Ўзбекистон Республикаси

Олий ва ўрта махсус таълим вазирилиги

Наманган мухандислик – педагогика институти

П.Каримов, С.Ирисқулов, А.Исабаев.

Дастурлаш

Тошкент-2002

Сўз боши

Азиз ўқувчи! Энг аввал шуни айтиб ўтиш жоизки, сиз ажиб бир замонда яшамоқдасиз. Бу замоннинг энг муҳим хусусияти халкимизнинг истиқлол шароитида ҳаёт кечириётгани ва ижтимоий ҳаётнинг барча жабҳаларида чукур инқилобий ўзгаришлар содир бўлаётгандир. Республикамиз Президенти И.Каримов XXI асрни маънавият ва маърифат асри, илм-фан, маданият ва ахборотлар асри деб таърифлар экан, бу билан ҳозирги даврнинг муҳим белгиси ва хусусиятини ҳам белгилаб берганини пайқаб олиш қийин эмас.

Ўзбекистон дархақикат келажаги буюк давлат. Бироқ бу келажакнинг қай даражада яқин ёки узоклиги, мамалакатимизни қачон жаҳоннинг ривожланган давлатлари каторидан ўрин олиши, қолаверса ҳалқимизнинг эркин ва фаровон ҳаёт кечириши ҳал қилувчи даражада унинг ақлий салоҳиятига, ёшларнинг замонавий билимларини, фан, техника ва технологияларнинг энг сўнгги ютуқларини ўзлаштирганига боғлиқ.

Миллий истиқлол мафкураси, ўзининг мазмун-моҳиятига кўра, мамлакатимизнинг ҳар бир фуқароси онгига халқимизнинг дунёдаги ҳеч бир халқдан кам эмаслиги ва кам бўлмаслиги ғоясини сингдиришга қаратилгандир, десак хато бўлмас. Мана шу ғоя билан қуролланган ёшларимиз мамлакат тақдирини ҳал қиладилар, унинг буюк келажак сари қўяётган қадамларини тезлаштирадилар.

Мамалакатимизда қабул қилинган Кадрлар тайёрлашнинг миллий дастури замон талабларига тўла-тўкис жавоб бера оладиган, чинакам маърифатли, бозор муносабатлари шароитида ўз билим ва кўникмалари билан ватан истиқболи манфаатлари йўлида самарали фаолият кўрсата оладиган мутахассислар тайёрлашни назарда тутади.

Ҳозирги замонда инсон фаолиётининг бирон бир соҳасини ҳисоблаш техникаси (компьютерлар) ва ахборот технологияларисиз тассавур қилиб бўлмайди. Улардан оқилона ва самарали фойдалана олиш бугунги кунда ҳар бир ўқимишли ва зиёли инсон учун сув ва ҳаводай зарур бўлиб қолмоқда. Шу нуқтаи назардан қарайдиган бўлсак, қўлингиздаги ўқув қўлланмаси нақадар долзарб масала ва муаммоларни ҳал этишга бағишлангани ўз-ўзидан аён бўлади.

Ҳозирги кунда мамлакатимиз таълим тизимида янги босқич бўлган касб-ҳунар таълимини зарур меъёрий ҳужжатлар, замонавий техника ва технологиялар билан таъминлаш борасида бир қатор ишлар амалга оширилди ва оширилмоқда. Жумладан, Ўзбекистон Республикаси Вазирлар махкамасининг қарори билан «Ўрта махсус, касб-ҳунар таълимнинг давлат таълим стандарти» ва «Ўрта махсус, касб-ҳунар таълимнинг йўналиш фанлар давлат таълим стандарти» тасдиқланди. Уларда кўрсатиб ўтилганидек, ўрта махсус, касб-ҳунар таълим муассаса битирувчиларига фундаментал фанлар ва аниқ касб соҳаси доирасида назарий ва амалий билимларга эга бўлиш, компьютер ва телекоммуникация воситаларидан фойдалана олиш, ўқув фанлари бўйича олий таълим муассасаларида таҳсил олиш учун зарур бўлган билимлар мажмуасига эга бўлиш вазифаси қўйилган. Шу талаблардан келиб чиққан ҳолда, касб-ҳунар коллежлари учун «Программалаш» фанидан намунавий ўқув дастури ишлаб чиқилди ва Ўзбекистон Республикаси Олий ва ўрта махсус таълим вазирлигининг Ўрта махсус, касб-ҳунар таълими маркази томонидан тасдиқланди.

Бугунги кунда касб-ҳунар коллежлари ва улардаги таълим йўналишларининг замонавийлиги уларнинг компьютерлаштирилганлик даражаси билан белгиланмоқда. Информатика ва янги ахборотлар технологиялари жамиятимиз ижтимоий ва иқтисодий ҳаётига шиддат билан кириб келмоқда. Шу сабабли информатика фанининг сир-синоатларини ўзлаштириш давр талабига айланиб қолди.

Ўйлаймизки, юқорида кўрсатиб ўтилган дастур асосида тайёрланган ушбу қўлланма ўқувчиларга программалаш асосларини ўзлаштириш, компьютердан ўз касбий фаолиятларида бемалол ва самарали фойдалана олиш учун зарур бўлган малака ва кўникмаларни ҳосил қилишга хизмат қилади.

Албатта, мазкур қўлланма илк бор ёзилганлиги боис айрим камчиликлардан холи бўлмаслиги мумкин. Уни янада такомиллаштиришга қаратилган фикр мулоҳазаларни муалифлар мамнуният билан қабул қиладилар.

Проф. Т.Эргашев

I.БОБ. Алгоритмлар назарияси элементлари

1.1. Алгоритм тушунчаси ва унинг хоссалари

Алгоритм хозирги замон математикасининг энг кенг тушунчаларидан бири ҳисобланади.

Алгоритм (алгорифм) сўзи ўрта асрларда пайдо бўлиб, бу ўзбек мутафаккири Ал-Хоразмийнинг (783-855) ишлари билан европаликларнинг биринчи бор танишиши билан боғлиқдир. Бу ишлар уларда жуда чуқур таассурот қолдириб алгоритм (algorithmi) сўзини келиб чиқишига сабаб бўлдики, у Ал-Хоразмий исмининг лотинча айтилишидир. У пайтларда бу сўз арабларда қўлланиладиган ўнлик саноқ системаси ва бу саноқ системасида ҳисоблаш усулини билдирар эди. Шуни таъкидлаш лозимки, европаликлар томонидан араб саноқ системасининг Ал-Хоразмий ишлари орқали ўзлаштирилишига, кейинчалик ҳисоблаш усулларининг ривожланишига катта туртки бўлган.

Хозирги пайтда ўнлик саноқ системасида арифметик амалларни бажариш усуллари ҳисоблаш алгоритмларига соддагина мисол бўла олади холос. Хозирги замон нуқтаи назаридан алгоритм тушунчаси нимани ифодалайди? Маълумки, инсон кундалик турмушида турли-туман ишларни бажаради. Ҳар бир ишни бажаришда эса бир қанча элементар (майда) ишларни кетма-кет амалга оширишга тўғри келади. Мана шу кетма-кетликнинг ўзи бажариладиган ишнинг алгоритмидир. Аммо бу кетма-кетликка эътибор берсак, биз ижро этаётган элементар ишлар маълум қоида бўйича бажарилиши керак бўлган кетма-кетликдан иборат эканлигини кўрамиз. Агар бу кетма-кетликдаги қоидани бузсак, мақсадга эришмаслигимиз мумкин.

Масалан, шахмат ўйинини бошлашда шоҳни юра олмаймиз, чунки бу ўйин алгоритмида юришни бошқа бир шахмат доналаридан бошлаш керак ёки палов пишириш алгоритмида биринчи навбатда қозонга сув солиб кўрингчи, ош қандай бўлар экан. Берилган математик ифодани соддалаштиришда амалларнинг бажарилиш кетма-кетлигига эътибор бермаслик нотўғри натижага олиб келиши барчага маълум.

Демак ишни, яъни қўйилган масалани бажаришга майда элементар ишларни маълум кетма-кетликда ижро этиш орқали эришилади. Бундан кўриниб турибдики, ҳар бир иш қандайдир алгоритмнинг бажарилишидан иборатдир. Алгоритмни бажарувчи алгоритм ижрочисидир. Алгоритмнинг ижрочиси масалани қандай қўйилишига эътибор бермай, натижага эришиши мумкин. Бунинг учун у фақат аввалдан маълум қоида ва кўрсатмаларни қатъий бажариши шарт. Бу эса алгоритмнинг жуда муҳим хусусиятлардан биридир.

Умуман, алгоритмларни икки гуруҳга ажратиш мумкин. Биринчи гуруҳ алгоритмнинг ижрочиси фақат инсон бўлиши мумкин (масалан паловни фақат инсон пишира олади), иккинчи гуруҳ алгоритмларнинг ижрочиси ҳам инсон, ҳам ЭҲМ бўлиши мумкин (фақат ақлий мехнат билан боғлиқ бўлган масалалар). Иккинчи гуруҳ алгоримтларнинг ижрочисини ЭҲМ зиммасига юклаш мумкин. Бунинг учун алгоритмни ЭҲМ тушунадиган бирор программалаш тилида ёзиб, уни машина хотирасига киритиш кифоя.

Шундай қилиб, биз алгоритм деганда, берилган масалани ечиш учун маълум тартиб билан бажарилиши керак бўлган чекли сондаги буйруқлар кетма-кетлигини тушунамиз.

Бирор соҳага тегишли масалани ечиш алгоритмини тузиш алгоритм тузувчидан шу соҳани мукаммал билган ҳолда, қўйилган масалани чуқур таҳлил қилишни талаб қилади. Бунда масалани ечиш учун керак бўлган ишларнинг режасини туза билиш муҳим аҳамиятга эга. Шунингдек, масалани ечишда иштирок этадиган объектларнинг қайсилари бошланғич маълумот ва қайсилари натижалигини аниқлаш, улар ўртасидаги ўзаро боғланишни аниқ ва тўла кўрсата билиш, ёки программа тузувчилар тили билан айтганда, масаланинг маълумотлар моделини бериш лозим.

Берилган масала алгоритмини ёзишнинг турли усуллари мавжуд бўлиб, улар қаторига сўз билан, блок-схема шаклида, формулалар, операторлар ёрдамида, алгоритмик ёки программалаш тилларида ёзиш ва хоказоларни киритиш мумкин.

Энди бирор усулда тузилган алгоритмнинг айрим хоссалари ва алгоритмга қўйилган баъзи бир талабларни кўриб чиқайлик.

1.Алгоритм ҳар доим тўлиқ бир қийматлидир, яъни уни бир хил бошланғич қийматлар билан кўп марта қўллаш ҳар доим бир хил натижа беради.

2.Алгоритм биргина масалани ечиш қоидаси бўлиб қолмай, балки турли-туман бошланғич шартлар асосида маълум турдаги масалалар тўпламини ечиш йўлидир.

3.Алгоритмни қўллаш натижасида чекли қадамдан кейин натижага эришамиз ёки масалани ечимга эга эмаслиги ҳакидаги маълумотга эга бўламиз.

Юқорида келтирилган хоссаларни ҳар бир ижрочи ўзи тузган бирор масаланинг алгоритмидан фойдаланиб текшириб кўриши мумкин. Масалан:

$$a\,x^2 + bx + c = 0$$

квадрат тенгламани ечиш алгоритми учун юқорида санаб ўтилган алгоритмнинг хоссаларини қуйидагича текшириб кўриш мумкин.

Агар квадрат тенгламани ечиш алгоритми бирор усулда яратилган бўлса, биз ижрочига бу алгоритм қайси масалани ечиш алгоритми эканлигини айтмасдан a, b, c ларнинг аниқ қийматлари

учун бажаришни топширсак, у натижага эришади ва бу натижа квадрат тенгламанинг ечими бўлади. Демак, алгоритмни ижро этиш алгоритм яратувчисига боғлиқ эмас.

Худди шунингдек a, b, c ларга ҳар доим бир хил қийматлар берсак, алгоритм ҳар доим бир хил натижа беради, яъни тўлиқдир.

Яратилган бу алгоритм фақатгина битта квадрат тенгламани ечиш алгоритми бўлиб қолмай, балки у **a, b, c** ларнинг мумкин бўлган барча қийматлари учун натижа ҳосил қилади, шу турдаги барча квадрат тенгламаларнинг ечиш алгоритмидир.

Алгоритмнинг охирги хоссаси ўз-ўзидан бажарилади, яъни квадрат тенгламани ечиш албатта чекли қадамда амалга оширилади.

Программа тузувчи учун ЭҲМнинг иккита асосий параметри ўта муҳимдир: ҳисоблаш машинаси хотирасининг хажми ва машинанинг тезкорлиги. Шунингдек, алгоритм тузувчидан икки нарса талаб қилинади. Биринчидан, у тузган программа машина хотирасида энг кам жой талаб этсин, иккинчидан, энг кам амаллар бажариб масаланинг натижасига эришсин. Умуман олганда, бу икки талаб бир-бирига қарама-қаршидир, яъни алгоритмнинг ишлаш тезлигини ошириш, алгоритм учун керакли хотирани оширишга олиб келиши мумкин. Бу хол, айниқса мураккаб масалаларни ечиш алгоритмини тузишда яққол сезилади. Шунинг учун ҳам бу икки параметрнинг энг мақбул ҳолатини топишга ҳаракат қилиш керак.

1.2. АЛГОРИТМНИ ТАВСИФЛАШ УСУЛЛАРИ

Алгоритм сўзлар, математик формулалар, алгоритмик тиллар, геометрик схемалар, программалаш тиллари ва бошқалар ёрдамида тавсифланади.

Алгоритмнинг сўзлар ёрдамида берилишига, тавсифланишига мисол тариқасида лифтда керакли қаватга кўтарилиш алгоритмини келтириш мумкин. Бу куйидагича кетма-кетликда бажарилади:

1. Лифтга киринг.
2. Керакли қават тартиб сонига мос тугмачани босинг.
3. Лифтни ҳаракатга келтиринг.
4. Лифт тўхташини кутинг.
5. Лифт эшиги очилгандан кейин ундан чиқинг.

Алгоритм математик формулалар ёрдамида тавсифланганда ҳар бир қадам аниқ, формулалар ёрдамида ёзилади. Мисол тариқасида

$$ax^2 + bx + c = 0 \quad (a \neq 0)$$

квадрат тенглама ечимлари x_1, x_2 ни аниқлаш алгоритмини кўриб чиқайлик.

1. *a, b, c* коэффициентлар қийматлари берилсин.
2. $D = b^2 - 4ac$ дискриминант ҳисоблансин.

3. D<0 бўлса, тенгламанинг ҳақиқий ечимлари йўқ. Фақат ҳақиқий илдизлар изланаётган бўлса, масала ҳал бўлди.

4. Dқ0 бўлса, тенглама иккита бир-бирига тенг, яъни каррали ечимга эга бўлади ва улар формулалар билан ҳисобланади. Масала ҳал бўлди.

5. D>0 бўлса, тенглама иккита ҳақий ечимга эга, улар

$$x_1 = (-b - \sqrt{D})/2a \quad ва \quad x_2 = (-b + \sqrt{D})/2a$$

формулалар билан ҳисобланади. Яни масала ҳал бўлди.

Шундай қилиб, квадрат тенглама ҳақиқий ечимларини аниқлашда:

1. “Тенгламанинг ҳақиқий ечимлари йўқ” матни;

2. “Тенглама каррали ечимга эга x_1қx_2” матни ва x_1, x_2 қийматлари;

3. “Тенглама иккита ечимга эга” матни, x_1 ва x_2 қийматлари натижалар бўлади.

Алгоритмик тиллар — алгоритмни бир маъноли тавсифлаш имконини берадиган белгилар ва қоидалар мажмуидир. Ҳар кандай тиллардагидек улар ҳам ўз алифбоси, синтаксиси ва семантикаси билан аниқланади.

Бизга ўрта мактабдан маълум бўлган, академик А. П. Ершов раҳбарлигида яратилган, ЭҲМсиз алгоритмлашга мўлжалланган алгоритмик тизим алгоритмик тилнинг наъмунасидир. Алгоритмик тилга мисол сифатида яна алгоритмларни белгили операторлар тизими шаклида тавсифлашни ҳам кўрсатиш мумкин. Бу тиллар одатдаги тилга ўхшаш бўлиб, ЭҲМда бевосита бажаришга мўлжалланмаган. Улардан мақсад алгоритмни бир хил шаклда ва тушунарли қилиб, таҳлил қилишга осон қилиб ёзишдир.

Алгоритмларни геометрик схемалар ёрдамида тавсифлаш кўргазмали, шу сабабли тушунарлирок, бўлгани учун кўп қўлланилади. Бунда хар бир ўзига хос операция алохида геометрик шакл (блок) билан тавсифланади ва уларнинг бажарилиш тартиби, улар орасидаги маълумотлар узатилиши ва йуналиши блокларни бир-бири билан кўрсаткичли тўғри чизиқлар ёрдамида туташтириб кўрсатилади. Алгоритмнинг геометрик схемасига унинг блок-схемаси дейилади.

Блокларга мос геометрик шакллар, уларнинг ўлчамлари ва улар ёрдамида блок-схемаларни чизиш қоидалари давлат стандартларида берилган. 1-жадвалда энг кўп ишлатиладиган блоклар шакли ва уларнинг маъноси келтирилган. Бу давлат стандартларига кўра блокларни туташтирувчи тўғри чизиқўтар ёзув текислигига вертикал ёки горизонтал ҳолатда бўлиши керак, яъни уларни оғма чизиқлар билан туташтириш тақикланади. Блокларни бажариш табиий ёзиш тартибида бўлса, яъни юқоридан пастга ёки чапдан ўнгга бўлса, туташтирувчи чизиқ, кўрсаткичсиз бўлиши мумкин.

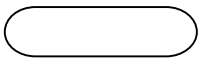
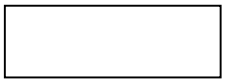
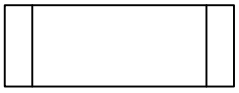
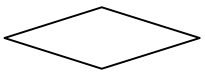
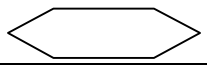
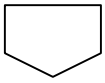
Бошқа барча ҳолларда маълумот оқими йўналишини кўрсатувчи кўрсаткич қўйилиши шарт. Блокнинг тартиб сони туташтирувчи чизикдан чапга, алоҳида ажратилган бўш жойга қўйилади. Чизиқўтарнинг бирлашган жойи йирикроқ нуқта ёрдамида кўрсатилади. Блокда кўзда тутилган операция унинг ичига ёзиб қўйилади. Схемалар давлат стандарти форматларида бажарилади.

Амалда ечиладиган масалалар ва демак, алгоритмлар турлари ҳам жуда кўп бўлишига қарамасдан улар асосан беш хил : чизиқли, тармоқланувчи, циклик, итерацион ва чексиз такрорланувчи тузилишларда бўлади деб айтиш мумкин.

Агар мураккаб масалалар алгоритмларининг блок-схемасини бир бино десак, бу тузилишдаги алгоритмлар уни ташкил қилувчи ром, ғишт, тўсин, устун ва бошқаларини ифодалайди деб айтиш мумкин. Ҳар кандай мураккаб бино ана шу ашёлардан қурилганидек, мураккаб алгоритмлар ҳам юқоридагидек схемалардан тузилади. Аслида охирги учта тузилишдаги алгоритмларни битта ном билан такрорлаш алгоритмлари деб аташ мумкин. Аммо уларнинг ҳар бири ўзига хос бўлганлиги учун алоҳида номланади.

(1-жадвал)

Асосий блокларнинг шакллари ва уларнинг вазифалари

Шакл номи	Шакл	Вазифаси
Ишга тушириш, тўхтатиш		Бошлаш, тамомлаш, маълумотларни қайта ишлаш жараёни ёки программа бажарилишини тўхтатиш
Киритиш-чиқариш		Маълумотларни қайта ишлашга (киритиш) ёки қайта ишлаш натижаларини акслантиришга (чиқариш) яроқли ҳолга келтириш
Жараён		Бажарилиши натижасида маълумотлар-нинг қиймати, тасаввур шакли ёки ўрнини ўзгартирадиган амал ёки амаллар гуруҳи
Аввалдан маълум жараён		Илгари тузилган ва алоҳида тавсифланган алгоритм ва программадан фойдаланиш
Ечим		Ўзгарадиган шартга боғлиқ, ҳолда алгоритм ёки программани бажариш йуналишини танлаш
Модификация (турлаш)		Программани ўзгартирадиган буйруқ ёки буйруқлар гуруҳини бажариш
Ҳужжат		Маълумотларни босма қоғозига чиқариш
Боғловчи		Маълумотлар оқимининг узилган жойларини туташтириш
Бетлараро боғловчи		Турли варақда жойлашган алгоритм ва программа бўлаклари орасидаги боғланишни қўрсатиш

Изоҳ		Схема элементлари ва тушунтириш ўртасидаги боғланиш

1.3. ЧИЗИҚЛИ ТУЗИЛИШДАГИ АЛГОРИТМЛАР

Чизиқли тузилишдаги алгоритмларда кўрсатмалар ёзилиш тартибида бажарилади. Уларнинг блок-схемалари ишга тушириш, тўхтатиш, киритиш-чиқариш жараён ва аввалдан маълум жараён блоклари ёрдамида тузилиб, бир чизиқ бўйлаб кетма-кет жойлашган бўлади.

Чизиқли тузилишдаги алгоритмни тузиш масалани ечиш учун керак бўладиган бошланғич маълумотларни ташкил қилувчи ўзгарувчилар номи, уларнинг тури ва ўзгариш кўламини аниқлашдан бошланади. Кейин оралиқ ва якуний натижалар ўзгарувчиларининг номлари, турлари ва мумкин бўлса ўзгариш кўламини аниқлаш керак. Энди алгоритм мана шу бошланғич маълумотларни қандай қайта ишлаб оралиқ ва якуний натижаларни олиш кераклигини аниқлашдан иборат бўлади.

М и с о л . Томонлари мос равишда *a, Ь, c* бўлган *ABC* учбурчак юзини ҳисоблаш алгоритмини тузайлик.

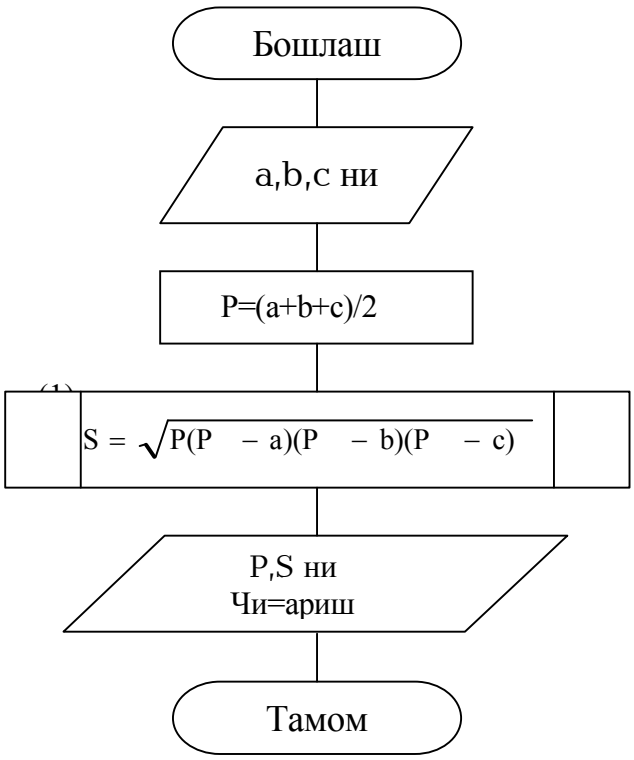
Томонлари маълум бўлганда ABC учбурчакнинг юзи

$$s = \sqrt{p(p-a)(p-b)(p-c)}$$

Герон формуласи билан ҳисобланади. Бунда

Рқ(аҚЬҚс)F2 (2)

учбурчак ярим периметри.



1- Расм.

1. Бошланғич маълумотлар: *a, Ь, c* учбурчак томонлари. Шунинг учун *a, b, c ∈R* ва *a>0, Ь>0, c>0*, яъни *a, Ь, c — ўзгарувчилар* номи; улар ҳақиқий сон қийматлар қабул қилади. Акс ҳолда кесма узунлиги бўлмаган бўларди. Яна, бу уч сон учбурчак томонларини ифода қилиши учун уларнинг исталган бири қолган иккитаси йиғиндисидан катта бўлмаслиги, яъни

a<bҚc, b<aҚc, c<aҚb (3)

шартлар бажарилиши керак. Шундай қилиб, ўзгариш кўлами (3) муносабатлар билан аниқланар экан.

2. Натижалар: (1) формула билан учбурчак юзини ҳисоблаш учун унинг ярим периметрининг қиймати керак. Демак, *P ўзгарувчининг* қиймати оралиқ маълумот бўлади. Юқоридаги шартларда *P∈ R* ва

P>Ь. Якуний натижа: *S*— учбурчак юзи. У *S∈R* ва S>0 қийматлар қабул қилади.

Шундай қилиб, ихтиёрий АВС учбурчак юзини ЭҲМда ҳисоблаш ва босмага (ёки Дисплей экранига) чиқариш:

*1) а, Ь, с қ*ийматларини ЭҲМ хотирасига киритиш:

2) *P қ*ийматини (2) формула билан ҳисоблаш:

3) *S қи*йматини (1) формула билан ҳисоблаш;

4) *P* ва *S* қийматларини босмага чиқариш операцияларидан иборат бўлади(1-расм).

Ҳар қандай алгоритм блок-схемаси ишга тушириш блокидан бошланади. Уни ЭҲМни ишга тайёрлаш, бошланғич маълумотларни аниқлаш ва тайёрлаш деб тушуниш керак. Ҳисоблашларнинг тугаганлиги ана шундай геометрик шакл билан кўрсатилади. Шунинг учун расмдаги 1- ва 6-блоклар ичига мос келган операциялар ҳақида маълумот ёзиб куйилган.

Бошланғич маълумотларни ЭҲМга ҳар-хил қурилмалардан киритиш мумкин. Аниқ биттасини танлаб олиш иш шароитига боғлик,. Шунинг учун умумий киритиш-чиқариш блокларидан фойдаланилади (2- ва 5-блоклар).

Учинчи блокда бевосита ҳисоблаш жараёни, тўртинчи блокда эса квадрат илдиздан чиқариш учун тузилган кичик алгоритм (ёрдамчи алгоритм) дан фойдаланиш - аввалдан маълум жараён кўзда тутилган. Алгоритм кўрсатмалари ёзилиш тартибида кетма-кет бажарилади. Маълумотлар блокдан-блокка юқоридан пастга узатилади. Шунинг учун уларни туташтирувчи чизиқўтарга кўрсаткичлар кўйилмаган.

Алгоритмдан фойдаланувчи бошланғич маълумотларни (3) шартларни бажариладиган қилиб олиши керак. Акс ҳолда алгоритмни бажариб бўлмайди. У натижавийлик хоссасига эга бўлмайди.

1.4.Тармоқланувчи тузилишдаги алгоритмлар

Турли масалаларни ечганда кўрсатмаларни бажариш тартиби бирор бир шартнинг бажарилишига боғлиқ ҳолда бажарилади, яъни алгоритм тармоқланади. Тармоқланиш “Ечим” блоки орқали ифодаланади.

Шартни текшириш натижаси фақат икки ҳил бўлганда: бажарилган ҳол учун “Ҳа” (ёки “Қ”), бажарилмаган ҳол учун “Йўқ” (ёки “-“) белгилари қўйилади (мантиқий шарт, 2-а расм).

Тармоқланиш математик ифода қйматининг ишораси бўйича бўлганда (арифметик шарт): “>” - мусбат, “<” - манфий ва “қ” - нолга тенг белгилар куйилади *(2-б* расм),

Текшириш натижаси учдан кўп бўлганда 2-*в* расмдагидек тармоқланиш бўлиши мумкин.

3-м и с о л . ax^2*ҚbxҚсқ0* тенглама ечимларининг аниқлаш алгоритмини ва унинг блок-схемасини тузинг.

Бунда *a, Ь, c* ∈ R ва тенгламанинг барча ечимлари изланаётган бўлсин. Агар бу сонлар берилган бўлса, тенглама илдизлари қуйидаги тартибда ҳисобланади:

1. Тенгламани квадрат тенглама эканлиги текширилади. Бунинг учун *aқ0* шарт текширилади. Агар бу шарт бажарилса тенглама квадрат тенглама бўлмайди, 4-бандга ўтилади.

2. Тенглама дискриминанти *Dқb²—4ac* ҳисобланади.

3. *D* нинг ишораси текширилади. Агар *D>0* бўлса, тенгламани иккита хақиқий илдизлари бор, улар

$$x_1 = (-b - \sqrt{D})/2a \quad ва \quad x_2 = (-b + \sqrt{D})/2a$$

формулалар билан ҳисобланади.

Натижалар: 1-маълумот: “Тенглама иккита ҳақиқий ечимга эга” матн ва x_1, x_2 ларнинг қийматлари босмага чиқарилади. Масала ечилди.

Агар *D*< O бўлса, тенглама

$$x_1 = (-b - \sqrt{|D|}i)/2a \quad ва \quad x_2 = (-b + \sqrt{|D|}i)/2a$$

формулалар билан ҳисобланадиган иккита кўшма комплекс илдизга эга бўлади. Илдизларнинг ҳақиқий қисмини x₁қ—bF2a, мавҳум қисмини

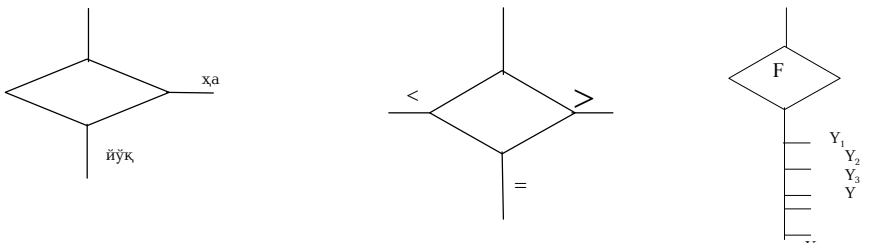
$x_2 = \sqrt{|D|}i/2a$ билан белгиланса, x_1, x_2 ни ҳисоблаб, x_1 ё x_2i комплекс сонлари ҳосил қилинади.

Натижалар: 3-маълумот: “Тенглама комплекс ечимга эга, ҳақиқий қисми қ... “ мавҳум қисми қ”, матни ва x_1 x_2 ларнинг қийматлари босмага чиқарилади. Бу ҳолда ҳам масала ечилди.

4. Тенгламанинг чизиқлилиги *Ьқ0* шартни текшириш билан аниқланади. Агар бу шарт бажарилса, тенглама чизиқли эмас, 6-бандга ўтади.

5. Тенгламанинг ечими: хқ-cFb. Натижалар: 6-маълумот: “Тенглама чизиқли *хқ”* матни *ва x* нинг қиймати босмага чиқарилади. Масала ечилди.

6. Агар *cқ0* бўлса, тенглама $0x^2Қ0xҚ0қ0$ айниятга айланади ва ихтиёрий $x \in R$ сон унинг ечими бўлади. Натижа: 4-маълумот: “Тенглама чексиз кўп ечимга эга” матни босмага чиқарилади. Агар $c \neq 0$ бўлса, тенглама маънога эга бўлмайди. Бу ҳолда натижа: 5-маълумот:

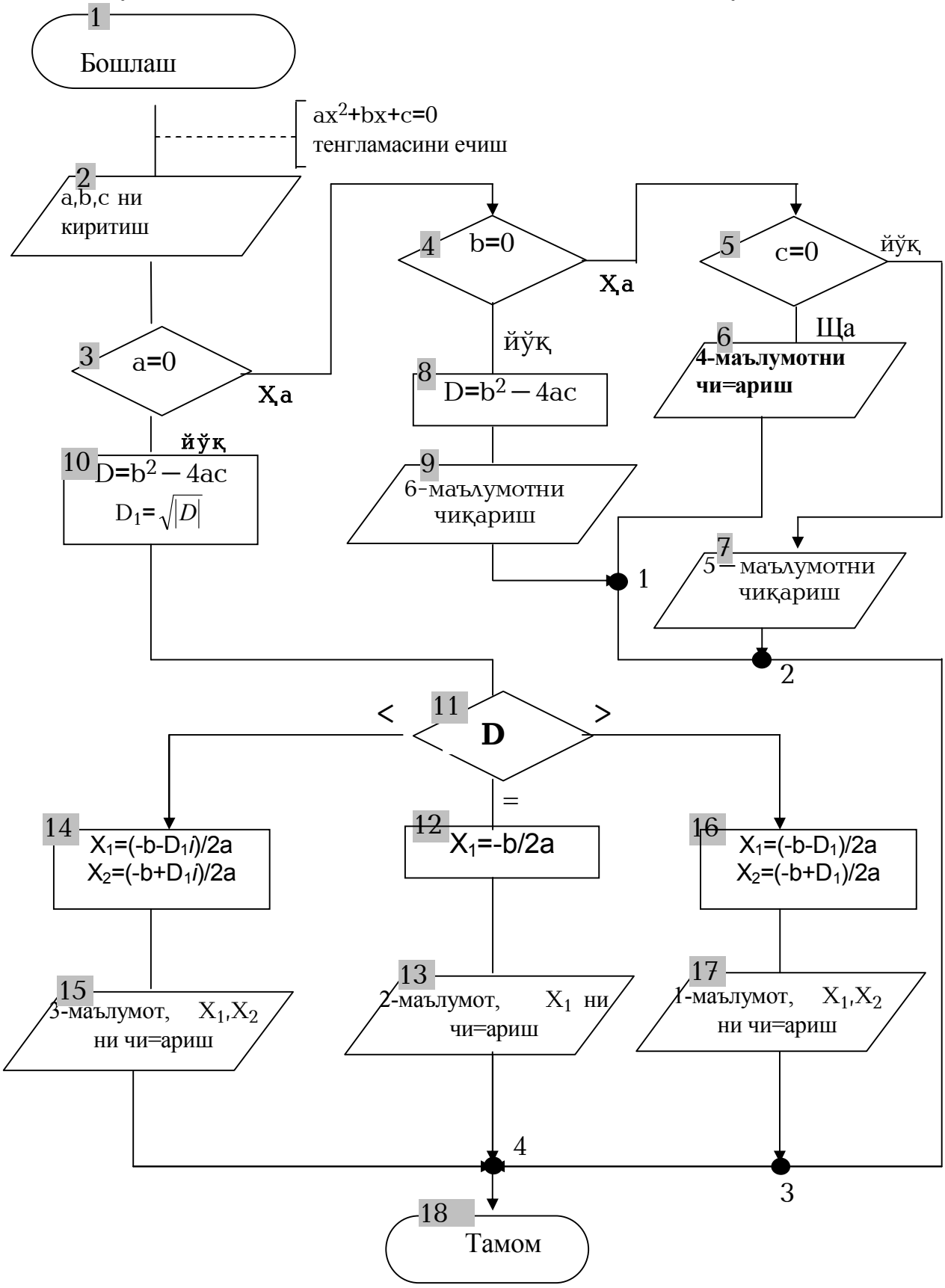


а) б) в)

2- Расм

“Тенглама ечимга эга эмас” матни бўлади.

Шундай қилиб, ҳамма ҳолларда ҳам маълум бир натижага



3-расм

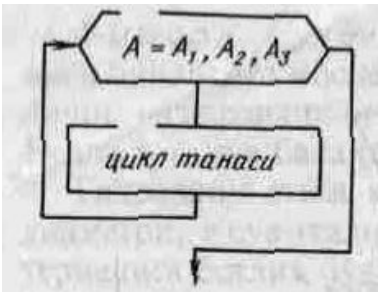
келинади. Бу алгоритмнинг блок-схемаси 3-расмда келтирилгандек

бўлади. Учинчи, бешинчи ва тўртинчи “ечим” блокларида мантиқий шартлар мос ҳолда *a, Ь, c* коэффициентларнинг нолга тенглилиги текширилмоқда. Бу шартларни текшириш натижаси 2 хил бўлиши мумкин: бажарилган (жавоб - ҳа) ёки бажарилмаган (жавоб - йўқ). Алгоритм икки тармоққа ажралади. Булардан фарқли ўларок, 11 ечим блокида *D* ўзгарувчининг ишораси текширилмоқда. Натижа уч хил бўлиши мумкин. Схеманинг 1,2,3,4 нуқталарида маълумотлар оқимининг қўшилиши рўй бермоқда. Шунинг учун бу нуқталар ажратиб кўрсатилган. Маълумот оқимини кўрсатувчи чизиқ «синган» ва оқим ўнгдан чапга йўналган ҳолларда туташтирувчи чизиқлар кўрсаткич билан белгиланган.

1.5.ТАКРОРЛАШ АЛГОРИТМЛАРИ

Такрорлаш алгоритмлари цикл танаси деб номланувчи кўп марта такрорланадиган қисмни ўз ичига олади. Такрорлаш бирор шарт бажарилгунча давом этади. Юқорида айтилганидек циклик, итерацион ва чексиз давом этувчи такрорлаш алгоритмлари фарқланади.

Циклик тузилишдаги алгоритмлар такрорлаш ўзгарувчиси (цикл параметри) арифметик прогрессия турида ўзгарганда ҳосил бўлади. Алгоритмнинг блок-схемасида улар модификация блоки билан берилади (4-расм). Расмда *A*- цикл ўзгарувчисининг номи, A_1 - цикл ўзгарувчисининг бошланғич A_2 - охирги қийматлари, A_3 - цикл ўзгарувчисининг ўзгариш қадами (цикл рақами ёки оддийгина қадам дейилади). Бунда A_1, A_2, A_3 ихтиёрий сон ёки ифода бўлиши мумкин. Агар $A_3>0$ бўлса, $A_1<A_2$ бўлиши, $A_3<0$ бўлганда эса $A_1>A_2$ бўлиши керак. A_3қ1 бўлганда уни блок ичидаги ёзувда кўрсатмасликка келишилган.



4-расм

Алгоритмнинг бажарилиши:

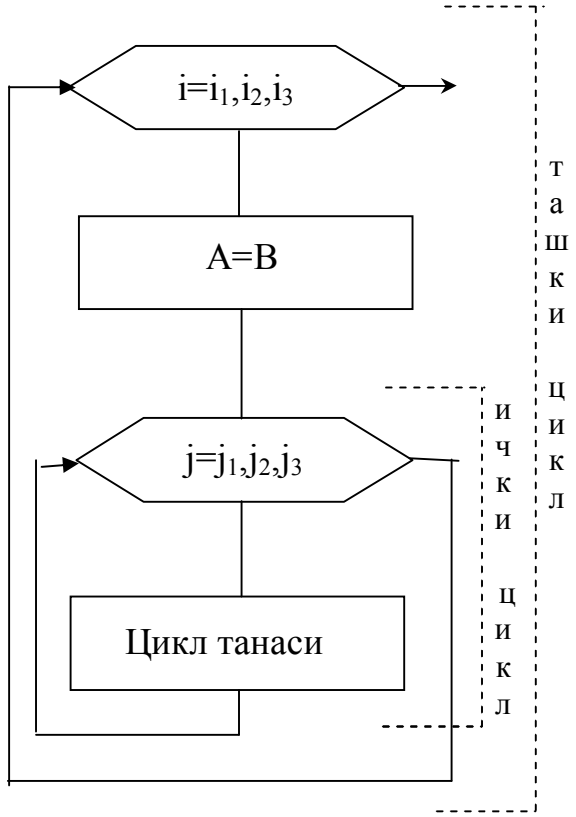
1) $AқA_1$ қилиб олинади;

2) Цикл танасига кирувчи амаллар бажарилади, бунда бирор шарт бажарилганда цикл ташқарисига чиқиб кетиш мумкин;

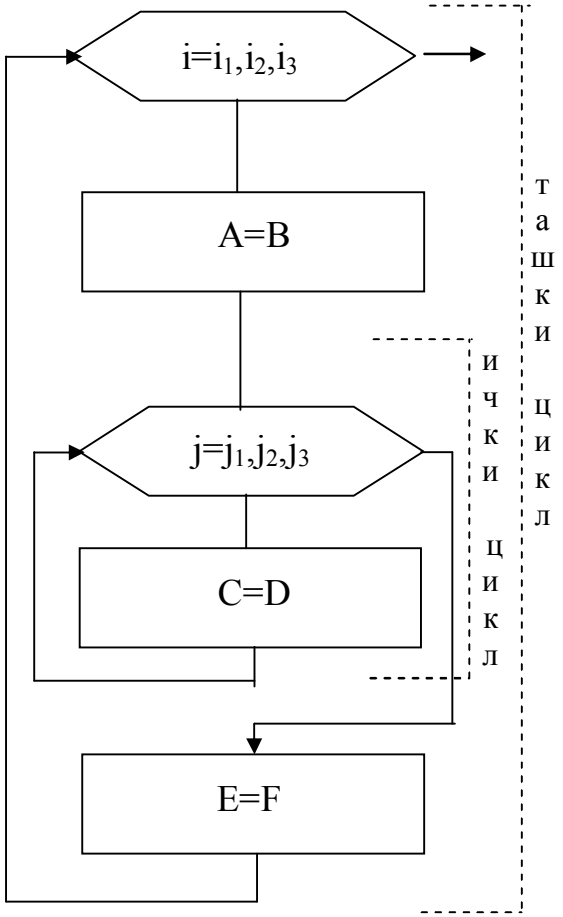
3) $AқAҚA_3$ қилиб олиниб: $A_3>0$ бўлганда $A<A_3$, ёки $A_3<0$ бўлганда $A>A_2$ такрорлаш шарти текширилади.

Агар такрорлаш шарти бажарилса, цикл танасидаги амаллар унинг ўзгарувчисининг янги қийматида бажарилади. Бунда улар цикл ўзгарувчисининг қийматига боғлик, бўлиши ҳам, бўлмаслиги ҳам мумкин.

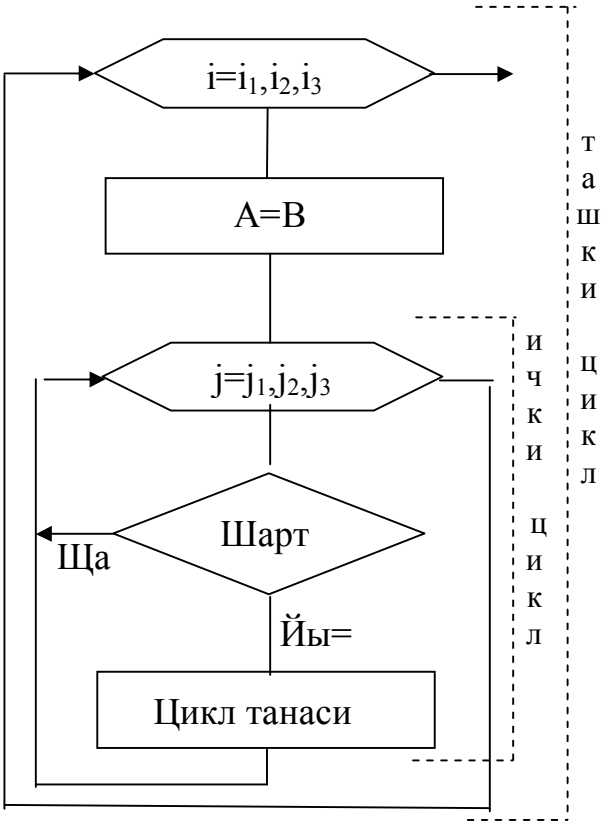
Циклик тузилишдаги алгоритмда такрорлаш сони аввалдан берилган бўлиши ёки у *пқ*(A2—A1) FA3+* формула билан ҳисобланади. Бунда *•+ белги соннинг бутун қисмини ифодалайди.


i=i1,i2,i3
A=B
j=j1,j2,j3
Цикл танаси
ички цикл
ташки цикл

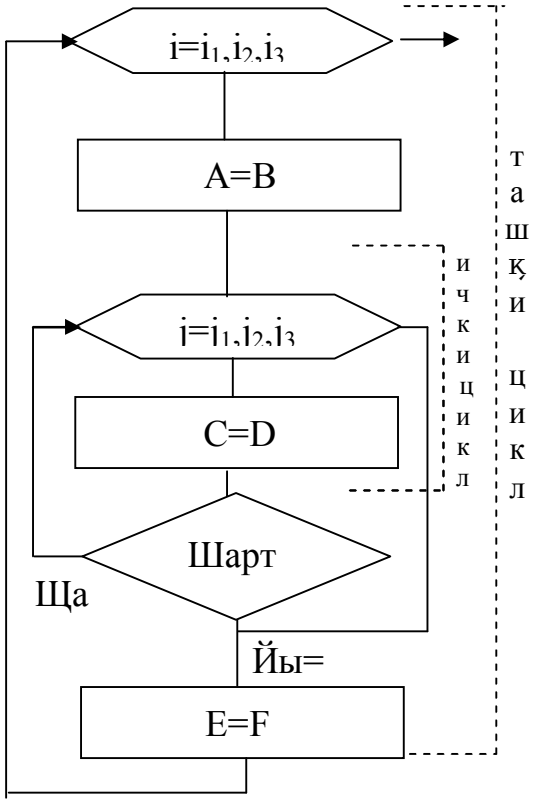
а)


i=i1,i2,i3
A=B
j=j1,j2,j3
C=D
E=F
ички цикл
ташки цикл

б)


i=i1,i2,i3
A=B
j=j1,j2,j3
Шарт
Ща
Йы=
Цикл танаси
ички цикл
ташки цикл

в)


i=i1,i2,i3
A=B
i=i1,i2,i3
C=D
Шарт
Ща
Йы=
E=F
ички цикл
ташқи цикл

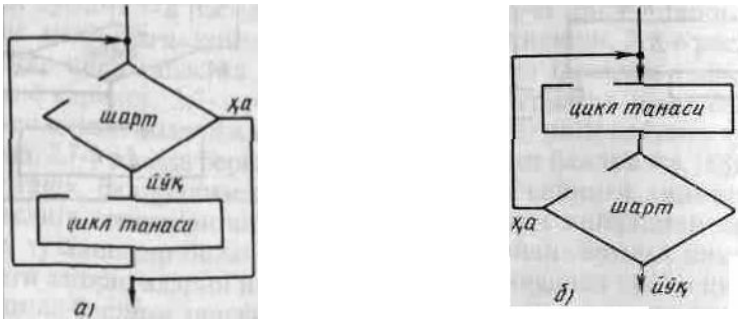
г)

5-расм

Ичма-ич жойлашган цикллар. Бир цикл танасида бошқа бир ёки бир неча цикллар жойлашган алгоритмлар ҳам бўлади (5-расм).

Расмда *j* ўзгарувчили цикл *i* ўзгарувчили цикл ичига жойлашган бўлиб 5-*а* расмда *j* бўйича цикл тугаши билан *i* нинг навбатдаги қийматига ўтилиши, 5-б расмда *i* ўзгарувчили цикл танасига *j* бўйича циклдан ташқари амаллар ҳ*ам* кириши, 5-*в* расмда берилган шарт бажарилмаса, ҳеч қандай амал бажармасдан *j* нинг кейинги қийматига утиш, 5-*г* расмда берилган шарт бажарилса, j бўйича цикл тугаши, бажарилмаса *j* нинг кейинги қийматига ўтиш кераклиги тасвирланган. Ичма-ич жойлашган цикл турлари бу тузилишлар билан тугамайди. Бундай циклик тузилишдаги алгоритмларни ишлаб чиқанда ташқи циклдан ички цикл бошини ташлаб ўтиб, унинг ичига кириш мумкин эмаслигини унутмаслик керак.

Итерацион алгоритмлар. Кўп ҳолларда амалларни неча марта такрорлашни аввалдан аниқлаб бўлмайди. Такрорлаш маълум бир шарт бажарилгунча давом этади. Бундай ҳолларда алгоритмни 5-расмда кўрсатилгандек ташкил қилиш мумкин.



6-расм

Такрорлаш ўзгарувчиси ўзгармас қадам билан ўзгармайди. 6-а расмда такрорлаш шарти аввалдан текшириладиган, 6-б расмда эса кейин текшириладиган тузилишдаги итерацион алгоритмнинг схемаси кўрсатилган (италянча iteratixo - такрорлаш сўзидан).

Биринчи ҳолда цикл танасидаги амаллар бирор марта ҳам бажарилмаслиги мумкин бўлса, иккинчи ҳолда эса камида бир марта бажарилади. Албатта циклик тузилишдаги алгоритмни ҳам итерацион тузилишда тасвирлаш мумкин. Лекин модификация блокини ишлатганда блок-схема ихчамрок, бўлади. Такрорлаш алгоритмларини тузганда цикл ташқарисидан унинг танасига кириш мумкин эмаслигини унутмаслик керак. Чунки бунда цикл ўзгарувчисининг қиймати аниқланмаган бўлиб қолади. Аксинча, цикл танасидан чиқиш мумкин. Бунда цикл ўзгарувчисининг қиймати ундан чиққан пайтдагидек бўлади.

Чексиз такрорланувчи алгоритм. ЭҲМ билан мулоқот олиб боришга имкон берувчи операторлари мавжуд бўлган программалаш тилларида программалашда ишлатилади (масалан, БЕЙСИКда). Бунда такрорлаш фойдаланувчи томонидан тугатилгунча давом этади.

1-м и с о л. y қ $log_2(ex^2ĶlxĶk)$ функциянинг x∈ *a, b+ кесмадаги қийматларини h қадам билан ҳисоблаш алгоритмини тузамиз.

Бошланғич маълумотлар: a, b, e, I, k, h ўзгарувчиларнинг қийматлари бўлади. Улар $a, b, e, I, k, h \in R$ бўлиб, кўлами $ex^2ĶlxĶk>0$ шарт билан аниқланади.

Тузилган алгоритм: 1. y нинг қийматига тенг бўлган хақиқий сон; 2. «Логарифм остидаги ифода қиймата мусбат эмас» матнли натижаларга эга бўлади. Бу натижаларни олишнинг икки хил алгоритмини кўриб чиқайлик.

1-алгоритм. Модификация блокидан фойдаланилган ҳол. Ҳисоблаш кетма-кетлиги куйидагича бўлади:

1) $h, a, Ъ, e, I, k$ қийматларини ЭҲМ хотирасига киритиш;

2) x ўзгарувчи бўйича a дан b гача h қадам билан цикл тузиш;

3) $Aқex^2ĶlxĶk$ ни ҳисоблаш;

4) Агар $A>0$ шарт бажарилса, уқ$\log_2$A ни ҳисоблаш ва 6-қадамга ўтиш;

5) «Логарифм остидаги ифода мусбат эмас» матни 1-ахборотни чиқариш ва x нинг бошқа қиймати учун ҳисоблашга ўтиш;

6) x ва y қийматларини босмага чиқариш ва 3-қадамга ўтиш;

Бу алгоритмнинг блок-схемаси 6-расмдагидек бўлади.

2-алгоритм. Модификация блокидан фойдаланилмаган ҳол.

Натижаларни олиш кетма-кетлиги куйидагича:

1) h, a, b, e, I, k қийматларини ЭҲМга киритиш;

2) хқа қилиб олиш;

3) $Ақ\ ex^2ĶlxĶk$ ни ҳисоблаш;

4) $A>0$ шартни текшириш. Агар бу шарт бажарилса, 5-, бажарилмаса 7-қадамга ўтиш;

5) $уқlog_2A$ ни ҳисоблаш;

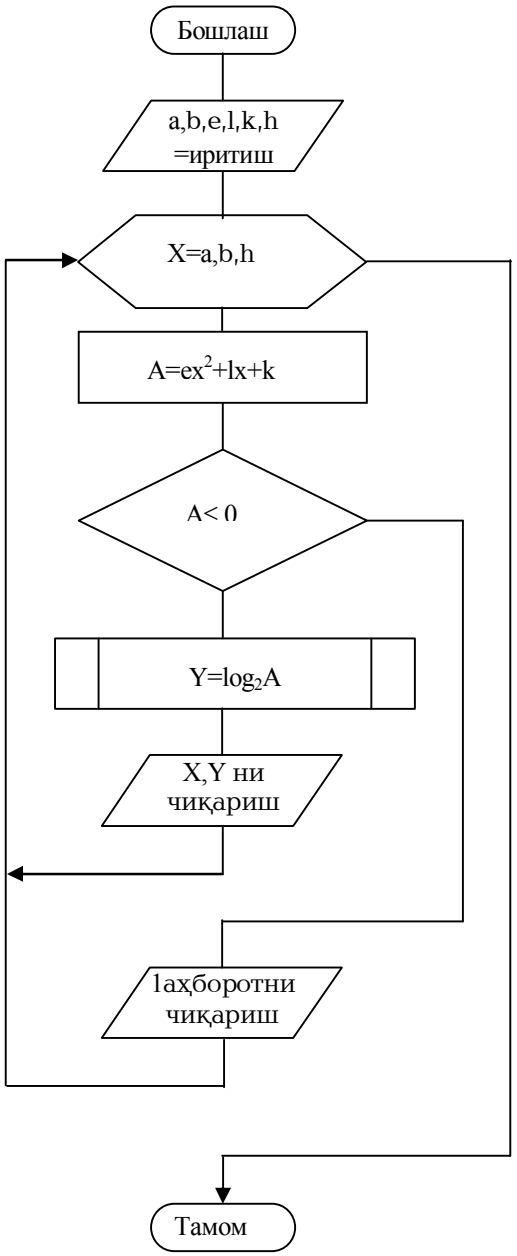
6) x, y қийматларини босмага чиқариш ва 8-қадамга ўтиш;

7) 1 -ахборотни босмага чиқариш;

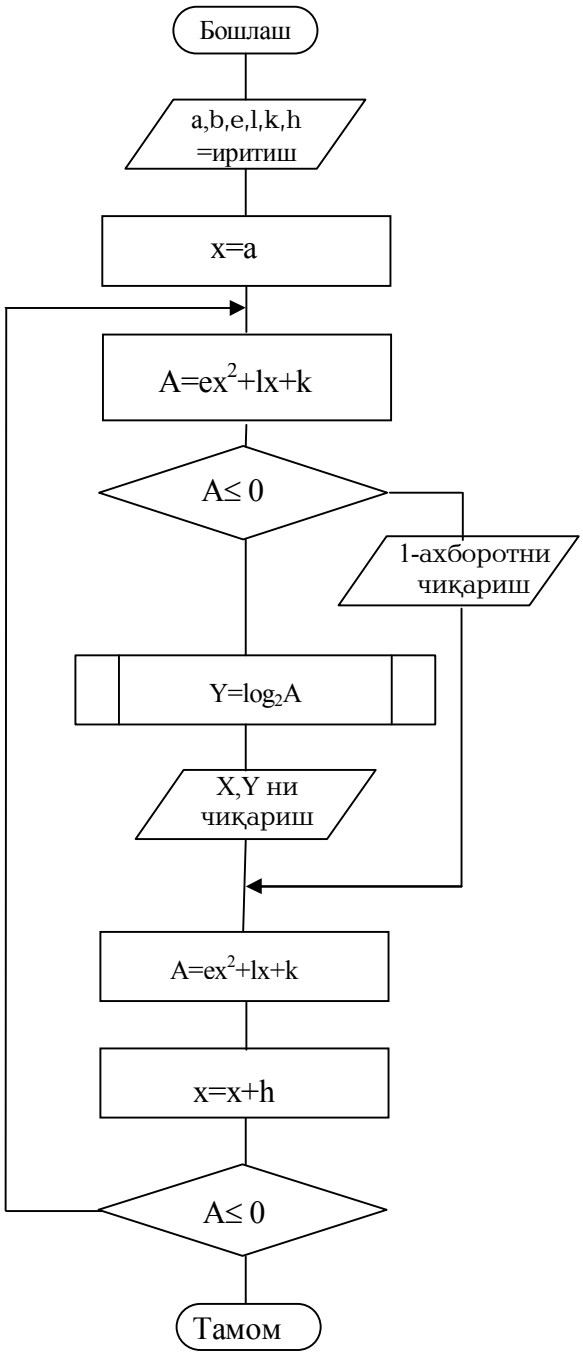
8) $ххxĶh$ қилиб олиб, $x<Ъ$ шартни текшириш. Агар бу шарт бажарилса, 3-қадамга ўтиш, бажарилмаса, ҳисоблашни тугатиш.

Алгоритмнинг блок схемаси 7-расмда келтирилган.

Биринчи алгоритмнинг блок-схемаси иккинчи алгоритмникидан ихчамроқ эканлигини кўрамиз. Бунга модификация блокидан фойдаланилганлиги сабабли эришилди.



А) 1- алгоритм



Б) 2- алгоритм

7 -расм

2-БОБ. Программалаштириш тилларига норасмий кириш.

2.1. Масалаларни ЭҲМ да ечиш босқичлари

ЭҲМ билан бевосита ишлашдан олдин қандай босқичларни бажариш кераклигини кўриб чиқамиз. Исталган ҳаётий ёки математик, физик ва хоказо масала шартларини ифода қилиш дастлабки маълумотлар ва фикрларни тасвирлашдан бошланади ва улар қатъий таърифланган математик ёки физик ва хоказо тушунчалар тилида баён қилинади. Сўнгра ечишнинг мақсади, яъни масалани ечиш натижасида айни нимани ёки нималарни аниқлаш зарурлигини кўрсатилади.

Масалани қўйиш - биринчи босқич.

Масала шартини аниқ ифодаси масаланинг математик (физик ва хоказо) қўйилиши деб ҳам аталади ва исталган масалани ечиш энг аввал унинг қўйилишидан бошланади. Масалани қўйилишида бошланғич маълумотлар ёки аргументлар ҳамда қийматлари аниқланиши керак бўлган катталиклар, яъни натижалар ажратилади. Масалани қўйиш уни ечишнинг биринчи босқичи бўлади. Бунга турли туман мисоллар келтириш мумкин:

1.Томонларнинг узунлиги маълум бўлган тўғри тўртбурчакнинг юзи ҳисоблансин.

2.Босиб ўтилган йўл ва кетган вақт маълум бўлса, йўловчининг тезлиги аниқлансин.

3. Машҳур Пифагордан сўрашди: Сизнинг мактабингизга нечта ўқувчи қатнашади ва суҳбатингизни тинглайди? Пифагор жавоб берди: Менинг ўқувчиларимни ярми математикани ўрганади, чораги мусиқани ўрганади, еттидан бири жимгина фикрлайди, қолгани эса 3 та. Пифагор мактабида нечта ўқувчи бўлган?

4. Шахмат тахтасининг катакларидан бир катакка қайта юрмаслик шарти асосида, от билан юриб ўтинг.

5. Ўтлоқдаги қўйларнинг саккиздан бирининг квадрати ўтлаётган, қолган 12 таси ётган бўлса, ҳаммаси бўлиб нечта қўй бор?

Математик моделни қуриш - иккинчи босқич.

Амалий масалаларни ҳал этишда объектлар - табиат ҳодисалари физик ёки ишлаб чиқариш жараёнлари, маҳсулот ишлаб чиқариш жараёнлари, маҳсулот ишлаб чиқариш режалари ва шу кабилар билан иш кўришга тўғри келади. Ана шундай масалаларни қўйиш учун аввал текширилаётган объектни математик атамаларда тавсифлаш, яъни иложи бўлса, унинг математик моделини (ифодасини) қуриш керак, бу ифода эса ҳақиқий объектни текширишни математик масалани ечишга келтириш имконини беради. Моделни ҳақиқий объектга мослик даражаси амалиётда тажриба орқали текширилади. Тажриба- қурилган моделни баҳолаш ва лозим бўлган ҳолда уни аниқлаштириш имконини беради. Бу

босқич масалаларни ЭҲМ да ечишнинг иккинчи босқичини ташкил этади. Шуни таъкидлаш лозимки, ҳар доим ҳам қўйилган масаланинг математик моделини яратиб бўлавермайди.

Юқорида келтирилган масалаларнинг математик моделларини тузамиз.

Биринчи масала учун математик модел **S қ a b** кўринишдаги формуладан иборат. Бунда бошланғич маълумотлар томонлари узунлиги **a** ва **b** бўлса, натижа тўғри тўртбурчакнинг юзи **S** дан иборатдир.

Иккинчи масала учун босиб ўтилган йўлни **s**, кетган вақт **t** деб белгиласак, йўловчининг тезлиги **v** физика курсидан маълум бўлган

v қ sFt

математик модел билан ифодаланади. Бунда **s** ва **t** бошланғич маълумот, **v** эса натижадир.

Учинчи масалада **x** деб ўқувчилар сонини белгиласак, у

$$\frac{x}{2}+\frac{x}{4}+\frac{x}{7}+3=x$$

ёки 3x-84қ0 кўринишдаги чизиқли тенгламага келади. Бу ерда 3 ва 84 бошланғич маълумотларни, **x** эса натижани ифодалайди.

Тўртинчи масала учун ошкор кўринишдаги математик модел мавжуд эмас, шунинг учун ҳам бу масалани ечишда биринчи босқичдан кейин, тўғридан-тўғри учинчи босқичга ўтиш мумкин.

Бешинчи масалада **x** ни барча қўйларнинг сони деб белгиласак, уни топиш

x- (xF8) ^{2} қ12

кўринишдаги квадрат тенгламани ечишга келади. Умуман бу масалалар

a x ^{2} ҚbxҚсқ0

квадрат тенглама шаклидаги математик модел билан ифодаланади. Бунда **a, b, c** лар бошланғич маълумот бўлса, **x (x$_1$, x$_2$)** натижа бўлади.

Шундай қилиб, биз ходисаларни ифодаловчи математик моделлар билан танишдик. Албатта, ҳозир қурган бу моделлар жуда содда. Ҳаётда шундай мурракаб масалалар учрайдики, улар учун математик модел яратиш жуда кўп куч ва вақт талаб этади, баъзи масалаларни эса моделини умуман тузиш мумкин эмас.

Ечиш усулини аниқлаш - учинчи босқич.

Масалани математик модели яратилгандан сўнг, уни ечиш усули излана бошланади. Айрим ҳолларда масалани қўйилишидан кейин тўғридан тўғри, масалани ечиш усулига ҳам ўтиш керак бўлади. Бундай масалалар ошкор кўринишдаги математик модел билан ифодаланмаслиги мумкин. Бу босқич масалаларни ЭҲМ да ечишнинг учинчи босқичини ташкил қилади. Бунга мисол қилиб юқорида келтирилган математик моделларни ечиш усулларини келтириш мумкин. Улар (1,2,3,5-масалалар) билан сиз математика

курсида танишсиз. Тўртинчи масала учун ечиш усули нима ёки қандай бўлиши мумкин. Шахматдан хабардор ҳар бир кишига маълумки, шахмат тахтасининг ихтиёрий катагида турган отни юқоридаги шарт асосида ҳар доим ҳам юриш мумкин эмас. Ҳамма катаклардан ўтишнинг ягона усули мавжуд ва у қуйидагилардан иборат:

Фараз қилайлик, от шахмат тахтасининг ихтиёрий бир катагида турибди. Умуман олганда бу катакдан бошқа 8 та катакка юриш мумкин. Юрилиши мумкин бўлган бу катакларнинг ҳар биридан ҳам яна нечадир катакларга юриш мумкин. Мана шу мумкин бўлган юришларнинг энг камини танлаш керак, агар улар бир қанча бўлса, ихтиёрий биттасини танлаш мумкин. Демак, отни шундай бир катакка юриш керак эканки, бу катакдан юрилиши мумкин бўлган катаклар сони энг кам бўлсин. Фақат ва фақат шу усул билан қўйилган масалани ҳал қилиш мумкин.

Ечиш алгоритмини тузиш - тўртинчи босқич.

Навбатдаги босқичда, яъни тўртинчи босқичда, масалани ЭҲМ дан фойдаланиб ечиш учун унинг ечиш алгоритми тузилади. Алгоритмни турли-туман кўринишда ёзиш мумкин. Дастурлаш фанининг вазифалардан бири ҳам алгоритм тузиш усулларини ўрганишдан иборатдир. Бу жараёнда талабаларда масалани ечишнинг алгоритми, яъни алгоритмик фикрлаш усули вужудга келади.

Алгоритмни программалаш тилига кўчириш - бешинчи босқич.

Алгоритмнинг ЭҲМ да бажарилиши учун бу алгоритм программалаш тилида ёзилган бўлиши лозим. Масалани ечишнинг бу босқичи бешинчи босқич бўлиб, унда бирор бир усулда тузилган алгоритм маълум бир программалаш тилига кўчирилади. Масалан,агар алгоритм блок-схема кўринишида тасвирланган бўлса, уни программалаш тилига кўчириш учун ҳар бир блокни тилнинг мос буйруқлари билан алмаштириш етарли.

Программани бажарилиши - олтинчи босқич.

Бу босқичда - программа кўринишида ёзилган алгоритмни ЭҲМ ёрдамида бажариш. Бу босқич натижа олиш билан тугалланади. Бу программа тузувчилар учун энг қийин ҳисобланади. Чунки программани машина хотирасига киритишда айрим хатоликларга йўл қўйиш мумкин. Шунинг учун программани ЭҲМ хотирасига киритишда жуда эҳтиёт бўлиш керак.

Олинган натижаларни таҳлил килиш - еттинчи босқич.

Ниҳоят, масалани ечишнинг якунловчи еттинчи босқичи олинган натижаларни таҳлил қилишдир. Бу босқич олинган натижалар қанчалик ҳақиқатга якинлигини аниқлаш мақсадида бажарилади. Натижаларни таҳлил қилиш, зарур бўлган ҳолларда алгоритмни, ечиш усули ва моделини аниқлаштиришга ёрдам беради.

Шундай қилиб, биз масалаларни ЭҲМ ёрдамида ечиш босқичлари билан танишиб ўтдик. Шуни таъкидлаш керакки, ҳар

доим ҳам бу босқичлар бир биридан яққол ажралган ҳолда бўлмасдан, бир-бирига қўшилиб кетган бўлиши ҳам мумкин.

2.2. Металингвистик формулалар тили

Алгоритмик тилнинг синтаксис қоидаларини ёзиш ва тушунтириш учун ҳам қандайдир тил лозим бўлади. Бу тил билан программалашнинг алгоритмик тилларини алмаштириб юбормаслик керак. Киритилиши керак бўлган бу тил программалаш тилини аниқлаш учун керак бўлади. Бу тил "мета тили" деб аталади. Тилни ифодалашда Бэкус-Наурнинг металингвистик формулаларидан (БНФ) фойдаланилади.

БНФ тилида программалаш тилларининг синтаксиси ихчам ва формулалар кўринишида аниқланади. Бу формулалар оддий арифметик формулаларга ўхшаб кетади, шунинг учун ҳам уларни металингвистик формулалар (қисқача метаформула) деб аталади.

Метаформуланинг ўнг ва чап қисмлари "::қ" белгиси билан ажратилади. Белгининг маъноси "аниқланиши бўйича шундай" жумласига яқинроқ. Бу белгининг ўнг томонида мета ўзгарувчи, чап томонида эса мета ўзгарувчининг қийматлар тўплами ётади. Тушунишга осон бўлиши учун мета ўзгарувчилар, ёки уларнинг қийматлари бурчак қавслар ("<" ва ">") ичига олиб ёзилади. Масалан <сон>, <арифметик ифода>, <оператор> ва ҳакозо.

Мета ўзгарувчиларнинг мета қийматлари бир неча мумкин бўлган конструкциялардан ташкил топиши мумкин. Бу ҳолда конструкциялар ўзаро тик чизиқ (|) билан ажратилади. Бу белгининг маъноси "ёки" сўзига яқинроқ тушунчадир.

Метаформулаларга мисоллар:

1. <ўзгарувчи>::қ A | B

<ифода>::қ <ўзгарувчи> | <ўзгарувчи> Қ <ўзгарувчи> | <ўзгарувчи> - <ўзгарувчи>

бу формуладан <ўзгарувчи> деганда A ёки B ҳарфлари тушунилади, <ифода> тушунчаси остида қуйидаги 10 та ҳолнинг бири бўлиши мумкин:

A, B, A Қ A, A Қ B, B Қ A, B Қ B, A - A, A - B, B - A, B - B.

2. <иккилик рақам>::қ 0|1

3. <иккилик код>::қ<иккилик рақам> | <иккилик код> <иккилик рақам>

3-мисолда ўнг томонда аниқланаётган тушунча ётибди, бу мета формулаларнинг рекурсив хоссасига эга эканлигини кўрсатади.

Мета формулаларни ёзишда "{", "}" қавслари учраб туради. Бу қавс ичига олиб ёзилган конструкция такрорланувчи конструкция ҳисобланади.

Мисол:

<иккилик код>::қ <иккилик рақам>{<иккилик рақам>}

3-БОБ. Паскал тилига кириш

3.1. Алгоритмик тилларнинг умумий тавсифи

Маълумки ЭҲМлари берилган алгоритмларни формал бажарувчи автомат ҳисобланади, шунинг учун бирор масалани ЭҲМда ечишда унга мос алгоритмни бериш зарур. Алгоритмни ЭҲМга узатишда эса уни махсус «машина тили»га ўгириб машина кодида ёзилган программага айлантирилади. Шу билан бир қаторда ЭҲМнинг турли хил типлари турлича тилларга эга бўлади, яъни бирор ЭҲМ учун ёзилган программа бошқа ЭҲМ учун тушунарсиз бўлиши мумкин. Шундай қилиб, ҳар бир ЭҲМ фақат ўзининг «машина тили»да ёзилган программаларнигина тушуниши ва бажариши мумкин.

Машина кодида ёзилган программаларнинг кўриниш сифати жуда камбағалдир, чунки бу программалар фақат 0 ва 1 ларнинг махсус кетма-кетлигидан ташкил топади. Бу эса мутахассис бўлмаган одамга тушунарсиз бўлиб, программа тузишда ноқулайликлар келтириб чиқаради.

Айтиб ўтилганлардан шуни хулоса қилиш мумкинки, машина тилидан фойдаланиш одам учун уни қизиқтирган, яъни ечиши лозим бўлган масаланинг алгоритмини ишлаб чиқишда ва ёзишда жуда катта қийинчиликлар ва муаммолар туғдиради.

Юқорида айтиб ўтилган қийинчиликларни бартараф қилиш, дастурчининг ишини осонлаштириш ва яратилган программаларнинг ишончлилик даражасини ошириш мақсадида юқори даражадаги программалаш тиллари ёки алгоритмик тиллар яратилган.

Алгоритмик тилларнинг машина тилларидан асосий фарқлари сифатида қуйидагиларни кўрсатиш мумкин;

-машина тили алфавитидан алгоритмик тил алфавитининг ўта кенглиги;

-тузилган программа матнининг кўриниш сифатини кескин оширади;

-ишлатилиши мумкин бўлган амаллар мажмуи машина амаллари мажмуига боғлиқ эмас;

-бажариладиган амаллар одам учун қулай кўринишда, яъни амалда қабул қилинган математик белгилашларда берилади;

-амаллар операндлари учун дастурчи томонидан бериладиган шахсий исмлар қўйиш мумкинлиги;

-машина учун кўзда тутилган маълумот типларидан ташқари янги типлар киритиш имконияти яратилганлиги.

Шундай қилиб, қайсидир маънода айтиш мумкинки, алгоритмик тиллар машина тилига боғлиқ эмас.

Юқорида айтилганлардан келиб чиққан ҳолда маълум бўлдики, алгоритмик тилда ёзилган масала ечимининг алгоритми тўғридан-тўғри ЭҲМда бажарилиши мумкин эмас экан. Бунинг учун эса,

алгоритм олдиндан ишлатилаётган ЭҲМнинг машина тилига транслятор (компилятор ёки интерпретатор) ёрдамида ўгирилиши лозим. Транслятор - машина тилида ёзилган махсус программа бўлиб, унинг асосий мақсади алгоритмик тилларда ёзилган программа матнини ЭҲМ тилига таржима қилишдан иборатдир.

Амалда программалашда фойдаланилаётган алгоритмик тиллар ўз маъносига кўра алгоритмни сўзли-формулали ёзиш услубига ўхшаб кетади, яъни маълум бир қисм кўрсатмалар оддий математик формулалар, бошқа қисмлар эса сўзлар ёрдамида ифодаланиши мумкин. Мисол сифатида n ва m натурал сонларнинг энг катта умумий бўлувчиси(ЭКУБ)ни топиш алгоритмини кўриб чиқайлик:

1. A=n, B=m дейлик
2. Агар A=B бўлса 5-пунктга, акс ҳолда 3-пунктга ўт.
3. Агар A>B бўлса A нинг янги қиймати деб A-B ни қабул қил, B ни қийматини ўзгартирма; акс ҳолда B нинг янги қиймати деб B-A ни қабул қил, A ни қийматини ўзгартирма.
4. 2-пунктга ўт.
5. ЭКУБ=A ва ҳисобни тўхтат.

Ушбу алгоритмни қисқароқ кўринишда қуйидагича ифодалашимиз ҳам мумкин:

1. A=n, B=m дейлик;
2. Агар A>B бўлса A=A-B акс ҳолда B=B-A, A=B бўлгунча 2-пунктни такрорла.
3. ЭКУБ=A ва ҳисобни тўхтат.

Ушбу мисолдан кўриниб турибдики алгоритмларнинг бундай ёзиш услуби одам учун ҳам қулай ва ҳам тушунарлидир. Лекин бу услубда ҳам маълум камчиликлар кўзга ташланади:

- алгоритмни ортиқча кўп сўзли ва узун дейиш мумкин;
- бир хил маънодаги кўрсатмани турли хил услубларда бериш мумкинлиги;
- бундай эркин кўринишда ифодаланган алгоритмни ЭҲМ тилига ўтказиш имконияти камлиги.

Юқоридаги каби камчиликларни бартараф қилиш учун формаллашган, қатъий аниқланган алгоритмик тиллар ишлаб чиқилган. Алгоритмик тиллар учта ўзакдан ташкил топади: тил алфавити, синтаксиси ва семантикаси.

Тил алфавити шу тилгагина тегишли бўлган чекли сондаги белгилардан ташкил топади. Программа матнини ёзишда фақат шу белгилардангина фойдаланиш мумкин, бошқа белгиларни эса тил танимайди, яъни улардан фойдаланиш мумкин эмас.

Тил синтаксиси алфавит ҳарфларидан ташкил топиб, мумкин бўлган конструкцияларни аниқловчи қоидалар системасидир. Мазкур тилда ифода этилган тўла алгоритм ва унинг алоҳида ҳадлари шу конструкциялар орқали ифода қилинади. Шундай қилиб,

белгиларнинг ҳар қандай кетма-кетлигини, мазкур тилнинг матни тўғрилиги ёки нотўғрилигини тил синтаксиси орқали билиб оламиз.

Тил семантикаси алгоритмик тилнинг айрим конструкциялари учун қоидалар системасини тушунтиришга хизмат қилади.

Энди алгоритмик тилларнинг қайсилари амалда кўпроқ ишлатилиши ҳақида фикр юритсак. Маълумки 70-йилларда бир гуруҳ муаммоли-йўналтирилган алгоритмик тиллар яратилган бўлиб, бу программалаш тилларидан фойдаланиб жуда кўп соҳалардаги муаммоли вазифалар ҳал қилинган. Ҳисоблаш жараёнларининг алгоритмларини ифодалаш учун Алгол-60 ва Фортран тиллари, иқтисодий масалалар алгоритмлари учун Кобол ва Алгэк тиллари, матнли ахборотларни тахрирлаш учун эса Снобол тиллари ишлатилган. Санаб ўтилган бу алгоритмик тиллар асосан катта хажмли, кўпчиликнинг фойдаланишига мўлжалланган ЭҲМлари учун мўлжалланган эди.

Ҳозирда инсоният фаолиятининг барча жабҳаларига шахсий электрон ҳисоблаш машиналари (ШЭҲМ) шахдам қадамлар билан кириб бормоқда. Асосан ШЭҲМларга мўлжалланган, ҳамда мураккаб жараёнларнинг ҳисоб ишларини бажариш ва жуда катта маълумотлар тизими билан ишлашни ташкил этувчи янги алгоритмик тиллар синфи борган сари кенгайиб бормоқда. Бу тиллар жумласига қуйидаги кўпроқ ишлатилаётган тилларни киритиш мумкин:

- Бейсик тили;
- Паскал тили;
- Си тили ва ҳакозо.

Программа тузишни ўрганишни бошловчиларга мўлжалланган, савол-жавоб системасида ишлайдиган, турли-туман жараёнлар алгоритмини ёзишга қулай бўлган тиллардан бири БЕЙСИК(BASIC) тилидир. Бейсик тилининг номи инглиз сўзи (Beginner's All-purpose Symbolic Instruction Code) нинг ўқилишига мос келиб, бошловчилар учун белгили кўрсатмалар коди(тили) деган маънони англатади. Бейсик тилини яратиш устидаги ишлар 1963 йилнинг ёзидан бошланган. Тилнинг ижодкорлари таниқли олимлар Т.Курц ва Ж.Кемени ҳисобланади. Ҳозирга келиб Бейсик тилининг турли хил янги кўринишлари ишлаб чиқилмоқда ва улардан фойдаланиб миллионлаб дастурчилар ажойиб программалар яратишмоқда.

Энди нисбатан мукаммалроқ бўлган Паскал ва Си алгоритмик тиллари ҳақида қисқача фикр юритсак.

Паскал тили 1969 йили Н.Вирт томонидан яратилиб машҳур олим Блез Паскал номи билан аталди. Бу тил Н.Виртнинг ўйлаши бўйича программалашнинг замонавий технологиясига ва услубига, структурали программалаш назариясига асосланган ва бошқа программалаш тилларидан муайян ютуққа эга тил бўлиши лозим эди. Мазкур тил:

1. Программалаштириш концепциясини ва структурасини системали ва аниқ ифодалайди;

2. Программа тузишни системали олиб бориш имконини беради;
3. Программа тузиш учун бой термин ва структура схемаларига эга;
4. Йўл қўйилган хатоликларни таҳлил қилишнинг юқори даражадаги системасига эга.

1981 йили Паскал тилининг ҳалқаро стандарти таклиф этилди ва IBM PC типидаги шахсий компьютерларда Паскал тилининг Borland фирмаси томонидан ишлаб чиқилган Турбо-Паскал оиладош тили кенг қўлланила бошланди. Ҳозирда Турбо-Паскалнинг бир қанча версиялари яратилиб, юқори даражадаги программалар яратиш имкониятлари борган сари кенгайтирилиб борилмоқда:

4.0 версиясидан бошлаб программа ёзишни, таҳрирлашни ва натижалар олишни осонлаштириш учун янги интегралашган муҳит ҳосил қилинди;

5.5 версиясининг пайдо бўлиши билан Турбо-Паскалда объектли программалаш имконияти пайдо бўлди;

6.0 версиясидан бошлаб эса Паскал программаси ичига қуйи программалаш тили бўлмиш Ассемблер тилида ёзилган программаларни қўшиш ҳолати ҳосил қилинди. Шу билан бир қаторда тилнинг интегралашган муҳити ҳам бир қатор ўзгаришларга эга бўлди.

Си тили 1972 йили Д.Ричи томонидан турли хил ЭҲМлар учун универсал тил сифатида ишлаб чиқилган ва дастурчи программа тузиш жараёнида ҳисоблаш машинасининг имкониятларидан кенг фойдаланиши мумкин. Шунинг учун, бу тил барча нарсани қилишга қодир деган тушунча ҳосил бўлган.

Ҳозирда амалда фойдаланилаётган кўпгина операцион системалар Си тилида яратилган.

3.2. Тилнинг алфавити

Маълумки, ҳар қандай тилни ўрганиш унинг алфавитини ўрганишдан бошланади. Тилнинг алфавити - шу тилгагина тегишли бўлган асосий белгилари ва тушунчалар тўпламидан иборат бўлади. Паскал тилининг алфавитини ташкил этувчи асосий белгилар жамламасини 3 гуруҳга ажратиш мумкин: ҳарфлар, рақамлар ва махсус белгилар.

Тил алфавитининг металингвистик (Бэкус - Наур) формуласи қуйидагича бўлади:

<асосий белги>::қ<ҳарф>|<рақам>|<махсус белги>

Ҳарф сифатида катта ва кичик лотин ҳарфлари ишлатилади. Лекин, матнлар ва программага изоҳлар ёзиш учун кирилл алифбосининг бош ва кичик ҳарфларини ҳам алфавитга киритилган. Рақамлар сифатида оддий араб рақамлари олинган:

<рақам>::қ0|1|2|3|4|...|9

Махсус белгилар кўп сонли ва бир жинссиз бўлганлиги учун уларни ўз навбатида 4 та гуруҳга ажратамиз:

<махсус белги>::қ<арифметик амал белгиси>|<солиштириш амали белгиси>|<ажратгич>|<хизматчи сўз>.

<арифметик амал белгиси>::қ = | F | Қ | -

Бу амаллар мос равишда кўпайтириш, бўлиш, қўшиш ва айириш белгилари ҳисобланади.

Солиштириш амалларининг белгилари, уларнинг математик ифодаси ва амалларнинг маъноси 2-жадвалда ўз ифодасини топган. Бу ерда шу нарсага аҳамият бериш керакки, баъзи бир амаллар иккита белги орқали ифодаланган.

2-жадвал

Солиштириш амали белгисининг Паскалдаги ёзилиши	Амалнинг математик ифодаси	Амалнинг маъноси
қ	қ	Тенг
<>	≠	Тенгмас
<	<	Кичик
<қ	≤	Кичик ёки тенг
>	>	Катта
>қ	≥	Катта ёки тенг

Ажратгичлар гуруҳини қуйидаги белгилар ташкил қилади:

<ажратгич>::қ . | , | : | ; | (|) | * | + | { | } | ' | :қ

Ажратгичларнинг вазифаларини тилни ўрганиш давомида аниқлаб борамиз.

Хизматчи сўзлар гуруҳи жуда кенг, шунинг учун бу сўзларни ҳаммасини бирданига ёдлаб, эслаб қолиш шарт эмас, балки улардан фойдаланиш давомида кетма-кет эслаб қолинаверади:

<хизматчи сўзлар>::қand | array | begin | case | const | div | do | downto | else | end | for | function | goto | if | in | label | mod | nil | not | of | or | packed | program / procedure | record | repeat | set | then | to | type | until | var | while | with

3.3. Тилнинг асосий тушунчалари

3.3.1.Операторлар

Оператор тушунчаси тилнинг энг асосий тушунчаларидан бири бўлиб, ҳар бир оператор тилнинг якунланган жумласи ҳисобланади ва маълумотлар таҳлилининг тугалланган босқичини ифодалайди.

Операторларни икки гуруҳга ажратиш мумкин. 1-гуруҳ операторларининг таркибида бошқа операторлар қатнашмайди ва бу операторларни асосий операторлар деб аталади. Асосий операторлар жумласига қуйидаги операторлар киради: ўзлаштириш оператори, процедура оператори, ўтиш оператори, бўш оператор. 2-гуруҳ операторларининг таркибида эса бошқа операторлар ҳам қатнашиб, улар таркибий операторлар деб аталади. Улар жумласига қуйидаги операторлар киради: ташкилий оператор, танлов оператори, такрорлаш оператори, улаш оператори.

Масалани ечиш алгоритмида юқоридаги икки гуруҳ операторларининг кетма-кетлиги чекланмаган миқдорда қатнашиши мумкин. Бу кетма-кетликдаги операторлар ўзаро ";" ажратиш белгиси орқали ажратилади, яъни программа матнининг ёзуви алоҳида операторларга бўлинади. Шундай қилиб, S орқали ихтиёрий ёзиш мумкин бўлган операторни белгиласак, масала ечилишининг алгоритми қуйидаги кетма-кетлик бўйича ифодаланиши мумкин:

S; S; ...;S.

Операторларнинг бу кетма-кетлиги уларнинг программада ёзилиш тартиби бўйича бажарилади. Шундай қилиб, операторнинг издоши ундан кейин ёзилган оператор ҳисобланади. Операторлар бажарилишининг бу табиий кетма-кетлигини фақат ўтиш оператори ёрдамида бузиш мумкин. Таркибий операторларда эса операторларнинг бажарилиш тартиби ўзига хос қоидалар билан аниқланади.

3.3.2.Исмлар ва идентификаторлар

Маълумки, маълумотларнинг таҳлили жараёнини ифодаловчи алгоритм турли хил объектлар (ўзгармаслар, ўзгарувчи миқдорлар, функциялар ва ҳоказо) устида иш олиб боради. Бу объектларга уларнинг вазифаси ва қабул қиладиган қийматларига қараб махсус исмлар берилади. Шу исмларни одатда, идентификаторлар деб аталади. Идентификатор деб ҳарф ёки "_" белгисидан бошланувчи, ҳарф, рақам ва "_" белгисининг ихтиёрий кетма-кетлигига айтилади:

<идентификатор>::қ<ҳарф>|<идентификатор><ҳарф> |
<идентификатор> <рақам>

Агар қуйидаги оралиқ тушунчани киритсак:

<ҳарф ёки рақам>::қ <ҳарф>|<рақам>

Юқоридаги аниқлашни қуйидагича ҳам ёзиш мумкин:

<идентификатор>::қ <ҳарф> {<ҳарф ёки рақам>}.

Хизматчи сўзлардан идентификатор сифатида фойдаланиш мумкин эмас. Одатда идентификатор сўзининг ўрнига қулайроқ ва қисқароқ қилиб исм дейиш мумкин. Программада қатнашувчи объектларга исмларни программа тузувчи ўз ихтиёрига кўра танлаб олиши мумкин. Бир хил исм билан бир неча хил объектларни номлаш мутлақо мумкин эмас. Turbo Pascal муҳитида исмда қатнашувчи белгилар сони (исм узунлиги) 63 та белгидан ошмаслиги керак.

Исмларга мисоллар:

_Burchak, _A1, Ahmad_Berdiev, C, Summa, Time, A, S1, …

3.3.3.Эълонлар

Паскал тилининг асосий тушунчаларидан бири эълон қилиш ҳисобланади. Программада қатнашувчи барча объектларнинг исмлари мос равишда программанинг бош қисмида, уларнинг қандай типдаги қийматлар қабул қилиши мумкинлигига қараб, эълон қилиниб қўйилиши керак. Паскал тилида эълон қилишнинг 5 хил тури мавжуд:

- меткалар эълони;
- ўзгармаслар эълони;
- тип аниқлаш учун эълон;
- ўзгарувчилар эълони;
- процедура ва функциялар эълони.

Умуман олганда, юқорида санаб ўтилган эълонларнинг вазифалари уларнинг номларидан ҳам сезилиб турибди, эълоннинг вазифалари эса кейинроқ тўла очиб берилади.

3.3.4. Ўзгарувчилар

Ўзгарувчи, программа объекти бўлиб, турли хил қийматларни хотирада маълум ном билан сақлаб туриш учун ишлатилади. Ўзгарувчи ўз қийматини программани бажарилиш давомида, ўзлаштириш оператори ёрдамида қабул қилади. Қабул қилинган қиймат, ўзгарувчига бошқа янги қиймат берилмагунча сақланиб турилади ва янги қиймат берилиши билан эски қиймат бутунлай ўчиб, йўқ бўлиб кетади. Ҳар бир ўзгарувчига маълум бир типга тегишли қийматларнигина қабул қилиш ҳуқуқи берилади. Бошқа типдаги қийматларни ўзлаштиришга уриниш программанинг хатолигини таъминлайди.

Ўзгарувчи - бу идентификатордир. Унинг исми ўзгарувчининг қийматига мурожаат қилишда ишлатилади. Бошқача айтганда, программа матнидаги исм, шу ўзгарувчининг қийматини ифодалайди.

3.3.5. Функциялар ва процедуралар

Ўрта мактаб курсидан функция тушунчаси бизга яхши маълум. Алгоритмик тилларда, фақат қийматини ҳисоблаш алгоритмлари маълум бўлгангина функциялар ишлатилади. Программа тузувчи программа учун лозим бўлган кераклича функцияларни ўз программасига киритиши мумкин.

Худди функциялар каби ҳал қилинаётган масаланинг маълум бир тугалланган босқичларини ҳисоблаш вазифасини процедуралар зиммасига юкласа ҳам бўлади. Фукнцияни ҳисоблаш натижасида фақат, ягона натижавий қийматга эришилади, процедурадан фойдаланганда эса, натижавий қийматлар сони етарлича кўп бўлиши мумкин.

Программада аниқланган функция ва процедуралар ўзгарувчиларнинг эълони бўлимида эълон қилиниб қўйилиши керак. Бунда ҳар бир функция ва процедурага, уларнинг бажарадиган вазифасига мос исмлар бериб қўйилади. Уларни аниқлашда формал параметрлардан фойдаланилади. Ўз навбатида, бу параметрларнинг типлари функция ва процедуранинг ичида аниқланилиб, эълон қилинади.

Программада аниқланган функция ва процедуралардан фойдаланиш учун программа матнида уларнинг исмлари ва формал параметрларга мос, фактик параметрлари берилиши керак.

Маълумки, математика курсидаги элементар функциялардан программа тузишда жуда кўп фойдаланишга туғри келади. Масалан $\sin x, \cos x, \ln x, e^x$ ва ҳоказо. Бундай функцияларни бошқа тиллардаги каби стандарт функциялар деб аталади ва стандарт функцияларнинг исмларидан бошқа мақсадда фойдаланиш мақсадга мувофиқ эмас.

3.3.6. Программа матнини ёзиш қоидалари

Ҳар бир алгоритмик тилнинг программа матнини ёзиш қоидалари турлича бўлади. Программалаш тилларидан энг соддаси Бейсик тилининг маълум версияларида, программанинг ҳар бир оператори қатъий аниқланган қатор номерлари орқали ёзилади. Паскал тилида эса, операторлар кетма-кет ёзилиб, ўзаро ";" белгиси билан ажратиб борилади. Бундан ташқари, ёзилган программанинг ўқишга осон ва ундан фойдаланиш қулай бўлишлиги учун программада "матнни ажратиш" тушунчасидан фойдаланилади (бўш жой, қаторни тугаши ва изоҳлар).

Бўш жой (пробел) график тасвирга эга эмас белги бўлиб, қатордаги бўш жойни англатади. Лекин, бўш жой белгиси ўзининг сонли кодига эга ва программа матнидаги бошқа белгилар каби компьютерга киритилади.

Қатор охири (тугаши) бошқарувчи белги бўлиб, у ҳам график тасвирга эга эмас. Маълумки, программа матнини ёзиш давомида уни

табиий равишда янги қаторларга ажратилиб ёзилади. Чунки, шу матнни ёзмоқчи бўлган қоғознинг ҳам, компьютер экранининг ҳам ўлчамлари чекланган. Программа матнини алоҳида қаторларга ажратмай ёзиш ҳам мумкин, лекин бир сатрга 256 тадан ортиқ белги сиғмайди. Программа матнини алоҳида қаторларга ажратиш, программа тузувчининг хоҳишига қараб бажарилади. Маълум бир қатор тугамай туриб, янги қаторга ўтиш учун "қатор охири" тугмачаси босилади. Бу тугмача ҳам ўзининг махсус сонли кодига эга.

Изоҳлар программани ўқишга осон бўлиши, уни қийналмай текшириб, йўл қўйилган хатоларни тўғрилаш ва программада бажарилаётган ишларни тушунтириб бориш учун қўйилади. Изоҳсиз ёзилган программани ҳужжат сифатида қабул қилинмайди. Муваффақиятли қўйилган изоҳ программанинг ва программа тузувчининг катта ютуғи ҳисобланади. Изоҳлар ихтиёрий вақтда программа матнига киритилиши ёки олиб ташланиши мумкин. Бу билан программанинг иши ўзгариб қолмайди. Изоҳларни "{" ва "}" қавслари ичига олиниб ёзилади.

Программа "матн ажратгич"ларидан фойдаланишнинг қуйидаги қоидаларига амал қилиш лозим:

- тилнинг кетма-кет ёзилган иккита конструкцияси орасига албатта бўш жой ёзилиши керак;
- ажратгичларни хизматчи сўзлар, сонлар ва исмлар орасига қўйиш мақсадга мувофиқ эмас.

Қуйида юқоридаги қоиадалар асосида ёзилган программага доир мисол келтирилган.

Мисол. Қуйидаги берилган функцияларнинг қийматларини ⁺a,b⁻ оралиғидаги хқаҚih, $h=\frac{b-a}{n}$ лар учун (*n*-берилган сон) ҳисоблаш программасини тузинг: $f_1(x)$қx^2,$f_2(x)$қ3-x, $f_3(x)$қ0,5-sinx

```
Program P1;
{ f1(x)қx=x; f2(x)қ3-x; f3(x)қ0.5-sin(x) функциялар қийматини ⁺a,b⁻
оралиғида ҳисоблаш программаси }
  const
      nқ10; {⁺a,b⁻ оралиқни 10 та бўлакчаларга ажратдик}
     Var
            a,b:real;
            i:integer;
            x,h,y1,y2,y3:real;
     Begin
            read(a,b);                    {⁺a,b⁻ оралиқни чегараларини ажратиш}
       h:қ(b-a)Fn;  x:қa;  i:қ0;  {Бошланғич маълумотлар ҳисобланди}
    Repeat
        y1:қx=x;
        y2:қ3-x;
        y3:қ0.5-sin(x);
     Writeln (x,y1,y2,y3); {Функциялар ҳисобланиб, натижалар чоп этилмоқда}
         x:қxҚh;  i:қiҚ1;
     Until iқnҚ1
```

{Ҳисоб ишлари якунланди}

end.

3.3.7. Турбо-Паскал муҳитини ўрнатиш

Турбо-Паскалнинг профессионал программалаш - 6.0 версияси (Turbo Pascal Professional) Borland International Inc фирмасининг бир нечта дискеталарида берилади. Шу дискеталарнинг бири «INSTALL/COMPILER» деб номланади ва бу дискетада INSTALL.EXE программаси мавжуд.

Турбо Паскални ўрнатиш учун дискетани дисководга қўйиб, программани ишга туширилади. Программа Сиздан бир неча саволларга жавоб беришни таклиф қилади:

- қайси дисководдан Турбо Паскални ишга туширмоқчисиз?
- Турбо Паскални винчестерга ўрнатасизми?
- қайси каталогларда Турбо Паскалнинг ишчи файлларини жойлаштириш лозим?

Агар Сизда етарли асос бўлмаса, таклиф қилинган каталог исмлари билан чегараланган маъқул. Бу ҳолда сиздан фақатгина диск исмини ўзгартириш талаб этилади. Шундан сўнг INSTALL.EXE программаси ўзининг ишини давом эттиради ва унинг сўрови бўйича навбат билан кўрсатилган номли дискеталарни дисководга қўйилади. Натижада Турбо Паскал ишга тайёр ҳолга келади.

Фараз қилайлик, Турбо-Паскални ўрнатишда одатдаги «C» дискнинг ўрнига «D» дискни танладингиз ва бу ҳолда система қуйидаги каталогларда жойлашди:

D:\TP, D:\TP\BGI, D:\TP\DEMOS, D:\TP\DOC,
D:\TP\DOCDEMOS, D:\TP\TURBO3, D:\TP\TVDEMOS,
D:\TP\TVISION, D:\TP\UTILS

Бу ерда D:\TP\ каталогида Турбо Паскалнинг асосий файллари жойлашган:

- Turbo.exe –программани яратиш учун лозим бўлган интегралашган муҳит (Turbo Pascal Integrated Development Environment) файли;
- Turbo.hlp –тезкор ёрдам маълумотларини сақловчи файл;
- Turbo.tp – Turbo.exe программаси фойдаланадиган система конфигурациясининг файли;
- Turbo.tpl – Турбо Паскалнинг резидент модуллари (Resident units);
- Tptour.exe –интегралашган муҳитда ишлашни таништирувчи программа.

Булардан ташқари, D:\TP\BGI каталогида Турбо Паскалнинг график режим ишини таъминловчи файллар мавжуд:

- Graph.tpu – Турбо Паскалнинг барча график программаларини ишлаши учун зарур файл;

- BGI кенгайтмали бир нечта файл – видеосистемаларнинг турли типлари билан ишлашни таъминловчи драйверлар;
- CHR кенгайтмали бир нечта файл – векторли шрифтларни ўз ичига олувчи файллар.

3.4. Турбо-Паскал тили

3.4.1. Паскал тилининг асосий типлари

Одатда, программада ишлатилувчи маълумотлар қуйидаги типларнинг бирортасига тегишли бўлади: бутун қийматли типлар, ҳақиқий қийматли типлар, белгили ва сатрли типлар, мантиқий қийматли ва кўрсаткичли типлар. Умуман олганда, типларни иккита гуруҳга ажратиш мумкин: асосий (ёки оддий) ва ҳосилавий. Юқорида санаб ўтилган типлар асосий гуруҳга тегишли бўлган типлардир. Ҳосилавий типлар эса, асосий ёки ҳосилавий гуруҳга тегишли типлардан ҳосил қилинади.
Бутун қийматли типга тегишли сонга мисоллар: -1501, 0, 9999.
Бутун қиймат қабул қилувчи ўзгарувчиларни эълон қилиш учун ***Integer, ShortInt, Byte, LongInt*** ва ***Word*** хизматчи сўзларидан фойдаланиш мумкин.
Ҳақиқий қийматли типга тегишли сонларга мисоллар: 25.0956, 6.75, -321.936, 1.2E+02, -3.57E-01
Ҳақиқий (каср) қийматли типга тегишли ўзгарувчиларни эълон қилиш учун ***Real, Single, Double, Extended*** ва ***Comp*** хизматчи сўзларидан фойдаланиш мумкин.
Ҳамма ҳарфлар, белги ва рақамлар, масалан A, b, '', !, $, S белгили типга тегишлидир. Белгили типни қабул қилувчи ўзгарувчиларни эълон қилиш учун ***Char*** хизматчи сўзидан фойдаланиш мумкин.
Белгиларнинг ихтиёрий йиғилмаси (кетма-кетлиги) қаторлар деб аталади. Мисол 'Аҳмад', '$25', '_СТАРТ'. Қатор хатто бўш ҳам бўлиши мумкин (' '). Бу типдаги ўзгарувчиларни эълон қилиш учун String хизматчи сўзидан фойдаланилади.
Мантиқий ўзгарувчилар фақат ***True*** (рост) ва ***False*** (ёлғон) қийматларининг биттасинигина қабул қилиши мумкин. Бу тип ўзгарувчиларини эълон қилиш учун ***Boolean*** ҳизматчи сўзи ишлатилади.
Кўрсатгичлар маълумотларнинг компьютер хотирасидаги турар жойи (адреси)ни аниқлаб беради ва уларни эълон қилиш учун Pointer хизматчи сўзидан фойдаланилади.
Ҳосилавий типларни ҳосил қилиш ва уларни эълон қилиш йўлларини келгуси бўлимларда тўлиқ тушунтириб ўтилади.
Юқорида санаб ўтилган типлар ҳақида тўлиқроқ маълумотлар келтириб ўтамиз.

3.4.2.Бутун сонлар

Бутун қийматли типларнинг барчаси 3- жадвалда келтирилган:

3-жадвал.

Тип кўриниши	Мазкур типли ўзгарувчининг қабул қиладиган қийматлар оралиғи	Ўзгарувчининг компьютер хотирасидан эгаллайдиган жойи
ShortInt	-128..127	8 бит
Integer	-32768..32767	16 бит
LongInt	-2147483648.. 2147483647	32 бит
Byte	0..255	8 бит
Word	0..65535	16 бит

Бу санаб ўтилган типлар ўзларининг қийматлар қабул қилиш оралиғи ва хотирадан эгаллаган жойининг катта ёки кичиклиги билан фарқланади. Шунинг учун, ўзгарувчиларнинг қабул қиладиган қийматларини катта ёки кичиклигига қараб, юқоридаги типлардан мосини танлаш мақсадга мувофиқдир.

Энди шу типдан фойдаланишга доир қуйидаги мисолни кўриб чиқайлик:

Берилган m ва n бутун сонлари устида қуйидаги арифметик амаллар бажариш программасини тузинг: m+n,m-n,m=n. Умуман Паскал тилида программа тузиш унчалик мураккаб эмас, ҳозир шуни амалда кўрсатамиз. Системали қавс ({,})лар ичига турли изоҳ ва тушунтиришлар ёзиб, улар билан программани жиҳозлаймиз.

```
{Программа сарлавҳасини ёзамиз}
    Program Sonlar;
Var {программада фойдаланиш мумкин бўлган барча ўзгарувчилар
шу var сўзидан сўнг эълон қилинади}
m,n:integer;{m ва n ўзгарувчилар ўртача катталикдаги бутун сонлар}
 k1,k2,k3:integer;{k1=m+n, k2=m-n, k3=m=n – бажарилган арифметик
амаллар натижасини хотирада сақлаш учун танланган бутун типли
ўзгарувчилар}
    begin {Паскал программаси begin (бошланмоқ) сўзи билан бошланиб,

end.(тамом) сўзи билан тамомланади}
Readln(m,n); {m ва n бутун сонларини киритиш сўраляпти, агар
киритилаётган сон бутун бўлмаса “Error 106:Invalid numeric format.”
хатоси ҳабар қилинади ва программа ишини тўхтатади}
    k1:=m+n;
    k2:=m-n;
    k3:=m=n;      {сўралган амаллар бажарилди}
```

writeln (k1,k2,k3); {ҳисобланган k1=m+n, k2=m-n, k3=m=n натижавий қийматларни чоп этиш ташкил этилди}
end. {Программа тамом бўлди}
Бундан ташқари, Турбо-Паскалда ўн олтилик саноқ системасида ёзилган бутун сонлардан фойдаланишга ҳам руҳсат берилади. Ўн олтилик саноқ системасидаги бутун сонни аниқлашда унинг олдига “$” (доллар) белгиси қўйилади.
Мисол, $11 ўнли саноқ системасидаги 17 га, $12 сони эса 18 га тенг.
Энди шу ҳолатга доир қуйидаги содда программани келтирамиз:

```
Program Sanoq_sistema;
        Var
        N:integer;
Begin
    N:=12; {N бутун қийматли ўзгарувчига ўнлик саноқ системасидаги 12 сони
    ўзлаштириляпти}
    N:=$12; {N бутун қийматли ўзгарувчига ўн олтилик саноқ системасидаги 12 сони
    ўзлаштириляпти. Бу сон амалдаги ўнли саноқ системасида 18 га тенг}
End.
```

3.4.3.Ҳақиқий сонлар

Ҳақиқий сонлар математика курсидан маълум бўлган оддий ўнлик каср сонлардир. Агар компьютерингиз математик сопроцессорли бўлса қуйидаги 4-жадвалда санаб ўтилган барча типлар ўринли бўлади:

4-жадвал

Тип кўриниши	Мазкур типли ўзгарувчининг қабул қиладиган қиймат оралиғи	Ўзгарувчининг компьютер хотирасидан эгаллайдиган жойи
Real	2.9E-39..1.7E38	6 байт
Single	1.5E-45..3.4E38	4 байт
Double	5.0E-324..1.7E308	8 байт
Extended	3.4E-4932..1.1E4932	10 байт
Comp	-9.2E18..9.2E18	8 байт

Агар компьютерда математик сопроцессор бўлмаса, фақат ***Real*** типинигина ишлатиш мумкин.
Ҳақиқий сонларга доир қуйидаги мисолни кўрайлик:
Берилган m ва n ҳақиқий сонлари устида тўрт математик амални бажариш программасини тузинг.

```
Program arifmetika;
Var
        m,n:real; {m ва n ўзгарувчиларни ҳақиқий сонлар типига
        тегишли деб эълон қилдик}
Begin
Readln (m); {m сонини киритиш сўраляпти}
```

```
    Readln (n); {n сонини киритиш сўраляпти}
    Writeln (‘сонлар йиғиндиси=’,m+n);
    Writeln (‘сонлар айирмаси=’,m-n);
    Writeln (‘сонлар кўпайтмаси=’,m*n);
    If n<>0 then Writeln (‘сонлар бўлинмаси=’,m/n);
{Агар n сони нулга тенг бўлмаса m/n амалининг натижаси мос изоҳ
билан чоп этилади}
end.
```

3.4.4.Белгилар ва қаторлар

Белгили типли ўзгарувчилар ***Char*** хизматчи сўзи билан эълон қилиниб, бу типнинг қийматлари хотирадан 1 байт жой эгаллайди. Паскал тилининг барча белгилари бу типнинг қийматлар соҳасига тегишлидир. Белгили қийматни ′(апостроф) белгиси ичига олиб, ёки # белгисидан кейин унинг ASCII кодини ёзиб аниқлаш мумкин. Мисол: ‘A‘, ёки # 60.

Қатор – бу ′(апостроф) белгиси ичига олиб ёзилган белгиларнинг оддий кетма-кетлигидир: 'Ab21#9!cd', 'dasturchi Saidkarim Gulomov'. Қатор бўш ёки битта белгили бўлиши ҳам мумкин. Қаторли ўзгарувчи узунлиги 255 гача бўлган белгили қийматларни қабул қилиши мумкин. Умуман олганда, ҳар бир қаторли ўзгарувчига хотирадан 256 байт жой ажратилади. Хотирани тежаш учун, қаторнинг типини қуйидагича кўрсатиш мақсадга мувофиқдир: ***String***[N], N - қатордаги белгилар сони. Бу ҳолда белгили ўзгарувчи учун N байт жой ажратилади.

Белгилар ва қаторлар устида бир қанча амаллар бажариш мумкин, яъни қатордан керакли бўлакни кесиб олиш, қаторларни бир-бирига қўшиш ва натижада янги қаторлар ҳосил қилиш. Қаторлар ҳақидаги тўлиқ маълумотни керакли бўлимдан олиш мумкин.

Белгилар ва қаторларга доир қуйидаги содда программани келтирамиз:

```
Program String;
Var
   ch: char; {ch ўзгарувчи белгили қиймат қабул қилади}
          qator1,qator2:String; {qator1 ва qator2 ўзгарувчилар
          узунлиги 255 дан  ортмаган қаторларни ўзлаштириши
          мумкин}
   N:String[5]; {N ўзгарувчиси 5 та белгидан ташкил топган
қаторларни ўзлаштиради}
Begin
      ch:=’A’; {ch ўзгарувчиси A белгини ўзлаштирди}
```

N:=’Ascar’; {N ўзгарувчиси 5 та ҳарфли Ascar сўзини ўзлаштирди}
qator1:=ch+’ли ’+N; {qator1 ўзгарувчиси натижавий Али Ascar сўзини ўзлаштирди}
qator2:=’’; {qator2 ўзгарувчиси бўш қаторни ифодалаяпти лекин, бу ўзгарувчи учун хотирадан 256 байт жой ажратилган}
end.

3.4.5.Маълумотларнинг мантиқий типлари

Паскал тилида мантиқий тип ***boolean*** стандарт номи билан аниқланади. Мантиқий типли ўзгарувчилар фақат икки хил қиймат: ***True***(рост) ва ***False*** (ёлғон) ларнигина қабул қилиши мумкин. Мантиқий типли қийматлар ҳам тартибланган, яъни ***False<True***.
Паскал тилида асосан қуйидаги учта мантиқий амалдан кўпроқ фойдаланилади: ***not*** - рад этмоқ, ***and*** - мантиқий кўпайтириш, ***or*** - мантиқий қўшиш.
Бу амалларни фақат мантиқий ўзгармаслар устидагина ишлатиш мумкин ва натижада яна мантиқий ўзгармас ҳосил бўлади. Қуйида мантиқий ўзгармаслар устида амаллар 5-жадвалда кўрсатилган:

5-жадвал

Мантиқий кўпайтириш	Мантиқий қўшиш	Мантиқий рад этмоқ
True and true қ true	true or true қ true	not true қ false
True and falseқ false	true or falseқ true	not falseқ true
False and true қ false	False or true қ true	
False and false қ false	False or false қ false	

Ихтиёрий, қийматларни солиштириш амали ҳам мантиқий қийматни беради:
Мисол: 3>2 натижаси ***true***
0<-1 натижаси ***false***.

3.4.6.Янги типларни лойиҳалаш

Паскал тилида тилнинг стандарт типларидан ёки олдин ҳосил қилинган янги типлардан фойдаланиб яна янги типлар яратиш мумкин. Программада янги типларни киритиш учун махсус тип аниқлаш бўлими мавжуд. Бу бўлим ***type*** хизматчи сўзидан кейин бошланади.
Ҳар бир янги типни эълон қилишдан олдин унинг номи (типнинг идентификатори), сўнг эса типни нимадан ташкил топганлиги кўрсатилади.
Янги тип ёзув ҳам бўлиши мумкин, унинг майдони эса стандарт типдан ёки олдинги киритилган типлардан ташкил топиши мумкин.

Ўз ўрнида киритилган янги тип программани ёзишда жуда қўл келади ва программанинг сифатини кескин оширади.

3.5. Паскал программанинг структураси

Паскал - программаси – программа сарлавҳаси ва нуқта билан туговчи программа танасидан ташкил топган. Программа сарлавҳаси ва программа танасини ; (нуқта вергул) белгиси билан ажратилади:
<Паскал программаси>::қ<программа сарлавҳаси>;<программа танаси>.
Программа сарлавҳаси ***program*** хизматчи сўзидан бошланади ва ундан сўнг программага фойдаланувчи берган ном ёзилади:
<программа сарлавҳаси>::қ***program***<программа исми>;
Программанинг асосий қисми унинг танаси ҳисобланади. Программанинг танасини қисқача қилиб блок ҳам деб аташ мумкин. Умуман олганда, блок қатъий кетма-кетликда ёзилувчи олтита бўлимдан ташкил топган:

```
<блок>::қ  <меткалар бўлими>
        <ўзгармаслар бўлими>
        <янги типлар бўлими>
        <ўзгарувчилар бўлими>
        <процедура (қисм программа) ва функциялар бўлими>
        <операторлар бўлими>
```

Программа танасининг асосий қисми бу операторлар бўлимидир. Ҳар қандай программада бу бўлим албатта бўлиши керак. Программага қўйилган масалани ечиш шу бўлимда амалга оширилади. Бошқа бўлимлар эса ёрдамчи бўлимлар бўлиб, типларни эълон қилиш бўлимлари деб аталади. Бу ёрдамчи бўлимлар программада қатнашиши ёки қатнашмаслиги ҳам мумкин, лекин уларнинг ёзилиш кетма-кетлиги сақланиб қолиниши зарур. Паскал - программанинг умумий кўринишини қуйидаги кўринишда ёзиб олайликда, сўнг ҳар бир бўлимни тўлароқ таҳлил қилиб чиқамиз:

```
Program <программа исми>;
label
    <меткалар рўйхати>;
const
    <ўзгармаслар ва уларнинг қийматлари>;
type
  <маълумотларнинг янги,  ностандарт типларини аниқлаш>;
var
    <ўзгарувчиларни, процедуралар ва
     функцияларни эълон қилиш>;
begin
    <операторлар бўлими>
end.
```

Меткалар бўлими. Программанинг ихтиёрий операторини бошқа операторлар орасида ажратиб кўрсатиш мумкин. Бунинг учун, операторнинг олдидан икки нуқта (:) белгиси орқали метка (тамға ёки исм деб ҳам аташ мумкин) қўйилади ва бундай операторни метка билан жихозланган оператор деб аталади. Меткалар, программада ўтиш операторидан фойдалангандагина ишлатилади. Метка сифатида оддий идентификаторлардан ёки сонлардан фойдаланилса бўлаверади. Программада ишлатилган барча меткалар ***label*** хизматчи сўзидан кейин бошланувчи меткалар бўлимида эълон қилиниб қўйилиши керак:

<метка бўлими>::қ<бўш> | ***label***<меткалар рўйхати>;

Меткаларнинг номлари оригинал, яъни ўхшаши йўқ бўлиши керак.

Мисол: *label* L1, L2, A3;

Бу ерда L1, L2, A3 лар программада ишлатилувчи меткаларнинг номлари.

```
Label  m10, m20, StopLabel, 1;
Var
      I:ShortInt;
Begin
       1:
          If  i<10 Then Goto m10 Else Goto m20;
    m10:  Writeln('i  кичик 10 дан');
          Goto StopLabel;
    M20:   i:қi-1;
          Goto 1;
    StopLabel:
End.
```

Ўзгармаслар бўлими. Ўзгармас - бу программани ишлаши давомида ўзгармай қоладиган миқдордир. Агар миқдор программада кўп марта ишлатилса, уни программа матнида қайта - қайта ёзгандан кўра, бу миқдорни ўзгармас деб аниқлаб олиб, программадаги миқдорни ўрнига ўзгармасни исмини ёзиш қулай бўлади. Масалан, ҳаммага маълум π (πқ3,1415926535…) сони. Бу сонни бир неча марта такроран программада ёзиш ноқулай, шунинг учун, уни ўзгармас сифатида аниқлаб олиш мақсадга мувофиқдир.

Ўзгармас ***const*** хизматчи сўзидан кейин эълон қилинади (аниқланади):

<ўзгармасни аниқлаш>::қ<ўзгармас исми>қ<ўзгармас>;

бу ерда <ўзгармас>::қ<скаляр миқдор>|<белгили қатор>|<ўзгармас исми>|Қ<ўзгармас исми>|-<ўзгармас исми>;

<ўзгармаслар бўлими>::қ<бўш>|***const***<ўзгармасни аниқлаш>;

```
Мисол:     Program L1;
           const Рiқ3.1415926535;
```

```
var A,B: real;
Begin
        A:қPiҚSin(Pi=(Pi-1));
        B:қsqrt(a);
end.
```

Маълумотларнинг типларини аниқлаш бўлими. Паскал тилида қийматларнинг тўртта стандарт типлари мавжуд: ***integer*** (бутун), ***real*** (ҳақиқий), ***Char*** (белгили) **ва** ***boolean*** (мантиқий). Бу типлар билан бир қаторда, программанинг автори ўзи учун зарур бўладиган типларни аниқлаб олиб, улардан фойдаланиши мумкин. Бунинг учун, муомалага киритилаётган ҳар бир янги типга ўзига хос исм бериш кифоя ва ўзгарувчиларни эълон қилиш бўлимида бу типдан бемалол фойдаланиш мумкин бўлади.

Тип эълон қилиш қуйидаги метаформула асосида амалга оширилади:

<тип эълони>::қ<тип исми>қ<тип>;

<тип>::қ<тип исми>|<тип вазифаси>;

Типларни аниқлаш бўлими эса қуйидагича аниқланади:

<тип аниқлаш бўлими>::қ<бўш>|***type***<тип эълони>

Мисол:

```
        type
        Mantiqқboolean;
        bқinteger;
Haftaқ(dush, sesh, chor, pay, ju, shan, yaksh);
Ish_Kuniқdush..ju;
```

Ўзгарувчилар бўлими. Ҳар қандай программада ўзгарувчилар деб аталувчи программа объектларидан фойдаланилади. Ўзгарувчи - қиймат қабул қилувчи объектдир. Уларга қиймат программани бажарилиши давомида берилади. Ҳар бир ўзгарувчига уни қабул қиладиган қиймати ва вазифасига қараб исмлар берилади. Шундай қилиб, ўзгарувчи ўзининг номи ва қабул қиладиган қиймати билан характерланади.

Программада ишлатилувчи ҳар бир ўзгарувчи ўзи қабул қиладиган қийматларининг типларига мос ҳолда ўзгарувчилар бўлимида эълон қилиниб қўйилиши керак:

<ўзгарувчилар бўлими>::қ<бўш>|***Var***<ўзгарувчилар эълони>;

<ўзгарувчилар эълони>::қ<ўзгарувчи исми>:<тип>

Мисол: Var

```
n,m,k : integer;
d : real;
x,y : real;
        S : boolean;
```

Программада ишлатилган ўзгарувчилар фақат бир мартагина эълон қилиниши керак.

Процедура ва функциялар бўлими. Бошқа тиллардаги каби, Паскал тилида ҳам программани ёзувчи ўзи учун қулай ва зарур бўлган процедура ва функцияларни муомалага киритиши мумкин. Табиийки бу процедура ва функциялар ҳам худди ўзгарувчилар каби эълон қилиниб қўйилиши керак. Ҳамма ишлатилувчи процедура ва

функцияларга исм берилиб, шу исми билан улар чақирилиб программанинг зарур жойида ишлатилади. Уларга мурожаат худди оддий стандарт функцияларга мурожаат каби амалга оширилади. Процедура ва функциялар бўлими ўзгарувчилар бўлимининг давоми бўлиб ***procedure*** ёки ***function*** хизматчи сўзлари билан бошланиб, ихтиёрий кетма-кетликда эълон қилинади.
Операторлар бўлими. Бу бўлим программанинг энг асосий бўлими бўлиб, программа бўйича ҳисобланувчи барча ишлар шу бўлимда бажарилади. Бўлимнинг метаформуласи қуйидагича ёзилади:
<операторлар бўлими>::қ***begin*** <операторлар рўйхати> ***end***.
Программа ўз ишини шу бўлимда ёзилган операторлар рўйхати бўйича, қатъий кетма-кетликда бажаради.

3.6. Тилнинг операторлари

Программанинг асосий вазифаси бошланғич маълумотларни қайта ишлаб, қўйилган масаланинг натижасини берувчи амалларни бажаришдан иборат. Алгоритмик тилларда бирор-бир масалани ечишда маълумотлар устидаги амалларни бажариш операторлар зиммасига юклатилади. Программалаш тилларидаги ҳар бир оператор, маълумотларни қайта ишлаш жараёнининг мустақил босқичи бўлиб, мантиқан якунланган ҳисобланади. Программада ёзилган операторларни тўғри талқин қилиш учун уларни ёзиш қоидалари (операторнинг синтаксиси) қатъий аниқланган бўлиши шарт.
Шундай қилиб, айтиш мумкинки программа бу - турли хил вазифаларни бажарувчи ва ягона мақсадга элтувчи операторларнинг тўпламидир. Ҳар бир оператор ; (нуқта-вергул) белгиси билан якунланади. Мавжуд программалаш тили рухсат берган операторлардан унумли ва оқилона фойдаланиб, мукаммал программалар яратиш дастурчининг билимига, тажрибасига ва санъатига боғлиқдир.
Қуйида Паскал тилининг асосий операторлари билан тўлиқроқ танишиб, улардан программалашда фойдаланиш йўлларини ўрганамиз.

3.10.1.Ўзлаштириш оператори

Одатда программа натижасини ҳосил қилиш учун жуда ҳам кўп оралиқ ҳисоб ишларини бажаришга тўғри келади. Оралиқ натижаларни эса маълум муддатга сақлаб туриш лозим бўлади. Бу ишларни бажариш учун тилнинг энг асосий операторларидан бири бўлмиш - ўзлаштириш оператори ишлатилади:
<ўзлаштириш оператори>::қ<ўзгарувчи>:қ<ифода>;
Бу ерда :қ ўзлаштириш белгиси ҳисобланади, бу белгини қ (тенглик) белгиси билан алмаштирмаслик зарур. Ўзлаштириш операторида :қ белгисининг ўнг томонидаги <ифода> қиймати аниқланилиб, сўнг

чап томондаги ўзгарувчига ўзлаштирилади ёки бошқача қилиб айтганда, ифода қиймати ўзгарувчи номи билан хотирада эслаб қолинади. Ўзгарувчининг олдинги қиймати эса (агар у бўлса) йўқ бўлиб кетади.
Ўзлаштириш операторини ёзишдаги энг муҳим нарса, бу ифода ва ўзгарувчиларнинг бир хил типли бўлишлигидир.
Ўзлаштириш белгисининг ўнг томонидаги ифоданинг натижавий типига қараб, ўзлаштириш операторини уч хил гуруҳга ажратиш мумкин: арифметик ўзлаштириш оператори, мантиқий ўзлаштириш оператори, белгили ўзлаштириш оператори.

3.10.2.Арифметик ўзлаштириш оператори.

Бутун ёки ҳақиқий типли, сонли натижа берувчи ифодани (одатда бундай ифодани арифметик ифода деб аталади) ҳисоблаш учун арифметик ўзлаштириш операторидан фойдаланилади. Арифметик ифодада қатнашувчи барча ўзгарувчилар ҳақиқий ёки бутун типли бўлиши керак. Арифметик ифода- сонлар, ўзгармаслар, ўзгарувчилар ва функциялардан ташкил топади, ҳамда Қ, -, =, F, div, mod каби амаллар ёрдамида ёзилади. Арифметик амалларни бажарилиши қуйидаги тартибда бўлади : =, F, *div, mod*, Қ, -.
Ифодани бажарилишидаги бу тартибни ўзгартириш учун кичик қавслардан фойдаланилади. Ифоданинг қавслар ичига олиб ёзилган қисмлари мустақил ҳолда биринчи галда бажарилади.
Санаб ўтилган арифметик амалларнинг вазифалари бизга математика курсидан маълум. Лекин, бу рўйхатдаги div ва mod амаллари билан таниш эмасмиз. Div – бутун бўлишни англатади, бўлинмани бутун қисми қолдирилиб, қолдиқ ташлаб юборилади. Мисол:

7 div 2 қ 3
5 div 3 қ 1
-7 div 2 қ -3
-7 div –2 қ 3
2 div 5 қ 0
3 div 4 қ 0

Mod – бутун сонлар бўлинмасининг қолдиғини аниқлайди. M mod n қиймат фақат n>0 дагина аниқланган. Агар m≥0 бўлса m mod nқm-((m div n)=n), m<0 бўлса m mod nқm-((m div n)=n)Қn, m mod n нинг натижаси доим мусбат сондир.
Мисол:

7 mod 2 қ 1
3 mod 5 қ 3
(-14) mod 3 қ 1
(-10) mod 5 қ 0

Паскал тилида ҳам бошқа алгоритмик тиллар каби арифметик стандарт функциялари мавжуд. Бу функцияларни математик

ёзилиши ва Паскал тилида ифодаланиши қуйидаги 6-жадвалда келтирилган:

6-жадвал

Функциялар	Паскал тилида ифодаланиши
sin x	**sin(x)**
Cosx	**cos(x)**
Arctg x	**arctan(x)**
Lnx	**ln(x)**
e^x	**exp(x)**
$\sqrt{x}$	**Sqrt(x)**
|x|	**Abs(x)**
x^2	**sqr(x)**
Аргументнинг каср қисмини топиш функцияси	**Frac(x)**
Аргуметнинг бутун қисмини топиш функцияси	**Int(x)**

Арифметик ифодага доир мисоллар :

2=5 - 4=3,
9 div 4F2,
45F5F3,
a Қ bF2=7.2 - sqrt(7),
exp(2 - a)=9.7 - 6.1=6.1

Паскал тилида даражага кўтариш амали йўқ, шунинг учун, бу амални бажаришда логарифмлаш қоидасидан фойдаланамиз.

Мисол: уқa^n ,a>0 ифодани ҳисоблашни кўриб чиқайлик. Тенгликни иккала томонини логарифмлаймиз:

lnуқlna^n , логарифм хоссасига кўра

lnуқnlna , бу тенгликдан "у" ни аниқлаймиз,

уқe^{nlna} - бу тенгликни Паскал тилида қуйидагича ёзиш мумкин:

уқexp(n=ln(a)).

Энди сал мураккаброқ арифметик ифодаларни Паскал тилида ёзилишини кўриб чиқайлик.

Математик ёзуви	Паскал тилидаги ёзуви
aҚb / cҚd	(aҚb)F(cҚd)
$\frac{a\cdot(a+b)}{bc}$	a=(aҚb)F(b=c)

$$\frac{1}{1-\frac{1}{1-\frac{1}{x}}}$$

1F(1-1F(1-1Fx))

$2-(x-b)^2-e^{ax}Қsin CX$

2-sqr(x-b)-exp(a=x)Қsin(c=x)

$$x\cdot\frac{e^{x^2+y^2}-1}{\sqrt{|x^2+y^2|}}$$

x=(exp(x=xҚy=y)-
1)Fsqrt(abs(x=xҚy=y))

Энди арифметик ўзлаштириш операторига доир мисоллар кўриб чиқамиз:

x:қ 0;
c:қ sqrt(a=aҚb=b);
y:қ 2=pi=r; i:қ iҚ1; i:қ 5F4; x:қ a - bF2;

Ўзлаштириш операторининг ўнг томонидаги ифодада қатнашувчи ўзгарувчилар, албатта, бу оператордан олдин ўзининг қийматларига эга бўлиши керак. Акс ҳолда, ўзлаштириш оператори ўз ишини бажара олмайди. Программа тузишда кўпчилик йўл қўядиган хатоликни қуйидаги мисолда тахлил қилиб кўринг:

Тўғри тузилган программа

```
Program Misol;
Var
   a,x,y:Real;
Begin
   a:қ2.3;
   x:қ3.1;
   y:қa=x;
    Writeln('уқ',y);
End.
```

Нотўғри тузилган программа

```
Program Misol;
Var
   a,x,y:Real;
Begin
   a:қ2.3;
   y:қa=x;
{ўзлаштириш операторининг ўнг томонидаги
"X" ўзгарувчининг қиймати аниқланмаган}
    Writeln('уқ',y);
End.
```

3.10.3.Мантиқий ўзлаштириш оператори.

Агар ўзлаштириш операторининг чап томонидаги ўзгарувчи ***boolean*** (мантиқий) типига тегишли бўлса, операторнинг ўнг томонида натижаси ***true*** ёки ***false*** бўлган мантиқий ифода бўлиши шарт. Мантиқий ифода- арифметик ифода, солиштириш белгилари ва мантиқий амаллардан ташкил топади. Мантиқий ифоданинг натижавий қиймати ***true*** (рост) ёки ***false*** (ёлғон) бўлади. Мантиқий ифодада амалларнинг бажарилиш тартиби қуйидагича:

1. Not
2. =, F, div, mod, and
3. Қ,-,or
4. қ, <, >, <қ, >қ,<>

Мантиқий ифодада ҳам амаллар кетма-кетлигини ўзгартириш учун кичик қавслардан фойдаланилади.
Мантиқий ифодага доир мисоллар:

1. x<2=y
2. true
3. not not d
4. (x>yF2)
5. d and (xқy) and b
6. (C or D) and (xқy) or not B

Мантиқий ўзлаштириш операторига доир мисоллар:

```
d:қtrue;
b:қ (x>y) and (kқ0);
c:қD or B and true;
VAR Global Flag:Boolean;
FUNCTION GETSQR( x:real );
Const  SQRMAXқ100;
Begin
X:қx=x;
GlobalFlag:қ( x>SQRMAX );
If  GlobalFlag  then  x:қSQRMAX;
GetSQR:қx;
End;
```

3.10.4.Белгили ўзлаштириш оператори.

Агар ўзлаштириш операторининг чап томонида ***char*** (белгили) ёки String (қаторли) типдаги ўзгарувчи кўрсатилган бўлса, у ҳолда операторнинг ўнг томонида белгили ифода бўлиши зарур. Белгили қийматлар устида фақатгина қўшиш (улаш) амалинигина бажариш мумкин . Шунинг учун, белгили ифода-белгили ўзгармас, белгили ўзгарувчи ёки белгили типли функция бўлиши мумкин.
Белгили ўзлаштириш операторига мисоллар:

```
s:қ'Қ-';
d:қ'=F';
k:қsҚd;
p:қ'Turbo Pascal';
```

```
Program M1;
Var
    s1,s2:String;
Begin
    s1:қ'олий';
    s2:қ' таълим';
    s2:қs1Қs2;
    Writeln(s2);
End.
```

Натижа:

олий таълим

3.10.5.Ташкилий операторлар

Бир нечта операторларнинг кетма-кетлигини битта операторга бирлаштириш учун ташкилий оператор зарур бўлади. Ташкилий оператор -бу ***begin*** ва ***end*** хизматчи сўзлари орасига олиб ёзилган, ихтиёрий операторларнинг кетма-кетлигидир :

<ташкилий оператор>::қ *begin* <оператор>{;<оператор>} *end*

Хусусий ҳолда, операторлар кетма-кетлиги битта оператордан ҳам ташкил топиши мумкин.

Ташкилий операторга доир мисоллар :

1. Begin k:қ 5 end
2. Begin y:қ xF7=exp(xҚ5); z:қ ln(abs(y)) end
3. Begin
   k:қ0;
   begin
   i:қ0;
   z:қi=(iҚk);
   end;
   k:қ2=k;
   end
4. if x>0 then begin a:қ5; c:қа=Sin(a) end

Юқоридаги 3-мисолда кўрсатилгандай, ташкилий оператор рекурсив характерга ҳам эга.

3.10.6.Ўтиш оператори

Одатда, программа ўз ишини ёзилган операторлар кетма-кетлиги бўйича амалга оширади. Операторларнинг табиий бажарилиш кетма-кетлигини бузиш учун, шартсиз ўтиш операторидан фойдаланиш мумкин. Программанинг бирор операторидан бошқаришни бошқа операторга узатиш учун, бошқарилиш узатиладиган оператор олдига тамға (метка) қўйилиши керак . Бошқаришни шартсиз узатиш оператори қуйидаги формада ёзилади :

<ўтиш оператори>::қ *goto* <метка>

бу ерда goto - … га ўтмоқ. Бу оператор ёрдамида бошқариш кўрсатилган меткали операторга узатилади. Юқорида айтганимиздек, программада қатнашган барча меткалар, программанинг меткалар бўлимида эълон қилиниши керак.

Ўтиш операторига доир мисоллар:

```
1)  a:қ 5.75;
    b:қ sqr(a);  goto L5;
    c:қ 9.76;
    L5:        d:қ aҚb;
2) L:  a:қ5; goto L;
3) 1:  x:қ0; d:қx=x; goto 1; y:қx;
```

1. программада C:қ9.76 операторидан бошқа барча операторлар бажарилади;
2. программа a:қ5 қийматни тинимсиз ҳисоблайди;

3. программа ҳам x:қ0 ва d:қx=x ифодани қайта-қайта ҳисоблаб, y:қx ифодани ҳисоблашга навбат келмайди.

Умуман олганда, программа тузувчи иложи борича ўтиш операторида фойдаланмасликка ҳаракат қилгани маъқулдир. Чунки ўтиш операторидан фойдаланиш, программанинг ўқилишини қийинлаштириб,унинг сифатини кескин пасайтиради.
Ўтиш операторидан фойдаланишга доир қуйидаги тўлиқ программани кўриб чиқайлик:

```
Program m1;
        Label 1;
        Var  a,y:real;
Begin
        1: Readln (a);
        If a<=0 then goto 1;{a нинг қиймати a>0 шартини
        қаноатлантирмагунча қайтадан киритилмоқда}
                y:=ln(a)
                Writeln('уқ ',y);
end.
```

3.10.7.Шартли оператор

Алгоритмлар назариясидан маълумки ҳисоблаш жараёнларини шартли равишда уч хил гуруҳга ажратиш мумкин:

1. Чизиқли жараёнлар;
2. Тармоқланувчи жараёнлар;
3. Такрорланувчи жараёнлар.

Чизиқли жараённи ҳисоблаш алгоритми қатъий кетма-кетлик асосида амалга оширилади. Бундай жараённи ҳисоблаш учун ўзлаштириш операторининг ўзи етарли бўлади.
Тармоқланувчи жараённи ҳисоблаш йўли маълум бир шартни бажарилиши ёки бажарилмаслигига қараб танланади. Тармоқланувчи жараёнларни ҳисоблаш учун шартли оператордан фойдаланилади.
Шартли оператори икки хил кўринишда бўлади:

- тўлиқ шартли оператор;
- чала шартли оператор.

Тўлиқ шартли оператор қуйидаги формада ёзилади:
<тўлиқ шартли оператор>::қ ***if*** <мантиқий ифода>
then <оператор> ***else*** <оператор>
бу ерда ***if*** (агар), ***then*** (у ҳолда), ***else*** (акс ҳолда) - хизматчи сўзлар.
Шундай қилиб, тўлиқ шартли операторни соддароқ қуйидагича ёзиш мумкин:

if S then S1 else S2;

бу ерда S – мантиқий ифода;
S1 – S мантиқий ифода рост қиймат қабул қилганда ишловчи оператор;

S2 – S мантиқий ифода ёлғон қиймат қабул қилганда ишловчи оператор.

Шартли операторнинг бажарилиши унда ёзилган S1 ёки S2 операторларидан фақат бирини бажарилишига олиб келади, яъни агар S мантиқий ифода бажарилишидан сўнг ***true*** (рост) қиймати ҳосил бўлса S1 оператори, акс ҳолда эса S2 оператори бажарилади.

Тўлиқ шартли операторга доир мисоллар:

```
1. if ақ2 then d:қ хҚ2 else d:қ х-2;
2. if (x<y) and z then begin y:қ x = sin(x);
      t:қx = cos(x) end else begin y:қ 0; t:қ1 end;
3. if (x<0) or (xқ3) then y:қ x=xҚ1 else if x<2
then y:қ sqr(abs(x-1)) else y:қ x=x;
```

Чала (тўлиқмас) шартли операторнинг ёзилишини қуйидагича ифодаласа бўлади:

if S then S1;

бу ерда S - мантиқий ифода, S1 - оператор.

Агар S ифода қиймати ***true*** (рост) бўлса S1 оператори бажарилади, акс ҳолда эса, бошқариш шартли оператордан кейин ёзилган операторга узатилади.

Юқорида аниқланган шартли операторлардан бир хил мақсадда бемалол фойдаланиш мумкин.

Бу иккала оператордан фойдаланиб, программа тузиш учун қуйидаги мисолни кўриб чиқайлик:

$$y = \begin{cases} ax + b & \text{агар } x > 0 \\ cx + d & \text{агар } x \le 0 \end{cases}$$

бу ерда фараз қилайликки a қ 1,5 ; b қ 4 ; c қ 3,7; d қ - 4,2. x - эса қиймати бериладиган номаълум ўзгарувчи.

"y" тармоқ функциясини ҳисоблаш программасини тузиш талаб этилсин.

1. Тўлиқ шартли оператордан фойдаланиб тузилган программа:

```
program misol1;
      var  x, y, a, b, c, d: real;
begin
    readln (x); { x нинг қийматини клавиатурадан киритиш
    сўралмоқда}
      a:қ1.5; b:қ4; c:қ3.7; d:қ-4.2;
      if  x>0  then  y:қ a=xҚb  else  y:қc=xҚd;
      writeln (y);
end.
```

2. Чала шартли оператордан фойдаланиб тузилган программа:

```
program  misol2;
label L1;
var   x, y, a, b, c, d: real;
begin
     readln (x);
      a:қ1.5; b:қ4; c:қ3.7; d:қ-4.2;
      if  x>0  then begin y:қa=xҚb; goto L1  end;
      y:қ c=xҚd;
L1:   writeln (y);
```

end.

Шартли операторнинг синтаксис қоидасига кўра then ва else хизматчи сўзларидан сўнг фақат битта оператор ёзилиши мумкин, агар бир нечта операторларни ёзиш лозим бўлса у ҳолда, бу операторлар кетма-кетлиги begin ва end хизматчи сўзлари орасига олиниб ташкилий оператор ҳосил қилинади.

Мисол:

```
if a>b then
begin
    y:=a=cos(a);
    z:=Sqr(y);
    p:=Sqrt(abs(y+z));
    writeln(z)
end
else
begin
    y:=a=sin(a);
     z:=Sqr(y)=y;
    p:=tan(y+z);
    writeln(z)
end;
```

Кўпгина операторлар каби, шартли оператор ҳам рекурсивлик хоссасига эга яъни, шартли оператор ичида яна шартли оператор қатнашиши мумкин. Лекин, чала шартли операторнинг ичида яна шартли оператор ёзишда эҳтиёт бўлмоқ зарур, чунки ёзилган операторни икки хил маънода тушуниш мумкин:

if B1 then if B2 then S1 else S2

бу оператор қуйидагича тушунилиши мумкин:

1) **if** B1 **then** begin **if** B2 **then** S1 **else** S2 end
2) **if** B1 **then** begin **if** B2 **then** S1 end **else** S2

3.10.8.Такрорловчи (цикл) операторлар

Юқорида санаб ўтилган жараёнлардан бири, такрорланувчи жараёнларни ҳисоблашни шартли операторлардан фойдаланиб ҳам ташкил этса бўлади, лекин бундай жараёнларни ҳисоблашни такрорлаш операторлари ёрдамида амалга ошириш осонроқ кечади. Такрорлаш операторларининг 3 хил тури мавжуд:

- параметрли такрорлаш оператори;
- repeat такрорлаш оператори;
- while такрорлаш оператори.

Ечилаётган масаланинг моҳиятига қараб, программа ёзувчи ўзи учун қулай бўлган такрорлаш операторини танлаб олиши мумкин.

Параметрли такрорлаш оператори. Операторнинг қуйидаги кўринишдаги ҳоли амалда кўпроқ ишлатилади:

for k:қ k1 ***to*** k2 ***do*** S;

бу ерда **for** (учун), **to** (гача), **do** (бажармоқ) - хизматчи сўзлари;
k - цикл параметри (ҳақиқий типли бўлиши мумкин эмас);
k1 - цикл параметрининг бошланғич қиймати;
k2 - цикл параметрининг охирги қиймати;
S - цикл танаси.
Операторнинг ишлаш принципи:

- цикл параметри (цп) бошланғич қиймат k1 ни қабул қилиб, агар бу қиймат k2 дан кичик бўлса, шу қиймат учун S оператори бажарилади;
- цп нинг қиймати янгисига ўзгартирилиб (агар k сон бўлса ўзгариш кадами 1 га тенг, белгили ўзгарувчи бўлса навбатдаги белгини қабул қилади, ва ҳ.к.) яна S оператори бажарилади ва бу жараён $k > k2$ бўлгунча давом эттирилади. Шундан сўнг, цикл оператори ўз ишини тугатиб бошқаришни ўзидан кейинги операторга узатади.

Агар биз операторларнинг неча марта такроран ҳисобланишини аниқ билсак, у холда параметрли такрорлаш операторидан фойдаланиш мақсадга мувофиқдир.

Мисол: $S = \sum_{i=1}^{n} \frac{1}{i}$ йиғиндини чекли n та ҳадининг йиғиндисини топиш программасини тузиш.

```
Program  sum1;
var
        S: real;
              i,n: byte; {i ва n ўзгарувчилар 255 дан катта бўлмаган,
бутун, натурал сонлар}
begin
readln (n);  S:қ 0;
        for  i:қ1  to   n  do
                S:қ  S Қ 1Fi;
writeln (S);
end.
```

Айрим пайтларда, цикл параметрини ўсиб бориш эмас, балки камайиш тартибида ўзгартириш мумкин, бу ҳолда цикл оператори қуйидаги формада ёзилади:

```
for  k:қ k2  downto  k1  do S;
```

бу ерда ***down to*** (гача камайиб) – тилнинг хизматчи сўзи.
Бу операторда k параметри k2 дан токи k1 гача камайиш тартибида (агар k - бутун қийматли ўзгарувчи бўлса цикл қадами - 1 га тенг) ўзгаради. Операторнинг ишлаш принципи олдинги операторникидай қолаверади.
Мисол: юқорида кўрсатилган мисолни программасини қайтадан тузайлик.
Бу ҳолда программадаги цикл операторигина ўзгаради холос:

```
for  i:қ n  downto  1 do
```

қолган операторлар эса ўз ўрнида ўзгармай қолади.

Программада параметрли такрорлаш операторидан фойдаланиш жараёнида, цикл параметрининг қийматини цикл танаси ичида ўзгартирмаслик лозим, акс ҳолда операторнинг иш ритми бузилиши мумкин. Буни қуйидаги мисолларда кўриш мукин:

Тўғри тузилган программа қисми

```
for i:=1 to 10 do
Begin
        s:=i=i;
        writeln(s);
end;
```

Нотўғри тузилган программа қисми

```
for i:=1 to 10 do
Begin
        s:=i=i;
        writeln(s);
        i:=i+3
end;
```

Маълум бир жараёнларнинг такрорлаш параметрлари ҳақиқий қийматлар қабул қилиши мумкин, бу ҳолда параметрли такрорлаш операторидан тўғридан-тўғри фойдаланиб бўлмайди. Қуйидаги мисолда бундай такрорлашларни қандай ташкил қилиш мумкинлигини кўрамиз:

Мисол: $y=e^x$ функциясини $[-2;2]$ оралиқдаги «x» лар учун ҳисоблаш программасини тузинг («x» нинг ўзгариш қадами 0,5 га тенг деб ҳисоблансин).

Функцияни неча марта ҳисоблаш кераклигини $N=\frac{2-(-2)}{0,5}=\frac{4}{\frac{1}{2}}=8$

формула билан аниқлаймиз.

```
Program Function;
Var  x:real;
     y:real;
     i:integer;
begin
       x:= -2;
       for i:=1 to 9 do
       begin
              y:=exp(x);
              writeln(x,y);
              x:=x+0.5
       end
end.
```

3.10.9.Repeat такрорлаш (цикл) оператори

Юқорида айтиб ўтганимиздек, циклдаги такрорланишлар сони олдиндан маълум бўлса, параметрли (***for***) цикл оператори фойдаланиш учун жуда қулай. Лекин, кўпгина ҳолларда, такрорланувчи жараёнлардаги такрорланишлар сони олдиндан маълум бўлмайди, циклдан чиқиш эса маълум бир шартнинг бажарилиши ёки бажарилмаслигига боғлик ҳолда бўлади. Бу ҳолларда ***repeat*** ёки ***while*** цикл операторларидан фойдаланиш

зарур. Агар циклдан чиқиш шарти, такрорланувчи жараённинг охирида жойлашган бўлса repeat операторидан, бош қисмида жойлашган бўлса ***while*** операторидан фойдаланиш мақсадга мувофиқдир.

Repeat операторининг ёзилиш формаси қуйидагича бўлади:

repeat S1; S2; ... SN ***until*** B;

бу ерда ***repeat*** (такрорламоқ), ***until*** (гача) - хизматчи сўзлар;
S1, S2, ..., SN лар эса цикл танасини ташкил этувчи операторлар;
B - циклдан чиқиш шарти (мантиқий ифода).

Операторнинг ишлаш принципи жуда содда, яъни циклнинг танаси B мантиқий ифода рост қийматли натижа бермагунча такрор - такрор ҳисобланаверади. Мисол сифатида, яна юқоридаги йиғинди ҳисоблаш мисолини олайлик.

```
Program  Sum2;
var      i, n: Byte;
         S: real;
begin
      readln(n);
      S: қ0;  i:қ1;
      repeat
         S:қ SҚ1Fi;
         i:қiҚ1;
      until  i>n;
      writeln (S)
end.
```

Айрим такрорланиш жараёнларида циклдан чиқиш шартини ифодаловчи мантиқий ифода ҳеч қачон True (рост) қийматга эришмаслиги мумкин. Бу холда программанинг такрорлаш қисми чексиз марта қайтадан ҳисобланиши мумкин, яъни дастурчилар тили билан айтганда «**программа осилиб қолади**» шунинг учун, операторлаги шартни танлашда эътиборли бўлиш лозим. Эътиборингизга яъна бир, исмни қидириб топиш программасини хавола қиламиз:

```
Program ism;
Var
  a,b:String*20+;
Begin
  a:қ'Jamshid';
  Repeat
    Writeln('Танлаган исмингизни киритинг');
     Readln(B);
     if a<>b Then writeln('Нотугри') else writeln('Яшанг тугри
топдингиз');
  Until АқB;
End.
```

3.10.10.While такрорлаш (цикл) оператори.

Аҳамият берган бўлсангиз, ***repeat*** операторида циклнинг тана қисми камида бир марта ҳисобланади. Лекин, айрим пайтларда, шу бир марта ҳисоблаш ҳам ечилаётган масаланинг моҳиятини бузиб юбориши мумкин. Бундай ҳолларда, қуйидаги формада ёзилувчи ***while*** цикл операторидан фойдаланиш мақсадга мувофиқдир:

while B do S;

бу ерда ***while*** (ҳозирча), ***do*** (бажармоқ) - хизматчи сўзлари;
B - циклдан чиқишни ифодаловчи мантиқий ифода;
S - циклнинг танасини ташкил этувчи оператор.
Бу операторда олдин B шарти текширилади, агар у ***false*** (ёлғон) қийматли натижага эришсагина цикл ўз ишини тугатади, акс ҳолда циклни тана қисми қайта - қайта ҳисобланаверади.
While операторига мисол сифатида, яна юқорида берилган йиғинди ҳисоблаш мисолини кўриб чиқайлик:

```
program  sum3;
var     i, n: byte;
         S: real;
begin
        readln(n);
        i:қ1; S:қ 0;
        while i<қn  do
        begin
             S:қ S Қ 1Fi;
               i:қ iҚ1;
       end;
       writeln (S)
end.
```

3.10.11.Бўш оператор

Бу оператор ўзидан кейинги операторни аниқлаб беради холос. Операторлар кетма-кетлиги орасида бошқа операторлардан ";" белгиси билан ажратилиб турилади. Бундан ташқари, бўш оператор метка билан жиҳозланган ҳам бўлиши мумкин.
Мисол:

1) **begin** L1:; k:қ5; M:қkҚ6; **end**.
2) **begin** M:қ5; k:қM-2.7; L4: **end**.

Айрим пайтларда, баъзи бир операторларга бир нечта метка билан мурожаат қилишга тўғри келганда бўш операторлардан фойдаланиш қўл келади.

S5:; S6:; S7: x:қ0.5;

3.11. Қийматларнинг скаляр типлари

Шу пайтгача биз қийматларнинг стандарт-скаляр типлари устида сўз олиб бордик ва улардан программалашда фойдаландик. Бу типлар Паскал тилининг ўзида аниқланган типлар эди. Лекин, Паскал тили программа тузувчи ўзи учун қулай бўлган янги типлар киритиш

имкониятини беради. Шундай янги типлардан бири сифатида чекланган ва саналма типларни кўрсатиш мумкин.

3.11.1.Саналма типлар

Тилнинг стандарт типларига биз "бутун сон"лар, "ҳақиқий сон"лар, "белги"лар ва "мантиқий қиймат"ларни киритган эдик. Лекин, амалда турли хил типдаги қийматлар билан ишлашга тўғри келади. Масалан, ранг тушунчаси қизил, қора, оқ, сариқ, кулранг ва ҳ. к.ларни ўз ичига олади, ёки йил ойлари тушунчаси январ, феврал, ..., декабр каби 12 та ойни ўз ичига олади. Бундай қийматли типларни сонлар орқали ифодалаб олса ҳам бўлади, лекин бу белгилаб олиш уларнинг моҳиятини йўқотиб, тушунишга қийин ҳолни ҳосил қилади.
Масалан:
if k қ 7 then
программа қаторини ўқиб, гапни нима ҳақида кетаётганилигини дабдурустдан англаш қийин. Эҳтимол, гап бу ерда 7 - ой ҳақидадир, балки "k" ўзгарувчини 7 бутун сони билан текширилаётгандир. Шундай қилиб, "7" сони остида нима яширинганини билиш жуда қийин. Лекин, программанинг бу қатори

if k қ июл ***then***

бўлса, гап йилнинг июл ойи ҳақида кетаётганлигини осонгина англаш мумкин.
Юқоридаги каби тушунмовчиликларни бартараф қилиш, программани ўқишга қулайлигини ошириш учун қийматлар типларининг саналма типи киритилган.
Стандарт типлар ичида бу типга мисол қилиб ***boolean*** (мантиқий) типини кўрсатиш мумкин: ***boolean*** қ (***false***, ***true***).
Саналма қиймат типини қуйидагича аниқланади:
<саналма типи>::қ(<исм>,<исм>,...) ёки <саналма типи>::қ(<исм> {,<исм>})
бу ерда кичик қавс ичидаги ўзаро вергул билан ажратилган <исм>лар аниқланган типнинг ўзгармаслари ҳисобланади, уларнинг қавс ичига олиб ёзилган бирикмаси эса, шу типнинг қийматлар тўплами ҳисобланади. Саналма тип қийматлари қатъий нолдан бошлаб номерланган.
Масалан, (душанба, сешанба, чоршанба, пайшанба, жума, шанба, якшанба) саналма типи 7 та ҳаддан иборат бўлиб, бу ерда қуйидаги ҳол ўринлидир:
душанба < сешанба < чоршанба < пайшанба < жума < шанба < якшанба яъни душанба 0 - тартиб рақамига , сешанба 1-тартиб рақамига эга ва ҳ.к.
Бу тип программанинг янги типлар бўлимида аниқланади.
Саналма типни аниқлашга доир мисоллар:

```
type
Rang қ (qizil, safsar, sariq, kuk, havorang, kulrang, qora, oq);
```

Hafta қ (**dush, sesh, chor, pay, jum, shan, yaksh**);
Mevalar қ (**olma, nok, shaftoli, uzum**);
Gul қ **Rang**;

бу ерда биз тўртта саналма тип киритдик, охирги *Gul* типи *Rang* типи билан бир хил қилиб аниқланди.

Шуни эсда тутиш керакки, бир исмда ҳар хил тип қийматлари бўлиши мумкин эмас. Масалан юқоридаги типларнинг сафига

Zirovor қ (zira, qalampir, olma)

типини қўшиш мумкин эмас, чунки olma қиймати Mevalar типида аниқланган эди. Бундай тип эълон қилиш, программанинг хатолигини англатади.

Программанинг ***type*** бўлимида аниқлаб қўйилган типлардан ўзгарувчиларни типларини эълон қилиш бўлимида худди стандарт типлар каби фойдаланилса бўлади:

```
var
    Kun: Hafta;
    shar, kub: Rang;
```

Янги саналма типларни ўзгарувчиларнинг типларини эълон қилиш бўлимида ҳам киритиш мумкин:

```
var
    A, B: (stul, divan, stol, shkaf, parta);
```

Лекин, киритилган бу тип исмсиз бўлганлиги учун бу типга программанинг бошқа жойларидан мурожаат қилиш мумкин эмас. Шунинг учун, саналма типни аниқлашнинг биринчи усули маъқулроқдир.

Саналма типли "x" аргументи учун ***Succ***(x), ***pred***(x) ва ord(x) стандарт функциялари мавжуд. Юқоридаги аниқланган типларга доир мисоллардан кўриб чиқайлик.

Фараз қилайлик, хқ sesh қийматли бўлсин.

succ (x) қ chor (x дан кейинги қиймат),
pred (x) қ dush (x дан олдинги қиймат),
org (x) қ 1 (x қийматнинг тартиб рақами),

```
for x қ sesh to yaksh do S;
```

Саналма типли қийматлар устида ҳеч қандай амални бажариб бўлмайди.

Саналма типли қийматларни чоп этиш учун одатда вариант танлаш операторидан фойдаланилади.

3.11.2.Вариант танлаш оператори

Айрим алгоритмларнинг ҳисоблаш жараёнлари ўзларининг кўп тармоқлилиги билан ажралиб туради. Умуман олганда, тармоқли жараёнларни ҳисоблаш учун шартли оператордан фойдаланиш етарлидир. Лекин, тармоқлар сони кўп бўлса, шартли оператордан фойланиш алгоритмнинг кўринишини қўполлаштириб юборади. Бу ҳолларда шартли операторнинг умумлашмаси бўлган вариант танлаш операторидан фойдаланиш мақсадга мувофиқдир.

Вариант танлаш операторини синтаксис аниқланмаси қуйидагича:

<вариант танлаш оператори>::қ ***case*** <оператор селектори>
of <вариант рўйхатининг ҳадлари> ***end***;
бу ерда <оператор селектори>::қ <ифода>;
<вариант рўйхатининг ҳади>::қ<вариантлар меткаларининг рўйхати>:<оператор>;
<вариантлар меткаларининг рўйхати>::қ<вариант меткаси>{,<вариант меткаси>};
<вариант меткаси>::қ<ўзгармас>
Вариант танлаш операторини бажарилиш пайтида, олдин селекторнинг қиймати ҳисобланади, шундан сўнг селекторнинг қийматига мос метка билан жиҳозланган оператор бажарилади ва шу билан вариант танлаш оператори ўз ишини якунлайди. Шуни эсда тутиш керакки, <вариант метка>си билан <оператор метка>си бир хил тушунча эмас ва вариант меткаси меткалар *(Label)* бўлимида кўрсатилмаслиги керак. Бундан ташқари, улар ўтиш операторида ишлатилиши мумкин эмас.
Мисоллар:

```
1. Case i mod 3 of
       0:  m: қ 0;
       1:  m: қ -1;
       2:  m: қ 1
   end.
2. Case sum of
     'қ' : k: қ 1;
    '=', 'Қ', 'F', '-' : ;
    '!' : k: қ 2;
              ':', ';' : k: қ 3
   end.
3. Case kun of
      dush, sesh, chor, pay, jum: writeln('иш куни');
      shan, yaksh: writeln('дам олиш куни')
  end.
```

Вариант танлаш операторининг тана қисмига кириш фақат ***case*** орқали амалга оширилади.
Энди шартли операторни, вариант танлаш оператори орқали ифодасини кўриб чиқайлик:

Қуйидаги шартли ва вариант танлаш операторлари бир-бирига мос келади

```
if B then S1 else S2
```

ва

```
Case B of
    true: S1;
    false: S2;
end.
```

Қуйидаги чала шартли ва вариант танлаш операторлари бир-бирига мос келади

2. **if** B **then** S ва Case B of
true: S;
false:
end.

Вариант танлаш операторларидан саналма типлар қийматларини кўришга қулай ҳолда чоп этиш учун фойдаланиш мумкин. Буни қуйидаги мисол устида кўриб чиқамиз:

```
type
        haftaқ(Du,Se,Cho,Pa,Jy,Sha,Jak);
  var
      kun:hafta;
  begin
      kun:қDu;
       case kun of
Du,Se,Cho,Pa,Ju: writeln('Иш куни');
Sha,Jak:writeln('Дам олиш куни');
       end;
    for kun:қDu to Jak do
   begin
      case kun of
            Du: writeln('Душанба');
         Se:  writeln('Сешанба');
           Cho:writeln('Чоршанба');
         Pa:  writeln('Пайшанба');
           Ju:   writeln('Жума');
         Sha:writeln('Шанба');
         Jak:writeln('Якшанба');
          end ;
      end;
 end.
```

Саналма типли қийматларни киритиш, анча қийин масала ҳисобланиб, бу ишни ташкил қилиш учун тўғридан-тўғри вариант танлаш операторидан фойдаланиш мумкин эмас. Маълумотларни киритишда асосан қаторли (String) типлар имкониятларидан фойдаланилади.

3.11.3.Чекланган типлар

Фараз қилайлик, "n" ўзгарувчиси программада қайсидир ойнинг бирор кунини ифодаловчи бутун сон бўлсин. Бу ўзгарувчини ***integer*** типи билан эълон қилсак, "n"га ихтиёрий бутун сонни ўзлаштириш мумкин. Лекин, ечилаётган масаланинг моҳиятига кўра "n"нинг фақат *1; 31+ оралиқдаги қийматларигина маънога эга холос. Ўзгарувчининг бошқа қийматга эга бўлиши унинг хато ҳисобланаётганлигини ёки программага берилган маълумотларнинг хатолигини англатади. Шунга ўхшаш, дастурчи программани тузиш давомида маълум бир ўзгарувчилар қийматларининг ўзгариш

оралиқлари ҳақида маълумотга эга бўлади. Бундай маълумотларнинг программада кўрсатиб қўйилиши, программа ишининг тўғри бажарилаётганлиги устидан назорат қилиб туриш имкониятини яратади.
Шундай чеклашларни амалга ошириш учун, Паскал тилида чекланган типлар киритилган. Ҳар бир шундай тип олдиндан маълум бўлган типларга чеклашлар киритиш орқали аниқланади.
Чекланган типлар қуйидагича аниқланади:
<чекланган тип>::қ<ўзгармас1>..<ўзгармас2>
бу ерда <ўзгармас1> ва <ўзгармас2>лар чекланган тип киритилаётган, олдиндан аниқланган типнинг ўзгармасларидир, ҳамда <ўзгармас1> < <ўзгармас2> шарти бажарилиши керак.
Чекланган типларга доир мисоллар:
1..20 (integer типидаги чекланма);

1) dush..jum (саналма типли чекланма);
2) 'A'..'Z' (char типидаги чекланма).

Юқорида айтганимиздек, чекланма олдиндан аниқланган типга нисбатан аниқланади. Масалан, (dush, sesh, chor, pay, jum, shan, yaksh) саналма типини аниқламасдан туриб dush..jum чекланган типини киритиш мумкин эмас.
Чекланма тип ҳам бошқа типлар каби, янги тип аниқлаш бўлимида ёки ўзгарувчиларнинг типларини эълон қилиш бўлимида аниқланиши мумкин.
Чекланган типларни эълон қилишга доир бўлган мисоллардан яна кўриб чиқайлик:

```
type
        Hafta қ (dush, sesh, chor, pay, jum, shan, yaksh);
        Figura қ (piyoda, ot, fil, ferz, shoh);
        Engil_Figura қ piyoda..fil;
        Ish_Kunlari қ dush..jum;
        Indeks қ 10..20;
var
        x, y, z: real;
        i, j: integer;
Kun: Hafta;
L: Indeks;
        Ish_Kuni: Ish_Kunlari;
        Dam_Olish_Kuni: Shan..yaksh;
```

3.7. Комбинацияли типлар (ёзувлар)

Эътиборингизга яна бир янги, бошқа алгоритмик тилларда мавжуд бўлмаган, Паскал тилининг ҳосилавий типларидан бирини - комбинацияли типни ҳавола қиламиз. Комбинацияли типнинг қиймати ҳам худди массивларники каби (массивлар ҳақида бошланғич маълумотларга эгасиз деб ўйлаймиз) бир нечта ҳаддан ташкил топади, лекин массивдан фарқли ўлароқ, унинг ҳар бир ҳади турлича типли бўлиши мумкин. Бу типнинг ҳадларига, уларнинг

жойлашган тартиб рақамлари билан эмас, балки исмлари орқали мурожаат қилинади.
Комбинацияли типни одатда соддагина қилиб - ёзувлар деб аталади. Комбинацияли тип қийматлари асосан мураккаб, бир жинссиз ташкил этувчиларга эга бўлган объектларни ифодалашга бағишланган. Бу типдан амалда турли хил маълумотлар жамғармасини яратишда кенг фойдаланилади. Масалан, бирор ташкилотда ишловчи ходимлар ҳақидаги маълумотлар жамғармаси - ходимларнинг турли хил анкетали ҳабарларини ўзида жамлайди:
фамилияси,
исми-шарифи,
туғилган йили-ойи-куни,
уй адреси ишчи ва уй телефон рақамлари,
маълумоти, мутахассислиги,
оилавий аҳволи,
ҳарбийга алоқадорлиги ва ҳ.к.
Санаб ўтилган ва битта шахсга тегишли бўлган ушбу маълумотларнинг турли хил типга тегишлигига аҳамият беринг: телефонлар,туғилган йили-ойи-кунлари – бутун сонлардан, бошқа маълумотлар эса белгили қаторлардан ташкил топган.
Шундай қилиб, комбинацияли тип қиймати – майдонлар деб аталмиш чекли сондаги ҳадлардан ташкил топган маълумотлар структурасидир. Ёзувнинг ҳар бир майдонига ўзига хос исм берилади ва бу майдон қийматининг типи кўрсатилади. Бунда, майдон қийматининг типига ҳеч қандай чеклашлар қўйилмайди. Шунинг учун, ёзув ҳадлари ўз навбатида яна ёзув бўлиши мумкин. Демак, ёзув аниқ ифодаланган иерархик структурага эга бўлиши мумкин. Бир хил даражада турган, битта ёзувнинг барча исмлари турли хил бўлиши лозим, худди шунингдек, турли ёзувлар бир хил исмли майдонларни ўз ичига олиши мумкин. Бунда бу майдонларга мурожаат қилишда ҳеч қандай англашилмовчилик бўлмайди, чунки мурожаат ташқи ёзув орқали амалга оширилади.

Ёзувлардан фойдаланиб мукаммал программалар яратишдан олдин, содда ҳолларда унинг имкониятлари билан танишиб чиқайлик.
Оддий комбинацияли типлар. Соддалик учун, бир авлодли структура орқали ёзилган комбинацияли тип маълумотлари билан танишиб чиқайлик.
Ёзувларни аниқлаш (киритиш) қуйидаги синтаксис қоида бўйича амалга оширилади:
<комбинацияли типни аниқлаш>::қ record < майдонлар рўйхати> end
<майдонлар рўйхати>::қ<`ёзув секцияси>{;<`ёзув секцияси>}
<`ёзув секцияси>::қ<майдон исми>{,<майдон исми>}:<тип>
Шу қоидага асосланиб, математика фанидан яхши таниш бўлган комплекс сонли тип киритайлик (a Қ bi -комплекс сон, a, b - ҳақиқий сонлар, i^2қ-1) ва тип номини complex деб атайлик:

```
type
```

```
        CompLex қ record
                re: real;
                im: real;
    end
```

Бу типни қуйидагича тушунтириш мумкин: ***CompLex*** типига тегишли ихтиёрий қиймат иккита ҳадли (майдонли) ёзувдан ташкил топган структурадир. Ёзув майдонлари re ва im номлари билан аталади ва улар **real** типига тегишлидир.

re ва im бир хил типли бўлгани учун, уларни битта рўйхатга бирлаштириб ёзса ҳам бўлади:

```
type
    CompLex қ record
            re, im: real;
              end
```

Энди барча комплекс (мавҳум) қийматлар қабул қилувчи ўзгарувчиларнинг типларини ***var*** бўлимида аниқлаш мумкин:

```
var
    x, y, z: CompLex;
```

Бу типдаги ўзгарувчиларга бирор қиймат ўзлаштириш учун, уларнинг майдонларини ташкил этувчиларга қиймат бериш керак бўлади. Масалан, x ўзгарувчига 4,5 Қ i = 6,75 қийматини ўзлаштириш учун, re ва im исмли майдонларга қиймат бериш керак:

```
x.re :қ 4.5;   x.im :қ 6.75;
```

Бир хил комбинацияли типга тегишли ўзгарувчилар учун фақат ўзлаштириш амалигина ўринли холос:

```
y :қ  x;
```

Ёзувларга доир қуйидаги мисол устида, улар билан ишлаш малакамизни оширайлик:

Мисол: x ва y мавҳум сонлари устида қўшиш, айириш ва кўпайтириш амалларини бажариш программасини тузинг.

Масалани ечиш алгоритми:

Агар x қ Re x Қ i Im x, y қ Re y Қ i Im y бўлса, улар устида санаб ўтилган амалларни бажариш алгоритми қуйидагича бўлади:

```
 u қ x Қ y,     Re u қ Re x Қ Re y,  Im u қ Im x Қ Im y;
 v қ x - y ,    Re v қ Re x - Re y,  Im u қ Im x - Im y;
 w қ x = y,     Re w қ Re x = Re y - Im x = Im y,
                Im w қ Re y = Im x Қ Re x = Im y
```

Энди мазкур алгоритмни программада ифода этамиз:

```
Program L1;
type
    comp қ record
    re, im: real
end;
var
x, y, u, v, w: comp;
begin {x ва y мавҳум сонларнинг ҳақиқий (Re x, Re y) ва
      мавҳум (Im x, Im y) қисмларини киритиш}
    readln (x.re, x.im, y.re, y.im);
    { uқxҚy}
```

```
u.re:қ x.re Қ y.re;  u.im :қ x.im Қ y.im;
{vқx-y}
v.re :қx.re - y.re;   v.im :қ x.im - y.im;
{wқx=y}
w.re  :қ x.re = y.re - x.im = y.im;
w.im :қ x.re = y.im Қ x.im = y.re;
writeln ( ‘x Қ y қ’,u.re,’Қ’,u.im,’=i’);
writeln ( ‘x - y қ’,v.re,’Қ’,v.im,’=i’);
writeln ( ‘x = y қ’,w.re,’Қ’,w.im,’=i’);
end.
```

Иерархик ёзувлар. Олдинги мавзуда биз киритган ёзувларнинг майдонлари скаляр миқдорлар эди. Паскал тилида эса, юқорида таъкидлаганимиздек, ёзув майдонининг типларига ҳеч қандай чеклашлар қўйилмайди, шунинг учун ёзувнинг ҳадлари яна ёзувлардан, уларнинг ҳадлари ҳам ўз навбатида ёзувлардан ташкил топиши мумкин.

Ёзув майдонининг типи икки хил усул билан аниқланиши мумкин:

- тўғридан - тўғри комбинацияли тип ичида аниқланиши;
- олдиндан аниқланган типлар орқали аниқланиши.

Иерархик (авлодли) структура бўйича аниқланган ёзувдаги энг паст даражада фақат оддий скаляр типларгина қатнашади. Авлод даражасининг ўсиб бориши давомида, типларнинг мураккаблик даражаси ҳам ортиб боради.

Мисол сифатида «Ilmiy_xodim» деган ёзувли типни кўриб чиқайлик. Бу типни киритишдан аввал, бир нечта ёрдамчи типларни аниқлаб оламиз. Бу типлар асосий «Ilmiy_xodim» типини аниқлашда керак бўлади:

```
type rab қ packed array *1..15+ of  char;
        data қ record
           kun : 1..31;
           oy : (yanv, fev, mart, apr,may, iyun, iyul, avg, setnt, oct, noy,
dek);
           yil: integer;
          end;
Ilmiy_xodim қ record
        fish : record  {fish – фамилияси, исми, шарифи}
fam, ism, shar: rab
end;
        tug : data;
        jins: (erkak, ajl);
        malum: (boch, orta, ortmax, olij);
        maosh: integer;
        ildar: (darajasiz, fan_nom, doktor); {ildar – илмий даражаси}
        ilunvon: (unvonsiz, dos,  kat_ul_xodim, prof); {Ilunvon – илмий
унвони}
        uyadr: record
                ind: integer;
                        shah, kucha: rab;
                        uy, kvar: integer;
        end;
```

```
        tel: record
    uy, hizmat: integer
            end;
end;
```

Энди *«shaxs»* деган *«Ilmiy_xodim»* типидаги ўзгарувчини киритамиз:

```
var
    shaxs: Ilmiy_xodim;
```

Бу ўзгарувчининг қийматларини барча майдонларнинг қийматларини аниқлаш орқали берилади, масалан:

```
shaxs.fish.ism:қ ‘Аҳмад’;
shaxs.fish.fam:қ’Пўлатов’;
shaxs.fish.Shar:қ’Анварович’;
shaxs.tug.kun:қ’15’;
shaxs.tug.oy:қmaj;
shaxs.tug.yil:қ’1953’;
shaxs.jins:қerkak;
shaxs.malum:қolij;
shaxs.maosh:қ’3000’;
shaxs.ildar:қfan_nom;
shaxs.ilunvon:қdos;
shaxs.uyadr.ind:қ’717325’;
shaxs.uyadr.shah:қ’Наманган’;
shaxs.uyadr.kucha:қ’Навоий’;
shaxs.uyadr.uy:қ’20’;
shaxs.uyadr.kv:қ’5’;
shaxs.tel.uy:қ’45215’;
shaxs.tel.hiz:қ’40625’;
```

Боғлама оператори. Олдинги мавзуда ҳавола қилинган программадан кўриниб турибдики, ёзувлар билан ишлаш программанинг матнини жуда узун қилиб, ҳар доим ёзувнинг тўлиқ исмларини ёзишга тўғри келади. Жумладан, бу мисолда 18 марта «shaxs» сўзи такрорланди. Шундай такрорланувчи ёзув ҳадларини камайтириш мақсадида боғлама оператори киритилган:

<боғлама оператори>::қ<сарлавҳа> <оператор>
<сарлавҳа>::қwith<ёзувли ўзгарувчилар рўйхати> do
<ёзувли ўзгарувчилар рўйхати>::қ<ёзувли ўзгарувчи>{,<ёзувли ўзгарувчи>}

Энди бу операторларни амалда ишлатилишини кўриб чиқамиз:

with R ***do*** S

бу ерда ***with*** ва ***do*** - хизматчи сўзлар;

R - комбинацияли типга тегишли ўзгарувчи;

S - ихтиёрий оператор.

Боғлама операторининг бажарилиши – S операторининг бажарилиши демакдир. Фақат S операторининг ички қисмида R.p ёзуви ўрнига (p- майдон исми) фақат p ёзилади холос.

Масалан, олдинги мисолимиз учун ***with*** shaxs ***do*** операторидан кейин ўзгарувчиларнинг қийматларини ёзсак:

fish.ism :қ’Аҳмад’; ва ҳ.к.

Бу ерда кўриниб турибдики, программа матни ўзлаштириш операторининг чап томонидаги исмларда “Shaxs” сўзи тушириб қолдирилганлиги сабабли анча ихчамлашди.
Операторнинг ишлашини яхшироқ тушуниш учун қисқароқ программани кўриб чиқайлик.
Қуйидаги ўзгарувчиларнинг қийматлар типларини эълон қилиш бўлими мавжуд бўлсин:

```
var
        R2: record
A, B, C: integer
       end;
        R3: record
A, D: integer;
                B: record
        C, E : integer
              end
               end;
```

У ҳолда боғлама операторининг киритилиши:

```
with R3, B, R2 do
begin
A:қ1;  B:қ2;  C:қ3; D:қ4; E:қ5
end
```

қуйидаги ташкилий операторга тенг кучлидир:

```
begin
R2.A:қ1;  R2.B:қ2; R2.C:қ3; R3.D:қ4;  R3.B.E:қ5
End.
```

3.8. Тўпламли типлар

3.8.1.Паскал тилида тўпламларни белгилаш

Тўплам тушунчаси математика курсидан яхши маълум бўлиб, программалаш тилларида ҳам кенг маънода ишлатилади. Тўпламнинг ҳадлари бир хил типли ва чекли сондаги бўлиши керак. Тўпламдаги ҳадларнинг сони 256 тадан ортмаслиги ёки умуман тўплам бўш бўлиши ҳам мумкин. Тўпламнинг ҳадлари ўрта кавс ичига олиниб, ўзаро вергул билан ажратилиб ёзилади.
Паскал тилида тўплам типлари сифатида олдинги мавзуларда кўриб чиқилган ихтиёрий скаляр тип қабул қилиниши мумкин, фақат Real типини қабул қилиши мумкин эмас.
Паскал тилида ёзилган тўпламларга мисоллар:

* +	- бўш тўплам;
*2,3, 5, 7, 11+	- 2, 3, 5, 7, 11 бутун сонларидан тузилган тўплам;
*1..10+	- 1 дан 10 гача бўлган бутун сонлар тўплами;
*‘a’, ‘b’, ‘d’+	- a, b, d ҳарф (белги)лардан тузилган тўплам;
*оқ,қора, сариқ+	- 3 та ҳадли, саналма типдаги қийматлардан ҳосил қилинган тўплам;

*‘d’..’a’+	- бўш тўплам;
*‘a’..’d’,’f’..’h’, ‘k’+	- a, b, c, d, f, g, h, k ҳарфларидан ташкил топган, белгили типли тўплам;
*1..1,5..1+	- бир қийматли, битта ҳадли, сонли тўплам, яъни *1+ га тенг кучли.

Тўпламларнинг умумий назариясидаги каби Паскалда ҳам, тўплам ҳадларининг тартиб жойи ҳеч қандай вазифа бажармайди ва ҳар бир ҳад фақат бир марта ҳисобга олинади:

1) *true, false+ ва *false, true+ тўпламлари эквивалентдир;
2) *1,2,3,2..5,6,4,3+ тўплами *1,2,3,4,5,6+ тўпламига эквивалент, у эса ўз навбатида *1..6+ тўпламига эквивалент.

Нотўғри ёзилган тўпламлардан намуналар:

1) *5.3, 2.5+ – тўплам ҳадлари Real типидаги сон бўлиши мумкин эмас;
2) *9, 7, “P”,0+ – тўплам ҳадлари бир хил типли бўлиши лозим эди;
3) *‘abc’,’Axad’+ – тўплам ҳадлари ҳосилавий типларга тегишли бўлиши мумкин эмас.

Тўпламлар устидаги муносабат амалларининг қуйидаги жадвалини эътиборингизга ҳавола қиламиз:

Паскалдаги ёзилиши	Математик ёзилиши	Натижавий қиймат	
		True	False
AқB	A⊆B	А ва В тўпламлари мос тушади	акс ҳолда
A.<>B	A≠B	А ва В тўпламлари мос тушмайди	акс ҳолда
A<қB	A⊆B	А тўпламининг барча ҳадлари В тўпламга тегишли	акс ҳолда
A>қB	A⊇B	В тўпламининг барча ҳадлари А тўпламга тегишли	акс ҳолда
x in A	X∈A	х ҳади А тўпламга тегишли	акс ҳолда

3.8.2.Тўпламлар устида амаллар

Иккита А ва В тўпламлари тенг (бир хил) дейилади, агарда уларнинг барча ҳадлари ўзаро бир хил бўлса (АқB).
Мисол: Ақ*4,1,3+, Вқ*4,1,3+.
А тўплам В тўпламининг тўплам остиси дейилади, агарда А нинг барча ҳадлари В нинг ҳам ҳадлари бўлса (A<B).
Мисол: Ақ*1, 2, 3+ , Вқ*1, 2, 3, 4, 5, 6+.

А ва В тўпламларининг кесишмасидан яна тўплам ҳосил бўлади ва натижавий тўпламнинг ҳадлари А тўпламга ҳам, В тўпламга ҳам тегишли бўлади (С қА=В).
Мисол: *1,2,3,4,5+ = *2,5,6,7,8+ қ *2,5+. Агар А ва В тўпламлари бир хил ҳадларга эга бўлмаса, С қ А=В натижа – бўш тўплам бўлади С қ * +.
А ва В тўпламларнинг бирлашмаси (С қАҚВ) уларнинг ҳеч бўлмаса бирортасига тегишли бўлган ҳадлардан ташкил топади.
Мисол: *1,2,3,4,5+ Қ *2,5,6,7,8+ қ *1,2,3,4,5,6,7,8+.

1) А тўпламдан В тўпламни айирмаси деб, шундай тўпламга айтиладики, натижавий тўпламнинг ҳадлари А тўпламга тегишли лекин В тўпламга тегишли бўлмайди (С қ А - В). Мисол: *1,2,3,4,5+ - *2,5,6,7,8+ қ *1,3,4+.
2) х қийматни А тўпламига тегишлилигини аниқлаш учун «*in*» амалидан фойдаланилади. х *in* А қ *true* , агар х қиймат А тўпламига тегишли бўлса; х in А қ *false* , агар х қиймат А тўпламига тегишли бўлмаса.

Мисол: А қ*2, 3, 4, 7+ бўлса, 6 *in* А ифодасининг натижаси *false:* 3 in А ифодани натижаси эса *True.*

3.8.3.Тўпламли типни бериш ва тўпламли ўзгарувчилар

Тўпламли типларни аниқлаш қуйидагича амалга оширилади:
Set of <тўплам типи>;
бу ерда *Set*, *of* - Паскал тилининг хизматчи сўзлари.
Мисол:
1. *Set of* 1..3 - тўпламнинг базавий типи 1 дан 3 гача бўлган бутун сонлар оралиги, шунинг учун бу тўпламнинг қийматлари сифатида 1, 2 ва 3 сонларидан тузилган барча тўпламларни олиш мумкин (хатто бўш тўпламни ҳам): * +, *1+, *2+, *3+, *1, 2+, *1, 3+, *2, 3+ ва *1, 2, 3+.
2. Фақат шу тўпламларгина, юқорида аниқланган тўпламли типнинг қийматлари бўлиши мумкин.
Set of boolean;
бундай аниқланган тўпламнинг қийматлари тўплами:
* +, **true*+, **false*+, **true*, *false*+

```
3. type
        oy қ (yanv, fev, ..., dek);
        yiloyi қ Set of oy
        bool қ Set of boolean;
   var
        bolalar: Set of (ugil, kiz);
        oylar: yiloyi;
        mantiq: bool;
```

Тўпламли қийматларни тўпламли ўзгарувчиларга ўзлаштириш учун ўзлаштириш оператори ишлатилади:

```
V:қS;
```

бу ерда V - тўплам типига тегишли ўзгарувчи;
S - тўпламли ифода.

Тўпламли ифодага мисоллар:

1) *1, 2, 5, 6,7+ = *2, 6+ Қ *3,9+, натижаси *2, 3, 5, 6, 9+;
2) (*3,4,5+Қ*1-,3,6,7+) = *5..7+-*6+, натижаси *5,7+.

Энди тўпламлар устида бажариладиган амалларга доир қуйидаги мисол программасини яратайлик.

Киритилган қаторнинг фақат сон, лотин ҳарфлари ва бўш жойлардан ташкил топганлигини аниқлаш программасини тузинг.

```
Program Tuplamlar;
Var
      Str:string;
      L:Byte;
      Tru:Boolean;
Begin
Writeln('Қаторни киритинг');
      Readln(Str);
      L:қLength(Str);                          {Киритилган символлар сони}
      Tru:қL>0;                                 {True, агар бўш қатор бўлмаса}
      While Tru and (l>0) do                   {Қатор охиригача текшириш }
      Begin
            Tru:қStr*L+ in *'0'..'9', 'A'..'Z', 'a'..'z', ' '+;
          {Символлар тўғрилигини текшириш }
            Dec(L);                             {Олдинги символ }
      End;
      If  Tru  Then Writeln('Тўғри ёзилган қатор')
Else Writeln('Нотўғри ёзилган қатор');
End.
```

Қуйида фақат ҳақиқий сонларни қабул қила оладиган функцияни яратиш программаси келтирилган:

```
Program Set_Of;
Uses Crt;
Var
   R:Real;
Function Input_R:Real;  {Ҳақиқий сонларни киритиш функцияси}
Var
   S:String*15+;          {S ўзгарцвчига кўпи билан 15 белги сиғади}
   S1:Set Of Char;        {Белгиларни текшириш учун тўплам}
   Ch:Char;               {Белгиларни қабул қиладиган ўзгарувчи}
   Code:Integer;
   R1:Real;
Begin
    S:қ'';
    S1:қ*'0'..'9','.','-'+;
    Repeat
      Ch:қReadkey;
      If Ch in S1 Then    {Ch га киритилган белгини S1 тўплам ичидан
текшириш}
      Begin
          S:қSҚCh;
```

```
            Write(Ch);
          End;
      Until Chқ#13;                {Enter клавиши босилгунча цикл
айланади}
      Val(S,R1,Code);
      Input_R:қR1;
End;
Begin
   R:қInput_R;              {Input_R функциясидан фойдаланиш}
   Writeln;
   Writeln('1-натижақ ',R:10:4);
   Writeln('2-натижақ ',Input_R:10:4); {Input_R функциясидан
фойдаланиш}
   Readln;
End.
```

3.9. Массивлар (жадвал катталиклар)

Биз шу пайтгача қийматларнинг оддий (скаляр) типларидан фойдаланиб, турли хил программалар яратишни ўргандик. Скаляр типга тегишли ҳар бир қиймат ягона маълумот ҳисобланиб, тривиал структурага эгадир.

Амалда эса, турли хил ҳосилавий типлар билан ишлашга, улардан фойдаланиб мураккаб программалар яратишга тўғри келади. Бу типларга тегишли қийматларнинг ҳар бири тривиал бўлмаган структурага эга, яъни бу қийматлар ўз навбатида яна бир нечта қийматлардан ташкил топади.

Энди шундай типлардан бири бўлган, программалашда энг кўп қўлланиладиган программа объекти – массивлар билан танишиб чиқамиз.

3.14.1.Бир ўлчамли массивлар.

Массив - бу бир хил типли, чекли қийматларнинг тартибланган тўпламидир. Массивларга мисол сифатида математика курсидан маълум бўлган векторлар, матрицалар ва тензорларни кўрсатиш мумкин.

Программада ишлатилувчи барча массивларга ўзига хос исм бериш керак. Массивнинг ҳар бир ҳадига мурожаат эса, унинг номи ва ўрта қавс ичига олиб ёзилган тартиб ҳади орқали амалга оширилади:

<массив номи>*<индекс>+

бу ерда <индекс> - массив ҳадининг жойлашган ўрнини англатувчи тартиб қиймати.

Умуман олганда, <индекс> ўрнида <ифода> қатнашиши ҳам мумкин. Индексни ифодаловчи ифоданинг типини – индекс типи деб аталади. Индекс типининг қийматлар тўплами албатта номерланган тўплам бўлиши, шу билан бир қаторда, массив ҳадлари сонини аниқлаши ва уларнинг тартибини белгилаши керак.

Массивларни эълон қилишда индекс типи билан бир қаторда массив ҳадларининг типи ҳам кўрсатилиши керак. Бир ўлчамли массивни эълон қилиш қуйидагича амалга оширилади:
array *<индекс типи>+ of <массив ҳадининг типи>;
Кўпинча <индекс типи> сифатида чекланма типлардан фойдаланилади, чунки бу типга тегишли тўплам тартибланган ва қатъий номерлангандир. Мисол учун, 100 та ҳақиқий сонли ҳадлардан иборат массив қуйидагича эълон қилинади:
array *1..100+ of real;
Массивларни эълон қилиш ҳақида тўлиқроқ маълумот бериш учун турли типдаги индексларга оид мисолларни эътиборингизга ҳавола қиламиз:

1. array *1000..5000+ of integer;
2. array *-754..-1+ of byte;
3. array *0..100+ of real;
4. array *0..10+ of boolean;
5. array *10..25+ of char;
6. type
   chegara қ 1..100;
   vektor қ array *chegara+ of real;
   massiv1 қ array *115..130+ of integer;
   massiv2 қ array *-754..-1+ of integer;
   var
   A,B: vektor;
   c,d : massiv1;
   e: massiv2;
7. var
   r, t: array *chegara+ of real;
   s, q: array *115..130+ of integer;
   p: array *-754..-1+ of integer;
   k, m: array *1..50+ of (shar, kub, doira);
8. type kv1 қ (yanvar, fevral, mart);
   var t, r: array *kv1+ of real;
9. type
   belgi қ array *boolean+ of integer;
   belgi_kodi қ array *char+ of integer;
   var
   k : belgi;
   p : belgi_kodi;

Энди массивлар устида типик амаллар бажарувчи бир нечта программа билан танишиб чиқайлик.
1. Бир ўлчамли, n та ҳадли (nқ30) массив ҳадларини йиғиш.

```
Program L1;
const nқ30;
var
i: integer;
```

```
x: array *1..n+ of real;
S: real;
begin
  for i: қ1 to n do readln (x*i+); { массив ҳадларини киритиш}
      S: қ0;
for i: қ1 to n do S: қSҚx*i+;
      writeln ('натижақ', S)
end.
```

2. Бир ўлчамли, n та ҳадли (nқ30) массив ҳадларининг энг каттасини топиш ва унинг жойлашган жойини аниқлаш.

```
Program L2;
const nқ30;
type
      gran қ 1..30;
      vector қ array *gran+ of real;
var
      x: vector;
      S: real;
      i, k: integer;
begin
      writeln (' x - массиви ҳадларини киритинг');
      for i: қ1 to n do readln (x*i+);
      S: қx*1+; k: қ1;
      for i: қ2 to n do
      if x*i+ > S then
      begin
            S: қx*i+; k: қi
      end;
      writeln ('x массивининиг энг катта ҳади');
      writeln (S);
      writeln ('max(x) нинг ўрни', k)
end.
```

3. n та ҳадли (n қ 15) векторларнинг скаляр кўпайтмасини аниқлаш.

```
Program L3;
const nқ15;
type
gran қ 1..n;
     mas қ array *gran+ of real;
var
i: byte;
S: real;
     x, y: mas;
begin
writeln ('x ва y массив ҳадларини киритинг');
     for i: қ1 to n do readln (x*i+);
     for i: қ1 to n do readln (y*i+);
S: қ0;
     for i:қ1 to n do S:қ S Қ x*i+ = y*i+;
     writeln ('натижа', S)
end.
```

3.14.2.Кўп ўлчамли массивлар

Бир ўлчамли массивларнинг ҳадлари скаляр миқдорлар бўлган эди. Умумий ҳолда эса, массив ҳадлари ўз навбатида яна массивлар бўлиши мумкин, агар бу массивлар скаляр миқдорлар бўлса, натижада икки ўлчамли массивларни ҳосил қиламиз. Икки ўлчамли массивларга мисол сифатида математика курсидаги матрицаларни келтириш мумкин. Агар бир ўлчамли массивнинг ҳадлари ўз навбатида матрицалар бўлса натижада уч ўлчовли массивлар ҳосил қилинади ва ҳ.к.

Икки ўлчамли массив типини кўрсатиш қуйидагича бажарилади:

array *<индекс типи>+ of array *<индекс типи>+ of <скаляр тип>;

Икки ўлчамли массивларнинг типларини бир неча хил йўлда аниқлашни қуйидаги мисол устида кўриб чиқайлик (А матрица 10 та сатр ва 20 та устундан иборат бўлиб, унинг хадлари ҳақиқий типга тегишли бўлсин):

```
1. var
       A: array *1..10+ of array *1..20+ of real;
2. type matr қ array *1..10+ of array *1..20+ of real;
   var
     A: matr;
3. type gran1 қ 1..10; gran2 қ 1..20;
       matr қ array *gran1, gran2+ of real;
       var    A: matr;
4. var A: array *1..10, 1..20+ of real;
```

Яна шуни ҳам айтиш мумкинки, икки ўлчамли массив индексларининг типлари турли хил ҳам бўлиши мумкин. Бу ҳолни қуйидаги мисол устида кўриб чиқайлик:

```
Program L1;
const  n қ 24;
type   hafcun қ (dush, sesh, chor, pay, jum, shan, yaksh);
       Ishkun  қ dush..jum;
       detson  қ array *1..n+ of char;
var    A: array *boolean+ of array *1..n+ of char;
       B: detson;
       C: array *1..365+ of detson;
```

Икки ўлчамли массивлар устидаги бир нечта тугалланган программалар билан танишиб чиқайлик.

1. Матрицаларни қўшиш.

```
Program L2;
const  n қ 3;  m қ 4;
{ n - матрица сатрлари сони, m - устунлар сони}
var    i, j: integer;
       A, B, C: array *1..n,1..m+ of real;
begin {A, B матрица ҳадларини киритиш}
       for i : қ 1 to n do
  for j: қ1 to m do
    readln (A*i,j+, B*i,j+);
```

```
        for i : қ 1  to  n  do
          for  j: қ 1  to m  do
  begin
    C*i,j+ :қ A*i,j+ Қ B*i,j+;
          writeln (C*i,j+)
        end
end.
```

2. Матрицани векторга кўпайтириш.

```
        Program  L3;
        const  n қ3; m қ 4;
        type  matr  қ array *1..n, 1..m+  of  real;
              vect қ array *1..m+  of  real;
        var   i, j: byte;
              A: matr;
              B, C: vect;

        begin
        writeln ('A матрица ҳадларини киритинг');
              for i:қ1  to  n  do
                  for  j:қ1  to  m  do
                      readln (A*i,j+);
              writeln ('B вектор ҳадларини киритинг');
              for i:қ1  to  n  do readln (B*i+);
              for i:қ1  to n  do
                      begin
        C*i+:қ0;
                              for j:қ1  to m  do
                              C*i+:қ C*i+ Қ A*i,j+ = B*j+;
                              writeln (C*i+);
                      end
        end.
```

3. Матрица ҳадларининг энг каттасини топиш ва унинг жойлашган жойини аниқлаш.

```
Program  L4;
const nқ3;  mқ4;
var   A: array *1..n, 1..m+ of  real;
      R: real;
      i, j: byte;  K, L: byte;
begin {A матрица ҳадларини киритиш}
      for i: қ1  to  n  do
for  j: қ1  to m  do
            readln (A*i,j+);
      R:қ A*1,1+;  L:қ 1;  K:қ 1;
      for i: қ1  to  n  do
          for  j: қ1  to  m do
        begin
              if  R< A*i,j+  then
          begin
                R: қA*i,j+;
            L: қ i;  K: қj;
          end;
         end;
       writeln ('max Aқ', R);
```

```
            writeln (‘сатрқ’,L,’устун қ’,K);
    end.
```

3.10. Процедура - операторлар

Программа тузиш жараёнида, унинг турли жойларида маъносига кўра бир хил, мустақил характерга эга бўлган ва ечилаётган асосий масаланинг бирор қисмини ҳал қилишни ўз бўйнига олган мураккаб алгоритмдан бир неча маротаба фойдаланишга тўғри келади. Масалан, матрицаларни кўпайтириш, матрицани векторга кўпайтириш, чизиқсиз тенгламани ечиш, чизиқли алгебраик тенгламалар системасини ечиш, факториал ҳисоблаш, йиғинди ҳисоблаш ва ҳоказо каби масалаларни ҳал қилиш алгоритлари жуда ҳам кўп масалаларни ечишнинг бош алгоритмларида қайта-қайта, турли бошланғич маълумотлар билан қатнашиши мумкин. Бундай ҳолларда, малакали дастурчи программа матнини ихчамлаштириш, программанинг ишончлилик даражасини ошириш, программани тахрирлаш (отладка) ни тезлаштириш ва программанинг умумийлигини (универсаллигини) таъминлаш учун процедура ва функциялардан кенгроқ фойдаланиб, мукаммал программа яратишга ҳаракат қилади.

Процедура ва функциялар мустақил программали объектлар ҳисобланади. Бу мустақил программали объектни дастурчи ўз ҳохишига ва ундан олинадиган натижаларига кўра процедура ёки функция кўринишида аниқлаши мумкин. Одатда олинадиган натижа ягона қийматли бўлса функциядан, олинадиган натижалар сони бир нечта бўлса процедурадан фойдаланиш мақсадга мувофиқдир.

Процедурани ёзиш структураси худди асосий программа структураси каби бўлиб, фақат сарлавҳалари билангина фарқ қилади холос :

```
procedure <процедура исми>(<формал параметрлар рўйхати>);
label <меткалар рўйхати>;
const <ўзгармасларни  киритиш>;
type <янги типларни аниқлаш>;
var <ўзгарувчиларнинг типларини эълон қилиш>;
<қисм программагагина тегишли бўлган ички процедура ва
функциялар эълони>;
begin
      <процедуранинг тана қисми>
end;
```

Процедуралар ва функцияларни аниқлаш асосий программанинг ***var*** (ўзгарувчиларнинг типларини эълон қилиш) бўлимида бажарилади. Процедурадан программада фойдаланиш учун унинг исми ва фактик параметрлар рўйхати ёзилади. Шунда процедура ўзига белгиланган ишни бажариб, ўзининг фактик параметрлари орқали асосий программага ўз натижасини беради.

Процедуранинг эълони ва унга мурожаат қилишни кейинчалик кўриладиган мисоллар орқали ўзлаштириб оламиз.

3.10.1.Параметрсиз процедуралар

Юқорида айтиб ўтганимиздек, процедура ҳисоблаб берган натижалар унинг фактик параметрлари орқали асосий программага узатилади. Лекин, айрим пайтларда процедура параметрсиз ҳам бўлиши мумкин. Бу ҳолда асосий программанинг барча параметрлари процедура параметрлари ролини бажаради. Параметрсиз процедурада ҳам процедуранинг барча бўлимлари сақланиб қолади, фақат параметрлар рўйхатигина қатнашмайди.
Процедураларни аниқлаш ва улардан фойдаланишни қуйидаги мисол устида кўриб чиқайлик:
Мисол: u қmax (x Қ y, x = y), v қ max (0.5, u) – берилган x ва y ҳақиқий сонлардан фойдаланиб u ва v қийматларни аниқлаш.
бу ерда x, y - қийматлари киритиладиган ҳақиқий типли ўзгарувчилар.
1. Масалани ечиш программасининг процедурадан фойдаланмай тузилган ҳоли:

```
Program  max;
var
x, y, u, v: real;
a, b, s: real;
begin
{x, y - миқдорларни киритиш};
     readln (x,y);
a:қ x Қy;  b:қ x=y;
     if  a > b  then S:қ a else S:қb;
u :қ S;
     a:қ 0.5; b:қu;
     if  a > b  then  S:қ a   else  S:қb;
     v:қS;
{олинган натижалар};
     writeln (u, v)
end.
```

Аҳамият берсангиз, программадаги шартли оператор икки марта такрорланиб, бир хил иш бажарди.
2. Масалани ечиш программасини параметрсиз процедурадан фойдаланиб тузилган ҳоли (энди юқоридаги программада йўл қўйилган камчиликни процедуралар орқали тузатишга ҳаракат қиламиз):

```
Program max;
var x, y, u, v: real; a, b, S: real;
procedure  max1;
begin
if  a>b  then  S:қa  else  S:қb;
end;
begin
readln (x, y);
     a:қ x Қ y;  b:қx = y;
 max1; {max1 процедурасига 1-марта мурожаат    қилинмоқда}
```

```
 u:қS;
        a:қ 0.5;   b:қ u;
max1;  {max1 процедурасига 2-марта мурожаат қилинмоқда}
v:қS;
        writeln (u,v);
end.
```

Асосий программанинг операторлар қисмида икки марта ёзилган max1 параметрсиз процедурасига мурожаат, эълон қилинган процедурани икки марта асосий программага олиб келиб ишлатишни ташкил қилади. Аҳамият берилса, иккинчи программа биринчи процедурасиз тузилган программага кўра ихчамроқ ва соддароқдир. Биз киритган процедура ҳозирча фақат иккита ҳақиқий сон ичидан каттасини аниқлаб берди холос, шунинг учун программа матнинг хажмини камайтиришдан эришган ютуқ салмоқли бўлмади. Лекин, процедуралар асосан кўп хажмли матндаги амалларни, вазифаларни бажаришга мўлжалланади ва бу ҳолда эришилган ютуқ салмоғи анча юқори бўлади.

Параметрсиз процедуранинг асосий камчилиги, унинг асосий программага ва ундаги маълум параметрларга боғланиб қолганлигидир.

3.10.2.Параметрли процедуралар

Процедура билан асосий программани боғлайдиган асосий фактор бу – процедура параметрларидир. Параметрларни иккита типга ажратилади: қийматли параметрлар (параметр-қиймат), ўзгарувчили параметрлар (пара-метр - ўзгарувчи).

Параметр - қиймат бу процедурани ишлаш жараёнини таъминловчи параметрлар ҳисобланади, яъни асосий программа қийматларини процедурага узатадиган параметрлардир.

Энди, юқорида кўриб чиқилган сонларни энг каттасини топиш алгоритмининг программасини қийматли параметр билан ёзилган процедуралар орқали амалга оширайлик:

```
Program max;
var x, y, u, v: real;  S: real;
procedure max2 ( a, b: real );
begin
if a>b  then S:қa  else  S:қb;
end;
begin
read (x, y);
        max2 (x Қ y, x = y);  u:қS;
        max2 (0.5 , u);  v:қS;
        writeln (u, v)
end.
```

бу ерда a, b - процедуранинг қийматли формал параметрлари.

Процедурага мурожаат қилишда формал ва фактик параметрларнинг типлари ўзаро мос келиши керак, акс ҳолда программа хато тузилган

ҳисобланади. Юқоридаги программадан кўриниб турибдики, a ва b формал параметрлар ўрнига натижавий қийматлари маълум ифодалар қўйилди. Демак, қийматли фактик параметрлар ўрнига, шу типли натижага эришувчи ифода ёзилиши мумкин. Бундан ташқари, процедурада киритилган a ва b параметрлари фақат процедуранинг ичидагина маънога эга, ташқарида, мисол учун асосий программада улар тушунарсиз, қийматлари аниқланмаган миқдорлардир. Шунинг учун, қийматли параметрларга процедура натижаларини ўзлаштириб, асосий программага узатиб бўлмайди.

Юқорида тузилган программанинг асосий камчилиги, топилган катта сон доим S ўзгарувчисига ўзлаштирилаяпти. Мисолимиз шартига кўра, натижалар u ва v ўзгарувчиларига ўзлаштирилиши керак. Шунинг учун, программада икки марта қўшимча u:қS ва v:қS ўзлаштириш операторлари ёзилди.

Бу камчиликни тузатиш учун процедурага яна бир параметрни киритамиз. Лекин, киритилган бу параметр процедурага қиймат олиб кирмайди балки, процедура натижасини асосий программага олиб чиқиб кетади. Бундай параметрни параметр - ўзгарувчи деб аталади. Параметр-ўзгарувчини параметр-қийматдан фарқ қилиш учун процедурани аниқлашдаги параметрлар рўйхатида ўзгарувчи олдидан ***var*** хизматчи сўзи ёзилади. Параметр - ўзгарувчидан сўнг албатта, унинг типи кўрсатиб қўйилади. Юқорида айтганимиздек, формал параметр - қиймат ўрнига процедурага мурожаат вақтида шу типли ифода ёзиш мумкин бўлса, параметр - ўзгарувчи учун бу ҳол мутлақо мумкин эмас.

Процедурани мукаммаллаштириб бориш динамикасини ҳис этиш учун яна, юқорида кўрилган максимум топиш мисолининг программасини параметр - ўзгарувчи ишлатган ҳолда кўриб чиқамиз:

```
Program max;
var
x, y, u, v: real;
procedure  max3(a, b: real; var S: real);
begin
if  a>b  then  S:қa  else  S:қb;
end;
begin
readln (x, y);
max3( x Қ y, x = y, u); {x+y ва x=y ифодаларининг каттаси u
ўзгарувчисига ўзлаштирилмоқда}
max3( 0.5, u, v); {0.5 ва u ифодаларининг каттаси V ўзгарувчисига
ўзлаштирилмоқда}
        writeln( u, v)
end.
```

Шундай қилиб, битта программани процедуранинг уч хил варианти учун тузиб чиқиб, натижада ихчам ва содда программага эга бўлдик. Процедураларни аниқлашда шу пайттача оддий типли параметрлардан фойдаланиб келдик. Лекин, биз шуни яхши биламизки, Паскал тилида ҳосилавий типлар ҳам мавжуд. Параметр

- ўзгарувчига ҳосилавий, янги типлар бериш худди оддий скаляр тип бериш каби амалга оширилаверади. Аммо, параметр - қийматларда янги типлар масаласига батафсилроқ ёндашиш керак.
Биз юқорида эслатиб ўтдикки, фактик параметр формал параметр - қийматга мос типли ихтиёрий ифода бўлиши мумкин. Лекин, Паскал тилида ихтиёрий типли қийматлар учун шу типдаги натижа берувчи ҳеч қандай амал кўзда тутилмаган. Шунинг учун, бу типлар учун фактик параметрлар фақат шу типга мос ўзгарувчилар бўлиши мумкин холос. Бундай ҳол, хусусий ҳолда массивлар учун ҳам ўринлидир.
Фараз қилайлик, программада ўзгарувчилар қуйидагича эълон қилинган бўлсин:

```
const  nқ20;
type vector қ array *1..N+  of  real;
var     u, v: real;
        x, y: vector;
```

Бу ерда uқ max{x_i}, vқ max {y_i} ларни аниқлаш талаб қилинаётган бўлсин.
Векторнинг энг катта ҳадини топишни албатта процедура кўринишида ташкил қиламиз:

```
procedure max1 (A:vector;  var S: real);
var  i: integer;
begin
S:қA*1+;
        for i:қ2  to  n  do  if  A*i+ > S    then   S:қ A*i+
end.
```

Бу процедурага асосий программада мурожаат

```
max1 (x, u);  max1 (y, v);
```

кўринишида амалга оширилади.
Процедурадаги А векторини параметр - қиймат сифатида ёзиб қўйганимиз учун, процедурага қилинаётган ҳар бир мурожаатда А векторга мос равишда X ва Y векторлари кўчириб ёзилади ва сўнг процедура ўз ишини бажаради. Биз биламизки, бир тарафдан, массивларнинг устида кўчириш амалини бажаришга анча вақт кетади, иккинчи тарафдан, ҳар сафар янгидан процедурага қилинган мурожаатда А вектор учун хотирадан қўшимча жой ажратилади. Шунинг учун, процедуранинг сарлавҳасида қуйидагича алмаштириш қилсак, юқоридаги икки камчиликни бартараф қилган бўламиз:

```
procedure max1 (var A: vector; var S: real);
```

Энди процедурани эълон қилиш, ундан фойдаланиб программа яратиш малакасини ҳосил қилганимиздан сўнг, уни эълон қилишнинг синтаксис қоидаларини кўриб чиқайлик.
Процедурани аниқлаш (эълон қилиш) қуйидагича амалга оширилади:
<процедурани аниқлаш>::=<процедура сарлавҳаси>;<блок>
Бу ерда <блок> тушунчаси тўлиқлигича <программа танаси> тушунчаси билан бир хил синтаксис қоида асосида аниқлангани учун, бу тушунчага ортиқ қайтиб ўтирмаймиз.

Энди эса <процедура сарлавхаси>га таъриф берамиз:
<процедура сарлавҳаси>::қProcedure <процедура исми>| Procedure <процедура исми>(<формал параметрлар рўйхати>)
Процедура исми дастурчи томонидан танланадиган оддий идентификатор ҳисобланади.
Формал параметрлар рўйхати қуйидагича аниқланади:
< формал параметрлар рўйхати >::қ<формал параметрлар секцияси > {; < формал параметрлар секцияси >}
Формал параметрлар секцияси деганда процедура параметрларининг параметр-қиймат ва параметр-ўзгарувчи лардан иборат бўлишлиги тушунилади:
< формал параметрлар секцияси >::қ{< исм > {, < исм >}: <исм типи> | var <исм>{,<исм>}:<исм типи>
бу ерда <исм> - формал параметрлар сифатида ишлатиладиган идентификатор.
Энди юқоридаги аниқлашларга тушунтиришлар бериб ўтсак.
Юқоридаги Бекус-Наур формулаларидан кўриниб турибдики, формал параметрлар рўйхати (агар у мавжуд бўлса) битта ёки бир нечта ўзаро нуқта- вергул (;) белгиси билан ажратилган секциялардан ташкил топган. Ҳар бир секцияда эса ўз навбатида, битта ёки бир нечта ўзаро нуқта-вергул билан ажратилган формал параметрлар қатнашиши мумкин. Процедурадаги формал параметрлар сонини, дастурчининг ўзи процедурани аниқлаш мохиятидан келиб чиққан ҳолда танлайди.
Мисол:

```
Procedure P(A:Char; B:Char; Var C:Real; Var D:Real; E:Char);
```

бу ерда формал параметрлар рўйхати бешта секциядан иборат: A,B,E – лар Char типли қийматлар, C, D – лар Real типидаги ўзгарувчилар. Шу билан бир қаторда ҳар бир секция фақат, битта параметрни ўз ичига олмоқда.
Бир хил типли, ҳамда кетма-кет жойлашган қийматларни ва ўзгарувчиларни битта секцияга бирлаштириб процедура сарлавҳасини қуйидагича ёзиш ҳам мумкин:

```
Procedure P(A,B:Char; Var C,D:Real; E:Char);
```

Шундай қилиб, секция деганда бир хил типли параметр - қийматлар ёки параметр - ўзгарувчиларнинг рўйхатини тушуниш мумкин.
Кўпчилик бошловчи дастурчилар йўл қўядиган қуйидаги хатоликлардан эҳтиёт бўлмоқ зарур:

```
Procedure P(Var X:Real; Y:Real);
```

бу сарлавҳа

```
Procedure P(Var X,Y:Real);
```

сарлавхаси билан бир хил эмас.
Аниқланган процедурага мурожаат ёки процедура операторидан қандай фойдаланишни аниқлашни кўриб чиқайлик (яратилган процедурани «Активлаштириш», яъни ишлатиш):

<процедура оператори>::қ<процедура исми>|<процедура исми>(<фактик параметрлар рўйхати>)

Агар процедура аниқланишида параметрсиз бўлса, унга мурожаат қилиш ҳам фақат, процедура исмини ёзиш билангина амалга оширилади.

Агар процедура аниқланишида параметрли бўлса, албатта процедура-оператор ҳам унга мос фактик параметрлар рўйхатига эга бўлади. Шу параметрлар орқали процедурага мурожаат қилинаётганида, формал параметрлар фаоллаштирилади:
< фактик параметрлар рўйхати >::қ< фактик параметр > {, < фактик параметр >}

3.11. Локаллаштириш принципи

Ҳажми катта ва мураккаб программаларни ишлаб чиқишда, табиийки катта қийинчиликларга дуч келинади. Катта, комплекс программаларни зарур муддатда яратишга битта дастурчининг эса вақти етмайди. Бундай ҳолларда, яъни муҳим аҳамиятга эга бўлган ва қисқа муддатларда яратилиш керак бўлган программаларни ишлаб чиқиш учун дастурчиларнинг катта гуруҳини жалб этишга тўғри келади. Бундай, ягона программани яратишдаги паралел иш олиб боришда процедура ва функцияларнинг роли жуда катта бўлади. Бажарилиши керак бўлган ишни мустақил бўлимларга ажратилиб, ҳар бир мустақил иш алоҳида программаланиб, кейинчалик улар ягона - асосий программага бирлаштирилади.
Асосий программада ишлатилувчи ўзгарувчилар ва процедура параметрларини қандай танлаб олиш керак деган муаммо, бажариладиган ишнинг энг оғир қисмларидан бири бўлиб қолади. Агар уларни бир-бирларига боғлаб юборилса у ҳолда асосий программадаги бирор ўзгарувчига киритилган ўзгартириш, процедурада ишлатилган ва шу ўзгарувчига боғлиқ барча ишларни қайтадан таҳлил қилиб, текшириб чиқишга олиб келади. Бундай чалкаш ва оғир ишни бажаришнинг қийинлиги программа тузишда параллел, бир нечта дастурчининг иш олиб боришига ҳалақит беради.
Шунинг учун, процедура ва функцияларни ёзишда ҳар бир программага ўзи ечаётган масалага мувофиқ ҳолда, турли хил ички ўзгарувчилар, программали объектлар ўзгарувчиларининг турли қийматларини танлаб олиш ҳуқуқи берилади. Хаттоки, битта ўзгарувчини турли хил вазифаларда ишлатса ҳам бўлади. Паскал тилида бундай масалани ҳал қилиш учун локаллаштириш принципи ишлаб чиқилган, яъни процедура ёки функцияда ишлатилган ўзгарувчи шу процедура ёки функциянинг таъсир доирасида (ичида) гина ўз қийматини сақлаб қолади. Процедура ва функцияларнинг ичида аниқланиб, қийматланган ўзгарувчиларни локал (ички) ўзгарувчилар деб аталади. Ташқарида, яъни асосий программада киритилган ўзгарувчилар эса умуман олганда программанинг

ихтиёрий жойида ўз қийматини сақлаб қола олади. Бу ўзгарувчиларни глобал (ташқи) ўзгарувчилар деб аталади.
Қуйидаги мисолда локаллаштириш принципи яққол кўзга ташланади:

```
Program L1;
 const
 n қ 1;
 var
 t: real;
 x: char;
 procedure P (x, y: real);
 var n: real;
 begin
   n:қ xҚt; t:қy;
   writeln( n, t, x);
 end;
begin
  t:қ nF2; x:қ 'Қ';
 P(n,0.8); writeln(n,t,x);
end.
```

бу ерда t – асосий программанинг глобал ўзгарувчиси;
x, y – P процедурасининг формал параметрлар;
n – P процедурадаги локал ўзгарувчи.

3.12. Процедура - функциялар

Математика курсидан функция тушунчаси бизга яхши таниш бўлиб, унинг ёрдамида функция ва аргумент ўртасидаги боғлиқлик аниқланади. Паскал тилида ҳам функция тушунчаси киритилган бўлиб, уни шартли равишда икки турга ажратсак бўлади: стандарт функциялар, дастурчи томонидан аниқланган процедура - функциялар. Стандарт функциялар ҳар бир алгоритмик тил учун аниқланиб, амалда кўп учраб турувчи функцияларнинг қийматларини ҳисоблаб беришга мўлжалланган. Масалан, sin (x), cos(x), exp(x), abs(x), sprt(x) ва ҳ.к.
Худди стандарт функциялар каби дастурчи ҳам ўзи учун зарур, мустақил программа объектларини функциялар кўринишида аниқлаб, ундан керакли пайтда фойдаланиши мумкин.
Функция Паскал тилида қуйидаги структура бўйича аниқланади:

```
function <функция исми>(<формал параметрлар рўйхати>):
<функция қийматининг типи>;
label
      <меткалар рўйхати>;
const
      <ўзгармасларнинг қийматларини аниқлаш>;
type  <янги типларни киритиш>;
var
      <ўзгарувчиларнинг типларини эълон қилиш>;
```

<функция учун ички процедуралар ва функцияларни аниқлаш>;
begin
<функциянинг тана қисми>
end;

Юқорида эслатиб ўтганимиздек, функциялар ҳам процедуралар каби мустақил программалар ҳисобланиб, асосий программа орқали бошқарилади ва худди асосий программа ва процедурага ўхшаш структурада ёзилади.

Функцияни ҳам процедура каби программанинг ***var*** (ўзгарувчилар типларини эълон қилиш) бўлимида аниқлаб қўйилади. Процедура учун айтилган гапларнинг деярли барчаси функция учун ҳам ўринлидир. Функциянинг процедурадан асосий фарқи қуйидагилардир:

- функция сарлавҳаси бошқача аниқланади;
- функциянинг иши давомида олинадиган натижа функциянинг исмига ўзлаштирилади, яъни функциянинг тана қисмида албатта, функция исмига мос типли қиймат ўзлаштирилган бўлиши керак;
- функциядан асосий программага унинг исми орқали биттагина қиймат берилади.

Функцияга мурожаат ҳам худди процедурадаги каби амалга оширилади, лекин функциянинг мос типли ифодада қатнашиш каби қўшимча имконияти мавжуд.

Энди функцияни аниқлаш ва унга мурожаат қилишни тўлиқроқ ўрганиш учун қуйидаги мисолни эътиборингизга ҳавола қиламиз:

Мисол: f(n) қ n! (n! қ 1 = 2 = 3 = ...= n - факториал) функциядан фойдаланиб,

$Y = \frac{20!+3!}{5!+31!} * \frac{(k+1)!}{m!}$ - ифодани ҳисоблашни ташкил қилинг:

```
Program L1;
var
k, m, i :integer; y: real;
function fact (n: integer): integer;
var
j: integer; P: byte;
begin
j:қ1;
for p:қ1 to  n do j:қ j = p;
fact :қj;
end;
begin
readln (k, m);
        y:қ (fact (20) Қ fact(3))F(fact(5) Қ fact (31)) = fact(kҚ1) F
fact(m);
        writeln( y);
end.
```

Функцияларни аниқлашда доим шундай ҳаракат қилиш лозимки, унинг тана қисмида формал параметрлар ва функцияни аниқлаш учун зарур бўлган локал ўзгарувчиларгина қатнашсин.
Программанинг глобал ўзгарувчисига иложи борича процедура ёки функция ичидан туриб қиймат бермаслик керак, акс ҳолда программа хато натижа бериши мумкин.
Мисол:

```
Program m1;
Var x,y: integer;
Funktion  f(t: integer): integer;
Begin
    f:қt=t;
    x:қ7;
end;
begin
    x:қ5; writeln(x);
    y:қf(2)Қx
    writeln(x,y)
end.
```

Бу программанинг ишлаши натижасида хқ5, уқ11 ва хқ7 қийматлар экранга чиқарилади, яъни функциянинг ички қисмидаги хқ7 қиймати асосий программадаги натижавий қийматларга ўз таъсирини ўтказмоқда.

Паскал тилида процедура – функциялар билан ишлашда, функцияларнинг рекурсивлик хоссасидан фойдаланиш имконияти яратилган.
Рекурсия тушунчасига мисол қилиб оддий факториал ҳисоблашни келтириш мумкин:

$$n! = \begin{cases} 1 & агар \quad n = 0 \\ n \cdot (n-1)! & агар \quad n > 0 \end{cases}$$

бу ерда кўриниб турибдики n! қиймати (n-1)! орқали аниқланаяпти, яъни рекурсия дегани ўзи орқали ўзини аниқлаш маъносини англатади.
Паскал тили ҳам функцияларни рекурсив аниқлаш имкониятини беради. Функцияни рекурсив аниқлаш унинг тана қисмида ўзига - ўзи мурожаат қилиш орқали амалга оширилади.
Юқоридаги факториал ҳисоблашни рекурсив функциялар орқали амалга оширайлик:

```
program L1;
var
n: integer; y: integer;
function fact(m: integer): integer;
var
k: integer;
begin
if   mқ0  then fact:қ1 else fact:қ fact(m-1) = m;
end;
begin
readln (n);
```

```
        y:қ fact (n);
        writeln(y);
end.
```

Функцияларни рекурсив аниқлаш қисқа ва тушунарли тилда бўлади, рекурсив эмас аниқлаш эса узоқ ва функцияни кўриниш эффектини бузади, лекин биринчи ҳолда сарфланган ЭҲМ вақти ва хотира нисбатан анча юқоридир.

Юқорида кўриб чиқилган ва аниқланган барча процедура ва функцияларнинг параметрлари ёки қандайдир типли қиймат, ёки ўзгарувчилар бўлган эди. Аммо, шундай ҳоллар ҳам учраб турадики айрим параметрларни функциялар ёки процедуралар орқали аниқлаш лозим бўлади. Бу ҳолга мисол сифатида

$$y=\int_a^b f(x)dx \quad \text{ва} \quad z=\int_c^d g(x)dx$$

аниқ интегралларни трапеция усулида тақрибий ҳисоблаш программасини кўриб чиқамиз.

Бу ерда a, b, c, d - қийматлари бериладиган ўзгарувчилар;

f(x) қe^{2x} Қ sin6x, g(x)қ x^2 - $3x^3$cosx

Аниқ интегралларни трапеция усули ёрдамида ҳисоблаш алгоритми қуйидаги формула асосида бажарилади:

$$y=\int_a^b f(x)dx \approx h\left[\frac{f(a)+f(b)}{2}+\sum_{k=1}^{n-1} f(a+kh)\right]$$

бу ерда $h=\frac{b-a}{n}$, n - * a, b+ оралиқни бўлишлар сони.

```
Program  L1;
var
a, b, c, d, y, z: real;
function f(x: real) : real;
begin
f:қexp(2=x)Қsin(6=x)
end;
function g(x: real) : real;
begin
g:қsqr(x) - 3=x=sqr(x)=cos(x)
end;
procedure  int(A, B: real; function F(x: real): real; var R: real);
const
nқ20;
var
x, h: real; k: integer;
begin
h:қ (B - A)Fn;  R:қ(f(A)Қf(B))F2;
for k:қ1  to  n-1  do
          R:қ RҚ f(a Қ k = h);
R:қ R = h;
end;
```

{асосий программанинг операторлар бўлими}

```
begin {1 - интеграл чегараларини киритинг}
```

```
        read ( a, b);
    int (a, b, f, y);
        writeln( 'уқ',y);
{2-интеграл чегараларини киритинг}
    read (c, d); int ( c, d, g, z);
    write ('zқ', z);
    end.
```

3.13.Турбо-Паскалда модуллар

3.13.1. Турбо-Паскалнинг модуллари ва фойдаланувчи модулини яратиш.

Шахсий компьютерларнинг энг катта камчилиги уларда амалий программалар кутубхонасининг тўлиқ эмаслигидадир. Катта ЭҲМларда программа тузувчилар учун жуда катта программалар кутубхонаси хизмат қилар ва улардан фойдаланиб тузилган программалар ўзларининг ишончлилик даражаси билан юқори турар эди. Шахсий компютерларнинг бу камчилигини йўқотиш учун Турбо-Паскалда модуллар тушунчаси киритилган. Умуман олганда, ҳар бир малакали программа тузувчи ўз программасини процедура ва функциялардан фойдаланиб тузади. Лекин, бу процедура ва функциялардан бошқа программаларда фойдаланиш учун уларнинг матнларини қайта кўчириб ёзиш лозим бўлади.
Турбо-Паскалда бу масалани ечиш учун модуллар яратилиб, уларни компиляция қилинади ва бу модулдан бошқа программаларда бемалол фойдаланилаверилади.
Турбо-Паскал тилининг яратувчилари қуйидаги зарур ва фойдали модулларни яратиб, дастурчилар учун жуда катта қулайликлар яратишган:

1) System модули - стандарт процедура ва функцияларни ўз ичига олиб, автоматик тарзда барча программалар учун очиқдир;
2) DOS модули - MS DOS операцион системаси билан ишлашни ташкил қилувчи функция ва процедуралардан ташкил топган;
3) Crt модули – экран, клавиатура ва IBM русумидаги компьютерларнинг товушли динамиги билан ишлаш процедураларини ўз ичига олган;
4) Graph модули - компьютернинг график имкониятларидан фойдаланиб яратилган функция ва процедураларнинг катта тўплами;
5) Printer модули - бу кичкинагина модул принтер қурилмаси билан ишлашни осонлаштиради;
6) Overlay модули – катта программаларни бир нечта бўлакларга ажратишнинг кучли воситаси бўлиб, бир қанча процедуралар ва функциялардан ташкил топган.

Модуллардан фойдаланиш учун программа сарлавҳасидан

Program <программа номи>;
кейин қуйидаги қатор ёзилиши керак:
Uses <модул исми>;
Агар программада бир нечта модул ишлатилса, уларнинг исмлари кетма-кет ёзиб қўйилади:
Uses <модул исми1>,<модул исми2,>,...,<модул исмиN>;
Турбо-Паскал бизга ўзимизнинг модулларимизни яратиб олиш имконини ҳам беради. Фойдаланувчи модуллари қуйидаги структурада бўлади:
Unit<модул исми>;
Interface
...
{ очиқ эълонлар бўлими - интерфейс секцияси}
...
Implementation
...
{ ёпиқ эълонлар бўлими}
...
Begin
...
{ Инициализация бўлими}
...
End.
Агар модул ўз ичида бошқа модуллардан фойдаланса Interface хизматчи сўзидан кейин
Uses <модуллар рўйхати>;
ёзилади.
Интерфейсли бўлим модулнинг бир қисми бўлиб, ***Interface*** ва ***Implementation*** сўзлари орасида жойлашади. Бу бўлимда ўзгармаслар, маълумотлар типи, ўзгарувчилар, процедура ва функцияларни аниқлаш мумкин. Бу киритилганлар мазкур модулда қатнашувчи барча программалар ва модулларда бемалол ишлатилиши мумкин. Бўлимда санаб ўтилган процедура ва функцияларнинг тана қисмлари ***Implementation*** сўзидан кейин аниқланади (уларнинг сарлавҳалари айнан сақланиб қолиши керак). Бу бўлимда ҳам, фақат шу бўлим учунгина "Кўринадиган" (ишлатиши мумкин бўлган) эълонлар бўлими қатнашиши мумкин. Инициализация секцияси ***Begin*** ва ***End*** сўзлари ичига олиб ёзилади. Агар ***Begin*** сўзи тушириб қолдирилган бўлса, демак бу секция йўқ ҳисобланади. Инициализация секциясида бошқаришни асосий программага узатгунгача бажариладиган операторлар жойлашган бўлади. Бу операторлар асосан программани ишга туширишга тайёрлаб беради.
Мисол сифатида X ва Y бунинг сонларининг максимуми ва минимумини аниқловчи модулни яратайлик:
Unit Stud;
Interface {очиқ эълонлар бўлими – интерфейс секцияси}

```
    function min(x,y:integer):integer;
    function max(x,y:integer):integer;
Implementation     {ёпиқ эълонлар бўлими}
function min(x,y:integer):integer;
Begin
    if  x<қу        then min:қх else  min:қу;
End;
function max(x,y:integer):integer;
Begin
   if  x>y   then max:қх else  max:қу;
End;
Begin
  {Инициализация секцияси йўқ}
End.
```

Биз зарур модулни ҳосил қилдик, энди уни компиляция қилишимиз лозим. Компиляция натижасида ***Stud.tpu*** исмли файл ҳосил қилиниши керак. Компиляция қилинмаган модулнинг исми эса шунга мос ҳолда ***Stud.pas*** бўлиши керак.

Бу модулдан фойдаланиш программаси қуйидагича бўлиши мумкин:

```
Uses Stud;
Var
   A,b,c,d:integer;
Begin
   Write('A ва B ларни киритинг >');
   Readln(a,b);
   C:қmax(a,b);
   Writeln('Махқ ',C);
    C:қmin(a,b);
    Writeln('Minқ ',C);
    D:қmax(a,b)Қmin(a,b);
    Writeln('MaxҚMinқ',D);
End.
```

Қуйида эса экран рангини танлаш модули мисол сифатида кўрсатилган:

```
Unit Colors;
Interfase
Type
     Colortype қArray*0..15+ of Byte;
Const
     Black:bytеқ0;blue:bytеқ1;
     Green:bytеқ2;cyan:bytеқ3;
     Red:bytеқ4;magenta:bytеқ5;
     Brown:bytеқ6;lightgray:bytеқ7;
     Darkgray:bytеқ8;lightblue:bytеқ9;
     Lightgreen:bytеқ10;lightcyan:bytеқ11;
     Lightred:bytеқ12;lightmagenta:bytеқ13;
     Yellow:bytеқ14;white:bytеқ15;
Var
      Currcolors:colortype absolute Black;
      Procedure setMonoColors;
     Procedure setColorColors;
Implementation
```

```
Const
      ColorColors:Colortypeқ(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15) ;
       MonoColors:ColorTypeқ(0,1,7,7,7,7,7,7,7,7,7,7,7,7,15,15);
Procedure SetMonoColors;
Begin
       CurrColors:қMonoColors;
End;
Procedure SetColorColors;
Begin
        CurrColors:қColorColors;
End;
Var
        Ch:Char;
Begin
   Write(
    Readln(ch);
    If ch in *‘M’,’m’,’M’,’m’+ then SetMonoColors;
End.
```

3.13.2. System модулининг процедура ва функциялари

Юқорида айттанимиздек, System модулининг процедуралари ва функциялари барча программалар учун очиқ бўлиб, улардан кенг фойдаланиш мумкин. System модулининг исмини модуллар рўйхатида кўрсатиш шарт эмас.

Қуйида ушбу модулнинг маълум бир қисм процедуралар ва функциялари билан қисқача танишиб чиқамиз:

1) Программа ишини бажариш процедуралари.

Exit процедураси

Вазифаси: актив ишчи блокидан чиқиш, бажарилаётган ишни якунлаш;

Аниқланиши: Exit.

Halt процедураси. Вазифаси: программа бажарилишини тўхтатади ва ОС га бошқаришни қайтаради;

Аниқланиши: Halt*(ExitCode:Word)+;

бу ерда мажбурий бўлмаган (ўрта қавс мажбурий эмас белгиси) ExitCode параметри программанинг якунланиш кодини беради, агар бу параметр бўлмаса бу код нолга тенг бўлади.

RunError процедураси. Вазифаси: программанинг бажарилишини тўхтатади ва бажарилиш вақтидаги хатоларни аниқлайди;

Аниқланиши: RunError*(ErrorCode:Word)+;

бу ерда ErrorCode параметрининг хатолик номери ҳақидаги маълумоти экранга чоп этилади.

2) Типларни алмаштириш функциялари.

Chr функцияси.Вазифаси: ASCII жадвалидаги тартиб рақами берилган бутун сонга мос бўлган белгини аниқлайди;

Аниқланиши: Chr(N:Byte):Char;

бу ерда N белгининг ASCII жадвалидаги тартиб рақамини ифодаловчи бутун, мусбат сон.

Ord функцияси.Вазифаси: саналма типли қийматлар бўйича унинг тартиб сонини аниқлаш;

Аниқланиши: Ord(X):LongInt;

бу ерда X – саналма типли қиймат.

Round функцияси.Вазифаси: ҳақиқий типли қийматни яхлитлаб, катта бутун, сон ҳосил қилади;

Аниқланиши: Round(X:Real):LongInt;

Trunc функцияси.Вазифаси: ҳақиқий типли қийматнинг каср қисмини ташлаб юбориб, бутун сон ҳосил қилади;

Аниқланиши: Trunc(X:Real):LongInt;

3) Арифметик функциялар.

бу функцияларнинг ёзилиши ва улардан фойдаланиш қоидалари ўзлаштириш оператори мавзусида тўлиқ бериб ўтилганлиги учун, уларга тўхталиб ўтирмаймиз.

4) Саналма тип функциялари ва процедуралари.

Dec процедураси. Вазифаси: ўзгарувчи қийматини камайтиради;

Аниқланиши: Dec(Var X*;n+:LongInt);

бу ерда ёзилиши мажбурий бўлмаган “n” ўзгарувчи аргументнинг қийматини қанчага камайтириш лозимлигини кўрсатади. Бу ўзгарувчи ёзилмаса аргумент қиймати бир сонига камаяди.

Inc процедураси. Вазифаси: ўзгарувчи қийматини орттиради;

Аниқланиши: Inc(Var X*;n+: Integer);

бу ерда Dec процедурасига тескари иш бажарилади.

Odd функцияси.Вазифаси: аргументнинг тоқ ёки жуфт сонлигини текширади;

Аниқланиши: Odd(X:LongInt):Boolean;

бу ерда натижавий қиймат рост (True) бўлса сон тоқ, акс ҳолда сон жуфт.

Pred процедураси.Вазифаси: аргуметни олдинги қийматини аниқлаш;

Аниқланиши: Pred(X);

Succ процедураси.Вазифаси: аргуметни кейинги қийматини аниқлаш;

Аниқланиши: Succ(X);

5) Сатрлар билан ишлаш функциялари ва процедуралари.

ConCat функцияси.Вазифаси: сатрларни кетма-кет улайди;

Аниқланиши: ConCat(S1*,S2,...,SN+:String):String;

бу эрда S1 қаторни кейинги санаб ўтилган қаторлар билан уларни ёзилиш тартибида улайди.

Copy функцияси.Вазифаси: сатр ичидан янги сатр ҳосил қилиш;

Аниқланиши: Copy(S:String;Index,Count:Ineger):String;

бу эрда S– берилган сатр;

Index–S сатрининг нечанчи белгисидан бошлаб, янги сатр ҳосил қилиш кераклигини аниқлайди;

Count – янги сатрдаги белгилар сони.

Delete процедураси.Вазифаси: берилган сатр ичидан сатр остини олиб ташлайди;
Аниқланиши: Delete(Var S:String;Index:Integer;Count:Integer);
бу эрда S – берилган сатр;
Index – шу тартиб рақамли белгидан бошлаб, S сатрдан сатр ости олиб ташланади;
Count – олиб ташланадиган сатрдаги белгилар сони.
Insert процедураси.Вазифаси: берилган сатрга янги сатр қўшади;
Аниқланиши: Insert(S1:String; Var S2:String; Index:Integer);
бу эрда S2–берилган сатр;
S1– қўшиладиган сатр;
Index–S2 сатрнинг қайси тартиб рақамли ҳадидан бошлаб янги сатр қўшилишини англатади.
Length функцияси.Вазифаси: сатр узунлигини аниқлайди;
Аниқланиши: Length(S:String):Integer;
бу ерда S сатридаги белгиларнинг сонини аниқланади:
Pos функцияси.Вазифаси: сатрдан сатр остини қидиради;
Аниқланиши: Pos(SubStr,S:String):Byte;
бу эрда SubStr – S сатрда қидирилаётган сатр ости.
Агар SubStr сатр ости S сатрида топилса, Pos функцияси мос келган биринчи белгининг тартиб рақамини беради, агар бу сатр ости S сатрда бўлмаса, функция нул қиймат беради.
Str процедураси.Вазифаси: сонли қийматни унинг сатрли кўринишига ўтказади;
Аниқланиши: Str(X*:Width*:decmals+ +;Var S:String);
бу ерда ёзилиши шарт бўлмаган Width ва decmals параметрлари мос равишда, S сатрининг ҳадлар сонини ва ҳақиқий соннинг вергулдан кейинги хадлар сонини ифодалайди.
Val процедураси.Вазифаси: сатрли қийматни унинг сонли кўринишига ўтказади;
Аниқланиши: Val(S:String;Var V; Var Code:Integer);
бу ерда S – берилган сатр;
V – S сатрнинг унинг сонли кўринишига ўтказганидан кейин ҳосил бўлган сонни сақлаш жойи;
Code – бутун типли ўзгарувчи.
Агар S сатрни сонга айлантириб бўлмаса Val процедураси бажарилганидан сўнг Code ўзгарувчиси нул қийматни қабул қилади.
6) Параметрлар билан ишлаш функциялари.
Программани ишга туширишда унга узатиладиган қийматларни параметрлар деб ҳисоблаймиз.
Мисол: Системали программалашнинг қуйидаги Copy буйруғини кўрайлик:

```
Copy  file.dat  c:₣prog₣file2.dat
```

яъни, активлашган каталогдаги file.dat файлининг нусхасини «C» дискнинг prog каталогига file2.dat номи билан кўчириш. Бу ерда

Copy буйруғига параметр сифатида ”file1.dat”ва “file2.dat” қатори узатилмоқда.
Турбо-Паскалда параметрлар билан ишлаш учун ParamCount ва ParamStr функциялари мавжуд.
ParamCount функцияси.Вазифаси: буйруқли қатордан программага узатилган параметрлар сонини аниқлайди;
Аниқланиши: ParamCount:Word;
ParamStr функцияси.Вазифаси: кўрсатилган номердаги параметрни аниқлайди;
Аниқланиши: ParamStr(N:Integer):String;
7) Адреслар билан ишлаш функцияси.
Addr функцияси.Вазифаси: кўрсатилган объектнинг адресини аниқлайди;
Аниқланиши: Addr(X):Pointer;
бу ерда X – ихтиёрий типли ўзгарувчи ёки программада эълон қилинган процедура ёки функция исми.
Seg функцияси.Вазифаси: кўрсатилган ўзгарувчи адресининг сегмент қийматини аниқлайди;
Аниқланиши: Seg(X):Word;
бу ерда X – ихтиёрий типли ўзгарувчи ёки программада эълон қилинган процедура ёки функция исми.
8) Бошқа процедура ва функциялар.
FillChar процедураси.Вазифаси: кўрсатилган қиймат билан кетма-кет келган ва чекли сондаги байтларни тўлдиради;
Аниқланиши: FillChar(Var X;Count:Word;Value);
Hi функцияси.Вазифаси: аргументнинг катта байтини аниқлайди;
Аниқланиши: Hi(X):Byte;
Lo функцияси.Вазифаси: аргументнинг кичик байтини аниқлайди;
Аниқланиши: Lo(X):Byte;
Move процедураси.Вазифаси: кўрсатилган миқдордаги байтлардан тезкор (оператив) хотиранинг бир қисмидан иккинчи қисмига нусха кўчиради;
Аниқланиши: Move(Var Cource,Dest;Count:Word);
Random функцияси.Вазифаси: тасодифий сонни аниқлайди;
Аниқланиши: Random(*Range:Word+);
бу ерда аниқланадиган тасодифий сон 0 ва Range сонлари оралиғида ётади.
Randomize процедураси.Вазифаси: тасодифий сонни ҳосил қилувчи генераторни ишга туширади;
Аниқланиши: Randomize;
UpCase функцияси.Вазифаси: кичик лотин ҳарфларни катта ҳарфларга ўтказади;
Аниқланиши: UpCase(Ch:Char):Char;
функция рус алфавити ҳарфларини ҳам катта ҳарфларга ўтказади.

3.13.3 Crt модулининг процедура ва функциялари.

Юқорида қисқача айтиб ўтганимиздек, Crt модули матнли экран, клавиатура ва товуш динамиги билан ишлашни ташкил этувчи программалар билан жиҳозланган.
Матнли режимдаги экран одатда 25 та қатор ва ҳар бир қаторда 80 та белги сиғдириш имкониятига эга. Экранда ҳосил қилиш мумкин бўлган ранглар сони эса 16 та (оқ-қора рангли экран рангини кўзда тутмаяпмиз).
Экранда керакли белгини зарур жойда чиқариш, матн ва фон рангини ўзгартириш ва экраннинг ишини тўлиқ бошқара олиш Crt модулининг вазифаларидан ҳисобланади:

- GotoXY(I,J:Byte) процедураси экраннинг I устун ва J сатрига курсорни кўчириб берди;
- Write(S) процедураси курсор турган жойдан S қаторни чоп этади;
- TextColor(Color:Byte) процедураси экранга чиқариладиган матн рангини ўрнатади;
- TextBackGround(Color:Byte) процедураси чоп этилувчи матн учун экран фонининг рангини ўрнатади;
- ClrScr процедураси TextBackGround процедураси ўрнатган ранг билан экранни тозалайди, агар ранг ўрнатилмаган бўлса экран қора ранг билан тозаланади, курсор экраннинг чап юқори қисмига чиқарилади;
- AssignCrt(Var F:Text) процедураси Crt модулини ўрнатиш учун матнли файлни таъминлайди;
- ClrEol процедураси курсор турган жойдан бошлаб экранни тозалайди. Агар Window процедураси билан ойна очилган бўлса тозалов шу ойна ичида бўлади. Курсор ўз ўрнини ўзгартирмайди;
- Delay(N:Word) процедураси N милли секунд давомида тўхтовни ташкил этиб беради;
- DelLine процедураси курсор турган қаторни ўчиради. Қолган қаторлар бир сатрга юқорига силжийди ва экраннинг охирги қатори тозаланади;
- HighVideo процедураси чоп этилаётган белгилар учун юқори ёруғлик ўрнатади;
- InsLine процедураси курсор турган жойга бўш қатор ўрнатади;
- KeyPressed:Boolean функцияси агар клавиш босилса True, клавиш босилмаган бўлса False қийматни қайтаради;
- LowVideo процедураси чоп этилаётган белгилар учун паст ёруғлик ўрнатади;
- NormVideo процедураси чоп этилаётган белгилар учун ўртача ёруғлик ўрнатади;

- NoSound процедураси товуш динамигини ўчиради;
- ReadKey:Char функцияси клавиатурадан белгини ўқийди;
- Sound(X:Word) процедураси товуш динамигини улайди ва уни берилган X частотада товуш чиқаришга мажбур қилади;
- TextMode(N:Integer) процедураси керакли N-матнли режимни ўрнатади;
- WhereX:Byte функцияси курсор турган жойнинг устун номерини (қаралаётган ойнада), яъни X координатасини аниқлайди;
- WhereY:Byte функцияси қаралаётган ойнага нисбатан курсор жойининг қатор номерини, яъни Y координатасини аниқлайди;
- Window(X1,Y1,X2,Y2:Byte) процедураси экранда матнли ойна ҳосил қилади. Бу ойна ўзини худди тўлиқ экрандек ҳис этади.

Ранглар билан ишлашни қулайлаштириш мақсадида Crt модулида қуйидаги ўзгармаслар киритилган:

```
Const
Blackқ0;{қора}
Blueқ1; {кўк}
Greenқ2; {зангор}
Cyanқ3; {ҳаворанг}
Redқ4; {қизил}
Magentaқ5; {малинали}
Brownқ6; { }
LightGrayқ7; {оч кулранг}
DarkGrayқ8; {тўқ кулранг}
LightBlueқ9; {оч ҳаворанг}
LightGreenқ10; {оч зангори}
LightCyanқ11; {оч}
LightRedқ12; {оч қизил}
LightMagentaқ13; {оч малинали}
Yellowқ14; {сариқ}
Whiteқ15; {оқ}
Blinkқ128; {ёниб-ўчиш}
```

3.13.4 MSDOS модулининг процедура ва функциялари

DOSнинг системали модули 6 килобайт атрофидаги жой эгаллаб, MSDOS муҳити билан ишлашга ва унинг имкониятларидан фойдаланишга мўлжалланган процедуралар ва функцияларга эга. Аслида бу процедуралар ва функциялар Паскал тилининг синтактик қоидаларига мослаштирилган MSDOSни чақириш функцияларидир. DOS модулидаги процедура ва функцияларни бажарадиган ишларининг маъноларига кўра олтита функционал гуруҳга ажратиш мумкин:

- MSDOS параметрларини сўрови ва уларни ўрнатиш;
- ШЭҲМда календар ва соатлар билан ишлаш;

- Диск ресурсларининг анализи;
- Каталог ва файллар билан ишлаш;
- MSDOS узилишлари(прерўвание) билан ишлаш;
- Резидент программалар ва субпроцесслар билан ишлаш.

1) MSDOS параметрларини сўрови ва уларни ўрнатишнинг процедуралари ва функциялари:

- DOSVersion:Word функцияси MSDOS версиясининг кодлаштирилган номерини аниқлайди;
- GetCbreak (Var B: Boolean)-процедураси Break параметрининг қийматини ўқийди;
- SetCbreak(Var B: Boolean) процедураси Break қийматини ўрнатади;
- GetVerify(Var V:Boolean) процедураси Verify параметрининг қийматини ўқийди;
- SetVerify(V:Boolean) процедураси Verify қийматини ўрнатади;
- EnvCount:Integer функцияси MSDOSнинг системали ўзгарувчилари сонини аниқлайди;
- EnvString(N:Integer):String функцияси N номерли MSDOS ўзгарувчисининг тўлиқ вазифа қаторини беради;
- GetEnv(E:String):String функцияси E системали ўзгарувчининг қийматини аниқлайди.

2) ШЭҲМда календар ва соатлар билан ишлаш процедура ва функциялари:

DOS модулида компьютер соати ва календар билан ишлаш ҳамда файлларни ташкил қилиш куни ва вақтини ўрнатиш процедуралари мавжуд:

- GetDate(Var Year,Month,Day,Dw:Word) ШЭҲМ соатидан йил, ой, кун ва ҳафта кунини ўқийди;
- SetDate(Year,Month,Day:Word) процедураси ШЭҲМнинг соатига йил, ой ва кунни ўрнатади;
- GetTime(Var Hour,Min,Sec,Sec100:Word) процедураси ШЭҲМ соатидан ҳозирги вақтни аниқлайди;
- SetTime(Var Hour,Min,Sec,Sec100:Word) процедураси ШЭҲМ соатига янги вақтни ўрнатади;
- PackTime(Var Dt:Datetime; Var T:Longint) процедураси кун,ой, йил ва вақт маълумотларини файлга ёзиш учун ихчам ҳолда тайёрлайди.
- UnPackTime(Var T:Longint;Var Dt:Datetime) процедураси файлдан ўқилган кун, ой, йил ва вақт ёзувини очиб беради;
- GetFtime(Var F; Var T:Longint) процедураси очиқ F файли учун кун, ой, йил ва вақтнинг ихчам ёзувини ўқиб беради;
- SetFtime(Var F; Var T:Longint) процедураси очиқ F файли учун йил, ой, кун ва вақтнинг ихчам ёзувини ёзиб қўяди.

3) Диск ресурслари таҳлилининг функциялари:

DOS модули ўз ичига диск ҳолатини таҳлил қилишнинг иккита функциясини олади:

- DiskFree(D:Word):Longint функцияси дискдаги бўш жой ўлчовини аниқлайди;
- DiskSize(D:Word):Longint функцияси дискнинг тўлиқ хажмини байтларда аниқлаб беради. Бу ерда D бутун қийматли параметр ёки бутун сон бўлиб, унинг қиймати аниқ бир дискни кўрсатади:
- Агар Dқ0 бўлса ишлатилаётган жорий диск, Dқ1 бўлса А диск, Dқ2
- бўлса В диск ва ҳоказолар таҳлил қилинади;
- Агар мазкур система Dнинг киритилган қийматига мос дискни топа олмаса функция -1га тенг қиймат беради;

4) Каталоглар ва файллар билан ишлаш функциялари ва процедуралари.

Паскал тилининг ташқи файллар билан ишлаш имкониятлари жуда чекланган бўлиб, улар асосан қуйидагилардан ташкил топган: файлни очиш, ёпиш, қайта ном бериш ва ўчириш. Бу камчиликларни бартараф қилиш учун DOS модулида бир қанча функциялар ва процедуралар кўзда тутилган:

- FindFirst(Path:String;Attr:Word;Var Sr:SearchRec) процедураси Path сўрови бўйича Attr атрибутли биринчи мос исмни топади;
- FindNext(Var Sr:SearchRec) процедураси FindFirst процедурасидан кейин чақирилиб, кейинги мос исмларни топиш учун ишлатилади;
- FSearch(Path:PathStr; DirList:String):PathStr функцияси DirList каталоглар рўйхатидан Path исмли файлни қидиради;
- GetFattr(Var F:File; var Fa:Word) процедураси F файли билан боғлиқли дискдаги файлнинг Fa атрибутини ўқийди;
- SetFattr(Var F:File; var Fa:Word) процедураси F файли билан боғлиқли дискдаги файлга атрибут ўрнатади;
- Fsplit(Path:PathStr; Var Dir:DirStr; Var Name:NameStr; Var Ext:ExtStr) процедураси Path файлининг тўлиқ исмини уни ташкил этувчиларига ажратади: Dir-йўли, Name-исми, Ext-кенгайтмаси.

5) MSDOS узилишлари билан ишлаш.

MSDOS нинг системали функцияларини чақириш узилишлар кўринишида ҳал қилинади. Ҳар бир узилиш активлаштирилгандан кейин турли хил функциялар тўпламига мурожаат қилиш имкониятини яратади. Масалан, 16H номерли узилиш, операцион системаси даражасидаги клавиатура билан мулоқотни ташкил этувчи функцияларга йўл очиб беради, 25H номерли узилиш эса дискни ўқиш ишларини бошқаришни ўз зиммасига олади ва ҳоказо. Жуда катта вазифалар 21H узилиши зиммасидадир, чунки у операцион система (ОС)ни ташкил этувчи ўнлаб функциялардан фойдаланишни ташкил қилиб беради.

Қуйида эътиборингизга Паскал-программасидан туриб MSDOSнинг функцияларига мурожаат қилишни ташкил қилиб берувчи DOS модули процедураларининг вазифалари ва аниқланишларини ҳавола қиламиз:

- **GetIntVec(N:byte; Var Adress:pointer)** процедураси Adress кўрсаткичига берилган N номерли узилиш қисм программаининг адресини аниқлаб беради;
- **SetIntVec(N:byte; Var Adress:pointer)** процедураси N-номерли узилишнинг янги қисм программасини DOS га ўрнатиб, адреснинг эски қийматини Adress кўрсткичига жойлайди;
- **Intr(N:byte; Var R:Register)** процедураси N программали узилишни активлаштириб, функция номери ва параметрларини R ўзгарувчисига узатади;
- **MsDos(Var R:Register)** процедураси 21H номерли узилишни махсус чақириш вазифасини бажаради.

Охирги процедурада янги Register типи киритилди. Бу тип DOS модулида аниқланган қуйидаги ёзувдан иборат;

```
Type
    registerқRecord
      case Integer of
            0:(AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags:Word);
            1:(AL,AH,BL,BH,CL,CH,DL,DH:Byte);
      end;
```

Бу типдаги ўзгарувчилар микропроцессор регистрларга Intr ва MsDos ларни чақиришда йўл очиш учун ишлатилади:
0-вариантида-16 разрядли регистрларга 1-вариантда эса 8 разрядли процессор ячейкаларига мурожаат қилинади.
6) Резидент программалар ва субпроцесс (субжараён)ларни ташкил килиш.
MS-DOS операцион системаси субпроцессларини ташкил қилиш бўйича катта имкониятларга эга. Субпроцессда бир программа бошқа бир программани ишга туширса, бошка бир программа эса ўз навбатида кейинги программани ишга туширади ва ҳакозо. Бу ҳолда ишга тушурувчи программа (процесс) субпроцессни ишини ташкил қилиб ўзи ишдан тўхтайди, лекин ШЭҲМ хотирасида сақланиб туради. Субпроцесс эса хотиранинг қолган қисмида ихтиёрий программа каби бажарилади. У ўз ишини якунлагач кейинги жараён программаси иш бошлайди.
MS-DOS нинг бу хоссаси интегратор-программаларни яратиш имкониятини ҳосил қилади. Ҳаммага таниш бўлган Norton Commander программаси субпроцесслардан актив фойдаланувчи, резидент бўлмаган программа ҳисобланади.
Оддий программалар ўз ишини якунлагач ШЭҲМ хотирасини бўшатиб қўяди. Субпроцесслар эса ишини якунлагач, бошқаришни ўзларини чақирган программаларга узатади, улар эса ўз навбатида

кейингиси ва жараён охирида бошқариш ОС (операцион система)га узатилади.
Резидент программалар эса бошқача ишлайди. Улар ишга тушгач дарров қандайдир амалларни бажариши мумкин, ёки ОС узилишлари билан турли хил тушунарсиз ишларни амалга ошириши мумкин. Уларни асосий ҳислати шуки, ўз ишини якунлагач ШЭҲМ хотирасида сақланиб қолади ва бирор бир шартнинг бажарилишига қараб қайта "тирилиши" яъни яна ишлай бошлаши мумкин. Резидент программаларга мисол қилиб қуйидаги программаларни кўрсатиш мумкин:
Side Kick системаси, ALFA ва BETA кирилл драйверлари, антивирус программалари, вирус программалари ва ҳакозо. Резидент программаларнинг барчасини бир хил нарса бирлаштиради, бу ҳам бўлса уларни ички(тезкор) хотирада (ОЗУ) жойлашиши ва махсус шартларда (ALFA, BETA клавишларини босишда, антивирус программаларга мурожаатда, қўйилган вақтни етиб келишида ва ҳакозо) қайта "тирилиши". Пассив ҳолатларда резидент программалар фақат ортиқча жой ушлаб туришади, бошқа оддий программалар ишига ҳалақит бермайди.
ШЭҲМ хотирасидан унумли фойдаланган ҳолда субпроцессларни ташкил қилиш ва резидент программалар билан ишлашнинг махсус процедураларидан фойдаланиш мумкин:

- SwapVectors процедраси ишга туширилаётган субпроцесслар учун системали ёки вақтинчалик узилишлар векторларини тиклайди;
- Exec (ExeFile, ComLine: String) процедураси ComLine сатрли параметр билан ExeFile (субпроцесс) бажарилувчи файлини ишга туширади;
- DosExitCode: Word функцияси субпроцесс якунланганлиги ҳақидаги кодни аниқлайди;
- Keep(ExitCode:Word) процедураси ШЭҲМ хотирасидан ўчмаган ҳолда программа ишини якунлайди (резидент кодини ташкил қилади).

3.13.5. Printer модули.

Паскал тилининг кутубхонасида санаб ўтилган модуллар қаторида Printer модули ҳам ўзининг муносиб жойини эгаллаган, чунки олинган натижаларни принтер орқали қоғозга чоп этиш муҳим ишдир. Бу модулнинг асосий вазифаси программа ва чоп этувчи қурилма (одатда принтер) ўртасида алоқа ўрнатишдир.
Бу кичик модул бор-йўғи битта ўзгарувчи ва бажарилувчи матннинг икки қаторидан ташкил топган. Бу модул Lst ўзгарувчиси ва LPT1 (биринчи параллел порт) системали қурилма мослигини ўрнатади.
Модулнинг тўлиқ матни қўйидагича:

```
Unit Printer;
```

```
Interface
Var
   Lst:Text;
Implementation
Begin
   Assign(Lst,'LPT1');
   Rewrite(Lst)
End.
```

Printer модулини улангандан сўнг Write(Lst,...) ва WriteLn(Lst,...) операторлари натижавий маълумотларни тўғридан-тўғри принтерга чиқаради.

Агар принтер бошқа қурилма орқали уланган бўлса, масалан иккинчи порт **LPT2** ёки кетма-кетли COM1 порти орқали, у ҳолда дастурчи ўзининг Printer2 номли модулини яратиб олиши мумкин. У ҳолда юқоридаги модул программаси матнидаги LPT1 ёзувини бошқа исмга (LPT2 ёки COM1) алмаштирилиб трансляция қилиб олинади. Натижаларни ёки махсус буйруқли кодларни принтерга Lst файли (ёки шунга ўхшаган) орқали чиқариш, чоп этишни бошқаришнинг ягона йўли эмас. БСВВ17Н узилиши орқали принтерга белгилар ёки уларнинг кетма-кетлигини узатиш мумкин. Бундан ташқари мазкур узилишдан фойдаланиш, чоп этиш қурилмасининг ҳолатини сўраш имкониятини яратади.

17Н узилиши принтерга хизмат кўрсатишнинг 3 хил функциясидан фойдаланишга рухсат беради:

- белгиларни принтерда чоп этиш (0-номерли функция);
- принтер портини ишга созлаш (1-номерли функция);
- принтер ҳолатини ўқиш (2-номерли функция).

3.13.6. Overlay модулининг процедура ва функциялари.

Жуда кўп ҳолатларда, программа компиляция қилингандан сўнг маълум бўладики программа учун ШЭҲМ нинг ички тезкор хотираси етишмай қолади, яъни программа талаб қилган хотира жойининг миқдори ШЭҲМ нинг хотирасидан катта бўлади. Бу ҳолатда экранда Not enough memory («хотира етишмаяпти») маъносидаги хато кўрсатилган маълумот ҳосил бўлади Бундай ҳолатларда дастурчига Overlay модули ёрдамга ошиқади

Программани оверлейли қуриш.Overlay модули программани алоҳида бўлакларга ажратишнинг кучли воситаси ҳисобланади. бундай бўлаклар сони ихтиёрий сонда бўлиши ва бу бўлакларга зарур бўладиган жой миқдори ШЭҲМ имкониятидан анча юқори бўлиши мумкин. Оверлейли тарзда тузилган программа битта . EXE кенгайтмали файлдан ва шундай номли лекин .OVR кенгайтмали яна битта файлдан ташкил топган бўлади. Бунда EXE – файл программанинг доимий қисмини, OVR – файли эса зарур пайтда хотирага юкланадиган кодларни сақлаб туради. Оверлейли тарзда

тузилган программада хотирада фақат хозирдагина зарур бўладиган оверлейли процедура ва функцияларгина сақланади, улар керакли пайтдагина ўзларининг жойларини бошқа процедура ва функцияларга бўшатиб беради. Бу ҳолда қўшимча хотира жойи талаб қилинмайди чунки программанинг оверлейли қисмлари хотиранинг фақат бир қисминигина навбат билан ишлатади холос. Хотиранинг мазкур қисми оверлейли буфер деб аталади. Оверлейларни хотирага юклаш ва хотирадан бўшатиш каби ички амалларнинг барчасини оверлей администрацияси автоматик тарзда бажаради. Дастурчидан эса фақат программанинг оверлейли парчаларини эълон қилиш ва унинг администрациясини ишга солиш талаб қилинади холос.

Оверлейли программаларни расмийлаштириш қоидалари.Турбо Паскалда оверлейлар модуллар даражасидагина яратилиши мумкин. Оверлейлар худди оддий модуллар каби эълон бўлими (INTERFACE) ва хисоблаш бўлими (IMPLEMENTATION), ҳамда уни фаоллаштирадиган қисмлардан ташкил топади. Шу билан бир қаторда қуйидаги шартларни бажарилиши ҳам талаб қилинади.

- хар бир оверлей модул учун унга олдиндан оверлейлилик ҳуқуқини берувчи {$FK} компиляция режими ўрнатилиши лозим;
- оверлейли қисм-программалардан бевосита ва билвосита фойдаланувчи барча процедура ва функциялар {$FK} компиляция қилиниши лозим.

Бу шартларни бажариш учун оверлейли модулнинг бош қисмига {$FK, $OK} директиваларини ўрнатиб, асосий программанинг биринчи қаторларига {$FK}, компиляция режимининг калити ёзиб қўйилади.

Бундан ташқари, асосий программа overlay модулига уланган бўлиши ва uses директивасидаги қайси модуллар оверлейли эканлиги кўрсатиб қўйилади.

Мисол сифатида қуйидаги программанинг бош қисмини кўрсатиб ўтамиз:

```
Program MultiCalc;
{$FK}
uses crt,dos,Overlay, MultiCalc, divCalc AddCalc, Subcalc;
{ $O MultiCalc}              {  MultiCalc модули –оверлей }
{ $O DivCalc}                { DivCalc модули –оверлей }
{ $O AddCalc}                { AddCalc модули –оверлей }
{ $O SubCalc}                { SubCalc модули –оверлей }
. . . . . .
```

Аҳамият берган бўлсангиз { $O <модул исми >} директиваси оверлейли модулларни кўрсатиш учун ёзилишга мажбурдир, бундан ташқари, uses директивасининг модуллари рўйхатида overlay модули оверлейли модуллар рўйхатининг бошида келиши шарт.
Оверлейли программалар хотирада компиляция қилинмайди, энди улар фақат дисклардагина ҳосил қилинади. Программа компиляциясидан сўнг (бу программани MULTICALC.PAS номи

билан атайлик) MULTICALC.EXE бажарувчи файли ва оверлейли деб эълон қилинган барча модуллардаги процедураларнинг кодларини сақловчи йўлдош файли ҳосил қилинади.

3.13.7.Оверлей ишини инилизация қилиш.

1. Оверлейлар администраторини юклаш
Оверлей администраторини юклаш программани ишлаши давомида фақатгина бир марта бажарилиши шарт. Бу иш қуйидаги overlay модулида эълон қилинган процедурага мурожаат орқали амалга оширилади:

OvrInit (Ovr File Name: string)

Бу процедура оверлей администраторини ишга юклайди ва ovr оверлейли файлини очади.
2. Оверлейни юклаш натижаларининг таҳлили учун Overlay модулидаги олдиндан аниқлаб қўйилган бутун типли OvrResult ўзгарувчиси мавжуд бўлиб, у мазкур модул процедура ва функцияларининг, шу жумладан OvrInit процедурасининг ишларини якунлаш кодини ўзида сақлайди. OvrResult ўзгарувчиси етти ҳил қиймат қабул қилиши мумкин ва бу қийматларнинг ҳар бирига мос равишда ўзгармаслар бириктириб қуйилган.

Бу маълумотлар 7- жадвалда келтирилган

7-жадвал

Ўзгармаслар	*Ўзгармаснинг маъноси*
OvrOk = 0 OvrEror = -1 OvrNotFound = -2 Ovr NoMemory = -3 OvrIOEror = -4 OvrNoEMSDriver = -5 OvrNoEMSMemory = -6	тўғри якунланиши Overlay ни бошқариш хатоси Ovr файли топилмади Буфер учун хотира етишмаяпти оверлейли файлни ўқишда бузилиш рўй берган EMS дравейри ўрнатилмаган EMS – хотира ҳажми етарли эмас

Энди санаб ўтилган хатоликлар ҳақида тўлиқроқ тўхталиб ўтайлик:

- OvrEror хатоси - оверлейли бўлмаган файлни юклашга харакат қилинганда рўй беради ;
- OvrNotFound хатоси - оверлейли файлни дискка нотўғри жойлаш оқибатида юзага келиши мумкин ;
- OvrNoMemory хатоси - ШЭҲМ нинг бўш хотираси етишмаслигидан рўй бериши мумкин ;
- OvrIOEror хатосини - келиб чиқишига кўпроқ ташқи факторлар сабабчи бўлади (файлни шикастланиши,MS –DOS ичидаги бузилишлар) ;

- OvrEMSDriver ва OvrNoEMSMemory хатоликлари OvrInit процедурасининг эмас, балки OvrInitEMS қўшимча процедурасининг ишидаги хатоликларидан келиб чиқади.

Оверлей буферини бошқариш.Оверлей буферининг ўлчовларини бошқариш ва уни тозалаш учун Overlay модулида маҳсус процедура ва функциялар киритилган:

OvrGetBuf функцияси.Вазифаси : ишчи буфернинг байтлардаги ўлчовларини аниқлайди ;

Аниқланиши : OvrGetBuf :Longint ;

бу ерда функциянинг қиймати баъзи ҳолларда 64 кб дан ошиши мумкин, шунинг учун, функция қийматини longint деб эълон қилинади.

OvrSetBuf процедураси.Вазифаси :буфер ўлчовини мослаштириб туради;

Аниқланиши : OvrSetBuf (Size: longint); бу процедура OvrInit ва OvrInitEMS процедуралари ёрдамида оверлейлар администрацияси юклангандан сўнг ишга тушиши лозим. Буфернинг талаб қилинаётган байтлардаги миқдори Size параметри орқали берилади.

OvrClearBuf – процедураси.Вазифаси : оверлей буферини мажбурий тарзда тозалаш учун ишлатилади.

Аниқланиши : OvrClearBuf ;

бу процедура буфердаги турган барча оверлейли программаларни чиқариб ташлаш вазифасини бажаради. Оверлейлар администраторининг ўзи, бу процедурани чақирмайди.

3.13.8. Graph модулининг процедура ва функциялари

Барча программалаш тиллари каби Паскал тили ҳам, программачига фақат матнли ахборотлар билангина эмас, балки график маълумотлар билан ҳам ишлаш имкониятини яратади. Graph модули турли хил дисплей адаптерлари (CGA,EGA,VGA,MCGA,Hercules,PC3270,AT&T6300 ва IBM8514)нинг график режимларини тўлиқ бошқариш имкониятини яратувчи саксондан ортиқ процедура ва функцияларнинг кутубхонасини ифода этади.

Графика режимида экран холати. Маълумки, экран тўртбурчак майдон бўлиб, жуда кўп нуқта(Pixel)лардан ташкил топган. График экран ишчи майдон ва бордюр (экраннинг четки қисми)дан иборат бўлади.

График режимда экрандаги барча нуқталарнинг рангларини ўзгартириб чиқиш мумкин (эслатиб ўтамиз, матнли режимда бу мумкин эмас). Турли рангга бўялган нуқталар ёрдамида чизиқлар, матнлар ва бошқа хар ҳил тасвирлар ҳосил қилинади. Экран турига қараб ранглар сони турли ҳил бўлиши мумкин, энг ками билан эса 2 ҳил ранг бўлади.

Компьютер экрани ёки матнли режим ёки график режим ҳолатида бўлади. Бир вақтнинг ўзида экраннинг бир қисми график режимда, бир қисми матнли режимда бўлиши мумкин эмас. Чунки экранда кўринаётган барча тасвирлар видео хотирадаги маълумотларнинг акси - тасвиридир.
Экраннинг график ёки матнли режимларига қараб, видеохотирадаги маълумотлар турлича бўлади.
Видеохотира, ёруғлик трубкасининг контроллери, кириш-чиқиш портлари ва ҳ.к. лар битта платада жойлашади ва дисплей адаптери деб аталади. Амалда бир неча хил дисплей адаптерлари мавжуд бўлиб, улар қуйидаги кўсаткичлари билан бир-биридан фарқ қилади :
- экраннинг тасвирни кўрсатиш сифати;
- экранда бир вақтда кўрсатиш мумкин бўлган ранглар сони.
Қуйида амалда кенг тарқалган видеоадаптерлар санаб ўтилган:

- CGA(Color Graphics Adapter) ;
- MCGA(Multi Color Graphics Array);
- EGA(Enhanced Graphic Adapter);
- VGA(Video Graphics Array).

Компьютерда қандай адаптернинг ўрнатилишига қарамай Турбо-Паскалнинг процедура ва функцияларидан тўлиқ фойдаланиш мумкин. Уларни ўзаро созлашуви автоматик тарзда кечади. Бу созлашларни график драйверлар деб аталувчи махсус программалар бажаради. Драйверлар =.BGI кенгайтмаси бор бўлган файлларда жойлашган.
Мисол учун, EGA ва VGA адаптерлари билан ишлаш учун зарур драйверлар EGAVGA.BGI файлида,CGA ва MCGA адаптерларига мос драйверлар эса CGA.BGI файлида жойлашган.
Турли хил драйверларни кўрсатиш учун *Graph* модулида қуйидаги ўзгармаслар аниқланган:

```
const
Detectқ0;{Драйверни автоматик тарзда аниқлаш}
CGAқ1;
MCGAқ2;
EGAқ3;
EGA64қ4;
EGAMONOқ5;
IBM8514қ6;
HERCMONOқ7;
ATT400қ8;
VGAқ9;
PC327қ10.
```

Graph модулида график режимларни кўрсатиш учун эса қуйидаги ўзгармаслар аниқланган (CGA,EGA ва VGA адаптерлари учун);

```
const
CGAC0қ0;        {320x200 та нуқта, 4 хил ранг}
CGAC1қ1;        {320x200 та нуқта, 4 хил ранг}
CGAC2қ2;        {320x200 та нуқта, 4 хил ранг}
```

```
CGAC3қ3;          {320x200 та нуқта, 4 хил ранг}
CGAHiқ4;          {640x200 та нуқта, 4 хил ранг}
EGALoқ0;          {640x200 та нуқта, 16 хил ранг, 4 та вароқ}
EGAHiқ1;          {640x350 та нуқта, 16 хил ранг, 2 та вароқ}
EGA64Loқ0; {640x200 та нуқта, 16 хил ранг,1 та вароқ}
EGA64Hiқ1;        {640x350 та нуқта, 4 хил ранг, 4 та вароқ}
VGALoқ0;          {640x200 та нуқта, 16 хил ранг, 4 та вароқ}
VGAMedқ1; {640x350 та нуқта, 16 хил ранг, 2 вароқ}
VGAHiқ2.          {640x480 та нуқта, 16 хил ранг, 1 вароқ}
```

Дисплейни график режимга ўтказиш.Дисплейнинг доимий режими, матнли режим ҳисобланади. Дисплейни график режимига ўтказиш учун Graph модулининг InitGraph процедурасидан фойдаланилади*:*

InitGraph(GD,GM,Path)
бу ерда GD-драйвер номери ;
GM-режим номери ;
Path-керакли драйвер жойлашган файл йўли.

GD ва *GM* номерларни қандай аниқлашни олдинги мавзуда кўриб чиқдик. Агар *Path* ўзгарувчиси бўш сатр қаторини ташкил қилса (Pathқ‘ ‘) драйверни ишчи каталогдан қидирилади.
GD ва *GM* ўзгарувчи параметрлар ҳисобланади. Агар *InitGraph* процедурасини ишга туширишда *GD*қ0 бўлса, зарур драйвер ва бу драйвер учун энг яхши режим автоматик тарзда аниқланади.
Graph модулида қулайлик учун *Detect* ўзгармаси (Detectқ0) киритилган. Мисолларда ундан қандай фойдаланганлигига аҳамият беринг.
InitGraph процедурасига симметрик процедура *CloseGraph* ҳисобланади. Бу процедура драйверни хотирадан чиқариб ташлайди ва олдинги видеорежимни тиклайди.
Қуйидаги программа дисплейни график режимга ўтказади ва шу заҳотиёқ уни асли ҳолига қайтаради:

```
uses  Graph;
var
      Gd,Gm:Integer;
begin
GD:қdetect;{Драйверни автоматик тарзда аниқлайди, чунки
Detectқ0}
      InitGraph(GD,GM,’d:ғtp’);  {График режимни ўрнатиш}
      ReadLn;              {Enter босқичини босилишини кутиш}
      CloseGraph;
end.
```

Баъзи ҳолларда, масалан .BGI кенгайтмали файл йўлини хато кўрсатилса ёки график режимни ўрнатиш учун тезкор хотира етишмаса InitGraph процедураси ўз ишини муваффақиятли якунлай олмайди. Бундай ҳолларда йўл қўйилган хатоликларни аниқлаш учун *GraphResult* функциясидан фойдаланиш мумкин. Бу функция график

режимни ишга туширишга ҳаракат қилганда қуйидаги қийматлардан бирини беради:

- 0 – хатолар йўқ;
- 2 – график плата ўрнатилмаган;
- 3 – драйвер файли топилмади;
- 4 – нотўғри драйвер ўрнатилган;
- 5 – график режимни ўрнатиш учун тезкор хотира етишмаган;

GraphResult функциясидан фойдаланишга доир қуйидаги программани кўриб чиқайлик:

```
uses Graph;
var
      GD,GM,ErrCode:integer;
begin
      GD:қDetect; {Драйверни  автоматик тарзда аниқлаш}
      InitGraph(GD,GM,'      '); {График режимни ишга тушириш}
      ErrCode:қGraphResult;
      if ErrCode<>0 then WriteLn(ErrCode,'рақамли  хато'){Хатога йўл
қўйилган}
      else CloseGraph;{Хато топилмаган ва график режимни иши
якунланди}
end.
```

Нуқта, чизиқ ва ранглар. Биз экранда график режимни қандай ўрнатишни кўриб чиқдик. Энди экранда турли хил нуқталар, чизиқлар ва шакллар чизишни ташкил қилайлик. *Graph* модулида 80 тага якин функция ва процедуралар мавжуд бўлиб, улардан фойдаланиб қуйидаги ишларни қилиш мумкин:

- нуқталар қуриш;
- кесмалар чизиш;
- эллипс ва айланалар чизиш;
- тўғри тўртбурчаклар ва кўпбурчаклар чизиш;
- ёпиқ шакларни турли рангларга бўяш, ҳамда бир неча ҳил стандарт ва керагича ностандарт усуллар билан соҳаларни штрихлаш;
- экранга турли шрифтдаги матнларни чиқариш;
- экран соҳаларини эслаб қолиш ва уларни суриш.

График режимда ишлаш худди математика фанидаги Декарт координаталар системасида нуқталар орқали турли ҳил шакллар ясашга ўхшаб кетади. Экрандаги ҳар бир нуқта ўзининг координаталарига эга. Экраннинг чапдан энг тепадаги нуқтасининг координатаси (0,0)га тенг. X координатаси чапдан ўнгга ўсиб борса, Y координатаси юқоридан пастга қараб ўсиб боради. Нуқта координатасини аниқловчи (x,y) жуфтликдаги биринчи қиймат OX ўқидан, иккинчи қиймат эса OY ўқидан аниқланади. Экраннинг ўнгдан энг пастидаги нуқтаси охирги нуқта ҳисобланади ва унинг координатаси график режимнинг турига боғлиқ. Масалан,VGAHi режимида экрандаги нуқталар сони 640x480га тенг. Ўнгдан энг пастки нуқта координатаси эса (639,479) га тенг бўлади.

Энди ***Graph*** модулининг энг кўп ишлатиладиган функция ва процедуралари билан танишиб чиқайлик.
1.***PutPixel***(x,y,color) процедураси -(x,y) координатали нуқтани ***Color*** параметри билан аниқланган рангга бўяб беради;
Мисол: **PutPixel**(100,120,Red) - экранда (100,120) координатали қизил нуқта пайдо бўлади.
2.***GetPixel***(x,y) функцияси - (x,y) координатали нуқтанинг рангини аниқлаб беради;
Мисол: Color:қGetPixel(100,120); (100,120) координатадаги нуқтага қўйилган ранг қийматини аниқлайди.
3. ***Line***(x1,y1,x2,y2) процедураси -(x1,y1) ва (x2,y2) координатали нуқталарни туташтирувчи кесма чизади;
4. ***Circle***(x,y,Radius) процедураси - маркази (x,y) нуқтада жойлашган радиуси Radius га тенг айлана чизади;
5. ***Rectangle***(x1,y1,x2,y2) процедураси-чап юқоридаги бурчаги (x1,y1) ва ўнгдан пастки бурчаги (x2,y2) координатали нуқталар орқали ўтказилаган тўғри тўртбурчак чизади;
6. ***SetColor***(Color) процедураси - чизиш рангини ўрнатади;
График режимда рангларни белгилаш учун худди матнли режимдаги каби қуйидаги ўзгармаслар ишлатилади:

```
const
Blackқ0;{Қора}
Blueқ1;{Кўк}
Greenқ2;{Яшил}
Cуанқ3;{Зангори ранг}
Redқ4;{Қизил}
Magentaқ5;{Пуштиранг}
Brownқ6;{Жигарранг}
LightGrayқ7;{Оч кулранг}
DarkGrayқ8;{Тўқ кулранг}
LightBlueқ9;{Оч ҳаворанг}
LightGreenқ10;{Оч кўк}
LightCyanқ11;{Оч зангориранг}
LightRedқ12;{Оч қизил}
LigthMagentaқ13;{Оч пуштиранг}
Yellowқ14;{Сариқ}
Whiteқ15;{Оқ}
```

Мисол:

```
uses Graph;
var
        GD,GM:integer;
        Rang,Radius:word;
begin
        GD:қDetect;          {Драйверни  автоматик тарзда аниқлаш}
        InitGraph(GD,GM,'      '); {График режимни ишга тушириш}
        for Rang:қ15 downto 0 do
        begin
            setcolor(Rang); {Чизиш рангини ўрнатиш}
            Radius:қRang=10;
             Circle(GetMaxX div 2, GetMaxY div 2,Radius);
```

```
    {Маркази экран марказида жойлашган айлана чизиш}
      end;
      Readln;
      CloseGraph;
end.
```

Турли хил фигуралар.Юқоридаги мавзуда кўриб чиқилган функция ва процедуралар ёрдамида фақат чизиқлар чизиш мумкин. Энди бошқа турли хил ранглар билан тўлдирилган фигуралар чизишни ташкил этишга ёрдам берувчи яна бир нечта процедура ва функциялар билан танишиб чиқамиз.

1.**SetFillStyle**(Style,Color) процедураси - Color ранги билан соҳаларни тўлдириш ва уларни кўрсатилган услубда тўлдириш (штриховка қилиш) учун ишлатилади;

Соҳани турли ранглар билан тўлдириш ўзгармаслари:

```
const
EmptyFillқ0;{соҳани экран фонининг рангига бўяйди}
SolidFillқ1;{соҳани белгиланган рангда узлуксиз тўлдириш}
LineFillқ2;{соҳани қалин горизонтал (-----) чизиқлар билан тўлдиради}
LtSlashFillқ3;{соҳани ингичка"FFF " белгилари билан тўлдиради}
SlashFillқ4;{соҳани қалин "FFF " белгилари билан тўлдиради}
BkSlashFillқ5;{соҳани қалин "ғғғ" белгилари билан тўлдиради}
LtBkSlashFillқ6;{ соҳани "ғғғ" қатлами билан тўлдиради}
HatChFillқ7;{соҳани тўр билан тўлдиради}
XHatChFillқ8;{ соҳани эгри тўр билан тўлдиради}
InterLeaveFillқ9;{ соҳани зич эгри штриховка билан тўлдиради}
WideDotFillқ10;{соҳани кам учровчи нуқталар билан тўлдиради}
CloseDotFillқ11;{соҳани зич нуқталар билан тўлдиради}
UserFillқ12;{соҳани дастурчи аниқлаган штриховка билан
                тўлдиради}
```

2.**Bar**(x1,y1,x2,y2) процедураси экрандаги ранг ва штриховка устига тўғри тўрт бурчак қуради;

3.**Bar3D**(x1,y1,x2,y2,Depth,Top) процедураси ҳам шундай ранг ва штриховка билан тўлдирилган параллелепипед чизади. **Depth** ўзгарувчиси параллелепипед баландлигини англатади. **Top** мантиқий ўзгарувчиси **true** қийматли бўлса параллелипипеднинг юқори асоси чизилади акс холда, чизилмай очиқ қолади;

4. **FillEllipse**(x,y,XRadius,YRadius) процедураси олдин ўрнатилган рангга тўлдирилган эллипс чизади. Эллипс ўқлари координата ўқларига паралел деб олинади. **XRadius** - эллипс эни, **YRadius** - эллипс баландлиги.

График режимдаги матнлар.График режимда матнларни ёзиш учун икки хил типдаги шрифтдан фойдаланиш мумкин: нуқталар матрицаси ва символни ташкил этувчи векторлар қатори орқали. Шрифтлар файллари .CHR кенгайтмасига эга бўлади ва шрифтни ишга созлаганда керакли файллар ишчи каталогда ёки .BGI график драйвери жойлашган каталогда бўлиши керак.

Шрифтни танлаш ва масштаб ўрнатиш **SetTextStyle** процедураси ёрдамида амалга оширилади:

SetTextStyle(Font, Direction, Size) - керакли шрифтни ўрнатади, матнни чиқариш йўналишини аниқлайди ва белгилар ўлчовини белгилаб беради. **Font**- шрифтни аниқловчи ўзгарувчи, **Direction**- матнни чоп этиш йўналишини кўрсатувчи ўзгарувчи (чапдан ўнгга ёки пастдан юкорига), **Size**-шрифт ўлчовини аниқловчи ўзгарувчи. Матрицали шрифтда ўлчов Sizeқ1, вектор шрифтида эса Sizeқ4 қийматларида эришилади.

Турли хил шрифтларни кўрсатиш ва матнларни чоп этиш йўналишларини танлаш учун қуйидаги ўзгармаслар аниқланган:

```
        const
{шрифтлар}
        DefaultFontқ0;{8x8 нуқтали стандарт матрицали шрифт}
        TriplexFontқ1;{векторли шрифт}
        SmallFontқ2;{векторли шрифт}
        SansSerifFontқ3;{векторли шрифт}
        GothicFontқ4;{векторли шрифт}
{матн йўналиши}
        HorizDirқ0;{чапдан ўнгга}
        VertDirқ1;{пастдан юқорига}
```

OutTextXY(x,y,TextString) процедураси - олдиндан аниқланган шрифтда, йўналишда ва белги ўлчовида *TextString* қаторини (x,y) нуқтадан бошлаб чоп этади.

SetTextJustify(Horiz,Vert) процедураси *OutTextXY* процедураси чоп этадиган матнни автоматик тарзда текислаб беради. *Horiz* - горизонтал, *Vert* - вертикал текислашлар.

Матнларни текислаш учун қуйидаги ўзгармаслар аниқланган:

```
    const
{горизонтал текислаш учун}
    LeftTextқ0;{чап томонга нисбатан текислаш}
    CenterTextқ1;{марказга нисбатан текислаш }
    RightTextқ2;{ўнг томонга нисбатан текислаш}
{вертикал текислаш учун}
    BottomTextқ0;{пастги томонга нисбатан текислаш}
    CenterTextқ1;{марказга нисбатан текислаш}
    TopTextқ2;{юқори томонга нисбатан текислаш}
```

Экран соҳалари. GetImage, PutImage процедуралари ва ImageSize функцияси ёрдамида тасвирларнинг тўғри тўртбурчакли соҳаларини ҳотирада эслаб қолиш ва уларни экранга чиқаришимиз мумкин.

1.*ImageSize*(x1,y1,x2,y2) функцияси – экраннинг тўғри тўртбурчак-ли соҳасини сақлаш учун зарур бўлган хотира ўлчовини (байтларда) беради.(x1,y1) тўғри тўртбурчакли кўринишнинг чапдан юқоридаги, (x2,y2) - эса пастдан ўнгдаги бурчак нуқталари учун координаталар.

2.*GetImage*(x1,y1,x2,y2,Area) процедураси хотиранинг *Area* соҳасида тўғри тўртбурчакли экран тасвирини сақлайди. (x1,y1) ва (x2,y2) лар юқоридаги маънода қайта ишлатилмоқда.

3.*PutImage*(x,y,Area,Mode) процедураси экраннинг кўрсатилган жойига тасвир кўринишини чоп этади. (x,y) – хотиранинг *Area*

соҳасидаги тасвир кўриниши нусхасини чоп этиладиган, экраннинг чапдан юқоридаги нуқтасининг координатаси. *Mode*-тасвирни экранга чиқариш режими.
Тасвирларни экранга чиқариш режимини аниқлаш учун фойдаланиладиган ўзгармаслар:

```
const
{PutImage процедураси учун ўзгармаслар}
NormalPutқ0;{ мавжуд тасвирни алмаштириш}
XorPutқ1;{XOR мантиқий амали}
OrPutқ2;{OR мантиқий амали}
AndPutқ3;{AND мантиқий кўпайтириш амали}
NotPutқ4;{NOT мантиқий рад этмоқ амали}
```

Хатолар таҳлили. График режимни ўрнатиш мавзусида йўл қўйилган хатолар диагностикаси учун *GraphResult* функциясидан фойдаланган эдик. Ҳозир шу функция берадиган хатолар коди билан тўлиқроқ танишиб чиқайлик.
Қуйида *GraphResult* функцияси берадиган кодларга мос ўзгармаслар рўйҳати келтирилган:

```
const
grOkқ0;{хатолар йўқ}
grNoInitGraphқ-1;{График режим ўрнатилмаган(InitGraph процедурасини ишга туширинг )}
grNotDetectedқ-2; {График плата ўрнатилмаган}
```

-6 - соҳаларни кўчиришда хотира чегарасидан чиқиш;
-7 - зарур соҳани бўяш пайтида хотира чегарасидан чиқиш;
-8 - шрифт файли топилмаган;
-9 - шрифт файлини ишга тушириш учун хотира етишмаяпти;
-10 - танланган драйвер учун нотўғри график режим.

Graph модулининг процедура ва функциялари.

1.*Arc* процедураси - айлана ёйини чизади.
Аниқланиши : Arc(x,y : integer; StAng, EndAng, Radius: Word);
x,y - айлана марказининг координатаси;
StAng, EndAng - мос равишда ёйнинг бошланғич ва охирги бурчаклари;
Radius-айлана радиуси.
2.*Bar* процедураси - рангга бўялган тўғри тўртбурчак чизади.
Аниқланиш: Bar(x1,y1,x2,y2:integer);
(x1,y1) ва (x2,y2) мос равишда тўғри тўртбурчакнинг четки нуқталари координаталари.
3.Bar3D процедураси рангга бўялган параллелипипед чизади.
Аниқланиши :Bar3D(x1,y1,x2,y2:integer;Depth:word;Top:boolean);
(x1,y1) ва (x2,y2) асосни ташкил этувчи тўғри тўртбурчак учларининг координаталари;
Depth -параллелипипед чуқурлиги;
Top- мантиқий ўзгарувчи.

4.*Circle* процедураси - айлана чизади;
Аниқланиши: Circle(x,y:integer;Radius:word);
(x,y) айлана марказининг координатаси;
Radius-айлана радиуси.
5. *CloseGraph* процедураси график режимини узади.
Аниқланиши :Closegraph;(параметрсиз процедура)
6.*DrawPoly* процедураси - кўп бурчак чизади.
Аниқланиши :DrawPoly(NumPoints:word; var PolyPoints);
NumPoints - кўпбурчак томонлари сони;
PolyPoints - кўпбурчак учларининг координаталаридан тузилган массив.
7.*Ellipse* процедураси - эллипс ёйини чизади.
Аниқланиши:
Ellipse(x,y:integer;StAng,EndAng:word;XRadius,YRadius:word);
(x,y) – эллипс марказнинг координатаси;
StAng ва EndAng - ёйнинг бошланғич ва охирги бурчаклари;
Xradiues ва Yradius мос равишда эллипс баландлиги ва эни.
8.*FillPoly* процедураси - рангли кўпбурчак чизади.
Аниқланиши: FillPoly(NumPoints:word; var PolyPoints);
NumPoints - кўпбурчакнинг учлари сони;
PolyPoints - кўпбурчак учлари координаталаридан тузилган массив.
9.*GetArcCoords* процедураси - охирги марта ишлатилган *Arc* процедурасининг координаталарини аниқлайди.
Аниқланиши: GetArcCoords(var ArcCoords:ArcCoords Type);
10.*GetColor* функцияси - экран рангини аниқлайди.
Аниқланиши: GetColor:word;
11.*GetGraphMode* функцияси - график экранни қайтаради.
Аниқланиши: GetGraphMode:integer;
12.*GetImage* процедураси - экраннинг берилган соҳасини Area да сақлайди.
Аниқланиши: GetImage(x1,y1,x2,y2:integer;var Area);
13.*GetMaxColor* функцияси - рангнинг энг катта қийматини ҳисоблайди.
Аниқланиши: GetMaxColor:word;
14.*GetPixel* функцияси - берилган нуқта рангини аниқлайди.
Аниқланиши: GetPixel(x,y:integer):word;
15.*GraphErrorMsg* функцияси - берилган код бўйича хато ҳақида сатр маълумот беради.
Аниқланиши: GraphErrorMsg(Code:integer):string;
16.*LineTo* процедураси - олдинги аниқланган нуқтадан берилган нуқтагача кесма чизади.
Аниқланиши: LineTo(x,y:integer);
17. *PieSlice* процедураси сектор чизади.
Аниқланиши: PieSlice(x,y:integer;StAng,EndAng,Raduis:word);

3.14.Файлли типлар ва динамик объектлар

3.14.1.Файлли типлар

Файлли типдаги ўзгарувчиларни дискдан маълумот ўқиб олувчи ёки дискка маълумот ёзиб қўювчи программаларда ишлатиш мумкин. Файлли типдаги ўзгарувчиларни эълон қилишда ***file*** ва ***text*** хизматчи сўзлари ишлатилади:

```
var     mfile 1, mfile 2: file;
        afile: file;
        Prima: text;
```

text хизматчи сўзи файлнинг матнли эканлигини англатади. Матнли файллар махсус белгилар билан ажратилган, узунлиги номаълум бўлган қаторлардан ташкил топади.

Айрим пайтларда файлларни бир хил типли ҳадлар кетма-кетлиги кўринишида қараш қулайрок бўлади. Бу кетма-кетлик қаторлар, бутун сонлар ёки ёзувлардан ташкил топиши ҳам мумкин:

```
var     A1: file  of  byte;
{A1 файли байтлар кетма - кетлигидан ташкил топган}
        A2: file  of  integer;
        {A2 файли бутун сонлар кетма-кетлигидан ташкил топган}
        A3: file  of  string;
        {A3 файли каторлар кетма-кетлигидан ташкил топган}
        A4: file of  string*20+;
{A4 файли 20та белгили қаторларнинг кетма-кетлигидан ташкил топган}
        A5: text;
{A5 файли матнли файл ҳисобланади}
```

Агар файлнинг ҳадлари учун тип аниқланган бўлса, бундай файлларни типлаштирилган, акс ҳолда типлаштирилмаган деб аталади:

```
var     A: file ; { типлаштирилмаган файл}
        B: file of char; { типлаштирилган файл}
```

Файллар билан ишлайдиган қуйидаги программани кўриб чиқайлик.

```
Var
        mydata: file of integer;
        i, j, sum: integer;
begin
        assign (mydata, 'd:ғtpғmyfile.dat');
        {mydata файл ўзгарувчиси билан файлнинг  исмини myfile.dat ва унинг
аниқ йўли аниқланмоқда}
                rewrite (mydata);   {файл ёзув учун очиқ}
writeln ('Салом номаълум ўртоқ...');
writeln ('Биринчи сонни киритинг');
readln (i);
writeln ('Киритилган сонни дискдаги myfile.dat
файлига ёзилмокда');
        write (mydata, i); {бу оператор ёрдамида дискдаги myfile.dat  файлига i
сонини ёзилади}
        writeln ('Иккинчи сонни киритинг');
        readln (j);
```

```
writeln (‘Киритилган иккинчи сонни дискдаги myfile.dat
файлига ёзилмоқда’);
        write (mydata, j);  {Дискка ёзиш бажарилмоқда}
        sum :қ i Қ j;
        writeln (‘Йиғинди қ’, sum);
writeln (‘Йиғинди дискдаги myfile.dat файлига ёзилмоқда’);
        write (mydata, sum);        {Дискка ёзиш бажарилмоқда}
        close (mydata);  {mydata файли ёпилди}
        writeln (‘Хайр номаълум ўртоқ...’);
end.
```

Эътиборингизга ҳавола этилган программада ***Assign, Rewrite, Write*** ва ***Close*** процедураларидан фойдаланилди. Энди шу процедураларнинг ва кейинги программада ишлатилувчи ***Reset*** ва ***Read*** процедураларнинг вазифалари ва қандай аниқланганлиги ҳақида қисқача маълумот бериб ўтайлик:

Assign процедураси.

Вазифаси: Файлли ўзгарувчига ташқи файл исмини ўзлаштиради.

Аниқланиши: Assign (f; name: string);

бу ерда f – ихтиёрий типли файлли ўзгарувчи;
name - қаторли типдаги ифода ёки қатор, файл исми (агар файлнинг тўлиқ йўли кўрсатилмаган бўлса файл ишланаётган каталогда жойлашган бўлади).

Close процедураси.

Вазифаси: очиқ файлни ёпади.

Аниқланиши: Close (f);

бу ерда f - олдиндан очилган файлга мос келувчи файлли ўзгарувчи.

Read процедураси.

Вазифаси: файл ҳадини ўзгарувчига ўқийди.

Аниқланиши: Read (f, v);

бу ерда f - файлнинг ихтиёрий типига мос файлли ўзгарувчи (фақат матнли типли эмас);
v - файл ҳадининг типи билан бир хил типли ўзгарувчи.

Reset процедураси.

Вазифаси: мавжуд файлни очади.

Аниқланиши: Reset (f: file);

бу ерда f – файлнинг ихтиёрий типига мос файлли ўзгарувчи ва бу ўзгарувчи файл билан Assign процедураси орқали боғланган бўлиши керак. Reset процедураси мазкур файлни очади.

Rewrite процедураси.

Вазифаси: янги файлни яратади ва очади.

Аниқланиши: Rewrite (f: file);

бу ерда f – ихтиёрий файлли типдаги файлли

ўзгарувчи. Rewrite процедурасини ишлатишдан олдин f ўзгарувчи Assign процедураси ёрдамида дискдаги файл билан боғланиши керак. Rewrite процедураси янги файл ташкил қилади.

Write процедураси.

Вазифаси: файл ҳадига ўзгарувчини ёзиб қўяди.

Аниқланиши: Write (f, v);

бу ерда f – файлли ўзгарувчи;

v - f файлининг ҳади билан бир хил типли ўзгарувчи.

Олдинги тузган программамиз «d:» дискдаги tp каталогида myfile.dat файлини ташкил қилди. Энди шу файлдан қандай қилиб маълумотларни ўқишни кўриб чиқайлик.

```
Var
        mydata: file of integer;
        i, j, sum: integer;
begin
        assign (mydata, 'd:ғtpғmyfile.dat');
        reset (mydata);                         {файл ўқиш учун очилмоқда}
        writeln ('Салом номаълум ўртоқ...');
        read (mydata, i);
        writeln ('myfile.dat файлидан биринчи сон ўқилди');
        read (mydata, j);
        writeln ('дискдаги myfile.dat файлидан
иккинчи сон ўқилди');
        read (mydata, sum);
        writeln ('myfile.dat файлидан учинчи сон ўқилди');
        close (mydata);                 {mydata файли ёпилади}
        writeln ('Хайр номаълум ўртоқ...');
end.
```

Text стандарт файлли тип матнли файлларни аниқлайди. Матнли файллар ўзаро янги қаторга ўтиш белгилари билан ажратилган қаторлардан ташкил топади.

Матнли файллар билан ишлаш учун махсус киритиш (*Readln*) чоп этиш (*Writeln*) процедуралари кўзда тутилган. Бу процедуралар узунлиги номаълум каторларни файллардан ўқиш ва файлларга ёзиш учун ишлатилади.

Энди матнли файллар билан ишлашга доир қуйидаги программа билан танишиб чиқайлик:

```
var
        mytext: text;
        s: string;
begin
        assign (mytext, 'd:ғtpғmytext.txt');
                {mytext файлли ўзгарувчи оркали файл
                        исми ва йўли аниқланмоқда}
        rewrite (mytext);
                {файл ёзиш учун очиқ}
        writeln ('Сизнинг исмингиз?');
```

```
        readln (s);
        writeln (‘Исмингизни дискдаги mytext.txt файлига ёзилмоқда’);
        writeln (mytext, s);
                {s - қатори mytext.txt файлига ёзилмоқда}
        close ( mytext);
                {mytext файли ёпилди}
end.
```

3.14.2. Хотиранинг динамик соҳаси

Программанинг (ташқи) ўзгарувчилари компьютер хотирасининг ва ўзгармаслар маълумотлар сегментида жойлашган бўлади. Сегмент узунлиги эса 64 Кб ни ташкил қилади. Демак, шундай ҳоллар бўлиши мумкинки, программанинг барча ўзгарувчиларини маълумотлар сегментига жойлаб бўлмайди, яъни маълумотлар сегментга сиғмайди. Масалан, куйидаги программани компиляция қилишга уриниш натижасиз тугайди, яъни экранда "*Error 49: Data Segment too large*" (берилган маълумотлар сегменти жуда катта) йўл қўйилган хато ҳақида маълумот чиқади:

```
var
        A: array *1..40000+ of   byte;
        B: array *1..25534+  of  byte;
begin
end.
```

Шунинг учун, катта хажмли маълумотлар билан ишлашга тўғри келганда бу маълумотларни хотиранинг динамик соҳасига ёзиш мақсадга мувофиқдир. Хотиранинг динамик соҳаси (тўплама деб ҳам аташади) программа эгаллаган деярли барча хотира ҳисобланади. Фақат бу соҳага маълумотлар кодлари, стек ва маълумотлар сегменти хотиралари кирмайди. Одатда программа эгаллаган хотира кўриниши қуйидагича бўлади:

```
Хотиранинг катта
адреслари

                                  ←----------
Бўш хотира                          HeapEnd
                                  ←----------
Тўплама (хотиранинг                 HeapPtr
динамик соҳаси)

                                  ←----------
Стек соҳаси                         HeapOrg
Киритилган маълумотлар
сегменти
глобал ўзгарувчилар
ўзгармаслар

                                  ←----------
Программа коди                      Dseg:0000
                                  ←----------
```

PrefixSeg

Хотиранинг кичик адреслари

Тўпламанинг боши ва охирини ***System*** модулининг ***Heaporg*** ва ***HeapEnd*** махсус ўзгарувчилари кўрсатиб туради. Зарур пайтдаги тўплама соҳасининг тугаш жойини ***HeapPtr*** ўзгарувчиси аниқлаб беради. Тўпламанинг ўлчови киритилган маълумотлар сегментининг ўлчовидан бир неча марта катта бўлиб, тахминан 500 кб (агар компьютернинг умумий хотираси 640 Кб бўлса) жой эгаллаши мумкин.
Маълумотлар сегментида жойлашган маълумотлар статик маълумотлар деб аталади. Бунга асосий сабаб, улар учун хотира программани компиляция қилиш давомида ажратиб қўйилади. Программани ишлаши давомида эса бу хотира ўзгармай колади. Тўпламадаги хотира эса программани ишлаши давомида тўлдириб борилади ва зарур бўлган пайтда бу хотира бўшатиб қўйилиши мумкин. Шунинг учун, тўпламада жойлашган ўзгарувчиларни динамик ўзгарувчилар деб аталади. Тўпламадан хотира ажратишни ***GetMem*** ва ***New*** процедуралари орқали амалга оширилади, уни бўшатиб қўйишни эса ***FreeMem*** ва ***Dispose*** процедуралари бажаради.

3.14.3. Кўрсаткичлар ҳақида бошланғич маълумотлар

Кўрсаткич шундай ўзгарувчики, у маълумотнинг қийматини эмас, балки уларнинг хотирадаги жойлашган адресини ўзида сақлайди. Кўрсаткичларнинг икки хил типи мавжуд: типланган ва типланмаган. Типланмаган кўрсаткичларнинг эълон қилиш учун ***pointer*** хизматчи сўзидан фойдаланилади, типланган кўрсаткичларни эълон қилиш учун эса кўрсатилган тип олдидан "^" белгиси қўйилади.
Мисол:

```
var
        p: pointer; {типланмаган кўрсаткич}
        pint: ^ integer ; {pint - типланган, бутун типли кўрсаткич}
        x, y: ^ byte; {x, y - байт типидаги кўрсаткич}
        pst: ^ string ; {pst - қатор типидаги кўрсаткич}
```

Адреси кўрсаткичда жойлашган зарур маълумотга кўрсаткич қилиш учун кўрсаткич исмидан кейин "^" белгисини ёзиш керак:

```
GetImage (0, 0, 100, 100, p^); {график экран соҳасининг
                                маълумотлари p да кўрсатилган
                                адресдаги хотира соҳасига ёзилади}
pint^ :қ2;      {pint  адреси бўйича 2 бутун сони ёзилади}
x^ :қ # 12; y^ :қ $2; {x^ байтига # 12 қийматини ўзлаштирилади,  y^ байтига эса
$ 2}
        pst^ :қ ' Ахмад'; {pst  қаторига 'Аҳмад' сўзини ўзлаштирилади}
```

Кўрсаткични ишлатиш учун уни фақат эълон қилибгина қолмай, унинг қандай жойни кўрсатаётганлигига ҳам катта аҳамият бериш зарур. Яъни, кўрсаткич оператив хотиранинг керакли маълумотлар жойлашган адресини аниқ кўрсатиши керак. Кўрсаткич адресини хато кўрсатиш кутилмаган натижалар олишга ёки бутунлай программа ишини чиппакка чиқаришга олиб келиши мумкин. Шунинг учун, Паскал тилида махсус процедуралар мавжудки, улар ёрдамида хотира адреси аниқ ҳисобланади.

1. *GetMem* процедураси (типланмаган курсаткичлар учун) - тезкор хотирадан зарур миқдорда жой ажратади ва унинг адресини кўрсаткичга жойлаштиради.

Аниқланиши: GetMem (var p: Pointer; size : word);

p - кўрсаткич, size - ажратилган хотиранинг байт ўлчовидаги миқдори.

Мисол:

```
var
        p: pointer;
begin  GetMem (p, 1000); {1000 байт узунликдаги хотира
соҳаси ажратилди}
{p^   соҳадан фойдаланиш блоки}
        FreeMem (p, 1000); {p адресдаги 1000 байт узунликли
хотира соҳаси бўшатиб қўйилди}
end.
```

2. *New* процедураси (типланган кўрсаткичлар учун) - хотирадан зарур жойни банд қилиб, унинг адресини берилган кўрсаткичга жўнатади.

Аниқланиши: New (var p: pointer);

p - берилган кўрсаткич.

New ва *GetMem* процедуралари нисбатан жуда катта ҳисобланмиш хотиранинг динамик соҳасидан зарур миқдордаги хотира сўраб олади. Шу хотирани банд қилиб, унинг адресини мос кўрсаткичга жўнатади. Шундай ҳоллар ҳам бўладики, хотиранинг динамик соҳасида бўш жой қолмайди, у ҳолда шу хотирага юқоридаги процедуралар билан қилинган кўрсаткич қуйидаги маълумотли хатони юзага келтиради:

        Runtime Error 203 ...

Шунинг учун, хотиранинг динамик соҳасини вақти-вақти билан бўшатиб туриш керак. Бу ишларни *Dispose* ва *FreeMem* процедуралари бажаради.

1. *Dispose* процедураси - типланган кўрсаткичдаги адресда жойлашган хотирани бўшатади.

Аниқланиши: Dispose (var p: pointer);

2. *FreeMem* процедураси - типланмаган кўрсаткичда кўрсатилган адресдаги хотирани бўшатади.

Аниқланиши: FreeMem (var p: pointer; size: word);

Кўрсаткични вақтинча ишламай туришини билдириш учун *Nil* хизматчи сўзидан фойдаланиш мумкин.

Бу процедуралар билан бир қаторда қуйидаги функциялар билан ҳам танишиб чиқайлик:
1. *MemAvail* функцияси - хотиранинг динамик соҳасидаги бўш соҳаларнинг умумий хажмини аниқлайди.
Аниқланиши: MemAvail : LongInt;
2. *MaxAvail* функцияси - хотиранинг динамик соҳасидаги энг катта соҳа ўлчовини аниқлайди.
Аниқланиши: MaxAvail: LongInt;
Мисол сифатида GetMem ва FreeMem процедураларидан фойдаланилган қуйидаги программани кўриб чиқайлик:

```
Var
        P: Pointer;
Begin
GetMem(P,1024);  {Хотирадан 1024 байт жой ажратилди, P шу соҳани
кўрсатмоқда}
......
        FreeMem(P,1024); {P соҳадаги  1024 байт жой бўшатилди}
End.
```

New ва Dispose процедуралари эса қуйидагича ишлатилади:

```
Var
                PS:^String;
        Begin
New(PS);  {Хотирадан 256 байт жой ажратилди }
PS^:қ'Динамик хотира';
Writeln(PS^);
Dispose(PS); {PS соҳасиги  тегишли бўлган жой бўшатилди}
End.
```

4- БОБ. Си программалаш тили.

4.1. Си тили элементлари.

4.1.1. Тилнинг асосий тушунчалари.

Си программалаш тили Юникс операцион системаси билан боғланган бўлиб, шу системада ишлатиладиган бир қанча программалар Си тилида яратилган. Си программалаш тили машинанинг барча имкониятларидан тўлалигича фойдаланишга имкон беради.

Си мўлжалланишига кўра умумий бўлиб, у юқори даражали программалаш тиллари билан Ассеблер орасида оралиқ вазиятни эгаллайди.
Программани бошқарувчи операторлар- структурали программалаш талабларига жавоб беради. Унда киритиш-чиқариш, хотирани динамик тақсимлаш, мультипрограммалаш, параллел ҳисоблаш

воситалари йўқ. Бу воситалар ташқи функциялар орқали амалга оширилади.
Си тилида программа тузиш -ҳисоблаш машиналари ишини ва унинг операцион системасини тушуниш имкониятини яратади, шунинг учун ҳам у касбий программистларга мўлжалланган.
Лекин,Си программалаш тили бирон-бир система ёки машина билан қаттиқ боғланиб қолмаган.
Си тилида программалар маълумотлар устида натижа олиш мақсадида қилинадиган ҳаракатни билдиради. Программада ҳаракатлар операторлар орқали берилади, маьлумотлар эса обьектларни аниқлаш ва тавсифлаш орқали амалга оширилади.
Программада фойдаланиладиган ҳар бир объект тавсифланиши зарур. Тавсифлаш объект билан бир неча характеристикаларни боғлайди. Бу характеристикалар: тип, белгилаш, хотира синфи, ҳаракат соҳаси, ва бошланғич қийматлар кабилардир.
Тилда асосий типлар: белги (литер) билан биргаликда бутун ва сузувчи вергулли сонлар ишлатилади. Бундан ташқари кўрсаткичлар, массивлар, операндалардан чиқарилувчи маълумотларнинг тўла иерархиясини ҳосил қилиш мумкин.
Ифода-операндалар ва кўрсаткичлардан ташкил топади. Қиймат узатиш функциясини чақаришни қўшиб ҳисоблаганда, ҳар бир ифода кўрсатма бўлиши мумкин.
Кўрсаткич - бу берилган типдаги объектни далиллашдан иборат.
Массив - бу бир типдаги объектлар кетма-кетлиги бўлиб, уларга яқинлашиш индекслари орқали амалга оширилади, лекин қўшимча аниқлаш рекурсив қоидаси ёрдамида исталган ўлчамдаги массивларни аниқлаш мумкин.
Тузилмалар-турли типдаги объектлар кетма-кетлиги бўлиб, уларга яқинлашиш-исмлар бўйича амалга оширилади. Икки хил тузилмалар мавжуд бўлиб, оддий тузилмаларда (struct) элементлар хотирада тавсифланган тартибда кетма-кет жойлашади. Исталган пайтда уларга яқинлашиш мумкин.
Вариантли тузилмаларда (union) ҳамма элементлар хотирада тузилма бошланишида жойлаштирилади. Бу ҳолат энг кейинги қиймат берилган элементгагина яқинлашиш имконини беради холос.
Ўтказиш - бу бутун типдаги объектлар кетма-кетлиги бўлиб, уларга яқинлашиш исми бўйича амалга оширилади.
Функция - бу мураккаб объектдан иборат, уни ҳисоблаш натижасида берилган типдаги қиймат ҳосил қилинади.
Кўрсаткичлар боғлиқмас адресли-машина арифметикасини белгилайди. Си да кўрсатма тузувчи ({ }), шарт бўйича тармоқланиш (if), кўпдан бирон-бир муқобилларини (switch) танлаш, юқорига қараб такрорлаш (for, while) қуйига қараб такрорлаш (do), шунингдек такрорлашни тўхтатиш (break) каби бошқарувчи тузилмалар мавжуд.

Си тилида ҳар бир функцияга рекурсив (якка тартибда) ёндашилади. Лекин бир функцияниинг тасвири иккинчи функциянинг орасида ётиши мумкин эмас.
Тилнинг бошқа тилларга ўхшаш камчилиги ҳам мавжуд бўлиб, тил тузилмалари синтаксиси ноқулай.
Си программалаш тили биринчи марта Денис Ритчи ("BELL LABARATORIES" АҚШ) тамонидан 1972 йилда лойихалаштирилган. Бу тилда сколяр қийматлар билан ҳам иш олиб бориб, улар ўртасида операциялар бажариш мумкин. Такрибий қийматлар эса элементларга яқинлашишни тартиблаш учун ишлатилади.
Тилнинг лексикаси ва синтаксисида Бэкус-Нуар формасига яқин қоидалар қабул қилинган ва бу Метаформалар "қ" ифода билан ажратилади. Бу белгининг чап томонида мета ўзгарувчи ва ўнг томонида эса унинг қиймати ётади.
Си тили алфавитига қуйидагилар киради:

- лотин алифбосининг босма ва ёзма ҳарфлари (A,B,...,Z,a,b,...,z)
- рақамлар: 0,1,2,3,4,5,6,7,8,9
- махсус белгилар: ", [], {}, |, (), +, –, /, F, ; ' . ! ? < қ > _ ! & = # ~ ^
- аксланмайдиган белгилар (бўш жой, табуляция, янги қаторга ўтиш)

Изоҳларда эса бошқа белгилар иштирок этиши ҳам мумкин.
F= =F белгиси ёрдамида изоҳлар келтирилади.
*+, (), {}, ;, :, …, =, қ ажратувчи белгилар дейилади.
[ ] - катта қавс бир ёки кўп ўлчамли массивларни чегаралаш учун ишлатилади.
Масалан:
a[5] - а бир ўлчамли 5 та элементдан иборат бўлган массив;
e [3] [2] - е икки ўлчамли массив (матрица);
( ) - кичик қавслар қуйидаги ҳолларда ишлатилади:
1) агар операторида ифода шартини ажратади.
If (x<0) хқ-x;
2) функцияни аниқлашда ва уни тавсифлашда;
float F(float x, int k) /= функцияни аниқлаш =/ {функция - танлаш};
float F(float x, int); /= функцияни тавсифлаш =/;
3) функцияни кўрсатувчиларини аниқлашда int (=pfink)(); /= pfink функция кўпайтмасини аниқлаш учун =/;
4) операцияларни бажариш кетма-кетлигини аниқлашда уқ(aҚb)Fc;
5) такрорлашнинг элементи сифатида for(iқ0, jқ1,iқ<j;i Ққr ;jҚ2; jҚҚ) цикл танаси; do цикл танаси while (k>0);
6) макро аниқликларда
#define R(x,y) sqrt((x)=(x)Қ(y)=(y))

Ташкилий оператор ёки блокни бошланишини ёки тугалланган қисмини билдириш учун фигурали қавслар ишлатилади. Уларни массивларни инициализациялашда ва улар таркибини аниқлашда ҳам ишлатиб, нуқта вергул билан тугаланади.

Int day *+қ{1,2,3,4,5,6,7,8,9,10,11,12};

,-вергул белгиси ажратувчи ёки операция сифатида ишлатилиши мумкин.

;- Си тилида ҳар бир аниқлаш нуқта вергул билан якунланади.

: -меткани аниқлаш учун ишлатилади.

...- кўп нуқта функцияда ўзгарувчини параметрлари сонини аниқлашда ва тасвирлашда ишлатилади.

Қиймат узатишни белгилаш 'қ' тенглик белгиси ёрдамида амалга оширилади.

=- операция сифатида кўпайтириш,

#- белгиси эса процесор олди буйруқларини аниқлаш учун хизмат қилади.

Операциялар сифатида: Қ, -, =, F,<, >, !, &, ˜, →, ←, ˆ белгилари ишлатилади.

Улар билан кейинги параграфларда танишиб ўтамиз.

4.1.2. Идентификаторлар.

Идентификатор ("_"), ҳарфлардан ё белгилардан ёки остки чизиқ билан бошланган рақамлар кетма-кетлиги идентификатор ҳисобланади.

Масалан:

KOM_ 15, sizl 98, - MAX,TIME, time.

Босма ва ёзма идентификатор бир-биридан фарқланади. Юқоридаги охирги иккита идентификатор бир-биридан фарқли. Идентификаторлар турли узунликга эга бўлиши мумкин, лекин компютер 31 тагача бўлган белгини ҳисобга олади холос. Айрим компютерларда бу чеклов янада қаттиқроқ қўйилиб, фақат дастлабки ихтиёрий идентификаторнинг 8 та белгисигача ҳисобга олинади.

Бу ҳолда NUMBER_OF_ROOM ва NUMBER-OF_TECT идентификаторлари программа бажарилишида бир хил бўлади.

Хизматчи (калит) сўзлар. Программист томонидан эркин танлаб олиб ишлатиш сифатида қўлланилмайдиган идентификаторлар хизматчи сўзлар деб аталади. Хизматчи сўзлар маълумотлар типини, хотира синфини, тип хусусиятини,ва операторларини аниқлаш учун хизмат қилади.Тил стандартида қуйидаги хизматчи сўзлар мавжуд:

auto, break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, long, register, return, short, signed, siseof, static, struct, with, typedef, union, unsigned, void, volatile, while.

Хизматчи сўзлар маъно жиҳатдан қуйидаги гуруҳларга ажратилади.

Маълумотлар турларини синфларга ажратиш учун:

char-белгили

double-сузувчи вергули иккиланган аниқликдаги ҳақиқий сонлар

enum-саналма тип

float- сузувчи вергули ҳақиқий

int-бутун

long- узун бутун

short- қисқа бутун

struct- тузилмали тип

signed- ишорали

union- бирлаштирувчи тип

unsigned-ишорасиз

void- қиймат иштирокисиз

typedef-типни аниқлаш синонимини киритиш.

Тип хусусиятларини аниқлаш:

const-ягона қийматга эга бўлган фақат ўқиш учун аниқланадиган синф хусусияти.

volatile- програмистнинг аниқ кўрсатмаларисиз қиймат ўзгартирадиган объект синфи.

Хотира синфини белгилаш учун:

auto- автоматик

extern- ташқи

register- регистрли

static- статитик

идентифигаторлари ишлатилади.

Операторларини қуриш учун қуйидаги хизматчи сўзлар ишлатилади.

Break-циклдан ёки ўтказувчидан чиқиш

continue- циклни давом эттириш

do- бажармоқ(шарт остида циклни бажариш оператори бош қисми)

for- учун(параметрли такрорлаш операторининг бош қисми)

goto- ўтиш(шартсиз ўтиш)

if- агар- шартли операторнинг белгиланиши

return- қайтиш (функциядан)

switch- ўтказувчи

wile- ҳозирча(do цикли якунловчиси)

Хизматчи сўзларга кирадиган қуйидаги идентификатор мавжуд бўлиб:

default-switch операторида керакли вариант топилмаса ҳаракатни давомини аниқлаш

case- switch оператори вариантларни танлаш

else- ёки-if шартида муқобил тармоқни танлаш

sizeof- операнда ўлчамини (байтларда) аниқлаш операцияси.

Хизматчи сўзлар ёзма ҳарфлар билан белгиланади.

Изоҳлар F= =F белгиси орасида қўйилиб, ундан программа исталган қисмида фойдаланиш мумкин.

4.1.3. Операция белгилари

Қуйидаги 8-жадвалда операциялар ранглари билан группаларга ажратилган.

8-жадвал

№	Операция	Ассоциативлиги
1.	() * + → ←&	→
2.	! ~ Қ - ҚҚ -- & = (min) sizeof	←
3.	= F % (мультипликатив) бинар	→
4.	Қ - (аддитив бинар)	→
5.	>> << (разрядли суриш)	→
6.	>< қ > қ > (муносабат)	→
7.	қ қ! (муносабат)	→
8.	& (разрядли конъюнкция "ва")	→
9.	^ (разраядли "ёки")	→
10.	| (дизъюнкция "ёки")	→
11.	&& (конъюнкция "ва")	→
12.	|| (дизъюнкция "ёки")	→
13.	? : (шрифтли операция)	←
14.	қ = қ F қ % қ Қ қ & қ ^ қ F қ << қ >> қ	←
15.	, (вергул операцияси)	→

Муносабат ва мантиқий ифодалар. Муносабат операциялари қуйидагича аниқланади:

қ қ тенг; ! қ тенг эмас;
< кичик; < қ кичик ёки тенг;
> катта; > қ катта ёки тенг;

1 - ранг операцияларни юқорида кўриб ўтдик.
2 - ранг операциялари унар ёки бир жойли операциялар деб аталади.
! - эмас – мантиқий инкорни билдиради;
масалан: !2қ0; !(-5)қ0; !0қ1;
~- битли инкорни ифодалайди;
- - унар минус, арифметик операцияни ишорасини унар минусга ўзгартиради;
Қ - унар плюс, унар минусга симметрик равишда киритилган (инкремент ёки автоматик ортириш);
ҚҚ - бир бирликка ортиши;
икки хил шакли мавжуд:
префиксли операция - операнда қиймати у фойдалангунча биттага ортиши;

постфиксли операция - операнда қиймати
фойдаланилгандан кейин биттага ортиши;

-- (декремент ёки автоматик камайтириш) камайтириш икки хил формада ишлатилади:

префиксли операция - операнда қийматини биттага у фойдаланилгунча камайиши;

постфиксли операция - операнда қийматини биттага у фойдаланилгандан кейин камайиши;

Масалан

ҚҚm m нинг қийматини биттага ортиради;
-- n n нинг қийматини биттага камайтиради;
iҚҚ i нинг қийматини операция бажарилгандан сўнг биттага ортади;
j-- j нинг қийматини операция бажарилгандан сўнг қийматини биттага камайтиради;

nқ4 бўлса nҚҚ=2 ифода қиймати 8 бўлиб n нинг қиймати эса 5 га тенг бўлади.
ҚҚn=2 ифода қиймати 10 бўлиб n нинг қиймати 5 га тенг бўлади.
xҚҚҚb ёки z---d ифодани кўрсак бу ифодалар (x ҚҚ)Қb ёки (z--)-d ифода билан тенг кучлидир.

Мултипликатив операциялар

= - арифметик типдаги операндаларни кўпайтириш;

- бинар минус - арифметик операндаларни айириш ёки кўрсаткичларини айириш учун;
F- арифметик типдаги операндаларни бўлиш;
%- модуль бўйича бўлиш;

Сурилиш операциялари
<< - чапга сурилиш;
>> - ўнга сурилиш;

Разрядли операциялар

& - разрядли конъюнкция (ва);
! - разрядли дизъюнкция ("ёки");
^ - разрядли ёки;

Сурилиш разрядли операцияларнинг бажаралиши натижасида :

4 << 2 16 га тенг
5 >> 1 2 га тенг

Эслатиб ўтамизки $4_{(10)}$қ$100_{(2)}$; $5_{(10)}$ қ$100_{(2)}$; $6_{(10)}$ қ$110_{(2)}$ ва х.к.
2 та сурилиш натажасида 100 коди 1000 айланади унинг ўнли қиймати 16 га тенг.

Муносабат операциялари
< -кичик;
> -катта;
<қ - кичик ёки тенг;
>қ - катта ёки тенг ;
қ қ - тенг;
!қ - тенг эмас;

муносабат операциясининг операндалари арифметик типга тегишли ёки кўрсаткичдан иборат бўлиши лозим .

& - операнда адресини олиш учун;
= - адресга мурожат қилиш;
sizeof - операнда типига эга бўлган операция ўлчамини ҳисоблаш;
sizeof ифода
sizeof (тип)
шаклида қўлланилади.

Бинар (икки жойли) операциялар қуйидаги группаларга бўлинади:

- аддитив;
- мультипликатив;
- сурилиш;
- разрядли;
- муносабат операциялари;
- мантиқий;
- қиймат узатиш;
- "вергул" операцияси;
- қавслар операция сифатида.

Аддитив операциялар

Қ - бинар плюс - арифметик операндалари қўйиш ёки бутун типли
операндалар кўрсаткичини қўйиш учун;
- - бинар минус - арифметик операндаларни айриш ёки кўрсаткичларини айириш учун;

Мантиқий бинар операциялар

&& - арифметик операндалар ёки муносабатлар конъюнкцияси(ва). Бутун сонли қиймат 0 (ёлғон) ёки рост (1)қиймат қабул қилади;
!! - арифметик операндалар ёки муносабатлар дизъюнкцияси (ёки) . Бутун қийматли ифода (0) ёлғон ёки рост (1) қиймат қабул қилади;
Муносабат ва мантиқий қиймат натижалари:

3<5 1 га тенг

3>қ5 0 га тенг;
3қ қ5 0 га тенг;
3!қ5 1 га тенг;
3!қ5 || 3қ қ5 1 га тенг;
3ҚҚ4>5 && 3ҚҚ5>4 && 4ҚҚ5>3 1 га тенг

Қиймат узатиш операторлари

қ - оддий қиймат узатиш ;
масалан: Рқ10.3-2=х;
=қ - кўпайтиришдан сўнг қиймат узатиш;
Р=қ2 ифода РқР=2 билан тенг кучли
Fқ - бўлишдан кейинги қиймат узатиш;
PFқ2.2-d ифода РқPF(2.2-d) ифода билан тенг кучли;
%қ- модуль буйича бўлишдан кейин қиймат узатиш;
Ққ- қўшишдан кейин қиймат узатиш;
аҚқb ифода ақаҚb ифода билан тенг кучли;
- қ - айришдан кейин қиймат узатиш х-қ4,3-z ифода-
хққх- (4,3-z) ифода билан тенг кучли;
<< қ - разрядларни чапга суришдан кейин қиймат узатиш
а<<қ4 ифода ақа<<4 ифода билан тенг кучли;
>> қ - разрядни ўнга суришдан кейин қиймат узатиш;
а>>қ4 ифода ақа>>4 ифода билан тенг кучли;
& қ - разрядли коньюнкциядан кейин қиймат узатиш;
е& қ 44 ифода еқе&44 ифода билан тенг кучли;
! қ -разрядли дизьюкциядан кейин қиймат узатиш;
а!қb ифода ақа!b ифода билан тенг кучли;
^ қ - разрядли ёки чиқариб ташлашдан кейин қиймат
узатиш z^қхҚу ифода zқ^(хҚу) ифода билан тенг кучли;

Вергул операция сифатида

Вергул билан ажратилган ифодалар чапдан ўнгга қараб ҳисобланади. Натижа типи сифатида энг ўнгдаги ифода қиймати типи сақланади:

Х бутун типга тегишли бўлса хқ3 бўлганда хқ2.3=х ифоданинг қиймати 6 бўлади.

4.2. Программанинг тузилиши.

4.2.1.Бошланғич программа.

Си тилида программани аниқлаш ва ёзиш ихтиёрий шаклда, матнли файл сифатида ёзилиши мумкин.

Унда ҳар бир программа процессор олди директивалари, бош функция, ва объектни тавсифлаш ва аниқлаш кетма-кетлигидан иборат.

Процессор олди директивалари буйруқлари программа матнини компиляция қилингунгача қайта ўзгартиришини бошқаради.

‘#‘ белгиси процессор директиваларини белгилаш учун ишлатилади.

Масалан:

include <stdio.h> процессор олди буйруғи программа матнини киритиш ва чиқариш финкциялари кутубхонасини киритади.

Std - standart (стандарт); i-input (киритиш);

o - output(чиқариш); h-head(сарлавҳа);

маъносини билдиради.

Форматланган чиқаришни амалга ошириш учун printf() буйруғи ишлатилади.

printf() функцияни чақириш қуйидагича амалга оширилади.

printf(формат-қатори, аргументлар-руйҳати);

Формат қатори қўштирноқ ичига олиниб, қатор ўзгармаслари, ихтиёрий матни бошқарувчи белгилар ва маълумотларни ўзгартириш хусусиятлари кўрсатилиши мумкин. Масалан экранга " Салом Олам " сўзини чиқариш учун қуйидагича программа тузиш зарур.

```
# include <stdio.h>
void main()
{
  printf("Салом оламfn");
}
```

"fn" белгиси қатор ўтказиб юборишни билдириб, курсорни янги қаторга ўрнатади.

Шунингдек қуйидаги белгилардан ҳам фойдаланиш мумкин:

‘f t’ - горизонтал табуляция;

‘fr’-курсорни координата бошига қайтариш;

‘f’- тескариси аниқ чизиқ;

‘f’’-апостроф;

‘f"’- қўштирноқ;

‘f0’- ноль белгиси;

‘fa’-сигнал қўнғироқ;

‘fb’- битта белгига қайтиш;

‘ff’- қатор ўтказиб юбориш;

‘fv’- вертикал табуляция;

‘f?’- сўроқ белгиси.

Юқоридаги белгилар бошқарувчи кетма-кетликлар деб юритилади. Бу бошқарувчи белгилар ёрдамида дисплейда ахборотни жойланишини ўзгартириш мумкин.

Ахборотларни ташқи шаклини ўзгартиришни бошқариш учун ўзгартиришнинг махсуслатгич белгиларидан фойдаланилади ва улар % белгиси билан биргаликда ишлатилади.

Ҳисоблаш характерига эга бўлган масалаларда биз кўпроқ қуйидаги ўзгартириш махсуслатгичларидан фойдаланамиз.

%d-ўнли бутун сонлар учун(int-типи);

%u-ишорасиз ўнли бутун сонлар учун(unsigned-типи);

%f-ҳақиқий сонлар учун(floatba double-типи);

%l-сўзувчи нуқтали ҳақиқий сонлар учун(double ва fioat-типи);

Агар summa ҳақиқий ўзгарувчи қиймати 2702.3 га тенг бўлса:

printf("ғn summақ%f, summa);

функцияси экранга:

summақ 2702.3 ни чиқаради.

Қуйидаги операторларни бажарилиши натижасида

float m,n;

int k;

mқ15.7; kқ-75; nқ15.33 ни ҳосил қиламиз.

Ўзгарувчилар программа асосий объекти ҳисобланиб, унинг бажарилиши давомида ўз қийматини ўзгартиради.

Арифметик ифодада сонлар, ҳарфлар ва ўзгарувчилар қатнашиши мумкин. Ҳамда К„-, =,F каби амаллар ёрдамида ёзилади. Си тилида бошқа стандарт математик функциялар мавжуд. Бу функциялар қуйидаги 9-жадвалда келтирилган.

9-жадвал

Функция	Си тилида ифодаланиши
smx	sin (x)
cosx	cos (x)
tgx	tan (x)
Inx	log (x)
Ixl	abs (x)
e^x	exp (x)
Vx	sqrt (x)

Си программалаш тилида ўзгарувчилар руйхати фойдаланилмасдан олдин киритилиши зарур, одатда ҳамма ўзгарувчилар фойдаланиладиган биринчи функциядан аввал ёзилади. Функциядаги ўзгарувчиларнинг хусусияти аввалдан эълон қилинади.

Int туридаги сонлар 16 разрядли (-32768 дан 32767 гача)

float 32 разрядли ($3.4=10'^{38}$дан $3.4=10'^{38}$гача).

Си да int ва float дан ташқари маълумотларнинг бир қанча базавий типлари мавжуд.

Кўрсатилган типларнинг ўлчамлари машина турларига қараб ўзгариши мумкин.

Базавий типдан массивлар, тузилмалар ва бирлашмалар, кўрсаткичлар ҳосил қилиш мумкин.

Юқорида кўриб ўтганимиздек printf функцияси объектни экранга чиқарибгина қолмай балки уни исталган кўринишда чоп этади.

Ахборотни киритиш учун тилда scanf, read функцияларидан фойдаланилади.

scanf ((формат катори), (адрес1), (адрес2)...)

Бу ерда printf даги сингари формат қаторларини қўллаш мумкин.

1 -Мисол

```
# include <stdoi.h>
main ()
{
    char name * 30 +;
    printf ("Сизнинг исмингиз нима ?ғп");
    scanf ("% s", name);
    printf  ("Салом % sғn", name);
      }
```

Бу программанинг бажарилиши натижасида:

Сизнинг исмингиз нима? .....

Салом.....

Ҳосил қилинади.

2-Мисол.

$y = \dfrac{x^2 - 3x + 2}{\sqrt{2x^3 - 1}}$ функциянинг х да қийматини ҳисоблаш программасини тузинг.

```
# include <stdio.h>
# include <math.h>
main()
{

    float x,y;
    printf ("X нинг қийматини киритинг:" );
    scanf ("%f", x);
```

$$y = \begin{cases} \dfrac{(m-2)(m-1)m}{6}, & агар\ \ n \le 9 \\ 2^{29-n}, & агар\ \ 9 < n < 29 \\ 1, & агар\ \ n = 29 \\ (n-9)!, & агар\ \ n > 29 \end{cases}$$

```
    y(x=x-3=xҚ2)Fsqrt (2=x=x=x- 1 );
    printf ("xқ% At уқ%f", x,y);
    }
```

3-мисол.

ифоданинг кийматини берилган n да ҳисоблаш программасини тузинг.

```
# include <stdio.h>

# include <math.h>

main()

{

float y ,a,p;
int n,i,a;
printf ("n-ни киритинг:" );
scanf ("%d", n);
if (n<қ9)
                уқ(n=(n-l)=(n-2))F6;
                 else if(n>29)
                 iқl;
                 pқl;
                 while (i<қn-9)

               ақa=i;

                iKҚ;

          уқa;

           else if(nқ29)

          уқl;

             else
               уқexp ((29-n)=log(2));
               printf ("уқ% f nқ%d", y,n);
               }
```

scanf функциясидан фойдаланиш бошқа бир муаммони вужудга келтиради. Биринчи мисолимизни қайта бажариб, исм ва фамилямизни киритсак, фақат исм чиқади холос. Нима учун? Чунки, исмдан кейин қуйиладиган бўш жой scanf функциясига қатор тугаганлигини билдиради. Бу вазиятдан қандай чиқиш мумкин?

Икки хил усул мавжуд. Мана булардан бири:

```
# include <stdio.h>
main( )
{
char name *60+;
printf ("Исмингиз нима?" );
gets (name);
printf ("Салом,% sғn", name);
}
```

gets функцияси сиз нимани терсангиз шуни ўқийди лекин қатор охирига ғO белгисини қўшади. Иккинчи усул қуйидагича:

```
# include <stdio.h>
main()
{
char first *20+,middle*20+,last*20+;
printf ("Исмингиз нима?" );
scanf("%s %s %s", firs,middle,last)
printf ("Салом,% s, ёки %s ғn",first, last);
}
```

Юқоридагилардан ташқари getch функцияси ҳам мавжуд бўлиб, у тугмачадан киритилган ягона белгини ўқийди.

4.2.2. Программа объектларининг “яшаш вақти” ва “ҳаракат” кўлами.

Ўзгарувчининг (глобал ёки локал) “яшаш вакти” қуйидаги қоида орқали аниқланади:

1. Эълон килинган глобал ўзгарувчилар программанинг бажарилиш давомида сақланади;
2. Локал ўзгарувчилар register ёки auto синфидаги ўзгарувчилар эълон қилинган блок бажарилиши давомидагина “яшайди”. Агар бу ўзгарувчи static ёки extern типига таълуқли бўлса, у ҳолда ўзгарувчи бутун программма бажарилиши давомида “яшайди”.

Ўзгарувчи ва функцияларнинг программадаги кўриниши қуйидаги қоидалар ёрдамида аниқланади:

1. Эълон қилинган ёки глобал аниқланган ўзгарувчи, эълон қилинган нуктадан ёки бажалувчи файл охиригача аниқланган ўзгарувчи кўринади. Ўзгарувчини бошқа бажарилувчи файлларда хам кўринадиган қилиш мумкин. Буниг учун бу файлларда уни extern хотира синфи билан элон килиш керак.
2. Эълон қилинган ёки локал аниқланган ўзгарувчи, эълон қилинган нуктадан ёки шу блок охиригача кўринади. Бундай ўзгарувчи локал ўзгарувчи деб аталади.
3. Қамраб олувчи блокларда ва шу жумладан глобал эълон қилинган ўзгарувчилар ҳам ички блокларда кўринади. Бу ичма-ич кўриниш деб аталади. Агар блок ичида эълон

қилинган ўзгарувчи қамраб олувчи блокдаги ўзгарувчининг номи билан бир хил бўлса улар ҳар-ҳил ўзгарувчилар. Қамраб олувчи блокдаги ўзгарувчи ички блокда қўринмасдир

4. **static** хотира синфдаги функциялар улар аниқланган бажарилувчи файлдагина кўринадилар. Бошқа ҳар қандай функциялар эса бутун программада кўринади.

Функциялардаги меткалар бутун функция бажарилиш давомида кўринади.

Функция прототипида эълон қилинган формал параметрлар исмлари уни эълон қилиш нуқтасидан бошлаб функцияни эълон қилиш тугагунгача кўринади.

4.3. Операторлар

4.3.1. Шартли операторлар

Яна оддий программанинг тузилишига қайтамиз, бу фақат биттагина main() функциясидан иборат эди.

```
Процессор олди директивалари
main()
{ обьектларни_аниқлаш;
  бажарилувчи_оператор
      }
```

Ҳар бир бажарилувчи оператор, программанинг кейинги қадамдаги ҳаракатини ифода этади. Оператор қийматга эга эмас. Ҳаракат хусусияти жиҳатидан операторлар икки типга ажратилади: маълумотларни ўзгартириш операторлари ва программа ишини бажарувчи операторлар.

Маълумотларни ўзгартириш типик операторлари-қиймат узатиш ва нуқта ва вергул билан якунланувчи ихтиёрий ифода.

Программа ишини бошқарувчи операторлар программа конструкциясини бошқарувчи операторлар дейилади. Уларга:

- таркибий операторлари;
- танлаш операторлари;
- такрорлаш операторлари;
- ўтиш операторлари киради.

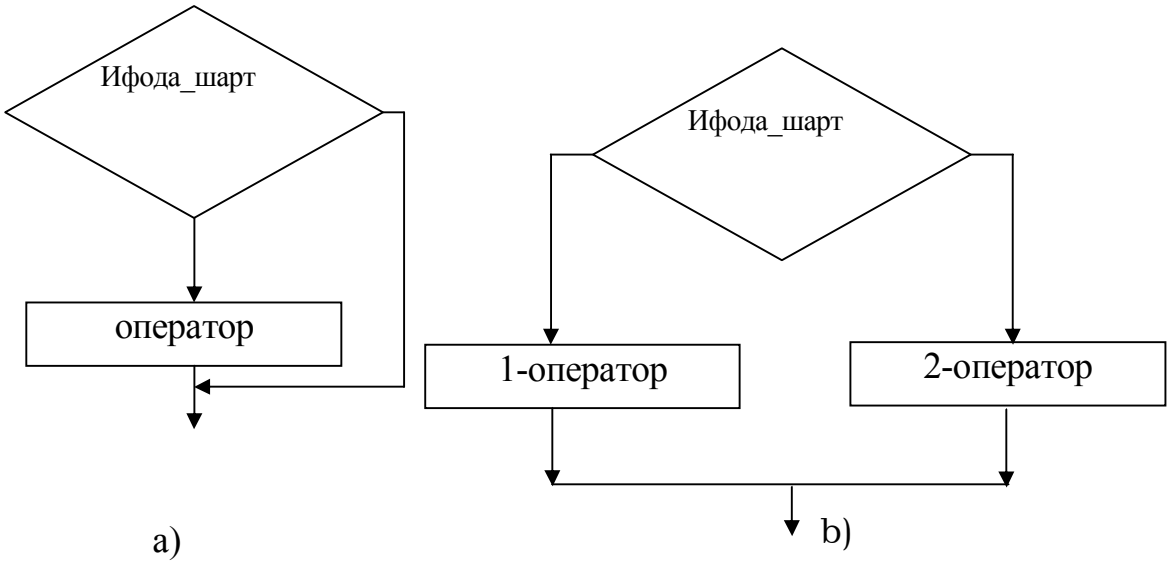
Қуйида таркибий операторларнинг қисми келтирилган.

```
{ nҚҚ ;
sumҚқ (float)n
 }
```

Танлаш операторлари эса бу шартли оператор (if) ва қайта уловчи (switch) дан иборат.

Такрорлаш (цикл) операторлари: шарти олдиндан берилган оператор (while), параметрли такрорлаш (for) ва шарти кейин келувчи оператор (do) лардан иборат.

Ўтиш операторлари бошқаришни шартсиз узатишни таъминлайди. Уларга goto (шартсиз ўтиш), continue (такрорлаш ҳозирги итерациясини якунлаш), break (такрорлашдан ёки қайта



a) b)

уловчидан чиқиш) ва return (функциялардан қайтиш) кабилар киради.

Биз бу параграфда шартли операторлар билан танишиб ўтамиз. У қуйидаги қисқа кўринишга эга:

If

If(x<o && x>-10) xқ-x;

Қисқа шаклдан ташқари, яна тўлиқ шаклга ҳам эга бўлиб, унинг кўриниши:

If(шартли_ифода)

1-оператор;

else

2-оператор;

Масалан:

If(x>0)

bқx;

else

bқ-x;

Уларнинг бажарилиш схемаси қуйидаги расмда (8-расм) келтирилган:

8-расм.

a) Тўлиқмас қисқа шакл. b) Тўлиқ шакл.

Масалан: ax^2 қbxқcқ0 квадрат тенгламанинг x_1, x_2 - ҳақиқий илдизларини аниқлаш:

```
# include(stdio.h)
main( )
{
float a,b,c,d,x1,x2;
dқb=b-4=a=c
```

```
if (d>қ0.0)
{
        x1қ(-bҚsqrt(d)F2Fa;
x2қ(-b-sqrt(d)F2Fa;
printf (“ғn x1қ %f  x2қ %f”, x1, x2 );
  }
else
printf (“ғn Ҳақиқий илдизлар мавжуд эмас”);
} кўринишида аниқланади.
```

4.3.2. Такрорлаш операторлари (for, while, do)

Бошқа програмалаш тиллари сингари СИ тилида ҳам такрорлашларни ҳосил қилиш учун махсус воситалар мавжуд. Кўпчилик программалаш тилларини такрорлаш оператори икки элементдан: бош қисм ва танадан иборат. Тана такрорлашда бажариладиган операторларни ўз ичига олади.

Бош қисм эса танадаги операторларни такрорий бажарилишини таъминлайди.СИ тилида такрорлаш операторларининг уч тури ишлатилади, улар қуйидаги хизматчи сўзлар билан белгиланади.

while, for, do.

while ва for қуйидаги схемада тузилади.

Такрорлаш бош қисми

такрорлаш танаси

do такрорлаши бошқача тузилмага эга бўлиб (юқоридан ва пастдан) конструкциясига асосланади. Шунинг учун ҳам do такрорлашида такрорлашнинг бош қисми ҳақида гапириш ўринсиздир.

Шуни эслатиб ўтамизки, ҳар уччала такрорлаш операторида такрорлаш танаси-бу {} белгиси орасидаги алоҳида ёки ташкилий оператор бўлиши мумкин. Такрорлаш танаси бўш оператор бўлиши ҳам мумкин.

while-такрорлаши қуйидаги кўринишга эга:

while (ифода_шарт)

такрорлаш танаси

Ифода_шарт сифатида кўпроқ муносабат ёки мантиқий ифода фойдаланилади. Агар у рост қиймат қабул қилса, такрорлаш танаси ифода_шарт ёлғон бўлгунча бажарилади.

Шунга эътибор берингки, ифоданинг тўғрилиги такрорлаш танасининг бажарилишини таъминлайди. Шундай қилиб, олдиндан ёлғон бўлган ифода_шартда такрорлаш бирор марта ҳам бажарилмайди.

Ифода_шарт арифметик ифода бўлиши ҳам мумкин

do такрорлаши қуйидаги кўринишга эга:

do

такрорлаш танаси

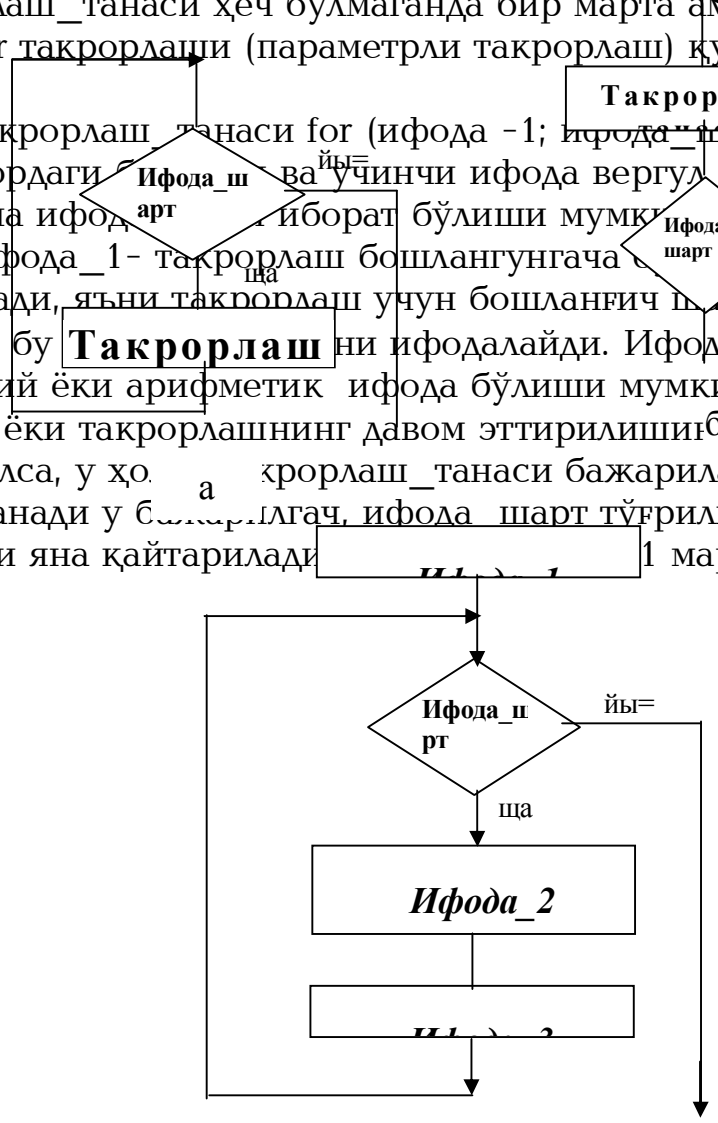
while (ифода_шарт)

Ифода_шарт while такрорлашдаги каби мантиқий ёки арифметик ифода бўлиши мумкин. do такрорлашда такрорлаш_танаси ҳеч бўлмаганда бир марта амалга ошади.

for такрорлаши (параметрли такрорлаш) қуйидаги кўринишга эга:

такрорлаш_танаси for (ифода -1; ифода_шарт; ифода-3) операторлардаги биринчи ва учинчи ифода вергул билан ажратилган бир неча ифодадан иборат бўлиши мумкин.

Ифода_1- такрорлаш бошлангунгача бажариладиган ҳаракатни билдиради, яъни такрорлаш учун бошланғич қийматни ифодалайди. Кўпроқ бу **Такрорлаш** ни ифодалайди. Ифода_шарт – одатда мантиқий ёки арифметик ифода бўлиши мумкин. Бу шартнинг тугаши ёки такрорлашнинг давом эттирилиши Агар у рост бўлса, у ҳо такрорлаш_танаси бажарилади. Сўнгра ифода_3 ҳисобланади у бажарилгач, ифода_шарт тўғрилиги ҳисобланади ва ҳаммаси яна қайтарилади 1 марта бажарилади,



а б в

9-расм

ифода_шарт ва ифода_3 такрорлаш танасининг ифода_шарт ёлғон бўлгунча давом этади.

9-расмда while, do ва for такрорлашларини ташкил қилиш келтирилган:

а) while буйруғи такрорлаши; б) do буйруғи такрорлаши;

в) for буйруғи такрорлаши.

Масалан ℓ^x нинг тақрибий қийматини ҳисоблашни кўриб ўтайлик. Бизга математик анализ курсидан маълумки

$$\ell^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \sum_{i=0}^{n} \frac{x^n}{n!}$$

формула билан ҳисобланади.

Биз қуйида программанинг бир қисмини кўриб ўтамиз.

```
:
iқ2;
bқ1.0;
rқx;
while (r>eps!!r<-eps)
{
bқbқr;
xқr=xFi;
iққ;
}
do такрорлаши орқали
iқ1;
bқ0.0;
rқ1.0;
do {
bқbқr;
rқr=xFi;
iққ;
}
while(r>қeps!!r<қ-eps);
for параметрли  такрорлаш орқали
iқ2;
bқ1.0;
rқx ;
for(;r>eps!!r<-eps;)
{
bқbқr;
rқr=xFi;
iққ;
}
```

Лекин for цикли юқоридагиларга қараганда бир мунча афзалликларга эга бўлиб, уни қуйидагича ёзишимиз мумкин.

```
for(iқ2,bқ1.0,xқx;x>eps!!x<-eps;iҚҚ)
{
bқbҚx;
xқx=xFi;
}
```

йиғиндини ҳисоблаш программасини тўла матни қуйидагича:

```
F=экспонентани ҳисоблаш программаси =F
Программани бажарилиши натижасида:
X нинг қийматини киритинг: xқ1
Аниқликни киритинг : epsқ0.01
натижа: 2.708333
аниқлик     0.008333
Қатор ҳадлари сони: 6 ни ҳосил қиламиз.
```

Программани бажарилишининг бошқа варианти.

```
X ни қийматини киритинг: xқ1
Аниқликни киритинг: eps қ -0.3
Аниқликни киритинг: epsқ0.0001
Натижа: 2.718254
Аниқлик: 0.000025
Қатор ҳадлари сони: 9та
```

Охирги вариантда дастлаб нотўғри epsқ-0.3 қиймат киритилди шунинг учун ҳам аниқликни қайта киритишга тўғри келди.

4.3.3. Break оператори

Юқорида кўриб ўтган мисолимизда бошланғич маълумотларни қайта киритиш унча қулай эмас, экранда хатолик ҳақида ҳеч қандай хабар берилмади. Бундай ҳолатлардан чиқишнинг турли йўллари мавжуд. Улардан бири метка ва ўтказиш оператори goto дан фойдаланишдир. Лекин структурали программалаш нуқтаи назаридан бундай усул корректсиз ҳисобланади.

Бундай натижага структурали программалаш принципларини бузмаган ҳолда, такрорлаш танасида узилиш оператори break ни қўллаш орқали ҳам эришиш мумкин.

Бу оператор такрорлаш бажарилишини тугатади ва бошқаришни такрорлашдан кейинги операторга узатади.

Масалан: $c = \sum_{i=1}^{n} a^i$, a,n(n>0)-ўзгарувчиларни киритиш ёрдамида ҳисобланг.

Программанинг кўриниши қуйидагича:

```
F =сон даражалари йиғиндиси  =F
# include <stdio.h>
main()
{
double a, c; F= c –йиғинди = F
int i,n;
```

```
printf(“ғn a нинг қийматини киритинг ақ”);
scanf (“%f”, &a );
while(1)
{
printf(“ғn n- нинг қийматини киритинг nқ” );
        scanf(“%d”,&n) ;
if (n>0) break ;
printf(“хато ! n>0 бўлиши керак ғn” );
}
for( сқ0.0, рқ1.0, іқ1 ; i<қn ;  іҚҚ)
{
p=қa;
cҚқр;
}
printf( ”ғn йиғинди сқ%f“,c);
}
```

Программанинг бажарилиши натижаси :

```
а нинг қийматини киритинг  ақ8.8
n- нинг қийматини киритинг  nқ-15
хато! n>0 бўлиши керак
n-нинг қийматини киритинг: nқ15
йиғинди сқ1. 65817е Қ14
```

4.3.4. Continue оператори

Такрорлаш танасида операторларни бажарилишига аралашиш имкониятларидан бири continue операторидир.

У такрорлашнинг ихтиёрий қисмида такрорлаш бажарилиш шартини текширишга ва унинг натижасида такрорлашни тугатиш ёки давом эттиришга хизмат қилади. Бу операторни бажарилишини қуйидаги программа орқали кўришимиз мумкин.

```
# include <stdio.h>
F= мусбат сонлар йиғиндисини топинг =F
main()
float s,x ; F= s –йиғинди,x-киритилувчи сон=F
int k;  F= k – мусбат ҳад сони=F
printf (“ғ n сонларни киритинг охирида 0 билан” )
for (хқ1.0, sқ0.0 , kқ0; x!қ 0.0 )
{
scanf(“%f”, &x) ;
if  (x<қ 0.0) continue;
   kҚҚ; sҚқx;
```

```
}
printf("ғn йиғинди қ%f  мусбат элемент сони қ%d",s,k) ;
}
```
Программанинг бажарилиш натижасида:
Сонларни киритинг охирида 0 билан
5. 0 -7.0 11 -4 10 0. 0
йиғинди қ 26.0 мусбат элемент сони қ 3.

4.4.Массивлар

Массивлар- бу бир ҳил типдаги (double, float, int ва х.қ.) элементлар гуруҳидан иборат. Компилятор эълон килинган массивдан унинг элементлари типи ва уларнинг сонги тўғрисида маълумотларга эга бўлиши керак. Массивни эълон қилишнинг икки хил формати мавжуд:

махсуслаткич_типи *константали - ифода+;

махсуслаткич_типи * +;

Махсуслаткич бу массивнинг идентификаторидир.

Махсуслаткич_типи эса эълон қилинаётган элементларнинг типини беради. Массив элементларнинг типи функция ва void типига таълуқли бўлмаслиги керак.

Константали-ифода массив элементлари сонини билдиради. Массивни эълон қилишда қуйидаги холлардагина константали-ифодани тушириб қолдириш мумкин:

- массив функциянинг формал параметри сифатида эълон қилинганда,
- эълон қилинган массив бошқа файлдаги аниқ массивга йўлланган бўлса.

Охирги мисолда w*3+*3+ массиви эълон қилинган. Фигурали қавислар ичига олинган ифодалар массивнинг сатрига мос келади, кавслар қолдириб кетилса инициализация нотоғри бажарилади.

СИ тилида массивларнинг кесишмасидан ҳам фойдаланиш мумкин, лейкин бу кесишмалардан фойдаланишга кўп чекловлар куйилади. Кесишмалар бир ёки бир неча жуфт квадрат қавсларни олиб ташлаш оркали амалга оширилади. Массивларнинг кесишмаси ҳисоблаш жараёнини ташкил қилишда ишлатилади.

Мисоллар:

int s*2+*3+;

Агар айрим функцияга s*0+ орқали мурожаат қилинса, s массивнинг нолинчи сатри олинади.

int b*2+*3+*4+;

b массивга b*1+*2+ орқали мурожаат қилинганда тўрт элементдан иборат вектор узатилади, b*1+ орқали мурожаат қилинганда эса 3x4 икки ўлчамли массив узатилади.

Белгили массивларни эълон қилиш куйидагича бўлади:

char str* + қ “белгили массивни эълон қилиш”.

4.4.1. Массивлар устида амаллар

Массивларни қўлланилишига доир мисоллар кўриб ўтамиз.

Мисол 1.

$$S = \sum_{j=1}^{10} \left[\prod_{i=1}^{5} b_{ij} + \sum_{i=1}^{5} b_{ji} \right]$$

Программасини тузамиз:

```
# include <stdio.h>
main ()
{ float p,c;
int i , j;
double b*10+ *5+
printf(“ғ n  Массив элементларини киритинг”) ;
for ( jқ1 ; j<10 ; jҚҚ)
{
for ( iқ1 ; i<5 ; iҚҚ)
{
printf(“b* %d  +  * %d+қ”, j,i )
scanf(“ %f %f , & x* j+  *i  +”)
}
 for ( sқ0.0 , jқ0, ; j< 10 ; jҚҚ)
  {
   for (pқ1.0 , cқ0. 0 , iқ0 ; i<5 ; iҚҚ)
        {
              p=қ a* j+  *i+  ;
              c Ққ a* j+  *i+  ;
              }
              sҚқ cҚp ;
          }
        }}
```

Мисол 2.

Бир ўлчамли массив элементларини тартиблаш.

a(n) 1< n< 100 массивнинг элементларини қийматлари ўсиб бориш тартибида жойлаштиринг.

```
          # include <stdio.h>
          main (){ int n , i, j ;
          float a *100+  ,b ;
          while (1)
           {
   printf(“ғn  Элементлар сонини киритинг nқ” ) ;
   scanf(“ %d”,& n);
   if ( n> 1 && <қ100) break ;
```

```
   printf(“ хато! 1<n<қ100 бўлиши зарур !”) ;
        }

   printf(“ғ n массив элементлари қийматларини киритинг : ғn”) ;
   for ( jқ0 ; j<n ; jҚҚ)
    {
      printf( a *%d  +  , jҚ) ;
      scanf(“ %f ” , &a* j+  ) ;
     }
    for (iқ0; i<n-1; iҚҚ)
                for (jқiҚ1; j<n; jҚҚ)
                if (a*i+>a*j+)
                                {
                                  bқ a*i+;
                                  a*i+қb*j+;
                                }
                        printf(“ғn Тартибланган массив: ) n”);
                                for (jқ0; j<n; jҚҚ)
                                printf(“a(%d)қ%f ғn”,jҚ1,a*j+);
}
```

Программанинг бажарилиши

```
 Элементлар сонини киритинг nқ3
a*1+қ15.8
a*2+қ11.2
a*3+қ-2.3
Тартибланган массив:
        a*1+қ-2.3
        a*2+қ11.2
        a*3+қ15.8
```

4.5. Функция.

4.5.1. Функцияни аниқлаш.

Си тили синтаксига кўра унда учта ҳосилавий тип аниқланади: массив, кўрсаткич, функция. Биз бу бўлимда функция ҳақида гапиришамиз.

Функциялар ҳақида гапирганда, Сида икки тамонлама қараш зарур. Биринчи тамондан, юқорида айтганимиздек функция ҳосилавий типга қараса. Иккинчи томондан Си тилида функция бажарилувчи модул ҳисобланади. Бу ерда иккинчи тушунча бошқа программалаш тилларидаги процедура, қисм программа, қисм программа-функция синонимлари билан мос келади. СИ тилида ҳамма функциялар стандарт талаби асосида ягона кўринишга эга:

тип функция _*исми*_ (параметрлар _ хусусияти)
функция _танаси.

Биринчи қатор – бу мазмунига кўра функциянинг бош қисми, ўзининг тимсолидан охирида *;+ нуқта вергул қатнашмаслиги ва албатта расмий параметрларни қатнашиши билан фарқ қилади.
Бу ерда тип қиймат қайтармайдиган ***void*** турига ёки қиймат қайтарувчи функциялар турига қарашли бўлиши мумкин.
Биз олдинги бўлимларда қиймат қайтарувчи базавий турларни кўриб ўтдик (char, int,double ва бошқалар).
Функция исми ё бош функция учун main(), ёки программист томонидан ихтиёрий танланадиган программадаги хизматчи сўзлар ва бошқа объектлар исми билан устма-уст тушмайдиган исм (идентификатор) бўлиши лозим.
Параметрлар _ хусусияти эса – бўш ёки расмий параметрлар рўйхати бўлиб, улар қуйидаги кўринишга эга бўлади:
типни _ белгиланиши
параметр _ исми.
Параметрлар рўйхати нуқта вергул сўнгида *. . .+ (кўп нуқта) билан тугаши мумкин. Кўп нуқта функцияга кўп сонли ҳақиқий параметрлар билан мурожаат қилишни англатади. Бундай имконият функция танасига махсус воситалар ёрдамида бириктирилиши зарур.

Масалан:
int printf(conct char = format, . . .);
int scanf(conct char = format, . . .);
функцияларни назарий жихатдан форматланган киритиш ва чиқаришни чекламаган миқдордаги ҳақиқий параметрлар учун қўллашни таъминлайди.
Функция_танаси – бу фигурали қавслар ичига олинган сарлавхадан кейин келувчи функцияларни аниқлаш қисмидир.
Функция_танаси ё блок ёки таркибий оператор бўлиши мумкин. Си да функция танаси ичида бошқа функцияни аниқлаш имкони йўқ.
Функция танасида функция чақарилган нуқтасидан чиқиш оператори бўлиши ҳамиша ҳам талаб этилавермайди, қайсики у икки шаклига эга:
return;
return;ифода;
Биринчи шакл ҳеч қандай қиймат қайтармайдиган ***void*** типига тўғри келади.
Шундай қилиб Си тилида функция кўрсатилган типда қиймат қайтарувчи, ҳеч нарса қайтармайдиган параметрлар билан ва параметрларсиз қўлланиши мумкин.
Функция аниқланиш сўнгида функцияни аниқлаш шаклини кўрсатамиз:
тип_натижа
функция_исми(параметрлар_рўйхати) параметрлар_хусусияти;
функция_танаси

4.5.2. Функцияни эълон қилиш ва чиқариш.

Функцияга аниқ мурожаат қилиш учун у ҳақидаги барча маълумотлар компиляторга олдиндан маълум бўлиши зарур, яъни функцияни чақиришдан аввал стандарт томонидан тавсия этилишига ўша фанга унинг аниқланишини ёки тавсифини жойлаштириш керак.
Стандарт бўйича функцияни аниқлаш учун унинг тимсоли хизмат қилади.
тип функция_*исми (*парметрлар _ хусусиятлари);
Масалан:
double f(int n, float x);
double f(int, float);
лар бир-бири билан эквивалент.
Функцияга мурожаат қилиш учун «()» операцияси билан ифодалар фойдаланилади:
функцияни_белгиланиши (ҳақиқий_параметрлар_рўйхати) «()» операциянинг операндалари сифатида функцияни_белгиланиши ва ҳақиқий_параметрлар рўйхати.
Функцияни белгиланиши бу унинг исми ҳақиқий параметрлар рўйхати, аргументлар деб аталиб, уларнинг сони функциянинг расмий параметрлари сонига тенг.
Ҳақиқий ва расмий параметрлар орасидаги мослик уларнинг рўйхатда ўзаро жойлашишига қараб аниқланади. Ҳақиқий параметрларни ҳисоблаш (ўнгдан чапга ёки чапдан ўнга) Си тили стандартида аниқланмаган. Расмий ва ҳақиқий параметрлар типлари бўйича мос бўлиши зарур. Ҳақиқий параметр типи расмий параметр типи билан бир хил бўлгани яхшироқ. Акс ҳолда, бундай типларни келтириш мумкин бўлса, компилятор типларни яратиш буйруқларига автоматик равишда қўшади.
Масалан функция тимсоли билан қуйидагича аниқланган бўлсин:
int g(int,long);
Программада уни чиқариш қуйидагича:
g(5.0Қm,6.3eҚ2)
Бу ерда иккала параметр ҳам double типига эга. Компилятор функция тимсолига асосланиб, автоматик равишда мана бундай алмаштиришни қарайди:
g((int)(3.0Қm), (long) 6.3eҚ2)
Функция ифода ҳисобланганлиги сабабли, унинг танасидаги операторлар бажарилгач, чақириш нуқтасига айрим қийматлар қайтарилади, унинг типи эса қатъий равишда унинг тимсолида аниқланган функция исми олдида келувчи типга мос келади.
Масалан:
float ft(douible x, int n)
{
if (x<n) return x;

return n;
}
ҳамиша float типидаги қийматни қайтаради.
Функцияни чақириш–ифода ҳисобланиб, бундай ифода программа матнида функциянинг қайтарадиган қийматларига боғлиқ бўлади.
Масалан:
void print (int gg, int mm, int dd)
 {
printf (“ғn йил:%d”, gg);
printf (“ғn ой:%d”, mm);
printf (“ғn кун:%d”, dd);
}
Унга мурожаат
print (2001, 10, 15);
Экранга қуйидагича натижани чиқариш билан якунланади.
Йил: 2001 ой:10 кун:15
Учбурчакнинг томонлари берилса, унинг периметри ва юзини ҳисобловчи программани кўриб ўтайлик.
Бизга геометрия курсидан маълумки, томонлари а, в, с бўлган учбурчакни ясаш учун аҚb≥c, aҚc≥b, bҚc≥a тенгсизликлар бажарилиши керак. Бу ҳолда, учбурчак периметри рқаҚвҚс унинг юзи эса

$$S=\sqrt{\frac{p}{2}(\frac{p}{2}-a)(\frac{p}{2}-b)(\frac{p}{2}-c)}$$

формулалари ёрдамида топилади.
Программанинг кўриниши қуйидагича.
# include <stdio.h>
# include <match.h>
main()
{
  float x,y,z,pp,ss;
        F =тимсол:=F
int triangle(float,float,float,float=,float=);
printf("ғ n x ни киритинг хқ");
scanf("%f ", &x);
printf("ғt y ни киритинг: уқ");
scanf("%f ", &y);
printf("ғt z ни киритинг: zқ");
scanf("%f ", &z);
if (triangle(x,y,z,&pp,&ss)қ қ1)
{
print("Периметрқ% f ",pp);
print ("юза қ% f", ss);
}
else

```
print ("ғn маьлумотлар хато");
}
F= функцияни аниқлаш =F
int triangle(float a,float b,float c,float =perimeter,float  =arrea);
{
float e;
        = perimeterқ=arreaқ0.0;
        if (aҚb<қc:: aҚc<қb:: bҚc<қa);
return 0
=perimeterқ aҚbҚc
        eқ=perimeterF2;
        =arreaқsqrt(e=(e-a)=(e-b)=(e-c));
return 1
}
```

Программанинг бажариш натижаси:

```
        x ни киритинг: xқ3
        y ни киритинг: yқ4
        z ни киритинг: zқ5
        периметрқ12.000000
        юзақ6.000000
```

4.6 Процессор директивалари ва компиляторга кўрсатма

4.6.1. Номланган ўзгарувчи ва макроаниқлашлар

Бизга маълумки Си тилида ҳар бир программа процессор олди директиваларидан иборат. Процессор олди буйруқлари директивалар деб аталади. Уларнинг ҳар бири # белгиси билан белгиланади. Биз бундан олдинги бобларда # include директивасини кўриб ўтдик. Директиваларнинг умумий формати қуйидагича:

директива_исми процессор_лексемлари

белгисидан олдин ва кейин пробеллар рухсат этилади. Матнли қатор охири процессор олди директивасининг тугашини билдиради.

#define директиваси кўп ишлатиладиган идентификаторларни ўзгармасларга алмаштириш учун хизмат килади ва номланган ўзгарувчи деб аталади.

Бу директива икки хил синтактик шақлга эга бўлади:

```
#define идентификатор матн;
#define идентификатор(параметрлар руйхати)матн.
```

Масалан:

```
#define WIDTH 80
#define LENGTH (WIDTHҚ10)
```

Агар экранда бирор ўзгарувчининг қийматини тез-тез чоп қилиш керак бўлса, уни қуйидагича амалга ошириш мумкин:

```
#define AK printf (“ғэлемент номериқ%d.”,N);
```

бу директивадан сўнг,

int Nқ4;
AK;
операторлар кетма-кетлиги,
Элемент номери қ4 ёзувини чиқаради.
#define k 50
#define PK printf(“n Элементлар сони %d.”,K)
....
PK;
....
Экранга “Элементлар сони PKқ50” ни чиқаради.
#undef исм (идентификатор)
буйруғи бажарилиши билан исм ёки макрос аниқланишини бекор қилиш мумкин.
Масалан:
#define M 16
#undef M
M қ 16 ни бекор қилади.

....
A қ 10
....
#define A x
....

A қ 5
#undef A
....
B қ A

B қ 10 га тенг қийматни қабул қилади.

4.6.2. Шартли компиляция, қаторни белгилашни бошқариш ва хатолар билан ишлаш директивалари.

Си да шартли компиляциялаш қуйидаги директивалар ёрдамида амалга оширилади.
#if бутун_ўзгармасли_ифода
#ifdef идентификатор
#ifndef идентификатор
#else
#endif
#elif

Биринчи учта директива шартни текшириш, кейинги иккитаси эса шартни ҳаракат диапазонини аниқлашни билдиради.
Шартли компиляциялаш директиваларини қўллаш умумий тузилиши қуйидагича:
#if...
1 – матн
#else

2 – матн
#endif
Шартли компиляциялаш ташқи қурилмалар учун программалар ёзишда, созланаётган чоп қилишни ажратиш ва шунга ўхшаш ҳолатларда ишлатилади.
Қуйидаги директива:
#ifdef идентификатор
ёрдамида #define директиваси ёрдамида ҳозирги вақтда #ifdef директивасидан кейин жойлаштирилган идентификатор аниқланган бўлса, у ҳолда матн_1 – компилятор томонидан фойдаланилади.
Қуйидагича:
#ifndef идентификатор
директиваси ёрдамида эса тескари шарт – идентификаторнинг аниқланмаганлиги яъни идентификатор #define функцияси фойдаланилмаганлиги ёки аниқлаш #undef буйруғи билан бекор қилинганлиги текширилади.
Программани созлашда назорат қилувчи ахборот воситаларини ёки чиқариб ташлаш учун шартли компиляцияни қўшиш қулай.
Масалан:
#define DEBUG
....
#ifdef DEBUG
printf (“Созлашда чоп қилиш”);
#endif
Мулти тармоқланишларни амалга ошириш учун:
#elif бутун_ўзгармасли_ифода
директиваси киритилган.
Бу ерда бутун_ўзгармасли_ифодасига қўйиладиган талаблар #if директивасila келтирилганига ўхшаш бўлади.
Бу директивани қўллаш структураси билан берилган матн қуйидагича тус олади:
#if шарт
if_учун_матн
#elif ифода_1
Матн_1
#elif ифода_2
Матн_2
#else
Матн_3
Else_учун_main
endif
Шундай қилиб матннинг фақат шартли директивалар томонидан ажратилган қисми ишлатилади.
Қаторларни номерлаш учун:
line ўзгармас

директиваси компиляторга кейинги қатор ўнли бутун ўзгармас билан аниқланадиган номерга эга бўлишини кўрсатади. Директива бир вақтнинг ўзида нафақат қатор номерини, балки файл исмини ҳам ўзгартириш имконини беради. Унинг умумий шакли:
line ўзгармас «файл исми».
Масалан AAA.C программа матни;
```
# define  N 3
voidmain ()
{
# line 23 “files.c”
double  z(3=n);
}
```
процессор олди ишловидан кейин “aaa.i” исмли файлда қуйидаги қатор ҳосил бўлади;
```
aaa.c 1;
aaa.c  2;
aaa.c  3;{
aaa.c  4;
file.c  23;  double z(3=3)
file.c  24;}
```

Қуйидаги директива:
error лексем кетма-кетлиги
хатоликларни олдини олиш хабарларини етказади.
Масалан:
```
# define aa 5
# if (aa ! қ5)
# error  aa  5 га тенг бўлиши керак!
```
интегралашган муҳитда (масалан TURBO C да) ахборот қуйидаги кўринишга эга бўлади:
Fatal <файл исми><қатор номери>;
Error directiv: AA 5 га тенг бўлиши керак!

4.6.3. Бўш деректива

Си тилида ҳеч қандай ҳаракатни ифодаламайдиган директива мавжуд бўлиб, у қуйидаги кўринишга эга:
#
программа
pragma лексем_кетма-кетлиги
Компилятор билан боғлиқ аниқ ҳаракатни ижро этишни билдиради. Айрим компиляторлар программа матни таркибида мавжуд ассемблер буйруқларини сони тўғрисидаги вариант бу деректива таркибига киради.
Бу ерда # pragma буйруғи муҳим ва турлича бўлиши мумкин. Улар учун стандарт йўқ. Агар аниқ бир процессор олди ишлови унга

нотаниш программани учратса, уни бўш деректива сингари бекор қилади.
Айрим компиляторларда
pragma pack(n)
бу ерда, n 1,2 ёки 4 бўлиши мумкин.
Бунда, pack программаси тузилмаларда ва бирлашмаларда қўшма элементларга таъсир этади.
Олдиндан аниқланган макроисмлар мавжуд бўлиб, улар қуйидаги хабарларни олиш имконини беради.

_ _ LINE _ _ - ўнли ўзгармас – Си програмаси билан ҳозирги ишланаётган қатор номери.

_ _ FILE _ _ - компиляция қилинаётган файл номи.
_ _ DATE_ _ - «ой кун йил» шаклидаги белгилар қатори.
_ _ TIME _ _ «соат: минут: секунд» белгилари қатори – ишлаш вақтини аниқлайди.

4.7. Хотира модели

Си программалаш тили бешта турдаги хотира моделига эга. Ушбу моделлар учун қуйидаги чекловлар ўринли:

- Ҳеч қайси такрорланувчи модул 64 кб дан юқори ҳажмни эгаламайди.
- Агар жуда катта хотира модели қўлланилмаса ва маълумот элементи (масалан масив) **huge** калит сўзи билан ёзилмаса, у 64кб юқори бўла олмайди).

4.7.1. Модел турлари

Кичик (smail) модел
Бунда программа бир сегмент маълумоти ва бир буйруқ сегменти билан яратилади (сегмент 64 кб.га тенг). Бундан келиб чиқадики, бу моделларда программа 128 кб дан ошмайди. Ушбу моделда ҳеч қандай калитлар ишлатилмаса ҳамма кўрсаткичлар ***near*** типига эга бўлади, лекин far, huge калит сўзлари билан ҳам киритишни амалга ошириш мумкин.
Ўрта (meditum)модел
Прогарамма буйруқларнинг бир неча сегменти ва битта маълумот сегменти билан яратилади. Одатда программалар учун буйруқларнинг катта ўлчами қўлланилади. Бунда ҳар бир бажарилувчи модуль ўзининг буйруқ сегментига эга бўлади.
Компакт (compact) модел
Программа бир неча маълумот сегментлари ва битта буйруқ сегменти билан яратилади. Олдиндан кўрсатилмаса ҳамма код кўрсаткичлари бу моделда **near** типига, маълумоталар эса – **far** типига эга бўлади. Бошқа код кўрсатичларини калит сўз **far** ёрдамида ва маълумотларни калит сўзлар **near** ва **huge** ёрдамида ҳам киритилиши мумкин.

Катта (large) модел

Программа бир неча маълумотлар сегменти ва бир неча буйруқлар сегменти ёрдамида яратилади. Аввалдан кўрсатилмаса бу моделда элемент коди ва маълумотлар **far** типидаги адресга эга бўлади. Бошқа код кўрсаткичларини калити сўз **near** ва **huge** ёрдамида ҳам киритиш мумкин.

Жуда катта (huge) модел

Программа бир неча буйруқ ва маълумот сегментлари билан тўлдирилади. Бундай сегментлар бир қанча бўлиши мумкин. Бу моделда маълумот ўлчамига (масалан массивга) 64 кб. ли чеклов олиб ташланади, лекин сегментларга қуйидаги чекловлар қолади:

— маълумотнинг ягона элементи (масалан массив элементи) 64 кб дан юқори бўлмаслиги;

— ихтиёрий 128 кб. дан юқори массивлар учун ҳамма элементлар 2 га каррали ўлчамга эга бўлиши керак.

Sizeof опреторининг бажарилиш натижаси ва икки кўрсаткичнинг фарқи int типига эга бўлади, бунинг учун **sizeof** операторини қуйидаги конструкцияда фойдаланиш керак:

(long)sizeof(huge_item),

Бунда **huge_item** талаб қилинган элемент.

Иккита кўрсаткичнинг фарқини **huge**да олиш учун, элементларни қуйидаги конструкцияда олиш керак:

(long) (huge_ptr1-huge_ptr2).

4.7.2.Турбо СИ хотира модели

Турбо-С программалаш тили юқорида кўрсатилган моделлардан ташқари яна бир хотира модели (**ting**)га эга. Бу энг кичик хотира моделидир. Бунда ҳамма тўрта (CS,DS,SS,ES) регистрлари битта ягона адресни сақлайди, шунинг учун 64 кб маълумот кодини массивларни сақлашга ажратилади. Ҳамиша **near** типидаги кўрсаткичлар қўлланилади. **Tiny** типидаги хотира моделидан фойдаланувчи программа **=.com** типидаги форматга эга бўлади.

Элементлар ёки кўрсаткичларни элементга калит сўзлар **near, far** ва **huge** ёрдамида модификатциялаш учун қуйидаги қоидани доимо эсда тутиш зарур:

1. Калит сўзлар элемент ёки кўрсаткични ўнгдан модификациялайди. Масалан, **char far == p**; шуни билдирадики, **p**- бу **far** га кўрсаткич **char** га кўрсаткичдир;
2. Агар ўнгда калит сўзидан кейин идентификатор келса, у ҳолда бу элемент стандарт маълумот сегменти **near**да ёки бошқа **far** ё **huge**да жойлашишини аниқлайди. Масалан, **char far A**; бу ерда **A char** типидаги **far** адресли элементларга эга;
3. Агар ўнгда калит сўзидан кейин кўрсаткич келса у ҳолда у адрес ўлчами: 16 бит **near** ёки 32 бит (**huge** ё **far**)ни аниқламайди. Масалан char far=p. Бу ерда p char типидаги элемент far кўрсаткичини (32 бит) билдиради;

Мисол:

Char a *3000+-a стандарт маълумот сегментида жойлашади.
Near ва **far** ларни функцияларга қўллаш худди маълумотлар учун қўлланиши қоидаларига ўхшамайди лекин калит сўз **huge** ни функцияларга қўллаб бўлмайди.
Мисол:
Char far fun();
Char far fun() {|=..=|} – бу ерда fun(), char типидаги элементни қайтарувчи ва 32 битни адресдан чақирилувчи функциялардир.

4.8. Файллар устида операциялар

Файллардан матнларни ўз ичига олиш учун #include директиваси қўлланилади ва уни ёзишнинг қуйидаги уч шакли мавжуд:
#include <файл_номи>
#include “файл_номи”
#include макрос_исми
Шу вақттача биз мисолларда биринчи шаклни қўлладик. Файл номи <> белгиси ичига олинса, процессор файлни каталогнинг стандарт системасидан қидиради. Агар файл номи “” ичига олинса, процессор фойдаланувчининг ҳозирги каталогини, сўнгра каталогнинг стандарт системасини қидиради.
Макрос_исми – бу #define директиваси орқали киритилган исм ёки макрослардан иборат.
Фойдаланувчи Си тили билан иш бошлашда киритиш-чиқариш воситалари билан тўқнаш келади. Бунинг учун программа матнида қуйидагича директива жойлаштирилар эди:
#include <stdio.h>
Бу директивани бажарилишида программа кутубхонадан киритиш ва чиқаришнинг алоқаларини улайди. Бунда **.h** кутубхона файллари тимсолини билдиради.
TURBO C да include файлларини айримлари қуйидагилар:
aloc.h – хотирани бошқариш функциясини эълон қилади (бўлиш, жойлаштириш ва ҳоказолар).
asert.h – созловчи макробуйруқ assert ни эълон қилади.
bios.h – IBM-PC ROM BIOS процедурасини чақиришда турли функцияларни эълон қилади.
conio.h – DOS системасида консолга киритиш-чиқаришда турли функцияларни эълон қилади.
ctype.h – макробуйруқларда фойдаланиладиган белгиларни ўзгартириш ва синфларга ажратиш структурасии сақлайди.
dir.h – каталоглар билан ишлаш функцияларини ишлаш процедураларини ўз ичига олади.
float.h – сузувчи вергулли маълумотлар билан операциялар бажарганда процедуралар учун сақлайди.

math.h – математик функциялар тимсоллари кутубхонасини ўз ичига олади.
stddef.h – айрим маълумотларнинг умумий типлари макробуйруқларини ўз ичига олади.
jtdio.h – киритиш чиқариш процедураларини ўз ичига олади.
stdlib.h – алмаштириш процедуралари, излаш процедураси, тартиблаш процедураси ва ҳоказоларни ўз ичига олади.
string.h – белгилар қатори билан ишлаш.
time.h – вақтни белгилаш учун.
graphics.h – графиклар билан ишлайди.

4.9. Функциялар графикларини хосил килиш

Ранглар билан ишлаш учун Си да турли функциялар мавжуд бўлиб, улар қуйидагилардан иборат:
Se1co1or(ранг)-бу функция кўрсатилган чизиш рангини ўрнатади;
GetColor-чизиш рангини қайтаради;
GetBkColor-фон рангини қайтаради;
GetMaxColor- ранглар сонини қайтаради;
Рангларнинг номи ва уларнинг белгиланиши 9-жадвалда келтирилган.
Масалан:
SetBkColor (GREEN)- экранга яшил фон беради.
Жадвал -10 Ранглар ва уларнинг белгиланиши.

Сон кетмакетлиги	Белгиланиши	Ранг номи
0	Black	Қора
1	Blue	Кўк
2	Green	Яшил
3	Cyan	Феруза (барги карам)
4	Red	Қизил
5	Magenta	Қирмизи
6	Brown	Жигарранг
7	LIGHTGRAY	Оч кул ранг
8	DARKGRAY	Тўқ кулранг
9	LIGHTBBLUE	Ҳаворанг
10	LIGHTGREEN	Оч яшил
11	LIGHTCYAN	Оч феруза
12	LIGHTRED	Оч қизил
13	LIGHTMA-CENTA	Оч қирмизи
14	MELLOW	Сариқ
15	WHfTP	Оқ

4.9.1. Нуқта тасвирини ҳосил қилиш

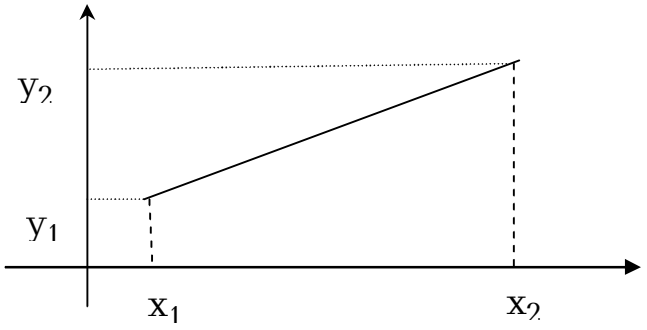
Нуқтани тасвирлаш учун putpixel операторидан фойдаланамиз. Унинг формати қуйидагича:

putpixel(x,y, ранг) -x,y лар нуқтанинг координатлари, ранг – эса унинг ранги.
Масалан: (240,220) координатали нуқтани тасвирини қизил рангда ясайлик .

```
#ihclude<conio.h>
# include<graphic.s.h>
rnain()
{       float x,y;
int gdқDETECT,gm; initgraph(&gd,&gm,"ғғtc");
putpixel(240,220,RED);
getch();
```

4.9.2. Тўғри чизиқ тасвирини ҳосил қилиш

Тўғи чизиқ, тўғрироги кесмани ясаш учун line функциясидан фойдаланилади. Унинг формати қуйдагича :
Iine(xl,yl,x2,y2)



10- расм. Кесма чизиш.

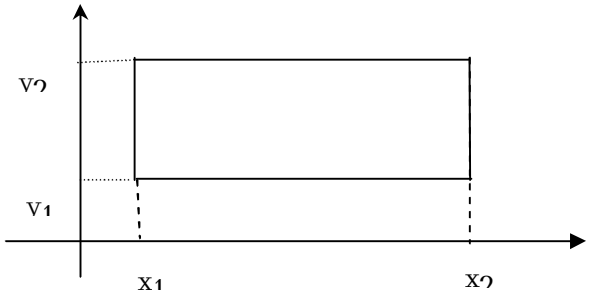
Бу ерда: xl, yl – кесманинг бошланғич нуқтасининг координаталари, x2,y2-кесма охирининг координаталари.

масалан:

line (100,29,80,15) - кўрсатилган (100,29) ва (80,15) нуқталар орасида кесмани-ясайди.

Rectangle(x_1,y_1,x_2,y_2) диоганал бўйича (x_1,y_1) ва (x_2,y_2) координаталарда ётган тўғрибурчакни ясайди. (11-расм).

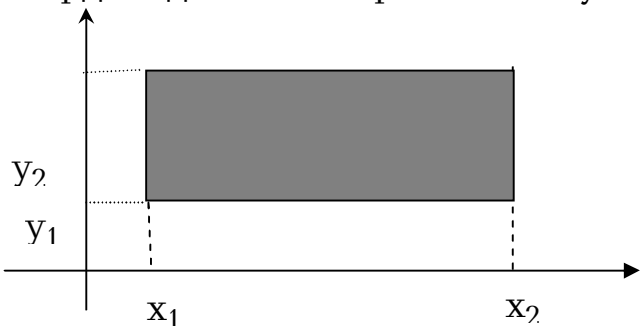
11-расм. Тўғри тўртбурчак чизиш.



bar(xl,yl,x2 ,y2)-учлари диагонал буйича (xl,yl,) ва (x2,y2)-координаталарда бўлган бўялган тўғри тўртбурчакни ясайди.

4.9.3. Айлана ва ёй тасвирини ҳосил қилиш

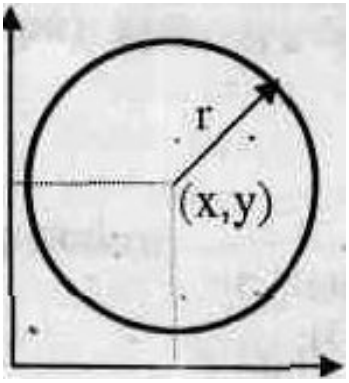
Circle-функцияси ёрдамида айланаларни ясаш мумкин. Унинг



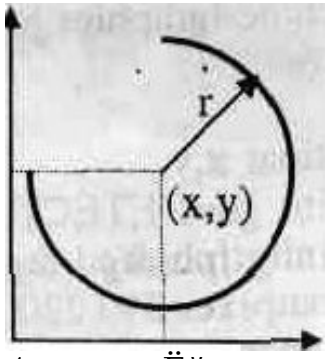
12-расм. Быялган ты\ри тыртбурчак чизиш.

формати:

Circle (x,y,r)-бунда x,y айлана марказининг кооридинаталир, r-радиуси: (13-Расм)



13- расм. Айлана чизиш



14- расм. Ёй чизиш.

Эллипсни ҳосил қилиш учун Ellipse функциясидан фойдаланамиз. Унинг формати.

Ellipse (x,y,α,β,x_r,y_r)-,бунда (x,y) – эллипснинг маркази координаталари, α-бошланғич бурчак, β-охирги бурчак, x_r-абциссалар ўқи бўйлаб радиуси, y_r – ординаталар ўқи бўйлаб радиуси.

Соҳа чегараларини бўяш.

Кўрсатилган соҳа чегарасини бўяш учун **floodfill(x,y, ранг чегараси)** – функциядан фойдаланилади, у **setfillstyle** билан биргаликда ишлайди.

void sefillstyle(int бўяш тури, int ранг)

Бу функция шаблон турини ва рангини танлаш имконини беради. Тўлдиришнинг 12 хил тури бўлиб, у қуйидаги 11-жадвалда келтирилган.

11-жадвал

Исми	Қиймати
EMPTY_FILL	0 (соҳа ранги билан тўлдирилади)
SOLID_FILL	1 (тўлдирувчи ранг билан бутунича бўяш)
LINE_FILL	2 (горизонтал-чизиқлар билан тўлдирилади)
LTSLAHS_FILL	3 (оғма чизиқлар билан тўлдирилади)
SLAHS_FILL	4 (йўғон оғма чизиқлар билан тўлдирилади)
BKSLAHS_FILL	5 (тескари йўғон оғма чизиқлар билан тўлдирилади)
LTBKSLASH_FILL	6 (тескари ингичка оғма чизиқлар билан тўлдирилади)
HATCH_FILL	7 (тўртбурчакли горизонтал штриховкалар билан тўлдирилади)
XHATCH_FILL	8 (эгри штриховка билан тўлдирилади)
INTERLEAVE_FILL	9 (бекилувчи эгри штриховка билан тўлдирилади)
WIDE_DOT_FILL	10(сийрақ нукталар билан тўлдирилади)
CLOSE_DOT_FILL	11(ўзаро якин жойлашган нуқталар билан тўлдирилади)

Си да ҳар-хил йўғонликдаги чизиқларни **selinestyle** функцияси ёрдамида ясаш мумкин.

void selinestyle(int linesigned, uppatern, int thickness);

linestyle(“ чизиқ тўғри”) қуйидаги 5 та кийматдан бирини қабул қилиши мумкин.

Қиймат	Белгили константа	Чизиқ тури
1	SOLID_LINE	Узуликсиз
2	DOTER_LINE	Узуқ-узуқ
3	CENTER_LINE	Марказий
4	DAHSED_LINE	Штрихли
5	USRBIT_LINE	Фойдаланувчи шаблони

Икки аргумент **upatter** (“фойдалувчи шаблони”) биринчи аргумент USERBIT_LINE бўлган ҳолдагина фойдаланилади.
Иккинчи аргумент **thickness** чизиқнинг йўғонлиги ҳакидаги ахборотни билдиради. Афсуски, унинг фақатгина икки кўриниши мавжуд холос.

Қиймат	Белгили константа	Изоҳ
1	NORM_WIDTH	эни 1 нукта
3	THIC_WIDTH	эни 3 нукта

Матнларни жойлаштириш учун **outtext** (қатор) функциясидан фойдаланиб, қаторни экранга чиқариш мумкин.
Масалан:

Outtextxy(x,y,”НаМПИ“)- НаМПИ қатори экранга чикади
Outtextxy(x,y,”матн“)-керакли ахборотни кўрсатилган координатадан бошлаб экранга чикариш.
Масалан:
Outtextxy(50,50,”Ўзбекистон келажаги буюк давлат“)
textwidth(қатор)- қатор узунлигини қайтаради.
textheight(қатор)- қатор бўйини қайтаради.
settextstyle(шрифт, йўналиш, ўлчам)
void far settextstyle(int шрифт,int йўналиш, int ўлчами)
font- ўлчами қуйидаги қийматлардан бирини қабул қилади.

Қиймати	Белгили константа	Йўналиши
0	DEFALT_FONT	Умумий ҳолда
1	TRPLEX_FONT	Триплекс
2	SMALL_FONT	Кичик
3	SANS_SERIF_FONT	Сансериф
4	GOTIC_FONT	Готик

Икки ўзгарувчи direction йўналиши қуйидаги қийматларни қабул қилади.

Қиймат	Белгили константа	Йўналиши
0	HORIZ_DIR	Горизонтал
1	VERT_DIR	Вертикал

Учунчи ўлчам **charsize** белги ўлчамини аниқлайди. 1,2,..,10 қийматларини қабул қилади.

charsize(1≤ i≤10)
void far setjistify (int horiz, int xert)
функцияси ёрдамида матни исталган жойга жойлаштириш мумкин.

Горизонтал, вертикал бўйича қийматлари қуйидаги 11-жадвалда жойлаштирилган.

11-жадвал

Қиймати	Белгили константа	Мўлжалланиши
0	LEFT_TEXT,BOTTOM_TEXT	ўнгдан,куйидан
1	CENTER_TEXT	Ўртасидан
2	RIGHT_TEXT,TOP_TEXT	Ўнгдан,юқоридан

Closegraph()-функцияси график режимидан чикиш учун ишлатилади.

Фойдаланилган адабиётлар рўйхати.

1. Йенсен К., Вирт Н. ПАСКАЛЬ: руководство для пользователя и описание язўка. - М.: Финансў и статистика, 1982.
2. Грогоно П. Программирование на язўке Паскаль. -М.: Мир, 1982.
3. Грэхем Р. Практический курс язўка Паскаль для микроЭВМ. М.: Радио и связь, 1986.
4. Абрамов В., Трифонов Н., Трифонова Г. Введение в язўк Паскаль. М.: Наука, 1988.
5. Файсман А. Профессиональное программирование на Турбо Паскале. Т.: "Инфомэкс Корпорейшн", 1992.
6. Findlay W., Watt D.A., Paskal. An introduction to methodical programmin. Third edition. London: Pitman, 1985.
7. Керниган, Д. Ритчи " Язўк программирования Си" М.: Финансў и статистика. 1992 г.
8. Руководство пользователя по язўку Си " Опсания язўка Си"1,2 книги
9. Д.Маслова " Введение в язўк Си" М, 1991 г.
10. Л. Амераль "Программирование графики на TURBO C" М.: "Сол систем", 1992 г.
11. Вирт Н. Алгоритмў структурў даннўх программў. - М.: Мир, 1985.

МУНДАРИЖА