Цифрлық әдіскер
ақпараттық-білім беру ортасы

ПРАКТИКАЛЫҚ САБАҚТАР

C++ негіздері

Объектіге бағытталған бағдарламалаудың классикалық тілі — синтаксис, функциялар, кластар және алгоритмдік есептер.

- C++ синтаксисі
- Функциялар мен модульдер
- Кластар және объектілер
- Алгоритмдік есептер

ӘДІСТЕМЕЛІК НҰСҚАУ

Тәжірибелік сабақ 1. Бағдарламалау және бағдарламалау тілдерінің даму эволюциясы

Бағдарламалау–бағдарлама құру процесі, алгоритм ұғымымен тығыз байланысты информатика саласының бір бөлігі. Бұл – соңғы кезде өте жылдам дамып келе жатқан пән саласы. Бұрынғы бағдарламалық және техникалық құралдарды дамыту мен жаңа мүмкіндіктердің пайда болу тәжірибесі бағдарламалауды тұрақты жетілдіріп келеді, осының нәтижесінде жаңа тәсілдер мен технологиялар шығып, олар қазіргі бағдарламалық жабдықтамаларды жасауға жаңа негіздеме болып табылады. Жаңа технологиялар жасау процесін зерттеу және олардың негізгі даму бағытын анықтау ісін жүргізуді қазіргі технологияларды бағдарламалау деңгейімен салыстыра отырып, қолда бар бағдарламалық және аппараттық құралдар ерекшеліктерін қолдану арқылы жетілдіру қажет. Бағдарламалау технологиясы деп бағдарламалық жабдықтамаларды жасау процесінде пайдаланылатын құралдар мен тәсілдер жиынын айтады. Кез келген басқа технологиялар сияқты бағдарламалау технологиясы да төмендегі мүмкіндіктерді қамтитын технологиялық нұсқаулардан тұрады:

- технологиялық операцияларды орындау тізбегін көрсету;
- кез келген операцияда орындала алатын шарттарды нақтылау;
- әрбір операцияны олардың әрқайсысы үшін анықталған бастапқы мәліметтер, нәтижелер және де нұсқаулар, нормативтер, стандарттар, бағалау критерийлері мен әдістері, т.б. арқылы сипаттау. Технология, операциялар мен олардың тізбегінен бөлек, жобаланатын жүйенің сипаттамасын, яғни оның жасалатын нақты кезеңінде пайдаланылатын моделін сипаттау тәсілін анықтайды. Технологиялар көбінесе мынадай екі түрге бөлініп қарастырылады:
- бағдарлама жасау процесінің нақты кезеңдерінде пайдаланылатын технология немесе осы кезеңдердегі жеке есептерді шығару;
- бірнеше кезеңдерді біріктіретін немесе барлық кезеңдерді толық қамтитын технология. Алғашқы технология негізінде көбінесе нақты есепті шығара алатын шектеулі түрде қолданылатын тәсіл жатады. Ал екіншісінің негізі ретінде бағдарлама жасаудың әртүрлі кезеңдерінде пайдаланылатын тәсілдер жиынын анықтайтын базалық тәсіл, яғни ортақ көзқарас (парадигма) немесе әдіснама (методология) қарастырылады. Бағдарламалау технологиясының дамуы компьютерлік техниканың дамуымен тығыз байланысты. Бірінші ЭЕМ ENIAC 1946 жылы жасалды. ЭВМ-нің өнімділігі секундына 500 мың амалдар көрсетіп, 100 шаршы метрлік ғимаратта орналасқан. Ол АҚШ ӘТК снарядтың ұшу бағытын анықтауға арналған болды. Кейін ЭЕМ қуатын арттырып, көлемін кішірейту жолмен компьютерлік техникалар дами бастады. ЭЕМ даму эволюциясы микроэлектрониканың дамуымен тығыз байланысты. Суретте ЭЕМ даму эволюциясы электронды компоненттердің дамуы бойынша көрсетілген.

Лампалар 40-50ж. Транзисторлар 50-60ж. ИС 60-70ж.

БИС, СБИС 70-80Ж.

ПЭЕМ 1982Ж.

ЭЕМ-ның бір буынынан басқа буынға асуы тек элементтік қорды жетілдірумен сипатталып қана қоймайды және ЭЕМ-нің құрылымының өзгеруімен, олардың фукнционалды-техникалық мүмкіндіктерінің және тұтынымдық сипаттамаларының өсуімен бейнеленеді. Бірінші буынның ЭЕМ-дері электронды лампалардан құрылды. Сыртқы есте сақтау құрылғылары ретінде магнитті барабандар қолданылды. Олардың өнімділігі мен сенімділігі төмен болды. Бірінші ЭЕМ-дер тек әр ЭЕМ-ге (тілдердің 1-ші буыны) тән машиналық командалар тілін түсінді. Барлық командалар цифрлармен берілді және оларды мүмкін емес еді.

Тәжірибелік сабақ 2. Бағдарламалау сапасын қамтамасыз ететін негізгі принциптер.

Бағдарламалау процесін жақсартатын және кең қолданылатын әдістердің бірі – құрылымдық бағдарламалау.

Құрылымдық бағдарламалау Өткен ғасырдың 60 жылдары – бағдарламалаудың «стихиялық» даму кезеңінде бағдарлама құрылымы, мәліметтер типтері сияқты ұғымдар болмады. Сол себепті бағдарламалар қайшылықтары көп, түсінуге қиын кодтардан тұратын еді. Ол кездегі бағдарламалау өзінше бір өнер тәрізді болып көрінетін. Бағдарламалаудағы сондай қиындықтан шығу үшін бағдарламалаудың құрылымдық парадигмасына көшу керек болды. Құрылымдық бағдарламалау есепті шағын құрылымдарға бөліп, олардың әрқайсысын жеке қарастыруды қажет етеді. Мұнда жобалау «жоғарыдан-төменге» қарай жүргізілді де, жалпы құрылым идеясын жүзеге асырып, ішкі интерфейс бағдарламаларын орындауды талап етті. Бұған қоса, алгоритм конструкцияларына шектеулер қойылып, оларды формальды модель түрінде сипаттау ұсынылды және алгоритм құрудың арнайы тәсілі – қадамдық түрде орындау тәсілі қолданылды. Құрылымдық бағдарламалау қағидасын сүйемелдеу процедуралық бағдарламалау тілдерінің негізіне жатқызылды. Олар көбінесе негізгі басқаруды беру «құрылымдық» операторларын қолдану, ішкі бағдарламаларды пайдалану және мәліметтердің «көріну» аймағын шектеу тәрізді мүмкіндіктерді сүйемелдеді. Бұл топтағы кең таралған тілдер ретінде PL/1, ALGOL-68, Pascal, C тілдерін атауға болады. Бағдарламалық жабдықтамаларды ары қарай көлемінің өсуі, күрделілігінің артуы мәліметтерді құрылымдауды да дамытуды талап етті. Осының нәтижесінде қолданушының мәліметтер типін анықтау мүмкіндігі пайда болды. Оған қоса, бағдарламаның ауқымды (глобальді) мәліметтерін пайдалануға шек қойып, олармен жұмыс істеу кезінде шығатын қателерді азайту мақсатында жұмыстар істеле бастады. Сондықтан келе-келе жаңа – технология модульдік бағдарламалау технологиясы жетіле бастады.

```
Құрылымдық бағдарламалаудың 3 бөлігі (құраушысы) бар:
```

1. Модульдік бағдарламалау
2. Құрылымдық кодтау
3. Жоғарыдан төменге қарай жобалау.

Модульдік бағдарламалау. Уақыт өте келе бағдарламаларға деген көзқарас өзгеріп, бұрынғыдай процедуралар құрудан алдын мәліметтерді ұйымдастыру жағына көңіл бөліне бастады. Бұған басты себеп болған жайттардың бірі бағдарламалар көлемінің ұлғаюы болды. Бір-бірімен өзара байланысқан процедуралар мен солар арқылы өңделетін мәліметтер жиынын көбінесе модуль деп атайтын еді. Модульдік бағдарламалау ауқымды айнымалылардың белгілі бір тобын қолданатын ішкі бағдарламалар жиынын жеке копиляцияланатын модульдерге (ішкі бағдарламалар кітапханасы) бөліп қарастыру тәсілі болып табылады. Бұл технологияда модульдер арасындағы байланыс арнайы интерфейс арқылы жүзеге асырылады да, сол модульдердің өздерін пайдалануға (ішкі бағдарламалар командаларын және олардың «ішкі» айнымалыларын) тиым салынады. Бұл технологияны Pascal, C/C++, Ада, Modula тілдерінің соңғы нұсқалары сүйемелдейді. Құрылымдық кодтау деп бағдарламада басқарушы конструкциялардың - шартты операторлардың, циклдің (параметрлі, цикл - әзір, цикл - дейін) қолданылуын айтады. Шартсыз көшу операторы бағдарламада сирек қолданылуы керек немесе шартты операторлардың, циклдің көмегімен өзгертілуі керек. Бағдарламаны жоғарыдан төмен қарай жобалаудың өз иерархиялық құрылымы бар және қысқа есеп қойылымынан басталады. Одан кейін есеп бірнеше ұсақ ішкі есептерге бөлінеді. Ішкі есептердің өзі де ішкі есептерге бөлінуі мүмкін. Әр қадамда ішкі есептің орындайтын негізгі функциялары анықталуы керек. Бөлу процесі әр ішкі есеп қарапайым болғанға дейін, яғни әр ішкі есепке бір модуль сәйкес келгенше созылады.

Объектіге бағытталған бағдарламалау Дәстүрлі процедураға бағытталған бағдарламалау тәсілдері күрделі бағдарламалардың талаптарын қанағаттандыра алмады. Сондықтан объектіге бағытталған бағдарламалау ұғымы пайда болды.

```
Объектіге бағытталған бағдарламалау (ОБП) – бұл жаңа технология.
```

Мұнда құрылымдық бағдарламалаудың ең тиімді жақтары алынып, олар жаңа идеялармен толықтырылады. Объектіге бағытталған тіл деп бағдарламалары өзара байланысқан объектілер тізімінен және солардың жұмысын сипаттаудан тұратын бағдарламалау тілін айтады. ОБП ерекшеліктері: ? бағдарламалық жабдықтамалардың күрделілік дәрежесін төмендету; ? бағдарламалық жабдықтамалардың сенімділігін жоғарылату; ? бағдарламалық жабдықтамалардың кейбір компоненттерін қайтадан қолдану мүмкіндігін қамтамасыз ету. ОБП негізгі ұғымдарының бірі - объект. Объект – мәліметтердің логикалық бірлігі ретінде бағдарламада құрастырылған жаңа типтегі айнымалы. Объектінің атқаратын басты міндеті – мәліметтерді және оларды өңдеуге арналған тәсілдердің (ережелердің) бірігуі. C++ тілінде мәліметтерді өңдеу тәсілдер, яғни функциялар арқылы атқарылады. Сондықтан C++ бағдарламалау тілінде объект мәліметтерден және соларға қолданылатын функциялардан тұрады.

Тәжірибелік сабақ 3. Бағдарламалау тілдерінің құрылымды даму концепциясы.

Өткен ғасырдың 50-ші жылдары компьютерлер қуаттылығы (ЭЕМ-дердің алғашқы буыны) жоғары деңгейде болған жоқ, олар бағдарламалау машиналық кодтар арқылы жүргізілді. Негізінен, ғылыми-техникалық есептер (формулалар бойынша) шығарылды, бағдарламалауға берілген тапсырмалар нақты есептер қойылымынан тұратын еді. Көбінесе бағдарламалаудың интуитивтік технологиясы қолданылатын, мұнда бірден тапсырма бойынша бағдарлама құруға кірісетін, оның үстіне жұмыс барысында тапсырмалар өзгертіліп те отыратын еді, қажетті құжаттама бағдарлама жұмыс істеген кезде барып құрастырылатын. Соған қарамастан, бағдарламалау технологиясының іргелі тұжырымдамасы болып саналатын модульдік бағдарламалау осы кезеңде (машиналық кодта бағдарламалаудың қиындығын жеңу мақсатында) пайда болды. Алғашқы жоғары деңгейде бағдарламалау тілдері пайда бола бастады, солардың ішіндегі ФОРТРАН тілі одан кейін де ұзақ уақыт пайдаланылып келеді. 1960-шы жылдары бірсыпыра жоғары деңгейде бағдарламалау тілдерінің (АЛГОЛ 60, ФОРТРАН, КОБОЛ, т.б..) өте жылдам дамығанын көреміз, сол кездерде бағдарламалау технологиясындағы олардың орны өте жоғары бағаланылып жүрді. Бірақ осы тілдер көлемді бағдарламалар жазуда барлық мәселелерді шешеді деген үміт ақталмады. Компьютерлердің қуаты артып, бағдарламалау тәжірибелері де жинақталған кездерде жоғары деңгейде бағдарламалау тілдерінің мүмкіндіктері шектеулі екендігі белгілі болды. Осының нәтижесінде бағдарламаларды модульдік түрде ұйымдастыру жүзеге асырыла бастады. Тек ФОРТРАН тілі ғана пайдаланыла берді, өйткені ол бұрыннан-ақ модульдік қағидамен жасалған еді және дайындалған программалар кітапханасы да өте бай болатын. Осы кездерде бағдарламаны қай тілде құру емес, қалай құру керек деген сауалдың маңызды екені түсінікті болды. Мамандар енді бағдарламалау технологиялары мен олардың әдіснамаларына көңіл бөле бастады. Екінші буын компьютерлеріндегі жұмысты уақытша тоқтатып, бағдарламаларды үзе тұру мүмкіндігінің пайда болуы мульти бағдарламалау ісінің жетілдіріліп, үлкен бағдарламалық жүйелер жасауға себепші болды. Бұлардың бәрі бағдарламаларды ұжым болып жасаудың әсерінен туындады да, көптеген жаңа технологиялық мәселелер пайда болды. Қазіргі кезде компьютердің көмегімен әр түрлі есептерді шығаруға мүмкіндік беретін сан алуан бағдарламалау тілдері бар. Жоғарғы деңгейлі бағдарламалау тілдерінің бірі – Си тілі. Си бағдарламалау тілі 1972 жылы Bell Laboratories фирмасының қызметкері Денис Ритчидің басшылығымен Америкада жасап шығарылды. Си тілі – құрылымы Паскаль тіліне ұқсас жоғарғы деңгейлі қуатты тіл.

Жоғары деңгейлі тілде типтік бағдарламаның құрылымы.

Жоғары деңгейдегі бағдарламалау тілдері пайда болысымен бағдарламаларды әзірлеу шапшаңдығы айтарлықтай артты, бағдарламаны құруға кететін уақыттың 80% құрайтын ретке келтіру процесін автоматтандыру мүмкіндігі пайда болды. Үшінші буындағы тілдерінде қосу командаларын жазу мысалын қарастырайық. Basic тілінде қосу командалары жазбасының мысалы шамамен мынадай:

```
Y=a+b
```

Pascal тілінде қосу командалары жазбасының мысалы шамамен мынадай:

```
Y:=a+b;
```

C++ тілінде қосу командалары жазбасының мысалы шамамен мынадай:

```
Y=a+b;
```

Мысалдарда көрініп тұрғандай командалар жазбасы адамға үйреншікті жазбаға максималды ұқсастырылған. Нақты бағдарламалау тілін таңдау шешілетін тапсырманың класына тікелей тәуелді. Жүйелік бағдарламалау тапсырмалары көп еңбекті талап ететін ретке келтіруге қарамастан дәстүр бойынша C++ тілінде шешіледі. Қолданбалы сипаттағы тапсырмалар PASCAL және BASIC тілдерінде шешіледі. Төртінші буынның ЭЕМ-дерінің басты ерекшелігі– ҮИС (БИС) мен АҮИС (СБИС) кең қолданылуларында. Бұл ЭЕМ-нің техникалық-экономикалық көрсеткіштерінің бірден өсуіне мүмкіндік берді. Микропроцессордың (МП) пайда болуымен есептеуіш техникасының жаңа дәуірі басталды. Бір кристалда орындалған МП сыртқы құрылғыларынан басқа ЭЕМ-ң барлық элементтерін қамтыды. МП негізінде құрылған ЭЕМ-н микро-ЭЕМ деп атайды. Олар өте кішкентай, жұмыс столында көп орын алмайды, пайдалануда еш қиындық тудырмайды және арнайы шарттар қажет етпейді. Төртінші буынға бағдарламалық қамтамасыздандыруды басқару мәселелік-бағытталған тілдері жатады. Бұл мәселелік-бағытталған тілдер Visual Basic, Delphi, C++, Java. Осы тілдердің артықшылығы компьютерлік желілерде, Интернетте жұмыс істеуге арналған қосымшаларды құру мүмкіндігі және бағдарламалаудың түрлі элементтерімен объектретінде жұмыс істеу мүмкіншілігі болып табылады. Бағдарламалаудың OLE, ActiveX, COM, DCOM және басқалар сияқты жаңа технологиялары пайда болды. Сонымен, машиналық тілдердің

өркениеті шынайы тілдер бағытында жүріп жатыр. Идеалды шешім– қолданушы өз мәселесін шынайы тілде құрастырады, ал өзгесін ЭЕМ орындайды. Суретте бағдарламалау тілдерінің өркениеті көрсетілген. Машиналық кодтар Ассемблер Жоғары деңгейдегі тілдер ОББ тілдері

1982 жылы IBM фирмасы дербес компьютерлердің (ДК) бірінші партиясын шығарды, олардың пайдаланушылық мүмкіншіліктері мен құрылыстық-технологиялық шешімдеріне қарап, бесінші буын ЭЕМ-на жатқызуға болады.

Тәжірибелік сабақ 4-5. Алгоритмдер және олардың типтері

Алгоритмдеу – есепті шығару алгоритмін құрастыру процесі, мұның нәтижесінде мәліметтерді өңдеу процесінің кезеңдері айқындалады да, кезеңдер мазмұны формальды (жасанды) түрде жазылып, солардың орындалу реттілігі анықталады. Алгоритм – бастапқы айнымалы түрде берілген мәліметтерден қажетті нәтижеге қол жеткізу жолында атқарылатын есептеу процесін анықтайтын дәлме-дәл нұсқаулар жиыны. Алгоритм қасиеттері:

1. детерминділік (анықтылық, бір мәнділік) – басқаша түсінуге жол бермей, тек қана көрсетілген әрекеттерді айқын түрде орындауға арналған нұсқаулар дәлдігі;
2. дискреттілік – есептеу процесін жекеленген қарапайым операцияларға бөлу қасиетінің болуы, яғни күрделі есепті атқарылуына күдік келтіруге болмайтын шағын бөліктерге жіктеу мүмкіндігінің болуы;
3. нәтижелілік – белгілі бір әрекеттер саны атқарылған соң, процестің қажетті нәтижесін алып, оны аяқтау мүмкіндігінің болуы немесе есептеу процесін ары қарай жалғастыруға болмайтындығы жайлы мәлімет алу;
4. жалпылық – алгоритмнің осы сияқты көптеген басқа да есептерге қолданылу мүмкіндігінің болуы Алгоритмдік тіл – алгоритмдерді жазуға арналған символдар мен сол символдардан тұратын конструкцияларды құрастыру және түсіндіру ережелерінің жиыны. Бағдарламалау тілі компьютерлерде бағдарламаларды орындау ісін атқарады.

БАҒДАРЛАМА – МАШИНАҒА ТҮСІНІКТІ ТҮРДЕ ЖАЗЫЛҒАН АЛГОРИТМ. БАҒДАРЛАМАДА БЕРІЛГЕН МӘЛІМЕТТЕРДІҢ СИПАТТАМАЛАРЫМЕН БІРГЕ ОЛАРДЫ ӨҢДЕЙТІН КОМАНДАЛАР БОЛАДЫ. КОМАНДАЛАР ҚАНДАЙ МӘЛІМЕТТЕР ҚАНДАЙ ОПЕРАЦИЯЛАРҒА ҚАТЫНАСАТЫНЫН, ОЛАР ҚАНДАЙ РЕТТІЛІКПЕН ОРЫНДАЛАТЫНЫН ЖӘНЕ НӘТИЖЕНІҢ ҚАНДАЙ ТҮРДЕ ШЫҒАРЫЛАТЫНЫН КӨРСЕТЕДІ. БҰЛАР ОПЕРАТОРЛАР АРҚЫЛЫ ЖҮЗЕГЕ АСЫРЫЛАДЫ.

Мәліметтер – белгілі бір процесс көмегімен тасымалдап, өңдеуге болатын, формальды түрде бейнеленген фактілер мен идеялар. Айнымалы – бағдарлама орындалуы барысында өз мәнін өзгерте алатын шама. Оператор – операциялар мен мәндерді көрсететін немесе солардың элементтерінің қай жерде орналасқанын білдіретін символдар жиыны. Мысалы:

```
a = b + c;  // a, b, c – айнымалылар;}
k = 2;
if (t<0) ...
```

Бұл пәннің заттық негізі болып (компьютерде шығару мақсатында) бағдарламаларды құрастыру тәсілдері мен құралдары саналады. Бағдарламалар құру үшін бағдарламалау жүйелері пайдаланылады. Бағдарламалау жүйесі – бағдарламалауды автоматтандыру құралдары. Олар бағдарламалау тілінен, осы тілдің трансляторынан, құжаттамаларынан және де бағдарламаларды дайындау, әрі орындау құралдарынан тұрады. Транслятор – бір тілді екінші тілге аудару бағдарламасы. Ол интерпретатор және компилятор сияқты екі топқа бөлінеді. Интерпретатор – бұл командаларды аударып, оларды бірден орындауға арналған трансляторлық бағдарлама. Компилятор – бұл алгоритмдік тілдің конструкцияларын толығымен машиналық кодқа түрлендіретін бағдарлама. Есептің нәтижесін алу түшін машиналық кодты орындау керек.

Компьютерде есеп шығару кезеңдері Компьютерде есеп шығару күрделі процесс болып есептеледі, ол төмендегі кезеңдерден тұрады:

1. Берілген есепті математикалық түрде өрнектеу, яғни есепті мәселе ретінде қоя білу.
2. Есепті шығарудың компьютерге ыңғайлы сандық тәсілдерін анықтау.
3. Есепті шығару жолын алгоритм түрінде бейнелеу.
4. Есепті компьютерде шығару бағдарламасын жасап, оның қателерін түзету.

5. Есепке керекті мәліметтер дайындау.
6. Компьютерде есепті шығару және шыққан нәтижені іс жүзінде қолдану. Берілген есепті математикалық түрде өрнектеу дегеніміз – есептің берілген мәндерін математикалық таңбаларды қолданып жаза білу және керекті математикалық формулаларды анықтау болып саналады. Күрделі формулаларды, теңдеулерді арифметикалық амалдар тізбегіне айналдыру есепті шығарудың сандық тәсілдерін табу не анықтау жолы болып есептеледі. Қазіргі кезде барлық есептердің шығару жолының сандық тәсілдері белгілі десе де болады, тек солардың ішінен өзімізге тиімді жолын таңдап алуымыз керек. Бұл мақсатта есепті шығару дәлдігін, нәтижені жылдам табу мүмкіндігін, мәліметтерді дайындау мен есепті шығарудың бағасын салыстыра отырып қарастыру қажет. Есептің алгоритмін жасағанда, оның шығару жолын тізбектелген іс-әрекеттер ретінде схема түрінде өрнектеледі.

БАҒДАРЛАМА ЖАСАҒАНДА ҚАЗІРДЕ КЕҢІНЕН ТАРАҒАН БАҒДАРЛАМАЛАУ ТІЛІНІҢ БІРІНДЕ АЛГОРИТМ НАҚТЫ ТҮРДЕ ЖАЗЫЛАДЫ. БІЗДЕ КЕҢ ТАРАҒАН ТІЛДЕРГЕ – ПАСКАЛЬ, ДЕЛЬФИ, C/C++ ЖАТАДЫ. ЖАЗЫЛҒАН БАҒДАРЛАМАНЫҢ ҚАТЕСІН ТҮЗЕТУ КОМПЬЮТЕРДІҢ КӨМЕГІМЕН ШЕШІЛЕДІ, ӨЙТКЕНІ ЖІБЕРІЛГЕН ҚАТЕЛЕРДІ ТЕК КОМПЬЮТЕР ҒАНА ЖЫЛДАМ АҢҒАРЫП, ТҮЗЕТУ МҮМКІНДІГІН БЕРЕДІ.

Есепті шығаруға керекті деректерді сұрыпталған күйінде алдын ала қағазға, әйтпесе магниттік дискіге жазып, компьютердің жадына реттей отырып енгіземіз. Есептің нәтижесін алған соң шешім қабылдау және оны іс жүзінде қолдану – мамандардың жұмысы. Тек солар ғана белгілі бір шешім қабылдай алады. Бірақ оқып-үйрену барысында кездесетін, яғни студенттерге арналған есептерде жоғарыда көрсетілген сатылардың бірсыпырасы болмайды, өйткені олар бірден формула күйінде беріледі, шығарудың сандық тәсілі формулада айқын көрініп тұрады (интеграл, туынды болмаса), нәтижені алған соң, оны жазып алу жеткілікті. Мәселені шешудің немесе есеп шығарудың көрсетілген алты сатысы күрделі өндірістік есептерде, дипломдық немесе курстық жұмыстарда жиі кездеседі.

Алгоритмді жазу тәсілдері. Алгоритмдерді бейнелеудің негізгі тәсілдеріне оларды жазудың келесідей түрлері жатады: ? табиғи тіл сөздері арқылы; ? формулалық-сөздік тәсіл арқылы; ? графикалық түрде бейнелейтін блок-схемалар арқылы;

```
? псевдокодтар (жалған кодтар) арқылы;
```

? бағдарламалау тілі арқылы. Алгоритмдерді табиғи тіл сөздері арқылы бейнелеуде – есептеу кезеңдері мазмұны кез келген түрде табиғи тілде жазылады. Осы тәсілмен келесі мысалдың алгоритмін жазып шығайық. 1-мысал. Сандар жиымы (массиві) берілген делік. Осы жиым сандарының көрсетілген аралықта, яғни интервалда толығынан жататынын немесе жатпайтынын тексеру керек. Интервал өзінің шекаралық А және В мәндерімен берілген. 1-кесте.

Алгоритмді графикалық түрде блок-схемалар арқылы көрсету – оның логикалық құрылымын графикалық түрде бейнелеу болып саналады. Мұнда мәліметтерді өңдеудің әрбір кезеңі атқарылатын операцияға сәйкес әр түрлі геометриялық фигуралар (блоктар) түрінде көрсетіледі. Сонымен алгоритм блоктармен немесе геометриялық көпбұрыштар түріндегі фигуралармен өрнектеледі. Әр блоктың ішіне орындалатын іс-әрекеттің (амалдың) мазмұны жазылады. Символдардың (блоктардың) бір кіру және біршығу сызықтары болуға тиіс.

2-кесте.

Жалған кодтар (псевдокодтар) – нақты бағдарламалау тілінің синтаксистік ерекшеліктерін есепке алмай, тек формальды түрде бағдарлама логикасын бейнелеуге мүмкіндік береді. Бұл тәсіл бағдарламалау тілінің операторлары мен табиғи тіл сөздерінің араласуы арқылы құрастырылып, блок-схема орнына пайдалануға болатын бағдарлама логикасын бейнелеу құралы болып табылады. мысал. Z = ax 2+bx+cos(ax2+b)-tg(ax2+b) функциясының мәнін есептеп шығару алгоритмін жасау керек болсын делік. Мұнда алдымен a, b, x нақты мәндерін енгізіп, жақшада тұрған ax2+b өрнегін есептеп алу керек, соңынан Z функциясының мәні нәтиже ретінде шығарылуы тиіс. алг Z функциясын есептеу нақ a, b, x, t, z арг a, b, x нәт z басы a, b, x енгізу t := ax 2 + b z := t+cost-tgt x, z шығару соңы

Бағдарламалау тілдері – бағдарламаларды компьютерде тікелей орындауға арналған алгоритмдерді жазу тәсілі. Бағдарлама – алгоритмнің компьютерге түсінікті түрде жазылуы. Әрбір машинаның өз тілі (машиналық тіл) болады және ол тек осы тілдегі бағдарламаларды, яғни командалар тізбегін орындай алады. Бағдарламаларды машиналық тілде жазу өте күрделі, әрі адамды шаршататын жұмыс болып табылады. Бағдарламалаушылардың жұмыс өнімділігін арттыру мақсатында жасанды тілдер, яғни бағдарламалау тілдері қолданылады. Мұндайда жасанды тілде жазылған бағдарлама

машиналық тілге аударылуы тиіс. Аудару жұмысын, яғни бағдарламаны бір тілден екінші тілге түрлендіру ісін транслятор атқарады. Жиі қолданылатын, тікелей интерпретация жасайтын транслятор түріне Бейсик тілінің трансляторы жатады, ол командаларды аударады да, оны бірден орындайды.

```
Бірінші элементті таңдаймыз ( i := 1)
```

IF A > x i немесе x i > B THEN хабарлама шығару және алгоритм соңына көшу ELSE

```
келесі элементке көшу (i := i +1)
IF жиым біткен жоқ (i ? N)  THEN
```

интервалды тексеруге көшу ELSE интервалға элементтер толық кіретіні жайлы хабарлама шығару Соңы жұмысының қорытындысы қажетті нәтижелерді алу болып табылады. Паскаль, C/C++ тілдерінің трансляторы – компилятор түрінде болады. Мұнда бастапқы жазылған бағдарлама мәтіні машина тіліне аударылады да, объектілік модуль деп аталатын бағдарлама коды шығарылады. Сонан соң объектілік модуль бағдарлама аралық байланыс редакторы деп аталатын бағдарлама арқылы өңделгеннен кейін барып қана жұмыс істеуге дайын болады.

Қарапайым алгоритмдер құру Күрделі алгоритмдерді құру үшін қарапайым канондық (бірыңғайланған) алгоритмдік құрылымдар қолданылады. Олар сызықтық, тармақталу және цикл құрылымдарынан тұрады. Бағдарламалау теориясында кез келген күрделі бағдарламаны осы үш түрлі құрылымнан құрастыруға болатыны дәлелденген. Сызықтық құрылым бірінен кейін бірі орындалып тізбектеле орналасқан бірнеше операторлардан тұрады. Тармақты – шартқа байланысты екі оператордың бірінің орындалуы. Цикл – операторлар бөлігінің бірнеше рет қайталана орындалуы. Негізгі конструкцияларды пайдалану мақсаты–қарапайым құрылымды бағдарлама алу болып саналады. Мұндай бағдарламалар оңай оқылады, түзетіледі және керек болса, оңай өзгертіледі.

1-сурет. Блок-схема түріндегі алгоритм

Алгоритмді жоғары деңгейлі тілде іске асыру

1. Сызықтық құрылымды алгоритм немесе қарапайым сызықтық алгоритм іс-әрекеттердің орындалу ретіне қарай тізбектеле орналасқан блоктардан тұрады. Амалдардың бұлай бірінен соң бірі реттеліп орындалу тәртібін табиғи атқарылу дейді. Мысал. y = a+b формуласы бойынша y -ті табу керек болсын. Мұнда формула бойынша есептеу тіктөртбұрыш арқылы кескінделетін есептеу блогы (3-блок) арқылы өрнектеліп, нәтижені қағазға басу үшін көпбұрышты құжат алу блогын (4-блок) пайдаланып, оның ішіне нәтиженің атауларын жазамыз. Осы көрсетілген y=a+b формуласын есептеу үшін a және b-ның сандық мәндерін бағдарламаға енгізіп (2-блок), содан кейін қосу амалын орындап, ақырында y-ті экранға (қағазға) басып шығарып, жұмысты тоқтатамыз. Осы алгоритмнің схемасы 2-суретте көрсетілген, ал оның жанында Паскаль және C тіліндегі бағдарламасы жазылған. Бағдарлама мәтінін құру жолдарын келесі тарауларда қарастырамыз.
2. Тармақталу алгоритмдері. Тұрмыста кездесетін алгоритмдер әр түрлі болып келеді. Олардың жиі кездесетін түріне алгоритмнің белгілі бір шарттың орындалуына не орындалмауына байланысты тармақталып бірнеше жолдарға бөлінуі жатады.

2-сурет. Бағдарламалау тілінде жазылуымен алгоритмі.

1. Циклдік алгоритмдер. Математикада, экономикада көптеген есептерді шығару кезеңінде бір теңдеуді пайдаланып, ондағы айнымалының өзгеруіне байланысты оны бірнеше рет қайталап есептеуге тура келетін сәттер де жиі кездеседі. Осындай қайталап орындалатын есептеу процесінің белгілі бір бөліктерін цикл деп атайды. Осы бірнеше рет қайталанатын бөлігі бар алгоритмдер тобы циклдік алгоритмдерге жатады. Циклдік алгоритмдерді пайдалану оларды кейіннен бағдарламаларда цикл операторы түрінде қысқартып жазу мүмкіндігін береді. Циклдер қайталану санының алдын ала белгілі және белгісіз болуына байланысты екі топқа бөлінеді. Қайталану сандары алдын ала белгілі болып келген циклдер тобы арифметикалық цикл болып есептеледі, ал орындалу саны белгісіз циклдер – қадамдық (итерация) цикл болып аталады.
2. Қадамдық циклдер. Циклді орындаудың алдында, оның қайталану саны белгісіз болған жағдайда қадамдық циклдер пайдаланылады. Мұнда циклді жазу үшін тек "шартты тексеру" блогын қолдану қажет, ол циклді аяқтау үшін белгілі бір шартты тексереді. Қадамдық циклдердің схемасын сызғанда модификаторды (алтыбұрышты) қолдана алмаймыз, себебі алдынала циклдің неше рет қайталанатыны белгісіз.

3-сурет. Циклға арналған блок-схема түріндегі алгоритм

Тәжірибелік сабақ 6-7. Деректер құрылымы мен типтері. Жиымдар, тізімдер, ағаштар, стектер, кезектер, файлдар.

Кез келген бағдарламаның негізгі мақсаты мәліметтерді өңдеу болып табылады. Әр түрлі типтегі мәліметтер компьютер жадында басқаша сақталып, олардың өңделуінде де айырмашылықтар болады. Кез келген алгоритмдік тілде әрбір константа, айнымалы, өрнекті немесе функцияны есептеу нәтижесі белгілі бір типте болуы тиіс. Мәліметтер типі мыналарды:

```
–   компьютер жадындағы мәліметтің ішкі бейнелену түрін (көлемін);
```

- белгілі бір типтегі шамалардың қабылдай алатын мәндер жиынын;
- осы типтегі шамаларға қолдануға болатын операциялар мен функцияларды анықтайды. Осыларға байланысты бағдарламада пайдаланылатын нақты объектілерді бейнелеу үшін бағдарламалаушы әрбір шамалардың типін алдын ала таңдап алады. Типті міндетті түрде сипаттау қажеттілігі компиляторға бағдарламадағы әр түрлі конструкцияларды пайдалануға болатындығын тексеру мүмкіндігін береді. Мәліметтерді өңдеу үшін пайдаланылатын машиналық командалар шамалардың типіне байланысты болып келеді. С++ тілінің барлық типтері негізгі және құрама болып екіге бөлінеді. Мұнда бүтін, нақты, символдық және логикалық шамаларды бейнелеу үшін алты негізгі тип қолданылады. Бағдарламалаушы осы типтерді негізге ала отырып, құрама типтерді сипаттай алады. Құрама типтерге жиымдар (массивтер), тізбелер (пере числения), функциялар, құрылымдар (структуралар), сілтемелер (ссылки), нұсқауыштар (указатели), біріктірмелер (объединения) және кластар жатады.

Си тілінің типтерінің құрамы

С тілінде мәліметтердің бірнеше негізгі типтері қолданылады. Олар:

```
    •    char (8 бит) – символдық, яғни таңбалық тип,
```

- int – бүтін сан типі,
- float – нақты сан типі, яғни жылжымалы нүктелі сандар,
- double – екі еселенген нақты сан типі. Алғашқы екі тип бүтін сандарды сипаттайтын негізгі (стандартты) тип, ал соңғы екеуі – жылжымалы нүктелі типтер болып табылады. Компилятордың бүтін шамаларды өңдеу үшін жасайтын кодтары жылжымалы нүктелі сандарды өңдеу кодтарынан басқашалау болады. Төмендегі кестеде әр түрлі типтердің ұзындықтары көрсетілген. Ал С++ тіліне жоғарыдағыларға қосымша тағы екі тип:
- wchar_t – кеңейтілген символдық тип,
- bool – логикалық тип енгізілді. Стандартты сандық типтердің мәндерін бейнелеу диапазонын анықтауда төрт тип спецификаторы қолданылады, олар:
- short (қысқартылған);
- long (ұзартылған);
- signed (таңбалы);
- unsigned (таңбасыз).

Негізгі бағдарламалау құрылымының түсінігі мен оларды іске асыру тәсілдері. Int бүтін сандар типі Int типін стандарт бекітпеген, ол компьютерге немесе компиляторға байланысты өзгеріп отырады. 16-разрядты процессорде ол 2 байт, ал 32-разрядтысында – 4 байт. Егер int алдында short спецификатор сөзі тұрса, онда ол әрқашан 2 байт, ал егер спецификаторы long болса, 4 байт болады. Санға компьютер жадында берілген орынға қарай олардың мәндері өзгереді. short int – 2 байт, оның диапазоны –32768 ..+32767; long int – 4 байт, оның диапазоны –2 147 483 648..+2 147 483 647. Int типі 16-разрядты компьютер үшін short int типімен бірдей, ал 32-разрядты компьютер үшін long int типімен бірдей. Signed және unsigned модификаторлары да сандар шамасына әсер етеді, олар: unsigned short int – 2 байт, оның диапазоны 0 ..65536; unsigned long int – 4 байт, диапазоны 0..+4 294 967 295. Айнымалыларды сипаттау кезінде бүтін тұрақтылар–костанталар мәндерін де көрсетуге болады. Мысалы:

```
int k=0; (бір ғана сан сипатталған және оған мән берілген)
int k1,k3=0; (біреуі сипатталған, екіншісіне мән берілген)
```

Unsigned типі int, long, short түйінді сөздерімен сипатталатын типтердің модификаторы ретінде қолданылады. Мысалы:

```
unsigned int sum=0;
```

Бүтін типті шаманың компьютер жадында ішкі бейнеленуі – екілік жүйедегі код түріндегі бүтін сан. Signed спецификаторын пайдаланғанда, санның ең жоғарғы биті санның таңбасын (0 – оң сан, 1– теріс сан) көрсетеді. Unsigned спецификаторы тек оң сандарды бейнелейді, өйткені оның жоғарғы разряды да санның коды болып қарастырылады. Сонымен, int типті мәндердің диапазоны спецификаторға байланысты өзгеріп отырады екен. IBM PC тәрізді компьютерлер үшін әр түрлі спецификаторы бар бүтін типті шамалардың өзгеру диапазоны 2 кестеде келтірліген. Алдын ала келісім бойынша барлық бүтін санды типтер таңбалы болып саналады, яғни signed спецификаторын жазбаса да болады. Бағдарламада кездесетін константалардың жазылуына қарай, яғни солардың сыртқы бейнесіне сәйкес белгілі бір тип тағайындалады. Егер ол тип, кейбір жағдайларға байланысты, бағдарламалаушыны қанағаттандырмайтын болмаса, онда санның соңына жалғастырылып керекті типтің атына сәйкес бір әріп – L, l(long) немесе U, u (unsigned) жазылады. Мысалы, 32L константасының типі long және ол компьютердің жедел жадында 4 байт орын алады. Қажет болса, Lжәне U әріптерін қатарластыра да қолдануға болады, мысалы, 0x22UL немесе 05Lu. Осы атаулардағы short int, long int, signed int және unsigned int типтерінshort, long, signed және unsigned деп қысқаша жазуға болады. Бүтін типтердің мүмкін болатын ең кіші және ең үлкен мәндері компиляторға байланысты болып келеді де, C++ тілінде <limits.h> (<climits>) тақырыптық файлында көрсетіледі, нақты типтердің сипаттамалары – <float.h>(<cfloat>) файлында және де numeric_limits класының үлгілерінде беріледі).

Күрделі және құрамдас деректер типтері. C/C++ тілдерінде негізгі типтерден бөлек, солар арқылы жасалатын туынды типтерді де пайдалануға болады. Туынды типтердің үш түрі бар, олар:

- берілген типтегі элементтер массиві немесе жиымы;
- берілген типтегі объектіге нұсқауыш;
- берілген типтегі мән қайтаратын функция. Жиым немесе массив – бір типтегі элементтердің реттелген жиыны. Жиымның әрбір элементіне компьютер жедел жадынан нақты орын беріледі. Бір жиым элементтері тізбектеле қатар тұрған жады ұяларында орналасады. Жиым элементтері саны оның өлшемі (ені) болып табылады. Компьютер жадынан орын бөліп беру үшін жиымның өлшемін білу керек. Жиымға орын бөліп беру бағдарламаны компиляциядан өткізу кезінде атқарылады. Жиым бір атаумен – идентификатормен аталады да, индексті айнымалы ұғымына сәйкес келеді. Мысалы, бүтін сандардан тұратын a 100 жиымы былай анықталады: int a[100];// бүтін типтегі 100 элементтен тұратын a жиымымұнда sizeof(a) операциясының мәні 400 болады, яғни әрқайсысы 4 байттан тұратын 100 элемент. Жиым элементтері 0-ден бастап нөмірленеді.

```
Жиым элементін пайдалану үшін оның нөмірін (индексін) көрсету керек:
a[0] – индекс константа түрінде берілген,
a[55] – индекс константа түрінде берілген,
a[i] – индекс айнымалы түрінде берілген,
a[2*i] – индекс өрнек түрінде берілген.
```

Мысалы, мынадай тізбек 0 1 1 2 3 5 8 13 21 Фибоначчи тізбегінің 9 элементін құрайды (алғашқы екі санды таңдап алып, келесі санды алдыңғы екеуін қосу жолымен алады). Ал мынау өзіне және бірге бөлінетін жай сандар тізбегінің алғашқы 7 элементі: 1 3 5 7 11 13 17 Осындай бір текті тізбектерді жиым түрінде сипаттап, оған бастапқы мән беріп инициалдау үшін былай жазамыз:

```
int fib[8]={0, 1, 1, 2, 3, 5, 8, 13, 21};  немесе
int fib[]={0, 1, 1, 2, 3, 5, 8, 13, 21};  деп
```

көрсетеміз. мұндағы fib – жиым аты, оның элементтерінің типі int, ал ені, яғни ұзындығы – 9, жиым элементтерінің индекстері 0-ден бастап нөмірленеді, сол себепті 9 элемент 8 индекспен көрсетіледі. Мәндері толық көрсетілсе,индексті жазбаса да болады. Ал былай болса,

```
int fib[8]={0, 1, 2, 3};  қалған элементтері 0 болып саналады.
n=10; k=2; fib[n-k]={0, 1, 2, 3};  десе де болады.
```

Жоғарыдағы тізбектің 7-ші элементін бір бүтін айнымалыға меншіктеу үшін былай жазамыз.

```
int a = fib[6]; // a = 8
Жиымды сипаттау кезінде оның ені нақты санмен көрсетіледі, мыс., a[20], ал индексті  a[n]  деп жазу үшін алдын ала #define n 20    жолы көрсетіледі немесеconst n=20;    болып жазылады.
```

Бір өлшемді жиымдарды өңдеу Жиым элементтерін енгізу немесе оларды түрлендіру үшін цикл операторлары қолданылады. Төменде 10 элементі бар жиымды 0-ден 9-ға дейінгі сандармен толтырып, сонан кейін оларды кері бағытта экранға шығару мысалы көрсетілген: ...

```
main ()
{int ia[10];
```

int index;

```
for (index = 0; index < 10; index ++)
ia[index] = index;
for (index = 9; index >= 0; index --)
printf(" %i", ia[index]);
```

} ... C/C++ тілдерінде жиымды жиымға бірден теңестіруге болмайды, мысалы, A0, a1, a2, ... , a9 және c0, c1, c2, ... , c9 жиымдары үшін a = c деп жазуға рұқсат етілмейді. Олардың элементтерін цикл ішінде бір-біріне біртіндеп теңестіру керек. Мысалы, мынадай цикл жазылуы тиіс:

```
int a[10], c[10];
for (int i=0; i<9; ++i)
a[i]=c[i];
```

Ұзындығы өзгермелі динамикалық жиымдар. Жиымды сипаттау кезінде міндетті түрде оның элементтері санын көрсету керек. Сол арқылы компилятор компьютер жадынан оған керекті орын бөледі. Бұл онша ыңғайлы емес, өйткені жиымдағы элементтер саны есепке байланысты өзгеріп отыруы мүмкін. Сондықтан C/C++ тілінде ұзындықтары өзгеріп отыратын динамикалық жиымдар қарастырылған (олар төменде айтылған). Ал динамикалық жиымға ұқсас мүмкіндікті былай да жүзеге асыруға болады:

1. жиымды анықтау кезінде оған компьютер жадынан артығымен орын бөлінеді, мысалы: const int n = 100; // ат қойылған константа int b[n];
2. бағдарламалаушы жұмыс барысында жиым элементтері санын n-нен аз етіп алады. int m; // C тілінде printf("\nJyim elementteri sani: "; scanf("%d", &m); немесе int m; // C++ тілінде cout << "\nEnter the size of array" << m << ":"; cin >> m;
3. мұнан кейінгі жұмыс жиымның осы m элементімен ғана орындалады, яғни жиымның тек белгілі бір бөлігі ғана (m<n) пайдаланылады. Жиымды толтыру үшін кездейсоқ сандарды пайдалану. C/C++ тілдерінде кездейсоқ сандар беретін функциялар бар. Олар: int rand() – 0..RAND_MAX аралығынан кез келген кездейсоқ бүтін сан береді, мұндағы RAND_MAX константасы (бас әріптемен жазылуы тиіс) ең үлкен бүтін санға 32767-ге тең болып саналады; int random(n) функциясы 0..n аралығынан кез келген кездейсоқ бүтін сан береді, мұндағы n – кез келген бүтін сан. Осы функцияларды пайдалану <stdlib.h> тақырып файлы арқылы орын-далады. Мысал: //a[n] жиымына кездейсоқ сандар енгізу #include <conio.h> #include <stdio.h> #include <stdlib.h> void main() { int a[100]; int n; printf("\nEnter the size of array:", n); scanf("%i",&n); for(int I=0;I<n;I++) {a[I]=rand()%100-50; // 0..50 аралығындағы // оң және теріс сандар алу printf(" %i ", a[I]);} // жиым элементтерін// экранға шығару getch(); }

```
Жиымды өңдеу есептерінің түрлері (кластары)
```

Жиымды өңдеу есептері көбінесе бірыңғайланған төрт түрге бөлінеді.

1. Есептердің 1-түріне жиым элементтерінің барлығын немесе көрсетілгендерін бірдей бір тәсілмен өңдеу есептері жатады.
2. Есептердің 2-түріне (класына) жиым элементтерінің орналасу реттілігін өзгерту тәсілдері жатады.

3. Есептердің 3-класына бірнеше жиымдарды қатар өңдеу немесе бір жиымның ішкі элементтерін бірнеше топқа бөліп жеке-жеке өңдеу тәсілдері жатады. Жиымдар бір тәсілмен – синхронды өңделеді немесе әр түрлі тәсілмен – асинхронды түрде өңделеді.
4. Есептердің 4-класына жиымның берілген санға тең бірінші элементін табу, яғни іздеу есептері жатады. Екі өлшемді жиымдар. Екі өлшемді жиымды – матрицаны пайдалану үшін тік жақшалар ішінде олардың екі өлшемінің де енін көрсету керек. Мысалы: int a[4][3]; алғашқы сан жолдар санын, ал екінші сан бағаналар санын көрсетеді, а жиымы 12 элементтен тұрады. Оларға бастапқы мәнді былай береміз: int a[4][3]={{0,1,2}, {3,4,5}, {6,7,8}, {9,10,11}}; ішкі жүйелі жақшаларды қоймаса да болады: int a[4][3] = {0,1,2,3,4,5,6,7,8,9,10,11}; Келесі түрде сипаттау жолдардың тек бірінші элементтерін ғана анықтайды,қалған элементтер 0-ге тең болып саналады: int a[4][3]={ {0},{3},{6},{9} }; Егер ішкі жүйелі жақшалар алынып тасталса, онда мағынасы өзгереді. int a[4][3]={ 0,3,6,9 }; мұнда бірінші жолдың 3 элементі мен екінші жолдың бірінші элементі анықталады да, қалғандары 0 болып саналады. Екі өлшемді жиымды инициалдау қабаттасқан циклдер арқылы орындалады. 1-мысал. /* a[3][4] жиымы элементтерін rand( ) арқылы енгізу және экранға шығару */ #include <stdio.h> #include <stdlib.h> #include <conio.h> main() { const int jol=3, bag=4; int a[jol][bag]; clrscr(); for (int i=0; i<jol; i++) for (int j=0; j<bag; j++) a[i][j]=rand()%100-50; printf("\na[3][4] жиым элементтері мәндері:"); for (i=0; i<jol; i++) for (j=0; j<bag; j++) printf(" %i",a[i][j]); getch(); }

Тәжірибелік сабақ 8. Заманауи бағдарламалау тілдерінің базалық конструкциялары. Тізбектелген және тармақталған процесстерді бағдарламалау.

Си тілінің алфавиті:

```
•    Латын алфавитінің бас және кіші әріптері  (A–Z, a–z).
```

- 0 ден 9-ға дейінгі араб цифрлары.
- Арнайы символдар : "'.,{}|[]()+-*%\;:?⇔_!&#~^ Пробелмен бөлінбеген кейбір символдар комбинациясы бір символ ретінде қарастырылады. Оларға мыналар жатады: ++,==,&&,||,<<,>>,>=,<=,+=,-=,*=,/=,**/,// Әріптермен немесе _ (астын сызу) белгісінен басталған латын әріптерінің, цифр, _ белгілерінің тізбегі идентификатор болып табылады. Pascal – дан айырмашылығы Си – де бас және кіші әріптер айырылады. Си тілінде сандық деректердің негізгі 4 типі бар. Int, char – бүтін сандар, Float, double – нақты сандар. Алфавит. СИ тілінің символдарын бес топқа бөлуге болады.
- Идентификаторлар мен қызметші сөздерді тудыру үшін керек символдары (1 кесте). Бұл топқа ағылшын алфавитінің жолдық кіші және бас әріптерді, сонымен қатар астын сызу символы да кіреді. Бірдей жолдық кіші және бас әріптердің кодтары әртүрлі болғандықтан олар әртүрлі символ болып саналады.

1 кесте Латын алфавитінің жолдық бас әріптері A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Латын алфавитінің жолдық кіші әріптері A b c d e f g h I j k l m n o p q r s t u v w x y z Астын сызу символы _

```
2. Орыс алфавитінің жолдық кіші және араб цифрлары (2 кесте).
```

2 кесте Орыс алфавитінің жолдық бас әріптері

А Б В Г Д Е Ж З К Л М Н О П Р С Т У Ф Ч Ц Ч Ш Щ Ы Ь Э Ю Я ОРЫС АЛФАВИТІНІҢ ЖОЛДЫҚ КІШІ ӘРІПТЕРІ

а б в г д е ж з и к л м н о п р с т у ф х ц ч ш щ ы ь э ю я Араб цифрлары 0 1 2 3 4 5 6 7 8 9

1. Номерлеу белгілері және арнайы символдар (3 кесте). Бұл символдар бір жағынан есептеу процесін ұйымдастыру үшін, ал екінші жағынан копиляторға инструкцияларды беру үшін қолданылады.

3 кесте Символ Атауы Символ Атауы , Үтір) Дөңгелек оң жақша . Нүкте (Дөңгелек сол жақша ; Үтірлі нүкте } Фигуралық оң жақша : Қос нүкте { Фигуралық сол жақша ? Сұрау белгісі < Аз белгісі ‘ Апостроф > Көп белгісі I Леп белгісі [Квадрат жақша | Тік сызық] Квадрат жақша / Бөлшек сызық # Номер \ Кері сызық % Процент ~ Тильда & Амперсанд * Жұлдызша ^ Логикалық жоқ + Қосу = Тең - Алу “ Тырнақшалар

1. Басқаратын және бөлетін символдар. Бұл топқа кіретін символдар: бос орын (пробел), табуляция символдары, жолды көшіру, каретканы қайтару, жаңа жол және жаңа бет. Бұл символдар қолданушы анықтайтын константа мен идентификатор сияқты объектілерді бір – бірінен ажыратады.

1. Сонымен бірге Си тілінде басқарушы тізбектер, яғни информацияны енгізу – шығару функцияларында арнайы символдық комбинациялар кең қолданылады. Басқарушы тізбек кері бөлшек сызығы (\) (міндетті түрде бірінші символ) және латын әріптері мен цифрларының (4 кесте). Негізінде құрылады.

---

4 КЕСТЕ БАСҚАРУШЫ ТІЗБЕК

Атауы Он алтылық ауыстыру \a Қоңырау 007 \b Қадамға кері қайту 008 \t Көлденең табуляция 009 \n Жаңа жолға көшу 00A \v Тік табуляция 00B \r Каретканы қайтару

\f

```
Форматты көшіру (перевод)
```

\” Тырнақшалар

\’ Апостроф

\0 Нөлдік символ

\\ Кері бөлшек сызық

\ddd Сегіздік түрдегі ПЭЕМ-ң кодтар символы

\xddd Оналтылық түрдегі ПЭЕМ-ң кодтар символы

Комментарий – сол жағынан /*, ал оң жағынан */ белгілерімен шектелетін символдар жолы. Мысалы: /* Бұл комментарий */ . Си тілінде 6 лексем класы бар: еркін таңдалатын және қолданылатын идентификаторлар, қызметші сөздер, константалар, (жолдық константалар), операция (операция белгілері), бөлгіштер (пунктуация белгілері). Идентификатор. Идентификатор деп тек әріптен немесе арнайы символдан басталуы міндетті әріптер, цифрлар, сонымен қатар арнайы символдар тізбегін айтады. Мысалы: KOM_16, size88, _MIN, TIME, time. Мұнда бас және кіші әріптерден айырмашылығы бар, яғни соңғы екі идентификатор әр түрлі. Идентификаторларды жазу үшін латын алфавитінің жолдық кіші және бас әріптері қолданылады. Арнайы символ ретінде астын сызу символы (_) қолданылуы мүмкін. Басты айырмашылық болып компилятордың идентификатордың қанша символ қолбылдаса да, оның тек 31 символы ғана оқылады. Идентификатор құрылымының, функцияның, айнымалының, т.б. хабарлануы кезінде құрылады. Осыдан соң оны бағдарламаның келесі операторларында қолдануға болады. Идентификаторды таңдау кезінде маңызды айырмашылықтарды білу керек. Біріншіден, идентификатор қызметші сөзбен, кейінгі сақталған сөздермен және СИ тілінің компиляторынан библиотеканың функциясының атымен сәйкес келмеуі керек. Екіншіден, (_) символына идентификатордың бірінші символы ретінде ерекше көңіл бөлу керек, идентификаторлар осылай құрылғандықтан, бір жағынан жүйелі функциялардың атымен және айнымалылармен сәйкес келуі мүмкін, ал екінші жағынан бұндай идентификаторларды басқа типтегі компьютерлерде қолдануға болмайды. Қызметші сөздер. Программистің өзі еркін таңдап қолдана алмайтын, тілдегі қабылданып сақталған идентификаторлар қызметші сөздер деп аталады. Қызметші сөздер мәліметтер типі, жады кластары, тип квалификаторлары, модификатор және операторлар типіне анықтайды. Қызметші сөздер тізімі: auto double int struct break else long switch

```
register  tupedef  char  extern  return  void  case  float
```

unsigned default for signed union do if sizeof volatile continue enum short while

Қызметші сөздер идентификатор ретінде қолданыла алмайды. Олар мағынасына қарай келесі жолмен топталады. Мәліметтер типтерін анықтау үшін типтердің спецификаторлары мен квалификаторлары қолданылады.

Тәжірибелік сабақ 9. Тізбектелген және тармақталған процесстерді бағдарламалау. Түрлі айнымалылардың, тұрақтылар мен операция символдарының қандай да болмасын бір жинақ түрінде жазылуы өрнек деп аталады. Си тілінде өрнек соңына нүктелі үтір (;) символы қойылып жазылады. Осы түрде жазылған өрнек не функция оператор делінеді. Мысалы, операторлар:

```
Z= (3* x+y +5);
y= sin (x);
```

Сипаттаманың соңына да нүктелі үтір қойылып жазылады, сондықтан ол да оператор . Мысалы: floaty, z; сипаттамасы – оператор. Бір символдық оператор мәні бір не бірнеше қатарларда жазылуы мүмкін, тек онда пайдаланатын символдар саны байтпен көрсетілген аралықтан аспаса болғаны. Функция да С және С++ тілдерінде түрлі ұғымды білдіреді. Мысалы:

- блок басында сипатталатын айнымалы;
- бағдарлама ішінде орындалатын блок;
- арнайы ат беріліп, кітапханада мәнін есептейтін ішкі бағдарлама бойынша жазылып сақталған функция, (мысалы, sin(), sgrt(), т.б.). Мысалдар: int radius (x); -тип, функция аты және аргумент { операторлар – құрама оператор – функция } float x,y - тип, айнымалылар аттары. Сипаттамаға сәйкесх, у айнымалыларының мәндері үшін алты мәндік ондық цифрлар алынады. Y=sin (6.5)+3; - оператор; , sin() функциясының мәні үшінарнайы стандартты кітапхана шақырылып, мәні автоматты түрде есептеледі. (Си тілінде тригонометриялық функциялардың мәндері радиан арқылы өлшенеді). Мұндағы ескеретін жайт: соңғы өрнек – оператор. Оның соңына қойылған қойылатын нүктені үтір алып тасталған кезде қалған бөлігі (y=sin (6.5)+ 3) функция, ал sin () –осы функция ішіндегі стандартты функция. sin (6.5)- өрнегі функцияны шақыру деп аталады. Операциялар Си, С++ тілдерінде пайдаланылатын негізгі операциялар арифметикалық, қатыс, логикалық, меншіктеу, биттік және шарт.

Арифметикалық операциялар Операция Айнымалы типтері

```
Қосу  (+)
```

int,char,float

```
Азайту (–)
```

int,char,float

```
Көбейту (∗)
```

int,char,float

```
Бөлу (/)
```

int,float

```
Қалдық (%)
```

int

Егер операндтар бір типті болса, алдыңғы үш операция басқа алгоритмдік тілдердегі операциялар сияқты пайдаланылады. Бүтін типті айнымалылардың мәндерін бөлу (/) операциясының нәтижесі де бүтін типті (int) бірақ ерекшелігі - бұл операция нәтижесінде бөліндінің тек бүтін бөлігі шығарылады да, қалдық алынып тасталады. Мысалы, 17/3=5; -17/3=-5. сондықтан бағдарламада кездескен бөлшектермен амал істеу кезінде оларды float типі арқылы сипаттау керек. Мысалы,

```
    •    float k,y,s; for(k=1; k<=3; k++)y=1/k; s=s+y;}
    •    float s; s=1.+1./2+1./3;
```

үзінділерінің нәтижелері бірдей.

```
Қалдық (%) операциясы да тек бүтін типті сандарды бөлуде қолданылады:17% 3=2;
```

-17% 3= -2. Мұндай ерекшеліктер төмендегі бағдарламада көрсетілген.

```
# include <stdio.h>
      #include <conio.h>
      main()                                (2)
```

{

```
      int x,y, z1, z2, clrscr ();
      printf (“сандар:x,y=?”);
      scanf(“%d %d”, &x, &y);
      getch (); return 0;
```

} Ескерту. Бірнеше айнымалыларды бірден енгізу қажет болса, scanf() операторында формат спецификаторларын жоғарыдағы сияқты ретімен толық енгізіп қою керек.

Қатыс операциясы. Логикалық операциялар. Қатыс операциясы салыстыру үшін шарт түрінде жазылатын операция. Си, С++ тілдерінде алты қатыс операциясы, үш логикалық операция бар. Қатыс операциялары:

```
      Теңдікті тексеру(= =)
      Тең еместікті тексеру(!=)
      Кіші(<)
      Кіші не тең (< =)
      Үлкен (>)
      Үлкен не тең(> =)
```

Ескерту. Теңдікті тексеру және басқа операциялар шартты тексеруге арналған if операторында пайдаланады. (= = және= операциялары бірдей емес). Логикалық операциялар – және, немесе, жоқ шарттары түрінде жазылады. Жазылу түрі:

```
&& –және (and),
      | | –немесе (or),
      ! –терістеу (not).
```

Мысалдар:

```
      if (k>50 && j= =24)
      if (k<=2 | | k>5)
```

Қатыс және логикалық операциялардың нәтижелері әдеттегідей екеу ақ: ақиқат (true) не жалған(false). Нәтиже ақиқат болса, мәні=1, жалған болса мәні=0 деп белгіленеді. Ескерту. Логикалық операцияларда if, for сияқты басқарушы операторлар да пайдаланылады. Тағы бір ескертетін жайт: келесіде Си тілі деп жазылса, Си және С++ тілдері деп түсіну деп керек. Егер ол тек Си тіліне қатысты болса, арнайы ескертіледі. Берілген екі айнымалының мәндері түрлі болғанда ғана мәні шын болатын болғызбайтын және (XOR) логикалық операциясы Си тіліне енгізілмеген. Меншіктеу операциясы Меншіктеу операторы өрнек мәнін айнымалыға меншіктеу операциясының соңына; таңбасын қоюдан тұрады. Меншіктеу операциясы әдеттегідей теңдік (=) таңбасымен белгіленеді. Айнымалы lvalue (адрестік шама) деп аталады. Мысалы,

y=3*x+5; операторындағы y айнымалысы – lvalue. Си тілінде қосарлы меншіктеу операциясы да бар. Мысалы, y = x = a*b+5; операторында алдымен a*b+5 өрнегінің мәні x айнымалысына меншіктеледі, ал ол y айнымалысына меншіктеледі. Си тіліне қосымша меншіктеу операциялары да енгізілген:

```
      +=, – =, *= және %=
```

Олар – айнымалы мәніне +, –, * не % операциясы қосылған өрнекті бастапқы айнымалыға меншіктеу операциялары. Мұндай операцияларды пайдалану тәсілі төменде көрсетілген. Бірдей операциялар:

```
m+=10   және   m=m+10
m – =10 және   m=m –10
m*=10   және   m=m*10
m%=10  және  m=m%10
```

Мұндағы m+=10 командасы m айнымалысының мәнін 10 ға өсіруді білдіреді, ал m=m+10 командасы оң жақтағы m –нің мәнін орнына қойып, және оған 10–ды қосып, нәтижені m айнымалысына меншіктеуді білдіреді. Олардың ерекшеліктері жоқ, тек бірінші түрде жазылған команда екінші түрде жазылған командадан тезірек орындалады.

Биттік операциялар Процедуралық тілдерде жад ұяшықтары мен байттар бойынша жұмыс істеу мүмкін екені белгілі. Си тілінде тек байттық ұяшықтармен емес, олардың биттік разрядтарын өзгерту операциялары да бар:

```
<<, >>, &, I, ^  және   ~
```

Разрядттық операциялар бүтін типті шамалардың биттік шамаларын өңдеуге арналған.

```
<< – солға жылжыту операциясы. Жазылу түрі:
мән    << биттер саны
```

Мұндағы <мән >берілген санның екілік жүйеде жазылуы. Мысалы, x=11012(=1310) санын 3 битке солға жылжыту керек болсын:

int x=11012

```
 int y=x<<3
```

Яғни, x өзгермей өз күйінде қалады. :x=11012 x мәнінің екілік кодының оң жағына үш нөл қосылып қоятын y- тің мәні: y=11010002 (оның ондық мәні : y-10410).

```
>>– оңға жылжыту операциясы: мән >> биттер саны
```

Операция нәтижесінде берілген екілік мәннің сол жағына операцияда көрсетілген биттер санындай 0-дер енгізіліп қойылады (сан кішірейеді).

```
        Мысалы: int x=110002; (=2420)
                        int y=x>>3
үшін нәтиже: y =112(=310) .
Ескерту. x>>3  операциясы x мәнін  810– ге бөлумен,    x<<3 операциясы  x  мәнін 810–ге
```

көбейтумен бірдей, себебі : 10002=810>> операциясының нәтижесінде қалдық алынып тасталады.

?! Шарт операциясы ?! шарт операциясы - үш операндтан тұратын жалғыз операция. Оның жазылу синтаксисі:

```
<өрнек1> ? <өрнек2>: <өрнек3>
```

Бұл операцияда алдымен өрнек1-дің мәні (ақиқат не жалған екендігі) анықталады. Егер ол ноль (жалған) болмаса, онда өрнек2-нің мәні нәтиже түрінде қабылданады. Егер өрнек1-дің мәні жалған (ноль) болса, онда өрнек3-тің мәні есептеледі де, оның мәні операция нәтижесі үшін алынады. Операцияны орындау үшін алдымен оны кез келген айнымалыға меншіктеп алу керек. Сонымен, операцияны орындау кезінде соңғы екі өрнектің бірінің ғана мәні нәтижеленеді. Мысалы, max=(x>y) ? x:y өрнегі үшін x>y болса, max айнымалысына x-тің мәні, ал x<y болса, y-тің мәні меншіктеледі.

БАҒДАРЛАМА:

```
# Include <stdio.h>
#Include <conio.h>
 main()                                (4)
```

{

```
floatmax,x,y;  clrscr ();
      printf (“x,y=?”);
      scanf(“%f %f”, &x, &y);
      max=(x>y) ? x:y;
      printf (“max=%f”,max);
      getch (); return 0;
```

{ Нәтиже: x,y=? 13.2 5.7 Ent

```
                    max=13.200000
```

Ескерту: Математикалық операциялардың орындалу басымдылығы белгілі, ал басқа кейбір операциялардың басымдылық реті мынадай:

```
      1. ()–функцияны шақыру,
```

1. ++, - - ,
2. <<, >>,
3. Логикалық операциялар,
4. Жай меншіктеу: = , + -, -= және т.б.

Тәжірибелік сабақ 10. Си тілінің операторлары printf() – экранға мәтінді не аргументтердің мәнін форматты түрде шығаруға арналған стандартты функция.Мұндағы ескеретін жайттар: 1) Егер шығаруда форматтың қажеті болмаса, онда операторды (1-) бағдарламаның бірінші printf операторы сияқты форматсыз пайдалана беруге болады. 2.Екінші printf операторында %f форматы %2.3f түрінде берілсе, сәйкес нәтиже 0.001 дәлдікпен шығады. scanf() – форматты түрде енгізу функциясы. (соңына нүктелі үтір қойылып жазылғандықтан, ол–операторда). getch()– кез келген перне басылғанша мәндермен жұмыс жасалатын терезелік бетті уақытша ұстап тұруға арналған оператор. (Турбо Паскальдағы ReadKey операторы сияқты, ол басылған перненің кодын қабылдайды). Кезкелген перне басылған кезде бағдарлама жазылған экран қайта көрінеді.

- printf – монитор экранына хабарламалар мен айнымалылар мәнін шығаратын функция.
- printf функциясының бірінші параметрі болып шығарылатын мәтін мен айнымалы мәнін шығару форматын анықтайтын шығару қатары болып табылады.
- айнымалы мәнін шығару форматы спецификатор түрінде беріледі: % - символымен басталады.
- Сандық мәндерді шығаруда көбінесе мына спецификаторлар қолданылады:

- %i,%d – бүтін сандарды таңбасымен шығару,
- %u - бүтін сандарды таңбасыз шығару,
- %f - жылжымалы нүкте түрінде берілген бөлшек сандарды шығару,
- %n.mf – жылжымайтын нүкте форматындағы бөлшек сандарды шығару,
- n – санның бүтін бөлігі, m – бөлшек бөлігі;
- \n – жаңа қатар,
- \t – табуляция
- \''- екеулік тырнақша
- \\ - \ символы.
- printf функциясының орнына puts функциясын қолдануға болады, ол экран бетіне мәтінді шығарғаннан кейін автоматты түрде келесі қатардың басына әкеледі.

- Клавиатурадан берілгендерді енгізу үшін scanf функциясы қолданылады.
- scanf функциясының бірінші параметрі басқарушы жол болып табылады, қалған параметрлері – мәндері енгізілетін айнымалылар адресі.
- басқарушы жол екі тырнақшаға алынған спецификатор тізімін береді:
- %i – бүтін сандарды таңбасымен енгізу,
- %u - бүтін сандарды таңбасыз енгізу,
- %f - бөлшек сандарды енгізу
- %s- жолды енгізу.

Мысал. Шеңбер ұзындығын есептеу керек. Бағдарламаны мынадай түрде құруға болады:

```
# include <stdio.h>
#include <conio.h>
main()                                (1)
```

{

```
int r; float length;   clrscr();
printf(“ радиус r=?”);
scanf(“%d”,&r);
length=3.14159 *2*r;
printf(“радиус=%d  ұзынд.=%f \ n”,r,length);
getch(); return 0;
```

} // Нәтиже: r=? 5

```
радиус=5   ұзынд.=31.415899
```

Бағдарламаға енгізілген нұсқаулар: Мысалы, бағдарламада екінші printf операторы төрт бөлімнен тұр: басталымы – функция аты, оның ішіне үтірлер арқылы бөлініп үш аргумент жазылған. Біріншісін (тырнақшалар ішінде жазылған “Радиус=%d ұзынд.=%f \ n” аргументін басқару жолы не форматтық жол деп атайды. Ол экранға шығаратын келес ir, length аргументтерінің мәндер форматын (жазылу үлгісін) анықтайды.

Шартты оператор. IF операторы. Тармақталу (if) операторы, процедуралық бағдарламалау тіліндегі сияқты, екі үлгіде жазылады:

```
1)  if   (шарт)  оператор;              /* қысқа  құрылым*/
2)  if   (шарт)  оператор1;
```

else оператор2; /* толық құрылым*/ Соңғы құрылымды мынадай түрде алуға болады:

```
if   (шарт1)  оператор1;
else  if   (шарт2)  оператор2;
```

Егер if операторының құрамында бірнеше оператор болса, олар фигуралық жақшаларға алынып жазылады.

- мысал. функциясының мәнін есептеу керек. #include <stdio.h> #include <conio. h> main() { float x,y; clrscr(); printf(“x=?”); scanf(“%f”,&x); if (x<0 || x>2) y=2* x + 1; else if (x>=0 && x<=2) {y=x*x+3; printf (“Нәтиже:”);} printf(“y=%f ”,y); getch(); }

2-Мысал. Екі санның бөліндісін есептейтін бағдарлама жазу. Бағдарлама пайдаланушының енгізген мәліметтерінің дұрыстығын тексеру қажет. Егер олар дұрыс болмаса (бөлгіш нөлге тең болса) қате жөнінде хабарлама беру қажет. Төменде: Бөліндіні есептеу Бір қатарға бөлінгіш, бөлгішті енгізіп, «Enter»-ді басыңыз—12 0 Сіз қателестіңіз. Бөлінгіш нөлге тең болмауы керек.

```
# include <stdio. h>
# include <conio.h>
void main ( )
{   float a, b, c; // бөлгіш, бөлінгіш және бөліндінің мәні
     printf ( “/n Бөліндінің мәнін есептеу/n”);
     printf ( “ Бір қатарға бөлінгіш, бөлгішті енгізіп,”);
printf ( “ «Enter»–ді басыңыз”);
       printf ( “ –> ”);
scanf ( “% f % f ”, & a, & b);
if ( b ! = 0)
{         c = a / b ;
           printf ( “ Бөлшектің бөліндісі % 5.2 f   % 5.2f ”, a, b );
           printf (“ тең  % 5.2 f “, c );
```

} else

```
                    printf ( “Қате ! Бөлінгіш нөлге тең болмауы керек ”);
         printf (“\ n Аяқтау үшін <Enter> –ді бас”);
        getch ( );   }
```

Тәжірибелік сабақ 11.Енгізу- шығару функциялары puts – Синтаксис:

```
puts (const char* қатар)
```

Экранға символдар қатарын шығарады және курсорды экранның келесі қатарының басына алып келеді. Функция параметрі ретінде қатарлық тұрақтыны немесе қатарлық айнымалыны қолдануға болады. Тақырып файлы: <stdio.h> gets Синтаксис:

```
char *gets (char* s);
```

Клавиатурадан символдар қатарын енгізеді. Енгізілетін қатарлар бос орыннан тұруы мүмкін. Тақырып файлы: <stdio.h> putch Синтаксис:

```
int putch (int c)
Символды экранға шығарады. Тақырып файлы: <conio.h>
```

- Мәтінді экран бетіне түрлі түсте шығару үшін cprintf және cputs функцияларын қолдану керек.
- Жаңа жолға өту \n\r –түрінде беріледі.
- cprintf және cputs функцияларымен шығарылатын символдар түсі textcolor (түс) функциясы арқылы орнатылады.
- Түстің фонын textbackground (Түс) орнатады.

- Clrscr, textcolor және textbackground функцияларын қолдану үшін бағдарлама мәтініне # include <conio.h>директивасын қосу қажет.

Циклдық есептелімді процестерді бағдарламалау. Процедуралық бағдарламалау тілдерінде GOTO (сөзсіз өту) операторы пайдалана берілмейтіні, бірақ ол ентаңбаға (метка label) сілтеме ретінде қолданылатыны белгілі. С және С++ тілдерінде де осы сияқты. Ентаңбаға сілтеме тек бір блок (функция) ішінде пайдаланылады; ентаңба жөнінде кеңірек түсінік келесі тақырыпта берілген.

```
2-мысал.  M (x,y)  нүктесінің  D облысында  жатуын  анықтау  керек (1-сурет):

          {x²+y²,    x>=0, y>=0
D=       {x+y>=-1,   x<0,  y<0
```

Нұсқау. (x,y) нүктесінің берілген дөңгелек ширегінде жату шарты: x>=0 , y>=0, x²+y²<=4. Нүктенің берілген үшбұрышта жату шарты: x<=0; y<=0 және x+y>= -1

```
#include  <stdio.h>
#include <conio. h>
main()
```

{

```
float x,y; int ml, ul, rl ;clrscr();                    1-сурет
ml:                                                        (6)
puts (“Нүкте  координаттары: x,y-?”);
printf(“x,y=?”);scanf(“%f %f”,&x,&y);

if  ((x>=0 && y>=0&& x*x+y*y<=4) ||
  (x<=0 && y<=0 && x+y>=-1))
  {printf (“Нүкте облыста жатады”);
 goto ul ;}
 else  printf (“Нүкте облыста жатпайды”);
 ul: printf (“Жалғастыру керек пе-?”);
printf (“Иә үшін енгізу  -10 ,Жоқ –басқа сан енгізу”);
scanf (“%d”,&rl);
if (rl==10)  goto ml; else
printf (“Жұмыс соңы. Енг пернесін басыңыз”);
getch() ;  return 0;
```

} Бағдарламаға енгізілген puts операторы printf операторы сияқты орындалады, бірақ ерекшелігі : ол тек мәтін шығаруда пайдаланылады және енгізілген мәтінді шығарып болған соң курсорды жаңа жолдың басына орналастырып қояды.

Тәжірибелік сабақ 12. Циклдық есептелімді процестерді бағдарламалау.

С, С++ тілдерінде циклдік операторлардың үш түрі бар :

for, while, do – while, Құрама цикл денесі фигуралық жақшалар ішінде жазылады, ал дене бір операторлан тұрса, оны жақшаларға алу міндетті емес. Ескерту. Әдетте кілттік сөздер мен операторлар латын алфавитінің кіші әріптерінен жазылады. For циклі.

```
Инкремент (++) және декремент (--)  операторлары
  for  (өрнек1; өрнек2; өрнек3)
{ операторлар}
```

өрнек1 –циклді киниалдау. Қарапайым түрде ол- параметрге бастапқы мән меншіктеу командасы. өрнек2- циклдің аяқталу шарты. Ол ақиқат кезде циклдің денесі орындалып, басқару ретімен 3- өрнекке өтеді.

```
өрнек3–  параметрдің өзгеру шамасы (цикл қадамы).
```

C, Turbo C++тілдерінде параметр бүтін сан болса, ол 1 –ге өседі не 1-ге кемиді. Мысалы, параметр k арқылы белгіленсе, оның өзгеруінің жазылу түрі: k++не k-- ++ - инкремент (өсімше) операторы деп аталады. Ол өз операндын 1-ге өсіреді (k++ және k=k+1 операторлары бірдей). -- - декремент (төмендету) операторы делінеді. Ол өз операндын 1-ге кемітеді (k-- және k=k-1 операторлары бірдей) For цикліне енгізілген әр өрнек нүктелі үтір (;) таңбасымен ажыратылып жазылады. Мысалы : for (k=0; k<10; k++) printf(“%d\n”,k); бағдарламасының үзіндісі бағаn бойынша 0,1,2,3,.....,9 сандарын , ал,

```
for  (k=9;  k>=0; k––)     printf(“%d\n”,k);
```

үзіндісі бағаn бойынша 9,8,.....2,1,0 сандарын басып шығарады. Цикл параметрін символдық етіп алуға болады. Мысалы ,

```
 for (ch=’A’ ;  ch=’R’; ch ++)   printf(“%c”,ch);
```

циклі кезегі мен A-дан R –ге дейін латын алфавитінің бас әріптерін басып шығарады. C,C++ тілдерінде функция бөлімінде пайдаланылатын бүтін тұрақтының орнына айнымалы арқылы белгіленген атауын пайдалануғада болады. Ол үшін тұрақтыны main () функциясының алдында CONST стандартты операторы (спецификаторы) арқылы сипаттап қою керек. Сипаттау үлгісі :

```
const<атау>=<мән>
Мысалы, const ms=12; Тұрақтының типі оның мәнінің типі мен анықталады. Мәнді айнымалы атауы арқылы сипаттаудың
да қатесі жоқ, мысалы,
const ms;
```

1-мысал. For циклін пайдаланып, y=x5 функциясының мәнін есептеу керек.

```
#include <stdio.h>
#include <conio.h>
const a=5;
main()
```

{

```
float x, y; int k; clrscr();
puts(“x мәні : x=?”);
scanf(“%f”, &x); y=1;
for (k=1; k<=a; k++)
y=x*y;
printf (“дәреже=%f\n”,y); getch ()
; return 0;
```

} Цикл шексіз болуыда мүмкін. Мысалы,

```
For (k=10; k>8; k++)
```

Шексіз не жай циклден break операторын пайдаланып шығуға болады.

While, do-whileциклдері

```
Циклдердің жазылу үлгісі (while – әзірше, do – орындау):
1) while (шарт) {операторлар}
```

1. do {операторлар} while (шарт) while циклінің денесінде бір ғана оператор бар болса, оны фигуралық жақшаларға алмай жазуғада болады. 1-мысал. Жалпы мүшесі (k=1,2,...) болатын тізбектің алғашқы алты мүшесінің қосындысын табу керек. #include<stdio.h> # include<conio.h> main() { int k, a, s; clscr(); while (k<=6) { a=k*k; s=s+a; k++; } Printf("s=%d\n",s); getch(); return 0; } // Нәтиже: s=91

БАҒДАРЛАМА ДЕНЕСІНІҢ WHILE БЛОГЫН DO-WHILE ҚҰРЫЛЫМЫМЕН ДАЙЫНДАУҒА БОЛАДЫ: DO

```
{ a=k*k; s=s+a; k++; }
while (k<=6);
```

Ескерту: 1.Циклді пайдалану кезінде бөлшек түрінде берілген тізбектің мүшелері 1-ден кем, мысалы, ak=1/(k*k) түрінде берілген болса, олардың қосындысын есептеу үшін a, s айнымалыларын float типі арқылы сипаттау керек. While циклінен ерте шығу үшін break не goto операторын пайдалануға болады.

Рекурсия. Болдырмауды өңдеу. C және C++ тілдерінде тармақталу тобынан тұратын және ентаңбамен белгіленген деректердің (берілгендердің) бірін таңдау үшін switch операторы, сұрыптаушы үшін case операторы пайдаланылады. (switch – ауыстырып қосқыш , case -жағдай). Switch операторының жазылу үлгісі:

```
switch  (өрнек)
```

{

```
case <тұрақты 1>: операторлар;
```

break ;

```
case <тұрақты 2>: операторлар;
```

break ; .....

```
case <тұрақты N> : операторлар;
```

break ; default : операторлар; Мұндағы тұрақтылар – ентаңбалар (k=1,2.....,N). Ентаңба-оператор алдында жазылатын (оператор жолын белгілейтін) бүтін сан, символ не символдар тізбегінен тұратын атау. Ентаңба соңына қос нүкте (:) қойылып жазылады. Мысалдар (мұндағы ентаңбалар: 10,'C'): case 10: m-3 ;

```
case 'C': printf("Ура!");
```

Ентаңбаға өту үшін switch (өрнек) операторы пайдаланылады; мұндағы өрнек– мәні айқындалатын айнымалы не математикалық өрнек. switch операторының орындалуы әдеттегідей алгоритмдік тілдердегі сияқты: алдымен switch кілттік сөзінен соң жақшалар ішінде жазылған өрнек мәні есептеледі не айнымалы мәні оқылады. Ол ентаңбалар енгізілген сұрыптаушылар мен салыстырылады да, жүйе басқаруды ентаңба мәніне сәйкес келетін операторға (операторлар блогына) өткізеді. Егер оқылған мән бірде – бір ентаңбаға тең болмаса, басқару default (жоқболғанда)

кілттік сөзінен соң жазылған операторға беріледі. Үлгіде әр case операторының кейінгі жолда break (тоқтату) операторы енгізілген. Ол сәйкес case операторы орындалған соң оны тоқтатадыда, жүйе басқаруды Switch операторынан соң жазылған бағдарламаның келесі операторына өткізеді.

CONTINUE операторы Бір цикл екінші циклдің ішінде орналасса, олар бір-біріне салынған циклдер не цикл ішінде циклделінетіні белгілі. С, С++ тілерінде сыртқы цикл денесі фигуралық жақшаларға алынып жазылады. Егер ішкі цикл де бірнеше операторлардан тұрса, ол да осындай түрде жазылады.

```
1-мысал. (9x9) көбейту кестесін дайындау керек.

#include<stdio.h>
#include<conio.h>
main()
```

{

```
int k, j; clrscr();
for(k=1; k<10; k++)
```

{

```
For (j=1; j<10; j++)
Printf("%d*%d=%d ",k,j,k*j);
Printf("\n");
} getch(); return 0;
```

} Continue (жалғастыру) – for, while, do – while циклдерінің кез-келгенінде пайдаланатын оператор. Оның орындалу түрі: ол оқылған соң жүйе цикл денесінің қалған бөліктерін орындауды тоқтатады да, циклді қайталап орындайды, яғни басқаруды циклдің жаңа қадамын орындауға өткізеді. Бұл операторлың орындалуы break операторының орындалуына қарама-қарсы.

Тәжірибелік сабақ 13. Бағдарламалаудың құралдық саймандары, әдістері мен технологиялары

Turbo C бағдарламалау жүйесі С тілінде бағдарлама құрастырып, оны орындауға мүмкіндіктер береді. Жалпы, тұтынушы мынадай әрекеттерді орындай білуі керек:

- бағдарламаның мәтінін жазып, дискіде бағдарлама файлы ретінде сақтау; -бағдарламаны компиляциядан өткізіп, егер синтаксистік қателері бар болса, оларды түзету;
- бағдарламаны орындап, нәтижесін алу. Сонымен, кез келген бағдарлама оны теру, компиляциялау, құрастыру, атқарылушы модульді жасау және орындап нәтиже алу сатыларынан өтуі тиіс. С тілінің біріктірілген (интегралданған) ортасында бағдарлама орындау төмендегідей қадамдардан тұрады:
- компилятор қажет файлдарды іздеп тауып алуы үшін ортаның параметрлерін тағайындау;
- бағдарламалық файлды редактор ортасына жүктеу немесе теру;
- атқарылатын модульді (орындалатын файлды) жасау;
- бағдарламаны іске қосу және орындау;
- бағдарламада қате болса, оны жөндеп түзету.

БАҒДАРЛАМА ҚҰРУ ҮШІН, ЕҢ АЛДЫМЕН, ТУРБО С БІРІКТІРІЛГЕН ОРТАСЫМЕН ЖҰМЫС ІСТЕЙ БІЛУ КЕРЕК.ТУРБО С ОРТАСЫ – АРНАЙЫ КӨП ТЕРЕЗЕЛІ МӘТІНДІК РЕДАКТОР. ОЛ КӨБІНЕСЕ C:\TC\BIN\TC.EXE ФАЙЛЫН ІСКЕ ҚОСУ АРҚЫЛЫ НЕМЕСЕ СОНЫҢ ЖАРЛЫҚТАРЫНЫҢ БІРІН ІСКЕ ҚОСУ ЖОЛЫМЕН ЖҮКТЕЛЕДІ:

TC.EXE файлының жұмыс істеу барысында экранда Турбо С ортасының негізгі терезесі ашылады. Турбо С ортасын автоматты түрде жүктеп, орыс (қазақ) әріптерін теру мүмкіндігін беретін C.bat пакеттік файлының мәтінін мынадай түрде құрастыруға болады: C.bat файлын немесе оның жарлығын шерту арқылы да Турбо С ортасы ашылады. Ол іске қосылған соң, орыс әріптеріне және ағылшын әріптеріне ауысу қос Shift пернелерін (ол басқаша да болуы мүмкін) қатар басу арқылы орындалады. Турбо С редакторы DOS ортасында жұмыс істейтін көкшіл экранға шығады. Ол экран төрт бөліктен тұрады:

- меню жолы,
- түзету/теру терезесі,
- мәлімедеме хабарлар терезесі,
- қалып қатары. Терезені үлкейту үшін Alt+Enter пернелерін басу керек, редактор терезесін экранды толық алатындай етіп үлкейту үшін, оның оң жақ жоғарғы бұрышындағы тілсызық батырманы шерту керек. Ал терезені жабу үшін – сол жақ жоғарғы шеттегі төртбұрышты батырманы шертеміз. Турбо С-ден шығу үшін Alt және X (латын) пернелерін қатар басу керек.

Терезе нөмірі оның оң жақ жоғары бұрышында орналасады. Керекті терезеге (1 – 9) көшу үшін:

- Alt+0 пернелерін басқанда шығатын терезелер тізімінен керектісін таңдау арқылы;
- Alt+5 деп терезе нөмірін 5-ті енгізіп көшу;
- F6 пернесі арқылы терезелердің бірінен біріне көшуге болады. F5 пернесі арқылы терезені үлкейтуге немесе аздап кішірейтуге болады. Меню жолында 11 команда бар, олар әртүрлі қызметтер атқарады, әр менюді таңдағанда, оның ішкі командалары ашылады. Солардың кез келгенін таңдап орындай аламыз. Бағдарламалау жүйесінің саймандарының негізгі функциялары және ұйымдастыру.

БАҒДАРЛАМА ІСКЕ ҚОСЫЛЫП, ТЕРЕЗЕ АШЫЛҒАННАН КЕЙІН, КУРСОР ЖҰМЫС АЛАБЫНДА ТҰРАДЫ. МЕНЮ ҚАТАРЫНА F10 ПЕРНЕСІ АРҚЫЛЫ ШЫҒЫП, ESC АРҚЫЛЫ ЖҰМЫС АЛАБЫНА ОРАЛАМЫЗ. МЕНЮ ҚАТАРЫНЫҢ КОМАНДАЛАРЫН ЖӘНЕ ТӨМЕНГІ САТЫЛЫ КОМАНДАЛАРЫНЫҢ ҚАЖЕТТІСІН ← ↑ → ↓ БАҒЫТТАУЫШТАРЫ АРҚЫЛЫ ТАҢДАЙ АЛАМЫЗ. КОМАНДАНЫ ОРЫНДАУ ҮШІН ENTER ПЕРНЕСІН БАСАМЫЗ.

Осы әрекеттерді тышқан қолтетігімен де қалыптағыдай етіп орындауға болады. Командалар тізімі •Ё – қосымша әрекеттер орындау.

```
•File (файл) — файлдармен жұмыс істеу.
•Edit (түзету) — ашық терезедегі мәтінді түзету режимдерін орындау.
•Search (іздеу) — іздеп табу әрекеттерін орындау.
•Run (атқару) —  бағдарламаны орындау.
•Compile (компиляция) — бағдарламаны компиляциядан өткізу.
•Debug ( отладка) — бағдарламаны жөндеу.
•Options (нұсқалар) — орта параметрлерін тағайындау.
•Windows (терезе) — тереземен жұмыс істеу.
• Help (көмек) — анықтамалық жүйеден көмек алу.
```

Қосымша әрекеттер жұмыс істеп тұрған файлдарды анықтау, ішкі Ассемблерді іске қосу сияқты әрекеттерді орындайды.

```
File менюінің ішкі командалары (17.2-сурет):
Open  F3  –  бұрын жазылған файлды ашу {A:\LB1.C}
```

New – жаңа файл ашу Save F2 – файлды дискіге жазып сақтау Save as ... – файлды басқаша түрде жазып сақтау Save all – барлығын да жазып сақтау Change dir – директорийды өзгерту Print – ашық терезедегі файлды қағазға басу

```
Dos shell  –   DOS-қа (Turbo C-ден) уақытша шығу
```

Exit Alt-X – Turbo C-ден шығып кету Файлды дискіге өзіңіз қойған атпен сақтауға арналған Басқаша сақтау (Save as – Сохранить как...) терезесі төмендегі 17.2-суретте көрсетілген.

Undo (болдырмау) Alt + Bksp – соңғы орындалған команданың әрекетін болдырмай алып тастайды. Redo Shift +Alt + Bksp – Undo командасының кері қайтарған командасы әрекетін қайтадан іске қосады. Cut (қиып алу) Shift+Del – белгіленген бөлікті буферге қиып алады (бұрынғы орнында қалмайды). Copy (көшіру) Ctrl+Ins – белгіленген бөлікті буферге көшіреді (бұрынғы орнында сақталады). Paste (кірістіру) Shift+Ins – курсор орналасқан жерге буфердегі ақпаратты кірістіріп қояды, яғни енгізеді.

```
Clear (өшіру) Ctrl+Del – белгіленген бөлікті өшіру.
```

Copy Examples – мысалды көшіру. Show Clipboard (буферді ашу) – редактор терезесінен буферге алынған мәтінді сақтайтын терезені ашады. Search менюінің ішкі командалары:

```
Find (табу)  –  табуға қажетті сөзді енгізу мүмкіндігін    беретін сұхбат терезені ашады.
```

Replace (орын алмастыру) Alt+S+R – іздейтін мәтін мен оны алмастыратын мәтінді енгізу мүмкіндігін беретін сұхбат терезені ашады. Search Again (қайтадан іздеу) Ctrl+L – Find немесе Replace командаларының соңғы әрекетін қайталайды. Go to line number (нөмір қатарына бару) – курсорды нөмірі көрсетілген қатарға орналастырады. Previous error Alt+F7 – алдыңғы қате орнына бару. Next error Alt+F8 – келесі қате орнына бару.

Run менюінің ішкі командалары: Run (орындау) Ctrl+F9 – орнатылған параметрлерді қолдана отырып, редактор терезесіндегі екпінді бағдарламаны орындайды. Program reset Ctrl+F2 (сброс программы) – жөндеушінің орындап жатқан әрекетін тоқтатып, бағдарламаға бөлінген орынды босатып, барлық ашық файлдарды жабады. Go to cursor (курсорға өту) F4 – екпінді терезедегі бағдарламаны курсор тұрған орындағы қатарға дейін орындайды. Trace into (қосалқы бағдарламаға кіріп, қадамдық орындау) F7 – програм-мадағы операторларды қадам бойынша көрсете отырып орындайды. Step over (бағдарламаны қадам бойынша орындау) – бағдарлама мәтінінің бір жолына сәйкес келетін кезекті операторды біртіндеп орындайды.

```
Compile менюінің ішкі командалары (17.4-сурет):
```

Compile Alt+F9 – екпінді терезедегі бағдарламаның қатесін тексереді. Синтаксистік қате жөнінде хабарлама береді де, курсор қате жіберілген орынға орналасады. Қате жоқ болса, компилляцияның сәтті болғаны жөнінде хабарлама береді.

Бағдарламалау жүйесінің интегралданған мүмкіндіктері мен түсініктері. Make (бағдарламаны жинақтау). Егер негізгі бағдарламада немесе негізгі модульде қолданылатын жеке модульдердің мәтінінде объектілік файлды алғаннан кейін өзгеріс болса, соған сәйкес модульдер қайта тексеріледі, одан кейін негізгі бағдарлама немесе негізгі модульден тұратын файл қайта құрылады. Build all (Бағдарлама құру) – бұл команда орындалғанда, негізгі бағдарлама және негізгі модульде қолданылатын барлық модульдер қайта компиляциядан өткізіледі.

```
Debug (жөндеу) менюі ішкі командалары:
```

Inspect... (Alt+F4) – Inspector терезесін ашады, ол объектілер мәнін талдауға көмектеседі. Evaluate/modify... (Ctrl+F4) үш өрісі бар терезе ашады: Expression, Result, New value — олар айнымалы мәндерін көріп, оларды өзгерту мүмкіндігін береді. Call stack (Ctrl+F3) – бағдарламада қолданылған функциялар тізбегін – стекті көрсететін қосалқы бағдарлама

терезесін көрсетеді. Watches 8 суырылып шығатын менюді ашып, жаңа өрнектер енгізіп, оның нәтижесін көрсете алады. Breakpoints... – түзету режимінде тоқтау нүктесімен жұмыс істеу мүмкіндігін беретін терезе ашады. Project командалары қажет болғанда, жобалар ашу, толықтыру және жабу ісін атқарады. Жоба — бір-бірімен байланысты файлдар жиыны, бірнеше объектілік файлдар бірден компиляциядан өткізіліп, атқарылатын бір бағдарлама жасайды. Жоба көп файлды бағдарламалар кезінде керек. Кейде бір файлмен жұмыс істегенде де қолданылады. Модульдік бағдарламалау барысында көпфайлдық компиляциялаусыз жұмыс істеу мүмкін емес. Көлемі үлкен бағдарламалармен жұмыс істеу кезінде ол бағдарламаның бөліктерін бірнеше файлдарға сақтау анағұрлым ыңғайлы. Әрбір файл бүтіндей бір немесе бірнеше функцияны қосуға тиіс. Ол файлдардың аттары арнайы файл-жобаға жазылады, IDE ол ақпараттан мәтіндік файлдардың қайсысын орындалатын файлға (.EXE) біріктіру керек екендігінен хабардар болады. Файл-жобалармен жұмыс істеуге қажетті бұйрықтардың барлығы Projectмәзіріне қосылған. Файл-жобаларды ұйымдастыру үшін ол файл-жобаны ашу керек. Ол үшін Project\Open Project... командасы орындалады және IDE экранның төменгі жағындағы арнайы Project терезесін іске қосады. Қажетті файл-жобаны жүктейтін немесе берілген атпен жаңа файл-жоба құратын диалог терезесі ашылады. Жаңа файл-жоба құрылған болса, Project терезесі бастапқыда бос болады. Жобаға файлдарды қосу немесе оларды жою Project\Add item... және Project\Delete item бұйрықтарымен орындалады. Меңзер (курсор) Projectтерезесінде орналасқан жағдайда осы мақсатта Ins және Del батырмаларын басса да жеткілікті. Файлдарды жобаға қосу кезінде қажетті файлды таңдауға мүмкіндік беретін диалог терезесі ашылады. Project терезесі жобаға қосылған бір файлдан оларды редакциялау барысында келесісіне көшуді жеңілдетеді. Ол үшін Project терезесіндегі қажетті жолға меңзерді (курсорды) апарып ENTER пернесін басса жеткілікті. Borland C++ ортасында жұмыс істеу кезінде прог-рамма бір ғана файлдан тұрғанның өзінде жобаны қолданған ыңғайлы.

Options менюінің командалары (17.5-сурет) Турбо С ортасының келісім бойынша тағайындалатын параметрлерін көру және оларды өзгерту мүмкіндігін береді. Олардың көптеген мәндерін өзгертпей, қалдыруға болады. Мұнда түйінді сөздер түсін (16 түс) өзгерту мүмкіндігі бар (Options/Enviroment/Colors/Syntax). Directories командасы тақырыптық файлдар каталогын (Include Directories),кітапханалық функциялар (Library Directories) каталогын және терілген файлдар мен олардың нәтижелік файлдары (Output Directory) қайда орналасатынын көрсетіледі. Мысалы, егер TC бағдарламалары C:\TC каталогында орналасса, онда Include Directories өрісінде C:\TC\INCLUDE деп көрсеткен дұрыс болады, ал Library Directories өрісіне — C:\TC\LIB деп жазу керек. Output Directoryжолына нәтижелік файлдар орналасатын каталогты, мысалы, C:\TC\BIN\STUDENT деп көрсеткен ыңғайлы болып саналады немесе C:\TC\BIN каталогын қалдыру үшін — нүкте —.‖ енгізе салу керек. Керекті параметрлер енгізілген соң, оларды Options – Save... командасымен есте сақтап қою қажет. Window (Терезе) меню командалары терезені ашу, жабу, экранды жылжыту әрекеттерін орындау мүмкіндігін береді.

Help (көмек) меню командалары жүйедегі анықтамалық ақпаратты оқу мүмкіндігін береді. Contents (экранға шығарылған ақпарат жөнінде мәлімет) ағымдағы уақытта экранға шығарылған мәлімет жөнінде (екпінді терезе, таңдалған меню командасы, жіберілген қате) мәліметті сұхбат терезеге шығарады. Index (түйінді сөздер) Shift+F1 – жүйеде бар барлық анықтамалық ақпарат тізімін әліпбилік ретімен түйінді сөздер бойынша шығарады. Topic search (сөз бойынша іздеу) Alt+F1 – курсор орналасқан сөз жөнінде анықтамалықты шығарады. Егер сол сөз жөнінде анықтамалық жоқ болса, алғашқы символдарының саны көп сәйкес келетін түйінді сөздер тізімін береді. Prevoius topic (алдыңғы тақырып) алдыңғы сұранысқа сәйкес келетін анықтамалықты шығарады. Жүйе 20 сұранысты сақтай алады.

Интегралданған бағдарламалау жүйесінде жұмыс істеу нәтижелері. С тіліндегі бағдарлама құрылымы төмендегідей болады: #препроцессор директивалары . . . . . #препроцессор директивалары

```
функция a()
{операторлар}
функция b()
{операторлар}
void main ()   // бағдарламаның орындалуын бастайтын функция
```

{операторлар: сипаттау операторлары меншіктеу операторлары функция бос оператор құрама операторлар таңдау операторлары цикл операторлары көшу операторлары } Препроцессор директивалары – бағдарламаны компиляциядан өткізгенге дейінгі түрлендіру ісін басқарады. C/C++ тілдерінде мәтіндік файл түрінде даярланған бастапқы бағдарлама түрлендірудің 3 кезеңінен өтеді:

```
1)  мәтінді препроцессорлық түрлендіру;
2)  бағдарламаны компиляциядан өткізу;
3)  біріктіру (байланыстарды түзету және жинақтау).
```

Осындай үш кезеңнен өткен соң (5.1-сурет), бағдарламаның орындалатын екілік коды қалыптасады. Препроцессордың міндеті – бағдарлама мәтінін компиляцияға дейін түрлендіру. Препроцессорлық өңдеу ережелерін бағдарламалаушы препроцессор директивалары көмегімен анықтайды. Директива # таңбасынан басталады. Мысалы:

1. #define – бағдарлама мәтініндегі алмастыру ережелерін көрсетеді. #define ZERO 0.0 Соңғы жол бағдарламадағы ZERO сөзінің әрқайсысы 0.0 санына ауысатынын білдіреді.
2. #include <тақырыптық файл аты> – бағдарлама мәтініне стандартты кітапханамен бірге берілетін «Тақырыптық файлдар» каталогынан мәтін қосылатынын білдіреді. Тақырыптық файлдардың біреуінде С тілінің әрбір кітапханалық функциясының атына сәйкес сипаттамасы болады. Тақырыптық файлдар тізімі тіл стандартымен анықталған. Include директивасын пайдалану соған сәйкес стандартты кітапхананы іске қоспайды, ол тек көрсетілген тақырыптық файлдан алынатын керекті сипаттамаларды бағдарлама мәтініне енгізуге мүмкіндік береді. Кітапханадан алынатын қажетті кодтар бағдарламаға компиляциядан кейін орындалатын біріктіру кезеңінде қосылады. Тақырыптық файлдарда стандартты функциялардың барлығының да сипаттамалары болғанмен, бағдарлама кодына тек соның ішінде қолданылатын функциялар ғана кірістіріледі. Препроцессорлық өңдеу кезінде бағдарлама мәтініндегі препроцессор директивалары (#include, #define) анықталып, бағдарламаға тақырыптық файлдар каталогынан мәтіндік файлдар қосылады немесе кейбір сөздерді алмастыру орындалады. С/С++ тілдерінің стандарты бойынша анықталған функциялар бір тақырыптық файлда анықталады. Препроцессорлық өңдеу орындалғаннан кейін бағдарлама мәтінінде бірде бір препроцессорлық директива қалмайды. Компиляция кезінде бағдарлама компьютерге түсінікті кодтарға түрлендіріледі. Егер осы түрлендіру кезінде стандартқа сәйкес келмейтін қателер кездессе, оны компилятор бірден көрсетеді. Ал қате жоқ болса, компилятор объектілік код немесе объектілік модуль болып табылатын бағдарлама мәтінін береді. Барлық бағдарлама бөліктері – функциялар компиляциядан өткен соң, объектілік модульдер біріктіргішке (компоновщикке) беріледі. Ол модульдерді біріктіріп, оған стандартты кітапхана функцияларын қосады да, функцияда қате болса, соны анықтайды. Біріктіргіш жұмысы нәтижесінде бағдарламаның екілік сандар түрінде жазылған атқарылатын машиналық коды жасалады да, ол орындалып жұмыс нәтижесін береді. Барлықбағдарламалау тілдері осы схемамен жұмыс істейді.

БАҒДАРЛАМА СИПАТТАМАЛАР МЕН АНЫҚТАМАЛАРДАН, ОПЕРАТОРЛАРДАН ҚҰРАСТЫРЫЛҒАН БІРНЕШЕ ФУНКЦИЯЛАР ЖИЫНЫНАН ТҰРАТЫН МӘТІН ТҮРІНДЕ БОЛАДЫ. СОЛ ФУНКЦИЯЛАР ІШІНДЕ MAIN АТТЫ БАСТЫ ФУНКЦИЯ МІНДЕТТІ ТҮРДЕ БОЛУЫ ТИІС.

```
Ондай функциясыз бағдарламаорындалмайды. Функция атының алдында сол функция қайтаратын мәннің типі (нәтиже типі) көрсетіледі. Егер функция ешқандай нәтиже қайтармайтын болса, онда void типі көрсетіледі, мысалы: void main(). Әрбір функцияның, соның ішінде main функциясының да, параметрлері болуы тиіс, бірақ олар жоқ та болуы мүмкін, ондай жағдайда жақша ішінде (void) сөзі көрсетіледі.
```

Функция тақырыбынан соң, жүйелі жақша ішінде оның операторлары, яғни ішкі тұлғасы (денесі) орналасады. Функция тұлғасы дегеніміз – анықтаулар, сипаттамалар, орындалатын операторлар жиынынан тұратын, жүйелі жақшаға алынған символдар тізбегі. Әрбір анықтама, сипаттама немесе оператор нүктелі үтірмен аяқталады. Анықтаулар – бағдарламадағы өңделетін мәліметтерді бейнелеуге қажет объектілер (объект – компьютер жадының ат қойылған аймағы, мысалы, айнымалы) енгізеді. Анықтаулар төмендегідей түрде болады: int y = 10; // бүтін сан түріндегі ат қойылған константafloat x; // нақты (аралас сан түріндегі) айнымалымұндағы // белгілерінен кейін орналасқан сөздер түсініктеме рөлін атқарады да, бағдарламаға еш әсерін тигізбейді. Сипаттамалар – компиляторға бағдарламаның басқа бөліктерінде жазылған объектілер мен функциялардың аттары және қасиеттері жайлы мәлімет береді. Операторлар – бағдарламаның орындалатын әрбір қадамында қандай іс-әрекеттер атқарылатынын анықтайды. С бағдарламасының мысалы:

```
#include <stdio.h>    // препроцессорлық директива
void main()      // функция тақырыбы
```

{ // функция тұлғасының басы

```
printf(—Hello!‖);  // экранға Hello! сөзін шығару
```

} // соңы

Тәжірибелік сабақ 14-15. Функциялар мен әдістер. Тапсырма тәсілдері. Сипаттау.

Символдық айнымалыларға әрекеттер жасау үшін Си тілінде string.h тақырыптық файлда анықталған функциялар қолданылады. Символдық айнымалылар былайша сипатталады: Char айнымалылар тізімі; Символдық жол символдық массив түрінде сипатталады. String.h тақырыптық файлда төмендегідей функциялар анықталған.

1. strlen() – функциясы жолдың ұзындығын табу үшін қолданылады, функция нәтижесінде бүтін сан шығады.
2. strcad ()- функциясы 2 жолды біріктіру үшін қолданылады. Жалпы жазылуы:strcad (1-жол, 2- жол); Функция 2-жолды 1-жолға біріктіреді.
3. strepy () функциясы жолдық айнымалыны басқа жолдық айнымалыға түгелдей көшіру үшін қолданылады. Функция str басынан 0-дік символға дейінгі символдарға санын қайтарады. Бұл функцияның қолданылуы 1- ші листингте көрсетілген. Жолдарды көшіру С кітапханасында жолдарды көшірудің екі функциясы бар. Спецификациялық техниканың әсерінен жолдармен жұмысты С-да бір жолды екіншісіне оңай меншіктеуге болмайды, бірақ басқа бағдарламалау тіліндерінде бұлай жасауға жол берілген. Міндетті түрде орындалушы жолдың өзі алып орналасқан жерге жолдарды көшіру функциялары strcpy () және strncpy () дан тұрады. Екеуі де string.h тақырып файлында хабарланған, оны міндетті түрде олардың қолданылуы үшін қосу қажет. Strcpy () функциясы Strcpy () кітапханалық функциясы берілген жадыда бүтін бір жолды көшіреді. Оның протипі:char * strcpy(char * destinution, const char*source); Анықтама бойынша, Strcpy () функциясы жолдардыкөшіруді орындайды және ол soyrce адресінде орындалады (оған қоса аяқтаушы ()-дік символмен) б сілтеуіште destination адресінен басталады. Strcpy () функциясын шақырмайтын бұрын жаңа жол үшін шартты түрде жадыны ерекшелеу керек. Жаңа сілтенген адрес бойынша жеткілікті жады бар немесе жоқ екендігін функция өздігінен тексере алмайды. Егер жады реттеуді орындамаса, онда функция strlen(source) жаңа ақпаратқа әсер етеді. Оның ішіде destination сілтенген адрестен бастайды. Мұндай жағдайда күтпеген қателіктер шығуы мүмкін. Strcpy () функциясының қолданылуы 2-ші листингте айқын көрсетілген. Strncpy () функциясы Strncpy () функциясы strcpy() функциясымен белгіленуі жөнінен сілтейді. Бір еске сақтар жағдай бар, оның көмегімен белгілі бір мөлшерде символдар көшіріледі. Ол төмендегідей прототипке ие: Char * strncpy (char * destination, const char* source, size_t n); Source және destination аргументтері бастапқы және соңғы жолдарға сілтейді. Функция destination –дағы source аргументтерінен n-нен астам символдарды көшіреді. Егер source жолы n символдан қысқа болса, онда destination да n-ге тең болуы, көшірілуі үшін, оған жеткілікті мөлшерде нөлдік символдар қосылады. Егер source жолы n символынан ұзынрақ болса, онда destination-ға аяқталушы нөлдік символ қосылмайды. Функция destination сілтеуішін кері қайтарады. Strdup функциясы Бізге берілген жөн, тағы бір жолдарды көшіру функциясыбар. Оның аты strdup().Берілгендері бойынша strdup() функциясына ұқсас, бірақ буфер үшін жадының реттелуін орындайды. Нақты алғанда, ол тура соны жасайды. Біз оны өз күшімізбен 2- кейін strdup() жолды көшіру үшін. Білген жөн, strdup() функциясы ANSI стандартында анықталмаған. Ол компилятордың кітапханасына кіреді. Мысалы, Microsoft Borland және Symantec C, бірақ басқа да компиляторлар оны жүзеге асырап алады. strdup() келесідей типке ие: Char *strdup(char*source); Source аргументі жолдарды көшуруші сілтеуішті көрсетеді. Функция жолға сілтеуішті қайтарады, онда орындалған буферге көшіру ma11oc() немесе NULL көмегімен құру, оның ішінде жадының реттелуі сәтті болмаған жағдайда ғана. strdup() функциясының қолданылуы source 4-ші листингте көрсетілген. Конкатенация (жолдардың тіркесуі немесе айқасуы) Егер сіз әлі конкатенация терминін кездестірмеген болсаңыз (ол “тіркесу” немесе “айқасу” мағынасын береді), онда сізде бұл не?,ол заңды ма?-деген сұрақ тууы мүмкін. Бұдан былай жауап береміз: конкатенация кезінде бір жол екінші жолдың соңына жай ғпнп жплғпнпды, бұл үшін сіздерді құрдымға сүйреп кетпейді. С стандартты кітапханасында 2 жолды айқастырушы функциялар бар. Олар strcat() немесе strncad(). Олармен жұмыс істеу үшін міндетті түрде string.h тақырыптық файлы қажет. Strcat () функциясы Strcat () функциясы төмендегідей прототипке ие: Char * strcat (char*srr1,const char*str2) Бұл функция str1 жолының соңына str2 жолының көшірмесін сыйғызады және жаңа алған жолдың соңына апарып аяқтаушы нөлдік символды қояды. Алдын- ала міндетті түрде екі жолдың тіркесуі немесе айқасуының шешімін сақтау үшін, str1 жолында жеткілікті орын болуын қамтамасыз етіңіз. Strcat () функциясы str1- ге сілтеуішті кері қайтарады. Оның қолданылуы төмендегі 5-ші листингіде

бейнеленген. Strncat() функциясы Strncat() кітапханалық функциясы сонымен қатар, жолдардың конкатенциясын орындайды, бірақ шығушы жолдан белгіленген жолдың соңына тура қанша символ қосу керек екендігін сілтеу керек. Ол төмендегідей прототипке ие: Char *strcat (char*str1, const char*str2,size_t n); Егер str2 n-нен астам символға ие болса, онда str1-ге тек алғашқв п символы қосылады. Әр осындай жағдайда жолға нөлдік аяқтаушы символ автоматты түрде қосылып отырады. Міндетті түрде str1-ге жеткілікті жадының үлестірілуін қамтамасыз еткен жөн, себебі оның ішіне нәтиже сыюы қажет. Функция str1-ге сілтеуішті кері қайтарады. 6-шы дистингте Strncat() функциясының көмегімен тура 5-ші дистингтегідей нәтижеге қол жетеді. Жолдарды салыстыру Жолдарды өзара тең немесе тең емес екендігін анықтау үшін салыстырылады. Егер жолдар өзара тең болмаса, онда олардың біреуі ұзынрақ, ал екіншісі қысқарақ болады. Қысқарақ және ұзынрақ анықтамалары ASCII символдардың кодтарына сүйекнеді. Кодтардың кезектесуі тура алфавиттегі реттік бойынша, бірақ бір өзгеше қасиет бар- ол жоғарғы регистрдің әріптері, төменгі регистрдің әріптерінен қысқарақ. Екі бүтін жолды салыстыру Strcmp() функциясы екі жолды символ бойынша салыстыруға арналған. Оның прототипі: Int strcmp(const char*str16const char*str2) str1 және str2 аргументтері салыстырылып жатқан жолдарға сілтейді. Кері қайтарушы функцияның мағынасы 1-ші кестеде берілген назар аударыңыз, екі жолдық аргумент те константалармен хабарланған. Себебі олардың біреуі функция ішінде өзгермейді. Strcmp() функциясының қолданылуы 7-ші листингте

Жолдардың фрагментін салыстыру Strcmp() кітапханалық функциясы бір жолдан екінші жолға берілген ұзындықтан фрагментін салыстырады. Ол төмендегідей прототипке ие:

```
Int strncmp(const char* str1,const char*str2,size_t n)
```

Бұл функция str1 жолымен бірге str2 жолынан п символды салыстырады. str1-дің соңына жетіп немесе барлық п символдар тамамдалғанша салыстыру жалғаса береді.Кері мағынасы және салыстыру әдісі тура strncmp() Функциясындай. Жолдар регистірдің тіркеуімен салыстырылады. Бұл функцияның қолданылуы 8-ші листингте көрсетілген. Екі жолды регистрдің тіркеуінсіз салыстыру Өкінішке орай, ANSI C стандартты кітапханасында регистрдің символдарының айырмашылығынсыз жолдарды салыстыру функциясы жоқ. Бірақ көп тараған C компиляторлары бұл мақсат үшін стандартты емес функцияларды көрсетеді. Мысалы, Symantec компилятор strncmp1() функциясын ұсынады, ол Microsoft кітапханасында бұл жағдайда _stricmp() функциясы барады, ал Borland кітапханасында бірден екі функция strmpi() stricmp() бар.

Параметрлерді жіберу. Шақыру тәсілдері. Енгізу- шығару функциялары puts – Синтаксис:

```
puts (const char* қатар)
```

Экранға символдар қатарын шығарады және курсорды экранның келесі қатарының басына алып келеді. Функция параметрі ретінде қатарлық тұрақтыны немесе қатарлық айнымалыны қолдануға болады. Тақырып файлы: <stdio.h> gets Синтаксис:

```
char *gets (char* s);
```

Клавиатурадан символдар қатарын енгізеді. Енгізілетін қатарлар бос орыннан тұруы мүмкін. Тақырып файлы: <stdio.h> putch Синтаксис:

```
int putch (int c)
Символды экранға шығарады. Тақырып файлы:<conio.h>
```

- Мәтінді экран бетіне түрлі түсте шығару үшін cprintf және cputs функцияларын қолдану керек.
- Жаңа жолға өту \n\r –түрінде беріледі.
- cprintf және cputs функцияларымен шығарылатын символдар түсі textcolor (түс) функциясы арқылы орнатылады.
- Түстің фонын textbackground (Түс) орнатады.
- Clrscr, textcolor және textbackground функцияларын қолдану үшін бағдарлама мәтініне # include <conio.h>директивасын қосу қажет.