriladigan apparatlarning aerodinamik trubalardagi sinovini, poligonlardagi yadroviy hamda termoyadroviy qurilmalarni portlatish o‘rnini sezilarli darajada bosdi.

Zamonaviy axborot texnologiyalari tibbiyotda ham qo‘llaniladi. Analiz ma’lumotlarini yig‘ish va tahlil etish kasalliklarga o‘z vaqtida tashxis qo‘yish imkoniyatini beradi. Masalan, kompyuterli tomograf katta massivdagi ma’lumotlarni qayta ishlashning matematik usullaridan foydalanish bo‘yicha sifatli tibbiyot vositasini olishga asos bo‘ladi.

Bu yerda aniq bir xususiyatga bog‘liq bo‘lmagan, turli xil fan sohalari uchun umumiy bo‘lgan matematik modellarni qurish va tahlil qilishga qaratilgan asosiy yondashuvlar bayon etilgan. Insonlarni o‘rab turgan olam yagona. Xususan, bu matematik modellarning mukammalligida, turli xil hodisa va Ob’yektlarni ta’riflash uchun qo‘llaniladigan matematik qurilmalarning bir xilligida namoyon bo‘ladi.

Ilmiy-tadqiqotlardagi nazariy va amaliy usulli hisoblash tajribasining umumiy xususiyatlar ko‘rsatib o‘tilgan. Quyida hisoblash tajribasining har xil turlariga qisqacha ta’rif keltirilgan. Hisoblash tajribasi matematik modellarni o‘rganish uchun kompyuterlar va sonli usullardan foydalanish natijasida paydo bo‘lgan. Unga matematik modellashtirishning eng yuqori pog‘onasi sifatida qaraladi.

Matematik modellashtirish. Mazmuni matematik tushunchalarni tabiiy va ijtimoiy fanlarda, texnikada qo‘llashdan iborat bo‘lgan ilmiy bilimlarni matematikalashtirish zamonaviy davr udumi hisoblanadi. Ko‘pincha u yoki bu fanning rivojlanish darajasi ham matematik usullarni qo‘llash darajasi bo‘yicha xarakterlanadi. «Har qanday bilimda matematika qancha bo‘lsa, shuncha fan bor» degan mashhur hikmatli ta’rif bu fikrni ifodalab beradi.

Bilimlarni matematikalashtirish. Fan rivojlanishining empirik bosqichida kuzatilayotgan hodisalar ta’riflanadi, tajribalar o‘tkaziladi, tajriba ma’lumotlari yig‘iladi va guruhlashtiriladi. Nazariy bosqich uchun uning yadrosini tashkil etuvchi asosiy qonunlarni, yangi abstraksiyalar va ideallashtirish tushunchalarini kiritish xos xususiyat. Bunda o‘rganilayotgan Ob’yekt to‘g‘risida umumiy tasavvur hosil qilinadi, tajriba ma’lumotlarining umumiy majmuiga ta’rif beriladi.

Nazariyaning evristik ahamiyati Ob’yekt, hodisa yoki jarayon to‘g‘risidagi yangi, oldin ma’lum bo‘lmagan tavsiflarni aytib bera olishida namoyon bo‘ladi.

Fanning rivojlanish tarixi neptun, pozitronning kashf etilishiga doir yorqin misollarga ega. Matematik g‘oyalar va usullar nafaqat matematik bezaklar vazifasini, balki sonli hamda sifatli tahlilning muhim vositalari bo‘lib xizmat qiladi.

Matematik modellardan foydalanish. Ilmiy bilimlarni matematik xodisaning aniq tabiatidan chetlashish, ideallashtirish va uning matematik shaklini ajratib ko‘rsatish bosqichi mavjud, (matematik model quriladi). Aynan matematik modelning abstraktligi uning aniq hodisa yoki jarayonga nisbatan qo‘llanilishida ma’lum bir qiyinchiliklar tug‘diradi. Hozirda, to‘plangan tajriba tufayli turli fanlardagi ideallashtirish, chetlashish jarayoni nisbatan tinchroq va tezroq o‘tadi.

Matematikalashtirishning ikkinchi bosqichi matematik modellarni abstrakt Ob’yektlar sifatida o‘rganishdir. Ushbu maqsadda matematikaning yaratilgan va maxsus qurilgan vositalari qo‘llaniladi. Hozirgi paytda matematik modellarni o‘rganish uchun hisoblash vositalari - kompyuterlar va sonli usullar katta imkon yaratib beradi.

Matematikani amaliy tadqiqotlarda qo‘llashda uchinchi bosqich interpretasiya-matematik chetlashishlarga aniq bir amaliy mazmun kiritish bilan tavsiflanadi. Amaliy matematik modellashtirish bo‘yicha mutaxassis amaliy sohadagi mutaxassislar bilan yuzma-yuz ishlash paytida matematik chetlashishlar ortida har doim aniq bir amaliy mazmunni ko‘radi.

Matematik modellar sof matematik an’analari bo‘yicha o‘rganilishi mumkin. Bunday holatda matematik modellar amaliy mazmun bilan hech qanday aloqasiz, matematikada qabul qilingan qat’iylik darajasi bo‘yicha o‘rganiladi. Bu esa ularga mukammallik va zaruriy umumiylikni ta’minlaydi. Bu yerda yirik matematiklar - D.Gilbert, A.M.Lyapunov va boshqalarning fikriga yondashish o‘rinli. Mazkur nuqtai nazar quyidagiga olib keladi.

Amaliy muammoni matematik jihatdan sharhlab bo‘lgach sof matematika darajasida ko‘rib chiqish kerak. Matematik modellarni bevosita o‘rganish matematikaning rivojlanishida eng katta turtki hisoblanadi.

Matematik modellashtirishning evristik ahamiyati shunda namoyon boʻladiki, unda naturali tajriba oʻrniga matematik tajriba oʻtkaziladi. Oʻrganilayotgan Ob'yektga u yoki bu ta'sirni oʻrganish oʻrniga matematik model parametrik jihatdan oʻrganiladi. Yechimning u yoki bu parametrga bogʻliqligi aniqlanadi. Bunday tajriba naturaviylikni toʻldirib, hodisa yoki jarayonni chuqurroq oʻrganishga imkoniyat beradi.

Elektron hisoblash mashinalarining paydo boʻlishi, hisoblash matematikasining tez sur'atlar bilan rivojlanishi, hisoblash texnikasining turmushimizda keng qoʻllanilishi matematik modellashtirish imkoniyatlarini sezilarli darajada kengaytirdi.

Matematikaning yangi imkoniyatlari. Kompyuterlar va hisoblash vositalari ilgarilari oʻrganish imkoniyati boʻlmagan masalalarni ma'lum bir aniqlikda va deyarli kam vaqt ichida yechishga, yirik ilmiy-texnik loyihalarni ishlab chiqishga imkon berdi.

Kosmik kemalarni uchirishda va boshqarishda, foydali qazilmalarni seysmik tekshirish natijasida toʻplangan ma'lumotlarni qayta ishlashda kompyuterlardan foydalanish, samolyotning haqiqiy konfiguratsiyasi aerodinamikasini sonli modellashtirish bunga misol boʻla oladi. Sof matematikada isbotlovchi hisoblashlarni bajarish, toʻrtta boʻyoqqa doir mashhur muammoda ham kompyuterlar oʻzining oʻrnini topdi.

Amaliy muammolarni nazariy oʻrgangan holda hisoblash vositalaridan keng foydalanishga asoslangan yangi ilmiy sohalar, yoʻnalishlar tez sur'atlar bilan rivojlanmoqda. Bu borada, avvalo, hisoblash fizikasini, hisoblash gidrodinamikasini, hisoblash geometriyasini, hisoblash algebrasini, hisoblash issiqlik fizikasini qayd etib oʻtamiz.

Matematik modellarni oʻrganish deganda, avvalo, matematik modellarni sifatli oʻrganish hamda aniq yoki taqribiy yechimni olish nazarda tutiladi. Kompyuter nafaqat taqribiy yechimlarni sonli usullarda olishga, balki matematik modellarni sifatli oʻrganishga imkon beradi.

Matematik modellarni o‘rganishning analitik usullari. Sifatli tadqiqot masalani o‘lchamli tahlil etishdan boshlanadi. Masalani o‘lchovsiz ko‘rinishga keltirish uning aniqlovchi o‘zgaruvchilari sonini qisqartirishga imkon beradi. Kichik yoki katta o‘lchovsiz parametrlarni ajratish bir qator holatlarda joriy matematik modellarni sezilarli darajada soddalashtirishga, uni yechishning sonli usullarini ishlab chiqarishda masalaning xususiyatlarini hisobga olishga imkonini yaratadi.

Matematik modelning o‘zi ancha murakkab, chiziqli bo‘lmasligi mumkin. Buning natijasida uni amaliy matematikaning an’anaviy usullari yordamida sifatli o‘rganib bo‘lmaydi. Aynan shuning uchun ko‘p hollarda anchagina sodda, lekin joriy matematik modelga nisbatan mazmunliroq masalada sifatli tadqiqot o‘tkaziladi. Bunday hollarda asosiy modelning soddalashtirilgan masalalari (model uchun model) to‘g‘risida so‘z yuritish lozim.

Matematik modellarni sifatli o‘rganishda korrektlilik muammolariga katta e’tibor qaratiladi. Avvalo, yechimning mavjudlilik masalasi ko‘riladi. Unga mos bo‘lgan qat’iy natijalar (mavjudlilik teoremasi) matematik modelning korrektliligiga kafolat beradi. Bundan tashqari mavjudlilik teoremalarining konstruktivlik isbotlari qo‘yilgan masalani taqribiy yechish usullariga asos qilib olinishi mumkin.

Amaliy matematik modellashtirishda kiruvchi ma’lumotlarning nisbatan kichik chetlanishlarida yechimning turg‘unlik masalasi muhim ahamiyat kasb etadi. Turg‘unmaslik (kichik chetlanishlarda yechimning cheksiz ortib ketishi) teskari masalalar uchun xarakterli bo‘lib, taqribiy yechimni olishda hisobga olinishi kerak.

Yechimning ko‘pligi, yagona emasligi chiziqli bo‘lmagan matematik modellar uchun xos bo‘lishi mumkin. Matematik modellarni sifatli o‘rganishda tarmoqlanish nuqtalari, zaruriy yechimning ajratib ko‘rsatilish masalalari o‘rganiladi.

Tajribaviy ma’lumotlarni qayta ishlash. Amaliyotchi o‘zining tadqiqoti bo‘yicha umumiy sxemasida o‘rganilayotgan Ob’yektga ta’sir o‘tkazadi, ushbu

ta’sir natijalari to‘g‘risida ma’lumot oladi va uni qayta ishlaydi. Bu ma’lumotlar o‘lchovning tasodifiy xatoliklari bilan cheklangan. Shu bois tajribaviy ma’lumotlarni birinchi marta qayta ishlashda asosiy matematik apparat ehtimollar nazariyasi hamda matematik statistikaga asoslanadi. Tajribaviy tadqiqotlar tajriba paytida olingan ma’lumotlarni saqlashga va qayta ishlashga imkon beruvchi o‘lchov-hisoblash komplekslari yordamida o‘tkaziladi.

Har bir amaliy tadqiqotda sinov ma’lumotlari statistik jihatdan qayta ishlanadi. Alohida omillarning ta’sirini sonli baholash tajribaviy ma’lumotlarni u yoki bu aniqlikda interpolyasiyalaydigan emperik bog‘liqlarni qurishda bilinadi. Bunday holda mazmunli matematik modellar umuman bo‘lmagan aproksimasiyali matematik modellardan foydalanish to‘g‘risida gapirish mumkin. U yoki bu masalani yechish uchun o‘tkaziladigan tajribalar soni va sharti tajribani rejalashtirish bosqichida tanlanadi. Bu yerda muqobil tajriba matematik nazariyasi, tajribani rejalashtirish nazariyasining natijalari jalb qilinadi.

Uskunaning matematik modeli. Tajribaviy tadqiqotlarning zamonaviy rivojlanish bosqichi mukammal uskunalarning keng qamrovda qo‘llanilishi bilan izohlanadi. Uskunalarning o‘zi o‘rganilayotgan hodisa yoki jarayonga chetlanishlar kiritadi. Bunday xatoliklardan qutulish uchun uskunaning matematik modeli quriladi.

Tajribalarni o‘tkazish paytida ikkita mutlaqo turli holatni nazarda tutish kerak. Ulardan birinchisi o‘rganilayotgan hodisa yoki ob’yekt uchun nazariy ta’rif, matematik model yo‘q bo‘lib, keyinchalik matematik ta’rif berish maqsadida tajribaviy materialni to‘plash masalasining qo‘yilishi bilan bog‘liq. Bu holda matematik usullar ma’lumotlarni saqlash va qayta ishlash, xususan emperik bog‘liqliklarni o‘rnatish uchun qo‘llaniladi.

Aproksimasiyali matematik modellarni qurishda emperik formulalarning parametrlarini aniqlash, formulaning o‘zini moslashtirish holati tabiiydir. Tajribaviy ma’lumotlar to‘plamidan aproksimasiyali modellarning parametrlarini shunday tanlash kerakki, natijada tajribaviy ma’lumotlar katta aniqlikda

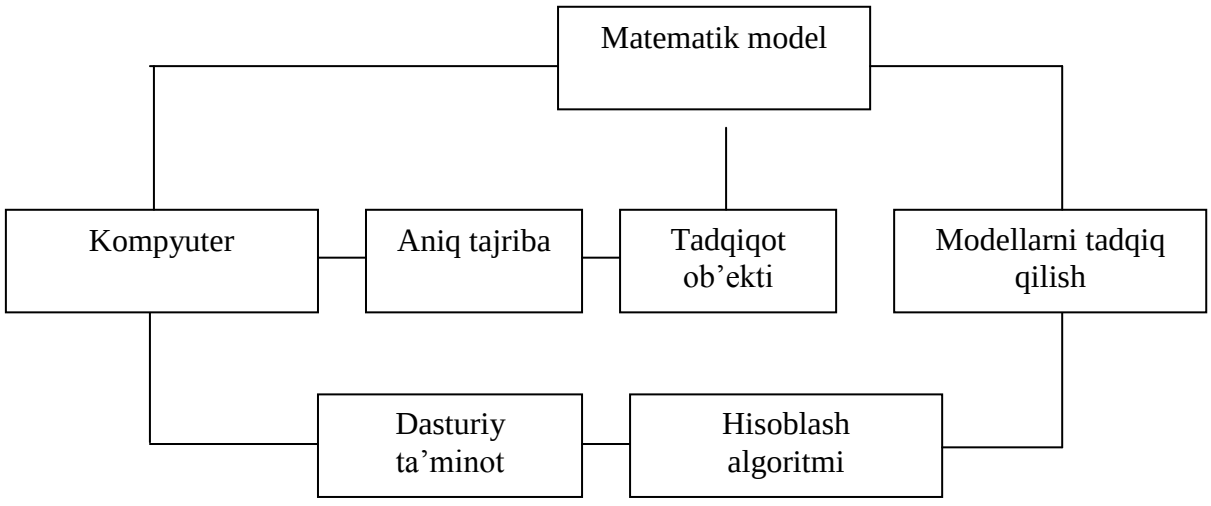
ta’riflanishi mumkin bo‘lsin. Bunda biz minimallashtirish masalalarini taqribiy yechish zaruriyatiga duch kelamiz.

Tajribalarning ikkinchi sinfi o‘rganilayotgan ob’yektning nazariy ta’rifi berilgan sharoitda o‘tkaziladi. Matematik model tarkibining aniq va modelning parametrlarini aniqlash masalasi qo‘yiladi. Naturali tajribaning o‘zi Ob’yektning u yoki bu xususiyatini aniqlashga, Ob’yektning matematik modeliga aniqlik kiritishga qaratilgan.

Bunday tadqiqotlarning tajribaviy ma’lumotlarini qayta ishlashda ko‘pincha teskari masalalar bilan ish tutishga to‘g‘ri keladi. Bunday masalalar klassik nuqtai nazardan to‘liq bo‘lmasligi, shuning uchun sonli tadqiqotni o‘tkazish uchun qiyin bo‘lishi mumkin. Tajribaviy tadqiqotlarning ma’lumotlarini qayta ishlash va interpolyasiyalash bosqichida matematik modellarning turli xil sinflarini o‘zida mujassamlashtirgan hisoblash vositalari keng qamrovda qo‘llanilmoqda.

Hisoblash tajribasi. Nazariy va tajribaviy tadqiqotlarning avtonomik darajasi yuqori. Fundamental modellar aniq, tekshirilgan sharoitda nazariy hamda tajribaviy tadqiqotlarning mustahkam koordinatsiyalashuvi va aloqasiga oid masalasi qo‘yilishi mumkin.

Gap ilmiy tadqiqotlarning birlashtiruvchi yangi texnologiyasi bo‘lgan matematik modellashtirish hamda hisoblash tajribasi to‘g‘risida boradi (1.16-rasm).



1.16-rasm. Hisoblash tajribasining umumiy shemasi

Hisoblash tajribasining asosiy bosqichlari. Avvalo, hisoblash tajribasining umumiy sxemasini bayon etamiz, so‘ngra uning asosiy bosqichlariga qisqacha ta’rif keltirib o‘tamiz. Hisoblash tajribasini tor ma’noda, o‘rganilayotgan ob’yektning matematik modellarini hisoblash vositalari yordamida yaratish va o‘rganish deb tushunib, unga asos qilib «*model-algoritm-dastur*» uchligini ajratib ko‘rsatish mumkin.

Hisoblash tajribasining keng metodologik mazmuni sifatida ilmiy tadqiqotlarning yangi texnologiyasi tushuniladi.

O‘rganilayotgan ob’yekt uchun, avvalo, matematik model quriladi. U ma’lum fundamental modellarga asoslanadi. Hisoblash tajribasi o‘z mazmuniga ko‘ra bir-biriga yaqin modellar guruhi o‘rganilishini nazarda tutadi. Avvalo, o‘rganilayotgan jarayonlar ta’rifi, tajribaviy ma’lumotlarga yaqinlik nuqtai nazariga ko‘ra anchagina mazmunli, hamda to‘liq, lekin sodda model quriladi.

Hisoblash tajribasining keyingi sikllarida modelga aniqlik kiritiladi, yangi omillar hisobga olinadi va h.k. Shuning uchun biz har doim haqiqatni u yoki bu aniqlikda ta’riflovchi matematik modellarning tartiblangan majmui to‘g‘risida gapirishimiz mumkin. Nisbatan sodda model doirasida ham tajriba bilan hamohanglikni saqlash kerak. Aynan shu, oxir oqibatda, hisoblash tajribasining maqsadi bo‘ladi.

Matematik model amaliy matematikaning an’anaviy usullarida qurib bo‘lganidan so‘ng matematik model oraliq tadqiqotdan o‘tkaziladi. Hisoblash tajribasining mohiyati kompyuterda matematik modellarni sonli usullarda o‘rganishdan iborat. Bu yerda faqatgina matematik modelni oraliq sinovdan o‘tkazish to‘g‘risida gap boradi. Bu bosqichda mavjud aniqlikda, matematikada qabul qilingan qat’iylik darajasida tor matematik mazmunli to‘liq masalaning korrektliligi hal etiladi.

Matematik modellar va ularning sinflari . “Model” lotincha modulus so‘zidan olingan bo‘lib, biror ob’yekt yoki ob’yektlar tizimining obrazi yoki namunasi sanaladi. Modellashtirish – bu, biror A ob’yektni boshqa B ob’yekt bilan almashtirishdir. Bu yerda A ob’yekti original yoki modellashtirish ob’yekti, uni

almashtiruvchi B esa model deyiladi. Boshqacha aytganda, model – bu original ob’yektning unga almashtirilgan shunday ob’yektidirki, u original ob’yektning ayrim xossalarini o‘rganishni ta’minlab beradi. Masalan, Yerning modeli –globus, osmon va undagi yulduzlar modeli – sayyoralar ekrani, pasportdagi suratni shu pasport egasining modeli deb qarash mumkin.

Modellashtirishdan maqsad tashqi muhit va o‘zaro aloqada bo‘lgan ob’yektlar haqidagi ma’lumotlarni olish, ishlatish, tasvirlash va qayta ishlashdir. Bu o‘rinda model esa ob’yekt holati, xossalari va qonuniyatlarini o‘rganish uchun vosita sifatida ishtirok etadi.

Modellashtirish inson faoliyatining turli sohalarida keng ishlatiladi. Asosan olingan ma’lumotlar bo‘yicha samarador yechimlar qabul qilish jarayonida, loyihalash va boshqarish sohalarida modellashtirish ko‘p qo‘llaniladi.

Model har doim ma’lum bir maqsadda quriladi, masalan, uning qaysi bir xossasi ob’yektiv jarayonga ta’sir etishi muhim rol o‘ynaydi yoki qaysilari muhim rol o‘ynamaydi.

Modellashtirish nazariyasi ob’yekt moduli asosida original ob’yekt xossalarini o‘rganadigan usullarni tekshiruvchi ilmiy yo‘nalishning bir bo‘limi hisoblanadi. Modellashtirish nazariyasi asosida o‘xshashlik nazariyasi yotadi.

Barcha modellarni ikki sinfga ajratish mumkin:

1. Moddiy
2. Ideal

O‘z navbatida moddiy modellarni quyidagilarga ajratish mumkin:

1. Tabiiy
2. Fizik
3. Matematik

Ideal modellarni quyidagilarga ajratish mumkin:

1. Ko‘rgazmali
2. Belgili
3. Matematik.

Moddiy tabiiy model – bu real ob’yekt, jarayon va tizimlar bo‘lib, ular ustida turli ilmiy, texnik va ishlab chiqarish tajribalari o‘tkaziladi.

Moddiy fizik model– bu, maketlar, mulyajlar, original fizik xossa nusxasi (masalan, kinematik, dinamik, gidravlik, issiqlik, elektrik, yorug‘lik modellari).

Moddiy matematik model – bu analogik, tuzilmali, geometrik, grafik, raqamli va kibernetik modellardir.

Ideal aniq model – bu sxemalar, kartalar, chizmalar, grafiklar, graflar, analog, tuzilmali va geometrik modellardir.

Ideal belgili modellar - bu simvollar, alfavit, dasturlash tili, tartiblangan yozuv, topologik yozuv, tarmoqli tasvirlashlardir.

Ideal matematik modellar – bu tahliliy, funksional, imitatsion, kombinatsiyalashgan modellardir.

Matematik modellashtirish. Matematik modellarni qurishning shakl va tamoyillari. Matematik modellashtirish barcha modellar ichida eng universal modellashtirish sifatida fizik modelni tuzmasdan turib, ob’yekt holati to‘g‘risidagi ma’lumotlarni olish imkonini beradi va u eng samarali va qimmatlidir. *Matematik modellashtirish* - real Ob’yektni, jarayonni yoki tizimni matematik modelga almashtirish yo‘li bilan o‘rganuvchi, hamda kompyuter yordamida tajriba tadqiqotlarini o‘tkazish uchun mo‘ljallangan eng qulay vositadir

Matematik model - real ob’yekt, jarayon yoki tizimning matematik terminlarda ifodalangan va uning mavjud belgilarini ifodalovchi, unga taqriban yaqin bo‘lgan nusxasidir. Modelning taqribiylik xarakteri turli ko‘rinishda namoyon bo‘lishi mumkin. Masalan, tajriba o‘tkazish mobaynida foydalanadigan asboblarning aniqligi olinayotgan natijaning aniqligiga ta’sir etadi. *Matematik model* mantiqiy matematik konstruksiyalar yordamida sanoqli formada ob’yekt, jarayon yoki tizimning asosiy xossalarini ifodalaydi va bundan tashqari, uning parametrlarini, ichki va tashqi aloqalarini ham tasvirlaydi.

Umuman olganda, real Ob’yekt, jarayon yoki tizimni funksional tizim ko‘rinishida tasvirlaydi. Masalan:

$$F_i\,(X,Y,Z,t)=0.$$

Bu yerda:

X - vektor bo‘lib kiritiladigan o‘zgaruvchilardir,

$X=[x_1,x_2,x_3, \dots , x_N]^t$,

Y - vektor bo‘lib chiqariladigan o‘zgaruvchilardir,

$Y=[y_1,y_2,y_3, \dots , y_N]^t$,

Z - vektor bo‘lib tashqi aloqani bildiradi,

$Z=[z_1,z_2,z_3, \dots , z_N]^t$,

t – vaqt koordinatasi.

Matematik model qurishda tadqiqot o‘tkazishdan oldin oxirgi olingan natijaga uncha ta’sir qilmaydigan faktorlarni aniqlash va qarashdan o‘chirish masalasi kelib chiqadi. Matematik modelga real modeldagiga qaraganda ancha kam sondagi faktorlar kiritiladi. Tajriba ma’lumotlari asosida oxirgi natijada ifodalanadigan kattaliklar bilan matematik modelga kiritilgan faktorlar orasidagi bog‘lanish (aloqa) haqida gipotezalar aniqlanadi. Bunday bog‘lanish ko‘proq xususiy hosilali differensial tenglamalar bilan ifodalanadi. Masalan, qattiq jismlar, suyuqlik va gaz mexanikasi masalalarida, filtrlash nazariyasida, issiqlik tarqalish, elektrostatik va elektrodinamik maydonlar nazariyasida buni ko‘rish mumkin.

Matematik modelni tasvirlash forma va tamoyillari ko‘p faktorlarga bog‘liq bo‘ladi.

Matematik modelni qurish tamoyili quyidagilarga bo‘linadi:

1. Analitik;

2. Imitasion.

Analitik modelda real ob’yekt, jarayon yoki tizim funksiya qilish jarayoni aniq funksional bog‘lanishlar ko‘rinishida yoziladi.

Analitik model matematik muammoga bog‘liq ravishda quyidagi turlarga bo‘linadi:

1. Tenglama (algebraik, transsendent, differensial, integral),

2. Approksimatsiya masalalari (interpolyatsiya, ekstrapolyatsiya, sonli integrallash va differensiallash),

3. Optimizatsiya masalalari,

4. Stoxastik muammolar.

Ob’yektni modellashtirishda uning murakkabligi ko‘p hollarda analitik modelni qurishda qiyin muammolarni keltirib chiqaradi. Bunday hollarda tadqiqotchi imitatsion modellashtirishni ishlatishga majbur bo‘ladi.

Funksiya qaralayotgan ob’yekt, jarayon yoki tizim imitatsion modellashtirishda algoritmlar to‘plami ko‘rinishida tasvirlanadi. Algoritmlar real elementar hodisani imititatsiya (taqlid) qiladi. Imitatsion modellashtirish boshlang‘ich ma’lumotlar asosida ma’lum bir vaqt oralig‘ida jarayon yoki tizim holati haqida ma’lumotni olish imkonini beradi. Lekin ob’yekt, jarayon yoki tizim holati haqida bashorat qilish qiyin. Aytish mumkinki, imitatsion model – bu matematik modellar asosida real ob’yekt, jarayon yoki tizim holatini kompyuterda hisoblash tajribalarini o‘tkazish yo‘li bilan imitatsiya qilishdir.

Real ob’yekt, jarayon yoki tizim xarakterini o‘rganish bilan bog‘liq ravishda matematik model quyidagicha bo‘lishi mumkin:

1. Determinlashgan,

2. Stoxastik.

Determinlashgan modelda turli tasodifiy ta’sirlar yo‘q, model elementlari (o‘zgaruvchilar, matematik bog‘lanishlar) yetarlicha aniq qo‘yilgan va tizim holatini aniq ifodalash mumkin deb faraz qilinadi. Determinlashgan model ko‘proq algebraik tenglamalar, integral tenglamalar va matritsalar algebrasi ko‘rinishida ifodalanadi.

Stoxastik model o‘rganilayotgan ob’yekt va tizimda tasodifiy xarakterdagi jarayonlarni hisobga oladi va ularda ehtimollar nazariyasi va matematik statistika usullaridan foydalanadi.

Kiritiladigan ma’lumotlar turiga qarab modellar quyidagilarga bo‘linadi:

1. Uzluksiz;

2. Diskret.

Agar ma’lumotlar va parametrlar uzluksiz bo‘lib, matematik aloqalar (bog‘lanishlar) turg‘un bo‘lsa, u holda matematik model *uzluksiz* bo‘ladi.

Aksincha, agar ma’lumotlar va parametrlar diskret bo‘lib, matematik aloqalar (bog‘lanishlar) turg‘un bo‘lmasa, u holda matematik model *diskret* bo‘ladi.

Vaqtni hisobga olgan holda model holati bo‘yicha modellar quyidagilarga bo‘linadi:

1. Statistik,

2. Dinamik.

Statistik modellar ma’lum bir vaqt ichida ob’yekt, jarayon yoki tizim holatini ifodalaydi. Dinamik modellar ob’yekt, jarayon yoki tizim holatini vaqt bo‘yicha aks ettiradi.

Matematik model va real ob’yekt, jarayon yoki tizim matematik modeli orasidagi mos daraja bo‘yicha quyidagilarga bo‘linadi:

1. *Izomorf* (forma bo‘yicha bir xillik),

2. *Gomomorf* (forma bo‘yicha har xillik).

Agar u bilan real ob’yekt, jarayon yoki tizim orasida elementlari bo‘yicha to‘liq moslik bo‘lsa, *model izomorf deyiladi*. Agar model va ob’yektning juda ahamiyatli mos qismlari orasida moslik mavjud bo‘lsa *gomomorf deyiladi*.

Amaliy masalalarni yechishda kompyuterni qo‘llash uchun oldin amaliy masalani formal matematik tilga o‘tkazish lozim, ya’ni real ob’yekt, jarayon yoki tizim uchun matematik model tuzilgan bo‘lishi kerak. Matematik model sonli formada mantiqiy-matematik konstruksiya yordamida ob’yekt, jarayon yoki tizimning asosiy xossalarini, uning parametrlarini, ichki va tashqi aloqalarini tasvirlaydi.

Matematik model qurish uchun quyidagilar kerak:

1. Real ob’yekt yoki jarayonni chuqur tahlil qilish kerak;

2. Mavjud eng asosiy xossa va xususiyatlarni ajratish kerak;

3. O‘zgaruvchilarni aniqlash kerak, ya’ni ob’yekt xossalariga hamda uning asosiy xususiyatlariga ta’sir qiluvchi parametrlar va ularning qiymatlari aniqlanishi kerak;

4. Ob’yekt, jarayon yoki tizimning asosiy xossalari o‘zgaruvchilar qiymatlariga bog‘liqligi mantiqiy-matematik munosabat (aloqa) yordamida aniqlanishi kerak (tenglama, tenglik, tengsizlik, mantiqiy-matematik konstruksiya);

5. Ob’yekt, jarayon yoki tizim ichki bog‘lanishlarini cheklanishlar, tenglama, tenglik, tengsizlik va mantiqiy-matematik konstruksiyalar yordamida ajratish zarur;

6. Tashqi aloqalarni aniqlash va ularni cheklanishlar, tenglama, tenglik, tengsizlik va mantiqiy-matematik konstruksiyalar yordamida tavsiflash zarur.

Matematik modellashtirish, ob’yekt, jarayon yoki tizimni tadqiqot qilishdan va uning matematik tavsifini tuzishdan tashqari yana quyidagilarni ham o‘z ichiga oladi:

1. Ob’yekt, jarayon yoki tizim holatini modellashtiruvchi algoritm tuzish;

2. Hisoblash va tabiiy tajribalar o‘tkazish yordamida model, ob’yekt, jarayon yoki tizimning mosligini tekshirish;

3. Modelni to‘g‘rilash;

4. Modelni ishlatish.

O‘rganilayotgan ob’yekt, jarayon yoki tizimni matematik ifodalash quyidagilarga bog‘liq:

1. Tabiatdagi barcha real jarayon yoki tizimlar fizika, kimyo, mexanika, termodinamika, gidrodinamika, elektrotexnika, elastiklik va elastiklik nazariyasi qonunlariga asoslangan holda tuziladi.

2. Real jarayon yoki tizimni talab qilingan ishonchli va aniqlikda o‘rganish va tadqiqot qilish.

Matematik modelni tanlash bosqichida quyidagilar aniqlanadi: ob’yekt, jarayon yoki tizim chiziqli va chiziqsiz, dinamik yoki statik, stasionar yoki nostasionar. Bulardan tashqari, o‘rganilayotgan ob’yekt yoki jarayonning diterminallash darajasi ham aniqlanadi.

Matematik model hech vaqt qaralayotgan ob’yekt, jarayon yoki tizimni aynan bir xil ifodalamaydi. U soddalashtirishga asoslangan holda ob’yektni taqribiy ifodalaydi. Shu sababli natija, tahlil qilish natijasida olingan model taqribiy

xarakterga olib keladi. Uning aniqligi ob’yekt va modelning moslik darajasi bilan baholanadi.

Matematik modelni qurish qaralayotgan ob’yekt, jarayon yoki tizimning oddiy va ancha qo‘pol bo‘lgan matematik modelini qurish va tahlil qilish bilan boshlanadi. Keyinchalik, kerak bo‘lgan holda, model aniqlashtiriladi.

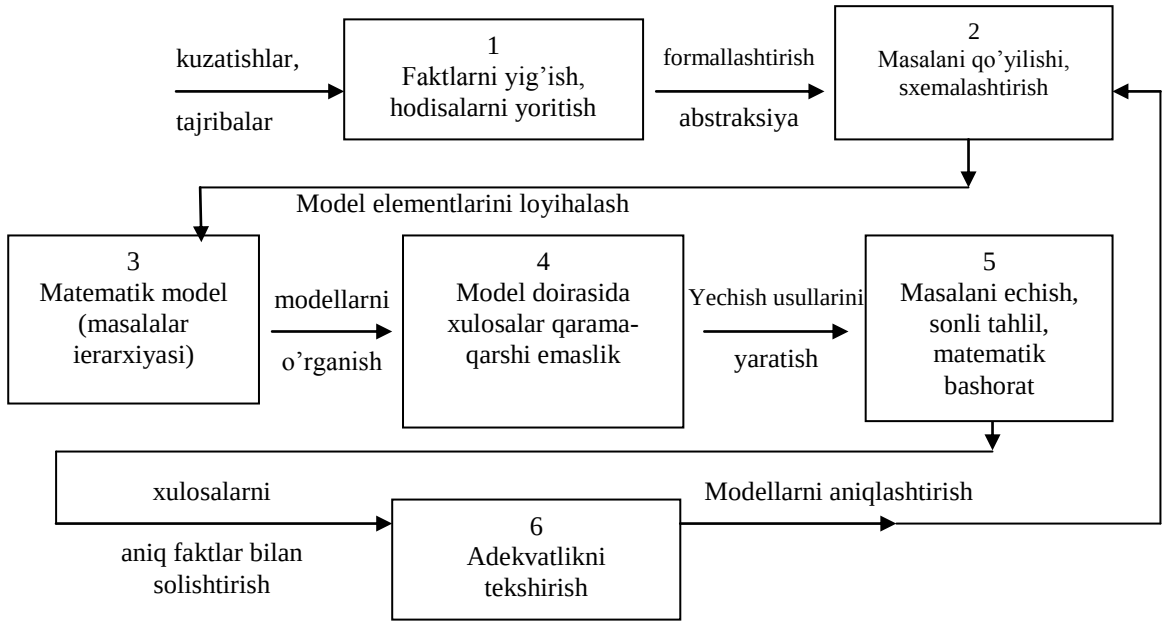
Matematik modellashda berilgan fizik jarayonlarning matematik ifodalari modellashtiriladi. Matematik model tashqi dunyoning matematik belgilar bilan ifodalangan qandaydir hodisalari sinfining taqribiy tavsifidir. Matematik model tashqi dunyoni bilish, shuningdek, oldindan aytib berish va boshqarishning kuchli uslubi hisoblanadi.

Quyidagi sxemada matematik model tuzish bosqichlari keltirilgan (1.17-rasm).

Hodisa va jarayonlarni matematik model yordamida o‘rganish quyidagi ketma-ketlikda amalga oshiriladi.

Birinchi – modelning asosiy ob’yektlarining bog‘lovchi qonuniyatlarini ifodalash.

Ikkinchi – modeldagi matematik masalalarni tekshirish.

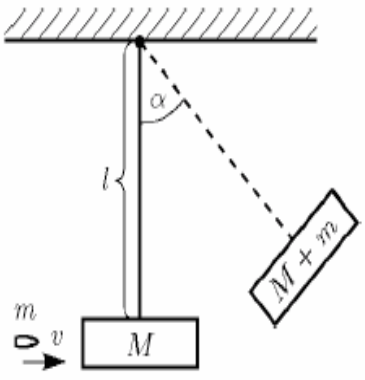


1.17- rasm. Matematik model tuzish bosqichlari

Matematik modellarni yaratishning matematik usullari. Tabiatning fundamental qonunlari, variatsion tamoyillar, o‘xshashliklar, iyerarxik zanjirlarni qo‘llashga asoslangan eng sodda matematik modellarni qurishning yondashuvlarini ko‘rib chiqaylik. Soddalikka qaramay, jalb etilayotgan material modellarning mosligi, ularning «ta’minlanganligi», nochiziqligi, sonli amallashtirish va bir qator boshqa materiallarni muhokama qilishga imkon beradi. Biz quyida elementar matematik modellarni qurishni ko‘rib chiqamiz.

Elementar matematik modellar. Tabiatning fundamental qonunlari. Modellarni qurishning keng tarqalgan usuli tabiatning fundamental qonunlarini aniq bir vaziyatda qo‘llashdan iborat. Bu qonunlar hamma tomondan tan olingan, tajriba asosida ko‘p marotaba tasdiqlangan, ilmiy-texnik yutuqlarning asosi bo‘lib xizmat qiladi. Shu bois ularning asoslanganligi shubha tug‘dirmaydi, bu esa izlanuvchiga kuchli ruhiy quvvat bag‘ishlaydi. Birinchi qatorda ma’lum bir vaziyatda qanday qonun (qonunlar) qo‘llash kerak degan savol tug‘iladi.

a) Energiyaning saqlanishi. Bu qonun qariyb 200 yildan beri ma’lum bo‘lib, tabiatning buyuk qonunlari orasida faxriy o‘rinni egallaydi. Unga asoslanib, o‘qning tezligini tezkor usulda aniqlamoqchi bo‘lgan ballistika bo‘yicha ekspert oldida laboratoriyasi bo‘lmasa ham, mayatnikka o‘xshash nisbatan oson qurilma - yengil va erkin aylanuvchi sterjenga osilgan yukdan foydalanishi mumkin. (1.18-rasm). Yukda tiqilib qolgan o‘q «o‘q-yuk» tizimiga o‘zining kinetik energiyasi haqida ma’lumot beradi, bu tizim esa o‘z navbatida sterjen vertikaliga nisbatan eng ko‘p darajada og‘ganda to‘laligicha potensial energiyaga o‘tadi. Bu transformasiyalar quyidagi tengliklar zanjiri ko‘rinishida yoziladi:



1.18-rasm

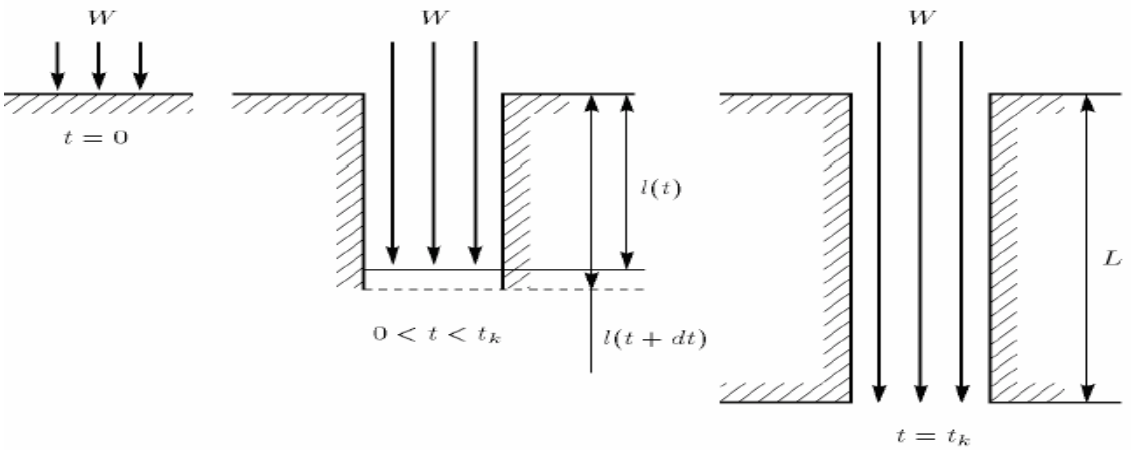
$$\frac{mv^2}{2} = (M+m)\frac{V^2}{2} = (M+m)gl(1-\cos\alpha).$$

Bu yerda $mv^2/2$ – v tezlikka ega bo‘lgan m massa yo‘lining kinetik energiyasi, M – yukning massasi, V – «o‘q-yuk» sitemasining to‘qnashishdan keyingi tezligi, g-erkin tushish tezlanishi, l-sterjen uzunligi, α - eng kata og‘ish burchagi. Qidirilayotgan tezlik quyidagi formula bilan topiladi:

$$v = \sqrt{\frac{1(M+m)gl(1-\cos\alpha)}{m}}. \qquad (1.1)$$

Agar energiyaning o‘q hamda yukni qizdirishga, havoning qarshiligini yengishga sarf bo‘lishini hisobga olmasak, tezlikning qiymati aniqroq bo‘ladi. Bir qarashda bu to‘g‘ri mulohaza, lekin aslini olganda unday emas. O‘q bilan mayatnikning «yopishishi» paytida sodir bo‘ladigan jarayonlar faqatgina mexanik emas. Shuning uchun v kattalikni hisoblashda qo‘llanilgan mexanik energiyaning saqlanish qonuni unchalik to‘g‘ri emas: tizimning mexanik emas, to‘la energiyasi saqlanadi. Mexanik energiya o‘z yo‘lidagi tezlikni baholashda quyi chegarani yaratadi (bunday sodda masalani to‘g‘ri yechish uchun impulsning saqlanish qonunini ham hisobga olish kerak 1-misolga qarang).

Shunga o‘xshash mulohazadan muhandis nurlanishi material yuzasiga perpendikulyar yo‘nalgan W quvvatli lazer bilan L qalinlikdagi metall qatlamini o‘yishning t_k vaqtini baholashda foydalanishi mumkin (1). Agar lazerning energiyasi LS_ρ (S-nurlanayotgan yuza, LS – ustun hajmi, ρ – modda zichligi) to‘laligicha ustunni o‘yish uchun sarf bo‘lsa, u holda energiyaning saqlanish qonuni quyidagi tenglik bilan ifodalanadi:



1.19-rasm. Metallni lazer bilan o‘yishning boshlang‘ich, oraliq va yakuniy bosqichlari

$$E_0 = Wt_k = hLS\rho, \qquad (1.2)$$

Bu o‘rinda h – birlik massaning bug‘lanishi uchun kerak bo‘ladigan energiya. h kattalik tarkibiy qismdan iborat: $h = (T_{n.n} - T)h_1 + h_2 + h_3$, formulaning mazmuni quyidagicha: materialni avvalo erish harorati T_{pl} gacha olib kelib, so‘ngra suyultirib, bug‘ga aylantirish (T – boshlang‘ich harorat, h_1 – solishtirma issiqlik sig‘imi, h_2 va h_3 – mos ravishda solishtirma erish va bug‘lanish issiqligi) kerak.

O‘yma chuqurligining *l*(t) vaqt o‘tishi bilan o‘zgarishi energiyaning *t* dan *t*+*dt* gacha bo‘lgan vaqt oralig‘idagi balansidan $[l(t+dt) - l(t)]S\rho = dl\ \mathrm{S}\rho$ topiladi.

Bu vaqt ichida bug‘langan massaga W dt ga teng bo‘lgan $dl\ \mathrm{hS}\rho$ energiya sarf bo‘ladi $dl\ \mathrm{hS}\rho = W\ \mathrm{dt}$, bu energiya haqidagi ma’lumot esa lazer orqali moddaga yetkaziladi.

Bu yerdan quyidagi differensial tenglama kelib chiqadi:

$$\frac{dl}{dt} = \frac{W}{hS\rho}.$$

Uni integrallash natijasida (o‘ymaning boshlang‘ich chuqurligi nolga tengligini hisobga olgan holda):

$$l(t) = \frac{W}{hS\rho}t = \frac{E(t)}{hS\rho} \qquad (1.3)$$

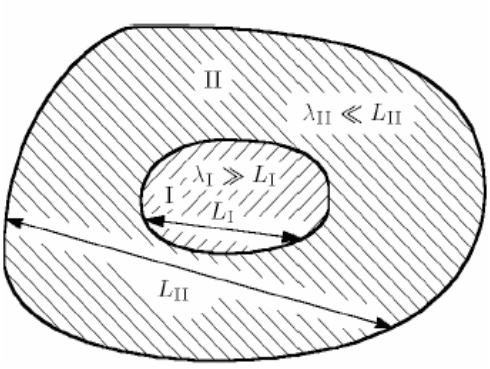
formula hosil qilinadi, bu o‘rinda E(t) – lazer tomonidan t vaqtgacha ajralgan butun energiya. Demak, o‘ymaning chuqurligi sarf bo‘lgan energiyaning qiymatiga proporsional (jumladan, $l(t_k) = L$ dagi t_k ning qiymati (1.2) formula bo‘yicha hisoblangan qiymat bilan ustma-ust tushadi).

Hayotda esa o‘yish jarayoni ko‘rib chiqilgan sxemadan ancha murakkabroq energiya moddani qizdirishga, noto‘g‘ri shaklga ega bo‘lgan o‘ymadan bug‘larni olib tashlashga sarf bo‘ladi. Shuning uchun taklif etilgan matematik ta’rifning to‘g‘riligiga bo‘lgan ishonch o‘q bilan bog‘liq holdagidan kamroqdir. Ob’yekt bilan modelning mos kelishi - matematik modellashtirishning markaziy masalasi bo‘lganligi uchun, unga keyinchalik ko‘p marta murojaat etamiz.

b) Materiyaning saqlanishi. Maktab o‘quvchisi ikkita quvurdan oquvchi hamda kiruvchi suv bilan basseynni to‘ldirish haqidagi masalani yechayotganida

aynan shu mulohazadan foydalanadi. Albatta, bu qonunning qo‘llanilish doirasi ancha keng.

Masalan, bizda «oddiy» material (qo‘rg‘oshin) ning qalin qatlami bilan o‘ralgan kam miqdorda radioaktiv modda (uran) berilgan bo‘lsin - bunday vaziyat bo‘linadigan materiallarni saqlashda, yoki energetika ishlarida (1.20-rasm) ko‘p uchraydi. «Kam miqdor» degan ibora ostida soddalashtirilgan hol, aniq qilib aytganda, yemirilishning hamma mahsulotlari atomlar bilan to‘qnashmay, I sohani hech qanday qiyinchiliklarsiz tark etadi degan mazmun yashirilgan. Boshqa so‘z bilan aytganda, yemirilishdagi moddalarning erkin yugurish uzunligi λ_1 birinchi moddada materialning xususiy o‘lchamlari L_1 dan ham kattaroq, ya’ni $\lambda_1 >> L_1$. «Qalin qatlam» iborasi xafvsizlik maqsadida ajraluvchi mahsulotlar II sohada butunlay yutilib ketilishini anglatadi. Bu esa $\lambda_{II} << L_{II}$ teskari shart bajarilganida amalga oshadi, bu yerda λ_{II} -ikkinchi moddadagi yemirilish mahsulotlarining erkin yugurish yo‘li, L_{II} – uning xususiy o‘lchami.



1.20-rasm

Shunday qilib, bir sohadan uchib chiquvchi hamma narsa, II sohada yutiladi va ikkala moddaning umumiy massasi vaqt o‘tishi bilan ham o‘zgarmaydi. Bu esa berilgan vaziyat uchun qo‘llanilgan materiyaning saqlanish qonunidir. Agar boshlang‘ich vaqt momenti t=0 da $M_I(0)$ va $M_{II}(0)$ moddalarning massalari teng bo‘lsa, vaqtning ixtiyoriy momentida quyidagi balans o‘rinli:

$$M_I(0)+ M_{II}(0)= M_I(t)+ M_{II}(t). \qquad (1.4)$$

Bitta tenglamaning o‘zi ikkita massa $M_I(t)$ va $M_{II}(t)$ larning joriy qiymatlarini aniqlash uchun yetarli emas. Matematik mulohazani tugallash uchun yemirilish bilan bog‘liq qo‘shimcha ta’rifni kiritish kerak. Unga ko‘ra, yemirilish tezligi (birlik vaqtda yemirilgan atomlar soni) radioaktiv moddadagi atomlarning umumiy soniga proporsionaldir. Kichik vaqt dt ichida t va t+dt momentlar orasida jami:

$$N_I(t+dt) - N_I(t) = -\alpha N_I(t+\xi dt), \;\; \alpha > 0, \;\; 0 < \xi < 1,$$

atomlar yemiriladi. Bu yerda moddaning saqlanish qonuni ikkinchi marotaba qo‘llanilgan bo‘lib, u butun jarayonga emas, faqatgina dt vaqt oralig‘iga tegishli. Atomlarning balansini ta’riflovchi tenglamaning o‘ng tomonida minus ishorasi (modda kamayib ketmoqda) turibdi, $N_I(t+\xi dt)$ qiymat esa ko‘rib chiqilayotgan vaqt ichidagi atomlar sonining o‘rta qiymatiga mos keladi. Uni differensial ko‘rinishda yozib olamiz:

$$\frac{dN_I(t)}{dt} = -\alpha N_I(t).$$

$M_I(t)=\mu_1 N_I(t)$ ligini hisobga olsak (bu yerda μ_1 – I moddaning atom massasi) quyidagi tenglikka ega bo‘lamiz:

$$\frac{dM_I(t)}{dt} = -\alpha M_I(t). \qquad (1.5)$$

Erkli radioaktivlikka ega bo‘lgan ixtiyoriy atom atrofdagi moddaning holatiga bog‘liq bo‘lmagan yemirilish ehtimoliga ega. Shu bois eng radioaktiv modda qanchalik ko‘p (kam) bo‘lsa, birlik vaqt ichida yemirilish mahsuloti shuncha ko‘p (kam) ajraladi. Proporsionallik koeffisiyenti $\alpha > 0$ (yemirilish doimiysi) aniq bir modda orqali aniqlanadi. (1.4), (1.5) tenglamalar $\lambda_I >> L_I$, $\lambda_{II} << L_{II}$ shartlar hamda α, $M_I(t)$, $M_{II}(t)$ kattaliklar bilan birgalikda ko‘rib chiqilayotgan ob’yektning matematik modelini tashkil etadi. (1.5) integrallab, shuni guvohi bo‘lamizki, bo‘linayotgan materialning massasi eksponent qonun bo‘yicha kamayadi:

$$M_I(t) = M_I(0)e^{-\alpha t},$$

$t \to \infty$ da I sohadagi modda butunlay yo‘q bo‘lib ketadi.

Umumiy massa (1.4) ga ko‘ra o‘zgarmas bo‘lib qolishi tufayli, II sohadagi modda miqdori ortadi:

$$M_{II}(t) = M_{II}(0) + M_I(0) - M_I(0)e^{-\alpha t} = M_{II}(0) + M_I(0)(1-e^{-\alpha t}),$$

va $t \to \infty$ da yemirilish mahsulotlari I sohadan butunlay II sohaga o‘tib ketadi.

Muhokama savollari

1. Model va modellashtirish tushunchalari.

2. Model turlari.

3. Formallashtirish.

4. Modellashtirishning yashash sikli.

5. Kompyuterda masala yechish bosqichlari.

6. Matematik modellashtirish.

Nazorat savollari:

1. Model nima?

2. Modellashtirish deganda nima tushuniladi?

3. Modellashtirishdan maqsad nima?

4. Modellarni qanday sinflarga ajratish mumkin?

5. Modellarga tushuntirish bering.

6. Matematik modellashtirish nima?

7. Matematik modelni qurish tamoyillari nimalarga bo‘linadi?

8. Analitik model nima?

9. Imitasion model nima?

10. Matematik model qurish uchun nimalar kerak?

11. O‘rganilayotgan obyekt, jarayon yoki tizimni matematik ifodalash nimalarga bog‘liq?

12. Matematik modelni tanlash bosqichida nimalar aniqlanadi?

13. Matematik model tuzishning asosiy bosqichlarini tushuntiring.

1.11 Bob bo‘yicha xulosalar

Ushbu bobda siz dasturlash jarayonini o‘rganish uchun zarur bo‘lgan asosiy tushunchalar: axborotni kodlash, axborotni o‘lchash, mantiq algebrasining asosiy tushunchalari va funksiyalari, algoritmlash asoslari, dasturiy va texnik ta’minot, operatsion tizim, modellashtirishning asosiy tushunchalari haqidagi tushunchalarga ega bo‘ldingiz. Axborotni o‘lchash uchun turli yondoshishlar va usullardan, masalan, R.Xartli va K.Shennon usullari bo‘yicha axborot o‘lchamini topishdan foydalaniladi.

Axborot miqdori - baholanayotgan tizimdagi koʻp shakllilikni (tarkiblilik, aniqlik, holatlarni tanlash va h.k) adekvat tavsiflovchi son.

Har qanday qoʻyilgan masalani dasturlash avval uning modelini yaratish, algoritmini tuzish dasturlash jarayonini yengillashtirishini tushundingiz deb oʻylaymiz.

II BOB. DASTURLASH. DASTURLASH TILLARI, ULARNING KLASSIFIKASIYASI VA ISHLATILISHI

2.1. C++ dasturlash tili

C++ tili Byarn Straustrup tomonidan 1980 yil boshlarida ishlab chiqilgan. C++ tilida yaxshi dastur tuzish uchun “aql, farosat va sabr” kerak bo‘ladi. Bu til asosan tizim sathida dasturlovchilar uchun yaratilgan.

C/C++ algoritmik tilining alifbosi:

1. 26 ta lotin va 32 ta kirill harflari (katta va kichik);
2. 0 dan 9 gacha bo‘lgan arab raqamlari;
3. Maxsus belgilar: - + * / : ; . , % ? ! = “” № < > { } [ ] ( ) $ # & ^ va h.k.

Dastur bajarilishi jarayonida o‘z qiymatini o‘zgartira oladigan kattaliklar o‘zgaruvchilar deyiladi. O‘zgaruvchilarning nomlari harfdan boshlanuvchi xarf va raqamlardan iborat bo‘lishi mumkin. O‘zguruvchilarni belgilashda katta va kichik harflarning farqlari bor. (A va a harflari 2 ta o‘zgaruvchini bildiradi) Har bir o‘zgaruvchi o‘z nomiga, toifasiga, xotiradan egallagan joyiga va son qiymatiga ega bo‘lishi kerak. O‘zgaruvchiga murojaat qilish uning ismi orqali bo‘ladi. O‘zgaruvchi uchun xotiradan ajratilgan joyning tartib raqami uning adresi hisoblanadi. O‘zgaruvchi ishlatilishidan oldin u aniqlangan bo‘lishi lozim.

O‘zgaruvchilarning son qiymatlari quyidagi ko‘rinishda yoziladi:

- Butun toifali o‘nlik sanoq tizimida: ular faqat butun sondan iborat bo‘ladilar. Masalan: 5; 76; -674 va h.k.
- Sakkizlik sanoq tizimidagi sonlar: 0 (nol) dan boshlanib, 0 dan 7 gacha bo‘lgan raqamlardan tashkil topadi. Masalan: *x*=0453217; *s*=077;
- O‘n oltilik sanoq tizimidagi sonlar: 0 (nol) dan boshlanadi va undan keyin x yoki X harfi keladi, so‘ngra 0-9 raqamlari va a-f yoki A-F harflaridan iborat ketma-ketliklar bo‘ladi. Masalan: 10 s.s.dagi 22 soni 8 s.s. da 026, 16 s.s.da 0x16 shaklida bo‘ladi.

- Haqiqiy toifali sonlar: ular butun va kasr qismlardan iborat bo‘ladilar. Masalan: 8,1; -12,59 va x.k. Haqiqiy toifali sonlarning bu ko‘rinishi oddiy ko‘rinish deyiladi. Juda katta yoki juda kichik haqiqiy toifali sonlarni darajali (eksponensional) formada yozish qulay. Masalan: $7,204*10^{12}$ yoki $3,567*10^{-11}$ kabi sonlar 7.204e+12 va 3.567e-11 ko‘rinishda yoziladi.
- Simvolli konstantalar. Ular qatoriga dastur bajarilishi’’ ichida qabul qilinadigan simvollar kiradi.

C/C++ tilida har qanday o‘zgaruvchi ishlatilishidan oldin e’lon qilinishi kerak. E’lon qilish degani ularning toifalarini aniqlab qo‘yish demakdir.

C++ tilida quyidagi toifali o‘zgaruvchilar ishlatiladi:

- Butun toifali kichik sonlar yoki simvollar uchun: char uning o‘zgarish intervali -128 dan +127 gacha yoki apostrof ichidagi ixtiyoriy 1ta simvol. Xotiradan 1 bayt joy oladi. Simvollar ASCII kodlariga mos keladi. (ASCII – American Standart Code for Information Interchange)
- Butun toifali o‘zgaruvchilar: int. Masalan: int a, i, j ; Bu yerda dasturda ishlatilayotgan a, i, j o‘zgaruvchilarining toifasi butun ekanligi ko‘rsatildi. Bu toifadagi o‘zgaruvchilar 2 bayt joy egallaydi. Ularning o‘zgarish intervali: -32768 dan +32767 gacha; (Hozirgi 32 razryadli kompyuterlarda 4 bayt joy oladi va oralig‘i 2 marta oshgan).
- Butun toifali katta (uzun) o‘zgaruvchilar: long. Masalan: long s, s2, aa34; Bu toifadagi o‘zgaruvchilar 4 bayt joy egallaydi. Ular –2147483648 dan +2147483647 oraliqdagi sonlarni qabul qilishi mumkin.
- Ishorasiz butun o‘zgaruvchilar: unsigned short – 2 bayt joy oladi, o‘zgarish intervali 0 dan 65535 gacha; unsigned long – 4 bayt joy oladi, o‘zgarish intervali: 0 dan 4294967295 gacha; unsigned char – 1 bayt joy oladi, o‘zgarish chegarasi 0 dan 255 gacha.
- Haqiqiy toifadagi o‘zgaruvchilar: float. Masalan: float a, b: Bu yerda dasturda ishlatilayotgan a, b o‘zgaruvchilarining toifasi haqiqiy ekanligi ko‘rsatilgan. Bu toifadagi o‘zgaruvchilar 4 bayt joy egallaydi va qabul qilish chegarasi 10^{-38} dan 10^{+38} gacha.

- Katta yoki kichik qiymatli o‘zgaruvchilarni ifoda etishda double toifasi ishlatiladi. Ular uchun 8 bayt joy ajratiladi va qabul qilish chegarasi 10^{-304} dan 10^{+304} gacha.
- Juda katta yoki juda kichik qiymatli o‘zgaruvchilar uchun long double toifasi ishlatiladi, u 10 bayt joy oladi va qabul qilish chegarasi $3.4*10^{-4932}$ dan $1.1*10^{-4932}$ gacha.
- Qator toifasidagi o‘zgaruvchilar uchun ham char toifasi belgilangan. Ular ham 1 bayt joy oladi va 0 dan 256 tagacha bo‘lgan simvollar ketma-ketligidan iborat bo‘lishi mumkin. Satr toifasidagi o‘zgaruvchilar qo‘shtirnoq (“) ichida yoziladi.

C++ tilida o‘zgaruvchilarni inisializasiya qilish degan tushuncha ham mavjud. Inisializasiya qilish degani o‘zgaruvchini e’lon qilish barobarida unga boshlang‘ich qiymatini ham berish demakdir. Masalan: int a=5, b, s=-100; - a, b, s o‘zgaruvchilari butun toifali ekanligi ko‘rsatildi va a o‘zgaruvchisiga 5 (a=5), s o‘zgaruvchisiga esa –100 (s=-100) boshlang‘ich qiymatlar berildi.

Dastur bajarilishi jarayonida o‘z qiymatini o‘zgartira olmaydigan kattaliklar o‘zgarmaslar deyiladi. Masalan: x=1; bo‘lsa keyinchalik x=x+5 deb yozib bo‘lmaydi. O‘zgarmaslarni const so‘zi bilan ko‘rsatiladi. Maslan: const int x=95; float y=9.17; (const lar simvol yoki nol (NULL) bo‘lishi xam mumkin.)

C++ tilida standart funksiyalarning yozilishi:

Funksiya	Ifodalanishi	Funksiya	Ifodalanishi
Sin x	sin(x)	$\sqrt{x}$	sqrt(x); pow(x,1/2.)
Cos x	cos(x)	$\lvert x\rvert$	abs(x) yoki fabs(x)
Tg x	tan(x)	Arctan x	atan(x)
e^x	exp(x)	Arcsin x	asin(x) ?
Ln x	log(x)	Arccos x	acos(x) ?
Lg x	log10(x)	$\sqrt[3]{x^2}$	pow(x,2/3.)
x^a	pow(x,a)	Log_2x	log(x)/log(2)

Masalan: $\frac{-b+\sqrt{b^2-4ac}}{2a}$ → (-b + sqrt(b*b-4*a*c)/(2*a); yoki

(-b+pow(b*b-4*a*c,1/2.)/(2*a);

$e^{\sin x} + tg^2(x+3) \rightarrow \exp(\sin(x)) + \text{pow}(\tan(x+3),2);$

k=(m*5)+((7 % n) / (9+x));

C++ tilidagi dastur quyidagi tarkibdan tashkil topadi:

1. Direktivalar – #include <file.h> direktiva – instruksiya degan ma’noni beradi. C++ tilida dasturning tuzilishiga, ya’ni ehtiyojiga qarab, kerakli direktivalar ishlatiladi. Ular < > belgisi orasida keltiriladi. Umuman olganda quyidagi direktivalar mavjud (jami 32 ta):

- #include <stdio.h> - C da oddiy kiritish/chiqarish dasturi uchun. Bu yerda std - standart, i – input, o - output degani.
- #include <iostream.h> - C++ da kiritish/chiqarish uchun, oddiy amallar bajarilsa.
- #include <math.h> - standart funksiyalarni ishlatish uchun.
- #include <conio.h> - dasturning tashqi koʻrinishini shakllantirish uchun.
- #include <string.h> - satr toifasidagi oʻzgaruvchilar ustida amallar bajarish uchun.
- #include <stdlib.h> - standart kutubxona fayllarini chaqirish uchun.
- #include <time.h> - kompyuter ichidagi soat qiymatlaridan foydalanish uchun.
- #include <graphics.h> - C++ tilining grafik imkoniyatlaridan foydalanish uchun.

Bu fayllar maxsus kutubxona e’lon fayllari hisoblanadilar va ular alohida INCLUDE deb nomlanadigan papkada saqlanadi. Hozirda C++ kutubxonasini yangilandi va undagi fayllarning nomlaridan .h (head – bosh ma’nosida) kengaytmasi olib tashlandi va oldiga c harfi qoʻshildi (C dan qolgan 18 tasiga). Bu fayllarda funksiya prototoifalari, toifalari, oʻzgaruvchilar, oʻzgarmaslar ta’riflari yozilgan boʻladi.

Direktivalar dasturni uni kompilyasiya qilinishidan oldin tekshirib chiqadi.

3. Makroslar - # define makro qiymati. Masalan:

#define y sin(x+25) – u = sin(x+25) qiymati berildi;

#define pi 3.1415 - pi = 3.1415

#define s(x) x*x - s(x) = x*x (; belgisi qo‘yilmaydi)

4. Global o‘zgaruvchilarni e’lon qilish. Asosiy funksiya ichida e’lon qilingan o‘zgaruvchilar lokal, funksiyadan tashqarida e’lon qilinganlari esa global o‘zgaruvchilar deyiladi. Global o‘zgaruvchilar dastur davomida ishlaydi va xotiradan ma’lum joyni egallaydi. O‘zgaruvchini bevosita ishlatishdan oldin e’lon qilsa ham bo‘ladi, u holda o‘z lokal bo‘ladi. Global o‘zgaruvchilar nomi lokal o‘zgaruvchilar nomi bilan bir xil bo‘lishi ham mumkin. Bunday holatda lokal o‘zgaruvchining qiymati joriy funksiya ichidagini qiymatini o‘zgartiradi, funksiyadan chiqishi bilan global o‘zgaruvchilar ishlaydi.

5. Asosiy funksiya - main ( ) hisoblanadi. Bu funksiya dasturda bo‘lishi shart. Umuman olganda C++ dagi dastur funksiyalardan iborat deb qaraladi. main ( ) funksiyasi { boshlanadi va dastur oxirida berkitilishi shart } . *main* – asosiy degan ma’noni beradi. Bu funksiya oldida uning toifasi ko‘rsatiladi. Agar main ( ) funksiyasi beradigan (qaytaradigan) javob oddiy so‘z yoki gaplardan iborat bo‘lsa, hech qanday natija qaytarmasa, void so‘zi keltiriladi. main ( ) funksiyasi dastur tomonidan emas, balki OS tomonidan chaqiriladi. OSga qiymat qaytarish shart emas, chunki u bu qiymatdan foydalanmaydi. Shuning uchun main ( ) funksiyasining turini *void* deb ko‘rsatganimiz ma’qul. Har bir funksiyaning o‘z argumenti bo‘ladi, shuning uchun main funksiya ( ) lari ichiga uning parametri keltiriladi. Ba’zan u bo‘sh bo‘lishi ham mumkin. Bu funksiyadan chiqish uchun odatda *return* operatori ishlatiladi. 0 (nol) qiymatining qaytarilishi operasion tizimga ushbu dastur normal bajarilib turganini bildiradi. return orqali qaytadigan qiymat toifasi funksiya e’lonidagi qaytish toifasi bilan bir xil bo‘lishi kerak. Masalan int main ( ) va 0 (nol) qiymat butun toifalidir. Bu funksiyadan so‘ng lokal o‘zgaruvchilar, qism dasturlar, ularning haqiqiy parametrlar e’lon qilinadi. So‘ngra dasturning asosiy operatorlari (kiritish/chiqarish, hisoblash va h.k.) yoziladi. Agar bu operatorlar murakkab toifali bo‘lsalar, ularni alohida {} qavslarga olinadi. C++ tilida dastur kichik harflarda yoziladi. Ba’zi operatorlar katta harflar bilan kelishi mumkin, bunday xollarda ular alohida aytib o‘tiladi. Operatorlar oxiriga ; belgisi qo‘yiladi. Operatorlar bir qatorga ketma-ket yozilishi mumkin. Dasturda izohlar

xam kelishi mumkin, ular /* ....*/ belgisi orasiga olinadi. Agar izoh bir qatorda tugasa, uni // belgisidan keyin yoziladi. Masalan:

main ( ) // C++ tilining asosiy funksiyasi

Tilda quyidagi amallardan foydalanish mumkin:

1. Arifmetik amallar: +, -, /, *, %. Barcha amallar odatdagidek bajariladi, faqat bo‘lish amali butunga bo‘lish bajariladi, ya’ni agar butun sonlar ustida bajarilayotgan bo‘lsa, natija doim butun bo‘ladi, ya’ni kasr qism tashlab yuboriladi (9/5=1; vaxolanki 1,8 bo‘lishi kerak). Shuning uchun surat yoki maxrajiga nuqta (.) qo‘yilsa, natija ham xaqiqiy bo‘ladi (9./5=1.8). % belgisi (modul operatori) esa butun sonni butun songa bo‘lgandan hosil bo‘ladigan qoldiqni bildiradi.

Masalan: 9 % 5=4

2. Taqqoslash amallari: = = (tengmi?); != (teng emas); < ; > ; >=; <=

3. Mantiqiy amallar: && (and) mantiqiy ko‘paytirish; || (or) mantiqiy qo‘shish; ! (not) mantiqiy inkor. Mantiqiy amallarni ixtiyoriy sonlar ustida bajarish mumkin. Agar javob rost bo‘lsa, natija 1 bo‘ladi, agar javob yolg‘on bo‘lsa, natija 0 bo‘ladi. Umuman olganda 0 (nol) dan farqli javob rost deb qabul qilinadi.

Masalan: i>50 && j==24 yoki s1 < s2 && (s3>50 || s4<=20);

Yoki $6 \le x \le 10$ yozuvini x>=6 && x<=10 deb yoziladi

4. Qiymat berish amallari:

a. a=5; b = 2*c; x = y = z =1; a = (b = c)*d // 3=5 deb yozib bo‘lmaydi

b. qabul qildim va almashtirdim deb nomalandigan amallar:

+ = : a+=b → a = a + b;

- = : a-=b → a = a - b;

* = : a*=b → a = a * b;

/ = : a/=b → a = a / b;

% = : a%=b → a = a % b;

- inkrement operatsiyasi (++) ikki ma’noda ishlatiladi: o‘zgaruvchiga murojaat qilinganidan keyin uning qiymati 1 ga oshadi (a++ postfiks ko‘rinishi) va o‘zgaruvchining qiymati uning murojaat qilishdan oldin 1 ga oshadi (++a prefix ko‘rinishi);

- dekrement operatsiyasi (--), xuddi inkrement operatsiyasi kabi, faqat kamaytirish uchun ishlatiladi. Masalan: s = a + b++ (a ga b ni qoʻshib keyin b ning qiymatini 1 ga oshiradi); s = a+(--b) (b ning qiymatini 1 ga kamaytirib, keyin a ga qoʻshadi).

Yuqoridagi standart funksiyalardan tashqari yana quyidagi funksiyalar ham ishlatiladi:

- ceil (x) - x ni x dan katta yoki unga teng boʻlgan eng kichik butun songacha yaxlitlash. Masalan: ceil (12.6) = 13.0; ceil (-2.4) = -2.0;
- floor (x) - x ni x dan kichik boʻlgan eng katta butun songacha yaxlitlash. Masalan: floor (4.8) = 4.0; floor (-15.9) = -16.0; floor(12.1) = 12; floor(-12.1)=-13;
- fmod (x,y) – x / y ning qoldigʻini kasr son koʻrinishida berish. Masalan: fmod(7.3, 1.7) = 0.5;

Muhokama savollari

1. Tilning alifbosi.
2. Oʻzgarmaslar
3. Oʻzgaruvchilarning toifalari.
4. Standart funksiyalarning koʻrinishi.

Nazorat savollari

1. C++ tilida oʻzgarmaslar.
2. C++ tilida oʻzgaruvchilarning toifalari
3. Kompanovka bosqichlarini ayting.
4. Standart funksiyalarning qoʻllanishi.
5. Ifodalar haqida tushuncha.
6. Dastur tuzilishi.
7. Preprosessor direktivalari
8. Identifikator, oʻzgaruvchilar va oʻzgarmaslar.
9. Oʻzgaruvchilarning oddiy toifalari.
10. Amallar va ifodalar.

11. Standart matematik funksiyalar.
12. Oddiy arifmetik ifodalar.

2.2 Operatorlar. C++ tilida ma’lumotlarni kiritish va chiqarish

Ma’lumotlarni kiritish/chiqarish (Input/Output), ular ustida amallarni bajarish va natijalarni olish C++ da oqim Ob’yektlari orqali bajarilishi mumkin. Kiritish/chiqarishni C dagi kabi funksiyalar bilan ham amalga oshirsa bo‘ladi. C++ falsafasiga ko‘ra har bir kiritish/chiqarish jixozi (ekran, printer, klaviatura va hokazo) baytlar oqimi bilan ishlagandek qabul qilinadi. Normal holatda bu oqimlar ekranga va klaviaturaga ulangan bo‘ladi. Bu oqim ikki xil: >> va << ko‘rinishda qabul qilingan. Ma’lumotlarni chiqarish oqimi cout<< ko‘rinishida va ma’lumotlarni kiritish cin >> ko‘rinishida yoziladi. cout (console output) va cin (console input) ma’nolarini bildiradi.

Ma’lumotlarni kiritishda cin so‘zidan keyin o‘zgaruvchilarning nomlari keltiriladi. Agar ular bir nechta bo‘lsa, har bir o‘zgaruvchi nomidan so‘ng >> belgisi qo‘yiladi.

Masalan: cin >> a >> b; cin >>GG >>lola >> a12; , bu o‘zgaruvchilarning son qiymatlari dastur kompilyasiyadan o‘tganidan keyin klaviatura orqali beriladi. Sonlarni bo‘sh joy (probel) orqali yoki Enter klavishasi orqali har bir satrda bittadan kiritish mumkin. cin operatori orqali faqat son qiymatlar kiritiladi, ifodalar yozilishi mumkin emas.

Ma’lumotlarni chiqarish uchun

cout<< identifikator;

operatori ishlatiladi. Bu yerda o‘zgaruvchi nomlari, qo‘shtirnoq ichida istalgan so‘z yoki gaplar yozilishi mumkin, ushbu esa izoh sifatida qabul qilinadi. Agar natijasi berilayotgan o‘zgaruvchilar bir nechta bo‘lsa ularni alohida holatda yoki ketma-ket chiqarilishi mumkin. Masalan:

cout<< a; cout<< a << b;

cout << “y=” << y;

cout << “funksiyaning qiymati teng =”<< f;

Chiqarish oqimini kiritish oqimi bilan birga qo‘llansa, dastur ko‘rinishi tushunarli va chiroyliroq bo‘ladi. Masalan:

cout << “1 - sonni kiriting”;

cin >> x;

cout << “2 - sonni kiriting”;

cin >> y;

Bir nechta natijalar chiqarilayotgan bo‘lsa, ular ekranda ketma-ket ko‘rinishda namoyon bo‘ladilar. Bu holat esa natijalarni o‘qishda noqulaylik tug‘dirishi mumkin. Shuning uchun cout << oqimi oxirida endl (end line-satr oxiri) so‘zi qo‘yilsa, kursor keyingi qatorga o‘tadi va keyingi natija ko‘rinadi.

Masalan: cout << “Natijalar:”<< endl;

cout << “1-funksiyaning qiymati=”<<f1<<endl;

cout << “2-funksiyaning qiymati=”<<f2<<endl;

Ekranda quyidagi javoblar ko‘rinadi:

Natijalar:

1-funksiyaning qiymati=2.35847

2-funksiyaning qiymati=-95.3687

cout << ichida bevosita arifmetik amallarni ham keltirsa bo‘ladi.

cout <<”a*sin(b) ning javobi=”<< a*sin(b)<<endl; yoki

cout <<”a*sin(b) ning javobi=”<< endl;

cout << a*sin(b)<<endl;

Natijalarni 8 s.s va 16 s.s larida ham chiqarsa bo‘ladi. Buning uchun oct (8 s.s uchun) hex (16 s.s uchun) so‘zlari ishlatiladi. Misol:

cin >> a;

cout << a << endl;

cout << oct << a << endl;

cout << hex << a << endl;

Natijada kiritilgan son avval 10 s.s., keyin 8 s.s. va 16 s.s.da ko‘rinadi.

cout operatori bilan yana quyidagi belgilar ham ishlatilishi mumkin (ularni ESC – eskeyp simvollar ham deyiladi):

\n – yangi satr. Kursor yangi qator boshidan joy oladi.

\t – gorizontal tabulyasiya, kursor bir nechta harf o‘ngga siljiydi.

\v – vertikal tabulyasiya, kursor bir nechta satr tashlab o‘tadi.

\r – qaytish, ya’ni kursor ayni satr boshiga qaytadi, yangi satrga o‘tmaydi.

\a – kompyuter dinamik ovoz chiqaradi.

Izoh: \n belgisi bilan endl so‘zi orasida farq bor.

Agar chiqarilayotgan natija haqiqiy toifada bo‘lsa, uning aniqligini nazorat qilish mumkin, ya’ni nuqtadan keyin necha xonagacha aniqlikda olishni boshqarish mumkin. Buning uchun cout. precision(n); funksiyasi ishlatiladi. Bu yerda n – aniqlik ko‘rsatgichi.

Masalan: cout.precision(3);

```
cout << x;
```

Bu funksiya dasturda bir marta yozilsa kifoya qiladi.

C++ tilidagi dastur quyidagi tarkibdan tashkil topadi:

```
#preprossor direktivalari
void main ()
{
// operatorlar
}
```

Dasturda formatli kiritish va chiqarishni ham amalga oshirish mumkin.

Formatli kiritish:

scanf (“kiritish formati”, identifikatorlar adresi);

Kiritish formati - kiritilayotgan (chiqarilayotgan) sonlar toifasini ko‘rsatuvchi simvolli satr.

%d – butun son

%f – haqiqiy son

%c – 1 simvol

%s – simvolli satr

Formatli chiqarish:

printf (“chiqarish formati”, identifikatorlar);

Masala. Ikkita butun son kiritilib, ularning yig‘indisi ekranga chiqarilsin.

```
#include <stdio.h>
#include <conio.h>
main()
{
 int a, b, c;
 printf (“Ikkita butun sonni kiriting\n”);
 scanf (“%d%d”, &a, &b);
 c = a + b;
 printf (“%d”, c);
 getch();
}
```

Ishlatilishiga misollar:

printf (“%d”,c);

printf (“natija: %d”, c);

printf (“%d+%d=%d”, a, b, c);

printf (“%d+%d=%d”, a, b, a+b);

printf (“%9.3f”, x); -chiqarishda pozitsiyalar sonini ko‘rsatish

printf (“%e”, x);

printf (“%10.2e”, x);

Muhokama savollari

1. Tilda ishlatiladigan kiritish operatorlari.
2. Tilda ishlatiladigan chiqarish operatorlari
3. Qiymatlarni kiritish va chiqarish.
4. C++ tilida ma’lumotlarni oqim bilan kiritish .
5. C++ tilida ma’lumotlarni oqim bilan chiqarish.
6. C++ tilining asosiy operasiyalari.

7. Boshqarish qatori.
8. Format spesifikatori.
9. Format modifikatori.
10. Dastur kompilyasiyasi va bajarilishi.

Nazorat savollari

1. C++ tilida ma’lumotlarni oqim bilan kiritish.
2. C++ tilida ma’lumotlarni oqim bilan chiqarish.
3. C++ tilining asosiy operasiyalari.
4. Boshqarish qatori.
5. Format spesifikatori.
6. Format modifikatori.
7. Dastur kompilyasiyasi va bajarilishi.

2.3 Chiziqli tuzilmali algoritmlarni dasturlash

Endi yuqorida ko‘rib o‘tilgan operatorlar yordamida birinchi chiziqli dasturni tuzib ko‘ramiz.

```
#include <iostream.h>
void main ()
{
cout << “Biz C++ tilida dastur tuzishni o‘rganamiz \ n”;
}
```

Ekranda: Biz C++ tilida dastur tuzishni o‘rganamiz so‘zi paydo bo‘ladi.

2-misol. $y=x^2$ funksiyani hisoblash uchun dastur tuzing. Bu yerda x – ixtiyoriy haqiqiy o‘zgaruvchi.

```
#include <iostream.h>
int main ( )
{
```

```
float x,y;
cin >> x;
y=x*x;
cout << “y=”<< y <<endl; }
```

Dasturning yana bir ko‘rinishi:

```
#include <iostream.h>
int main ( )
{
float x=3.26; // x inisalizasiya qilindi;
cout << “y=”<< x*x <<endl; // y ning qiymati bevosita yozildi;
}
```

Dasturning yana bir ko‘rinishi:

```
#include <iostream.h>
#include <math.h> //standart funksiyalar kutubxonasi chaqirildi
 int main ( )
{
float x=3.26; // x inisalizasiya qilindi;
cout << “y=”<< pow(x,2) <<endl; // y ning qiymati bevosita yozildi;
}
```

Izoh: O‘zgaruvchi toifasi butun deyilsa-yu, unga haqiqiy qiymat berilsa ham, natija butun bo‘ladi. Masalan:

int x=1.2, y;
y = x+5;
cout <<y;
Natija: 6 chiqadi
False
	float x=1.2, y;
	y = x+5;
	cout << y;
	Natija 6.2 chiqadi
	True
		int x=1.2; float y;
		y = x+5;
		cout << y;
		Natija 6 chiqadi
		False

int x=1.2, y;
y = sin(x)+5;
cout <<y;
Natija: 5 chiqadi
False
	float x=1.2, y;
	y = sin(x)+5;
	cout <<y;
	Natija: 5.9320 chiqadi
	To‘g‘ri!!! True
		int x=1.2; float y;
		y = sin(x) + 5;
		cout <<y;
		Natija: 5.8414 b-di
		??? false

O‘zgaruvchilarni e’lon qilish ularni ishlatilishi vaqtida bo‘lishi ham mumkin. Bunday holatda o‘zgaruvchining toifasi qavslarda yoziladi. Masalan:

float b = 2 * a;

double p = 5.5 * sin (a);

Dasturni ishga tushirish uchun F9 klavishasi bosiladi.

Translyator – dasturni kompilyatorga yetkazadi, olib boradi.

Kompilyator – dasturni mashina kodiga aylantiradi va undagi xatolarning barchasini ko‘rsatadi, natijani oladi.

Error – xatolik degani;

Waring – ogohlantirish degani;

Uchburchak masalasi. Uchburchakning 2 ta tomoni va ular orasidagi burchagi berilgan. Uchburchakning 3-tomonini, uning yuzasini, unga tashqi va ichki chizilgan aylana radiuslarini va qolgan burchaklarini topish dasturini tuzing.

Demak: a, b, γ berilgan. Topish kerak: c, s, R, r - ?

c= $\sqrt{a^2+b^2-2ab\cos\gamma}$; s = $\frac{ab\sin\gamma}{2}$;

r = $\frac{abc}{4s}$; R = $\frac{2s}{a+b+c}$; $\frac{a}{\sin\alpha}=\frac{b}{\sin\beta}=\frac{c}{\sin\gamma}=2R$

```
#include <iostream.h>
#include <math.h>
int main ( )
{ float a, b, c, r, R, s, alf, bet, gam, x, pi=3.14;
cout << “uchburchak tomonlarini kiriting:\n”;
cin >>a>>b;
 gam=pi/3;  c = sqrt (a*a+b*b-2*a*b*cos(gam));
s = a*b*sin(gam)/2;
r = a*b*c / (4*s);
R = 2*s / (a+b+c);
x = b *sin(gam) / c;
bet = atan (sqrt (1- x*x)/x);
```

eslatma:

$\arcsin x = arctg\frac{x}{\sqrt{1-x^2}}$

$\arccos x = arctg\frac{\sqrt{1-x^2}}{x}$

```
alf = pi – bet – gam;
cout << “Natijalar:”<<endl;
cout << “c=”<<c<<endl;
cout << “r=”<<r<<endl;
cout << “R=”<<R<<endl;
cout << “s=”<<s<<endl;
cout << “alfa=”<<alf<<endl;
cout << “betta=”<<bet<<endl;
}
```

Muhokama savollari

1. Chiziqli dastur tushunchasi.
2. Chiziqli dasturga misollar.
3. Dasturning kompilyasiya qilinishi.

Nazorat savollari

1. Chiziqli dastur tarkibi.
2. Chiziqli dasturda qo‘llaniladigan operatorlar.
3. Kiritish va chiqarishni standart ravishda amalga oshirish.
4. Kiritish va chiqarishni oqim bilan amalga oshirish.

2.4 Tarmoqlanuvchi algoritmlarni dasturlash

Ko‘p masalalarning yechimi ma’lum bir shart yoki shartlarning qo‘yilishiga qarab bajariladi. Bunday jarayonlarni tarmoqlanuvchi hisoblash jarayoni deyiladi. Tarmoqlanuvchi hisoblash jarayonlari tarkibida yana tarmoqlanish bo‘lishi mumkin. Bundaylarni murakkab tarmoqlanuvchi jarayonlar deb ataladi. Algoritmik tilda kattaliklarning istalgan xossasi shu ondagi qiymatlari uchun bajarilishi yoki bajarilmasligi shart deyiladi. Masalan: a = v tenglik uchun a=3 va

v=3,1 bo‘lganda shart bajarilmaydi. N tub son deyilsa va N=19 bo‘lsa, shart bajariladi, N=15 bo‘lganda shart bajarilmaydi [6].

Tarmoqlanuvchi jarayonlarni tashkil etishda shartsiz o‘tish va shartli o‘tish operatorlaridan foydalaniladi. Shunday jarayonlar mavjudki, shartning bajarilishiga qarab, dasturning u yoki bu qismiga o‘tishga to‘g‘ri keladi. Bunday hollarda shartsiz o‘tish operatori ishlatiladi.

1. Shartsiz o‘tish operatori:

goto n;

bu yerda n – belgi bo‘lib, jarayon o‘tishi kerak bo‘lgan joyning belgisi hisoblanadi. Belgi harf, son va ular aralashmasidan iborat bo‘lishi mumkin. 1 ta operatorga bir nechta belgilarni qo‘yish mumkin. (Usta dasturchilar goto n operatoridan kamroq foydalanadilar).

2. Shartli o‘tish operatori:

if (shart) operator;

Uning ishlashi quyidagicha: agar shart rost bo‘lsa keltirilgan operator bajariladi, agar shart yolg‘on bo‘lsa, keyingi qatorga o‘tiladi. Ko‘pincha bu ko‘rinish ishlatilganda 2 ta operatorlar aralashib ketmasligi uchun goto operatori ishlatildi. Agar if so‘zidan keyin bir nechta operatorlar keladigan bo‘lsa, ularni alohida {} qavslarga olinadi. (bu usul kamroq ishlatiladi)

Masalan:

$$Y=\begin{cases}\sin x, & agar \quad x<5\\ \sqrt[3]{x^2}, & agar \quad x\geq 5\end{cases}$$

```
#include <iostream.h>
#include <math.h>
int main ( )
{ float x, y;          // x va y ning toifasi haqiqiy
  cin >> x;          // x ning son qiymati kiritiladi
  if (x<5)                  // agar x<5 bo‘lsa
 { y=sin(x); goto cc; }         // 1-funksiya ishlaydi
   y=pow(x, 2/3.);         // aks holda 2-funksiya ishlaydi
```

cc: cout << “y=”<< y<< endl; // y- ning javobi beriladi. cc-belgi

} // main funksiyasi berkitildi.

3. Shartli o‘tish operatorning uzun ko‘rinishi

if (shart) 1-operator(lar); else 2-operator(lar);

Masalan: yuqoridagi misolni ko‘rib o‘tamiz:

```
#include <iostream.h>
#include <math.h>
int main ( )
{ float x, y;
  cin >> x;
  if (x<5)  y=sin(x); else  y=pow(x, 2/3.);
  cout << “y=”<< y<< endl;  }
```

Izoh: if – else konstruksiyasi ichida yana if – else konstruksiyasi ishlatilishi mumkin. Bunda bir nechta if operatoridan iborat ichma-ich joylashgan konstruksiya hosil bo‘ladi. Bunday hollarda else so‘zi o‘ziga yaqin turgan if ga tegishli bo‘ladi.

4. ? operatori

shart ? 1-operator (lar) : 2-operator (lar);

Masalan: x <5 ? y=sin(x) : y=pow(x,2/3.);

(agar shart rost bo‘lsa, 1-operator, aks holda 2-operator bajariladi).

5. **Tanlash operatori** – O‘zgaruvchining qiymatiga qarab ko‘p tarmoq ichidan bittasi tanlanadi.

Bu operatorning ko‘rinishi quyidagicha:

```
switch (ifoda yoki o‘zgaruvchi)
{ case 1-qiymat : operator(lar); break;
 case 2-qiymat : operator(lar); break;
……
  case n - qiymat : operator(lar); break;
  default : aks holdagi operator (lar);  }
```

Masalan:

$$Y = \begin{cases} \sin x, \ agar \ x = 1 \\ \cos x, \ agar \ x = 2 \\ tgx, \ agar \ x = 3 \\ \sqrt{x}, \ boshqa \ barcha \ hollarda \ (x > 0) \end{cases}$$

```
#include <iostream.h>
#include <math.h>
int main ( )
{ int x; float y;
cin >> x;
switch (x)
{ case 1 : y=sin(x); break;
  case 2 : y=cos(x); break;
  case 3 : tan(x);    break;
  default : y=sqrt(x); }
cout << “y=”<<y<<”x=”<<x<<endl;
getch ( ); }
```

Izoh: 1) switch operatoridagi ifoda yoki o‘zgaruvchi butun yoki simvolli toifali bo‘lishi shart!

2) switch operatori satrlaridagi break so‘zi tushib qolsa, joriy case operatoridan keyingi case bloklari ichidagi ifodalar ham bajarilaveradi.

Tekshirilayotgan shartlar bir nechta bo‘lishi ham mumkin. Bunday hollarda ularni murakkab shart deyiladi. Bunday shartlarni quyidagi mantiqiy amallar orqali ifoda etiladi:

&& - mantiqiy ko‘paytirish (va)

|| – mantiqiy qo‘shish (yoki)

! - mantiqiy inkor (emas)

Masalan: $6 \le x \le 10$ bo‘lsa, (x >=6) && (x<=10)

$Y > 0$ va $x < 4$ yoki $z \ge 5$ bo‘lsa, (y >0) && (x<4) || (z>5)

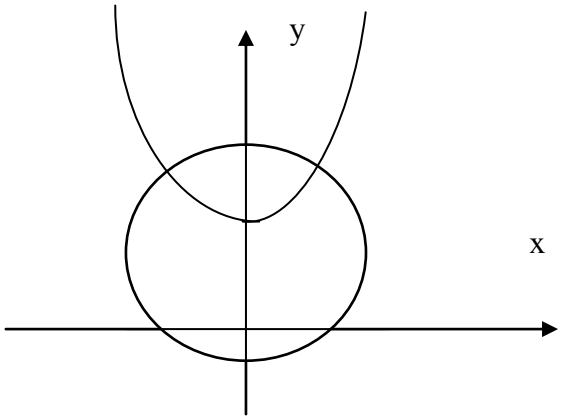
if (sin(x) > 1) && (5 / 2 = 0) y:=1; else y:=0; {ikkala shart ham yolg‘on qiymatga ega, shuning uchun y=0 bo‘ladi.}

Mantiqiy qiymatlar ustida amallar bajarilganida quyidagi natijalar olinadi:

(+ true 1; - false 0 degan ma’noda)

A	B	!A	!B	A&&B	A||B
0	0	1	1	0	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	1	1

Mantiqiy masala: Ixtiyoriy berilgan M(x,y) nuqta $y=x^2$ va $x^2+y^2=4$ aylana bilan kesishgan sohaga yoki shu aylananing 4-choragi tashqarisiga tushishini tekshiring.

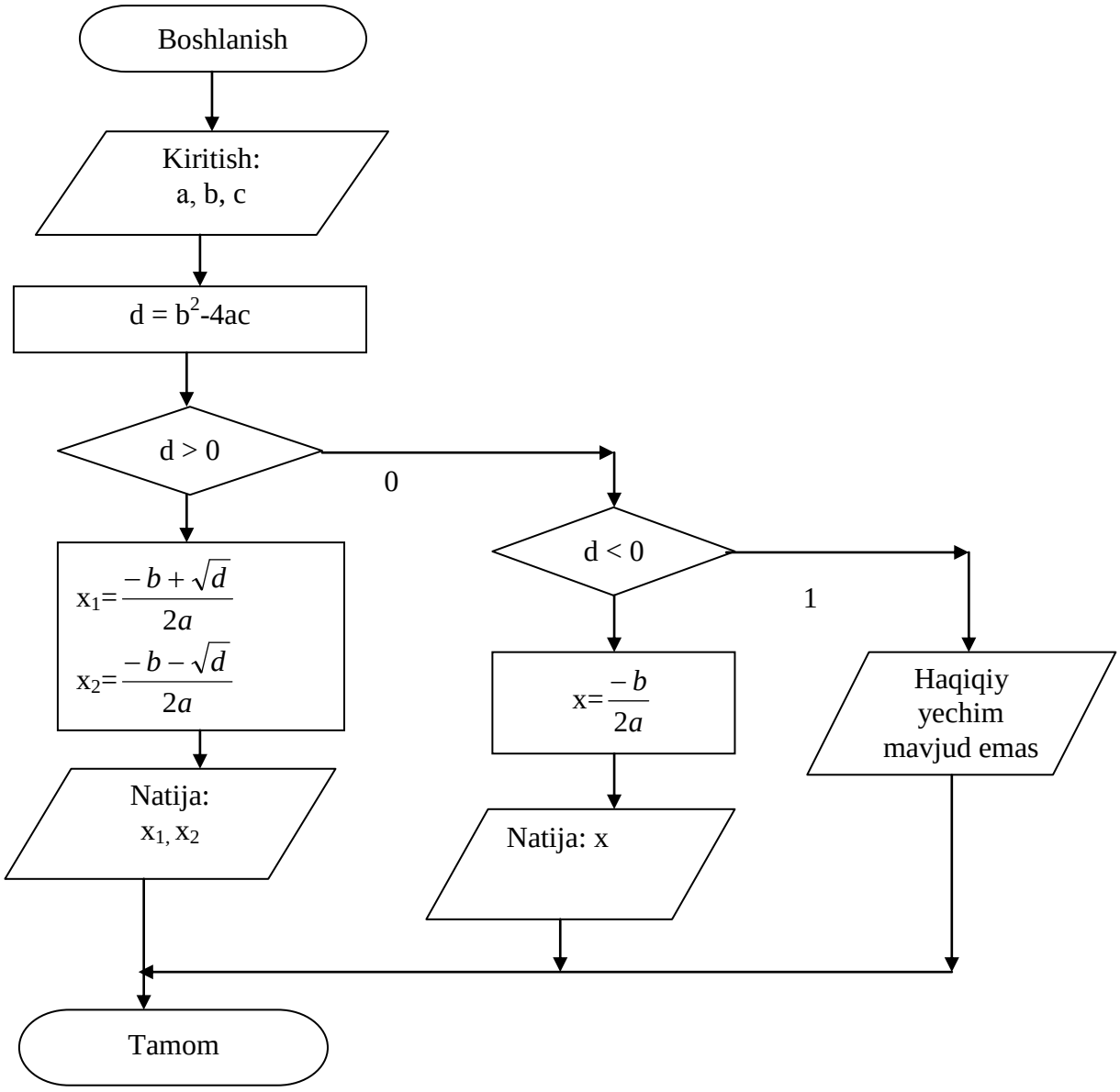


Demak:

$y > x^2$ and $x^2 + y^2 <= 4$
or $x > 0$ and $y < 0$ and
$x^2 + y^2 > = 4$
$x = 1, y = 1 \rightarrow$ false
$x=1, y=0 \rightarrow$ true
$x = -2, y = 0.5 \rightarrow$ false
$x =2, y= -2 \rightarrow$ true

```
#include  <iostream. h>
int main ( )
{  float x, y;  int n;
cout << “nuqtaning koordinatalarini kiriting:”;
 cin >> x>>y;
if ((y>=x*x && x*x+y*y<=4) ||
(x>0 && y<0 && x*x+y*y>=4)) n=1;
else n=0;
cout << “x=”<<x<<endl;
cout << “y=”<<y<< endl; cout << “n=”<<n<< endl; }
```

3-misol: $ax^2+bx+c=0$ ko‘rinishdagi kvadrat tenglamaning haqiqiy yechimlarini topish algoritmini tuzing. (a≠0; b≠0; c≠0;)



include <iostream.h> #include <math.h> int main ( ) float a, b, c, d, x, x1, x2; cout<< “Tenglamaning koeffisientlari: ”; cin >> a>>b>>c; d = b*b - 4*a*c;	#include <iostream.h> #include <math.h> int main ( ) float a, b, c, d, x, x1, x2; int v; cout << “Tenglamaning koeffisientlari:”; cin >> a>>b>>c; d = b*b - 4*a*c;

if (d == 0) { x=- b / (2*a);
cout <<"x="<<x<<endl; go to b15; }
if (d > 0)
{ x1 = (- b + sqrt(d)) / (2*a);
x2 = (- b - sqrt(d)) / (2*a);
cout
<<"x1="<<x1<<"x2="<<x2<<endl; }
else cout << "yechimi yo'q"<< endl;
b15 : }
	if (d<0) v=0;
	if (d = = 0) v=1; else v=2;
	switch (v)
	{ case 0: cout <<"yechimi yuk"<<
	endl; break;
	case 1 : { x=- b / (2*a);
	cout <<" x="<<x<<endl; } break;
	case 2: { x1 = (- b + sqrt(d)) / (2*a);
	x2 = (- b - sqrt(d)) / (2*a);
	cout
	<<"x1="<<x1<<"x2="<<x2<<endl; }
	break; }

Muhokama savollari

1. O'tish operatori.
2. Shartli operatorning qisqa ko'rinishi.
3. Shartli operatorning uzun ko'rinishi.
4. Tanlash operatori.

Nazorat savollari

1. Shartli o'tish operatorining vazifasi
2. Shartli o'tish operatorlarining ko'rinishlari
3. Murakkab operatorlar qachon va qanday qo'llaniladi?
4. Shartli o'tish operatori ichida yana shartli operator qatnashishi mumkinmi?
5. Shartni ifodalovchi blok sxema tuzing.
6. Tanlash operatorining vazifasi.
7. Tanlash operatoridagi **break** ning vazifasi.
8. Tanlash operatoridagi o'zgaruvchilarning toifalari qanday bo'lishi kerak?

2.5 C++ tilida takrorlanuvchi jarayonlarni dasturlash

Agar dastur bajarilish jarayonida operator yoki operatorlar guruhi bir necha marta qayta-qayta bajarilsa, bunday jarayonlarni takrorlanuvchi (siklik) jarayon deyiladi. C++ tilida siklni 3 xil ko‘rinishda tashkil qilish mumkin.

1. <u>Sharti avval tekshiriladigan takrorlanish (oldshartli sikl):</u>

while (shart) operator (lar);

Bu yerda operatorlar while da ko‘rsatilgan shart yolg‘on bo‘lgunicha takrorlanadi. Takrorlanish tanasi murakkab bo‘lsa, ya’ni 1 tadan ortiq operatorlar qatnashsa, ularni alohida {} ichiga olish kerak bo‘ladi.

Masalan: b = 2*(a+5); a € [1, 10]; h=1;

```
#include <iostream.h>
#include <math.h>
int main ( )
{  int a=1, b;
 while (a<=10)
 {  b = 2*(a+5); cout << “b=” <<b;
 cout << “a=” <<a << endl;
 a++ ; }
}
```

Ekranda 10 ta a va b larning qiymatlari paydo bo‘ladi.

2- misol.

```
#include <iostream.h>
 int main ( )
 {  int i = 10;
    while ( i++ < =15)
    cout << “Salom!!!”<< endl;  }
```

Ekranda 5 marta “Salom!!!” yozuvi paydo bo‘ladi.

2. <u>Sharti keyin tekshiriladigan takrorlanish (so‘ngshartli sikl):</u>

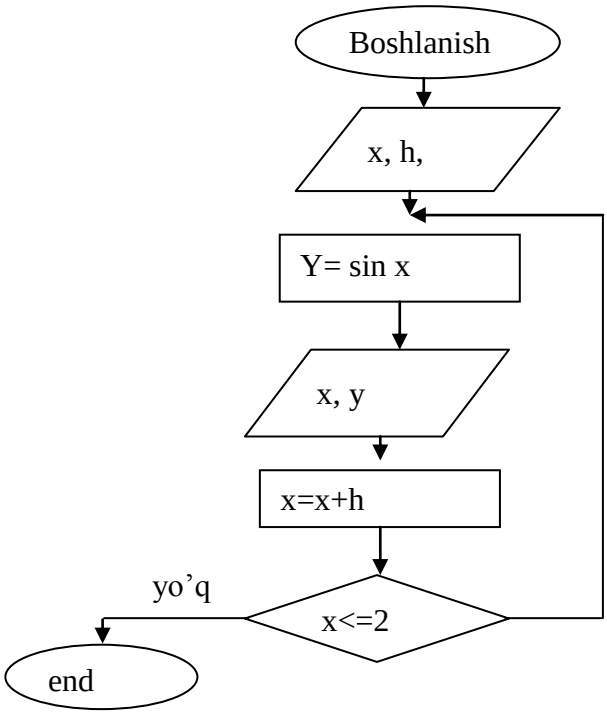
```
do
operator (lar)
while (shart);
```

Takrorlanish <u>while</u> da ko‘rsatilgan shart <u>yolg‘on</u> bo‘lgunicha davom etadi.

Masalan: y=sinx; x € [1,2] ; h=0.1

```
#include <iostream.h>
#include <math.h>
 int main ( )
{ float x=1, y;
 do
  { y=sin(x); cout << “x=”<<x<<”  y=”<<y<<endl;    x+ = 0.1;  }
   while (x<=2); getch(); }
```

Masalaning algoritmi quyidagi ko‘rinishga ega bo‘ladi:



2- misol. Dastur klaviaturadan 20 soni kiritilishini kutadi.

```
#include <iostream>
int main ( )
{ int n;
  do
  { cin >> n;
  cout << “Sonni qayta kiriting!=”<<n; }
  while (n!=20);
 }
```

3. <u>Parametrli takrorlanish (sikl):</u>

<u>Umumiy ko‘rinishi</u>

for (bosh qiymat; shart; o‘zgarish qadami)

operator (lar);

Operatorlar 1 tadan ortiq bo‘lsa ularni alohida qavslar -{} ichiga olinadi.

1-misol. Y = cos x; x € [2,3] ; h=0,2;

```
#include <iostream.h>
#include <math.h>
int main ( )
{ float x, y;
  for (x=2; x<=3; x+ = 0.2)
  { y=cos(x);  cout << “x=”<< x << “  y=”<< y << endl; }
}
```

2-misol. 100 gacha bo‘lgan juft sonlarni ekranga chiqarish dasturi.

```
#include <iostream.h>
  int main ( )
 {   int i = 2;
  while (i<=100)
  { cout << “i=”<< i;  i += 2; }  }
```

…. for (int i=2; i<=100; i +=2) cout << “i=”<< i;	do cout << “i=”<< i; i += 2; while (i<=100);

3-misol: 1 dan 100 gacha bo‘lgan 3 raqami bilan tugaydigan sonlarni ekranga chiqarish dasturini tuzing (2 xil usulda).

……
int i=3;
while (i <=100)
{ cout << “i=”<< i;
  i + = 10; }
	…..
	for (i = 3;i <= 100; i + = 10)
	cout << “i=”<< i;

4-misol. Qadimiy masala. Bir odam 100 so‘m bilan bozorga bordi. Bozorda 1 ta sigir 10 so‘m, 1 ta qo‘y 3 so‘m, 1 ta echki 0.5 so‘m va xarid qilingan qoramollarning umumiy soni 100 ta bo‘lsa, nechta sigir, qo‘y va echki sotib olindi?

Sigirlar soni: x, qo‘ylar soni y, echkilar soni z deb olinsa,

#include <iostream.h>
int main ( )
{
int x, y, z, s;
for (x=1; x<=100; x++)
for (y=1; y<=100; y++)
if (19*x + 5*y = = 100)
{ z = 100 – x – y;
cout << “x=”<<x;
cout << “y=”<<y;
cout << “z=”<<z; }
return 0;
}
	#include <iostream.h>
	int main ( )
	{
	int x, y, z, s;
	for (x=1; x<=100; x++)
	for (y=1; y<=100; y++)
	for (z=1; z<=100; z++)
	if (x + y + z = = 100)
	{ cout << “x=”<<x;
	cout << “y=”<<y;
	cout << “z=”<<z; }
	return 0;
	}

continue operatori

Bu operator yordamida sikl parametrining biror qiymatida hisoblashni to‘xtatib, keyingi qiymatida hisoblashni davom ettirish mumkin. Masalan: y = 2x funksiyasini x € [1,18] oraliqda h=1 qiymat bilan hisoblash kerak, lekin x=6 yoki x=13 qiymatlarida hisoblashni bajarmaslik kerak.

```
#include <iostream.h>
int main ( )
{ int x, y;
for (x=1; x<=18; x++)
 { if (( x = = 6) || (x = = 13)) continue;
  y = 2*x; cout << “x=”<< x << “  y=” << y << endl;
 }  }
```

2 - misol. 10 ta ketma-ket kiritiladigan butun musbat sonlar yig‘indisini hisoblash dasturini tuzing. Agar son manfiy bo‘lsa, yig‘indiga qo‘shmaslik kerak.

```
#include <iostream.h>
int main ( )
{   int x, i, s=0;
   for ( i =1; i <=10; i ++)
   { cin >> x;   if ( x < 0)  continue;
   s = s + x; }          // s +=x deb yozsa ham bo‘ladi.
   cout << “s=”<< s << endl;    }
```

3-misol. $Y = x^n$ funksiyasini rekurrent formula orqali hisoblash dasturini tuzing. Bu yerda n - butun son, x – ixtiyoriy haqiqiy son.

```
#include <iostream.h>
#include <conio.h>
int main ( )
{ float x=2.56, y=1;
for ( int n=1; n<=10; n++)
y = y * x;              // y * = x;
cout <<"y="<<y<<endl;
getch ( );  }
```

4-misol. $Y= \sum_{n=1}^{8} \frac{x^2}{n!} = x^2 + \frac{x^2}{1*2} + \frac{x^2}{1*2*3} + ... + \frac{x^2}{1*2*3*4*5*6*7*8}$

x - ixtiyoriy haqiqiy son.

```
#include <iostream.h>
#include <conio.h>
int main ( )
{ float x=3.75, y=0; long p=1;
for ( int n=1; n<=8; n++)
{ p = p * n; y = y + x*x / p; }
cout << "y="<<y<<endl;
getch ( );
}
```

5-misol. $S=\cos x + \frac{\cos 2x}{2} + \frac{\cos 3x}{3} + ... + \frac{\cos nx}{n}$; bu yerda $\frac{\pi}{5} \le x \le \frac{9\pi}{5}$ n=10.

```
#include <iostream.h>
#include <conio.h>
#include <math.h>
 int main ( )
{ float a, b, h, x, s, pi=3.14;
a = pi / 5;  b=9 * pi / 5; h=(b-a) / 10;
x = a; cout. precision (3);
while ( x<=b )
{ s = 0;
for ( int n=1; n<=10; n++)
s = s + cos(n*x) / n;
```

```
cout <<“s= ”<<s<<endl;
x = x + h;
}
getch ( ); }
```

6-misol. Boy bilan olim bahslashibdilar. Olim boyga har kuni (30 kun) 100000 so‘m beradigan bo‘libdi. Boy esa olimga 1-kun 1 tiyin, 2-kun 2 tiyin, 3-kun 4 tiyin, 4-kun 8 tiyin va h.k. pul beradigan bo‘libdi. Bahsda kim yutadi? Dasturini tuzing.

Olim → 30*100000 = 3000000 so‘m

Boy → $\sum_{i=0}^{30} 2^i$ sum → 10737418 so‘m

```
#include <iostream.h>
#include <conio.h>
#include <math.h>
int main ( )
{ int s, s1=1, k = 0;
for ( int i=1; i<=30; i++)
{ k = k + s1;
s1 = s1*2; }
k = k / 100;
s = 30*100000;
cout <<”boy olimga beradi:”<<k<<endl;
cout <<”olim boyga beradi:” << s << endl;
getch ( );
}
```

Muhokama savollari

1. Oldshartli takrorlanish operatori.
2. So‘ngshartli takrorlanish operatori.
3. Parametrli takrorlanish oyeratori.
4. Continue, break operatorlari.

Nazorat savollari

1. Sharti avval tekshiriladigan takrorlanish.

2. Sharti keyin tekshiriladigan takrorlanish.
3. Parametrli takrorlanish.
4. Dasturda takrorlanishlarni tashkil etish.
5. Takrorlanuvchi dastur nima?
6. Murakkab takrorlanishlar.
7. continue operatori.
8. return operatori.

2.6 Massivlar va ularning o‘lchamlari

Umumiy nomga ega va tartiblangan kattaliklarning chekli to‘plamiga massiv deyiladi. Massiv – indeksli o‘zgaruvchi degan ma’noni bildiradi. Massiv elementlarining toifasi bazaviy toifa deb yuritiladi. Masalan: a = (a_1, a_2, …. , a_{10}); bu yerda: a – massiv nomi, a_1, a_2, …. , a_{10} - massiv elementlari, 1,2,..10 – element indekslari. Indeks o‘zgaruvchilarning tartiblangan o‘rnini bildiradi va u C/C++ tilida to‘rtburchak -[ ] qavslarda keltiriladi. Massiv ta’riflanganda uning toifasi, nomi va indekslari ko‘rsatiladi. Misol:

int a[5]; char ww[20]; double f [100];

Massiv elementlarining indekslari doim 0 dan boshlanadi, demak a massivda a[0],a[1],a[2],a[3],a[4] element, ww massivida ww[0], ww[1], … ww[19] element, f massivida f[0],f[1],f[2]…f[99] element bor.

Massiv elementlariga son qiymatlarini berish usullari.

1. Massivlar ta’riflanganda ular bevosita inisializasiya qilinishi mumkin. Masalan: float c[4] = {1, 0.1, -45, 7.23}; Bu yozuvni quyidagicha ham yozsa bo‘ladi: float c[] = {1, 0.1, -45, 7.23}; demak agar massiv chegarasi ko‘rsatilamagan bo‘lsa, son qiymatlarga qarab aniqlanishi ham mumkin. Massiv chegarasi ko‘rsatilgan, lekin unga beriladigan son qiymatlar kam bo‘lishi ham mumkin. U holda qolgan qiymatlar aniqlanmagan deb qaraladi. Masalan: float c[4]

= {1.56, 7.23}; , ya’ni s[1]=1.56, c[2]=7.23 , ya’ni qolgan 2 tasi aniqlanmagan deyiladi. Lekin son qiymatlari ko‘p bo‘lishi mumkin emas.

Masalan: float c[4] = {1, 0.1, -45, 7.23, -8.96, 7.78};

2. Kiritish operatori yordamida son qiymatlarni aniqlashtirish.

Masalan:

```
#include <iostream.h>
int main ( )
{ int a[10];
for (int i=0; i<10; i++)
cin >> a[i];   }
```

dastur bu usulda tuzilganda massiv elementlari klaviaturadan kiritiladi.

3. Massiv elementlarining son qiymatlari const orqali ham ko‘rsatilishi mumkin, bu holda ularning son qiymatlarini keyin o‘zgartirib bo‘lmaydi.

4. Massiv elementlarining son qiymatlarini chiqarish ham takrorlanish operatori yordamida bajariladi.

Misollar ko‘rib o‘tamiz.

1-misol. Massiv elementlaridan musbatlarining soni va summasini hisoblash dasturi.

```
      #include <iostream.h>
      #include <conio.h>
      int main ( )
{int x[ ]={1,2,56,78,-7,-45,34,12,9,-1};
      int s=0, n=0; clrscr ( );
      for (int i = 0; i < 10; i++)
      { if (x[i] < 0) continue;
      s = s + x[i];  n++;}
cout << “s=”<< s << “n=”<<n<<endl;
      getch ( );
      }
```

yoki

```
int x[10], s=0, i, n=0;
for (i = 0; i < 10; i++)
cin >> x[i];
for (i=0; i<10; i++)
{ if (x[i] < 0) continue;
s + = x[i]; n++ ;
}
cout << “s=”<< s; }
```

2-misol. 10 ta elementdan iborat massivning eng katta, eng kichik elementlarini va ularning o‘rta qiymatini hisoblash dasturi.

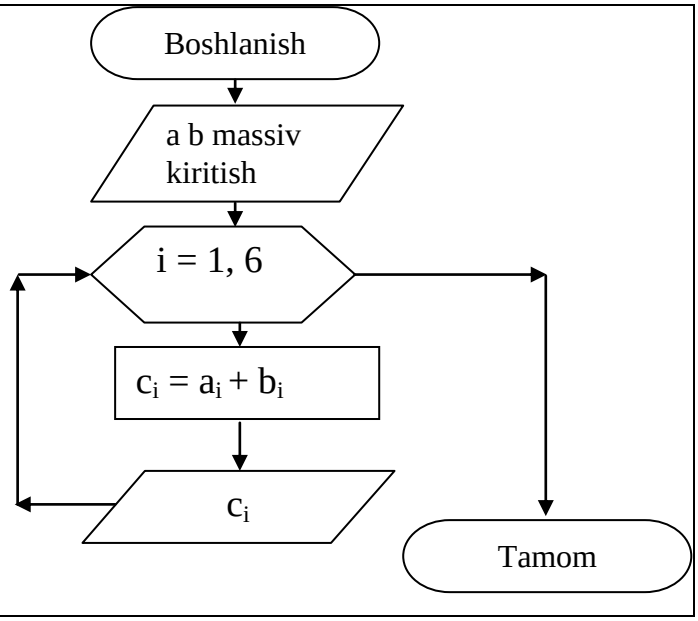
```
#include <iostream.h>
```

```
#include <conio.h>
 int main ( )
{
float  x[] = {1,2.23,5.6,-78,-7,-45.12,34.0,12,9,-1};
float s, min, max; int i; clrscr ( );
min = x[0];  max=x[0];
for (i=0; i<10; i++)
{ if (min > x[i])  min = x[i];
  if (max < x[i])  max = x[i]; }
s = (min + max) / 2;
cout << “min=”<<min<<endl;
cout <<”max=”<<max<<endl;
cout <<”o‘rta qiymat=”<<s<<endl;
getch ( );
 }
```

Izoh: for operatorida i=1 deb olinsa ham bo‘ladi, u holda n nechta bo‘lsa, shuncha element olinadi. Agar i=0 deb olinsa, n-1 element olinadi.

1-misol. 6 ta elementdan iborat a va b massivlari berilgan. Ularni o‘zaro qo‘shib yangi c massivini hosil qiling.

```
#include <iostream.h>
#include <conio.h>
int main ( )
{ int a[6] = {1,2,3,4,5,6},
b[6] = {7,8,9,10,11,12}, c[6];
for (int i=0; i<6; i++)
{ c[i] = a[i] + b[i];
cout <<”c=”<<c[i]<<endl; }
getch ( );
}
```



2-misol. Vektorlarning skalyar ko‘paytmasini hisoblash dasturini tuzing.

$S = \sum_{i=1}^{n} x_i y_i$; n=5;

```
#include <iostream.h>
```

```
#include <conio.h>
int main ( )
{ int x[5]={1, 2, 3, 4, 5}, y[5]={6, 7, 8, 9, 10}, s=0;
for (int i=0; i < 5; i++)
s = s + x[i] * y[i];
cout << ”s= ” << s << endl;
getch ( ); }
```

3-misol. Bazaviy toifasi haqiqiy bo‘lgan 10 ta elementli A massivi berilgan. Juft indeksli elementlardan alohida, toq indeksli elementlardan alohida massiv hosil qiling.

```
#include <iostream.h>
#include <conio.h>
int main ( )
{ float a[10], b[5], c[5];
for (int i=0; i<10; i++)
cin >> a[i];
for (int i=0; i<5; i++)
{ c[i] = a[2*i +1];
  b[i] = a[2*i];
cout << ”c=” << c[i];
cout << ”b=” << b[i] <<endl; }
getch ( );
}
```

4-misol. A va B massivlari berilgan. Yangi C massivini quyidagicha hosil qiling: A massivining elementlari yangi massivning toq elementlari, B massivining elementlari esa yangi massivning juft elementlarini tashkil etadi.

```
#include <iostream.h>
#include <conio.h>
int main ( )
{ float a[5], b[5], c[10]; int i;
for ( i=0; i<5; i++)
cin >> a[i] >> b[i];
for ( i=0; i< 5; i++)
{ c[2*i+1] = b[i];
  c[2*i] = a[i]; }
```

```
for (i = 0; i < 10; i++)
  cout << ”c[” << i << ”]= ” << c[i] <<endl ;
getch ( );
}
```

Muhokama savollari

1. Bir o‘lchamli massivlarni tavsiflash. ir o‘lchamli massivlarga ishlov berish.
2. Massiv elementlarini kiritish.
3. Massiv elementlariga ishlov berish, yangi element kiritish.
4. Massiv elementlarini o‘zgartirish.

Nazorat savollari

1. Massiv. Massiv tushunchasi.
2. Indeksli o‘zgaruvchilar. Masivlarni kiritish va chiqarish.
3. Ikki o‘lchamli massivlar.
4. Massivning eng katta yoki eng kichik elementini topish algoritmi.
5. Turli o‘lchamli vektor uzunliklarini hisoblash algoritmi.

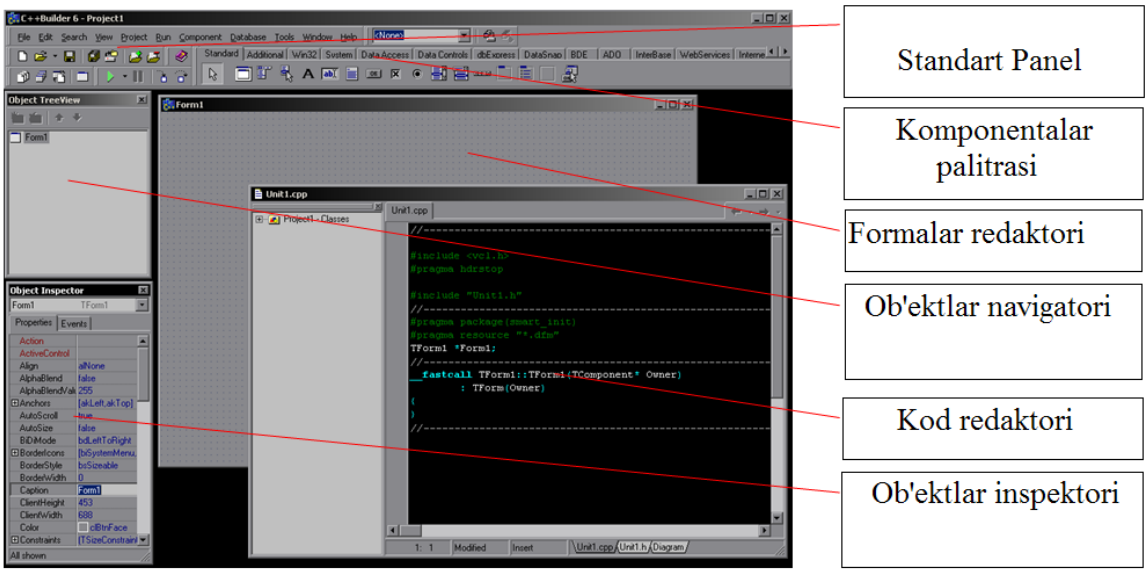
2.7 C++ BUILDER muhiti

Ishlab chiqishning integratsiyalashgan muhiti Komponentalar palitrasini birlashtiradi. Shakllar muharriri, Kod muharriri, Ob’yektlar noziri, Ob’yektlar xazinasi - bular hammasi kod va zahiralar ustidan to‘liq nazoratni ta’minlovchi dasturiy ilovalarni tez ishlab chiqish instrumetlari (2.1-rasm).

Komponentlar palitrasi ilovalarni qurishda taklif qilinadigan 100 dan ortiq takroran qo‘llanadigan komponentlardan iborat.

- *Shakllar muharriri* dasturning foydalanuvchi bilan interfeysini yaratish uchun mo‘ljallangan.

- *Kod muharriri* dastur matnini, xususan, voqealarga ishlov berish funktsiyalarini yozish uchun mo‘ljallangan.



2.1-rasm.Ishlab chiqish muhitining tuzilishi

- *Ob’yektlar noziri* qotib qolgan chigal dasturlash zaruratisiz Ob’yektlar xususiyatlarini vizual o‘rnatish imkonini beradi hamda shunday voqealarni o‘z ichiga oladiki, bu voqealarni ularning paydo bo‘lishiga nisbatan Ob’yektlar reaksiyasi kodlari bilan bog‘lash mumkin bo‘ladi.
- *Ob’yektlar xazinasi* ma’lumotlarning shakl va modullari kabi Ob’yektlarga ega bo‘lib, ular ishlab chiqishda muvaqqat sarflarni kamaytirish maqsadida ko‘plab ilovalar bilan bo‘linadi.

C++Builder ilovalarni qurishning vizual metodikasini Komponentlar palitrasidan kerakli boshqarish elementlarini tanlab olish vositasida joriy etadi. Har bir komponenta (masalan, tugma) bilan ushbu komponenta turini va xulq-atvorini o‘zgartiradigan xususiyatlarga bog‘liq bo‘ladi. Har qanday komponenta ushbu komponentaning turli xildagi ta’sirlarga reaksiyasini (munosabatini) aniqlab beradigan voqealar seriyasini keltirib chiqarishi mumkin. Bundan keyin => belgilari siz C++Builder muhitida amalga oshiradigan xatti-harakatlarni bildiradi.

=>C++Builder ni chaqiring va bosh menyudagi File | New Application komandasi bo‘yicha yangi ilovalar ustida ishlashni boshlang.

=>sichqonchani Komponentalar palitrasining qo‘shimcha ilovalari ustida bosib, foydalanuvchi ish ko‘radigan dastur interfeysi elementlarining mavjud assortimentini ko‘rib chiqing.

Palitraning bir qo‘shimcha ilovasidan ikkinchisiga o‘tib, kirish mumkin bo‘lgan komponentlar to‘plami o‘zgarayotganining guvohi bo‘lishimiz mumkin. Sichqoncha kursori komponentlar belgisi ustida to‘xtaganda, aytib turish nomi paydo bo‘ladi. Agar F1 klavishasini bossak, tizimning ma’lumotnomalar xizmati tanlab olingan komponenta haqida to‘liq ma’lumot chiqarib beradi.

Vizual loyihalash: Bizning birinchi ilovamiz bolalarning “O‘nta negr bolasi” sanoq she’rini generatsiya qiladi. Dastlabki versiyada faqat uchta Ob’yekt kerak bo‘ladi: ro‘yxat, tahrir qilish maydoni va tugma. Komponentalarni loyihalash shakliga olib o‘tamiz hamda ilovani asta-sekin rivojlantira boshlaymiz. Tashib olib o‘tish metodi (drag-and-drop) quyidagilardan iborat: sichqoncha tugmasini tanlab olingan komponenta ustida bosing, kursorni shaklning to‘g‘ri kelgan yeriga o‘tkazing, keyin esa sichqoncha tugmasini yana bosing. Boshida faqat “standart” Palitra Komponentlari bilan cheklanamiz:

- Standard qo‘shimcha ilovani tanlab oling.
- Ro‘yxat komponentasini ListBox shakliga olib o‘ting.
- Tahrir qilinayotgan kiritish maydoni EditBox ni olib o‘ting.
- Button tugmasi komponentasini olib o‘ting.
- Komponentalarni o‘zingizning ilovangizdagi darchada qanday ko‘rmoqchi bo‘lsangiz, shunday joylashtiring va o‘lchamlarini shunday o‘zgartiring.

Ob’yekt noziri yordamida komponentalar xususiyatlarining boshlang‘ich qiymatlarini aniqlang. Items ro‘yxatining xususiyatlar qiymatlari katagida tugmani bosing, ochilgan muharrir darchasida she’rning dastlabki 7 satrini kiriting. Shakl va tugmaning Caption xususiyatida ularning ma’noli nomlarini ko‘rsating (mos ravishda, “O‘nta negr bolasi” va “Natija”). Tahrir qilish maydonining Text xususiyatiga natijani aytib berish satrini kiriting (“To‘qqizta negr bolasi”).

Endi Kod muharririga ulanish hamda, avval qabul qilinganidek, C++ tilidagi har qanday dasturni yozish mumkin, shu jumladan, ANSI/ISO standartining so‘nggi kengaytmalarini ham. Biroq, avval ilovalarni tez ishlab chiqishning yangi vositalari hamda C++ Builder da mavjud bo‘lgan qo‘shimcha komponentalar atributlaridan foydalanishga harakat qilib ko‘ramiz.

Ikki yo‘nalishli ishlanma texnologiyasi

Loyihaviy shablonlarni qo‘llash

Xususiyatlar, metodlar va voqealar:[4]

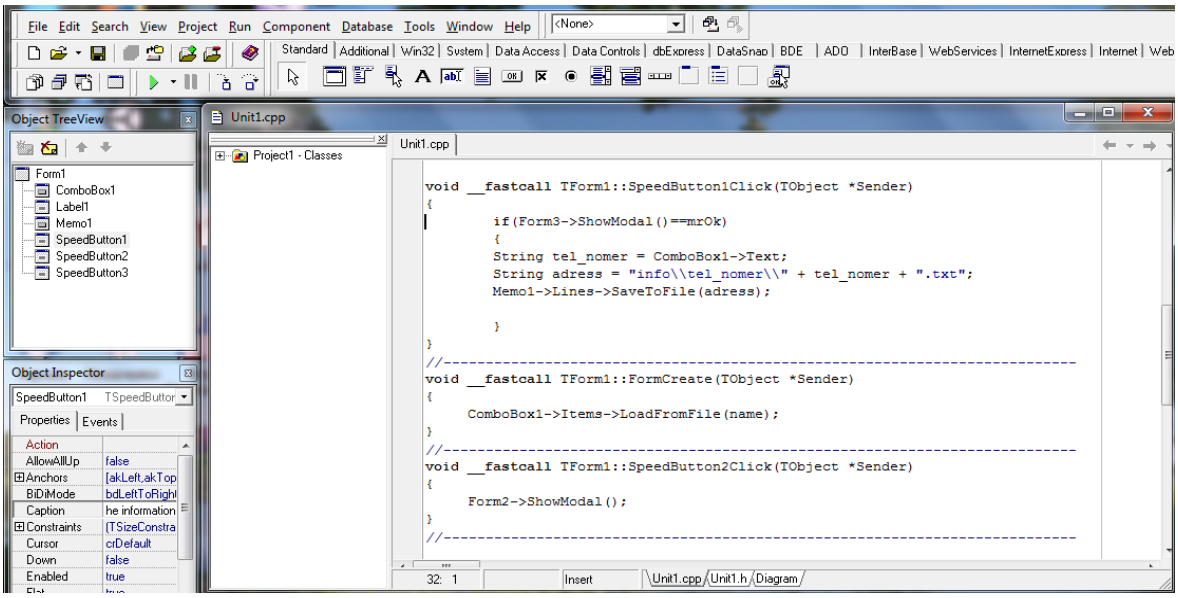
Ilovalarning tez ishlab chiqilishi Ob’yektga yo‘naltirilgan dasturlash doirasida xususiyatlar, metodlar va voqealarning qo‘llab-quvvatlanishini bildiradi. *Xususiyatlar* komponentalarning nomlar, matnli aytib berishlar yoki ma’lumotlar manbalari kabi turli xildagi tavsiflarini osongina o‘rnatish imkonini beradi. *Metodlar* (a’zo-funksiyalar) komponentadagi Ob’yekt ustida ma’lum operatsiyalarni amalga oshiradi. Bunday operatsiyalar jumlasida qayta tiklash yoki multimedia qurilmasini qayta ulash kabi murakkab operatsiyalarni ham ko‘rsatish mumkin. Bundan tashqari voqealar komponentalar holatlarida sodir bo‘ladigan ayrim o‘ziga xos o‘zgarishlar paytida ham yuzaga kelishi mumkin. Bunday o‘ziga xos o‘zgarishlar qatorida ma’lumotlar bazasiga kirishning interfeysli elementlarida ma’lumotlarni yangilashni ko‘rsatib o‘tish kifoya. Xususiyatlar, metodlar va voqealar, birgalikda ish olib borar ekan, ular Windows uchun ishonchli ilovalarni intuitiv tarzda dasturlash muhiti - RAD ni hosil qiladi.

=>Tanlangan Ob’yekt bilan assotsiatsiyalanadigan (birgalikda yodga olinadigan) voqealarni ko‘rish uchun, Ob’yektlar nozirida voqealar (Events) qo‘shimcha ilovasini ko‘rsating.

=>O‘zingiz shaklga joylashtirgan tugma komponentasini sichqoncha bilan ikki marta bosing.

=>Ochilgan Kod muharriri darchasida kursor Button1Click funksiyasi tanasiga instruktsiyalarni kiritish uchun pozitsiyani ko‘rsatadi. Bu funksiya esa tugmani bosishda yuzaga keladigan OnClick voqeasiga ishlov berish uchun yo‘naltirilgan.

2.2-rasmda oddiy kod ko‘rsatilgan bo‘lib, u “Add the number” tugmasi bosilishiga javoban Form3 ochiq turgan holda, info direktoriyasining ichidagi tel_nomer papkasidagi telefon nomerlarini matn ko‘rinishiga o‘tkazib, Memo1 maydonida akslanishini ta’minlaydi. Bu esa SpeedButton1 tugmasini bosish bilan yuzaga keladigan voqeaga ishlov berish funksiyasining chaqirilishlari o‘rtasida o‘zining joriy qiymatini saqlaydi.



2.2-rasm. Kod muharriri bajarilayotgan modul matnining Unit1.cpp faylida kiritilishi va tahrir qilinishini ta’minlaydi

Birinchi versiyali ilovani loyihalash bosqichi shuning bilan tugallanadi va ishchi dasturni yaratishga kirishish mumkin bo‘ladi.

=>Run | Run bosh menyusi komandasi bilan ilovani kompilyatsiya qilish (ko‘chirish) va yig‘ish jarayonini ishga tushirib yuboring [4].

=>Dastur chaqirilgach, bir necha marta “Natija” tugmasini bosing.

Ikki yo‘nalishli ishlanma texnologiyasi:

C++ Builder muhiti dasturchi va uning kodi o‘rtasida hech qanday to‘siqlarni qo‘ymaydi. Two-Way Tools ikki yo‘nalishli ishlanma texnologiyasi vizual loyihalash instrumentlari va Kod muharriri o‘rtasida moslashuvchan, integrallashgan va sinxronlashtirilgan o‘zaro aloqa vositasida sizning kodingiz ustidan nazoratni taminlaydi. Ikki yo‘nalishli ishlanma instrumentlari qanday amal qilishini kuzatib borish uchun, quyidagi operatsiyalarni bajaring:

=>Sichqonchaning o‘ng tugmasini bosib, Kod muharririning kontekstli menyusini oching, keyin Swap Cpp/Hdr Files operatsiyasi yordamida Unit1.h. e’lonlar fayliga ulaning.

=> Instrumentlarning ekrandagi aksini shunday tashkil qilingki, bunda Kod muharriri darchasida bir paytning o‘zida loyihalanayotgan shakl va Unit.h fayli ko‘rinsin.

=>OK Button tugmasining yana bitta komponentasini shaklga olib o‘ting. Tugmaning Caption xususiyatida uning ma’noli nomi “Yangi band” deb ko‘rsating.

Quyidagilarni kuzatib boring: siz tugmani shaklga olib o‘tishingiz bilan, shu ondayoq Unit1.h faylida Button2 Ob’yektining e’loni paydo bo‘lishi kerak, OnClick voqeaning aniqlanishi esa ushbu vorqeaning qayta ishlovchisi bo‘lgan Button2Click metodining e’lon qilinishini generatsiyalaydi. Shaklni loyihalashning va kodni avtomatik generatsiyalash jarayonlarining mana shunday sinxronlashtirilishi C++ ilovaning vizual ishlanmasini haqiqatan ham tezlashtiradi va shuning bilan birga dasturning dastlabki matni ustidan nazoratni to‘la saqlab qoladi.

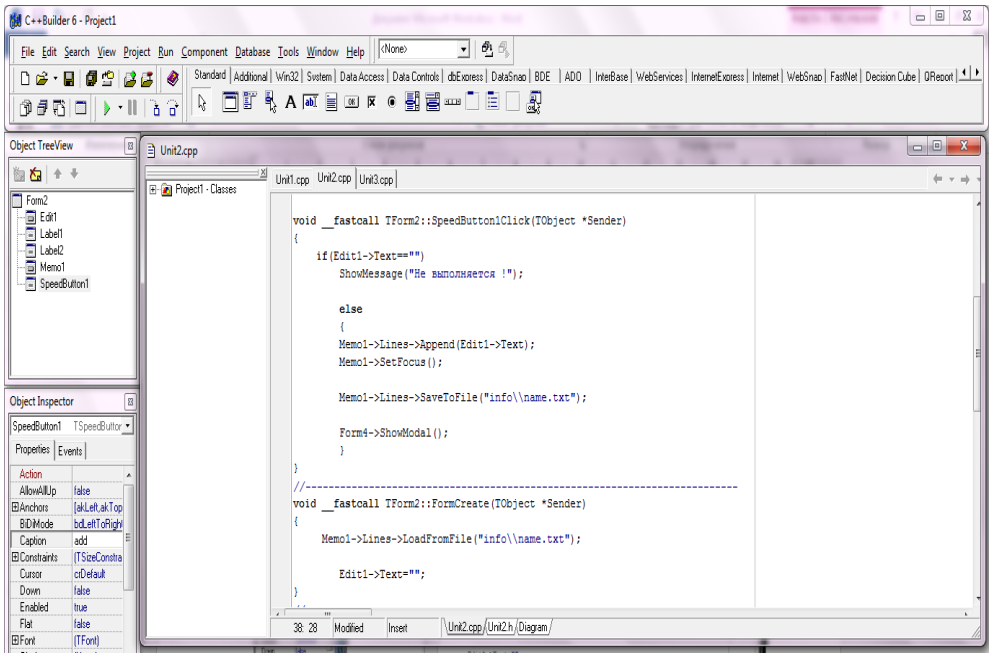
O‘zimizning birinchi ilovamizni ishlashda yana bir qadam qo‘yamiz - uni she’r bandini avtomatik tarzda generatsiyalashga majbur qilamiz. Buning uchun OnClick voqeasi ishlanmasining funksiyasini “Yangi band” tugmasini bosib, mazmun bilan to‘ldirishga to‘g‘ri keladi [4].

2.3-rasmda oddiy kod ko‘rsatilgan bo‘lib, u “Add” tugmasining navbatdagi bosilishiga javoban kiritilgan raqamlarni yuqorida keltirilgan papkalarda saqlanishini ta’minlaydi.

C++ Builder har bir ilova bilan yashirin nomlari quyidagicha bo’lgan uchta dastlabki faylni eslatishini yodda saqlab qolish kerak:

- Unit1.cpp ilovangizning bajarilayotgan ishga tushirish kodini saqlaydi. Aynan shu yerda siz foydalanuvchining komponentalar Ob’yektlariga ta’siri paytidagi dastur reaksiyasiga javob beradigan voqealarning qayta ishlatgichlarini yozib qo’yasiz.

- Unit1.h barcha Ob’yektlar va ularning konstruktorlarining e’lonlariga ega. Voqealarni qayta ishlash funktsiyalari e’lonlaridagi _fastcall kalit-so’zga e’tibor abering (C++ Builder bu funksiyalarni avtomatik tarzda generatsiya qiladi). _fastcall tufayli parametrlar stek orqali emas, balki markaziy protsessor registrlari orqali uzatiladi. Voqealarni qayta ishlatgichlarning chaqirishlari tez-tez ro’y berib turadi, Shuning uchun stek xotirasidan parametrlarni tanlab olishga sarflanadigan vaqtning tejalishi ancha sezilarli natijalarni beradi. C++ Builder kompilyatsiya qiladigan va to’playdigan ilovalarning yuqori darajada tez harakatlanishining sabablaridan biri ham shu yerda yashiringan.



2.3-rasm. Unit2.cpp faylida voqeaning yangi qayta ishlatgichi

- Project1.cpp ilovada mujassamlangan barcha Ob’yektlarga xizmat ko’rsatadi. Har qanday yangi shakl, dasturiy modul yoki ma’lumotlar moduli avtomatik tarzda loyihaviy faylga kiritiladi. Siz bosh menyu komandasi - View | Project Source yordamida yoki Loyiha Administratorining kontekstli menyusidan shu nomdagi opsiyani tanlab olib, Kod muharriri darchasida loyihaviy fayl dastlabki matnining

mazmunini ko’rib chiqishingiz mumkin. Hech qachon loyihaviy faylni qo’lda tahrir qilmang!

Balki siz, birinchi ilova ishlanmasini tugatib, dastlabki fayllarni keyingi seans uchun saqlab qolishni xoxlarsiz. Buning uchun quyidagi xatti-harakatlardan birini bajarishingiz kerak:

=>File | Save All komandasi ilovaning hamma dastlabki fayllarini saqlaydi.

File | Save komandasi dasturiy modulning ikkala komandasini saqlaydi, File | Save As komandasi esa ularga yangi nom berishga ruxsat etadi.

File | Save Project As komandasi, fayllarning joriy nomlaridan foydalanib, loyihaviy fayl tarkibiy qismlarining hammasidagi o’zgarishlarni saqlaydi.

Ikki yo’nalishli ishlanma texnologiyasi. Loyihaviy shablonlarni qo’llash

Ob’yektlar xazinasidagi tayyor loyihaviy shablonlardan foydalanar ekansiz, siz dasturni ishlab chiqishda ko’pchilik ilovalar uchun toifalik bo’lgan operatsiyalarni chetlab o’tish imkoniyatiga ega bo’lasiz. Bu qanday operatsiyalar dersiz. Bular, masalan, menyu va tez chaqirib olish tugmalari panelini tuzish, standart chaqirishlar dialogi va fayllarni tuzishni tashkil etish bilan bog’liq operatsiyalardir. Siz shablonga kiritgan o’zgartirishlar xuddi shu loyihaviy shablondan boshqa ishlab chiquvchilarning foydalanishiga ta’sir qilmaydi.

Ko’p hujjatli interfeys (MDI) rejimida ishlash uchun loyihaviy shablon asosida ilova prototoifasini yaratish uchun quyidagi xatti-harakatlarni amalga oshiring:

Filtrlar muharriri darchasida TOpenDialog komponentasining Filter xususiyati qiymatlari ustunida matnli hujjatlar fayllarining nomlari va kengaytmalarini ko’rsating.

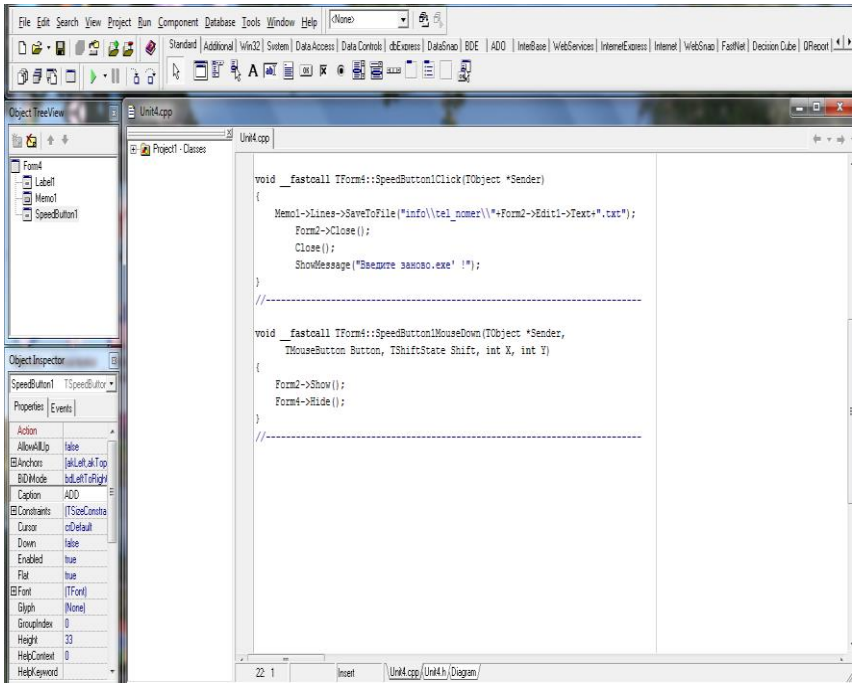
Agar siz shunday ilovani kompilyatsiya qilib, to’play olsangiz, bu holda u faqat MDI rejimida darchalar bilan amallar bajara olishini hamda, darchani tanlab olingan fayllarning matnli mazmuni bilan to’ldirmay turib, fayllarni ochish dialogini chaqirib olish “mumkinligi”ni ko’rishingiz mumkin. Ya’ni prototip nofunksional va amaliy jihatdan befoyda bo’lib qoldi. Ilova qandaydir ongli xulq-atvorga ega bo’lishi uchun, quyidagi xatti-harakatlarni bajaring:

Bosh menyudan View | Forms komandasini bering va ro’yxatdan MDIChild nomli shaklni tanlab oling.

Memo tahrir qilish maydonining ko’p satrli komponentasini Palitraning Standart qo’shimcha ilovasidan sho’ba shaklga olib o’ting.

Lines xususiyatli satriy muharrirni tugmani bosish bilan chaqirib olib, TMemo komponentasining tahrir maydonini tozalang. Tahrir maydoni sho‘ba darchasining hammasini egallashi uchun, Align xususiyatli alClient qiymatini o‘rnating. Uzun matnli fayllarni ko‘rib chiqishni osonlashtirish maqsadida, ScrolBars xususiyatli Ssboth qiymatini o‘rnating.

=>Bosh shaklni sichqoncha yordamida faollashtirib, yana unga qayting hamda ilovalar menyusidan File|Open komandasini tanlab oling.



2.4-rasm. Unit4.cpp faylida sho’ba darcha yuklanishining amalga oshirilishi

=>Kod muharriri darchasida kursor menyuning tegishli elementini tanlashda yuzaga keladigan OnClick voqeasining qayta ishlatgichiga yo’riqnomani kiritish uchun kerakli pozitsiyani ko’rsatib beradi. C++ Builder TOpenDialog bosh shakli (komponentalar Palitrasining Dialogs kiritmasidan) komponentasi uchun ushbu funksiyaning e’lonini avtomatik tarzda generatsiya qiladi.

2.4-rasmda shu voqeaning qayta ishlatgichi bo’lgan FileOpenltemClick funksiyasi tanasini tashkil qiluvchi zarur yo’riqnomalar ko’rsatilgan.

Ajratib olingan yo’riqnoma tel_nomer nomli ochiq matnli faylning ichidagilari bilan yuklatadi.

Bu faylning ishlanishi hali tugallanganicha yo‘q, albatta. Siz uni kompilyatsiya qilib, to‘plab bo‘lsangiz, bir paytning o‘zida bir necha darchalardagi matnli fayllarni tahrir qila olasiz. Biroq natija beruvchi fayllarning saqlanishi hozircha ko‘zda tutilgan emas - o‘quvchining o‘zi File [Save va File | Save As menyulari komandalari uchun osongina kod yozib oladi.

Ilovani mantiqan eng sodda matnli muharrirga aylantirish uchun bu Edit nomli bosh menyu elementining tushib qoluvchi ro‘yxatiga qidirish va almashtirish komandalarini qo‘shish kerak.

Muhokama savollari

1. Vizual loyihalash
2. Xususiyatlar, metodlar va voqealar
3. Ikki yo‘nalishli ishlanma texnologiyasi
4. Loyihaviy shablonlarni qo‘llash

Nazorat savollari

1. Komponentlar Palitrasi haqida tushuncha.
2. Shakllar Muharriri haqida tushuncha.
3. Kod Muharriri haqida tushuncha.
4. Ob’ektlar Noziri haqida tushuncha.
5. Ob’ektlar Xazinasi haqida tushuncha.
6. Vizual loyihalash haqida tushuncha.
7. Ikki yo‘nalishli ishlanma texnologiyasi haqida tushuncha.
8. Loyihalash shablonlari haqida tushuncha.
9. Two Way Tools texnologiyasi haqida tushuncha.

2.8 Ko‘p o‘lchamli massivlar

C++ algoritmik tilida faqat bir o‘lchamli massivlar bilan emas, balki ko‘p o‘lchamli massivlar bilan ham ishlash mumkin. Agar massiv o‘z navbatida yana massivdan iborat bo‘lsa, demak ikki o‘lchamli massiv, ya’ni matrisa deyiladi. Massivlarning o‘lchovi kompyuterda ishlashga to‘sqinlik qilmaydi, chunki ular xotirada chiziqli ketma-ket elementlar sifatida saqlanadi. Ko‘p o‘lchamli massivlarni xuddi 1 o‘lchamli massivga o‘xshab e’lon qilinadi, faqat indeks toifasi sifatida massivning satrlari (qatorlari) va ustunlari toifasi ko‘rsatiladi va ular alohida [ ][ ] qavslarda ko‘rsatiladi. Masalan: A nomli butun sonlardan iborat 2 o‘lchamli massiv berilgan bo‘lsa va satrlar soni n ta, ustunlar soni m ta bo‘lsa: int a[n][m]

Ikki o‘lchovli massiv elementlarini kiritish-chiqarish, ular ustida amallar bajarish ichma-ich joylashgan parametrli sikllar ichida bo‘ladi, ya’ni 1-sikl satrlar uchun, 2-sikl ustunlar uchun. Masalan:

```
for ( i=0; i<3; i++)
for ( j=0; j<3; j++)
cin >>a[i][j];
```

Agar ularni klaviaturadan kiritish kerak bo‘lsa, ya’ni cin operatori yordamida tashkil etilsa, quyidagicha kiritiladi:

1 2 3

4 5 6

7 8 9

undan tashqari massiv elementlarini e’lon qilish bilan birga ularni inisalizasiya ham qilish mumkin:

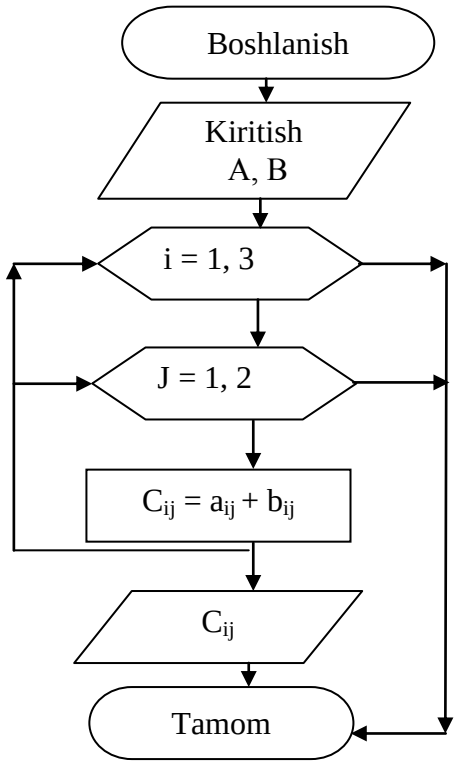
```
int  a[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
```

Natijalar chiroyli ko‘rinishda bo‘lishi uchun chiqarish operatorini quyidagicha qilib tashkil etish kerak:

```
for (int i=0; i<3; i++)
{ for (int j=0; j<3; j++)
```

```
 cout <<”a[“<<i<<”,”<<j<<”]=”<<a[i][j];
cout <<endl; }
getch ( );
}
```

1-misol. A va B matrisalari berilgan. Quyidagi formula orqali yangi C matrisasini hosil qiling: $C_{ij} = A_{ij} + B_{ij}$; bu yerda i=1,3; j=1,2;




```
#include <iostream.h>
#include <conio.h>
int main ( )
{ float a[3][2]={{24.3,-4.15},
{0,18.4},{-8.8,-15.75}},
b[3][2]={{0.1,-4.8},{6.8,7.1},{-
2.8,0.40}};
float c[3][2];
int i, j;
for (i = 0; i < 3; i++)
{ for (j = 0; j < 3; j++)
{ c[i][j] = a[i][j] + b[i][j];
cout <<”c[“<<i<<”,”<<j<<”]=”<<c[i][j]; }
cout <<endl; }
getch ( );
}
```

Massiv ementlariga son qiymat berishda kompyuter xotirasidagi tasodifiy butun sonlardan foydalanish ham mumkin. Buning uchun standart kutubxonaning rand ( ) funksiyasini ishga tushirish kerak. rand ( ) funksiyasi yordamida 0 ÷ 32767 oraliqdagi ixtiyoriy sonlarni olish mumkin. Bu qiymatlar umuman tasodifiydir. (psevdo – tasodifiy degani).

Agar dastur qayta-qayta ishlatilsa, ayni tasodifiy qiymatlar takrorlanaveradi. Ularni yangi tasodifiy qiymatlar qilish uchun srand ( ) funksiyasini dasturda bir marta e’lon qilish kerak. Dastur ishlashi jarayonida ehtiyojga qarab rand ( ) funksiyasi chaqirilaveradi. Tasodifiy qiymatlar bilan ishlash uchun <stdlib.h> faylini e’lon qilish zarur. srand ( ) funksiyasidagi qiymatni avtomatik ravishda o‘zgaradigan holatga keltirish uchun srand (time (NULL)) yozish ma’qul, shunda kompyuter ichidagi soatning qiymati time ( ) funksiyasi yordamida o‘rnatiladi va

srand ga parametr sifatida beriladi. NULL yoki 0 deb yozilsa, qiymat sekundlar ko‘rinishida beriladi. Vaqt bilan ishlash uchun <time.h> ni e’lon qilish kerak.

```
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
int main ( )
{ srand ( time (0));
int a[5], b[5], i;
for (i = 0; i < 5; i++)   a[i] = rand ( );
for (i = 0; i < 5; i++)
{ b[i] = a[i] + 64;
cout << “b=”<<b[i]<<endl; }
getch ( ); }
```

Izoh: tasodifiy sonlar ichida manfiy sonlarning ham qatnashishini ihtiyor etsak, a[i] = 1050 - rand ( ); yoki a[i] = rand ( )-1000; deb yozish ham mumkin.

2-misol. A matrisani B vektorga ko‘paytirish algoritmi.

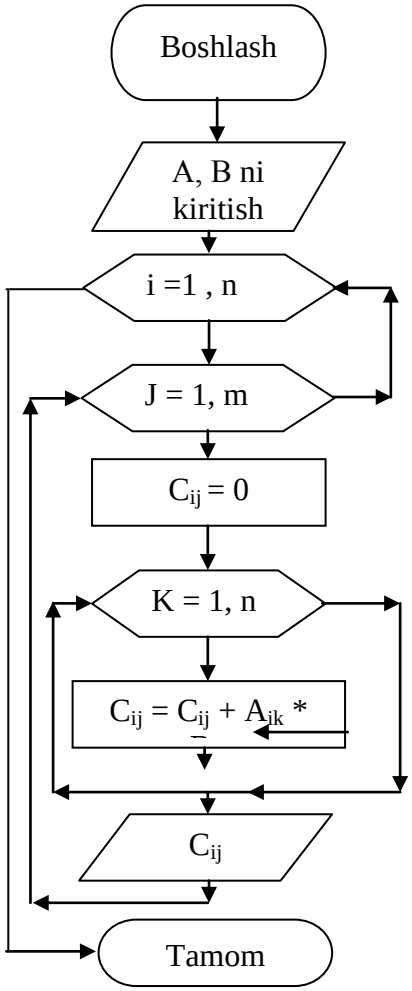
$$C_i = \sum_{\substack{i=1 \\ j=1}}^{n} a_{ij} * b_j$$

Izoh: matrisaning satrlari soni vektorning satrlariga teng bo‘lishi kerak.

Masalan:

```
#include <iostream.h>
#include <conio.h>
int main ( )
{ int a[3][3] = {{1,2,3},{4,5,6},{7,8,9}}, b[3] = {1,3,6}, c[3], i, j;
for (i=0; i<3; i++)
{ c[i] = 0;
  for (j=0; j<3; j++)
  c[i] = c[i] + a[i][j] * b[j];
  cout <<”c=”<<c[i]<<endl; }
getch ( );
}
```

3-misol. 2 ta matrisa berilgan. Ularni o‘zaro ko‘paytirib yangi matrisa hosil qiling. Bu yerda 1-matrisaning ustunlar soni 2-matrisaning satrlar soniga teng bo‘lishi kerak.




```
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
int main ( )
{ srand ( time (0));
int a[3][3], b[3][3],c[3][3], i, j, k;
for (i=0; i<3; i++)
for (j=0; j<3; j++)
a[i][j] = rand ( );
for (i=0; i<3; i++)
for (j=0; j<3; j++)
b[i][j] = rand ( );
for (i=0; i<3; i++)
{ for (j=0; j<3; j++)
{ c[i][j] = 0;
for (k=0; k<3; k++)
c[i][j] = c[i][j] + a[i][k]*b[k][j];
cout <<"c="<<c[i][j]<<"\t"; }
 cout << endl; }
getch ( );}
```

4-misol. Matrisani transponirlash dasturini tuzing. Matrisani transponirlash deb, ustun va satr elementlarini o‘zaro o‘rin almashtirishga aytiladi, ya’ni $A_{ij} = B_{ji}$

```
#include <iostream.h>
#include <conio.h>
int main ( )
{
int a[3][3] = {{1,2,3},{4,5,6},{7,8,9}},
b[3][3],
 i, j;
for ( i=0; i<3; i++)
{ for ( j=0; j<3; j++)
 {  b[i][j] = a[j][i];
  cout <<"b["<<i<<","<<j<<"]="<<b[i][j];  }
cout << endl;  }
getch ( );
}
```

```
Berilgan matrisa:
1 2 3
4 5 6
7 8 9
Hosil bo’lgan matrisa:
1 4 7
2 5 8
3 6 9
```

5-misol. 3 ta qator va 4 ta ustunga ega A matrisa berilgan. Undagi eng kichik elementni va uning indeksini topish, hamda o‘sha qatorni massiv shaklida chiqarish dasturini tuzing.

```
#include <iostream.h>
#include <conio.h>
int main ( )
{
int a[3][4] = {{1,2,3,4},{4,5,6,7},{7,8,9,10}}, i, j, k, h, min;
int b[4];
min = a[0][0];
for (i=0; i<3; i++)
for (j=0; j<4; j++)
{ if ( a[i][j] > min) { min = a[i][j]; k = i; h = j; } }
cout << “min=”<<min<<” k=”<<k<<”  h=”<<h<<endl;
for ( j=0; j<4; j++)
{ b[j] = a[k][j];
  cout <<”b=”<<b[j]; }
getch ( );
}
```

6-misol. Saralash masalasi. Massiv elementlarini o‘sib borish tartibida saralash dasturini tuzing. (pufaksimon saralash)

Avval bir o‘lchovli massiv elementlarini saralashni ko‘rib o‘tamiz.

```
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
int main ( )
{
srand (time (0));
float  a[10] , b; int i, j;
for (i = 0; i<10; i ++)
a[i] = rand( ) /33.;
for( j = 0; j<10; j++)
for ( i = 0; i < 10; i ++)
{
 if (a[i] < a[i+1])
```

```
{ b = a[i];
a[i] = a[i+1];
a[i+1] = b; }
}
cout. precision (3);
for (i = 0; i < 10; i++)
cout << a[i]<<endl;
getch ( );
}
```

Endi ikki o‘lchamli massiv elementlarini saralashni ko‘ramiz:

```
#include <iostream.h>
#include <conio.h>
int main ( )
{ float  a[3][3] = {{.....},{.....},{.....}}, b;
int i, j, k;
for ( k=0; k<3; k++)
for ( i=0; i<3; i++)
for ( j=0; j<2; j++)
{ if (a[i][j] > a[i][j+1] )
{ b = a[i][j]; a[i][j] = a[i][j+1]; a[i][j+1] = b; } }
for ( i=0; i<3; i++)
{ for ( j=0; j<3; j++)
cout <<”a=”<<a[i][j]; cout << endl; }
getch ( );
}
```

Yuqoridagi dastur saralashni qator bo‘yicha olib borish uchun mo‘ljallangan. Agar saralashni ustun bo‘yicha amalga oshirish kerak bo‘lsa, quyidagicha yozish kerak bo‘ladi:

```
for ( i=0; i< 2; i++)
for ( j=0; j<3; j++)
{ if ( a[i,j] > a[i+1, j] ) {  b:= a[i, j]; a[i, j]:= a[i+1, j]; a[i+1, j]:= b; }
```

Agar saralashni o‘sib borish tartibida amalga oshirish kerak bo‘lsa, if operatoridagi solishtirish belgisi > bo‘lishi kerak, agar kamayish tartibida kerak bo‘lsa, solishtirish belgisi < ko‘rinishida bo‘lishi kerak.

7-misol. Matrisaning izini hisoblash dasturini tuzing. Matrisaning izi deb bosh diagonal elementlarining yig‘indisiga aytiladi. Shu dasturda teskari (qo‘shimcha) diagonal elementlarining yig‘indisini ham hisoblashni ko‘rib o‘ting.

```
#include <iostream.h>
#include <conio.h>
int main ( )
{
float  a[3][3] = {{.....},{.....},{.....}}, s1=0, s2=0;
int i, j;
for ( i=0; i<3; i++)
s1 = s1 + a[i][i];
for ( i=0; i<3; i++)
for ( j=0; j<3; j++)
if ( i+j = = 2)  s2 = s2 + a[i][j];
cout <<”s1=”<<s1<<”  s2=”<< s2<< endl;
getch ( ); }
```

8-misol. Har bir hadi $a_n = \frac{n!}{(2n)!}$ formulasi orqali hisoblanadigan satr yig‘indisini 0,0001 aniqlikda hisoblash dasturini tuzing.

```
#include <iostream.h>
#include <conio.h>
int main ( )
{ int n =1; float s1 = 0, s2 = 0;
float p1 =1, p2 = 1;
while (s2 > 0.0001)
{ p1 = p1 * n;            // p1*=n;
p2 = p2 * 2*n*(2*n-1);     // p2*= 2*n*(2*n-1);
s2 = p1 / p2;
s1 = s1 + s2;             // s1+ = s2;
n ++; }
cout. precision (3);
cout << “s1=”<<s1 <<” n=” << n << endl;
}
```

Muhokama savollari

1. Ikki o‘lchamli massivlarni tavsiflash. Ikki o‘lchamli massivlarga ishlov berish.
2. Ikki o‘lchamli massiv elementlarini kiritish va chiqarish.
3. Massiv satr va ustunlariga ishlov berish.
4. Mavjud massivdan yangi massivni tashkil qilish

Nazorat savollari

1. Ikki o‘lchamli massivlar.
2. Saralash usullari. To‘g‘ri tanlash usuli.
3. Maksimal element joylashgan satr yoki ustunni o‘chirish algoritmi.
4. Matrisani matrisaga ko‘paytirish algoritmi.

2.9. Saralash algoritmlari

Umuman olganda saralashning maqsadi berilgan Ob’yektlar to‘plamini aniq bir tartibda guruhlab chiqish jarayoni ta’riflanadi. Saralashning maqsadi keyinchalik, saralangan to‘plamning qidirilayotgan elementini topishdan iborat. Bu qariyb universal, fundamental jarayon. Biz bu jarayon bilan har kuni uchrashamiz – telefon daftaridagi saralash, kitoblar sarlavhasida, kutubxonalarda, lug‘atlarda, pochtada va hokazo [6].

Xatto yosh bolalar ham o‘z narsalarini tartiblashga o‘rganadi.

Saralashning juda ko‘p usullari mavjud. Ular turli to‘plamlar uchun turlicha bo‘lishi mumkin.

$$a_1, a_2, a_3, \ldots\ldots\ldots, a_n$$

elementlar berilgan bo‘lsin, u holda massivni saralash deganda, uni elementlarini o‘rinlariga almashtirish tushuniladi.

$$a_{k1}, a_{k2}, a_{k3}, \ldots\ldots\ldots\ldots, a_k$$

bu yerda, quyidagi tartiblashtirilgan funksiya bajariladi.

$$f(a_{k1}) \leq f(a_{k2}) \leq f(a_{k3}) \leq \ldots f(a_{kn})$$

Massivlarni saralash. Massivni saralash uchun ishlatiladigan usul unga berilgan xotirani ixcham holda ishlatishdir. Boshqacha qilib aytganda, saralanayotgan massiv xuddi shu massivni o‘zida amalga oshirilishi lozim. Saralanayotgan *a* massiv elementlarini kiritib, uni boshqa bir d massivda saralangan holda saqlanishi bizda hech qanday qiziqish uyg‘otmaydi.

Saralash usullari kam mashina vaqtini talab qilishi lozim. Eng yaxshi tez algoritmlar n*log *n* tartibidagi saralashlarni talab etadi.

Biz saralash bo‘yicha bir nechta sodda va ma’lum usullarni ko‘rib chiqamiz. Ular to‘g‘ri usullar deb aytiladi.

Saralash usullari to‘g‘risida quyidagi fikrlarni bildirish mumkin:

1. To‘g‘ri usullar ko‘plab saralashning asosiy tamoyillarining xarakterini ochib berishi uchun qulay.

2. Bu usullarni dasturlar oson tushunadi va ular qisqa. Eslatib o‘tamiz, dasturning o‘zi ham xotira egallaydi.

3. Murakkab usullar ko‘p sondagi amallarni talab qiladi, lekin bu amallarning o‘zlari yetarlicha murakkab bo‘lganlari uchun, kichik n larda tez va katta n larda sekin ishlaydi. Ammo ularni katta n larda ishlab bo‘lmaydi.

Bitta massivni o‘zida saralashni ularni mos aniqlangan tamoyillari bilan uch kategoriyaga ajratish mumkin:

1. Qo‘shish orqali saralash (by insertion);

2. Ayirish orqali saralash (by selection);

3. Almashish orqali saralash (by exchange).

To‘g‘ridan-to‘g‘ri qo‘shish orqali saralash. Bu usul qarta o‘yinida ko‘p qo‘llaniladi. Qartaning elementlari fikran tayyor holdagi $a_1, a_2, \ldots\ldots, a_{i-1}$ ketma-ketliklarga va dastlabki ketma-ketliklar $a_i, \ldots\ldots, a_n$ bo‘linadi.

Har qadamda i=2 dan boshlab i ta element ketma-ketlikdan chiqariladi va tayyor ketma-ketlikka qo‘yiladi. Bunda u har doim kerakli joyga qo‘yiladi. i ning qiymati har doim bittaga oshirib boriladi.

Saralashning maqsadi berilgan ob’yektlar to‘plamini aniq bir tartibda guruhlab chiqish jarayoni ta’riflanadi.

a_0, a_1... a_n ketma – ketlik berilgan bo‘lsin.

Ketma – ketlikning elementlarini saralash (masalan: $a_i <= a_{i+1}$, i = 0 dan n-1 gacha) masalasi qo‘yilgan bo‘lsin. Bu masalani ishlash algoritmini tanlaganda quyidagilarni baholash zarur: *Saralash vaqti* – algoritmni baholaydigan asosiy parametr hisoblanadi. *Hotira* – algoritm ishlashi uchun ketadigan qo‘shimcha hotira hajmi. Bunda berilganlar va dastur kodi uchun ketadigan hotira hajmi hisobga olinmaydi. *Turg‘unlik* – dasturni ketma – ketlikning boshqa qiymatlarda ham turg‘un ishlashi tushuniladi. Saralash algoritmlari klassifikasiyasi. Berilgan ketma – ketlikni saralashda ketma – ketlikning harakteristikasiga mos ravishda u yoki bu saralash algoritmi olinadi. Aks holda algoritmlar kerakli natijani bermaydi.

1. “Tez” saralash algoritmi:

“Tez” saralash algoritmi bosqichlari:

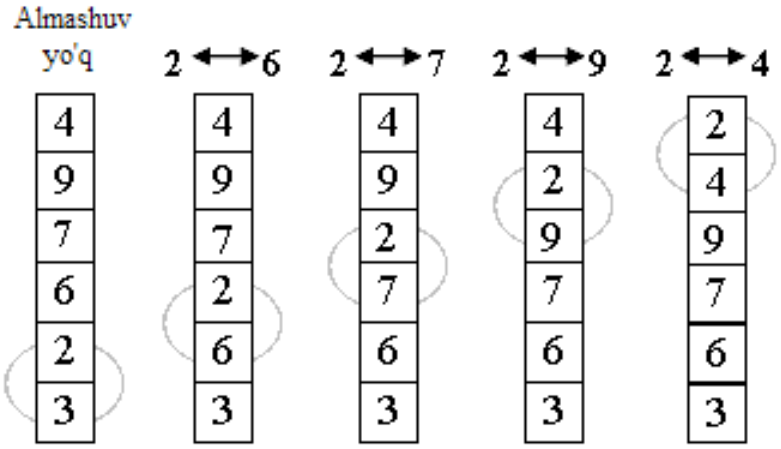
2. Massivning o‘rta elementini tanlab olamiz.

3. O‘rta element chap tomoniga o‘rta elementdan kichik elementlarni joylashtiramiz, o‘rta elementning o‘ng tomoniga esa o‘rta elementdan katta elementlarni joylashtiramiz.

```
int i=quyi;
 int j=yuqori;
 int x=A[(quyi+yuqori)/2];
 do {
   while(A[i]<x) ++i;
   while(A[j]>x) --j;
   if(i<=j){
     int temp=A[i];
     A[i]=A[j];
     A[j]=temp;
     i++; j--;
   }
 } while(i<=j);
```

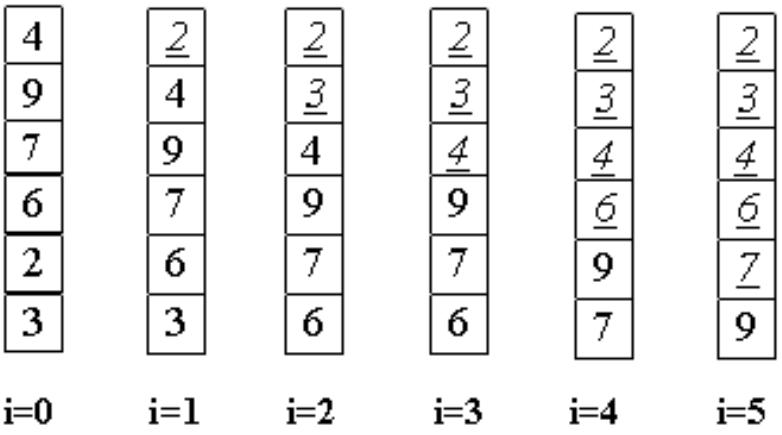
Pufaksimon saralash algoritmi.

Massiv elementlarini tepadan pastga qarab saralaymiz. Bunda faqat juft elementlar $a_i <= a_{i+1}$ (i = 0 dan n-1 gacha) shart bilan tekshiriladi, agar shart bajarilmasa ular o‘zaro o‘rin almashtiriladi.



Bu jarayon ohirgi element qolguncha bajariladi. Natijada massiv elementlari o‘sish tartibida saralanadi.

Dasturdagi bosqichlar:




```
long i, j,
float x, a[];
for( i=0; i < size; i++)
for( j = size-1; j > i; j-- )
{ if ( a[j-1] > a[j] )
{x=a[j-1]; a[j-1]=a[j]; a[j]=x; } }
```

Saralashning maqsadi keyinchalik, saralangan to‘plamni qidirilayotgan elementini topishdan iborat. Bu qariyb universal, fundamental jarayon. Biz bu

jarayon bilan har kuni uchrashamiz – telefon daftaridagi saralash, kitoblar sarlavhasida, kutubxonalarda, lug‘atlarda, pochtada va h.k. Xatto yosh bolalar ham o‘z narsalarini tartiblashga o‘rganadi. Saralashning juda ko‘p usullari mavjud. Ular turli to‘plamlar uchun turlicha bo‘lishi mumkin. Massivlarni saralash uchun ishlatiladigan usul unga berilgan xotirani ixcham holda ishlatish lozim. Boshqacha qilib aytganda, saralanayotgan massiv xuddi shu massivni o‘zida amalga oshirilishi lozim. Saralanayotgan *a* massivni elementlarini kiritib, unda boshqa bir d massivda saralangan holda tashkil topgan bizga hech qanday qiziqish uyg‘otmaydi. tartibidagi saralashlarni talab etadi. Biz quyidagi saralash bo‘yicha bir nechta sodda va ma’lum usullarni qaraymiz. Ular to‘g‘ri usullar deb aytiladi. Saralash usullari to‘g‘risida quyidagi fikrlarni bildirish mumkin:

1. To‘g‘ri usullar ko‘plab saralashning asosiy tamoyillarining xarakterini ochib berishi uchun qulay.

2. Bu usulli dasturlar oson tushuniladi va ular qisqa. Eslatib o‘tamiz, dasturning o‘zi ham xotira egallaydi.

3. Murakkab usullar ko‘p sondagi amallarni talab qiladi, lekin bu amallarning o‘zlari yetarlicha murakkab bo‘lganlari uchun, kichik n larda tez va katta n larda sekin ishlaydi. Ammo ularni katta n larda ishlab bo‘lmaydi.

Bitta massivni o‘zida saralashni ularni mos aniqlangan tamoyillari bilan uch kategoriyaga ajratish mumkin:

1. Qo‘shish orqali saralash (by insertion);

2. Ayirish orqali saralash (by selection);

3. Almashish orqali saralash (by exchange).

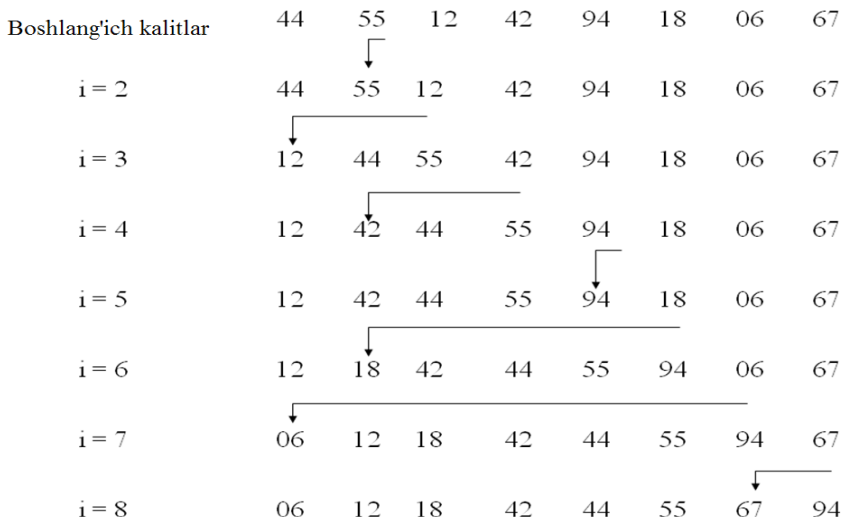
To‘g‘ridan-to‘g‘ri qo‘shish orqali saralash

Bu usul karta o‘yinida ko‘p qo‘llaniladi.

Kartaning elementlari fikran tayyor holdagi ketma-ketlik qismlarga bo‘linadi.

Har qadamda i=2 dan boshlab i ta element ketma-ketlikdan chiqariladi va tayyor ketma-ketlikka qo‘yiladi. Bunda u har doim kerakli joyga qo‘yiladi.

i ning qiymati har doim bittaga oshirib boriladi.



To‘g‘ridan to‘g‘ri tanlash yordamida saralash

• Eng kichik kalitli element tanlanadi.

• Uni birinchi element a_1 bilan o‘rinlari almashtiriladi.

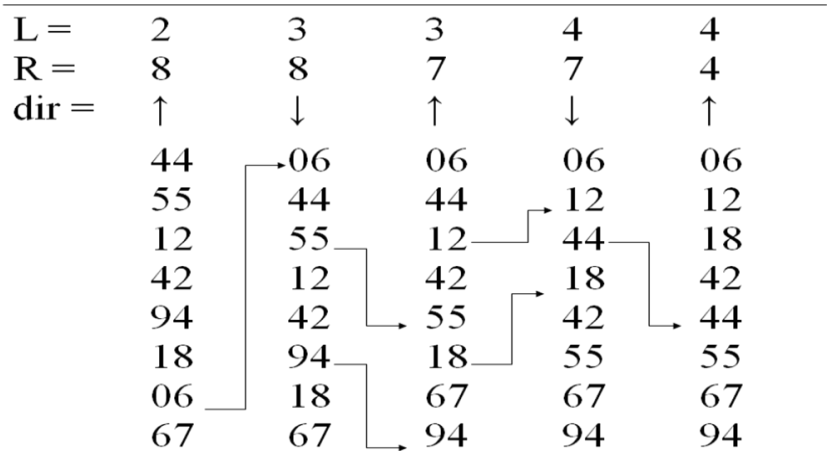
So‘ng bu jarayon qolgan n-1 element bilan, so‘ngra n-2 element bilan va h.k. bitta eng katta element qolmaguncha davom ettiriladi.

Boshlang'ich kalitlar	44	55	12	42	94	18	06	67
i = 2	06	55	12	42	94	18	44	67
i = 3	06	12	55	42	94	18	44	67
i = 4	06	12	18	42	94	55	44	67
i = 5	06	12	18	42	94	55	44	67
i = 6	06	12	18	42	44	55	44	67
i = 7	06	12	18	42	44	55	94	67
i = 8	06	12	18	42	44	55	67	94

Pufaksimon saralash:

i =	1	2	3	4	5	6	7	8
	44	06	06	06	06	06	06	06
	55	44	12	12	12	12	12	12
	12	55	44	18	18	18	18	18
	42	12	55	44	42	42	42	42
	94	42	18	55	44	44	44	44
	18	94	42	42	55	55	55	55
	06	18	94	67	67	67	67	67
	67	67	67	94	94	94	94	94

Sheyker saralash usuli



Muhokama savollari

1. Ketma-ketlikni saralash.
2. Ichki saralash.
3. Binar va qayta terish usullari.
4. Shell, Sheyker va sharsimon saralash algoritmlari.

Nazorat savollari

1. Tanlash yo‘li bilan saralash algoritmlari.
2. Almashuv yo‘li bilan saralash algoritmlari.
3. Hisoblash yo‘li bilan saralash algoritmlari.
4. Shell usuli bilan saralash algoritmi.
5. Sheyker usuli bilan saralash algoritmi
6. Sharsimon saralash usuli.
7. Binar va qayta terish bilan saralash usullari.

2.10 Funksiyalarni tashkil etish. Funksiya -proseduralari

Umuman olganda C++ tilida barcha yozuvlar funksiyadan iborat deb qaraladi. Funksiya bu ma’nosiga ko‘ra bajariluvchi modul bo‘lib hisoblanadi. Funksiyani

boshqa dasturlash tillarida qism dastur, prosedura, prosedura funksiya deb yuritiladi. C++ tilida funksiya standart formaga asosan quyidagicha ifodalanadi [9]:

funksiya toifasi funksiya nomi (rasmiy parametrlar ro‘yxati)

{ funksiya tanasi }

Funksiya toifasi istalgan toifa yoki void (bo‘sh) toifa bo‘lishi mumkin.

Funksiya nomi istalgan lotin harfi yoki harflaridan iborat bo‘lib, xizmatchi so‘zlar bilan bir xil bo‘lmasligi lozim.

Rasmiy parametrlar ro‘yxatida ishlatiladigan parametrlarga mos toifali o‘zgaruvchilar toifalari bilan alohida-alohida keltiriladi yoki bu soxa bo‘sh bo‘lishi ham mumkin. Eslatib o‘tish lozimki, funksiya aniqlashtirilayotganda nuqta vergul belgisi qo‘yilmaydi.

Funksiya tanasi o‘zining figurali qavslariga ega bo‘lib, o‘zida shu funksiyani tashkil etuvchi operatorlar yoki operatorlar blokini mujassamlashtiradi. Bir funksiya tanasi ichida boshqa funksiya aniqlanishi mumkin emas.

Funksiya tanasidan chiqish

return;

yoki

return ifoda;

ko‘rinishida bo‘ladi. Agar funksiya hech qanday qiymat qaytarmaydigan, ya’ni toifasi void bo‘lsa, birinchi ko‘rinishdagi chiqish ishlatiladi. Agar funksiya uning toifasiga mos biror qiymat qaytaradigan bo‘lsa, ikkinchi ko‘rinishdagi chiqish ishlatiladi. C tilida quyidagi ko‘rinishlar ekvivalent xisoblanadi, lekin birinchi ko‘rinish ko‘proq ishlatiladi:

```
double f (int n, float x)
{
  funksiya tanasi;
  }
```

```
double f ( n, x)
int n;  float x;
{
funksiya tanasi;  }
```

Dasturda funksiya ishlatiladigan bo‘lsa, uni albatta e’lon qilish shart. Funksiyani e’lon qilishda uning toifasi, nomi va qaytaradigan parametrlari haqida

xabar beriladi. Dasturda biror funksiyani oldindan e’lon qilmasdan turib uni chaqirish mumkin emas. Funksiyani asosiy funksiya main( ) dan oldin va keyin aniqlanishi mumkin. Agar funksiya asosiy funksiyadan oldin aniqlansa, u aniqlanishi bilan birga e’lon qilingan deb hisoblanadi va uni alohida main( ) ichida e’lon qilish shart bo‘lmay qoladi. Agar funksiya asosiy funksiyadan keyin aniqlanayotgan bo‘lsa, uni main( ) ichida albatta e’lon qilish shart bo‘ladi. Funksiyani main( ) ichida e’lon qilinadigan bo‘lsa, uning nomi bilan birga ishlatiladigan parametrlarining faqatgina toifalari ko‘rsatilishi ham mumkin. Masalan:

```
int myFuncsion ( int, float);
double Area (float, float);
```

Funksiyaga murojaat qilishdan uning rasmiy parametrlari aniqlangan bo‘lishi, ya’ni haqiqiy parametrlar berilgan bo‘lishi lozim. Funksiyaga murojaat qilish quyidagicha amalga oshiriladi:

```
funksiya_toifasi funksiya_nomi (haqiqiy parametrlar ro‘yxati);
```

Masalan:

```
myFuncsion (78, 3.0+m);
Area (a, b);
g (6.4e-2, 5, 70);
```

Funksiyaning rasmiy va haqiqiy parametrlarining toifasi, parametrlar soni va ularning kelish o‘rinlari albatta bir biriga mos kelishi shart!

Funksiyaga murojaat qilinganidan so‘ng aniqlangan funksiya tanasi bajariladi va mos toifali qiymat chaqirilgan joyga qaytib keladi.

Masalan: quyidagi funksiya chaqirilganida float toifali natija qaytaradi:

```
float ft (double x, int n)
{
 if (x < n) return x;
 else return n;
 }
```

Funksiyalarga murojaat qilinganida uning uzatiladigan parametrlariga alohida e’tibor berish kerak. Parametrlarning uzatilishi quyidagi bosqichlardan iborat:

- Funksiyani tashkil etadigan rasmiy parametrlar uchun xotiradan joy ajratiladi. Agar parametrlar haqiqiy toifaga ega bo‘lsa, ular double toifaga, agar char va short int toifali bo‘lsalar, ular int toifasi sifatida tashkil etiladilar. Agar parametrlar massiv shaklida bo‘lsalar, massiv boshiga ko‘rsatgich qo‘yiladi va u funksiya tanasi ichida massiv parametr bo‘lib xizmat qiladi.
- Funksiya chaqirilganida kerak bo‘ladigan ifodalar yoki haqiqiy parametrlar aniqlanadi va ular rasmiy parametrlar uchun ajratilgan joyga yoziladi;
- Funksiya chaqiriladi va aniqlangan haqiqiy parametrlar yordamida hisoblanadi. Bu yerda ham agar parametrlar haqiqiy toifaga ega bo‘lsa, ular double toifaga, agar char va short int toifali bo‘lsalar, ular int toifasi sifatida tashkil etiladilar.
- Natija funksiya chaqirilgan joyga qaytariladi.
- Funksiyadan chiqishda rasmiy parametrlar uchun ajratilgan xotira qismi bo‘shatiladi.

Funksiyaga murojaat qilish ifodani tashkil etadi, lekin agar funksiyaning qaytaradigan qiymati bo‘sh (void) bo‘lsa, u ifoda bo‘lmasligi ham mumkin. Unda bunday funksiyalarga murojaat qilish quyidagicha bo‘ladi:

funksiya nomi (haqiqiy parametrlar);

Masalan:

```
void print (int gg, int mm, int dd)
{
 cout<< ”\n yil:”<< gg;
 cout << “ \n oy: ” << mm;
 cout << “ \n kun: ” << dd;
}
```

ko‘rinishidagi funksiyaga print (1966, 11, 22); deb murojaat qilinsa, quyidagi natija chiqadi:

yil: 1966

oy: 11

kun: 22

Ba’zan umuman hech qanday parametrsiz funksiyalar ham ishlatiladi. Masalan:

```
void Real_Time (void)
{
 cout << “ Hozirgi vaqt: “ << TIME “(soat: min: sek)”;
}
```

funksiyasiga Real_Time ( ); deb murojaat qilinsa, ekranga

Hozirgi vaqt: 14: 16: 25 (soat: min: sek) degan axborot chiqadi.

Funksiya - bu mantiqan to‘g‘ri tugatilgan dasturiy qismdir. Ular yordamida katta va murakkab hisoblashlarni qayta - qayta yozish mashaqqatidan xalos bo‘linadi va dastur bajarilishi yengillashadi. Uni bir marta tashkil etib yozib qo‘yiladi va unga dasturning istalgan yeridan murojaat qilish mumkin bo‘ladi. Funksiyani tashkil qilishda funksiyaning toifasi, uning nomi va tashkil etuvchi parametrlari haqida axborot keltiriladi. Bu parametrlar rasmiy parametrlar deb yuritiladi.

Rasmiy va haqiqiy parametrlar soni, ularning toifasi va kelish o‘rni bilan albatta bir biriga mos bo‘lishi shart! Rasmiy va haqiqiy parametrlar nomlari bir xil bo‘lishi mumkin. Funksiyani bosh funksiya ichida e’lon qilinganida haqiqiy parmetrlar nomlarini ko‘rsatmasdan, faqat ularning toifalarini keltirish ham mumkin [12].

Funksiyalar main ( ) funksiyasidan avval ham, keyin ham aniqlanishi mumkin. Agar bosh funksiyadan avval aniqlangan bo‘lsa, uni main ( ) funksiyasi ichida alohida e’lon qilish shart emas, agar bosh funksiyadan keyin keladigan bo‘lsa, uni main ( ) funksiyasi ichida albatta e’lon qilish kerak. Masalan: sonning kubini hisoblash uchun funksiya tashkil eting va undan foydalaning.

```
#include <iostream.h>
#include <conio.h>
int main ( )
{ int k, n, kw (int n); // kw - funksiya nomi (ixtiyoriy)
cin>>n;         // n - berilayotgan son
k=kw(n);       // kw funksiyasiga murojaat qilinmoqda
```

```
cout << «k=«<<k<<endl;
getch( );
}
int kw (int a) // funksiya aniqlanmoqda. Bu yerda a rasmiy parametr
{  int c;    // lokal o‘zgaruvchi
 c=a*a*a;   // hisoblash
 return c; }  // funksiyaga natijani qaytarish
```

Yuqoridagi s lokal o‘zgaruvchisini ishlatmasdan, to‘g‘ridan-to‘g‘ri return a*a*a; deb yozsa ham bo‘ladi.

Bu yerda funksiya bosh funksiyadan keyin aniqlandi, shuning uchun uni bosh funksiya ichida e’lon qildik. Dasturni yana quyidagicha yozsa ham bo‘ladi:

```
#include <iostream.h>
#include <conio.h>
int kw (int a)
{ return a*a*a;  }
int main ( )
{ int k, n ;
cin>>n;
k=kw(n);
cout << “k= ” <<k<<endl;
getch( );
}
```

2-misol. Ikkita sondan eng kattasini topish uchun funksiya tashkil qiling va undan foydalaning.

```
#include <iostream.h>
#include <conio.h>
int main( )
{ float a=7, b=9, c, max(float , float );
c = max(a, b);
cout << “c=”<<c<<endl;
getch( );
}
float max ( float x, float y)
{  if  (x > y)  return x;  else  return y; }
```

Funksiyaga yana quyidagicha ham murojaat qilish mumkin:

```
c = max( 7.23, 9.145);
c = max( a, 9.145);
```

3-misol. Uchburchak uchlarining koordinatalari berilgan. Shu koordinatalar yordamida uchburchak qursa bo‘ladimi? Agar mumkin bo‘lsa shu uchburchakning yuzini hisoblash dasturini tuzing.

Demak, berilgan koordinatalar yordamida uchburchak tomonini ko‘rish funksiyasini, shu tomonlar asosida uchburchak qurish mumkinmi yoki yo‘qligini va uning yuzini hisoblash funksiyalarini tuzing.

```
#include <iostream.h>
#include <math.h>
#include <conio.h>
// uchburchak tomonini topish funksiyasi
float line (float x1, float x2, float y1, float y2)
{ (float) p = sqrt ((x1-x2)*(x1-x2)+ (y1-y2)*(y1-y2));
  return p; }
// uchburchak qurib bo‘ladimi? funksiyasi
int uch ( float a, float b, float c)
{ if ( a+b>c && b+c>a && c+a>b ) return 1;
else return 0; }
// uchburchakning yuzini topish funksiyasi
float s (float a, float b, float c)
{ float p, s ;
p = ( a + b + c ) / 2; s = sqrt (p*(p-a)*(p-b)*(p-c));
return s; }
int main ( )
{ float x1, x2, x3, y1, y2, y3, p1, p2, p3; clrscr ( );
cin >> x1>> x2>> x3>> y1>> y2>> y3;
p1 = line (x1, x2, y1, y2);
p2 = line (x1, x3, y1, y3);
p3 = line (x2, x3, y2, y3);
t = uch (p1, p2, p3);
if ( t = = 1)
{ yuza = s ( p1, p2, p3); cout << “yuza = ”<< yuza << endl;
else cout <<”uchburchak qurib bo‘lmaydi !!!”<< endl;
```

```
} getch ( ); }
```

Bir funksiya ichida boshqa funksiya aniqlanishi mumkin emas, lekin funksiya ichida o‘zini-o‘zi chaqirishi mumkin. Bunday holatni rekursiya holati deyiladi. Rekursiya 2 xil bo‘ladi: to‘g‘ri rekursiya va bilvosita rekursiya. Agar funksiya o‘zini-o‘zi chaqirsa, bu to‘g‘ri rekursiya deyiladi. To‘g‘ri rekursiyada funksiyaning nusxasi chaqiriladi. Agarda funksiya boshqa bir funksiyani chaqirsa va u funksiya o‘z navbatida 1-sini chaqirsa, u holda bilvosita rekursiya deyiladi. Rekursiya 2 xil natija bilan yakunlanadi: biror natija qaytaradi yoki hech qachon tugallanmaydi va xatolik yuz beradi. Bunday holatlarda rekursiv funksiyalar uchun rekursiyani to‘xtatish shartini berish zarur, chunki rekursiyada xotira yetishmasligi xavfi bor.

4-misol. F = n! ni xisoblash uchun funksiya tashkil eting va undan foydalaning.

```
#include <iostream.h>
#include <conio.h>
int main( )
{ int n, f, fac(int);
cout << “sonni kiriting:”; cin >> n;
f = fac(n); cout << “sonning factoriali=”<<f<<endl;
getch( );
}
int fac(int i)
{ return i <=1 ? 1 : i * fac( i - 1); }
```

5-misol. Fibonachi sonlarini xosil qilish dasturini tuzing. Fibonachi sonlari quyidagicha topiladi:

$f_0 = 1; f_1 = 1; f_2 = f_1 + f_0 ; \ldots\ldots$

$f_n = f_{n-1} + f_{n-2} ;$

Rekursiv jarayonni to‘xtatish sharti n < 2 deb olinadi. Masalan 9-o‘rindagi Fibonachi sonini topish kerak.

```
#include <iostream.h>
int main ( )
{ int n, f ; int fib ( int );
cout << “Nomerni kiriting =”;
```

```
cin >> n;
f = fib (n);
cout << “Fibonachi soni=”<< f << endl;
}
int fib ( int n )
{  if ( n < 2) return 1; else return ( fib (n-2) + fib (n-1)); }
```

6-misol. Z = $\frac{a^5 + a^{-4}}{2a^n}$ hisoblash dasturi tuzilsin. Bu yerdagi darajani hisoblash funksiya sifatida tashkil etilsin. $y = x^n$ ni funksiya deb tashkil etamiz, bu yerda x, n - rasmiy parametrlar

```
#include <iostream.h>
float dar (float x, int n)
{ float y=1;
for (int i=0; i<=n; i++)
  y = y*x;
  return y; }
void main( )
{
int n=3 ;  float a, z;
cin>>a;
z = ( dar(a, 5) + dar(1/a, 4))/ (2* dar(a, n)) ;
cout << “z=”<<z<<endl;  }
```

Bir xil nomdagi funksiyalarni har xil toifali o‘zgaruvchilar ro‘yxati bilan murojaat qilib chaqirish mumkin. Parametrlar soni ham har xil bo‘lishi mumkin. Bunday holatda parametrlar ro‘yxati va qiymatlarga qarab kompilyator o‘zi qaysi funksiyani chaqirish kerakligini aniqlaydi [2]. Masalan:

```
1. double multi (float x)
  {return x*x*x; }
2. double multi (float x, float y)
  { return x*y*y; }
3. double multi (float x, float y, float z)
{ return x*y*z;  }
```

va quyidagi murojaatlarning hammasi to‘g‘ri yozilgan:

```
multi (0.5);
multi (1.45, 7);
multi (10, 39, 54);
```

Funksiyalarning bir xil nom bilan atalishi polimorfizm deb ataladi. Poli – ko‘p, morfe – shakl degan ma’noni bildiradi.

Masalan:

```
#include <iostream.h>
int max (int a, int b)
{ if (a>b) return a; else return b;}
float max (float a, float b)
{ if (a>b) return a; else return b;}
int main ( )
{
int a1, b1; float a2, b2;
cin >> a1>>b1;
cout << “butun max=”<<max(a1, b1)<<endl;
cin >>a2>>b2;
cout << “haqiqiy max=”<< max(a2, b2)<<endl;
}
```

1-misol. B va C vektorlarining uzunliklarini hisoblash dasturini tuzing. Vektor uzunligini hisoblash uchun funksiyadan foydalaning.

```
#include < iostream. h>
float vector (int d[ ], int k)
{ float s=0; int i ;
for (i=0; i<k; i++)
s = s + d[i] * d[i];
s = sqrt (s);
return s; }
int main ( )
{
int b[3] = {10,20,30}, c[4] = {14,15,16,17};
float  s1, s2;
s1 = vector (b, 3);
s2 = vector (c, 4);
cout <<”s1=” << s1 <<” s2=” << s2 << endl;
```

}

2-Misol. Butun sonli 4x5 matrisasi berilgan. Aniq bir sondan kichik bo‘lgan xadlarining yig‘indisini topish dasturini tuzing. Matrisa elementlarini kiritish (tasodifiy sonlar yordamida), chiqarish va yig‘indini hisoblash jarayonlarini funksiya sifatida tashkil eting.

Funksiya ichida 2 o‘lchovli massivlardan foydalanilganda uning 1-parametrini, ya’ni satrlar sonini ko‘rsatmaslik ham mumkin, lekin 2-parametrini, ya’ni ustunlar sonini albatta ko‘rsatish shart.

```
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
void kir(int m[ ][5], int k);
void chiq(int m[ ][5], int k);
int summa(int m[ ][5], int k, int x);
int i, j ;
int main ( )
{ int matr[4][5]; int a, s; int b[ ][3];
cout<< “sonni kiriting=”; cin>>a;
kir(matr, 4);  chiq(matr, 4);
s = summa(matr, 4, a);
cout<< “s=”<<s<< endl;
getch( );  }

void kir(int m[ ][5], int k)
{ srand(time(0));
for (i=0;i<k;i++)
for (j=0;j<5;j++)
m[i][j]=rand( ) - 200; }

void chiq(int m[ ][5], int k)
{ for (i=0;i<k;i++)
for (j=0;j<5;j++)
cout <<m[i][j]<<endl; }
```

```
int summa(int m[ ][5], int k, int x)
{ int s1 = 0;
for (i=0; i<k; i++)
for (j=0; j<5; j++)
if (m[i][j] < x) s1 = s1 + m[i][j];
return s1; }
```

Funksiyalarga murojaat qilish quyidagi bosqichlardan iborat bo‘ladi:

1. Funksiya bajarilayotganda rasmiy parametrlar uchun xotiradan joy ajratiladi, ya’ni ular funksiyaning ichki parametrlariga aylantiriladi. Bunda parametr toifasi float toifasi double toifasiga, char va shortint toifalari int toifasiga aylantiriladi.

2. Haqiqiy parametrlar qiymatlari qabul qilinadi yoki hisoblanadi.

3. Haqiqiy parametrlar rasmiy parametrlar uchun ajratilgan xotira qismiga yoziladi.

4. Funksiya tanasi ichki parametrlar yordamida bajariladi va qiymat qaytarish joyiga yuboriladi.

5. Funksiyadan chiqishda rasmiy parametrlar uchun ajratilgan xotira qismi bo‘shatiladi.

Dasturdagi har bir o‘zgaruvchi – Ob’yekt hisoblanadi. Uning nomi va qiymati bo‘ladi. Har bir Ob’yekt xotiradan ma’lum joy egallaydi va ular ma’lum adresga ega bo‘ladi. Dasturlashning ma’lum etaplarida o‘zgaruvchining o‘ziga emas, balki uning adresiga murojaat qilishga to‘g‘ri keladi. Bunday paytlarda ko‘rsatgichlardan foydalaniladi. Ko‘rsatgich - bu biror o‘zgaruvchining adresini o‘zida saqlovchi o‘zgaruvchidir. Adres - bu xotira yacheykasining tartib nomeri. Umuman olganda adres 4 bayt joy oladi. Ko‘rsatgichlarni e’lon qilishda uning toifasidan keyin * belgisi va o‘zgaruvchi nomi keltiriladi.

Masalan: int a; char *d; int *p;

Ko‘rsatgichlar ham inisalizasiya qilinishi mumkin. *r = 6; *d =‘$';

cout <<*p bilan cout <<p ning farqi bor. *p da shu yerdagi qiymat chiqadi, p ning o‘zini yozsak, shu yerning adres nomeri chiqadi. Masalan:

int a=10, b=5, e, *m;

e = a + b;

*m = e;

cout <<*m; deb yozilsa, m = 15 chiqadi;

cout <<m; deb yozilsa, m = 0xfff2 chiqadi, ya’ni shu yerning adres nomeri chiqadi.

Ularning qiymatlarini “adresini ol!” (&) operatsiyasi orqali amalga oshirsa ham bo‘ladi, ya’ni m=&e; cout <<*m; deb yozish ham mumkin, u holda m=15 chiqadi, ya’ni e- ning adresidagi son qiymat chiqadi. Buni bilvosita murojaat operatori ham deyiladi.

Masalan:

int h;

int *p=35;

h = &p;

Natija: h = 35;

Adresni olish amali (&) son yoki ifodalarga qo‘llanilmaydi, ya’ni &3.14 va &(a+b) yozuvlari xatodir.

Ko‘rsatgichlar ustida quyidagi amallarni bajarish mumkin:

Ko‘rsatgichlar ustida arifmetik amallarni bajarish:

*p1-*p2; *p1+*p2

Ko‘rsatgichlarga biror sonni qo‘shish yoki ayirish:

*p1 - 25; *p1+3.45

Ko‘rsatgichlarni bittaga oshirish yoki kamaytirish:

*p1++ yoki --*p1

Misol.

```
#include <iostream.h>
#include <conio.h>
int main ( )
{
int x=10, y=10; int *xp, *yp;
```

```
*xp = &x; *yp = &y;
if (xp = = yp) cout << “ular teng!”<<endl;
else cout << “ular teng emas!”<<endl;
if (*xp = = *yp) cout << “ular teng!”<<endl;
else cout << “ular teng emas!”<<endl;
getch( ); }
```

1- if da ular teng emas chiqadi, chunki ularning adres qiymatlari har xil.

2- if da ular teng chiqadi, chunki ularning adreslaridagi son qiymatlari bir xil

2-misol.

```
#include <iostream.h>
int main ( )
{ int m = 5, *p = 0;
p = &m;
cout << m << endl;
cout << *p<<endl;
*p = 7;
cout << m << endl;
cout << *p << endl;
m = 9;
cout << m << endl;
cout << *p << endl;
}
```

Natija:
m = 5
*p = 5
m = 7
*p = 7
m = 9
*p = 9

Ba’zi masalalarda funksiya bilan ishlaganda funksiya tanasi ichida haqiqiy parametrlar qiymatlarini o‘zgartirish zaruriyati tug‘iladi, ya’ni natija bir emas, balki bir nechta hosil bo‘lishi kerak bo‘ladi. Bunday jarayonni proseduralar hosil qilish deyiladi va bu muammoni xal qilish uchun ko‘rsatgichlardan foydalaniladi. Funksiyani aniqlashtirishda rasmiy parametrlar bilan bir satrda natijalar nomlari ham ko‘rsatiladi. Shuning uchun proseduralar bilan ishlaganda funksiya toifasini bo‘sh (void) deb olish maqsadga muvofiqdir, return shart bo‘lmay qoladi [11].

Masalan: tomonlari berilgan to‘rtburchakning perimetrini va yuzini hisoblash uchun funksiyani quyidagicha aniqlashtiriladi:

void tt (float a,float b, float* p, float* s)

{ *p = 2*(a+b); *s = a*b; } Bu yerda float a, float b beriladigan kattalik hisoblanadi, float* p, float* s lar esa natijalar hisoblanadi.

Bu funksiyaga murojaat qilish quyidagicha bo‘ladi:

tt (2.3, 4, &p, &s); ya’ni 2*(a+b); va a*b ning qiymatlari adreslar bo‘yicha olinadi.

{Proseduralarni beriladigan kattaliklarsiz ham tashkil etish mumkin. Bunda funksiya tanasi ichida ishlatilgan barcha kattaliklar beriladiganlar hisobiga o‘tadi. }

Hosil qilingan proseduralarga murojaat qilish adres (&) operatsiyasi orqali amalga oshiriladi.

Masalan: $Z = \frac{a^5 + a^{-4}}{2a^n}$ hisoblash dasturi tuzilsin. Bu yerdagi darajani hisoblash prosedura sifatida tashkil etilsin. $y = x^n$ ni prosedura deb tashkil etamiz, bu yerda x, n - rasmiy parametrlar

```
#include <iostream.h>
void dar1 (float x, int n, float *y)
{ *y=1;
for (int i=0; i<=n; i++)
  *y = y*x; }
int main( )
{
int n=3 ;  float a, z, z1, z2, z3;
cin>>a;
dar1(a, 5,&z1); dar2( 1/a, 4, &z2); dar1(a, n, &z3);
z = ( z1+z2) / z3 ;
 cout << "z= "<<z<<endl;
}
```

2-misol. 2 ta vektor berilgan. Vektorlar orasidagi burchak quyidagi formula bilan hisoblanadi:

$$\varphi = \arccos \frac{(x, y)}{\sqrt{(x, x)(y, y)}}$$

bu yerda (x,u), (x,x), (u,u) - vektorlarning skalyar ko‘paytmasi. Vektorlarning skalyar ko‘paytmasini dasturda prosedura sifatida tashkil eting.

```
#include <iostream.h>
#include <math.h>
typedef float mm[4];
void vec(mm a, mm b, float* s)
```

```
{*s=0;
  for (int i=0; i<4; i++)
  *s=*s+a[i]*b[i];   }
int main ( )
{float fi, f1, f2, f3; int i;
mm  x, y; // mm x={1,2,3,4}, y={5,6,7,8};
for (i=0; i<4; i++)
cin >>x[i] >> y[i];
vec (x, y, &f1); vec (x, x, &f2); vec (y,y,&f3);
fi = f1 / sqrt( f2*f3); fi = atan(sqrt (1-fi*fi) / fi);
cout << «fi=«<<fi*180/3.1415<<endl;  // natija gradusda chiqadi
getch( ); }
```

Proseduralarni tashkil etishda koʻrsatgichlardan tashqari yana havolalardan ham foydalaniladi. Bu usul yanada qulay hisoblanadi. Unda (*) amalining oʻrniga toʻgʻridan-toʻgʻri adres olish (&) amali ishlatiladi va proseduraga murojaat qilish osonlashadi.

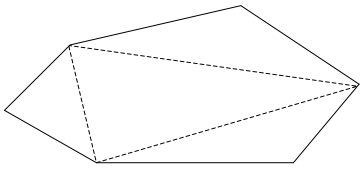
Masalan:

```
void tt (float a,float b, float& p, float& s)
{ p = 2*(a+b); s = a*b; }
```

Bu funksiyaga murojjat qilish quyidagicha boʻladi:

```
tt (2.3, 4, p, s);
```

Misol. Bir fermerning yer yuzasini va shu yerga toʻlaydigan yer soligʻini hisoblash dasturini tuzing. Yer maydoni quyidagi koʻrinishda:



```
#include <iostream.h>
#include <math.h>
#include <conio.h>
```

```
# define pi 3.1415
void yuza (int a, int b, int al, float& c, float& s)
{ c = sqrt(a*a+b*b-2*a*b*cos(al*pi/180));
  s = a*b*sin(al*pi/180)/2;  }
int main ( )
{ int a1=10, b1=30, a2=40, b2=40, a3=30,b3=50,al1=85, al2=145, al3=125;
float c1, c2, c3, s1, s2, s3, s4, s, sol, p;
yuza (a1, b1, al1, c1, s1); // yuza (10, 30, 85, c1, s1) deb yozsa ham bo‘ladi
yuza (a2, b2, al2, c2, s2);
yuza (a3, b3, al3, c3, s3);
p = (c1+ c2 + c3) / 2;
s4 = sqrt ( p*(p - c1)*(p - c2)*(p - c3));
s = s1+s2+s3+s4;
s = s/100; sol = s * 8560; //    (som)
cout <<«er yuzasi=«<< s << endl;
cout << «soliq=«<<sol<<endl;
getch( ); }
```

2-misol. Kvadrat tenglamaning haqiqiy yechimlarini topish dasturini tuzing.

```
#include <iostream.h>
#include <math.h>
int kvad (float a, float b, float c, float &x1, float &x2)
{ float d ;
d = b * b – 4*a*c;
if ( d < 0 ) return 0;
x1 = (-b + sqrt (d)) / (2*a);
x2 = (-b - sqrt (d)) / (2*a);
if ( x1 = = x2) return 1; else return 2;
}
int main ( )
{ float a, b, c, x1, x2;  int k;
```

```
cin >> a >> b >> c;
k = kvad (a, b, c, x1, x2);
switch ( k )
{
case 0 : cout << “echimi yo‘q”<< endl; break;
case 1 : cout << “x=”<< x1 << endl; break;
case 2 : cout << “x1=”<< x1 << “ x2=” << x2 << endl; break;
}
}
```

3-misol. 4x4 va 4x5 o‘lchamli matrisalar berilgan. Ulardagi juft ustunlari xadlari yg‘indisini topish dasturini tuzing. (natija vektor ko‘rinishida chiqadi)

```
#include <iostream. h>
typedef  float mmm[10][10];
typedef float mm[10]; int i, j;
void nodir (mmm a, int n, mm b)
{
for (j=0; j<4; j+=2)
{ b[j] = 0;
for (i=0; i<n; i++)
b[j] = b[j] + a[i][j]; } }
int main ( )
{ mmm d = {{1,2,3,4},{1,2,3,4},{1,2,3,4},{1,2,3,4}};
mmm d1 = {{1,2,3,4,5},{1,2,3,4,5},{1,2,3,4,5},{1,2,3,4,5}};
mm c, c1;
nodir (d, 4, 4, c);
nodir (d1, 4, 5, c1);
for (i=0; i<4; i++)
{ cout <<”c=”<< c[i];
```

```
cout << “c1=”<< c1[i] << endl;
}
```

4 –misol. Ikki o‘zgaruvchining qiymatini ayirboshlash dasturi tuzilsin.

```
#include <iostream.h>
void main( )
{
 float x, y;
 void aa( float *, float *);
 cout <<” x=”<< x <<endl; cin >> x;
 cout <<” y=” << y << endl;  cin >> y;
  aa ( &x, &y);
  cout << “\nNatija: \n”;
 cout <<”x=”<<x<<”y=”<<y;
 }
void aa (float *b, float *c)
 { float e;
  e = *b;
 *b = *c;
 *c = e;
 }
```

Asosiy dasturda x va y o‘zgaruvchilarining qiymatlari klaviaturadan kiritiladi. Masalada ikkita son o‘zaro o‘rin almashishi so‘ralmoqda. aa ( ) funksiyaning rasmiy parametrlari sifatida float * tavsiya etilgan. aa () funksiyasiga murojaat qilinganida x va y larning son qiymatlari haqiqiy parametrlar sifatida qabul qilinadi. Bu dasturning ishlashi jarayonida quyidagi natijalar olinadi:

x=33.3 y=66.6 qiymatlar kiritilsa

Natija:

x=66.600000 y=33.300000

5-misol. Uchburchakning perimetri va uning yuzasini hisoblash uchun dastur tuzilsin.

```
#include <iostream.h>
#include <math.h>
int main ( )
{
 float x, y, z, pp, ss;
 int tria (float, float, float, float *, float *);
 cout <<" x="; cin >> x;
 cout <<" y="; cin >> y;
 cout <<" z="; cin >> z;
 if (tria (x, y, z, &pp, &ss)==1)
 cout << "Uchburchak yuzasi="<< ss << "va perimetri="<< pp <<endl;
  else
 cout << "Ma'lumotlar noto'g'ri kiritilgan!" << endl;
}
 int tria ( float a, float b, float c, float *pp, float *ss)
{
 float e;
 if (a+b<=c || a+c<=b || b+c<=a) return 0;
 else
  { *pp = a+b+c;
   e=*pp/2;
  *s=sqrt(e*(e-a)*(e-b)*(e-c));
   return 1;
}
```

Dasturning bajarilishiga misol:

x=3

y=4

z=5

pp=12.00000

ss=6.00000

Funksiyada parametrlar sifatida massivlar va satrlar ishlatilishi mumkin. Agar funksiyaning parametri sifatida massivlar ishlatilsa, funksiya ichida massiv boshlanishining adresi uzatiladi. Bunga misol tariqasida vektorlarning skalyar ko‘paytmasini hisoblovchi funksiya sarlavxasini ko‘rib chiqamiz:

float skalyar(int n, float a[ ], b[ ]) yoki

float skalyar (int n, float *a, float *b)

Bu yerda float a[ ] va float *a yozuvlari parametr sifatida bir xil ma’noni anglatadi.

Satrlar funksiya parametri sifatida.

Satrlar funksiya parametri sifatida ishlatiladigan bo‘lsa, char [ ] yoki char* toifali ko‘rsatgichlar kabi tavsiflanadi. Oddiy massiv parametridan farqli o‘laroq, satrning uzunligini ko‘rsatish shart emas. Bunda \0 belgisi satr oxirini avtomatik ravishda ko‘rsatadi. Misol tariqasida satrlarni qayta ishlovchi bir nechta dasturlarni ko‘rib chiqamiz. Bu dasturlarning o‘xshashi standart kutubxonalarda saqlanadi va ularni ishga tushirish uchun string.h stdlib.h fayllaridan foydalanish kerak bo‘ladi.

1. Satr toifali o‘zgaruvchining uzunligini aniqlash uchun funksiya:

```
int len(char e[ ])
{ int m;
 for(m=0; e[m]!=’\0’; m++)
 return m; }
```

Yoki bu dasturdagi massivni ko‘rsatgichlar orqali quyidagicha ifoda etish ham mumkin:

```
int len(char *s)
{ int m;
 for(m=0; *s!=’\0’; m++)
 return m; }
```

2. Satr toifali massiv elementlarini teskari ifoda etish uchun funksiya:

```
void invert (char e[ ])
```

```
{ char s; int i, j, m;
  for (m=0; e[m]!='\0'; m++)
  for(j=0, j=m-i; i<j; i++, j- -)
 { s=e[i]; e[i] = e[j]; e[j] =s; } }
```

Dasturdagi void toifadan ma'lumki, bu funksiya hech qanday qiymat qaytarmaydi.

Masalan:

```
#include <iostream.h>
int main( )
{ char ct[ ] ="0123456789";
 void invert (char [ ]);
 invert(ct); cout << ct; }
```

Natija: 9876543210

3. Satrning chap tomonidan kiritilgan boshqa satrni qidirish funksiyasi:

```
int index(char *ct1, char *ct2)
{ int i, j, m1, m2;
 for(m1=0; ct1[m1]!='\0'; m1++)
 for(m2=0; ct2[m2]!='\0'; m2++)
 if (m2>m1) return -1;
 for(i=0; i<m1-m2; i++)
 { for(j=0; j<m2; j++)
  if (ct2[j] !=ct1[i+j] ) break;
  if (j==m2) return 1; }
  return -1; }
```

Funksiyaning ishlashiga misol:

```
#include <iostream.h>
int main ( )
{ char c1[ ] ="og'irlik_yig'indisi";
```

```
int index(char[ ], char[ ]);
char c2[ ] = “non”;
char c3[ ] = “olma”;
cout<< index(c1,c2);
cout<< index(c1,c3);
}
```

Muhokama savollari

1. Funksiyalarni tashkil etish
2. Funksiya parametrlari-rasmiy, haqiqiy.
3. Funksiyalarga qiymatlarni joʻnatish-qiymat parametrlar.
4. Funksiyalarga oʻzgaruvchilarni joʻnatish-oʻzgaruvchi parametrlar.
5. Funksiyalardan qiymatlarni va oʻzgaruvchilarni qaytarish.

Nazorat savollari

1. Funksiyalarni tashkil etish.
2. Funksiya nomi, tavfsilotchisi, parametrlari.
3. Funksiyaga murojaat.
4. Koʻrsatgichlarni e’lon qilish.
5. Koʻrsatgichlar qiymatini aniqlash.
6. Koʻrsatgichlar ustida amallar.
7. Vektor uzunligini hisoblovchi funksiyani tashkil etish.
8. Funksiyalarni polimorfizm xususiyati.
9. Funksiyalarga murojaat qilish bosqichlari.
10. return operatorining vazifasi.

2.11 Qidirish algoritmlari

Chiziqli va binar qidiruv algoritmlari.

Masala. Massivda X ga teng bo‘lgan elementni topish, yoki bunday element mavjud emasligini aniqlash.

- Yechim: ixtiyoriy massivda chiziqli qidiruvni amalga oshirish kamchiligi: tezlik juda past.
- Tezlikni qanday oshirish mumkin? – massivni qidiruvga tayyorlash
- Qanday qilib tayyorlash?
- “tayyorlangan” massivni qanday qo‘llash lozim?

Chiziqli qidiruv

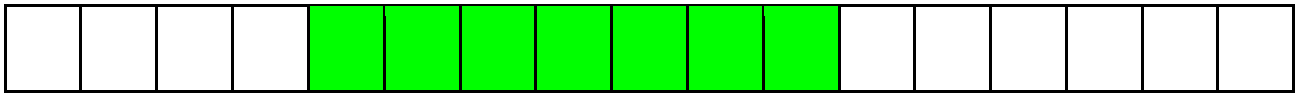
```
nX = -1; // hali topilmadi \\ X – massivda kerakli elementning tartib raqami
for ( i = 0; i < N; i ++) \\ elementlar bo‘yicha sikl
if ( A[i] == X )    \\ agar topsak, unda...
nX = i; // ... Tartib raqamni saqlaymiz
if (nX< 0) printf(“Topilmadi...»)
else printf(«A[%d]=%d», nX, X);
```

Yaxshilash: X topilgandan so‘ng, sikldan chiqamiz, buning uchun break operatoratoridan foydalanamiz.

Binar (ikkilik)qidiruv

1. A[c]- o‘rta elementni ajratamiz, X bilan solishtiramiz.
2. Agar X=A[c] bo‘lsa, topildi (chiqish).
3. Agar X<A[c], birinchi yarmidan qidirish.
4. Agar X>A[c], ikkinchi yarmidan qidirishni davom ettirish.

BINAR QIDIRUV



```
nX = -1;
L = 0; R = N-1; // chegaralar: A[0] dan A[N-1] gacha qidiramiz
while ( R>= L ){
```

```
c = (R + L) / 2;  //  O‘rta element tartib raqami
if (X = = A[c])// Agar topilsa…
{nX = c;
break;// Sikldan chiqish
  }
if (X<A[c]) R = c - 1; // Chegaralarni o‘zgartiramiz
if (X > A[c]) L = c + 1;  }
if (nX< 0) printf (“Topilmadi...»);
else printf («A[%d]=%d», nX, X);
```

Qidiruv usullarini taqqoslash

	Chiziqli	Binar
tayyorlash	yo‘q	ha
	Qadamlar soni	
N = 2	2	2
N = 16	16	5
N = 1024	1024	11
N= 1048576	1048576	21
N	$\leq N$	$\leq \log_2 N+1$

Muhokama savollari

1. Chiziqli va binar qidiruv algoritmlari.
2. Elementlarga qidirish orqali ishlov berish.
3. Binar va qayta terish usullari.
4. Qayta terish algoritmi.

Nazorat savollari

1. Qidiruv haqida ma’lumot bering.
2. Chiziqli qidiruvni tushuntiring.

3. Ikkiga bo‘lib qidirmoq (ikkilamchi qidiruv).

4. Jadvalda izlashni tushuntiring

Misollar

1. 1-999 oraliqdagi sonlar berilgan. Berilgan sonni “ikki xonali juft son”, “uch xonali toq son” va x.k. ko‘rinishda ekranga yozadigan dastur tuzilsin.

```
#include <iostream>
using namespace std;
int main()
{   int n;
cout<< «n = «; cin >> n;
if (0 < n && n < 10) cout << “Bir xonali”;
else if (n >= 10 && n < 100 ) cout << “Ikki xonali ”;
else if (n >= 100 && n < 1000) cout << “Uch xonali ”;
if (n % 2 == 0 ) cout << “juft son”  << endl;
else cout << “toq son” << endl;
system(“pause”);
return 0;
 }
```

2. Yil berilgan (musbat butun son). Berilgan yilda nechta kun borligini aniqlovchi dastur tuzilsin. Kabisa yilida 366 kun bor, kabisa bo‘lmagan yilda 365 kun bor. Kabisa yil deb 4 ga karrali yillarga aytiladi. Lekin 100 ga karrali yillar ichida faqat 400 ga karrali bo‘lganlari kabisa yil hisoblanadi.

Masalan 300, 1300 va 1900 kabisa yili emas. 1200 va 2000 kabisa yili.

```
#include <iostream>
using namespace std;
int main()
{int yil;
cout<< “yilni kiriting:”; cin >> yil;
if ((yil % 400 == 0) || (yil % 4 == 0 && yil % 100 != 0))
```

```
cout<< “Kabisa yil” << endl;
else cout << “Kabisa yil emas” << endl;
system(“pause”);
return 0; }
```

3. To‘g‘ri to‘rtburchakning qarama–qarshi uchlari koordinatalari berilgan. Uning tomonlari koordinata o‘qiga parallel. To‘g‘rito‘rtburchakning perimetri va yuzasi aniqlansin.

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{  float x1, x2, y1, y2;
cout<< “Birinchi nuqta A(x1;y1)” << endl;
cout<< “x1=”; cin >> x1;
cout<< “y1=”; cin >> y1;
cout<< “\nIkkinchi nuqta B(x2;y2) ” << endl;
cout<< “x2=”; cin >> x2;
cout<< “y2=”; cin >> y2;
cout<< “\nTo‘g‘ri to‘rtburchak perimetri P=” << 2 * (fabs(x2 - x1) +fabs(y2-y1)) << endl;
cout<< «To‘g‘ri to‘rtburchak yuzasi S=” << fabs(x2 - x1) * fabs(y2 -y1) << endl;
return 0;}
```

11. Uch xonali son berilgan. Uning chapdan birinchi raqamini o‘chirib oэng tarafiga yozishdan hosil boэlgan sonni aniqlovchi dastur tuzilsin.

```
#include<iostream>
using namespace std;
int main()
```

```
{int a, b, c, d, e;
cout<< “Uch xonali son kiriting: a =”;
cin>> a;
  d = a / 100;
  b = (a % 100) / 10;
  c = a % 10;
  e = 100 * b + 10 * c + d;
cout<< a << “sonini chapdagi birinchi raqamini o‘ngga yozishdan hosil bo‘lgan son :” << e << endl;
system (“pause”);
return 0; }
```

12. Kun boshidan boshlab N sekund vaqt o‘tdi. Kun boshidan boshlab qancha soat va sekund o‘tganini aniqlovchi dastur tuzilsin.

```
#include<iostream>
using namespace std;
int main()
{ int s, soat;
cout<< “Kun boshidan necha sekund o‘tganligini kiriting: s =”;
cin>> s;
soat = s / 3600 ;
  s = s % 3600;
cout << “Kun boshidan”<< soat << “soat” << s << “sekund o‘tdi.” <<endl;
system ( “pause” );
return 0;
 }
```

13. Hafta kunlari quyidagicha tartibda berilgan. 0-yakshanba,1-dushanba, 2-seshanba, 3-chorshanba, 4-payshanba, 5-juma, 6-shanba. 1-365 oraliqda yotuvchi K soni berilgan. Agar 1-yanvar dushanba bo‘lsa, kiritilgan K - kun haftaning qaysi kuniga to‘g‘ri kelishini aniqlovchi dastur tuzilsin.

```
#include<iostream>
using namespace std;
int main()
{int k;
cout<< “( 1 < k < 365 ): k =”;
cin>> k;
cout<< k % 7;
system ( “pause” );
return 0; }
```

14. Og‘irlik birliklari quyidagi tartibda berilgan.1-kilogramm, 2-milligramm, 3-gramm, 4-tonna, 5- sentner. Og‘irlik birligini bildiruvchi soni berilgan va shu birlikdagi og‘irlik qiymati berilgan. Og‘irlikni kilogramda ifodalovchi dastur tuzilsin.

```
#include<iostream>
#include<math.h>
using namespace std;
int main()
{ float a; int n;
cout<< “n=”; cin>>n;
cout<< “a=”; cin>>a;
switch(n)
  {
case 1: cout<<a<< “kg=”<<a<< “kg”<<endl; break;
case 2: cout<<a<< “milligramm=”<<a/1000000<< “kg”<<endl; break;
case 3: cout<<a<< “gramm=”<<a/1000<< “kg”<<endl; break;
case 4: cout<<a<< “tonna= “<<a*1000<< “kg”<<endl; break;
case 5: cout<<a<< “sentner= “<<a*100<< “kg”<<endl; break;
default: cout<< “bunaqa o‘lchov yo‘q”;
  }
```

```
system(“pause”);
return 0;
}
```

15. O‘quv masalalarini aniqlovchi 10-40 gacha butun son berilgan.Son kiritilganda unga mos so‘zlarda ifodalovchi dastur tuzilsin. (“yigirmata masala”, “o‘n uchta masala” va h.k.)

```
#include<iostream>
#include<math.h>
using namespace std;
int main()
{ int n;
cout<< “masala raqami=”; cin>>n;
switch(n/10)
  {
case 1: cout<< “o‘n”, break;
case 2: cout<< “yigirma”; break;
case 3: cout<< “ottiz”; break;
case 4: cout<< “qirq”; break;
  }
switch(n%10)
  {
case 1: cout<< “bitta masala”; break;
case 2: cout<< “ikkita masala”; break;
case 3: cout<< “uchta masala”; break;
case 4: cout<< “to‘rtta masala”; break;
case 5: cout<< “beshta masala”; break;
case 6: cout<< “oltita masala”; break;
case 7: cout<< “yettita masala”; break;
case 8: cout<< “sakkizta masala”; break;
```

```
case 9: cout<< “to‘qqizta masala”; break;
default: cout<< “10 dan 40 gacha masala kirit”; break;
  }
system(“pause”);
return 0;
}
```

16. n butun soni va x haqiqiy soni berilgan (n > 0). Quyidagi yig‘indini hisoblovchi dastur tuzilsin. (Olingan natija taxminan sin(x) ga yaqinlashadi)

x – x^3 / (3!) + x^5/ (5!) - … +?(-1)?^n x^(2n+1) /((2*n+1)!)

```
#include <vcl.h>
#pragma hdrstop
#include<iostream.h>
using namespace std;
#pragma argsused
int main(int argc, char* argv[])
{
int n, ishora=1;
float x, maxraj, surat, s=0;
float const pi=3.1415;
cout<< “n = “; cin>>n;
cout<< “x = “; cin>>x;
  x=x*pi/180;
surat=x;
maxraj=1;
  s=x;
for(int i=1; i<=n; i++)
  {ishora*=-1;
surat*=x*x;
maxraj*=2*i*(2*i+1);
     s+=ishora*surat/maxraj;
```

```
cout<<surat<< “/”<<maxraj<<endl;
  }
cout<< “natija =” <<s<<endl;
system(“pause”);
return 0;
}
```

17. n butun soni berilgan (n > 1). Fibonachi ketma – ketlikning dastlabki n ta hadini chiqaruvchi dastur tuzilsin: F1 = 1; F2 = 1; F_k = F_(k-2) + F_(k-1);k = 3, 4, …

```
#include <cstdlib>
#include <iostream>
#include <math.h>
using namespace std;
int main(int argc, char *argv[])
{int n, f1=1,f2=1,f;
cout<< “n=“; cin>>n;
for(int k=1; k <=n; k++)
{if((k==1) || (k==2))
     {
         f=1;
     }
else f=f1+f2;
cout<< f<< “ “;
     f2=f1;
     f1=f;
  }
system(“PAUSE”);
return 0;
}
```

18. A va B butun soni berilgan (A < B). A va B sonlari orasidagi barcha butun sonlarni chiqaruvchi dastur tuzilsin. Bunda har bir son o‘zining qiymaticha chiqarilsin. (Ya’ni 3 soni 3 marta chiqariladi.)

```
#include <cstdlib>
#include <iostream>
#include <math.h>
using namespace std;
int main(int argc, char *argv[])
{
int a,b;
cout<< “a=”; cin>>a;
cout<< “b=”; cin>>b;
for(int i=a; i <=b; i++)
  {
for(int j=1;j<=i;j++)
cout<<i<< “ ”;
cout<<endl;
  }
system(“PAUSE”);
return 0;
}
```

19. Bankka boshlang‘ich S so‘m qo‘yildi. Har oyda bor bo‘lgan summa p foizga oshadi (0 < p < 25). Necha oydan keyin boshlang‘ich qiymat 2 martadan ko‘p bo‘lishini hisoblovchi dastur tuzilsin. Necha oy k – butun son. Bankda hosil bo‘lgan summa haqiqiy son ekranga chiqarilsin.

```
#include<iostream>
#include<conio.h>
using naespace std;
int main()
{
```

```
int p, k=0; float Summa=10000;
cin>> p;
while (Summa <= 20000)
  {
k ++;
     Summa += Summa*p/100;
  }
cout<< “oy =” << k << “\nSumma: “ << Summa;
getch ();
return 0;
}
```

20. N natural soni va N ta butun sondan iborat to‘plam berilgan. Birinchi uchragan eng katta va oxirgi uchragan eng kichik element tartib raqamini aniqlovchi dastur tuzilsin.

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{
int a, n, max, min, mintr, maxtr;
cout<< “n=”; cin >> n;
cin>> a;
max = min = a;
mintr = maxtr = 1;
for (int i = 2; i <= n; i++)
  {
cin>> a;
if (max < a)
    {
max  = a;
```

```
maxtr = i;
   }
if (min >= a)
   {
min  = a;
mintr = i;
   }
  }
cout<< endl;
cout<< “max=” << max << “tartib raqami=”<< maxtr << endl;
cout<< “min=” << min << “ tartib raqami=” << mintr << endl;
getch();
return 0;
  }
```

14. N natural soni va N ta butun sondan iborat to‘plam berilgan. To‘plamdagi ketma – ket keladigan eng katta elementlarning minimal sonini aniqlovchi dastur tuzilsin.

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{
int a, n, soni1, max, soni_min;
cout<< “n=”; cin >> n;
soni_min = n; // faqat bir xil qiymat kiritlsa
cin>> max;
  soni1 = 1;
for (int i = 2; i <= n; i++)
  {
cin>> a;
```

```
if (a > max)
   {
max = a;
     soni1 = 1;
   }
else if (max == a) soni1++;
else
   {
if (soni_min > soni1 && soni1 != 0)
       soni_min = soni1;
     soni1 = 0;
   }
  }
  // max qiymat oxirda bitta kiritilishi mumkin
if (soni_min > soni1 && a == max)
       soni_min = soni1;
cout<< endl;
cout<< “Ketma-ket kelgan max larning min soni=: ” << soni_min << endl;
cout<< “max=” << max << endl;
getch();
return 0;
}
```

15. N natural soni va N ta haqiqiy son berilgan. Bu sonlar kamayish tartibida bo‘lsa – 0, aks holda qonuniyatni buzgan element nomerini chiqaruvchi dastur tuzilsin.

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{
float a1, a;
```

```
int n, nomer = 1;
bool kamayuvchi = true;
cout<< “n=”; cin >> n;
cout<< n << “ta son kiriting\n”;
cin>> a1; // birinchi sonni kiritish
for( int i = 2; i < n; i++ )
  {
cin>> a;
if (a > a1 && kamayuvchi)
    {
kamayuvchi = false;
nomer = i;
    }
    a1 = a;
  }
if (kamayuvchi)
cout<< “kamayuvchi – 0” << endl;
else
cout<< “kamayuvchi emas” << nomer << “son qonuniyatni buzgan” << endl;
system(“pause”);
return 0;
 }
```

2.12 Bob bo‘yicha xulosalar

Mazkur bob sizga C++ tilida dasturlash asoslarini taqdim etdi. Endi siz har qanday chiziqli, tarmoqanuvchi, takrorlanuvchi dasturlar algoritmini tuzish bilan bir qatorda, ushbu algoritmlar asosida dasturlar ham yoza olasiz. Turli muammoviy masalalarni yechishda odatda massivlar tushunchasi keng qo‘llanadi. Shunga ko‘ra, bo‘limlarda keltirilgan misollar ushbu bo‘yicha masalalarni yechishda sizga yordam beradi deb o‘ylaymiz.

III BOB. C++ TILINING QO‘SHIMCHA IMKONIYATLARI

3.1 Grafik imkoniyatlar

C++da grafik holatida ishlash uchun maxsus graphics.h sarlavha fayli mavjud. Bu direktiva o‘zgarmaslar, o‘zgaruvchilar va turli qism dasturlardan tashkil topgan bo‘lib, ular yordamida turli grafik adapterlar bilan har xil tasvirlar chizish mumkin. Adapter kompyuterda graphics.h fayli bilan ishlash imkoniyatini yaratadigan maxsus qurilmadir. Grafik holatiga o‘tilganda ekran alohida-alohida nuqtalarga bo‘linadi. Har bir nuqta o‘z koordinatasiga egadir.

Eng ko‘p ishlatiladigan adapterlar:

1. CGA - color graphics Adapter
2. MCGA - multi color graphics array
3. EGA - enhenced graphics Adapter
4. VGA - video graphics array .

Drayverlarni ko‘rsatish uchun quyidagi o‘zgarmaslar ishlatiladi:

Detect = 0 CGA = 1; MCGA = 2; EGA=3; VGA=9.

Matn holatidan grafik holatiga o‘tish uchun maxsus protseduradan foydalaniladi:

initgraph (&gd, &gm, “ path “);

bu yerda:

gd - drayver nomi;

gm - rejim nomi;

path - kerakli drayver faylning yo‘li. Ko‘pincha gd=0 deb olinadi.

Drayverlar .bgi kengaytmali fayllarida saqlanadi. Agar drayver ishchi katalogning o‘zida joylashgan bo‘lsa, u holda Path = “ “ (bo‘sh belgisi) bo‘ladi.

Grafik holatidan yana matn holatiga o‘tish kerak bo‘lsa, closegraph( ) funksiyasi ishlatiladi.

Chizmalarni hosil qilish uchun quyidagi prosedura va funksiyalar ishlatiladi:

1. putpixel (x, y, color) - x va y koordinatadagi nuqtani colour rangda chizish;

2. getpixel (x, y) - x va y koordinatadagi nuqtaning rangini aniqlaydi;

3. line (x1, y1, x2, y2) - x1 va y1 koordinatadagi nuqtadan x2 va y2 koordinatadagi nuqtagacha kesma chizish;

4. circle (x, y, r) - markazi x va y koordinatada va radiusi R boʻlgan aylana chizish;

5. rectangle (x1, y1, x2, y2) - yuqori chap nuqtasi x1 va y1 koordinatada, oʻng pastki nuqtasi x2 va y2 koordinatada boʻlgan toʻgʻri toʻrtburchakni chizish;

6. setbkcolor (color) - orqa fonga rang berish;

7. setcolor (color) - chizish rangini oʻrnatish (rangli qalam); Bu yerda colour - rang nomeri yoki nomi. Agar rang nomi yoziladigan boʻlsa, uni katta xarflarda yoziladi.

8. bar (x1, y1, x2, y2) - joriy rang va chiziqlar yordamida ichi boʻyalgan toʻgʻri toʻrtburchak chizish;

9. fillellpse (x, y, xr, yr) - markazi x va y da, xr kenglikda va xr balandlikda ichi boʻyalgan rangli ellips chizadi;

10. setfillstyle (style, color) - boʻyash usul va rangni oʻrnatish. Bu yerda style - oʻzgarmas kattalik boʻlib, u quyidagicha boʻlishi mumkin:

0 - sohani fon rangi bilan toʻldirish;

1- sohani rang bilan uzluksiz toʻldirish;

2 -gorizontal chiziqlar

3 - ingichka ogʻma chiziqlar

4 - yoʻgʻon ogʻma chiziqlar

5 - yoʻgʻon ogʻma chiziqlar (boshqa stil)

6 - ogʻma yoʻllar

7 - toʻrtburchakli chiziqlar

8 - ogʻma toʻrtburchaklar

9 - zich ogʻma shrtixlar

10 - siyrak nuqtalar (u yer - bu yerda)

11 - zich nuqtalar bilan

11. getmaxx - joriy rejim va drayverlar uchun nuqtalar sonini aniqlash; getmaxy - joriy rejim va drayverlar uchun vertikal nuqtalar soni. Bu protsedura yordamida kompyuterning oʻzi ekrandagi maksimal nuqtalar sonini aniqlaydi.

12. linerel (x, y) - x va y koordinatali nuqtadan joriy nuqtagacha kesma chizish;

lineto (x, y) - joriy nuqtadan x va y koordinatali nuqtagacha kesma chizish;

13. bar3D (x1, y1, x2, y2, h, top) - parallelepiped chizadi. Bu yerda h - parallelepipedning uzunligi; top - yuqori qismini chizish uchun kerak. Agar topon - boʻlsa tomi bor, agar topoff - boʻlsa tomi yoʻk.

14. arc (x, y, a, b, r) - yoy chizish uchun. Bu yerda x va y - markazning koordinatalari, a - bosh burchak, b - oxirigi burchak, r - yoy radiusi. Burchaklar gradusda qabul qilinadi.

15. ellipse (x, y, a, b, xr, yr) - huddi shu tartibda ellips yoyini chizadi.

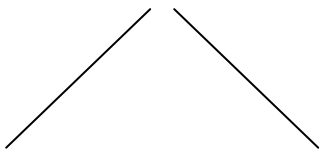
16. drawpoly (n, p) - koʻpburchak chizish uchun. Bu yerda n - koʻpburchakning uchlari soni; p -. Koʻpburchak uchlarining koordinatalari.

17. floodfill (x, y, color) - joriy rang va usuldan foydalangan holda chegaralangan sohani boʻyash. Bu yerda x va y - shu sohaga tegishli boʻlgan biror nuqta koordinatasi. Avval rang, stili keyin chizmalar koʻrsatiladi. Masalan:

setcolor (4); {qizil rangli qalam, chegara rangi}

setfillstyle (1, 2); {1-stil bilan yashil rang bilan boʻyash}

circle (50, 50, 35); {radusi 35 boʻlgan aylana chizish}



floodfill (50, 50, 4); {aylana ichiga rang toʻkish, boʻyaladigan chegara rangi rangli qalam bilan bir xil boʻlishi kerak}

18. setlinestyle (s, a, b) – turli stildagi chiziqlarni chizish uchun; Bu yerda s-style nomeri; a –foydalanuvchi stilini yaratishi mumkin boʻlgan parametr, odatda a=1 deb olinadi; b- chiziqning qalinligini koʻrsatadigan parametr

0 – oddiy chiziq

1 – mayda punktir chiziq

2 – qalin va uzunchoq punktir chiziq

3 – yupqa va uzunchoq punktir chiziq

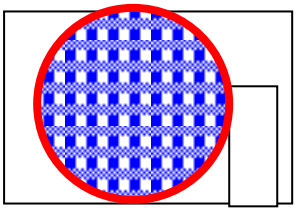
4 – siyrak nuqtali chiziq.

1- Misol:

```
#include <graphics.h>
#include <conio.h>
int main ( )
{ int i, j, gd, gm ;
gd= 0;
initgraph (&gd, &gm, « «);
setcolor (14);                    // sariq qalam
for ( i=0; i<=20; i++)
for ( j=0; j<=20; j++)
circle (i*50, j*30, 55);              // sariq rangli aylanalar
rectangle (0, 0, getmaxx, getmaxy);  //ekran bo‘ylab to‘g‘ri to‘rtburchak
setcolor (11);                    // to‘q feruza rangli qalam
bar3d(200, 300, 100, 150, 30, topon); // parallelepiped, ichi oq
setcolor (CYAN);                  // och feruza rangli qalam
fillellipse (350, 360, 135, 90);       //ellips, ichi oq rangda
getch( );
closegraph( ); }
```

2-misol.

```
int main ( )
{ gd=0;
initgraph (&gd, &gm,’’);
  setbkcolor (BLUE);
  setcolor (14);
 rectangle (120, 130, 240, 250);
setcolor (6);
line (120, 130, 180, 80);
```



```
setcolor (2);
line (180, 80, 240, 130);
setcolor (14);
rectangle (160, 160, 200, 250);
setcolor (4);
setfillstyle(7, 9);
circle( 300, 300, 50);
floodfill (300, 300, 4);
getch( ); closegraph ( );  }
```

Grafik holatida turli shriftlardan foydalanib matnlarni ham yozsa bo‘ladi. Shriftlar .chr kengaytmali fayllarda saqlanadi. Ular .bgi fayllari bilan bitta katalogda saqlanishi shart.

1. outtextxy (x, y,’matn'); - matnni yozish; bu yerda x va y matn boshlanadigan nuqta koordinatalari; masalan: outtextxy (10, 10,’Mirzaev K. 212-15 kif’);

2. settextstyle (sh, n, r); mant shriftini o‘rnatish; bu yerda sh - shrift nomeri (0 - vektorli shrift, 1 - standart shrift); n - shrift yo‘nalishi (0 - chapdan o‘ngga, 1 - quyidan yuqoriga yozish); r - shrift razmeri (oddiy shriftda 1, vektorli shriftda 4 deb olinadi);

3. settextjustify (h, v) - yozilgan satrni tekislaydi. U outtextxy prosedurasidan keyin yoziladi. Bu yerda h - gorizontal tekislash; v - vertikal tekislash; Gorizontal tekislash uchun: 0 - chapga; 1- markazga; 2 – o‘ngga. Vertikal tekislash uchun: 0 - pastga; 1 - markazga; 2 - yuqoriga.

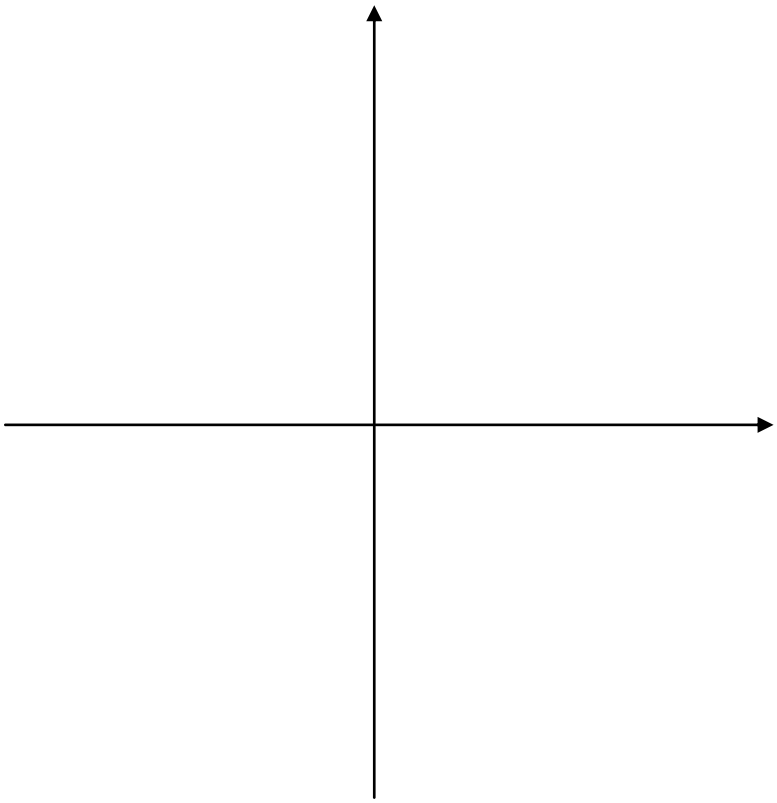
4. setusercharsize - vektor shriftlari uchun bir xil simvollarning eni va bo‘yini o‘rnatadi. Masalan: setUserCharSize(x1, y1, x2, y2);

3-misol. Funksiyalarning grafiklarini chizish.

```
#include <graphics.h>
#include <conio.h>
```

```
#include <math.h>
int main ( )
{ int i, j, gd, gm ; float x, y;
gd=0;  initgraph (&gd, &gm, « «);
setcolor (14);
line (320, 0, 320, 480);
line (0, 240, 640, 240);
line (480, 0, 480, 235);
line (325, 120, 635, 120);
line (160, 245, 160, 475);
line (0, 360, 315, 360);
line (480, 245, 480, 475);
line (325, 360, 635, 360);
x =-10; outtextxy(10, 20,’ y=sin(x) grafigi');
do
{ y = sin(x);
putpixel (160 + 10*x, 120 - y, 5);
 x = x+0.001;  }
while (x<=10);
x1 = -10; outtextxy(10, 20,’ y=cos(x) grafigi');
do
{ y = cos(x1);
putpixel (480 + 20*x1, 120 - 20*y1, 6);
 x1 = x1+0.001;  }
while (x1 < =10);
x2 = -10; outtextxy(10, 20,’ y=exp(x) grafigi');
do
{ y2 = exp(x2);
putpixel (160 + 10*x2, 360 - 20*y2, 7);
 x2 = x2+0.001;  }
```

```
while (x2 <=10);
x3 = -10; outtextxy(10, 20,’ y=ln(x) grafigi');
do
{ y3 = ln(x3);
putpixel (480 + 10*x3, 360 - y3, 8);
 x3 = x3+0.001; }
while (x3 <=10); getch( ); closegraph( );  }
```



#include <graphics.h> direktivasining yana shunday protseduralari mavjudki, ular yordamida chizmalarni ekran bo‘ylab harakatga keltirish mumkin. Figuralarni harakatga keltirishning bir necha usullari bor. Ulardan biri harakatni takrorlanish buyrug‘i orqali tashkil qilishdir. Ikkinchi usul ekranda chizilgan chizma joylashgan sohani massiv ko‘rinishida eslab qolib, uni maxsus protsedura yordamida ekranning kerakli nuqtasiga ko‘chirishdir. Bunda dinamik xotiradan foydalaniladi.

Katta miqdordagi ma’lumotlar ishlatiladigan masalalarni yechishda, kompyuterning grafik imkoniyatlaridan foydalanganimizda xotira hajmi yetishmasligi mumkin. Bunday hollarda dinamik xotira juda qo‘l keladi. Dinamik xotira bu kompyuterning dasturga ma’lumotlar segmentidan tashqari yuklatilgan

tezkor xotiradir. Bu xotira taxminan 200-300 Kbni tashkil qiladi. Dinamik xotiradan foydalanish uchun koʻrsatgichlar ishlatiladi. Bu oʻzgaruvchilarni (koʻrsatgichlarni) xotirada joylashtirishni kompilyator amalga oshiradi. Koʻrsatgich shunday oʻzgaruvchiki, uning qiymati oʻzgaruvchi qiymatiga emas, balki shu oʻzgaruvchi joylashgan xotira adresiga tengdir.

Dinamik xotira sohasidan joy ajratish uchun new operatori ishlatiladi. Bu soʻzdan keyin xotiraga joylashtiriladigan Ob'yekt toifasi aniqlanadi. Masalan: new int; deb yozsak, dinamik xotiradan 2 bayt joy ajratgan boʻlamiz. Masalan: int *p;

p = new int;

yoki int *p = new int ;

Ajratilgan xotira sohasiga biror qiymatni joylashtirish mumkin: *r = 750;

Bu yozuvni quyidagicha oʻqiladi: « r koʻrsatgichida adresi saqlanayotgan xotiraga 750 sonini yozing «.

Dinamik xotira sohasi chegaralangan, u toʻlib qolganda new operatori orqali joy ajratish xatolikka olib keladi. Bu holni biz xotiraning toʻlib ketishi yoki oqib ketishi deymiz (utechka pamyati). Shuning uchun xotira boshqa kerak boʻlmasa uni boʻshatish zarurdir. Buni delete operatori yordamida bajariladi. Masalan: delete p;

Ekranda chizmalarni harakatlantirish uchun kerak boʻladigan proseduralar:

1.imagesize (x1, y1, x2, y2) – ekranning chap yuqori nuqtasi va ung pastki nuqtasi koordinatalaridan toʻgʻri toʻrtburchakli sohani saqlash uchun kerak boʻladigan xotiraning oʻlchami (baytlarda olinadi);

2. getimage (x1, y1, x2, y2, p) – dinamik xotiraning berilgan r maydonida toʻgʻri toʻrtburchakli tasvirni saqlash. Bu yerda r – tasvir saqlanadigan joyning adresini saqlaydigan oʻzgaruvchi, ya'ni koʻrsatgich.

3. putimage (x, y, p, m) – ekranning berilgan joyiga tasvirni chiqarish; bu yerda x va y – xotiraning p maydonidagi tasvirdan nusxa koʻchiriladigan ekran maydonining chap yuqori nuqtasi; m – tasvirni ekranga chiqarish holati. Agar:

m = 0 (NormalPut) - tasvirni koʻchirish. Bunda eskisi oʻchib, yangisi paydo boʻladi (xuddi yurib ketayotgandek)

m = 1 (XorPut)

m = 2 (Orput) –

m = 3 (AndPut)

```
#include < graphics.h >
#include <conio.h >
#include <dos.h >
int main ( )
{ int gd = 0, gm, I, j, s; int *a;
 initgraph(&gd,&gm,»«);
setcolor ( 4 );
circle ( 30, 30, 20 ); putpixel ( 30, 30, 2);
rectangle ( 10, 10, 50, 50);
s = imagesize ( 9, 9, 51, 51);
 *a = new int;  *a = s;
getimage (9, 9, 51, 51, a);
for ( i = 0; i < = 585; i + +)
{ putimage ( i, 10, a, 0); sound (20);  delay (10); nosound ( ); }
for ( j = 10; j < = 420; j + +)
{ putimage(585, j, a, 0 ); sound ( 30 ); delay (10); nosound( );}
for (i = 585; i > = 10; i - -)
{ putimage(i, 420, a, 0); delay ( 10 ); }
for (j = 420; j >10; j - -)
{ putimage(10, j, a, 0); delay( 10 ); }
delete a;
getch(); //closegraph( );
}
```

Muhokama savollari

1. Ekranni matnaviy holatdan grafik holatga o‘tkazish.
2. Ekranni grafik holatdan matnaviy holatga o‘tkazish.

3. Tasvirlarni hosil qilish.

4. Tasvirlarni harakatlantirish.

Nazorat savollari

1. C++ tilining grafik imkoniyatlari.

2. Tasvirlarni hosil qiluvchi funksiyalar.

3. Tasvirlarni harakatlantirish.

4. C/C++ tilida grafika.

5. initgraph protsedurasi.

6. Grafik adapterlar haqida

7. Ekranning fon rangini o‘zgartirish

8. To‘g‘ri chiziq, aylana, to‘g‘rito‘rtburchak, ellips, arka, parallelopipedlarni va hakozolarni chizish asoslari

9. Bo‘yash usullari.

10. Chegaralangan sohani bo‘yash algoritmi

11. Chizish uslublari (stillari)

12. Funksiyalarni chizish asoslari

13. Grafik holatda matnlarning ishlatilishi

3.2 Satrlar va ularga ishlov berish

C++ tilida satrlar (1ta so‘z) simvolli massiv sifatida ishlatiladi. Simvolli massivlar quyidagicha ifodalanadi:

char soz[10];

Simvolli massivlar quyidagicha initsalizasiya qilinishi mumkin:

char shahar[ ]= “TOSHKENT”; Bu holda massiv elementlarining soni avtomatik ravishda aniqlanadi va massiv oxiriga kursorni keyingi satrga ko‘chiradigan \0 belgisi qo‘yiladi.

Yuqoridagi misolni quyidagicha inisalizasiya qilish ham mumkin:

char shahar[ ]= {'T',’O‘,'S','H','K','E','N','T',\0}; bu holda so‘z oxiriga \0 belgisini qo‘yish shart.

Masalan: Berilgan so‘zdagi bir simvolni olib tashlash dasturi.

```
#include <iostream.h>
int main ( )
{
char s[100], c; int i, j;
cin >> s;          // kiritilayotgan so‘z
cin >>c;           // kiritilayotgan simvol
for (i=j=0; s[i]!='\0'; i++)
if (s[i] != c) s[j++] = s[i];
// kir. simvolning barchasini olib tashlaydi
s[j]='\0';
cout << “yangi soz=” <<s;
}
```

C++ tilida so‘zlar massivlari ikki o‘lchamli massiv sifatida ishlatiladi. Masalan: char nom [4][6]; bu ta’rif yordamida har biri 6ta simvoldan iborat 4ta so‘zli massiv berilgan. So‘zli massivlar quyidagicha initsalizatsiya qilinishi mumkin:

char ismlar [3][10] = { ”Anvar”, ”Mirkomil”, ”Abdulla”};

Har bir so‘z uchun xotiradan 10 bayt joy ajratiladi va har bir so‘z oxiriga \0 belgisi qo‘yiladi.

So‘zli massivlar initsalizatsiya qilinganida so‘zlar soni ko‘rsatilmasligi ham mumkin, bu holda so‘zlar soni berilganlarga qarab avtomatik ravishda aniqlanadi. Masalan:

char comp[ ][9]={”kompyuter”, ”printer”, ”kartrij”};

1-Misol: berilgan simvoldan (harfdan) boshlanadigan so‘zlar ro‘yxatini chiqaring.

```
So’zlar soni: 3
So’zlarni kiriting:
Nodir
Bobur
Sobir
Simvolni kiriting:
B
Natija:
Bobur
```

```
#include <iostream.h>
int main ( )
{
char a[] [10], b;  int i, n;
cout<< “so‘zlar sonini kiriting=”;  cin>>n;
cout<<“so‘zlarni kiriting=”;
for (i=0; i<=n; i++)
cin >> a[i];
cout << ”simvolni kiriting=”;  cin >> b;
for (i = 0; i <= n; i++)
if (a[i][0] = = b)  cout << a[i] << endl;
}
```

2-misol. Talabalar ro‘yxati berilgan (familiyasi). 1ta fan nomi va shu fandan talabalarning olgan baholari kiritilgan. Yomon baho olgan talabalar ro‘yxatini chiqarish dasturi tuzilsin.

```
#include <iostream.h>
main ( )
{ char a[20][10], s[10]; int k[20], i;
cout << “fan nomini kiriting=”;  cin >>s;
cout << “familiyalar:”<<endl;
for (i=0; i<20; i++)
cin >>a[i];
cout << “baholar:”<<endl;
for (i=0; i<20; i++)
cin >>k[i];
for (i=0; i<20; i++)
if (k[i] = =2) cout << “Yomon baho olganlar:”<<a[i]<<endl;
```

}

Massivlarni tashkil etishda ko‘rsatgichlardan foydalanilsa ham bo‘ladi. Bunda massivlar quyidagicha e’lon qilinadi:

toifa *massiv nomi [con];

Ko‘rsatgichlarni simvolli massivlarga ishlatish qulaydir. Masalan:

char fam[][10] = {“Olimov”, “Rahimov”, “Ergashev”} deb yozilsa, xotirada 30 elementdan iborat massiv hosil bo‘ladi, chunki har bir familiya 10ta harfga yetmasa ham oxiri 0(nol)lar bilan to‘ldiriladi. Uni ko‘rsatgichlar yordamida ta’riflansa, ya’ni

char *fam[][10] = {“Olimov”, “Rahimov”, “Ergashev”}

deb yozilsa, xotirada 24ta elementdan iborat bo‘lgan massiv hosil bo‘ladi, chunki familiyalardagi har bir harflar sanaladi. Shunda xotira ham tejaladi.

Izoh: har bir familiya oxiriga \0 belgisi qo‘yiladi.

C++ tilida satrli ma’lumotlar ustida bir qancha amallarni bajarish mumkin. Buning uchun avvalo #include <string.h> ni ulash kerak bo‘ladi.

1. a) 2ta satrli o‘zgaruvchilarni bir-biriga ulash

strcat (s1, s2); ya’ni s2 satrni s1 satrga ulash.

b) 2-satrning n ta simvolini 1-satrga ulash

strncat (s1, s2, n); ya’ni s2 satrning n ta simvolini s1 satrga ulash.

Natija doim 1-ko‘rsatilgan satrga o‘zlashtiriladi.

Masalan;

1) #include < string.h >

#include < iostream.h>

void main( )

{ char s1[10] = “Gul”, s2[10] = “bahor”;

strcat (s1, s2);

cout << s1 << endl; // yoki cout << strcat (s1, s2)<< endl;

}

Natija; Gulbahor

Agar cout << strcat (s2, s1)<< endl;

deb yozsak natija bahorGul chiqadi.

2) #include < string.h >

#include < iostream.h>

void main( )

{ char s1[10] = ”Gul”, s2[10] = “bahor”;

strncat (s1, s2,4);

cout << s1 << endl; // yoki cout << strncat (s1, s2,4)<< endl;

}

Natija; Gulbaho

2. a) 2-satrning nusxasini 1-satrga qo‘yish

strcpy (s1, s2);

b) 2-satrdan n ta simvolni 1-satrga nusxasini qo‘yish

strncpy (s1, s2, n);

Masalan:

#include < string.h >

#include < iostream.h>

void main( )

{ char s1[10] = ” Dilorom ”, s2[10] = “ Gul ”;

strncpy (s1, s2, 3);

cout << s1 << endl; // yoki cout << strncpy (s1, s2, 3)<< endl;

}

Natija: Dilorom

Agar strncpy (s2, s1, 3); deb yozilsa, natija Dil chiqadi.

3. 2ta satr o‘zgaruvchisini solishtirish.

a) strcmp (s1, s2);

Agar s1 > s2; natija + son

Agar s1 < s2 ; natija – son

Agar (s1=s2) ular aynan teng bo‘lsalar 0 chiqadi.

Masalan;

char s1[10] = “ Gulnora “, s2[10]=” Guli “; int n;

n=strcmp (s1, s2); n +

n=strcmp (s2, s1); n –

yoki:

char s1[10] = “ Guli “, s2[10]=” Gulnora “; int n;

n=strcmp (s1, s2); n -

n=strcmp (s2, s1); n +

b) strncmp (s1, s2, n); - 1-satrning n ta simvolini 2-satr bilan solishtirish.

4. Satr uzunligini hisoblash strlen (s);

Masalan:

char s = “Dastur”; int n;

n = strlen (s);

cout << n << endl;

Natija ; n=6

5. a) Satr o‘zgaruvchisining boshidan simvollarini berilgan simvol bilan almashtirish:

strset (s, c);

s – berilgan satr o‘zgaruvchisi; c – simvol

b) Satr o‘zgaruvchisining boshidan n ta simvolini berilgan simvol bilan almashtirish:

strnset (s, c, n);

Masalan: char s = “Hilola”, c =‘Z’; int n = 1;

strnset (s, c, n);

cout << s<< endl;

Natija; Zilola

char s = “Hilola”, c =‘Z’;

strset (s, c);

```
        cout << s<< endl;
```

Natija; ZZZZZZ

1-misol. Palindrom masalasi. Palindrom so‘z deb o‘ngiga ham, teskarisiga ham bir xil o‘qiladigan so‘zga aytiladi.

```
#include <iostream.h>
#include <string.h>
int main ( )
{ char s[20], sp[20]=” “; int n, i;
cin >> s ;              // berilgan so‘z
n = strlen (s);          // so‘z uzunligi
for ( i=0; i < n; i++)
sp [(n-1) – i] = s [ i ];
if ( strcmp( s, sp) = = 0)  // so‘zlar bir hil bo‘lsa
cout <<”palindrom so‘z=”<< sp <<endl;
else cout <<”palindrom so‘z emas =”<< sp <<endl;
}
```

2-misol.

```
#include <iostream.h>
#include <string.h>
int main ( )
{ char d[20], *b = “ “, *c = “Dilafruz”, *a = “Aripova”;
strcpy ( d, a );
strcat ( d, b );
strcat ( d, c );
cout << d << endl;
}
```

Natija: Aripova Dilafruz

Yuqoridagi funksiyalardan tashqari yana satr toifasidagi o‘zgaruvchilar ustida quyidagi funksiyalarni ham ishlatish mumkin va ularni ishlatish uchun #include <stdlib. h> ni ulash kerak.

1. Satr toifasidagi sonni butun son toifasiga o‘tkazish

```
char → int  atoi ( s );
int n; char *s = “12345”;
n = atoi ( s );
cout << n <<endl;
H : n = 12345;
```

```
int n; char *s= “12345.67”;
n = atoi ( s );
cout << n <<endl;
H : n = 12345;
```

2. Satr toifasidagi sonni haqiqiy son toifasiga o‘tkazish

```
char → float  atof ( s );
float n; char *s = “12345.67”;
n = atof ( s );
cout << n <<endl;
H : n = 12345.67;
```

```
float n; char *s = “123e+30 ”;
n = atof ( s );
cout << n <<endl;
H : n = 1.23e+32;
```

3. Satr toifasidagi sonni uzun butun son toifaga o‘tkazish

```
char → long  atol ( s );
long n; char *s = “1234567890”;
n = atol ( s );
cout << n <<endl;
H : n = 1234567890;
```

```
long n; char *s = “123456789.67 ”;
n = atol ( s );
cout << n <<endl;
H : n = 123456789;
```

4. Butun toifadagi o‘zgaruvchini satr toifasiga o‘tkazish

float → char itoa (a, b, c);

Bu yerda:

a – int toifasidagi o‘zgaruvchi

b – char toifasidagi o‘zgaruvchi

c – c.c. asosi

```
int a = 1234, c = 10; char *b;
itoa (a, b, c);
cout << b <<endl;
H : b = 1234;
int a = 1234, c = 8; char *b;
itoa (a, b, c);
cout << b <<endl;
H : b = 30071;
```

```
int a = 12.34, c = 10; char *b;
itoa (a, b, c);
cout << b <<endl;
H : b = 12;
```

5. Uzun butun toifasidagi o‘zgaruvchini satr toifasiga o‘tkazish

long → char ltoa (a, b, c);

Bu yerda:

a – long toifasidagi o‘zgaruvchi

b – char toifasidagi o‘zgaruvchi

c – c.c. asosi

```
long a = 123456789, c = 10;
char *b;
ltoa (a, b, c);
cout << b <<endl;
H : b = 123456789;
```

7. Ishorasiz uzun butun toifasidagi o‘zgaruvchini satr toifasiga o‘tkazish

unsigned long → char ultoa (a, b, c);

Bu yerda:

a – ishorasiz uzun toifasidagi o‘zgaruvchi

b – char toifasidagi o‘zgaruvchi

c – c.c. asosi

```
unsigned long a = 1234567890, c = 10; char *b;
ultoa (a, b, c);
cout << b <<endl;
H : b = 1234567890;
```

Muhokama savollari

1. Simvolli massivlar.
2. Simvolli massivni kiritish.
3. Matnga ishlov berish.
4. Satrlarni tashkil etish

Nazorat savollari

1. Simvolli massivlar tushunchasi, xotiraning taqsimlanishi.
2. Simvolli massivni kiritish usullari.
3. Matnga ishlov berish yollari.
4. Satrlarni tashkil etish bo‘yicha tavsiyalar.
5. Dasturda satrlardan foydalanish.
6. Satrlarga ishlov beruvchi standart funksiyalar.

3.3 Xotiraning taqsimlanishi. Dinamik xotira

Dasturchi o‘z dasturi ustida ish boshlashi bilan operatsion tizim (DOS, WINDOWS) kompilyator talabiga muvofiq xotira sohasidan joy ajratadi. Mavjud xotira 5ta sohaga ajratiladi:

1. Global o‘zgaruvchilar sohasi
2. Bo‘sh yoki dinamik xotira
3. Registrli xotira
4. Dastur segmenti
5. Stekli xotira

Lokal o‘zgaruvchilar va funksiya parametrlari stekli xotiraga joylashtiriladi. Stek – bu dasturdagi funksiya chaqirilganda undagi ma’lumotlar uchun talab qilinadigan xotiraning maxsus sohasidir. Uning stek deb atalishiga sabab, «oxirigi kelgan 1-ketadi» jarayoni asosida ishlashidir. Agar bir funksiya ikkinchisini chaqirsa, ikkinchi funksiya o‘z ishini tugatganidan so‘ng 1-funksiya o‘z ishini

yakunlaydi, ya’ni oxirgi chaqirilgan funksiya birinchi bo‘lib ishini tugatadi. (taxlangan tarelkalar kabi). Lokal o‘zgaruvchilarning o‘ziga xos xususiyati shundan iboratki, dastur boshqaruvi unga tegishli funksiyadan chiqishi bilan unga ajratilgan xotira sohasi bo‘shatiladi, ya’ni bu o‘zgaruvchilar o‘chiriladi.

Dastur kodi, ya’ni barcha operatorlar va amallar ketma-ketligi dastur segmentida joylashtiriladi.

Registrli xotira dasturdagi o‘zgaruvchilarni operativ xotirada emas, balki markaziy protsessorning biror registrida saqlash uchun xizmat qiladi.

Global o‘zgaruvchilar dastur ishga tushirilganidan boshlab toki o‘z ishini yakunlagunicha xotiradan o‘chirilmaydi. Ixtiyoriy joyda undan foydalanish mumkin, ya’ni bu o‘zgaruvchilar xotiradan doimiy joy olib turadi. Ularni o‘chirib bo‘lmaydi. Bu muammoni dinamik xotira hal etadi.

Dinamik xotirani axborotlar yozilgan yacheykalarning nomerlangan to‘plami sifatida qarash mumkin. Bu xotira yacheykalarini nomlash mumkin emas. Ularga murojaat kerakli yacheyka adresini o‘zida saqlovchi ko‘rsatgichlar orqali amalga oshiriladi.

Dinamik xotira dastur ishini tugatganida ham bo‘shatilmaydi, bunday xotirani bo‘shatish bilan dasturchining o‘zi shug‘ullanadi.

Xo‘sh, dinamik xotirani bo‘shatish nima uchun kerak?

O‘zgaruvchi uchun ajratilgan xotira sohasi bo‘shatilmagunicha bu joydan foydalanish mumkin emas. Agar dinamik xotira funksiya bilan ishlash jarayonida ajratilgan bo‘lsa, funksiya ishini tugatganidan keyin ham biz undan bemalol foydalanishimiz mumkin. Dinamik xotiraning yaxshi tomoni undagi ma’lumotlarga faqatgina uning ko‘rsatgichiga murojaat qilish orqaligina bo‘ladi. Bu esa bizga joriy ma’lumotlarni o‘zgartirishni qat’iy nazorat qilish imkoniyatini beradi.

Dinamik xotira sohasidan joy ajratish uchun new operatori ishlatiladi. Bu so‘zdan keyin xotiraga joylashtiriladigan ob’yekt toifasi aniqlanadi.

Masalan: new int; deb yozsak, standart holatda dinamik xotiradan 2 bayt joy ajratgan bo‘lamiz.

new operatori natija sifatida belgilangan xotira yacheykasining adresini qaytaradi. Bu adress koʻrsatgichga oʻzlashtiriladi.

Masalan: int *p;

p = new int;

yoki int *p = new int ;

Ajratilgan xotira sohasiga biror qiymatni joylashtirish mumkin:

*r = 750 ;

Bu yozuv quyidagicha oʻqiladi: « r koʻrsatgichida adresi saqlanayotgan xotiraga 750 sonini yozing «.

Dinamik xotira sohasi chegaralangan, u toʻlib qolganda new operatori orqali joy ajratish xatolikka olib keladi. Bu holni biz xotiraning toʻlib ketishi yoki oqib ketishi deymiz (utechka pamyati). Shuning uchun xotira boshqa kerak boʻlmasa uni boʻshatish zarurdir. Buni delete operatori yordamida bajariladi. Masalan: delete p;

Bunda koʻrsatgich oʻchirilmaydi, balki unda saqlanayotgan adresdagi xotira sohasi boʻshatiladi. Belgilangan xotiraning boʻshatilishi koʻrsatgichga ta’sir qilmaydi, unga boshqa adresni oʻzlashtirish ham mumkin.

Masalan:

```
#include < iostream.h >
int main ( )
{ int t = 5, h = 7 ;
  int *p = &t ;
  cout << t << endl;
  cout << *p << endl;
  cout << &h << endl;
  p = new int ;  *p=9;
  cout << *p << endl;
  cout << t << endl;
  delete p; }
```

Натижада: t=5; *p = 5;
&h = 0xfff2;
*p = 9; t = 5;

Muhokama savollari

1. Xotira turlaridan foydalanish.
2. Dinamik xotira sohasidan joy ajratish.
3. Adres bilan ishlash.

Nazorat savollari

1. Dinamik xotira tushunchasi.
2. Joy ajratish operatori.
3. Adres bilan ishlash.
4. Koʻrsatgichlarni qoʻllash.
5. Dinamik xotirani nima uchun boʻshatish kerak?

3.4 Foydalanuvchi toifasini yaratish

C++ tilida foydalanuvchi oʻzining sarlavha faylini yaratishi va undan kerakli vaqtlarda keng foydalanishi imkoniyati berilgan. Buning uchun kerakli amallar algoritmi funksiya va protsedura sifatidagi dasturi bloknotga yoziladi. Soʻngra bu dasturga ixtiyoriy nom (lotincha) beriladi, fayl nomidan soʻng .h kengaytmasi beriladi. h - header - sarlavha degan maʼnoni bildiradi. Bu faylni INCLUDE papkasiga saqlash shart! Foydalanuvchi bu include da avvaldan mavjud boʻlgan include-lardan, istalgan oʻzgaruvchi, oʻzgarmaslar, funksiya va protseduralardan foydalanishi mumkin. Dasturchi oʻzining shaxsiy include ini ishlatishi uchun asosiy dasturda (C/C++ muhitida) ularni eʼlon qilishi va undagi funksiya va proseduralarning nomlari hamda haqiqiy parametrlarining oʻrnini bilishi kerak. Shaxsiy include-larni chaqirishda < > yoki « « belgilari ishlatilishi mumkin.

Masalan: arccos x = arctg ( ) formulasi yordamida tashkil etuvchi include yaratamiz va undan foydalanamiz.

Bloknotdagi dasturi:

```
#include < math.h >
float  acos (float x)
{ float y;
y = atan( sqrt ( 1-x*x) / x );
return y;
}
```

Bu faylga ixtiyoriy nom beramiz, masalan: farruh.h so‘ngra undan foydalanib dastur tuzamiz: (C/C++ muhitida)

```
#include <iostrream.h >
#include <conio.h >
#include <farruh.h >
int main ( )
{ float x, y;
cin >> x;
y = acos (x);
cout << “y=” << y << endl;
getch ( );
}
```

Odatda yaratilayotgan includelarga barcha kerakli funksiyalar guruhlab joylashtiriladi va bir yo‘la chaqirib ishlatiladi. Masalan:

```
#include <math. h >
float acos ( float x)
{ ……  }

float  asin ( float x )
{ ……. }

float sh ( float x )
{ …….. }
 va hokazo.
```

2-misol. Uchburchak va beshburchak chizadigan protsedura uchun include yarating va undan foydalaning.

Bloknotda: (nomi chiz . h bo‘lsin)

```
#include < graphics. h >
int gd=0, gm; initgraph (&gd, &gm, “ ”);
void uchb ( int x1, int y1, int x2, int y2, int x3, int y3)
{ line (x1, y1, x2, y2);
  line (x1, y1, x3, y3);
```

```
  line (x2, y2, x3, y3 ); }
void besh ( int a[ ] )
{ drawpoly ( 6, a );}
```

Endi bu sarlavha fayldan foydalanamiz:

```
#include < chiz. h >
#include < conio. h >
{ int main ( )
setcolor (4);
uchb ( 10, 20, 100, 80, 250, 150);
int a[ ] = { 10, 200, 30, 50, 70, 50, 90, 200, 50, 350, 10, 200);
besh (a);
getch ( );
}
```

Muhokama savollari

1. Foydalanuvchining direktivalarini (include) yaratish.
2. Tuzilmalarni tashkil etish
3. Foydalanuvchi toifasini tashkil etish.
4. Massivlar- tuzilma elementlari.

Nazorat savollari

1. Foydalanuvchining sarlavha faylini yaratishida dasturga talablar?
2. Foydalanuvchining sarlavha faylini yaratish yo‘llari.
3. h –kenggaytmali fayllarning vazifasi?

3.5 Tuzilmalar va birlashmalar

Bir necha xil toifadagi o‘zgaruvchilarni bitta nom bilan birlashtirib ishlatish tuzilma deyiladi. Tuzilmalar murakkab kattaliklarni o‘zaro bog‘lab, yagona toifa ostida dastur tuzishda yordam beradi. Tuzilmalar quyidagicha ta’riflanadi:

```
struct tuzilma nomi
{
  elementlar ta’rifi;
}
```

Tuzilma nomi identifikatorlarni aniqlash qoidasiga bo‘ysunadi.

Misol uchun ombordagi mollarni tasvirlovchi tuzilma hosil qilamiz. Unda: mol nomi (char[10]); sotib olish narxi (long); ustama haq, foiz(float); mollar soni (int); mol kelib tushgan sana (char[9]). Bu tuzilma dasturda quyidagicha ta’riflanadi:

```
struct ombor
{
char name[10]; // mol nomi 10ta simvoldan oshmaydi
long nn;      // mol narxi
float f;       // foizi
int i;        //  mollar soni
char date[9];  //  sana 9ta simvoldan iborat
}
```

Tuzilmalar tarkibida ixtiyoriy toifadagi o‘zgaruvchilar, massivlar qatnashishlari mumkin.

Tuzilmalar ta’riflanganda konkret ro‘yxatni kiritish mumkin, ya’ni

```
struct tuzilma nomi
{
  elementlar ta’rifi;
} konkret ro‘yxat;
```

Masalan: talabalar ro‘yxati uchun tuzilma hosil qilish:

```
struct student
{ char name[10];
  char fam[10];
  int ty;} student1, student2,student3;
```

Bu holda student tuzilmali toifa bilan birga 3ta konkret roʻyxat kiritilgan va har biri uchun ismi, familiyasi, tugʻilgan yili beriladi.

Tuzilmali toifa taʼriflanganda tuzilma nomi koʻrsatilmasdan, toʻgʻridan toʻgʻri konkret roʻyxat koʻrsatilishi ham mumkin:

```
struct
{
  elementlar taʼrifi;
} konkret roʻyxat;
```

Masalan: yuqoridagi yozuvni quyidagicha yozish mumkin:

```
struct
{ char name[10];
  char fam[10];
  int ty;} student1, student2,student3;
yoki
struct
{ char prossesor[10];
  char matplat[10];
  int memory;
  int disk; } IBM_486, Pentium_4, Compaq;
```

Tuzilma elementlariga quyidagicha murojaat qilish mumkin:

tuzilma nomi. element nomi;

Bu yerda nuqta amali tuzilma elementiga murojaat qilish amali deyiladi. Bu amal qavs amaliga oʻxshab yuqori ustuvorlikka egadir.

Masalan: student1.name; student2.ty; student3.fam;

student1.name = “Anvar”; student2.ty = 1990; student3.fam = “Solijonov”;

Tuzilmalarni ishlatishda koʻpincha typedef soʻzidan foydalaniladi. Bundan maqsad tuzilmadagi kattaliklardan yangi toifa yaratib, undan dasturda keng

foydalanish imkoniyatini yaratishdir. Uni foydalanuvchining (dasturchining) toifasi deb yuritiladi. typedef da yangi nom ishlatiladi va shu nom keyinchalik boshqa o‘zgaruvchilarni belgilashda ishlatiladi. Misol:

```
typedef struct
{ float f;
  char st[10];
  double d, d1;} bbb;
bbb  nnn[25];
```

Misol uchun quyidagi dasturni ko‘ramiz. Unda kompleks sonlarni qo‘shish amali qurilgan. Kompleks sonlar haqiqiy va mavhum qismlardan tashkil topadi. Ular alohida alohida qo‘shiladi.

```
#include <iostream.h>
#include <conio.h>
typedef struct
{ double real;
  double imag; } complex;
void main( )
{ complex x, y, z;
cin >>x.real; cin >>x.imag;
cin >>y.real; cin >>y.imag;
z.real = x.real + y.real;
z.imag = x.imag + y.imag;
cout << z.real; cout << z.imag;
getch( );
}
```

Massivlarni ham tuzilma elementlari sifatida ishlatish mumkin. Massiv elementlarining son qiymatlarini initsalizatsiya qilish ham mumkin. Masalan:

```
struct { float a; int mas[4] } pp = { 5.34, {1,2,3,4}};
```

Masalan: guruhda 20ta talaba bor. Talabalarning familiyasi va informatika fanining 3ta modulidan olgan baholari berilgan. Shu fan bo‘yicha qarzdor talabalar haqida ma’lumot beruvchi dastur tuzing.

```
#include <iostream.h>
#include <conio.h>
int main ( )
{
typedef struct
{ char fam[15]; int mod1, mod2, mod3;} gr;
gr gr211_08[20]; int i, f;  clrscr( );
cout <<”malumotlarni kiriting:\n”;
for (i = 1; i<=20; i++)
{ cout <<”familiyasini kiriting: \n”;
cin >>gr211_08[i].fam;
cout <<”ballarini kiriting:\n”;
{ cin >> gr211_08[i].mod1;
cin >> gr211_08[i].mod2;
cin >> gr211_08[i].mod3;} }
f=0;
for(i=1; i<=20; i++)
if (gr211_08[i].mod1= =2 || gr211_08[i].mod2= =2 || gr211_08[i].mod3= =2)
{
cout <<”Qarzdor talaba:”<<gr211_08[i].fam<<endl;
f=f+1; }
cout <<”qarzdor talabalar soni:”<<f<<endl;
getch(); }
```

Ma’lumki, biror predmet sohasidagi masalani yechishda undagi Ob’yektlar bir nechta, har xil turdagi parametrlar bilan aniqlanishi mumkin. Masalan, tekislikdagi nuqta haqiqiy turdagi x- absissa va y- ordinata juftligi - (x,y) ko‘rinishida beriladi. Talaba haqidagi ma’lumotlar: satr turidagi talaba familiya,

ismi va sharifi, mutaxassislik yo‘nalish, talaba yashash adresi, butun turdagi tug‘ilgan yili, o‘quv bosqichi, haqiqiy turdagi reyting bali, mantiqiy turdagi talaba jinsi haqidagi ma’lumot va boshqalardan shakllanadi.

Dasturda holat yoki tushunchani tavsiflovchi har bir berilganlar uchun alohida o‘zgaruvchi aniqlab masalani yechish mumkin. Lekin bu holda ob’yekt haqidagi ma’lumotlar «tarqoq» bo‘ladi, ularni qayta ishlash murakkablashadi, ob’yekt haqidagi berilganlarni yaxlit holda ko‘rish qiyinlashadi.

C++ tilida bir yoki har xil turdagi berilganlarni jamlanmasi *struktura* deb nomlanadi. Struktura foydalanuvchi tomonidan aniqlangan berilganlarning yangi turi hisoblanadi. Struktura quyidagicha aniqlanadi:

struct <struktura nomi>
{
  <tur_1 > <nom_1>;
  <tur_2 > <nom_2>;
 …
  <tur_n> <nom_n>;
};

Bu yerda <struktura nomi> - struktura ko‘rinishida yaratilayotgan yangi turning nomi, “<tur_i> <nom_i>;” - strukturaning i-maydonining (nom_i) e’loni.

Boshqacha aytganda, struktura e’lon qilingan o‘zgaruvchilardan (maydonlardan) tashkil topadi. Unga har xil turdagi berilganlarni o‘z ichiga oluvchi *qobiq* deb qarash mumkin. Qobiqdagi berilganlarni yaxlit holda ko‘chirish, tashqi qurilmalar (binar fayllarga) yozish, o‘qish mumkin bo‘ladi.

Talaba haqidagi berilganlarni o‘z ichiga oluvchi struktura turining e’lon qilinishini ko‘raylik.

struct Talaba
{
 char FISh[30];
 unsigned int Tug_yil;
 unsigned int Kurs;

```
char Yunalish[50];
float Reyting;
unsigned char Jinsi[5];
char Manzil[50];
bool status;
};
```

Dasturda strukturalardan foydalanish, shu turdagi o‘zgaruvchilar e’lon qilish va ularni qayta ishlash orqali amalga oshiriladi:

Talaba talaba;

Struktura turini e’lonida turning nomi bo‘lmasligi mumkin, lekin bu holda struktura aniqlanishidan keyin albatta o‘zgaruvchilar nomlari yozilishi kerak:

```
struct
{
unsigned int x,y;
unsigned char Rang;
} Nuqta1, Nuqta2;
```

Keltirilgan misolda struktura turidagi Nuqta1, Nuqta2 o‘zgaruvchilari e’lon qilingan.

Struktura turidagi o‘zgaruvchilar bilan ishlash, uning maydonlari bilan ishlashni anglatadi. Struktura maydoniga murojaat qilish’.’ (nuqta) orqali amalga oshiriladi. Bunda struktura turidagi o‘zgaruvchi nomi, undan keyin nuqta qo‘yiladi va maydon o‘zgaruvchisining nomi yoziladi. Masalan, talaba haqidagi struktura maydonlariga murojaat quyidagicha bo‘ladi:

```
talaba.Kurs=2;
talaba.Tug_yil=1988;
strcpy(talaba.FISh,”Abdullaev A.A.”);
strcpy(talaba.Yo‘nalish, “Kompyuter injiniringi”);
strcpy(talaba.Jinsi, “Erkak”);
strcpy(talaba.Manzil, “Toshkent,Yunusobod 6-3-8, tel: 224-45-78”);
talaba.Reyting=123.52;
```

Keltirilgan misolda talaba strukturasining son turidagi maydonlariga oddiy ko‘rinishda qiymatlar berilgan, satr turidagi maydonlar uchun strcpy funksiyasi orqali qiymat berish amalga oshirilgan.

Struktura turidagi ob’yektning xotiradan qancha joy egallaganligini sizeof funksiyasi (operatori) orqali aniqlash mumkin:

int i=sizeof(Talaba);

Ayrim hollarda struktura maydonlari o‘lchamini bitlarda aniqlash orqali egallanadigan xotirani kamaytirish mumkin. Buning uchun struktura maydoni quyidagicha e’lon qilinadi:

<maydon nomi> : <o‘zgarmas ifoda>

Bu yerda <maydon nomi>- maydon turi va nomi, <o‘zgarmas ifoda>- maydonning bitlardagi uzunligi. Maydon turi butun turlar bo‘lishi kerak (int, long, unsigned, char).

Agar foydalanuvchi strukturaning maydoni faqat 0 va 1 qiymatini qabul qilishini bilsa, bu maydon uchun bir bit joy ajratishi mumkin (bir bayt yoki ikki bayt o‘rniga). Xotirani tejash evaziga maydon ustida amal bajarishda razryadli arifmetikani qo‘llash zarur bo‘ladi.

Misol uchun sana-vaqt bilan bog‘liq strukturani yaratishning ikkita variantini ko‘raylik. Struktura yil, oy, kun, soat, minut va sekund maydonlaridan iborat bo‘lsin va uni quyidagicha aniqlash mumkin:

```
struct Sana_vaqt
{
 unsigned short Yil;
 unsigned short Oy;
 unsigned short Kun;
 unsigned short Soat;
 unsigned short Minut;
 unsigned short Sekund;
};
```

Bunday aniqlashda Sana_vaqt strukturasi xotirada 6 maydon*2 bayt=12 bayt joy egallaydi. Agar e’tibor berilsa, strukturada ortiqcha joy egallangan holatlar mavjud. Masalan, yil uchun qiymati 0 sonidan 99 sonigacha qiymat bilan aniqlanishi yetarli (masalan, 2008 yilni 8 qiymati bilan ifodalash mumkin). Shuning uchun unga 2 bayt emas, balki 7 bit ajratish yetarli. Xuddi shunday oy uchun 1..12 qiymatlarini ifodalashga 4 bit joy yetarli va hokazo.

Yuqorida keltirilgan cheklovlardan keyin sana-vaqt strukturasini tejamli variantini aniqlash mumkin:

```
struct Sana_vaqt2
{
 unsigned Yil:7;
 unsigned Oy:4;
 unsigned Kun:5;
 unsigned Soat:6;
 unsigned Minut:6;
 unsigned Sekund:6;
};
```

Bu struktura xotiradan 5 bayt joy egallaydi.

Strukturalar funksiya argumenti sifatida ishlatilishi mumkin. Buning uchun funksiya prototipida struktura turi ko‘rsatilishi kerak bo‘ladi. Masalan, talaba haqidagi berilganlarni o‘z ichiga oluvchi Talaba strukturasi turidagi berilganlarni Talaba_Manzili() funksiyasiga parametr sifatida berish uchun funksiya prototipi quyidagi ko‘rinishda bo‘lishi kerak:

```
void Talaba_Manzili(Talaba);
```

Funksiyaga strukturani argument sifatida uzatishga misol sifatidagi dasturning matni:

```
#include <iostream.h>
#include <string.h>
struct Talaba
{
```

```
 char FISh[30];
 unsigned int Tug_yil;
 unsigned int Kurs;
 char Yunalish[50];
 float Reyting;
 unsigned char Jinsi[5];
 char Manzil[50];
 bool status;
};
void Talaba_Manzili(Talaba);
int main(int argc,char* argv[])
{
 Talaba talaba;
 talaba.Kurs=2;
 talaba.Tug_yil=1988;
 strcpy(talaba.FISh, “Abdullaev A.A.”);
 strcpy(talaba.Yunalish, “Kompyuter injiniringi”);
 strcpy(talaba.Jinsi, “Erkak”);
 strcpy(talaba.Manzil, “Toshkent, Yunusobod 6-3-8, tel: 224-45-78”);
 talaba.Reyting=123.52;
 Talaba_Manzili(talaba);
return 0;
}
void Talaba_Manzili(Talaba t)
{
 cout<< “Talaba FIO: “ <<t.FIO<<endl;
 cout<< “Manzili: “<<t.Manzil<<endl;
}
```

Dastur bosh funksiyasida talaba strukturasi aniqlanib, uning maydonlariga qiymatlar beriladi. Keyin talaba strukturasi Talaba_Manzili() funksiyasiga

argument sifatida uzatiladi. Dastur ishlashi natijasida ekranga quyidagi ma’lumotlar chop etiladi.

Talaba FIO: Abdullaev A.A.

Manzili: Toshkent, Yunusobod 6-3-8, tel: 224-45-78

Strukturalar massivi

O‘z-o‘zidan ma’lumki, struktura turidagi yagona berilgan bilan yechish mumkin bo‘lgan masalalar doirasi juda tor va aksariyat holatlarda, qo‘yilgan masala strukturalar majmuasini ishlatishni talab qiladi. Bu turdagi masalalarga berilganlar bazasini qayta ishlash masalalari deb qarash mumkin.

Strukturalar massivini e’lon qilish xuddi standart massivlarni e’lon qilishdek, farqi massiv turi o‘rnida foydalanuvchi tomonidan aniqlangan struktura turining nomi yoziladi. Masalan, talabalar haqidagi berilganlarni o‘z ichiga olgan massiv yaratish e’loni quyidagicha bo‘ladi:

```
const int n=25;
Talaba talabalar[n];
```

Strukturalar massivining elementlariga murojaat odatdagi massiv elementlariga murojaat usullari orqali, har bir elementning maydonlariga murojaat esa’.’ orqali amalga oshiriladi.

Quyidagi dasturda guruhdagi har bir talaba haqidagi berilganlarni klaviaturadan kiritish va guruh talabalarini familiya, ismi va sharifini chop qilinadi.

```
#include <iostream.h>
#include <conio.h>
const int n=3;
struct Talaba
{
char FISh[30];
unsigned int Tug_yil;
unsigned int Kurs;
char Yonalish[50];
float Reyting;
```

```
char Jinsi[6];
char Manzil[50];
bool status;
};
void Talaba_Kiritish(Talaba t[]);
void Talabalar_FISh(Talaba t[]);
int main(int argc, char* argv[])
{
Talaba talabalar[n];
Talaba_Kiritish(talabalar);
Talabalar_FISh(talabalar);
return 0;
}
void Talabalar_FISh(Talaba t[])
{
 for(int i=0; i<n; i++)
cout<<t[i].FISh<<endl;
}
void Talaba_Kiritish(Talaba t[])
{
 for(int i=0; i<n; i++)
 {
 cout<<i+1<< “-talaba malumotlarini kiriting:”<<endl;
 cout<< “Talaba FISh :”;
 cin.getline(t[i].FISh,30);
 cout<< “Kurs:”;
 cin>>t[i].Kurs;
 cout<< “Reyting bali:”;
 cin>>t[i].Reyting;cout<< “Tug‘ilgan yili:”;
 cin>>t[i].Tug_yil;
```

```
cout<< “Ta'lim yo‘nalishi:”;
cin.getline(t[i].Yo‘nalish,50);
cout<< “Jinsi(erkak,ayol):”;
cin.getline(t[i].Jinsi,6);
cout<< “Yashash manzili:”;
cin.getline(t[i].Manzil,50);
}
}
```

Strukturalarga ko‘rsatgich

Struktura elementlariga ko‘rsatgichlar [7] orqali murojaat qilish mumkin. Buning uchun strukturaga ko‘rsatgich o‘zgaruvchisi e’lon qilinishi kerak. Masalan, yuqorida keltirilgan misolda Talaba strukturasiga ko‘rsatgich quyidagicha e’lon qilinadi:

```
Talaba * k_talaba;
```

Ko‘rsatgich orqali aniqlangan struktura elementlariga murojaat “.” bilan emas, balki “->” vositasida amalga oshiriladi:

```
cout<<k_talaba ->FISh;
```

Strukturalarni ko‘rsatgich va adresni olish (&) vositasida funksiya argumenti sifatida uzatish mumkin. Quyida keltirilgan dastur bo‘lagida strukturani Talaba_Kiritish() funksiyasiga ko‘rsatgich orqali, Talabalar_FISh() funksiyasiga esa adresni olish vositasida uzatishga misol keltirilgan.

```
...
void Talaba_Kiritish(Talaba *t);
void Talabalar_FISh(Talaba & t);
int main( )
{
 Talaba * k_talaba;
 k_talaba=(Talaba*)malloc(n*sizeof(Talaba));
 Talaba_Kiritish(k_talaba);
 Talabalar_FISh(*k_talaba);
```

```
 return 0;
}
void Talabalar_FISh(Talaba & t)
{
 for(int i=0; i<n; i++)
 {cout<<(&t+i)->FISh<<endl;}
}
void Talaba_Kiritish(Talaba *t)
{
 for(int i=0; i<n; i++)
 {
 cout<<i+1<<"-talaba malumotlarini kiriting:"<<endl;
 cout<< "Talaba FISh :";
 cin.getline((t+i)->FISh,30);
 cout<< "Kurs:";
 cin>>(t+i)->Kurs;
 ...
 }
}
```

Shunga e'tibor berish kerakki, dinamik ravishda hosil qilingan strukturalar massivi elementi bo‘lgan strukturaning maydoniga murojaatda "*" belgisi qo‘llanilmaydi.

Masala. Futbol jamoalari haqidagi ma'lumotlar - jamoa nomi, ayni paytdagi yutuqlar, durang va mag‘lubiyatlar sonlari, hamda raqib darvozasiga kiritilgan va o‘z darvozasidan o‘tkazib yuborilgan to‘plar sonlari bilan berilgan. Futbol jamoalarining turnir jadvali chop qilinsin. Jamoalarni jadvalda tartiblashda quyidagi qoidalarga amal qilinsin:

1) jamoalar to‘plagan ochkolarini kamayishi bo‘yicha tartiblanishi kerak;

2) agar jamoalar to‘plagan ochkolari teng bo‘lsa, ulardan nisbatan ko‘p g‘alabaga erishgan jamoa jadvalda yuqori o‘rinni egallaydi;

3) agar ikkita jamoaning to‘plagan ochkolari va g‘alabalar soni teng bo‘lsa, ulardan nisbatan ko‘p to‘p kiritgan jamoa jadvalda yuqori o‘rinni egallaydi.

Jamoa haqidagi berilganlar struktura ko‘rinishida, jadval esa struktura massivi sifati aniqlanadi:

```
struct Jamoa
{
 string Nomi;
 int Yutuq, Durrang, Maglub, Urgan_top, O‘tkazgan_top;
 int O‘yin, Ochko;
};
```

Bu yerda O‘yin maydoni Yutuq, Durrang va Mag‘lub maydonlar yig‘indisi, jamoa to‘plagan ochkolar - Ochko=3*Yutuq+1*Durrang ko‘rinishida aniqlanadi. Jamoalar massivi Ochko, Yutuq va Urgan_to‘p maydonlari bo‘yicha tartiblanadi.

Dastur matni:

```
struct Jamoa
{
 string Nomi;
 int Yutuq, Durrang, Maglub, Urgan_top, Utkazgan_top;
 int O‘yin, Ochko;
};
const nom_uzunligi=10;
int jamoalar_soni;
Jamoa * Jamoalar_Jadvali()
{
 char *jm_nomi=(char*)malloc(nom_uzunligi+1);
 cout<< “Jamoalar soni:”;
 cin>>jamoalar_soni;
 Jamoa * jm=new Jamoa[jamoalar_soni];
 for(int i=0; i<jamoalar_soni; i++)
 {
 cin.ignore();
 cout<<i+1<<”-jamoa ma'lumotlari:\n”;
 cout<<” Nomi: “;
 cin.getline(jm_nomi,nom_uzunligi);
 while(strlen(jm_nomi)<nom_uzunligi)
  strcat(jm_nomi, “ ”);
 jm[i].Nomi.assign(snomi);
```

```
 cout<< “Yutuqlar soni: “;
 cin>> jm[i].Yutuq;
 cout<<”Durranglar soni: “;
 cin>>jm[i].Durang;
 cout<<”Mag‘lubiyatlar soni: “;
 cin>>jm[i].Maglub;
 cout<<” Raqib darvozasiga urilgan to‘plar soni: “;
 cin>>jm[i].Urgan_top;
 cout<<” O‘z darvozasigan o‘tkazgan to‘plar soni: “;
 cin>>jm[i].Utkazgan_top;
 jm[i].O‘yin=jm[i].Yutuq+jm[i].Durrang + jm[i].Mag‘lub;
 jm[i].Ochko=jm[i].Yutuq*3 +jm[i].Durrang;
 }
free(snomi);
return jm;
}
void O‘tkazish(Jamoa & jamoa1, const Jamoa & jamoa2)
{
 jamoa1.Nomi=jamoa2.Nomi;
 jamoa1.Yutuq=jamoa2.Yutuq;
 jamoa1.Durrang=jamoa2.Durrang;
 jamoa1.Maglub=jamoa2.Maglub;
 jamoa1.Urgan_top=jamoa2.Urgan_top;
 jamoa1.Otkazgan_top=jamoa2.Otkazgan_top;
 jamoa1.Oyin=jamoa2.Oyin;
 jamoa1.Ochko=jamoa2.Ochko;
}
Jamoa * Jadvalni_Tartiblash(Jamoa * jm)
{
 bool urin_almashdi=true;
 for(int i=0;i<jamoalar_soni-1 && urin_almashdi; i++)
 {
 Jamoa Vaqtincha;
 urin_almashdi=false;
 for(int j=i; j<jamoalar_soni-1; j++)
 {
  // j-jamoaning ochkosi (j+1)- jamoa ochkosidan katta
  // bo‘lsa, takrorlashning keyingi qadamiga o‘tilsin.
  if(jm[j].Ochko>jm[j+1].Ochko) continue;
```

```
 //j va (j+1)-jamoalarning ochkolari teng va j-jamoa
 // yutuqlari (j+1)- jamoa yutuqlaridan ko‘p bo‘lsa,
 // takrorlashning keyingi qadamiga o‘tilsin.
 if(jm[j].Ochko==jm[j+1].Ochko &&
  jm[j].Yutuq>jm[j+1].Yutuq) continue;
 //j va (j+1)-jamoalarning ochkolari va yutuqlar soni
 // teng va j-jamoa urgan to‘plar soni (j+1)- jamoa
 //urgan to‘plardan ko‘p bo‘lsa, takrorlashning keyingi
 // qadamiga o‘tilsin.
 if(jm[j].Ochko==jm[j+1].Ochko &&
  jm[j].Yutuq==jm[j+1].Yutuq &&
  jm[j].Urgan_tup>jm[j+1].Urgan_tup) continue;
 //yuqoridagi shartlarning birortasi ham bajarilmasa,
 //j va (j+1)-jamoalar o‘rinlari almashtirilsin.
 urin_almashdi=true;
 O‘tkazish(Vaqtincha,jm[j]);
 O‘tkazish(jm[j],jm[j+1]);
 O‘tkazish(jm[j+1], Vaqtincha);
 }
 }
 return jm;
}
  void Jadavlni_Chop_Qilish(const Jamoa *jm)
  {
char pr=’’;
cout<<«  FUTBOL JAMOALARINING TURNIR JADVALI\n» ;
cout<<«----------------------------------------\n»;
cout<<«| JAMOA  | O | Y | D | M |UrT|O‘T|OCHKO|\n»;
cout<<«----------------------------------------\n»;
 for(int i=0; i<jamoalar_soni; i++)
 {
 cout<<«|»<<jm[i].Nomi.substr(0,10);cout<<’|’;
 if(jm[i].O‘yin<10)cout<<pr;cout<<jm[i].Uyin<<” |”;
 if(jm[i].Yutuq<10)cout<<pr;cout<<jm[i].Yutuq<<” |”;
 if(jm[i].Durrang<10)cout<<pr;
 cout<<jm[i].Durrang<<” |”;;
 if(jm[i].Maglub<10)cout<<pr;
 cout<<jm[i].Maglub<<” |”;
 if(jm[i].Urgan_top<10)cout<<pr;
 cout<<jm[i].Urgan_top<<” |”;
 if(jm[i].Otkazgan_top<10)cout<<pr;
```

```
 cout<<jm[i].Otkazgan_top<<” |”;;
 if(jm[i].Ochko<10)cout<<pr;
 cout<<jm[i].Ochko<<” | “<<endl;
 }
cout<<”----------------------------------------\n”;
}
int main()
{
 Jamoa *jamoa;
 jamoa=Berilganlarni_kiritish();
 jamoa=Jadvalni_Tartiblash(jamoa);
 Jadvalni_Chop_Qilish(jamoa);
 return 0;
}
```

Dastur bosh funksiya va quyidagi vazifalarni bajaruvchi to‘rtta funksiyadan tashkil topgan:

1) Jamoa * Jamoalar_Jadvali()- jamoalar haqidagi berilganlarni saqlaydigan Jamoa strukturalaridan tashkil topgan dinamik massiv yaratadi va unga oqimdan har bir jamoa berilganlarni o‘qib joylashtiradi. Hosil bo‘lgan massivga ko‘rsatgichni funksiya natijasi sifatida qaytaradi;

2) Jamoa*Jadvalni_Tartiblash(Jamoa*jm) - argument orqali ko‘rsatilgan massivni masala sharti bo‘yicha tartiblaydi va shu massivga ko‘rsatgichni qaytaradi;

3) void Utkazish(Jamoa & jamoa1, Jamoa & jamoa2) - jamoa2 strukturasidagi maydonlarni jamoa1 strukturasiga o‘tkazadi. Bu funksiya Jadvalni_Tartiblash() funksiyasidan massivdagi ikkita strukturani o‘zaro o‘rinlarini almashtirish uchun chaqiriladi;

4) void Jadavlni_Chop_Qilish(const Jamoa *jm) - argumentda berilgan massivni turnir jadvali qolipida chop qiladi.

Uchta jamoa haqida ma’lumot berilganda dastur ishlashining natijasi quyidagicha bo‘lishi mumkin:

FUTBOL JAMOALARINING TURNIR JADVALI

| JAMOA | O | Y | D | M |UrT|O‘T|OCHKO|

|Bunyodkor |20 |15 | 3 | 2 |30 |10 | 48 |

|Paxtakor |20 |11 | 5 | 4 |20 |16 | 38 |

|Neftchi |20 | 8 | 5 | 7 |22 |20 | 29 |

Muhokama savollari

1. Tuzilmalarni tashkil etish va qo‘llash?
2. Birlashmalar haqida tushuncha?
3. Massivlarning tuzilma elementlari sifatida qo‘llanishi?

Nazorat savollari

1. Tuzilmani ta’riflash.
2. Tuzilmalar bilan ishlash asoslari
3. Tuzilma elementlariga murojaat.
4. Massivlarning tuzilma elementi sifatida qo‘llanishi.
5. Tuzilmada foydalanuvchi toifasini tashkil etish.

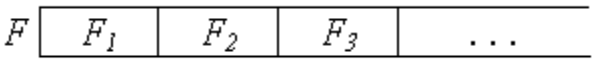
3.6 Fayllar haqida tushuncha

C++ tilidagi standart va foydalanuvchi tomonidan aniqlangan turlarning muhim xususiyati shundan iboratki, ularning oldindan aniqlangan miqdordagi chekli elementlardan iboratligidir. Hatto berilganlar dinamik aniqlanganda ham, operativ xotiraning (uyumning) amalda cheklanganligi sababli, bu berilganlar miqdori yuqoridan chegaralangan elementlardan iborat bo‘ladi. Ayrim bir tadbiqiy masalalar uchun oldindan berilganning komponentalari sonini aniqlash imkoni yo‘q. Ular masalani yechish jarayonida aniqlanadi va yetarlicha katta hajmda bo‘lishi mumkin. Ikkinchi tomondan, dasturda e’lon qilingan o‘zgaruvchilarning

qiymatlari sifatida aniqlangan berilganlar faqat dastur ishlash paytidagina mavjud bo‘ladi va dastur o‘z ishini tugatgandan keyin yo‘qolib ketadi. Agar dastur yangidan ishga tushirilsa, bu berilganlarni yangidan hosil qilish zarur bo‘ladi. Aksariyat tadbiqiy masalalar esa berilganlarni doimiy ravishda saqlab turishni talab qiladi. Masalan, korxona xodimlarining oylik maoshini hisoblovchi dasturda xodimlar ro‘yxatini, shtat stavkalari va xodimlar tomonidan olingan maoshlar haqidagi ma’lumotlarni doimiy ravishda saqlab turish zarur. Bu talablarga fayl turidagi Ob’yektlar (o‘zgaruvchilar) javob beradi.

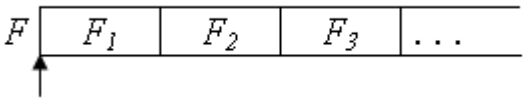
Fayl - bu bir xil turdagi qiymatlar joylashgan tashqi xotiradagi nomlangan sohadir.

Faylni, boshida ketma-ket ravishda joylashgan yozuvlar (masalan, musiqa) bilan to‘ldirilgan va oxiri bo‘sh bo‘lgan yetarlicha uzun magnit tasmasiga o‘xshatish mumkin.



3.1-rasm. Fayl tasviri

3.1-rasmda F-fayl nomi, F_1,F_2, F_3 - fayl elementlari (komponentalari). Xuddi yangi musiqani tasma oxiriga qo‘shish mumkin bo‘lgandek, yangi yozuvlar fayl oxiriga qo‘shilishi mumkin.



3.2-rasm. Fayl ko‘rsatgichi

Yana bir muhim tushunchalardan biri fayl ko‘rsatgichi tushunchasidir.*Fayl ko‘rsatgichi* - ayni paytda fayldan o‘qilayotgan yoki unga yozilayotgan joy (yozuv o‘rnini) ko‘rsatib turadi, ya’ni fayl ko‘rsatgichi ko‘rsatib turgan joydan bitta yozuvni o‘qish yoki shu joyga yangi yozuvni joylashtirish mumkin. 3.2-rasmda fayl ko‘rsatgichi fayl boshini ko‘rsatmoqda.

Fayl yozuvlariga murojaat ketma-ket ravishda amalga oshiriladi: n- yozuvga murojaat qilish uchun n-1 yozuvni o‘qish zarur bo‘ladi. Shuni ta’kidlab o‘tish zarurki, fayldan yozuvlarni o‘qish jarayoni qisman «avtomatlashgan», unda i-yozuvni o‘qilgandan keyin, ko‘rsatgich navbatdagi i+1 yozuv boshiga ko‘rsatib turadi va shu tarzda o‘qishni davom ettirish mumkin (massivlardagidek indeksni oshirish shart emas). Fayl- bu berilganlarni saqlash joyidir va shu sababli uning yozuvlari ustida to‘g‘ridan-to‘g‘ri amal bajarib bo‘lmaydi. Fayl yozuvi ustida amal bajarish uchun yozuv qiymati operativ xotiraga mos turdagi o‘zgaruvchiga o‘qilishi kerak. Keyinchalik, zarur amallar shu o‘zgaruvchi ustida bajariladi va kerak bo‘lsa natijalar yana faylga yozilishi mumkin.

Operasion tizim nuqtai-nazaridan fayl hisoblangan har qanday fayl C++ tili uchun *moddiy fayl* hisoblanadi. MS DOS uchun moddiy fayllar <fayl nomi>.<fayl kengaytmasi> ko‘rinishidagi «8.3» formatidagi satr (nom) orqali beriladi. Fayl nomlari satr o‘zgarmaslar yoki satr o‘zgaruvchilarida berilishi mumkin. MS DOS qoidalariga ko‘ra fayl nomi to‘liq bo‘lishi, ya’ni fayl nomining boshida adres qismi bo‘lishi mumkin: ”C:\\USER\\Misol.spp”, “A:\\matn.txt”.

C++ tilida *mantiqiy fayl* tushunchasi bo‘lib, u fayl turidagi o‘zgaruvchini anglatadi. Fayl turidagi o‘zgaruvchilarga boshqa turdagi o‘zgaruvchilar kabi qiymat berish operatori orqali qiymat berib bo‘lmaydi. Boshqacha aytganda fayl turidagi o‘zgaruvchilar ustida hech qanday amal aniqlanmagan. Ular ustida bajariladigan barcha amallar funksiyalar vositasida bajariladi.

Fayllar bilan ishlash[8] quyidagi bosqichlarni o‘z ichiga oladi:

- fayl o‘zgaruvchisi albatta diskdagi fayl bilan bog‘lanadi;
- fayl ochiladi;
- fayl ustida yozish yoki o‘qish amallari bajariladi;
- fayl yopiladi;
- fayl nomini o‘zgartirish yoki faylni diskdan o‘chirish amallarini bajarilishi mumkin.

Matnli va binar fayllar

C++ tili C tilidan o‘qish-yozish amalini bajaruvchi standart funksiyalar kutubxonasini vorislik bo‘yicha olgan. Bu funksiyalar <stdio.h> sarlavha faylida e’lon qilingan. O‘qish-yozish amallari fayllar bilan bajariladi. Fayl matn yoki binar (ikkilik) bo‘lishi mumkin.

Matnli fayl - ASCII kodidagi belgilar bilan berilganlar majmuasi. Belgilar ketma-ketligi satrlarga bo‘lingan bo‘ladi va satrning tugash alomati sifatida CR(karetkani qaytarish yoki’\r’)LF (satrni o‘tkazish yoki’\n’) belgilar juftligi hisoblanadi. Matn fayldan berilganlarni o‘qishda bu belgilar juftligi bitta CR belgisi bilan almashtiriladi va aksincha, yozishda CR belgisi ikkita CR va LF belgilariga almashtiriladi. Fayl oxiri #26 (^Z) belgisi bilan belgilanadi.

Matnli faylga boshqacha ta’rif berish ham mumkin [10]. Agar faylni matn tahririda ekranga chiqarish va o‘qish mumkin bo‘lsa, bu matn fayl. Klaviatura ham kompyuterga faqat matnlarni jo‘natadi. Boshqacha aytganda dastur tomonidan ekranga chiqariladigan barcha ma’lumotlarni stdout nomidagi matn fayliga chiqarilmoqda deb qarash mumkin. Xuddi shunday klaviaturadan o‘qilayotgan har qanday berilganlarni matn faylidan o‘qilmoqda deb hisoblanadi.

Matnli fayllarining komponentalari *satrlar* deb nomlanadi. Satrlar uzluksiz joylashib, turli uzunlikda va bo‘sh bo‘lishi mumkin. Faraz qilaylik, T matn fayli 4 satrdan iborat bo‘lsin:

1-satr#13#10	2-satr uzunroq #13#10	#13#10	4-satr#13#10#26

3.3-rasm. To‘rtta satrdan tashkil topgan matn fayli

Matnni ekranga chiqarishda satr oxiridagi #13#10 boshqaruv belgilari juftligi kursorni keyingi satrga tushiradi va uni satr boshiga olib keladi. Bu matn fayl ekranga chop etilsa, uning ko‘rinishi quyidagicha bo‘ladi:

1-satr[13][10]

2-satr uzunroq[13][10]

[13][10]

4- satr[13][10]

Matndagi [n] - n-kodli boshqaruv belgisini bildiradi. Odatda matn tahrirlari bu belgilarni koʻrsatmaydi.

Binar fayllar- bu oddiygina baytlar ketma-ketligi. Odatda binar fayllardan berilganlarni foydalanuvchi tomonidan bevosita «koʻrish» zarur boʻlmagan hollarda ishlatiladi. Binar fayllardan oʻqish-yozishda baytlar ustida hech qanday konvertasiya amallari bajarilmaydi.

Oʻqish-yozish oqimlari. Standart oqimlar

Oqim tushunchasi berilganlarni faylga oʻqish-yozishda ularni belgilar ketma-ketligi yoki oqimi koʻrinishida tasavvur qilishdan kelib chiqqan. Oqim ustida quyidagi amallarni bajarish mumkin:

- oqimdan berilganlar blokini operativ xotiraga oʻqish;
- operativ xotiradagi berilganlar blokini oqimga chiqarish;
- oqimdagi berilganlar blokini yangilash;
- oqimdan yozuvni oʻqish;
- oqimga yozuvni chiqarish.

Oqim bilan ishlaydigan barcha funksiyalar buferli, formatlashgan yoki formatlashmagan oʻqish-yozishni taʼminlaydi.

Dastur ishga tushganda oʻqish-yozishning quyidagi standart oqimlar ochiladi:

stdin - oʻqishning standart vositasi;

stdout - yozishning standart vositasi;

stderr- xatolik haqida xabar berishning standart vositasi;

stdprn - qogʻozga chop qilishning standart vositasi;

stdaux - standart yordamchi qurilma.

Kelishuv boʻyicha stdin - foydalanuvchi klaviaturasi, stdout va stderr-terminal (ekran), stdprn- printer bilan, hamda stdaux - kompyuter yordamchi portlariga bogʻlangan hisoblanadi. Berilganlarni oʻqish-yozishda stderr va stdaux oqimidan boshqa oqimlar buferlanadi, yaʼni belgilar ketma-ketligi operativ xotiraning bufer deb nomlanuvchi sohasida vaqtincha jamlanadi. Masalan,

belgilarni tashqi qurilmaga chiqarishda belgilar ketma-ketligi buferda jamlanadi va bufer to‘lgandan keyingina tashqi qurilmaga chiqariladi.

Hozirdagi operatsion tizimlarda klaviatura va displeylar matn fayllari sifatida qaralad [6]. Haqiqatdan ham berilganlarni klaviaturadan dasturga kiritish (o‘qish) mumkin, ekranga esa chiqarish (yozish) mumkin. Dastur ishga tushganda standart o‘qish va yozish oqimlari o‘rniga matn fayllarni tayinlash orqali bu oqimlarni qayta aniqlash mumkin. Bu holatni *o‘qishni (yozishni) qayta adreslash ro‘y berdi* deyiladi. O‘qish uchun qayta adreslashda’<’ belgisidan, yozish uchun esa’>’ belgisidan foydalaniladi. Misol uchun gauss.exe bajariluvchi dastur berilganlarni o‘qishni klaviaturadan emas, balki massiv.txt faylidan amalga oshirish zarur bo‘lsa, u buyruq satrida quyidagi ko‘rinishda yuklanishi zarur bo‘ladi:

gauss.exe < massiv.txt

Agar dastur natijasini natija.txt fayliga chiqarish zarur bo‘lsa

gauss.exe > natija.txt

satri yoziladi.

Va nihoyat, agar berilganlarni massiv.txt faylidan o‘qish va natijani natija.txt fayliga yozish uchun

gauss.exe < massiv.txt > natija.txt

buyruq satri teriladi.

Umuman olganda, bir dasturning chiqish oqimini ikkinchi dasturning kirish oqimi bilan bog‘lash mumkin. Buni *konveyrli jo‘natish* deyiladi. Agar ikkita junat.exe dastursi qabul.exe dastursiga berilganlarni jo‘natishi kerak bo‘lsa, u holda ular o‘rtasiga’|’ belgi qo‘yib yoziladi:

junat.exe | qabul.exe

Bu ko‘rinishdagi dasturlar o‘rtasidagi konveyrli jo‘natishni operatsion tizimning o‘zi ta’minlaydi.

Simvollarni o‘qish-yozish funksiyalari

Belgilarni o‘qish-yozish funksiyalari makros ko‘rinishida amalga oshirilgan.

getc() makrosi tayinlangan oqimdan navbatdagi belgini qaytaradi va kirish oqimi ko‘rsatgichini keyingi belgini o‘qishga moslagan holda oshiradi. Agar o‘qish

muvaffaqiyatli bo‘lsa getc() funksiyasi ishorasiz int ko‘rinishidagi qiymatni, aks holda EOF qaytaradi. Ushbu funksiya prototipi quyidagicha:

int getc(FILE * stream)

EOF identifikator makrosi

#define EOF(-1)

ko‘rinishida aniqlangan va o‘qish-yozish amallarida fayl oxirini belgilash uchun xizmat qiladi. EOF qiymati ishorali char turida deb hisoblanadi. Shu sababli o‘qish-yozish jarayonida unsigned char turidagi belgilar ishlatilsa, EOF makrosini ishlatib bo‘lmaydi.

Navbatdagi misol getc() makrosini ishlatishni namoyon qiladi.

```
#include <iostream.h>
#include <stdio.h>
int main()
{
char ch;
cout<<”Belgini kiriting: “;
ch=getc(stdin);
cout<<”Siz ”<<ch<<” belgisini kiritdingiz.\n”;
return 0;
}
```

getc() makrosi aksariyat holatlarda stdin oqimi bilan ishlatilganligi sababli, uning getc(stdin) ko‘rinishiga ekvivalent bo‘lgan int getchar() makrosi aniqlangan. Yuqoridagi misolda «ch=getc(stdin);» satrini «ch=getchar();» satri bilan almashtirish mumkin.

Belgini oqimga chiqarish uchun putc() makrosi aniqlangan va uning prototipi

int putc(int c, FILE*stream)

ko‘rinishida aniqlangan. putc() funksiyasi stream nomi bilan berilgan oqimga s belgini chiqaradi. Funksiya qaytaruvchi qiymati sifatida int turiga aylantirilgan s belgi bo‘ladi. Agar belgini chiqarishda xatolik ro‘y bersa EOF qaytariladi.

putc() funksiyasini standart stdout oqimi bilan bog‘langan holati - putc(c,strout) uchun putchar(c) makrosi aniqlangan.

Satrlarni o‘qish - yozish funksiyalari

Oqimdan satrni o‘qishga yo‘naltirilgan gets() funksiyasining prototipi

char*gets(char *s);

ko‘rinishida aniqlangan.gets() funksiyasi standart oqimdan satrni o‘qiydi va uni s o‘zgaruvchisiga joylashtiradi. Joylashtirish paytida oqimdagi’\n’ belgisi’\0’ belgisi bilan almashtiriladi. Bu funksiyani ishlatishda o‘qilayotgan satrning uzunligi s satr uchun ajratilgan joy uzunligidan oshib ketmasligini nazorat qilish kerak bo‘ladi.

puts() funksiyasi

int puts(const char *s);

ko‘rinishida bo‘lib, u standrat oqimga argumentda ko‘rsatilgan satrni chiqaradi. Bunda satr oxiriga yangi satrga o‘tish belgisi’\n’ qo‘shiladi. Agar satrni oqimga chiqarish muvaffaqiyatli bo‘lsa puts() funksiyasi manfiy bo‘lmagan sonni, aks holda EOF qaytaradi.

Satrni o‘qish-yozish funksiyalarini ishlatishga misol tariqasida qo‘yidagi dasturni keltirish mumkin.

```
#include <stdio.h>
int main()
{ char*s;
puts(”Satrni kiriting: “);
gets(s);
puts(”Kiritilgan satr: ”);
 puts(s);
 return 0;
}
```

Formatli o‘qish va yozish funksiyalari

Formatli o‘qish va yozish funksiyalari -scanf() va printf() funksiyalari C tilidan vorislik bilan olingan[8]. Bu funksiyalarni ishlatish uchun «stdio.h» sarlavha faylini dasturga qo‘shish kerak bo‘ladi.

Formatli o‘qish funksiyasi scanf() quyidagi prototipga ega:

int scanf(const char * <format>[<adres>,...])

Bu funksiya standart oqimdan berilganlarni formatli o‘qishni amalga oshiradi. Funksiya, kirish oqimidagi maydonlar ketma-ketligi ko‘rinishidagi belgilarni birma-bir o‘qiydi va har bir maydonni <format> satrida keltirilgan format aniqlashtiruvchisiga mos ravishda formatlaydi. Oqimdagi har bir maydonga format aniqlashtiruvchisi va natija joylashadigan o‘zgaruvchining adresi bo‘lishi shart. Boshqacha aytganda, oqimdagi maydon (ajratilgan belgilar ketma-ketligi) ko‘rsatilgan formatdagi qiymatga akslantiriladi va o‘zgaruvchi bilan nomlangan xotira bo‘lagiga joylashtiriladi (saqlanadi). Funksiya oqimdan berilganlarni o‘qish jarayonini «to‘ldiruvchi belgini» uchratganda yoki oqim tugashi natijasida to‘xtatishi mumkin. Oqimdan berilganlarni o‘qish muvafaqqiyatli bo‘lsa, funksiya muvafaqqiyatli aylantirilgan va xotiraga saqlangan maydonlar sonini qaytaradi. Agar hech bir maydonni saqlash imkoni bo‘lmagan bo‘lsa, funksiya 0 qiymatini qaytaradi. Oqim oxiriga kelib qolganda (fayl yoki satr oxiriga) o‘qishga harakat bo‘lsa, funksiya EOF qiymatini qaytaradi.

Formatlash satri -<format> belgilar satri bo‘lib, u uchta toifaga bo‘linadi:

– to‘ldiruvchi belgilar;
– to‘ldiruvchi belgilardan farqli belgilar;
– format aniqlashtiruvchilari.

To‘ldiruvchi-belgilar – bu probel,’\t’,’\n’ belgilari.Bu belgilar formatlash satridan o‘qiladi, lekin saqlanmaydi.

To‘ldiruvchi belgilardan farqli belgilar – bu qolgan barcha ASCII belgilari,’%’ belgisilan tashqari. Bu belgilar formatlash satridan o‘qiladi, lekin saqlanmaydi.

3.1–jadval. Format aniqlashtiruvchilari va ularning vazifasi

Komponenta	Bo‘lishi shart yoki yo‘q	Vazifasi
[*]	Yo‘q	Navbatdagi ko‘rib chiqilayotgan maydon qiymatini o‘zgaruvchiga o‘zlashtirmaslik belgisi. Kirish oqimidagi maydon ko‘rib chiqiladi, lekin o‘zgaruvchida saqlanmaydi.
[<kenglik>]	Yo‘q	Maydon kengligini aniqlashtiruvchisi. O‘qiladigan belgilarning maksimal sonini aniqlaydi. Agar oqimda to‘ldiruvchi belgi yoki almashtirilmaydigan belgi uchrasa funksiya nisbatan kam sondagi belgilarni o‘qishi mumkin.
[F|N]	Yo‘q	O‘zgaruvchi ko‘rsatgichining (adresining) modifikatori: F– far pointer; N- near pointer
[h|l|L]	Yo‘q	Argument turining modifikatori. <tur belgisi> bilan aniqlangan o‘zgaruvchining qisqa (short - h) yoki uzun (long –l,L) ko‘rinishini aniqlaydi.
<tur belgisi>	Ha	Oqimdagi belgilarni almashtiriladigan tur belgisi

Format aniqlashtiruvchilari– oqim maydonidagi belgilarni ko‘rib chiqish, o‘qish va adresi bilan berilgan o‘zgaruvchilar turiga mos ravishda almashtirish jarayonini boshqaradi. Har bir format aniqlashtiruvchisiga bitta o‘zgaruvchi adresi mos kelishi kerak. Agar format aniqlashtiruvchilari soni o‘zgaruvchilardan ko‘p bo‘lsa, natija nima bo‘lishini oldindan aytib bo‘lmaydi. Aks holda, ya’ni o‘zgaruvchilar soni ko‘p bo‘lsa, ortiqcha o‘zgaruvchilar inobatga olinmaydi.

Format aniqlashtiruvchisi quyidagi ko‘rinishga ega:

% [*][<kenglik>] [F|N] [h|l|L] <tur belgisi>

Format aniqlashtiruvchisi’%’belgisidan boshlanadi va undan keyin 3.1–jadvalda keltirilgan shart yoki shart bo‘lmagan komponentalar keladi.

3.2–jadval. Almashtiriladigan tur alomati belgilari

Tur alomati	Kutilayotgan qiymat	Argument turi
Son turidagi argument		
d, D	O‘nlik butun	int * arg yoki long * arg
E,e	Qo‘zg‘aluvchi nuqtali	float * arg

	son	
F	Qo‘zg‘aluvchi nuqtali son	float * arg
G,g	Qo‘zg‘aluvchi nuqtali son	float * arg
O	Sakkizlik son	int * arg
O	Sakkizlik son	long * arg
I	O‘nlik, sakkizlik va o‘n oltilik butun son	int * arg
I	O‘nlik, sakkizlik va o‘n oltilik butun son	long * arg
U	Ishorasiz o‘nlik son	Unsigned int * arg
U	Ishorasiz o‘nlik son	Unsigned long * arg
X	O‘n oltilik son	int * arg
X	O‘n oltilik son	int * arg
Belgilar		
S	Satr	char * arg (belgilar massivi)
C	Belgi	char * arg (belgi uchun maydon kengligi berilishi mumkin (masalan, %4s). N belgidan tashkil topgan belgilar massiviga ko‘rsatgich: char arg[N])
%	‘%’ belgisi	Hech qanday almashtirishlar bajarilmaydi,’%’ belgisi saqlanadi.
Ko‘rsatgichlar		
N	int * arg	%n argumentigacha muvaffaqiyatli o‘qilgan belgilar soni, aynan shu int ko‘rsatgichi bo‘yicha adresda saqlanadi.
P	YYYY:ZZZZ yoki ZZZZ ko‘rinishidagi o‘n oltilik	Ob’yektga ko‘rsatgich (far* yoki near*).

Oqimdagi belgilar bo‘lagini almashtiriladigan tur alomatining qabul qilishi mumkin bo‘lgan belgilar 3.2-jadvalda keltirilgan.

3.3–jadval. Format aniqlashtiruvchilari va ularning vazifasi

Komponenta	Bo‘lishi shart yoki yo‘q	Vazifasi
[bayroq]	Yo‘q	Bayroq belgilari. Chiqarilayotgan qiymatni chapga yoki o‘ngga tekislashni, sonning ishorasini, o‘nlik kasr nuqtasini, oxirdagi nollarni, sakkizlik va o‘n oltilik sonlarning alomatlarni chop etishni boshqaradi. Masalan,’-‘ bayrog‘i qiymatni ajratilgan

		o‘ringa nisbatan chapdan boshlab chiqarishni va kerak bo‘lsa o‘ngdan probel bilan to‘ldirishni bildiradi, aks holda chap tomondan probellar bilan to‘ldiradi va davomiga qiymat chiqariladi.
[<kenglik>]	Yo‘q	Maydon kengligini aniqlashtiruvchisi. Chiqariladigan belgilarning minimal sonini aniqlaydi. Zarur bo‘lsa qiymat yozilishidan ortgan joylar probel bilan to‘ldiriladi.
[.<xona>]	Yo‘q	Aniqlik. Chiqariladigan belgilarning maksimal sonini ko‘rsatadi. Sondagi raqamlarning minimal sonini.
[F|N|h|l|L]	Yo‘q	O‘lcham modifikatori. Argumentning qisqa (short - h) yoki uzun (long –l,L) ko‘rinishini, adres turini aniqlaydi.
<tur belgisi>	Ha	Argument qiymati almashtiriladigan tur alomati belgisi

Formatli yozish funksiyasi printf() quyidagi prototipga ega:

intprintf(const char * <format>[,<argument>,...])

Bu funksiya standart oqimga formatlashgan chiqarishni amalga oshiradi. Funksiya argumentlar ketma-ketligidagi har bir argument qiymatini qabul qiladi va unga <format> satridagi mos format aniqlashtiruvchisini qo‘llaydi va oqimga chiqaradi.

3.4–jadval. printf() funksiyasining almashtiriladigan tur belgilari

Tur alomati	Kutilayotgan qiymat	Chiqish formati
Son qiymatlari		
D	Butun son	Ishorali o‘nlik butun son
I	Butun son	Ishorali o‘nlik butun son
O	Butun son	Ishorasiz sakkizlik butun son
U	Butun son	Ishorasiz o‘nlik butun son
X	Butun son	Ishorasiz o‘n oltilik butun son (a,b,c,d,e,f belgilari ishlatiladi)
X	Butun son	Ishorasiz o‘n oltilik butun son (A,B,C,D,E,F belgilari ishlatiladi)
F	Qo‘zg‘aluvchi nuqtali son	[-]dddd.dddd ko‘rinishidagi qo‘zg‘aluvchi nuqtali son
E	Qo‘zg‘aluvchi nuqtali son	[-]d.dddd yoki e[+/-]ddd ko‘rinishidagi qo‘zg‘aluvchi nuqtali son
G	Qo‘zg‘aluvchi nuqtali	Ko‘rsatilgan aniqlikka mos e yoki f

	son	shaklidagi qo‘zg‘aluvchi nuqtali son
E, G	Qo‘zg‘aluvchi nuqtali son	Ko‘rsatilgan aniqlikka mos e yoki f shaklidagi qo‘zg‘aluvchi nuqtali son. e format uchun’E’ chop etiladi.
Belgilar		
S	Satrga ko‘rsatgich	0-belgisi uchramaguncha yoki ko‘rsa-tilgan aniqlikka erishilmaguncha belgilar oqimga chiqariladi.
C	Belgi	Bitta belgi chiqariladi
%	Hech nima	‘%’ belgisi oqimga chiqariladi.
Ko‘rsatgichlar		
N	int ko‘rsatgich(int* arg)	%n argumentigacha muvaffaqiyatli chiqarilgan belgilar soni, aynan shu int ko‘rsatgichi bo‘yicha adresda saqlanadi.
P	Ko‘rsatgich	Argumentni YYYY:ZZZZ yoki ZZZZ ko‘rinishidagi o‘n oltilik songa aylantirib oqimga chiqaradi.

Har bir format aniqlashtiruvchisiga bitta o‘zgaruvchi adresi mos kelishi kerak. Agar format aniqlashtiruvchilari soni o‘zgaruvchilardan ko‘p bo‘lsa, natijada nima bo‘lishini oldindan aytib bo‘lmaydi. Aks holda, ya’ni o‘zgaruvchilar soni ko‘p bo‘lsa, ortiqcha o‘zgaruvchilar inobatga olinmaydi. Agar oqimga chiqarish muvaffaqiyatli bo‘lsa, funksiya chiqarilgan baytlar sonini qaytaradi, aks holda EOF.

printf() funksiyasining <format> satri argumentlarni almashtirish, formatlash va berilganlarni oqimga chiqarish jarayonini boshqaradi va u ikki turdagi Ob’yektlardan tashkil topadi:

– oqimga o‘zgarishsiz chiqariladigan oddiy belgilar;

– argumentlar ro‘yxatidagi tanlanadigan argumentga qo‘llaniladigan format aniqlashtiruvchilari.

Format aniqlashtiruvchisi quyidagi ko‘rinishga ega:

% [<bayroq>][<.kenglik>] [.<xona>][F|N|h|l|L] <tur belgisi>

Format aniqlashtiruvchisi ’%’ belgisidan boshlanadi va undan keyin 3.3–jadvalda keltirilgan shart yoki shart bo‘lmagan komponentalar keladi.

Almashtiriladigan tur belgisining qabul qilishi mumkin boʻlgan belgilar 3.4-jadvalda keltirilgan.

Berilgan qiymatlarini oqimdan oʻqish va oqimga chiqarishda scanf() va prinf() funksiyalaridan foydalanishga misol:

```
#include <stdio.h>
int main()
{
 int bson, natija;
 float hson;
 char blg, satr[81];
 printf(“\nButun va qo‘zg‘aluvchi nuqtali sonlarni,”);
 printf(“\nbelgi hamda satrni kiriting\n”);
 natija=scanf(“%d %f %c %s”, &bson, &hson,&blg,satr);
 printf(“\nOqimdan %d ta qiymat o‘qildi”,natija);
 printf(“va ular quyidagilar:”);
 printf(“\n %d %f %c %s \n”,bson, hson, blg, satr);
 return 0;
}
```

Dastur foydalanuvchidan butun va qoʻzgʻaluvchi nuqtali sonlarni, belgi va satrni kiritishni soʻraydi. Bunga javoban foydalanuvchi tomonidan

10 12.35 A Satr

qiymatlari kiritilsa, ekranga

Oqimdan 4 ta qiymat oʻqildi va ular quyidagilar:

10 12.35 ASatr

satrlari chop etiladi.

Fayldan oʻqish-yozish funksiyalari

Fayl oqimi bilan oʻqish-yozish amalini bajarish uchun fayl oqimini ochish zarur. Bu ishni, prototipi

FILE * fopen(const char* filename, const char *mode);

koʻrinishida aniqlangan fopen() funksiyasi orqali amalga oshiriladi. Funksiya filename nomi bilan faylni ochadi, u bilan oqimni bogʻlaydi va oqimni identifikatsiya qiluvchi koʻrsatgichni javob tariqasida qaytaradi. Faylni ochish muvaffaqiyatsiz boʻlganligini fopen() funksiyasining NULL qiymatli javobi bildiradi.

Parametrlar roʻyxatidagi ikkinchi - modeparametri faylni ochish rejimini aniqlaydi. U qabul qilishi mumkin boʻlgan qiymatlar 3.5- jadvalda keltirilgan.

3.5-jadval. Fayl ochish holatlari

Mode qiymati	Fayl ochilish holati tavsifi
R	Fayl faqat oʻqish uchun ochiladi
W	Fayl yozish uchun ochiladi. Agar bunday fayl mavjud boʻlsa, u qaytadan yoziladi (yangilanadi).
A	Faylga yozuvni qoʻshish rejimi. Agar fayl mavjud boʻlsa, fayl uning oxiriga yozuvni yozish uchun ochiladi, aks holda yangi fayl yaratiladi va yozish rejimida ochiladi.
r+	Mavjud fayl oʻzgartirish (oʻqish va yozish)uchun ochiladi.
w+	Yangi fayl yaratilib, oʻzgartirish (oʻqish va yozish) uchun ochiladi. Agar fayl mavjud boʻlsa, undagi oldingi yozuvlar oʻchiriladi va u qayta yozishga tayyorlanadi.
a+	Faylga yozuvni qoʻshish rejimi. Agar fayl mavjud boʻlsa, uning oxiriga (EOF alomatidan keyin) yozuvni yozish (oʻqish) uchun ochiladi, aks holda yangi fayl yaratiladi va yozish rejimida ochiladi.

Matn fayli ochilayotganligini bildirish uchun fayl ochilish rejimi satriga’t’ belgisini qoʻshib yozish zarur boʻladi. Masalan, matn fayl oʻzgartirish (oʻqish va yozish) uchun ochilayotganligini bildirish uchun “rt+” satri yozish kerak boʻladi. Xuddi shunday binar fayllar ustida ishlash uchun’b’ belgisini ishlatish kerak. Misol uchun fayl ochilishining“wb+” rejimi binar fayl yangilanishini bildiradi.

Fayl o‘zgartirish (o‘qish-yozish) uchun ochilganda, berilganlarni oqimdan o‘qish, hamda oqimga yozish mumkin. Biroq yozish amalidan keyin darhol o‘qib bo‘lmaydi, buning uchun o‘qish amalidan oldin fseek() yoki rewind() funksiyalari chaqirilishi shart.

Faraz qilaylik «C:\\USER\\TALABA\\iat1kuz.txt» nomli matn faylni o‘qish uchun ochish zarur bo‘lsin. Bu talab

FILE *f=fopen(”C:\\USER\\TALABA\\iat1kuz.txt”,”r+”);

ifodasini yozish orqali amalga oshiraladi. Natijada diskda mavjud bo‘lgan fayl dasturda f o‘zgaruvchisi nomi bilan aynan bir narsa deb tushuniladi. Boshqacha aytganda, dasturda keyinchalik f ustida bajarilgan barcha amallar, diskdagi «iat1kuz.txt» fayli ustida ro‘y beradi.

Fayl oqimi bilan ishlash tugagandan keyin u albatta yopilishi kerak. Buning uchun fclose() funksiyasidan foydalaniladi. Funksiya prototipi quyidagi ko‘rinishga ega:

int fclose (FILE *stream);

fclose() funksiyasi oqim bilan bog‘liq buferlarni tozalaydi (masalan, faylga yozish ko‘rsatmalari berilishi natijasida buferda yig‘ilgan berilganlarni diskdagi faylga ko‘chiradi) va faylni yopadi. Agar faylni yopish xatolikka olib kelsa, funksiya EOF qiymatini, normal holatda 0 qiymatini qaytaradi.

fgetc() funksiyasi prototipi

int fgetc(FILE *stream);

ko‘rinishida aniqlangan bo‘lib, fayl oqimidan belgini o‘qishni amalga oshiradi. Agar o‘qish muvaffaqiyatli bo‘lsa, funksiya o‘qilgan belgini int turidagi ishorasiz butun songa aylantiradi. Agar fayl oxirini o‘qishga harakat qilinsa yoki xatolik ro‘y bersa, funksiya EOF qiymatini qaytaradi.

Ko‘rinib turibdiki, getc() va fgetc() funksiyalari deyarli bir xil ishni bajaradi, farqi shundaki, getc() funksiyasi belgini standart oqimdan o‘qiydi. Boshqacha aytganda, getc() funksiyasi, fayl oqimi standart qurilma bo‘lgan fgetc() funksiyasi bilan aniqlangan makrosdir.

fputc() funksiyasi

int fputc(int c, FILE *stream);

prototipi bilan aniqlangan. fputc() funksiyasi fayl oqimiga argumentda koʻrsatilgan belgini yozadi (chiqaradi) va u amal qilishida putc() funksiyasi bilan bir xil.

Fayl oqimidan satr oʻqish uchun

char * fgets(char * s, int n, FILE *stream)

prototipi bilan fgets() aniqlangan. fgets() funksiyasi fayl oqimidan belgilar ketma-ketligini s satriga oʻqiydi. Funksiya oʻqish jarayonini oqimdan n-1 belgi oʻqilgandan keyin yoki keyingi satrga oʻtish belgisi (‘\n’) uchraganda toʻxtatadi. Oxirgi holda’\n’ belgisi ham s satrga qoʻshiladi. Belgilarnioʻqish tugagandan keyin s satr oxiriga, satr tugash alomati’\0’ belgisi qoʻshiladi. Agar satrni oʻqish muvaffaqiyatli boʻlsa, funksiya s argument koʻrsatadigan satrni qaytaradi, aks holda NULL.

Fayl oqimiga satrni fputs() funksiyasi yordamida chiqarish mumkin. Bu funksiya prototipi

int fputs (const char *s, FILE *stream);

koʻrinishida aniqlangan. Satr oxiridagi yangi satrga oʻtish belgisi va terminatorlaroqimga chiqarilmaydi. Oqimga chiqarish muvaffaqiyatli boʻlsa, funksiya nomanfiy son qaytaradi, aks holda EOF.

foef() funksiyasi aslida makros boʻlib, fayl ustida oʻqish-yozish amallari bajarilayotganda fayl oxiri belgisi uchragan yoki yoʻqligini bildiradi. Funksiya

int feof(FILE *stream);

prototipiga ega boʻlib u fayl oxiri belgisi uchrasa, noldan farqli sonni qaytaradi, boshqa holatlarda 0 qiymatini qaytaradi.

Quyida keltirilgan misolda faylga yozish va oʻqishga amallari koʻrsatilgan.

```
#include <iostream.h>
#include <stdio.h>
int main()
{
 char s;
```

```
 FILE *in,*out;
 if((in=fopen(“D:\\USER\\TALABA.TXT”, “rt”))==NULL)
 {
 cout<<”Talaba.txt faylini ochilmadi!!\n”;
 return 1;
}
 if((out=fopen(“D:\\USER\\TALABA.DBL”, “wt+”))==NULL)
 {
 cout<<”Talaba.dbl faylini ochilmadi!!\n”;
return 1;
}
 while (!feof(in))
 {
char c=fgetc(in);
cout<<c;
fputc(c,out);
}
 fclose(in);
 fclose(out);
 return 0;
}
```

Dasturda «talaba.txt» fayli matn fayli sifatida o‘qish uchun ochilgan va u in o‘zgaruvchisi bilan bog‘langan. Xuddi shunday, »talaba.dbl» matn fayli yozish uchun ochilgan va out bilan bog‘langan. Agar fayllarni ochish muvaffaqiyatsiz bo‘lsa, mos xabar beriladi va dastur o‘z ishini tugatadi. Keyinchalik, toki in fayli oxiriga yetmaguncha, undan belgilar o‘qiladi va ekranga, hamda out fayliga chiqariladi. Dastur oxirida ikkita fayl ham yopiladi.

Masala. G‘alvirli tartiblash usuli.

Berilgan x vektorini pufakcha usulida kamaymaydigan qilib tartiblash quyidagicha amalga oshiriladi: massivning qo‘shni elementlari x_k va x_{k+1} (k=1,..,n-

1) solishtiriladi. Agar $x_k > x_{k+1}$ bo‘lsa, u holda bu elementlar o‘zaro o‘rin almashadi. Shu yo‘l bilan birinchi o‘tishda eng katta element vektorning oxiriga joylashadi. Keyingi qadamda vektor boshidan n-1 o‘rindagi elementgacha yuqorida qayd qilingan yo‘l bilan qolgan elementlarning eng kattasi n-1 o‘ringa joylashtiriladi va h.k.

G‘alvirli tartiblash usuli pufakchali tartiblash usuliga o‘xshash, lekin x_k va x_{k+1} (k=1,2,3,..,n-1) elementlar o‘rin almashgandan keyin «g‘alvirdan» o‘tkazish amali qo‘llaniladi: chap tomondagi kichik element imkon qadar chap tomonga tartiblash saqlangan holda ko‘chiriladi [7]. Bu usul oddiy pufakchali tartiblash usuliga nisbatan tez ishlaydi.

Dastur matni:

```
#include <stdio.h>
#include <alloc.h>
int * Pufakchali_Tartiblash(int*,int);
int main()
{
char fnomi[80];
printf(“Fayl nomini kiriting:”);
scanf(“%s”, &fnomi);
int Ulcham,i=0,* Massiv;
 FILE * f1, *f2;
if((f1=fopen(fnomi, “rt”))==NULL)
 {
printf(“Xato:%s fayli ochilmadi!”,fnomi);
return 1;
 }
fscanf(f1,”%d”,&Olcham);
 Massiv=(int *)malloc(Olcham*sizeof(int));
while(!feof(f1))
fscanf(f1,”%d”,&Massiv[i++]);
```

```
fclose(f1);
 Massiv=Pufakchali_Tartiblash(Massiv,O‘lcham);
 f2=fopen(“natija.txt”, “wt”);
fprintf(f2,”%d%c”,Olcham);
for(i=0; i<Ulcham; i++)
fprintf(f2,”%d%c”,Massiv[i]);
fclose(f2);
return 0;
}
int * Pufakchali_Tartiblash(int M[],int n)
{
  int almashdi=1, vaqtincha;
  for(int i=0; i<n-1 && almashdi;i++)
  {
almashdi=0;
for(int j=0; j<n-i-1;j++)
if (M[j]>M[j+1])
{
almashdi=1;
vaqtincha=M[j];
M[j]=M[j+1];
M[j+1]=vaqtincha;
int k=j;
if(k)
while(k&& M[k]>M[k-1])
 {
vaqtincha=M[k-1];
M[k-1]=M[k];
M[k]=vaqtincha;
k--;
```

```
}
}
}
return M;
}
```

Dasturda berilganlarni oqimdan o‘qish yoki oqimga chiqarishda fayldan formatli o‘qish - fscanf() va yozish - fprintf() funksiyalaridan foydalanilgan. Bu funksiyalarning mos ravishda scanf() va printf() funksiyalaridan farqi - ular berilganlarni birinchi argument sifatida beriladigan matn fayldan o‘qiydi va yozadi.

Nomi foydalanuvchi tomonidan kiritiladigan f1 fayldan butun sonlar massivining uzunligi va qiymatlari o‘qiladi va tartiblangan massiv f2 faylga yoziladi.

Vektorni tartiblash Pufakchali_Tartiblash() funksiyasi tomonidan amalga oshiriladi. Unga vektor va uning uzunligi kiruvchi parametr bo‘ladi va tartiblangan vektor funksiya natijasi sifatida qaytariladi.

Navbatdagi ikkita funksiya fayl oqimidan formatlashmagan o‘qish-yozishni amalga oshirishga yo‘naltirilgan.

fread() funksiyasi quyidagi prototipga ega:

```
size_t fread(void*ptr,size_t size,size_t n,
FILE *stream);
```

Bu funksiya oqimdan ptr ko‘rsatib to‘rgan buferga, har biri size bayt bo‘lgan n ta berilganlar blokini o‘qiydi. O‘qish muvaffaqiyatli bo‘lsa, funksiya o‘qilgan bloklar sonini qaytaradi. Agar o‘qish jarayonida fayl oxiri uchrab qolsa yoki xatolik ro‘y bersa, funksiya to‘liq o‘qilgan bloklar sonini yoki 0 qaytaradi.

fwrite() funksiyasi prototipi

```
size_t fwrite(const void*ptr,size_t size,
size_t n,FILE *stream);
```

ko‘rinishi aniqlangan. Bu funksiya ptr ko‘rsatib to‘rgan buferdan, har biri size bayt bo‘lgan n ta berilganlar blokini oqimga chiqaradi. Yozish muvaffaqiyatli

bo‘lsa, funksiya yozilgan bloklar sonini qaytaradi. Agar yozish jarayonida xatolik ro‘y bersa, funksiya to‘liq yozilgan bloklar sonini yoki 0 qaytaradi.

Fayl ko‘rsatgichini boshqarish funksiyalari

Fayl ochilganda, u bilan «stdio.h» sarlavha faylida aniqlangan FILE tuzilmasi bog‘lanadi. Bu tuzilma har bir ochilgan fayl uchun joriy yozuv o‘rnini ko‘rsatuvchi hisoblagichni fayl ko‘rsatgichini mos qo‘yadi. Odatda fayl ochilganda ko‘rsatgich qiymati 0 bo‘ladi. Fayl ustida bajarilgan har bir amaldan keyin ko‘rsatgich qiymati o‘qilgan yoki yozilgan baytlar soniga oshadi. Fayl ko‘rsatgichini boshqarish funksiyalari - fseek(), ftell() va rewind() funksiyalari fayl ko‘rsatgichini o‘zgartirish, qiymatini olish imkonini beradi.

ftell() funksiyasining prototipi

long int ftell(FILE *stream);

ko‘rinishida aniqlangan bo‘lib, argumentda ko‘rsatilgan fayl bilan bog‘langan fayl ko‘rsatgichi qiymatini qaytaradi. Agar xatolik ro‘y bersa funksiya -1L qiymatini qaytaradi.

int fseek(FILE *stream, long offset, int from);

prototipiga ega bo‘lgan fseek() funksiyasi stream fayli ko‘rsatgichini from joyiga nisbatan offset bayt masofaga surishni amalga oshiradi. Matn rejimidagi oqimlar uchun offset qiymati 0 yoki ftell() funksiyasi qaytargan qiymat bo‘lishi kerak.from parametri quyidagi qiymatlarni qabul qilishi mumkin:

SEEK_SET (=0) - fayl boshi;

SEEK_CUR (=1) - fayl ko‘rsatgichining ayni paytdagi qiymati;

SEEK_END (=2) - fayl oxiri.

Funksiya fayl ko‘rsatgichi qiymatini o‘zgartirish muvaffaqiyatli bo‘lsa, 0 qiymatini, aks holda noldan farqli qiymat qaytaradi.

rewind() funksiyasi

void rewind(FILE *stream);

prototipi bilan aniqlangan bo‘lib, fayl ko‘rsatgichini fayl boshlanishiga olib keladi.

Quyida keltirilgan dasturda binar fayl bilan ishlash ko‘rsatilgan.

```
#include <iostream.h>
#include <stdio.h>
#include <string.h>
struct Shaxs
{
 char Familiya[20];
 char Ism[15];
 char Sharifi[20];
};
int main()
{
int n,k;
 cout<<"Talabalar sonini kiriting: " ;cin>>n;
 FILE *oqim1,*oqim2;
 Shaxs *shaxs1, *shaxs2, shaxsk;
 shaxs1=new Shaxs[n];
 shaxs2=new Shaxs[n];
 if ((oqim1=fopen("Talaba.dat", "wb+"))==NULL)
 {
 cout<<"Talaba.dat ochilmadi!!!";
return 1;
}
 for(int i=0; i<n; i++)
 {
 cout<<i+1<<"- shaxs ma'lumotlarini kiriting:\n";
cout<<"Familiyasi: ";gets(shaxs1[i].Familiya);
cout<<"Ismi: "; gets(shaxs1[i].Ism);
cout<<"Sharifi: ";gets(shaxs1[i].Sharifi);
}
 if (n==fwrite(shaxs1,sizeof(Shaxs),n,oqim1))
```

```
cout<<”Berilganlarni yozish amalga oshirildi!\n”;
 else
 {
 cout<<”Berilganlarni yozish amalga oshirilmadi!\n”;
return 3;
 }
 cout<<”Fayl uzunligi: “<<ftell(oqim1)<<'\n';
 fclose(oqim1);
 if((oqim2=fopen(“Talaba.dat”, «rb+»))==NULL)
 {
 cout<<”Talaba.dat o‘qishga ochilmadi!!!”;
return 2;
 }
 if (n==fread(shaxs2,sizeof(Shaxs),n,oqim2))
 for(int i=0; i<n; i++)
 {
 cout<<i+1<<”- shaxs ma'lumotlari:\n”;
cout<<”Familiyasi: “<<shaxs2[i].Familiya<<'\n';
cout<<”Ismi: “<<shaxs2[i].Ism<<'\n';
cout<<”Sharifi: “<<shaxs2[i].Sharifi<<'\n';
cout<<”*****************************\n”; }
 else
 {
 cout<<”Fayldan o‘qish amalga oshirilmadi!\n” ;
return 4;
 }
 do
{
cout<<”Yo‘zuv nomerini kiriting(1..”<<n<<”):”;
 cin>>k;
```

```
} while (k<0 && k>n);
 k--;
 cout<<”Oldingi Familiya: “;
cout<<shaxs2[k].Familiya<<'\n';
 cout<<”Yangi Familiya: “;
 gets(shaxs2[k].Familiya);
 if (fseek(oqim2, k*sizeof(Shaxs),SEEK_SET))
 {
 cout<<”Faylda”<<k+1;
 cout<<”-yozuvga o‘tishda xatolik ro‘y berdi???\n”;
 return 5;
}
 fwrite(shaxs2+k,sizeof(Shaxs),1,oqim2);
 fseek(oqim2, k*sizeof(Shaxs),SEEK_SET);
 fread(&shaxsk,sizeof(Shaxs),1,oqim2);
 cout<<k+1<<”- shaxs ma'lumotlari:\n”;
 cout<<”Familiyasi: “<<shaxsk.Familiya<<'\n';
 cout<<”Ismi: “<<shaxsk.Ism<<'\n';
 cout<<”Sharifi: “<<shaxsk.Sharifi<<'\n';
 fclose(oqim2);
 delete shaxs1;
 delete shaxs2;
 return 0;
}
```

Yuqorida keltirilgan dasturda, oldin «Talaba.dat» fayli binar fayl sifatida yozish uchun ochiladi va u oqim1 o‘zgaruvchisi bilan bog‘lanadi. Shaxs haqidagi ma’lumotni saqlovchi n o‘lchamli dinamik shaxs1 tuzilmalar massivi oqim1 fayliga yoziladi, fayl uzunligi chop qilinib fayl yopiladi. Keyin, xuddi shu fayl oqim2 nomi bilan o‘qish uchun ochiladi va undagi berilganlar shaxs2 strukturalar massiviga o‘qiladi va ekranga chop qilinadi. Dasturda fayldagi yozuvni

o‘zgartirish (qayta yozish) amalga oshirilgan. O‘zgartirish qilinishi kerak bo‘lgan yozuv tartib nomeri foydalanuvchi tomonidan kiritiladi (k o‘zgaruvchisi) va shaxs2 tuzilmalar massividagi mos o‘rindagi tuzilmaning Familiya maydoni klavia-turadan kiritilgan yangi satr bilan o‘zgartiriladi. oqim2 fayl ko‘rsatgichi fayl boshidan k*sizeof(Shaxs) baytga suriladi va shaxs2 massivning k - tuzilmasi (shaxs2+k) shu o‘rindan boshlab faylga yoziladi. Keyin oqim2 fayli ko‘rsatgichi o‘zgartirish kiritilgan yozuv boshiga qaytariladi va bu yozuv tuzilmasiga o‘qiladi hamda ekranga chop etiladi.

Masala. Haqiqiy sonlar yozilgan f fayli berilgan. f fayldagi elementlarning o‘rta arifmetigidan kichik bo‘lgan elementlar miqdori aniqlansin.

Masalani yechish uchun f faylini yaratish va qaytadan uni o‘qish uchun ochish zarur bo‘ladi. Yaratilgan faylning barcha elementlarining yig‘indisi s o‘zgaruvchisida hosil qilinadi va u fayl elementlari soniga bo‘linadi. Keyin f fayl ko‘rsatgichi fayl boshiga olib kelinadi va elementlar qayta o‘qiladi va s qiymatidan kichik elementlar soni - k sanab boriladi.

Faylni yaratish va undagi o‘rta arifmetikdan kichik sonlar miqdorini aniqlashni alohida funksiya ko‘rinishida aniqlash mumkin.

Dastur matni:

```
#include <iostream.h>
#include <stdio.h>
#include <string.h>
int Fayl_Yaratish()
{
 FILE * f;
 double x;
// f fayli yangidan hosil qilish uchun ochiladi
 if ((f=fopen(“Sonlar.dbl”, “wb+”))==NULL)return 0;
 char *satr=new char[10];
 int n=1;
 do
```

```
{cout<<"Sonni kiriting(bo‘sh satr tugatish): “;
gets(satr);
if(strlen(satr))
{x=atof(satr);
 fwrite(&x,sizeof(double),n,f);
}
}while(strlen(satr));// satr bo‘shbo‘lmasa,takrorlash
fclose(f);
return 1;
}
int OAdan_Kichiklar_Soni()
{
FILE * f;
double x;
f=fopen(“Sonlar.dbl”, “rb+”);
double s=0; // s - f fayl elementlari yig‘indisi
while (!feof(f))
{
if (fread(&x,sizeof(double),1,f)) s+=x;
}
long sonlar_miqdori=ftell(f)/sizeof(double);
s/=sonlar_miqdori;// s- o‘rta arifmetik
cout<<"Fayldagi sonlar o‘rta arifmetiki="<<s<<endl;
fseek (f,SEEK_SET,0);// fayl boshiga kelinsin
int k=0;
while (fread(&x, sizeof(x),1,f))
{
k+=(x<s); //o‘rta arifmetikdan kichik elementlar soni
}
fclose(f);
```

```
 return k;
}
int main()
{
if(Fayl_Yaratish())
 {
 cout<<”Sonlar.dbl faylidagi\n”;
int OA_kichik=OAdan_Kichiklar_Soni();
cout<<”O‘rta arifmetikdan kichik sonlar miqdori=”;
cout<<OA_kichik;
 }
else // f faylini yaratish muvaffaqiyatsiz bo‘ldi.
 cout<<”Faylini ochish imkoni bo‘lmadi!!!”;
return 0;
}
```

Dasturda bosh funksiyadan tashqari ikkita funksiya aniqlangan:

int Fayl_Yaratish() - diskda «Sonlar.dbl» nomli faylni yaratadi. Agar faylni yaratish muvaffaqiyatli bo‘lsa, funksiya 1 qiymatini, aks holda 0 qiymatini qaytaradi. Faylni yaratishda klaviaturadan sonlarning satr ko‘rinishi o‘qiladi va songa aylantirilib, faylga yoziladi. Agar bo‘sh satr kiritilsa, sonlarni kiritish jarayoni to‘xtatiladi va fayl yopiladi;

int OAdan_Kichiklar_Soni()-diskdagi»Sonlar.dbl» nomli fayli o‘qish uchun ochiladi va fayl elementlarining s o‘rta arifmetigidan kichik elementlari soni topiladi va funksiya natijasi sifatida qaytariladi.

Bosh funksiyada faylni yaratish muvaffaqiyatli kechganligi tekshiriladi va shunga mos xabar beriladi.

Oldindan berilgan kattaliklarni - Ob’yektlarni kiritish –chiqarish C++ tilida kiritish-chiqarish oqimlarining sinflari mavjud bo‘lib, ular kiritish-chiqarish standart kutubxonasining Ob’yektga yo‘naltirilgan ekvivalentidir. Ular quyidagilar:

istream - kiritish oqimi

ostream - chiqrish oqimi

iostream - kiritish/chiqarish oqimi

Satrli oqimlar xotirada joylashtirilgan satrli buferlardan ma’lumotlarni kiritish-chiqarish uchun xizmat qiladi.

istrstream - satrli kiritish

ostrstream - satrli chiqarish

strstream - satrli kiritish/chiqarish

Quyidagi faylli okimlar fayllar bilan ishlash uchun xizmat qiladi.

ifstream - faylli kiritish

ofstream - faylli chiqarish

fstream - faylli kiritish/chiqarish

Odatda bu oqimlar include < …..> sifatida yoziladi. ifstream, ofstream va fstream oqimlari dasturda fayllar hosil qilish, ulardagi ma’lumotlardan foydalanish uchun ishlatiladi. Ularning qo‘llanilishi quyidagicha:

ofstream name (“ path\ file_name”); - ma’lumotli fayl hosil qilish, ya’ni ma’lumotlar bazasi uchun ochish;

Masalan: ofstream farruh(“c:\tcpp\bin\d11.dat”);

ofstream alibek (“nnn.txt”);

Bu yerda

name - ixtiyoriy nom (lotincha), ya’ni oqim nomi. Keyinchalik fayldagi ma’lumotlarni yozish yoki o‘kish uchun shu nomdan foydalanamiz.

d11.dat va nnn.txt biz hosil qilgan fayl nomlari bo‘lib, ular oqim nomlari bilan bog‘langandir.

Endi hosil bo‘lgan ma’lumotlardan foydalanish uchun uni ochishni ko‘ramiz:

ifstream name (“path”);

Masalan: ifstream farruh («c:\ tcpp\bin\d11.dat»);

ifstream alibek(«nnn.txt»);

Ochilgan fayllarni albatta yopish kerak! Bu jarayonni quyidagicha amalga oshiriladi: name.close( );

Masalan, farruh.close( ); yoki alibek.close( );

Demak, farruh bilan d11.dat, alibek bilan nnn.txt nomlari o‘zaro ma’lumot almashinuvini ta’minlaydi.

Masalan:

2 ta butun sonni va ularning yig‘indisini o‘zida saqlovchi fayl hosil qiling va undan keyingi dasturda foydalaning.

Sonlarni a, b, yigindini s, faylni ttt.dat, oqim nomini jasur deb ataymiz.

```
#include <iostream.h>
#include <fstream.h>
#include <conio.h>
int main ( )
{
int a=12, b=13, s;
ofstream jasur («ttt.dat»);
s = a + b;
sout <<”s=” << s << endl;
jasur << s <<endl;
jasur.close ( );
getch ( );
}
```

Endi undan foydalanamiz:

```
// #include <iostream.h>
#include <fstream.h>
#include <conio.h>
#include <math.h>
int main ( )
{
int  s; float s1;
```

```
ifstream jasur («ttt.dat»);
jasur >>s;
s1 = sin (s);
cout <<«s1=«<<s1<<endl;
jasur.close ( );
getch( );  }
```

2-misol. Matrisa va vektorlar berilgan. Son qiymatlari ixtiyoriy. Ushbu qiymatlardan foydalanib, matrisani vektorga ko‘paytirish, matrisaning izini hisoblash va vektorning yig‘indisini hisoblash dasturini tuzing.

Avval matrisa va vektorlarning son qiymatlarini o‘zida saqlovchi fayl hosil qilamiz.

So‘ngra bu ma’lumotlardan foydalanamiz.

```
#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include <time.h>
int main ( )
{ srand (time (0));
int a [3][3], b[3], i, j;
ofstream said («akbar.txt»);
for ( i=0; i<3; i++)
{ for (j=0; j<3; j++)
{ a[i][j] = rand( );
said <<a[i][j]; } }
for (i=0; i<3; i++)
{ b[i] = rand( );
said << b[i]; }
said.close ( );
}
```

```
#include <iostream.h>
#include <fstream.h>
int main ( )
{
int a[3][3], b[3], i, j, c[3], s1=0, s2=0;
ifstream said («akbar.txt»);
for ( i=0; i<3; i++)
for (j=0; j<3; j++)
said >>a[i][j];
for ( i=0; i<3; i++)
said >> b[i];
for ( i=0; i<3; i++)
{ c[i] = 0;
for (j=0; j<3; j++)
c[i] = c[i] + a[i][j] * b[i];
cout << «c=«<<c[i]<<endl; }
for ( i=0; i<3; i++)
s1 = s1 + a[i][i];
for ( i=0; i<3; i++)
s2=s2 + b[i];
cout << «s1=«<<s1<<«  s2=«<<s2<<endl;
}
```

Muhokama savollari

1. Matn va binar fayllar
2. O‘qish-yozish oqimlari. Standart oqimlar
3. Fayllarni tashkil etish
4. Simvollarni o‘qish-yozish funksiyalari
5. Fayllarni oqim ko‘rinishida tashkillashtirish.

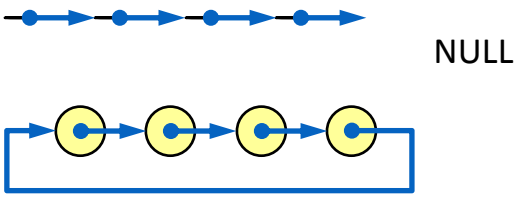
Nazorat savollari

1. Fayllar bilan ishlash asoslari.
2. Simvollarni o‘qish-yozish funksiyalari
3. Fayldan o‘qish-yozish funksiyalari
4. Fayl ko‘rsatgichini boshqarish funksiyalari
5. Kerakli include larni ko‘rsating
6. Foydalanuvchining direktivasini yaratish

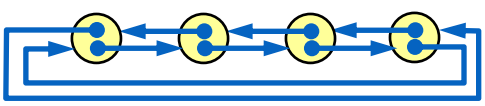
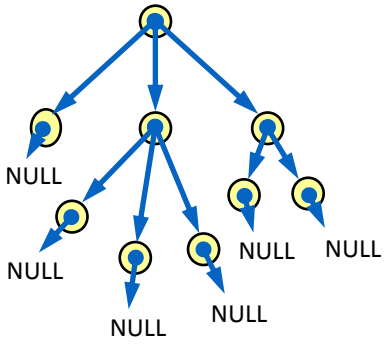
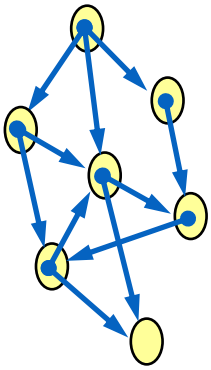
3.7. Ma’lumotlarning dinamik informasion tuzilmasi: ro‘yxatlar, steklar, navbatlar

Ro‘yhat havolalar yordamida birlashgan bog‘lamalar to‘plamini tashkil etadi[6]. Bog‘lama tuzilmasi bevosita ma’lumot va boshqa bog‘lamalarga havolalarni o‘z ichiga oladi.

Strukturalar toifasi:

<table>
<tr>
<td>Ro‘yhatlar
Bir yo‘nalishli
 

ikki yo‘nalishli
 

siklik
  </td>
<td>daraxtlar
 
</td>
<td>graflar
  </td>
</tr>
</table>

Dasturlashda ro‘yhatlar zaruratini tushunish uchun quyidagi masalani ko‘rib chiqamiz: Faylda matn yozilgan. Ikkinchi faylga matndagi barcha so‘zlar ustun ko‘rinishida alifbo tartibida yozilishi, hamda har bir so‘zning takrorlanish soni aniqlanishi lozim.

Bunda quyidagi muammolarga duch kelish mumkin:

1) So‘zlar soni noma’lum (statik massiv);
2) So‘zlar soni faqat xarakatlar oxirida aniqlanadi (dinamik massiv).

Ushbu muammolar yechimi dinamik tuzilma-ro‘yhatni qo‘llash orqali topiladi. Masalani yechish uchun quyidagi algoritmdan foydalanamiz:

1) Agar fayldagi so‘zlar tugagan bo‘lsa, to‘xtash.
2) So‘zlarni o‘qib, ularni ro‘yhatdan qidirish;
3) Agar so‘z topilgan bo‘lsa, qaytarilish soni hisobchisi qiymatini oshirish, aks holda so‘zni ro‘yhatga qo‘shish;
4) 2 qadamga o‘tish.

Yuqoridagilarga ko‘ra ro‘yhatga quyidagi rekursiv ta’rif berish mumkin:

5) bo‘sh struktura
6) Bosh bog‘lama va u bilan bog‘liq ro‘yhat.

Bog‘lama strukturasi:

```
struct Node {
 char word[40]; // so‘z
 int     count;   // hisobchi
 Node *next;   // keyingi elementga havola
 };
```

Ushbu strukturaga ko‘rsatgich

```
typedef Node *PNode;
```

Ro‘yhat boshining adresi:

```
PNode Head = NULL;
```

Ro‘yhatga murojaat uchun uning boshi adresini bilish yetarli!

Ro‘yhatlar ustida quyidagi xarakatlarni bajarish mumkin:

1. Yangi bog‘lamani tashkil etish.

2. Bog‘lamani qo‘shish:
   a) Ro‘yhat boshiga;
   b) Ro‘yhat oxiriga;
   c) Belgilangan bog‘lamadan so‘ng;
   d) Belgilangan bog‘lamagacha.
3. Ro‘yhatdan kerak bog‘lamani qidirish.
4. Bog‘lamani o‘chirish.

Bog‘lamani tashkil etish uchun CreateNode funksiyasi (bog‘lamani tashkil etish)dan foydalanamiz. Bunda

kirish: fayldan o‘qilgan yangi so‘z;

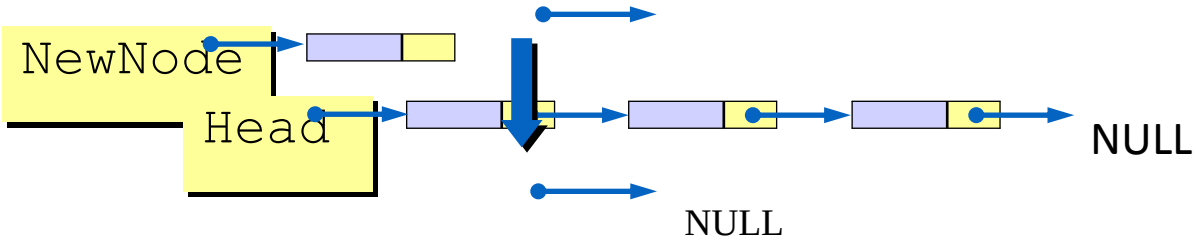
chiqish: xotirada tashkil etilgan yangi bog‘lama adresi.

```
PNode CreateNode // Tashkil etilgan bog‘lama adresini qaytaradi
( char NewWord[] )// Yangi so‘z
{
PNode NewNode = new Node;
strcpy(NewNode->word, NewWord);
NewNode->count = 1;
NewNode->next = NULL;
return NewNode;
}
```

Bog‘lamani ro‘yhat boshiga qo‘shishni quyidagicha bajarish lozim:

1) Yangi bog‘lamaga havolani ro‘yhat boshiga o‘rnatamiz:

NewNode->next = Head;



2) Yangi bog‘lamani ro‘yhat boshi qilamiz:

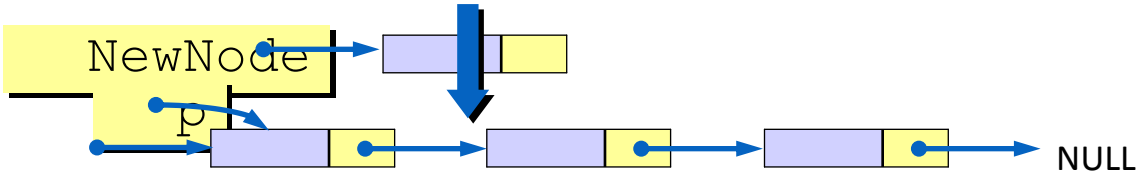
void AddFirst (PNode & Head, PNode NewNode)// Bosh elementning adresi o‘zgarmoqda

```
{
        NewNode->next = Head;
        Head = NewNode;
       }
```

Belgilangandan keyin bog‘lamani qo‘shishni quyidagicha bajaramiz:

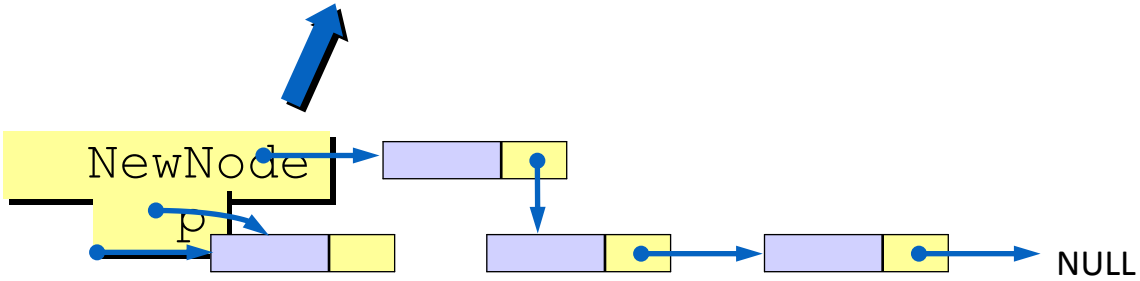
1) P dan keyingi bog‘lamaga yangi bog‘lamaga havolani o‘rnatamiz:

NewNode->next = p->next;



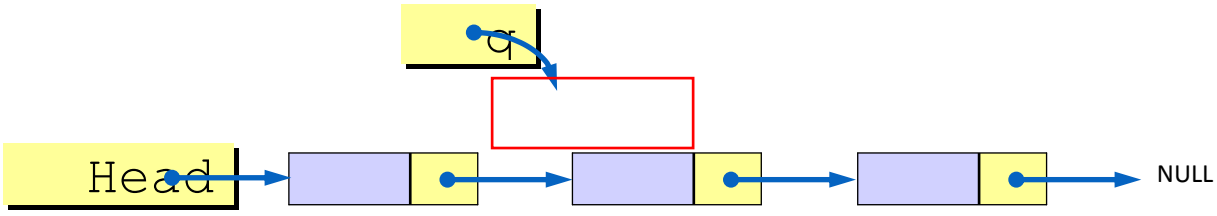
2) P bog‘lamaning havolasini yangisiga o‘rnatish:

p->next = NewNode;




```
void AddAfter (PNode p, PNode NewNode)
{
  NewNode->next = p->next;
  p->next = NewNode;
}
```

Ro‘yhat elementining har biri ustida biror bir xarakat bajarish uchun quyidagi algoritmdan foydalanamiz:



Algoritm:

1.Qo‘shimcha ko‘rsatgich q ni ro‘yhat boshiga o‘rnatamiz.

2. Agar q = NULL (ro‘yhat oxiri) bo‘lsa, algoritm ishini tugallaydi

3. Q adresli bog‘lama ustida xarakat bajariladi

4. keyingi bog‘lamaga o‘tish bajariladi: q->next.

```
...
PNode q = Head;    // boshidan boshlandi
while ( q != NULL ) { // oxiriga yetmagunga qadar
 ...        // q ning ustida xarakat
 q = q->next;    // keyingi bog‘lamaga o‘tish

 }
...
```

Ro‘yhat oxiriga bog‘lama qo‘shish uchun quyidagi xarakatlarni bajaramiz:

Algoritm:

1) q->next = NULL bo‘lgan q ni qo‘shish;

2) Q adreslidan keyin qo‘shish (AddAfter prosedurasi).

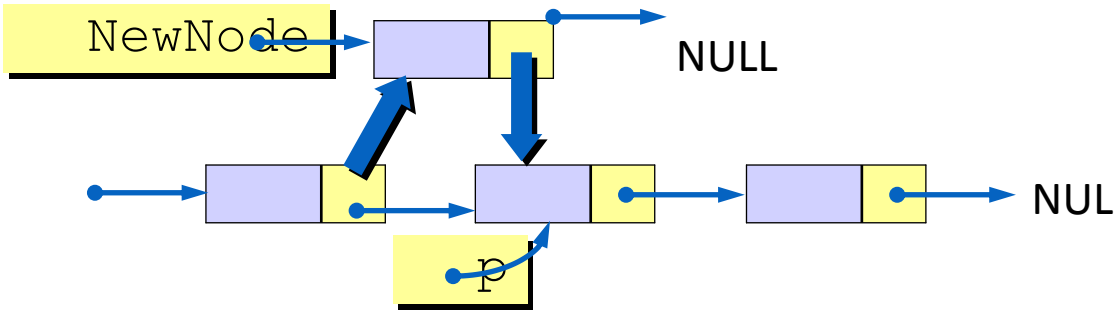
Alohida holat: bo‘sh ro‘yhatga qo‘shish.

```
void AddLast ( PNode &Head, PNode NewNode )
{
PNode q = Head;
if ( Head == NULL ) {
  AddFirst( Head, NewNode );// Alohida holat: bo‘sh ro‘yhatga qo‘shish
  return;
  }
while ( q->next ) q = q->next;// Oxirgi bog‘lamani qidiramiz
```

AddAfter (q, NewNode);// Q dan keyin yangi bog‘lamani qo‘shish

}

Belgilangandan avval bog‘lama qo‘shish quyidagi ko‘rinishga ega:



Bunda muammo sifatida oldingi bog‘lama adresi zarurati tug‘iladi, ammo orqaga yurish mumkin emas!

Yechim: Oldingi q bog‘lamani topish (ro‘yhat boshidan o‘tish).

void AddBefore (PNode & Head, PNode p, PNode NewNode)

{

PNode q = Head;

if (Head == p)

{

AddFirst (Head, NewNode);// Alohida holat: ro‘yhat boshiga qo‘shish

return;

}

while (q && q->next != p) q = q->next;// Keyingisi p bo‘lgan bog‘lamani qidiramiz

if (q) AddAfter(q, NewNode);// q dan keyin bog‘lama qo‘shish

}

Ro‘yhatdan so‘z qidirish uchun:

Ro‘yhatdan kerakli so‘zni topish yoki mavjud emasligini aniqlash lozim.

Find funksiyasi:

kirish: so‘z (simvolli satr);

chiqish: so‘zga ega bog‘lama adresi, yoki NULL.

Algoritm sifatida ro‘yhat bo‘yicha o‘tishni tanlashimiz mumkin.

PNode // natija – bog‘lama adresi

Find (PNode Head, char NewWord[]) // So‘zni qidiramiz

{

PNode q = Head;

while (q && strcmp(q->word, NewWord))

q = q->next;// so‘z topilmagunga yoki ro‘yhat oxiriga yetmagunga qadar

return q;

}

Yangi so‘zni qayerga qo‘shish kerak?

Ro‘yhatda alifbo tartibini saqlagan holda so‘z qo‘shilishi lozim bo‘lgan bog‘lamani topish lozim.

FindPlace funksiyasi:

kirish: so‘z (simvolli satr);

chiqish: so‘z qo‘shilishi kerak bo‘lgan bog‘lama adresi yoki

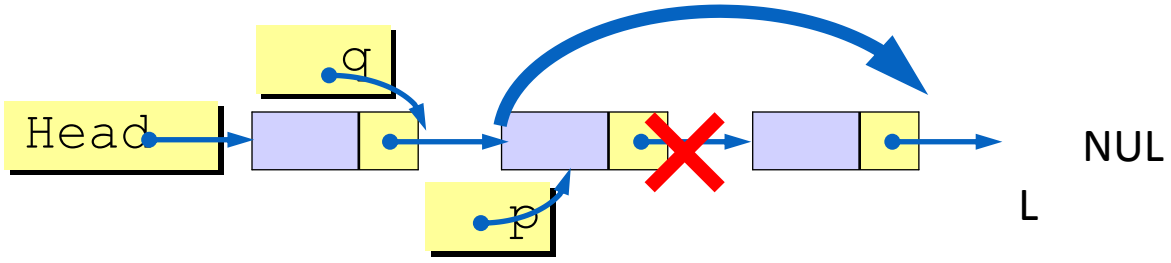
NULL, agar so‘z ro‘yhat oxiriga qo‘shilishi lozim.

```
PNode FindPlace ( PNode Head, char NewWord[] )
{
PNode q = Head;
while ( q && strcmp(NewWord, q->word) > 0 )
 q = q->next;
return q;// NewWord so‘zi alifbo bo‘yicha  q->word gacha turadi

}
```

Bog‘lamani o‘chirish uchun

Oldingi q bog‘lama adresini bilish lozim.



```
void DeleteNode ( PNode &Head, PNode p )
{
PNode q = Head;
if ( Head == p )
  Head = p->next;
else // Alohida holat: birinchi bog‘lamani o‘chiramiz
{
 while(q&&q->next!=p) // q->next == p bo‘lgan avvalgi bog‘lamani qidiramiz
 q = q->next;
  if ( q == NULL ) return;
  q->next = p->next;
  }
delete p; // Xotirani bo‘shatish
 }
```

Alifbo-chastotali lug‘atni tashkil etish algoritmi:

1. Faylni o‘qish uchun ochish;

```
FILE *in;
in = fopen ( «input.dat», «r» );// read
```

2. So‘zni o‘qish:

```
char word[80];
...
n = fscanf ( in, «%s», word );// Faqat bitta so‘z kiritiladi (probelgacha)!
```

3. Fayl tugasa (n!=1), 7 qadamga o‘tish;
4. So‘z topilgan bo‘lsa, hisobchini oshirish (count);
5. Agar so‘z ro‘yhatda bo‘lmasa, unda
6. Yangi bog‘lama tashkil etib, maydonlarni to‘ldirish (CreateNode);
7. So‘z qo‘shilishi lozim bo‘lgan bog‘lamani topish (FindPlace);
8. bog‘lamani qo‘shish *(*AddBefore*);* 2 qadamga o‘tish;
   a) Ro‘yhat bo‘yicha o‘tib, so‘zlar ro‘yhatini chiqarish.
   b) 2 qadamga o‘tish;

c) Ro‘yhat bo‘yicha o‘tib, so‘zlar ro‘yhatini chiqarish

Stek

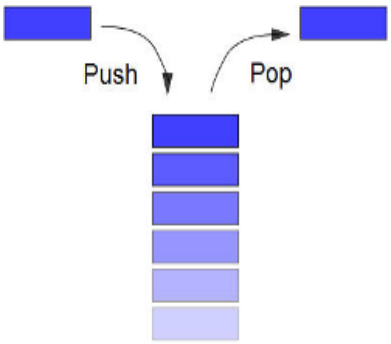



Stek – ma’lumotlarning chiziqli strukturasi bo‘lib, unga elementlarni qo‘shish yoki o‘chirish faqat bir uchdan (stek cho‘qqisidan) amalga oshiriladi. (*Stack* = uyum, to‘plam)

LIFO = *Last In – First Out* « Oxirgi kirgan birinchi chiqadi».

Stek ustida amallar:

1) Cho‘qqiga element qo‘shish (*Push = itarib qo‘shish*);
2) Cho‘qqidan element chiqarish *(Pop = itarib chiqarish).*



Masalan: Tarkibida [], {} i () qavslar mavjud bo‘lgan simvolli satr kiritilgan. Qavslar to‘g‘ri qo‘yilganligi aniqlansin (boshqa simvollarni e’tiborga olmasdan).

Masalan: [()]{}][[({})]

Masalani soddalashtirilgan holda ko‘rib chiqib, bir toifa qavslarni e’tiborga olamiz.

Yechim: qavslarning ichma-ich joylashganligi hisobchisini kiritamiz. Agar hisobchining qiymati satr oxirida nolga teng bo‘lsa va satr ko‘rilishida biror marta ham manfiy bo‘lmasa, qavslar soni teng.

(	(	)	)	(	)
1	2	1	0	1	0

(	(	)	)	)	(
1	2	1	0	-1	0

NewNode->next = p->next;

Dastlabki masalani xuddi shunday, ammo uchta hisobchi bilan ishlash mumkinmi?

Masala yechimi.

Algoritm:

1) dastlab stek bo‘sh;
2) Siklda tartib bilan satrning barcha simvollarini ko‘rib chiqamiz;
3) Agar navbatdagi simvol – ochiluvchi qavs bo‘lsa, uni stek cho‘qqisiga joylashtiramiz;
4) agar simvol – yopiluvchi qavs bo‘lsa, stek cho‘qqisini tekshiramiz: unda mos yopiluvchi qavs bo‘lishi kerak (aks holda , xato);
5) Agar oxirida stek bo‘sh bo‘lmasa, ifoda noto‘g‘ri.

Stek tadbiqi

Struktura-stek:

```
const MAXSIZE = 100;
struct Stack {
  char data[MAXSIZE]; // 100 simvolli stek
  int size;
  };
```

Element qo‘shish:

```
int Push ( Stack &S, char x )
{
if ( S.size == MAXSIZE ) return 0;// xato: stek to‘lib ketdi

S.data[S.size] = x;// Element qo‘shilsin
S.size ++;
return 1;// Xato yo‘q
}
```

Stek tadbiqi (massiv).

Elementni cho‘qqidan olish:

```
char Pop ( Stack &S )
{
if ( S.size == 0 ) return char(255);// xato: stek bo‘sh
S.size --;
return S.data[S.size];
}
// Bo‘shmi?
int isEmpty ( Stack &S )
{
if ( S.size == 0 )
   return 1;
else return 0;
}
int isEmpty ( Stack &S )
{
return (S.size == 0);
}
```

Dastur.

```
void main()
{
 char br1[3] = {‘(’,’[’,’{' }; // Ochiluvchi qavslar
 char br2[3] = {‘)’,’]’,’}' }; // Yopiluvchi qavslar
 char s[80], upper;// Stekdan olingan qiymatlar
 int i, k, error = 0;// Xatolik ko‘rsatgichi
 Stack S;
 S.size = 0;
 printf(“Qavsli ifodani kiriting > ”);
 gets ( s );
```

```
 ... // bu yerda ishlov beruvchi asosiy sikl bo‘ladi
 if ( ! error && (S.size == 0) )
    printf(”\n Ifoda to‘g‘ri \n”);
 else printf(“\n Ifoda noto‘g‘ri\n”);
}
      int isEmpty ( Stack &S )
      {
      return (S.size == 0);
      }
```

Satrga ishlov berish (asosiy sikl)

```
for ( i = 0; i < strlen(s); i++ ) // s satrning barcha simvollari bo‘yicha sikl
 {
 for ( k = 0; k < 3; k++ ) // Turli qavslar bo‘yicha sikl
  {
  if ( s[i] == br1[k] ) // agar ochiluvchi qavs
   {
   Push ( S, s[i] ); // stekka kiritish
   break;
   }
  if ( s[i] == br2[k] ) // agar yopiluvchi qavs
   {
   upper = Pop ( S ); // yuqoridagi elementni olish
   if ( upper != br1[k] ) error = 1;// xato: stek bo‘sh yoki mos qavs emas
   break;
   }
  }
 if ( error ) break;// xato bo‘lsa, davomini tekshirishga xojat yo‘q
 }
```

Stek tadbiqi (ro‘yhat)

Bog‘lama strukturasi:

```
struct Node {
char data;
  Node *next;
  };
typedef Node *PNode;
```

Element qo‘shish:

```
void Push (PNode &Head, char x)
{
 PNode NewNode = new Node;
 NewNode->data = x;
 NewNode->next = Head;
 Head = NewNode;
}
```

Stek tadbiqi (ro‘yhat)

Elementni cho‘qqidan olish:

```
char Pop (PNode &Head) {
 char x;
 PNode q = Head;
 if ( Head == NULL ) return char(255); // Stek bo‘sh
 x = Head->data;
 Head = Head->next;
 delete q;
 return x;
}
```

Asosiy dasturdagi o‘zgarishlar:

```
Stack S; //PNode S = NULL;
S.size = 0;
...
```

```
if ( ! error && (S.size == 0) )// (S == NULL)
   printf(«\n Ifoda to‘g‘ri \n»);
else printf(«\nIfoda noto‘g‘ri \n»);
```

Arifmetik ifodani hisoblash

Avtomatik tarzda hisoblash:

Infiks yozuv.

(a + b) / (c + d – 1)

(amallar operandlar orasida)

 Qavslar zarur!

Prefiks yozuv (amallar operandlardan oldin)

/ a+b c+d-1 Polyakcha yozuv, Jan Łukasiewicz (1920)

 Qavslar zarur emas, hisoblash mumkin!

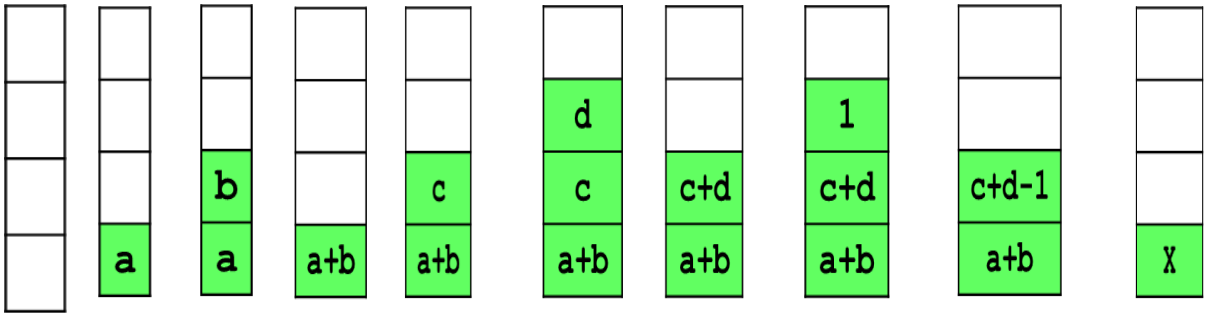
Postfiks yozuv (amallar operandlardan keyin)

a+b c+d-1 Polyakcha teskari yozuv, F. L. Bauer and E. W. Dijkstra

Ifodani hisoblash.

Postfiks ko‘rinish:

x=a b + c d +1 - /



Algoritm:

1) Navbatdagi elementni olish;
2) Agar bu amal bo‘lmasa, uni stekka qo‘shish;
3) Agar bu amal bo‘lsa, unda
   - Unda stekdan ikkita operand olish;

- amalni bajarib, natijani stekka yozish;

4) 1 qadamga o‘tish.

Navbat

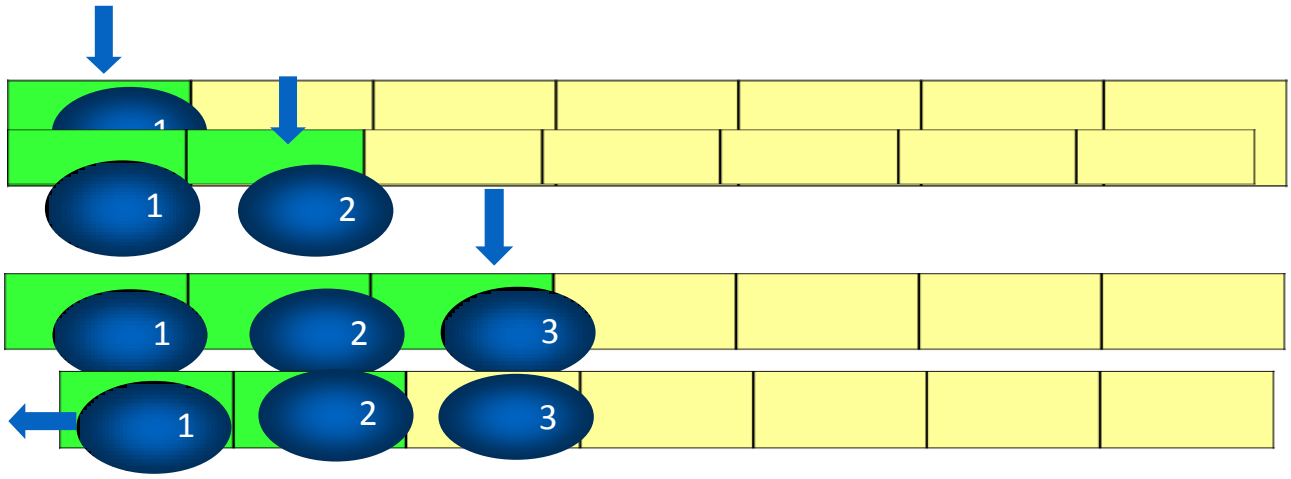
Navbat – ma’lumotlarning chiziqli strukturasi bo‘lib, unga element kiritish bir uchdan(navbat boshidan), element o‘chirish boshqa uchdan amalga oshiriladi (navbat oxiridan).

FIFO = *First In – First Out*

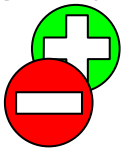
Navbat bilan amallar:

1) Elementni navbat oxiriga qo‘shish (*PushTail = oxiriga kiritish*);
2) Navbat boshidan elementni olib tashlash *(Pop)*.

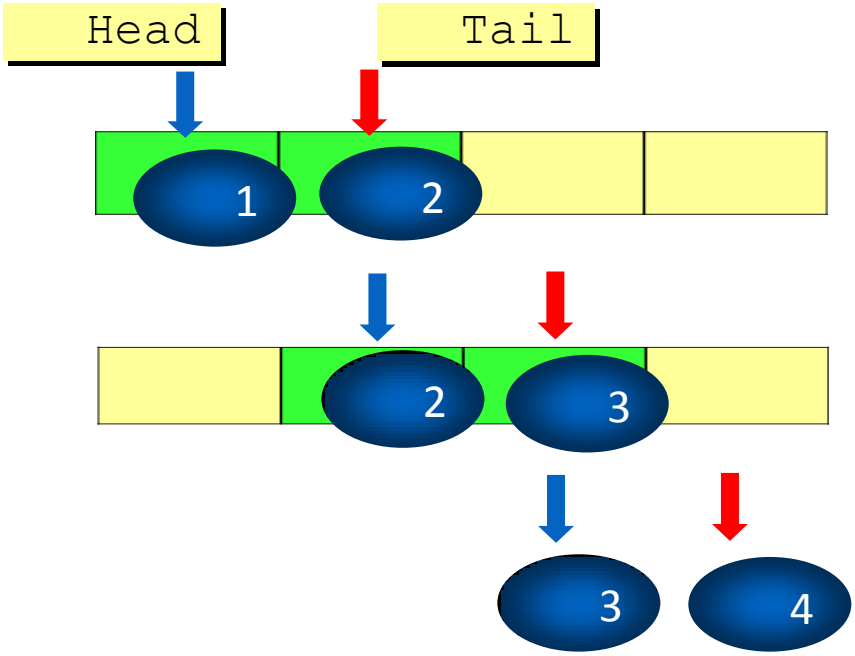
<u>Navbat tadbiqi (massiv).</u>



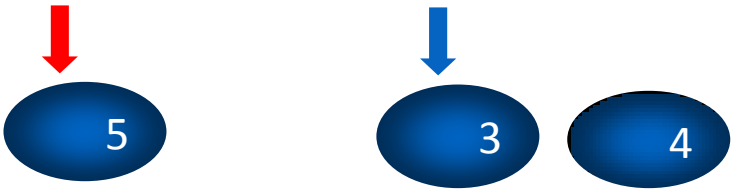
eng oddiy usul



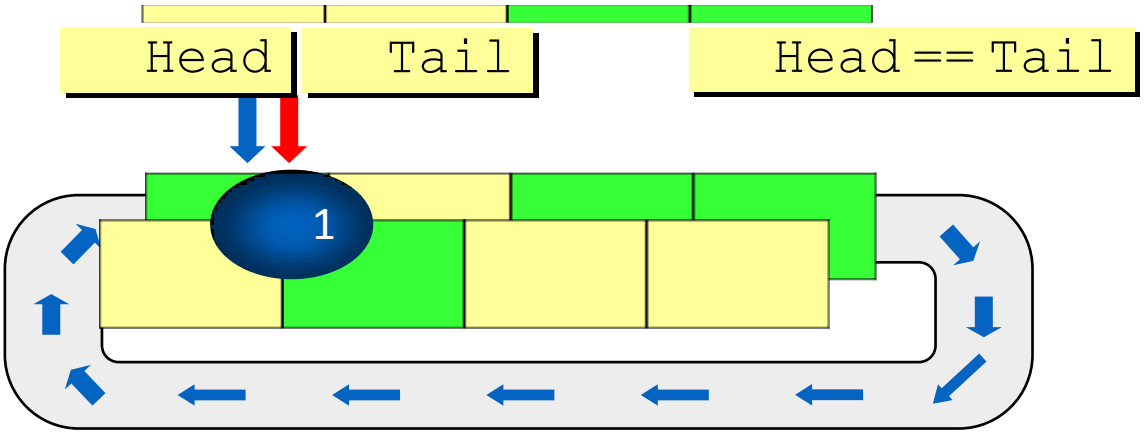
1) oldindan massivni ajratish;
2) Navbatdan tanlashda barcha elementlarni siljitish lozim



Navbat tadbiqi (doiraviy massiv).



Navbat tadbiqi (doiraviy massiv). Navbatda 1 element:



Navbat tadbiqi (doiraviy massiv).

Ma’lumotlar strukturasi:

```
const MAXSIZE = 100;
struct Queue {
  int data[MAXSIZE];
```

```
  int head, tail;
  };
```

Navbatga qo‘shish:

```
int PushTail ( Queue &Q, int x )
{
  if ( Q.head == (Q.tail+2) % MAXSIZE )// Doirani birlashtirish
 // Navbat to‘la, qo‘shish mumkin emas
   return 0;
  Q.tail = (Q.tail + 1) % MAXSIZE;
  Q.data[Q.tail] = x;
  return 1; // Muvaffaqiyatli qo‘shish
}
```

Navbat tadbiqi (doiraviy massiv)

Navbatdan tanlash:

```
int Pop ( Queue &Q )
{
  int temp;
  if ( Q.head == (Q.tail + 1) % MAXSIZE ) //Navbat bo‘sh
   return 32767;
  temp = Q.data[Q.head];// Birinchi elementni olish
  Q.head = (Q.head + 1) % MAXSIZE;// Uni navbatdan o‘chirish
  return temp;
}
```

Navbat tadbiqi (ro‘yhat)

Bog‘lama strukturasi:

```
struct Node {
  int data;
  Node *next;
  };
typedef Node *PNode;
```

«navbat» ma’lumotlar toifasi:

```
struct Queue {
  PNode Head, Tail;
  };
```

Navbat tadbiqi (ro‘yhat).

Elementni qo‘shish:

```
void PushTail ( Queue &Q, int x )
{
  PNode NewNode;
  NewNode = new Node; // Yangi bog‘lamani tashkil etish
  NewNode->data = x;
  NewNode->next = NULL;
  if ( Q.Tail )
    Q.Tail->next = NewNode;// Agar ro‘yhatda nimadir bo‘lsa, oxiriga qo‘shamiz
  Q.Tail = NewNode;
  if ( Q.Head == NULL ) //Agar ro‘yhatda hech nima bo‘lmasa, …
    Q.Head = Q.Tail;
}
```

Navbat tadbiqi (ro‘yhat)

Elementni tanlash:

```
 int Pop ( Queue &Q )
{
  PNode top = Q.Head;
  int x;
  if ( top == NULL ) //Agar ro‘yhat bo‘sh bo‘lsa, …
   return 32767;
  x = top->data;// Birinchi elementni eslab qolamiz
  Q.Head = top->next;
  if ( Q.Head == NULL ) //Agar ro‘yhatda hech nima qolmagan bo‘lsa
    Q.Tail = NULL;
```

```
 delete top;// Xotirani bo‘shatish
 return x;
}
```

Muhokama savollari

1. Ro‘yhat tushunchasi va toifalari
2. Bog‘lama tushunchasi
3. Bog‘lamani tashkil etish
4. Ro‘yhat oxiriga bog‘lama qo‘shish
5. Ro‘yhatdan so‘z qidirish
6. Alifbo-chastotali lug‘atni tashkil etish algoritmi
7. Navbat tushunchasi va tadbiqi

Nazorat savollari

1. Ro‘yhat dinamik tuzilmasi ko‘rinishi.
2. Ro‘yhatlar ustida xarakatlar
3. Bog‘lamani ro‘yhat boshiga qo‘shish
4. Belgilangandan keyin bog‘lamani qo‘shish
5. Belgilangandan avval bog‘lama qo‘shish
6. Bog‘lamani o‘chirish

Misollar

1. n ta elementdan tashkil topgan massiv berilgan. Dastlab massiv elementlariorasidan juftlarini indekslari o‘sish tartibida chiqaruvchi, keyin massiv elementlari orasidan toqlarini indekslari kamayish tartibida chiqaruvchi dastur tuzilsin.

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{int a[100],b,s=0,d,n,k=0;
```

```
cout<<”n=«”cin>>n;
for(int i=0;i<n;i++)
{ cout<<”a[“<<i<<”]=”;
cin>>a[i];
  }
for(int i=0;i<n;i++)
ifa[i]%2==0)
{cout<<a[i]<<”\t”;
k++;
  }
cout<<k<<endl;
  k=0;
for(int i=0;i<n;i++)
if(a[i]%2==1)
{cout<<a[i]<<”\t”;
k++;
    }
cout<<k<<endl;}
```

2. n ta elementdan tashkil topgan massiv berilgan. Massiv elementlari arifmetik progressiyani tashkil qilsa, ayirmaniaks holda nolni chiqaruvchi dastur tuzilsin.

```
 #include <iostream>
 #include <conio.h>
using namespace std;
int main()
{int n, a[10], d;
cout<< “n=”; cin >> n;
for (int i = 0; i < n; i++)
{cout<< “a[“ << i << “]=”;
cin>> a[i];
```

```
   }
   d = a[1] - a[0];
bool arif = true;
for (int i = 1; i < n; i++)
if (a[i] != a[i - 1] + d)
{ arif = false;
break;
   }
if ( arif ) cout << d << endl;
else cout << 0 << endl;
getch();
return 0;
 }
```

3. n ta elementdan tashkil topgan massiv berilgan. Massiv lokal maksimumlari orasidan kichigini chiqaruvchi dastur tuzilsin.

```
#include <iostream>
#include <cstdlib>
#include <conio.h>
using namespace std;
int main()
{  int a[100],n,min=0;
srand(time(NULL));
cout<<”n=”;cin>>n;
for(int i=0;i<n;i++)
a[i]=rand()%20+1;
for(int i=0;i<n;i++)
cout<<”a[“<<i<<”]=”<<a[i]<<”\n”;
cout<<endl;
for(int i=1;i<n-1;i++)
if(a[i-1]<a[i]&&a[i]>a[i+1])
```

```
{ if(min==0)
min=a[i];
if(min<a[i])
min=a[i];
  }
cout<<min<<endl;
getch();
return 0;}
```

4. n ta(100tagacha bo‘lishi mumkin) elementdan tashkil topgan massiv berilgan. Ushbu massivda tashqaridan kiritiladigan x o‘zgaruvchisining uchrash soni aniqlanuvchi dastur tuzilsin.

```
#include <stdio.h>
#include<iostream>
#define MAX 100
using namespace std;
int main()
{int i, x, n, count, continueFlag=0;
int array[MAX];
 /* input n */
do {  printf(«Insert n: «);
scanf(«%d», &n);
 } while (n > MAX);
 /* array initialization */
for (i=0; i<n; i++) {
printf(“Value [%d] = “, i);
scanf(“%d”, &array[i]);
 }
do {
  /* input the search value */
```

```
printf(“Insert x: “);
scanf(“%d”, &x);
  /* count the occurrences */
count = 0;
for (i=0; i<n; i++) {
if (array[i] == x) {
count++;
   }
  }
  /* output the result */
printf(«The value %d appears %d times in the sequence.\n», x, count);
  /* keep going on? */
printf(“Would you like to continue (1=yes, 0=no)? “);
scanf(“%d”, &continueFlag);
 } while (continueFlag != 0);
return 0;
}
```

5. m x n o‘lchamli matritsa berilgan. Matritsaning 2 ga karrali (0, 2, 4, …) satrlarini chiqaruvchi dastur tuzilsin. Shart operatori ishlatilmasin.

```
#include <iostream>
using namespace std;
int main()
{ int m, n, a[10][10];
cout<< “Satrlar sonini kiriting \nm=”;
cin>> m;
cout<< “Ustunlar sonini kiriting \nn=”;
cin>> n;
cout<<”Massiv elementlarini kiriting \n”;
for(int i = 0; i < m ; i++)
```

```
for(int j = 0; j < n; j++)
 {
cout<< “a[“ << i << “][“ << j << “]=”;
cin>> a[i][j];
 }
 // matritsani jadval shaklida chiqarish
cout<< “Matritsaning barcha elementlari” << endl;
for(int i = 0; i < m; i++)
 {
for(int j = 0; j < n; j++)
cout<< a[i][j] << “  ”;
cout<<«\n»;
 }
cout<< «Matritsaning juft indeksli elementlari» << endl;
for(int i = 0; i < m; i += 2)
 {
for(int j = 0; j < n; j++)
cout<< a[i][j] << « «;
cout<<”\n”;
 }
system(“pause”);
return 0;
}
```

6. m x n o‘lchamli matritsa berilgan. Matritsaning har bir satri elementlari yig‘indisini chiqaruvchidastur tuzilsin.

```
#include <iostream>
using namespace std;
int main()
{
```

```
int m, n, yig, a[10][10];
cout<< «Satrlar sonini kiriting \nm=«;
cin>> m;
cout<< «Ustunlar sonini kiriting \nn=«;
cin>> n;
cout<<«Massiv elementlarini kiriting \n»;
for(int i = 0; i < m ; i++)
for(int j = 0; j < n; j++)
cin>> a[i][j];
 // matritsani jadval shaklida chiqarish
for(int i = 0; i < m; i++)
 {
yig = 0;
for(int j = 0; j < n; j++)
  {
cout<< a[i][j] << « «;
yig += a[i][j];
  }
cout<< «\t sum=« << yig << «\n»;
 }
system(«pause»);
return 0;
}
```

7. m x n o‘lchamli matritsa berilgan. Elementlari yig‘indisi eng katta bo‘lsan ustunning, eng kichik elementini chiqaruvchi dastur tuzilsin.

```
#include <iostream>
#include <stdlib.h>
using namespace std;
int main()
```

```
{
int m, n, a[10][10];
cout<< «Satrlar sonini kiriting \nm=«; cin >> m;
cout<< «Ustunlar sonini kiriting \nn=«; cin >> n;
cout<<«Massiv elementlarini kiriting \n»;
for (int i = 0; i < m ; i++)
for (int j = 0; j < n; j++)
cin>> a[i][j];
  // matritsani jadval shaklida chiqarish
cout<< «Kiritilgan matritsa\n»;
for (int i = 0; i < m; i++)
  {
for(int j = 0; j < n; j++)
cout<< a[i][j] << «\t»;
cout<<«\n»;
  }
int ustun = 0, min, max;
for (int j = 0; j < n; j++)
  {
int k = 0;
for (int i = 0; i < m; i++)
    k += a[i][j];
    // boshlang‘ich qiymat sifatida 0 - ustunni olamiz
if (j == 0) max = k;
if (max < k)
    {
max = k;
ustun = j;
    }
  }
```

```
min = a[0][ustun];
for (int i = 0; i < m; i++)
if (min > a[i][ustun])
min = a[i][ustun];
cout<< ustun << « ustun elementlari eng kichikki=« << min << endl;
system(«pause»);
return 0;
}
```

8. m x n o‘lchamli matritsa berilgan. Elementlari har xil bo‘lgan ustunlar sonini aniqlovchi dastur tuzilsin.

```
#include <iostream>
#include <stdlib.h>
using namespace std;
int main()
{
int m, n, a[10][10], jami = 0;
cout<< «Satrlar sonini kiriting \nm=«; cin >> m;
cout<< «Ustunlar sonini kiriting \nn=«; cin >> n;
cout<<«Massiv elementlarini kiriting \n»;
for (int i = 0; i < m ; i++)
for (int j = 0; j < n; j++)
cin>> a[i][j];
  // matritsani jadval shaklida chiqarish
cout<< «Kiritilgan matritsa\n»;
for (int i = 0; i < m; i++)
  {
for(int j = 0; j < n; j++)
    {
cout<< a[i][j] << «\t»;
```

```
    }
cout<< «\n»;
  }
bool b[10] = { 0 }; // massivning hamma elementiga false qiymat berish
for (int j = 0; j < n - 1; j++)
  {
int soni = 1; // j - ustunga o‘xshashlar soni
    // j - ustunga o‘xshashlari oldin sanalgan bo‘lsa, keyingi ustunga o‘tamiz (j++)
if (b[j]) continue;
for (int k = j + 1; k < n; k++)
    {
int same = 0; // same - bir xillar soni
for (int i = 0; i < m; i++)
if (a[i][j] == a[i][k]) same++;
     // bir xillar soni satrlar soniga teng bo‘lsa
if (same == m)
     {
soni++;
       // k - ustunga o‘xshashlar sanaldi
b[k] = true;
     }
    }
    // j - ustunga o‘zidan boshqa o‘xshashlar bo‘lsa jamiga qo‘shamiz
if (soni > 1) jami += soni;
  }
cout<< «O‘xshash ustunlar soni=« << jami << endl;
system(«pause»);
return 0;
}
```

9. m x n o‘lchamli matritsa va k1, k2 butun sonlari berilgan (0 <= k1 < k2 < n). k1 va k2 ustun elementlarini almashtiruvchi dastur tuzilsin.

```
#include <iostream>
#include <stdlib.h>
using namespace std;
void matrix_print(int a[10][10], int m, int n)
{
  // matritsani jadval shaklida chiqarish
for (int i = 0; i < m; i++)
  {
for (int j = 0; j < n; j++)
    {
cout<< a[i][j] << «\t»;
    }
cout<< «\n»;
  }
}
int main()
{
int m, n, a[10][10], k1, k2;
cout<< «Satrlar sonini kiriting \nm=«; cin >> m;
cout<< «Ustunlar sonini kiriting \nn=«; cin >> n;
do {
cout<< «0 <= k1 < k2 < n oraqlida k1 va k2 ni kiriting\n»;
cout<< «k1=«; cin >> k1;
cout<< «k2=«; cin >> k2;
  } while (!(0 <= k1 && k1 < k2 && k2 < n));
cout<<«Massiv elementlarini kiriting \n»;
for (int i = 0; i < m ; i++)
for (int j = 0; j < n; j++)
```

```
cin>> a[i][j];
cout<< «Kiritilgan matritsa\n»;
  matrix_print(a, m, n);
  // ustun elementlarini almashlash
for (int i = 0; i < n; i++)
  {
int t = a[i][k1];
a[i][k1] = a[i][k2];
a[i][k2] = t;
  }
cout<< «Natijaviy matritsa\n»;
  matrix_print(a, m, n);
system(«pause»);
return 0;
}
```

10. Kiritilgan belgining nimaligini aniqlovchi dastur tuzilsin. Agar kiritilgan belgi raqam bo‘lsa - “digit”, lotincha harf bo‘lsa – “lotin” yozuvhi chiqarilsin. Boshqa xolatlar uchun nol chiqarilsin.

```
#include <iostream>
using namespace std;
int main()
{
char c;
cout<< «Ixtiyoriy belgi kiriting» << endl;
cin>> c;
int kod = (int) c;
if (kod < 0) kod += 256;
if (isdigit(c))
cout<< «digit»;
else
```

```
if (isalpha(c))
cout<< «lotin»;
else
if (((kod >= 160) && (kod <= 175)) || // kichik kirill harflari [a, п]
        ((kod >= 224) && (kod <= 239)))  // kichik kirill harflari [p, я]
cout<< «kirill»;
else
cout<< 0;
cout<< endl;
system(«pause»);
return 0;
 }
```

11. Satr berilgan. Satrdagi hamma katta harflarini kichigiga almashtiruvchi dastur tuzilsin.

```
#include <iostream>
#include <ctype.h>
//tolower funksiyasidan foydalanish uchun
//ctype.h sarlavha faylini qo‘shish lozim
using namespace std;
int main()
{
char s[20];
cout<< «Satr kiriting» << endl;
  cin.getline(s, sizeof(s));
for (int i = 0; s[i] !=‘\0'; i++)
  {
int kod = s[i];
s[i] = (char) tolower(kod);
  }
```

```
cout<< s << endl;
system(«pause»);
return 0;
}
```

12. N ta haqiqiy sonlardan iborat bo‘lgan A massivni quyidagicha tekislovchi Smooth2(A, N) protsedurasi tuzilsin: ya’ni A1 element o‘zgartirilmaydi, AK (K=2,..,N) element esa AK-1 va AK boshlang‘ich elementlar yig‘indisining yarimga almashtirilsin. A massiv ham kiriuvchi ham chiquvchi parametr hisoblanadi. Shu protsedura yordamida o‘lchovi Nga teng bo‘lgan A massiv har bir tekislanishini chop qilgan holda 3 marta tekislansin.

```
#include <iostream>
#include <conio.h>
using namespace std;
void Smooth(float a[], int n)
{
float b1, b2;
  b1 = a[0];
for (int i = 1; i< n; i++)
  {
    b2 = a[i];
a[i] = (b1 + b2) / 2;
    b1 = b2;
  }
}
int main()
{
int n, ak;

cout<< «Massiv o‘lchamini kiriting\nn = «;
```

```
cin>> n;
float *a = new float [n];
cout<< «Massivni elementlarini kiriting:» << endl;
for (int i = 0; i < n; i++)
cin>> a[i];
Smooth(a, n);
for (int i = 0; i < n; i++)
cout<< a[i] << «\t»;
cout<< endl;
Smooth(a, n);
for (int i = 0; i < n; i++)
cout<< a[i] << «\t»;
cout<< endl;
delete []a;
getch();
return 0;
}
```

13. O‘lchami MxNga teng bo‘lgan A haqiqiy sonlar matritsasining normasini hisoblovchi haqiqiy toifadagi Norm2(A,M,N) funksiyasi tuzilsin:

Norm2(A,M,N) = max{|A_(i,0)|+|A_(i,1)|+…+|A_(i,N-1)|}, bu yerda barcha ustunlarning maksimumi yig‘indisi olinadi. Berilgan MxN o‘lchamdagi A matritsa uchun Norm2(A,K,N), K=0,…,M – 1 topilsin.

```
#include <iostream>
using namespace std;
float SumRow( float *a[], int m, int n, int k)
 {
if (k >= m)
return 0;
float s = 0;
```

```
for (int j = 0; j < n; j ++)
   s +=a[k][j];
return s;
 }
int main()
 {
int m, n, k;
cout<< «A Matrissani tartibini kiriting (MxN):» << endl;
cout<< «m =«; cin >> m;
cout<< «n =«; cin >> n;
float **a = new float *[n];
for (int i = 0; i < m; i++)
a[i] = new float [n];
cout<< «A Matrissani elementlarini kiriting (MxN):» << endl;
for (int i = 0; i < m; i++)
for (int j = 0; j < n; j++)
cin>> a[i][j];
for (int i = 0; i < m; i++)
   {
for (int j = 0; j < n; j++)
cout<< a[i][j] << « «;
cout<< «\n»;
   }
cout<< «k=«; cin >> k;
cout<< «SumRow = « << SumRow( a, m, n, k) << endl;
for (int i = 0; i < m; i++)
delete []a[i];
delete []a;
system(«pause»);
return 0; }
```

14. S satrdagi barcha katta lotin harflarini kichik harflarga o‘zgartiruvchi LowCaseLatin(S) funksiyasi tuzilsin. (S satrning qolgan belgilari o‘zgartirilmaydi). S satr chiquvchi va kiruvchi parametr hisoblanadi. LowCaseLatin funksiyasidan foydalangan holda berilgan 3 ta satrlaar qayta tashkil qilinsin.

```
#include <iostream>
using namespace std;
string LowCaseLatin (string s);
int main ()
 {
string s;
cout<< «Satr kiriting:\n»;
getline (cin, s);
cout<< «Natija:\n» << LowCaseLatin (s) << endl;
system(«pause»);
return 0;
 }
string LowCaseLatin (string s)
 {
for (int i = 0; i < s.length (); i++)
s[i] = tolower (s[i]);
return s;
 }
```

15. S satrdagi K – so‘zini qaytaruvchi WordK(S, K) funksiyasi tuzilsin. (so‘z deb tarkibida probellar bo‘lmagan, probellar bilan chegaralangan yoki satrning boshi/oxiri bo‘lgan belgilar to‘plamiga aytiladi). Agar satrdagi so‘zlar soni K dan kichik bo‘lsa, u holda funksiya bo‘sh satr qaytarsin. Shu funksiyadan foydalangan holda, berilgan S satrdan K, K2, K3 nomerli so‘zlar ajratilsin.

```
#include <iostream>
using namespace std;
```

```
string WordK (string s, int k);
int main ()
 {
int k;
string s;
cout<< «Matn kiriting:\n»;
getline (cin, s);
cout<< «Butun son kiriting:\nk = «;
cin>> k;
cout<< WordK (s, k) << endl;
system(«pause»);
return 0;
 }
string WordK (string s, int k)
 {
k--; // massiv 0 dan boshlangani uchun
string a[100] = {«««}, x = «««;
int n = 0;
   s = s + « «;
for (int i = 0; i < s.length (); i++)
   {
if (s[i] !=‘‘)
       x = x + s[i];
else
     {
a[n++] = x;
       x = «««;
     }
   }
if (k < n)
```

```
return a[k];
else
return «\n»;
 }
```

16. Fayl nomi va N butun soni berilgan (N > 1). Berilgan nomdagi fayl hosil qilinsin va unga N ta birinchi musbat juft sonlari chop qilinsin (2, 4, …).

```
#include <stdio.h>
using namespace std;
#pragma argsused
int main(int argc, char* argv[])
{
  FILE *f;
int n, s = 0;
char filename[30];
cout<< «Fayl nomini kiriting: « ;
cin.getline(filename, sizeof(filename));
cout<< «n=«; cin >> n;
  // binar faylni o‘qish va yozish uchun ochamiz
  f = fopen(filename, «wb+»);
for(int i = 1; i <= n; i++)
  {
    s = s + 2;
fwrite(&s, sizeof(int), 1, f); // s sonini faylga yozamiz
  }

rewind(f); // fayl ko‘rsatgichibi fayl boshiga qo‘yish
cout<< filename << « fayli ma'lumotlari\n»;
  // fayldan o‘qish
```

```
while (fread(&s, sizeof(int), 1, f))
  {
cout<< s << endl;
  }
fclose(f);
system(«pause»);
return 0;
 }
```

17. Haqiqiy sonlar fayli berilgan. Berilgan fayl elementlarini teskari tartibda saqlovchi yangi fayl hosil qilinsin.

```
#include <vcl.h>
#pragma hdrstop
 #include <iostream>
 #include <stdlib.h>
 #include <stdio.h>
using namespace std;
#pragma argsused
int main(int argc, char* argv[])
{
float n;
  FILE *f, *g;
  f = fopen («haqiqiy.bin», «r»);
if (f == NULL)
  {
cout<< «fayl topilmadi\n»;
system («pause»);
return 0;
  }
```

```
cout<< «Birinchi fayl elementlari\n»;
while (fread (&n, sizeof (n), 1, f))
  {
cout<< n << « «;
  }
cout<< endl;
  g = fopen («teskari.bin», «w+»);
int pos = ftell (f);
while (pos > 0)
  {
pos -= sizeof(n);
    // fayl ko‘rsatgichini fayl boshiga nisbatan joylashtirish
fseek (f, pos, SEEK_SET);
fread (&n, sizeof (n), 1, f); // fayldan o‘qish
fwrite (&n, sizeof (n), 1, g); // faylga yozish
  }
  // fayl ko‘rsatgichini fayl boshiga qo‘yish
rewind(g);
cout<< «Yangi fayl elementlari\n»;
while (fread (&n, sizeof (n), 1, g))
  {
cout<< n << « «;
  }
cout<< endl;
fclose (f);
fclose (g);
system («pause»);
return 0;
 }
```

18. Haqiqiy sonlar fayli berilgan. Boshlang‘ich faylning kamayib boruvchi elementlar ketma-ketliklari uzunligiga ega bo‘lgan yangi butun sonlar fayli hosil qilinsin. Masalan, 1.7, 4.5, 3.4, 2.2 elementlariga ega bo‘lgan boshlang‘ich fayl uchun natijaviy yaratilgan fayl tarkibi quyidagicha bo‘ladi: 3, 2.

```
include<iostream>
 #include <stdlib.h>
 #include <stdio.h>
 #include <ctime>
using namespace std;
int main()
 {
float k, a;
int n = 0;
   FILE *f, *g;
srand (time (NULL));
   f = fopen («haqiqiy.bin», «w+»);
   g = fopen («soni.bin», «w+»);
if (f == NULL ||
g == NULL)
   {
cout<< «Faylni ochishda xato!\n»;
system («pause»);
return 0;
   }
cout<< «Fayl elementlari:\n»;
for (int i = 1; i <= 10; i++)
   {
     k = rand () % 10;
fwrite(&k, sizeof (k), 1, f);
```

```
cout<< k << « «;
  }
cout<< endl;
rewind(f);
fread(&a, sizeof(k), 1, f);
while (!feof(f))
  {
fread (&k, sizeof(k), 1, f);
if (a > k) // agar a == k bo‘lsa kamayuvchi bo‘lmaydi
n++;
else
    {
n++;
if (n != 1) // faqat 1 elementlik kamayish oraliq bo‘lmaydi
fwrite (&n, sizeof(n), 1, g);
      n = 0;
    }
    a = k;
  }
if (n != 0) // oxirgi element saqlanmagan bo‘lsa, saqlansa n = 0 bo‘ladi
fwrite (&n, sizeof(n), 1, g);
rewind(g);
cout<< «Kamayuvchi oraliqlar elementlari soni\n»;
while(fread(&n, sizeof(n), 1, g))
cout<< n << « «;
cout<< endl;
fclose(f);fclose(g);
system («pause»);
return 0;
 }
```

19. N ta talabaning familiyasi, kursi, guruhi, stipendiyalari miqdori berilgan. Stipendiya miqdori katta bo‘lgan talabalar familiyasini chop etish dasturi tuzilsin.

```
#include <conio.h>
void main ()
{
Const N=30; int i; float maxs;
Struct student { char fam [15];
        Int kurs;
        Char grup [3];
        Float stip;
        };
Student stud [N];
Clrscr ();
for (i=0; i<N; i++)
{ printf (“%d- talaba”, i );
  Printf (“\n”familiya:”); scanf (“%s”, &stud [i]. fam);
  Printf (“kurs:”); scanf(“%d”,&stud [i].kurs);
  Printf (“guruh:”); scanf (“%s”, &stud [i].grup);
  Printf (“stipendiya:”); scanf (“%f”, &stud [i]. stip);
}
Maxs=0;
for(i=0; i<N; i++)
If (stud[i.] stip>maxs) maxs=stud[i].stip;
Printf (“\n maksimal stipendiya %f sum.”,maxs);
for(i=0; i<N; i++)
If(stud[i].stip= =maxs) printf(“\n%s”, stud[i].fam);
}
```

20. Klaviaturadan kiritilgan simvolning ikkilik kodini tashkil etuvchi dastur tuzilsin.

```
#include <sydio.h>
```

```
#include <conio.h>
// Bit maydonlarining tuzilmasi
Struct byte {int b1:1;
      Int b2:1;
      Int b3:1;
      Int b4:1;
      Int b5:1;
      Int b6:1;
      Int b7:1;
       Int b8:1;
};
// Birlashma – o‘zgaruvchi
Union bits
  {char ch;
   Byte cod;}  u;
// Dekodirlash fuksiyasi prototipi
  Void decode (bits x);
// Asosiy dastur
  Void main ()
{ do {u.ch=getche ();  //klaviaturadan simvol kiritish
  Printf(“:”);
  Decode (u);
} while (u.ch!=’q’);   // q-simvoli tugallanish belgisi
                    }
// dekodirlash funksiyasi
Void decode (bits.u)
{ if (u.cod.b8) printf(“1”); else printf(“0”);
  if (u.cod.b7) printf(“1”); else printf(“0”);
  if (u.cod.b6) printf(“1”); else printf(“0”);
  if (u.cod.b5) printf(“1”); else printf(“0”);
```

```
  if (u.cod.b4) printf(“1”); else printf(“0”);
  if (u.cod.b3) printf(“1”); else printf(“0”);
  if (u.cod.b2) printf(“1”); else printf(“0”);
  if (u.cod.b1) printf(“1”); else printf(“0”);
printf (“/n”);
}
```

21. Quyidagi maydonlardan tashkil topgan MARSH strukturasi tashkil etilsin:

- marshrut boshlang‘ich punkti nomi;
- marshrut so‘nggi punkti nomi;
- marshrut tartib raqami.

Yozuvlari marshrut nomerlari bo‘yicha tartiblangan, toifasi MARSH bo‘lgan m-ta elementli massivni kiritib, tartib raqami klaviaturadan kiritilgan marshrut haqidagi axborotni chiqaruvchi dastur tuzilsin. Agar bunday marshrut bo‘lmasa, mos yozuv chiqarilsin.

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
struct MARSH{
char pun[20];
char kon[20];
int n;};
void swap_int(int &a,int &b){
int c;
c=a; a=b;b=c;}
void swap_char(char *a,char *b){
int i; char c;
for(i=0;a[i]||b[i];i++){
c=a[i]; a[i]=b[i];b[i]=c;}}
```

```
void sort(MARSH *x,int n){
int i,j;
for(i=0;i<n-1;i++)
for(j=i+1;j<n;j++)
if(x[i].n>x[j].n){
swap_int(x[i].n,x[j].n);
swap_char(x[i].pun,x[j].pun);
swap_char(x[i].kon,x[j].kon);}}
void poisk(MARSH *x,int n,int m){
int i; bool q=true;
for(i=0;i<n;i++)
if(x[i].n==m){ if(q){printf(« %d raqamli poyezd:\n»);q=false;}
printf(«Boshlang‘ich punkt nomi %s, so‘nggi punkt s\n»,x[i].pun,x[i].kon);}}
int main(){
FILE *fp; int i,j,n,m; MARSH *o; bool fl=true;
fp=fopen(«baza.txt»,»r»);
for(i=0;!feof(fp);i++);
o=new MARSH [i];
for(i=0;!feof(fp);i++)
fscanf(fp,»%s%s%d»,&o[i].pun,&o[i].kon,&o[i].n);
fclose(fp);
sort(o,i);
while(fl){fl=false;
cin>>m;
poisk(o,i,m);
cout<<«Qidiruvni davom ettirasizmi?!\n1.Yes 2.No «;
cin>>j; if(j==1) fl=true;}}
```

22. Bo‘sh bo‘lmagan ikkita ikki taraflama ro‘yhatning birinchi, oxirgi va joriy elementlariga ko‘rsatgichlar berilgan. TList toifasini qo‘llagan holda, L2

po‘yhatning (tartibni saqlangan holda) barcha elementlari L1 ro‘yhatning oxiriga yozuvchi AddList(L1, L2) prosedurasi tuzilsin, bunda L2 ro‘yhat bo‘sh holatga keladi. L1 ro‘hatning joriy elementi sifatida qo‘shilganlarning dastlabkisi o‘rnatiladi. Prosedurada xotira ajratish va boshatish vazifalari bajarilmasin. Prosedura yordamida ikkinchi ro‘yhatni birinchi ro‘hatning oxiriga qo‘shish, birlashgan ro‘yhatning birinchi, oxirgi va joriy elementlari adreslarini yozuvga chiqarish amalga oshirilsin.

```
#include<iostream.h>
#include<conio.h>
struct Node{
int Data;
Node* Prev;
Node* Next;};
typedef Node* PNode;
void AddFirst(PNode &HNode,int n){
PNode NNode=new Node;
NNode->Data=n;
NNode->Prev=0;
NNode->Next=0;
HNode=NNode;}
void Add(PNode &HNode,int n){
if(!HNode){AddFirst(HNode,n);return;}
PNode NNode=new Node;
NNode->Data=n;
NNode->Prev=HNode;
NNode->Next=0;
HNode->Next=NNode;
HNode=NNode;}
void AddList(PNode &L1,PNode &L2){
PNode P=L2;
```

```
while(L2->Prev) L2=L2->Prev;
L1->Next=L2;
L2->Prev=L1;
L1=P;
L2=0;}
void OutAtFirst(PNode P){
while(P->Prev)
P=P->Prev;
while(P){
cout<<P->Data<<« «;
P=P->Next;}}
int main(){
PNode L1=0,L2=0;
int n=rand()%10+10;
cout<<«Birinchi ro‘yhat elementlari :\n»;
for(int i=0;i<n;i++){
int j=rand()%100;
Add(L1,j);
cout<<j<<« «;}
cout<<«\n\nIkkinchi ro‘yhat elementlari:\n»;
n=rand()%10+10;
for(int i=0;i<n;i++){
int j=rand()%100;
Add(L2,j);
cout<<j<<« «;}
AddList(L1,L2);
cout<<«\n\nO‘zgartirilgan ro‘yhat:\n»;
OutAtFirst(L1);
while(L1){
PNode buf=L1;
```

```
L1=L1->Prev;
delete buf;}
getch();}
```

23. Tasodifiy elementlardan roʻyhat tashkil etilsin, roʻyhatdan maksimal va minimal elementlar aniqlanib, ushbu elementlar olib tashlangan roʻyhat hosil qilinsin.

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
struct Node{
int Data;
int Index;
Node* Prev;
Node* Next;};
typedef Node* PNode;
void AddFirst(PNode &HNode,int n){
PNode NNode=new Node;
NNode->Data=n;
NNode->Index=1;
NNode->Prev=0;
NNode->Next=0;
HNode=NNode;}
void Add(PNode &HNode,int n){
if(!HNode){AddFirst(HNode,n);return;}
PNode NNode=new Node;
NNode->Data=n;
NNode->Index=HNode->Index+1;
NNode->Prev=HNode;
NNode->Next=0;
```

```
HNode->Next=NNode;
HNode=NNode;}
PNode MAX(PNode HNode){
PNode m=HNode;
while(HNode){
if(m->Data<HNode->Data) m=HNode;
HNode=HNode->Prev; }
return m;}
PNode MIN(PNode HNode){
PNode m=HNode;
while(HNode){
if(m->Data>HNode->Data) m=HNode;
HNode=HNode->Prev; }
return m;}
void Delete(PNode P1,PNode P2){
if(P1->Index<P2->Index){
PNode P0=P1->Next;
for(int i=P1->Index;i<P2->Index;i++){
PNode buf=P0;
P0=P0->Next;
delete buf; }
P1->Next=P2;
P2->Prev=P1; }
else {
PNode P0=P2->Next;
for(int i=P2->Index+1;i<P1->Index;i++){
PNode buf=P0;
P0=P0->Next;
delete buf; }
P2->Next=P1;
```

```
P1->Prev=P2; } }
int main(){
srand(100);
PNode HNode=0,Min=0, Max=0;
int n=rand()%10+5;
cout<<«Ro‘yhat elementlari:\n»;
for(int i=0;i<n;i++){
int j=rand()%100-rand()%100;
Add(HNode,j);
cout<<j<<« «;}
cout<<endl;
Min=MIN(HNode);
Max=MAX(HNode);
cout<<«\nMaksimal element: «<<Max->Data<<endl;
cout<<«\nMinimal element: «<<Min->Data<<endl;
Delete(Min,Max);
cout<<«\nO‘zgartirilgan ro‘yhat: «<<endl;
while(HNode){
PNode buf=HNode;
HNode=HNode->Prev;
cout<<buf->Data<<« «;
delete buf;}
getch();}
```

24. Ro‘yhatdan barcha manfiy elementlar olib tashlanuvchi dastur tuzilsin.

```
#include<iostream.h>
#include<conio.h>
struct Node{
int n;
Node* prew;
```

```
Node* next;};
typedef Node* PNode;
void AddFirst(PNode &HNode,int n){
PNode NNode=new Node;
NNode->n=n;
NNode->prew=0;
NNode->next=0;
HNode=NNode;}
void Add(PNode &HNode,int n){
if(!HNode){
AddFirst(HNode,n);
return;}
PNode NNode=new Node;
NNode->prew=HNode;
HNode->next=NNode;
NNode->next=0;
NNode->n=n;
HNode=NNode;}
void DelNode(PNode &P){
PNode p1=P->prew,p2=P->next,buf=P;
P=P->prew;
if(p1)
p1->next=p2;
if(p2)
p2->prew=p1;
delete buf;}
void DeleteAll(PNode &HNode){
while(HNode){
PNode buf=HNode;
cout<<HNode->n<<« «;
```

```
HNode=HNode->prew;
delete buf;}}
int main(){
int n=rand()%20+10;
PNode HNode=0;
cout<<«Ro‘yhatni tashkil etish:\n»;
for(int i=0;i<n;i++){
int j=rand()%20-rand()%20;
cout<<j<<« «;
Add(HNode,j);}
cout<<«\n\nmanfiy elementlarni olib tashlash:\n»;
PNode P=HNode;
while(P){
if(P->n<0){cout<<P->n<<« «; DelNode(P);}
else
P=P->prew;}
cout<<«\n\nO‘zgartirilgan ro‘yhat:\n»;
DeleteAll(HNode);
getch();
return 0;}
```

25. Soni beshtadan kam bo‘lmagan elementlardan tashkil topgan stek cho‘qqisiga bo‘lgan P1 ko‘rsatgich berilgan. S stekdan birinchi (cho‘qqisidagi) elementni chiqarib olib, uning qiymatini qaytaruvchi va ushbu element egallab turgan xotirani bo‘shatuvchi TStack toifali Pop(S) funksiyasi tuzilsin. Ushbu funksiya orqali stekdagi barcha elementlar chiqarib olinsin va qiymatlari bosmaga chiqarilsin. Shuningdek, stekning yangi cho‘qqisiga bo‘lgan ko‘rsatgich qiymati ham chiqarilsin (agar natijaviy stek bo‘sh bo‘lsa, ko‘rsatgich qiymati NILga teng bo‘ladi).

```
#include<iostream.h>
```

```
#include<conio.h>
struct Stack{
int n;
Stack* prew;};
typedef Stack* PStack;
void Push(PStack &HStack,int n){
PStack NStack=new Stack;
NStack->prew=HStack;
NStack->n=n;
HStack=NStack;};
int Pop(PStack &HStack){
PStack buf=HStack;
int n=HStack->n;
HStack=HStack->prew;
delete buf;
return n;}
int main(){
PStack P1=0;  //Stek cho‘qqisiga ko‘rsatgich
int n=rand()%20+5;
cout<<«Stekka element qo‘shish:\n»;
for(int i=0;i<n;i++){
int j=rand()%90+10;
cout<<j<<« «;
Push(P1,j);}
cout<<«\n\nstek cho‘qqisi adresi: «<<P1<<endl;
cout<<«\n5 elementni chiqarib olish: «;
for(int i=0;i<5;i++){
cout<<Pop(P1)<<« «;}
cout<<«\n\nCho‘qqining yangi adresi : «<<P1<<endl;
cout<<«\nStekning qolgan elementlari:\n»;
```

```
while(P1) //qolgan elementlarni olb tashlash
cout<<Pop(P1)<<« «;
getch();
return 0;}
```

3.8 Bob bo‘yicha xulosalar

Ana endi Siz mohir dasturchi bo‘lishingiz mumkin. Dasturlashtirishda ushbu bobda keltirilgan satrlar, xotiraning taqsimlanishi, dinamik xotira, tuzilmalar, birlashmalar, faylllar, ro‘yhatlar, steklar, navbatlar tushunchalari keng qo‘llanadi. Ushbular yordamida dasturlash jarayonini engillatishimiz mumkin. O‘quvchilarga bo‘limlarda va bob oxirida keltirilgan misollar bayon etilgan nazariy bilimlar mustahkamlanishida yordam beradi deb o‘ylaymiz.

IV BOB. OB’YEKTGA YO‘NALTIRILGAN DASTURLASH ASOSLARI

4.1. Inkapsulyasiya, polimorfizm, vorislik

Ob’yektga yo‘naltirilgan dasturlash (OYD) – bu dasturlashga yangi bir yondashuvdir. Hisoblash texnikasining rivojlanishi va yechilayotgan masalalarni tobora murakkablashuvi dasturlashning turli modellarini (paradigmalarini) yuzaga kelishiga sabab bo‘lmoqda. Birinchi kompilyatorlarda (masalan, FORTRAN tili) dasturlashning funksiyalardan foydalanishga asoslangan prosedura modelini qo‘llab quvvatlagan. Bu model yordamida dastur tuzuvchi bir nechta ming satrli dasturlarni yozishi mumkin edi. Rivojlanishning keyingi bosqichida dasturlarning strukturali modeli paydo bo‘ldi va ALGOL, Pascal va C tillari kompilyatorlarida o‘z aksini topdi. Strukturali dasturlashning mohiyati – dasturni o‘zaro bog‘langan proseduralar (bloklar) va ular qayta ishlaydigan berilganlarning majmuasi deb qarashdan iborat. Ushbu model dastur bloklari keng qo‘llashga, GOTO operatoridan imkon qadar kam foydalanishga tayangan va unda dastur tuzuvchi o‘n ming satrdan ortiq dasturlarni yarata olgan. Yaratilgan dasturni prosedurali modelga nisbatan sozlash va nazorat qilish oson kechgan.

Murakkab masalalarni yechish uchun dasturlashning yangi uslubiga zarurat paydo bo‘ldiki, u OYD modelida amalga oshirildi. OYD modeli bir nechta tayanch konsepsiyalarga asoslanadi.

Berilganlarni abstraksiyalash – berilganlarni yangi turini yaratish imkoniyati bo‘lib, bu turlar bilan xuddi berilganlarning tayanch turlari bilan ishlagandek ishlash mumkin. Odatda yangi turlarni berilganlarning abstrakt turi deyiladi, garchi ularni soddaroq qilib “foydalanuvchi tomonidan aniqlangan tur” deb atash mumkin.

Inkapsulyasiya – bu berilganlar va ularni qayta ishlovchi kodni birlashtirish mexanizmidir. Inkapsulyasiya berilganlar va kodni tashqi ta’sirdan saqlash imkonini beradi.

Yuqoridagi ikkita konsepsiyani amalga oshirish uchun C++ tilida *sinflar* ishlatiladi. *Sinf* termini bilan Ob'yektlar turi aniqlanadi. Sinfning har bir vakili (nusxasi) *Ob'yekt* deb nomlanadi. Har bir Ob'yekt o'zining alohida holatiga ega bo'ladi. Ob'yekt holati uning *berilganlar-a'zolarning* ayni paytdagi qiymati bilan aniqlanadi. Sinf vazifasi uning *funksiya-a'zolarining* sinf ob'yektlari ustida bajaradigan amallar imkoniyati bilan aniqlanadi.

Berilgan sinf ob'yektini yaratish *konstruktor* deb nomlanuvchi maxsus funksiya-a'zo tomonidan, o'chirish esa destruktor deb nomlanuvchi maxsus funksiya–a'zo orqali amalga oshiriladi.

Sinf ichki berilganlarini murojaatni cheklab qo'yishi mumkin. Cheklov berilganlarni ochiq (public), yopiq (private) va himoyalangan (protected) deb aniqlash bilan tayinlanadi.

Sinf, shu turdagi ob'yektning tashqi dunyo bilan o'zaro bog'lanishi uchun qat'iy muloqot shartlarini aniqlaydi. Yopiq berilganlarga yoki kodga faqat shu ob'yekt ichida murojaat qilish mumkin. Boshqa tomondan, ochiq berilganlarga va kodlarga, garchi ular ob'yekt ichida aniqlangan bo'lsa ham, dasturning ixtiyoriy joyidan murojaat qilish mumkin va ular ob'yektni tashqi olam bilan muloqotni yaratishga xizmat qiladi. Yaratilgan ob'yektlarni, ularni funksiya–a'zolariga oddiygina murojaat orqali amalga oshiriluvchi *xabarlar* (yoki *so'rovlar*) yordamida boshqarish mumkin. Keyinchalik Windows xabarlari bilan adashtirmaslik uchun so'rov termini ishlatiladi.

Vorislik – bu shunday jarayonki, unda bir ob'yekt boshqasining xossalarini o'zlashtirishi mumkin bo'ladi. Vorislik orqali mavjud sinflar asosida hosilaviy sinflarni qurish mumkin bo'ladi. Hosilaviy sinf (*sinf–avlod*) o'zining ona sinfidan (*sinf-ajdod*) berilganlar va funksiyalarni vorislik bo'yicha oladi, hamda ular satriga faqat o'ziga xos bo'lgan qirralarni amalga oshirishga imkon beruvchi berilgan va funksiyalarni qo'shadi. Ajdod sinfdagi himoyalangan berilgan-a'zolarga va funksiya-a'zolarga ajdod sinfda murojaat qilish mumkin bo'ladi. Bundan tashqari, hosilaviy sinfda ona sinf funksiyalari qayta aniqlanishi mumkin. Demak, vorislik asosida bir-biri bilan "ona-bola" munosabatidagi sinflar shajarasini yaratish

mumkin. *Tayanch sinf* termini sinflar shajarasidagi ona sinf sinonimi sifatida ishlatiladi. Agar ob'yekt o'z atributlarini (berilganlar-a'zolar va funksiyalar–a'zolar) faqat bitta ona sinfdan vorislik bilan olsa, *yakka (*yoki *oddiy) vorislik* deyiladi. Agar ob'yekt o'z atributlarini bir nechta ona sinflardan olsa, *to'plamli vorislik* deyiladi.

Polimorfizm – bu kodning, bajarilish paytidan yuzaga keladigan holatga bog'liq ravishda o'zini turlicha amal qilish xususiyatidir. Polimorfizm – bu faqat ob'yektlar xususiyati bo'lmasdan, balki funksiyalar-a'zolar xususiyatidir va ular xususan, bitta nomdagi funksiya-a'zoni, har xil turdagi argumentlarga ega va bajaridagan amali unga uzatiladigan argumentlar turiga bog'liq bo'lgan funksiyalar uchun (o'rnida) foydalanish imkoniyatida namoyon bo'ladi. Bu holatga *funksiyalarni qayta yuklash* deyiladi. Polimorfizm amallarga ham qo'llanishi mumkin, ya'ni amal mazmuni (natijasi) operand (berilgan) turiga bog'liq bo'ladi. Polimorfizmning bunday turiga *amallarni qayta yuklash* deyiladi.

Polimorfizmning yana bir ta'rifi quyidagicha: polimorfizm – bu tayanch sinfga ko'rsatgichlarning (murojaatlarning), ularni virtual funksiyalarni chaqirishdagi turli shakl (qiymatlarni) qabul qilish imkoniyatidir. C++ tilining bunday imkoniyati *kechiktirilgan bog'lanish* natijasidir. Kechiktirilgan bog'lanishda chaqiriladigan funksiya-a'zolar adreslari dastur bajarilishi jarayonida dinamik ravishda aniqlanadi. An'anaviy dasturlash tillarida esa bu adreslar statik bo'lib, ular kompilyasiya paytida aniqlanadi (*oldindan bog'lanish*). Kechiktirilgan bog'lanish faqat virtual funksiyalar uchun o'rinli.

C++ tili ob'yektga yo'naltirilgan dasturlash prinsiplarini qo'llab quvvatlaydi. Bu prinsiplar quyidagilardan iborat:

1. Inkapsulyasiya
2. Vorislik
3. Polimorfizm

Inkapsulyasiya. Agar muxandis ishlab chiqarishda diod, triod yoki rezistorni ishlatsa, u bu elementlarni yangitdan ihtiro qilmaydi, balki do'kondan sotib oladi [9]. Demak, muxandis ularning qanday tuzilganligiga e'tiborini qaratmaydi, bu

elementlar yaxshi ishlasa yetarli. Aynan shu tashqi konstruksiyada ishlaydigan yashirinlik yoki avtonomlik xossasi inkapsulyasiya deyiladi.

Vorislik. Yangi ob'yekt yaratilayotgan bo'lsa, ikkita variantdan biri tanlanadi: mutlaqo yangisini yaratish yoki mavjud modelning konstruksiyasini takomillashtirishdir. Ko'pincha 2-variant tanlanadi, demak, ba'zi xususiyatlari o'zgartiriladi xolos. Bu narsa vorislik prinsipiga asos soladi. Yangi sinf oldin mavjud bo'lgan sinfni kengaytirishdan hosil bo'ladi. Bunda yangi sinf oldingi sinfning merosxo'ri deb ataladi.

Polimorfizm. Poli – ko'p, morfe – shakl degan ma'noni bildiradi. C++ tili bir xil nomdagi funksiya turli ob'yektlar tomonidan ishlatilganda turli amallarni bajarish imkoniyatini ta'minlaydi. Polimorfizm – shaklning ko'p xilligidir.

Dasturda ishlatiladigan har bir o'zgaruvchi o'z toifasiga ega va u quyidagilarni aniqlaydi:

1. Xotiradagi o'lchovini;
2. Unda saqlanayotgan ma'lumotlarni;
3. Uning yordamida bajarilishi mumkin bulgan amallarni.

C++ tilida dasturchi o'ziga kerakli ixtiyoriy toifani hosil qilishi mumkin. Bu yangi toifa ichki toifalarning xossalari va ularning funksional imkoniyatlarini o'zida ifodalaydi. Yangi toifa sinfni e'lon qilish orqali tuziladi. Sinf bu – bir-biri bilan funksional bog'angan o'zgaruvchilar va usullar (funksiyalar) to'plamidir.

Masalan: Mushuk nomli sinf tuzmoqchimiz. Bu yerda uning yoshi, og'irligi kabi o'zgaruvchilar va miyovlash, sichqon tutish kabi funksiyalardan ishdatiladi. Yoki Mashina sinfi g'ildirak, eshik, o'rindiq, oyna kabi o'zgaruvchilar va xaydash, to'xtatish kabi funksiyalardan iborat.

Sinfdagi o'zgaruvchilar – sinf a'zolari yoki sinf xossalari deyiladi.

Sinfdagi funksiyalar odatda o'zgaruvchilar ustida biror bir amal bajaradi. Ularni sinf usullari (metodlari) deb ham ataladi.

Sinfni e'lon qilish uchun class so'zi , { } ichida esa shu sinfning a'zolari va usullari keltiriladi. Masalan:

class non

```
{ int ogirlik ;
  int baho ;
  void yasash ( );
  void yopish ( );
  void eyish ( );
}
```

Sinfni e’lon qilishda xotira ajratilmaydi. Sinf e’lon qilinganda kompilyator faqat shunday (non) sinf borligini, hamda unda qanday qiymatlar (ogirlik, baho) saqlanishi mumkinligini, ular yordamida qanday amallarni (yasash, yopish, yeyish) bajarish mumkinligi haqida xabar beradi. Bu sinf ob’yekti hammasi bo‘lib 4 bayt joy egallaydi (2 ta int).

Ob’yekt sinfning biror bir nusxasi hisoblanadi.

C++ tilida toifalarga qiymat o‘zlashtirilmaydi, balki o‘zgaruvchiga o‘zlashtiriladi. Shuning uchun to‘g‘ridan-to‘g‘ri int = 55 deb yozib bo‘lmaganidek non.baho=1200 deb ham bo‘lmaydi. O‘zlashtirishda xatolikka yo‘l qo‘ymaslik uchun oldin non sinfiga tegishli patir ob’yektini hosil qilamiz keyin esa unga kerakli qiymatlarni beramiz.

Masalan:

```
int a;     // butun toifali a o‘zgaruvchisi, ob’yekti
non patir; //
```

Endi non sinfining real ob’yekti aniqlanganidan so‘ng uning a’zolariga murojaat

```
patir.baho = 1200;
patir.ogirlik = 500;
patir.yasash ( ) ;
```

Sinfni e’lon qilishda quyidagilardan foydalaniladi:

public - ochiq

private – yopiq

Sinfning barcha usul va a’zolari boshlang‘ich holda avtomatik ravishda yopiq bo‘ladi. Yopiq a’zolarga esa faqat shu sinfning usullari orqaligina murojaat qilish mumkin. Ob’yektning ochiq a’zolariga esa dasturdagi barcha funksiyalar murojaat qilishi mumkin. Lekin sinf a’zolariga murojaat qilish ancha mushkul ish hisoblanadi. Agar to‘g‘ridan to‘g‘ri:

non patir;

patir.baho = 1200;

patir.og‘irlik = 500; deb yozsak xato bo‘ladi.

A’zolarga murojaat qilishdan oldin uni ochiq deb e’lon qilish kerak:

```
#include < iostream.h >
class non
{ public :
  int baho;
  int ogirlik;
  void yasash ( );  };
int main ( )
{ non patir;
patir.baho = 1200;  patir.ogirlik = 500;
cout <<“men olgan patir” <<patir.baho <<“so‘m”<<endl;
cout <<“uning og‘irligi =”<<patir.og‘irlik <<endl ;   }
```

Muhokama savollari

1. Obyektga yo‘naltirilgan dasturlash tamoyillari.
2. Sinflar va ob’yektlar
3. Joylashtiriladigan (inline) funksiyalar–a’zolar
4. Inkapsulyasiya tushunchasi
5. Vorislik tushunchasi
6. Polimorfizm tushunchasi

Nazorat savollari

1. Inkapsulyasiya nima?
2. Polimorfizm haqida tushuncha.
3. Vorislikning qo‘llanishi.
4. Sinfdagi o‘zgaruvchilar – sinf a’zolarining qo‘llanishi.

4.2 Sinflar va ob’yektlar

Sinf tushunchasi C++ tilidagi eng muhim tushunchalardan biridir. Sinf sintaksisi struktura sintaksisiga o‘xshashdir va uning ko‘rinishi quyidagicha:

```
class <sinf nomi>
 {
 // sinfning yopiq berilganlar–a’zolari va funksiyalar –
 // a’zolari
  public:
 // sinfning ochiq berilganlar–a’zolari va funksiyalar –
// a’zolari
}
<ob’yektlar ro‘yxati>
```

Odatda sinf tavsifida <ob’yektlar ro‘yxati> qismi shart emas. Sinf ob’yektlari keyinchalik, zarurat bo‘yicha e’lon qilinishi mumkin. Garchi <sinf nomi> qismi ham majburiy bo‘lmasa ham, uning bo‘lgani ma’qul. Chunki <sinf nomi> berilganlarning turining yangi nomi bo‘lib, uning yordamida shu sinf ob’yektlari aniqlanadi.

Sinf ichida e’lon qilingan funksiya va berilganlar shu sinf a’zolari hisobalandi. Sinf e’lonining ichida e’lon qilingan o‘zgaruvchilar *berilganlar-a’zolar*, sinf ichida e’lon qilingan funksiyalar *funksiyalar–a’zolar* deyiladi. Kelishuv bo‘yicha sinf ichidagi barcha funksiya va o‘zgaruvchilar shu sinf uchun yopiq hisoblanadi, ya’ni ularni faqat shu sinf a’zolari ishlatishi mumkin. Sinfning

ochiq a’zolarini e’lon qilish uchun public kalit so‘zi va “:” belgisidan foydalaniladi. Sinf e’lonidagi public so‘zidan keyin e’lon qilingan funksiyalar va o‘zgaruvchilarga sinfning boshqa a’zolari va dasturning shu sinf ishlatilgan ixtiyoriy joyidan murojaat qilish mumkin bo‘ladi.

Sinf e’loniga misol:

```
class Sinf_1
{
// sinfning yopiq elementi
int a;
public:
int get_a();
void set_a(int_num);
}
```

Garchi int get_a()va void set_a(int_num) funksiyalari Sinf_1 sinf ichida e’lon qilingan bo‘lsa ham, ular hali aniqlangani yo‘q. Funksiyani aniqlash uchun sinf nomi va “::” belgilarini yozish orqali amalga oshiriladi. Bu yerda “::” – *ko‘rish sohasini kengaytirish amali(yoki taaluqlilik amali)* deyiladi. Funksiya-a’zoni aniqlashning umumiy shakli quyidagicha:

```
<tur><sinf nomi>::<funksiya nomi> (<parametrlar ro‘yxati>)
{
 // funksiya tanasi
}
```

Yuqorida e’lon qilingan Sinf_1 sinfning int get_a() va void set_a(int_num) funksiya-a’zolarini aniqlashga misol keltirilgan:

```
int Sinf_1 :: get_a()
{ return a;}
void Sinf_1:: set_a(int num) {a=num;}
```

Sinf_1 sinfini e’lon qilish shu sinf turidagi ob’yektlarini yuzaga keltirmaydi. Sinf ob’yektlarini yuzaga keltirish uchun sinf nomini berilganlar turi spesifikatori sifatida ishlatish zarur bo‘ladi. Masalan,

Sinf_1 obj1,obj2;

Sinf ob’yekti yaratilgandan keyin “.” yordamida sinfning ochiq a’zolariga murojaat qilish mumkin bo‘ladi. Misol uchun

obj1.set_a(20);

obj2.set_a(50);

murojaatlar orqali obj1 va obj2 ob’yektlarning *a* o‘zgaruvchilariga qiymatlar beriladi. Har bir ob’yekt sinfda e’lon qilingan o‘zgaruvchilar o‘z nusxalariga ega bo‘ladi. Shu sababli, obj1 ob’yektidagi *a* o‘zgaruvchi obj2 ob’yektdagi *a* o‘zgaruvchidan farq qiladi.

Sinf elementlariga ko‘rsatgichlar orqali ham amalga oshirish mumkin. Quyidagi misol buni namoyon etadi.

```
class Nuqta
{
 public:
int x,y;
void Koord_Qiymat_Berish( int _x, int _y);
};
voidNuqta::Koord_Qiymat_Berish(int _x, int _y)
{
 x= _x; y=_y;
}
int main()
{
Nuqta x0y;
Nuqta * Koord_kursatgich= & x0y;
 //…
 x0y.x=0;
 Koord_kursatgich –> y=0;
 Koord_Kursatgich –> Koord_Qiymat_Berish(10,15);
 //…
```

return 0;

}

Shuni qayd qilish kerakki, sinf ob’yektlariga “.” va ”–>” orqali murojaat qilishning kompilyasiya nuqtai-nazaridan hech bir farqi yo‘q. Kompilyator “.” bilan murojaatni ”–>” almashtiradi. Masalan,

obj1.set_a(20);

ko‘rsatmasi kompilyator tomonidan

(&obj1)–> set_a(20);

ko‘rinishidagi ko‘rsatma bilan almashtiriladi.

C++ tili sinfning berilganlar-a’zolariga ma’lum bir cheklanishlar qo‘yadi:

– berilganlar–a’zolar auto, extern yoki register modifikatorlari bilan aniqlanishi mumkin emas;

– sinfning berilganlar–a’zolari shu sinf turidagi ob’yekt bo‘lishi mumkin emas, lekin ular shu sinfga ko‘rsatgich yoki murojaat(&) bo‘lishi, boshqa sinf ob’yekti bo‘lishi mumkin.

Sinf [9], uning a’zolari ishlatilishidan oldin e’lon qilingan bo‘lishi kerak. Biroq, ayrim hollarda sinfda hali e’lon qilinmagan sinfga ko‘rsatgich yoki murojaat (&) e’lon qilishga zarurat bo‘lishi mumkin. Bu holda sinfning to‘liq bo‘lmagan e’lonidan foydalanishga to‘g‘ri keladi. Sinfning to‘liq bo‘lmagan e’loni quyidagi ko‘rinishga ega:

class <sinf nomi>;

Misol ko‘raylik.

```
class Sinf2; // sinfning to‘liqmas e’loni
class Sinf1
{
int x;
 Sinf2 * sinf2; // sinf2 sinfiga ko‘rsatgich
public:
Sinf1(int _x) {x=_x;}
};
```

```
int main()
{
 //…
return 0;
}
class Sinf2 // Sinf2 sinfining to‘liq e’loni
{
int a;
public:
Sinf2();
};
```

Shuni qayd etish kerakki, sinf e’loni struktura e’loniga o‘xshash, farqli ravishda:

– sinf e’lonida public, protected yoki private murojaat modifikatorlari ishlatiladi;

– struct kalit so‘zi o‘rnida class yoki union kalit so‘zlari ishlatilishi mumkin;

– odatda sinf tarkibida berilganlardan tashqari funksiya-a’zolari kiradi;

– sinf konstruktor yoki destruktor deb nomlanuvchi maxsus funksiya-a’zolariga ega bo‘ladi.

Quyida struct va union kalit so‘zlari bilan aniqlangan sinflarga misol keltirilgan.

```
struct Nuqta
{
private:
int x; int y;
public:
int Olish_X();
int Olish_Y();
void Qiymat_Berish_X(int _x);
```

```
void Qiymat_Berish_Y(int _y);
};
union Bit
{
Bit(unsigned int n);
void Bit_Chop_Qilish();
unsigned int num;
unsigned char c [sizeof(unsigned int)];
};
```

Murojaat spesifikatori, undan keyin joylashgan barcha sinf elementlariga qo‘llaniladi, toki boshqa spesifikator uchramaguncha yoki sinf e’loni tugamaguncha.

Quyida spesifikatorlar tavsifi keltirilgan.

Sinfga murojaat spesifikatorlari:

private - berilganlar-a’zolarga va funksiyalar-a’zolarga faqat shu sinf funksiyalar-a’zolari murojaat qilishi (ishlatishi) mumkin;

protected-berilganlar-a’zolarga va funksiyalar-a’zolarga faqat shu sinf va shu sinfdan hosil bo‘lgan sinflar funksiyalar-a’zolariga murojaat qilishi (ishlatishi) mumkin;

public - berilganlar-a’zolariga va funksiyalar-a’zolariga faqat shu sinf funksiya-a’zolari va sinf ob’yekti mavjud bo‘lgan dastur funksiyalari murojaat qilishi (ishlatishi) mumkin;

C++ tilida struktura va birlashmalar sinf turlari deb qaraladi. Struktura va sinflar bir-biriga o‘xshash, faqat kelishuv bo‘yicha murojaat bilan farq qiladi: strukturada kelishuv bo‘yicha barcha elementlar public murojaatiga ega bo‘lsa, sinfda ular private murojaatida bo‘ladi. Birlashmada ham, xuddi strukturadek kelishuv bo‘yicha elementlar public murojaatda bo‘ladi.

Struktura va birlashmalar o‘z konstruktor va destruktorlariga ega bo‘lishi mumkin. Shu bilan birgalikda birlashmalarni sinf sifatida ishlatishga ma’lum bir cheklovlar mavjud. Birinchidan, ular birorta sinf vorisi bo‘lishi mumkin emas,

o‘zlari ham boshqa sinflar uchun ona sinf bo‘la olmaydi. Ular static atributli a’zolarga, hamda konstruktor va destruktorli ob’yektlarga ega bo‘lishi mumkin emas.

Birlashmani sinf sifatida ishlatilishiga misol ko‘raylik.

```
#include <iostream.h>
union Bit
{
Bit(unsigned int n);
void Bit_Chop_Qilish();
unsigned int num;
unsigned char c[sizeof(unsigned int)];
};
Bit::Bit(unsigned int n){num=n;};
void Bit::Bit_Chop_Qilish( )
{
int i,j;
for (j=sizeof(unsigned int)-1; j>=0; j--)
 {
 cout<<“Baytning ikkilik ko‘rinishi”<<j<<’:’;
for (i=128; i; i>>=1)
 {
if (i & c[j]) cout<<’1’;
else cout<<’0’;
}
cout<<endl; }
}

int main( )
 {
 Bit bit(2000);
```

```
 Bit.Bit_Chop_Qilish();
return 0;
 }
```

Bu misolda ob’yektga berilgan ishorasiz butun qiymatning ikkilik ko‘rinishi chop etiladi.

Muhokama savollari

1. Sinflar va ob’yektlar
2. Berilganlar-a’zolar
3. Funksiyalar–a’zolar
4. Ko‘rish sohasini kengaytirish amali
5. Sinfning yopiq berilganlari
6. Sinflarni tashkil etish, ob’yektlar.
7. Sinf usullari va amallari.
8. Sinflarni e’lon qilish, ochiq va yopiq sinflar.

Nazorat savollari

1. Inkapsulyasiya nima?
2. Polimorfizm haqida tushuncha.
3. Vorislikning qo‘llanishi.
4. Sinfdagi o‘zgaruvchilar – sinf a’zolarining qo‘llanishi.
5. Sinfning ochiq berilganlari.
6. Murojaat spetsifikatorlari.

4.3 Konstruktorlar va destruktorlar

Ob’yektni yaratishda uni inisializasiyalash kerak. Bu maqsadda C++ tilida konstruktor deb nomlanuvchi maxsus funksiya-a’zo aniqlangan. Sinf konstruktori har safar sinf ob’yekti yaratilishi paytida chaqiriladi. Konstruktor nomi o‘zi a’zo

bo‘lgan sinf nomi bilan ustma–ust tushadi va qaytaruvchi qiymatga ega bo‘lmaydi. Masalan,

```
#include <iostream.h>
class Sinf
{
 int var;
 public:
 Sinf(); // Konstruktor
 void Chop_etish_var();
 };
Sinf::Sinf()
{
 cout<< “Konstruktor ishladi \n”;
 var=0;
}
Sinf::Chop_etish_var(){cout<<var;}
int main()
{
 Sinf ob;
ob.Chop_Etish_var();
//…
 return 0;
}
```

Bu misolda Sinf konstruktori ekranga xabar chiqaradi va yopiq var o‘zgaruvchini inisializasiyalaydi.

Shuni qayd etish kerakki, dastur tuzuvchi konstruktor chaqiradigan kod yozmasligi kerak. Barcha zarur ishni kompilyator amalga oshiradi. Yuqorida qayd qilingandek konstruktor u tegishli sinf ob’yekti yaratilayotgan paytda chaqiriladi. O‘z navbatida ob’yekt bu ob’yektni e’lon qiluvchi operator bajarilishida yaratiladi.

Shuning uchun ham C++ tilida o‘zgaruvchini e’lon qiluvchi operator bajariluvchi operator hisoblanadi.

Global ob’yektlar uchun konstruktor dastur bajarilishi boshlanganda chaqiriladi. Lokal ob’yektlar uchun konstruktor o‘zgaruvchi e’lonining har bir bajarilishida chaqiriladi.

Konstruktorga nisbatan teskari amal bajaradigan funksiya-a’zolariga destruktorlar deyiladi. Bu funksiya-a’zo ob’yekt o‘chirilishida chaqiriladi. Odatda destruktor ob’yekt tomonidan egallangan xotirani bo‘shatish uchun xizmat qiladi. Uning nomi sinf nomi bilan mos tushadi, faqat oldiga “~” belgisi qo‘yiladi.

Quyida destruktor aniqlangan sinfga misol keltirilgan.

```
#include <iostream.h>
class Sinf;
{
int var;
 public:
 Sinf(); // Konstruktor
 ~Sinf(); // Destruktor
 void Chop_etish_var();
 Sinf::Sinf()
 {
 cout<< “Konstruktor ishladi \n”;
  var=0;
}
 Sinf::~Sinf( )
 {
 cout<< “Destruktor ishladi \n”; }
void Sinf::Chop_etish_var( )
{
 cout<<var<<endl;
}
```

```
int main()
{
 Sinf ob;
ob.Chop_Etish_var();
//…
 return 0;
}
```

Destruktor ob’yekt o‘chirilishida chaqiriladi. Global ob’yektlar dastur tugashida o‘chiriladi. Lokal ob’yektlar – ularni ko‘rish sohasidan chiqishda o‘chiriladi.

Shuni alohida qayd etish kerakki, konstruktor va destruktorlarga ko‘rsatgichlar hosil qilish mumkin emas.

Agar sinf o‘zgaruvchilarini inisializasiya qilish zarur bo‘lsa, parametrli konstruktor ishlatildi. Yuqorida keltirilgan misolga o‘zgartirish kiritamiz.

```
#include <iostream.h>
class Sinf
{
 int a, b;
 public:
 Sinf(int x, int y);     // Konstruktor
 ~Sinf();                          // Destruktor
 void Chop_etish_var();
}
Sinf::Sinf(int x, int y)
{
 cout<< “Konstruktor ishladi \n”;
a=x; b=y;
}
Sinf::~Sinf()
{
```

```
 cout<< “Destruktor ishladi \n”;
}
void Sinf::Chop_etish_var()
{
 cout<<a<<b<<endl;
}
int main()
{
Sinf ob (5,10);
ob.Chop_Etish_var();
//…
 return 0;
}
```

Bu yerda ob ob’yekti e’lonida konstruktorga uzatilgan qiymatlar sinf tarkibidagi a va b yopiq o‘zgaruvchilarni inisializasiya qilishda ishlatiladi.

Parametrli konstruktorga qiymat uzatish sintaksisi quyidagi ifodaning qisqargan shakli hisoblanadi:

“Sinf ob(5,10);”

ifodasi

“Sinf(5,10);”

ifodasi bilan ekvivalent. Aksariyat hollarda qisqartirilgan shakl ishlatiladi.

Konstruktordan farqli ravishda destruktor parametrga ega bo‘lishi mumkin emas, chunki o‘chirilayotgan ob’yekt o‘zgaruvchilariga qiymat berish ma’noga ega emas.

Konstruktor uchun aniqlangan bir nechta qoidalarni keltiramiz:

- konstruktor uchun qaytariluvchi qiymat turi ko‘rsatilmaydi;
- konstruktor qiymat qaytarmaydi;
- konstruktor vorislik bilan o‘tmaydi;
- konstruktor const, volatile, static yoki virtual modifikatorlari bilan e’lon qilinmaydi.

Agar sinf aniqlanishida konstruktor e’lon qilinmasa, kompilyator o‘zi kelishuv bo‘yicha parametrsiz konstruktorni hosil qiladi.

Destruktor uchun quyidagi qoidalar aniqlangan:

– destruktor parametrlarga ega bo‘lishi mumkin emas;

– destruktor qiymat qaytarmaydi;

– destruktor vorislik bilan o‘tmaydi;

– sinf bittadan ortiq destruktorga ega bo‘lishi mumkin emas;

– destruktor const, volatile, static yoki virtual modifikatorlari bilan e’lon qilinmaydi.

Agar sinfda destruktor e’lon qilinmasa, kompilyator o‘zi kelishuv bo‘yicha konstruktorni hosil qiladi.

Odatda sinf berilganlari-a’zolari konstruktor tanasida inisializasiyalanadi. Lekin inisializasiyaning boshqa usul bilan – *elementlarni inisializasiyalash ro‘yxati* orqali amalga oshirish mumkin. Elementlarni inisializasiyalash ro‘yxati funksiya sarlavhasi aniqlanishidan keyin ikki nuqta (“:”)joylashadi va unda vergul bilan ajratilgan holda berilganlar-a’zolar va tayanch sinflar yoziladi. Har bir element uchun qavs ichida inisializasiyada ishlatiladigan bir yoki bir nechta parametrlar ko‘rsatiladi. Quyidagi misolda elementlarni inisializasiyalash ro‘yxati orqali sinf o‘zgaruvchilarini inisializasiya qilish ko‘rsatilgan.

```
class Sinf
{
int a, b;
 public:
 Sinf(int x, int y); // Konstruktor
};
Sinf:: Sinf( int x, int y): a(x), b(y);
  {
   cout<< “Konstruktor ishladi \n”;
  }
//…
```

Keltirilgan dastur bo‘lagida konstruktor bajaradigan ish oldingi misoldagi konstruktor ishi bilan ekvivalent.

Albatta, sinf elementlarini inisializasiya qilishning qaysi shaklini qo‘llash dastur tuzuvchiga bog‘liq. Biroq, shunday holatlar bo‘ladiki, unda elementlarni inisializasiyalash ro‘yxatidan foydalanmaslikning iloji yo‘q: sinfning berilganlar–konstantalariga va ko‘rsatgichlariga boshlang‘ich qiymat berishda; sinf a’zosi ob’yekt bo‘lganda va bu ob’yekt konstruktori bir yoki bir nechta parametrlarga qiymat berishni talab qilgan hollarda.

Kelishuv bo‘yicha konstruktorlarni ishlatishda kolliziyadan (bir narsani ikki xil tushunishdan) qochish kerak, ya’ni sinfda bir nechta konstruktor bo‘lganda kompilyator qachon ularning qaysi biri chaqirilishini aniq bilishi kerak. Quyidagi misolda bu holat bilan bog‘liq xato ko‘rsatilgan.

```
class S
{
 public:
  S();            // Kelishuv bo‘yicha konstruktor
 S(int i=0);   // Kelishuv bo‘yicha konstruktor o‘rnida
//ishlatilishi mumkin bo‘lgan konstruktor
};
int main()
{
 S ob1(10);  // S::S(int) konstruktori ishlatiladi.
S ob2;                  //Noto‘g‘ri. S::S(int) yoki S::S()
// konstruktorlarining qaysi biri
// chaqirilishi noma’lum.
```

Bu ziddiyatni hal qilish yo‘li – bu sinf e’lonidan kelishuv bo‘yicha konstruktorni o‘chirishdir.

Muhokama savollari

1. Konstruktorlar va destruktorlar

2. Parametrli konstruktor
3. Konstruktor uchun aniqlangan qoidalar

Nazorat savollari

1. Kostruktorlik va destruktorlik?
2. Destruktor uchun qoidalar
3. Elementlarni inisializasiyalash ro‘yxati
4. Kelishuv bo‘yicha konstruktorlar

4.4 Nusxalash konstruktori

Nusxalash konstruktori sinf ob’yektini yaratadi va shu sinfning mavjud ob’yektlaridan berilganlarni (ularning qiymatini) nusxasini oladi. Shu sababli u sinf ob’yektiga konstanta ko‘rsatgichi (const S&) yoki oddiygina ko‘rsatgich (S&) bo‘lgan yagona parametrga ega bo‘ladi. Parametrlarning birinchisini ishlatish ma’qul hisoblanadi, u konstanta ob’yektlarni nusxalash imkonini beradi [5].

Quyidagi misolda nusxalash konstruktorini ishlatish ko‘rsatilgan.

```
classNuqta
{
int x,y;
public:
Nuqta(const Nuqta& koor);
Nuqta(int x0, int y0);
void Abstissa();
void Ordinata();
void Uzgartirish (int delta_x, int delta_y);
};
Nuqta::Nuqta(const Nuqta& koor)
{
```

```
 x = koor.x; y = koor.y;
};
Nuqta::Nuqta(int x0, int y0){x = x0; y = y0;};
void Nuqta::Abstissa(){cout<<«x=«<<x<<endl;};
void Nuqta::Ordinata(){cout<<«y=«<<y<<endl;};
void Nuqta::Uzgartirish(int delta_x, int delta_y)
{
 x+=delta_x; y+=delta_y;
};
int main( )
{
Nuqta koord1(5,10);
Nuqta koord2=koord1;
Nuqta koord3(koord1);
 koord1.Uzgartirish (3, -2);
 koord2.Uzgartirish (1, 2);
 cout<<” koord1 Ob’yekt berilgan-a’zolari qiymati:\n”;
koord1.Abstissa();
koord1.Ordinata();
 cout<<” koord2 Ob’yekt berilgan-a’zolari qiymati:\n”;
koord2.Abstissa();
koord2.Ordinata();
 cout<<” koord3 Ob’yekt berilgan-a’zolari qiymati:\n”;
koord3.Abstissa();
koord3.Ordinata();
return 0;
}
```

Dastur ishlashi natijasida ekranga quyidagilar chop etiladi:

koord1 Ob’yekt berilgan-a’zolari qiymati:

x=8

y=8

koord2 Ob’yekt berilgan-a’zolari qiymati:

x=6

y=12

koord3 Ob’yekt berilgan-a’zolari qiymati:

x=5

y=10

Muhokama savollari

1. Nusxalash konstruktori.
2. Sinfning kontansta obyektlari va konstanta funksiya-a’zolari.
3. Sinf usullarining aniqlanishi.
4. Sinfning statik a’zolari.

Nazorat savollari

1. Sinfning statik a’zolari.
2. Sinflar haqida tushuncha.
3. Sinf xossalari va usullari?
4. Sinf usullarining aniqlanishi.

4.5 this ko‘rsatgichi

C++ tilidagi har bir ob’yekt kompilyator tomonidan yaratiladigan va ob’yektga ko‘rsatuvchi *this* deb nomlanuvchi maxsus ko‘rsatgichga ega. this ko‘rsatgichining turi S* bo‘lib, bu yerdagi S – ushbu ob’yekt sinfi turidir. this ko‘rsatgichi sinfda aniqlanganligi uchun uning amal qilish sohasi o‘zi aniqlangan sinf bo‘ladi. Boshqacha aytganda, this ko‘rsatgichini kompilyator tomonidan qo‘shiluvchi sinfning yashiringan parametri deb qarash mumkin. Sinf funksiya-

a’zosi chaqirilganda unga this ko‘rsatgichi, go‘yoki birinchi argument sifatida uzatiladi. Ya’ni funksiya-a’zoni chaqirishning quyidagi ko‘rinishi

Obekt1.Funktsiya(arg1,arg2);

kompilyator tomonidan

Obekt1.Funktsiya(&Obekt1, arg1,arg2);

ko‘rinishida talqin qilinadi.

Funksiya chaqirilganda uning qavs ichidagi argumentlari stekka o‘ngdan chapga tomonga qarab joylashtirildi. Birinchi argument (this) stekka eng oxirda joylashadi. Funksiya–a’zo ichida ob’yektlar adresi this orqali aniqlanadi. Quyida this ko‘rsatgichini ishlatish bilan bog‘liq dastur matni keltirilgan.

```
#include <iostream.h>
#include <string.h>
class S
{
char Ism[20];
public:
S(char*);
void Salom();
};
S::S(char * _Ism)
{
strcpy(Ism, _Ism);
Salom();                    // uchta murojaat
this -> Salom();       // o‘zaro
 (*this).Salom();      // ekvivalent
}
void S::Salom()
{
cout<<”Salom “<<Ism<<’\n’;
cout<<”Salom “<<this ->Ism<<’\n’;
```

```
}
int main( )
{
S obekt (“Muqumov Rustam”);
return 0;
}
```

Dastur matnidan koʻrinib turibdiki, funksiya-aʼzo ichida boshqa funksiya-aʼzolarga va berilganlar–aʼzolarga bevosita ularning nomlari bilan yoki this koʻrsatgichi orqali murojaat qilish mumkin. Shu sababli amaliyotda this koʻrsatgichidan kam foydalaniladi. Asosan, this koʻrsatgichi funksiya qaytaruvchi qiymati sifatida (return this; yoki return *this;) va operatorlarni qayta yuklash masalalarida keng qoʻllaniladi.

Muhokama savollari

1. Maxsus koʻrsatgich.
2. this koʻrsatgichining turi.
3. this koʻrsatgichining amal qilish sohasi.
4. this koʻrsatgichi funksiya qaytaruvchi qiymati sifatida.

Nazorat savollari.

1. Sinfning statik aʼzolari.
2. Sinflar haqida tushuncha.
3. Sinf xossalari va usullari?
4. Sinf usullarining aniqlanishi.

4.6 Joylashtiriladigan (inline) funksiyalar–a’zolar

Funksiyalar bo‘limida inline funksiyalar haqida ma’lumot berilgan edi. Ularning boshqa funksiyalardan farqi – kompilyator dasturda bunday funksiyalar chaqirilgan joyga funksiyani chaqirish buyrug‘ini emas, balki funksiya tanasini qo‘yadi. inline funksiyalarning afzalligi shunda ediki, dasturda oddiy funksiyani chaqirishdagi argumentlarni uzatish, stekka qiymatlarni joylashtirish va qayta olish bilan bog‘liq qo‘shimcha amallar bajarilmaydi. Noqulay tomoni, inline funksiyaga murojatlar ko‘p bo‘lganda dastur hajmi oshib ketadi. Bu o‘rinda inline funksiyalar qo‘llanishini makroslarni ishlatishga o‘xshatish mumkin. Shu sababli, odatda juda sodda funksiyalar inline funksiyalar sifatida ishlatiladi. Funksiyalar aniqlanishda inline spesifikatorini qo‘yishni kompilyatorga funksiya chaqirilgan joylarga funksiya tanasini qo‘yishga talab deb qaraladi. Agar kompilyator funksiya tanasini qo‘yishni amalga oshira olmasa, inline funksiya oddiy funksiya sifatida ishlatiladi. Kompilyator quyidagi holatlarda inline funksiyani chaqiruv joyiga qo‘ya olmaydi, agarda funksiya:

- tanasida takrorlash operatorlari ishlatilgan bo‘lsa (for, while, do while);
- tanasida switch, goto operatorlari bo‘lsa;
- static o‘zgaruvchilarni ishlatsa;
- rekursiv bo‘lsa;
- void turidan farqi qaytaruvchi turga ega va tanasida return operatori bo‘lmasa;
- tanasida assembler kodli bo‘laklar mavjud bo‘lsa.

Joylashtiriladigan funksiyalar sifatida nafaqat oddiy funksiyalar, balki sinfning funksiya– a’zolari ham aniqlanishi mumkin. Buning uchun sinf e’lonida funksiya–a’zoni e’loniga uning aniqlanishini qo‘yish yetarli yoki sinfdan tashqaridagi funksiya aniqlanishida, funksiya sarlavhasi oldiga inline spesifikatorini qo‘yish zarur bo‘ladi. Quyida keltirilgan misolda joylashtiriluvchi funksiya – a’zolarining ikki xil variantdagi e’loni ko‘rsatilgan.

class Nuqta

```
  {
  int x;
int y;
public :
  Get_x(){return x;}
  Get_y(){return y;}
void Set_x(int _x);
void Set_y(int _y);
 }
inline void Nuqta::Set_x(int _x) {x= _x;}
inline void Nuqta::Set_y(int _y) {y= _y;}
```

Muhokama savollari

1. inline funksiya tushunchasi
2. inline funksiyaning makroslarga o‘xshash tomoni

Nazorat savollari

1. inline funksiyalar haqida ma’lumot.
2. inline funksiyalarning noqulay tomoni.
3. inline spesifikatori.

4.7 Sinfning statik a’zolari

Sinf a’zolari static modifikatori bilan e’lon qilinishi mumkin. Sinf statik a’zosini sinf sohasi chegarasida murojaat qilish mumkin bo‘lgan global o‘zgaruvchi yoki funksiya deb qarash mumkin. Sinfning static deb e’lon qilingan berilganlar-a’zolari sinfning barcha ob’yektlari tomonidan birgalikda ishlatiladi, chunki bunday o‘zgaruvchining yagona nusxasi bo‘ladi. Amalda sinfning statik berilganlari uchun xotiradan joy ajratiladi, hattoki sinfning birorta ob’yekti

bo‘lmasa ham. Shu sababli sinf statik berilganini e’lon qilib qolmasdan, uni aniqlash shart. Masalan:

```
slass Sinf
{
public:
 Sinf();
static int Sanagich;//statik berilgan–a’zo e’loni
   }
int Sinf::Sanagich=0; // statik berilgan–a’zo e’loni
```

Bu misolda, garchi Sanagich statik berilgan – a’zo public bo‘limida e’lon qilingan sinf ob’yekti nomini ishlatish yordamida murojaat qilish mumkin.

```
Sinf sinf1;
Sinf1.Sanagich++;
Sinf sinf2;
Sinf2->Sanagich--;
```

Statik berilganlar – a’zolarga sinf nomi orqali murojaat qilgan ma’qul bo‘ladi.

```
  Sinf::Sanagich++;
```

Bu holat Sanagich statik berilgan–a’zo barcha sinf ob’yektlari uchun yagona ekanligini ta’kidlaydi.

Agarda statik berilganlar yopiq deb e’lon qilingan bo‘lsa, ularga funksiyalar – a’zolar orqali murojaat qilish mumkin.

Umuman olganda statik berilganlar–a’zolarni ishlatishda quyidagi takliflarni berish mumkin:

- statik berilganlar-a’zolarni bir nechta sinf ob’yektlari tomonidan birgalikda ishlatish uchun aniqlang;
- statik berilganlar-a’zolarini private, protected modifikatorlar bilan e’lon qilish orqali ularga murojaatni cheklang.

Sinfning statik berilgan-a’zosini ishlatishga misol.

```
  class S;
  {
```

```
public:
S() {ob_soni++;}
  ~S() {ob_soni--;}
static int ob_soni ;
private:
int x;
 };
int S::ob_soni=0;
int main ()
 {
 S* p_ob=new S[5];
 cout<<”Sinfning ”<<S::ob_soni<<”Ob’yekti mavjud./n”;
delete [] p_ob;
return 0;
 }
```

Dastur ishlash natijasida ekranga

Sinfning 5 ob’yekti mavjud.

Sinfning statik funksiyalarni ishlatishning o‘ziga xosligi shundaki, ular ham yagona nusxada aniqlanadi va birorta sinf ob’yektining “shaxsiy” funksiyasi bo‘lmaydi. Shu sababli, bu funksiyalarga this ko‘rsatgichi uzatilmaydi. Statik funksiyalarning bunday xususiyatidan Windows uchun dasturlashda keng foydalaniladi.

Yuqorida aytilgan fikrlardan bir nechta muhim xulosalar kelib chiqadi:

– statik funksiya –a’zolarni sinfning birorta ham vakili (ob’yekti) mavjud bo‘lmasa ham chaqirish mumkin;

– sinfning statik funksiyasi faqat sinfning statik berilganlarini qayta ishlashi mumkin va faqat sinfning statik funksiya–a’zolarini chaqirishi mumkin;

– statik funksiya-a’zo virtual modifikatori bilan e’lon qilinishi mumkin emas.

Quyida keltirilgan dastur funksiya–a’zoni ishlatishga misol bo‘ladi:

```
#include <iostream.h>
class S
 {
public:
S() {sanagich++};
   ~S() {sanagich--};
   //..
  static int Sinf_Sanagichi(){return sanagich;}
private:
int x;
static int sanagich;
 };
int S::sanagich=0;

int main()
 {
 S * pOb= new S[10];
cout<<"S sinfning "<<S::Sinf_Sanagichi()
<<" Ob'yekti mavjud"<<endl;
delete [] pOb;
return 0;
 }
```

Muhokama savollari

1. static modifikator tushunchasi
2. static modifikatorli o‘zgar uvchilarga xotiradan joy ajratish

Nazorat savollari

1. statik berilganlar yopiq deb e'lon qilinganda murojaat
2. statik berilganlar–a'zolarni ishlatishdagi takliflar

4.8 Sinfning konstansta ob’yektlari va konstanta funksiya-a’zolari

Sinfning funksiya-a’zolari parametrlar ro‘yxatidan keyin keluvchi const modifikatori bilan e’lon qilinishi mumkin. Bunday funksiya sinf berilganlar-a’zolari qiymatlarini o‘zgartira olmaydi va sinfning konstanta bo‘lmagan funksiya–a’zolarini chaqirishi mumkin emas. Konstanta funksiya-a’zosi bo‘lgan sinfga misol keltiramiz.

```
classNuqta
 {
int x,y;
  public:
Nuqta(int _x, int _y);
void Qiymat_Berish(int x0, int y0);
//konstanta funksiya-a’zo
void Qiymat_Olish(int xx, int yy) const;
};
Nuqta::Nuqta(int _x, int _y)
{ x=_x; y=_y;}
voidNuqta::Qiymat_Berish(int x0, int y0)
{ x=x0; y=y0;}
voidNuqta::Qiymat_Olish(int &xx, int &yy)const
{ xx=x; yy=y; }
```

Xuddi shunday, konstanta ob’yektlar yaratish mumkin. Buning uchun ob’yekt e’loni oldiga const modifikatori qo‘yilishi kerak.

```
const Nuqta koord(3,6);
```

const kalit so‘zi kompilyatorga ushbu ob’yektning holati o‘zgarmasligi kerakligini bildiradi. Shu sababli ob’yekt berilgan-az’olari qiymatini o‘zgartiradigan funksiya–a’zosini chaqirishi uchrasa, kompilyator xato haqida xabar beradi. Bu qoidaga konstanta funksiya-a’zolarni chaqirish rioya qilmaydi,

chunki o‘z mazmuniga ko‘ra ular berilgan–a’zolarni qiymatini o‘zgartira olmaydi. Yuqorida e’lon qilingan Nuqta sinfini konstanta ob’yekt ishlatishiga misol.

```
// Nuqta sinf e’loni va aniqlanishi
int main()
 {
Nuqtanuqta1(3,7);
constNuqta nuqta2(8,10);//Konstanta ob’yekt
int a,b;
 nuqta1.Qiymat_Olish (a,b);
 nuqta2. Qiymat_Berish(2,3);// Xato
nuqta2. Qiymat_Olish(a,b); // To‘g‘ri
 return 0;
 }
```

Konstanta funksiya–a’zolarning berilganlar–a’zolar bilan ishlashda bog‘liq cheklovlarni “aylanib o‘tish” uchun mutable kalit so‘zi aniqlangan. Bu kalit so‘z sinfning qaysi berilgan-a’zosi konstanta funksiya-a’zolar tomonidan o‘zgartirilishi mumkin ekanligini ko‘rsatadi. Statik va konstanta berilganlar-a’zolariga mutable kalit so‘zini ishlatish mumkin emas, u berilganlar turining modifikatori sifatida ishlatiladi.

Misol.

```
#include <iostream.h>
class Sinf
{
mutable int count;
mutable const int * intPtr;//O‘rinli, garchi
// ko‘rsatgich konstanta butun songa ko‘rsatsa
// ham o‘zi konstanta emas
public:
```

```
int Funktsiya(int i=0) const
 {
count=i++;
intPtr =&i;
cout<<*intPtr;
return count;
 }
};
int main()
{
 Sinf S;
S.Funktsiya();
return 0;
}
```

Dastur ishlashi natijasida ekranga

1

chop etiladi.

Shu vaqtgacha tuzgan dasturlarimiz berilgan ma’lumotlar ustida biror-bir amallarni bajaruvchi proseduralar ketma-ketligidan iborat edi. Prosedura va funksiyalar ham o‘zida aniqlangan ketma-ket komandalar to‘plamidan iboratdir.

Ob’yektga yo‘naltirilgan dasturlashning asosiy maqsadi berilganlar va ular ustida amal bajaruvchi proseduralarni yagona ob’yekt deb qarashdan iboratdir. Masalan: non, avtomobil, odam … ob’yektlari.

Muhokama savollari

1. Sinfning funksiya-a’zolari
2. Konstanta funksiya-a’zosi

Nazorat savollari

1. Konstanta ob’yektlar yaratish
2. Konstanta funksiya–a’zolarning berilganlar
3. mutable kalit so‘zi

4.9 Sinf usullarining aniqlanishi

Sinf usulini aniqlash uchun sinf nomidan so‘ng ikkita ikki nuqta (::) belgisi, funksiya nomi va uning parametrlari ko‘rsatiladi.

Masalan: non sinfining patir ob’yektidagi usullarni aniqlash dasturi:

```
#include < iostream.h >
class non
{ public :
  int baho;
  int og‘irlik;
  void yasash ( );  };
void non :: yasash ( );
{ cout << “hamir qoriladi, zuvala tutiladi, doira holiga keltiriladi”<< endl; }
int main ( )
{ non patir;
patir.baho = 1200;  patir.og‘irlik = 500;
cout <<”men olgan patir “<<patir.baho <<” so‘m”<<endl;
cout <<” uning og‘irligi =”<<patir.og‘irlik <<endl ;
cout <<” u quyidagich yasaladi:”;
patir . yasash ( );  }
```

Misollar: 1.

```
#include <vcl.h>
    #pragma hdrstop
    #include<iostream.h>
    #include<conio.h>
```

```
#include<string.h>
#pragma argsused
int i;
struct waw
{
  char fam[15];
  char ism[15];
  int tel;
  int date[3];
};
class note
{
  waw A[3],t;
  public:
  void kiritish();
  void saralash();
  void taqqoslash();
  void chiqarish();
};
void note::kiritish()
{
  for( i=0; i<3; i++)
  {
    cout<<i+1<<«. «;
    cin>>A[i].fam;
    cin>>A[i].ism;
    cin>>A[i].tel;
    cin>>A[i].date[0]>>A[i].date[1]>>A[i].date[2];
  }
}
```

```
// saralash tel. raqamining 1-chi 3 ta raqami bo‘yicha
void note::saralash()
{
  int x,y;
  for( i = 0; i < 2; i++)
  {
    x = A[i].tel / 10000;
    for(int j = i; j < 3; j++)
    {
      y = A[j].tel / 10000;
      if(x > y)
      {
        t = A[i];
        A[i] =A[j];
        A[j] = t;
      }
    }
  }
}
void note::taqqoslash()
{  char f[15];
  int w=0;
  cout<<”\n\nfamilyani kiriting = “;
  cin>>f;
  for(i=0; i<3; i++)
  if( strcmp(A[i].fam, f)==0)
    {
   cout<<«\n\nfamiliya ismi   --->”<<A[i].fam<<”\t”<<A[i].ism<<endl;
    cout<<«telifon nomeri   --->   “<<A[i].tel<<endl;
```

```
          cout<<«tug‘ilgan sanasi  --->   “<<A[i].date[0]<<”
“<<A[i].date[1]<<” “<<A[i].date[2]<<endl;
          w++;  break;
          }
         if(w==0)
         cout<<”bunday familiya haqida ma'lumot yo‘q”;
        }
        void note::chiqarish()
        {
         for(i = 0; i < 3; i++)
         {
          cout<<”\n\nfamiliya ismi --->”<<A[i].fam<<”\t”<<A[i].ism<<endl;
          cout<<”telifon nomeri   --->   “<<A[i].tel<<endl;
          cout<<”tug‘ilgan sanasi  --->   “<<A[i].date[0]<<”
“<<A[i].date[1]<<” “<<A[i].date[2]<<endl;
          }
        }
        void main()
        {
         note B;          //ob’yekt tavsifi
         cout<<”ma'lumotlarni kiriting: (familiya, ism, tel.nomer, tug‘ilgan
sana):\n”;
         B.kiritish();
         cout<<”\n tartiblangan holatda : \n\n”;
         B.saralash();
         B.chiqarish();
         cout<<”\n\n kiritilgan familiya haqida malumot chiqarish:\n\n”;
         B.taqqoslash();
        getch();
        }
```

2. y = Ax+B chiziqli tenglama. A koeffisient sifatida first maydoni – kasr son; B koeffisient sifatida kasr son bo‘lgan second maydoni olinmoqda. Chiziqli tenglama ildizini aniqlovchi root () usuli tatbiq qilinsin. Usul B koeffisientning nolga teng emasligini tekshirmog‘i lozim.

```
#include <vcl.h>
#pragma hdrstop
#include <iostream.h>
#include <conio.h>
#include <stdio.h>
#include <string.h>
#pragma argsused
struct kasr
{
  float surat;
  unsigned int maxraj;
};
class tenglama
{
  public:
      kasr A, B;
      float y, x;
      void vvod (void);
      void root(void);
};
  void tenglama::vvod(void)
  {
    cout << «\n x = « ;
    cin >> x;
    cout << «\nA kasr surati = « ;
    cin >> A.surat;
```

```
  cout << «\nA kasr maxraji = « ;
  cin >> A.maxraj;
  cout << «\nB kasr surati = « ;
  cin >> B.surat;
  cout << «\nB kasr maxraji = « ;
  cin >> B.maxraj;
 }
 void tenglama::root(void)
 {
  y = x * (float)(A.surat / A.maxraj) + (float)(B.surat / B.maxraj);
 }
int main(int argc, char* argv[])
{
 tenglama C;
 C.vvod();
 C.root();
 cout << "\n\nnatija :\n y =”<< C.y ;
 getch();
 return 0;
}
```

3. Qiymatlari gradus va minutlarda berilgan tekislikdagi burchaklar bilan ishlovchi Angle sinfi tuzilsin.

Bunda radianlarga o‘tkazish, 0-360 diapazonga keltirish, burchakni berilgan qiymatga ko‘paytirish yoki kamaytirish, sinusni tashkil etish, burchaklarni taqqoslashni tatbiq etilsin.

```
#include <vcl.h>
#pragma hdrstop
#include <iostream.h>
#include <conio.h>
```

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#pragma argsused
class angle
{
   float gradus;
   float minut;
   float radian;
   float sinus;
 public :
   void read(void);//float *gradus, float *minut, float *sinus);
   void converter(void);
   void write(void);
};
void angle :: converter(void)
{
 radian = (gradus + minut / 60) / 180 * 3.14;
}
void angle :: read(void)//float *gradus_value, float *minut_value, float
*sinus_value)
{
 cout << «\ngradus = « ;
 cin >> gradus ;
 if(gradus > 360)
 {
  int qoldiq = gradus;
  qoldiq = qoldiq % 360;
  gradus = qoldiq;
 }
```

```
  cout <<”minut =”;
  cin >> minut ;
  radian = (gradus + minut / 60) / 180 * 3.14;
  sinus = sin(radian * 180 / 3.14);
}
void angle :: write(void)
{
  cout <<”\ngradus = ” << gradus;
  cout <<”\nminut = ” << minut;
  cout << ”\nradian = ” << radian;
  cout <<”\nburchak sinusi = ” << sinus;
}
int main(int argc, char* argv[])
{
  angle info[10];
  int n, i, min, max;
  cout << ”elementlar soni =” ;
  cin >> n ;
  for(i = 0; i < n; i++)
  {
    cout << i + 1 <<”- inchi ma’lumotni kiriting” ;
    info[i].read();
  }
  cout << ”\nnatija : \n” ;
  for(i = 0; i < n; i++)
  {
    cout <<”\n\n”<< i <<” - inchi ma’lumotni : “ ;
    info[i].write();
  }
  getch();
```

```
    return 0;
  }
```

Student sinfi tashkil etilib, guruh talabalari orasidan 4 va 5 baho olgan talabalarni familiyasi, guruhi chiqadigan dastur tuzilsin.

```
#include <conio.h>
#include <stdio.h>
#include<math.h>
#include<iostream.h>
#include<string.h>
#include <stdlib.h>
class student
{ public :
    char fam[20];
  int baho;
  int guruh;
  int bilimdon (int );  };
int student :: bilimdon (int x)
{ if(x>=4) return x;
else return 0; }
int main ( )
{student alo[10]; int i,a=0;
for (i=0;i<5;i++){
  cout<<i+1<<”-talaba:”<<endl;
  cout<<”familiya:”;
  cin>>alo[i]. fam;
  cout<< ”guruhi: ”;
  cin>>alo[i].guruh;
  cout<< ”bahosi: ”;
  cin>>alo[i].baho;} cout<<”4 va 5 olgan talabalar: ”<<endl;
  for(i=0;i<5;i++){
```

```
int y=alo[i] . bilimdon (alo[i].baho );
if(y!=0){a++;
cout<< familiya: ”<<alo[i].fam<<'\t'<<”guruhi: ”<<alo[i].guruh<<endl;}}
if(a==0) cout<<”talabalarning hammasi ikkichi”;
 getch();
return 0;
}
```

5. Uchburchakni aniqlashtiruvchi Triangle sinfi tashkil etilsin. Ma’lumotlar maydoni tarkibiga burchaklar va tomomnlari kiritilsin. Berilganlar maydonlarini olish va o‘zgartirish, yuzasini hisoblash, perimetrni hisoblash, balandliklarni hisoblash hamda uchburchak turini aniqlash amallarini joriy etish talab etiladi.

```
#include <vcl.h>
#pragma hdrstop
#include <iostream.h>
#include <conio.h>
#include <math.h>
#pragma argsused
  class triangle
  {
    float x[3];
    float y[3];
    float tomon[3];
    float h[3];
    float s;
    float p;
    char stil_treugolnik[101];
  public :
    void vvod(void);
    void schitat(void);
  };
```

```
void triangle :: vvod(void)
{
 cout << “\nA uchni \nX =” ;
 cin >> x[0];
 cout << “Y =” ;
 cin >> y[0];
 cout << “\nB uchni \nX = ” ;
 cin >> x[1];
 cout << “Y = ” ;
 cin >> y[1];
 cout << “\nC uchni \nX = ” ;
 cin >> x[2];
 cout << “Y = ” ;
 cin >> y[2];
}
void triangle :: schitat(void)
{
 tomon[0] = pow( (pow((x[0]-x[1]), 2) + pow((y[0]-y[1]), 2) ), 1/2);
 tomon[1] = pow( (pow((x[1]-x[2]), 2) + pow((y[1]-y[2]), 2) ), 1/2);
 tomon[2] = pow( (pow((x[2]-x[0]), 2) + pow((y[2]-y[0]), 2) ), 1/2);
 cout <<”\n ucburchak tomonlari : “<<tomon[0]<<”, “<<tomon[1]<<”,
“<<tomon[2];
 p = tomon[0] + tomon[1] + tomon[2];
 cout <<”\nperimetr =”<<p;
 float p2 = p;
 s = sqrt(p2 * (p2-tomon[0])*(p2-tomon[1])*(p2-tomon[2]) );
 cout << “\nyuza =”<< s;
 h[0] = s / tomon[0];
 h[1] = s / tomon[1];
 h[2] = s / tomon[2];
```

```
        cout <<"\n balandliklar :"<<h[0] <<","<<h[1]<<","<<h[2];
        if(tomon[0] == tomon[1] && tomon[1] == tomon[2])
        strcpy(stil_treugolnik, «teng tomonli»);
        else if(tomon[0] == tomon[1] || tomon[1] == tomon[2] || tomon[0] == tomon[2])
        strcpy(stil_treugolnik, "teng yonli");
        else strcpy(stil_treugolnik, "turli tomonli");
        cout <<"\n uchburchak turi :"<<stil_treugolnik;
       }
      int main(int argc, char* argv[])
      {
       triangle info;
       info.vvod();
       info.schitat();
       getch();
       return 0;
      }
```

6. Dasturda to‘rtburchaklar baza sinfi va voris sinflar sifatida parallelogramlar, romblar, kvadratlar sinfi tavsiflangan.

```
  #include <iostream.h>
#include <math.h>
// to‘rtburchaklar baza sinfi
 classFourAngle
{ protected :
  double  x1,y1,x2,y2,x3,y3,x4,y4,
A,B,C,D,D1,D2,
Alpha,Beta,Gamma,Delta, P,S;
public:
voidInit(void);
void Storony(void);
```

```
void Diagonali (void);
voidAngles (void);
void Perimetr(void);
void Ploshad(void)
void PrintElements(void)
};
// to‘rtburchaklar sinfi vorisi parallelogrammlar sinfi
class Parall: public Four Angle
{public:
void Storony(void);
voidPerimetr(void);
voidPloshad (void);
//romblar sinfi parallelogrammlar sinfi vorisi
class Romb: public Parall
{ public:
voidStorony(void);
voidPerimetr(void);
};
//Kvadratlar sinfi romblar sinfi vorisi
classKvadrat: publicRomb
{ public:
voidAngles  (void);
voidPloshad(void);
};
//Sinf a’zolari funksiyasi tavsifi
void Four Angle:: Init(void)
{ cout<<”\n, Uchlar koordinatasini kiriting:\n”;
cin>>x1>>y1>>x2>>y2>>x3>>y3>>x4>>y4;
}
voidFour Angle:: Storony(void)
```

```
{ A=sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
  B=sqrt((x3-x2)*(x-3-x2)+(y3-y2)*(y3-y2));
 C=sqrt  ((x4-x3)*(x4-x3)+(y4-y3)*(y4-y3));
 D=sqrt ((x4-x1)*(x4-x1)+(y4-y1)*(y4-y1));
}
void FourAngle::Diagonali(void)

{  D1=sqrt ((x1-x3)*(x1-x3)+(y1-y3)*(y1-y3));
   D2=sqrt ((x2-x4)*(x2-x4)+(y2-y4)*(y2-y4));
// Ugol funksiyasi Angles uchun qo‘shimcha imkoniyatlar funksiyasidir.
//Ushbu funksiya bevosita asosiy dasturda uchburchak tomonlari bilan
//aniqlashtiriluvchi burchaklarini aniqlash uchun ishlatilishi mumkin.
Double Ugol (double Aa, double Bb, double Cc)
{ doubleVspCos,  vspSin,  Pi;
  Pi=4*atan (1.0);
VspCos=(Aa*Aa+Bb*Bb-Cc*Cc)/2/Aa/Bb;
VspSin=sqrt(1-VspCos*VspCos);
If (abs(VspCos)>1e-7)
return(anat(VspSin/VspCos) +Pi* (VspCos<0)) / Pi / 180;
else return   90.0;
}
void Four Angle: :Angles (void)
{ Alpha  = Ugol (D,A,D2):  Beta=Ugol (A,B,D1);
   Gamma=Ugol (B,C,D2) ;  Delta=Ugol (C,D,D1);
}
void FourAngle: :Perimetr (void)
 {  P=A+B+C+D; }
void FourAngle: :Ploshad (void)
{ double Per1, Per2;
Per1=( A+D+D2)/2; Per2=(B+C+D1) /2;
```

```
S=sqrt (Per1*( Per1*-A) *( Per1-D)*( Per1-D2))+
sqrt ((Per2*( Per2-B) *(Per2-C)*( Per2-D1));
 }
void Four Angle: :PrintELements (void)
{  cout<<”tomonlari:\n ”<<A<<” ”<<B<<” ”<<C<<” ”<<D<<”\n ;
cout<<”Burchaklar:\n”
<<Alpha <<Beta<<” ” Gamma<<” ”<<Delta<<” \n”;
cout<<”Perimetr:\n ”<<P<<”\n” ;
cout<<”Uza:\n”<<S<<”\n ”;
cout<<”Diagonallar:\n ”<<D1<<” ”<<D2<<”\n” ;
}
voidParall : : Storony(void)
{ A=sqrt ((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
   B=sqrt ((x3-x2)*(x3-x2)+(y3-y2)*(y3-y2));
   C=A;  D=B;
 }
voidParall : : Perimetr(void)
{ P=2*(A+B); }
voidParall : :Ploshad(void)
{ double Per ;
Per= (A+D+D2)/2;
S=2*sqrt(Per*(Per-A)*(Per-D)*(Per-D2) );
}
voidRomb : : Storony(void)
{ A=B=C=D=sqrt ((x2-x1)*( x2-x1)+(y2-y1)*( y2-y1));
}
voidRomb : : Storony(void)
{P=4*A;}
voidKvadrat : : Angles(void)
{ Alpha=Beta=Gamma-Delta=90.0;}
```

```
voidKvadrat : : Ploshad(void)
{S=A*A;}
// Asosiy funksiyada kvadrat uchlari koordinatalariga ko‘ra
// uning parametrlarini hisoblaydi va chop etadi
Void main(void)
{ Kvadrat obj;// “kvadrat” sinfi ob’yektlari tavsifi
Obj. Init();
 Obj. Storony();
Obj. Dioganali();
Obj.Angles();
Obj.Perimetr();
Obj.Ploshad();
Obj.PrintElements();
}
```

7. Oddiy kasrlar sinfi tavsiflanib, ushbu sinf o‘zgaruvchisi tavsiflanmoqda va o‘zgaruvchiga ishlov berish bajarilmoqda.

```
#include <iostream.h>
#include <math.h>
struct Frac{int P; int Q;} ;
Frac F;
//Sinf tavsifi
class Drob{
Frac A;
public:
void Vvod (void) ;
int NOD (void) ;
void Sokr (void) ;
void Stepen (int N) ;
void Print (void) ;
```

```
};
// Sinf a’zolari – funksiyalar tavsifi
void Drob: :Vvod (void)
{ cout<<“Surati?” ; cin>>A.P;
cout<<“maxraji?” ; cin>>A.Q;
}
int Drob : :NOD (void)
{ int M,N;
M=abs (A.P); N=A.Q;
while (M!=N)
{ if (M>N)
if (M%N!=0) M=M%N; else M=N;
else if (N%M!=0) N=N%M; else N=M
}
return M;
}
void Drob: :Sokr (void)
{ int X;
X=NOD ( );
if (A.P!=0)
{ A.P=A.P/X; A.Q=A.Q/X;}
else A.Q=1;
}
void Drob: : Stepen (int N)
{ int i;
F.P=F.Q=1;
for (i=1; i<=N; i ++)
{ F.P*=A.P; F.Q*=A.Q;}
 }
void Drob: :Print (void)
```

```
{ cout<<“\n”<<A.P<<“/”<<A.Q<<“\n”;}
//Asosiy fuksiya
void main (void)
{ Drob Y;  //Ob’yekt tavsifi
cout <<“kasrni kiriting!” <<“\n”;
Y.Vvod () ;
Y. Sokr () ;
cout<<“qisqartirilgan kasr: “<<”\n”;
Y.Print ();
Y.Stepen (2);
cout<<“kvadratga oshirilgan kasr:” <<“\n”;
cout<<F.P<<“/”<<F.Q<<“\n”;
}
```

Vector sinfi evklid fazosidagi uch o‘lchamli vektorni aniqlaydi. Sinfda (+) qo‘shish va (=) o‘zlashtirish qayta yuklash amallari uch o‘lchamli vectorlar ustidagi amallar sifatida qo‘llanmoqda. Ikkita vector yug‘indisi, mos elementlarining yig‘indisi kabi hisoblanayotgan vector sifatida, = amali vektorlarning mos elementlarini o‘zlashtirish kabi bajariladi.

```
#include <iostream.h>
class vector
{ int x, y, z; // vector komponentalari
public:
vector operator+ (vector t) ;
vector operator= (vector t) ;
void show (void);
void assign (int mx, int my, int mz);
}
// + amalini qayta yuklash
vector vector : :operator+ (vector t)
{vektor temp;
```

```
temp.x=x+t.x;
nemp.y=y+t.y;
temp.z=z+t.z;
}
// = amalini qayta yuklash
vector vector : :operator = (vector t)
{ x=t.x; y=t.y; z=t.z;
return *this; // this qo‘llanmoqda
}
void vector : :show (void)
{coun<<x<<“, “;
cout<<y<<“, “;
cout<<x<<\n’;
}
void vektor : :assign (int mx , int my, int mz)
{ x=mx; y=my; z=mz; }
//asosiy dastur
void main (void)
{ vector a, b, c;
a. assign (1, 2, 3);
b. assign (10, 10, 10);
a. show ();
b. show ();
c=a+b; // qayta yuklangan +, = amallari ishlamoqda
c. show () ;
c. a+b+c;
c. show;
c=b=a;
c. show;
b. show;
```

}

Quyidagi maydonlardan tashkil topgan MARSH sinfi tashkil etilsin:

• marshrut boshlang‘ich punkti nomi;

• marshrut so‘nggi punkti nomi;

• marshrut tartib raqami.

Yozuvlari marshrut nomerlari bo‘yicha tartiblangan, toifasi MARSH bo‘lgan m-ta elementli massivni kiritib, tartib raqami klaviaturadan kiritilgan marshrut haqidagi axborotni chiqaruvchi dastur tuzilsin. Agar bunday marshrut bo‘lmasa, mos yozuv chiqarilsin.

```
#include<iostream.h>
#include<conio.h>
class MARSH{
string punA;
string punB;
int num;
public:
void Input(){
cin>>punA>>punB>>num;}
void Output();
int GetNum(){
return num;}
void operator^(MARSH &a);};
void MARSH::Output(){
cout<<”\nBoshlang‘ich punkt:”<<punA<<endl;
cout<<”Oxirgi punkt:”<<punB<<endl;
cout<<”marshrut nomeri:”<<num<<endl;
cout<<«----------\n»;}
void MARSH::operator^(MARSH &a){
string st1=punA,st2=punB; int n=num;
punA=a.punA; punB=a.punB; num=a.num;
```

```
a.punA=st1; a.punB=st2; a.num=n;}
void Sort(MARSH *a,int n){
for(int i=0;i<n-1;i++)
for(int j=i+1;j<n;j++)
if(a[i].GetNum()>a[j].GetNum()) a[i]^a[j];}
void Poisk(MARSH *a,int n,int num){
for(int i=0;i<n;i++)
if(a[i].GetNum()==num) a[i].Output();}
int main(){
MARSH A[8];
for(int i=0;i<8;i++) A[i].Input();
Sort(A,8);
for(int i=0;i<n;i++) A[i].Output();
int n;
cin>>n;
Poisk(A,8,n);
getch();}
```

4.10 Bob bo‘yicha xulosalar

Ushbu bobda Siz sinflarni tashkil etish, ob’ektlardan foydalanish xaqidagi tushunchaga ega bo‘ldingiz. Sinf turli toifali o‘zgaruvchilardan, shu jumladan boshqa sinfdan tashkil topgan berilganlar-a’zolarga ega. Bundan tashqari sinf tarkibiga usullar deb ataluvchi funksiy-a’zolar ham kiradi. Ularning har biri himoyalsh usulini beruvchi maxsus spetsifikatorga ega bo‘ladi. Agar o‘zgaruvchi-a’zolar public sifatida tavsiflangan bo‘lsa, ushbu o‘zgaruvchi tarkibiga kirgan sinf ob’ektlarini ishlatmasdan turib, faqat nomi orqali unga murojaat qilish mumkin. Statik o‘zgaruvchi a’zolarni ob’ektlar hisobchisi sifatida qo‘llash mumkin.

Glossariy

Bayt- axborot o’lchovi birligi.

Bit- bir yoki nol qabul qiluvchi xotira yacheykasi.

Bufer-kiritish chiqarish amallari bajarilish vaqtida vaqtinchalik berilganlarni saqlash uchun ishlatiladigan xotira qismi.

Bufer – nusxasi olingan axborotni saqlash uchun maxsus xotira maydoni.

Vinchester – sistemali blok tarkibidagi qattiq disk.

Tashqi xotira – vinchester, kompakt-disk yoki disketlar.

Windows – operasion tizim.

Delete – o’chirish komandasi yoki tugmasi

Disketa – egiluvchan magnit diski.

Disk qurilmasi – axborotni disketadan o’qish va yozish qurilmasi.

Exit – dasturdan chiqish komandasi yoki tugmasi.

Fayl belgisi – fayl tashkil etilgan ilova belgisi ko’rinishga ega bo’lgan nomli piktogramma.

Informatika – kompyuter texnikasini qo’llashga asoslangan fan bo’lib, inson faoliyatining turli sohalarida axborotni saqlash, qidirish, o’zgartirish, uzatish va qo’llash, axborotning umumiy xususiyatlari va tuzilmasini, uni tashkil etish usullari va qonuniyatlarini o’rgatadi.

Interfeys – monitorda akslangan, foydalanuvchi va dastur o’rtasidagi “muloqot”ni amalga oshiruvchi dastur qismi.

Kiritish- chiqarish qurilmasi- kompyuter va tashqi qurilmalar o’rtasida ma’lumot almashinuvini ta’minlab beruvchi qurilma.

Klaviatura – kompyuterga turli axborot va komandalar kirituvchi, harf**,** raqam va boshqa belgilar tasvirlangan klavishlar.

Kompakt disk – ma’lumotlarni optik usulda yozuvchi plastik disk.

Kompyuter – (ingl. hisoblagich**)** turli xajmdagi va ko’rinishdagi ma’lumotga ishlov beruvchi avtomatik qurilma.

Kontekst menyu – sichqonchaning o’ng tugmasi bosilishida akslanuvchi qo’shimcha xarakatlar ro’yhati.

Quti (korzina) – o’chirilgan fayllarni vaqtincha saqlash uchun maxsus katalog.

Mashina komandasi -ma’lum qoida asosida mikroprosessorga biror amalni bajarish uchun mo’ljallangan buyruq.

Magnit disklar – kompyuterning xotira qurilmasida qo’llanuvchi, dumaloq plastina yoki bir o’qda parallel joylashgan magnit qatlamli bir nechta plastina ko’rinishidagi axborot tashuvchi.

Menyu – dastur imkoniyatlariga ko’ra guruxlangan ro’yhat (satr); odatda menyu satri ilovaning yuqori qismida joylashtiriladi.

Sichqoncha (mishka) – kompyuterni boshqarish uchun qulay qurilma.

Operativ xotira- dasturda bajarilishi uchun vaqtincha foydalaniladigan xotira bo’lib dastur bajarilish tezligini aniqlaydi.

Operasion tizim (sistema) – kompyuter qurilmalari va dasturlari ishini ta’minlovchi, kompyuter ishga tushirilganida yuklanuvchi dastur. Masalan, **WINDOWS 98, WINDOWS XP, LINUX, UNIX, MACHINTOSH**.

Faylni ochish –faylni bajarish uchun ishga solish.

Prosessor- kompyuter qurilmalarini boshqaruvchi va berilganlarni qayta ishlovchi qurilma.

Palitra – ranglar majmuasi.

Uskunalar paneli – dasturning ma’lum buyruqlariga mos piktogrammalar akslangan panel.

Topshiriqlar paneli – Pusk tugmasi, tezkor yuklash paneli, ishlayotgan dastur sarlavhasi va b. akslangan Windows ishchi stoli qismi.

Papka – (katalog) ma’lum fayllar guruxi saqlanuvchi tashqi xotira qismi.

Piksel – kompyuter ekranidagi ma’lum rangli nuqta. Ekrandagi har bir tasvir piksellardan tashkil topgan.

Piktogramma – monitordagi biror ob’yektga (fayl, ilova va b.) mos bo’lgan kichik o’lchamdagi tasvir.

Tuzatish (Pravka) – faylga tuzatish kiritish: masalan, xujjatga qism kiritish yoki olib tashlash va x.

Dastur – ma’lum masalani yechish uchun kompyuterga beriladigan buyruqlar ketma-ketligi.

Printer – kompyuterda tayyorlangan turli xujjatlarni qohozga chop etuvchi qurilma.

Probel – bo’sh joyni bildiruvchi klavisha; matndagi so’zlarni ajratish uchun qo’llanadi.

Prosessor – kompyuterning barcha qurilmalarini boshqaruvchi elektron sxema.

Pusk tugmasi – Windows operasion tizimining imkoniyatlarini ochuvchi Pusk menyusi.

Registr- xotiraning tez murojaat qilish mumkin bo’lgan qismi bo’lib, xajmi operativ xotiradan kichik bo’ladi.

Ishchi maydon – piktogrammalardan holi ishchi stol qismi.

Ishchi stol – WINDOWS yuklangandan keyin monitorda tasvirlanuvchi ko’rinish.

Sistemali blok – korpus bilan himoyalangan elektron sxema va qurilmalar majmuasi; unda asosiy plata joylashadi.

Software – yumshoq qism, kompyuter dasturlari shunday nomlanadi; ularni oddiy ravishda o’chirish va boshqasiga almashtirish mumkinligi sababli ular shunday nomlangan.

CD-ROM –kompakt-diskdan axborot o’qish qurilmasi.

Struktura- foydalanuvchi tomonidan aniqlangan toifa.

Tayanch toifalar-C++ tilida mavjud toifalar (int , float, double, char).

Ovozni eshittirish qurilmasi – kolonka, naushnik; ular yordamida musiqani eshitish, kliplarda, qo’shiqlarda, kinofilmlarda tovushni tiklash, eshittirish mumkin.

Fayl – ma’lum nom bilan tashqi xotirada saqlangan ixtiyoriy axborot.

Xotira segmenti- operativ xotira qismi bo’lib, xajmi (o’lchami) o’zgaruvchan bo’ladi.

Hardware – qattiq qism, odatda kompyuterning barcha qurilmalari shunday ataladi.

Yorliq (Yarlik) –quyi chap burchagida ko’rsatkich aks ettirilgan tasvir; har bir yorliq biror bir fayl yoki papkaga mos keladi.

FOYDALANILGAN ADABIYOTLAR RO‘YXATI

1. Karimov I.A. Xavfsizlik va barqarorlik yo‘lida. 6-jild. Toshkent “O‘zbekiston”. 1998. 409 b.

2. Horstman C.S. C++ for Everyone, 2 edition-2011, 562 p

3. Herb Sutter. More Exceptional C++. 2007- 304 p.

4. Nazirov Sh.A., Qobulov R.V., Bobojanov M.R., Raxmanov Q.S. C va C++ tili. “Voris-nashriyot” MCHJ, Toshkent 2013, 488 b

5. Aripov M., Begalov B., Begimqulov U., Mamarajabov M. Axborot texnologiyalari. Toshkent: Noshir, 2009. -368 s.

6. Deitel P.J., Deitel H.M. C++. How to Program, 9 th Edition-2011.-1070 p.

7. Подбелский В., Фомин С. Программирование на языке Си. Учебное пособие. -2-издание.- М.: Финансы и статистика, 2004. -600 с.

8. Павловская Т., Шупак Ю. С++ Объектно-ориэнтированное программирование. Практикум. - СПб.: Питер, 2005-265с.

9. Романов Б. Практикум по программированию на С++: Учебное пособие. Новосибирск: НГТУ, 2006.- 432с.

10. “Dasturlash asoslari” fani laboratoriya ishlari uchun uslubiy qo‘llanma 1-qism. T.: “NISIM”, 2013.-148b.

11. Raxmanov Q.S., Mo‘minov B.B., Qosimova Sh.T., Mahmudov A.Z. “C/C++ da dasturlash” kursidan laboratoriya ishlari uchun uslubiy ko‘rsatma. TATU, Toshkent 2014y. 126 b.

12. Raxmanov Q.S., Qosimova Sh.T., Abidova Sh.B. “Informatika” fanidan laboratoriya ishlarini bajarish uchun uslubiy ko‘rsatmalar (Maxsus fakultet talabalari uchun). TATU, Toshkent 2015y. 156 b.

13. Рахманов К.С., Касымова Ш.Т., Гапурова А.А., Гулямова Д.Р. Сборник заданий и методических пособий по выполнению лабораторных работ по предмету “Программирование на С/С++” (част 3). ТУИТ, Ташкент – 2015, 148 с.

14.O‘zbekiston Respublikasi Axborot texnologiyalari va kommunikatsiyalarini rivojlantirish vazirining birinchi o‘rinbosari A.N.Fayzullayevning ma’ruzasidan. 2015 yil noyabr.