ҚАЗАҚСТАН РЕСПУБЛИКАСЫ
БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

ҚАЗАҚСТАН
тәуелсіздігіне
20
ЖЫЛ

Б. Бөрібаев

ПРОГРАММАЛАУ ТЕХНОЛОГИЯЛАРЫ

Алматы, 2011

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ
БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Б. Бөрібаев

ПРОГРАММАЛАУ ТЕХНОЛОГИЯЛАРЫ

Оқулық

Қазақстан Республикасы
Білім және ғылым министрлігі бекіткен

Алматы, 2011

ӘОЖ 004.42(075.8)
КБЖ 32.973-018я73
Б 79

Пікір берушілер:

техника ғылымдарының докторы, профессор **А. Ж. Сейкетов**;
физика-математика ғылымдарының докторы, доцент **Ш. Ә. Жомартова**.

Бөрібаев Б.

Б 79 **Программалау технологиялары:** Оқулық. – Алматы: ЖШС РПБК «Дәуір», 2011. – 352 бет.

ISBN 978-601-217-280-5

"**Программалау технологиялары**" оқулығы жоғарғы оқу орындарындағы техникалық бағытта білім алып жатқан студенттерге арналған, мұнда С және С++ тілдерін негізге ала отырып программалау тәсілдерінен теориялық мағлұматтар беріліп, әрбір тақырып бойынша жинақталған тапсырмаларда студенттердің өз беттерімен орындауларына арналған есептер келтірілген. Ұсынылып отырған оқулық С/С++ тілдерінде программалауды өз бетінше оқып үйренгісі келетін оқырмандардың да қажетіне жарайды деген сенімдеміз.

ӘОЖ 004.42(075.8)
КБЖ 32.973-018я73

ISBN 978-601-217-280-5

КІРІСПЕ

Өткен ғасырдың 80-ші жылдары басында UNIX операциялық жүйесінің басқаруымен жұмыс істейтін компьютерлер әлемінде ең белгілі тілдің бірі С (си) тілі болды. Содан бері ол дербес компьютерлер мен мейнфреймдерде (үлкен компьютердер) жұмыс істейтін негізгі тілдердің біріне айналды. Программалық жабдықтамалармен айналысатын көптеген фирмалар бұл тілді мәтіндік процессорларды, электрондық кестелерді, компиляторларды құру үшін кең қолданып келеді. Соңғы кездерде C/C++ тілдері ең кең таралған тілдерге айналды.

C/C++ тілдерінің артықшылықтарына – ондағы программалардың тез орындалуы, мәтінінің қысқа болуы, түйінді сөздерінің аз болуы, тс.с. жатады. Бұл тілдердің ассемблерге ұқсас басқару мүмкіндіктері де бар. Программаларды өз қалауыңызша ең жоғары орындау жылдамдығына немесе жадты тиімді пайдалануға үйлестіруіңізге болады. Белгілі бір операциялық жүйеге арнап жазылған программаларды басқа жүйелерге кедергісіз немесе аздаған өзгертулер енгізе отырып көшіре аласыз. Ол аппараттық жабдықтамалар жұмысына тікелей араласып, жедел жадтың жекелеген биттерімен арифметикалық және логикалық амалдар орындау мүмкіндігін туғызады. Функциялардың көлемді кітапханасы программалаушылар алдында тұрған күрделі мәселелерді жеңіл және жылдам шешуге мүмкіндік береді.

C++ тілінде программалауды үйрену компьютерлік технологияларды игергісі келетін әрбір маманның алғашқы қадамдарының бірі болуы тиіс. Олай дейтініміз C/C++ және осыларға ұқсас тілдер қазіргі кезде үздіксіз даму үстінде. Сол себепті назарларыңызға ұсынылып отырған оқулықтың алдына қойған мақсаты студенттерді тиімді де қысқа мәтінді программалар жазуға және олардың нәтижелерін алып, түрлендіру істеріне машықтандыру болып табылады. Оқу құралы тұтынушының программалау дәрежесінің кез келген деңгейіне арналған. Программалауды енді үйрене бастаған студенттер үшін ол оқулық болса, тәжірибелі программалаушылар үшін анықтамалық құрал ретінде пайдаланылуы мүмкін.

Бұл оқулық өмір талабына сай жазылып, жаратылыстану ғылымдары мен техникалық салада білім алу барысында жеке оқу пәндері болып қалыптасқан **Алгоритмдер және мәліметтер құрылымы, Алгоритмдеу және программалау тілдері, Программалау технологиялары, Объектіге бағытталған программалау** сабақтарында қолдану үшін шығарылып отыр. Мұнда алгоритм құру жайлы түсініктер беріліп, қазіргі кездегі кең тараған C/C++ тілдерінің ерекшеліктері қарастырылып, есептерді компьютерде орындау шаралары жүзеге асырылады.

Оқулықта теориялық мағлұматтардың көптеген нақты мысалдар мен жаттығулар арқылы түсіндіріліуі кітаптың негізгі бір ерекшелігі деуге болады.

Мемлекеттік тілімізге әлі информатика терминдері толық аударылып, олардың стандарты бекітілмегендіктен, кітап соңында қолданылған терминдердің бірсыпыра аудармасы келтірілген.

Қазақ тілінде жарық көріп отырған бұл оқу құралы төл тілімізде дәріс алып жатқан студенттеріміз үшін программалау негіздерінің қыры мен сырын меңгеру жолында көмекші құрал бола алады деген сенімдеміз.

1 ДЕРБЕС КОМПЬЮТЕРЛЕРДІҢ ПРОГРАММАЛЫҚ ЖАБДЫҚТАМАЛАРЫ

XX ғасырдағы ғылым мен өндірістің жедел өркендеп дамуы адамға керекті ақпараттың тез артуына әкелді. Бұдан ақпаратты жіктерге (кластарға), тақырыптық топтарға бөліп, оларды сақтау, қажет кезінде шапшаң іздеп таба білу және ақпараттың өзгеру заңдылықтарын зерттеу қажеттігі туды. Осы мәселелерді ғылыми тұрғыдан зерттейтін техника саласын алдымен кибернетика, ал кейіннен оның ақпарат өңдеуге қатысты бөлігін информатика деп атады. Бұл сөз француз тіліндегі "Informatigue" деген ұғым атауынан шыққан, ал ағылшын тілінде сөйлейтін елдердегі "Computer science" деп аталатын ғылым саласы да осы информатика сөзінің баламасы болып табылады.

Сонымен, ақпаратты қабылдау, сақтау, түрлену және тасымалдау жолдарын зерттейтін *информатиканың* негізгі аппараты – электрондық есептеуіш машинасы, яғни компьютер болып саналады. Сол себепті компьютерді зерттеу ілімдерін де информатика деп түсінген жөн. Информатика 1.1-суретте көрсетілген үш бөліктен тұрады.

1.1-сурет. Информатиканың құрамы

Техникалық жабдықтамалар (ағылшынша hardware) мен программалық жабдықтамалар (ағылшынша software) әрбір оқулықта егжей-тегжейлі түсіндірілетін болса, ал осы екеуінің негізі болып есептелетін алгоритмдік бөлік соңғы кезде ғана толық түрде бір жүйеге келтіріліп, информатиканы оқытудың өзекті мәселесі ретінде кеңінен қарастырыла бастады.

Егерде *информатиканың* мағынасын толық ашатын болсақ, оны

1) ақпаратты автоматты түрде жинау, сақтау, іздеу және тасымалдау,

2) ақпаратты өңдеу және түрлендіру тәсілдерін жасау (ойлап табу),

3) ақпаратты өңдейтін электрондық есептеуіш техникасын дамыту жолдары деп түсіну керек.

Қазіргі кезде кез келген ғылым саласында өте көп ақпарат жинақталған. Мысалы, физика, химия, математика, т.б. салаларда өте көп формулалар, ұғымдар, схемалар бар. Оларды оқып-үйрену үшін информатиканы кеңінен қолдануға болады. Осы тұрғыдан алғанда информатиканы оқытудың негізгі мақсаты – компьютерлердегі ақпаратты сақтау, өңдеу жолдарымен танысу және үйрену; қалаған мамандықты игеру жолында кездесетін есептерді шығару үшін компьютерді кеңірек қолдана білу. Сондықтан есептер шығару кезінде информатиканың алгоритмдік және программалық бөліктерінің атқаратын маңызы зор. Бірақ бұл екеуінің жұмыс нәтижесін компьютерсіз алу мүмкін емес.

Енді информатиканың программалық бөлігін толығырақ қарастырайық.

1.1 Программалар мен олардың түрлері

Компьютердегі информация екі үлкен топтан тұрады:

Программалар *– алгоритмдердің компьютерге түсінікті командалар немесе операторлар тізбегі түрінде жазылған нұсқасы.*

Мәліметтер *(данные, операнды) – программалар өңдейтін немесе солардың жұмысы нәтижесінде пайда болатын информациялар жиыны, құжаттар.*

Кез келген программаның міндеті, атқаратын жұмысы – аппараттық құралдарды басқару.

Компьютерде программаның дұрыс жұмыс істеуі үшін ол қатесіз, дұрыс құрастырылып, өзінің жеке түсініктеме құжаты (документация) болуы тиіс. Сондықтан оны программалық жабдықтама деген терминмен атайды.

Компьютердің программалық жабдықтамасы құрамы **программалық конфигурация** деп аталады.

Программалық жабдықтама қызметі:

– компьютер жұмысын қамтамасыз ету;

– тұтынушы мен компьютер байланысын жеңілдету;

– есептеу жүйесінің мүмкіндігін кеңейту;

– құрылғыларды пайдалану тиімділігін арттыру;

– адамның жұмыс сапасы мен өнімділігін көтеру.

Программа жасауға қатысатындар:

– *Тұтынушы, пайдаланушы (user) – есепті, мәселені қоюшы*. Ол программа жазбайды, бірақ қай есепті шығарып, қандай нәтиже алу керектігін біледі.

– *Программалаушы – программа жасаушы, яғни оны жазатын адам.* Ол тұтынушының талабына сәйкес компьютерге нұсқаулар дайындап, соларды машина командалары ретінде жазып шығады.

– *Компьютер – командаларды атқарушы, орындаушы құрал.* Ол командаларға сәйкес тұтынушы мәліметтерін жылдам өңдейді. Мәліметтер мен программаларды өзіне түсінікті екілік (он алтылық) жүйеге алмастырып, программаны орындап, нәтижесін алып, оны да екілік жүйеден тұтынушыға түсінікті тілге түрлендіріп береді. Дербес компьютер деп бір адамның жеке пайдаланатын есептеу машинасын айтады.

Программалар жазу үшін үш тіл қажет:

1. Машиналық кодтар тілі – компьютерге түсінікті тіл (екілік және 16-лық кодтар түрінде жазылады). Ассемблер тілі осыған жақын.

2. Программалау тілі – программалаушыға түсінікті тіл (мәтін, сөз, команда түрінде жазылады). Бейсик, Паскаль, C/C++, Ява (Java) тілдері.

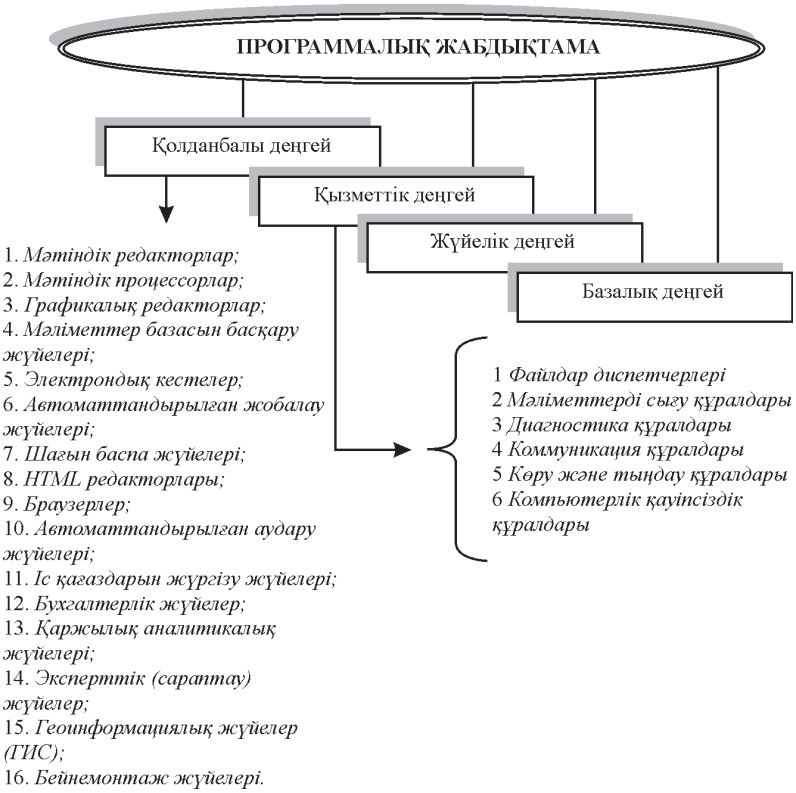
3. Бейнелер (образдар) тілі – тұтынушыға түсінікті тіл (мәтін және графика), яғни қарапайым кітап тілі деуге негіз бар. Біз екінші топты қарастырамыз.

1.2 Программалық жабдықтама деңгейлері

Программалар арасында өзара байланыс болады, яғни көптеген программалар жұмысы төменгі деңгейдегі программалар жұмысына негізделіп жасалады.

Программаралық интерфейс – программалардың бір-бірімен байланысты бірнеше деңгейлерге бөлінуі.

Программалық жабдықтама деңгейлері пирамида тәріздес, жоғары деңгей алдыңғы (төменгі) деңгейлерге сүйенеді.



1.2 сурет. Программалық жабдықтама құрылымы

Базалық деңгей

Базалық деңгей – ең төменгі деңгей. Ол негізгі аппараттық құралдармен байланысу ісін атқарады. *Базалық программалық жабдықтамалар базалық аппараттық құралдар құрамындағы тұрақты жады микросхемаларында жазылып сақталады* да, базалық енгізу-шығару жүйесін (BIOS) құрайды. Программалар мен мәліметтер аппараттарды зауыттан шығару кезінде микросхемаларға жазылады, кейіннен өзгертілмейді.

Жүйелік деңгей

Жүйелік деңгей – өтпелі деңгей. Бұл деңгей программалары басқа программаларды төменгі базалық программалармен және

тікелей аппаратурамен байланыстырады. Осы программаларға байланысты компьютердің қызмет ету көрсеткіштері қалыптасады.

Компьютерге жаңа құрылғы қоссақ, сол құрылғының *жүйелік деңгей программасы* басқа программаларды осы құрылғымен байланыстырады. Бір құрылғыны осылай басқа программалармен байланыстыру программасы **драйвер** деп аталады (принтер, тышқан, диск, дисплей драйверлері болады).

Тағы бір жүйелік деңгей программасы құрылғыны адаммен байланыстырады. Соның көмегімен компьтерге мәлімет енгіземіз, басқарамыз, нәтиже ала аламыз. Бұлар тұтынушы интерфейсін басқару құралдары, осылар компьютер жұмыс өнімділігін, тиімділігін қамтамасыз етеді.

Жүйелік деңгейдегі программалар жиыны компьютер **операциялық жүйесінің (системасының) ядросын** құрайды. Операциялық система (ОС) ядросының болуы адамның компьютермен жұмыс істеуінің алғашқы шарты.

ОС ядросы компьютер жадын, енгізу-шығару процесін, файл жүйесін, командаларды өңдеуді басқару, т.б. жұмыстар атқарады.

Қызметтік деңгей

Бұл деңгей программалары базалық деңгей программаларымен де және жүйелік деңгей программаларымен де байланыса алады.

Қызметтік программалар **(утилиттер**) жұмысы – компьютерлік жүйені тексеру мен баптау (параметрлерін тағайындау) жұмыстарын автоматтандыру және де жүйелік программалар жұмысын жақсарту. Утилиттер жиі қайталанатын операцияларды – мысалы, магниттік дискіні форматтау, дискіні дефрагментациялау, файлдарды архивтеу, вирустарды тауып аластау, т.б. істерді орындайды.

Қызметтік программаларға

1. файл диспетчерлері;
2. архиваторлар;
3. диагностикалық программалар;
4. инсталляция жасау программалары;
5. коммуникациялық программалар, т.б. жатады.

Қолданбалы деңгей

Қолданбалы деңгей нақты есептер шығаратын қолданбалы программалар жиынынан (өндіріс, шығармашылық, ойын және оқыту) тұрады. Қолданбалы және жүйелік программалар арасында тығыз байланыс бар.

Қолданбалы программаларға мәтіндік редакторлар, мәтіндік процессорлар, графикалық редакторлар, мәліметтер базасын басқару жүйелері (МББЖ), электрондық кестелер, автоматтандырылған жобалау жүйелері, баспа жүйелері, аударма программалары, браузерлер, HTML-редакторлар, бухгалтерлік жүйелер, т.б. жатады.

1.3 Программалық жабдықтамалардың жіктелуі

Жүйелік программалар компьютердің жұмыс істеуі үшін керек. Олар операциялық жүйелерге, драйверлер мен утилиттерге және тест программаларына бөлінеді.

Операциялық жүйелер *немесе системалар (ОС) – компьютерді басқаратын және құрылғыларды қолданбалы программалармен байланыстыратын программалар тобы.* Дербес ком-

1.3 сурет. Программалық жабдықтамалардың жіктелуі

пьютерлерде кең тараған операциялық жүйелерге MS (PC) DOS, WINDOWS 95/98, WINDOWS 2000/XP, WINDOWS 7, OS/2, UNIX, т.б. жатады.

Программалық қоршаулар – дербес компьютер мен жұмыс істеуші адамның арасындағы іс-әрекеттерді жеңілдететін программалар. MS DOS үшін әдетте Norton Commander қоршауы, ал Windows ортасы үшін – Norton Desktop, Norton Navigator, FAR пайдаланылады.

Драйверлер – ОС жүйелерін сыртқы құрылғылармен (принтерлермен, дискілермен, пернетақтамен) байланыстыратын программалар тобы.

Утилиттер – ОС-ны кеңейтіп, қосымша қызмет атқаруға керекті программалар тобы. Оларға орыс-қазақ әріптерін енгізуді, антивирустік (вирустерге қарсы) программаларды, мәліметтерді архивтеу (қысу) программаларын жатқызуға болады.

Тест программалары компьютердің жұмыс істеу қабілетін (диагностика) тексеретін программалар, олар ақауы бар құрылғыларды анықтап, экранға мәліметтер шығарады.

Программалау жүйелері – программалау тілдері мен трансляторлардан (түрлендіргіш программалар) тұрады, олар арқылы жүйелік және қолданбалы программалық жабдықтамалар жасалады. Программалау тілдері – ФОРТРАН, ПАСКАЛЬ, БЕЙСИК, ЛИСП, С, С++, АДА, ПРОЛОГ, ЯВА, т.б. Олар ағылшын тіліне жақын жасанды тілде жазылады да, кейіннен **трансляторлар** көмегімен машиналық кодтарға түрлендіріледі. Трансляторлар интерпретаторларға және компиляторларға бөлінеді.

Интерпретаторлар әрбір команданы машина тіліне аудара салып, бірден орындайтын трансляторлар тобы, олардың артықшылығы – тұтынушымен диалог режимінде жұмыс атқарады, кемшілігі – жұмыс өнімділігі төмен болады.

Компилятор – командаларды орындамай тұрып, машиналық кодтарға толық аударатын программа. Осының нәтижесінде объектілік модуль деп аталатын программа жасалады, олардың бірнешеуін біріктіру арқылы жүктеу модулі жасалып барып, осы модульді орындау арқылы мәселе (есеп) шешіледі.

Программалық жүйелер жасайтын мықты фирмаларға Microsoft Corporation және Borland International жатады. Microsoft

фирмасы: Quick Basic, Visual Basic, Microsoft C++ программаларын, ал Borland фирмасы – Turbo Basic, Borland C, Turbo Pascal программалық тілдерін жасап таратуда.

Қолданбалы программалар

Қолданбалы (кәделі) программалық жабдықтамалар – белгілі бір мамандық саласында нақты есептер шығара алатын программалар жиыны. Оларға қарапайым программадан бастап күрделі есептерді шығара алатын қуатты мамандандырылған жүйелерді (мәтіндік процесссор, графикалық редактор, баспаханалық жүйелер т.б.), экономикалық, ғылыми мәселелерге арналған және жалпы көпшілікке қызмет ету кешендерін де жатқызуға болады.

Дербес компьютерлердің кең тарауына басты себеп болған қолданбалы программалардың кең таралуы еді, бұл программалар тек бір есепті шығарып қана қоймай, белгілі бір мамандық саласында есеп жұмыстарын түгел автоматтандыруды немесе информацияның белгілі бір түрлерін өңдеуді түгел қамти алатын болды.

Офистік программалар осы қолданбалы программаларға жатады. Олар мынадай түрлерге жіктеледі:

– мәтін редакторлары (Блокнот, WordPad, Word, PageMaker, т.с.с.);

– графикалық редакторлар (Paint, Fotoshop, CorelDraw);

– электрондық кестелер (кестелік процессорлар – Excel, Lotus, Works, т.с.с.);

– мәліметтер базасы (Access, Dbase, FoxPro, т.б.);

– Интернетпен жұмыс істейтін программалар тобы (Netscape Navigator, Internet Explorer, Opera, т.б.);

– автоматтық жобалау программалары (AutoCad);

– оқыту және ойнау программалары, т.б.

1.4 Программаларды коммерциялық түрде тарату

Программаларды қызметтеріне қарай жіктеуден басқа оларды тарату, сату істеріне қарай да жіктеуге болады.

Тегін таратылатын программалар (freeware) – көптеген программалаушылар жасайтын шағын утилит-программалар және басқа сатылатын программаларға тегін қосымша толықтырмалар. Бұлар Интернеттен көшіріп алынады.

Коммерциялық программалар (commercialware) – сатылатын программалар, бұларға барлық ірі-ірі программалық дестелер кіреді. Интернет-дүкендерде, сайттарда несиелік карточкалар (кәртішкелер) арқылы және қолма-қол ақшаға сатылады.

Шартты түрде тегін программалар (shareware) – көп тараған программалар тобы, утилиттер, кейде керекті көлемді программалар да осыған кіреді. Бұларды шамалы мерзім ішінде тегін қолданып, сонан кейін аздап ақша төлеу керек. Егер сол мерзімде ақша төленбесе, олар мүмкіндіктерінің біразын орындамай қояды немесе жиі-жиі ақша сұрап мазаны алады.

Тәжірибелік нұсқалар – дұрыс жұмыс атқаратын, сатылатын программалар. Біраз уақыт өткен соң, жұмыс істемей, ақша сұрайды. Кейбір мамандар оны бұзып, ішіне кіріп уақытын ұзартып алады немесе сағаты мен датасын кейінге ауыстырып қойып пайдаланады. Ал кейбірі бұрынғысын деинсталлятормен өшіріп, қайтадан жазып алады немесе көшіріп аларда жүйелік мерзімін алдағы уақытқа ауыстырады, сонда ол ұзақ уақыт тегін істейді.

Демоверсиялар (demoware) – жарнамалық нұсқалар, жұмыс істегенмен, нәтижесін дискіде сақтамайды немесе қолдану мерзімі аз ғана уақытқа жасалады.

Соңғы кезде информатика саласында программалық және аппараттық құралдардаға байланысты ақпараттық технология деген термин жиі айтылып жүр.

Бақылау сұрақтары

1. *Информатиканың құрамы қандай? Информатиканың негізгі аппараты не?*
2. *Информатиканың толық мағынасы.*
3. *Программа дегеніміз не? Мәліметтер ше?*
4. *Программалық конфигурация ұғымы.*
5. *Программалық жабдықтама қандай қызмет атқарады?*
6. *Программа жасауға кімдер қатынасады?*
7. *Программаралық интерфейс түсінігі.*
8. *Программалық жабдықтама деңгейлері. Базалық деңгей деген не?*
9. *Жүйелік және қызметтік деңгейлер сипаттамалары.*
10. *Қолданбалы деңгейдің атқаратын қызметі.*
11. *Программалау жүйелері деген не? Оның құрамы.*
12. *Программалық жабдықтамалар қалай жіктеледі?*

13. *Операциялық жүйелер қызметі.*
14. *Программалық қоршаулар деген не?*
15. *Драйверлер, утилиттер ұғымы.*
16. *Тест программалары қандай қызмет атқарады?*
17. *Интерпретатор, компилятор түсініктері, олардың ұқсастықтары мен айырмашылықтары.*
18. *Қолданбалы (кәделі) программалық жабдықтамалар ұғымы.*

2 АҚПАРАТТЫҚ ТЕХНОЛОГИЯ ЖӘНЕ ТЕХНИКА

2.1 Ақпараттық технология түсінігі

Технология грек тілінен аударғанда (techne) өнер (искусство), шеберлік, білгірлік деген сөз, ал мұны қазірде процесс ұғымымен де беруге болады.

Процесс – алдағы мақсатқа жету жолында атқарылатын белгілі бір әрекеттер жиыны. Процесс адамдар таңдап алған стратегиямен анықталуы тиіс, ол әр түрлі құралдар және тәсілдер көмегімен жүзеге асырылады.

Ақпарат кәдімгі материалдық ресурстар – мұнай, газ, пайдалы қазба байлықтар тәрізді қоғамның аса бағалы ресурстарына жатады, сондықтан оны өңдеу процесін материалдық ресурстарды өңдеу сияқты *технология* деп қабылдауға болады.

2.1 сурет. Ақпараттық технологияны материалдық ресурстарды өңдеу технологиясының аналогы тәрізді қарастыру

Ақпараттық технология – объектінің, процестің немесе құбылыстың (ақпараттық өнімнің) қалып-күйі жайлы жаңа сападағы ақпарат алу мақсатында мәліметтерді (алғашқы ақпаратты) жинау, өңдеу және тасымалдау құралдары мен тәсілдері жиынын пайдаланатын процесс.

Ақпараттық технология – компьютердің көмегімен ақпаратты жинау, енгізу, тасымалдау, сұрыптау, іздеу, реттеу, өзгерту және өңдеу жұмыстарының тізбегі деп те айтуға болады.

Материалдық технология өндірісінің мақсаты — адамның немесе жүйенің талаптарын қанағаттандыратын өнім алу.

Ақпараттық технологияның мақсаты — адамдар талдай ала-

тын ақпарат өндіру, соның негізінде белгілі бір әрекетті орындауға қажет шешім қабылдау ісі атқарылады.

Ақпараттық технологияның өндірістік технологиядан айырмашылығы оны адамның қатысуынсыз роботтар, компьютерлер іске асыратын үздіксіз процеске айналдыруға болмайды. Өйткені ол құжаттың көшірмесін алу, есепке енгізу, есеп шығару секілді жұмыстармен қатар шығармашылық ізденісті талап ететін шешімдер қабылдау, мәселені жүргізу жолын өзгерту, бірнеше мүмкіндіктерді салыстыра отырып, олардың ішінен ең тиімдісін іріктеп алу сияқты таңдау жұмыстарын кеңінен жүргізумен өте тығыз байланысты болады. Сондықтан ақпараттық технология "адамдық факторға" үлкен көңіл бөледі және оның шеберлік деңгейінің өсуіне әсер етеді.

Бір материалдық ресурсқа әр түрлі технологиялар қолдану арқылы әр түрлі бұйымдар, өнімдер алуға болады. Дәл осындай тұжырым ақпарат өңдеу технологияларына да сәйкес келеді.

Өнімдер алуға арналған технологиялар құрамы

Материалдық өнім	Ақпараттық өнім
Шикізаттар мен материалдар даярлау	Мәліметтер немесе алғашқы ақпарат жинау
Материалдық өнім өндірісі	Мәліметтерді өңдеу және нәтижелік ақпарат алу
Шығарылған өнімді тұтынушыларға сату	Нәтижелік ақпаратты шешім қабылдау үшін қолданушыға беру

Ақпараттық технология қоғамдағы ақпараттық ресурстарды пайдалану процесінің маңызды құраушысы болып табылады. Қазіргі кезге дейін ол ғылыми-техникалық прогрестің дамуына және ақпарат өңдейтін жаңа техникалық құралдардың шығуына байланысты бірнеше эволюциялық кезеңдерден өтті.

Қазіргі ақпарат өңдеу технологиясының негізгі техникалық құралы компьютер болып саналады, ол технологиялық процестің құрылу концепциясына және нәтижелік ақпарат сапасына да бірсыпыра әсерін тигізді.

Дербес компьютердің ақпараттық ортаға енуі және байланыстың телекоммуникациялық құралдарын пайдалану ақпараттық технологияның жаңа кезеңі келгенін анықтады. Осыған орай технология атауына синоним сөздер болып табылатын *"жаңа", "компьютерлік"* немесе *"заманауи"* тәрізді тіркестер қосылып қолданыла бастады. **"Жаңа"** сөзі бұл технологияның эволюциялық сипатын емес новаторлық мүмкіндігін айқындайды. Оның новаторлық мүмкіндігі мекемелердің әр түрлі жұмыстарының мазмұндарының да жаңаруына сәйкес пайда болды.

Жаңа ақпараттық технология ұғымына ақпаратты тасымалдайтын телефон, телеграф, телекоммуникация, факс тәрізді құрылғылардан тұратын коммуникациялық технологиялар да кіреді.

Жаңа ақпараттық технологиялардың негізгі сипаттамалары

Методологиясы (әдіснамасы)	Негізгі белгісі	Нәтижесі
Жаңадан шыққан ақпарат өңдеу құралдары	Басқару технологиясына "кірістіру"	Коммуникациялардың жаңа технологиясы
Біртұтас технологиялық жүйелер	Мамандар мен менеджерлер қызметін интеграциялау	Ақпарат өңдеудің жаңа технологиясы
Ақпаратты мақсатқа сәйкес түрде жасау, тасымалдау, сақтау және бейнелеу	Әлеуметтік орта заңдылықтарын есепке алу	Басқару шешімдерін қабылдаудың жаңа технологиясы

Жаңа ақпараттық технология – дербес компьютерлер мен телекоммуникациялық құралдарды пайдаланатын, қолданушылармен "жылы жүзді" интерфейс арқылы жұмыс істейтін ақпараттық технология түрі.

Жаңа ақпараттық технологияның үш негізгі қағидасы (принципі) бар, олар:

– компьютермен интерактивті режимде жұмыс істеу;

– басқа программалық өнімдермен бірігуі (байланысы, қатынас жасауы);

– мәліметтерді де, мәселені қоюды да өзгерту процесінің икемділігі.

Ақпараттық технологиялар оның негізгі ортасы болып табылатын ақпараттық жүйелермен тығыз байланысты. Ақпараттық технологиялар – компьютерде сақталатын мәліметтермен шектелген ережелерге сәйкес орындалатын кез келген операциялардан, әрекеттерден, кезеңдерден тұратын процесс.

Ақпараттық технологияның негізгі мақсаты – алғашқы мәліметтерден белгіленген әрекеттерді орындау нәтижесінде қолданушыға керекті ақпарат алу.

Ақпараттық жүйенің құраушы элементтері компьютерлер, компьютерлік желілер, программалық өнімдер, мәліметтер базасы, адамдар, байланысудың техникалық және программалық жабдықтары болып келетін орта болып табылады. Ақпараттық жүйенің негізгі мақсаты – ақпаратты сақтау мен өңдеуді ұйымдастыру. Ақпараттық жүйе мәлімет өңдеудің адамдық-компьютерлік жүйесі болып табылады. Ақпараттық жүйеге бағытталған ақпараттық технологияны білмей, жүйе функцияларын жүзеге асыру мүмкін емес. Ақпараттық технология ақпараттық жүйе аймағынан тыс жұмыс істей береді.

Екі ақпараттық технологияны – басқару және компьютерлік технологияларды сәйкестендіре білу – ақпараттық жүйенің табысты жұмыс істеуінің кепілі бола алады.

Осы айтылғандарды есепке ала отырып ақпараттық технология және ақпараттық жүйе ұғымдарының бұрынғы анықтамаларын толықтыра түсуге болады.

Ақпараттық технология – компьютерде мәлімет өңдеу кезінде персоналдың мақсатқа жету жолында атқаратын нақты анықталған әрекеттер тізбегі.

Ақпараттық жүйе – компьютерлік ақпараттық технологияларды пайдалану арқылы шешім қабылдауды сүйемелдеу және ақпараттық өнім өндіру істерінің адамдық-компьютерлік жүйесі.

Қазіргі кездегі ақпараттық технология ұғымын мәліметтер өңдеуге байланысты үш түрге жіктеуге болады, олар: есепке алу, талдау және шешім қабылдау. Осы уақытқа дейін бұл жұмыстар адамның басқаруымен қағазға жазылған құжатты өңдеу түрінде өтетін, ал енді басқару, ұйымдастыру жұмыстарын компьютер көмегімен автоматты түрде атқару жаңа ақпараттық технологияның тууына себепші болды. Бұл жаңа технологияның

бұрынғы технологиядан (құжатты машинкада басу, байланысты телефонмен жүргізу, сөзді диктофонға жазу) айырмашылығы – тізбектеле орналасқан ақпараттық техника, "жылы жүзді" программалық жабдықтама және дамытылған компьютерлік байланысты кеңінен қолдану. Сонымен жаңа ақпараттық технология төмендегі мынадай мүмкіндіктерді орындауы тиіс:

– әрбір адам мәліметтерді дұрыс орналастыра білуі (программалау емес) керек;

– мәліметті өңдеу барысында оларды өрнектеу, сақтау, іздеу, бейнелеу және өзгерістен қорғаудың бірыңғай үлгісіне келтіріп, жинақталған мәліметтер қоймасын пайдаланатын ақпараттық қолғабыс көрсету;

– кез келген құжатты қағазсыз өңдеу ісін ұйымдастыру, мұнда қағазға құжаттың ең соңғы нұсқасы ғана түсіріледі, ал басқалары электрондық машинаның мәлімет жинақтағыштарында (дискілер) сақталады да, экранда көрсетіледі;

– адам үшін көптеген жеңілдіктері бар есепті шығарудың сұхбаттасу режимін (интерактивті режим) жүргізу;

– бір-бірімен байланысқан компьютерлерді пайдалана отырып, құжатты ұжымдық түрде даярлау;

– есепті шығару барысында мәліметтерді бейнелеу тәсілін жылдам өзгерту мүмкіндігі болу.

2.2-сурет. Ақпараттық техника құрамы

Бұл күндері жаңа ақпараттық технология информатиканың негізгі бір саласы болып табылады, оның болашақта басқа салаларда істейтін мамандар жұмысына тигізетін әсері зор болады деп күтілуде. *Ақпараттық техника* – ақпаратты өңдеуге керекті техникалық аспаптар жиыны. 2.2-суретте көрсетілген алғашқы екі техника түрі үздік-үздік цифр түрінде берілген дискретті мәліметтермен жұмыс істейді, олар: Компьютерлер үшін екілік сандар, ал

телеграфияда – морзе әліппесі. Қалған техника түрлері үздіксіз электр сигналы ретінде берілетін мәліметтерді пайдаланады.

Болашақта дискретті мәліметтермен жұмыс істейтін аспаптар көбейе түсуі тиіс, өйткені олар жоғары дәлдікпен сенімді түрде тыңғылықты қызмет атқарады. Сол себепті қазіргі кезеңде цифрлы телефондар мен теледидарлар шығарыла бастады.

Халық шаруашылығында ақпаратты кең қолдану үшін оны сақтайтын, жеткізетін және өңдейтін құралдарды көптеп жасау керек болды. Соның нәтижесінде ақпараттық техника күннен-күнге көбеюде, әсіресе оның ішінде электрондық машиналарға өте көп көңіл бөлінуде, өйткені ақпаратты жылдам өңдеу үшін қолданылатын негізгі электрондық тетік есептеу машинасы болып табылады. Сонымен қазіргі кезде информатиканы кеңінен қолданатын орындарға мыналарды жатқызуға болады:

1) еңбек ету аспаптарын басқару, яғни станоктарды, роботтарды, машиналарды басқару;

2) ғылыми-зерттеу жұмыстарын жүргізуді автоматтандыру;

3) компьютерлерді халық шаруашылығындағы экономикалық ұйымдастыру, басқару жұмыстарында пайдалану;

4) компьютерлерді оқыту және халық шаруашылығына керекті мамандар даярлауда қолдану;

5) компьютерлерді халыққа ақпараттық қызмет көрсетуге пайдалану (медицина, сауда, тұрмыс қажетін өтеу, энергетика, т.б.).

Информатиканың дамуы елдің экономикалық құрылымының және еңбек өнімділігінің өсу деңгейін көрсетеді, сол себепті ол елдің экономикалық қуаттылығының көрсеткіші бола алады.

2.2 Ақпараттық ресурстар және өнімдер

Ресурстар – *сақтаулы заттар, белгілі бір нәрселердің шығу көздері* деген мағына береді.

Мүмкіндіктердің басым бөлігі материалдық өндіріске бағытталған біздің қоғамда классикалық экономикалық категориялар (санаттарға) болып саналатын ресурстардың бірнеше негізгі түрлері бар, олар:

– *материалдық* ресурстар – қоғамдық өнім өндіру процесінде пайдалануға арналған еңбек заттары, мысалы, шикізат, матери-

алдар, отын, энергия, полуфабрикаттар (жартылай шикізаттар), құралдар, т.б.;

– *табиғи* ресурстар – адамдардың материалдық және рухани талаптарын қанағаттандыру үшін пайдаланылатын объектілер, процестер, табиғи жағдайлар;

– *еңбек* ресурстары – қоғамда жұмыс істеу үшін жалпы және кәсіби білімі бар адамдар;

– *қаржылық* ресурстар – мемлекеттік немесе коммерциялық құрылымдардың қолындағы ақшалық қаржылар;

– *энергетикалық* ресурстар – энергия көздері, мысалы, көмір, мұнай, мұнай өнімдері, газ, гидроэнергия, электр энергиясы, т.б.

Ақпараттық қоғамда дәстүрлі ресурс түрлерінен гөрі ақпараттық ресурстарға көбірек назар аударылады. Олар бұрыннан болғанменен, кезінде, экономикалық немесе басқа категориялар ретінде қарастырылмаған; ешкім олар туралы арнайы пікір айтпаған, әрине олар жайлы анықтамалар, түсініктер де болмаған еді.

Қазіргі ақпараттық қоғамға өту кезеңіндегі ең өзекті түсініктердің бірі *"ақпараттық ресурстар"* ұғымы болып отыр. Бұған соңғы кезде көптеген соны пікірлер, анықтамалар, түсініктер беріліп келеді.

Ақпараттық ресурстар – жеке құжаттар және жеке құжаттар жиымдары, ақпараттық жүйелердегі (кітапханаларда, мұрағаттарда, қорларда (фондтарда), мәліметтер банктерінде, басқа да ақпараттық жүйелерде) құжаттар мен құжаттар жиымдары.

Ақпараттық ресурстар – қоғамдағы әлеуметтік мақсатта пайдалануға арналған және белгілі бір материалдық жинақтауышта тіркелген адамдардың дайындаған білімдері. Қоғамның ақпараттық ресурстары білім ретінде оларды жасаған, жинақтаған, талдаған адамдардан бөлек сақтала береді. Адамзат білімі құжаттар, мәліметтер базасы, білім базасы, алгоритмдер, компьютерлік программалар және де өнер, әдебиет шығармалары, ғылыми еңбектер түрінде жазылып сақтала береді. Мемлекеттің, аймақтың, ұйымның ақпараттық ресурстары шикізат, энергия қорлары, пайдалы қазба байлықтар сияқты стратегиялық ресурстар ретінде қарастырылады.

Ақпараттық ресурстар *ақпараттық өнім* жасаудың негізгі базасы болып табылады. Кез келген ақпараттық өнім оны жасаушының ақпараттық моделін бейнелейді де, оның қарастырып отырған нақты пәндік аймағы жайлы жеке көзқарасын көрсетеді.

Ақпараттық өнім адамның интеллектуалдық еңбегінің нәтижесі бола отырып, белгілі бір материалдық мәлімет жинақтайтын кітап, программа, мақала, құжат, т.с.с. түрінде тіркеліп қалуы тиіс.

Ақпараттық өнім – заттық немесе заттық емес формада таратуға арнай отырып, ақпаратты жасаушы адам қалыптастырған мәліметтер жиыны.

Ақпараттық өнім кез келген материалдық тауар (өнім) сияқты қызмет көрсету жолымен таратыла алады. Көрсетілетін қызмет түрлері ақпараттық ресурстарды пайдалану аймағындағы оның көлемімен, сапасымен, қолданылу бағытымен және солардың негізінде жасалатын ақпараттық өнімдер арқылы анықталады.

Ақпараттық қызмет тек компьютерлік немесе компьютерлік емес нұсқадағы мәліметтер базасы болған кезде ғана туындайды.

Мәліметтер базасы ақпараттық қызмет көрсетудің негізгі көзі және шикізаты болып табылады. Мәліметтер базасы дәл осылай аталмағанмен, компьютерлік кезеңге дейін де кітапханалар, мұрағаттар, анықтамалық бюролар түрінде болған еді. Оларда оқиғалар, құбылыстар, объектілер, процестер, жарияланымдар жайлы барлық мәліметтер сақталатын болатын.

Мәліметтер базасы – ұйымдастыру ережелері мәліметтерді сипаттау, сақтау, өңдеу қағидаларына негізделіп, бір-бірімен байланысқан мәліметтер жиыны.

Компьютерлер пайда болған соң, мәліметтер базасының көлемі тез ұлғайды және соған сәйкес ақпараттық қызмет көрсету аймағы да кеңейді.

Қызмет көрсету – мекеменің немесе жеке адамның басқа бір адамның не мекеменің әр түрлі өнімдерді пайдалану талаптарын қанағаттандыруға бағытталған өндірістік емес іс-әрекеттерінің нәтижесі.

Ақпараттық қызмет көрсету – ақпараттық өнімдердің қолданушы (тұтынушы) қолына пайдалануға берілуі. Ақпараттық

қызмет көбінесе компьютер арқылы берілетін қызмет түрі болып саналады, бірақ одан оның ауқымы кең.

Қызмет көрсету кезінде оны беруші мен алушы арасында келісім-шарт жасалып, онда қызмет көлемі, бағасы, қолданылатын өнім көрсетіледі. Ақпараттық қызмет көрсету компьютер немесе басқа бір мүмкіндік арқылы мәліметтер базасы түрінде беріледі.

Енді ақпараттық технология ұғымымен сабақтас программалау технологиясы мәселелеріне кеңірек тоқталайық.

Бақылау сұрақтары

1. *Технология, процесс ұғымдары.*
2. *Ақпараттық технология мен материалдық технология айырмашылықтары.*
3. *Ақпараттық технология дегеніміз не? Ақпараттық жүйе ше?*
4. *Жаңа ақпараттық технология ұғымы.*
5. *Ақпараттық техника құрамы.*
6. *Ақпараттық ресурстар және өнімдер ұғымдары.*
7. *Ақпараттық қызмет көрсету дегеніміз не?*

3 ПРОГРАММАЛАУ ТЕХНОЛОГИЯЛАРЫ

Программалау – программа құру процесі, алгоритм ұғымымен тығыз байланысты информатика саласының бір бөлігі. Бұл – соңғы кезде өте жылдам дамып келе жатқан пән саласы. Бұрынғы программалық және техникалық құралдарды дамыту мен жаңа мүмкіндіктердің пайда болу тәжірибесі программалауды тұрақты жетілдіріп келеді, осының нәтижесінде жаңа тәсілдер мен технологиялар шығып, олар қазіргі программалық жабдықтамаларды жасауға жаңа негіздеме болып табылады. Жаңа технологиялар жасау процесін зерттеу және олардың негізгі даму бағытын анықтау ісін жүргізуді қазіргі технологияларды программалау деңгейімен салыстыра отырып, қолда бар программалық және аппараттық құралдар ерекшеліктерін қолдану арқылы жетілдіру қажет.

Программалау технологиясы деп программалық жабдықтамаларды жасау процесінде пайдаланылатын құралдар мен тәсілдер жиынын айтады. Кез келген басқа технологиялар сияқты программалау 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:

1) программа жасау процесінің нақты кезеңдерінде пайдаланылатын технология немесе осы кезеңдердегі жеке есептерді шығару;

2) 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 программа жасаудың әртүрлі кезеңдерінде пайдаланылатын тәсілдер жиынын анықтайтын базалық тәсіл, яғни ортақ көзқарас (парадигма) немесе әдіснама (методология) қарастырылады.

Программалаудың даму тарихында бір-бірінен айырмашылығы бар бірнеше парадигмалық сатылардың болғаны туралы айтылады.

3.1 Құрылымдық программалау

Өткен ғасырдың 60 жылдары – программалаудың "стихиялық" даму кезеңінде программа құрылымы, мәліметтер типтері сияқты ұғымдар болмады. Сол себепті программалар қайшылықтары көп, түсінуге қиын кодтардан тұратын еді. Ол кездегі программалау өзінше бір өнер тәрізді болып көрінетін. Программалаудағы сондай қиындықтан шығу үшін программалаудың құрылымдық парадигмасына көшу керек болды.

Бұрынғы прогаммалау тілдері пайда болған кездерден бастап, мынадай парадигма үстемдік еткен болатын.

"Қандай процедуралар керек екендігін анықтап алып, қолдануға болатын ең тиімді алгоритмдерді пайдаланыңыз".

Прогаммалау тілдерінде осы парадигма параметрлерді қабылдап алып, нәтижелік мән қайтаруды қамтамасыз ететін функциялар арқылы сүйемелденеді. Стильдің бұл түрі оқулықтарда параметрлер функцияға қалай беріледі, параметрлердің, функциялардың (процедуралар, ішкі программалар, макроанықтаулар) әртүрлі типтері қалай ажыратылады, т.с.с. пікірталас түрінде беріліп келеді. Фортран – алғашқы процедуралық тіл; Алгол60, Алгол68, Паскаль және С – сол нұсқаға сәйкес кейінгі шыққан тілдер болып табылады.

"Жақсы программалау стилінің" бір мысалы – функцияның аргумент-параметрі ретінде берілген санның квадрат түбірін табу функциясы; егер функцияның аргументі (параметрі) берілсе, ол нәтиже береді. Мұндай есеп белгілі математикалық есептеулер арқылы шығарылады. Мысалы:

```
double sqrt ( double arg)
{
    // квадрат түбірді табу программалық коды
}
void some_function ()
{
    double root2 = sqrt(2);
    // …
}
```

Мұндай функция алгоритмдер қоймасына белгілі бір тәртіп орнатады да, оның атқаратын жұмысы осы есептеумен аяқталады.

Сонымен, программаны құрудың негізгі әдістемесі процедура болып табылады. Бұл әдістеменің негізі – алгоритм. Программа көлемінің ұлғаюына байланысты үлкен көлемдегі есептерді бірнеше есептерге бөлу мәселесі туындады. Сондықтан программалауда процедура түсінігі енгізілді. Процедура программалық кодтардың көлемін азайтуға мүмкіндік туғызды.

Программалаудағы құрылымдық көзқарас – программалық жабдықтаманы жасаудың барлық кезеңдерін орындауды қамти отырып, қолдануға болатын технологиялық әдістер жиыны. Құрылымдық программалау негізінде күрделі жүйелерді декомпозициялау (бөліктерге бөлу) тәсілі жатыр, мұнда көлемді программа шағын ішкі программаларға бөлініп барып, соларды жеке-жеке орындалу жолымен іске асырылады. Декомпозициялаудағы басқа да әдістердің (объектілік, логикалық, т.б.) пайда болуына байланысты бұл тәсіл процедуралық декомпозиция деп аталды. Процедуралық декомпозицияның екінші бір ерекшелігі ретінде программа құруда тек базалық алгоритмдік құрылымдарды пайдаланып, GOTO операторын қолдануға тиым салынғанын айтуға болады.

Негізгі ***үш базалық алгоритмдік құрылымдарға*** тізбекті, тармақты және қайталау алгоритмдері жатқызылды.

Тізбекті құрылым көрсетілген кезекпен орындалатын екі командадан тұрады да, ары қарай солай жалғаса береді:

<алдыңғы команда>;

<келесі команда>.

Тармақты құрылымның толық түрі белгілі бір шарттың орындалуы немесе орындалмауына байланысты екі мүмкіндіктің бірінің ғана атқарылуын көрсетеді. Бұл құрылым С тілінде мынадай түрде жазылады:

if <шарт>

<шарт орындалғанда, атқарылатын команда>;

else <шарт орындалмағанда, атқарылатын команда>;

Тармақты құрылымның қысқаша түрі – алдыңғы команданың дербес түрі, мұнда шарт орындалмаған жағдайда, ешқандай команда атқарылмай, кезек келесі командаларға беріледі. Бұл құрылымның С тіліндегі жазылуы:

if <шарт>

<шарт орындалғанда, атқарылатын команда>;

Қайталау құрылымы (*цикл*) команданың белгілі бір параметрінің өзгеруі барысында, қайталанып бірнеше рет орындалатын операциялар тобын ыңғайлы түрде жазу үшін қолданылады.

Мысалы, "әзірше (while)" қайталау құрылымы С тілінде былай жазылады:

while <қайталану шарты>

do

<қайталанатын команда>;

Цикл тұлғасы деп қайталанатын командалар тізбегін айтады, ол кейде бос та болуы мүмкін (сирек кездесетін жағдай).

Қайталау циклінің орындалу ережесі: егер шарт орындалмаса (жалған болса), циклді қайталау тоқтатылады; ал егер шарт орындалса (ақиқат болса), онда цикл тұлғасы атқарылып, қайтадан шартты тексеру жүргізіледі.

Құрылымдық программалау есепті шағын құрылымдарға бөліп, олардың әрқайсысын жеке қарастыруды қажет етеді. Мұнда жобалау "жоғарыдан-төменге" қарай жүргізілді де, жалпы құрылым идеясын жүзеге асырып, ішкі интерфейс программаларын орындауды талап етті. Бұған қоса, алгоритм конструкцияларына шектеулер қойылып, оларды формалды модель түрінде сипаттау ұсынылды және алгоритм құрудың арнайы тәсілі – қадамдық түрде орындау тәсілі қолданылды.

Құрылымдық программалау қағидасын сүйемелдеу процедуралық программалау тілдерінің негізіне жатқызылды. Олар

көбінесе негізгі басқаруды беру "құрылымдық" операторларын қолдану, ішкі программаларды пайдалану және мәліметтердің "көріну" аймағын шектеу тәрізді мүмкіндіктерді сүйемелдеді. Бұл топтағы кең таралған тілдер ретінде PL/1, ALGOL-68, Pascal, C тілдерін атауға болады.

Программалық жабдықтамалардың ары қарай көлемінің өсуі, күрделілігінің артуы мәліметтерді құрылымдауды да дамытуды талап етті. Осының нәтижесінде қолданушының мәліметтер типін анықтау мүмкіндігі пайда болды. Оған қоса, программаның ауқымды (глобальді) мәліметтерін пайдалануға шек қойып, олармен жұмыс істеу кезінде шығатын қателерді азайту мақсатында жұмыстар істеле бастады. Сондықтан бара-бара жаңа технология – модульдік программалау технологиясы жетіле бастады.

3.2 Модульдік программалау

Уақыт өте келе программаларға деген көзқарас өзгеріп, бұрынғыдай процедуралар құрудан гөрі мәліметтерді ұйымдастыру жағына көңіл бөліне басады. Бұған басты себеп болған жайттардың бірі программалар көлемінің ұлғаюы болды. Бірбірімен өзара байланысқан процедуралар мен солар арқылы өңделетін мәліметтер жиынын көбінесе модуль деп атайтын еді. Сондықтан программалау парадигмасы аздап өзгеріске ұшырады:

"Өзіңізге қандай модуль керек екендігін анықтап алыңыз да, мәліметтер модуль ішінде көрінбей тұратындай етіп программаны бірнеше бөліктерге бөліңіз".

Бұл қағида "мәліметтерді пайдалануды шектеу" деген атпен тарала бастады. Мәліметтер оларды өңдейтін программалармен бірікпеген жағдайда бұрынғы процедуралық программалау қағидасын қолданатын болды.

Модула-2 тілі модульдік программалау тұжырымын тікелей сүйемелдейтін деңгейде құрылды, ал C тілі болса, оны тек пайдалану мүмкіндігін ғана беретін дәрежеде болды.

Мәліметтер программадағы жасырылып қойылатын элементтердің тек біреуі ғана болғандықтан, мәліметтерге қол жеткізуді шектеу, яғни "ақпаратты жасыру" қағидасын енгізгеннен кейін, оған қоса айнымалылар, константалар, функциялар және олардың типтерін де бір модуль ішінде ғана белгілі болатындай

етіп, шектеу мүмкіндігі туды. C++ тілін жасауда модульдік программалауды енгізуге арнайы көңіл бөлінбесе де, C++ кластары ұғымы, негізінде, модульділікті сүйемелдейтін мүмкіндік болып шықты. Оның үстіне C++ тілі программа бөліктерін жекежеке компиляциялау ісін орындайтын болғандықтан, C тіліне қарағанда, мұны модульдік тіл деп айтуға толық негіз бар еді.

Модульдік программалау ауқымды айнымалылардың белгілі бір тобын қолданатын ішкі программалар жиынын жеке компиляцияланатын модульдерге (ішкі программалар кітапханасы) бөліп қарастыру тәсілі болып табылады. Бұл технологияда модульдер арасындағы байланыс арнайы интерфейс арқылы жүзеге асырылады да, сол модульдердің өздерін пайдалануға (ішкі программалар командаларын және олардың "ішкі" айнымалыларын) тиым салынады. Бұл технологияны Pascal, C/C++, Ада, Modula тілдерінің соңғы нұсқалары сүйемелдейді.

3.3 Мәліметтер абстракциясы

Модульдерді пайдаланып программалау, ақырында, бір типтегі барлық мәліметтерді бақылауды бір басқару блогында жинақтауға әкеліп соқтырды. Бұл дәстүрлі тәсілді, кластарды пайдаланбай жұмыс істейтін көзқарасты жетілдіру, бірақ бұлай ұйымдастырылған типтер тіл құрамындағы негізгі типтерден басқаша болып шықты.

Мұндайда, біріншіден, белгілі бір типтегі мәліметтерді басқаратын әрбір модульдің осы типтегі айнымалыларды енгізетін бөлек механизмі болуы керек.

Екіншіден, жаңа типтегі айнымалылар компилятор мен программалау ортасына белгісіз болып келеді, ақырында, мұндай "айнымалылар" көріну аймағының бұрынғы ережелеріне бағынбайды және олар функцияларға қарапайым параметр ретінде беріле алмайды.

Модуль түрінде жасалатын осындай типтер негізгі типтерден басқаша болады және олар сүйемелдеудің де төменгі деңгейінде орналасады. Осыдан барып компилятор да бірсыпыра төменгі қателерді таба алмайтын болады. Басқаша айтқанда, мәліметтерді жасыра алатын модульдер концепциясы мәліметтерді абстракциялауды толық көлемде атқара алмайды.

Сонымен, осыған орай, программалау парадигмасы тағы да өзгеріске ұшырап, енді былай айтылатын болды:

"Өзіңізге қандай типтер керек екенін анықтап алыңыз да, әрбір тип үшін операциялардың толық жиынын дайындаңыз".

Мұндай ортада әрбір типтегі объект тек біреу болған жағдайда, модульдік программалауды пайдалана беруге болады. Арифметикалық типтер, мысалы, рационал комплекс сандар қолданушы анықтаған типтердің нақты мысалы бола алады.

Класты сипаттауда, яғни қолданушы анықтаған complex типінде комплекс сандарды бейнелеу форматы мен олармен орындалатын операциялар жиыны толық көрсетіледі. Мұндайда мәліметтер жабық әрі жекеше (приватты) болып саналады. Бірақ бірсыпыра модульдер (барлығы емес) қолданушы анықтаған типтер ретінде көрсетіле алады.

3.4 Объектіге бағытталған программалау түсінігі

Дәстүрлі процедураға бағытталған программалау тәсілдері күрделі программалардың талаптарын қанағаттандыра алмады. Сондықтан объектіге бағытталған программалау ұғымы пайда болды.

Объектіге бағытталған программалау (ОБП) – бұл жаңа технология. Мұнда құрылымдық программалаудың ең тиімді жақтары алынып, олар жаңа идеялармен толықтырылады.

Объектіге бағытталған тіл деп программалары өзара байланысқан объектілер тізімінен және солардың жұмысын сипаттаудан тұратын программалау тілін айтады. ОБП ерекшеліктері:

– программалық жабдықтамалардың күрделілік дәрежесін төмендету;

– программалық жабдықтамалардың сенімділігін жоғарылату;

– программалық жабдықтамалардың кейбір компоненттерін қайтадан қолдану мүмкіндігін қамтамасыз ету.

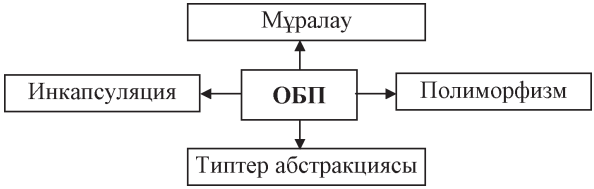
ОБП негізгі ұғымдарының бірі – объект. Объект – мәліметтердің логикалық бірлігі ретінде программада құрастырылған жаңа типтегі айнымалы.

Объектінің атқаратын басты міндеті – мәліметтерді және оларды өңдеуге арналған тәсілдердің (ережелердің) бірігуі.

C++ тілінде мәліметтерді өңдеу ісі тәсілдер, яғни функциялар арқылы атқарылады. Сондықтан C++ программалау тілінде объект мәліметтерден және соларға қолданылатын функциялардан тұрады.

Объектіге негізделген мәліметтер мен функциялар жабық (private), қорғалған (protected) және ашық (public) бола алады.

– ОБП технологиясының негізгі төрт сипаттамасы бар:
– инкапсуляция;
– мұралау;
– полиморфизм;
– типтердің абстакциясы.



3.1 сурет. ОБП негізгі қасиеттері

Инкапсуляцияда программа бірлігі ретінде мәліметтер және функциялар тобының бірлестігі қарастырылады. Объектіде мәліметтер қорғалған болып саналады, егер мәліметтер мен функция-мүшелер жабық немесе қорғалған деп айтылатын болса, онда олар программаның басқа бөлігінен көрінбейтін болады.

Программа сенімді жұмыс істеуі үшін объектінің өрісін тікелей пайдалану қажет емес. Сондықтан мұндайда оның мазмұнын ашық тәсілдерді шақыру арқылы қалыптастыру керек.

Белгілі бір класс ішінен ішкі класс құрастыру **мұралау** болып табылады. Мұралау кезінде бір объектінің атрибуттары мен қасиеттерін екіншісінің иемденуіне мүмкіндік беріледі. Егер бұрынғы кластан аз ғана айырмашылығы бар жаңа класс құрғымыз келсе, онда бастапқы кластың бұрынғы айнымалылары мен тәсілдерін қайтадан сипаттап жазу қажет емес. Сонда базалық класс *аталық*, ал жаңадан пайда болған класс – оның *ұрпағы (туындысы)* деп аталады. Ұрпақ кластың, қажет болса, өз құрамына жаңа айнымалылар мен тәсілдер қосу мүмкіндігі бар.

Мұралау өз кезегінде объектілердің иерархиясын құрастыруға жағдай туғызады, соның нәтижесінде жалпыдан жалқыға көше алады, ол объектіні нақтылайды, айқындайды.

Полиморфизмде кластың бір атаулы функция-мүшесі әр түрлі, бірақ өзара логикалық байланысқан мақсаттарда пайдаланылады. Мысалы, sum(x,y) функциясы x + y қосындысын табады делік. Аргументтерінің типіне қарай функция әр түрлі типтегі мәнді қайтара алады. Бүтін немесе нақты типті x пен y айнымалылары үшін атаулары әр түрлі екі функцияны тағайындаудың орнына бір sum(x,y) функциясын сипаттау тиімді болады. Мұндағы + амалы x пен y типтеріне тәуелді болады.

C++ тілінің негізгі ұғымдарының бірі – класс болып табылады. Класс дегеніміз – құрамында мүшелері (айнымалылары) мен тәсілдері (функциялары) бар тіл құрылымы. Кластық құрылым – программалаушының өзі анықтайтын тип. C++ тілінде объектіні анықтау үшін алдымен оның форматын class түйінді сөзімен белгілеу керек. Кластың анықталуы құрылымның (struct) анықталуына ұқсайды. Құрылымның кластан айырмашылығы – оның құрамындағы барлық мүшелер мен тәсілдердің ашық сипатталуында жатыр.

Программалау парадигмасын енді мынадай түрде айтуға болады:

"Өзіңізге қандай кластар керек екендігін анықтап алыңыз да, әрбір класс үшін барлық операциялар жиынын дайындаңыз; жалпы қасиеттерді айқын түрде көрсетіп, мұралауды пайдаланыңыз"

Жалпы стандарт бойынша, кластың барлық элементтері (мүшелері мен тәсілдері) *жабық* болып табылады. Жабық элементтерді осы кластан тыс ешбір сыртқы класс немесе функция пайдалана алмайды. Бұл инкапсуляция қағидасына сәйкес келеді. Инкапсуляция мәліметтерге программа кодынан қол жеткізуді бақылау, оларды жабық деп жариялау арқылы жүргізіледі.

ОБП қолданбалы программалауда жедел дамып келе жатқан бағыттардың бірі болып табылады.

3.5 Программалау технологияларының даму кезеңдері

Программалау технологияларының онжылдықтар бойынша көрсетілген даму кезеңдеріне қысқаша тоқталайық.

Өткен ғасырдың 50-ші жылдары компьютерлер қуаттылығы (ЭЕМ-дердің алғашқы буыны) онша жоғары болған жоқ, ал олар үшін программалау машиналық кодтар арқылы жүргізілді. Негізінен, ғылыми-техникалық есептер (формулалар бойынша) шығарылды, программалауға берілген тапсырмалар нақты есептер қойылымынан тұратын еді. Көбінесе программалаудың интуитивтік технологиясы қолданылатын, мұнда бірден тапсырма бойынша программа құруға кірісетін, оның үстіне жұмыс барысында тапсырмалар өзгертіліп те отыратын еді, қажетті құжаттама программа жұмыс істеген кезде барып құрастырылатын. Соған қарамастан, программалау технологиясының іргелі тұжырымдамасы болып саналатын модульдік программалау осы кезеңде (машиналық кодта программалаудың қиындығын жеңу мақсатында) пайда болды. Алғашқы жоғары деңгейде программалау тілдері пайда бола бастады, солардың ішіндегі ФОРТРАН тілі одан кейін де ұзақ уақыт пайдаланылып келеді.

1960-шы жылдары бірсыпыра жоғары деңгейде программалау тілдерінің (АЛГОЛ 60, ФОРТРАН, КОБОЛ, т.б. өте жылдам дамығанын көреміз, сол кездерде программалау технологиясындағы олардың орны өте жоғары бағаланылып жүрді. Бірақ осы тілдер көлемді программалар жазуда барлық мәселелерді шешеді деген үміт ақталмады. Компьютерлердің қуаты артып, программалау тәжірибелері де жинақталған кездерде жоғары деңгейде программалау тілдерінің мүмкіндіктері шектеулі екендігі белгілі болды. Осының нәтижесінде программаларды модульдік түрде ұйымдастыру жүзеге асырыла бастады. Тек ФОРТРАН тілі ғана пайдаланыла берді, өйткені ол бұрыннанақ модульдік қағидамен жасалған еді және дайындалған программалар кітапханасы да өте бай болатын. Осы кездерде программаны қай тілде құру емес, қалай құру керек деген сауалдың маңызды екені түсінікті болды. Мамандар енді программалау технологиялары мен олардың әдіснамаларына көңіл бөле бастады.

Екінші буын компьютерлеріндегі жұмысты уақытша тоқтатып, программаларды үзе тұру мүмкіндігінің пайда болуы мультипрограммалау ісінің жетілдіріліп, үлкен программалық жүйелер жасауға себепші болды. Бұлардың бәрі программалар-

ды ұжым болып жасаудың әсерінен туындады да, көптеген жаңа технологиялық мәселелер пайда болды.

Өткен ғасырдың 70-ші жылдары ақпараттық жүйелер мен мәліметтер базасы кең етек жайды. Бұған итермелеген негізгі себеп – компьютердің мәлімет жинақтауыштарындағы бір бит ақпаратты сақтау құны бұрынғы дәстүрлі мәлімет сақтаудан арзан болып кетті. Программалау технологиялары жылдам алға басып, құрылымдық программалаудың дамуы (мысалы, мәліметтер құрылымын жасыра алатын модульдердің шығуы), мәліметтердің абстракты типтерінің жетілдірілуі, программалық жабдықтамалардың сенімділігі мен мобильдігін қамтамасыз етілуі, ұжымдық түрде программа құруды басқару әдістемесінің шығуы, программалау технологияларын сүйемелдейтін саймандық программалық құралдардың пайда болуы жалпы осы бағыттағы жұмыстар түрін жаңа сатыға көтерді.

1980-ші жылдар жеке пайдаланылатын дербес компьютерлердің адам өмірінің әртүрлі салаларына кең араласуымен, программалық жабдықтамаларды пайдаланушылар санының шұғыл артуымен сипатталды. Осылардың нәтижесінде қолданушы интефейстерінің жылдам жетіліп, программалық жабдықтамалар сапасы деген тұжырымдаманың туындауына әкелді. Программау технологияларының жаңа талаптарына жауап бере алатын программалау тілдері (мысалы, Ада) жасалды. Программалау жабдықтамаларын спецификациялау тәсілдері мен тілдері дамыды. Алдыңғы қатарға объектіге бағытталған программалар жасау тәсілдері шықты. Компьютерлік желілер тұжырымдамасы пайда болды.

Өткен ғасырдың соңы мен жаңа ғасырдың басында бүкіл әлемді қамтитын ғаламтордың кеңеюі арқасында барлық дербес компьютерлер Интернет желісіне қосылды. Бұл компьютерлік-желілік ақпаратқа қол жеткізуді реттеу үшін мамандар алдына бірсыпыра тығыз шешімін табатын мәселерді (технологиялық, заңнамалық және этикалық сипаттағы) қойды. Желі бойынша тасымалданатын компьютердегі ақпаратты қорғау өзекті мәселеге айналды, программалық жабдықтамалар жасайтын компьютерлік технологиялар (CASE-технология) дамытылып, программалар спецификацияларымен тығыз байланысты сауалдар туындады.

Сонымен, қоғамды толық ақпараттандыру мен компьютерлендірудің шешуші кезеңі басталды.

Бақылау сұрақтары

1. *Программалау технологиясы деген не? Оның программалаудан айырмашылығы.*
2. *Құрылымдық программалау және оның парадигмасы.*
3. *Жүйелерді декомпозициялау деген не?*
4. *Модуль, модульдік программалау ұғымдары және оның парадигмасы.*
5. *Мәліметтерді абстракциялау және оның парадигмасы.*
6. *Объектіге бағытталған программалау технологиясы түсінігі.*
7. *ОБП технологиясының негізгі сипаттамалары.*
8. *Инкапсуляция түсінігі.*
9. *Мұралау деген не?*
10. *Полиморфизм ұғымы.*
11. *Программалау технологияларының даму кезеңдері. Олардың ерекшеліктері.*
12. *Қазіргі программалу тілдерінің ерекшеліктері.*

4. МӘЛІМЕТТЕРДІ ӨҢДЕУ ПРОЦЕСІН АЛГОРИТМДЕУ

4.1 Негізгі түсініктер мен анықтамалар

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.

Программалау тілі компьютерлерде программаларды орындау ісін атқарады.

Программа – машинаға түсінікті түрде жазылған алгоритм. Программада 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 – программа 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) ...

Алгоритмдеу мен программалаудың заттық негізі болып (компьютерде шығару мақсатында) программаларды құрастыру тәсілдері мен құралдары саналады. Программалар құру үшін программалау жүйелері пайдаланылады.

Программалау жүйесі – программалауды автоматтандыру құралдары. Олар программалау тілінен, осы тілдің трансляторынан, құжаттамаларынан және де программаларды дайындау, әрі орындау құралдарынан тұрады.

Транслятор – бір тілді екінші тілге аудару программасы. Ол интерпретатор және компилятор сияқты екі топқа бөлінеді.

Интерпретатор – бұл командаларды аударып, оларды бірден орындауға арналған трансляторлық программа.

Компилятор – бұл алгоритмдік тілдің конструкцияларын толығымен машиналық кодқа түрлендіретін программа. Есептің нәтижесін алу үшін машиналық кодты орындау керек.

4.2 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 программасын 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 немесе 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.

Программа жасағанда қазірде кеңінен тараған программалау тілінің бірінде алгоритм нақты түрде жазылады. Бізде кең тараған тілдерге – Паскаль, Дельфи, C/C++ жатады. Жазылған программаның 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) да, нәтижені алған соң, оны жазып алу жеткілікті болып табылады. Мәселені шешудің немесе есеп шығарудың көрсетілген алты сатысы күрделі өндірістік есептерде, дипломдық немесе курстық жұмыстарда жиі кездеседі.

4.3 Алгоритмдерді бейнелеу 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;
- программалау тілі арқылы.

Алгоритмдерді **табиғи тіл сөздері арқылы** бейнелеуде – есептеу кезеңдері мазмұны кез келген түрде табиғи тілде жазылады.

Осы тәсілмен келесі мысалдың алгоритмін жазып шығайық.

4.1-мысал. Сандар жиымы (массиві) берілген делік. Осы жиым сандарының көрсетілген аралықта, яғни интервалда толығынан жататынын немесе жатпайтынын тексеру керек. Интервал өзінің шекаралық A және B мәндерімен берілген.

1. Бірінші санды аламыз.
2. Осы сан A-дан B-ға дейінгі интервалға кіретінін салыстыру жолымен тексереміз; егер жауабы "Иә" болса, онда **3-пунктке көшу**, әйтпесе (жауабы – "Жоқ" болса) – **6-пунктке көшу**.
3. Жиымның барлық элементтері қарастырылды ма? Егер жауабы "Иә" болса, онда **5-пунктке көшу**, жауабы "Жоқ" болса, – **4-пунктке көшу**.
4. Келесі элементті қарастырамыз. **2-пунктке көшу.**
5. Мынадай хабарлама шығару: барлық элементтер осы интервалға кіреді. **7-пунктке көшу**.
6. Мынадай хабарлама шығару: элементтер интервалға толығынан кірмейді.
7. Соңы.

Бұл тәсілде көрнекілік жоқ, яғни толық формальдау мүмкіндігі жоқ.

Алгоритмдерді **формулалық-сөздік тәсіл арқылы** бейнеленуі – тапсырманың математикалық символдар мен өрнектердің және сөздердің араласуымен берілуі болып табылады.

4.2 мысал. Үшбұрыш ауданын оның берілген (*a, b, c*) үш қабырғасының ұзындығы арқылы есептеу алгоритмін Герон формуласы арқылы құру керек болсын делік. Мұны былай көрсете аламыз.

1. үшбұрыштың жарты периметрін есептеу $p=(a+b+c)/2$;
2. үшбұрыштың ауданын есептеу

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

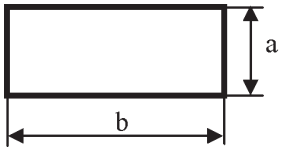
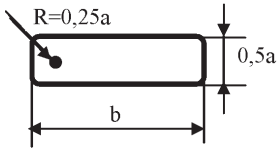
3. нәтиже ретінде S мәнін шығарып, алгоритм жұмысын аяқтау.

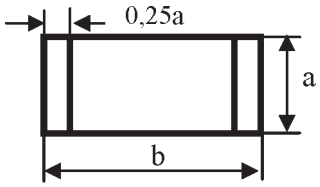
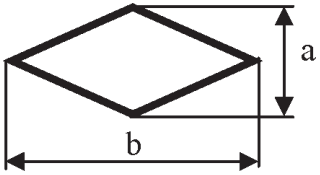
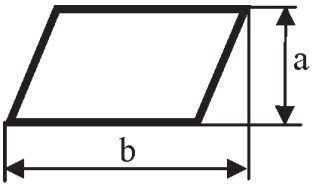
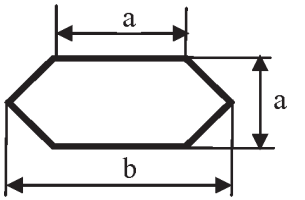
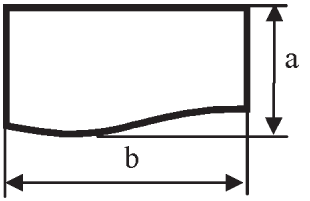
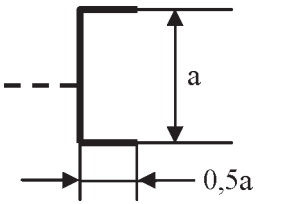
Бұл тәсілді пайдаланғанда, алгоритмді кез келген деңгейде айқындап көрсетуге болады, бірақ формальды түрде анық бейнелеу қиын.

Алгоритмді **графикалық түрде блок-схемалар арқылы** көрсету – оның логикалық құрылымын графикалық түрде бейнелеу болып саналады. Мұнда мәліметтерді өңдеудің әрбір кезеңі атқарылатын операцияға сәйкес әртүрлі геометриялық фигуралар (блоктар) түрінде көрсетіледі (4.1-кесте).

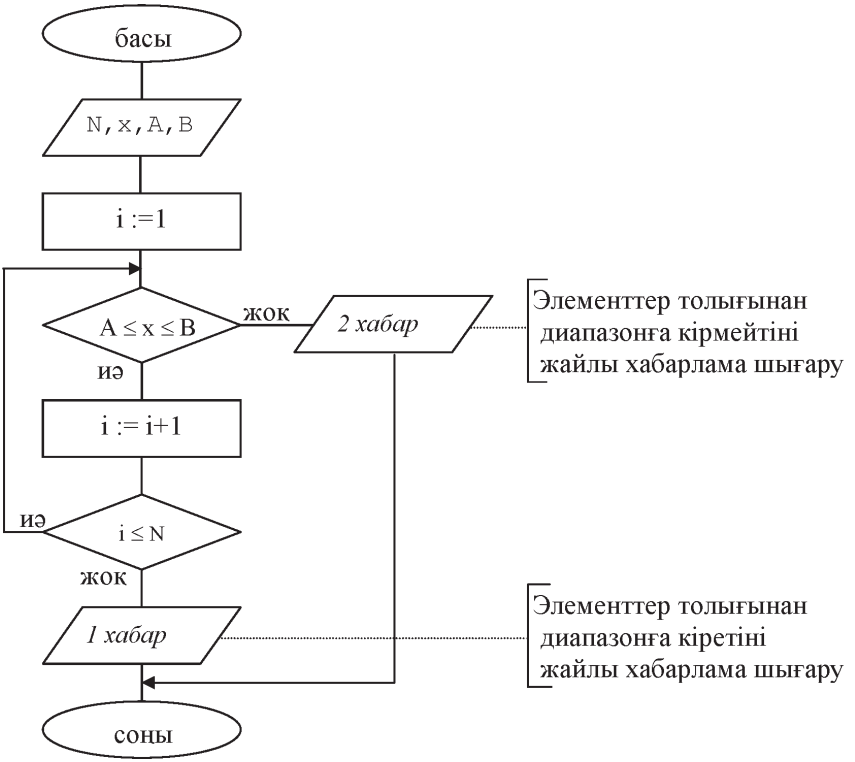
Сонымен алгоритм блоктармен немесе геометриялық көпбұрыштар түріндегі фигуралармен өрнектеледі. Әр блоктың ішіне орындалатын іс-әрекеттің (амалдың) мазмұны жазылады. Символдардың (блоктардың) бір кіру және бір шығу сызықтары болуға тиіс.

4.1 кесте. **Алгоритмдерді бейнелеу блоктары**

Іс-әрекет аты	Блоктың пішімі	Атқаратын жұмысы
Процесс		Математикалық өрнектерді есептеу
Басы – соңы		Алгоритмдерді бастау, аяқтау

Алдын ала анықталған процесс (ішкі программа)		Қосалқы программаларға кіру және шығу
Шешім		Есеп шығару жолын таңдау
Енгізу-шығару		Мәліметтерді енгізу және шығару
Модификация		Цикл басы
Құжат		Нәтижені баспаға (қағазға) шығару
Түсініктеме		Схеманы, формулаларды түсіндіру

4.1-мысалды графикалық тәсілмен орындау келесі 4.1-суретте бейнеленген.



4.1 сурет. Алгоритмді блок-схема арқылы графикалық түрде бейнелеу

Жалған кодтар (псевдокодтар) – нақты программалау тілінің синтаксистік ерекшеліктерін есепке алмай, тек формальды түрде программа логикасын бейнелеуге мүмкіндік береді. Бұл тәсіл программалау тілінің операторлары мен табиғи тіл сөздерінің араласуы арқылы құрастырылып, блок-схема орнына пайдалануға болатын программа логикасын бейнелеу құралы болып табылады.

Алдыңғы 4.1 мысалдағы алгоритмді псевдокод арқылы былай жазуға болады:

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 ($i \le N$) THEN
интервалды тексеруге көшу
ELSE
интервалға элементтер толық кіретіні жайлы хабарлама шығару
Соңы

Ал қарапайым алгоритмдік тіл деп аталып, алгоритмді программалау тілінің терминдерін қазақ (орыс) тілінде жазу арқылы да өрнектеу *жалған кодтар арқылы* жазу сияқты алгоритмді компьютерге дейін жазу үшін кеңінен қолданылып жүр. Мұны олардың ағылшын тіліне негізделіп жасалған программалау тілдеріне жақындығымен түсіндіруге болады.

4.3 мысал. $Z = ax^2+bx+cos(ax^2+b)-tg(ax^2+b)$ функциясының мәнін есептеп шығару алгоритмін жасау керек болсын делік. Мұнда алдымен *a, b, x* нақты мәндерін енгізіп, жақшада тұрған ax^2+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

Программалау тілдері **–** программаларды компьютерде тікелей орындауға арналған алгоритмдерді жазу тәсілі. *Программа – алгоритмнің компьютерге түсінікті түрде жазылуы*.

Әрбір машинаның өз тілі (машиналық тіл) болады және ол тек осы тілдегі программаларды, яғни командалар тізбегін орындай алады. Программаларды машиналық тілде жазу өте күрделі, әрі адамды шаршататын жұмыс болып табылады. Программалаушылардың жұмыс өнімділігін арттыру мақсатында жасанды тілдер, яғни программалау тілдері қолданылады. Мұндайда жасанды тілде жазылған программа машиналық тілге аударылуы тиіс. Аудару жұмысын, яғни программаны 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. Мұндай транслятор жұмысының қорытындысы қажетті нәтижелерді алу болып табылады.

Паскаль, C/C++ тілдерінің трансляторы – компилятор түрінде болады. Мұнда бастапқы жазылған программа мәтіні машина тіліне аударылады да, *объектілік модуль* деп аталатын программа коды шығарылады. Сонан соң объектілік модуль *программа аралық байланыс редакторы* деп аталатын программа арқылы өңделгеннен кейін барып қана жұмыс істеуге дайын болады.

4.4 Алгоритмдердің негізгі канондық құрылымдары

Программалаудың ең негізгі әрі күрделі кезеңі алгоритм құру болып табылады. Программалауды үйретудің басында алгоритмді тілдермен байланыстырмай, блок-схема немесе псевдокод тәрізді тәсілдермен құруды түсіндіру керек. Осындай "таза" алгоритмдеу жолын меңгерген студенттер кейін программалау тіліне оңай көшеді. Бұл тарауда алгоритмдерді блок-схема немесе алгоритм схемасын құру арқылы құру негіздері қарастырылады.

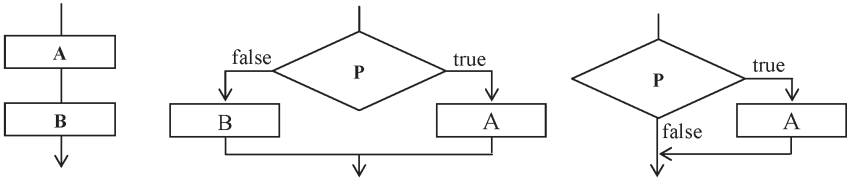
Программалау теориясында кез келген күрделі есептің шығару жолын, яғни кез келген программаны төмендегідей үш түрлі басқару құрылымын пайдалану арқылы жазып шығуға болатындығы дәлелденген:

- сызықтық құрылым немесе операторлар тізбегі;
- тармақты құрылым немесе шартты оператор;
- қайталау немесе циклдік оператор.

Осындай канондық құрылымдардан тұратын программаны регулярлық программа деп атайды, олардың бір ғана кіру нүктесі мен бір ғана шығу нүктесі болады. Дәл осы үш құры-

лым құрылымдық программалаудың базалық конструкциялары деп аталады. Программадағы кез келген операторды оның кіру нүктесі арқылы тауып орындауға болады (осы тәсілмен табылмайтын операторлар және шексіз циклдер болмауы тиіс). Мұндай программаны басқару ісі жоғарыдан төмен қарай жүргізіледі. Түсініктеме мәтін (комментарий) қосылған осындай программалар оқуға және түсінуге жеңіл болып есептеледі.

if P then A ; **(Паскаль, Бейсик тілдерінде)**
if P A ; **(C/C++ тілдерінде)**



а) сызықтық құрылым ә) толық тармақты құрылым б) толымсыз тармақты құрылым

4.2 сурет. Алгоритмдерді бейнелеудің канондық құрылымдары

1) сызықтық құрылым (4.2 а сурет), ол программалау тілдері арқылы былай жазылады:

A; B; (Паскаль, C/C++ тілдерінде)

мұндағы **A** және **B** әрекеттері мыналардың бірі болуы тиіс:

- кез келген жеке оператор;
- белгілі бір функцияны шақырып, одан кері оралу;
- басқа бір басқару құрылымы.

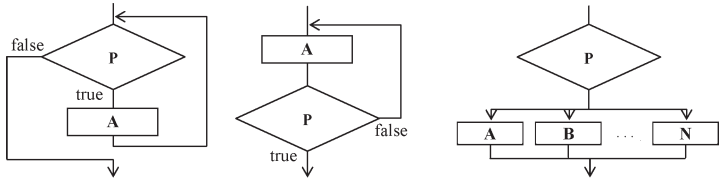
2) тармақты құрылым (4.2 ә сурет), программалау тілдерінде былай жазылады:

if P then A else B; (Паскаль тілінде)
if P A; else B; (C/C++ тілдерінде)

P-ны тексеру предикат болып табылады, яғни ол мәні ақиқат (true) немесе жалған (false) бола алатын логикалық өрнектен немесе шарттан тұратын болады. Бұл құрылым толымсыз (қысқаша) түрде болуы мүмкін, онда логикалық өрнектің мәні жалған болғанда ешқандай әрекет орындалмайды. Мұндай құрылым түрі төменде көрсетілген (4.2 б сурет), ал программалау тілдерінде:

if P then A ; (Паскаль тілінде)
if P A; (C/C++ тілдерінде)

Енді циклдік құрылымдарды қарастырайық, олар алгоритмнің белгілі бір бөлігін бірнеше рет қайталау ісін жүзеге асырады. Циклдік алгоритмдердің бірнеше түрі бар.



а) «әзірше» циклдік құрылымы ә) «дейін» циклдік құрылымы б) таңдау тармақты құрылымы

4.3 сурет. Алгоритмдерді бейнелеудің циклдік және таңдау құрылымдары

3) қайталау – "әзірше" циклі (4.3 а сурет)

while P do A ; (Паскаль, C/C++ тілдерінде)

Мұнда А әрекеті Р предикат мәні ақиқат болып тұрса, қайталана береді. Сондықтан А әрекеті орындалуы кезінде Р-ға әсер ететін айнымалылар мәні өзгеруі тиіс. Бұлай болмаған жағдайда шексіз цикл орын алады. Предикат мәні А әрекетіне **дейін** анықталады, сол себепті кейде А әрекеті бір де бір рет орындалмауы да мүмкін.

қайталау – "дейін" циклі (4.3 ә сурет)

repeat A until P; (Паскаль тілінде)

do A while P; (C/C++ тілдерінде)

Мұндағы қайталау кем дегенде бір рет орындалады, өйткені шарт А әрекетінен кейін тексеріледі. А әрекеті предикат мәні жалған болған кезде орындалмайтын болады.

4) таңдау – **switch (case)** ауыстырғыш (көп тармақты) құрылымы программалауды жеңілдететін мүмкіндік болып табылады. Таңдау құрылымы бірнеше мүмкіндіктердің біреуін ғана орындау кезінде қолайлы болып табылады (4.3 б сурет), ал программалау тілдерінде:

Паскаль тілінде: **C/C++ тілдерінде:**

```
case P of                    switch (P)
   A: операторлар;           { case A: операторлар; break;
   B: операторлар;             case B: операторлар; break
   ...                               ...
   N: операторлар;             case N: операторлар; break
    end;                     }
```

Р мәніне байланысты A, B, …, N әрекеттерінің бірі орындалады да, сонан соң мұнан кейін орналасқан құрылымдар атқарылады.

4.5 Қарапайым алгоритмдер құру

Күрделі алгоритмдерді құру үшін қарапайым канондық (бірыңғайланған) алгоритмдік құрылымдар қолданылады. Олар сызықтық, тармақталу және цикл құрылымдарынан тұрады.

Программалау теориясында кез келген күрделі программаны осы үш түрлі құрылымнан құрастыруға болатыны дәлелденген.

Негізгі конструкцияларды пайдалану мақсаты – қарапайым құрылымды программа алу болып саналады. Мұндай программалар оңай оқылады, түзетіледі және керек болса, оңай өзгертіледі.

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.

4.4 мысал. $y = a+b$ формуласы бойынша y -ті табу керек болсын.

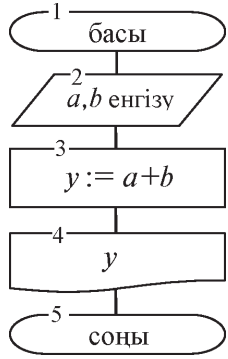
Мұнда формула бойынша есептеу тіктөртбұрыш арқылы кескінделетін есептеу блогы (3-блок) арқылы өрнектеліп, нәтижені қағазға басу үшін көпбұрышты құжат алу блогын (4-блок) пайдаланып, оның ішіне нәтиженің атауларын жазамыз. Осы көрсетілген $y=a+b$ формуласын есептеу үшін a және b-ның сандық мәндерін программаға енгізіп (2-блок), содан кейін қосу амалын орындап, ақырында y-ті экранға (қағазға) басып шығарып, жұмысты тоқтатамыз. Осы алгоритмнің схемасы 4.4-суретте көрсетілген, ал оның жанында Паскаль және С тіліндегі программасы жазылған. Программа мәтінін құру жолдарын келесі тарауларда қарастырамыз.

Паскальдағы программа:

```
Program P(input,output);
  var a,b,y: integer;
begin
write('a, b=');
  readln(a,b);
y:= a + b;
writeln('y=',y:6:3);
readln;
end.
```

С тіліндегі программа:

```
#include <stdio.h>
#include <conio.h>
main()
{
 int a,b,y;
 printf("a, b =");
 scanf("%i%i",&a,&b);
 y = a + b;
 printf("y = %i", y);
 getch();
}
```

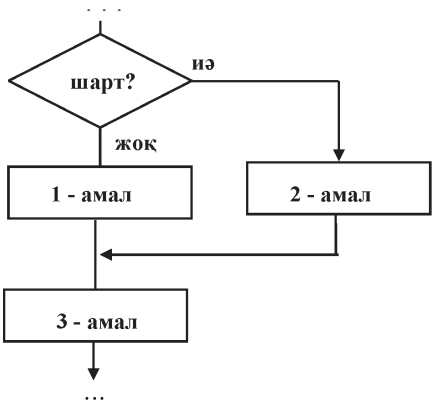


4.4-сурет. Алгоритм схемасы

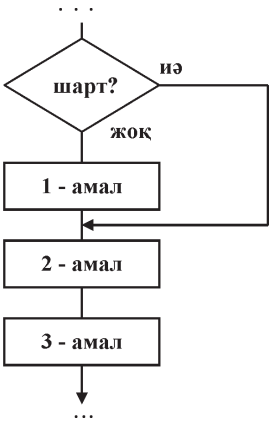
2. Тармақталу алгоритмдері. Тұрмыста кездесетін алгоритмдер әр түрлі болып келеді. Олардың жиі кездесетін түріне алгоритмнің белгілі бір шарттың орындалуына немесе орындалмауына байланысты тармақталып бірнеше жолдарға бөлінуі жатады.

Тармақталу алгоритмінің құрылымы қарапайым болып келеді. Мұнда арифметикалық теңсіздік (теңдік) түрінде берілген логикалық шарт тексеріледі. Егер ол орындалса, онда алгоритм бір жолмен, ал орындалмаса екінші жолмен жүзеге асырылады, яғни есепті шығару жолы тармақталып екіге бөлініп кетеді. Тармақталу алгоритмдеріне шартты тексеру блогы міндетті түрде кіреді. Ол ромб түрінде кескінделіп, басқа блоктармен 1 кіру және 2 шығу сызықтары арқылы байланысады. Көбінесе тармақталу алгоритмдері екі түрде кездеседі, олар жоғарыда көрсетілген канондық құрылымдардың толымды және толымсыз түрлеріне сәйкес келеді де, "таңдау" және "аттап өту" мүмкіндіктерін іске асыруға көмектеседі.

"Таңдау" жолымен тармақталуда берілген шарт тексеріледі (4.5-сурет), егер шарт орындалса (шарт ақиқат - true болса), онда 2-амал орындалып, содан кейін келесі 3-амалға көшеміз. Ал, егерде шарт орындалмаса (орындалу мүмкіндігі жалған - false болса), онда 1-амал атқарылып, сонан соң 3-амал атқарылады.



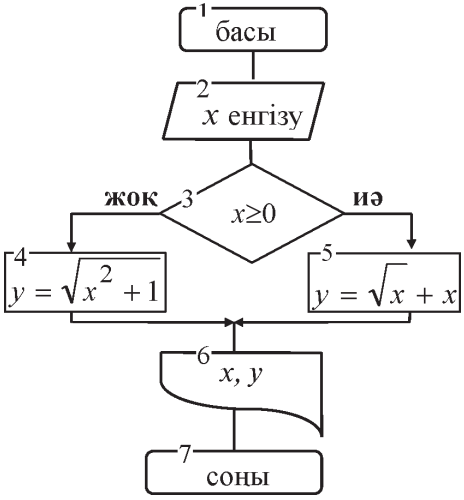
4.5 -сурет. "Таңдау" алгоритмі



4.6-сурет. "Аттап өту" алгоритмі

"Аттап өту" (4.6-сурет) алгоритмінде шарт орындалса, 1-амалды аттап өтіп, бірден 2-амалды, содан кейін 3-амалды орын-

даймыз. Ал шарт жалған болса, онда 1-амал міндетті түрде орындалып, одан кейін 2- және 3-амалдар жүзеге асырылады. Тармақталу кезеңінде шартты тексеру блогы орындалуы барысында, алгоритмнің екі мүмкіндігінің тек біреуі ғана таңдап алынып жүзеге асырылады да, ал екінші таңдап алынбаған тармақ біріктіру нүктесіне дейін орындалмай қалады. Енді осыған нақты мысалдар келтірейік.



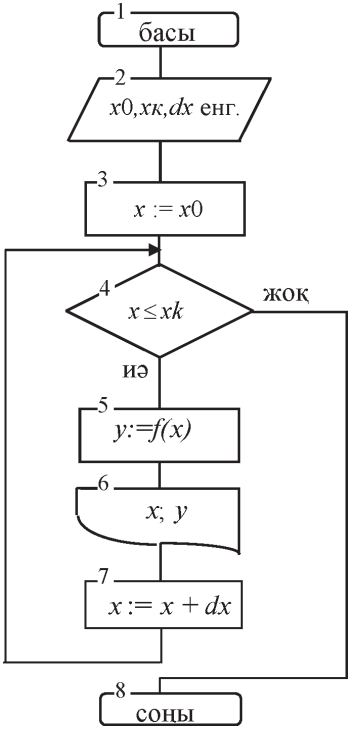
4.7-сурет. Тармақталу алгоритмі

4.5-мысал. y функциясын төмендегі формула бойынша есептеп шығарайық.

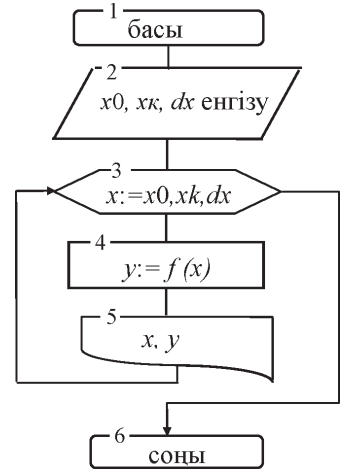
$$y = \begin{cases} \sqrt{x} + x, & x \geq 0 \\ \sqrt{x^2 + 1}, & x < 0 \end{cases}$$

Мұнда *x* айнымалысының таңбасына (оң, теріс) байланысты не жоғарғы, не төменгі формуланы таңдап алып, сол арқылы *y* функциясының мәнін табамыз (4.7 сурет). 2-блоктың орындалу барысында *x* айнымалысына белгілі бір мән беріледі де, ол мән енгізу операторлары арқылы программаға енгізілуі тиіс. Бұдан кейін енгізілген мәннің оң немесе теріс екендігі үшінші шартты тексеру блогы арқылы айқындалады. Шарттың "ақиқат" (иә) немесе "жалған" (жоқ) болуына байланысты 4- не 5-блоктардың бірі ғана орындалып, "таңдау" орындалады. 6-блок *x* айнымалысының және *y* функциясының сандық мәндерін экранға немесе қағазға басып шығарады.

3. 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 бөлік-



4.8-сурет. Қарапайым циклдік алгоритм



4.9-сурет. Модификаторлы циклдік алгоритм

терін цикл деп атайды. Осы бірнеше рет қайталанатын бөлігі бар алгоритмдер тобы циклдік алгоритмдерге жатады. Циклдік алгоритмдерді пайдалану оларды кейіннен программаларда 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.

Практикада белгілі бір айнымалының сандық мәніне байланысты орындалатын арифметикалық циклдер жиі кездеседі. Мұнда арифметикалық прогрессияға ұқсас болып келетін циклдер ең қарапайым арифметикалық цикл болып табылады.

Оны басқару қайталану кезеңінде прогрессияның заңына сәйкес тұрақты шамаға өзгеріп отыратын цикл параметрінің сандық мәнімен байланысты болуы тиіс.

Цикл орындалуы алдында оның айнымалы аргументі – параметрі (x) алғашқы мәнге ($x0$) ие болуы керек, сонан кейін қайталану кезеңінде цикл параметрі белгілі бір шамаға (dx қадамға) өзгере отырып, ол алдын ала берілген ең соңғы мәнге (xk) дейін жетуі қажет.

Алгоритмнің орындалу барысы-

нда цикл параметрі, мысалы, *x* өзінің ең алғашқы *x*0 мәнінен ең соңғы *xk* мәніне дейін тұрақты шамаға (*dx*) өзгеріп отырады. Осының нәтижесінде *x* мынадай мәндерді қабылдайды: *x*0, *x*0+*dx*, *x*0+2*dx*, ... , *x*0+*(n*-1*)dx*, *xk*, мұндағы *n* – циклдің қайталану саны, ол былай анықталады:

$$n=\left[\frac{xk\text{-}x0}{dx}\right]+1,$$

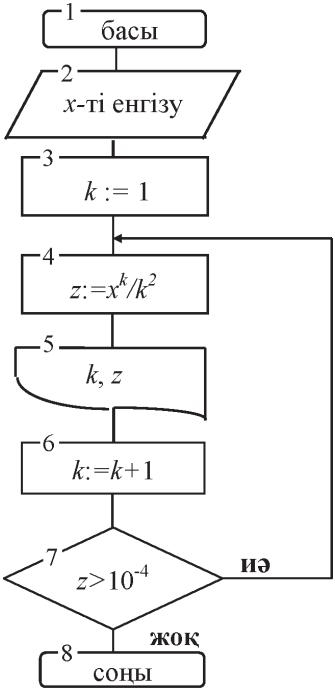
мұнда [...] – өрнектің бүтін бөлігі алынатынын көрсетеді, *n* – циклдің қайталану саны әрқашанда бүтін натурал сан болуы тиіс, егер ол аралас сан болса, онда оның бөлшегі алынып тасталады.

Арифметикалық цикл үшін *y=f(x)* функциясының есептелу жолы алгоритм ретінде 4.8-суретте көрсетілген. Мұндағы 3-ші, 4-ші, 7-блоктар циклді ұйымдастыру үшін қажет. Олар цикл параметрінің алғашқы мәнін, өзгеру қадамын белгілеп және оның ең соңғы мәніне жеткен-жетпегенін тексереді. Ал 5- және 6-блоктар бірнеше рет қайталанып циклдің өзін құрайды. 4-блок шартты тексеріп қайталану процесін ұйымдастырады.

Алгоритм схемасын салуды және программаны жазуды жеңілдету үшін цикл алгоритмдері ықшамдалған түрде "модификатор" немесе "цикл басы" блогын пайдалану арқылы жазылады. Онда 4.8-суретте көрсетілген 3-ші, 4-ші, 7-блоктардың орнына "цикл басы" блогы орналасады.

Ол алтыбұрыш тәрізді геометриялық фигурадан тұрады және оның міндетті түрде екі кіру және екі шығу сызығы болуға тиіс. Осы блокты пайдалану арқылы жоғарыда келтірілген алгоритм 4.9-суретте көрсетілген түрде кескінделеді. Параметрдің алғашқы *x0* мәні оның соңғы *xk* мәнінен кем болса, онда оның қадамы *dx* оң сан болады. Керісінше, параметрдің алғашқы мәні оның соңғы мәнінен артық болса, онда қадам теріс сан болады.

4. 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 алдын ала циклдің неше рет қайталанатыны белгісіз. Енді осындай циклдерге мысал келтірейік.



4.10-сурет. Қадамдық цикл алгоритмі

4.6 *мысал.* $Z = \dfrac{x^k}{k^2}$ функциясының мәндерін k = 1, 2, 3, ... және Z 0.0001-ден артық болған жағдайда есептейік, мұндағы $0 \le x \le 1$. Бұл мысалда алдын ала цикл неше рет қайталанатынын айта алмаймыз, өйткені бізде тек k параметрінің алғашқы мәні мен қадамы ғана белгілі. Сонымен қатар Z функциясының 0.0001-ден артық болуы циклді қайталау шарты болып есептеледі ($Z > 0.0001$). 4.10-суретте осы есептің алгоритм схемасы көрсетілген.

4.6 Программалау тілдері

Алгоритмдерді компьютерге түсінікті мәтін ретінде жазуға арналған қарапайым жасанды тіл программалау тілдері деп аталады. Әрбір компьютердің өзінің машиналық программалау тілі болады, оны командалар тілі немесе кодтар (арнайы таңбалау) тілі дейді. Компьютер тек өз тілінде, яғни машиналық тілде жазылған программаларды ғана орындай алады. Алайда, машина тілінде программа жазу күрделі жұмыс, өйткені ол тек екілік (он алтылық) жүйедегі кодтардан тұрады және әр машинада әр түрлі машиналық тіл қолданылады.

Программа жазуды жеңілдету үшін математикалық формулаларды кеңінен қолданатын, ағылшын тілінің негізінде жасалған алгоритмдік тілдер Бейсик, Паскаль, Фортран, С, С++, т.б. кеңінен қолданылады. Алгоритмдік немесе программалау тілі – жазу ережелері қарапайым жасанды тіл. Оның машина тілдерінен айырмашылығы – табиғи ағылшын тілі негізге алынып, кең тараған математикалық таңбалармен толықтырылып жасалған. Сондықтан алгоритмдік тілдерде программа жасау адамдарға әрі жеңіл, әрі ыңғайлы болып келеді. Алгоритмдік тілдер автоматты

түрде компьютердің көмегімен аудармашы программалар арқылы машиналық тілге көшіріледі.

Алгоритмдік тілді машина тіліне тікелей аударатын үлкен программаларды транслятор болып табылады. Алгоритмдік тілдерді пайдалану программалауды жеңілдете отырып, компьютерде есеп шығару процесін оңайлатады, алайда онда есеп шығару уақыты аздап көбейеді.

Алгоритмдік тілдер *машинаға* және *проблемаға бағытталған* болып екіге бөлінеді. Машинаға бағытталған тілдердің машина тілінен айырмашылығы, олар компьютердің ерекшеліктерін есепке ала отырып әріптерді де пайдаланады. Қазіргі кезде машинаға бағытталған тілдерде маман программалаушылар жұмыс істейді. Оларға автокод, макроассемблер, ассемблер тәрізді тілдер жатады.

Проблемаға бағытталған тілдер шығарылатын есептердің ерекшеліктерін еске ала отырып, есептің математикада жазылу тіліне жақындастырылады. Бұларға – Бейсик, Фортран, Паскаль, C/C++, т.с.с. тілдер жатады.

Қазіргі кезде бес жүзге жуық алгоритмдік тілдер тараған. Олардың әрқайсысы белгілі бір мақсаттарда қолданылады. Мысалы, Фортран – ғылыми-техникалық (инженерлік) есептерді шығару үшін, Паскаль – оқып үйренуде, ал C/C++ тілі үйрену үшін де, өндірістік есептерде де, операциялық жүйелер жазу үшін де қолданыла беретін кең қолданыстағы тіл болып табылады.

Әр түрлі процессорлар типтерінің машиналық командалары да әр түрлі болады. Егер программалау тілі нақты бір процессор типінің ерекшеліктерін есепке ала отырып жұмыс істейтін болса, онда ол *төменгі деңгейдегі программмалау тілі* деп аталады. Ең төменгі деңгейдегі программалау тілі *ассемблер* болып саналады, ол машиналық кодтың әрбір командасын мнемоника деп аталатын арнайы символдық белгілермен жазып шығады. Төменгі деңгейдегі программалау тілдері көмегімен өте тиімді және ықшам программалар жасалады, мұнда программалаушы процессордың барлық мүмкіндіктерін толық пайдалана алады. Процессорлардың әр түрлі модельдерінің өз ассемблерлері болатындықтан, мұнда жазылған программа тек осы ортада ғана

орындалуы тиіс. Сондықтан мұндай тілдер шағын жүйелік программалар мен драйверлер жазуға ыңғайлы болып табылады.

Жоғары деңгейдегі программалау тілдері нақты бір компьютерлік архитектура ерекшеліктерін есепке алмай жұмыс істейді, сондықтан олар процессорға сәйкес транслятор жазылған басқа компьютерлерге оңай көшіріледі. Жоғары деңгейдегі программалау тілдерінде программа жазу машиналық тілдерге қарағанда анағұрлым жеңіл.

Жоғары деңгейдегі программалау тілдері:

1. Фортран – өткен ғасырдың 50-ші жылдары жасалған, машиналық тілге компилятор арқылы аударылатын алғашқы тіл. Мұнда программалаудың бірсыпыра маңызды түсініктері жүзеге асырылған еді. Осы тілде статистикалық есептерге арналған қарапайым кешендерден бастап, жер серіктерін басқаруға арналған есептерге дейін көптеген кітапханалық программалар жасалған болатын, сондықтан ол көптеген мекемелерде осы кезге дейін пайдаланылып келеді.

2. Кобол – өткен ғасырдың 60-шы жылдары басында жасалған, экономикалық есептер шығаруда қолданылатын, машиналық тілге компилятор арқылы аударылатын тіл. Коболда сыртқы мәлімет сақтау құрылғыларында сақталатын көлемді мәліметтерді өңдеудің қуатты мүмкіндіктері жүзеге асырылды.

3. Паскаль – өткен ғасырдың 70-ші жылдары аяғында швейцар математигі Никлаус Вирттің программалауды үйрету мақсатында қолдану үшін арнайы жасаған тілі. Ол адамның алгоритмдік ойлау қабілетін қалыптастыратын, көпшілікке түсінікті қысқа программа жазуға қолайлы, алгоритмдеудің негізгі тәсілдерін көрсетуге және де ірі жобаларды да жүзеге асыра алатын тіл болды.

4. Бейсик (Basic) – бұл да өткен ғасырдың 60-шы жылдары программалауды үйренушілерге арналып жасалған қарапайым тіл болды. Бұған да арнайы компиляторлар мен интерпретаторлар жасалып, ол көпшілікке кең тараған, игеруге жеңіл программалау тілдерінің бірі болды.

5. С – 1970-ші жылдары шықты, ол алдымен көпшілікке арналып жасалмаған тіл еді. С тілі нақты процессорға тәуелді емес, қысқа, әрі тиімді программалар жасайтын ассемблер орнына пайдалану үшін жоспарланған тіл болатын. С тілінің көптеген

қасиеттері Паскаль тіліне ұқсас болғанымен, оның компьютер жадымен тікелей жұмыс істей алатын жаңа мүмкіндіктері бар еді. Осы тілде көптеген қолданбалы және жүйелік программалармен қатар Unix операциялық жүйесі де жазылып шыққан болатын.

6. C++ – объектіге бағытталған С тілінің кеңейтілген түрі, оны 1980 ж. Бьярн Страуструп жасап шығарды.

7. Java тілін 1990-шы жылдары басында C++ тілін негізге ала отырып, Sun компаниясы шығарды. Ол C++ тілінен төменгі деңгейдегі мүмкіндіктерді алып тастап, қолданбалы программалар жасауды оңайлату мақсатында жасалды. Тілдің басты ерекшелігі – оның машиналық кодқа түрленбей, бірден платформалы-тәуелсіз байт-кодқа (әр команда бір байт орын алады) түрленуі болды. Осындай код виртуалды Java-машина (JVM) интерпретаторы көмегімен орындалады.

8. C# ("Си-шарп" деп айтылады) тілін Microsoft компаниясының маманы Андерс Хейлзберг (Anders Hejlsberg) жаңа объектіге бағытталған программалау тілі ретінде С, C++, Java, Паскаль тілдерінің негізінде жасап шығарған. C# тілі Microsoft фирмасының .NET (дот-нет) архитектурасы үшін программалар жазуға арналған тілі. .NET – программалау технологиясындағы жаңа платформа, ол желіге қосылған компьютерлерге арналып жасалған. Бұл технология Visual Studio.NET деп аталады да, оның жұмыс ортасында Visual Basic, басқарылатын C++ және C# тілдерінде программалар жасау қарастырылған, бірақ ол осылармен ғана шектеліп қалмайды. C# тілінің мүмкіндігі Java тілімен қатарлас, осы екі тіл қазіргі алдыңғы қатарлы технологияларға жатады.

Бақылау сұрақтары

1. *Алгоритм және алгоритмдеу ұғымдары.*
2. *Компьютерде орындалатын алгоритмдердің қандай қасиеттері болады?*
3. *Алгоритм мен программаның қандай ұқсастықтары мен айырмашылықтары бар?*
4. *Оператор, айнымалы дегеніміз не? Транслятор, компилятор, интерпретатор ше?*
5. *Компьютерде есеп шығару кезеңдері.*

6. *Алгоритмдерді бейнелеу жолдары. Олардың ерекшеліктері.*
7. *Алгоритмдерді графикалық жолмен бейнелеу деген не?*
8. *Алгоритмдердің негізгі канондық құрылымдары.*
9. *Алгоритм схемаларының әр түрлі блоктары, олардың бейнеленуі, байланыстары.*
10. *Сызықтық, тармақталу және циклдік алгоритмдер.*
11. *Қадамдық циклдер және олардың ерекшеліктері.*
12. *Программалау тілдерінің түрлері, кең таралған тілдерге қандай тілдер жатады?*

Тапсырмалар

1. Берілген формулалар бойынша арифметикалық өрнектерді есептейтін алгоритмдердің блок-схемаларын сызып шығыңдар:

а) $z = \dfrac{1}{x-1} + \sqrt{x+1};$

ә) $y=(5x^2-4)(x^2+7);$

б) $u = \sin^3 \dfrac{x}{5} \cos x^2 + e^{\sqrt{a}x};$

в) $f(x) = \begin{cases} 5x^2 + 6, \text{егер } x < 5; \\ x^3 + 7, \text{егер } x \geq 5; \end{cases}$

г) $f(x) = \begin{cases} xtgx - \sin x, \text{егер } x < 1.5; \\ x^3 + \sin x, \text{егер } 1.5 \leq x < 2.5; \\ 3x^3 + 5, \text{егер } x \geq 2.5; \end{cases}$

д) $z = \dfrac{1}{x-1} + \sqrt{x+1};$ мұнда x айнымалысы 2-ден 5-ке дейін 0,5 қадамымен өзгереді.

е) Нақты a және бүтін n сандары берілген.

$$s = \frac{1}{a} + \frac{1}{a^2} + \frac{1}{a^4} + \ldots + \frac{1}{a^{2n}}$$

қосындысын табу алгоритмін құру керек.

ж) s = 1 + 1/2 + 1/3 + … + 1/n + … қосындысын 0,0001 дәлдігімен

табу керек.

2. Квадрат теңдеуді ($ax^2+bx+c=0; a \neq 0$) шешу алгоритмін тұрғызыңдар.
3. Жазықтықта үш нүкте - $A(x_1,y_1)$, $B(x_2,y_2)$ және $C(x_3,y_3)$ берілген:
   а) солардан үшбұрыш құруға бола ма? Егер болатын болса, Герон формуласы бойынша оның ауданын есептеп шығаратын алгоритм құрыңдар;
   ә) бұл үшбұрыш тең қабырғалы немесе тең бүйірлі болатынын анықтайтын алгоритм құрыңдар.
4. 1-ден *N*-ге дейінгі сандардың қосындысын есептейтін алгоритм құрыңдар. *N*-нің мәні енгізіледі.
5. Пернетақта арқылы 5 сан енгізіңдер. Егер сан 10-нан кем болса, онда осы санды және оның квадратын табыңдар.
6. Ұзындықтың 1-ден 20 дюймге дейінгі мәндерін сантиметрге (1 дюйм = 2,54 см) айналдыратын және оны экранға шығаратын алгоритм құрыңдар.
7. Бүтін x және y сандары берілген. Егер екі сан да жұп болса оларға 1-ді қосыңыз; егер тек біреуі жұп болса, онда олардың көбейтіндісін табыңыз; қалған жағдайда сандарды өзгеріссіз қалдырыңыз.
8. Бүтін a, b, c, d сандары берілген. Егер a>=b>=c>=d болса, онда барлық сандарды нөлге теңестіріңіз; егер a<b<c<d болса, онда әр санды 1-ге өсіріңіз; қалған жағдайда әр санды 1-ге кемітіңіз.

5. C/C++ ТІЛДЕРІ НЕГІЗДЕРІ

С тілі 1969 – 1973 жылдары арасында АҚШ-та Bell Telephon Laboratories компаниясының программалаушылары Дэннис Ритчи мен Кен Томпсон бастауымен дүниеге келді. Бұл тілдің негізі Алголдан басталып, Паскаль және ПЛ/1 тілдерімен қатар пайда болды.

С тілінің шығуы UNIX операциялық жүйесінде программалаумен тығыз байланысты, өйткені бұл жүйе ассемлерде және осы С тілінде жазылып шыққан болатын. UNIX жұмыс істеу ортасы С тілін жүйелік программалау тілі ретінде елге таныстырды, ол компиляторлар мен операциялық жүйелер жазу үшін қолайлы деп саналды, кейіннен С тілі кез келген салада программалар жазуға да өте қолайлы тіл болып табылатыны анықталды.

Алғаш рет UNIX 1969 жылы Нью-Джерси штатындағы Bell фирмасының лабораториясында PDP-7 компьютерінде жасалып шықты. UNIX жүйесі PDP-7 компьютерінің ассемблер тілінде жазылды. Бұдан соң сол лабораторияның жетекшісі Кен Томпсон 1970 жылы В деп аталған жаңа тілге арнап компилятор жасап шықты. Осы тілді С (ағылшынша Си) тілінің негізі деп атауға болады.

С тілінің алғашқы нұсқасы Б.Кернигана мен Д.Ритчидің "С программалау тілі" деген кітабы арқылы бірнеше рет жарық көрді (1985-1991 жж.). Осы кітап тілдің стандарты сияқты болды, бірақ мұнда аздаған кемшіліктер бар екендігі анықталды. Сондықтан оны жетілдіру мақсатында Америка ұлттық стандарттар институтында (ANSI) Техникалық комиссия құрылып, 1983 ж. ANSI С деп аталған тіл стандарты бекітілді. Мұнда тек программалау мәселелері ғана қарастырылмай, кең таралып келе жатқан IBM PC компьютерлеріне арналған компиляторлар жасау істері де қамтылды.

С тілін ары қарай жетілдіріп, сол Bell Telephon Laboratories қызметкері Бьерн Страуструп "Кластары бар С" тілін жасап шығарды, ол 1983-жылы С++ тілі деген атқа ие болды. С++ тілі объектіге бағытталған программалау тілі болып саналады, ол бұрынғы С тіліне кластардың енгізілуімен ерекшеленеді.

1986 жылы Бьерн Страуструп "С++ программалау тілі" кітабын шығарды, сол кезден бері тілдің бірнеше нұсқалары пайда болып, ол қазіргі ең көп тараған тілдердің біріне айналды. Тілдің халықаралық стандарты 1998 ж. бекітілді, бұл стандарт бойынша С++ тілі С тілінің 1990 ж. бекітілген стандартына негізделеді. Бұл оқулықтағы мысалдар осы стандартты пайдаланады.

Бірсыпыра фирмалар С++ тіліне арнап компиляторлар жазды, мысалы, Borland International фирмасы 1989 ж. жасаған біріктірілген программалау ортасы TurboC++ жүйесін дүниеге келтірді. Ол DOS ортасында жақсы жұмыс істеді. Ал 1992 ж. жасалған Borland C++ жүйесі Windows ортасында да жұмыс істейтін жақсы компилятор болып табылады.

Сонымен С/С++ программалары дайындалып орындалатын біріктірілген (интегралданған) орта DOS ортасында да және Windows жүйесінде де жұмыс істей береді. Біз қарастырғалы отырған С/С++ тілдерінің негізгі ұғымдары мен операторлары кез келген мектептің немесе жоғары оқу орнының компьютерлерінде орнатылған ТурбоС біріктірілген ортасының редакторы арқылы немесе Visual C++ программалау ортасында теріліп орындалады.

Оқулықтағы программалардың басым бөлігі С тілінде беріледі, С++ тілінде берілген программалардың бірсыпыра өзгешеліктері бар, сондықтан олар туралы мысалдарды талдау кезінде қосымша ескертіліп айтылады. Қазіргі кең қолданылып жүрген программаларды орындау ортасы С/С++ тілдерінің компиляторы болғандықтан, жалпы кітаптағы мысалдар кез келген трансляторда (ТурбоС, BorlandC, C++) орындала береді.

5.1 С/С++ тілдерінде жазылған программаның құрылымы

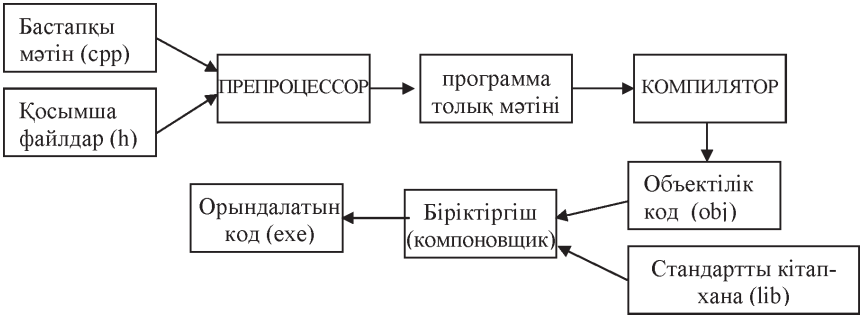
С тіліндегі программа құрылымы төмендегідей болады:

```
#препроцессор директивалары
. . . . . .
#препроцессор директивалары
функция a()
  {операторлар}
функция b()
  {операторлар}
void main ()
      // программаның орындалуын бастайтын функция
```

```
{операторлар:
      сипаттау операторлары
      меншіктеу операторлары
      функция
      бос оператор
      құрама операторлар
      таңдау операторлары
      цикл операторлары
      көшу операторлары
}
```

Препроцессор директивалары – программаны компиляциядан өткізгенге дейінгі түрлендіру ісін басқарады. C/C++ тілдерінде мәтіндік файл түрінде даярланған бастапқы программа түрлендірудің 3 кезеңінен өтеді:

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



5.1-сурет. C/C++ программасы мәтінін түрлендіру кезеңдері

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

1) **#define** – программа мәтініндегі алмастыру ережелерін көрсетеді.

#define ZERO 0.0

Соңғы жол программадағы ZERO сөзінің әрқайсысы 0.0 санына ауысатынын білдіреді.

2) **#include <тақырыптық файл аты>** – программа 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 программа мәтініне енгізуге мүмкіндік береді. Кітапханадан алынатын қажетті кодтар программаға компиляциядан кейін орындалатын біріктіру кезеңінде қосылады. Тақырыптық файлдарда стандартты функциялардың барлығының да сипаттамалары болғанмен, программа кодына тек соның ішінде қолданылатын функциялар ғана кірістіріледі.

Препроцессорлық өңдеу кезінде программа мәтініндегі препроцессор директивалары (#include, #define) анықталып, программаға тақырыптық файлдар каталогынан мәтіндік файлдар қосылады немесе кейбір сөздерді алмастыру орындалады. С/С++ тілдерінің стандарты бойынша берілген функциялар бір тақырыптық файлда анықталады.

Препроцессорлық өңдеу орындалғаннан кейін программа мәтінінде бірде бір препроцессорлық директива қалмайды.

Компиляция кезінде программа 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 программа мәтінін береді.

Барлық программа 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 программаның екілік сандар түрінде жазылған атқарылатын машиналық коды жасалады да, ол орындалып жұмыс нәтижесін береді. Барлық программалау тілдері осы схемамен жұмыс істейді.

Программа 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 программа орындалмайды. 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 – программадағы 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 константа
float x;   // нақты (аралас сан түріндегі) айнымалы
```

мұндағы // белгілерінен кейін орналасқан сөздер түсініктеме рөлін атқарады да, программаға еш әсерін тигізбейді.

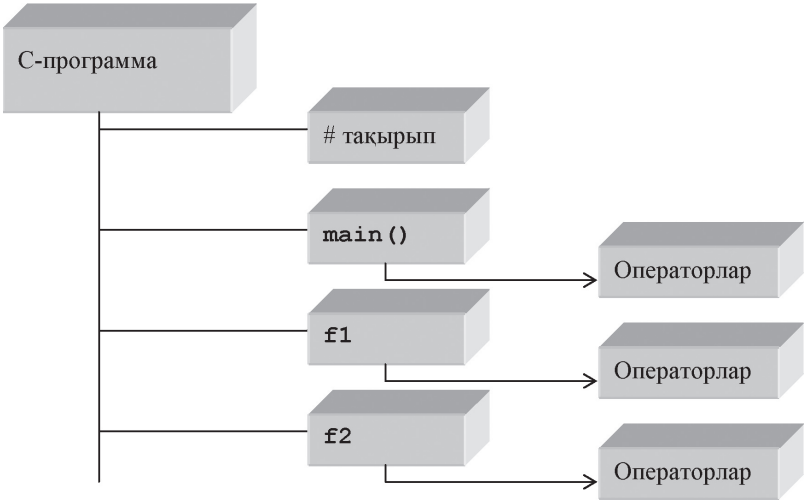
Сипаттамалар – компиляторға программаның басқа бөліктерінде жазылған объектілер мен функциялардың аттары және қасиеттері жайлы мәлімет береді.

Операторлар – программаның орындалатын әрбір қадамында қандай іс-әрекеттер атқарылатынын анықтайды.

С программасының мысалы:

```
#include <stdio.h>   // препроцессорлық директива
void main()          // функция тақырыбы
{                    // функция тұлғасының басы
 printf("Hello!");   // экранға Hello! сөзін шығару
}                    // соңы
```

5.2-суретте С/С++ программаларының жалпы құрылымы көрсетілген.



5.2-сурет. С/С++ программалары құрылымы

Суретке сәйкес, программа бірнеше функциялардан (main, f1, f2…) құралады және олардың біреуі міндетті түрде `main()` болуы қажет. Жалпы кез келген функция оның тақырыбы мен тұлғасынан (денесінен) тұрады.

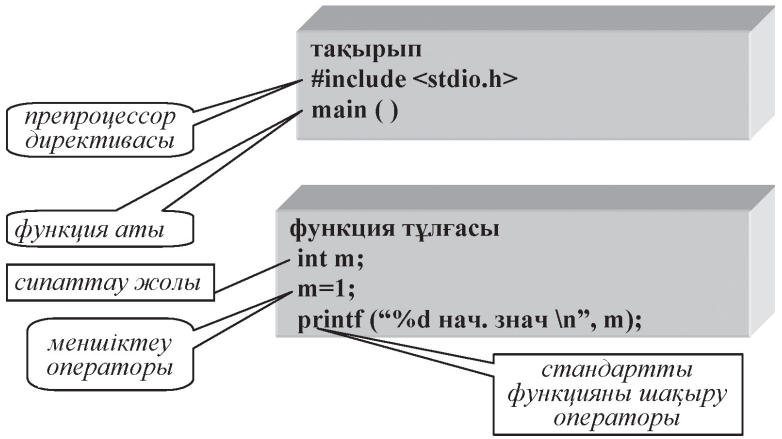
Программадағы кез келген функция тақырыбы препроцессордың директивасынан немесе функция атынан тұрады. Функция атына жалғасып, жақша ішіне параметрлер жазылуы мүмкін, кейде параметрлер болмайды, ондайда жақша ішіне ешнәрсе жазылмайды.

Функция тұлғасы операторлардан тұрады, олар жүйелі жақшалармен шектеледі. Әрбір оператордан кейін **;** таңбасы қойылады.

Енді бір программа мысалын келтірейік, олардың кейбірінің жұмыс істеу тәсілдері кейінірек баяндалады:

```
/* Герон формуласы арқылы үшбұрыш ауданын табу   */
#include <stdio.h> /* енгізу/шығару директивасы */
#include <math.h> /* математикалық функциялар
                     директивасы */
main()          /*  басты функцияны қолдану  */
{ int a,b,c; /*  бүтін айнымалыларды сипаттау  */
  float p,s;    /*  нақты айнымалыларды сипаттау  */
```

```
   printf("\nҮшбұрыш қабырғаларын енгіз: \n");
   scanf("%d%d%d",&a,&b,&c);
   p=(a+b+c)/2.0;
   s=sqrt(p*(p-a)*(p-b)*(p-c));
   printf("s=%f",s);
}
```



5.3-сурет. С программасы құрамы

Программада түсініктемелер беру үшін /* және */ таңбалары қолданылады, олардың ішіне қазақша, орысша, ағылшынша сөздер жазуға болады. Ал жол соңындағы түсініктемелер қос қиғаш сызықтан (//) кейін жазылады, бұл тәсіл C++ тілінің құрылымынан алынған.

Препроцессор директивалары **#include** сөзінен кейін жазылады, **stdio.h** тіркесі енгізу/шығару операциялары орындалатынын білдіреді (5.3-сурет). Ал **math.h** сөз тіркесі программада стандартты математикалық функциялар пайдаланылатынын көрсетеді.

Басты функция **main()** аргументсіз жазылған, сол себепті жақша ішінде ешнәрсе көрсетілмеген. Ал функция тұлғасы, яғни ішкі құрамы операторлардан (немесе басқа функциялардан) тұруы тиіс. **Int** түйінді сөзі **a,b,c** айнымалыларының бүтін мән қабылдайтынын, **float** түйінді сөзі **p, s** айнымалыларының нақты мән қабылдайтынын сипаттап тұр.

Келесі жол үшбұрыш қабырғаларын енгізуді талап ететін сөз тіркестерін экранға шығарады, мұндағы **\n** таңбалары сөз тіркесі алдында және одан кейін курсор бір жол төмен түсетінін көрсетеді. **Scanf** сөзінен басталатын жол **a,b,c** мәндерін бос орын таңбасымен бөле отырып енгізу арқылы пернелерден қабылдайды, сонан кейін жарты периметр есептеледі, 2 санын 2.0 түрінде жазу бөлу нәтижесінің нақты сан болатындығын білдіреді, әйтпесе бөлінді бүтін сан түрінде болады. Одан кейін аудан мәні анықталады да, соңғы нәтиже экранға шығарылады.

5.2 Тілдің құрамы

Кез келген табиғи тілдің мәтініндегі төрт негізгі элементті көрсетуге болады: символдар, сөздер, сөз тіркестері және сөйлемдер. Осындай элементтер алгоритмдік тілдерде де болады, мұнда бірақ сөздер – *лексемдер* (қарапайым конструкциялар) деп, сөз тіркестері – *өрнектер* деп, ал сөйлемдер – *операторлар* деп аталады. Лексемдер симолдардан тұрады, өрнектер – лексемдер мен символдардан, ал операторлар – символдардан, өрнектерден және лексемдерден тұрады (5.4-сурет):

5.4-сурет. Алгоритмдік тілдің құрамы

- Тілдің алфавиті немесе оның символдары – бұл бөлінбейтін негізгі белгілер, солардан тілдің барлық мәтіндері құрастырылады.

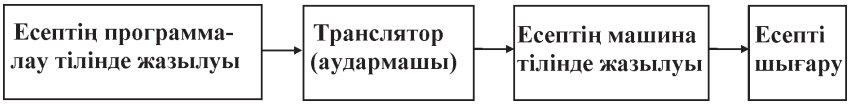
- Лексем немесе қарапайым конструкция – өзіндік мағынасы бар тілдің ең кіші бірлігі.
- Өрнектер белгілі бір мәнді есептеу ережесін береді.
- Оператор белгілі бір әрекеттің аяқталған сипаттамасынан тұрады.

Күрделі есепті шығару үшін операторлар тізбегін жазу керек. Кейде операторлар жүйелі жақшамен қоршалған құрама операторға немесе блокқа біріктіріледі. Мұндайда олар бір оператор тәрізді орындалады.

Операторлар *атқарылатын* және *атқарылмайтын* (орындалмайтын) болып бөлінеді. Атқарылатын операторлар белгілі бір әрекетті (амалды, операцияны) орындайды. Ал атқарылмайтын операторлар тек мәліметтерді сипаттайды, сондықтан оларды сипаттау операторлары немесе сипаттамалар деп те атайды.

Тілдің әрбір элементі синтаксиспен және семантикамен анықталады. *Синтаксистік* анықтамалар тіл элементтерінің құрылу ережелерін анықтайды, ал *семантика* олардың мағынасы мен қолдану ережелерінен тұрады**.**

Біртұтас алгоритм бойынша біріктірілген сипаттамалар мен операторлар жиыны программа құрайды. Оны орындау үшін оны процессорға түсінікті тіл – машиналық кодқа айналдыру керек. Бұл процесс бірнеше кезеңдерден тұрады. Ол үшін қажетті кезеңдер мен іс-әрекеттер 5.5-суретте көрсетілген.



5.5 сурет. Есепті шығару кезеңдері

Транслятор құрамындағы байланыс редакторы бір модульге басқа модульдерді қоса отырып, оның ішінде кез келген программада қолданылатын кітапханалық функциялар (мысалы, экранға мәлімет шығару үшін) бар, программаның машиналық кодтағы *атқарылатын модулін* қалыптастырады. Егер программа бірнеше бастапқы файлдардан тұратын болса, олар жекежеке компиляциядан өткізіліп (аударылып), құрастыру кезеңінде біріктіріледі. Атқарылатын модульдің тіркемесі *.exe* болып шығады да, ол кәдімгі программалардай бірден орындалады.

Тіл ережелерін сипаттау үшін оқулықтарда жасанды метатіл қолданылады, мысалы, Бэкус-Наур тәсілі немесе синтаксистік диаграммалар. Көрнекті және қарапайым ету мақсатында бұл оқулықта кең таралған сипаттау тәсілі қолданылады, онда тіл құрылымының міндетті түрде талап етілмейтін бөліктері тік жақшаға алынады; нақты мәнмен алмастырылатын мәтін қазақша (орысша) жазылады; ал бірнеше элементтердің бірін таңдау кезінде олар тік сызықшамен бөлініп жазылады.

Мысалы, мынадай

```
[ void | int ] аты();
```

тіркесте, **аты** сөзі орнына тіл ережелеріне сәйкес нақты атауды көрсету қажет, ал оның алдында **void** немесе **int** сөзі орналасады немесе ештеңе де жазылмайды. *Жүйелік жақшалар* тек бір элементі ғана таңдалып алынатын бірнеше элементтерді біріктіріп тұрады. Егер тік жақша синтаксис элементі болып келген жағдайда, ол туралы айрықша айтылады.

C++ тілі C тілінің жалғасы, олардың алфавиттері де, негізгі операторлары мен ұғымдары бірдей, енді C тілін және C++ тілінің C тіліне кірмейтін бірсыпыра ерекшеліктері мен күрделі элементтерін игере отырып, бірте-бірте объектіге бағытталған программалау үрдістерін үйрене бастаймыз.

Тілдің алфавиті. C/C++ тілдерінің алфавиттері бірдей деуге болады. Мұндағы түйінді сөздер (ключевое слово – keyword) мен идентификаторларды, өрнектерді құрастыру үшін қолданылатын символдар, яғни тіл алфавиті болып саналады.

C/C++ тілдері алфавитіне мыналар кіреді:

- ағылшын алфавитінің бас және кіші әріптері мен астын сызу символы кіреді;
- 0-ден 9-ға дейінгі араб цифрлары;
- арнайы таңбалар:

```
" { } , | [ ] ( ) + - / % * . \ ' : ? < = > !
& # ~ ; ^
```

- тіл элементтерін бір-бірінен бөліп тұратын көрінбейтін айыру символдары: босорын, табуляция символы, жаңа жолға көшу символы.

- қазақ (орыс) алфавитінің бас және кіші әріптері сөз тіркестері мен түсініктеме мәтін жазуда ғана пайдаланылады. Көбінесе олар-

ды сәйкес латын әріптерімен көрсету немесе ағылшын тіліндегі аудармаларын пайдалану қалыптасқан, өйткені көптеген трансляторлар қазақ әріптерін бейнелемейді.

Көрсетілгендерден басқа C/C++ тілдерінде **басқару тізбектері** деп аталатын мәліметтер енгізу мен шығаруда қолданылатын арнайы символдар тіркесі бар. Басқару тізбектері кері бөлу сызықшасы белгісінен (\) басталатын латын әріптері мен цифрлар тізбегінен тұрады (5.1 кесте).

5.1 кесте. **Басқару тізбектері**

Басқару тізбектері	Атаулары	Он алтылық кодтары
\a	Қоңырау	007
\b	Бір орынға кері қайтару	008
\t	Горизонталь табуляция	009
\n	Жаңа жолға көшу	00A
\v	Вертикаль табуляция	00B
\r	Сырғыманы (каретканы) қайтару	00C
\f	Форматты жылжыту	00D
\"	Қостырнақша	022
\'	Апостроф	027
\0	Нөл-символ	000
\\	Кері қиғаш сызық	05C
\odd	Сегіздік жүйедегі кодтар жиыны	
\xddd	Он алтылық жүйедегі кодтар жиыны	

Кестеде көрсетілген \ddd и \xddd (мұндағы d цифр тұрғанын көрсетеді) түріндегі тізбектер компьютердегі кодтар жиыны арқылы өрнектелетін символды сәйкесінше сегіздік және он алтылық цифрлар тізбегімен бейнелей алады. Мысалы, каретканы қайтару символы бірнеше тәсілмен өрнектеле алады:

\r – жалпы басқару тізбегі,

\015 – сегіздік сандардан тұратын басқару тізбегі,

\x0D – он алтылық сандардан тұратын басқару тізбегі.

Мысалы, жеке мынадай \n (жаңа жолға көшу) басқару тізбегін \010 немесе \xA түрінде де жазуға болады.

Басқару тізбектері тіркестер түріндегі константаларда да пайдаланылады, оларды *тіркестік литералдар* деп те атайды. Егер сөз тіркесі ішінде қостырнақша тұруы керек болса, онда оның

алдына қиғаш сызық қойылады, осыған байланысты компилятор оны шекарада тұратын қостырнақшадан ажырата алады, мысалы:

```
"\"Білім\" баспасы"
```

Тілдің қарапайым объектілеріне сан, идентификатор, константа, айнымалы және функция, өрнек ұғымдары кіреді.

Программадағы негізгі амалдардың орындалуына керекті мәліметтердің сандық, логикалық немесе символдық (литерлік) мәндері болады. Олармен жұмыс істеу қолайлы болуы үшін алгебра курсындағы белгілеулерге ұқсас шартты атаулар пайдаланылады. Бұл атаулар әр түрлі мәндерді (сандық мән, символдық мән, т.с.с.) қабылдауы мүмкін, сондықтан оның типі деген ұғым енгізіледі.

Сандар. Сандар мен айнымалылар бүтін және нақты болып екіге бөлінеді. *Бүтін сандар*: +4, -100, 15743, 0, т.с.с. Қазіргі дербес компьютерлер үшін қолданылатын бүтін сандар (ағылшынша INTEGER) -32768 бен +32767 аралығында ғана жазылады, бұдан үлкен сандар көбінесе нақты сандарға айналдырылады.

Нақты сандар кәдімгі табиғи аралас сандар тәрізді санның бүтіні мен бөлшегін нүкте арқылы бөлген күйде жазылады. Мысалы: 2.65, 0.5, -0.862, -6.0. Ал өте үлкен немесе өте кіші нақты сандар көрсеткіші бар экспоненциал сандар ретінде $mE\pm p$ түрінде жазылады да, олардың диапазоны әлде қайда кең болады, мұндағы m – санның мантиссасы деп аталады, E немесе e – онның дәрежесі дегенді білдіреді, ал p – дәреженің өз мәні. Мысалы:

Кәдімгі жазылуы	*С тілінде жазылуы*
145	145
147,125	147.125
-6,045	-6.045
$12*10^{14}$	12E+14
$-0,52*10^{4}$	-0.52e4
$5,2*10^{-12}$	5.2E-12
$-45*10^{6}$	- 45e6

Символдардан тіл лексемдері құралады, ал лексемдер константалар, айнымалылар, идентификаторлар, т.б. тұрады

Тұрақты немесе **константа** деп программаның орындалу барысында мәндері өзгеріссіз қалатын шамаларды айтады,

олар сандық, символдық немесе тіркестік мәнді тұрақты түрде қабылдайтын лексем болып табылады.

Тіл ережесі бойынша бірнеше константа типтері болады, мысалы, символдық, бүтін, нақты және тіркестік константалар. Компилятор константаны лексем (қарапайым конструкция) ретінде қарастырып, оны сыртқы түріне қарай бір типке жатқызады.

Әрбір типке сәйкес константалар форматтары 5.2 кестеде көрсетілген.

5.2 кесте. **C/C++ тілінің константалары**

Конс-танта	Форматы	Мысалдар
Бүтін	Ондық: ондық цифрлар тізбесі, нөлден басталмайды (егер сан 0 болмаса)	8, 0, 19226
	Сегіздік: нөлден басталатын сегіздік цифрлар (0,1,2,3,4,5,6,7)	01, 020, 07155
	Он алтылық: 0x немесе 0X таңбаларынан басталатын он алтылық цифрлар (0,l,2,3,4, 5,6,7,8,9,A,B,C,D,E,F)	0xA, 0x1B8, 0X00FF
Нақты	Ондық: [цифрлар].[цифрлар]	5.7, .001, 35.
	Экспоненциалдық [цифрлар][.][цифрлар]{E|e}[+|-][цифрлар]	0.2E6, .11e-3, 5E10
Сим-волдық	Апостроф таңбасымен қоршалған бір немесе екі символ	'A', 'ю', '*', 'db', '\0', '\n\', \012', '\x07\x07'
Тіркес-тік	Тырнақшаға алынған символдар тіркесі	"Мұнда Азат болды", "\tНәтижесіг=\0xF5\n"

Бүтін және нақты константалардың қабылдай алатын мәндер диапазондары келесі тарауда келтірілген.

Егер теріс таңбалы бүтін немесе нақты константа алғымыз келсе, оның алдына сан таңбасын керіге ауыстыратын унарлық, яғни бір орындық операция белгісі (-) қойылады, мысалы: -218, -022, -0x4C, -4.8, -0.1e4.

Константа және сан жазылуында бос орын таңбасы болмайды және де санның бүтіні мен бөлшегін айыру үшін үтір емес нүкте қойылады.

Бір ғана символдан тұратын символдық константалар стандартты **char** типінде болып, компьютер жадында бір байт орын алады. Екі таңбадан тұратын символдық константалар екі байт орын алып, **int** типінде болады да, алғашқы символ кіші адресті байтта сақталады.

Айнымалылар деп программаның орындалу барысында әр түрлі мәндерді қабылдай алатын шамаларды айтады. Әрбір айнымалы мен константа программа алдында сипатталуы тиіс. Олардың компьютер жадында алатын орны типтеріне байланысты болады. Константалар мен айнымалылар идентификатормен белгіленеді.

Айнымалы қасиеттері:

1. айнымалы белгілі бір мәнге ие болмағанша, анықталмаған болып саналады. Оған мән беру мынадай тәсілдермен орындалады:
   - сырттан енгізу арқылы;
   - константантаны меншіктеу арқылы;
   - бұрын анықталған айнымалының мәнін беру арқылы;
2. кез келген сәтте айнымалының белгілі бір мәні болады немесе ол анықталмаған болып есептеледі;
3. айнымалыға соңғы берілген мән оның алдыңғы мәнін жойып (өшіріп) жібереді. Айнымалыны таңдау (оқу) және оны пайдалану айнымалының мәнін өзгертпейді.

Әрбір айнымалы мен константа программа алдында сипатталуы тиіс. Олардың компьютер жадында алатын орны типтеріне байланысты болады. Константалар мен айнымалылар идентификатормен белгіленеді. C/C++ тілдеріндегі *айнымалы* – белгілі бір типтегі мәліметтер сақталатын компьютер жадының ат қойылған аймағы. Айнымалының аты мен мәні болады. Аты компьютер жадында мәні сақталған мәліметті пайдалану үшін керек. Пайдалану алдында кез келген айнымалы сипатталуы тиіс. Мысалы:

```
int a; float x;
```

Атау – идентификатор (identification – объектінің белгілі бір символдар тіркесіне сәйкестігін бекіту) программаны және программадағы тұрақтыларды, типтерді, айнымалыларды, функцияларды, файлдарды және тағы басқаларды белгілеп жазу

үшін қажет. Идентификаторлар тұрақтыларды, айнымалыларды, олардың түрлерін, функцияларды, программаларды, файлдарды, т.б. программа объектілерін белгілеу үшін қолданылады. Оның ұзындығын өте үлкен етудің қажеті жоқ, өйткені атауларды теру және кейіннен сақтау біраз уақыт керек етеді.

Идентификатор – латын әрпінен басталып, әріптер мен цифрлардан тұратын тізбек. Мысалы, a, beta, b5, baga, т.с.с. Айнымалыны сипаттау мынадай нұсқада орындалады:

```
char f;
long z, t;
int  a, beta, baga;
float b5, k, n;
int  y = 10;
```

Идентификаторлар латын алфавитінің бас және кіші әріптерінен және цифрлардан құралады. Әріп ретінде астын сызу символын (_) қолдануға рұқсат етілген. Бас әріп пен кіші әріп бірдей болып саналмайды, олар әр түрлі идентификаторлар болып есептеледі, мысалы, abc, ABC, A128B, a128b төрт түрлі идентификатор болып есептеледі.

Идентификатор ұзындығына шек қойылмайды, бірақ оның алғашқы 31 символы ғана мағыналы болып саналады. Идентификатор оны сипаттау кезінде анықталады да, кейінгі операторларда қолданыла береді. Сипатталатын идентификатор С тілінің алдын ала анықталған түйінді сөздерімен сәйкес келмеуі тиіс.

Константалардан, айнымалылардан, функциялардан және операциялар таңбаларынан ***өрнектер*** құралады. Әрбір *өрнек арифметикалық операциялар таңбаларымен қажетті жақшалар арқылы біріктірілген бірнеше операндтардан (сан, айнымалы, константа) тұрады*. Математикадағы формулалар, алгебрадағы көпмүшеліктер программалау тілінде тек осы өрнек ұғымы арқылы беріледі.

Егер өрнек мәні бүтін немесе нақты сан болатын болса, ол арифметикалық өрнек болып саналады. Арифметикалық өрнектерде мынадай операциялар: + – * / % (қалдық табу) болады. Жалпы өрнектер бір жол бойына жазылады және олардағы операция реттілігі жақшалармен анықталады.

Өрнектерді жазу мысалдары:

```
i = i+1; k = 5.35; x1=(-b+sqrt(b*b-4*a*c))/(2*a);
y = sqrt(sin(x)+1); c = 2*pi*r;  R = 19.36;
```

Қатынас таңбасы арқылы біріктірілген екі арифметикалық өрнек мәні басқа тілдердегідей ақиқат (0-ге тең емес) немесе жалған (0-ге тең) деп айтылады. Бірақ С тілінде логикалық тип түсінігі айтылмайды, ол С++ тілінде бар.

Түйінді сөздер (keyword) – мағынасы алдын ала анықталған идентификаторлар, олардың саны шектеулі. Программалаушы айнымалы, константа, өз функциялары аттарында тілдің түйінді сөздерін пайдаланбауы тиіс, олар тек өз мағынасында ғана қолданылады.

С тіліндегі бірсыпыра түйінді сөздер тізімін келтірейік.

```
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
```

С++ тілі бұларға тағы аздаған сөздер қосады:

```
asm catch class friend inline new operator private
protected public template this throw try virtual wchar_t
```

Бұған қоса операторлар мен стандартты функциялар аттары да түйінді сөздер тізімі секілді басқа мағынада қолданылмайды.

Стандартты функциялар. С/С++ тілдерінде алдын ала программалары жасалып стандартты модульге жинақталып қойылған, қажет кезінде пайдалануға болатын объектілердің бірі стандартты функциялар болып табылады. Олар жиі кездесетін математикалық және басқа да функцияларды есептеу үшін қолданылады. Стандартты функцияны жазу үшін міндетті түрде функцияның аты және жақшаның ішінде аргументі көрсетілуі қажет. Стандартты функциялар: fabs(x), sin(x), cos(x), asin(x), acos(x), tan(x), exp(x), log(x), sqrt(x), atan(x), т.с.с. Кітап соңындағы *А қосымшасында* бірсыпыра функция тізімдері мен олардың жазылу жолдары көрсетілген. Функцияны есептеу барысында аргумент пен функция типтерінің әр уақытта сәйкес келе бермейтінін есте сақтаған жөн. С/С++ тіліндегі стандартты функцияларды пайдалану үшін <**math.h**> тақырыптық файлы (прототипі) қолданылады.

Комментарий – түсініктеме ретінде қолдануға болатын символдар тізбегі. С тіліндегі комментарий басы мен аяғы мынадай /* ...*/ таңбалармен шектелуі тиіс. Олар бір немесе бірнеше жолдардан да тұра алады. Си++ тілінде жол соңында тұратын комментарийлер // сиволдарынан кейін орналасады. С және Си++ тілдерінде құрастырылған есептерді бір компилятор арқылы шығаруға болатындықтан, көбінесе түсініктемелердің жоғарыдағы екі түрін де пайдалана беруге болады. Комментарий ішінде тек С++ тілі алфавитіндегі символдарды ғана емес, компьютерде қолдануға болатын барлық символдарды пайдалана беруге болады. Мысалы, олар ұлттық алфавиттерді де (егер ол экранда бейнеленетін болса) пайдалана алады.

Түсініктеме ретінде мүмкіндігінше // сиволдарынан кейін орналасатын комментарийлерді қолдану ұсынылады, ал қос таңбалық жақша ішіндегі /* ... */ символдар тізбегін программаны түзетіп жөндеу кезінде кодтар бөлігін уақытша орындамайтын кездерде пайдаланған дұрыс болып табылады.

Программалау тілінің белгілі бір іс-әрекетті орындай алатын тиянақты мағынасы бар ең қарапайым сөйлемі ***оператор*** болып табылады. Тіл объектілерін, яғни программада пайдаланылатын мәліметтердің атаулары мен типтерін, олардың алғашқы мәндерін алдын ала тағайындау программаның *сипатталуы* болып саналады.

Енді Фаренгейт градустарын Цельсий градустарына ауыстыратын С тілінде жазылған программа мәтінін келтірейік.

```
/* Градустарды Фаренгейт бірлігінен Цельсий
 бірлігіне алмастыру, f = 0, 20, ...,300 */
#include <stdio.h>
#include <conio.h>
main()
{
int t0, tn, step;
float f,c;
t0 = 0;   /* төменгі температура */
tn =300;  /* жоғарғы температура */
step = 20; /* өзгеру қадамы */
clrscr(); f = t0;
```

```
printf("Град_Ф Град_Ц\n");
while (f <= tn)
{
 c = (5.0/9.0) * (f -32.0);
 printf("%4.0f %6.1f\n", f, c);
 f = f + step;
}
printf("\nАяқтау үшін ENTER басыңыз");
getch();
}
```

Бұл программаны теріп орындап көруге болады, оның әрбір жолының атқаратын қызметтерін келесі тараулардан оқып танысатын боламыз.

Бақылау сұрақтары

1. *С/С++ тілдерінің қысқаша даму тарихы.*
2. *С/С++ программасының жалпы құрылымы және құрамы.*
3. *Тақырыптық файлдар, препроцессор ұғымы.*
4. *Сипаттамалар қандай қызмет атқарады?*
5. *Операторлар түсінігі.*
6. *Компьютерде программаларды орындау кезеңдері.*
7. *С/С++ тілі алфавиті, басқару тізбектер.*
8. *Тілдің қарапайым объектілері.*
4. *С/С++ тіліндегі стандартты функциялар.*
5. *Тілдің алфавиті, оның құрамы.*
6. *С/С++ тілінің негізгі объектілері.*
7. *Тұрақтылар түрлері және олардың мүмкін мәндері.*
8. *Айнымалылар ұғымы және олардың қасиеттері.*
9. *Идентификатор және түйінді сөздер ұғымдары.*
10. *С және С++ тілдеріндегі түсініктемелерді беру тәсілдері.*

Тапсырмалар

1. Төмендегі сандардың ішіндегі бүтін сандарды көрсетіңіздер: 775; -832; 45.0; 2.98; -0.0; 2.65e02; 15248; -0.85e-01.
2. Төмендегі сандардың ішіндегі нақты сандарды көрсетіңіздер: 77.5; -844.0; 43.0; -2.98; -0.0; 2.65e02; 15248; -0.85e-01.
3. Төмендегі сандардың ішіндегі сегіздік сандарды көрсетіңіздер: 775; -081; 0x43; 0541; -065; 0x76A; 244; -0654.
4. Төмендегі сандардың ішіндегі оналтылық сандарды көрсетіңіздер: 7AC1; 0X8B; 0x4C; -0x2F1; -0141; -0xD6A; 0244; -0X654.

5. Мынадай арифметикалық өрнектерді программалау тілі заңдылықтарымен жазып шығыңыздар:

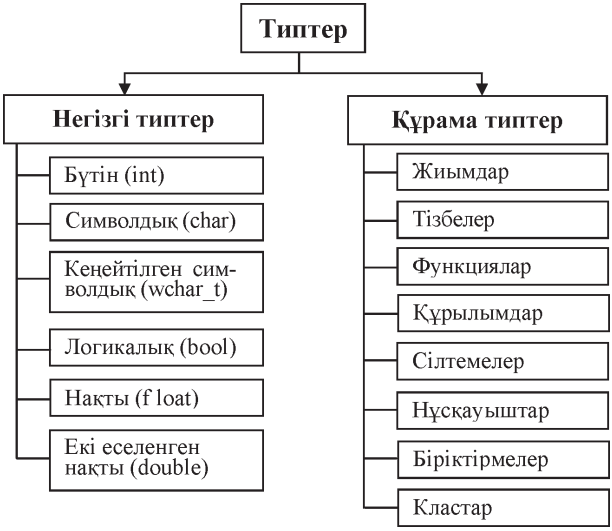
а) $t = \frac{x}{a} - \frac{1}{\pi}\lg(a+b);$ ә) $\varpi = e^{\left(\frac{a+b}{c-d}\right)} + 10^{-4}t;$

б) $v = \sqrt{\frac{8RT}{\pi\mu}},$ в) $E = \frac{b}{2c}\left(1 - \frac{a}{\sqrt{R^2 + a^2}}\right)$

6. Келесі айнымалыларды сипаттауда қандай қателер бар:

```
int  f; c; в;
long  2z, t5;
char  a2, 2beta, b_aga;
float b5, k=2, n;
int  y = 10.0;
```

Мәліметтер типі концепциясы. Кез келген программаның 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.



6.1 сурет. С++ тілінің типтері құрамы

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 программада пайдаланылатын нақты объектілерді бейнелеу үшін программалаушы әрбір шамалардың типін алдын ала таңдап алады. Типті міндетті түрде сипаттау қажеттілігі компиляторға программадағы әр түрлі конструкция-

ларды 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 (6.1 сурет). Программалаушы осы типтерді негізге ала отырып, құрама типтерді сипаттай алады. Құрама типтерге жиымдар (массивтер), тізбелер (перечисления), функциялар, құрылымдар (структуралар), сілтемелер (ссылки), нұсқауыштар (указатели), біріктірмелер (объединения) және кластар жатады.

6.1 С/С++ тілдеріндегі мәліметтер типтері

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 6.1 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).

6.1.1 Бүтін сан түріндегі мәліметтерді сипаттау

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 – константалар 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 6.1 кестеде келтірліген.

Алдын ала келісім бойынша барлық бүтін санды типтер таңбалы болып саналады, яғни **signed** спецификаторын жазбаса да болады.

Программада кездесетін константалардың жазылуына қарай, яғни солардың сыртқы бейнесіне сәйкес белгілі бір тип тағайындалады. Егер ол тип, кейбір жағдайларға байланысты, программалаушыны 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, **L** және **U** әріптерін қатарластыра да қолдануға болады, мысалы, **0x22UL** немесе **05Lu**. Осы атаулардағы **short int, long int, signed int** және **unsigned int** типтерін **short, 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).

6.1 кесте. **C/C++ тілдерінің ішкі құрамындағы мәліметтер типтері мен олардың ені (ұзындығы)**

Мәлімет типі	*Ұзындығы (бит – байт)*	*Сандар диапазоны*
bool	`true` және `false` – 1 бит	0 ... 1
char	8 бит – 1 байт	-128 ... +127
unsigned char	8 бит – 1 байт	0 ... 255
short int	16 бит – 2 байт	-32768 ... 32767
unsigned short	16 бит – 2 байт	0 ... 65 535
int	16 бит – 4 байт	-32768 ... 32767
unsigned [int]	32 бит – 4 байт	0 ... 4294967295
long	32 бит – 4 байт	-2 147 483 648 ... 2 147 483 647
unsigned long	32 бит – 4 байт	0 ... 4 294 967 295
float	32 бит – 4 байт	3.4×10^{-38} ... 3.4×10^{38}
double	64 бит – 8 байт	1.7×10^{-308} ... 1.7×10^{308}
long double	80 бит – 10 байт	3.4×10^{-4932} ... 3.4×10^{4932}

Нақты сандар типтері үшін кестеде солардың абсолюттік шамаларының ең кіші (минимал) және ең үлкен (максимал) мәндері көрсетілген.

Char типі

Char типін 0–255 аралығындағы таңбасыз бүтін сандарды сипаттауда қолдануға болады, компьютер жадында бұларға бір байт орын бөлінген. Мысалы:

```
char c1;
char ck = 'k';
```

Бұл тип мәндері реттелген символдар жиыны болып табылады. Әрбір символға бір бүтін сан сәйкес келеді, ол символ коды деп аталады. Символдық тип ені – 1 байт. **Char** типі де **signed** және **unsigned** спецификаторларымен қолданылады. **Signed char** типі диапазоны –128 .. +127. **Unsigned char** типін қолданғанда, оның мәндер диапазоны 0 .. 255 болады.

Символдарды кодтау үшін ASCII (American Standard Code for International Interchange) стандарты негізге алынған. **Unsigned char** типі 256-символдық ASCII кодтар жиынының кез келген символын бейнелеуге толық жарайды. **Char** типі осы сандар диапазонынан аспайтын бүтін сандарды сақтау үшін де пайдаланылады. **Char** символдарының 0 .. 31 кодтары басқару кодтарына жатады, олар тек енгізу-шығару кезінде ғана қолданылады.

Char типі символдарды олардың бүтін сан түріндегі кодтары арқылы сақтап, басқа шамалардың көрсетілген диапазоны сандарын да көрсету үшін де қолданыла алады.

Кеңейтілген символдық тип (wchar_t)

wchar_t типі (тек C++ тілінде) бір байттан асатын символдар жиынын кодтау үшін қолданылады, мысалы, **unicode** символдары. Бұл типтің көлемі компьютерге байланысты болып келеді де, көбінесе **short** типіне сәйкес келеді. **wchar_t** типіндегі тіркестік константалар **L** әрпінен басталып жазылады, мысалы, `L"bit"`.

Логикалық тип (bool)

Логикалық тип (бұл да тек C++ тілінде) түйінді сөз болып табылатын **true** және **false** мәндерін қабылдайды. Компьютер жадында **false** – 0 (нөл), басқа кез келген мән **true** болып табылады. Бүтін типке түрлендіргенде, **true** 1-ге сәйкес келеді.

6.1.2 Жылжымалы нүктелі нақты сандар типтері

Нақты сандар компьютерде 2 бөліктен – дәреже мен мантиссадан тұрады. IBM-PC компьютерлерінде float типінің ені – 4

байт, оның бір разряды – сан таңбасы, 7 разряды – дәреже, 24 бит3 – мантисса.

Егер `double` типі аты алдында `long` сөзі тұрса, онда оған 10 байт орын беріледі.

Программалау практикасында көбінесе жылжымалы нүктелі нақты (аралас) сандар пайдаланылады.

Жылжымалы нүктелі типтер компьютер жадында бүтін сандардан басқаша түрде сақталады. Нақты санның ішкі бейнесі екі бөліктен – *мантисса* және *дәрежеден* тұрады. IBM PC тәрізді компьютерлерде **`float`** типті шамалар 4 байттан тұрады, оның ішінде бір екілік разряд мантисса таңбасын бейнелейді де, 8 разряд дәрежені және 23 разряд мантиссаны көрсетеді. Мантисса – бұл 1.0-ден артық бірақ 2.0-ден аз сан. Мантиссаның алғашқы разряды әрқашанда 1 болғандықтан, ол сақталмайды.

8 байттан тұратын **`double`** типті шамалар үшін, дәреже мен мантиссаға сәйкесінше 11 және 52 разряд бөлінеді. Мантисса ұзындығы санның дәлдігін анықтайды, ал дәреже ені – оның диапазонын анықтайды. 1.4 кестеден көрініп тұрғандай, **`float`** және **`long int`** типтерінің ұзындықтарының байт саны бірдей болғанымен, ішкі бейнелену формасының айырмашылығына қарай, олардың диапазоны әр түрлі болып келеді.

Жылжымалы нүктелі константалар, келісім бойынша, **double** типінде болады. Олардың типін **F**, **f** (float) және **L**, **l** (long) әріптері арқылы нақты түрде көрсету мүмкіндігі бар. Мысалы, 2E+6L константасы **`long double`** типінде, ал 1.82f константасы **`float`** типінде көрсетілген.

Әртүрлі платформаларға ауысатын программалар үшін `int` типінің ені жайлы алдын ала болжам жасауға болмайды. Оны анықтау үшін **`sizeof`** операциясын пайдалану керек, оның нәтижесі сол типтің байтпен берілген еніне тең болады. Мысалы, MS-DOS операциялық жүйесі ортасында **`sizeof(int)`** нәтижесі – 2 болса, ал Windows 9X немесе OS/2 жүйесіндегі нәтижесі – 4 болады.

ANSI стандартында негізгі типтер мәндерінің диапазоны берілмейді, тек солардың көлемдерінің ара қатынасы ғана анықталады:

```
sizeof(float) ≤ sizeof(double) ≤ sizeof(long double)
sizeof(char) ≤ sizeof(short) ≤ sizeof(int) ≤
sizeof(long)
```

Sizeof стандартты операторын қолдану мысалы:

```
printf("double типіндегі мәліметтер ені %d байт\n",
sizeof((double));
```

Мәліметтерді бейнелеу диапазоны мен дәлдігі әр түрлі болып келетін бүтін және нақты типтердің осындай түрлері программалаушыға кез келген аппаратураның мүмкіндіктерін тиімді түрде пайдалануға мүмкіндік береді, өйткені есептеу жылдамдығы мен компьютер жадының пайдаланылатын көлемі типке тәуелді болып келеді. Бірақ компьютердің бір түріне икемделген программа басқа платформада қолдануға ыңғайсыз болуы ықтимал, сондықтан жалпы программаларды мәлімет типтеріне тәуелді етіп құруға тырыспау керек.

void типі

Негізгі типке жоғарыда көрсетілгендерден басқа **void** типі де жатады, бірақ бұл типтің мәндер жиыны бос болып келеді. Олар мән қайтармайтын функцияларды анықтау үшін қолданылады, функцияның аргументтерінің жоқ екенін білдіреді, нұсқауыштардың негізгі типі ретінде және типтерді келтіру операцияларында да пайдаланылады.

6.1.3 Символдық тіркестер (жолдар, қатарлар)

С тілінде символдық тіркестерді сипаттау үшін арнайы тип жоқ, олар көбінесе **char** типтегі элементтерден тұратын жиым (массив) ретінде қарастырылады. Өте ұзын тіркестік константаларды бірнеше жолға бөліп жазуға болады, мұндайда тасымалдау таңбасы ретінде кері қиғаш сызық қолданылады. Мысалы, мынадай сөз тіркесі:

```
"Көзің қайда көшеден мені іздеген,\
сөзің қайда екеуміз егіз деген,\
терезеңнің алдына келіп тұрмын, \
кептердей қысты күні жем іздеген"
```

төмендегі жолдармен бірдей болып саналады:

"Көзің қайда көшеден мені іздеген,сөзің қайда екеуміз егіз деген,терезеңнің алдына келіп тұрмын, кептердей қысты күні жем іздеген"

Жолдық немесе тіркестік символдар компьютер жадында көршілес ұяшықтарда сақталады да, олардың соңында **'\0'** символы тұрады.

Әрбір тіркестік литерал соңына компилятор **\0** тізбегімен көрсетілетін *нөлдік символ* қосып жазады, сондықтан тіркес ұзындығы сөздегі символдар санынан бірге артық болады. Символдар тіркесінің ұзындығын анықтау үшін **strlen()** функциясы қолданылады.Сонымен, бос тіркестің өзі компьютер жадынан 1 байт орын алады екен. Бір символдан тұратын сөз тіркесінен, мысалы, "А", бір таңбадан тұратын символдық константаның 'А' айырмашылығын білу керек. Бос символдық константа болмайды.

#define – символдарды немесе солардан тұратын константаларды анықтау мақсатында қолданылады. Мысалы:

```
#define NULL '/0'
#define VNAME "КазНУ"
```

6.2 Printf және scanf функциялары

С тілінде сыртқы ортамен мәліметтер алмасу **<stdio.h>** енгізу-шығару функциялары кітапханасын пайдалану арқылы орындалады. Ол тақырып файлы ретінде былай жазылады:

```
#include <stdio.h>
```

printf() функциясы мәліметтерді экранға шығару үшін қолданылады. Оның жалпы жазылу түрі:

printf(<формат тіркесі>,<аргументтер тізімі>);

(<формат тіркесі> – формат спецификаторларынан, бос орыннан және әр түрлі символдардан тұрады. Ол қостырнақшамен (″) шектеліп, аргументтердің қалай бейнеленетінін көрсетіп тұрады, экранға (баспаға) шығару алдында барлық аргументтер формат спецификациясына сәйкес түрлендіріледі, спецификация % символымен басталады және мәліметтер типін, оларды түрлендіру тәсілін көрсететін бір әріп – спецификация коды жазылады. Объектілер ретінде айнымалылар, константалар, өрнектер қолданылуы мүмкін. Мысалы:

```
printf (" Пи санының мәні = %f\n", pi);
```

Бұл жолдың нәтижесі : **Пи санының мәні = 3.14159**

Формат соңында тұрған **\n** тіркесі сан шығарылған соң, курсордың келесі жолға көшетінін білдіреді.

Формат тіркесінде мыналар болады:

1) мәтін ретінде шығарылатын символдар тіркесі;
2) түрлендіру спецификациялары;
3) басқару символдары.

Әрбір аргументке өз спецификациясы сәйкес келуі тиіс, олар:

%d – бүтін ондық сан шығарылуы тиіс,

%i – бүтін ондық сан шығарылуы тиіс,

%f – жылжымалы нүктелі нақты ондық сан ([-]dddd.dddd) жазылып шығады,

%e – жылжымалы нүктелі экспоненциалды сан ([-]d.dddde±dd) шығарылады,

%E – жоғарыдағы сияқты, тек е орнына E ([-]d.ddddE±dd) шығарылады,

%c – бір символ шығарылуы тиіс,

%s – символдар тіркесі (қатары) шығарылуы тиіс,

%g – нақты сан, сан ұзындығына қарай %e немесе %f қолданыла алады,

%u – таңбасыз ондық бүтін сан жазылып шығады,

%o – таңбасыз бүтін сегіздік сан шығады,

%x – таңбасыз бүтін он алтылық сан шығады.

\n – келесі жаңа жолға көшуді атқаратын басқару символы.

Мысалы:

```
printf("%d%f", x, y);
```

экранға бір бүтін (`%d`) – x және бір нақты сан (`%f`) – y шығару функциясы. Форматтарда санның ені де көрсетілуі мүмкін. Ол былай жазылады:

%9i – бүтін сан ені 9 цифрдан тұрады, сан ені аз болса, оның сол жағында бос орындар орналасады.

%9.3f – нақты сан ені 9 цифрдан тұрады, оның 3 таңбасы бөлшекке беріледі, сан ені аз болса, оның сол жағында бос орындар орналасады.

Әрбір спецификация % символынан басталып, түрлендіру символымен аяқталады. Ол екеуінің ортасында мыналар тұруы мүмкін:

- минус таңбасы, аргумент мәні сол жақ шетке ығыстырылып жазылады.

- цифрлар, бүтін санның жалпы орналасу енін анықтайды. Сан осы енге немесе одан артық болып шығарылады. Егер аргумент ені көрсетілген еннен аз болса, онда ол бос орындармен толтырылып жазылады.

- нүктеге дейін санның жалпы ені, нүктеден соң бөлшек цифрлар ені көрсетіледі.

- L модификаторы, сәйкес аргумент мәні **int** емес **long** екенін білдіреді.

Scanf() пернелерден (консольдан) мәлімет енгізу функциясы жоғарыда қарастырылған түрлендіру спецификациясының көбін пайдаланады. Жазылу ережесі:

scanf (<формат тіркесі>,<аргументтер тізімі>);

Аргументтер ретінде адрес нұсқауыштары – айнымалылар адрестері пайдаланылады. Мысалы:

scanf("%d%f", &x,&y);

мұндағы **&x,&y** – x және y айнымалыларының компьютер жадындағы адрестері. Бұл функция пернелерден бос орын немесе Enter пернесін басу арқылы бір бүтін және бір нақты сан енгізуді талап етеді. Әдетте **scanf** функциясы алдына қандай мән енгізілетіні жайлы мәтін шығарылады. Мысалы:

printf("x, y енгізіңіз:");
scanf("%f%f",&x,&y);

x және y мәндері 3,5 пен -2,5 болуы тиіс болса, экрандағы көрініс мынадай болады:

x, y енгізіңіз: 3.5 -2.5↵

мұндағы ↵ – Enter пернесін басу белгісі.

Scanf() функциясы форматтарының **printf()** функциясы форматынан кейбір айырмашылықтарын атап өтейік:

1) **%e** және **%f** спецификациялары енгізу кезінде бірдей болып табылады;

2) short типті бүтін санды енгізу кезінде **%h** спецификациясы қолданылады.

ЕСКЕРТУ. Айнымалы адресін беру үшін адрестерді жазғанда, айнымалы адресін анықтау үшін & символы қолданылады. Ал тіркестік (жолдық) айнымалыны енгізгенде, & символы жазылмайды.

Енді С тілінде енгізу/шығару функцияларын пайдаланатын бір мысал келтірейік.

```
#include <stdio.h>
main()
{   int i;
    clrscr();
    printf("\n Бүтін сан енгізіңіз:");
    scanf("%d",&i);
    printf("\nСіз %d санын енгіздіңіз.",i);
}
```

Бұл программаның алғашқы жолы – препроцессор директивасы, ол енгізу/ шығару операцияларын орындауды қамтамасыз етеді. **main()** функциясының бірінші жолы бүтін типтегі **i** айнымалысын сипаттап тұр, одан кейін тұрған **printf**() функциясы жаңа жолға көшіп **(\n)** экранға **Бүтін сан енгізіңіз:** деген сөздерді шығарады. **scanf()** функциясы пернеден енгізілген санды **i** айнымалысына меншіктейді (**&** таңбасы адрес алу операциясын көрсетеді). Келесі жол көрсетілген сөз тіркесін ондағы спецификаторды **i** санының мәніне алмастыра отырып экранға шығарады.

Программа нәтижесі экранда мынадай түрде бейнеленеді:

Бүтін сан енгізіңіз: 1256

Сіз 1256 санын енгіздіңіз.

Енді енгізілген санның көрсетілген дәрежесін есептейтін программа құрайық.

```
/* Санды дәрежелеу */
#include <conio.h>
#include <stdio.h>
#include <math.h>
main()
{ float x,y,s;
clrscr();
printf("\n x-ті және оның дәрежесін - y енгізіңіз:");
scanf("%f%f",&x,&y);
s=pow(x,y);
printf("\nНәтижесі s=%f",s);
}
```

Бұл программаны орындаудағы экран бейнесі төмендегідей болады:

x-ті және оның дәрежесін - y енгізіңіз: 2.5 4
Нәтижесі s=39.062500

Сонымен, **scanf()** функциясы символдарды, сандарды, сөз тіркестерін енгізу үшін қолданылады, енгізілетін сандар, сөздер бір-бірінен бос орын, табуляция символы немесе **Enter** пернесін басу арқылы ажыратылады екен.

6.3 Cin және cout функциялары

C++ тілі стиліндегі негізгі енгізу/шығару функциялары оның кластары кітапханасын пайдаланады. Оны түсіндіру үшін бір программа мысалын қарастырайық.

```
#include <iostream.h> // C++ тіліндегі программа
int main()
{
int i;
cout << "Бүтін сан енгізіңіз:\n";
cin >> i;
cout << "Сіз " << i << "санын енгіздіңіз, рахмет!";
return 0;
}
```

Мұндағы <**iostream.h**> – C++ тіліндегі енгізу/шығару кітапханасының стандартты тақырып файлы. Ол программадағы мәліметтер ағымы жайлы ақпарат береді, **iostream** сөзі – **input/output stream** сөздерінің қысқаша жазылуынан шыққан сөз. Бұл файлда мәліметті пернетақтадан енгізуге арналған стандартты **cin** ағымы және экранға мәлімет шығаратын **cout** ағымы анықталып, ағымға мәлімет беру << және ағымнан мәлімет оқу >> операциялары арқылы орындалады.

cin – ағылшынша "C" және "input" сөздерінен, ал **cout** – "C" және "output" сөздерінен құралған.

cout << "Бүтін сан енгізіңіз:\n"; жолы экранға қос тырнақшадағы сөз тіркесін шығарып, курсорды келесі жолға көшіреді.

cin >> i; жолы пернелерден енгізілген бүтін санды **i** айнымалысына меншіктейді.

cout << "Сіз " << i << "санын енгіздіңіз, рахмет!"; жолы қос тырнақшадағы тіркестерді өзгеріссіз, ал **i** айнымалысы орнына оның енгізілген сандық мәнін экранға шығарып береді.

С тілінің енгізу/шығару функцияларына қарағанда, С++ тілінің ағымдары әртүрлі типтермен жеңіл, әрі жылдам жұмыс істеуді қамтамасыз ете алады деп саналады.

Ағымнан мәлімет оқу оны *ағымнан алу* деп, ал ағымға мәлімет шығару оларды *ағымға қосу* деп айтылады.

С++ тілінде бүтін сандармен арифметикалық амалдар орындайтын тағы бір мысал қарастырайық.

```
/* ia және ib бүтін сандарын қосу, азайту,
көбейту және бөлу операцияларын орындау */
#include <iostream.h>
void main()
{    // мәліметтерді сипаттау
int ia, ib, iplus, iminus;
float del, mult;
     // сандық мәндерді енгізу
cout << "Input ia, ib: \n";
          //"ia, ib енгіз:" сөзін шығару
cin >> ia >> ib;
    // енгізілген мәндерді ia, ib-ға меншіктеу
    // есептеулер
iplus = ia + ib;
iminus = ia - ib;
mult = ia * ib;
del = ia / ib;
    // нәтижелерді шығару
cout << "plus = " << iplus << "\n";
cout << "minus = "<< iminus <<"\n";
cout << "del = " << del <<"\n";
cout << "mult = " << mult <<"\n";
}
```

Программа нәтижесі:

```
Input ia, ib:
4 5
plus = 9
minus = -1
del = 0
mult = 20
```

Пернетақтадан **ia** мен **ib** мәндерін енгізу кезінде арасында бос орын болуы тиіс. Мысалда 4 цифрын енгізіп, бос орынды басып, 5 цифрын теріп, **Enter** пернесін басу қажет.

Бөлу кезінде шығатын нәтижеге назар аударыңдар, ол дұрыс болмайды. Бұл бүтін санды бүтін санға бөлгендегі бөлінді де бүтін сан болатындығынан шығып отыр. Мұндайда дұрыс нәтиже алу үшін, **ia**, немесе **ib** айнымалыларының бірінің типін **float** деп көрсету керек.

Бақылау сұрақтары

1. *C/C++ тілдеріндегі мәліметтер типі нелерді анықтайды?*
2. *C тілінің мәліметтері типтері, C++ тілінің типтері.*
3. *Мәліметтері типтерін анықтауда қандай спецификаторлар қолданылады?*
4. *Бүтін сан түріндегі типтер түрлері, олардың ұзындықтары.*
5. *char және wchar_t типтерінің ерекшеліктері.*
6. *Жылжымалы нүктелі нақты сандар типтері.*
7. *`sizeof()` операциясының қолданылуы.*
8. *Символдық тіркестерді сипаттау және беру.*
9. *Сөз тіркестеріндегі нөлдік символдың атқаратын қызметі.*
10. *C тіліндегі мәліметтерді енгізу және шығару функциялары. Олардың форматтары, жазылу ерекшеліктері.*
11. *C++ тіліндегі мәліметтерді енгізу және шығару функциялары. Олардың форматтары, жазылу ерекшеліктері.*

Тапсырмалар

1. Келесі константалар қандай типтерде бейнеленген:
   45L, 08UL, -0X12Lu, 1245, 67.1, -125.0 , 42165421, 12.56e-02, 2E+3L, 2.1f
2. Төмендегі өрнектер мәні нешеге тең болады:
   а) 1/3; 1.0/3; 1/3.0 ; ә) int a=5, k=10, y; y=a/k; y=?
3. "Абай Құнанбаев", "Мұқағали Мақатаев", "Қасым Аманжолов",

"Баубек Бұлқышев" сөздерін бір жолға, екі жолға, үш жолға және төрт жолға шығару керек.

4. Төмендегі идентификаторлардың қайсысы дұрыс, қайсысы қате жазылған:
   Zat, kun, iBas, A_arpi, лфк_12, adet_ұұрып, dt15$tr, wiwa_1987_wini, gamma_8
5. 125 ондық санын сегіздік және оналтылық жүйеде экранға шығарыңдар.
6. `printf (" Пи санының мәні = %4.2f\n", pi);` жолы экранға қандай мәлімет шығарады?
7. `scanf("%d%f", &a,&b);` жолы қандай сандар енгізуді талап етеді?
8.
```
cout << "Бүтін сан енгізіңіз: ";
cin >> k;
cout << k << "тақ сан ба? ";
```
   жолдары қандай нәтиже береді?

7 C/C++ ТІЛДЕРІНДЕГІ ҚОЛДАНУШЫ ФУНКЦИЯЛАРЫ

Кез келген C/C++ тілінің программасы бірнеше функциялардан құралады және олардың біреуі міндетті түрде **main**() болады. Программаны орындау осы функцияның бірінші операторынан басталады.

Көбінесе функция белгілі бір мәнді есептеу үшін қолданылады, сондықтан функция аты алдында оның типі көрсетіледі. Функция жайлы төменде айтылады, әзірше ең керекті деген мағлұматтарды ғана қарастырамыз:

- □ егер функция мән қайтармайтын болса, онда оның типі **void** болады;
- □ функция тұлғасы (орындалатын ішкі операторлары) бір блок болып табылады, сондықтан ол жүйелі (фигуралық) жақшаға алынып жазылады;
- □ функциялар бірінің ішіне бірі кіріп қабаттаса орналаспайды;
- □ әрбір оператор нүктелі үтірмен аяқталады (құрама операторлардан басқалары).

Енді С тілі стиліндегі енгізу/шығару ішкі функцияларын ғана пайдаланатын программа мысалын қарастырайық.

7.1 мысал.

```
/* Шеңбер ұзындығын табу  */
#include <stdio.h>     /* енгізу/шығару директивасы */
int main()             /* басты функцияны қолдану */
{
 int radius;           /* бүтін айнымалыны сипаттау */
 float length;         /* нақты айнымалыны сипаттау */
 printf("\n Радиус мәнін енгізіңіз: \n");
 scanf("%d",&radius);
 length =2*3.14159*radius;
 printf("Радиусы - %d\n шеңбер ұзындығы -%f",
     radius,length);
}
```

Программада түсініктемелер беру үшін /* және */ таңбалары қолданылған, олардың ішіне қазақша, орысша, ағылшынша сөз тіркестерін жазуға болады.

Басты функция **main()** аргументсіз жазылған, сол себепті жақша ішінде ешнәрсе көрсетілмеген. Ал функция тұлғасы операторлардан (немесе басқа функциялардан) тұруы тиіс. **Int** түйінді сөзі **radius** айнымалысының бүтін мән қабылдайтынын, **float** түйінді сөзі **lenth** айнымалысының нақты мән қабылдайтынын сипаттап тұр.

Келесі жол радиус мәнін енгізуді талап ететін сөз тіркестерін экранға шығарады, мұндағы **\n** таңбалары сөз тіркесі алдында және одан кейін курсор бір жол төмен түсетінін көрсетеді. **Scanf** функциясы шеңбер радиусы **radius** мәнін пернелерден қабылдайды, оның алғашқы аргументі **%d** енгізілетін ондық бүтін сан екенін, ал екінші аргументі – енгізілетін сан меншіктелетін айнымалы аты болып табылады. Оның алдындағы **&** (амперсанд) символы **scanf()** функциясының дұрыс жұмыс істеуі үшін керек, ол туралы кейін айтылады. Сонан кейін шеңбер ұзындығы есептеледі де, ол нәтиже ретінде **printf()** функциясы арқылы экранға шығарылады. Функция параметрі ретінде тұрған **%f** формат спецификаторы (анықтауышы) **length** айнымалысының типі **float** екенін көрсетеді.

7.1 Қолданушы функциясын пайдалану

Қолданушы (тұтынушы) функциясын пайдалану үшін, өзіміз **main** типтес шағын функция жазамыз. Ол бұрынғыша шеңбер ұзындығын анықтау үшін қолданылатын болсын.

7.2 мысал.

```
/* Шеңбер ұзындығын табатын функцияны С тілінде пайдалану  */
#include <stdio.h>      /* енгізу/шығару директивасы */
void lenth(float radius); /* lenth() функциясын, яғни оның
                             прототипін хабарлау */
main()                  /* басты функцияны қолдану */
{
 float radius;          /* нақты айнымалыны сипаттау */
 radius = 5.5;
 lenth(radius);         /* lenth функциясын шақыру */
 printf("\n Радиус мәнін енгізіңіз: ");
 scanf("%f",&radius);
 lenth(radius);
```

```
}
void lenth(float r)   /* lenth() функциясын анықтау */
{
 printf("Радиусы - %f\n шеңбер ұзындығы -%f",
     r,2*3.14159*r);
}
```

Бұл программада екі функция бар: **main()** және **lenth(radius)**. Енгізу/шығару директивасынан кейінгі жолда **lenth()** функциясы хабарланған, С тілінің ережесі бойынша әрбір айнымалы, функция оны алғаш қолданғанға дейін хабарлануы тиіс. Нүктелі үтірмен аяқталып отырған бұл функция тақырыбы функция прототипі деп аталады да, ол функцияны хабарлау (жариялау) болып табылады. **Main()** функциясы аяқталып жүйелі жақша жабылған соң, **lenth(..)** функциясы толық сипатталып анықталуы тиіс.

lenth(..) функциясы анықтау оның атын және жақша ішінде типі көрсетілген аргументін жазудан тұрады, аргументтер бірнешеу болуы да мүмкін. Функция тақырыбында, алғашқы жолында көрсетілген аргументтер формальды параметрлер деп аталады. Функция тақырыбынан соң, жүйелі жақшаларға алынған оның ішкі жұмыс операторлары орналасады, мұнда ол бір ғана оператордан құралған. Бұл **lenth()** функциясы аргументінің нақты мәні **main()** функциясындағы оны шақыру кезінде анықталады, оны нақты параметр деп атайды. Функцияны шақыру кезіндегі нақты параметр типі оның бастапқы жариялануы кезінде берілген формальды параметрі типімен бірдей болуы тиіс.

Функцияны жариялау кезінде көрсетілген **void** түйінді сөзі функцияның ешқандай да мән қайтармайтындығын білдіреді. Көбінесе осындай қолданушы анықтаған функция оны шақырған **main()** функциясына мән қайтарып беруі керек. Қайтарылатын мән **return** операторы арқылы көрсетіледі.

return операторы. Функциядан кері оралу операторы **return** функция жұмысын аяқтап, басқаруды оны шақыру нүктесіне береді. Оператордың жазылуы:

```
return [ өрнек ];
```

Егер **өрнек** типі **void** түрінде сипатталса, өрнек жазылмауы тиіс.

С тілінің ANSI стандарты және C++ тілі қолданушы функциясының прототипін жариялауға қатаң талаптар қояды. Функция прототипінде оның типі, аты және формальды параметрлері саны мен типтері толық көрсетіледі, параметрлердің аттарын жазбауға да болады.

Сонымен, қолданушы функциясының прототипі былай көрсетіледі:

```
float func(int n, float f, long double g);
```

немесе

```
float func(int, float, long double);
```

Енді осы айтылған тұжырымдарды қолдана отырып, алдыңғы мысалдың тағы бір нұсқасын келтірейік.

7.3 мысал.

```
/* Шеңбер ұзындығын табатын функцияны пайдалану, 2 нұсқа */
#include <stdio.h> /* енгізу/шығару
                      директивасы, С тілінде */
float length(float radius); /* lenth() функция-
        сын, яғни оның прототипін хабарлау */
main()          /* басты функцияны қолдану */
{
 float radius;  /* нақты айнымалыны сипаттау */
 radius = 5.5;
 printf("Радиусы %f шеңбер ұзындығы %f\n",
      radius,length(radius));
                /* lenth функциясын шақыру */
 printf("Радиус енгізіңіз: ");
 scanf("%f",&radius);
 printf("Радиусы %f шеңбер ұзындығы %f\n",
     radius,length(radius));
                /* length функциясын шақыру */
}
float length(float r)
               /* length функциясын анықтау */
{
 return 2*3.14159*r;
}
```

Бұл мысалда **length()** функциясы **return** операторы арқылы **2*3.14159*r** мәнін қайтарады.

Сонымен, функцияны қарапайым анықтау форматы мынадай болады:

```
типі аты ([ параметрлері ])
{
  функцияның ішкі орындалатын операторлары
 }
```

Енді программалау оқулықтарында жиі кездесетін "Сәлем, әлем!" сөзін ағылшын тілінде экранға шығаруды, C++ тілінің қолданушы функциясы арқылы орындайық.

7.4 мысал.

```
#include <iostream.h>
void hello()
{
 cout << "Hello, world!\n";
}
void main()
{
  hello();
}
```

Программаны орындау нәтижесі:

```
Hello, world!
```

Мұнда қолданушы функциясы толығынан негізгі **main()** функциясынан бұрын анықталған, сондықтан функция прототипін жазу қажет емес, оның орнында функцияның анықталуы тұр.

Бір функция орындалуы барысында келесі басқа функцияны шақыруы мүмкін. Мысалы, келесі программада екі функция қолданылады. Алдымен **main** функциясы **hello_3** функциясын шақырады, ал ол функция ішінде **hello** функциясы цикл операторы арқылы үш қайтара шақырылады. Қолданушы функцияларын жазғанда, онда айнымалыларды пайдалану қажеттілігі туындайды. Айнымалылар **main** функциясындағы тәрізді функцияда да жариялануы, яғни алдын ала сипатталуы қажет.

7.5 мысал.

```
// C++ тілінде жазылған программа
#include <iostream.h>
void hello()
{
  cout << "Hello, world! \n");
}
```

```
void hello_3()
{
int n;
for (n = 1; n <= 3; n++)
  hello();
}
void main()
{
 hello_3();
}
```

Программаны орындау нәтижесі:

```
Hello, world!
Hello, world!
Hello, world!
```

Келесі бөлімдерде осы C және C++ тілдерінің мүмкіндіктерінің екеуінен де мысалдар келтіріледі, бірақ бір программада бұл екеуін араластырмай, жеке-жеке қарастыру ұсынылады.

7.2 Айнымалылар мен өрнектерді пайдалану ерекшеліктері

Айнымалылар деп мәліметтердің белгілі бір типін сақтайтын компьютер жадының ат қойылған аймағын айтуға болады. Қолданылудан бұрын әрбір айнымалы сипатталуы тиіс. Бүтін **a** айнымалысы мен нақты **x** айнымалысын сипаттау мысалы:

```
int a; float x;
```

Айнымалыларды *сипаттау операторының* жалпы жазылуы:

```
[жады класы] [const] типі аты [инициализатор];
```

Осы оператордың құрамдық бөліктерінің берілу ережелерін қарастырайық.

- Міндетті түрде жазылмайтын жады класы мынадай мәндердің **auto, extern, static** және **register** бірін қабылдайды. Олар жайлы төменде айтылады.
- Модификатор болып табылатын **const** түйінді сөзі айнымалының мәні өзгертілмейтінін көрсетеді. Мұндай айнымалыны *атаулы константа* немесе жай *константа* деп атайды.
- Сипаттау кезінде айнымалыға бірден бастапқы мән беруге болады, оны *инициалдау* деп атайды. Инициализатор мәнін екі түрде жазуға болады – меншіктеу таңбасы арқылы:

= мәні

немесе жай жақша ішінде:

(мәні)

Константалар хабарлау кезінде инициалдануы тиіс. Бір операторда типтері бірдей болып келген бірнеше айнымалыларды үтірлер арқылы бөле отырып сипаттауға болады.

Мысалдар:

```
short int a = 1;   // бүтін a айнымалысы
const char C = 'C'; // символдық C константасы
char s, sf = 'f';
          // тек sf айнымалысы инициалданған
char t (54);
          // t-ға 54-ті меншіктеу (инициалдау)
float c = 0.22,x(3),sum; // жариялау, инициалдау
```

Егер инициалдау кезінде меншіктелген мәннің типі айнымалы типіне сәйкес келмесе, онда белгілі бір ережелер бойынша *типті түрлендіру* операциясы орындалады.

Айнымалы сипаттамасында оның типі мен жады класынан басқа келісім бойынша айнымалының *әрекет ету аймағы* беріледі. Жады класы мен әрекет ету аймағы тек айнымалы сипаттамасына ғана байланысты емес, ол программадағы сипатталудың жазылған орнына да байланысты болады.

Идентификатордың әрекет ету аймағы – бұл оның өзімен байланысқан жады аймағымен қатынас құру үшін пайдалануға болатын программа бөлігі. Әрекет ету аймағына байланысты айнымалы локальдік (жергілікті) немесе глобальдік (ауқымды) болуы мүмкін.

Егер айнымалы блок ішінде (блок жүйелі жақшалармен шектелген бөлік) анықталған болса, онда ол *локальдік* болып табылады, оның әрекет ету аймағы – сипатталу нүктесінен бастап, осы блок соңына дейін, бұған ішкі блоктар да кіреді. Егер айнымалы кез келген блоктардан тыс, солардың сыртында анықталған болса, онда ол *глобальдік* болып саналады да, оның әрекет ету аймағы осы файлдағы сипатталу нүктесінен оның аяғына дейін болып есептеледі.

Айнымалының *пайдаланылу мерзімі* тұрақты (программаның орындалуы кезеңінде) және уақытша (блоктың орындалуы кезеңінде) болуы мүмкін.

Идентификатордың көріну аймағы деп идентификатормен байланысты жады аймағымен қарапайым қатынас құра алатын программа бөлігін айтады. Көбінесе көріну аймағы әрекет ету аймағымен бірдей болады, тек ішкі блокта сыртқы аймақтағы айнымалымен аттас болып келетін айнымалы сипатталған кезде ғана айырмашылық туындайды. Мұндайда сыртқы айнымалының әрекет ету аймағы блоктарға да қатысты болғанымен, ол ішкі блокта көрінбейді. Дегенмен, сыртқы айнымалы глобальдік болатын болса, көріну аймағына қатынасу операциясын (::) қолдана отырып, оны пайдалануға болады.

Жады класы программалық объектінің (немесе айнымалының) пайдаланылу (өмірлік) мерзімін және көріну аймағын анықтайды. Егер жады класы анық көрсетілмесе, онда компилятор айнымалының хабарлану мәтініне қарай оны өзі анықтайды.

Жады класын тағайындау үшін келесі спецификаторлар қолданылады:

`auto` – *автоматтық* айнымалы. Бұл айнымалы үшін жады стектен бөлінеді және қажеттілігіне қарай оның сипатталуы көрсетілген оператор орындалған сайын инициалданады. Осы айнымалы сипатталған блоктан шыққан кезде оған бөлінген жады босатылады. Оның қолданылу мерзімі – сипатталуынан бастап, блок соңына дейін жалғасады. Глобальдік айнымалыларда бұл спецификатор қолданылмайды, ал локальдік айнымалылар үшін ол үнсіз келісім бойынша орнатылады, сондықтан оны жазбауға болады, яғни айнымалы үшін жады класы нақты көрсетілмеген жағдайда, ол **`auto`** класына жатқызылады.

`extern` – сипатталатын айнымалының программаның басқа жерінде (басқа файлда немесе мәтіннің кейінгі жағында) анықталғанын білдіреді. Бұл спецификатор осы айнымалы сипатталған[1] программаның барлық модульдерінен оған қол жеткізуге болатын мүмкіндік жасауда қолданылады.

`static` – *статикалық* айнымалы. Қолданылу мерзімі –

1 Егер айнымалы белгілі бір операторда инициалданатын болса, онда мұндағы `extern` спецификаторы есепке алынбайды

тұрақты. Айнымалы анықталатын операторды алғашқы рет орындаған кезде бір рет инициалданады. Статикалық айнымалы оны сипаттайтын оператордың тұрған орнына байланысты глобальдік немесе локальдік болуы мүмкін. Глобальдік статикалық айнымалы тек өздері сипатталған модульде ғана көрінеді.

register – регистрлік айнымалы, **auto** спецификаторы секілді, бірақ мұнда жады мүмкіндігінше процессор регистрлерінен бөлінеді. Егер компилятордың ондай мүмкіндігі болмаса, онда регистрлік айнымалы **auto** айнымалысы сияқты өңделеді.

```
int a;          // 1 a глобальдік айнымалы
int main()
{ int b;         // 2 b локальдік айнымалы
extern int x;   // 3 x айнымалысы басқа жерде
                // анықталған
static int c;   // 4 c локальдік статикалық
                // айнымалы
a = 1;          // 5 глобальдік айнымалыға
                // меншіктеу
int a;          // 6 a локальдік айнымалы
a = 2;          // 7 локальдік айнымалыға
                // меншіктеу
::a = 3;        // 8 глобальдік айнымалыға
                // меншіктеу
return 0;
}
int x = 4;   // 9 x-ті анықтау және инициалдау
```

Бұл мысалда **a** глобальдік айнымалысы блоктардан тыс жарияланған. Оған жады – программа жұмысы басында мәліметтер сегментінде бөлінеді. Көріну аймағы – 6-8 жолдардан басқа программаның толық аймағы, өйткені 6-жолда глобальдік айнымалымен аттас локальдік айнымалы анықталған. Мұндағы **b** және **c** – локальдік айнымалылар, олардың көріну аймағы – блок, бірақ пайдаланылу кезеңдері әртүрлі: **b** үшін жады блокқа кірерде стекте бөлінеді де, одан шығарда жады босатылады. Ал **c** айнымалысы мәліметтер сегментінде орналасады да, оны программа жұмыс істеп тұрғанда пайдалана беруге болады.

Егер айнымалыға анықталу кезінде нақты мән берілмесе, компилятор глобальдік және статикалық айнымалыларға типтеріне сәйкес нөлдік мән меншіктейді, ал автоматтық айнымалыларға мән тағайындалмайды.

Айнымалы аты өз әрекет ету аймағында бірегей болуы тиіс (мысалы, бір блокта аттары бірдей екі айнымалы болмауы керек).

Айнымалы аттарын мағынасына қарай беруге тырысу қажет. Оның аты сол шаманың не екенін, не істейтінін көрсетіп, жеңіл оқылатындай дәрежеде болуы қажет. Атауларда бір-біріне ұқсас шатастырып алатындай символдар араласпағаны дұрыс, мысалы, `1` (бір), `l` (кіші `L`) немесе `I` (`i` бас әрпі). Жаңа атау табуға жалықпай ізденген абзал. Атаудың бөліктерін бір-бірінен айыру үшін астын сызу таңбасын пайдалануға болады. Кеңінен қолданылатын, көріну аймағы ауқымды айнымалыларға ұзынырақ ат (алғашқы әрпі типті көрсететін сивол) беріп, бірнеше жолдарда ғана пайдаланылатын айнымалыларға бір символдан тұратын ат беріп, оларға сипатталу кезінде түсініктемелер берген дұрыс.

Айнымалы сипаттамасы *хабарлау* немесе *анықтау* түрінде болуы мүмкін. *Хабарлау* компиляторға айнымалы типі мен жады класын көрсетеді де, *анықтау, оған қоса,* компиляторға айнымалы типіне сәйкес жады бөлінетінін білдіреді. C++ тілінде көптеген хабарлаулар, сонымен қатар, олардың анықтаулары да болып саналады. Жоғарыдағы мысалда 3 сипаттама берілген, олар хабарлау ғана, анықтау емес.

Айнымалы бірнеше рет жариялануы мүмкін, бірақ ол программаның бір-ақ жерінде анықталады, өйткені хабарлау айнымалының қасиеттерін сипаттайды, ал анықтау оны нақты жады аймағымен байланыстырады.

Өрнектер. Кез келген программада есептеу жұмыстары орындалады. Мәндерді есептеу үшін операндтардан, операция таңбаларынан және жақшалардан тұратын *өрнектер* қажет болады. Операндтар есептеуге керекті мәліметтерді береді. Операциялар орындалуға тиіс амалдарды анықтайды. Өз кезегінде, әрбір операнд өрнек немесе соның бір түрі, мысалы, константа немесе айнымалы болып табылады. Операциялар приоритеттеріне (математикалық реттіліктеріне қарай) сәйкес орындалады. Олардың орындалу тәртібін өзгерту үшін жай жақшалар пайдаланылады.

Сонымен, константалардан, айнымалылардан, функциялардан және операциялар таңбаларынан ***өрнектер*** құралады. *Әрбір өрнек арифметикалық операциялар таңбаларымен қажетті жақшалар көмегімен біріктірілген бірнеше операндтардан (сан, айнымалы, константа) тұрады*. Математикадағы формулалар, алгебрадағы көпмүшеліктер программалау тілінде тек осы өрнек ұғымы арқылы беріледі.

Егер өрнек мәні бүтін немесе нақты сан болатын болса, ол арифметикалық өрнек болып саналады. Арифметикалық өрнектерде мынадай операциялар: + - * / % болады. Жалпы өрнектер бір жол бойына жазылады және олардағы операция реттілігі жақшалармен анықталады. Өрнектерді жазу мысалдары:

```
i = i+1;  k = 5.35;  x1= (-b+sqrt(b*b-4*a*c))/(2*a);
y = sqrt(sin(x)+1);  c = 2*pi*r;  r = 19.36;
```

Енді өрнектер құрамы мен соларды жазу ережелерін қарастырамыз.

Көптеген арифметикалық операциялар жай алгебралық өрнектердегі амалдар приоритетін (реттілігін, басымдылығын) сақтай отырып жұмыс істейді, алдымен көбейту, бөлу және қалдық табу, соңынан — қосу мен азайту орындалады.

Егер өрнекте реттілігі бірдей бірнеше амал қатар тұрса, олар солдан оңға қарай орындалады. Мысалы, мына өрнекте

```
i = k + d * i / m % n - 1;
```

арифметикалық амалдар мынадай ретпен атқарылады: көбейту, бөлу, қалдық табу, қосу және азайту.

Бүтін сандарды бөлу операциясы: 7/4. Жауабы 1 болады. Нақты жауабын алу үшін 7/4.0 немесе 7.0/4, әйтпесе (float)7/4 деп жазу керек.

Бөлгендегі қалдықты табу операциясы **%** болып белгіленеді. Мысалы: **13%5** нәтижесі 3. **%** операциясын аралас, яғни нақты сандармен орындау синтаксистік қате береді.

Арттыру, кеміту операциялары. C/C++ тілдерінде инкремент ++ (1-ге арттыру) және декремент -- (1-ге кеміту) операциялары бар. i++ – бұл постфикстік соңынан жазу формасы; ал мынау ++j – префикстік форма (алдынан жазу. 3.1 кестеде осы инкремент пен декремент амалдарының орындалу ережелері көрсетілген.

Олардың бір-бірінен айырмашылығын төмендегі мысалдан көруге болады:

```
k = a + (i++) - d * (--j);
```

мұнда амалдар мынадай реттілікпен орындалады:

```
--j; k = a + i – d * j; i = i++;
```

7.1 кесте. **Инкремент пен декремент амалдарын пайдалану**

Операция	Операция аты	Өрнектің жазылуы	Атқарылатын әрекеттер
++	Инкременттің префикстік формасы	++a	Алдымен **a** 1-ге артады, содан кейін **a**–ның жаңа мәні осы **a** кездескен өрнекті есептеуде қолданылады
++	Инкременттің постфикстік формасы	a++	Осы тіркес кездескен өрнекті есептеуде **a**–ның ескі мәні қолданылады да, содан кейін барып **a**–ның мәні 1-ге арттырылады
--	Декременттің префикстік формасы	--b	Алдымен **b** 1-ге кемиді, содан кейін **b**–ның жаңа мәні осы **b** кездескен өрнекті есептеуде қолданылады
--	Декременттің постфикстік формасы	b--	Осы тіркес кездескен өрнекті есептеуде **b**–ның ескі мәні қолданылады, содан кейін барып **b** –ның мәні 1-ге азайтылады

Төменде көрсетілген белгі немесе белгілер тіркесі мағыналары:

= – меншіктеу белгісі,

== – теңдікті тексеру ("тең" дегенді білдіреді),

!= – теңсіздікті тексеру ("тең емес" дегенді білдіреді),

&&– логикалық ЖӘНЕ (мысалы, мынадай шартты тексеру: **if (x>10 && x<20)),**

|| – логикалық НЕМЕСЕ (мысалы, мынадай шартты тексеру: **if (x == 1 || y != 0)).**

Тағы бір мысал қарастырайық.

```
/* Программада C++ стилінде әртүрлі арифметикалық және логикалық операциялар орындалады */
#include <iostream.h> // енгізу/шығару үшін
                      // тақырыптық файл
```

```
void main()
{
int ii=2,ij=5,il,im,in;  /* жариялау (айнымалы-
лар типтерін көрсету) және 2 айнымалыны инициал-
дау (бастапқы мән беру) */
double dk,da=8.2,dd=.781,dc=-13.1;
                        // жариялау, инициалдау
/* ****** Арифметикалық операциялар орындау ****** */
il = 8 % ij; cout << "\n" << "8 % 5 = " << il;
// қалдық табу
dk = ii + ij * dc - 5 / ii + ij % ii;
cout << "\n"<< "k = " << dk;
dk = (ii + ij) * dc - 5 / (ii + ij) % ii;
cout << " k = " << dk; /* арифметикалық
 операциялар реттілігі жақшалармен өзгертілген */
dk = da + (ii++) - dd * (--ii);
cout <<" k =" << dk;
dk = da + (++ii) - dd * (ii--);
cout << " k = " << dk; // инкремент пен декременттің
       // постфикстік және префикстік формалары
/*
********* Логикалық операциялар *********
Мұнда if шартты операторы қолданылған. Егер
жақшадағы шарт ақиқат болса,жақшадан кейін
тұрған оператор, әйтпесе одан кейінгі оператор
орындалады */
cout << "\n" << "Input int m = n = 5 ";
cin >> im >> in;
if(im == in) cout << "The condition is fulfilled";
cout << "\n" << "Input int m = 6 and int n = 3 ";
cin >> im >> in;
if(im != 5 && im > in) cout << "Both conditions
are fulfilled";
cout << "\n" << "Input int n = -15 ";
cin >> in;
```

```
if ( im != 5 || in <= 1)
   cout << "One of the conditions is fulfilled\n";
}
```

Программаны орындау нәтижелері:

```
8 % 5 = 3
k = -64.5 k = -91.7 k =9.419 k = 8.857
Input int m = n = 5 5 5
The condition is fulfilled
Input int m = 6 and int n = 3 6 3
Both conditions are fulfilled
Input int n = -15 -15
One of the conditions is fulfilled
```

Мұндағы үш **if** операторларының да шарттары ақиқат болғандықтан, үш рет **The condition is fulfilled** (Шарт орындалды), **Both conditions are fulfilled** (Екі шарт орындалды), **One of the conditions is fulfilled** (Бір шарт орындалды) деген мәліметтер шығарылды.

Бақылау сұрақтары

1. *Стандартты функция мен қолданушы функциясының қандай айырмашылықтары бар?*
2. *Қолданушы функциясын анықтау дегеніміз не?*
3. *Формальды параметрлер мен нақты параметрлер қайда жазылады?*
4. *Функцияны жариялау кезінде көрсетілген* **void** *түйінді сөзі нені білдіреді?*
5. **return** *операторы қандай қызмет атқарады?*
6. *Функция прототипі дегеніміз не?*
7. *Қолданушы функциясын анықтау* **main()** *функциясына дейін орындала ма әлде одан кейін атқарыла ма? Неге?*
8. *Қолданушы функциясын анықтау форматы қандай болады?*
9. *Айнымалыға бастапқы мән беру қалай орындалады?*
10. *Константалар қалай хабарланады?*
11. *Айнымалының әрекет ету аймағы дегеніміз не?*
12. *Идентификатордың көрінү аймағының оның әрекет ету аймағынан айырмасы.*
13. *Жады класы түсінігі. Автоматтық айнымалылар дегеніміз не?*
14. *Статикалық айнымалының қандай ерекшелігі бар?*
15. *Класы* **extern** *болып келген айнымалылар қандай болады? Регистрлік айнымалылар ше?*

16. *Глобальдік айнымалы мен локальдік айнымалының айырмашылығы.*
17. *Өрнек дегеніміз не?*
18. *Инкремент және декремент операциялары. Олардың префистік дәне постфикстік формалары.*

Тапсырмалар

1. Үш натурал сандар берілген. Олардың ең үлкен ортақ бөлгішін (ЕҮОБ) анықтайтын функцияны құру керек.
2. Натурал N саны берілген. Ол екі *x* және *y* бүтін сандарының квадраттарының қосындысына тең болатын болса $N=x^2+y^2$, онда *x, y* сандарын анықтайтын функцияны құру керек.
3. Натурал *N* саны берілген. K^2-қа бөлінетін және K^3-қа бөлінбейтін барлық натурал *K*-ларды табу керек.
4. Егер *N* цифрдан тұратын натурал санның цифрларының қосындысын *n*-ші дәрежеге шығарғанда, сол санның өзіне тең болатын болса, ондай сан Армстронг саны деп аталады (мысалы, $153=1^3+5^3+3^3$). Екі, үш және төрт цифрдан тұратын барлық Армстронг сандарын табу керек.
5. Екі ұшбұрыштың төбелерінің координаталары берілген. Олардың қайсысының ауданы үлкен екенін анықтау керек.
6. Жазықтықтағы үш түзу $a_kx+b_ky=c_k$ (k = 1,2,3) теңдеулерімен берілген. Егер ол түзулер қос-қостан қиылысып, үшбұрыш құрайтын болса, сол үшбұрыштың ауданын табу керек.
7. Екі жай санның бір-бірінен айырмашылығы 2-ге тең болса, олар **"егіздер"** деп аталады (мысалы, 41 және 43 сандары). [n, 2n] аралығындағы барлық "егіздерді" анықтау керек, мұндағы *n*–2-ден үлкен бүтін сан.
8. $C_n^m = \frac{n!}{m!(n-m)!}$ анықтау керек, мұндағы $n!=1\cdot2\cdot3\cdot\ldots\cdot n$, яғни *n* – санының факториалы.

Операциялар таңбалары. Операция (амал) таңбасы – бұл операндтармен (сан, айнымалы немесе мәні анықталатын өрнек) атқарылатын амалды анықтайтын бір немесе бірнеше символдар тіркесі. Олардың арасына бос орын қоюға болмайды. Операциялар оған қатынасатын операндтар санына қарай унарлық, бинарлық және тернарлық болып бөлінеді. Бір таңбаның өзі тұрған орнына байланысты әртүрлі мағына беруі мүмкін.

C/C++ тілдеріндегі операциялар тізімі олардың басымдылықтарына, яғни приоритеттеріне (приоритеттерінің кемуіне қарай реттелген) байланысты төмендегі кестеде берілген. Ондағы *унарлық операция – бір орынды, бинарлық – екі орынды, тернарлы – үш орынды операция* деген ұғымды береді. Кестеге кірмеген басқа операциялар кездескен кезінде түсіндіріледі.

8.1 кесте. **C++ тілінің негізгі операциялары**

Операция	Қысқаша сипатталуы
Унарлық операциялар	
`++`	1-ге арттыру
`--`	1-ге кеміту
`sizeof`	мәліметтің компьютер жадындағы енін (көлемін) анықтау
`~`	разрядтар бойынша кері жазу
`!`	логикалық терістеу
`-`	арифметикалық терістеу (унарлық минус)
`+`	унарлық плюс
`&`	адресті алу
`*`	адрессіздендіру (нұсқауыш типі)
`new`	компьютер жадын бөлу (беру)
`delete`	компьютер жадын босату
`(type)`	типті түрлендіру, жақша ішінде тип аты жазылады
Бинарлық және тернарлық операциялар	
`*`	көбейту
`/`	бөлу
`%`	бөлгендегі қалдықты табу
`+`	қосу
`-`	азайту

<<	солға ығыстыру
>>	оңға ығыстыру
<	кіші
<=	кіші немесе тең (артық емес)
>	үлкен
>=	үлкен немесе тең (кем емес)
==	тең
!=	тең емес
&	разрядтар бойынша конъюнкция (ЖӘНЕ)
^	разрядтар бойынша арифметикалық (аласталған) НЕМЕСЕ
|	разрядтары бойынша дизъюнкция (НЕМЕСЕ)
&&	логикалық ЖӘНЕ
||	логикалық НЕМЕСЕ
?:	шартты операция (тернарлық)
=	меншіктеу
*=	көбейтіп алып меншіктеу
/=	бөліп алып меншіктеу
%=	қалдықты тауып алып меншіктеу
+=	қосып барып меншіктеу
-=	азайтып барып меншіктеу
<<=	солға ығыстырып алып меншіктеу
>>=	оңға ығыстырып алып меншіктеу
&=	разрядтар бойынша ЖӘНЕ амалын орындап алып меншіктеу
|=	разрядтар бойынша НЕМЕСЕ амалын орындап алып меншіктеу
^=	разрядтар бойынша арифметикалық НЕМЕСЕ амалын орындап алып меншіктеу
.	тізбектеп есептеу

Кестедегі шартты және **sizeof** операцияларынан басқалары асыра жүктеле береді.

8.1 Операцияларды орындау

C++ тіліндегі операциялар тізімі олардың приоритеттеріне (басымдылығына) қарай 8.1-кестеде берілген.

Енді негізгі операциялардың орындалуын толығырақ қарастырайық.

Арттыру немесе кеміту (инкремент және декремент) операциялары ++ және -- түрінде жазылады, олар туралы жоғарыда айтылды.

Префикстік түрде (**++x немесе --x**) алдымен операнд өзгертіліп, сонан соң оның мәні өрнектің нәтижелік мәніне айналады, ал постфикстік түрде (**x++** немесе **x--**) өрнек мәні операндтың бастапқы мәнін қабылдайды да, операнд сонан кейін өзгереді. Бұлар жасырын түрдегі меншіктеу амалы болып табылады. Олар жеке оператор түрінде де жазылады:

i++; немесе **++i;**

Бұл екеуі де мынадай амалмен бірдей болып саналады

```
i = i + 1;
```

Мысалы:

```
k=10;
x=k++; // x=10 k=11
x=++k; // x=12 k=12
```

Арттыру/кеміту операцияларын өрнек ішінде де орындау мүмкіндігі:

```
sum=a+b++;  //алдымен a, b қосылады, сонан соң b 1-ге артады
sum=a+ ++b;  //алдымен b 1-ге артады, сосын барып a, b
             // қосылады
```

Арттыру/кеміту операцияларының приоритеттері өте жоғары, тек жақша ішіндегі операциялардың приоритеті олардан жоғары болады.

8.1 мысал.

```
#include <stdio.h> //C тілі стиліндегі программа
int main()
{
int x = 3, y = 3;
printf("Префикстік өрнек мәні: %d\n", ++x);
printf("Постфикстік өрнек мәні: %d\n", y++);
printf("Өзгертілген x мәні: %d\n", x);
printf("Өзгертілген y мәні: %d\n", y):
return 0;
}
```

Программа жұмысының нәтижесі:

```
Префикстік өрнек мәні: 4
```

```
Постфикстік өрнек мәні: 3
Өзгертілген x мәні: 4
Өзгертілген y мәні: 4
```

Жалпы инкремент операциясының операнды *L-мәні (L-value)* болды деп айтылады. Мұндай түрде мән енгізуге болатын белгілі бір жады аймағын нөмірлеуді (адрестеуді) жүзеге асыратын кез келген өрнек белгіленеді. Бұлай атау меншіктеу операциясына байланысты туындаған, өйткені оның сол жақ бөлігі (Left) операция нәтижесінің жады аймағының қай жеріне орналасатынын анықтайды. Айнымалы L-мәнінің бір түрі болып табылады.

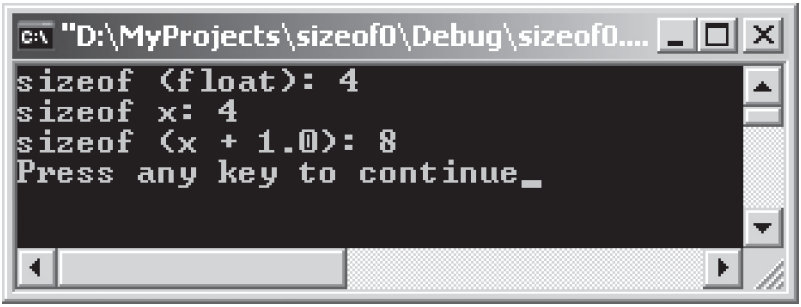
Компьютер жадындағы мәлімет көлемін (мөлшерін, енін) анықтау операциясы – **sizeof** объектінің немесе типтің алатын орнын байтпен анықтап береді, оның екі түрі бар:

```
sizeof  өрнек
sizeof ( тип )
```

Енді C++ тілінде бір мысал келтірейік.

8.2 мысал.

```
#include <iostream.h>
int main()
```



8.1-сурет. 8.2 -мысал нәтижесі

```
{
float x = 1;
cout << "sizeof (float) :"
   << sizeof (float);
cout << "\nsizeof x :" << sizeof x;
cout << "\nsizeof (x + 1.0) :"
```

```
   << sizeof (x + 1.0);
   return 0;
   }
```

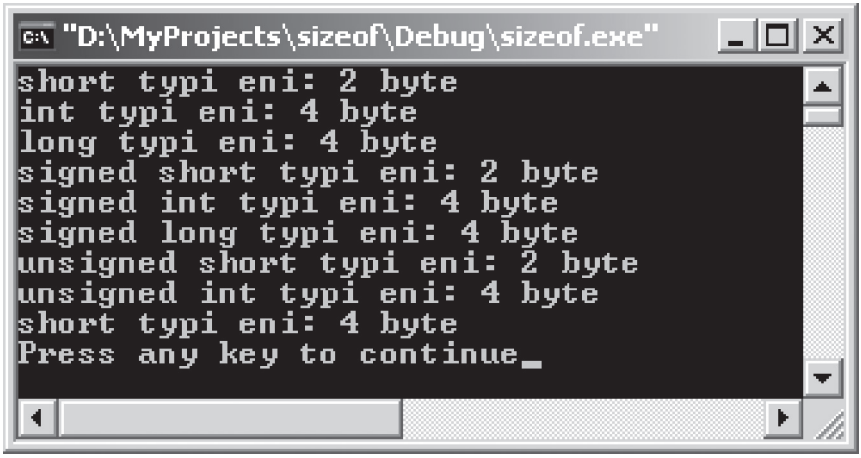
Программа жұмысы нәтижесі 8.1 суретте көрсетілген.

Компьютер жадындағы бүтін сандар типтерінің енін анықтау программасы төмендегідей болады.

8.3 мысал.

```
#include <iostream.h> // C++ тілінде берілген
int main()
{ cout << "short typi eni: " << sizeof(short) <<
          3"byte"  << endl;
 cout  << "int typi eni: " << sizeof(int) << " byte"
       << endl;
 cout   << "long typi eni: " << sizeof(long) << " byte"
       << endl;
 cout  << "signed short typi eni: " <<
      sizeof(signed short) << " byte" << endl;
 cout  << "signed int typi eni: " << sizeof(signed
      int)   << " byte" << endl;
 cout << "signed long typi eni: " << sizeof(signed
      long) << " byte" << endl;
 cout << "unsigned short typi eni: " <<
      sizeof(unsigned short) << " byte" << endl;
 cout << "unsigned int typi eni: "
      << sizeof(unsigned int) << " byte" << endl;
 cout << "short typi eni: "
      << sizeof(unsigned long) << " byte" << endl;
 return 0;
}
```

Терістеу операциясы (-, ! **және** ~). *Арифметикалық терістеу* (унарлық минус -) бүтін немесе нақты типтегі операндтың таңбасын қарама-қарсыға ауыстырады. *Логикалық терістеу* (!) операнд ақиқат болса, 0 мәнін және операнд нөлге тең болмаса (нөл емес), 1 мәнін береді. Операнд бүтін немесе нақты типте немесе нұсқауыш типінде болуы тиіс. *Разрядтар*



8.2-сурет. Бүтін сандар типтерінің енін анықтау программасы нәтижесі

бойынша терістеу (~), көбінесе биттік терістеу деп аталады, бүтін сан түріндегі операндтың екілік сан түріндегі бейнесінің әрбір разрядын терістейді (инверсиялайды). Мысалы, $\sim 118_{10} = \sim 1110110_2 = 0001001_2 = 9_{10}$.

Бөлу (/) **және қалдық табу** (%). *Бөлу операциясы* арифметикалық типтегі операндтарға қолданылады. Егер екі операнд та бүтін сан болса, бөлінді бүтін сан болады, әйтпесе бөлінді типі типтерді түрлендіру ережесіне сәйкес тағайындалады.

Қалдық табу операциясы тек бүтін типтегі операндтарға қолданылады.

8.4 мысал.

```
#include <stdio.h> // C тілі стилінде
int main(){
int x = 11, y = 4;
float z = 4;
printf("Бөлу нәтижесі: %d %f\n", x/y, x/z);
printf("Қалдық: %d\n", x%y):
return 0;
}
```

Программа жұмысы нәтижесі:

```
Бөлу нәтижесі: 2  2.750000
Қалдық: 3
```

Ығыстыру операциясы (<< және >>) бүтін сан түріндегі операндтарға қолданылады. Олар бірінші операндтың екілік бейнесін екінші операндта көрсетілген санға сәйкес оңға (>>) немесе солға (<<) ығыстыру ісін атқарады. *Солға* (<<) *ығыстыру* кезінде босаған разрядтар нөлдермен толтырылады. Ал *оңға* (>>) *ығыстыру* кезінде босаған разрядтар, бірінші операнд таңбасыз типте болғанда, нөлдермен толтырылады. Егерде бірінші операнд таңбалы типте болса, онда босаған орындар таңба разрядымен толтырылады. Ығыстыру операциялары толып кету (переполнение) және санның дәлдігін жоғалту әрекеттерін есепке алмайды.

Қатынас операциялары (<, <=, >, >=, ==, !=) бірінші операндты екінші операндпен салыстырады. Операндтар арифметикалық немесе нұсқауыштық типте болуы мүмкін. Операция нәтижесі **true** немесе **false** болады (нөлге тең емес кез келген мән нәтижесі **true**). Теңдікпен және теңсіздікпен салыстыру операцияларының приоритеті басқа салыстыру операцияларынан төмен болып саналады.

Екі шаманың тең екендігін (==) тексеру операциясы мен нәтижесі сол жақ операндқа берілетін мән болып табылатын меншіктеу операциясының (=) айырмашылығына назар аудару қажет.

Екілік разрядтар бойынша орындалатын операциялар (&, | , ^) тек бүтін санды типтегі екілік жүйедегі операндтарға қолданылады. Бұл операцияларды орындау барысында операндтар биттер бойынша қарастырылады (бірінші операндтың бірінші биті екінші операндтың бірінші битімен, бірінші операндтың екінші биті екінші операндтың екінші битімен, т.с.с. салыстырылады).

Разрядтық конъюнкцияда немесе *разрядтық ЖӘНЕ* (операция & болып белгіленеді) операциясында логикалық көбейту амалы орындалады да, тек екі операндтың да сәйкес орындардағы разрядтары 1-ге тең болған жағдайда нәтижелік бит 1-ге тең болады.

Разрядтық дизъюнкцияда немесе *разрядтық НЕМЕСЕ* (операция | болып белгіленеді) операциясында логикалық қосу ама-

1-операнд	0	0	1	1		1-операнд	1	0	1	0	0	1	1
2-операнд	0	1	0	1		2-операнд			1	1	0	1	0	1
Нәтиже	0	0	0	0		Нәтиже	0	0	1	0	0	0	1

лы орындалады да, екі операндтың тек біреуінің немесе екеуінің де сәйкес орындардағы разрядтары 1-ге тең болған жағдай-да нәтижелік бит 1-ге тең болады.

Разрядтар бойынша аластайтын НЕМЕСЕ (операция ^ болып белгіленеді) операциясында екі операндтың тек біреуінің

1-операнд	0	0	1	1		1-операнд	1	0	1	0	0	1	1
2-операнд	0	1	0	1		2-операнд			1	1	0	1	0	1
Нәтиже	0	1	1	1		Нәтиже	1	1	1	0	1	1	1

ғана сәйкес орындарындағы разряды 1-ге тең болған жағдайда нәтижелік бит 1-ге тең болады.

8.5 мысал.

```
#include <iostream.h>
```

1-операнд	0	0	1	1		1-операнд	1	0	1	0	0	1	1
2-операнд	0	1	0	1		2-операнд			1	1	0	1	0	1
Нәтиже	0	1	1	0		Нәтиже	1	1	0	0	1	1	0

```
int main() {
 cout << "\n 6 & 5 = " << (6 & 5);
 cout << "\n 6 | 5 = " << (6 | 5);
 cout << "\n 6 ^ 5 = " << (6 ^ 5);
 return 0;
}
```

Программа жұмысының нәтижесі:

```
6 & 5 = 4
6 | 5 = 7
6 ^ 5 = 3
```

Операция:	**&**	**|**	**^**
1-операнд	1 1 0	1 1 0	1 1 0
2-операнд	1 0 1	1 0 1	1 0 1
Нәтиже	1 0 0	1 1 1	0 1 1

Логикалық операциялар (&& және ||). ЖӘНЕ (&&) мен НЕМЕСЕ (||) логикалық операцияларының операндтары арифметикалық типте немесе нұсқауыш түрінде бола алады, мұнда

әрбір операциядағы операндтар типтері әр түрлі бола береді. Операциялар орындалуы кезінде типтерді түрлендіру жасалмайды, тек әр операндтың нөлге эквиваленттілігі бағаланады (нөлге тең операнд **false**, нөлге тең емесі – **true** болып саналады**).**

Логикалық операцияның нәтижесі **true** немесе **false** болады**.** *Логикалық ЖӘНЕ* операциясындағы екі операндтың да мәні **true** болғанда ғана оның нәтижесі **true** болады. Ал *логикалық НЕМЕСЕ* операциясындағы екі операндтың бірінің мәні **true** болғанда, оның нәтижесі **true** болады. Логикалық операциялар солдан оңға қарай орындалады. Егер бірінші операндтың мәні операция нәтижесін анықтауға жеткілікті болса, онда екінші операнд мәні есептелмейді.

Сонымен, бірнеше шарттық өрнектерді біріктіру үшін келесідей логикалық операциялар қолданылады:

1) ЖӘНЕ && (and) операциясы; f1 && f2
2) НЕМЕСЕ || (or) операциясы; f1 || f2 (| коды – 124)
3) ТЕРІСТЕУ ! (not) операциясы. !f1 (**!true = false, ! false = true**)

Терістеу (ЕМЕС) операциясының приоритеті өте жоғары, одан тек жақша ішіндегі мәндердің проритеті жоғарылау болады. && операциясының приоритеті || (or) операциясымен салыстырғанда жоғары, ал осы екеуінің приоритеттері қатынас операцияларымен салыстырғанда төмен, меншіктеу операциясынан жоғары болады.

8.2 Меншіктеу операциялары

Меншіктеу операциялары символдар (=,+=, −=, *=, т.с.с.) арқылы жазылады. Меншіктеу операциясы программада аяқталған оператор ретінде пайдаланылады. Кез келген ; таңбасымен аяқталатын өрнек меншіктеу операторы болып табылады. Өрнектің бір түріне бос оператор жатады, ол жай **;** операторы.

Қарапайым меншіктеу операциясының жазылуы:

```
1_операнд = 2_операнд;
```

Бірінші операнд – L_мән, ал екіншісі – өрнек болуы тиіс. Алдымен оң жақта тұрған өрнек есептеледі, сонан соң оның

нәтижесі сол жақта көрсетілген жады аймағына жазылады да (мұның мнемоникалық ережесі: "меншіктеу – мәліметтерді сол жаққа "беру""), онда бұрынғы сақталған мәлімет жойылады.

8.6 мысал.

```
#include <iostream.h> // C++ тілінде берілген
int main()
{
int a = 3, b = 5, c = 7;
a = b; b = a; c = c + 1;
cout << "a = " << a;
cout << "\t b = " << b;
cout << "\t c = " << c;
return 0;
}
```

Программа жұсысының нәтижесі:

```
a = 5 b = 5 c = 8
```

C/C++ тілдерінде меншіктеу операторының бірнеше түрі бар.

Жалпы меншіктеу операциясының жазылу форматы мынадай болады:

```
<айнымалы> = <айнымалы> <операция> <өрнек>;
```

Мұны қысқаша былай жазуға болады:

```
<айнымалы> <операция> = <өрнек>;
```

Төменде бірнеше мысал келтірілген.

```
a=a+b; → a+=b;        a=a*b; → a*=b;
a=a-b; → a-=b;        a=a/b; → a/=b;
```

C/C++ тілдерінде тізбектеле жазылған меншіктеу операцияларын да қолдануға болады. Мысалы:

```
sum = a = b;
```

Мұнда меншіктеу операциясы оңнан солға қарай орындалады, яғни b-ның мәні a-ға меншіктеледі, ал a-ның мәні sum-ға меншіктеледі.

Меншіктеу операцияларын былай да жазуға болады:

1) `a = (b = 1) + 2;`

мұнда `a=3, b=1`.

2) `a = b = 1 + 2;`

ал мұнда `a = 3, b = 3`.

Дөңгелек жақшаға алынған кез келген меншіктеу операторы анықталған мәні бар өрнек болып табылады, мысалы: **((s=13+12)<=30)** деген өрнек ақиқат мәнді болып табылады.

Шартты операция (? :) шартты өрнек жазуға мүмкіндік береді, яғни берілген шартқа байланысты әр түрлі мән қабылдайтын *шартты өрнектер* құрады. Бұл операция үшорынды болып табылады. Бірінші операнд арифметикалық типте немесе нұсқауыш болуы мүмкін. Ол нөлге эквиваленттілік тұрғысынан тексеріледі (нөлге тең операнд **false**, ал нөлге тең емесі – **true** болып табылады). Егер оның шарты (бірінші операнд) ақиқат болса, өрнек мәні екінші операндқа тең; егер жалған болса, онда – үшіншіге тең. Жазылуы:

```
1_операнд ? 2_операнд : 3_операнд;
```

8.7 мысал.

```
#include <stdio.h>
int main(){
  int a = 11, b = 4, max;
  max = (b > a)? b : a;
  printf("Максимум: %d", max);
  return 0;
  }
```

Программа жұмысының нәтижесі:

```
Максимум: 11
```

Тағы бір мысал келтірейік. Белгілі бір бүтін шама – **i** берілген **n** санынан артық болмаса, бірге артады, әйтпесе бірге тең болады:

```
i = (i < n) ? i + 1: 1;
```

Қалған операциялар кейін қарастырылады.

Өрнектер операндтардан, операция таңбаларынан және жақшалардан құралып, белгілі бір типтегі мәнді есептеу үшін қолданылады.

Өрнектерді жазу мысалдары:

i+1 5.35 (a+3.12)/5 (-b+sqrt(b*b-4*a*c))/(2*a) x && y || !z

Қатынас таңбасы арқылы біріктірілген екі арифметикалық өрнек мәні ақиқат (0-ге тең емес) немесе жалған (0-ге тең) деп айтылады. С тілінде логикалық тип түсінігі айтылмайды, ол С++ тілінде бар.

Операциялар өздерінің *приоритеттеріне* сәйкес орындалады, приоритеттер (басымдық) жай жақшалармен реттеледі. Егер өрнекте приоритеттері бірдей бірнеше операция – унарлық операциялар, шартты операция немесе меншіктеу операциясы жазылса, олар *оңнан солға* қарай, ал басқалары *солдан оңға* қарай орындалады. Мысалы, **a=b=c** тізбегі **a=(b=c)** болып саналады, ал **a+b+c** өрнегі **(a+b)+c** түрінде орындалады. Күрделі өрнектердің ішкі мүшелерінің есептелу реттілігін анық айту қиын, мысалы, мынадай өрнекте **(sin(x+2)+cos(y+1))** синус косинустан бұрын есептеледі деп айтуға болмайды және **x+2** өрнегі де **y+1** тізбегінен бұрын шығарылады деп те тұжырым жасауға болмайды.

Өрнектің есептелу нәтижесі оның мәнімен және типімен сипатталады. Мысалы, **a** мен **b** – бүтін типтегі айнымалылар ретінде мынадай түрде сипатталсын делік:

```
int a= 2, b = 5;
```

мұндағы **a+b** мәні 7 және типі **int**, ал **a=b** өрнегінің мәні **a** айнымалысында орналасқан мән болады да, оның тип де осының типіндей болып келеді. Сонымен, C++ тілінде **a=b=c** деп жазуға болады, мұнда алдымен **b=c** өрнегі есептеледі, сонан соң оның нәтижесі **a** айнымалысына меншіктелетін операцияның оң жақ операнды болып табылады.

Өрнекте әртүрлі типтегі операндтар қатар тұра береді. Егер олар бір типте болса, онда нәтиже де сол типте болады, ал операндтар әртүрлі типте болса, онда олардың дәлдігін сақтау үшін, қысқа типтер ұзынырақ типтерге белгілі бір ережелерге сәйкес түрлендіріледі.

8.3 Типтерді түрлендіру

Типтерді түрлендірудің екі түрі бар:

– шамалардың ішкі бейнесін өзгерту (дәлдікті жоғалту немесе жоғалтпау арқылы);

– шамалар ішкі бейнесінің тек интерпретациясын ғана өзгерту.

Бірінші типке, мысалы, бүтін санды нақты санға (дәлдікті жоғалтпай) және керісінше (дәлдікті жоғалту арқылы) түрлендіру әрекеті жатады, ал екінші типке – таңбалы бүтін сандарды таңбасыз типке ауыстыру әрекеттері жатады.

Жалпы, сандардың барлық мәндерін көрсете алатын кез келген жағдайда шамалардың **char, signed char, unsigned char,short int** және **unsigned short int** типтері **int** типіне түрлендіріледі, ал кері жағдайда **unsigned int** типіне ауыстырылады.

Егер өрнекте әр түрлі типтегі сандар мен айнымалылар қолданылса, онда олар жалпы бір типке түрлендіріледі. Біз қарастырған барлық негізгі типтер ішінде төменнен жоғары қарай бағытталған түрлендірілу реттілігі бар. Егер оларды оңға солға қарай реттеп орналастырсақ, мынадай болып шығады:

char → short → int → long → float → double

Оң жақтағылары сол жақтағылардан гөрі жоғары дәрежелі болып табылады.

Егер **char** мен **short** типтері араласса, нәтижесі – **short** болады,

ал **short** пен **int** типтері араласса, нәтижесі – **int** болады,

ал **int** пен **long** типтері араласса, нәтижесі **long**,

ал **long** пен **float** типтері араласса, нәтижесі **float**,

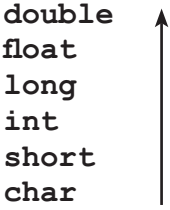
ал **float** пен **double** типтері араласса, нәтижесі **double** болады.

Егер екі-үш тип араласып, ең үлкен дәрежелісі – **float** болса, әрқайсысы да және нәтиже де осыған келтіріледі.

Компилятор типтерді автоматты түрде түрлендіру үшін төмендегі негізгі ережелер жиынын пайдаланады:

1. Егер операция екі түрлі типтегі мәліметтер үшін орындалатын болса, онда олар осы мәліметтер типтерінің арасындағы "жоғарғы" типке келтіріледі.

2. "Жоғары" типтен бастап, "төмен" типке дейін реттелген типтер аттарының тізбегі келесідей түрде көрсетіледі:



Меншіктеу операторында оң жақта орналасқан өрнектің есептелген нәтижесі осы оператордың сол жағына жазылған айнымалының типіне келтіріледі. Осындай процесс типтің "жоғарысына" немесе "төменіне" келтірілуі мүмкін.

8.8 мысал.

```
#include <stdio.h> // C тілі стилінде
#include <conio.h>
main ()
{
  char ch;
  int i; float fl;
  fl=i=ch='A';
  printf("ch=%c i=%d fl=%6.2f\n",ch,i,fl);
                // ch=A i=65 fl= 65.00
  ch=ch+1;      // ch=66
  i=fl+2*ch;    // i=65.00+2*66=197
  fl=2.0*ch+1;  // fl=2*66+1=133
  printf("ch=%c i=%d fl=%6.2f\n",ch,i,fl);
}               // ch=B i=197 fl=133.00
```

Келтіру операциясы. Жоғарыда көрсетілген типтердің түрлендірілуі автоматты түрде орындалады. Мәліметтердің көрсетілген қажетті типіне келтіру үшін C/C++ тілінде арнайы бір тәсіл бар. Бұл тәсілде типтерді бірыңғайлау үшін, айнымалының алдында дөңгелек жақшада қажетті типтің аты жазылады. Жалпы түрге келтіру операциясы мынадай болып жазылады:

(**тип**)**өрнек**. Мысалы:

```
int m;
float x,y;
y=pow(x,2)+sqrt((double)m);
```

Сонымен, ең қарапайым программаның өзі белгілі бір ережелерге сәйкес жазылса, ол сенімді түрде жұмыс істейтін болады екен.

Бақылау сұрақтары

1. *С тілінде қандай операциялар бар?*
2. *Меншіктеу операторының түрлері.*
3. *Меншіктеу операторының жазылу форматтары.*
4. *Арттыру немесе кеміту (инкремент және декремент) операциялары.*
5. *Префикстік және постфикстік операциялар.*
8. *Құрама операторлар қалай ұйымдастырылады?*
9. *Бос оператор деген не?*

10. Типтер ішінде төменнен жоғары қарай бағытталған түрлендірілу реттілігі.
11. Келтіру операцияларының жазылуы.

Тапсырмалар

1. А-ның берілген мәндері а) $a = 1.0$; ә) $a = 4$; б) $a = 5$ болған кездердегі x,b,a мәндерін анықтау керек:
   *b = 2.4*a;*
   *x = (a+b)/a*b-a;*
   x = a/b % b;
   *b = a*a-2*a;*
   a = (b++ + 2)(b - 1);*

2. Төмендегі операторларда жіберілген қателерді табыңдар:

а) $2\text{-}x = k + 4$;	д) $x = a/\text{-}\ b$;
ә) $x = x < 4$;	е) $y = y > 2$
б) $x = 3{,}74*a$;	ж) $5 = a - b$;
в) 3*k = m;	з) $p = 5.5\ \%\ 2$;
г) $\text{-}w = a + b$;	и) $x = \sin x + \cos x$;

3. Төмендегі өрнектерді алгоритмдік тілдерде жазыңдар:

а) $y = \dfrac{\sqrt{x^2+1} - \sqrt{5x-1}}{6x}$;

ә) $y = \dfrac{\sin 2x + tgx}{\ln|x|}$;

б) $y = \dfrac{\sin(\cos x) + 5{,}4 \sin\sqrt{|x|}}{\cos x - \sin x}$

4. Кубтың қабырғаларының ұзындығы берілген. Оның көлемі мен қабырғасының бетінің ауданын табыңдар.
5. Тікбұрышты үшбұрыштың катеттері берілген. Оның гипотенузасы мен ауданын табыңыздар.
6. Теңбүйірлі үшбұрыштың жақтары берілген. Үшбұрыштың ауданын табыңыздар.
7. x_1, y_1 және x_2, y_2 координаталарымен берілген нүктелердің ара қашықтығын табыңыздар.

8. Нақты x саны берілген. Тек көбейту, қосу және азайту амалдарын қолданып $2x^4 - 3x^3 + 4x^2 - 5x + 6$ есептеңіз. 4 көбейту, 4 қосу және 4 азайту амалын қолдануға болады.
9. Нақты x саны берілген. Тек көбейту, қосу және азайту амалдарын қолданып $1 - 2x + 3x^2 - 4x^3$ и $1 + 2x + 3x^2 + 4x^3$ есептеңіз. Тек 8 операция қолдануға болады.
10. x, y нақты сандары берілген. Тек 8 көбейту, 8 қосу және 8 азайту амалдарын қолданып, $3x^2y^2 - 2xy^2 - 7x^2y - 4y^2 + 15xy + 2x^2 - 3x + 10y + 6$ есептеңіздер.
11. x, y, z мәндері берілген, a және b мәндерін есептеңіздер.

$$a = \frac{2 * \cos(x - \pi / 6)}{1 / 2 + \sin^2 y}, \quad b = 1 + \frac{z^2}{3 + z^2 / 5}$$

12. x, y, z мәндері берілген, a және b мәндерін есептеңіздер.

$$a = \frac{1 + \sin^2(x + y)}{2 + \left| x - 2 * x / (1 + x^2 * y^2) \right|}, \quad b = \cos^2\left(\operatorname{arctg}\frac{1}{z}\right)$$

13. x, y, z мәндері берілген, a және b мәндерін есептеңіздер.

$$a = \ln\left|(y - \sqrt{|x|}) * \left(x - \frac{y}{z + x^2 / 4}\right)\right|, \quad b = x - \frac{x^3}{3!} - \frac{x^5}{5!}$$

14. a және b нақты сандары берілген. Осы сандардың қосындысын, айырмасын және көбейтіндісін табыңыздар.
15. Екі нақты оң сан берілген. Осы сандардың арифметикалық және геометриялық ортасын табыңыздар.
16. Екі нақты сан берілген. Осы сандардың абсолют шамасының арифметикалық және геометриялық ортасын табыңыздар.
17. Үшбұрыштың төбелерінің координатасы берілген. Үшбұрыштың периметрі мен ауданын табыңыздар.
18. Берілген *a, d, n* мәндері бойынша арифметикалық прогрессияның мүшелерінің

    $a, a + d, \ldots, a + (n - 1)d$

    қосындысын табыңыздар
19. Тікбұрышты үшбұрыштың гипотенузасы мен катеті берілген. Оның екінші катеті мен оған іштей сызылған шеңбердің радиусын табыңыздар.
20. Бір бірінен *r* арақашықтықта орналасқан, массалары m_1 және m_2 екі дене арасындағы *F* тартылыс күшін анықтаңыздар.
21. Тастың h биіктіктен жер бетіне құлау уақытын анықтаңыздар.
22. Шеңбердің ұзындығы белгілі. Осы шеңбермен шектелген дөңгелектің ауданын табыңыздар.
23. Ішкі радиусы 20-ға, ал сыртқы радиусы *r* ($r > 20$) болатын сақинаның ауданын табыңыздар.

9 С/С++ ПРОГРАММАЛАУ ТІЛДЕРІНІҢ ТАҢДАУ ОПЕРАТОРЛАРЫ

Компьютерлерде шығарылатын есептердің күрделіленуіне байланысты программалардың көлемдері өсіп, оларды жазу, оқу, түзету істері күннен күнге қиындап келеді. Белгілі бір өндірістік мәселенің, мезгіл-мезгіл қайталанып отыратын есептеулердің программалары ұзақ уақыт пайдаланылады, олар күнделікті өмір талабына сәйкес өзгертіліп, түзетіліп отыруы тиіс.

Осыларға байланысты программаларды құрастыруды, түсінуді, өзгертуді жеңілдететін тәсілдер жасалған, олар құрылымдық (структуралық) және объектіге бағытталған программалау жолдары деп аталады.

Программаларды адамның түсіну мен қабылдауын ыңғайлы түрде жүргізуге бағытталған тәсілдер жиынын ***құрылымды программалау*** деп атайды. Әрбір жасалған программа бөліктері бір-бірімен тығыз логикалық байланыста болып, оның бір жерінен екінші бір жеріне "секірулер" болмауы тиіс.

Құрылымды программалау – "**`go to`** операторысыз программалау" болып саналады, яғни шартсыз көшу операторын мүмкіндігінше пайдаланбауға тырысу керек, өйткені ол программа логикасын түсінуді қиындатады. Бірақ кейде **`goto`** операторын қолдану қажет болатын кездер болады.

Кез келген программа саны шектелген стандартты логикалық құрылымдардан тұрады. Негізгі логикалық құрылымдар туралы бұдан бұрын айтылған болатын, олар: *сызықтық, тармақталу* және *қайталау* құрылымдары. Осы үш құрылым программалаудың базалық негізгі конструкциялары болып саналады, олар күрделі программаларды жіктеп қарастыру мүмкіндігін береді.

Бұлар құрылымдардың ең көрнекті ерекшелігі – олардың орындалу алгоритмдерінің бір кіріс және тек бір шығыс сызығы болады. Мұндағы әрбір құрылымдық бірлік бір операторлан, біріктірілген операторлар тобынан немесе қабаттастырылған құрылымдардан да тұруы мүмкін. Осындай программалар оңай оқылады, түзетіледі және керек болса, оңай өзгертіледі.

С/С++ тілінде циклдің үш түрі, тармақталудың екі түрі бар. Олар программалауды жеңілдету мақсатында жасалған, сон-

дықтан әрбір жағдайда өзімізге ыңғайлысын таңдап алуымыз керек. Ең бастысы – кез келген программа айқын, әрі нақты түрде құрастырылған жеке блоктардан тұруы тиіс.

Программа жұмысын басқару операторларын программаның басқарушы конструкциясы деп атайды. Олар:

- құрама операторлар;
- таңдау операторлары;
- цикл операторлары;
- көшу операторлары.

"Өрнек" операторы. Нүктелі үтірмен аяқталатын кез келген өрнек белгілі бір мәнді есептейтін меншіктеу операторы болып саналады. Олар алдыңғы бөлімдерде қарастырылған болатын. Бос оператор да өрнектің бір түріне жатады, ол жай **;** операторы (бұл синтаксис бойынша оператор қажет етілгенмен, мағынасы бойынша ол керек болмайтын кезде пайдаланылады). Мысалдар:

```
k++;          // инкремент операциясы орындалады
a*= b+c;  // көбейте отырып меншіктеу операциясы
              // орындалады
fun(m,n);// функцияны шақыру операциясы орындалады
```

9.1 Тармақталу операторы

Тармақталу процестері бар алгоритмдерді ұйымдастыру үшін шартты операторлар пайдаланылады. Тармақталу белгілі бір шарттың орындалуы немесе орындалмауына тәуелді атқарылады. Кейде бір тармақ ішінде ешбір амал орындалмай да қала береді. Шарт ретінде логикалық өрнектің мәні пайдаланылады. C/C++ тілінде екі шартты оператор бар, олар: **if** және **switch**.

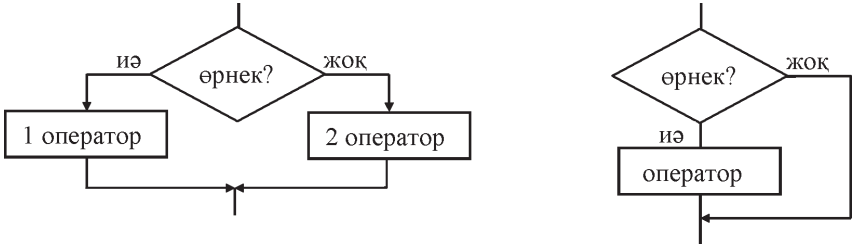
Шартты оператор. If *шартты операторы* – программадағы іс-әрекеттердің табиғи орындалу реттілігін өзгертетін мүмкіндіктің ең кең тараған тәсілі. Оның құрылымдық схемасы 9.1 суреттегідей болып өрнектеледі де, орындалуы кезінде есептеу жолы екіге тармақталып кетеді. Суретте көрсетілген құрылымдарға сәйкес бұл оператор мынадай түрде жазылады:

```
if (өрнек) 1-оператор; [ else 2-оператор; ]
```

(оқылуы `if` – иф, `else` – элс). Мұндағы 1-ші және 2-ші операторлардың өздері қарапайым немесе құрама оператор болуы мүмкін.

Алдымен **if** сөзінен соң жазылатын өрнек түрінде берілген шарт есептеледі, ол арифметикалық типте немесе нұсқауыштық типте болуы тиіс. Егер ол шарттың мәні **true** (иә), яғни ақиқат болса, онда 1-оператор атқарылады, онда **else** сөзінен кейінгі 2-оператор атқарылмайды. Екінші жағдайда, шарт мәні **false** (жоқ), яғни жалған болса, онда **else** сөзінен кейінгі 2-оператор атқарылып, 1-оператор атқарылмайды. **If** операторлары бірінің ішіне бірі кіріп қабаттасып та орындала береді.

Тармақтың бірі болмайтын жағдайда, **else** сөзінен кейінгі 2-оператор жазылмайды. Егер бір тармақта бірнеше операторларды орындау қажет болса, оларды блок түрінде жүйелі жақшаға алып жазу керек, әйтпесе компилятор тармақтың қай нүктеде аяқталатынын анықтай алмай қалады. Блокта кез келген операторлар жазыла береді, кейде онда тағы да шартты оператор орналасуы мүмкін.



9.1-сурет. Шартты операторлың құрылымдық алгоритмі

Әдетте, шарт өрнегін жазу үшін қатынас (салыстыру) белгілері ==, !=, >, >=, <, <= жиі пайдаланылады. Жалпы түрде шартты $k1 \otimes k2$ түрінде қарастыруға болады, мұнда $\otimes$– салыстыру белгілерінің бірі, $k1$, $k2$ – тұрақты, айнымалы немесе кез келген өрнек болуы мүмкін. Бір мезгілде бірнеше шартты, яғни күрделі құрама шартты жазу үшін логикалық амалдарды пайдалануға болады. Мысалдар:

```
if (a<0) b = 1;                                          // 1
if (a<b && (a>d || a==0)) b++; else {b *= a; a = 0;} // 2
if (a<b) {if (a<c) m=a; else m=c;}
  else {if (b<c) m=b; else m=c;}                         // 3
if (a++) b++;                                            // 4
if (b>a) max = b; else max = a;                          // 5
```

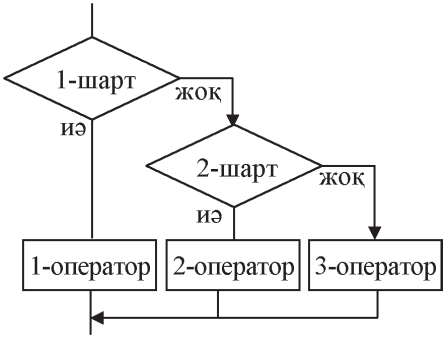
1 жолда **else** тармағы жоқ. Бұл аттап өту құрылымына сәйкес келеді, мұнда шартқа байланысты меншіктеу орындалады немесе атталып өтіледі.

Егер бірнеше шарттарды қатар тексеру керек болса, оларды логикалық операциялар таңбаларымен біріктіреді. Мысалы, *2 жолдағы* өрнек **a<b** шарты және жақшадағы шарттардың бірі орындалған жағдайда ғана ақиқат болады. Егер ішкі жақшаларды алатын болсақ, онда алдымен логикалық ЖӘНЕ (&&), сонан соң – НЕМЕСЕ (||) операциялары орындалады.

3 жолдағы оператор үш айнымалылардың ішіндегі ең кішісін анықтайды. Мұнда жүйелі жақшаларды жазу міндетті емес, өйткені компилятор операторлардың **else** бөлігін оған ең жақын тұрған **if** операторымен байланыстырады.

4 жол **if** операторында өрнек ретінде қатынас операциялары жиі пайдаланғанымен, ол да міндетті емес екенін көрсетеді. 5 *жолда* көрсетілгендей конструкцияларды шартты операция түрінде жазған көрнектілеу болып табылады (мысалы, мұнда: **max = (b > a) ? b : a;).**

Кейде қабаттасқан шартты операторлар кездеседі (9.2 сурет), мысалы:



9.2-сурет. Қабаттасқан шартты операторлар схемасы

```
if (1-шарт)
   1-оператор;
else if (2-шарт)
     2-оператор;
     else
     3-оператор;
```

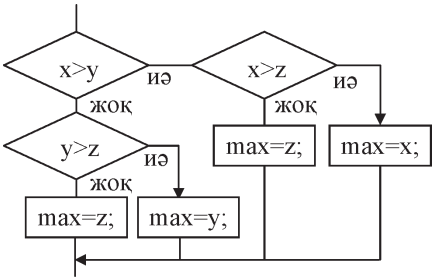
Мұнда егер 1-шарт ақиқат болса, 1-оператор орындалады, егер 1-шарт жалған болып, 2-шарт ақиқат болса, 2-оператор орындалады, ал 1-шарт және 2-шарт жалған болса, 3-оператор атқарылады. Ондағы кез келген **else** түйінді сөзі (keyword) оның алдында ең жақын тұрған **if** операторына қатысты болып саналады.

Мысалы, берілген ***x, y*** – екі санның үлкенін анықтау үшін жазылған шартты операторды былай жазуға болады:

```
if (x>y) max=x;
   else max=y;
```

Ал *x, y, z* сияқты үш санның үлкенін табу үшін, қабаттасқан шартты операторлар жазылады (9.3 сурет).

```
if (x>y)
 {if (x>z) max=x;
  else  max=z;}
else if (y>z) max=y;
     else max=z;
```

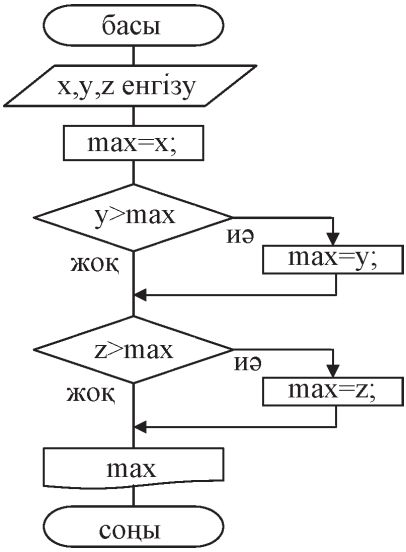


9.3-сурет. Үш санның үлкенін табу схемасы

Логикалық ЖӘНЕ операциясын пайдалана отырып, бұл есепті мынадай түрде де жаза аламыз.

```
if ((x>y)&&(x>z))
   max=x;
else if
 ((y>z)&&(y>x)) max=y;
     else max=z;
```

9.1 мысал. Осы алгоритмді толығынан қысқаша шартты оператор арқылы орындаудың блок-схемасы (9.4-сурет) мен программасын келесі түрде жазып шығайық.



9.4-сурет. Үш санның үлкенін табу схемасының екінші түрі

```
#include <stdio.h>
    // C тілі
#include <conio.h>
main ()
{
 clrscr();
 int max,x,y,z;
 printf("3 бүтін сан енгізіңіз:");
 scanf("%d%d%d",&x,&y,&z);
 max=x;
```

```
 if (y>max) max=y;
 if (z>max) max=z;
 printf("max=%d",max);
 getch();
}
```

9.2 мысал. Формула арқылы берілген төмендегі ***y*** функциясын есептеу программасын құрастыру керек.

$$y = \begin{cases} x+2, \text{ егер} & x < 0 \\ 2x^3, \text{ егер} & x \geq 0 \end{cases}$$

```
#include <stdio.h> // C тілі стилінде
#include <conio.h>
main ()
{
 clrscr();
 float x,y;
 printf("x нақты санын енгізіңіз:");
 scanf("%f",&x);
 if (x<0) y=x+2;
  else y=2*x*x*x;
 printf("\ny=%f",y);
 getch();
}
```

9.3 мысал. Программаға бір жыл нөмірін енгізіп, сол жылдың кәбисә (366 күн) немесе қарапайым жыл (365 күн) екендігін анықтау керек. Ол үшін жылды төртке бөлеміз, егер қалдық 0-ге тең болса, ол кәбисә жыл, әйтпесе қарапайым жыл болады.

```
#include <stdio.h>
#include <conio.h>
main ()
{ int gil;
int r; /* gil-ды 4-ке бөлгендегі қалдық */
clrscr();
printf("Жылды, мысалы, 2007 енгізіп, Enter
        басыңыз: ");
scanf("%i",&gil);
```

```
r=gil % 4;
if (r)
   printf("%i жыл - қарапайым \n", gil);
else
   printf("%i жыл - кәбисә \n", gil);
printf("\nАяқтау үшін Enter басыңыз");
getch();
}
```

9.4 мысал. Квадрат теңдеуді шешу программасын құру керек.

```
/* Квадрат теңдеуді шешу */
#include <stdio.h>
#include <conio.h>
#include <math.h>
main ()
{
float a,b,c;
float x1,x2,d;
clrscr();
printf("\n * Квадрат теңдеуді шешу * \n");
printf(" a,b,c мәндерін енгізіп, Enter басыңыз: ");
scanf("%f%f%f",&a,&b,&c);
d=b*b-4*a*c;
if (d < 0)
   printf("Теңдеудің шешуі жоқ \n");
else {
   x1=(-b+sqrt(d))/(2*a);
   x2=(-b-sqrt(d))/(2*a);
   printf("Теңдеу түбірлері: x1=%3.2f
           x2=%3.2f\n", x1,x2);
   }
printf("\nАяқтау үшін Enter басыңыз");
getch();
}
```

9.5 мысал. Төмендегі функция мәнін кез келген **x** үшін есептеу керек.

$$y = \begin{cases} 0, & егер \quad x \le 0 \\ x^2 - x, & егер \quad 0 < x \le 1 \\ \sin \pi x^2, & егер \quad x > 1 \end{cases}$$

```
#include <stdio.h>
#include <math.h>
#define pi 3.14159
main ()
{
float x,y;
clrscr();
printf("Нақты сан түріндегі x мәнін енгізіңіз: ");
scanf("%f",&x);
if (x <= 0)
   y = 0;
else if (x <= 1)
   y = x*x - x;
   else
   y = sin(pi * x*x);
printf("x = %f болғанда, y = %10.6f \n",x,y);
return(0);
}
```

9.6 мысал. Төменде алынған баллға сәйкес бағаны анықтау программасы келтірілген.

балл	баға
90...100	A
75...89	B
60...74	C
50...59	D
0...49	F

$$baga = \begin{cases} A, & егер \quad 90 \le ball \le 100 \\ B, & егер \quad 75 \le ball < 90 \\ C, & егер \quad 60 \le ball < 75 \\ D, & егер \quad 50 \le ball < 60 \\ F, & егер \quad 0 \le ball < 50 \end{cases}$$

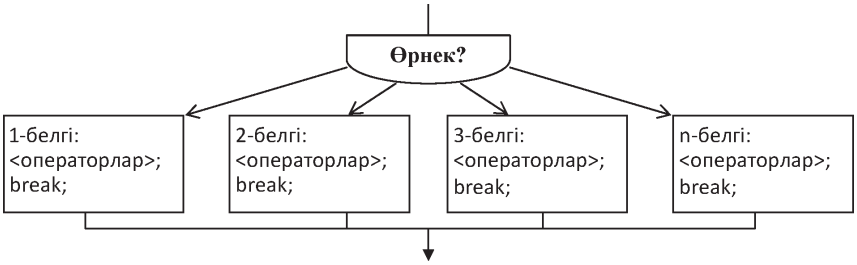
```
#include <stdio.h>
#include <conio.h>
main ()
{
int ball;
char baga;
clrscr();
printf("Балл мөлшері: ");
scanf("%i",&ball);
if (ball >= 90)
   baga = 'A';
else if (ball >= 75)
   baga = 'B';
else if (ball >= 60)
   baga = 'C';
else if (ball >= 50)
   baga = 'D';
else
   baga = 'F';
printf("Бағасы - %c, балл мөлшері - %i
        \n",baga,ball);
printf("\nАяқтау үшін Enter басыңыз");
getch();
}
```

9.2 Switch көп нұсқалы таңдау операторы

Программада кездесетін бірнеше нұсқаның бірін таңдап алу керек болған жағдайда, *switch* ауыстырғыш операторы қолданылады. Оператордың орындалу схемасы төмендегі 9.5 суретте келтірілген. Оның жалпы жазылу форматы:

```
switch <бүтін типті өрнек>;
{
 case белгі1: операторлар; break;
 case белгі2: операторлар; break;
 ................
 [default: операторлар;]
}
```

Мұнда switch сөзінен кейінгі өрнек мәні есептеледі, ол бүтін санды (char типі де) типте болуы тиіс. Сол мән **case** сөздерінен кейін жазылған константалар мәндерімен салыстырылады. Егер олардың біріне тең болса, сол жол орындалады, жол соңында көшу операторы болмаса, келесі жолдар толық орындалады. Ал бір жолды орындап болған соң, *switch* операторынан шығу үшін **break** операторы қолданылады. Егер **switch** сөзінен кейінгі өрнек мәні ешбір константамен сәйкес келмесе, онда **default** сөзінен кейінгі операторлар атқарылады. Кейде **default** сөзі болмауы да мүмкін.



9.5-сурет. Switch операторының орындалу схемасы

Default сөзі болмаса, онда *switch* операторынан кейінгі келесі операторлар орындала береді. *Switch* операторындағы өрнек түрінде нақты типтегі мәліметтерді, сөз тіркестерін (жолдарды) пайдалануға болмайды. Кейде бүтін мәндермен үйлестірілген мәліметтердің құрылымдық (структуралық) элементтері қолданылуы мүмкін.

9.7 мысал. Екі бүтін сан енгізіп, олармен арифметикалық 4 амалдың бірін орындау қажет.

```
#include <stdio.h> // C тілінде
#include <conio.h>
char symbol;
int x,y,z;
clrscr();
printf("Екі бүтін сан енгізіңіз: ");
scanf("%i%i",&y,&z);
printf("Қандай амал орындау керек: ");
```

```
scanf("%s",symbol);
switch (symbol)
{case '-' : x=y-z; break;
 case '+' : x=y+z; break;
 case '*' : x=y*z; break;
 case '/' : x=y/z; break;
 default: printf ("белгісіз операция\n");
 printf("\nНәтижесі - %d\n",x);
 getch();
}
```

9.8 мысал. Шығыс календары бойынша жылға сәйкес жануар атын анықтау.

```
#include <iostream.h> // C++ тілінде
main()
{   int god;
    cout << " Жылды енгізіңіз:\n";
    cin >> god;
    switch (god % 12)
 {  case 0 : cout << "Мешін жылы";break;
    case 1 : cout << "Тауық жылы"; break;
    case 2 : cout << "Ит жылы"; break;
    case 3 : cout << "Доңыз жылы"; break;
    case 4 : cout << "Тышқан жылы"; break;
    case 5 : cout << "Сиыр жылы"; break;
    case 6 : cout << "Барыс жылы"; break;
    case 7 : cout << "Қоян жылы"; break;
    case 8 : cout << "Ұлу жылы"; break;
    case 9 : cout << "Жылан жылы"; break;
    case 10 : cout << "Жылқы жылы"; break;
    case 11 : cout << "Қой жылы"; break;
  default: cout << "Таңбасыз бүтін сан енгізіңіз";
 }
}
```

Switch орындалуы кезінде цикл аяқталмай-ақ одан шығып, қалған операторларды аттап өтіп, осы цикл параметрінің келесі мәніне көшу үшін *continue* операторы қолданылады, яғни циклдің келесі итерациясына – қадамына басынан бастап ауысу жүзеге асырылады.

9.9 мысал:

```
#include <stdio.h> // C тілінде
main()
{     int i;
      printf("\nБүтін сан енгіз: )";
      scanf("%i",&i);
      switch(i)
      {case 1: printf("\nСан бірге тең!");
       case 2: printf("\n2*2=%d",i*i);
       case 3: printf("\n3*3=%d",i*i;break;
       case 4: printf("\n Сан төртке тең!";
       default: printf("\nАяқталды";
      }
}
```

Бұл программаның жұмыс нәтижесі:

1 енгізілгенде мыналар шығарылады:

Сан бірге тең!

2*2=1

3*3=1

2 енгізілгенде мыналар шығарылады:

2*2=4

3*3=4

3 енгізілгенде мыналар шығарылады:

3*3=9

4 енгізілгенде мыналар шығарылады:

Сан төртке тең!

Қалған сандар енгізілсе:

Аяқталды!

сөзі шығарылады.

Бақылау сұрақтары

1. *Құрылымдық программалау ұғымы.*
2. *Шартты операторлың толық және қысқа түрлері.*
3. *Қабаттасқан шартты операторлардың жазылуы. Олар қандай жағдайларда пайдаланылады?*
4. *Шарттық өрнектерді біріктіретін логикалық операторлардың қолданылуы.*
5. *Шартты операторды пайдаланып y=1/(x-1)+1/(x-2) мәнін есептейтін программа құрыңдар.*
6. *Шартты операция дегеніміз не?*
7. *Көп нұсқалы таңдау операторы не үшін қажет? Оның жазылу форматы қандай?*
8. *Көп нұсқалы таңдау операторына мысал келтіріңдер.*

Тапсырмалар

1. Тармақты алгоритмдерді программалау тәсілдерін пайдаланып, төмендегі функциялардың мәндерін есептейтін программа жазыңдар:

а) $y = \begin{cases} \dfrac{e^{\frac{\sin x}{2}} + \sin e^{\frac{x}{4}}}{x^2 + 2}, & \text{егер } x < 2 \\ 2x^2 + \dfrac{1}{\sqrt{3x}}, & \text{егер } x \geq 2 \end{cases}$

ә) $y = \begin{cases} \sin^2 2x + 5x^2, & \text{егер } x > 1,66 \\ \dfrac{61\sqrt{x} - 17}{\sqrt{4 + x^2 + \cos^2 4x}}, & \text{егер } x \leq 1,66 \end{cases}$

б) $y = \begin{cases} \dfrac{e^x + 1}{e^{x^2}}, & \text{егер } x < -2,4 \\ (x + \sin 4x) + \lg x^2, & \text{егер } x \geq -2,4 \end{cases}$

в) $y = \begin{cases} \sqrt{(\sin x + 1)} + \lg x^2, & \text{егер } x < -1 \\ \dfrac{\sin x + \cos x}{\cos x}, & \text{егер } -1 \leq x < 3 \\ e^{-\sin x} + \sin x, & \text{егер } x > 3 \end{cases}$

г) $z = \begin{cases} 5\sin y + \cos y, & \text{егер } y < 1 \\ \dfrac{y^2 - 2y - 5}{e^y}, & \text{егер } 1 \leq y < 4 \\ \sqrt{y^2 + 5} + \lg y, & \text{егер } y \geq 4 \end{cases}$ мұндағы $y = \begin{cases} \lg x + \sqrt{x}, & x \leq 2 \\ \lg 2x + \sqrt{3x}, & x > 2 \end{cases}$

$$д)\ y=\begin{cases}\sqrt{x}+\sin x, & егер\ \ x>0,4\\ \sqrt{2-x}+\cos x, & егер\ \ -1<x\le 0,4\\ e^{5x-1}+\lg x, & егер\ \ x\le -1\end{cases}$$

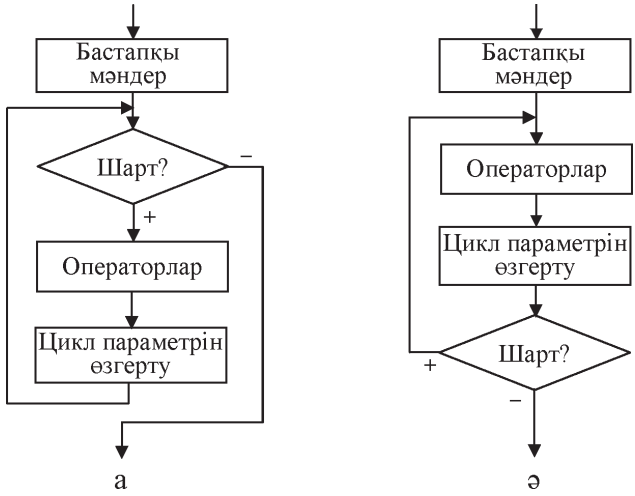
2. Жазықтағы нүктенің координаталары бойынша, оның қай ширекте жататындығын анықтайтын программа құрыңдар.
3. Жазықтықта үш нүкте берілген: $A(x_1,y_1)$, $B(x_2,y_2)$ және $C(x_3,y_3)$. Осы үш нүкте арқылы үшбұрыш құруға бола ма? Егер болса, Герон формуласын пайдаланып оның ауданын есептейтін программа құрыңдар.
4. Квадрат теңдеуді шешу программасын құрыңдар ($ax^2+bx+c=0$; $a \neq 0$)
5. Үшбұрыштың төбелері координаталар арқылы берілген: $A(x_1,y_1)$, $B(x_2,y_2)$, және $C(x_3,y_3)$. Бұл үшбұрыш тең қабырғалы, тең бүйірлі болатынын анықтау программасын құрыңдар.
6. Жазықтықта екі нүкте $N(x_1,y_1)$ және $M(x_2,y_2)$ берілген. Бұлардың қайсысы координаталар басына жақын болатынын анықтайтын программа құрыңдар.
7. Бүтін *n* санын берілген, ол бүтін *m* (*m*<*n*) санына қалдықсыз бөлінетінін анықтайтын программа құру керек.
8. Егер берілген *a* саны жұп болса, онда *p* атауына *true*, ал тақ болса *false* мәнін меншіктейтін программа құрыңдар.
9. Кез келген айдың бірінші жұлдызы аптаның қай күні екені белгілі болғанда, сол айдың енгізілген кез келген күнінің аптаның қандай күні болатынын анықтау программасын құру қажет.
10. Апта күндерінің реттік нөмірі бойынша олардың аттарын анықтайтын программа құру керек.
11. Нақты *x, y* ($x \neq y$) берілген. Кішісін олардың жарты қосындысымен, ал үлкенін - екі еселенген көбейтіндісімен алмастырыңыз.
12. Үш нақты сан берілген. Теріс емес сандарды квадраттаңыздар.
13. Егер берілген нақты *x, y, z* сандарының қосындысы 1-ден кем болса, онда бұл үш санның ең кішісін қалған екі санның жарты қосындысымен алмастырыңыз, кері жағдайда x және y-тің кішісін қалған екеуінің жарты қосындысымен алмастырыңыз.
14. Нақты *a, b, c, d* сандары берілген. Егер *a*<=*b*<=*c*<=*d* болса, онда әр санды ең үлкен санмен алмастырыңыз, егер *a*>*b*>*c*>*d* болса, садарды өзгеріссіз қалдырыңыз, кері жағдайда барлық сандарды олардың квадратымен алмастырыңыз.
15. Бүтін *x, y, z* сандары берілген. Егер *x y*-ке қалдықсыз бөлінсе және *y z*-ке қалдықсыз бөлінсе, онда барлық сандарға 1-ді қосыңыз, кері жағдайда барлық сандарды нөлге теңестіріңіз.

16. Нақты *a*, b, c, d сандары берілген. Осы сандардың терістерін квадраттап, ал оң сандарын түбірден шығарыңыз.
17. Нақты *a*, b, c, d сандары берілген. Егер кем дегенде бір сан нөлге тең болса, ол жайлы экранға мәлімет шығарыңыз, кері жағдайда a-ның b-ға және c-ның d-ға қалдықсыз бөлінетіндігін тексеріңіз.
18. Бүтін *a*, b, c сандары берілген. Егер *a*<=b<=c болса, онда барлық сандарды олардың квадратымен алмастырыңыз; егер *a*>b>c болса, онда әр санды ең үлкен санмен алмастырыңыз, кері жағдайда барлық сандардың таңбасын кері таңбаға ауыстырыңыз.
19. Нақты x, y, z сандары берілген. max (x + y + z, x*y*z) + 10 өрнегін есептейтін программа жазыңыз.
20. Нақты x, y, z сандары берілген. max($x^2 + y^2$, $y^2 + z^2$) – 1 өрнегін есептейтін программа жазыңыз.
21. Бүтін *k, l, m* сандары берілген. Нөлге тең сандардың санын анықтаңыз.
22. Бүтін *k, l, m* сандары берілген. Оң сандардың квадратының қосындысын есептеңіз. Егер бір де бір оң сан жоқ болса, ол жайлы экранға мәлімет шығарыңыз.
23. Бүтін *x* және *y* сандары берілген. Егер екі санда жұп болса оларға 1-ді қосыңыз; егер тек біреуі жұп болса, онда олардың көбейтіндісін табыңыз; қалған жағдайда сандарды өзгеріссіз қалдырыңыз.
24. Үш сан берілген. Солардың ішінен [0;1] аралығына кіретіндерін анықтау керек.
25. Нақты *x, y, z* оң сандары берілген. Қабырғаларының ұзындығы *x, y, z*-ке тең үшбұрыш тұрғызуға болатынын тексеріңіз. Сандарды енгізгенде олардың теріс емес және нөлге тең емес екендігін де тексеру қажет.
26. Нақты *x, y, z* сандары берілген. $\min^2(x+y+z/2, x*y-z) + 1$ өрнегін есептеңіз.
27. Бүтін *a, b, c, d* сандары берілген. Нөлге тең емес сандардың көбейтіндісін табыңыз. Егер барлық сандар нөлге тең болса, экранға мәлімет шығарыңыз.
28. Бүтін *a, b, c* сандары берілген. Тақ сандардың қосындысын табыңыз. Егер барлық сандар жұп болса экранға мәлімет шығарыңыз.
29. Бүтін *a, b, c, d* сандары берілген. Егер *a>=b>=c>=d* болса, онда барлық сандарды нөлге теңестіріңіз; егер *a<b<c<d* болса, онда әр санды 1-ге өсіріңіз; қалған жағдайда әр санды 1-ге кемітіңіз.
30. Бүтін *x, y, z* ($x \neq y, x \neq z, y \neq z$) сандары берілген. Осы сандардың ең кішісін тауып, оның жұп екендігін тексеріңіз.
31. 1 мен 7 аралығындағы кез келген сан берілген. Осы санға сәйкес апта күнінің атын шығарыңыз (1 – "дүйсенбі, 2 – "сейсенбі" және т.с.с.).

32. *K* бүтін саны берілген. Осы санға сәйкес бағаның сипаттамасын шығарыңыз (1 – "нашар", 2 – "қанағаттанарлықсыз", 3 – "қанағаттанарлық", 4 – "жақсы", 5 – "өте жақсы"). Егер сан 1–5 аралығында жатпаса, "қате" деген мәлімет шығарыңыз.
33. Айдың нөмірі 1–12 арасындағы бүтін сан арқылы берілген (1 – қаңтар, 2 – ақпан және т.с.с.). Ай атына сәйкес жыл мезгілінің атын шығарыңыз ("қыс", "көктем", "жаз", "күз").
34. Айдың нөмірі, 1–12 арасындағы бүтін сан арқылы берілген (1 – қаңтар, 2 – ақпан және т.с.с.). Сәйкес айдағы күндер санын анықтаңыз.
35. Арифметикалық амалдар келесідей ретпен нөмірленген: 1 – қосу, 2 – азайту, 3 – көбейту, 4 – бөлу. Амал нөмірі – *N* (1–4 аралығындағы бүтін сан) және нақты *A* мен *B* ($B \neq 0$) сандары берілген. Осы сандармен сәйкес амалды орындап, нәтижесін шығарыңыз.
36. Шеңбер элементтері келесі ретте нөмірленген: 1 – радиус (R), 2 – диаметр ($D = 2 \cdot R$), 3 – ұзындық ($L = 2 \cdot \pi \cdot R$), 4 – шеңбер ауданы ($S = \pi \cdot R^2$). Осы элементтердің бірінің нөмірі және оның мәні берілген. Шеңбердің қалған элементтерінің мәнін берілген реті бойынша шығарыңыз.

10 ЦИКЛ ОПЕРАТОРЛАРЫ

Цикл операторлары бірнеше рет қайталанатын есептеулерді орындау үшін қажет. Кез келген цикл сол цикл тұлғасынан (денесінен), яғни қайталанатын операторлар тізбегінен, бастапқы мәндер тағайындаудан, цикл *параметрлерін* өзгертуден және циклді қайталау шартын тексеруден тұрады (10.1 сурет). Циклдің бір рет орындалуы итерация (қадам) деп аталады. Шартты тексеру әрбір итерация сайын – цикл тұлғасына дейін (алғы шартты цикл) немесе цикл тұлғасынан соң (соңғы шартты цикл) атқарылады. Олардың айырмашылығы – соңғы шартты цикл, кем дегенде, бір рет орындалады да, содан кейін барып циклді қайталау шарты тексеріледі. Ал, алғы шартты циклде оны қайталау шарты цикл тұлғасынан бұрын тексеріледі, сондықтан кейде ол бір рет те орындалмай қалуы мүмкін.



10.1 сурет. Цикл операторларының құрылымдық схемасы:
а – алғы шартты цикл; ә – соңғы шартты цикл

Цикл тұлғасында мәні өзгертілетін айнымалылар цикл параметрлері болып табылады. Тұрақты қадаммен әрбір итерация сайын өзгертіліп отырылатын бүтін типтегі цикл парараметрлері *цикл санауыштары* деп аталады.

Бастапқы параметрлер мәндері айқын түрде берілмеуі де

мүмкін, олар циклге кіргенге дейін оның ішінде мәндері өзертілетін айнымалылардың алғашқы мәндерін беру үшін қажет.

Цикл оны қайталау шарты орындалмаған кезде аяқталады. Цикл қадамының немесе жалпы циклдің аяғына жетпей, доғарылатын кездері болады, олар **break, continue, return** және **goto** операторлары көмегімен іске асырылады. Сырттан цикл ішіне басқаруды беру болмауы тиіс.

C/C++ тілдерінде ыңғайлылығына қарай қолданылатын үш түрлі цикл операторлары бар, олар – **while, do while** және **for.**

10.1 Алғы шартты цикл (while – әзірше)

Орындалу саны алдын ала белгісіз болатын циклдер құру кезінде шарттары алдын ала немесе соңынан тексерілетін екі цикл түрі бар. Шарты алдын ала тексерілетін цикл операторының орындалу схемасы 10.1а-суретте көрсетілген. Оның жазылуы:

```
while (шарт-өрнек)  оператор;
```

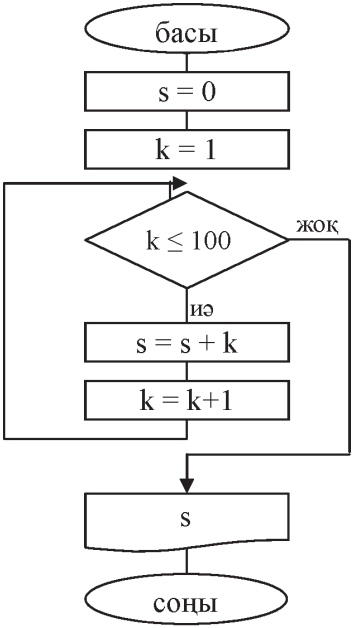
Мұнда шарт ретінде қатынас таңбалары кіретін шартты өрнек пайдаланылуы мүмкін. Өрнек типі арифметикалық немесе соған келтірілетін түрде болуы тиіс. Оператор қарапайым немесе құрама болуы мүмкін. Ол құрама оператор болса, онда операторлар жиыны жүйелі жақшаға алынып жазылады. **While** операторы орындалғанда, алдымен жақша ішіндегі өрнек есептеліп тексеріледі. Егер өрнек мәні ақиқат болса немесе жалпы жағдайда 0-ге тең болмаса, онда **оператор** атқарылады. Содан соң жақшадағы өрнек тағы да есептеледі. Егер өрнек мәні жалған болса (немесе жалпы жағдайда 0-ге тең болса), онда **while** цикл операторы өз жұмысын аяқтайды.

Мұнда шарт-өрнек құрамына кіретін айнымалы цикл ішінде өзгеріп отырады.

10.1 мысал (10.2 сурет).

```
/* 1-ден 100-ге дейінгі бүтін
сандар қосындысы */
#include <stdio.h>
#include <conio.h>
```

```
main ()
{
int s,k;
clrscr();
s=0; k=1;
while (k<=100)
 { s+=k;
  k++;
 }
printf("s= %d",s);
printf("\nАяқтау үшін Enter
басыңыз\n");
getch();
}
```



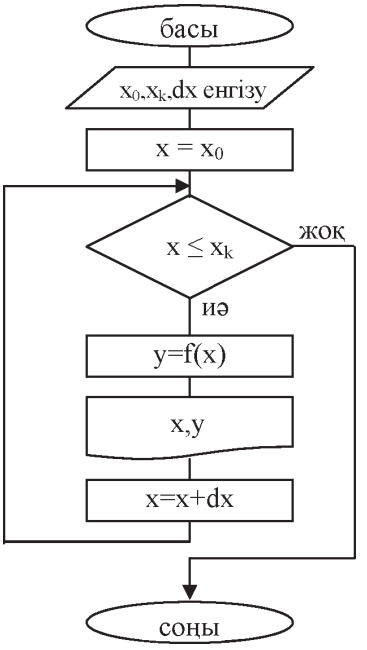
10.2-сурет. Бүтін сандарды қосу алгоритмі

10.2 мысал. $y=-2.4x^2+5x-3\sqrt{|x|}$ функциясы мәндерін оның аргументі x_0-ден x_k-ға дейін қадамы dx болып өзгерген кездерде анықтау керек. Мұнда программа С тілі стилінде жазылып, цикл алдында параметрге алғашқы мән меншіктеледі де, параметр цикл ішінде берілген қадамға өзгеріп отырады (10.3-сурет). Жалпы функция кез келген түрде беріле алады. Ол параметр мәніне байланысты тармақталып кететін функция да болуы мүмкін.

```
/* x тұрақты қадаммен x0-ден xk-ға дейін өзгергенде, функция мәндері кестесін алу,
x0,xk,dx (қадам) пернелерден енгізіледі */
#include <stdio.h>
#include <math.h>
#include <conio.h>
main () {
float x,y,x0,xk,dx;
clrscr();/* экранды тазалау */
printf("x-тің бастапқы,соңғы мәндері: ");
scanf("%f%f",&x0,&xk);
```

```
printf("x-тің өзгеру қадамы dx: ");
scanf("%f",&dx);
printf("----------------------\n");
printf("   x   |   y\n");
printf("----------------------\n");
x=x0;
while (x<=xk)
 {
y=-2.4*x*x+5*x-3*sqrt(fabs(x));
 printf("%6.2f  | %6.2f\n",x,y);
 x+=dx;
 }
printf("----------------------\n");
getch();
}
```

Келесі мысалда енгізілген **num** бүтін санының барлық бөлгіштері C++ тілі стилінде анықталады.



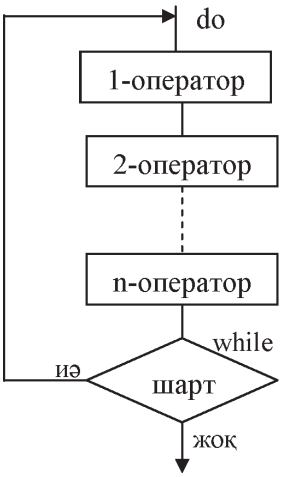
10.3-сурет. Функция мәндерін есептеу алгоритмі

10.3 мысал.

```
/* Бүтін оң санның бөл-
гіштерін анықтау */
#include <iostream.h>
int main(){
int num;
cout << "\nSan engizingiz: ";
cin >> num;
int half = num/2;
    // санның жартысы
int div = 2;
    // алғашқы бөлгіш
    //санды таңдау
while (div <= half) {
   if (!(num % div))
      cout << div <<"\n";
   div++;
 }
 return 0;
}
```

Программалауда жиі қолданылатын тәсілдердің бірі – шексіз цикл ұйымдастырып, одан белгілі бір шарт бойынша шығу болып табылады, ол үшін цикл басында **while (true)** немесе **while (1)** сөздері жазылады. **While** түйінді сөзінен кейінгі жақша ішіне циклде ғана қолданылатын айнымалыны сипаттауды жазып қоюға болады, мысалы:

```
while (int x = 0)
{ …
  /* x-тің пайдаланылу аймағы */
}
```



10.4-сурет. Шарты соңынан тексерілетін цикл

10.2 Соңғы шартты цикл (do .. while)

Шарты соңынан тексерілетін **do .. while (орындау .. әзірше)** циклінің орындалу схемасы 10.4-суретте көрсетілген. Осыған сәйкес операторлардың жалпы жазылу түрі:

```
do
  {
   1-оператор;
   2-оператор;
   … … …
   n-оператор;
  }
while (өрнек);
```

Цикл тұлғасы ретінде қарапайым немесе құрама оператор қолданылуы мүмкін. Жақшадағы өрнек цикл тұлғасынан кейін тексеріледі. Сондықтан **do while** цикл тұлғасы ең болмағанда бір рет орындалады. Цикл тұлғасынан кейін жазылған өрнек ақиқат болса (яғни ол 0-ге тең болмаса), цикл тұлғасы қайтадан орындалады. Ал өрнек жалған болса (немесе 0-ге тең болса), цикл аяқталады. Өрнек типі арифметикалық немесе соған келтірілетін типте болуы тиіс. Енді мысалдар келтірейік.

10.4 мысал.

```
// Енгізілген сандардың үлкенін (максимумын) табу
#include <stdio.h>
```

```
#include <conio.h>
main ()
{
int a, max;
clrscr();
printf("\n Сандар максимумын табу \n");
printf("Аяқтау үшін 0 енгізіңіз \n");
max = -32000;
// алдын ала максимумды ең кіші бүтінге теңейміз
do
 {
 printf("Сан енгізіңіз : ");
 scanf("%i",&a);
 if (a > max) max = a;
 }
 while (a!=0);
 printf("Сандардың максимумы: %i",m);
 getch();
}
```

10.5 мысал. Келесі программада шексіз сандар қосындысын

$$s = \sum_{i=1}^{\infty} \frac{1}{i^2} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \ldots + \frac{1}{n^2} + \ldots$$

алдын ала берілгін дәлдікпен $\varepsilon=10^{-5}$ анықтау керек, яғни келесі қосылатын қатар мүшесі осы ε санынан кіші болғанда, қосынды табу аяқталады.

```
#include <stdio.h>
#include <conio.h>
#define epsilon 1e-5
main ()
{
int i; float a,s;
clrscr();
s=0; i=1;
do
 {a=1.0/i*i;
```

```
  s+=a; i++;
 }
 while (a>epsilon);
 printf("s=%7.4f",s);
 getch();
}
```

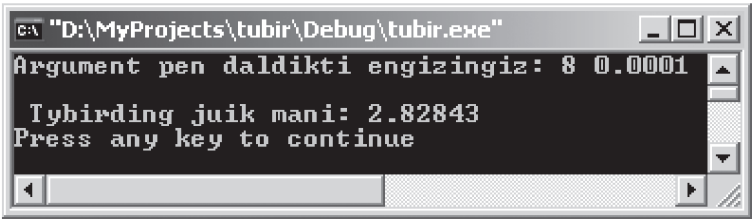
10.6 мысал. Төмендегі программада енгізілген бүтін санның тақ немесе жұп екендігі анықталады.

```
/* Санның жұп екендігін анықтау */
#include <stdio.h>
#include <conio.h>
main ()
{
int k;   /* енгізілетін сан */
char symbol;
textcolor(RED);
textbackground(WHITE);
clrscr();
printf("\n* Санның жұп/тақ екендігін анықтау *\n");
do
 {printf("\nБір бүтін сан енгізіңіз : ");
 scanf("%i",&k);
 printf("Бұл %i саны -",k);
 if (k % 2 == 0)
   printf("жұп сан.");
 else
   printf("тақ сан.");
 printf("\nТағы да енгісесіз бе? Иә-'Y',Жоқ-'N':");
 scanf("%s",&symbol);
 }
 while ((symbol=='Y')||(symbol=='y'));
}
```

10.7 мысал. Бұл программада енгізілген нақты аргумент – *x*-тің түбірін берілген дәлдікпен – **eps** жуық шамамен итерациялық формула арқылы анықтаймыз: $y_n = (y_{n-1} + x/y_{n-1})$,

мұндағы y_{n-1} – түбірдің алдыңғы жуық мәні (есептеу алдында бұл мән кез келген оң сан ретінде таңдалады), y_n – түбірдің келесі табылған жуық мәні. Есептеу процесі түбірдің анықталған екі жуық мәндері айырмасының абсолюттік мәні берілген дәлдіктен төмен болған сәтте тоқталады. Абсолюттік мәнді табу үшін стандартты **fabs()** функциясы қолданылады, ол **<math.h>** тақырыптық файлында анықталады.

```
#include <iostream.h> // C++ тілі стилінде
#include <math.h>
int main(){
double x, eps; //аргумент пен дәлдік
double Yp,Y=1; //түбірдің алдыңғы және келесі
жуық мәндері
cout << "Аргумент пен дәлдікті енгізіңіз: ";
cin >> x >> eps;
do{
```



10.5-сурет. 10.7 мысал нәтижесі

```
 Yp = Y;
 Y = (Yp + x/Yp)/2;
 }
while (fabs(Y - Yp) >= eps);
 cout << "\n Түбірдің жуық мәні: " << endl;
 return 0;
 }
```

10.3 For цикл операторы

For операторы айнымалы ретінде берілген цикл параметрінің алғашқы, соңғы мәні мен өзгеру қадамы белгілі болғанда, соған сәйкес бір немесе бірнеше операторларды қайталап орындау

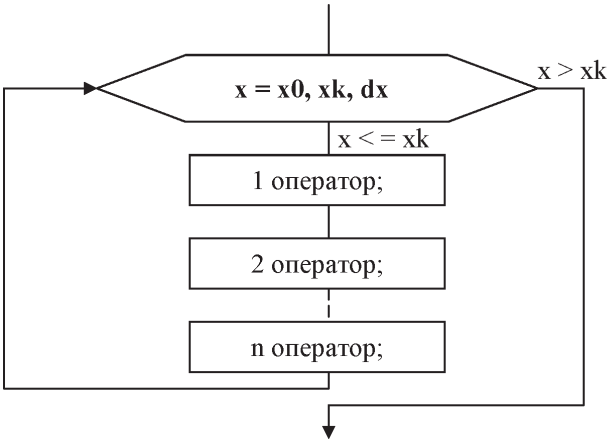
кезінде қолданылады. Бұл оператор параметрлі цикл операторы немесе арифметикалық цикл деп аталады.

Параметрлі циклдің орындалу схемасы:

For цикл операторының жалпы жазылу түрі:

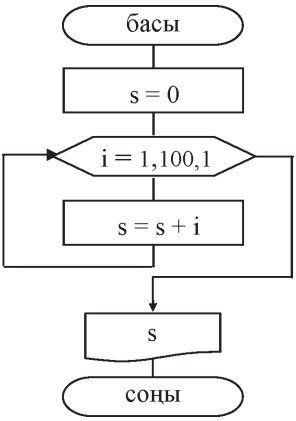
```
for (x=x0; x<=xk; x=x+dx)
  {
   <1-оператор>;
   <2-оператор>;
     . . .
   <n-оператор>;
  }
```

Мұнда **x=x0** – цикл айнымалысының бастапқы мәні, **x<=xk** – циклдің орындалу шарты, **x=x+dx** – цикл айнымалысының қадамы. **x=x0** цикл операторы орындаларда бір рет есептеледі,

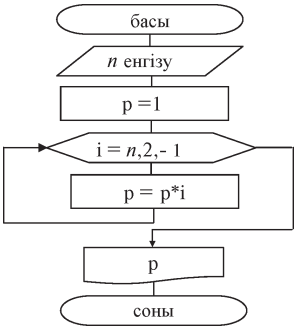


10.6-сурет. For циклінің орындалу алгоритмі

x<=xk ақиқат болса немесе 0-ге тең болмаса, цикл тұлғасы ретіндегі **операторлар** атқарылады. Содан соң **x=x+dx** есептеледі және **x<=xk** мәні қайта анықталады. **x<=xk** мәні жалған болса немесе жалпы жағдайда ол 0-ге тең болса, **for** операторының жұмысы аяқталады. Сонымен цикл тұлғасының келесі орындалуы немесе орындалмауы оның атқарылуы алдында анықталады.



10.7-сурет. Қосынды табу алгоритмі



10.8-сурет. Факториал табу алгоритмі

10.8 мысал (10.7 сурет).

```
/* 1-ден 100-ге дейінгі сандар
қосындысын анықтау */
#include <stdio.h>
#include <conio.h>
main ()
{ int s=0,i;
 clrscr();
 printf("1-ден 100-ге
 дейінгі сандар қосындысы:");
 for (i=1;i<=100;i++)
  s+=i;
 printf("s=%d",s);
 printf("\nАяқтау үшін Enter
      басыңыз\n");
 getch();
}
```

10.9 мысал. Бүтін сандардың көбейтіндісін өрнектейтін *n*! мәнін, яғни *n*!=1*2*... *n* табу қажет.

Бұл алгоритмді құру барысында **for** операторының кері қарай есептейтін мүмкіндігін пайдаланайық (10.8 сурет).

```
#include <stdio.h>
#include <conio.h>
main ()
{ int p=1,i;
 int n;
 clrscr();
 printf("n санын енгізіңіз де, Enter басыңыз:");
 scanf("%d",&n);
 printf("1-ден n-ге дейінгі сандар көбейтіндісі:");
 for (i=n;i>1;i--)
  p*=i;
 printf(" %d",p);
 getch();
}
```

10.10 мысал.

```
/* x айнымалысы берілген алғашқы
мәннен (x0) соңғы мәнге (xk) дейін тұрақты
қадаммен (dx) өзгеріп отырғанда, y функциясының
мәндерін анықтау */
#include <stdio.h>
#include <conio.h>
main ()
{ float x,y,x0,xk,dx;
clrscr();
printf("x-тің алғашқы,соңғы мәндері : ");
scanf("%f%f",&x0,&xk);
printf("x-тің өзгеру қадамы dx: ");
scanf("%f",&dx);
x=x0;
printf("----------------------\n");
printf("   x  |   y\n");
printf("----------------------\n");

for (x=x0;x<=xk;x+=dx)
{ y=-2.4*x*x+5*x-3;  /* функция */
  printf("%6.2f  | %6.2f\n",x,y);
}
 printf("----------------------\n");
 printf("\nАяқтау үшін Enter басыңыз");
 getch();
}
```

10.11 мысал. $y = \sum_{i=1}^{10} \frac{i^2}{2}$ қосындысын анықтау керек.

```
#include <stdio.h>
#define n 10
main ()
{
int i;
float s=0;
for(i=1,i<=n;i++)
```

```
    s+=i*i/2;
printf("нәтиже= %f\n",s);
}
```

For цикл операторындағы жақша ішіндегі соңғы өрнек ретінде жалпы дұрыс жазылған кез келген өрнекті пайдалануға болады.

Мысалы:

```
for (d=0.1; d<50; d*=5)
  printf("%f",d);
```

For цикл операторындағы жақша ішіндегі бір немесе бірнеше өрнектерді жазбауға да болады, бірақ мұндайда **;** символын міндетті түрде өз орындарына жазып отыру керек, мысалы:

```
x=2; for(n=4; x<=100;)
     x=x*n;
```

For цикл операторында құрама өрнектерді " , " операциясы арқылы жазуға да болады, " , " операциясы – құрама өрнекті ұйымдастыру үшін қолданылады. Осы операцияны қолданғанда, үтір арқылы бөлектенген өрнектер сол жақтан оң жаққа қарай есептеледі. " , " операциясы цикл операторының тиімді болуы үшін жиі пайдаланылады. Мысалы:

```
main ()
{
int x,y;
for (x=1,y=9;x<=10; x++,y--)
printf("%d%d\n", x,y);
}
```

Мұнда алғашқы **;** белгісіне дейін және соңғы өрнек арқылы осы цикл операторында екі параметр мәні беріліп (**x=1,y=9;**), олар **x,y** айнымалыларын өзгерту үшін қолданылып отыр.

Күрделі циклдер

Кейбір есептерді шығару үшін бірінің ішінде бірі жатқан *күрделі* деп аталатын қабаттаса орналасқан циклдерді пайдлануға тура келеді. Бұндай программаларды құрғанда ішкі цикл толығымен сыртқы циклдің ішінде орналасуы қажет. Ішкі циклдің

өзі де басқа ішкі циклдерді қамтуы мүмкін. Күрделі циклдің құрылымын төмендегі мысалдан көруге болады.

10.12 мысал. x = 1,2,3 және y = 2,4,6,8 болған кездерде $z = (2y+x)/\ln x$ функциясы мәндерін есептейтін программа құру керек.

```
/* z=(2y+x)/ln(xy) функциясы мәндерін
 x=1,2,3 және y=2,4,6,8 болғанда табу керек
*/
#include <stdio.h> // C тілі стилінде
#include <conio.h>
#include <math.h>
void lin() // сызық сызу функциясы
{ printf("--------------\n");}
main()   // басты функция
{ int x, y;
 float z;
 clrscr();
 printf(" x  y   z\n");
 lin();
 for(x=1;x<=3;x++)
  { y=2;
   while(y<=8)
   { z=(2*y+x)/log(x*y);
    printf("%2d%4d%8.2f\n",x,y,z);
    y+=2;
   }
   lin();
  }
 getch();
}
```

Программа нәтижесі:

```
 x y   z
-------------
 1 2 7.21
 1 4 6.49
 1 6 7.26
 1 8 8.18
-------------
```

```
 2  2  4.33
 2  4  4.81
 2  6  5.63
 2  8  6.49
-------------
 3  2  3.91
 3  4  4.43
 3  6  5.19
 3  8  5.98
-------------
```

10.13 мысал. Көбейту кестесін шығаратын программа құру керек.

```
// Толық көбейту кестесі
#include <iostream.h> // C++ тілі стилінде
void lin(int n)      // n сызықша сызу функциясы
{ for(int k=1; k<=n;k++)
  cout << '-';
 cout << "\n";
}
int main()            // басты функция
{ cout << "\t\t\tКөбейту кестесі << endl
       << "  1 \t2 \t3 \t4 \t5 \t6 \t7 \t8 \t9"
       << endl;
 lin(66);             // көлденең 66 сызықша
 for (int i=1; i < 10; i++)
  { cout << i << "| ";
   for (int j = 1; j < 10; j++)
     cout << j * i << '\t';
   cout << endl;
  }
 return 0;
}
```

10.4 Программаның орындалу тәртібін өзгерту операторлары

С++ тілінде программа жолдарының, яғни есептеу процесінің табиғи реттілігін өзгертетін төрт оператор бар:

- шартсыз көшу операторы – **goto**;
- циклден шығу операторы – **break**;
- циклдің келесі қадамына көшу операторы – **continue**;
- функциядан кері оралу, яғни қайту операторы – **return**.

Шартсыз көшу – goto операторы

Программаның орындалу реттілігін ешбір шартсыз өзгертетін **goto** операторының форматы;

```
goto <белгі>;
```

Программа ішінде осы белгі көрсетілген төмендегідей бір жол болуы тиіс:

```
белгі: оператор;
```

Бұл оператор белгіленген жолға көшу әрекетін орындайды. *Белгі – осы функция (программа) ішінде пайдаланылатын қарапайым идентификатор*. Жалпы дұрыс (құрылымдық) программалау ережесі бойынша бұл операторды қолданбауға тырысу керек. Ол программаны оқып түсіну ісін қиындатады. Бұл операторды құрылымдық тәртіппен құрылған программа ішінде пайдалануға болмайды. Сондықтан оны барынша аз қолдануға тырысып, мыналарды есте сақтаған жөн:

- оператор пайдаланылған жағдайда (онсыз программа құру мүмкіндігі жоқта ғана), оны программа мәтіні бойынша тек төмен қарай көшу үшін пайдаланған жөн; бұл операторды кері қайту үшін қолдану қажет болса, шартты оператормен алмастыруға тырысу керек;
- белгі тұрған орын мен көшу операторының арасы мәтін бойынша бір беттен (немесе дисплей экраны биіктігінен) аспауы тиіс.

Goto операторын қолданған шақта оның қатарына жүйелі жақша ішіне неліктен басқа қатарға көшу қажет екендігін түсіндіріп кету керек немесе оның орнына шартты көшу операторын пайдаланған абзал.

Бұл операторды цикл ішіне және **if, switch** операторларының ішіне көшу үшін қолданбау қажет.

```
int k = -4; …
goto belgi; …
{int a = 3, b = 4;
 k = a+ b;
 belgi: int m = k + 1; …
}
```

Break операторы

Циклдің (немесе **switch**) қайталану саны аяқталмай-ақ, оның ішінен сыртқа (одан кейін орналасқан жолға) шығу үшін **break** операторы қолданылады.

10.14 мысал. Нақты *x* аргументінің гиперболалық синусын оны шексіз қатарға жіктейтін төмендегі формула арқылы есептейтін программа құрайық.

$$sh\ x = 1 + x^3/3! + x^5/5! + x^7/7! + \ldots$$

мұндағы есептеу қатардың келесі мүшесінің абсолюттік мәні берілген дәлдіктен - ε кіші болған сәтте тоқталады.

```
#include <iostream.h> // C++ тілі стилінде
#include <math.h>
int main(){
const int MaxIter = 500; // итерация санын шектеу
double x, eps;
cout << "\nАргумент пен дәлдікті енгізіңіз: ";
cin >> x >> eps;
bool flag = true;      // дұрыс есептеу белгісі
double y = x, ch = x; // қосынды және қатардың
                      // 1-мүшесі
 for (int n = 0; fabs(ch) > eps; n++) {
   ch *= x*x/(2*n+2)/(2*n+3); // қатардың
                              // келесі мүшесі
    y += ch;
    if (n > MaxIter){
    cout << "\nҚосынды шексіздікке ұмтылады!";
    flag = false; break;}
 }
 if (flag) cout << "\nФункция мәні: " << y;
 return 0;
}
```

Continue операторы

Цикл орындалуы кезінде оның атқарылып жатқан қадамы аяқталмай-ақ, қалған операторларды аттап өте отырып одан шығып, цикл параметрінің келесі мәніне көшу үшін *continue* операторы қолданылады, яғни циклдің келесі итерациясына – қадамына басынан бастап ауысу жүзеге асырылады.

10.15 мысал. Программаға 10 бүтін сан енгізіп, солардың ішіндегі тақ сандар қосындысын анықтау керек.

```
/* енгізілетін 10 бүтін сандар ішіндегі тақ
   сандар қосындысын табу */
#include <iostream.h> // C++ тілі стилінде
#include <conio.h>
main()
{
 int num, sum=0;
 clrscr();
 for(int i=1;i<=10;i++)
  { cout <<"Бүтін сан енгізіңіз: "; cin >> num;
   if (num % 2 == 0) continue;
   sum+=num;
  }
 cout << "\nЕнгізілген тақ сандар қосындысы: "
      << sum;
 getch();
}
```

Бақылау сұрақтары

1. *Алғы шартты цикл операторының орындалу схемасы мен оның жазылуы.*
2. *while цикл операторының ішкі тұлғасы бір де бір рет орындалмауы мүмкін бе?*
3. *while цикл операторының шарты қатынас таңбаларысыз жазыла ма?*
4. *while цикл операторы қай кезде шексіз циклге айналады?*
5. *while цикл операторының тұлғасында оның шартына әсер ететін өрнектер жазыла ма?*
6. *Соңғы шартты do ... while циклінің орындалу схемасы мен жазылуы.*

7. *do ... while цикл операторының ішкі тұлғасы бір де бір рет орындалмауы мүмкін бе?*
8. *do ... while цикл операторының шартында қатынас таңбасы болмаса, оның ақиқат немесе жалған екенін қалай анықтауға болады?*
9. *do ... while цикл операторының ішкі тұлғасында шартсыз көшу операторын қолдануға бола ма?*
10. *Параметрлі циклдің орындалу схемасы мен жазылуы.*
11. *Параметрлі циклде "модификатор" блогын пайдалану ерекшеліктері.*
12. *Параметрлі цикл бір де бір рет орындалмауы мүмкін бе?*
13. *for операторының параметрі қандай типтерде бола алады?*
14. *for операторы параметрінің алғашқы мәні оның соңғы мәнінен кіші бола ма?*
15. *Параметрлі цикл операторының неше рет қайталанатынын алдын ала білуге бола ма, болса – ол қалай анықталады?*
16. *for цикл операторындағы жақша ішіндегі бір немесе бірнеше өрнектерді жазбауға бола ма?*
17. *for цикл операторы нүктелі үтірмен аяқтала ала ма?*
18. *for цикл операторында қай кезде құрама операторлар қолданылады?*
19. *for цикл операторы қай кезде шексіз циклге айналады?*
20. *for цикл операторында бірнеше құрама өрнектерді үтір арқылы қалай жазуға болады?*
21. *Қабаттаса орналасқан циклдерді пайдаланудың ерекшеліктері.*
22. goto *операторы қай кездерде қолданылады?*
23. *Белгі дегеніміз не?*
24. break *операторы цикл соңына көшіре ала ма? Неге?*
25. *Циклдің келесі қадамына көшу операторының жұмысы. Ол қай кездерде пайдаланылады?*

Тапсырмалар

1. 1-ден *N*-ге дейінгі сандардың қосындысын есептейтін программа құрыңдар. *N*-нің мәні пернетақтадан енгізіледі.
2. 1-ден *N*-ге дейінгі сандардың көбейтіндісін есептейтін программа құрыңдар. *N*-нің мәні пернетақтадан енгізіледі.
3. Пернетақтадан *N* сан енгізіледі. Енгізілген сандардың ішіндегі теріс, оң сандардың және нөлдердің санын анықтайтын программа құрыңдар.
4. Ұзындықтың 1-ден 20 дюймге дейінгі мәндерін сантиметрге (1 дюйм = 2,54 см) айналдыратын және оны экранға шығаратын программа құрыңдар.

5. Банктегі жылдық өсімі 9 пайыздық (проценттік) салымға S теңге салынды. N жылдан кейін салынған ақша неше теңгеге жетеді?
6. Пернетақтадан 10 бүтін сан енгізіп, солардың квадраты мен кубын анықтаңыз.
7. 20-дан 50-ге дейінгі натурал сандар берілген. Олардың ішіндегі 3-ке бөлінетін, бірақ 5-ке бөлінбейтін сандарды анықтаңдар.
8. 35-тен 87-ге дейінгі натурал сандар берілген. Олардың ішінде 7-ге бөлгенде, 1-ге, 2-ге немесе 5-ке тең қалдық қалатын сандарды табыңдар.
9. 1-ден 50-ге дейінгі натурал сандар берілген. Олардың ішіндегі 5-ке немесе 7-ге бөлінетін сандардың қосындысын табыңдар.
10. Пернетақтадан 10 сан енгізіңдер. Егер олардың ішінде 15-тен асқаны бар болса, онда оларды 15-пен алмастырыңдар. Сандарды экранға шығарыңдар.
11. Пернетақтадан он теріс және оң сан енгізіңдер. Барлық теріс сандарды олардың модульдерімен алмастырып сандарды экранға басып шығарыңдар.
12. Екі орынды сандардың ішіндегі 4-ке бөлінетінін, бірақ 6-ға бөлінбейтінін табыңдар.
13. 13-ке қалдықсыз бөлінетін екі орынды тақ сандардың көбейтіндісін табыңдар.
14. 100-ден 200-ге дейінгі сандардың ішіндегі 17-ге қалдықсыз бөлінетін сандардың қосындысын табыңдар.
15. Пернетақтадан 10 сан енгізіңдер. Егер сан 100-ден кем болса, онда осы санды және оның квадратын табыңдар.
16. 1-ден бастап өздерің енгізген бүтін n санына дейінгі сандардың квадраттарының қосындысын есептейтін программа құрыңдар.
17. 200-ге дейінгі 5-ке бөлгенде 4 қалдық қалатын сандар нешеу екенін табу керек.
18. 200-ге дейінгі бүтін сандардың 25-ке қалдықсыз бөлінетін сандары нешеу екенін анықтаңдар.
19. 20-дан үлкен және 100-ден кіші 3-ке қалдықсыз бөлінетін оң сандардың қосындысын табу керек.
20. Берілген a нақты және b бүтін сандарының мәндері бойынша a^b өрнегінің мәнін pow() функциясын пайдаланбай табатын программа жазыңдар.
21. Төмендегі есептерде бір өлшем бірлігін екінші өлшем бірлігіне түрлендіру қажет (цикл санауышы мәні 1-ден 20-ға дейін өзгереді):
    а) футпен берілген ұзындық өлшемін метрге (1 фут = 0,3048 м);
    ә) драхмды граммға (1 драхм = 3,7325 г;
    б) унцияны граммға (1 унция = 29,86 г);
    в) фунтты килограммға (1 фунт = 0,40951 кг);

г) аршынды метрге (1 аршын = 0,7112 м);
д) қадақты граммға (1 қадақ = 400 г);
е) қарысты сантиметрге (1 қарыс = 18 см);
ж) дюймді миллиметрге (1 дюйм = 25,3995 мм).

22. Төмендегі функциялар мәндерін x айнымалысы x0-ден xk-ға дейін dx қадамымен өзгерген кезде анықтаңдар.

а) $z=\begin{cases} 5\sin x+\cos x, & егер\ \ x<1 \\ \dfrac{x^2-2x-5}{e^x}, & егер\ \ 1\le x<4 \\ \sqrt{x^2+5}+lg\,x, & егер\ \ x\ge 4 \end{cases}$ мұндағы x_0=-1, x_k=5, dx =0.5

ә) $y=\begin{cases} \sin^2 2x+5x^2, & егер\ \ x>1,66 \\ \dfrac{61\sqrt{x}-17}{\sqrt{4+x^2+\cos^2 4x}}, & егер\ \ x\le 1,66 \end{cases}$ мұндағы x_0=0.5, x_k=3, dx =0.5

б) $y=\begin{cases} \sqrt{x}+\sin x, & егер\ \ x>0,4 \\ \sqrt{2-x}, & егер\ \ -1<x\le 0,4 \\ e^{5x-1}+lg\left|x\right|, & егер\ \ x\le -1 \end{cases}$ мұндағы x_0=-2, x_k=1, dx= 0.25

23. Төмендегі жалпы мүшесі берілген қатардың 10 мүшесінің қосындысын табыңдар:

а) $a_n=e^{-\sqrt{n}}$ ә) $a_n=n^3e^{-n}$ б) $a_n=\dfrac{2^n n!}{n^n}$

24. Жалпы мүшесі төменгі өрнекке сәйкес қатар қосындысын $\varepsilon=10^{-4}$ дәлдігімен анықтау керек.

а) $a_n=\dfrac{(-1)^{n-1}}{n^n}$ ә) $a_n=\dfrac{1}{2^n}+\dfrac{1}{3^n}$ б) $a_n=\dfrac{1}{((3n-2)(3n+1))}$

в) $a_n=\dfrac{(2n-1)}{2^n}$ г) $a_n=\dfrac{10^n}{n!}$ д) $a_n=\dfrac{n!}{(2n)!}$

11 ЖИЫМДАР ЖӘНЕ НҰСҚАУЫШТАРДЫ ПАЙДАЛАНУ

С/С++ 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 программаны компиляциядан өткізу кезінде атқарылады.

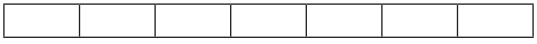
11.1 С/С++ тілдерінде жиымды анықтау

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 жиымы
```

мұнда **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, мысалы, **a[20],** ал индексті **a[n]** деп жазу үшін алдын ала

```
#define n 20;
```

жолы көрсетіледі немесе

```
const n=20;
```

болып жазылады.

11.2 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, a_0, a_1, a_2, ... , a_9 және c_0, c_1, c_2, ... , c_9 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];
```

11.3 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];
```

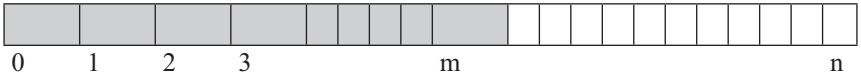
2) программалаушы жұмыс барысында жиым элементтері санын n-нен аз етіп алады.

```
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.

11.4 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 орындалады.

11.1 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 аралығын-
    дағы оң және теріс сандар алу */

 printf(" %i ", a[I]);}    // жиым элементтерін
                           // экранға шығару
 getch();
}
```

11.5 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 жекежеке 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.

11.5.1 1-түрдегі есептер

11.2 мысал. Бүтін сандардан тұратын a_{10} жиымының ең үлкен элементін – максимумын және оның индексін анықтау керек.

```
/* Жиым максимумын табу */
#include <conio.h>
#include <stdio.h>
#define n 10
main()
{ int i,t,a[n]={6,5,9,8,7,4,1,2,3,0},max;
textcolor(BLUE);           // мәтін түсі көк
textbackground(YELLOW);    // экран түсі сары
clrscr();
printf("a[10] элементтері : ");

for (i=0; i<n; i++)
 printf(" %d ",a[i]);
max=a[0]; t=0; // max - максимум,
               // t - оның индексі
for (i=1; i<n; i++)
 if (a[i] > max)
    {max = a[i]; t=i;}
printf("\nmax = %d, индексі = %d\n", max, t);
getch();
}
```

11.3 мысал. Жиымның жұп индексті элементтері қосындысын анықтау.

```
/* 0, 2, 4... индексті элементтер қосындысын табу */
#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;
 printf(" %i ", a[I]); }
int Sum=0;
for(I=0;I<n;I+=2)
Sum+=a[I];
printf("\nSum= %i ", Sum);
getch();
 }
```

Соңғы циклді басқаша да құрастыруға болады:

```
// Екінші тәсіл
for(I=0;I<n;I++)
if(I%2==0) Sum+=a[I];
printf("\nSum= %i ", Sum);
```

11.5.2 2-түрдегі есептер

Жиым ішіндегі екі элементтің бір-бірімен орнын ауыстыру үшін қосымша тағы бір айнымалы керек болады. Мысалы, a[I] және a[J] элементтерінің орнын ауыстыру үшін қосымша R айнымалысы керек:

```
R=a[I]; a[I]=a[J]; a[J]=R;
```

Жиым элементтерін кері бағытта орналастыру былай орындалады:

```
for(int i=0, j=n-1; i<j; i++, j--)
{int r=a[i];
 a[i]=a[j];
 a[j]=r;}
```

11.4 мысал. Жиымның қатар тұрған екі элементін: 1 және 2, 3 және 4, 5 және 6, т.с.с. элементтерін бір-бірімен орын ауыстыру есебі:

```
for(int i=0;i<n-1;i+=2)
{int r=a[i]; a[i]=a[i+1]; a[i+1]=r;}
```

11.5 мысал. Жиым элементтерін **k** орынға солға (оңға) ығыстыру, яғни жылжыту.

```
int k,i,t,r,n;
printf("k = ");
skanf("%d",&k);
for(t=0;t<k;t++)
{    r=a[0];
     for(int i=0; i<n-1; i++)
          a[i]=a[i+1];
     a[n-1]=r;
}
```

11.5.3 3-класс есептері

Жиымдарды синхронды түрде өңдеуде жиымдар элементін қарастыру кезінде индекстер бірдей қадамға өзгереді. Мысалы, бүтін сандардан құралған n элементтерден тұратын 2 жиым берілген делік. Жаңа **c** жиымы мынадай формула арқылы алынады: **c[I]=a[I]+b[I].**

```
for (int I=0; I<n; I++)  c[I]=a[I]+b[I];
```

Жиымдарды асинхрондық өңдеу кезінде әр жиым индексі өз реттілігімен өзгеріп отырады.

11.6 мысал. Бүтін сандардан құралған жиымдағы теріс элементтердің барлығын оның бас жағына орналастыру керек.

```
int b[10];  //қосымша жиым
int i,j=0;
for(i=0;i<n;i++)
  if(a[i]<0){b[j]=a[i];j++;}
// a-дан b-ға теріс элементтерді көшіріп жазу
for(i=0;i<n;i++)
  if(a[i]>=0){b[j]=a[i];j++;}
```

```
   // a-дан b-ға оң элементтерді көшіріп жазу
   for(i=0;i<n;i++) printf (" %d ", b[I]);
```

11.7 мысал. Жиымның барлық жұп элементтерін жою керек.

```
int b[10];
int i,j=0;
for(i=0;i<n;i++)
 if(a[i]%2!=0){b[j]=a[i];j++;}
for(i=0;i<j;i++) printf (" %d ", b[I]);
printf ("\n");
```

11.5.4 4-класс есептері

Іздеу есептерінде берілген шартқа сәйкес келетін элементті іздеп табу керек. Ол үшін жиым элементтерін біртіндеп тізбектей қарастырып отырып шартты тексеріп шығу қажет. Осылай ету барысында циклден шығудың екі жолы бар:

- керекті элемент табылғаннан кейін;
- жиым элементтері тегіс қаралып шықты, керекті элемент табылмады.

11.8 мысал. Берілген k санына тең жиымның алғашқы элементін табу.

```
int k;
printf("\nK=");
scanf("%i",&k);
int ok=0;//элемент табылғаны
         //табылмағаны белгісі
int i,nom;
for(i=0;i<n;i++)
     if(a[i]==k){ok=1;nom=i;break;}
if(ok==1) printf("\nnom=",nom);
 else printf("\nk-ға тең элемент жоқ!");
```

11.6 Жиымды сұрыптау (іріктеу, реттеу)

Сұрыптау – берілген объектілер жиынын (сандарды) ұсынылған реттілікпен қайта теріп орналастыру процесі.

Жиымдарды сұрыптау жылдамдығы әр түрлі болады. Қарапайым сұрыптау тәсілдері **n*n** рет салыстыруды керек етеді, мұндағы **n** – жиым элементтері саны; ал жылдам сұрыптау

тәсілі **n*ln(n)** рет салыстыруды қажет етеді. Қарапайым тәсілдер түсінуге жеңіл, өйткені алгоритмі түсінікті. Күрделі тәсілдер аз әрекеттер санын керек еткенмен, операциялары күрделірек болады, сондықтан элементтер саны аз жиымдарға қарапайым тәсілдерді қолданған дұрыс.

Қарапайым тәсілдер 3 топқа бөлінеді:

- жай таңдау жолымен сұрыптау;
- жай енгізу тәсілімен сұрыптау;
- жай алмастыру тәсілімен сұрыптау.

11.6.1 Жай таңдау жолымен сұрыптау

Жиымның ең кіші элементі анықталады да, ол бірінші элементпен орын ауыстырады. Қалған элементтермен де осы тәсіл қайталанады.

44	55	12	42	94	18

минимум

```
int i,min,n_min,j;
for(i=0;i<n-1;i++)
{ min=a[i];n_min=i; // минимумды іздеу
 for(j=i+1;j<n;j++)
 if(a[j]<min)
  { min=a[j];n_min=j; }
 a[n_min]=a[i];      //алмастыру
 a[i]=min;}
```

11.6.2 Жай енгізу (кірістіру) тәсілімен сұрыптау

Жиым элементтері екіге – бастапқы тізбекке және дайын тізбекке бөлінеді. Әрбір адымда **I=2** нөмірінен бастап, бастапқы берілген тізбектен **I-**ші элемент алынады да, ол дайын тізбектің керекті жеріне орналастырылады. Мұнан кейін **I-ге 1** қосылады да, сол әрекеттер қайталанады.

44	55	12	42	94	18
дайын тізбек	бастапқы тізбек				

Керекті орынды іздеу кезінде оң жақтағы келесі элементпен орын ауыстыру қарастырылады, яғни таңдалып алынған эле-

мент сұрыпталғандардың **J=I-1** нөмірінен басталатын кезекті элементімен салыстырылады. Егер таңдалып алынған элемент **a[I]**-ден артық болса, онда ол сұрыпталғандар ішіне қосылады, әйтпесе **a[J]** бір орынға ығысады да, таңдалған элемент сұрыпталғандар ішіндегі келесі элементпен салыстырылады. Керекті орынды іздеу әрекеті екі жағдайда:

- егер **a[J]>a[I]** болатын элемент табылса;
- дайын тізбектің сол жақ шетіне жеткен кезде аяқталады.

Мысалы:

```
int i,j,x;
for(i=1;i<n;i++)
 { x=a[i];//ауысатын элементті есте сақтау
  j=i-1;
  while(x<a[j]&&j>=0) //керекті орынды іздеу
  { a[j+1]=a[j]$       //оңға жылжыту
      j--;
  }
  a[j+1]=x;          //элементті кірістіріп қою
 }
```

11.6.3 Жай алмастыру арқылы сұрыптау

Мұнда ең соңғыдан бастап, екі элемент салыстырылады да, қажет болса, орындары алмастырылады. Осындай әрекет нәтижесінде ең кіші элемент жиымның ең сол жақ шетіне ығысады. Қалған жиым элементтері үшін де осы процесс қайталанады.

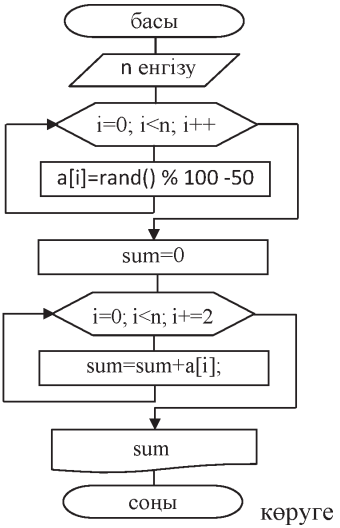
44	55	12	42	94	18

```
for(int i=1;i<n;i++)
for(int j=n-1;j>=i;j--)
  if(a[j]<a[j-1])
   {int r=a[j];a[j]=a[j-1];a[j-1]=r;}
```

11.9 мысал. Бүтін оң және теріс сандардан тұратын a[n] жиымының жұп нөмірлі элементтерінің қосындысын табу керек, мұнда **n** саны енгізіледі, ал жиым элементтерінің мәндері кездейсоқ бүтін сандардан тұрады.

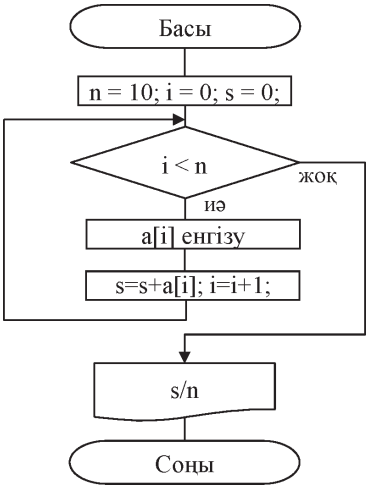
rand() функциясы 0...32767 аралығындағы бүтін сан береді.

Оны пайдалану үшін **stdlib.h** директивасын, яғни тақырып файлын қолдану қажет. Жиым элементтері екі разрядты оң және теріс сандардан тұруы үшін алынған кездейсоқ сан 100-ге бөлініп, қалдығынан 50 алып тасталынған (11.1-сурет).



11.1-сурет. Жиымның жұп элементтерін қосу

```
/*жиымның жұп элементтері
қосындысы*/
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
main()
{
int a[50];
int n;
printf("\nЖиымда неше элемент бар?");
scanf("%d",&n);
printf("Жиым элементтері:\n");
for(int i=0;i<n;i++)
  {
  a[i]=rand()%100-50;
 /* жиымға 0 - 50 аралығындағы кездейсоқ
     сандарды меншіктеу */
 printf("%d, ",a[i]);
 // сандарды экранда бейнелеу
 }
int sum=0;
for(i=0;i<n;i+=2)
sum+=a[i];
// 0, 2, 4... индексті элементтерді қосу
printf("\nЖиымның жұп элементтері
    қосындысы: %d",sum);
getch(); // нәтижелік экранды жапқызбай, көруге мүмкіндік беру
}
```



11.2-сурет. Жиымның арифметикалық ортасын анықтау

11.10 мысал. Бүтін сандардан құралған A(10) бір өлшемді жиымы берілген. Сол жиым элементтерінің арифметикалық ортасын табу керек (11.2 сурет).

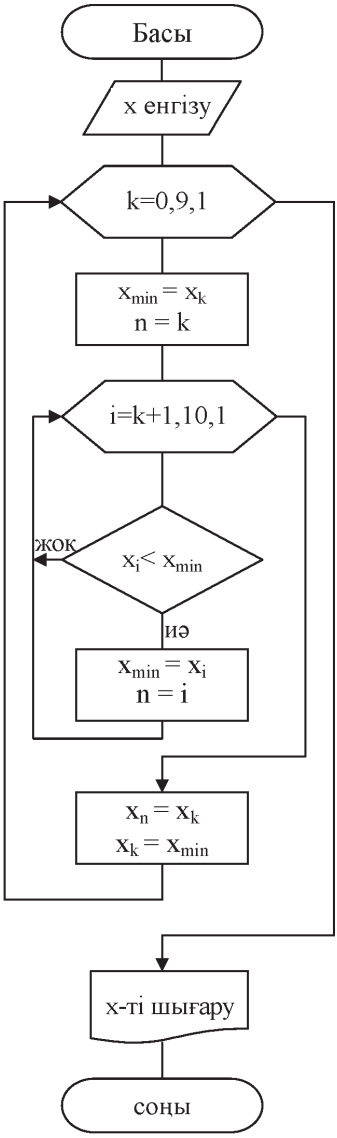
```
/* A[10] жиымының арифмети-
калық ортасын табу */
#include <conio.h>
#include <stdio.h>
#define n 10
main ()
{int i=0,s=0;
 int a[n];
 textcolor(RED);
// экран символдары кызыл түсті
 textbackground(GREEN);
// экран фоны жасыл түсті
 clrscr();
 printf ("Жиым элементтерін
  - 10 сан енгізіңіз:\n");
 while (i<n)
{
 printf("a[%i]=",i);
 scanf("%i",&a[i]);
 s=s+a[i];
 i=i+1;
}

printf("Жиым арифметикалық ортасы : %5.2f",
         (float)s/n);
printf("\nАяқтау үшін Enter басыңыз");
getch();
}
```

11.11 мысал. Нақты сандардан тұратын A[15] жиымы берілген. Жиымның оң элементтерінің геометриялық ортасын анықтау керек.

N санның геометриялық ортасы мынадай өрнекпен анықталады:

$$g = \sqrt[n]{a_0 \cdot a_1 \cdot \ldots \cdot a_n}$$

```
/* Нақты сандардан тұратын A[15] жиымының оң
элементтерінің геометриялық ортасын табу керек
*/
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#define n 15
main ()
{int i;
 float a[n];
 randomize(); // кездейсоқ сандарды өзгертіп
              // отыру функциясы
 textcolor(MAGENTA); textbackground(WHITE);
 clrscr();
 printf("\nЖиым элементтері:\n");
 for(i=0;i<n;i++)
 {a[i] = (float)(rand() % 100 - 50)/10;
  printf(" %5.1f",a[i]);}
          // көбейтінді мен оң элементтердің
          // санын табу
 i=0;     // бастапқы индекс = 0
 float p=1; // оң элементтер көбейтіндісі
 int k=0;  // оң элементтер саны
 do
 {
  if (a[i]>0) { k++; p*=a[i];}
  i++;
 }
 while (i<n);
 printf("\nоң элементтер саны : %d\n",k);
 p=pow(p,1.0/k);
 printf("геометриялық орта = %f",p);
 getch();
}
```



11.3-сурет. Жиым элементтерін өсуі бойынша реттеу

11.12 мысал. Берілген жиым элементтерін – x_{10} сол жиымда өсу реті бойынша орналастыру керек (11.3 сурет).

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
main()
{ int xmin,x[10];
 int n,k,i;
 clrscr();
  printf("\nБерілген жиым
элементтері:");
 for (k=0; k<10; k++)
 { x[k] = rand() % 100;
  /* 32767-ге дейінгі
   кездейсоқ сандар */
  printf(" %i",x[k]);
  }
 for (k=0; k<9; k++)
 { xmin=x[k];n=k;
  for (i=k+1; k<10; k++)
   if (x[i] < xmin)
        { xmin=x[i];
       n = i;
      }
  x[n]=x[k]; x[k]=xmin;
 }
 printf("\nРеттелген жиым
      элементтері:");
 for (k=0; k<10; k++)
  printf(" %i",x[k]);
 getch();
}
```

11.7 Адрестік операциялар

Адрестік операциялар үшін C/C++ тілінде екі арнайы оператор қолданылады.

& – адресті анықтау үшін қолданылатын операция;

* – адрес арқылы қатынас жасау үшін қолданылатын операция.

& операциясы берілген айнымалының адресін қайтарады. Мысалы, программа мәтінінде **sum** айнымалысы былай сипатталған болсын:

```
int sum;
```

онда **&sum** жазуы осы айнымалының компьютер жадындағы орналасқан ұясының адресі болып табылады.

Нұсқауыштар. Нұсқауыш мәліметтердің адресін сақтайтын айнымалы болып табылады. Жалпы алғанда, нұсқауыш адрестің символдық кескінделуі болып саналады. Қарастырылатын мысалда **&sum** – **sum** атты айнымалыға сілтейтін нұсқауыш болып табылады. Нақты адрес ретінде белгілі бір сан тұрады, ал **&sum** нұсқауыш типті константа болып табылады.

C/C++ тілінде нұсқауыш типті айнымалылар да бар. Нұсқауыш типті айнымалылар мәні болып белгілі бір шаманың адресі саналады.

Мысалы, нұсқауыш типті айнымалы **ptr** идентификаторы арқылы белгіленген болсын, онда төмендегі оператор **sum** айнымалысының адресін **ptr** атты нұсқауыш типті айнымалыға меншіктейді. **ptr** атты нұсқауыш типті айнымалы басқа да объектіге сілтеуі мүмкін.

Мысалы:

```
int *ptr;
ptr=&sum; ptr=&max;
```

* операциясы – адрес арқылы қатынас жасау үшін пайдаланылатын операция. Мысалы, **ptr** нұсқауыш типті айнымалысында **max** айнымалысына нұсқайтын сілтеме сақталған болсын. Осы айнымалының мәнін білу үшін * адресі бойынша қатынас жасау операциясын қолдануға болады. **ptr** нұсқауышы мәнін анықтау үшін келесі операцияны орындау қажет: **res=*ptr;**

Нұсқауышты сипаттау. Нұсқауыш типті айнымалыны сипаттағанда берілген нұсқауыш қандай типті айнымалыға сілтейтінін

көрсету қажет. Өйткені әр түрлі типті айнымалыға компьютер жадында ұялардың әр түрлі саны бөлініп беріледі.

```
int  *iptr;
char *cptr;
float *fptr;
```

Функциялар арасында байланыс жасау үшін нұсқауыштарды пайдалану

11.13 мысал. Бұл программада айнымалының мәндерін ауыстыру үшін нұсқауыштар пайдаланылған:

```
#include <stdio.h>
#include <conio.h>
void change (int *u, int *v)
 {int temp; temp = *u; *u = *v; *v = temp; }
main ()
{ int x=5, y=10;
 clrscr();
 printf ("x=%d y=%d\n",x,y);
 change (&x,&y);
 printf ("x=%d y=%d\n",x,y);
 getch();
}
```

Нәтижесі:

```
x=5 y=10
x=10 y=5
```

11.8 Жиымдар және жиымдарға қолданылатын нұсқауыштар

Жиымдарды сипаттағанда, элементтер типі және жалпы жағайда компьютер жадының қажетті класы көрсетіледі. Қарапайым айнымалыларда қарастырылатын қасиеттер жиымдарды да сипаттау кезінде қолданылуы мүмкін. Мысалы:

```
int b[30];
main()
{ float a[30];
static char c[20];
extern b[];
......
}
```

Жиымдарды инициалдауды қарастырайық. Жиым сипаттауында тек сыртқы немесе статистикалық жиымдар ғана инициалдануы мүмкін. Мысалы, 8 топтағы студенттер саны **stud**[8] жиымы ретінде көрсетілген:

```
#include <conio.h>
#include <stdio.h>
int stud[8]={15,16,14,18,15,20,17,19};
main()
{ int i;
clrscr();
extern int stud[];
for (i=0; i<8; i++)
printf("N %i тобында %i студент\n",
    i+1,stud[i]);
}
```

Жиымның нұсқауыштарын қарастырайық. Мысалы:

```
int *a;
```

Жиым аты нұсқауышты қолданған жағдайда, жиымның 0-ші элементін анықтайды, яғни **a** жиымы сипатталған болса, программа мәтініндегі **a** идентификаторы 0-ші элементті көрсетеді деп саналады:

```
a == &a[0];
```

бұл теңдеудің екі бөлігі де – **a** да және **&a[0]** де жиымның 0-ші элементінің адресін анықтайды. Осы екі белгілеу де нұсқауыш типті константа болып табылады. Сондықтан оларды мән ретінде нұсқауыш типті айнымалыға меншіктеуге болады немесе қажет болса, нұсқауыш типті айнымалының мәнін өзгертуге болады.

11.14 мысал. Нұсқауыштың мәніне санды қосуға болатынын көрсететін программа қарастырайық.

```
main()
{int a[4], *pti, i;
 float b[4], *ptf;
 pti=a;
 ptf=b;
 for (i=0; i<4; i++)
  printf("нұсқауыштар +%d: %8u %10u\n",i,pti+i,
          ptf+i);
}
```

Мұның нәтижесі:

```
нұсқауыштар + 0:  65518        65498
нұсқауыштар + 1:  65520        65502
нұсқауыштар + 2:  65522        65506
нұсқауыштар + 3:  65524        65510
```

Келесі мысалды қарастырайық.

```
(a+2)==&a[2];
*(a+2)==a[2];
```

Бұлар нұсқауыштар мен жиымның арасындағы байланысты анықтайды, яғни жиымның жеке элементін анықтау үшін немесе оның мәнін пайдалану үшін нұсқауышты қолдануға болады.

11.9 Нұсқауыштарды пайдаланып жиымдармен жұмыс істеу

Жиымдарды функция арқылы қарастырып, содан кейін осы функцияны нұсқауыштарды пайдаланып жазып шығу керек болсын.

11.15 мысал. Функция арқылы жиым қосындысын табу.

```
// функция арқылы жиым қосындысын табу
#include <conio.h>
#include <stdio.h>
int f1(int a[], int t)
{
 int i,sum=0;
 for (i=0; i<t; i++)
  sum+=a[i];
 return (sum);
}
main()
{ int i,s,b[10]={5,6,14,12,30,5,9,7,15,5};
 clrscr();
 s=f1(b,10);
 printf("s=%d",s);
 getch();
}
```

Негізгі программада **f1** функциясын шақыру үшін нақты параметрлерді жазып, функция келесі түрде шақырылып отыр:

```
   f1(b,10);
```

11.16 мысал. Нұсқауышты функцияда (**f1)** пайдалану программасын жазайық.

```
// функцияда нұсқауыш арқылы жиым қосындысын табу
#include <conio.h>
#include <stdio.h>
int f1(int *pa, int t)
{
 int i,sum=0;
 for (i=0; i<t; i++)
  sum+=*(pa+i);
 return (sum);
}
main()
{ int i,s,b[10]={5,6,14,12,30,5,9,7,15,5};
clrscr();
s=f1(b,10);
printf("s=%d",s);
getch();}
```

программада осы **f1** функциясын шақыру үшін, нақты параметрлер бұрынғыдай жазыла береді: **f1(b,10);**

11.10 Нұсқауыштарға қолданылатын операциялар

C/C++ тілінде нұсқауыш типті айнымалыларға бес негізгі оператор қолдануға болады:

1. меншіктеу операциясы. Нұсқауышқа адресті меншіктеуге болады. Жиымның атын қолданып немесе адресті анықтайтын & операторын пайдаланып, әдетте адресті меншіктеуге болады;

2. мәнді анықтау. Берілген адрес бойынша кейбір ұяшықта сақталатын мәнді анықтау үшін * операциясы қоланылады;

3. нұсқауыштың адресін анықтау. Кез келген айнымалылар сияқты нұсқауыш типті айнымалылар мәні немесе адресі болуы мүмкін. & операциясы арқылы нұсқауыштың адресін анықтауға болады;

4. нұсқауыштарды арттыру. Бұл амал әдеттегі + операциясы көмегімен немесе арттыру операциясы арқылы орындалуы мүмкін. Нұсқауышты арттырып, жиымның келесі элементіне

өтуге болады (қажет болса, нұсқауыш мәнін кемітуге де болады);

5. нұсқауыштардың айырмасы. Бір жиымның элементіне сілтейтін нұсқауыштың айырмасын табуға болады. Жиым элементінің арасындағы ара қашықтығын анықтау үшін нұсқауыштың айырмасын есептеуге болады.

11.17 мысал.

```
/* Жиымның максимумын тауып, одан кейінгі эле-
менттерін кемуі бойынша реттеп орналастыру */
#include <conio.h>
#include <stdio.h>
#include <math.h>
#define n 10
main()
{ int i,j,t,c;
 int a[n]={6,5,9,8,7,4,1,2,3,0};
 int *pa,max;
clrscr();
printf("a[10] элементтері : ");
for (i=0; i<n; i++)
 printf(" %d ",a[i]);
pa=a; max=*pa; t=0; //максимум мен оның
                    //индексін табу
for (i=1; i<n; i++)
 if((*(pa+i)) > (max))
 {max = *(pa+i); t=i; }
 printf("\nmax = %d оның индексі = %d\n", max, t);
for (i=t; i<n-1; i++) // элементтерді кемуі
                      // бойынша реттеу
for (j=i+1; j<n; j++)
if(*(pa+i) < *(pa+j))
 { c=*(pa+i); // жиымның 2 элементін алмастыру
  *(pa+i)=*(pa+j);
  *(pa+j)=c;
 }
 printf("\nнәтиже : ");
```

```
 for (i=0; i<n; i++)
  printf("%d ", *(pa+i));
 getch();
}
```

11.11 Динамикалық айнымалылар

Программада жарияланған барлық айнымалылар мәліметтер сегменті (64Кбайт) деп аталатын жедел жадының үздіксіз бір аймағында орналасады. Мұндай айнымалылар программа орындалуы барысында өз көлемін өзгертпейді де, статикалық айнымалылар деп аталады. Жиымдардың көлемі үлкен болған жағдайда, мәліметтер сегментіне олар сыймай қалуы мүмкін. Осындай сәттерде динамикалық жады қолданылады.

Динамикалық жады – бұл программаға мәліметтер сегментінен тыс бөлініп берілген стек (тізбекті) түріндегі жады, онда программаның және оның ішкі функцияларының жергілікті айнымалылары орналасады.

Динамикалық жадымен жұмыс істеу үшін нұсқауыштар қоданылады. Солар арқылы динамикалық айнымалы деп аталатын динамикалық жады мәліметтерін қолдануға болады. Динамикалық айнымалылар арнайы функциялар мен операциялар көмегімен жасалады. Олар программа жұмысы аяқталғанша немесе арнайы функциялар мен операциялар арқылы жойылғанша қолданылады.

C/C++ тілдерінде динамикалық айнымалыларды қолдану (жасау) үшін new операциясы қоланылады:

нұсқауыш = new тип_аты[инициализатор];

мұндағы **инициализатор** – жай жақшадағы өрнек.

new операциясы берілген типке сәйкес динамикалық жадыны пайдалануға мүмкіндік береді. Егер инициализатор берілген болса, онда осы жады аймағына инициализаторда көрсетілген мән жазылады.

```
int *x=new int(5);
```

Динамикалық айнымалыларды жою (өшіру) үшін **delete** операциясы қолданылады:

delete нұсқауыш;

мұндағы нұсқауыш бұрын **new** операциясы арқылы бөлінген жады аймағының адресін көрсетеді.

```
delete x;
```

C/C++ тілінде динамикалық жадымен жұмыс істеуді анықтайтын кітапханалық функциялар анықталған, олар **<stdlib.h>** тақырып файлында орналасқан:

1) **void *malloc(unsigned s)** – нұсқауышты ұзындығы **s** байт болып келетін динамикалық жады аймағының басына қайтарады, қателік болған жағдайда NULL мәнін береді;

2) **void *calloc(unsigned n, unsigned m)** – нұсқауышты әрқайсысы m байттан тұратын n элемент орналасатын динамикалық жады аймағының басына қайтарады, қателік болған жағдайда NULL мәнін береді;

3) **void *realloc(void *p,unsigned s)** – бұрын бөлініп берілген динамикалық жады аймағының көлемін s байтқа дейін өзгертеді, p – өзгертілетін блок адресінің басы, қателік болған жағдайда NULL мәнін береді;

4) **void *free(void *p)** – бұрын бөлініп берілген динамикалық жады аймағын босатады, **p** – сол аймақ басының адресі.

Мысалы:

```
int *u=(int*)malloc(sizeof(int)); /* функцияға керекті жады көлемі байтпен беріледі, бірақ функция void * типіндегі мән қайтаратын болғандықтан, оны нұсқауыш типіне (int *) түрлендіру керек */
free(u); // бөлініп берілген жады аймағын босату
```

Бақылау сұрақтары

1. *Жиым дегеніміз не?*
2. *Жиымдарға бастапқы мәндер қалай тағайындалады?*
3. *Жиымды сипаттау тәсілдері.*
4. *Жиым элементтерін енгізу және экранға шығару жолдары.*
5. *Кездейсоқ сандарды қалай шығаруға болады?*
6. *Жиымға кездейсоқ сандарды меншіктеу қалай орындалады?*
7. *Жиымды өңдеу есептерінің түрлері (кластары).*
8. *Жиымның ең үлкен (ең кіші) элементін анықтау.*
9. *Жиым элементтері қосындысын табу.*

10. *Жиым ішіндегі екі элементтің бір-бірімен орнын алмастыру.*
11. *Жиым элементтерін кері бағытта орналастыру.*
12. *Жиымның көрсетілген элементтерін өңдеу тәсілдері.*
13. *Жиымды сұрыптау (сорттау, реттеу) жолдары.*
14. *Адрестік операциялар түсінігі.*
15. *Нұсқауыш дегеніміз не? Ол қалай сипатталады?*
16. *Функциялар арасында байланыс жасау үшін нұсқауыштарды пайдалану.*
17. *Жиымдарға қолданылатын нұсқауыштар.*
18. *Нұсқауыштарға қолданылатын операциялар.*
19. *Динамикалық жады ұғымы.*
20. *Динамикалық айнымалыларды қолдану операциялары.*

Тапсырмалар

Жиымдарды өңдеу кезінде төмендегі ережелерді есте сақтаған жөн.

1. Жиым өлшемі (ені, көлемі) тек константа немесе константалық өрнек бола алады. Жиым өлшемін идентификатор түріндегі (ат қойылған) константа арқылы беру дұрыс деп саналады.
2. Жиым элементтері нөлден бастап нөмірленеді, сол себепті элементтің ең үлкен нөмірі элементтер санынан бірге кем болады.
3. Индекс нөмірлерінің жиым шекарасынан асып кетпеуін программа қадағаламайды, оны программалаушы өзі бақылауы тиіс.
4. Нұсқауыш — жедел жады аймағының адресін сақтайтын айнымалы.
5. Жиым аты оның 0-элементіне нұсқауыш болып табылады.
6. Егер программаға енгізілетін элементтер мәні алдын ала белгілі болса, онда жиымды сипаттау кезінде оның мәндерін көрсетіп (инициалдау) кету керек. Жиымды глобальді айнымалы емес, локальді айнымалы етіп сипаттаған дұрыс.
7. Жиым элементтерін пернеден енгізгеннен гөрі **rand()** және **random()** функцияларын пайдаланып, кездейсоқ сандар тізбегін беру арқылы енгізген ыңғайлы болады.
8. Жиымдарды сұрыптау алгоритмдерінің жұмыс істеу жылдамдығы, алатын көлемі және қолданылу аймағы әр түрлі болып келеді.

Тапсырмаларды орындауға арналған нұсқаулар

A. Әрбір студент журналдағы өз нөміріне сәйкес (мұғалімнің көрсетуі бойынша) төрт есеп шығаруы тиіс.
B. Әр есептің блок-схемасы және соған сәйкес программасы құрылып, компьютерде сол программаның нәтижесін алу керек.
C. Екі блок-схема Word программасындағы *Сурет салу-Автофигура-*

лар-Блок-схемалар мүмкіндігімен сызылып көрсетілуі тиіс.

D. Берілген жиым элементтері мен нәтижелік элементтерді экранға шығарып, нәтижелерді жазып алу қажет немесе нәтижені бірден файлға жазуға тырысу керек.

Есептер

1. Кез келген сандардың бірөлшемді *A*(10) жиымындағы оң элементтерді екі есе кемітіңдер, ал теріс элементтері болса, онда оларды индекстерінің мәнімен ауыстырыңдар.
2. Бірөлшемді *A*(10) жиымындағы теріс элементтердің ең үлкенін табыңдар.
3. Бірөлшемді *A*(20) жиымындағы –5-тен кіші элементтердің қосындысын және олардың жалпы санын, сонымен бірге 5 санына бөлінетін элементтердің нөмірлерін анықтаңдар.
4. Бірөлшемді *A*(10) жиымындағы оң элементтердің квадраттарының арифметикалық ортасын есептеңдер.
5. Бірөлшемді *A*(100) жиымындағы теріс элементтердің санын анықтаңдар.
6. Бірөлшемді жиым мәндері берілген диапазонда жататын элементтерінің нөмірін басып шығарыңдар.
7. Бірөлшемді *A*(10) жиымындағы оң элементтердің ішінде мәні ең кішісін және оның индексін (нөмірін) табыңдар.
8. Бірөлшемді жиымда 2-элементті бірінші орынға, 3-ні 2-шімен және т.б. алмастырулар орындай отырып, 1-элементті соңғы орынға қойыңдар.
9. Берілген оң сандар тізбегіндегі қосындысы берілген саннан асып кетпейтін элементтердің санын табыңдар.
10. Ұтыс билетінің нөмірі алты орынды сан. Билет "бақытты" (алдыңғы және соңғы үш цифрының қосындысы өзара тең) немесе "табысты" (жұп орындарда тұрған цифрлардың қосындысы тақ орында тұрғандардың қосындысына тең) болатындығын анықтаңдар.
11. Оң және теріс сандар тізбегіндегі бірінші теріс санға дейінгі орналасқан сандар тізбегінің өсу ретімен орналасатындығын анықтаңдар.
12. *N* кәсіпорынның бір жылғы электр энергиясын тұтынуы туралы дерек бар. Осы бойынша энергия тұтынудың арифметикалық ортасын және энергияны ең көп үнемдеген кәсіпорынды анықтаңдар.
13. 14 аудан бойынша жанармай қорының мөлшері белгілі. Жанармаймен ең жақсы қамтылған үш ауданды анықтаңдар.
14. Берілген *A*(15) жиымы бойынша мына шарттарды қанағаттандыратын *B* жиымын құрыңдар:
    $B(0) = A(1)$; $B(1) = A(1) * A(3)$; $B(2) = A(1) * A(3) * A(5)$; ...
    $B(7) = A(1) * A(3) * ... * A(13)$

15. Берілген $A(14)$ жиымы бойынша мына шарттарды қанағаттандыратын B жиымын құрыңдар :
$B(0) = A(0)$; $B(1) = A(0) * A(2)$; ... $B(7) = A(0) * A(2) * ... * A(13)$
16. Берілген $A(15)$ жиымы бойынша мына шарттарды қанағаттандыратын B жиымын құрыңдар:
$B(0) = A(0) – A(14)$; $B(1) = A(1) – A(13)$; $B(2) = A(2) – A(12)$; ...
$B(7) = A(7) – A(8)$
17. Берілген $A(16)$ жиымы бойынша мына шарттарды қанағаттандыратын B жиымын құрыңдар:
$B(0) = A(0)/A(15)$; $B(1) = A(1)/A(14)$; $B(2) = A(2)/A(13)$, ...
$B(7) = A(7)/A(8)$
18. $A(14)$ жиымы берілген. Жиым элементтерінің арифметикалық ортасын және ол мәнге еңжақын элементтің нөмірін анықтаңдар.
19. $A(14)$ жиымы берілген. Екі жиым құрыңдар: оның біріншісі индекстері жұп сандар, ал екінші жиым индекстері тақ сан болатын элементтер болсын.
20. N элементтен тұратын бірөлшемді жиым берілген. Әрбір жолында L элемент болатын екіөлшемді жиым құрастырыңдар.
21. $A(17)$ жиымы берілген. 7-ден үлкен барлық элементтерді берілген жиымның ең үлкен элементімен ауыстырыңдар.
22. $A(16)$ жиымы берілген. Алғашқы оң элементке дейінгі теріс элементтердің санын анықтаңдар.
23. $A(18)$ жиымы берілген. 2-ден кіші барлық элементтерді 3 санымен ауыстырыңдар.
24. $A(17)$ жиымы берілген. 0-ден үлкен барлық элементтерді 5-ке, ал қалғандарын 0-ге ауыстырыңдар.
25. $A(14)$ жиымы берілген. A жиымының барлық тақ элементтері индекс нөмірлерімен ауыстырылатын B жиымын құрыңдар.
26. $A(34)$ жиымы берілген. Тақ нөмірлі индекстерде тұрған элементтердің ең үлкенін анықтаңдар.
27. $A(N)$ жиымы берілген. Алғашқы ең кіші (минимал) элементтің орнын (индексін) анықтаңдар.
28. $A(N)$ жиымы берілген. Соңғы ең кіші элементтің орнын анықтаңдар.
29. $A(N)$ жиымы берілген. Соңғы ең үлкен (максимал) элементтің орнын анықтаңдар.
30. $A(N)$ жиымы берілген. Алғашқы ең үлкен элементтің орнын анықтаңдар.
31. $A(N)$ жиымы берілген. Мәндері 3 пен 11 саны интервалында жататын элементтердің арифметикалық ортасын анықтаңдар.
32. $A(12)$ = {2,5; 4,3; –0,57; 10,45; 1,5; –7,1; 11,4; 5,12; 4,9; 7,7; –12,3; 0,031} жиымы берілген. $B(I) = SIN(A(I))$ формуласы бойынша $B(12)$

жиымын құрастырыңдар және $P = A(0) * B(11) + A(1) * B(10) + ... + A(11) * B(0)$ өрнекті есептеп шығарыңдар.

33. Пернетақтадан жиымға бес бүтін мән енгізіңдер. Олар бір жолға үтір арқылы жазылады. Жиымның арифметикалық ортасын табыңдар.
34. Пернетақтадан *X* жиымының бес бүтін мәнін енгізіңдер. Жиым элементтерінің әрқайсысының түбірінің мәнін және квадратын экранға шығарыңдар.
35. Кездейсоқ сандар генераторының көмегімен *A*[0..7] жиымын құрыңдар және оны экранға шығарыңдар. Оның барлық элементтерін 2 есеге арттырыңдар.
36. Кездейсоқ сандар генераторының көмегімен элементтері –10 мен 10 сан аралығында болатын *A*[0..8] жиымын құрыңдар және оны экранға шығарыңдар. Жиымның теріс элементтерінің санын есептеңдер.
37. Кездейсоқ сандар генераторының көмегімен элементтері –20 мен 10 саны аралығында болатын A[0..12] жиымын құрыңдар және оны экранға шығарыңдар. Жиымның барлық теріс элементтерін 0-ге ауыстырыңдар.
38. Кездейсоқ сандар генераторы арқылы элементтері –15 пен 30 саны аралығында болатын бүтін саннан A[0..15] жиымын құрастырыңдар және оны экранға шығарыңдар. Жиымның ең үлкен элементін және оның индексін анықтаңдар.
39. 8 сағ-тан 20 сағ-қа дейінгі уақыт аралығында ауа температурасы сағат сайын өлшенеді. Осы аралықта температураның төмендегені мәлім. Температураның алғашқы теріс мәні қай сағатта пайда болғандығын анықтаңдар.
40. Қараша айында он күн бойына ауа температурасы туралы деректер жиымда сақталған. Температураның –10-нан қанша рет төмен болғанын анықтаңдар.
41. Балқаш көлі жағалауының температурасы туралы мәлімет қыркүйек айында он күн бойы жиымда сақталған. Осы уақыт ішінде неше күн шомылуға қолайлы болатынын анықтаңдар.
42. Сәуір айының он күндік ауа температурасы мен жауын-шашын мөлшері туралы мәлімет жиымда сақталады. Осы он күнде жауған жаңбыр және қардың мөлшерін анықтаңдар.
43. Желтоқсан айының он күндік ауа температурасының мәліметі жиымда сақталған. Он күндік температура мәліметі бойынша қанша рет орта температурадан жоғары ауытқу болатындығын анықтаңдар.
44. Қараша айының он күн ішіндегі желдің (солтүстік, оңтүстік, шығыс, батыс) бағыты және соғу күшінің мәліметтері жиым түрінде сақталған. Неше күн желдің жылдамдығы 8 м/с-тан артық жылдамдықпен соққандығын анықтаңдар?

45. Бүтін саннан тұратын 15 элементті жиым құрыңдар және олардың арасындағы ең кіші элементті анықтаңдар.
46. Жерге еркін түсу кезінде дененің 1, 2, ... , 10 с ішінде жүріп өткен қашықтықтарынан құралатын нақты сандардың сызықтық жиымын құрыңдар.
47. Бүтін сандардың сызықтық жиымы берілген. Оның кему ретімен орналасқандығын тексеріңдер.
48. Бүтін сандардың сызықтық жиымындағы оң элементтердің қосындысын табыңдар. Жиымның өлшемі — 10. Жиымды пернетақтадан толтырыңдар.
49. Бүтін сандар жиымындағы жұп нөмірлі элементтердің қосындысын табыңдар. Жиымның өлшемі 20. Жиымды 100 бен 200 арасындағы кездейсоқ сандармен толтырыңдар.
50. Бүтін сандар жиымындағы 7-ге қалдықсыз бөлінетін элементтердің көбейтіндісін табыңдар. Жиым өлшемі 15. Жиымды 10 мен 50 арасындағы кездейсоқ сандармен толтырыңдар.
51. Нақты сандар жиымындағы тақ нөмірлі элементтердің қосындысын табыңдар. Жиым өлшемі 20. Жиымды 100 бен 200 арасындағы кездейсоқ сандармен толтырыңдар.
52. Бүтін сандар жиымындағы барлық 0-ден кіші элементтердің көбейтіндісін табыңдар.
53. Бүтін сандар жиымында: "2-ге бөлгенде қалдық 3-ке тең" шартын қанағаттандыратын барлық элементтердің қосындысын табыңдар. Жиымның өлшемі 20. Жиымды 200 бен 300 аралығындағы кездейсоқ сандармен толтырыңдар.
54. Нақты сандар жиымындағы берілген саннан кіші барлық элементтердің қосындысын табыңдар. Жиымның көлемі 20. Жиымды 50 мен 100 аралығындағы кездейсоқ сандармен толтырыңдар.
55. Нақты сандар жиымындағы берілген саннан кіші барлық элементтердің көбейтіндісін табыңдар. Жиымның көлемі 10. Жиымды 50 мен 100 аралығындағы кездейсоқ сандармен толтырыңдар.
56. Жиымдағы 3 пен 9-ға қалдықсыз бөлінетін элементтердің көбейтіндісін табыңдар. Жиымның көлемі 10. Жиымды 5 пен 500 аралығындағы кездейсоқ сандармен толтырыңдар.
57. Бүтін сандар жиымында элементтердің арифметикалық ортасынан кіші болатын барлық элементтердің қосындысын табыңдар. Жиымның көлемі 20. Жиымды 150 мен 300 аралығындағы кездейсоқ сандармен толтырыңдар.
58. Бүтін сандар жиымында 5-ке де, 8-ге де бөлінетін элементтердің қосындысын табыңдар. Жиымның көлемі 30. Жиымды 100 бен 500 аралығындағы кездейсоқ сандармен толтырыңдар.
59. Науқастың тәулік бойы температурасын талдайтын программа жазыңдар: температураның минимал, максимал және ариф-

метикалық орта мәнін анықтаңдар. Температура тәулік бойына 6 рет өлшенеді және нәтиже пернетақтадан *T* жиымына енгізіледі.

60. Сызықтық жиымға бір жылдың әрбір 12 айының жауын-шашын мөлшері туралы мәлімет тіркелген. Жыл бойы жауған жауын-шашынның мөлшерін, оның орта айлық мөлшерін, құрғақшылық (жауын-шашын мөлшері 30 мм-ден аз) болған айлардың санын, жылдың ең құрғақ айын анықтайтын программа жазыңдар.
61. Бірөлшемді жиымда берілген *a* санына тең алғашқы жұп элементтердің санын табыңдар.

12. 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, *a* 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 };

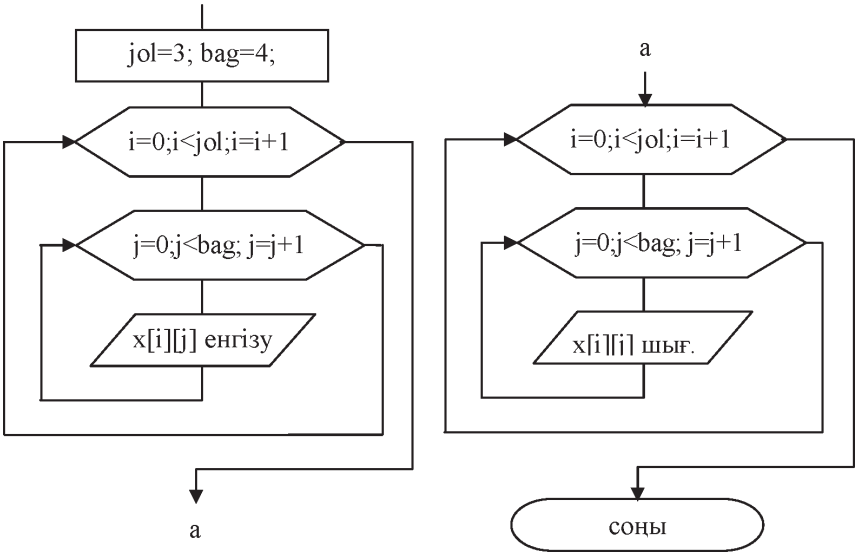
мұнда бірінші жолдың 3 элементі мен екінші жолдың бірінші элементі анықталады да, қалғандары 0 болып саналады.

Екі өлшемді жиымды инициалдау қабаттасқан циклдер арқылы орындалады.

12.1-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();
}
```



12.1-сурет. Жиым элементтерін енгізу – шығару

Матрицаларды өңдейтін негізгі алгоритмдер ретінде бір өлшемді жиымдарды өңдеу кезінде қолданылған алгоритмдер саналады. Жалпы матрицаларды өңдейтін барлық алгоритмдерді екі топқа бөліп қарастыруға болады, олар:

1. матрицаның барлық элементтерін өңдейтін алгоритмдер.

2. матрицаның әр жолы немесе әр бағанасы элементтерін жеке-жеке өңдейтін алгоритмдер.

12.1 Матрицаның барлық элементтерін өңдейтін алгоритмдер

12.2-мысал. Нақты сандардан тұратын $a_{4,6}$ матрицасы берілген. Мынадай өрнекті $Z = \frac{P1}{|P2|}$ есептеу керек, мұндағы P1 и P2 – сәйкесінше алынған матрицаның оң және теріс элементтерінің көбейтіндісі.

```
/* a[4][6] матрицасы берілген. z=p1/|p2| есеп-
теу керек, p1 және p2 – матрицаның оң және теріс
элементтерінің көбейтіндісі */
#include <math.h>
#include <conio.h>
#include <stdio.h>
void line()
 {printf("-----------------------------\n");
 return;}
main ()
{
 static int a[4][6]={
                          {5,-11,4,-2,5,6},
                          {3,3,-12,-5,7,8},
                          {2,3,-3,14,-9,-3},
                          {-9,3,-6,14,9,-3}
                     };
 int i,j;
 float p1, p2, z;
 clrscr();

 printf("Берілген матрица :\n");
 line();
 for (i=0; i<4; i++)
  {for (j=0; j<6; j ++)
   printf(" %3i ", a[i][j]);
  printf("\n");}
 line();
```

```
  /* Матрицаны өңдеу */
p1 = 1;
p2 = 1;
  for (i=0; i<4; i++)
  for (j=0; j<6; j ++)
   {if (a[i][j]>0) p1 = p1*a[i][j];
   if (a[i][j]<0) p2 = p2*a[i][j];
   }
   z = p1/abs(p2);
  printf("   z = %f\n",z);
  line();
  getch();
}
```

Матрица мынадай түрде беріледі:

A(0,0)	**A(0,1)**	**A(0,2)**	**A(0,3)**	**...**	**A(0, j)**	**…**	**A(0,n-1)**
A(1, 0)	**A(1,1)**	**A(1, 2)**	**A(1, 3)**	**...**	**A(1, j)**	**…**	**A(1, n-1)**
A(2, 0)	**A(2, 1)**	**A(2,2)**	**A(2, 3)**	**...**	**A(2, j)**	**…**	**A(2, n-1)**
A(3, 0)	**A(3, 1**	**A(3, 2)**	**A(3, 3**	**...**	**A(3, j)**	**…**	**A(3, n-1)**
…	**…**	**…**	**…**	**…**	**A(i, j)**	**…**	**…**
A(n-2, 0)	**A(n-2, 1)**	**A(n-2, 2)**	**A(n-2, 3)**	**...**	**A(n-2, j)**	**…**	**A(n-2, n)**
A(n-1, 0)	**A(n-1, 1)**	**A(n-1, 2)**	**A(n-1, 3)**	**...**	**A(n-1, j)**	**…**	**A(n-1, n-1)**

Бас диагональ элементтері белгісі: **i = j**
Бас диагональдан жоғары тұрған элементтері белгісі: **i < j**
Бас диагональдан төмен тұрған элементтері белгісі: **i > j**
Қосалқы диагональ элементтері белгісі: **i+j = n-1**
Қосалқы диагональдан жоғары элементтер белгісі: **i+j < n-1**
Қосалқы диагональдан төмен элементтер белгісі: **j+j > n-1**

12.3-мысал. Бүтін сандардан тұратын квадрат $b_{5,5}$ матрицасы берілген. Оның бас диагоналының сол жағында және оң жағында орналасқан нөлге тең элементтері санын анықтап, солардың айырмасының модулін табу керек.

Мынадай белгілеулер енгізейік:

L1 – бас диагональдың сол жағында (төменінде) орналасқан элементтер саны;

L2 – бас диагональдың оң жағында (жоғарысында) орналасқан

элементтер саны;
L= |L1-L2| – солардың айырмасы модулі.

```
#include <math.h>
#include <conio.h>
#include <stdio.h>
void line()
 {printf("------------------------\n");
 return;}
main ()
{ static int b[5][5]={    {5,0,0,0,0},
{0,3,12,0,0},
{0,33,13,14,0},
{0,23,0,14,0},
{35,0,13,14,9},
                 };
 int i,j;
 int L1,L2,L;
  clrscr();
 printf("Берілген матрица :\n");
 line();
 for (i=0; i<5; i++)
  {for (j=0; j<5; j ++)
   printf(" %3i ", b[i][j]);
  printf("\n");}
 line();
L1 = L2 = 0;
for (i=0; i<5; i++)
 for (j=0; j<5; j ++)
 if (b[i][j]==0)
  {if (i>j) L1 = L1+1;
  if (i<j) L2 = L2 +1;
  }
 L= abs(L1 - L2);
 printf("    L = %i ", L);
 getch();
}
```

12.2 Екінші типтегі есептер алгоритмдері

12.4-мысал. Бүтін сандардан тұратын $a_{3,6}$ матрицасы жолдарының алғашқы элементін осы жолдың минимальды элементімен алмастыру керек. Нәтижелік $a_{3,6}$ матрицасы элементтерін экранға шығару қажет.

```
/* a[3][6] матрицасы жолдарының алғашқы элементін
осы жолдың минимальды элементімен алмастыру ке-
рек. Натижелік X матрицасы элементтерін экранға
шығару кажет.*/
#include <math.h>
#include <conio.h>
#include <stdio.h>
void line()
 {printf("------------------------------\n");
 return;}
main ( )
{ static int a[3][6]={    {5,-11,4,-2,5,6 },
                          {2,3,-3,14,-9,-3},
                          {-9,3,-6,-14,9,-3}
                      };
 int i,j,jmin,min;
 clrscr();
 printf("Берілген матрица :\n");line();
 for (i=0; i<3; i++)
  {for (j=0; j<6; j ++)
   printf(" %3i ", a[i][j]);
  printf("\n");
  }
 line();
 for (i=0; i<3; i++)
  { min=+1E6;
  for (j=0; j<6; j ++)
   if (a[i][j]<min)
    {min=a[i][j];
     jmin=j;
    }
```

```
  a[i][jmin]=a[i][0];
  a[i][0]=min;
  }
  printf("Өңделген матрица :\n");line();
  for (i=0; i<3; i++)
  {for (j=0; j<6; j ++)
   printf(" %3i ", a[i][j]);
  printf("\n");
  } line();
 getch();
}
```

12.5-мысал. Бүтін сандардан тұратын $a_{3,4}$ матрицасының әрбір бағаналарының арифметикалық орташа мәнін анықтап, оларды бір өлшемді s_4 жиымы ретінде бейнелеу керек.

```
/* a[3][4] матрицасының әрбір бағаналарының
арифметикалық орташа мәнін анықтап,оларды бір
өлшемді s[4] жиымы ретінде бейнелеу керек. */
//әрбір бағана қосындысы және солардың орташа мәні
#include <conio.h>
#include <stdio.h>
main ()
{ static int a[3][4]={    {5,11,4,2},
                          {3,3,12,5},
                          {2,3,3,14}
                      };
 int i,j;
 float s[4];
 clrscr();
 printf("Берілген матрица :\n");
 printf("---------------------\n");
 for (i=0; i<3; i++)
  {for (j=0; j<4; j ++)
   printf(" %3i ", a[i][j]);
  printf("\n");}
  /* матрицаны өңдеу */
 printf("---------------------\n");
```

```
 printf("");
 for (j=0; j<4; j++)
  { s[j]=0;
   for (i=0; i<3; i ++) s[j]+=a[i][j];
    printf(" %4.2f",s[j]/3);
  }
 printf("\n----------------------");
getch();
}
```

Бұл программада **s[j]** жиымының әрбір элементін есептеуде қабаттасқан екі цикл қолданылған, онда **j** индексі сыртқы цикл параметрі, ал **i** индексі – ішкі цикл параметрі. Осы тәсіл матрица элементтерін бағаналар бойынша өңдеу ісін жүзеге асырады.

12.6-мысал. Берілген жиымның әрбір жолындағы элементтері қосындыларын және сол қосындылардың орташа мәнін анықтау керек.

```
// әр жол қосындысы және солардың арифметикалық ортасы
#include <conio.h>
#include <stdio.h>
main ()
{ static int a[3][4]= {
                          {5,3,4,2},
                          {3,3,4,5},
                          {2,3,3,4}
                       };
int i,j,s=0;
float c=0;
clrscr();

for (i=0; i<3; i++)
 {for (j=0; j<4; j ++) s+=a[i][j];
 printf("%i-жол косындысы:%i\n",i+1,s);
 c+=s;
 }
printf("--------------------");
printf("\nарифм.ортасы %5/2f", c/4) ;
}
```

12.7-мысал. Берілген жиымның әрбір бағанадағы элементтері қосындыларын және сол қосындылардың орташа мәнін анықтау керек.

```
// әрбір бағана қосындылары мен солардың орташа
мәнін анықтау
#include <conio.h>
#include <stdio.h>
main ()
{
 static int a[3][4]={
                      {5,11,4,2},
                      {3,3,12,5},
                      {2,3,3,14}
                     };
 int i,j,s; float c;
 clrscr();
 printf("Берілген матрица :\n");
 printf("---------------------\n");
 for (i=0; i<3; i++)
  {for (j=0; j<4; j ++)
   printf(" %3i ", a[i][j]);
  printf("\n");}
  /* Матрицаны өңдеу */
 printf("---------------------\n");
 printf("s=");
 for (j=0; j<4; j++)
  { s=0;
   for (i=0; i<3; i ++) s+=a[i][j];
    printf("%3i ",s); // қосынды s
  c+=s;
  }
 printf("\n---------------------");
 printf("\n арифм. ортасы %5.2f",c/4);
 getch();
}
```

12.8-мысал. Берілген $a_{4,4}$ жиымының бас диагоналындағы элементтерді нөлге, ал қосалқы диагоналындағы элементтерді – бірге теңестіру программасы.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

main()
{
 int a[4][4]={
                      {6,8,9,2},
                      {5,3,4,2},
                      {3,3,4,5},
                      {2,3,3,4}
             };
 int i,j;
 clrscr();
 printf("Берілген матрица :\n");
 for (i=0; i<4; i++)
  {for (j=0; j<4; j ++)
   printf(" %2i ", a[i][j]);
  printf("\n");}
  /* Матрицаны өңдеу */
 for (i=0; i<4; i++)
    a[i][i]=0;
 for (i=0; i<4; i++)
  for (j=0; j<4; j ++)
    if (i+j==3) a[i][j]=1;
  /* Өңделген матрица элементтерін шығару */
 printf("Нәтижелік матрица :\n");
 for (i=0; i<4; i++)
  {for (j=0; j<4; j ++)
   printf(" %2i ", a[i][j]);
  printf("\n");}
getch();
}
```

12.9-мысал. Берілген $a_{3,3}$ жиымының бас диагоналынан төмен орналасқан элементтер қосындысын анықтау программасы.

```
#include <conio.h>
#include <stdio.h>

main ()
{
 static int a[3][3] = {
                          {0,1,2},
                          {3,4,5},
                          {6,7,8}
                       };
 int i,j,r,s;
 clrscr();
printf("Енгізілгена[3][3]жиымэлементтері:\n");
 for (i=0; i<3; i++)
  {for (j=0; j<3; j++)
      printf(" %2i",a[i][j]);
  printf("\n");
  }
  /* бас диагоналдан төмен орналасқан элементтер
    қосындысын анықтау */
  s=0;
  for (i=0; i<3; i++)
  for (j=0; j<3; j++)
   if (j<i) s+=a[i][j];
  printf("\ns=%i",s);
  getch();
}
```

12.10-мысал. Берілген $a_{3,3}$ жиымының әрбір жолындағы элементтерді өсуі бойынша реттеп орналастыру программасы.

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

```
{ static int a[3][3] =  {
                          {8,7,6},
                          {5,4,3},
                          {2,1,0}
                        };
 int i,j,r,s,n=3,amin,m,k;
 clrscr();
  printf("Берілген матрица:\n");
 for (i=0; i<3; i++)
  {for (j=0; j<3; j++)
      printf(" %2i",a[i][j]);
  printf("\n");
  }
 for (i=0; i<n; i++)  //жолды таңдау
 {                    //мин тауып алмастыру
 for (k=0; k<n-1; k++)
  { amin=a[i][k];m=k;
  for (j=k+1; j<n; j++)
   if (a[i][j] < amin)
       {amin=a[i][j]; m = j;}
  a[i][m]=a[i][k]; a[i][k]=amin;
  }
 }
printf("\nНәтижелік матрица:\n");
for (i=0; i<3; i++)
  {for (j=0; j<3; j++)
      printf(" %2i",a[i][j]);
  printf("\n");
  }
getch();
```

12.11-мысал. Берілген $a_{3,3}$ жиымының әрбір бағанасындағы элементтерді өсуі бойынша реттеп орналастыру программасы.

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <conio.h>
main ()
{ static int a[3][3]= {
                              {8,7,6},
                              {5,4,3},
                              {2,1,0}
                            };
 int i,j,r,s,n=3;
 int amin;
 int m,k;
 clrscr();
  printf("Берілген матрица:\n");
 for (i=0; i<3; i++)
  {for (j=0; j<3; j++)
      printf(" %2i",a[i][j]);
  printf("\n");
  }

for (j=0; j<n; j++)      //бағананы таңдау
 {for (k=0; k<n-1; k++) //мин тауып алмастыру
  { amin=a[k][j];m=k;
  for (i=k+1; i<n; i++)
   if (a[i][j] < amin)
        {amin=a[i][j]; m = i;}
  a[m][j]=a[k][j]; a[k][j]=amin;
  }
 }
printf("\nСұрыпталған матрица элементтері:\n");
for (i=0; i<3; i++)
  {for (j=0; j<3; j++)
      printf(" %2i",a[i][j]);
  printf("\n");
  }
getch();
}
```

12.3 Екі өлшемді жиымдармен жұмыс істеу кезінде нұсқауыштарды қолдану

$A_{3,2}$ жиым берілген болсын. Олар бүтін сандар, яғни

```
int a[3][2];
int *pri;
pri = a; // бұл pri=a[0][0] деген сөз
```

a – жиымның аты немесе **a[0][0]** элементінің адресі

```
a=&a[0][0];
```

pri нұсқауышына 1-ді қоссақ, **pri+1** деген нұсқауыш **a[0][1]** элементіне сілтейді. Бұл жиымды қарастырғанда, келесі теңдеулер дұрыс болып табылады:

```
 pri == &a[0][0];
 pri+1== &a[0][1];
 pri+2== &a[1][0];
 pri+3== &a[1][1];
 pri+4== &a[2][0];
 pri+5== &a[2][1];
```

Екі өлшемді жиым бір өлшемді жиымдардан құрастырылған жиым ретінде қарастырылуы мүмкін. Яғни берілген екі өлшемді жиым үш жолдан тұрады, ал әрбір жол екі элементтен тұратын жиым болып табылады.

Бірінші жол аты – **a[0],**

екінші жол аты – **a[1],**

үшінші жол аты – **a[2].**

Жиымның аты берілген жиымның нұсқауышы болып табылады, яғни ол жиымның 1-элементіне сілтейді.

Екі өлшемді жиымның осы қасиеті бір өлшемді жиымға арналған функцияны екі өлшемді жиыммен жұмыс істеуге мүмкіндік береді.

12.12-мысал. Төмендегі **b[3][4]** матрицасы жолдарының арифметикалық ортасын табатын программада функция қолданылған.

```
/* b[3][4] матрицасы жолдарының қосындысы */
#include <conio.h>
#include <stdio.h>
float f1(int x[], int n)
 { int k; float s;
```

```
  for (k=0,s=0;k<n; k++)
     s+=x[k];
  return(s/n);
 }
main ()
{ int i,b[3][4]={    {6,4,3,3},
                     {7,5,3,3},
                     {8,4,2,6}
                 };
 clrscr();
 for (i=0;i<3;i++)
   printf("%d жолының орташа мәні: %f\n",
           i,f1(b[i],4));
 getch();
}
```

Бақылау сұрақтары

1. *Екі өлшемді жиымды – матрицаны сипаттау, бастапқы мәндерді тағайындау тәсілдері.*
2. *Екі өлшемді жиымды инициалдауды қабаттасқан циклдер арқылы орындау.*
3. *Матрица элементтерін толық өңдейтін алгоритмдер.*
4. *Матрицалардың көрсетілген элементтерін өңдеу тәсілдері.*
5. *Матрицалардың диагоналына байланысты орналасқан элементтерін өңдеу жолдары.*
6. *Матрицалардың жолдарының және бағаналарының ең үлкен (кіші) элементтерін, қосындыларын, көбейтінділерін табу.*
7. *Матрицалардың жолдарында және бағаналарында орналасқан элементтерді өсуі (кемуі) бойынша реттеу алгоритмдері.*

Тапсырмалар

1. Екіөлшемді $A(10,10)$ жиымда әрбір жол үшін оң элементтердің қосындысын есептеп шығарыңдар.
2. Екіөлшемді $A(10,10)$ жиымындағы оң элементтердің санын есептеп шығарыңдар.
3. Екіөлшемді $A(10,10)$ жиымындағы ең үлкен элементті анықтаңдар.

4. Екіөлшемді $A(10,10)$ жиымы берілген. A жиымының әрбір жолындағы элементтердің көбейтіндісіне тең болатын бірөлшемді B жиымын алыңдар.
5. Екіөлшемді $A(10,10)$ жиымы берілген. A жиымының әрбір жолындағы ең үлкен элементтің мәндеріне тең болатын бірөлшемді B жиымын алыңдар.
6. Екіөлшемді $A(10,10)$ жиымы берілген. A жиымының әрбір жолындағы теріс элементтің мәндеріне тең болатын бірөлшемді B жиымын алыңдар.
7. Екіөлшемді $A(10,10)$ жиымында теріс элементтері бар жолдардың санын есептеп шығарыңдар.
8. Екіөлшемді жиым жолдарындағы ең үлкен элементтердің қосындысын есептеп шығарыңдар.
9. Екіөлшемді жиым әрбір бағанадағы (тік жолдағы) элементтердің қосындысын есептеп шығарыңдар.
10. Екіөлшемді жиымның әрбір жолындағы теріс сандардың санын, қосындысын және арифметикалық ортасын есептеп шығарыңдар.
11. Екіөлшемді $A(4,4)$ жиымындағы индекстерінің қосындысы 4-ке тең болатын элементтерінің қосындысын есептеп шығарыңдар.
12. Екіөлшемді $A(4,7)$ жиымындағы оң элементтердің арифметикалық ортасын және нөлге тең элементтердің санын есептеп шығарыңдар.
13. Екіөлшемді $A(10,10)$ жиымының бас диагоналі бойындағы элементтердің ең үлкенін табыңдар.
14. Екіөлшемді $A(10,10)$ жиымының қосымша диагоналі бойындағы ең үлкен элементті табыңдар.
15. Екіөлшемді $A(10,10)$ жиымының қосымша диагоналі бойындағы ең үлкен элементті табыңдар.
16. Екіөлшемді $A(4,7)$ жиымының әрбір жолындағы берілген диапазонда жататын элементтердің арифметикалық ортасын есептеп шығарыңдар.
17. Екіөлшемді $A(10,10)$ жиымында элементтердің арифметикалық ортасы жиымның барлық элементтерінің арифметикалық ортасынан кем болатын бағананың нөмірлерін анықтаңдар.
18. Екіөлшемді $A(10,10)$ жиымында 3- және 1-жолдардың орындарын ауыстырыңдар.
19. Екіөлшемді $A(7,7)$ жиымында бас диагональдағы элементтерді әрбір жолдың ең үлкен мәнімен алмастырыңдар.
20. Екіөлшемді $A(10,10)$ жиымында бас диагональдан жоғарғы және төмен орналасқан элементтердің қосындысын есептеп шығарыңдар.
21. Екіөлшемді жиымда берілген жолдың ең үлкен элементінің мәнін және нөмірін анықтаңдар.
22. Екіөлшемді жиымда әр бағана үшін берілген бағананың мәнінен

кіші болатын элементтердің арифметикалық ортасын есептеп шығарыңдар.

23. Бүтін сандардан құралған 10Х10 матрицасының ең үлкен элементінің жолы мен бағана нөмірін шығарып беретін программа жазыңдар.
24. *A*(5,5) жиымы және *k* саны берілген. Әрбір жолдың элементтерін осы жолдағы бас диагональда орналасқан диагональдық элементке бөліңдер.
25. *A*(10,10) жиымы берілген. Осы жиымның бас диагоналінің элементтерінен тұратын бірөлшемді жиым құрыңдар .
26. *A*(10,10) жиымы берілген. *k*- және 1-жолдардың орындарын ауыстырыңдар.
27. *A*(10,10) жиымы берілген. Одан бірөлшемді жиым құрастырыңдар.
28. Бүтін санды ***x* [0...5, 0...4]** жиымы берілген. Оның 5-тен кіші барлық элементтерін 111 санымен ауыстырыңдар.
29. Бүтін санды ***B* [0...4, 0...3]** жиымы берілген. Оның элементтерін олардың квадраттарымен ауыстырыңдар.
30. Нақты сандар жиымы ***A*[0...5, 0...3]** берілген. Оның теріс элементтерінің индекстерін басып шығарыңдар.
31. Екіөлшемді бүтін санды ***A*[0..10, 0..7]** жиымын құрыңдар және оның екі тақ санды индекстерінің барлық элементтерінің қосындысын табыңдар.
32. Бүтін санды ***A* [0..10, 0..7]** жиымы берілген. Жиымның 5-ке қалдықсыз бөлінетін элементтерінің қосындысын табыңдар.
33. 3х3 матрицаның бүтін сандық элементтерін пернетақтадан енгізіңдер және әрбір баған элементтерінің қосындысын есептеп шығарыңдар.
34. Бүтін санды ***B*[0..5,0..5]** жиымы берілген. Оның диагональдарынан сол және оң жақта орналасқан элементтерін анықтаңдар.
35. Бүтін санды ***B* [0..5, 0..5]** жиымы берілген. Бас диагональдың оң жақтағы элементтерінің қосындысын, сол жақтағы элементтерінің көбейтіндісін табыңдар.
36. Бүтін санды ***B* [0..5, 0..5]** жиымы берілген. Жиымның ең үлкен элементін табыңдар және оның диагоналының қай жағында орналасқаны туралы хабарды экранға шығарыңдар.
37. Бүтін санды ***B* [0..5, 0..5]** жиымы берілген. Жиымның ең кіші элементін табыңдар және оның бас диагональдің қай жағында орналасқаны туралы хабарды экранға шығарыңдар.
38. Бүтін санды ***B* [0..5,0..5]** жиымы берілген. Жиым диагоналінің сол жағынан жоғары орналасқан элементтердің қосындысын табыңдар.
39. Бүтін санды ***B* [0..5,0..5]** жиымы берілген. Жиым диагоналінің сол

жағынан төмен орналасқан элементтердің көбейтіндісін есептеп шығарыңдар.

40. Бүтін санды ***B*** **[0..5,0..5]** жиымы берілген. Жиым диагоналінің сол жағынан төмен орналасқан теріс таңбалы элементтерінің санын табыңдар.
41. Бүтін санды ***B*****[0..5,0..5]** жиымы берілген. Жиым диагоналінің сол жағынан жоғары орналасқан оң таңбалы элементтерінің санын табыңдар.
42. Бес цехтың әрқайсысының 4 бөлімшесіндегі барлық шикізат туралы мәлімет кестесі берілген. Шикізаты ең аз цехтың нөмірін анықтаңдар.
43. ***A*****[0..3, 0..15]** жиымы берілген. Оның ішінде өзара тең екі элементтің индекстерін басып шығарыңдар.
44. a_1, a_2, a_3 сандары берілген. Элементтері $\boldsymbol{B[i,j] = a_i - 3a_j}$ болатын бүтін санды ***B*** **[0..3, 0..3]** жиымын анықтаңдар.
45. Нақты a_1, a_2, ... , a_{10}, b_1, b_2, ... , b_{20} сандары берілген. Элементтері $a_{i,j} = i + 2j$ болатын бүтін санды ***A*** **[0..10, 0..12]** жиымын алыңдар.
46. Өлшемдері 5*5 матрицаның әрбір элементінің мәні қиылатын жол мен бағана нөмірінің қосындысына тең болатын элементтерінің қосындысын есептеңдер.
47. Нақты **[0..7, 0...7]** жиымын алыңдар, оның 1-жолы $a_{1j} = 2_j + 3$ формуласымен, 2-жолы $a_{2j} = j + 3/(2+j)$ формуласымен беріліп, содан кейінгі әрбір жол алдыңғы екі жолдың қосындысына тең болатын болсын.
48. Натурал *n* саны берілген. Егер $a_{i,j} = \sin(i+j/2)$ болса, ***A*****[0..*n*, 0..*n*]** жиымында қанша оң элемент болатындығын анықтаңдар.
49. Бүтін санды ***A*****[0..4, 0..5]** жиымы берілген. Әрбір бағанның арифметикалық ортасын табыңдар.
50. Барлық элементтері нөлге тең емес ***n..m*** өлшемді нақты жиым берілген. Бұл жиымның модулі бойынша ең үлкен элементіне басқа барлық элементтерін бөлу арқылы жаңа жиым алыңдар.
51. Бүтін санды ***A*****[0..4, 0..5]** жиымы берілген. Соңғы жолдан басқа жолдарды әрбір элементі бойынша азайту арқылы матрицаны түрлендіріңдер.
52. Екіөлшемді **C** жиымының әрбір жолын өсу ретімен орналастыратын программа құрыңдар.
53. ***m..n*** өлшемді матрицаның әрбір жолын кему ретімен орналастыратын программа құрыңдар.
54. Бүтін санды ***A*****[0..4, 0..5]** жиымы берілген. Құрылымында кем дегенде бір рет 10-ға тең элементі бар жолдардың нөмірін анықтаңдар.
55. ***m..n*** өлшемді матрицаның әрбір бағанасын өсу ретімен орналастыратын программа құрыңдар.
56. ***A*****[0..5, 0..5]** жиымы берілген. Бұл жиымның әрбір жолының эле-

менттерін диагональдің сол жағында орналасқан элементтерге бөлу жолымен алынған жаңа жиым құрыңдар.

57. ***A*[0..5, 0..6]** жиымы берілген. Оның бірінші және соңғы жолдарының орындарын алмастырыңдар.
58. **A[0..5, 0..6]** жиымы берілген. Оның бірінші және соңғы бағаналарының орнын алмастырыңдар.
59. Тікбұрышты матрица берілген. Элементтерінің қосындысы ең үлкен болатын жолды табыңдар.
60. Тікбұрышты матрица берілген. Элементтерінің көбейтіндісі ең үлкен болатын бағананы табыңдар.
61. Өлшемі 4x8 болатын бүтін сандар жиымындағы барлық жұп нөмірлі элементтерінің қосындысын табыңдар.
62. Бүтін сандардың 5x5 өлшемді жиымындағы бас диагональда орналасқан барлық элементтердің қосындысын табыңдар.
63. Бүтін сандардың 7x4 өлшемді жиымындағы максимал элементтің жолы мен бағана нөмірлерін табыңдар.
64. Бүтін сандардың 6x5 екіөлшемді жиымы бар. Элементтерінің арифметикалық орта мәні максимал болатын жолдың нөмірін табыңдар.
65. Бүтін сандардың 5x9 өлшемді жиымында бірдей нөмірлі жол мен бағананың орнын алмастырыңдар.
66. Бүтін сандардың екі өлшемді жиымындағы әрбір жолдың максимал элементтерінің арасындағы минимал элементті табыңдар.
67. Бүтін сандардың екіөлшемді жиымында максимал элементі бар бағананы өшіріп тастаңдар.
68. Бүтін сандардың екіөлшемді жиымындағы қайталанбайтын барлық элементтерді табыңдар.
69. Екіөлшемді жиымды 1-ден 100-ге дейінгі бүтін сандармен орама (спираль) бойымен толтырыңдар.
70. Бүтін сандардың екіөлшемді жиымының барлық элементтерін сол жолдардағы элементтердің қосындысынан солардың ішіндегі ең кіші элементтер айырмасымен алмастырыңдар.
71. Бүтін сандар жиымының жолдарын кему реті бойынша сұрыптаңдар.
72. Жиымның тақ орындардағы бағаналарында тұрған элементтерді өсу ретімен орналастыра отырып сұрыптаңдар.
73. Шеберханада шығарылған түрлі тетік бөлшектер мен олардың бағасы берілген. Осы мәліметтерді а) бағалардың өсуі және ә) тетік бөлшек атауларын алфавиттік реті бойынша сұрыптаңдар.
74. Студенттердің аты-жөні және олардың телефон нөмірлері көрсетілген екіөлшемді жиым берілген. Студенттің фамилиясы бойынша оның телефон нөмірін табыңдар.
75. Екі матрица берілген. Олардың көбейтіндісін табыңдар.

76. Екіөлшемді жиымның элементтері сиқырлы квадрат (сиқырлы квадратта барлық вертикаль, горизонталь және екі диагональ бойынша сандардың қосындысы бірдей болады) құрайтындығын тексеретін программа құрыңдар.
77. Матрицаның элементтері қосалқы диагональ бойынша симметрия құра отырып орын алмастыратын программа құрыңдар .
78. Екіөлшемді *k* жиымының бағаналарын циклді түрде, оның *i*-ші бағанасын *i* + 1 бағанасымен алмастыра отырып, соңғы бағана бірінші болып орналасатындай деңгейге жеткізетін программа құрыңдар.
79. Екіөлшемді А жиымының нөлге тең элементтері жоқ жолдарының оң элементтерінің қосындысын есептеп шығаратын программа құрыңдар.
80. Квадрат пішінді кестенің ең кіші элементін анықтап, соған сәйкес жол мен бағана элементтерінің орындарын алмастырыңдар.
81. Бүтін сандардың екіөлшемді жиымы берілген. Сол жиымның ең кіші элементі орналасқан жолы мен бағанасын жойыңдар.
82. Тікбұрышты кестенің екінші қатарынан бастап, ондағы әрбір жолдың ең кіші элементін алдыңғы жолдың ең үлкен элементімен алмастырыңдар.
83. Бүтін сандардың 10х12 өлшемді матрицасы берілген. Оның барлық ершік нүктелерінің индекстерін басып шығарыңдар. (Ершік нүкте деп өзінің жолында ең кіші, бірақ бағанасында ең үлкен немесе, керісінше, өзінің жолында ең үлкен, бірақ өз бағанасында ең кіші болатын элементті айтады).

13. СӨЗ ТІРКЕСТЕРІН ӨҢДЕУ

PASCAL тілінде сөз тіркестерін өңдеу кезінде қолданылатын арнайы тип – **string** бар. Ал С тілінде мұндай арнайы тип жоқ. Сөз тіркестері **char** типті бір өлшемді жиым ретінде қарастырылады, яғни сөз тіркесі – нөлдік байтпен аяқталатын **char** типті бір өлшемді жиым. Нөлдік байт – барлық биттері де нөлге тең байт, ол `'\0'` символдық константасымен анықталады (тіркес соңы белгісі немесе нөл-терминатор). Сондықтан егер тіркесте **k** символ болса, онда жиымды сипаттауда **k+1** элемент көрсетілуі тиіс.

Мысалы, **char a[7]** деген сипаттау тіркестің 6 символдан тұратынын, ал соңғы байт нөлдік екенін білдіреді. С тіліндегі тіркестік (жолдық) константа – қос тырнақшаға алынған символдар жиыны. Мысалы, "Берілген тапсырма" тіркесі, оның соңына нөлдік байтты компилятор автоматты түрде өзі жазады.

Айнымалы мәні болатын сөз тіркесін сипаттау кезінде бірден көрсетуге болады, мысалы,

```
char S1[10]= "123456789", S2[]="Болат";
```

Соңғы сөз ұзындығы тіркестің символдары санымен анықталады.

Символдар тіркесін пернелерден енгізу үшін екі стандартты функция – **`scanf()`** немесе **`gets()`** қолданылады, ал олардың прототиптері **`stdio.h`** тақырыптық файлында болады.

13.1 Символдық таңбаларды енгізу/шығару

Символдарды біртіндеп енгізу/шығару үшін **printf()** және **scanf()** функцияларының **%c** форматы қолданылады.

`getch()` – параметрсіз функция, басылған перненің кодын (int) береді, экранға ешқандай символ шығармайды.

`getchar()` – параметрсіз функция. Пернеден символдарды бірбірлеп енгізеді. Сөз тіркесі <**Enter**> пернесі басылғанша енгізіле береді, оған дейін оны өзгертуге де болады.

`putch(c)` – бір символды (**c** – символдық айнымалы немесе константа), яғни бір таңбаны ғана экранға шығарады.

`putchar(c)` – бұл да тек бір таңбаны экранға шығарады.

Бұлар **conio.h** тақырып файлы бойынша жұмыс істейді. Символдық таңбаларды өңдеу туралы мәліметтер **Ә** қосымшасында келтірілген.

Мысалы, латын алфавиті әріптерін экранға шығару программасы төмендегідей болады:

```
#include <conio.h>
#include <stdio.h>
void main()
{ char z;
 clrscr();
 for(z='A';z<='Z';z++)
  putchar(z);
 getch();
}
```

Нәтижесі:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Ал енді осы символдарды ASCII-кодтарымен бірге шығаратын мына программаны көрейік.

```
/* латын алфавиті*/
#include <conio.h>
#include <stdio.h>
void main()
{
 char z;
 clrscr();
 for(z='A';z<='Z';z++)
 {
  if (z=='K' || z=='U') printf("\n");
  printf(" %c-%d ",z,z);
 }
 getch();
 }
```

Программа жұмысы нәтижесі:

```
A-65 B-66 C-67 D-68 E-69 F-70 G-71 H-72 I-73 J-74
K-75 L-76 M-77 N-78 O-79 P-80 Q-81 R-82 S-83 T-84
U-85 V-86 W-87 X-88 Y-89 Z-90
```

Келесі программа 0 мен 9 арасындағы цифрлық символдарды және олардың ASCII кодтарын басып шығарады:

```
#include <conio.h>
#include <stdio.h>
void main()
{
 char z;
 clrscr();
 for(z='0';z<='9';z++)
 {
  if (z=='0' || z=='5') printf("\n");
  printf (" %c-%в ",z,z);
 }
 getch();
 }
```

Жұмыс нәтижесі:

0 – 48 1 – 49 2 – 50 3 – 51 4 – 52

5 - 53 6 – 54 7 – 55 8 – 56 9 – 57

13.2 Символдық тіркестер

Символдық жолдарды немесе тіркестерді бірнеше тәсілмен өңдеуге болады, олардың негізгілері:

1. Тіркестік константаларды қолдану;
2. **Char** типті жиымды қолдану;
3. **Char** типіне сілтейтін нұсқауыштарды пайдалану;
4. Символдық тіркестерден тұратын жиымдарды қолдану.

Сөз тіркестері немесе тіркестік (жолдық) константа қостырнақшаға алынып жазылады. Тырнақшаға алынатын символдар тізбегінің ең соңына автоматты түрде **'\0'** символы жазылады. Компилятор жолдық символдарды компьютер жадына жазғанда, олардың көлемін анықтау үшін сол символдар санын есептейді. Символдық константа осы сөз тіркесі жазылған жады аймағына сілтейтін нұсқауыш болып табылады. Символдық тіркестер жиымын (массивін) беру кезінде компилятор компьютер жадының қажетті көлемін анықтау үшін жиымды сипаттағанда, оны тіркестік константа арқылы инициалдауға болады. Мысалы:

char c[]= "Атырау, Алтай - жеріміз";

Әдеттегі жиым қолданылатын жағдайлар сияқты бұл жиым аты **c** осы жиымның 1-элементіне сілтейтін нұсқауыш болып табылады.

```
c ==&c[0];
*c =='0';
*(c+1) == c[1] == 'n';
```

Сөз тіркестерін анықтау үшін нұсқауыштарды мынадай түрде сипаттауға болады:

```
char *c1 = "\n студенттер саны";
```

осы сипаттауға эквивалентті болып келесі сипаттау есептеледі:

```
static char c1[]="\n студенттер саны";
```

Осы қарастырылған екі сипаттау да **c1** тіркесінің нұсқауыш екенін білдіреді. Компьютер жадының қажетті көлемін айқын көрсетуге де болады. Мысалы, сыртқы сипаттауда келесі жолдың мынадай түрде жазылғаны көрсетілген.

```
char c[25]="Білім - өмір шырағы";
```

Элементтердің саны жолдың ұзындығынан бір символ артық болуы керек, яғни оның ең соңында **'\0'** символы болуы тиіс.

Статикалық немесе сыртқы жиымдағы бұрынғы қарастырылған әдеттегі жиымдар оларды қолдану кезінде автоматты түрде 0-мен инициалданған болатын. Ал сөз тіркестерін пайдалану кезінде де статистикалық немесе сыртқы жиымдар солар тәрізді 0 символымен инициалданады.

Келесі мысалды қарастыралық:

```
#include <stdio.h>
#include <string.h>
 main ()
{
 char msg[30];
 strcpy(msg, "Сәлем, Азат!");
 puts(msg);
}
```

Мұндағы msg сөзінен соң тұрған **[30]** саны компиляторға **29** символ үшін, яғни char типіндегі **29** айнымалыдан тұратын жиым үшін жады бөлуді қамтамасыз етеді (30-орын нөлдік символмен – **\0** толтырылады). **msg** айнымалысының символдық мәні жоқ; ол

тек char типіндегі 29 айнымалының алғашқысының адресін (компьютер жадындағы белгілі бір орын адресі) сақтайды.

Компилятор **strcpy(msg, "Сәлем, Азат!")** операторын кездестіргенде, екі түрлі әрекет орындайды:

- "Сәлем, Азат!" тіркесі соңына (\0) символын (ASCII коды 0) қосады.

- **strcpy** функциясын орындап, **msg** айнымалысы нұсқап тұрған жады аймағына сол сөз тіркесі символдарын біртіндеп көшіреді. Ол тіркесті көшіруді сөз соңындағы нөлдік символдан кейін барып аяқтайды.

puts(msg) функциясын орындағанда, оған **msg** мәні, яғни тіркес құрамындағы бірінші символ адресі беріледі. Одан кейін **puts** сол символдың нөлдік символ емес екенін анықтап, ары қарай адреске бірді қосып, келесі символы оқиды, т.с.с. тіркес соңына дейін жетеді. Нөлдік символға жеткен соң, **puts** жұмысты аяқтайды;

Осындай тәсіл тіркес ұзындығына шек қоймай, нөлдік символға дейінгі символдарды біртіндеп оқуды жүзеге асырады.

13.3 Символға нұсқауышты пайдалану

Екінші тәсіл – символдарға нұсқауыш жасау. Программаны келесі түрге келтірейік:

```
#include <stdio.h>
#include <string.h>
 main()
{
 char *msg;
 msg = "Сәлем, Азат!";
 puts(msg);
}
```

msg алдындағы жұлдызша (*) компиляторға оның символға нұсқауыш екенін білдіреді, яғни **msg** белгілі бір символ адресін сақтай алатын айнымалы. Бірақ мұнда компилятор символдар үшін ешқандай орын бөлмейді және msg да ешқандай мәнге ие болмайды.

Компилятор **strcpy(msg, "Сәлем, Азат!")** операторын кездестіргенде, ол тағы екі түрлі әрекет орындайды:

- объектілік код файлы ішіндегі бір орынға соңына (\0) символы қосылған "Сәлем, Азат!" тіркесін (ASCII коды 0) жазып қояды.
- сол тіркестің алғашқы символы адресін msg айнымалысына меншіктейді.

strcpy функциясын орындалып, **puts(msg)** командасы бұрынғыша нөлдік символға дейінгі мәліметті көшіреді.

Енді *символдық тіркестерден тұратын жиымдарды* қарастыралық. Бұл жиымдардың әрбір жолы символдық жиым болып табылады. Мысалы, статикалық жиымның сипатталуы келесідей түрде жазылуы мүмкін:

```
static char
   *m[4]={"регистр","жады","курсор","элемент"};
```

бұл жиым символдық тіркестерге сілтейтін 4 нұсқауыш болып табылады. Сонымен, символдық тіркестер жиымдар болып табылатын болса, онда осы жиымдарға сілтейтін 4 нұсқауыш қарастырылады. 1-жолға сілтейтін 1-нұсқауыш болып **m[0]** есептеледі, **m[1]**– 2-жолға сілтейтін 2-нұсқауыш болып табылады. Сонымен, әрбір нұсқауыш соған сәйкес жолдың немесе қатардың ең бірінші символына сілтейді.

```
*m[0]=='р'; *m[1]=='ж'; *m[2]=='к'; *m[3]=='э';
```

Тіркестерден құрылған жиымдарды сипаттағанда, символдық тіркестер көлемін көрсетуге де болады және бұл сипаттауда тіркестердің ұзындығын келесідей түрде көрсетуге болады:

static char m[10];

Символдар тіркестерін енгізу/шығару үшін **printf()** және **scanf()** функцияларының **%s** форматы қолданылады.

Тағы мысалдар келтірейік.

```
/* символдық тіркесті шығару */
#include <conio.h>
#include <stdio.h>
void main()
{char b[]="Сезам, ашыл!";
 clrscr();
 printf("%s",b);
 getch();
}
```

Мұндағы **b** жиымының ұзындығы 13 символ, яғни сөздер ұзындығынан 1-ге артық.

Енді бір сөйлем енгізіп, соның соңғы сөзін экранға шығарайық.

```
#include <conio.h>
#include <stdio.h>
void main()
{
 char s,ss; // s - енгізілетін символ
 // ss - алдыңғы символ
 char a[80]; // сөз жиымы
 int i,k;  // k - сөз ені
 clrscr();
printf("Соңында нүкте бар сөйлем енгізіңдер:\n");
  for(i='0',s=' ',k=0;i<=79;i++)
 {
  ss=s; s=getchar();
  if (s==' ') continue;
  if (s=='.') break;
  if (ss==' ') k=0;
  a[k]=s; k++;
 }
  //нүктеден кейін шығу не тіркес біткесін шығу
if (i==80 || k==0)
 printf("сөйлем дұрыс емес \n");
else
{
 a[k]='\0';  // жол соңы
 printf("ең соңғы сөз: %s",a);
}
 getch();
}
```

мұнда символдар біртіндеп **getchar()** функциясы арқылы енгізіледі. Егер бос орын енгізілсе continue операторы келесі қадамға көшіреді. Нүкте енгізілсе цикл тоқталады, бірақ алыңғы **k** символда соңғы сөз сақталады. Егер символ нүкте де, бос орын да

емес болса, онда алдыңғы символ қарастырылады. Егер ол бос орын болса, онда келесі сөз енгізіліп, **k** нөлге тең болады. Циклден шығу нүкте арқылы немесе 80 символ енгізілген соң орындалады.

Келесі мысалда сөз тіркесінің ұзындығы екі тәсілмен анықталады.

```
#include <conio.h>
#include <stdio.h>
#include <string.h>

void main()
{
 char st[80];
 int i;
 clrscr();
 puts("Сөз тіркесін енгізіп, Enter басыңыз:");
 gets(st);
 i=0;
 while (st[i++])
  ;
 printf("Енгізілген тіркес ұзындығы: %i\n",i-1);

 puts("Сөз тіркесін енгізіп, Enter басыңыз:");
 gets(st);
 printf("Енгізілген тіркес ұзындығы: %i\n",
     strlen(st));
 getch();
 }
```

Енді бір сөз тіркесін енгізіп, оның ішінде **"т"** символының неше рет кездесетінін табайық.

```
#include <conio.h>
#include <stdio.h>
#include <string.h>
void main()
{
 char str[80];
 int s=0;
```

```
 clrscr();
 puts("Сөз тіркесін (сөйлем) енгізіңіз:");
 gets(str);
 for (int i=0; i<strlen(str); i++)
  if (str[i] == 'т') s+=1;
 printf("'т' символы %i рет кездеседі\n",s);
 getch();
 }
```

Ендігі мысалда енгізілген сөздің палиндром (алды-артынан оқығанда, мәні бірдей – керек, қазақ) екенін анықтайық.

```
#include <conio.h>
#include <stdio.h>
#include <string.h>

void main()
{
 char str[80];
 int k,s=0;
 clrscr();
 puts("Бір сөз (палиндром) енгізіңіз:");
 gets(str);
 k=strlen(str);
 for (int i=0; i<k/2; i++)
  if (str[i] == str[k-i-1]) s+=1;
 if (s==k/2)
 puts("Сөз - палиндром.");
 else puts("Сөз - палиндром емес.");
 getch();
 }
```

13. 4 Сөз тіркестерін енгізу функциялары scanf(), gets(str)

Scanf() функциясы тіркестік айнымалылар мәнін **%s** форматымен енгізеді, бірақ ол тіркесті тек бірінші босорын таңбасына дейін ғана енгізе алады.

Ал **gets(str)** функциясы арасында босорыны бар тіркестерді енгізеді, енгізу ENTER пернесімен аяқталады.

Екі функция да тіркес соңына нөлдік байт қосып жазады. Оның үстіне тіркес – символдық жиым болып, ал жиым аты – оның компьютер жадындағы алғашқы адресіне сілтеме болғандықтан, тіркестік айнымалы атының алдына "&" символы жазылмайды.

13.5 Сөз тіркестерін шығару функциялары cprintf(),puts(),cputs()

printf() – экранға формат арқылы сөз тіркесін шығарады;

cprintf() – экранға **printf()** сияқты формат арқылы сөз тіркесін шығарады, тек олардың түстерін **textcolor()** және **textbackground()** функциялары арқылы өзгертуге мүмкіндік береді;

puts(str) – экранға сөз тіркесін шығарып, курсорды бірден келесі жолдың басына алып барады, мұндағы **str** – тіркестік константа немесе тіркестік айнымалы. Бұлар **stdio.h** тақырып файлы бойынша жұмыс істейді.

Екі функция да символдық жиымды нөлдік байтқа дейін шығарады. **printf()** функциясы символ тіркесі шығарылған соң, курсорды келесі жолға көшірмейді, ол үшін арнайы формат (**\n**) жазылуы тиіс. Ал **puts()** функциясы символдар шығарылған соң, автоматты түрде курсорды келесі жол басына көшіреді.

```
//puts функциясын пайдалану мысалы
#include <stdio.h>
#include <conio.h>
main()
{ char str1[]= "abc";
 char str2[]= "def\nghi\n";
 char str3[]= "jkl";
 puts(str1);
 puts(str2);
 puts(str3);
}
```

Нәтижесі:

abc
def
ghi
jkl

cputs(str) – экранға сөз тіркестерін шығарып, олардың түстерін **textcolor()** және **textbackground()** функциялары арқылы өзгертуге мүмкіндік береді, **conio.h** тақырып файлы бойынша жұмыс істейді.

Сөз тіркестерімен орындалатын басқа операциялар да стандартты функциялар арқылы атқарылады. Ол функциялар жұмыс істеуі үшін **string.h** тақырыптық файлы қажет.

Жалпы сөз тіркестеріне қолдану үшін **stdlib.h** немесе **string.h** тақырыптық файлдары қолданылады.

13.6 Сөз тіркестерімен жұмыс істейтін өзге функциялар

1) **strlen(str)** функциясы **str** сөз тіркесіндегі символдар санын (соңғы нөлді есепке алмайды), яғни жолдың ұзындығын анықтайды, оның типі **int**, тақырыптық файлы <**string.h**>.

Мысалы. Бірнеше сөз тіркестерінің ұзындығын анықтайтын программа құру керек.

```
// strlen(str) функциясын пайдалану
#include <conio.h>
#include <stdio.h>
#include <string.h>
main ()
{
static char t[]="Студенттер жайлы хабарлама.";
clrscr();
printf("%d\n",strlen(t));
printf("%d\n",strlen("Студенттер жайлы
          хабарлама."));
printf("%d\n",strlen("аль-Фараби ат.КазҰУ"));
printf("%d\n",strlen(""));
getch();
}
```

Мұның нәтижесі:

27
27
19
0

2) strcat(stroka1,stroka2) функциясы қатарларды біріктіру үшін қолданылады. Ол **stroka1** және **stroka2** тіркестерін біріктіі-

ріп, нәтижені **stroka1** айнымалысына меншіктейді, **stroka2** тіркесінің мәні өзгермейді

Мысалы:

```
// strcat(str1, str2) функциясын пайдалану
#include <conio.h>
#include <stdio.h>
#include <string.h>
main ()
{
char str1[50]="С тілін оқимыз, ";
char str2[]="жақында емтихан тапсырамыз.";
clrscr();
printf("%s\n",strcat(str1,str2));
puts(str1);  // қатарды экранға шығару
puts(strcat("Егер жақсы оқысақ, ",str2));
getch();
}
```

Мұның нәтижесі:

С тілін оқимыз, жақында емтихан тапсырамыз.
С тілін оқимыз, жақында емтихан тапсырамыз.
Егер жақсы оқысақ, жақында емтихан тапсырамыз.

Келесі мысалда студенттің аты пернелерден енгізіліп, ол екінші тіркеспен біріктіріледі.

```
// strcat(str1, str2) функцияларын пайдалану
#include <conio.h>
#include <stdio.h>
#include <string.h>
main ()
{
char name[80];
char stud[]= " - КазҰУ студенті";
clrscr();
puts("атын енгіз:");
gets(name);
strcat(name,stud);
puts(name);
```

```
getch();
}
```

3) **strcmp(stroka1,stroka2)** функциясы екі сөз тіркесін салыстыру үшін қолданылады. Егер олар бірдей болса, функцияның мәні 0-ге тең болады, әйтпесе ол екі тіркестің айырмасын береді. Егер **stroka1<stroka2** болса, нәтиже 0-ден кіші, ал **stroka1 > stroka2** болса, нәтиже 0-ден артық болады. Көбінесе бұл тәсіл екі тіркестің бірдей еместігін анықтау үшін ғана қолданылады.

Мысалы:

```
main ()
{
  printf("%d\n",strcmp("Сәлем","Сәлем"));
  printf("%d\n",strcmp("Azat","Izat"));
  printf("%d\n",strcmp("Абайда","Абайла"));
  getch();
}
```

Мұның нәтижесі:

0
-8
-7

Алғашқы екі сөз бірдей, нәтижесі – 0, келесі екі сөздің алғашқы әрпі әр түрлі, олардың ASCII-кодтарының айырмасы – -8 (A - 65, I - 73), ал 3-жолы -7 (д – 164, л – 171, олардың кодтарының айырмасы 164-171 = -7).

```
// strcmp(str1, str2) функциясын пайдалану
#include <conio.h>
#include <stdio.h>
#include <string.h>
#define NAME "Ритчи"
main ()
{
 char f[20];
 puts("С тілінің авторы кім:");
 gets(f);
 while(strcmp(f,NAME)!=0)
 {puts("басқа кім болуы мүмкін:");
```

```
  gets(f);
 }
puts("Жауап дұрыс!");
getch();
}
```
Нәтижесі:
```
С тілінің авторы кім:
Керниган
басқа кім болуы мүмкін:
Ритчи
Жауап дұрыс!
```
4) strcpy(str1,str2) функциясы сөз тіркесінің көшірмесін алу үшін қолданылады, мұнда **str2** айнымалысындағы сөз тіркесі **str1** айнымалысына көшіріледі. Мысалы:
```
// strcpy(str1, str2) функциясын пайдалану
#include <conio.h>
#include <stdio.h>
#include <string.h>
main ()
{
 char str1[21];
 strcpy(str1,"Хал қалай, Азат?");
 puts(str1);
 strcpy(str1,"Тамаша!");
 puts(str1);
 getch();
}
```
Нәтижесі:
```
Хал қалай, Азат?
Тамаша!
```
2-мысал:
```
// strcpy(str1, str2) функциясын пайдалану
#include <conio.h>
#include <stdio.h>
#include <string.h>
```

```
#define stroka "көшіру функциясы"

main ()
{
 char *ptr=stroka;
 char res[25];
 clrscr();
 puts(ptr);
 puts(res);
 strcpy(res,ptr);
 puts(ptr);
 puts(res);
 getch();
}
```

Нәтижесі:

көшіру функциясы

көшіру функциясы
көшіру функциясы

Мұнда **ptr** айнымалысы **көшіру функциясы** сөзін береді, **res** айнымалысы бос жол береді, ал келесі жолы екеуі де **көшіру функциясы** сөзін береді.

5) **strstr(str1,str2)** функциясы 2-ші көрсетілген жолды 1-ші жолдың ішінен іздейді.

6) **strset(str,ch)** функциясы берілген қатардағы барлық символдарды көрсетілген символға (char ch) ауыстырады.

7) **strtod(str1,str2)** функциясы берілген қатарды **double** типті санға ауыстырады.

8) **strchr(str,c)** функциясы берілген қатардағы коды көрсетілген символдың позициясын анықтайды.

9) **strrev(str)** функциясы берілген қатардың барлық символдарын керісінше бейнелейді.

10) **strpbrk(str1,str2)** функциясы 2-ші қатардың кез келген символын 1-ші қатардан іздейді.

Сөз тіркесін өңдейтін функциялар туралы мәліметтер Б қосымшасында келтірілген.

Бақылау сұрақтары

1. *Тіркестік айнымалылардың сипатталу тәсілдері қандай?*
2. *Тіркестік айнымалы қандай идентификатормен және қалай анықталады?*
3. *Бір тіркестік айнымалыға немесе тұрақтыға қанша символ жазуға болады?*
4. *Тіркестік айнымалының ұзындығы қалай анықталады?*
5. *Тіркестік өрнектер дегеніміз не?*
6. *Тіркестік айнымалылар мен тұрақтыларға қандай амалдар қолданылады?*
7. *Тіркестің ішкі символдарын қалай бөліп алуға болады?*
8. *С тілінде сөз тіркестерін өңдейтін қандай функциялар бар? Оларды қалай пайдаланады және олар қалай жазылады?*

Тапсырмалар

1. Бір топтағы фамилиялары бірдей студенттерді табыңдар.
2. Топ студенттері фамилиялары мен аттарының бірінші әрпін шығарыңдар.
3. Әрбір сөзден кейін бір бос орын қалдырылған сөйлемдер берілген. Құрамында дауысты әріптер ең көп кездесетін сөзді табыңдар.
4. Әрбір сөзден кейін бір бос орын қалдырылған сөйлемдер берілген. Құрамында берілген әріптен басталатын сөздердің санын анықтаңдар.
5. Әрбір сөзден кейін бір бос орын қалдырылған сөйлем берілген. Сөйлемдегі ең ұзын сөзді табыңдар.
6. Әрбір сөзден кейін бір бос орын қалдырылған сөйлем берілген. Сөйлем ішіндегі жақшалардың дұрыс қойылғанын тексеріңдер.
7. Әрбір сөзден кейін бір бос орын қалдырылған сөйлемдер берілген. Сөйлем ішіндегі жақшаға алынған мәтіндерді өшіріңдер.
8. Берілген сөз тіркесіндегі әрбір нүктені көп нүктемен (яғни үш нүктемен) алмастырыңдар.
9. Алдыңғы есеп шартындағы қатар келген нүктелердің әрбір тобын бір нүктемен алмастырыңдар.
10. Дүкендегі кассир көмекшісіне арналған программа құрыңдар. Программа тауардың бағасын, мөлшерін, сатып алынған тауарлар бағасының қосындысын есептеп, сатып алушының берген ақшасының мөлшерін сұрап, оған қайтарылатын соманы да анықтайды.
11. Әрбір сөзден кейін бір бос орын қалдырылған сөйлем берілген. Барлық сөздерді керісінше жазып шығыңдар.
12. Пернетақтадан енгізілген сөздердегі әріптердің санын есептейтін программа жазыңдар. Тапсырманы do … while циклында орындаңдар.
13. Натурал n саны және S_1, S_2, ... Sn символдары берілген. Осы символдардың арасында неше рет + символы кездесетінін анықтаңдар.

14. Натурал *n* саны және S_1, S_2, ... *Sn* символдары берілген. Осы символдардың арасында * символдарының санын есептеңдер.
15. Натурал *n* саны және S_1, S_2, ... *Sn* символдары берілген. Осы символдардың арасында қандай символдар көп: + немесе * символы ма?
16. Сөз тіркесі енгізілгеннен кейін оның құрамында бір символ қалғанша, тіркесті цикл сайын бір символға қысқарта отырып, сөздердің барлық нұсқаларын экранға шығаратын программа жазыңдар.
17. Енгізілген сөз тіркесіндегі сөздердің санын анықтайтын программа жазыңдар. Бір сөз екіншісінен бір бос орын арқылы айырылады деп санау керек.
18. Мәтіні пернетақтадан енгізілген телеграмманың бағасын есептейтін программа жазыңдар.
19. Берілген сөзде бірінші және соңғы әріптердің қайсысы көп кездесетінін анықтайтын программа жазыңдар.
20. "*а*" әрпімен аяқталатын атау септігіндегі зат есім берілген. Осы сөзді септеп, басып шығарыңдар.
21. Берілген сөздің жұп нөмірлі орындарында қанша "*о*" әрпі бар екендігін анықтайтын программа жазыңдар.
22. Студенттің фамилиясы, есімі және әкесінің аты бос орындармен бөлініп берілген. Студенттің аты-жөнінің инициалдарын (алғашқы әріптерін) басып шығаратын программа жазыңдар.
23. Сөз тіркесіндегі *а* әрпін өшіретін программа жазыңдар.
24. Мәтіндегі соңғы әріппен бірдей әріптерді жоятын программа жазыңдар.
25. *Z, X* сөздері берілген. *Z* сөзінде кездесетін барлық әріптерді *X* сөзінен өшіріп тастайтын программа жазыңдар.
26. Берілген сөздегі әр түрлі әріптердің санын есептейтін программа жазыңдар.
27. Сөздердің әрбір үшінші әрпін жоятын программа жазыңдар.
28. Берілген мәтіндегі "*Айна*" сөзін "*Асыл*" сөзіне өзгертетін программа жазыңдар.
29. Пернетақтадан енгізілген символды жазылған сөз тіркесінен өшіретін программа жазыңдар. Өшіру процесін жеке функция етіп қарастырыңдар.
30. Берілген мәтіндегі кездесетін "*а*" әрпін "*о*" әрпіне ауыстырыңдар.
31. Енгізілген сөз тіркесін керіге айналдыратын, яғни символдарды кері тәртіпте орналастыратын программа құрыңдар.
32. Енгізілген сөздің палиндром болатынын/болмайтынын анықтайтын программа құрыңдар.
33. Пернетақтадан енгізілген символдарды ASCII кестесінде нөмірлерінің өсу реті бойынша сұрыптайтын программа жазыңдар.

34. Бос орындармен бөлініп жазылған үш сөзден тұратын сөйлемдегі ең қысқа сөздің ұзындығын есептеп шығаратын программа жазыңдар.
35. Экранға жылжымалы жолды шығаратын программа жазыңдар.
36. Сөйлемдегі жүйелі жақшаға алынған мәтіндерді және жақшаның өзін өшіретін программа құрыңдар.
37. Ұзындығы 25 символдан артпайтын сөз тіркесін алып, одан мүмкіндігінше бірнеше жаңа сөз құрастырыңдар.
38. Сөз тіркесіндегі кездесетін "а" әрпін "ә" әрпімен ауыстыратын программа құру керек.
39. Пернелерден енгізілген сөздің ұзындығын анықтайтын программа жазу қажет. Программаны while do циклі арқылы ұйымдастырып, программа жұмысын аяқтауды '999' тіркесін енгізу арқылы жүзеге асыру керек.
40. Берілген тіркес құрамындағы сөздер бос орын арқылы бөлініп жазылған деп есептеп, олардың ішіндегі ең ұзын сөзді табу керек.
41. Берілген мәтінде өзің қалап алған сөз қанша рет кездесетінін анықтау керек.
42. Берілген мәтін сөздерінде ең көп кездесетін әріпті табу қажет.
43. Берілген мәтін сөздерінің ең жиі ұшырасатын алғашқы әрпін анықтау керек.
44. Берілген сөз тіркесіндегі "а" әрпінің санын анықтайтын программа құрыңдар.
45. Берілген мәтін сөздері арасындағы бос орынды үтірмен алмастырыңдар.
46. Берілген мәтіннің сөздері бір-бірінен бірнеше бос орын таңбасымен бөлініп жазылған, сол сөздердің арасына тек бір бос орын таңбасын қалдырып, қайта жазып шығыңдар.
47. Берілген мәтіннің неше сөзден тұратынын табу керек (сөздер арасындағы бос орын тұратынын пайдаланыңдар).
48. Берілген мәтінде "ас" тіркесі қанша рет кездеседі?
49. Берілген мәтіндегі "а" әрпімен аяқталатын сөздерді экранға шығару керек.
50. Енгізілген сөз тіркесі сан болатынын немесе болмайтынын анықтау қажет
51. Енгізілген үш сөздің ішіндегі ең қысқасының неше символдан тұратынын анықтаңдар.
52. Енгізілген мәтіннің жұп орында тұрған сөздерінің ішінде қанша 'е' әрпі кездесетінін табыңдар.

53. Енгізілген сөзде қандай әріптер қанша рет кездесетінін анықтау керек.
54. Сөзбен енгізілген сан аттарын цифр түрінде жазып шығатын программа құрыңдар.
55. Сөз тіркесі түрінде санмен енгізілген мәліметті цифрсыз жазылған сөз тіркесіне айналдырыңдар, мысалы, '56' – 'елу алты', '-125' – 'минус бір жүз жиырма бес', т.с.с.

Есептер

1 нұсқа

1. Құрамына сандар кіретін сөз тіркесінің ұзындығын – L анықтап, егер L жұп сан болса, онда тіркестегі барлық екілік сандарды өшіріп тастаңдар.
2. Берілген сөз тіркесінің дәл ортасында тұрған сөзді керісінше жазып шығыңдар.

2 нұсқа

1. Берілген сөз тіркесінің ұзындығын – L анықтап, егер L жұп сан болса, онда тіркестегі алғашқы сөзді өшіріңдер, ал тақ сан болса, соңғы сөзді өшіру керек.
2. Берілген сөз тіркесінде палиндром сөз бар екенін анықтап, ол жайлы мәлімет беру керек.

3 нұсқа

1. Берілген сөз тіркесінің ішінде ДОС сөзіне кіретін символдардың нешеу екенін анықтаңдар.
2. Берілген сөз тіркесінің ұзындығын – L анықтап, егер L >10 болса, соңғы сөзді өшіріп тастаңдар.

4 нұсқа

1. Берілген сөз тіркесінің ұзындығын – L анықтап, егер L тақ сан болса, онда тіркестің дәл ортасындағы символды анықтау керек, ал жұп болса – тіркес ортасындағы екі символды анықтау керек.
2. Берілген сөз тіркесінің ішіндегі барлық ! белгісін ? белгісіне алмастырыңдар.

5 нұсқа

1. Берілген тіркес ішіндегі бос орын таңбаларын астын сызу (_) таңбасына алмастырыңдар.
2. Берілген сөз тіркесінің ұзындығын – L анықтап, егер L >10 болса, соңғы сөзді өшіріп тастаңдар.

6 нұсқа

1. Берілген сөз тіркесінің ұзындығын – L анықтап, егер L 3-ке қалдықсыз бөлінетін болса, тіркестегі екінші сөзді өшіріп тастаңдар.
2. Берілген сөз тіркесінің ұзындығын – L анықтап, егер L 5-ке қалдықсыз бөлінетін болса, тіркестегі барлық жақша түрлерінің санын анықтаңдар.

7 нұсқа

1. Екі сөйлемнен тұратын сөз тіркесінің сөйлемдерінің орындарын алмастырыңдар.
2. Берілген сөз тіркесінің ұзындығын – L анықтап, егер L 3-ке қалдықсыз бөлінетін болса, тіркестегі екінші сөзді өшіріп тастаңдар.

8 нұсқа

1. Берілген сөз тіркесінің алғашқы нүктесіне дейінгі символдарды "с" әрпіне алмастырыңдар.
2. Берілген сөз тіркесіндегі бірінші және соңғы сөзді керісінше жазып шығыңдар.

9 нұсқа

1. Берілген сөз тіркесінің алғашқы сөзі мен екінші сөзін керісінше жазып шығыңдар.
2. Бірнеше сөйлемнен тұратын сөз тіркесіндегі екінші сөйлем ішіндегі "Е" әрпінің санын анықтаңдар.

10 нұсқа

1. Берілген сөз тіркесінде жақшалар бар. Алғашқы жақшалар ішіндегі сөзді анықтаңдар.
2. Берілген сөз тіркесіндегі ең ұзын сөздің енін тауып, оны керісінше жазып шығыңдар.

11 нұсқа

1. Сөз тіркесі берілген. Оның ішіндегі леп белгілерін нүктемен алмастырып, нүктелер санын анықтаңдар. Леп белгісі жоқ болса, ол туралы мәлімет беріңдер.
2. Құрамына сандар кіретін сөз тіркесінің ұзындығын – L анықтап, егер L жұп сан болса, онда тіркестегі барлық екілік сандарды өшіріп тастаңдар.

12 нұсқа

1. Сөз тіркесі берілген. Оның ішіндегі нүктелерді үш нүктемен алмастырып, нүктелер санын анықтаңдар, нүкте жоқ болса, ол жайлы мәлімет беріңдер.
2. Берілген сөз тіркесінің ұзындығын – L анықтап, егер L тақ сан болса, онда тіркестің соңғы сөйлемін анықтау керек, ал жұп болса – тіркес ортасындағы символды анықтау керек.

13 нұсқа

1. Сөз тіркесі берілген. Оның ішінде үтірлер бары белгілі. Алғашқы үтір мен соңғы үтірдің қай позицияда тұрғанын және оларды арасында неше символ бар екенін анықтаңдар.
2. Бірнеше сөйлемнен тұратын сөз тіркесіндегі соңғы сөйлем ішіндегі "А" әрпінің санын анықтаңдар.

14 нұсқа

1. Сөз тіркесі берілген. Оның ішінде нүктелер бары белгілі. Сол тіркесте неше нүкте бар екенін және бірінші нүктемен екінші нүкте арасында неше символ орналасқанын анықтау керек.
2. Берілген сөз тіркесінің ұзындығын – L анықтап, егер L жұп сан болса, онда тіркестегі алғашқы сөзді өшіріңдер, ал тақ сан болса, соңғы сөзді өшіру керек.

15 нұсқа

1. Сөз тіркесі берілген. Оның ішінде қатар орналасқан бірдей символдар бар екенін анықтау керек. Ондай символдар жоқ болса, ол туралы мәлімет беру керек.
2. Берілген сөз тіркесінің ішінде ДОС сөзіне кіретін символдардың нешеу екенін анықтаңдар.

16 нұсқа

1. Берілген сөз тіркесінде неше бос орын бар екенін анықтап, сол тіркестегі ең ұзын сөздің енін табу керек.
2. Берілген сөз тіркесінде "ба" сөзі неше рет кездесетінін анықтау керек, ондай тіркес жоқ болса, ол туралы мәлімет беріңдер.

17 нұсқа

1. Берілген сөз тіркесінде неше арифметикалық амалдар таңбасы бар екенін табыңдар.
2. Берілген сөз тіркесінің ұзындығын – L анықтап, егер L тақ сан болса, онда тіркестің дәл ортасындағы символды анықтау керек, ал жұп болса – тіркес ортасындағы екі символды анықтау керек.

18 нұсқа

1. Берілген сөз тіркесінде неше нүкте бар екенін және нүктелер арасында неше символ орналасқандарын анықтау керек.
2. Берілген сөз тіркесінің ішіндегі барлық ! белгісін ? белгісіне алмастырыңдар.

19 нұсқа

1. Берілген сөз тіркесіндегі барлық "А" әріптерін алып тастаңдар да, "Е" әріптерінің орнына "Э" әріптерін жазып шығыңдар.
2. Берілген сөз тіркесінің ұзындығын – L анықтап, егер L 5-ке қалдықсыз бөлінетін болса, тіркестегі барлық жақша түрлерінің санын анықтаңдар.

20 нұсқа

1. Берілген сөз тіркесіндегі екінші сөйлемді экранға шығарыңдар және оның неше символы бар екенін анықтаңдар.
2. Берілген сөз тіркесінің ұзындығын – L анықтап, егер L 5-ке бөлінетін сан болса, онда тіркестің дәл ортасындағы символды анықтау керек, ал жұп болса – тіркес ортасындағы сөзді анықтау керек.

21 нұсқа

1. Берілген сөз тіркесіндегі (кириллица) бас әріптерді кіші әріптерге айналдырыңдар.
2. Берілген сөз тіркесінің ұзындығын – L анықтап, егер L 3-ке бөлінетін сан болса, онда тіркестің бірінші сөйлемін экранға шығарыңдар, әйтпесе тіркестің дәл ортасындағы символды анықтау керек.

22 нұсқа

1. Берілген сөз тіркесінің ішіндегі бос орын таңбаларын астын сызу (_) таңбасына алмастырыңдар.
2. Берілген сөз тіркесіндегі "Г" әріптерінің барлығын "G" латын әріптеріне айналдырыңдар.

23 нұсқа

1. Берілген сөз тіркесіндегі (кириллица) кіші әріптерді бас әріптерге айналдырыңдар.
2. Екі сөйлемнен тұратын сөз тіркесінің сөйлемдерінің орындарын алмастырыңдар.

14 ҚОЛДАНУШЫ АНЫҚТАЙТЫН МӘЛІМЕТТЕР ТИПТЕРІ МЕН ҚҰРЫЛЫМДАРДЫ ПАЙДАЛАНУ

Практикалық есептерде өңделуге тиіс мәліметтердің құрылымы күрделі де болуы мүмкін. Соларды айқын, әрі анық бейнелеу үшін қарапайым типтер, жиымдар және нұсқауыштар негізінде жасалған мәліметтер типтері қолданылады. C++ тілі программалаушыға өз типін анықтап, солармен жұмыс істеу ережелерін де анықтауға мүмкіндік береді. Бұлар қолданушы немесе программалаушы анықтайтын типтер деп аталып жүр.

14.1 Типтердің атын өзгерту (typedef)

Программа жұмысын түсінуді жеңілдету үшін, **typedef** түйінді сөзі арқылы типке жаңа ат беріледі:

```
typedef тип жаңа_ат [ өлшемі ];
```

typedef сөзі тип спецификаторын анықтайды. Мұндай сипаттама (хабарлау) қолданушы анықтаған типтерге қысқа және мағыналы аттар беру үшін қолданылады. Мысалы:

```
typedef char name [100];
```

мұндағы тік жақша синтаксис элементі болып табылады. Өлшемі көрсетілмеуі де мүмкін. Мысалдар:

```
typedef unsigned int UINT;
typedef char sg[10];
```

Осылай енгізілген атаулар стандарттық типтердің аттары сияқты қолданылады:

```
UINT i, j;// unsigned int типті 2 айнымалы
sg a;     // символдардан тұратын a[10] жиымы
```

Тағы бір мысал:

```
typedef char Massiv[20];
typedef struct {
char fio[30];
int date, code;
double zarplata;} Worker;
```

Осылай енгізілген тип атын кейіннен пайдалануға болады:

```
Massiv str[10]; /* әрқайсысы 20 символдан
тұратын 10 жолы бар массив */
```

```
Worker sotrudniki[50]; /* 50 құрылымнан тұра-
тын массив */
```

14.2 Тізбелер (перечисления – enum)

Программа жазу кезінде бір типтегі әр түрлі мәндер қабылдай алатын шамаларды анықтау да кездесіп жатады. Ол үшін тізім түріндегі мәліметтер типін – тізбелерді қолданған жөн. **Enum** түйінді сөзі арқылы жасалған тізбелер программаны оқып түсінуді жеңілдетеді. Тізбелерге кіретін константалар **int** типінде болады. Келісім бойынша бірінші константа 0-ге тең, келесісі – 1-ге, сонан соң – 2-ге, т.с.с. бола береді. Бір тізбедегі барлық константалардың аттары әр түрлі болуы тиіс. Тізбенің жазылу форматы:

```
enum [тип_аты] {константалар_тізімі};
```

Тип аты программада осы типтегі айнымалыларды анықтау керек болған жағдайда беріледі. Компилятор осындай айнымалылардың константалар тізіміндегі мәндерді ғана қабылдауын қамтамасыз етеді. Константалар бүтін санды типте болуы тиіс және олар алғашқы мәндерді кәдімгідей түрде қабылдауы (инициалдануы) керек. Инициалдану болмаса, бірінші константа нөл болып саналады да, келесісі алдыңғысынан бірге артық мән қабылдайды. Мысалы:

```
enum {two = 2,three,four,ten = 10,eleven,
fifty = ten + 40};
```

мұнда **three** және **four** константаларына 3 және 4 мәндері, ал **eleven** константасына – 11 мәні меншіктеледі.

```
enum color {r,g,b}; // r=0, g=1, b=2
enum color {r=2,g=4,b=6}; // типті жариялау
                          // және инициалдау
enum color {r,g=3,b}; // r=0, g=3, b=4
```

Компилятор **int** және **enum** типтерінің арасында айырмашылық бар деп есептемейді, сондықтан тізбе типіндегі айнымалыларға программада бүтін санды мәндер меншіктеле береді. Бірақ C++ тілінде мұндай меншіктеу типті нақты түрде келтіру арқылы орындалады, әйтпесе компилятор ескертпе жасайды. Мысалы:

```
enum weekdays { /* жұмыс күндер: */ Monday,
Tuesday, Wednesday, Thursday, Friday};
 /* Monday – 0 , Tuesday – 1, т.с.с. */
weekdays today; today = (weekdays)1;
```

Тізбедегі константалар аттары бірегей (қайталанбайтын) болуы тиіс, ал мәндері бірдей де бола береді. Тізбелерді қолданудың жай константаларды және **#define** директивасын пайдаланудан артықшылығы байланысқан константалардың көрнекілігінде жатыр және компилятор константаларға бастапқы мән бергенде олардың типін тексере алады.

Келесі программада С тілінде тізбемен жұмыс істеу мысалы көрсетілген.

```
#include <stdio.h>
enum Months { /* айлар */
January = 1, February, March, April, May, June,
July, August, September, October, November, De-
cember } months;
/* мұнда тізбе түрінде 1-ден 12-ге дейінгі сан
тізбегі алынған, өйткені January айнымалысына 1
меншіктелген */
void main ()
{int present_month;
int diff;
/* үстіміздегі ай нөмірін (1 ... 12) енгізу */
printf ("Input the present month number (1-12): ");

scanf ("%d",&present_month);
months = December;
diff = (int) months - presentjnonth;

/* жыл соңына дейін ... ай қалды */
printf ("There are %d months till the end of the
year\n", diff);
}
```

Программаны орындау нәтижесі:

```
Input the present month number (1-12):  5
```

```
There are 7 months till the end of the year
Press any key to continue
```

14.3 Құрылымдарды пайдалану

С тіліндегі жиымдар бір типтегі мәліметтерді сақтайтын болса, құрылымдар өзара логикалық байланысқан әртүрлі типті мәліметтерді байланыстырады.

Мысалы, бір жиымда 50 қызметкердің жалақысын сақтай аламыз. Егер солармен байланыстыра отырып, қызметкерлердің аты-жөнін, жасын, реттік (табельдік) нөмірін сақтайтын болсақ, мынадай мәліметтер типтерін жазуға тура келеді:

```
char name;                    // фамилиясы
int age;                      // жасы
float salary;                 // жалақысы
unsigned employee_number;     // реттік нөмірі
```

Мұндайда байланысқан әртүрлі мәліметтерді бір атаумен сақтау мүмкіндігін беретін құрылымды пайдаланады. Мысалы:

```
struct Employee {
char name[64];                // фамилиясы
int age;                      // жасы
float salary;                 // жалақысы
unsigned employee_number;     // реттік нөмірі
};
```

Бұл құрылымның аты **Employee** идентификаторы болып табылады. Оны құрылымның *мәгі* деп, ал оның ішкі элементтерін құрылым *өрістері* деп те атайды. Өрістер кез келген типте немесе соларға нұсқауыш түріндегі құрылымның адресі арқылы қатынасу амалы ретінде де бола береді.

14.4 Құрылымдарды сипаттау

Мәліметтердің құрылымдық типтері келесі сипаттаумен анықталады:

struct құрылым_аты
{ элементтерді сипаттау; };

Құрылымның атауын, яғни тәгті жазу міндетті емес. Егер құрылым атаусыз болса, онда оны сипаттау кезінде осы типке кіретін бірнеше айнымалылардың аты бірден көрсетіледі:

```
// Құрылымдар жиымын және құрылымға нұсқауышты
анықтау
struct {
char name[64];              // фамилиясы
int age;                    // жасы
float salary;               // жалақысы
unsigned employee_number;   // қызметкердің
                            реттік нөмірі
} staff[50], *ps; /* бұл құрылымды жариялаудан
                кейінгі айнымалыларды анықтау */
```

Құрылымдар типіндегі айнымалыларды сипаттау екі жолмен жүзеге асырылады. Алғашқы тәсіл:

```
struct Shape {  /* фигуралар */
int type; // 0 - шеңбер, 1 - квадрат, 2 - үшбұрыш
int color;      // түсі
float radius;   // радиусы
float area;     // ауданы
float perimeter; // периметрі
};
Shape new_shape, old_shape;
```

Екінші тәсіл:

```
struct Shape {  /* құрылым аты – фигуралар */
int type; // 0 - шеңбер, 1 - квадрат, 2 - үшбұрыш
int color;      // түсі
float radius;   // радиусы
float area;     // ауданы
float perimeter; // периметрі
} new_shape, old_shape;
```

немесе мынадай түрде де беруге болады:

```
struct gr      /* құрылым аты – топ */
{char fio[30]; /* құрылым элементі – аты-жөні */
 char fak[25];/* құрылым элементі */
 int nomer;  /* құрылым элементі */
}
gruppa1;    /* құрылымдық айнымалы аты */
struct gr gruppa2; /* құрылымдық айнымалыны ха-
барлау */
```

Құрылымға компьютер жадынан орын бөлу үшін құрылымдық айнымалыны сипаттау керек:

struct құрылым_аты айнымалы_аты;

Мұндайда құрылым жарияланған соң оның атын бірден пайдалануға болады (анықтауды кейінірек беруге болады), бірақ мұндай тәсіл компиляторға құрылым мөлшерін білу қажет етілмейтін кездерде қолданылады:

```
struct First;
struct Second {
First *p;
Second *prev, *suc;
}; struct First { /* First құрылымын анықтау */ };
```

Бұл құрылымның байланысқан тізімдерін құру мүмкіндігін береді.

Құрылымды инициалдаудың (құрылым өрістеріне мән берудің) бір жолы – оның элементтерін сипатталу реттілігімен жүйелі жақша ішінде біртіндеп беру болып табылады:

```
struct Employee {
char name[64];              //   фамилиясы
int age;                    //   жасы
float salary;               //   жалақысы
unsigned employee_number; //   реттік нөмірі
} new_employee = {"Kadyrov", 32, 300.5, 1122};
```

Құрылымдағы жиымдарды инициалдау кезінде олардың элементтерін жүйелі жақшалар арқылы бөліп, құрылым өрістеріне мән беруге болады, мысалы:

```
struct complex{
unsigned code;
float salary;
} comp [2][3]={ {{1021, 301}, {1031, 289}, {1041,
250}},{{1121, 300}, {1131, 222}, {1141, 220}}};
```

немесе төмендегідей түрде де инициалдау мүмкіндіктері бар:

```
struct date { int day, month, year; };
d[5]={  {1,3,1980},
        {5,1,1990},
        {1,1,2002} };
```

Құрылымдарды анықтау барысында олардың элементтеріне бастапқы мәндерді пернелерден енгізіп те меншіктеуге болады. Құрылым элементтерінің мәндерін енгізу үшін ағымдық енгізу операторы ***cin***>> (C++ стилінде) немесе форматпен енгізу операторы – ***scanf*** қолданылады.

Егер құрылымды сипаттау жолы программадағы барлық функциялардың алдында орналасса, онда ол құрылымды осы программадағы барлық функциялар пайдалана алады.

14.5 Құрылым өрістерін пайдалану

Құрылымның элементін, яғни өрістерін пайдалану үшін нүкте (.) қою жолымен (тікелей ену) немесе -> таңбалары арқылы нұсқауыш бойынша ену амалы қолданылады, мысалы:

```
struct fruit {
char name[15] ;
int calories;
};
struct vegetable {
char name[15];
int calories;
};
fruit a;
vegetable b;
```

Құрылымды осылай жариялап алған соң оның өрістерін **a.calories** және **b.calories** деп, келесісін **a.name** және **b.name** деп пайдалана береміз.

Енді бір программаны толығырақ қарастырайық.

14.1 мысал. **new_employee** атты құрылым өрістерін пайдалану

```
#include <iostream.h>
struct Employee {
char name[64];
int age;
float salary;
unsigned employee_number;
```

```
} new_employee = {"Kadyrov", 32, 300.5, 1122};
void main()
{
cout << new_employee.name << endl <<
 new_employee.age  << endl <<
 new_employee.salary << endl <<
 new_employee.employee_number << endl;
}
```

Программа жұмысы нәтижесі:

```
Kadyrov
32
300.5
1122
```

Тағы бір мысал:

```
/* ойын карталарын сипаттайтын құрылымды анықтау
*/
enum suit {clubs, diamonds, hearts, spades};
      /* шыбын, қиық, табан, қарға */
typedef int pips;/* pips (ұпайлар) int синонимі
ретінде */
struct card {
  suit s;
  pips p; /* 1-ден 13-ке дейін сандар реттілігі
Тұз,2,3,...,10,Валет,Дама,Корольдерге сәйкес
келеді */
};
```

Келесі анықтаулар **card** типіндегі **m1**, **m2** идентификаторларына орын бөледі:

```
card m1,m2;
```

Құрылымның **m1** және **m2** өрістерін пайдалану үшін нүкте арқылы жазылатын таңдау жолын қолданамыз, **m1** өрісіне екілік қиық, ал **m2** өрісіне қарғаның дамасын тағайындағымыз келеді делік:

```
m1.s = diamonds;
```

```
m1.p = 2;
m2.s = spades;
m2.p = 12;
```

Келесідей анықтаулар мен меншіктеу амалдарын қарастырайық:

```
card cd, *pc = &cd;
cd.p = 2;
cd.s = spades;
```

Құрылым элементтерін пайдалануды нұсқауыш түріндегі -> операторы арқылы да орындауға болады.

Құрылым элементтерін пайдалану *14.1-кесте*

Өрнек	Оның басқаша жазылуы	Мәні
`cd.p`	`pc -> p`	`2`
`cd.s`	`pc -> s`	`spades(3)` `enum` арқылы шығады.
`(*pc).s`	`pc -> s`	`spades(3)`

Құрылымдар объектіге бағытталған программалаудың іргелі ұғымдарының бірі – кластар құру негізі болып саналатын маңызды мәліметтер типіне жатады.

14.6 Құрылымдар жасау

Құрылымдар жасауға толығырақ бір мысал келтірейік

14.2 мысал. Қайық атын, шыққан жылын және бағасын құрылым түрінде жазу

```
/* struct_2
Құрылым жасау мысалы */
#include <stdio.h>

# define STR15 16 /* аяқтау нөлі үшін бір символ
қосу */

struct stboat         // Қайық құрылымы
{ char model[STR15]; // моделі
```

```
int year;                    // шығарылған жылы
float price;                 // бағасы
} boat;                      // қайық
void main()
{ /* қайық моделін енгізу */
printf("\n Input a model of the boat: ");
gets(boat.model);
/* шығарылған жылын енгізу */
printf("\n Input the year of the creation of the
boat: "); scanf("%d",&boat.year);
/* бағасын енгізу */
printf("\n Input the price of the boat: ");
scanf("%f",&boat.price);
/* барлық енгізілген мәліметтерді шығару */
printf("\n\nThe boat %s, the year of the cre-
ation %d\n", boat.model, boat.year);
printf("The boat was sold for %8.2f tenge.\n",
boat.price);
}
```

Программаны орындау жұмысының нәтижесі:

```
Input a model of the boat: Tulpar
Input the year of the creation of the boat: 2009
Input the price of the boat: 168000
The boat Tulpar, the year of the creation 2009
The boat was sold for 168000.00 tenge.
```

Құрылымды функция аргументі ретінде беру тәсілін қарастырайық. Мұнда құрылым аргумент ретінде мәні бойынша функцияға беріледі, яғни функцияға тек мәліметтер көшірмесі ғана жіберіледі.

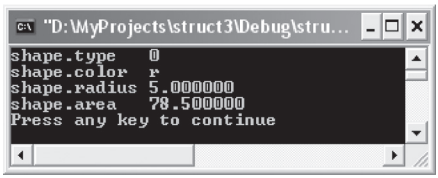
14.3 мысал. Геометриялық фигуралар түрлерін құрылым түрінде жазу

```
/* struct_3
Құрылымды функция аргументі ретінде қолдану */
#include <stdio.h> #define PI 3.14
struct Shape {  /* фигура */
int type;              // 0 - шеңбер, 1 - квадрат,
2 - үшбұрыш
```

```
char color;      // түсі
float radius;    // радиусы
double area; // ауданы
};
void show_struct (Shape ); /* show_struct
(құрылымды көрсету) функциясы прототипі */
void main()
{
Shape circle;
circle.type = 0;
circle.color = 'r';
circle.radius =5.0;
circle.area = PI * circle.radius * circle.ra-
dius;
show_struct (circle ); /* функцияны шақыру */
}
/* функцияны сипаттау */
void show_struct (Shape shape)
{
printf("shape.type  %d\n", shape.type);
printf("shape.color %c\n", shape.color);
printf("shape.radius %f\n", shape.radius);
printf("shape.area  %f\n", shape.area);
 }
```

Программа жұмысының нәтижесі (14.1-сурет):

```
shape.type 0
shape.color r
shape.radius 5.000000
shape.area 78.500000
Press any key to continue
```



14.1-сурет

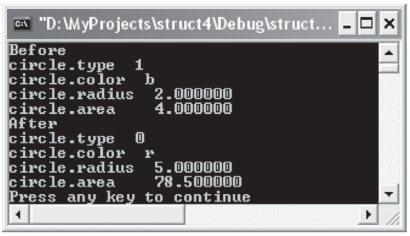
Құрылымдарға нұсқауыш ретінде берілетін мәліметтерді қарастырып, функция ішінде құрылымды өзгерту тәсіліне программа құрып көрейік.

14.4 мысал. Геометриялық фигуралар түрлерін құрылым түрінде жазу

```
/* struct_4
Құрылым өрістерінің мәндерін өзгерту керек
болса, онда құрылым ретіндегі айнымалы мән
бойынша емес, сілтеме бойынша берілуі тиіс.
Функция ішінде құрылым өрістерін пайдалану ->
амалы арқылы орындалады */
#include <stdio.h>
#define PI 3.14
struct Shape {
int type; /* 0 - шеңбер,1 - квадрат,2 – теңбүйірлі
үшбұрыш */
char color;    // түсі
float radius;  // радиусы немесе қабырғасы
double area;   // ауданы
} circle = {1,'b',2,4}; /* бастапқы мән беріп
инициалдау */
void change_struct (Shape *) ; /* change_struct
функциясы прототипі (ағылш. құрылым өзгерту) */
void main ()
{ /* функцияға мәлімет беруге дейін */
printf("Before\n");
printf("circle.type %d\n", circle.type);
printf("circle.color %c\n", circle.color);
printf("circle.radius %f\n", circle.radius);
printf("circle.area  %f\n", circle.area);
change_struct (&circle ); /* функцияны шақыру */
/* функцияға мән бергеннен кейін */
printf("After\n") ;
printf("circle.type %d\n", circle.type);
printf("circle.color %c\n", circle.color);
printf("circle.radius %f\n", circle.radius) ;
printf("circle.area  %f\n", circle.area);
}
/* функцияны сипаттау */
void change_struct (Shape *shape)
{
shape -> type = 0;
```

```
shape -> color = 'r';
shape -> radius =5.0;
shape -> area = PI * shape -> radius * shape ->
radius;
}
```

Программа жұмысы нәтижесі (14.2-сурет):



14.2-сурет

```
Before
circle.type 1
circle.color b
circle.radius 2.000000
circle.area  4.000000
After
circle.type  0
circle.color r
circle.radius 5.000000
circle.area  78.500000
Press any key to continue
```

Құрылымдардың өрістерінің өздері де құрылым бола алады. Енді соған бір мысал келтірейік.

14.5 мысал. Қызметкерлер туралы мәліметтерді құрылым түрінде жазу

```
struct Employee { /* "Қызметкер" құрылымы */
char name[64];        // фамилиясы
int age;              // жасы
unsigned employee_number; // реттік нөмірі
struct Date {         // датасы
int day;              // күні
int month;            //айы
int year;             //жылы
} hiredate;           //жұмысқа алынған мерзімі
 float salary;        //жалақысы
} new_employee;       //жаңа қызметкер
```

Ішкі құрылым элементін пайдалану үшін **.** (нүкте) операциясы қолданылады. Алдымен сыртқы құрылым элементі, сонан соң ішкі құрылым элементі көрсетіледі. Мысалы:

```
new_employee.hire_date.month = 12;
```

Құрылымдар жиымын құруға да бір мысал келтірейік.

14.6 мысал. Құрылымдар жиымы

```
/* struct_6  Құрылымдар жиымын құру */
#include <stdio.h>
#define STR15 16
#define MAX 50
struct boat  // "қайық" құрылымы
{
char model[STR15];
int year;
float price;
};
void main ()
{ int i, k;
boat Boats[MAX];
/* Қанша қайық бар? */
printf("How many boats? ");
scanf("%d", &k);
for (i = 0; i < k; i++)
{
_flushall() ; /* буферді тазалау алдыңғы
scanf() функциясын орындаудан қалған кіріс
ағынындағы жаңа жолға көшу символын алып
тастау үшін керек */
/* қайық моделін енгізу */
printf("\n Input a model of the boat: ");
gets(Boats[i].model);
/* қайықтың шыққан жылын енгізу */
printf("\n Input the year of the creation of
the boat: ");
scanf("%d",&Boats[i].year);
/* қайықтың бағасын енгізу */
printf("\n Input the price of the boat: ");
scanf("%f",&Boats[i].price);
}
printf("\n\n");
/* циклде барлық қайықтар туралы мәлімет
```

```
шығару */
for (i = 0; i < k; i++)
 {
 printf("The boat %s, the year of the creation
%d\n",
    Boats[i].model, Boats[i].year);
 printf("This boat was sold for %8.2f $.\n",
    Boats[i].price);
 }
}
```

Программаны орындау нәтижелері:

```
How many boats? 3
Input a model of the boat: Guldyz
Input the year of the creation of the boat: 2006
Input the price of the boat: 142200
Input a model of the boat: Tulpar
Input the year of the creation of the boat: 2009
Input the price of the boat: 162000
Input a model of the boat: Superi
Input the year of the creation of the boat: 2011
Input the price of the boat: 284000
The boat Guldyz, the year of the creation 2006
This boat was sold for 142200.00 tenge.
The boat Tulpar, the year of the creation 2009
This boat was sold for 162000.00 tenge.
The boat Superi, the year of the creation 2011
This boat was sold for 284000.00 tenge.
```

14.7 Құрылым жиымдарын функция аргументі ретінде пайдалану

Егер құрылымдар орнына оларға арналған нұсқауышты жазатын болсақ, онда құрылымдар функцияларға мәні бойынша беріледі де, программаның орындалуы жылдамдайды, өйткені мұндайда бастапқы мәліметтердің көшірмелері жасалмайды. Келесі мысалда құрылымдар орнына нұсқауышты пайдалану көрсетілген.

```
// struct_5
құрылымдар орнына оған арналған нұсқауышты
қолдану мысалы
#include <iostream.h>
#define STR15 16
#define MAX 50
int k;
struct boat
{ char model[STR15];
int year;
float price;
};
void print_data(boat *);
void main()
{ int i;
 boat Boats[MAX], *pastBoats;
pastBoats = &Boats[0];
cout << "How many boats? ";
cin >> k;
for (i = 0; i < k; i++)
{
cout << "\n Input a model of the boat: ";
cin >> pastBoats -> model;
cout << "\n Input the year of the creation of
the boat: ";
cin >> pastBoats -> year;
cout << "\n Input the price of the boat: ";
cin >> pastBoats ->"price;
pastBoats++;
}
pastBoats = &Boats[0];
print_data(pastBoats);
}
void print_data(boat *ptr_boat)
{
int i;
cout << "\n";
```

```
for (i = 0; i < k; i++) /* k айнымалысының
көріну аймағына назар салыңдар */
{
cout << "The boat " << ptr_boat -> model << "
the year of the creation " << ptr_boat -> year
<< endl;
 cout << " was sold for " << ptr_boat -> price
<< "$" << endl;
  ptr_boat++;
 }
}
```

Программаның орындалу нәтижесі:

```
How many boats? 3
 Input a model of the boat: Yakor
Input the year of the creation of the boat: 1987
Input the price of the boat: 18234
Input a model of the boat: Blue
Input the year of the creation of the boat: 2003
Input the price of the boat: 44321
Input a model of the boat: Volna
Input the year of the creation of the boat: 2000
Input the price of the boat: 39876
The boat Yakor the year of the creation 1987
 was sold for 18234$
The boat Blue the year of the creation 2003
 was sold for 44321$
The boat Volna the year of the creation 2000
 was sold for 39876$
Press any key to continue
```

14.8 Біріктірмелер (объединение – union)

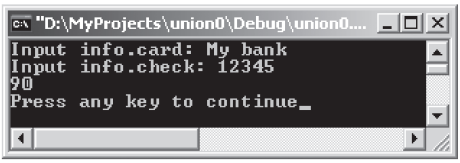
Біріктірмелер – бұл өрістерінің барлығы да бір адресте орналасатын құрылымның бір түрі. Оны сипаттау форматы да құрылымдардікі сияқты, тек **struct** түйінді сөзі орнына **union** сөзі қолданылады. Біріктірме ұзындығы оның өрістерінің ең көлемдісінің ұзындығына тең болады. Әрбір уақыт кезеңінде "Біріктірмелер" типті айнымалыда тек бір мән ғана сақталады.

Біріктірмелерді бір сәтте бір өріс қана пайдаланылатыны белгілі жағдайларда, тек компьютер жадын үнемдеу үшін қолданады.

Басқаша айтқанда, біріктірмелер – бұл әр түрлі уақыт кезеңдерінде әр түрлі типтегі объектілер сақталатын жады аймағы. Кез келген сәтте біріктірмеде көп дегеннің өзінде, тек бір объект сақталады, өйткені оның элементтері кезектесе отырып, жадының тек бір аймағын пайдаланады. Программалаушы біріктірмедегі мәліметті соның типіне сәйкес элементтің аты арқылы пайдаланылуын қадағалап отыруы тиіс. Біріктірмеге ең соңғы жазылған элементтен басқаларын пайдалану нәтижесі анықталмаған болып саналады. Енді бір мысал қарастырайық.

1-мысал

```
#include <iostream.h> // C++ тілі стилінде
void main ()
{ /* "карточка" немесе "чек" сияқты мәндері
бар жаңа pt типін (төлем типі) сипаттау */
 enum pt {CARD, CHECK};
 pt type;
 union payment
 { char card[25];
 long check;
 } info;
type = CARD;
cout << "Input info.card: ";
cin.getline(info.card, 25);
cout << "Input info.check: ";
cin >> info.check;
switch (type)
{case CARD : cout << info.card << endl; break;
case CHECK: cout << info.check << endl; break;
}
}
```



14.3-сурет

Прогамманың орындалу нәтижесі (14.3-сурет):

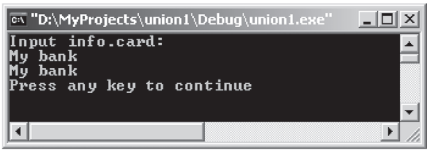
```
Input info.card : My bank
Input info.check : 12345
90
```

Алынған мәннің 90 болуы біріктірме элементін пайдалану

нәтижесінің анықталмағанын көрсетеді. Келесі мысалдағы біріктірмеде ең соңғы орналасқан элемент пайдаланылады.

```
#include <stdio.h> // C тілі стилінде
void main ()
{ enum pt {CARD, CHECK};
pt type;
union payment
{
char card[25];
 long check;
} info;
 type = CARD;
 info.check = 12345;
/* Мұнда компьютер жадына соңғы болып
орналасқан біріктірме элементі пайдаланылып
отыр: */
puts("Input info.card:");
gets(info.card);
switch (type)
{ case CARD : puts(info.card); break;
case CHECK: printf ("%d",info.check); break;
 }
}
```

Программаның орындалу нәтижесі (14.4-сурет):



14.4-сурет

```
Input info.card:
My bank
My bank
Press any key to continue
```

Бұл программада дұрыс нәтиже алынды.

Келесі мысал біріктірмелерді дұрыс пайдалану жауапкершілігі толығынан программалаушыда жатқанын көрсетеді.

```
#include <iostream.h> // C++ тілі стилінде
union Number
{
int x;
float y;
};
```

```
void main()
{ Number znachenie;
/* соңғы болып znachenie.y элементі мән
қабылдайды */
znachenie.x = 100;
znachenie.y = 50.;
cout << "znachenie.x: " << znachenie.x
<< "  znachenie.y: " << znachenie.y << endl;
/* соңғы болып znachenie.x элементі мән
қабылдайды */
znachenie.y = 50;
znachenie.x = 100;
cout << "znachenie.x: " << znachenie.x
 << "  znachenie.y: " << znachenie.y << endl;
}
```

Программа нәтижесі:

znachenie.x: 1112014848 znachenie.y: 50

znachenie.x: 100 znachenie.y: 1.4013e-043

Программа орындалуы барысындағы мынадай мәліметтер **znachenie.x: 1112014848 znachenie.y: 50** соңғы элемент **znachenie.y** болғанын көрсетеді.

Ал программадағы мынадай мәліметтер

znachenie. x: 100 znachenie. y: 1.4013e-043

соңғы элемент **znachenie.x** болғанын көрсетеді.

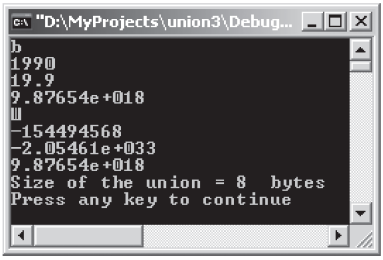
Біріктірмелер келесі программада көрсетілгендей компьютер жадын үнемді пайдалану мүмкіндігін береді.

```
#include <iostream.h>// C++ тілі стилінде
union many_types
{
char c;              //   1    байт
int ivalue;          //   4    байт
float fvalue;        //   4    байт
 double dvalue;      //   8    байт
 } my_union;
void main()
{
// дұрыс орындалатын енгізу-шығару операциялары
```

```
my_union.c = 'b';
cout << my_union.c << endl;
my_union.ivalue =1990;
cout << my_union.ivalue << endl;
my_union.fvalue = 19.90;
cout << my_union.fvalue << endl;
my_union.dvalue = 987654.32E+13;
cout << my_union.dvalue << endl;
// дұрыс орындалмайтын енгізу-шығару операция-
лары
cout << my_union.c << endl;
cout << my_union.ivalue << endl;
cout << my_union.fvalue << endl;
cout << my_union.dvalue << endl;
/* Программаның бірінші бөлігі дұрыс орын-
далды, өйткені біріктірмеге әр түрлі типтегі
мәліметтер бірден емес, біртіндеп меншіктелді.
Ал екінші бөлікте тек double типті мән
ғана дұрыс көрсетілді, өйткені ол соңында
енгізілген болатын. */
/* біріктірменің ең көлемді элементіне тең
ұзындығын есептеу */
cout << "Size of the union = "
 << sizeof (many_types) << " bytes" << endl;
}
```

Программаны орындау нәтижесі (14.5-сурет):

```
b
1990
19.9
9.87654e+018
Ш
-154494568
-2.05461e+033
9.87654e+018
Size of the union = 8 bytes
Press any key to continue
```



14.5-сурет

14.9 Биттік өрістер

Биттік өрістер – бұл құрылым өрістерінің ерекше бір түрі. Олар мәліметтерді жинақтап, мысалы, "иә/жоқ" сияқты жалаушаларды ықшамдап орналастыру үшін қажет. Мұндағы компьютер жадының ең кіші адрестелетін ұяшығы – 1 байт, ал жалаушаны сақтау үшін бір-ақ бит жеткілікті. Биттік өрістерді сипаттау кезінде оның атынан соң қос нүкте қойылады да, өріс ұзындығы битпен (бүтін оң константа) көрсетіледі:

```
struct Options{
bool centerX:1;
bool centerY:1;
unsigned int shadow:2;
unsigned int palette:4;
};
```

Биттік өрістер кез келген бүтін типте бола береді. Өріс аты жазылмауы да мүмкін, мұндай өрістер аппараттық шекараға туралау үшін қолданылады. Өрісті пайдалану әдеттегідей оның атын көрсету арқылы орындалады. Өріс адресін алуға болмайды, бірақ жалпы биттік өрістерді кәдімгі құрылым өрістері сияқты пайдалана беруге болады. Жеке биттермен амалдар орындау байт пен машиналық сөздерге қарағанда, онша үнемді орындалмайды, өйткені ол үшін компилятор арнайы кодтар жасап, программа кодының көлемін тым арттырып жібереді. Биттік өрістертерді компьютер жадында орналастыру компиляторға және аппаратураға байланысты болады.

Келесі құрылымды сипаттау мысалын қарастырайық:

```
   struct BitCard {
   /* көрсетілген цифрлар – бұл биттер саны */
   unsigned face : 4; /* 0 – тұз, ... ,12 – король; 4 бит 0
... 15 сандарын сақтай алады */
   unsigned suit : 2; /* 0 – қиық, 1 – табан, 2 – шыбын,
3 – қарға; 2 бит 0 ... 3 сандарын сақтайды */
   unsigned color : 1; /* 0 – қызыл, 1 – қара;
мұнда 1 бит жеткілікті */
```

```
} ;
```

Биттік өрістерді карта элементтерін сақтау үшін қолдану мысалы.

```
#include <iostream.h> // C++ тілі стилінде
#include <iomanip.h>
#include <stdio.h>
#include <math.h>
struct BitCard {
unsigned face: 4;  /* ағылш. face – карта;0 –
туз,..., 12 – король */
unsigned suit: 2;  /* ағылш. suit – карта
түрі;0 – қиық, 1 – табан,2 – шыбын,3 – қарға */
unsigned color:1; /* color – түсі; 0 – қызыл,
1 – қара */
};
void fillDeck(BitCard *); /* deck жиымын 52
картамен толтыру */
void deal (BitCard *); /* deal – карта тарату;
              экран 52 карта шығару */
void main ()
{
BitCard deck[52]; /* 52 картадан тұратын жиым */
fillDeck(deck) ;
deal(deck);
}
void fillDeck(BitCard *temp_deck)
{
 for ( int i = 0; i < 52; i++)
 {
temp_deck[i].face = i % 13;
temp_deck[i].suit = i / 13;
temp_deck[i].color = i / 26;
 }
}
void deal (BitCard *temp_deck)
```

```
{ int l = 0; /* бөліп шығару үшін */
/* ықшамды болуы үшін карталарды екі бағанға
бөлеміз: 0-25 карталар – бірінші баған, 26-51
карталар – екінші баған */
for (int k1 = 0, k2 = k1 + 26; k1 <= 25; k1++, k2++)
{cout << "Card: " << setw(3)
      << temp_deck[k1].face << " Suit: "
      << setw(2) << temp_deck[k1].suit
      << " Color: " << setw(2)
      << temp_deck[k1].color;
cout  << " Card: " <<setw(3)
      << temp_deck[k2].face << " Suit: "
      << setw(2) << temp_deck[k2].suit
      << " Color: " << setw(2)
      << temp_deck[k2].color << endl;
l++;
if (l > 12) { /* жалғастыру үшін Enter басу
                керек */
 printf("Press Enter for continuing ") ;
 getchar(); l = 0;}
}
}
```

Программаның орындалу нәтижелері:

```
Card:  0  Suit: 0 Color: 0 Card: 0  Suit:2Color: 1
Card:  1  Suit: 0 Color: 0 Card: 1  Suit:2Color: 1
Card:  2  Suit: 0 Color: 0 Card: 2  Suit:2Color: 1
Card:  3  Suit: 0 Color: 0 Card: 3  Suit:2Color: 1
Card:  4  Suit: 0 Color: 0 Card: 4  Suit:2Color: 1
Card:  5  Suit: 0 Color: 0 Card: 5  Suit:2Color: 1
Card:  6  Suit: 0 Color: 0 Card: 6  Suit:2Color: 1
Card:  7  Suit: 0 Color: 0 Card: 7  Suit:2Color: 1
Card:  8  Suit: 0 Color: 0 Card: 8  Suit:2Color: 1
Card:  9  Suit: 0 Color: 0 Card: 9  Suit:2Color: 1
Card: 10  Suit: 0 Color: 0 Card:10  Suit:2Color: 1
```

```
Card: 11  Suit: 0 Color: 0 Card:11  Suit:2Color: 1
Card: 12  Suit: 0 Color: 0 Card:12 Suit: 2Color: 1
Press Enter for continuing
Card:  0  Suit: 1 Color: 0 Card: 0 Suit: 3Color:1
Card:  1  Suit: 1 Color: 0 Card: 1 Suit: 3Color:1
Card:  2  Suit: 1 Color: 0 Card: 2 Suit: 3Color:1
Card:  3  Suit: 1 Color: 0 Card: 3 Suit: 3Color:1
Card:  4  Suit: 1 Color: 0 Card: 4 Suit: 3Color:1
Card:  5  Suit: 1 Color: 0 Card: 5 Suit: 3Color:1
Card: 6 Suit: 1 Color: 0  Card: 6 Suit:  3Color:1
Card: 7 Suit: 1 Color: 0 Card:  7 Suit:3 Color:1
Card: 8 Suit: 1 Color: 0 Card:  8 Suit:3 Color:1
Card: 9 Suit: 1 Color: 0 Card:  9 Suit:3 Color:1
Card: 10 Suit: 1 Color: 0 Card:  10 Suit:3 Color:1
Card: 11 Suit: 1 Color: 0 Card:11 Suit:3 Color:1
Card: 12 Suit: 1 Color: 0 Card:12 Suit:3 Color:1
Press Enter for continuing
Press any key to continue
```

Биттік өрістің адресін алуға болмайды, ал бұдан басқасында олар құрылым өрістері сияқты пайдаланыла береді. Енді қарастырылған құрылымдардың бірсыпыра ерекшеліктерін атап өтейік.

- Құрылым анықталғанда, компьютер жадында орын бөлінбейді.
- Құрылымды жариялауды аяқтаған соң, нүктелі үтір қоюды ұмытпау керек.
- Біріктірмелерде биттік өрістер болмайды.

Құрылымдар тек айнымалылар емес функцияларды да қамти алады (тек C++ ортасында). Құрылым функциялары осы құрылым ішіндегі айнымалылармен ғана жұмыс істейді. Құрылым мүшелерінің барлығы да ашық болып саналады.

Бақылау сұрақтары

1. ***Typedef*** *түйінді сөзі не үшін керек?* ***Enum*** *түйінді сөзі ше?*
2. *Тізбе түріндегі айнымалылар қалай сипатталып, олар қалай инициалданады?*
3. *Жиымдар мен құрылымдардың қандай айырмашылығы бар?*

4. *С тілінде құрылымдар қандай тәсілдермен сипатталады? Олардың бір-бірінен айырмашылықтарын түсіндіріңдер.*
5. *Құрылым элементтерін қалай инициалдауға болады?*
6. *"Құрылымға нұсқауыш" термині нені білдіреді?*
7. Құрылымның элементін пайдалану кезінде нүкте (.) немесе -> таңбалары не үшін қолданылады?
8. Құрылымдардың өрістерінің өздері құрылым бола ала ма?
9. Нұсқауыштарды құрылымдарға қатысты қолдану ерекшеліктері қандай?
10. Құрылымдарды функция аргументі ретінде пайдалануға бола ма?
11. Құрылымдармен жұмыс істеуде қандай амалдар қолданылады?
12. Құрылымдар жиымдарын қалай құруға болады?
13. Құрылым элементтерін экранға шығару мүмкіндіктерін көрсетіңдер.
14. Біріктірмелердегеніміз не? Олар қалай сипатталады?
15. Биттік өрістер не үшін қажет?Олар компьютер жадынан қанша орын алады?
16. Биттік өрістерді сипаттау қалай атқарылады?

Тапсырмалар

Келесі мәліметтерден құралған төрт студент туралы ақпаратты енгізіңіз:
- тегі және инициалы;
- туған жылы;
- оқуға түскен жылы;
- бірінші семестрдің бағасы:
- физика;
- жоғарғы математика;
- информатика;

1. Әліпби бойынша реттелген студенттердің тізімін шығару керек.
2. Туған жылы бойынша реттелген студенттердің тізімін көрсету қажет.
3. Оқуға түскен жылы бойынша реттелген озат студенттердің тізімін шығару керек.
4. Сессияны 4 және 5-ке тапсырған студенттердің анкеталық мәліметтері шығарылуға тиіс.
5. Тегі (фамилиясы) Б әрпінен басталатын студенттердің тізімін және олардың барлық пәндер бойынша бағаларын көрсету керек.
6. Оқу озаттарының анкеталық мәліметтерін шығару қажет.
7. Тегі А әріпінен басталатын студенттердің тізімін және олардың туған жылдары көрсетілуі тиіс.
8. Сессияда "3" деген бағалары бар студенттердің анкеталық мәліметтерін шығару керек.

9. Тегі В және Г әрпінен басталатын студенттердің тізімін және олардың бағаларын шығаратын программа құру керек.
10. Сессияны "үштік" баға алмай, емтихан тапсырған студенттердің тізімін және олардың туған жылдарын шығару керек.
11. Барлық студенттердің жалпы орта балын және осы орта балдан жоғары балл жинаған студенттердің тізімі көрсетілуі тиіс.
12. Барлық студенттердің жалпы орта балын және осы орта балға тең балл жинаған студенттердің тізімін шығару керек.
13. Сессияда "екілік" алған студенттердің анкеталық мәліметтерін шығару керек.
14. Информатика пәнін 5-ке тапсырған студенттердің анкеталық мәліметтерін экранға шығару қажет.
15. Физиканы 4-ке және жоғарғы математиканы 5-ке тапсырған студенттердің анкеталық мәліметтері шығарылуы тиіс.

Төмендегідей іс-әрекеттерді лабораториялық жұмыстар кезінде орындау керек.

1 нұсқа

Flat (бөлмелер саны, көлемі, қабаты, мекен жайы, бағасы) құрылымын сипаттаңыз. Пернетақтадан жиымға бес элементтен тұратын Flat типіндегі мәндер енгізіңіз. Экранға бағалары (пернетақтадан) енгізілген саннан аспайтын пәтерлер жайлы ақпаратты шығару керек. Егер, ондай пәтер болмаса, программа экранға сәйкесінше хабарламаны шығаруы тиіс.

2 нұсқа

PEREVOZKI (ұшақ типі, рейстер саны, осы кезге дейінгі сағатпен берілген ұшқан мерзімі, мың километрмен берілген ұшқан қашықтығы) құрылымын сипаттаңыз. Пернетақтадан жиымға алты элементтен тұратын PEREVOZKI типіндегі мәндер енгізіңіз. Экранға ұшқан мерзімі (пернетақтадан) енгізілген саннан артық ұшақтар жайлы ақпаратты шығару керек. Егер, ондай ұшақ болмаса, программа экранға сәйкесінше хабарлама шығарылуы тиіс.

3 нұсқа

AVTO (машина маркасы, түсі, бағасы, максималды (максимум) жылдамдығы) құрылымын сипаттаңыз. Пернетақтадан жиымға алты элементтен тұратын AVTO типіндегі мәндер енгізіңіз. Экранға жылдамдығы (пернетақтадан) енгізілген саннан (аспайтын) кем болатын машиналар жайлы ақпаратты шығару керек. Егер, ондай машина болмаса, программа экранға сәйкесінше хабарламаны шығаруы тиіс.

4 нұсқа

TELEFON (абоненттің ЖАТ, телефон нөмірі, телефонға төлеген төлемі, орнатқан жылы) құрылымын сипаттаңыз. Пернетақтадан жиымға бес элементтен тұратын TELEFON типіндегі мәндер енгізіңіз. Экранға орнатылған жылы (пернетақтадан) енгізілген саннан (аспайтын) артық болатын телефондар жайлы ақпаратты шығару керек. Егер, ондай телефон болмаса, программа экранға сәйкесінше хабарламаны шығаруы тиіс.

5 нұсқа

DET_SAD (бала бақша нөмірі, бүлдіршіндер саны, қала ауданы, бір айлық төлемі) құрылымын сипаттаңыз. Пернетақтадан жиымға алты элементтен тұратын DET_SAD типіндегі мәндер енгізіңіз. Экранға көрсетілген аудандағы (ауданның аты пернетақтадан енгізіледі) бала бақшалар жайлы ақпаратты шығару керек. Егер, бұл ауданда балабақшалар әлі болмаса, программа экранға сәйкесінше хабарлама шығарылуы тиіс.

6 нұсқа

MUSEUM (мұражай аты, жұмыс жасау уақыты, билеттің құны, мекен жайы) құрылымын сипаттаңыз. Пернетақтадан жиымға алты элементтен тұратын MUSEUM типіндегі мәндер енгізіңіз. Экранға билетінің құны (пернетақтадан) енгізілген саннан (аспайтын) аспайтын мұражайлар жайлы ақпаратты шығару керек. Егер, ондай мұражай болмаса, программа экранға сәйкесінше хабарламаны шығаруы тиіс.

7 нұсқа

TOUR (елі, турдың ұзақтығы, транспорт түрі, бағасы) құрылымын сипаттаңыз. Пернетақтадан жиымға жеті элементтен тұратын TOUR типіндегі мәндер енгізіңіз. Экранға билет құны (пернетақтадан) енгізілген саннан (аспайтын) аспайтын елдер жайлы ақпаратты шығару керек. Егер, ондай ел болмаса, программа экранға сәйкесінше хабарламаны шығаруы тиіс.

8 нұсқа

PRICE (тауардың аты, құны, дүкеннің аты) құрылымын сипаттаңыз. Пернетақтадан жиымға жеті элементтен тұратын PRICE типіндегі мәндер енгізіңіз. Экранға пернетақтадан енгізілген дүкендегі тауарлар жайлы барлық ақпаратты шығару керек. Егер, ондай дүкен болмаса, программа экранға сәйкесінше хабарламаны шығаруы тиіс.

9 нұсқа

FRIDGE (тауардың аты, дайындаған фирмасы, қоймадағы саны, бағасы) құрылымын сипаттаңыз. Пернетақтадан жиымға бес элементтен тұратын FRIDGE типіндегі мәндер енгізіңіз. Экранға

бағасы (пернетақтадан) енгізілген саннан (аспайтын) аспайтын тоңазытқыштар жайлы ақпаратты шығару керек. Егер, ондай тоңазытқыш болмаса, программа экранға сәйкесінше хабарламаны шығаруы тиіс.

10 нұсқа

ZODIAK (жұлдыз бойынша зодиак белгісі, аты және жөні, туылған күні) құрылымын сипаттаңыз. Пернетақтадан жиымға жеті элементтен тұратын ZODIAK типіндегі мәндер енгізіңіз. Экранға жұлдызы "мерген" болатын адамдардың жөнін, атын және туылған күнін шығару керек. Егер, ондайлар жоқ болса, программа экранға сәйкесінше хабарламаны шығаруы тиіс.

11 нұсқа

STUDENT (оқу орнының аты, студенттің тегі және инициалдары, топ нөмірі, бес бағадан тұратын жиым-үлгерімі) құрылымын сипаттаңыз. Пернетақтадан жиымға алты элементтен тұратын STUDENT типіндегі мәндер енгізіңіз. Экранға орта балы (пернетақтадан) енгізілген саннан артық болатын студенттер жайлы ақпаратты шығару керек. Егер, ондай студент болмаса, программа экранға сәйкесінше хабарламаны шығаруы тиіс.

12 нұсқа

WORKER (жұмысшының тегі және инициалдары, қызметі, жұмысқа орналасқан жылы, үш бағадан тұратын жиым-аттестатция нәтижесі) құрылымын сипаттаңыз. Пернетақтадан жиымға бес элементтен тұратын WORKER типіндегі мәндер енгізіңіз. Экранға аттестациясының орта балы (пернетақтадан) енгізілген саннан (аспайтын) төмен болмайтын жұмысшылар жайлы ақпаратты шығару керек. Егер, ондай жұмысшы болмасы, программа экранға сәйкесінше хабарламаны шығаруы тиіс.

15 ФАЙЛДАРДЫ ПАЙДАЛАНУ

Файл – сыртқы есте сақтау құрылғыларында (магниттік дискілерде) орналастырылған және мәлімет өңдеу, тасымалдау кездерінде біртұтас күйде қарастырылатын мәліметтер жиыны.

Файлдармен жұмыс істеу үшін оларды, алдымен, ашу керек, яғни файл туралы мәліметті – атын, адресін программаға белгілі ету қажет.

С тілінде файл ашу *fopen*() функциясы арқылы орындалады. Ол сыртқы құрылғыдағы физикалық файлды, мысалы, A:\NUR.DAT, программадағы оның логикалық атымен байланыстырады. Логикалық ат дегеніміз – файлға нұсқауыш, яғни файл туралы мәлімет сақталатын жады аймағына сілтеме жасау. Файлға нұсқауыш сипатталуы тиіс.

Сонымен, файлдармен жұмыс істегенде нұсқауыштар қолданылады. Файлға нұсқауыш мынадай түрде сипатталады:

```
FILE *fp;
```

мұндағы FILE типі – **<stdio.h>** тақырыптық файлында анықталған құрылым. Бұл нұсқауышты көрсетілген файлмен сол файл ашылғаннан бастап, байланыстырып қоюға болады. Ол үшін келесі функция пайдаланылады:

```
fopen("файл адресі/аты","қатынасу типі");
```

ол файлға нұсқауыш мәнін қайтарады немесе қате болса, NULL мәнін береді.

Мысалы, мынадай оператор орындалуы нәтижесінде

```
fp = fopen("ex1.txt","w");
```

жұмыс бумасындағы **ex1.txt** файлы оған мәлімет жазу (write) үшін ашылады, ал программада бұл файлды **fp** нұсқауышы арқылы пайдаланамыз (яғни ***fopen()*** функциясы файлдың сыртқы атын оның программада қолданылатын ішкі логикалық атымен байланыстырады).

Сонымен, файлды ашатын ***fopen()*** функциясының жалпы жазылуы:

```
fp = fopen(name, mode)
```

мұнда **fp** – файлға сілтейтін нұсқауыш; **name** – файлдың адресін де көрсетуге болатын аты, көбінесе символдық тіркес арқылы жа-

зылады; **mode** – файл қандай режимде қолданылатынын көрсететін параметр, бұл да символдар тіркесімен төмендегідей түрде жазылады:

"**r**" – файлды оқу үшін ашу (файл бұрын ашылған болуы тиіс);

"**w**" – бос файлды информация жазу үшін ашу;

"**a**" – файл соңына мәлімет қосып жазу үшін оны ашу;

"**r+**"– файлдан информация оқу және оған информация жазу үшін ашу);

"**w+**"– бос файлдан информация оқу және оған мәлімет жазу үшін файл ашу (бұрын файл болса, ол өшіріледі);

"**a+**"– файлдан информация оқу және оның соңына информация қосып жазу үшін ашу).

"**t**" – файл мәтіндік (текстік) режимде ашылады, ол **rt, wt, at,** т.с.с. болып көрсетіле береді.

"**b**" – файл екілік режимде ашылады, **rb, wb, ab,** т.с.с. болып көрсетіле береді.

Келісім бойынша файл көбінесе мәтіндік режимде ашылады.

Файлмен жұмыс істеп болған соң, оны жабу үшін келесі функция қолданылады:

fclose(файл_нұсқауышы).

Файлға мәлімет жазу/оқу үшін жалпы енгізу/шығару функциялары тәріздес бірсыпыра функциялар пайдаланылады:

fprintf(),fscanf(),fputs(),fgets(),getc(), putc(),fgetc(), fputc()

Бұлардың ішіндегі **getc()/fgetc(),putc()/fputc()** функциялары әрекеттері ұқсас, айырмашылықтары тек **getc**() пен **putc**() макроанықтаулар да, ал **fgetc**() және **fputc**() – нағыз функциялар болып табылады.

Барлық файлдық функциялар прототиптері **<stdio.h>** файлында орналасқан.

Файлдардан мәлімет жазу/оқу әрекеттерін үш топқа бөлуге болады:

- символдарды енгізу/шығару операциялары;
- сөз тіркестері жолдарын енгізу/шығару операциялары;
- блок бойынша енгізу/шығару операциялары.

Символдарды енгізу/шығару операцияларында файлдан бір ғана символ оқылады немесе оған бір символ жазылады. Мысалы:

fgetc(FILE *fp); ашылған файлдан символ оқиды.

fputc(int ch,FILE *fp); **ch** символы кодын файлға жазады.

Сөз тіркестері жолдарын енгізу/шығару операцияларында бір мәлімет алмасуы кезінде файлдан сөз тіркесі жолдары оқылады немесе оған сөз тіркесі жазылады. Мысалы:

gets(char *S); файлдан мәлімет байттарын '\n' символы кездескенше біртіндеп оқиды да, оларды **S** тіркесіне орналастырып, '\n' символын '\0' (нөл-терминатор) белгісіне айналдырады.

fgets(char *S,int m,FILE *fp); **fp** түрінде сипатталған файлдан байттарды оқып, оларды '\n' символы кездескенше немесе *m* байт оқылып біткенше S тіркесі мәні ретінде жазып шығады.

Блок бойынша енгізу/шығару операцияларында мәліметтер алмасу олардың бір блогын толық қамтиды. Мысалы:

fread (void *ptv, int size, int n, FILE *fp);

мұнда **fp** файлынан әрқайсысы **size** байттан тұратын мәліметтің **n** блогы **ptv** нұсқауышы көрсетіп тұрған жады аймағына оқылады (оқылатын блок үшін алдын ала орын дайындап қою қажет).

fwrite(void *ptv,int size,int n,FILE *fp);

мұнда **ptv** нұсқауышы көрсетіп тұрған жады аймағында орналасқан әрқайсысы **size** байттан тұратын мәліметтің **n** блогы ашық тұрған **fp** файлына жазылады.

15.1 fprintf және fscanf функцияларын пайдалану

fprintf файлға информация жазу үшін, ал **fscanf** файлдан информация оқу үшін қолданылады. Олардың жалпы жазылу түрі:

fprintf(fp, "спецификация шаблоны", p);

мұндағы **p** – өрнек**;**

fscanf(fp, "спецификация шаблоны", адрес);

Мысал: Бүтін сандар тізбегін (n=5) пернелерден енгізіп, оларды файлға жазу керек болсын.

```
// файл ашып, оған 5 сан жазу
#include <stdio.h>
#include <conio.h>
main()
{ int k,i,n=5;
 char fname[]="a:\\num.txt\0";
```

```
 clrscr();
 FILE *fp;
 fp=fopen(fname,"w");
 printf("Енгізілген сандар %s файлына жазылады
         \n",fname);
 puts("Әр сан енгізілген соң, Enter басыңыз\n");
  for(i=0; i<n;i++)
  {scanf("%i",&k);
  fprintf(fp,"%d ",k);
  }
  fclose(fp);
  printf("Енгізілген сандар %s файлына жазылды\n",
         fname);
  getch();
}
```

Нәтижесі:

Енгізілген сандар a:\num.txt файлына жазылады
Әр сан енгізілген соң, Enter басыңыз
1
2
3
4
5
Енгізілген сандар a:\num.txt файлына жазылды
Файлдағы мәлімет: 1 2 3 4 5

Бес бүтін санды n=5 бұрыннан бар **file1.txt** файлына жазу керек, сол файлды мәлімет қосу үшін ашып, оған 5 сан жазып, сонан соң ондағы мәліметті оқып тексеру керек.

```
/* файлға мәліметтер қосу */
#include <math.h>
#include <stdio.h>
#include <conio.h>
main()
{ int i, k=0, s=0, n=5, y, x;
 char fname[]="file1.txt";
 clrscr();
 FILE *fp;
 fp=fopen("file1.txt", "at");
```

```
 if ((fp=fopen(fname,"at")) == NULL)
  { printf("\nФайлды ашуда қате болды");
   getch();
   return(0);
  }
 puts("Enter арқылы бөліп, 5 сан енгізіңіз:");
 for(i=1; i<=n;i++)
  {scanf("%i", &x);
  fprintf(fp,"%d ", x);}
 fclose(fp);
 getch();
}
```

Нәтижесі:

```
Enter арқылы бөліп, 5 сан енгізіңіз:
11
22
33
44
55
```

Файлдағы мәлімет: **1 2 3 4 5 11 22 33 44 55**

Дискідегі **file1.txt** файлын мәліметтер оқу үшін ашып, ондағы мәліметті оқып, файлдағы тақ сандардың қанша екенін және қосындысын табу керек.

```
/* файлдан мәліметтер оқу */
#include <math.h>
#include <stdio.h>
#include <conio.h>
main()
 {  int i,k=0,s=0,n=5,y,x;
     char fname[]="file1.txt";
     clrscr();
     FILE *fp;
     fp=fopen(fname,"r");

 /* Оқу үшін файл ашу */
 fp=fopen("fp","rt");
```

```
  if ((fp=fopen(fname,"rt")) == NULL)
 { printf("\n Файл ашуда қате болды");
   getch();
   return(0);
 }
 puts("Файлдан оқылған сандар:");
 while(!feof(fp))
  {fscanf(fp,"%i",&y);
  printf(" %i ",y);
  if (y%2==0) {s+=y;k=k+1;}
  }
 fclose(fp);
 printf("\nТақ сандар қосындысы s = % i,
      олардың саны k=%i", s, k);
 getch();
}
```

15.2 fgets және fputs функцияларын пайдалану

Сөз тіркестерімен жұмыс істегенде **fgets** және **fputs** функциялары қолданылады:

fgets функциясының жалпы жазылу түрі:

```
fgets(нұсқауыш,MAXLEN,fp);
```

нұсқауыш – компьютер жадындағы орынға сілтейтін нұсқауыш; **MAXLEN** – оқылатын тіркестің максимальды ұзындығы; **fp** - файл нұсқауышы.

Мысалы:

```
#define L 20
main()
{
    FILE *fp;
    char *st[n];
    fp=fopen("stroka", "r");
    while(fgets(st, L, fp) != NULL)
          puts (st);
}
```

Файлдарға сөз тіркестерін жазу үшін **fputs** функциясы келесі түрде қолданылады:

status=fputs(қатар нұсқауышы, fp);

status – бүтін сан, оның мәні **eof** функциясында жазылады, егер **fputs()**функциясы файлдың соңына шыққан болса немесе қате тапса, **fputs** функциясы жазылатын жолдың соңына **/0** символын жазбайды.

Төменде осы функцияларды пайдалану мысалы келтірілген.

```
#include <stdio.h>
void main()
{
int n;
char str[50],str1[50],ch;
FILE *fp;
// Файлға мәлімет жазу
fp = fopen("ex.txt","w");
puts("Бүтін сан енгізіңіз: "); scanf("%d",&n);
fprintf(fp,"%d\n",n);
puts ("Символ енгізіңіз: "); ch=getchar();
putc (ch, fp);
puts ("Сөз тіркесін енгізіңіз: "); gets(str);
fputs(str,fp);
fclose(fp);
// Файлдан мәлімет оқу
if((fp = fopen("ex.txt","r")) != NULL)
 {
 fscanf (fp, "%d", &n); printf ("n=%d\n", n) ;
 ch = getc (fp); putchar (ch);
 fgets(str1, 50, fp); puts (str1);
 fclose (fp);
 } else printf ("\nФайлдан мәлімет оқылмайды!");
}
```

Мұндағы **fgets()** функциясының екінші параметрі N – оқылатын символдар саны, оған '\0' белгісі де қосылады. Бұл функция өз жұмысын N-1 символын оқығаннан кейін немесе '\0' белгісі кездескенде аяқтайды. Екеуінде де сөз тіркесі соңына '\0' белгісі қосылады. **fgets()** функциясы оқылған сөз тіркесі адресін қайтарады немесе файл оқылып болғанда (не қате шықса), NULL белгісін береді.

fputs() функциясы әрекет дұрыс орындалса, соңғы оқылған символ кодын қайтарады, ал қате болса, EOF (файл соңы) белгісін береді. Бұл функция курсорды автоматты түрде келесі жолға көшірмейді.

Жоғарыдағы функциялар файл мәліметтерін біртіндеп, символдан соң келесі символды қарастыра отырып өңдейді. С тілі файлдармен жиым сияқты жұмыс істеуге де мүмкіндік береді, яғни кез келген байтты жеке өңдеуге де болады. Файл ішіндегі белгілі бір орынды айқындау үшін мына функция қолданылады:

fseek(файлға нұсқауыш, бастапқы нүктеден ығысу, бастапқы нүкте);

Екінші аргумент типі *long*, оның мәні оң да, теріс те болуы мүмкін. Ол бастапқы нүктеден қанша орынға (байтпен) ығысу керек екендігін көрсетеді. Үшінші аргумент файлдағы бастапқы нүкте орнын анықтайтын код болып табылады. Осы код үшін мынадай мәндер тағайындалған:

0 - файл басы;
1 - ағымдағы позиция;
2 - файл соңы.

Дұрыс орындалғанда, *fseek()* функциясы 0 мәнін береді, ал егер қате (мысалы, файлдың сол жақ шекарасынан ары аспақшы болғанда) болса, онда 1 береді.

15.3 fwrite және fread функцияларын пайдалану

Құрылымдарды пайдаланатын файлдармен жұмыс істеу кезінде **fread()/fwrite()** функцияларын пайдаланған ыңғайлы. Олардың жазылуы:

```
fread(ptr, size, n, fp);
```

мұнда **fp** файлынан әрқайсысы **size** байттан тұратын мәліметтің **n** блогы **ptv** нұсқауышы көрсетіп тұрған жады аймағына оқылады (оқылатын блок үшін алдын ала орын дайындап қою қажет).

```
fwrite(ptr,size,n,fp);
```

мұнда **ptr** нұсқауышы көрсетіп тұрған жады аймағында орналасқан әрқайсысы **size** байттан тұратын мәліметтің **n** блогы ашық тұрған **fp** файлына жазылады.

Осы функцияларды қолданудың бір мысалын келтірейік:

```
  typedef struct
{
 char author [30];
 char title [50];
 int pages;
} BOOK;
BOOK b1={ "Kernighan", "C Language", 256 }, b2;
FILE *fp;
void main()
{...
 fp=fopen("struct.txt", "w+");
// файл әрі оқу, әрі жазу үшін ашылды
 fwrite(&b1, sizeof(BOOK), 1, fp);
 fseek(fp,0,0); // маркер файл басына
 fread(&b2, sizeof(BOOK), 1, fp);
  printf("Авторы – %s,аты – %s,беттер саны –
%d\n",b2.author, b2.title, b2.pages);
}
```

Бақылау сұрақтары

1. *Файл дегеніміз не? Ол не үшін пайдаланылады?*
2. *Файл қайда орналасады және қалай белгіленеді?*
3. *Файл ашу функциясы қалай жазылады?*
4. *Файл атын программада қалай анықтаймыз?*
5. *Файлдың қолданылу режимдері қалай көрсетіледі?*
6. *Файлға мәлімет жазу/оқу функциялары.*
7. *Файлға жаңа элемент қалай қосылады?*
8. *Мәлімет оқылған файлға мәлімет жазуға бола ма?*
9. *Файлдан ақпарат оқу үшін не істеу керек?*
10. *Файлға ақпарат жазу үшін не істеу керек?*
11. *Символдар мен сөз тіркестерін файлға жазу үшін не істеу керек?*
12. *Құрылымдарды пайдаланатын файлдар қандай функцияларды пайдаланады?*

Тапсырмалар

1. Файлдан сандар оқып, солардың ішіндегі теріс сандар қанша екенін анықтайтын программа құру керек.
2. Файлдан сөз тіркесін оқып, солардың ішіндегі 6 символдан артық сөздерді экранға шығарып, басқа файлға жазып шығыңдар.

3. Файлдан сандар оқып, солардың арифметикалық ортасын файл соңына қосып жазу керек.
4. Файлдан сөз тіркесін оқып, солардың керісінше жазылған нұсқасын басқа бір файлға жазып шығындар.
5. Файлдан сандар оқып, солардың максимумын анықтайтын программа құру керек.
6. Файлдан сөз тіркесін оқып, солардың ішіндегі бос орын орнына сызықша жазып оны басқа бір файлға жазып шығу керек.
7. Файлдан сандар оқып, солардың минимумын анықтайтын программа құру керек.
8. Бір файлда екі сөйлем жазылған, соның екінші сөйлемін басқа файлға жазып шығу керек.
9. Файлдан сандар оқып, солардың көбейтіндісін анықтайтын программа құру керек.
10. Сөз тіркесінен тұратын файлдағы бас әріптерді кіші әріптерге айналдырып, басқа файлға жазып шығу керек.
11. Файлдан сандар оқып, солардың нешеуі 5-тен артық екенін анықтау керек
12. Сөз тіркесінен тұратын екі файл берілген. Осы екі файлдағы сөз тіркестерін біріктіріп, үшінші файлға жазып шығындар.
13. Файлдан сандар оқып, солардың қосындысын анықтау керек
14. Сөз тіркесінен тұратын екі файл берілген. Осы екі файлдағы сөз тіркестерінің орнын ауыстырып жазып шығу керек.
15. Файлдан сөз тіркесін оқып, солардың ішіндегі ең ендісін анықтап, соның неше символдан тұратынын анықтау қажет.
16. Файлдан сандар оқып, солардың тақтарын бір файлға, жұптарын екінші файлға жазып шығындар.
17. Файлдан бірнеше сөйлем оқып, сол сөйлемдердің "м" әрпінен басталатын бір сөйлемін ғана экранға шығарып, соның ұзындығын анықтайтын программа құру керек.
18. Файлдан сандар оқып, солардың барлығын да бірге арттырып, шыққан сандарды басқа бір файлға жазып шығындар.
19. Файлдан бірнеше сөйлем оқып, сол сөйлемдердің ең соңғы сөйлемін ғана экранға шығарып, соның ұзындығын анықтайтын программа құру керек.
20. Файлдан сандар оқып, солардың керісінше тәртіппен екінші бір файлға жазып шығындар.
21. Файлдан сандар оқып, солардың цифрларын экранға сөзбен шығаратын программа құру керек, мысалы, 0 орнына "нөл", 1 орнына "бір", т.с.с. 9 орнына "тоғыз" деп жазатын болуы тиіс.
22. Файлдан сөз тіркесін оқып, солардың бірінші сөзі мен соңғы сөзін алмастырып, екінші бір файлға жазып шығындар.

23. Файлдан сөз тіркесін оқып, соларды керісінше жазып шығатын программа құру керек.
24. Файлдан сандар оқып, солардың алғашқы жартысы мен соңғы жартысының орнын ауыстырып, нәтижесін жаңа файлға жазып шығу керек.
25. 10 бүтін саннан тұратын файл жасайтын программа құру керек. Сол файлдағы сандарды оқып, олардың қосындысын анықтаңдар.
26. While операторы арқылы Char типті элементтерден тұратын файл жасау қажет. Циклден шығу шарты – z әрпін енгізу. Сол файлдың көшірмесін екінші бір файлға жазып, жазылған мәліметтерді экранға да шығару керек.
27. Integer типті N саннан тұратын файл жасап, сол файлдағы жұп сандарды экранға шығару қажет.
28. Бүтін сандардан тұратын файл жасап, сол сандарды басқа бір файлға кері тәртіппен жазып шығу керек.
29. Мынадай құрылымдағы бірнеше қатарлары бар файл жасау керек:
    - реттік нөмірі;
    - фамилиясы, аты-жөні;
    - жалақысы.

    Осы файлға бес адам туралы мәліметтер енгізіп, басқа файлға осылардың ішіндегі ең көп жалақы алатын адам туралы мәліметті көшіріп жазу керек.
30. Файлдағы мәтінді түгел оқып шығып, ондағы "о" әрпін "а" әрпімен алмастыратын программа құрыңдар.
31. Файлдағы мәтіндік ақпаратты экранға және қағазға шығаратын программа жасау керек.
32. Файлда N бүтін сан жазылған. Соларды өсуі бойынша реттеп, екінші файлға жазып шығару қажет.
33. Нақты сандардан тұратын бір өлшемді жиым элементтерін пернелерден енгізе отырып, бір файлға жазып шығып, сол жиымның жұп индексті элементтерін екінші файлға жазатын программа құрыңыздар.
34. Топтағы студенттердің үлгерімін бір файлға мынадай түрде жазу керек: рет нөмірі, аты-жөні, 5 сабақтың аты, әр сабақтан алған 3 бағасы. Программаға енгізілетін мәліметтер пернелерде теріліп, файлға жазылуы қажет. Жақсы оқитын студенттер тізімін екінші бір файлға бөлек жазып шығу керек.

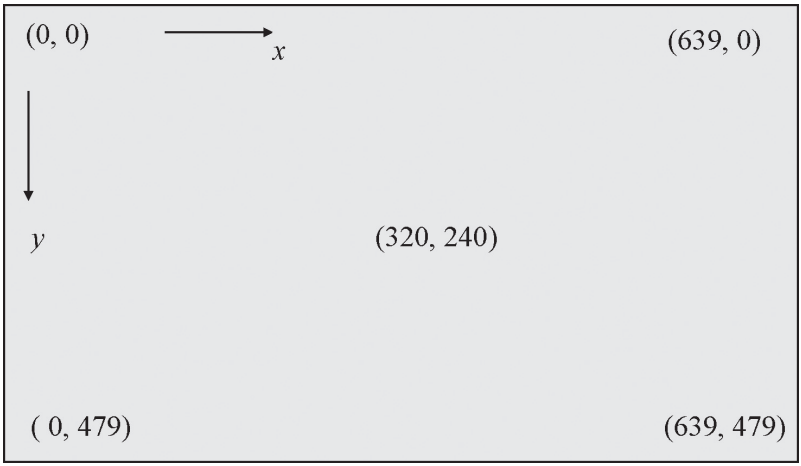
16. ГРАФИКАЛЫҚ РЕЖИМДЕ ЖҰМЫС ІСТЕУ

С тілінде растрлық графика жұмыс істейді, оның тақырыптық файлы **graphics.h**. График нүктелерден – пиксельдерден тұрады. *Пиксель – экранның адрестелетін ең кіші элементі*. Алдымен, графика шығара алатын бейнережимді іске қосу керек. Мұнда экранның пиксельмен берілген мөлшері және түстер саны беріледі.

Графикада үш координаталық жүйе: *абсолюттік*, *салыстырмалы* және *масштабталған* жүйе қолданылады. Тіке және көлденең өстер бойынша пиксельдер саны экран типіне байланысты болады.

Абсолюттік координатада координаталар басы – (0;0) нүктесі сол жақ жоғарғы бұрышта болып саналады, *x* координатасы солдан оңға қарай, *y* координатасы жоғарыдан төмен қарай өседі.

Салыстырмалы режимде координаталар басы экранның кез келген нүктесіне ауыстырыла алады.



16.1-сурет. Экрандағы нүктелердің координаталары

Масштабталатын режимде экран бетінде масштабталған координаталар беруге болады, онда *x* пен *y* өстері бойынша минимум және максимум мәндер енгізіп, жұмыс істеуге мүмкіндік бар. Жалпы абсолюттік графикалық режимде (16.1 сурет)

– әрбір пиксель берілген 16 түстің біріне боялады;

– (0,0) – экранның сол жақ жоғарғы бұрышы координатасы;

– (639,479) – оң жақ төменгі бұрышының координатасы болады;

– әртүрлі фигураларды экранға шығару үшін алдын ала графикалық режимді іске қосып алу керек.

Жалпы, дисплей адаптерлері графикалық режимде 200, 350, 600 нүктелерден тұратын экран жолдарының әрқайсысында 640, 720, 800 нүктелер тізбегін бейнелей алады. Мұндағы нүкте деп отырғанымыз – көлемі 0,8х1 мм2 шамасында болып келген (CGA) кішкентай тіктөртбұрыш, яғни пиксель. Әрбір нүктенің координаталары екі бүтін санмен (х, у) анықталады. Дисплей экранына график салу үшін оның нүктелерінің координаталарын көрсету қажет. Координаталар басы (0,0) болып экранның сол жақ жоғарғы бұрышы есептеледі. *X* координаталары (бағаналар немесе позициялар нөмірлері) солдан оңға қарай, ал *y* мәндері (жолдар немесе қатарлар) жоғарыдан төмен қарай өсіп отырады. Мысалы, VGA адаптерінің экран бұрыштарының координаталарын *x*=0..799, *y*=0..599 аралығында көрсету қажет. Экранда *x* өсі солдан оңға қарай, *Y* өсі жоғарыдан төмен қарай бағытталған, ал оның шеткі нүктелерінің координаталары суретте көрсетілген. Олардың ең жоғарғы мәндері пайдаланылған экран адаптеріне тәуелді болады, яғни (0,0)..(320х200), (0,0)..(640х480), (0,0).. (800х600) аралықтарында және т.б. болуы мүмкін. Сонымен, графикалық режимде экрандағы кез келген объект көрініп тұрған нүктелер тобынан тұрады. Мәтіндік режимнен графикалық режимге көшкенде экран тазартылады. Графикалық режимде экраннан курсор көрінбейді. Дисплей экранының бетіне (кейін қағазға) нүкте, түзу немесе қисық сызық, шеңбер, эллипс және кез келген тұйық сызық сызып шығаруға болады. Сонымен қатар тұйық сызықтардың ішін әртүрлі түске бояп қою мүмкіндіктері де бар. Сызықтарды жылжыту, айналдыру және басқа орынға көшіру арқылы көрнекі бейнелер мен мультфильмдер жасауға болады.

График тұрғызу үшін оны шығару немесе бастау нүктесін көрсету қажет. Мәтіндік режимде ол курсор позициясы болып саналады, ал графикалық режимде көрініп тұратын курсор жоқ, бірақ экранда көрінбейтін курсор тәрізді сілтеме белгі CP (current

pointer) бар. Негізінде оны да курсор деп қабылдауға болады.

Графикалық режимдегі жұмыстарды атқаратын Турбо С нұсқасында бірсыпыра графикалық функциялар бар, енді біз солардың негізгілеріне тоқталып өтеміз.

Графикалық режимде жұмыс істеу былай басталады:

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
void main()
{int gdriver = DETECT;     //драйвер
int gmode;                 //режим аты
int errorcode;             // қате коды
initgraph (&gdriver,&mode,"C:\\TCbgi\\");
errorcode = graphresult();
if errorcode != grOk)      //іске қосу қатесі
   {printf("Қате: %d\n", errorcode);
 puts("Аяқтау үшін ENTER басыңыз");
 getch();
 return;}
 ... Ары қарай программа мәтіні...
 getch();
 closegraph ();
}
```

16.1. Графикалық режим орнату, одан шығу, мәтін жазу, сызық салу функциялары

Графикалық режим драйверін іске қосу функциясы initgraph(&Driver, &Mode, Path);

Driver параметрі бейнелік жүйе драйверін анықтайды, *Mode* параметрі бейнелік жүйе жұмыс режимін береді, ал *Path* параметрі драйвер файлының орнын көрсетеді. Көбінесе *Driver* параметрі мәні ретінде ***detect*** болатын бүтін константа қолданылады. Мұндайда ***initgraph*** функциясының өзі графикалық драйвер типін анықтап, ең дұрыс режимді таңдап алады. Тақырыптық файлы: **<graphics.h>**

Драйвер дұрыс оқылған соң, *initgraph()* функциясы 4К көлемінде (келісім бойынша) ішкі графикалық буфер ұйымдастырады

да, экран түсі, сызықтар түсі анықталып, дисплей адаптері графикалық режимге кіреді. Экран тазаланып, курсор сол жақ жоғарғы бұрышқа орнатылады.

Егер BGI-файлдар ағымдағы бумада болса, *initgraph()* функциясының үшінші параметрі ретінде бос орын мәнін беруге болады

```
initgraph(&Driver,&Mode," ");
```

Графикалық режимнен шығу және оған бөлініп берілген жады бөлігін босату үшін, яғни бейнеадаптер буферін тазартып, бұған дейінгі мәтіндік режимді қалпына келтіру мақсатында мына функция қолданылады

```
closegraph();
```

Графикалық режимде сызық түстерін, тұйық сызықтар ішін түрлі түске бояуға болады. Ол үшін мәтіндік режимдегідей түстердің кодтары және олардың ағылшынша атаулары қолданылады.

Графикалық режимнен уақытша мәтіндік режимге ауысу мақсатында мынадай функция қарастырылған

```
restorecrtmode();
```

Ол *initgraph()* функциясын пайдаланғанға дейін болған мәтіндік режимді қайта орнатады, буферде (экран көрінісі де) сақталған мәтін қалпына келмейді, өйткені ол `initgraph()` функциясы арқылы өшірілген болатын.

Графикалық режимге қайтып оралу мына функция арқылы атқарылады

```
setgraphmode(gm);
```

Функция аргументі ретінде қолданылған драйверге қатысты режимнің бүтін сан түріндегі нөмірі қарастырылады.

Курсор орнын анықтау функциялары – `getx();` `gety();`

Функциялары курсор тұрған орын координаталарын *x (y)* береді.

```
maxx(); getmaxy();
```

getmaxx() функциясы экранның оң жақтағы ең шеткі нүктесінің *x* координатасын анықтайды, ал **getmaxy()** функциясы – экранның ең төменгі *y* нүктесінің координатасын анықтайды.

Сызықтар түсін тағайындау функциясы

```
setcolor(col);
```

мұндағы **`col`** – түс атауы немесе түс коды. Сызық және мәтіндер түсі осы функциямен беріледі. Ал сызық типі **`setlinestyle()`** функциясы арқылы тағайындалады.

Түстер бүтін сан түріндегі кодпен немесе константа түрінде бас әріппен жазылатын ағылшынша түс атауларымен беріледі.

Түстердің стандартты белгіленулері төмендегідей:

Көмескі түстер	**Кодтары**	**Ашық түстер**	Кодтары
Қара (BLACK)	0	Қара қошқыл (DARKGRAY)	8
Көк (BLUE)	1	Көкшіл (LIGHTBLUE)	9
Жасыл (GREEN)	2	Ақ жасыл (LIGHTGREEN)	10
Көгілдір (CYAN)	3	Ақшыл көк (LIGHTCYAN)	11
Қызыл (RED)	4	Қызғылт (LIGHTRED)	12
Күлгін (MAGENTA)	5	Қызғыш (LIGHTMAGENTA)	13
Қоңыр (BROWN)	6	Сары (YELLOW)	14
Сұр (LIGHTGRAY)	7	Ақ (WHITE)	15

Мысалы: **`setcolor(YELLOW); setcolor(3); setcolor(5);`**

Экранның фоны түсін өзгерту функциясы

`setbkcolor(түсі);`

Мысалы: **`setbkcolor(BLUE); setbkcolor(14);`**

Мәтін шығару функциялары

Графикалық режимде экранға мәтін шығарарда символдар көлемін, бағытын (тік, көлденең), бекітілген бірнеше қаріп түрінің бірін таңдау мүмкіндіктері бар. Осындай **символдар параметрлері мынадай функция** арқылы беріледі:

`settextstyle(Қаріп,Бағыты,Көлемі);`

outtextxy және **outtext** функциялары арқылы шығарылатын мәтіндердің қаріп түрін, көлемін және бағытын тағайындайды. **Қаріп** *(шрифт)* параметрі ретінде төмендегі константалардың бірін пайдалануға болады.

Константа	Мәні	Қаріп типі
DEFAULT_FONT	0	Стандартты. Әрбір символ 8 х 8 пиксел көлемдегі квадрат ішіне шығарылады.
TRIPLEX_FONT	1	Triplex шрифті (TRIP.CHR файлы)
SMALL_FONT	2	Майда шрифт (LITT.CHR файлы)
SANSSERIF_FONT	3	SansSerif тік шрифті (SANS.CHR файлы)
GOTHIC_FONT	4	Готикалық шрифт (GOTH.CHR файлы)

Орыс қаріптері тек стандартты түрде ғана (default_font) шығарылады. Бағыты параметрі берілген мәтінді тік немесе көлденең беруді тағайындайды, оның мәндері:

HORIZ_DIR	0	Символдар солдан оңға қарай жазылады
VERT_DIR	1	Символдар төменнен жоғары қарай жазылады

Көлемі 1-ден 10-ға дейін өзгере алады. Бұл шама DEFAULT_FONT стандартты қарпі (8x8 пиксел) үшін әр символды неше есе үлкейту керек екендігін көрсетеді (стандартты матрица 8*8 болғандықтан, егер көлемі 4 болса, онда символдар 32*32 пикселдік матрицаға дейін үлкейеді). Ал қалған қаріптер үшін бұл параметр сызықтық емес, экспоненциалдық масштабтау шкаласын көрсетеді. Символдың негізгі нұсқасы көлемі 4-ке тең болып саналады, сондықтан ол 7 болса, онда символдар 2 есе өседі; егер 8 болса - 3 есе; ал 9 болса - онда 4 есе ұлғаяды. Символдар әрқашанда үздіксіз жіңішке сызықтармен жазылады.

Экранға мәтін шығару функциясы –

```
outtext ("Мәтін");
```

қостырнақшаға алынған мәтінді курсор тұрған орыннан бастап экранға шығарады. Мәтін ішінде басқару символдары болмауы тиіс, мысалы, \n. Шығарылатын символдар түсі setcolor, шрифт типі — settextstyle функцияларымен беріледі. Мысалы:

```
setcolor(4);  moveto(250,10);outtext("Омаров Марат");
```

Экрандағы көрсетілген орынға мәтін шығару функциясы – outtextxy(x, y, "мәтін");

Курсорды алдымен x, y нүктесіне орналастырып алып барып, мәтінді экранға шығарады. Мәтін шығарылған соң, курсор бұрынғы орнында (x,y) қалады.

Шығарылатын символдар түсін – setcolor(), қаріп типін – settextstyle() функциясымен беруге болады. Мысалы:

```
setcolor(4); outtextxy(250,2,"Омарова Айман");
```

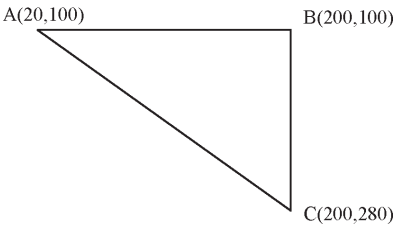
Компьютер бейнежадына пиксел жазу функциясы –

```
putpixel(x, y, Түсі);
```

координатасы (**x, y**) пиксел-нүктені ***Түсі*** санына байланысты бояп шығады. ***Түсі*** ағылшынша сөзбен немесе кодпен беріледі. Бұл

функцияны цикл ішіне орнатып, сызықтар сызуға болады. Мысалы:

```
setcolor(BLUE);
for(x=20; x<=200; x++)
  putpixel(x,100, BLUE);
for(y=100; y<=280;y++)
  putpixel(200,y, BLUE);
line(200,280,20,100);
```



16.2-сурет. Экранға үшбұрыш салу

Түрлі сызықтар салу үш функция арқылы орындалады:

1) line (x1, y1, x2, y2);

x1,y1 нүктесінен *x2,y2* нүктесіне дейін сызық сызады, бірақ мұнда курсор орны өзгермейді.

2) lineto (x, y);

Курсор тұрған нүктеден **x,y** нүктесіне дейін сызық салады, мұнда курсор бұрынғы орнынан жаңа орынға ауысады.

3) linerel (dx, dy);

Курсор тұрған **x, y** нүктесінен **x+dx, y+dy** нүктесіне дейін сызық салады, яғни сызық координаталары салыстырмалы түрде беріледі. Мұнда да курсор бұрынғы орнынан жаңа орынға ауысады.

Сызықтар осыған дейінгі *setcolor( )* функциясы тағайындаған түспен және *setlinestyle( )* функциясы арқылы орнатылған стильмен сызылады. Мысалы, ромб сызудың екі тәсілін келтірейік.

1 тәсіл

```
setcolor (GREEN);
moveto(100,10);lineto(50,90);lineto(100,170);
setcolor(RED);lineto(150,90);lineto(100,10);
```

2 тәсіл

```
setbkcolor(WHITE); setcolor(GREEN);
line(100,10,50,90); line(50,90,100,170);
setbkcolor (RED);
line(100,170,150,90);line(150,90,100,10);
```

16.2. Сызық стильдерін беру

Геометриялық фигуралар сызықтарының қалыңдығы мен түр сипатын беру функциясы

```
setlinestyle(tip,obr,tol);
```

сызықтар стилін береді, мұндағы **tip** – алдын ала анықталған константа, ол сызық ти-пін береді. **tol** – сызық қалыңдығын анықтайтын константа (**NORM_WIDTH** – қалыпты, **THICK_WIDTH** – қалыңырақ). Егер программалаушы анықтаған сызық типі қолданылатын болса, онда **obr** мәні төрт таңбалы оналтылық константа және ұзындығы 16 пиксел сызық кесіндісін кодтау үлгісі болуы тиіс.

tip константасының мүмкін мәндері (сызықтық элементтер үшін)

0	SOLID_LINE	(жай сызық)
1	DOTTED_LINE	(пунктир сызық – нүктелерден тұрады)
2	CENTER_LINE	(штрих-пунктир сызық – нүктелер мен сызықшалардан тұрады)
3	DASHED_LINE	(DOTTED_LINE сызығынан ұзындау келген пунктир сызық)
4	USERBIT_LINE	(программалаушы анықтаған сызық)

tol – қалыңдық параметрінің мүмкін мәндері

1	NORM_WIDTH	(қалыңдығы бір пиксел)
3	THICK_WIDTH	(қалыңдығы үш пиксел)

Obr параметрі **tip** константасы мәндері 4-ке тең болғанда ғана жұмыс істейді (қалған жағдайларда ол қарастырылмайды, сондықтан оны 0 деп алуға болады). Оның көмегімен кез келген периодты түрде қайталанатын сызық түрін бере аламыз, оның периоды 16 пиксел болуы тиіс. Егер сызықта жарықтанатын пиксел керек болса, үлгіде 1-ге тең бит, болмаса – 0-ге тең бит алу керек. Сонымен, **obr** мәні төрт таңбалы оналтылық константа болып, ол ұзындығы 16 пиксел сызық кесіндісін кодтау үлгісі болуы тиіс.

Мысалы, пунктир сызық үшін алынған үлгі мынадай болуы мүмкін: 0x3333, бұл мынадай биттер тізбегіне сәйкес келеді 0011 0011 0011 0011.

Графикалық режимде экрандағы тұйық сызық іші белгілі бір түспен боялуы мүмкін.

Штрихтау сызықтарын тағайындау және тұйық аймақты бояу функциясы – setfillstyle(stil,col);

аймақты бояу, толтыру стилін береді, мұндағы **stil** – алдын ала мәні анықталған константа, стильді береді, оның мәндері төменгі кестеде көрсетілген; **col** – түс кодына сәйкес бүтін сан немесе ағылшынша түс аты; ол **setcolor** функциясын анықтауда көрсетілген.

Бояу тәсілін анықтайтын stil параметрінің мүмкін мәндері

EMPTY_FILL	0	аймақты фон түсімен толтыру
SOLID_FILL	1	көрсетілген түспен толтыру
LINE_FIIILL	2	көлденең штрих сызықтармен толтыру
LTSLASH_FILL 3	4	5 градус оңға қиғаш жіңішке штрих сызықтармен толтыру
SLASH_FILL	4	45 градус оңға қиғаш қалың штрих сызықтармен толтыру
BKSLASH_FILL штрих	5	45 градус солға қиғаш жіңішке сызықтармен толтыру
LTBKSLASH_FILL штрих	6	45 градус солға қиғаш қалың сызықтармен толтыру
HATCH_FILL	7	торсызықты штрихтармен толтыру
XHATCH_FILL	8	45 градус оңға қиғаш сирек штрих сызықтармен толтыру
INTERLEAVE_FILL	9	45 градус қиғаш жиі штрих тор сызықтармен толтыру
WIDE DOT_FILL	10	сирек нүктелермен толтыру
CLOSE_DOT_FILL	11	жиі нүктелермен толтыру

Сызық стилі мен түсі көптеген функцияларда (bar,bar3d,sector, т.б.) қолданылады. Енді бір мысал келтірейік. / /

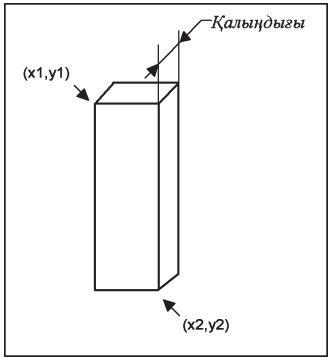
Әртүрлі сызықтар салу мысалы#include

```
<graphics.h>
#include <conio.h>
#include <stdio.h>
main(){
int gdriver=DETECT;
int gmode;
int x, y, xk = 500;
```

```
initgraph (&gdriver,&gmode,"C:\\TC\\bgi");
setcolor(RED); x=80; y=30;
moveto(x,y); outtext("жай сызық - SOLID_LINE");
setlinestyle(SOLID_LINE,0,NORM_WIDTH);
y+=20; line(x,y,xk,y);
y+=40; moveto(x,y);
outtext("пунктир сызық – DOTTED_LINE");
setlinestyle (DOTTED_LINE,0,NORM_WIDTH);
y+=20; line(x,y,xk,y); y+=40; moveto(x,y);
outtext("штрих-пунктир сызық – CENTER_LINE");
setlinestyle (CENTER_LINE,0,NORM_WIDTH);
y+=20; line(x,y,xk,y);
y+=40; moveto(x,y);
outtext("штрихі ұзынша пунктир сызық");
setlinestyle(DASHED_LINE,0,NORM_WIDTH);
y+=20; line(x,y,xk,y);
y+=40; moveto(x,y);
outtext("қалың сызық");
setlinestyle(SOLID_LINE,0,THICK_WIDTH);
y+=20; line(x,y,xk,y);
getch(); closegraph();}
```

16.3. Тұйық сызықтар салу

Тіктөрбұрыш контурын салу үшін
rectangle(xl,yl,x2,y2);
функциясы қолданылады, ол сол жақ жоғарғы бұрышы – *x1,y1,* оң жақ төменгі бұрышы - *x2,y2* болып келетін төртбұрыш сызады. Сызық типі setlinestyle функциясымен, ал түсі – setcolor функциясымен анықталады.



16.3 сурет. Параллелепипед салу

Іші боялған тіктөрбұрыш салу үшін
bar(xl,yl, x2,y2);
функциясы пайдаланылады, *x1, y1* сол жақ жоғарғы бұрыштың, ал *x2 , y2* – оң жақ төменгі бұрыштың координаталары.

Мұның қабырғалары өстерге параллель болады, контуры көрсетілмейді. Оны бояу стилі *setfillstyle()* функциясымен анықталады.

Алдыңғы жақ беті боялған параллелепипед салу функциясы

bar3d(*x1,y1,x2,y2,Қалыңдығы, Ж_Беті*);

сыртқы контуры сызылған паралелепипед салады (16.3 сурет), мұндағы *x1,y1* параметрлері сол жақ жоғарғы бұрыштың, ал *x2,*у*2* – оң жақ төменгі бұрыш координаталары. *Қалыңдығы* параметрі – алдыңғы және артқы жақтарының ара қашықтығы, егер ол 0 болса, онда айналасы сызылған төртбұрыш салынады. *Ж_Беті* параметрі – жоғарғы жақ шеттерін сызуды (1 – TOP_ON) көрсетеді. Егер ол 0 (TOP_OFF) болса, сызықтар сызылмайды (бірінің үстіне бірі қойылған бірнеше параллелепипед салу кезінде қажет болады).

Мысалы, бірінің үстіне бірі қойылған бірнеше паралелепипед салу:

```
   setbkcolor (WHITE);
setcolor (GREEN);
bar3d(230,50,250,150,15,1);
bar3d(220,150,260,180,15,1);
bar3d(300,150,340,180,15,0);
bar3d(300,50,340,150,15, 1);
```

Көпбұрыш салу функциясы

drawpoly(НүктелерСаны,Координаталары);

түзу сызықтардан тұратын тұйық аймақты көпбұрыш сызады. *НүктелерСаны* параметрі көпбұрыш төбелері санын, *Координаталары* параметрі сол төбелер координаталарын жиым элементтері ретінде береді. Жиымның 0-ші және 1-ші элементтері алғашқы нүкте координаталары, 2-ші, 3-ші элементтері – екінші нүкте координаталары, т.с.с. Енді осы функцияны пайдаланып, алты бұрыш салайық.

```
// көпбұрыш салу
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
```

```
main(){
int gdriver=DETECT; int gmode;
initgraph (&gdriver,&gmode,"C:\\TC\\bgi");
int x,y,t[14]={450,150,500,350,400,400,
               150,400,50,150, 250,80, 450,150 };
setcolor(WHITE); drawpoly(7,t);
settextstyle(3,HORIZ_DIR,1);
outtextxy(458,135,"C(450,150)");
outtextxy(508,335,"D(500,350)");
outtextxy(400,400,"E(400,400)");
outtextxy(150,400,"F(150,400)");
outtextxy(38,138,"A (50,150)");
outtextxy(243,55,"B(250,80)");
getch(); closegraph();}
```

Іші боялған көпбұрыш салу үшін қолданылатын функция fillpoly (НүктелерСаны, координаталары);

мұндағы **НүктелерСаны** – төбелердің саны, **координаталары** – жиым элементтері түрінде берілген төбе координаталары. Әрбір төбе координатасы екі бүтін санмен беріледі. Бұл функция төбелердің алғашқы нүктесін соңғы нүктесімен қосып, сызықтарды тұйықтап, ішін бояйды. Сызық типі мен бояу түсі *setfillstyle()* және *setfillpattern()* функцияларымен анықталады

Шеңбер сызу функциясы circle(x,y,r);

радиусы *r* (бүтін сан), центрінің координаталары (x, y) болатын шеңбер сызады.

Сызық түсі **setcolor()** функциясымен беріледі. Мысалы, қызыл түсті 5 шеңбер сызайық:

```
...
setcolor(RED);
for(r=5;r<=25; r+=5)
 circle(320,240,r); ...
```

Доға сызу функциясы

arc(x,y,БұрышБасы,БұрышСоңы, Радиус);

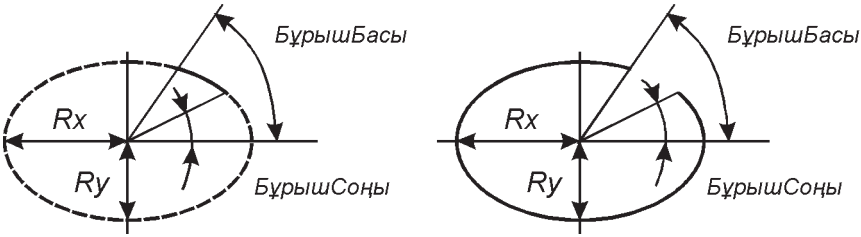
центрінің координатасы *(x, y)*, *радиусы* берілген доға сызады. **БұрышБасы, *БұрышСоңы*** параметрлері бұрышты градуспен горизонталь *x* өсінен бастап, сағат тіліне қарсы бағытта

береді. Радиус параметрі доға радиусын бүтін санмен береді. Бұрыштар мәні периодына сәйкес эквивалентті түрде [0..360] интервалындағы мәндерге келтіріледі. Мысалы, **arc(x,y,-45,45,r)** және **arc(x, y, 675, -315,r)** шеңбердің бір ширегіне сәйкес бір доғаның екі түрде берілуін көрсетеді.

Эллипс доғасын сызу функциясы

`ellipse(x,y,БұрышБасы,БұрышСоңы,rx,ry);`

центрінің координаталары *(x, y)* эллипс немесе эллипс доғасын (16.4 сурет) сызады. **БұрышБасы, БұрышСоңы** параметрлері доғаның басы мен соңын градуспен сағат тіліне қарсы береді. **rx**, **ry** параметрлері эллипстің көлденең және тік радиустарын береді. Эллипс өстері координаталар өстеріне параллель болады.



16.4 сурет. Эллипс доғасын салу

Іші боялған контурлы эллипс салу функциясы

`fillellipse (x,y, rx,ry);`

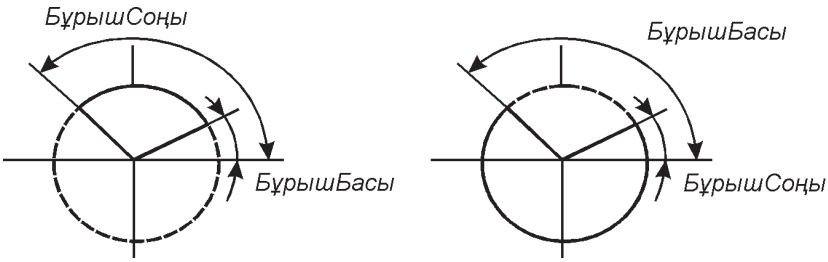
мұндағы *x, y* – центр координаталары; *rx, ry* – пикселмен берілген эллипс жарты өстері радиустары. Эллипс өстері координат өстеріне параллель болады. Эллипс ағымдағы түспен боялып шығады.

Іші боялған дөңгелек сектор сызу функциясы

`pieslice(x,y,БұрышБасы,БұрышСоңы,Радиус);`

радиусы *Радиус,* центрі **(x,y)** нүктесіндегі дөңгелек сектор сызады. **БұрышБасы,БұрышСоңы** параметрлері шеңбер секторының бастапқы және соңғы бұрыштарын градуспен сағат тіліне қарсы анықтайды. Егер **БұрышБасы** = 0, ал **БұрышСоңы** = 360 болса, онда **pieslice** функциясы шеңбер сызып шығады. Бұрыштарды [0..360] шегіне (диапазонына) келтірген соң, сектор

мәні кіші бұрыштан мәні үлкен бұрышқа қарай сызылады, сол себепті OX өсінің оң жақтағы бағытын кесіп өтетін сектор салуға



16.5 сурет. Секторлар салу

болмайды. Сектор контуры (доға мен екі радиус) сектор боялған соң сызылады, ал сызық типі мен қалыңдығы *setlinestyle()* функциясымен анықталады. Егер контурсыз сектор салу керек болса, мынадай тәсілді пайдалануға болады (16.5 сурет):

```
setcolor(BLACK); setbkcolor(BLUE) ;
setwritemode(XOR_PUT);
setfillstyle(WIDE_DOT_FILL,RED);
pieslice(200,100,45,90,50);
```

Іші боялған контурлы эллипс секторын салатын функция sector (x, y, бұрыш_басы, бұрыш_соңы, rx, ry);

Бұл функция *pieslice()* функциясы тәрізді жұмыс істейді.

```
// Жазуы бар секторлар sector2.cpp
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <graphics.h>
void main(){
int gd=DETECT,gm,i,x,y;
initgraph (&gd,&gm,"c:\\TC\\bgi");
setcolor(BLACK); setbkcolor(BLUE);
setwritemode(XOR_PUT);
setfillstyle(WIDE_DOT_FILL,RED);
```

```
pieslice(200,100,45,90,50);
setbkcolor(BLUE); setcolor(RED);
setfillstyle(1,3);
x=getmaxx()/2; y=getmaxy()/2;
pieslice(x,y,270,360,100);setfillstyle(1,2);
pieslice(x,y,0,270,100);settextstyle(1,0,2);
moveto(x-20,y-40); outtext("75%");
moveto(x+20,y+20); outtext("25%");
getch();closegraph();}
```

Тұйық сызықпен қоршалған аймақтың ішін бояу функциясы floodfill(x,y,шекара);

мұндағы *x, y* – боялатын аймақ ішіндегі нүкте координатасы. Тұйық аймақты қоршаған сызық контурында тесік болмауы тиіс, әйтпесе бояу бүкіл экранды сол түске бояп жібереді. Контур түсі *шекара* түсімен бірдей болуы тиіс. Бояу түсі мен типі *setfillstyle()* функциясымен орнатылады. Енді бір мысал келтірейік.

```
//боялған шеңберлер, эллипстер салу – kr_krug.cpp
#include <conio.h>
#include <stdlib.h>
#include <graphics.h>
void main(){
int gd=DETECT,gm,r,x=120,y=240;
initgraph(&gd, &gm,"C:\\TC\\BGI");
setcolor(RED);       //сызықтар қызыл түсті
setbkcolor(BLUE);    //фон көк түсті
for(r=0; r<80; r++) //концентрлі
 circle(x,y,r);      //80 қызыл шеңбер салу
setfillstyle(SOLID_FILL,RED);
x+=200; circle(x,y,80); //қызыл шеңбер
floodfill(x,y,RED);        // іші де қызыл
x+=200;
fillellipse(x,y-150,80,50); //қызыл эллипс
setfillstyle(SOLID_FILL,GREEN);
fillellipse(x,y,80,50);     //жасыл эллипс
setfillstyle(SOLID_FILL,YELLOW);
```

```
fillellipse(x,y+150,80,50); //сары эллипс
getch();
closegraph(); }
```

Терезе ашу функциясы

Графикалық режимде экран ішінде өз координаталық жүйесі бар төртбұрышты басқа терезені ашуда қолданылатын келесі функция қолданылады:

```
setviewport(x1,y1,x2,y2,clip);
```

Мұндағы ***x1, y1*** – терезенің сол жақ жоғарғы бұрышы координаталары; ***x2, y2*** – оң жақ төменгі бұрышының координаталары; ***clip*** – қию параметрі.Егер ***clip*** параметрі 1 болса, онда терезеге сыймайтын бейне элементтері қиылып алынып тасталады; ал егер де ол 0 болса, терезе шекаралары есепке алынбай, бейне толық экранда көрсетіледі. Бұл функция дұрыс орындалса, графикалық курсор терезенің координаталар басына орналасады.

Терезені тазалау функциясы `clearviewport ();`

Тағы бір мысал келтірейік.

```
//Диагональ сызу, жаңа терезелер ашу primer3.cpp
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <graphics.h>
void main(){
int gd=DETECT,gm,x,y;
initgraph(&gd,&gm,"c:\\TC\\bgi");
//жасыл экранда қалыңдығы 3 пиксел диагональ сызу
setbkcolor(GREEN); setcolor(RED);
setlinestyle(SOLID_LINE,0,3);
x=getmaxx(); y=getmaxy();
printf("  xmax=%d ymax=%d ",x,y);
line(0,0,x,y); // бір перне басып,экран тазалау
getch();
cleardevice(); // терезе ашу, оны тазалау
setviewport(100,100,600,400,1);
clearviewport(); setcolor(GREEN);
rectangle(10,30,450,255);
circle(230,145,55);
```

```
settextstyle(DEFAULT_FONT,HORIZ_DIR,3);
//орыс әріптері тек DEFAULT_FONT қарпінде
setbkcolor(BLUE);
setcolor(WHITE);
//Мәтін терезенің сол жақ жоғарысында
outtextxy(1,1, "Жаңа терезе");
getch();
closegraph();}
```

Бақылау сұрақтары

1. *IBM PC компьютерлері бейнемониторының қандай түрлері бар? Олардың айырмашылығы неде?*
2. *Графикалық режимнің атқаратын қызметі қандай? Оны программада қалай іске қосады?*
3. *Графикалық режим қандай функция арқылы орнатылады?*
4. *Адаптерлердің мынадай типтерінің CGA, EGA, VGA бір-бірінен айырмашылығы неде?*
5. *Монитордың мәтіндік және графикалық режимдерінің мүмкіндіктері неге әртүрлі болады?*
6. *Драйвер деп нені айтады? Графикалық драйвер ше? Ол қандай қызмет атқарады?*
7. *Экранның түсін және оған шығарылатын жол символдарының түсін қалай өзгертуге болады?*
8. *Графикалық режимде курсор бола ма?*
9. *Символдарды қалай жыпылықтатып қоюға болады?*
10. *Курсорды экранның кез келген нүктесіне қалай көшіреміз?*
11. *Графикалық режимде экранда қалай терезе құруға болады?*
12. *Графикалық режимде экранды қалай тазартуға болады?*
13. *Clrscr, Clreol, Delline және Insline функцияларының ортақ қасиеттері мен айырмашылықтары.*

Тапсырмалар

1. Уақытты есептеп отыратын сиқыршымен кездестіңдер делік және ол сендерге жұлдыздар сырын ашып береді деп ойлайық. Неге аспанда жұлдыздар жымыңдайды? Аспандағы әрбір жұлдыз — бір адамның өмірі. Сендерге көне жұлдыздарды дәл уақытында өшіріп, жаңасын дер кезінде жағып отыру керек. Уақыт есептеушінің жұмысын көрсететін программа жазу керек. Есеп модель құрудан басталады: экрандағы әрбір жұлдыз нүктемен көрсетіледі және барлығы 20 жұлдыз жанып тұрсын делік (кездейсоқ түрде таңдап алынған): жанып тұрған бірінші нүктені өшіріп, оны кез келген

басқа орыннан жандыру керек, содан кейін екіншісін, т.с.с. Жиырмасыншы жұлдыз өшкеннен кейін қайтадан біріншісіне көшу керек.

2. "Жаңбыр" программасын жазыңдар — экранда (кездейсоқ күйде) 50 нүкте жоғарыдан төменге қарай қозғалып келе жатқандай болуы тиіс. Нүктелердің бірі экранның төменгі шекарасына жеткенде, ол қайтадан экранның жоғарғы бөлігінде пайда болуы керек.
3. "Жаңбыр" программасын кейбір нүктелер шапшаң, кейбіреулері баяу қозғалатындай етіп өзгертіңдер.
4. "Жаңбыр" программасын нүктелер өз қозғалысын төменгі сол жақ бұрыштан бастап, жоғарғы оң жақ бұрышта аяқталатындай етіп өзгертіңдер.
5. Қағазға оралған "кәмпиттің" бейнесін: диагональдары жүргізілген горизонталь тіктөртбұрыш пен оның екі жағына жалғасып жатқан тең қабырғалы үшбұрыштар салыңдар. Тіктөртбұрыштың сол жақ төменгі төбесінің координатасы (x, y), биіктігі a-ға, ұзындығы $2a$-ға тең болсын. Үшбұрыштың биіктіктері a/2. Тікбұрыштың диагональдары мен қабырғаларынан пайда болған қарама-қарсы екі үшбұрышты бояңдар.
6. Үш тісті және жарты дөңгелек пішінді аркасы бар мұнара салыңдар. Мұнараның сол жақ төменгі төбесінің координатасы (x, y), табанының ұзындығы a. Қалған өлшемдерін a арқылы өрнектеп табыңдар. Мұнараны бояңдар.
7. Куб салыңдар. Кубтың алдыңғы сол жақ төменгі төбесінің координатасы (x, y), бүйір қыры a. Жоғарғы бүйір бетін бояңдар, оң жақ бүйір бетінде диагональ жүргізіңдер.
8. Алтыбұрышты тік пирамида салыңдар. Табанының алдыңғы сол жақ төбесінің координатасы (x, y), табанындағы қыры a. Қалған өлшемдерін a арқылы өрнектеп табыңдар. Пирамиданың екі бүйір жақтарын бояңдар.
9. Тіктөртбұрышты қиық пирамида салыңдар. Пирамида табанының төменгі сол жақ төбесінің координатасы (x,y), табанындағы қыры a, жоғарғы жақтың қыры — a/2. Қалған өлшемдерін a арқылы өрнектеп табыңдар. Пирамиданың оң жақ бүйір бетін бояңдар.
10. Қарайтылған "Е" әрпі тәрізді фигура салатын программа құрыңдар. Әріп құрайтын нүктелердің абсциссалары мен ординаталары жиым элементтері ретінде берілген. Сурет салу үшін циклге енгізілетін тек қана бір LINE операторын пайдаланыңдар.
11. Экранда боялған шырша салатын программа жазыңдар. Сурет элементтерін жиымда сақтап, өшіріп және содан кейін қайтадан экранға шығарыңдар.
12. Боялған 7 шыршаның суретін салатын программа жазыңдар. Шыршалар бір-біріне ұқсас, горизонталь бойында бір қалыпта орналасады, шыршалардың биіктігі солдан оңға қарай сызықтық түрде

өсуі керек. Шыршаларды бойлап Қызыл Телпек оларды уақытша "үсіп қалмасын" деп бір затпен жауып шығатын болсын.

13. Графикалық режимде ESC пернесін басқанша монитор экранына кездейсоқ күйде бірсыпыра нүктелер шығаратын программа құрыңдар.
14. Графикалық режимде ESC пернесін басқанша монитор экранында (100, 100) – (300, 200) тіктөртбұрышының нүктелерін кездейсоқ күйде жоятын программа құрыңдар.
15. Графикалық режимде ESC пернесін басқанша монитор экранының центрі (200, 200) нүктесімен дәл келетін және радиусы 80 болатын дөңгелек нүктелерін кездейсоқ күйде жоятын программа құрыңдар.
16. Шахмат тақтасының бейнесін салыңдар.
17. Экранда қарайтылған **М** әрпі пішіндес фигура салатын және бағыттауыш тілсызық (→, ←,↑, ↓) пернелер арқылы басқарылып, фигураның көлемін + және – пернелерінің көмегімен өзгертетін программа құрыңдар.
18. Циклдік операторды пайдаланып 10 басқышы бар саты салыңдар, оның жұп және тақ нөмірлі басқыштарының көлемі әртүрлі болатын программа құрыңдар.
19. Шеңберге іштей бесбұрышты жұлдыз салыңдар.
20. Қызыл экранға 0..9 аралығындағы 1000 кездейсоқ санды ақ түспен шығару қажет. Мұнан соң экран түсін жасылға бояп, сандарды сары түспен шығарыңдардар.
21. Экранды барлық (8) фон түстеріне 5 секунд кідіріспен бояп шығыңыздар. Экранның сол жақ жоғарғы бұрышына оның нөмірін жазып қою керек.
22. Қара экранға 1-ден 16-ға дейінгі сандарды 16 түрлі түске бояп шығару қажет. Әрбір сан бөлек жолда орналасатын болсын.
23. Алдын ала тазартылған экранды ақ түске бояп, жеті атаңыздың атарын жеті түрлі түспен шығарыңдардар.
24. Қара экранға өзгеріп отыратын кездейсоқ түстер арқылы 200 "+" белгісін кездейсоқ түрде берілетін координаталық нүктелерге басып шығару керек.
25. Экранның жоғарғы жағына екі терезе салып, оның біріншісінің ішіне өз атыңызды, екіншісіне – фамилияңызды жазып қойыңыз. 10 секундтан кейін терезелер ішін тазартып, өз аты-жөніңіздің орнына досыңыздың аты мен фамилиясын орналастырыңыз.
26. Экранның сол жақ шетіне көк түсті терезе салып, оны оң жаққа қарай қадамын бір символ етіп жылжытып отыратын программа құру керек.
27. Экранға біртіндеп үлкейе бастайтын шағын қызыл терезе салу қажет. Үлкею қадамын *X* координатасы үшін 3, ал *У* координатасы үшін 1 етіп алыңыздар.

17 С ПРОГРАМЗАСЫН ОРЫНДАУ ОРТАСЫ

17.1. Турбо С редакторының терезесі

Турбо С программалау жүйесі С тілінде программа құрастырып, оны орындауға мүмкіндіктер береді. Жалпы, тұтынушы мынадай әрекеттерді орындай білуі керек:

- программаның мәтінін жазып, дискіде программа файлы ретінде сақтау;
- программаны ***компиляциядан*** өткізіп, егер синтаксистік қателері бар болса, оларды түзету;
- программаны орындап, нәтижесін алу. Сонымен, кез келген программа оны теру, компиляциялау, құрастыру, атқарылушы модульді жасау және орындап нәтиже алу сатыларынан өтуі тиіс.

С тілінің біріктірілген (интегралданған) ортасында программа орындау төмендегідей қадамдардан тұрады:

1) компилятор қажет файлдарды іздеп тауып алуы үшін ортаның параметрлерін тағайындау;

2) программалық файлды редактор ортасына жүктеу немесе теру;

3) атқарылатын модульді (орындалатын файлды) жасау;

4) программаны іске қосу және орындау;

5) программада қате болса, оны жөндеп түзету. Программа 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 (17.1-сурет).

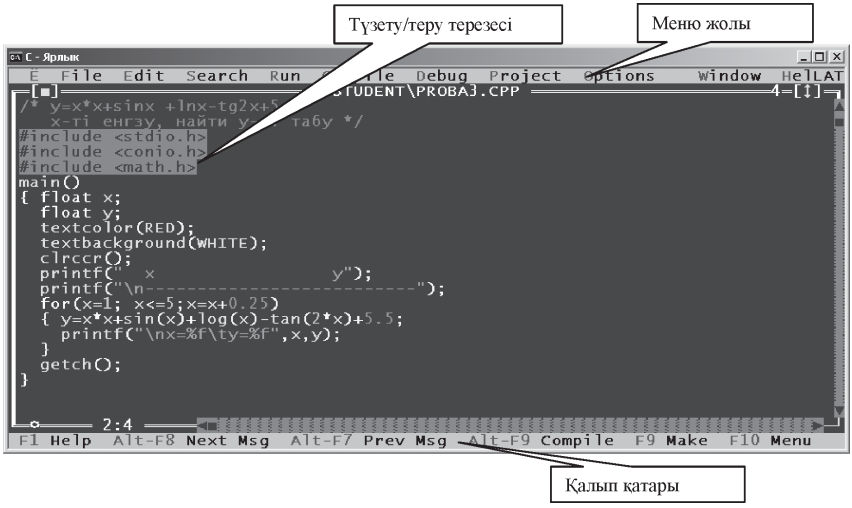
Турбо С ортасын автоматты түрде жүктеп, орыс (қазақ) әріптерін теру мүмкіндігін беретін C.bat пакеттік файлының мәтінін мынадай түрде құрастыруға болады:

```
c:
cd TC\BIN
rk.com
tc.exe
```

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 (17.1-сурет):

- 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.



17.1-сурет. Турбо С біріктірілген ортасы

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.

17.2 Меню командалары

Программа 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) – программаны орындау.

•**Compile** (компиляция) – программаны компиляциядан өткізу.

•**Debug** (түзету) – программаны 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

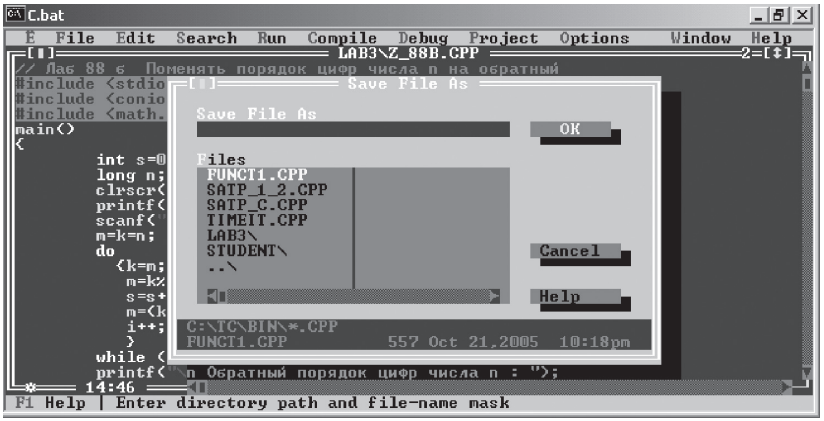
Change dir – директорийды өзгерту

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.



17.2-сурет. Дискіге мәлімет жазу терезесі

Edit менюінің ішкі командалары (17.3-сурет):

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.

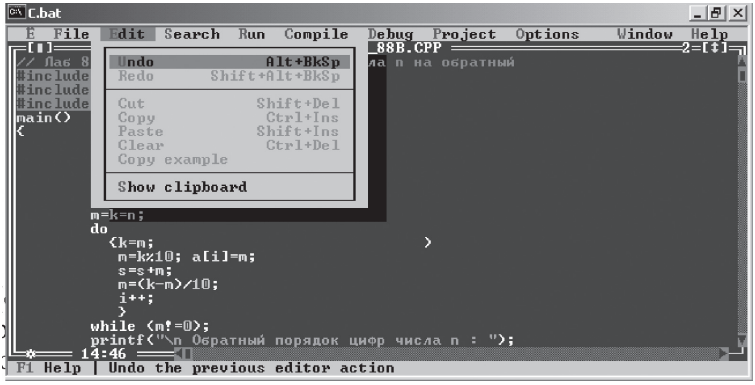
Search менюінің ішкі командалары:

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.



17.3-сурет. Edit менюінің ішкі командалары

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 программаны орындайды.**Program reset** Ctrl+F2 (сброс программы) – жөндеушінің орындап жатқан әрекетін тоқтатып, программаға бөлінген орынды босатып, барлық ашық файлдарды жабады.

Go to cursor (курсорға өту) F4 – екпінді терезедегі программаны курсор тұрған орындағы қатарға дейін орындайды.

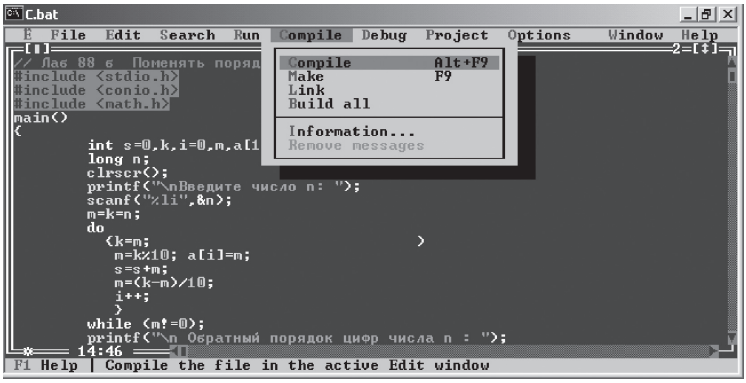
Trace into (қосалқы программаға кіріп, қадамдық орындау) F7 – программадағы операторларды қадам бойынша көрсете отырып орындайды.**Step over** (программаны қадам бойынша орындау) – программа мәтінінің бір жолына сәйкес келетін кезекті операторды біртіндеп орындайды.

Compile менюінің ішкі командалары (17.4-сурет):

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

Make (программаны жинақтау). Егер негізгі программада немесе негізгі модульде қолданылатын жеке модульдердің мәтінінде объектілік файлды алғаннан кейін өзгеріс болса, соған сәйкес модульдер қайта тексеріледі, одан кейін негізгі программа немесе негізгі модульден тұратын файл қайта құрылады.

Build all (Программа құру) – бұл команда орындалғанда, негізгі программа және негізгі модульде қолданылатын барлық модульдер қайта компиляциядан өткізіледі.



17.4-сурет. Compile менюінің ішкі командалары

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) – программада қолданылған функциялар тізбегін – стекті көрсететін қосалқы программа 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.

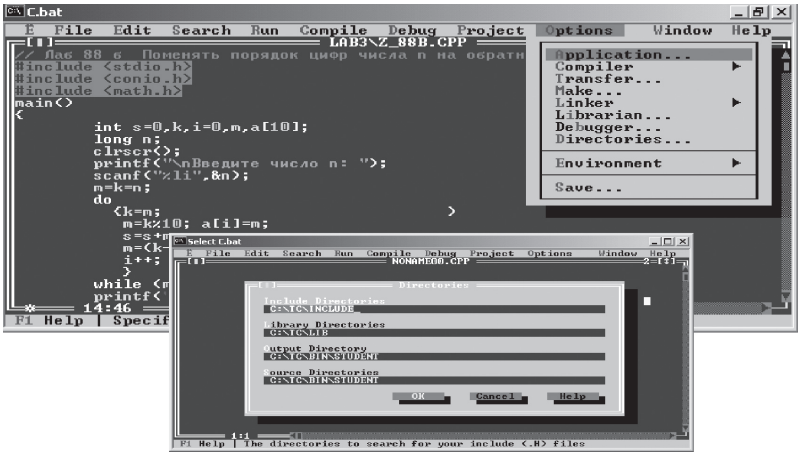
Жоба — бір-бірімен байланысты файлдар жиыны, бірнеше объектілік файлдар бірден компиляциядан өткізіліп, атқарылатын бір программа жасайды. Жоба көп файлды программалар кезінде керек. Кейде бір файлмен жұмыс істегенде де қолданылады.

Модульдік программалау барысында көпфайлдық компиляциялаусыз жұмыс істеу мүмкін емес. Көлемі үлкен программалармен жұмыс істеу кезінде ол программаның 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 *Project* менюіне 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) *Project* терезесінде 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 программа 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



17.5-сурет. Option менюінің ішкі командалары

мәндерін өзгертпей, қалдыруға болады. Мұнда түйінді сөздер арқылы түстерді (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** программалары **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 Directory** жолына 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.

Size/Move	Ctrl+F5	– терезе көлемі мен орны
Zoom	F5	– терезені ұлғайту
Tile		– сатылы (каскадты)
Next	F6	– келесі терезе

Close	Alt+F3	– терезені жабу
Output		– нәтижелік терезе
User screen	Alt+ F5	– тұтынушы экраны
List All	Alt+0	– басқа терезелерді ашу

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.

17.3 Қателер коды және олардың мәліметтері

Турбо С ортасы программа компиляциядан өткен кезде пайда болған қателер жайлы толық мәлімет береді. Қате кездескен кезде орта автоматты түрде бастапқы программа мәтінін экранға шығарып, курсорды қате табылған орынға орналастырады және редактордың жоғарғы жолында диагностикалық мәлімет пайда болады. F1 пернесінен басқа кез келген пернеге бассаңыз, жоғарғы жол бастапқы қалпына келіп, интегралданған орта редакциялау режиміне ауысады. Егер қате жайлы мәлімет шыққаннан кейін F1 пернесін басқанда, экранда қатені түзету жайлы нұсқаулар жазылған анықтама қызметінің терезесі пайда болады. Кейбір қателер бірден емес, программа мәтінін талдау барысында анықталады.

Қате нөмірі	Қате түсінігі
1	*Out of memory (Жады шекарасынан асып кету)*
2	*Identifier expected (Идентификатор көрсетілмеген)*
3	*Unknown identifier (Белгісіз идентификатор)*

4	*Duplicate identifier (Идентификатор қайталанған)*
5	*Syntax error (Синтаксистік қате)*
6	Error in real constant (*Нақты тұрақтыда қате бар*)
7	*Error in integer constant (Бүтін тұрақтыда қате бар)*
8	*String constant exceeds line (Тіркестік тұрақты берілген аймақтан тыс жатыр)*
9	*Too many nested files (Кіріктірілген файлдар саны тым көп)*
10	*Unexpected end of file (Файл соңы табылмады)*
11	*Line too long (Тым ұзын жол)*
12	*Type identifier expected (Тип идентификаторы қажет)*
13	*Too many open files (Ашылған файлдардың саны көп)*
14	*Invalid file name (Файл аты дұрыс емес)*
15	*File not found (Файл табылмады)*
16	*Disk full (Диск толып кеткен)*
17	*Invalid compiler directive (Компилятор директивасы қате)*
18	*Too many files (Файлдардың саны тым көп)*
19	*Undefined type in pointer definition (Көрсеткіш сипаттауында тип көрсетілмеген)*
20	*Variable identifier expected (Айнымалының иденфикаторы жоқ)*

17.3.1 Программа орындалу кезінде шығатын қателер

Программа орындалу кезінде анықталатын кейбір қателер экранда **Runtime error nnn at xxxx:yyyy** (xxxx:yyyy адресі бойынша nnn кезеңін орындаудағы қате) мәлімдемесінің шығуына әкеледі, мұнда nnn — қате нөмірі; **xxxx:yyyy** — адрес (сегмент немесе жылжу). Бұл мәлімдемеден кейін программа өз жұмысын тоқтатады. Программа орындалу кезінде шығатын қателер төртке бөлінеді: **СОЖ** (сұхбатты операциялық жады) **анықтайтын қателер** (1-ден 99-ға дейінгі қателер), **енгізу-шығару қателері** (100-ден 149-ға дейінгі қателер), **дағдарысты қателер** (критические ошибки) (150-ден 199-ға дейін) және **фаталды қателер** (200-ден 255-ке дейінгі қателер).

17.3.2 Операциялық жүйе анықтайтын қателер

Қате №	***Аты***
1	*Invalid function number (Функция нөмірі дұрыс жазылмаған)*
2	*File not found (Файл табылмады)*
3	*Path not found (Жол табылмады)*

4	*Too many open files (Ашылған файлдардың саны көп)*
5	*File access defined (Файлға қол жеткізуге рұқсат жоқ)*
6	*Invalid file handle (Ретсіз файлдық канал)*
12	*Invalid file access code (Файлға қол жеткізудің коды нақты емес)*
15	*Invalid drive number (Дискжетек нөмірі ретсіз)*
16	*Cannot remove current directory (Ағымдағы каталогты өшіруге болмайды)*
17	*Cannot rename across drives (Атты ауыстырғанда әртүрлі дискжетек аттарын көрсетуге болмайды)*

17.3.3 Енгізу-шығару қателері

Егер операторлардың біреуі {$1+} директивасымен компиляциядан өтсе, онда енгізу-шығару қатесі программаның орындалуын тоқтатады. {$1–} қалпында программа орындалуын жалғастырып, қате IORESULT функциясымен қайтарылады.

Қате №	***Аты***
100	*Disk read error (Дискіден оқу кезінде қате кетті)*
101	*Disk write error (Дискіге жазу кезінде қате кетті)*
102	*File not assigned (Файлға ат берілмеген)*
103	*File not open (Файл ашылмаған)*
104	*File not open for input (Файл енгізу үшін ашылмаған)*
105	*File not open for output (Файл жазу үшін ашылмаған)*
106	*Invalid numeric format (Сан форматы дұрыс емес)*

17.3.4 Дағдарысты қателер

Қате №	***Аты***
150	*Disk is write protected (Диск жазудан сақталған)*
151	*Unknown unit (Белгісіз модуль)*
152	*Drive not ready (Дискенгізгіш "дайын емес" қалып күйінде)*
153	*Unknown command (Бейтаныс команда)*
154	*CRC error in data (Бастапқы мәндерде қате бар)*
155	*Bad drive request structure length (Дискіден мәлімет аларда құрылым ұзындығы дұрыс көрсетілмеген)*
156	*Disk seek error (Дискіден оқитын құрылғы бастарын дискке орнату операциясы кезінде қате кетті)*
157	*Unknown media type (Тасымалдауыш типі белгісіз)*
158	*Sector not found (Сектор табылмады)*

159	*Printer out of paper (принтерде қағаз біттi)*
160	*Device write fault (Құрылғыға жазу кезінде қате кетті)*
161	*Device read fault (Құрылғыдан оқу кезінде қате кетті)*
162	*Hardware failure (Аппаратура жұмыс істемей тұр)*

17.3.5 Фаталды қателер Бұл қаталер программа жұмысының бірден тоқталуын жүзеге асырады.

Қате №	***Аты***
200	*Division by zero (Нөлге бөлу)*
201	*Range check error (Шекараларды тексеру кезінде қате шықты)*
202	*Stack overflow error (Стек толып кетті)*
203	*Heap overflow error (Мәлімет толып кетті)*
204	*Invalid pointer operation (Көрсеткішпен орындалатын белгісіз операция)*
205	*Floating point overflow (Жылжымалы нүктемен жұмыс жасау кезінде мәндер берілген аймақтан шығып кетті)*
206	*Floating point underflow (Жылжымалы нүктемен жұмыс жасау кезінде реті бұзылды)*
207	*Invalid floating point operation (Жылжымалы нүктемен жұмыс жасауға болмайтын операция)*
208	*Overlay manager not installed (Оверлейді басқарудың ішкі жүйесі орнатылмаған)*
209	*Overlay file read error (Оверлейлік файлды оқу кезінде қате жіберілген).*
210	*Object not initialized (Объект инициалданбаған)*
211	*Call to abstract method (Абстракты ережені шақыру)*
212	*Stream registration error (Тіркеуге алу ағынында қате бар)*
213	*Collection index out of range (Терілген индекс диапозон шекарасынан тыс жатыр)*
214	*Collection overflow error (Коллекция толып кеткен)*

Бақылау сұрақтары

1. *Turbo C біріктірілген программалау жүйесі қандай қызмет атқарады және оның терезесі неше бөліктен тұрады?*
2. *Turbo C ортасында программа орындау қандай қадамдардан тұрады?*
3. *Turbo C ортасы терезесін үлкейту/кішірейту үшін не істеу керек?*
4. *Turbo C ортасында программа теру үшін неше терезе пайдаланылады? Керекті терезеге қалай көшуге болады?*

5. *Меню қатарына және жұмыс алабына ауысу үшін не істеуге болады?*
6. File *менюінің ішкі командалары қандай?*
7. *Файлды дискіге жазу/оқу үшін қандай командалар мен пернелер қолданылады?*
8. *Файлды басқаша атпен сақтау қалай орындалады?*
9. Edit *менюінің ішкі командалары қандай қызметтер атқарады?*
10. *Turbo C терезесінде ерекшеленген бөлікті буферге қалай қиып (көшіріп) алады?*
1. *Курсор орналасқан жерге буфердегі ақпаратты қалай кірістіріп қояды?*
12. *Turbo C терезесінде ерекшеленіп белгіленген бөлікті қалай өшіріп тастайды?*
13. *Программа мәтіні ішінен тауып алуға қажетті сөзді қандай сұхбат терезеге және қалай енгізуге болады?*
14. *Run менюінің ішкі командалары қандай қызметтер атқара алады?*
15. *Орнатылған параметрлерді қолдана отырып, редактор терезесіндегі екпінді программаны қай команда орындайды?*
16. *Options менюінің командалары қандай?*
17. *Window (Терезе) меню командалары қандай қызметтер атқарады?*
18. *Программаны компиляциялауда шыққан қателер жайлы мәлімет қалай алынады?*
19. *Программа орындалу кезінде шығатын қателер қандай топтарға бөлінеді?*
20. *Қатені түзету жайлы нұсқауларды қайдан алуға болады?*

18. ПРОГРАММАЛАУ ТІЛДЕРІНІҢ ДАМУ ТАРИХЫ

Компьютердің машиналық тілінде жұмыс істеу программалаушының өте жоғары деңгейін талап етеді. Есептеу машиналарының бірте-бірте көбеюіне байланысты онымен қарым-қатынас жасауды оңайлату қажеттілігі пайда болды. Біріншіден істелген іс – компьютер жадын бөлу жұмысы автоматтандырылды және машиналық командалар үшін адамға түсінікті белгілеулер енгізілді. Осылай пайда болған компьютермен қатынасу тілі **Ассемблер тілі** деп аталды. Кейіннен көптеген тілдер дүниеге келіп, олар уақыт талабына сай мүмкіндігінше жетілдіріліп отырылды.

Программалау тілдерін жасау және дамыту жұмысының басында АҚШ-тағы атақты IBM фирмасы тұрды. Бұл фирма кәсіби программалаушылардан басқа да мамандарға алгебралық формулалармен жұмыс істеуге дағдыланған ғылыми қызметкерлер мен инженерлерге жеңіл қолдануға болатын компьютерлер шығаруға тырысты.

1953 жылы Джон Бэкус компьютерлердің IBM-704 атты нұсқасына арнап программалауды оңайлататын жаңа тіл ойлап шығаруды ұсынды. Осыған орай жаңа тілдің сөйлемдерін (операторларын) машиналық кодқа түрлендіретін программаны – трансляторды да жасап шығару керек болды.

Алдыңғы кезде мамандар Бэкустың жұмыс тобына үлкен сенімсіздікпен қараған болатын, өйткені машиналық кодты жетілдіру істерінің алғашқы ұсыныстары айналасында, "адам тілінде сөйлейтін" компьютерлер болады деген әңгіме-сөз көп болған еді.

Әйтсе де, FORTRAN *(FORmula TRANslator* — формулаларды түрлендіргіш) деп аталған тіл 1957 жылдың сәуірінде дайын болды және ол машина тіліне тек "формулаларды аударып" қана қоймай, циклдерді ұйымдастыру істерін де автоматтандыра білді.

Бұл жүйенің табысы күткен мүмкіндіктен бірнеше есе асып түсті, 1958 жылдың өзінде-ақ IBM-704 компьютерлеріндегі машиналық командалардың жартысынан астамы қолмен емес, осы Фортран тілінің трансляторы арқылы алынды.

Негізінде, Фортран жаңа программалау құралы емес, бұрынғы нақты машинаға көмекші ретінде, оны сату ісін арттыратын мүмкіндіктің бірі ретінде ғана жасалған жұмыс болатын.

Бірсыпыра кемшіліктерінің болуына қарамастан, мысалы, кейбір программалау идеологтары Фортранды "зердені тұйыққа тірейтін, синтаксис ережелерімен біріккен бұтақшалардың тізбегі, оны оқып үйренуге тиым салу" керек деп шулағанымен, бұл тіл көпшілікке кең таралып кетті және оны басқа машиналарға да икемдеп енгізу ісі жүргізіле бастады.

Икемдеу ісі (адаптация), Фортран тілінен басқа машина тілдеріне транслятор программалар жасау арқылы жүргізілді, әрине, ол оңай іс емес болатын.

Дегенмен, программалау тарихында тұңғыш рет программаны жақсы қойылған тапсырма ретінде (Фортран тілінің ережелерімен) жасау, "не жазарыңды білмейтін" тілде жазғаннан гөрі әлдеқайда жеңіл, әрі арзан екенін көрсете білді. Басқаша айтқанда, ешқандай программалаусыз-ақ есепті ықшам түрге келтіріп, оны түсінікті етіп қоя білу де үлкен өнер саналып, жоғары баға ала бастады.

Фортран тілі басқа да бір маңызды рөл атқарды: ол әртүрлі компьютерлер бір-бірімен ортақ тіл таба алатын тамаша құралға айналды.

Фортран бірнеше рет толықтырылып, өткен ғасырдың сексенінші жылдарына дейін кең пайдаланылды. Сонан кейін оны жаңа тілдер ығыстыра бастады. Соңғы кездерге дейін ол АҚШтың Аэронавтика мен ғарыштық зерттеулер жүргізетін ұлттық басқармасының (NASA) компьютерлерге арналған программалар жасайтын негізгі тілі болды. Оның әртүрлі нұсқалары осы кездерде де қуатты компьютерлер үшін есептеу программаларын жасау мақсатында кең қолданылып келеді.

Фортранның осындай кенеттен ашылған қасиеттері, оның әртүрлі компьютерлер үшін әмбебап тіл бола алатын мүмкіндіктері программалау идеологтарын қарапайым ойын программаларын жазудан босатып, оларды өмірдің кез келген кезеңінде пайдаға асатын "әмбебап, жақсы, қолайлы тамаша" тіл жасау ісіне бет бұрғызды. Оның үстіне трансляторларды жазатын олар емес, басқалар екені етенеден белгілі еді.

Программа жасаушылардың "Ең жақсы – жақсының жауы" деген ұранына қарамастан, Алгол *(ALGOrithmic Language)* деп аталған жаңа тілді жасау ісі екі жылдан аса уақыт алды және ол негізінен Европада ғана кең қолданыста болды. Бұл, сірә, Европада да программалау тілдерінің мамандары бар дегенді дәлелдеу үшін жасалған сияқты болды.

Алгол туралы белгілі информатик мамандардың бірі Грейс Хоппер былай деген болатын: "Ол үлкен поэмаға ұқсас: математика тұрғысынан қарапайым, әрі түсінікті, бірақ практикалық тұрғыдан олай дей алмаймыз".

Осылай программалау саласындағы европалық аналитиктер тілдің әмебаптығына көңіл бөлсе, біртіндеп компьютер шығарушылардың әлемдегі көш басшысына айналған IBM фирмасы одан да мықты тіл шығаруға бет алды.

IBM фирмасы 1964 жылы PL/1 *(Programming Language One* — Нөмірі Бірінші Программалау Тілі) тілін дүниеге келтірді. Сол кездегі АҚШ мамандары осы тілді – программалау тілдерін жасаушылар ойлап тапқан мүмкіндіктердің асқар шыңы болар деп үміттенді. Солай болды да. Бұл тілге көптеген жаңалықтар енгізілген еді.

Әйтсе де, супер әмбебап программалау тілдерін жасау идеясы кір жуғыш машинасы бар теледидар жасаған сияқты біраз жерде артық кетіп жатты, оның көптеген мүмкіндіктерін пайдалану оңай болмады.

Программалау тілдерінің ең жақсысын жасаймыз деп зерделерін жұқартып жатқан теоретиктер жұмысы алаңдағы жалғыз допты қуалаған футбол командасы тәрізді болды – солардың жасаған тілін ең жақсы, әрі әмбебап деп айтсын деуге ұмтылған жарысушылар күйінде жүрді.

Ал бұдан шығатын жол бұрыннан-ақ белгілі болатын. Балаларға арналған әңгімедегі "Хоттабыч шалдың" әрбір футболшыға бірбір доп сыйлағаны тәрізді әрбір тілдің өз қолданылу аясы болуы керек еді.

Осындай "доптардың" алғашқысы қазіргі программалау ісінің әжесі – Грейс Мюррей Хоппер атымен тығыз байланыста болды.

Жеті жасында-ақ Грейс суперпрограммалаушы қасиеттерінен белгі берді: ол үйіндегі барлық оятқыш сағаттардың қалай істелгенін білмек болып бұзып, артынан жинай алмай қойыпты.

Екінші дүниежүзілік соғыс қызып тұрған кезде математика докторы Грейс Хоппер АҚШ-тың соғыс-теңіз флотына кіріп, 1944 жылғы маусымда офицер атағын алады. Оның әскерде адмирал шеніне дейін жеткенін айта отырып, программалау саласындағы табыстарына тоқталайық.

Хоппер сол кездердегі "Марк-1" машинасына арнап программа жазумен шұғылданған. Ол сегіздік санау жүйесінде жұмыс істеген.

Бірде өзінің банктегі есебін анықтағысы келіп (математика докторы үшін қиын мәселе), Хоппер өзінің банкте істейтін бауырынан көмек сұрайды. Банкир інісі Грейстің жазбаларын қарап отырып, оның барлық есептеулерді сегіздік жүйеде орындағанын көріп таң қалады. Ал банк болса, бұрынғыша ондық жүйені пайдаланатын еді.

Осы кездерден бастап, Хоппер машинамен адам тіліне жақын тілмен қатынас құруға болатын мүмкіндіктерді зерттеумен айналыса бастайды. Оның еңбегінің нәтижесі ретінде А-1, А-2, А-3, В-0, т.б. трансляторларды келтіруге болады. Бірақ осы трансляторлардың барлығында да қолданылған тіл адам тілінен әлдеқайда алшақ болатын.

Грейс Хоппердің мақсаты ағылшын тілінде жасалған программалау мүмкіндігін табу болатын.

Банк есептерімен болған жағдай оның программалаудағы жұмысын физиктерге арнауға немесе әмбебап программалау тілдерін жасауға емес, қарапайым экономикалық есептерді шығаруды жеңілдетуге арнауға бет бұрғызды.

Осы жобаға байланысты болған жайттардың барлық бүгешігесіне тоқтамай-ақ, тек Хоппердің соғыс-теңіз флотымен байланысы арқасында оның АҚШ әскери министрлігін қолданбалы іс жүйесінде пайдаланылатын тілдермен айналысуға көндіргенін айту жеткілікті. Ол кезде бұл министрліктің қол астында көбінесе экономикалық есептеулермен айналысатын мыңнан аса әртүрлі компьютерлер бар болатын.

Осының нәтижесінде 1959 жылы жаңа тіл жасау комитеті құрылды. 1960 жылдың сәуірінде сол комитет тілдің сипаттамаларын құрастырып, жариялаған болатын. Ал сол жылдың аяғында бірнеше фирма бұл тілдің трансляторларын да жасап үлгерді.

COBOL *(COmmon Business Oriented Language* — бизнеске арналған әмбебап тіл) тілі бірден Американың іскерлері ортасынан өз бағасын алды. Әрине, программалау ісінің мықтылары оны көп сөзді және өте көлемді деп санайды, бірақ онда ағылшын тілінің қарапайым сөздері қолданылатындықтан, программа коды кез келген мамандар мен менеджерлерге түсінікті деңгейде болды.

Бүгіннің өзінде, шыққанына жарты ғасырдан асса да, ол әлі қолданыс табуда. Бұл тілде жазылған программалардың жалпы бағасы 50 млрд доллар шамасында деп саналады. Экономикалық есептер шығарудағы оның тиімділігі әлі де жоғары деңгейде. Кобол тілінің негізінде қазіргі кездегі мәліметтер базасымен жұмыс істеуге арналған Clarion тілі жасалып шықты.

Егер алғашқы программалау мамандары тіл және оған транслятор жасау жұмыстарын бөле жарып қарамайтын болса, кейіннен теоретиктердің күш салуы арқасында бұл екі жұмыс бір-бірінен ажыратылды да, теоретиктерге де арнайы жұмыс табылды. Тілді біреулер жасап, трансляторларды басқалар жазатын болды.

Дегенмен компьютер тұтынушылары үшін программалардың бағалы қасиеті – олардың әмбебаптығы сақталғаны дұрыс болар еді. Мысалы, бір типтегі компьютерлерге арналып Коболда жазылған программалар басқа компьютерлерде де жақсы жұмыс істеуі тиіс.

Осының бәрі тілдердің сипаттамаларын стандарттау қажеттілігін туғызды, өйткені кейіннен оларға арнап басқа фирмалар трансляторлар жазуы керек болатын.

Бұрынғы бір бағыт орнына енді үш бағытта жұмыс істеуге тура келді:

1. Тілді жасау.
2. Тілдің стандартын анықтау.
3. Программалау тіліне транслятор жазу.

Оның үстіне ақпарат өңдеуге арналған адамзат қызметінің әр саласының өз программалау тілі болғаны дұрыс екенін өмір

дәлелдеді (міне, әр ойыншыға бір-бір доп деген осы шығар), мысалы:

- қолданбалы іскерлік аймақтарға арналған тіл (COBOL);
- ғылыми-техникалық есептерге арналған тіл (FORTRAN);
- кестелерді өңдеуге арналған тіл (APL – *A Programming Language* – программалау тілі – жай ғана атау);
- металл өңдеуге арналған қондырғыларға (станоктарға) арналған программалау тілі (APT – *Automatically Programmed Tools* – автоматты түрде программаланатын аспаптар);
- мамандардың айтуы бойынша, ми жұмысын модельдейтін және жасанды зерде (интеллект) жасауға болатын тіл (IPL – *Information Processing Languages* – ақпарат өңдеу тілдері);
- әмбебаптықтан да аздап үміті бар, нақты уақыт режимінде жұмыс істейтін объектілерді басқару тілі (ADA);
- жедел жадтан өте аз орын алып, барынша жылдам істейтін программалар алуға мүмкіндік беретін жүйелік программалаушыларға арналған "орта" деңгейдегі тіл (C/C++);
- нақты уақыт режимінде жұмыс істейтін "сындарлы" (критикалық) есептерге және борттағы компьютерлерге арналған тіл (FORTH);
- программалауды үйренуге арналған тіл (PASCAL);
- студенттерге арналған программалау тілі (LOGO);
- программалауды үйренуге қабілеті болмаса да, өте программалағысы келетіндерге арналған тіл (BASIC), тағы да сол сияқты жалғасып кете береді.

АҚШ әскери министрлігінде өткен ғасырдың 70 жылдары 450-дей жоғары деңгейдегі тілдер мен солардың диалектілері қолданыста болғанын айту жеткілікті шығар. Осы жайт әмбебаптықтан үміткер ADA тілінің шығуына себепкер болған.

Компьютерлердің қуаты артқанмен, олардың бағалары тұрақты түрде төмен түсіп жатқан заманда, программалаудың мықты теоретигі Эдсгер Дийкстра айтқандай, трансляторлар көптеген "ысқырықтар мен сылдырмақтармен" толықтырылып, жаңа мүмкіндіктерге ие болып жатыр. Сол сылдырмақтар мен ысқырықтар программалаушылардың жап-жақсы өмірін одан да көріктендіре түспек. Ол қандай болмақ – бұл жағынан пікір айтушылар аспан мен жердей алшақтықпен айтысып келе жатыр.

Программалау тілдері туралы ой өрбіткенде, құрылымдық программалау идеологиясы жайлы айтпауға болмайды, бұл идеологияны алғаш ұсынған да сол Дийкстра болатын. Осы құрылымдық программалау ұғымы оқулығымыздың негізінде жатыр. Ал Паскаль тілінің конструкциясы Дийкстра ұсынған, қазіргі кездегі кең қолданыста жүрген барлық процедураға бағытталған құрылымдық тілдердің ядросы болып саналады. Жақсы ұйымдастырылған құрылымдық тілдердің басты белгісі – онда GOTO операторының, яғни шартсыз басқаруды беру операторының болмауы немесе оның сирек колданылуы болып есептеледі.

Дийкстра программаларда нақты математикалық логикалық құрылымның жоқ екеніне көңіл бөліп, зерделі мамандардың жиі қайталайтын пікіріне көбірек назар аудару қажеттілігін басып айтқан болатын. Оның идеяларын жүзеге асыра отырып, IBM фирмасы "Нью-Йорк таймс" газетінің мәліметтер банкін жасау кезінде жұмыс бағасының әжептеуір арзандағанын және оны орындау мерзімінің де біраз қысқарғанын анықтады.

Бір қарағанға өте қарапайым көрінетін Дийкстраның осы идеясының өзі көптеген елдерде миллиардтаған долларды үнемдеуге көмектесті. Қазіргі кезде жоғарыдан төмен қарай жүргізілетін құрылымдық бағыттағы жобалауды жүзеге асыра алмайтын бірде-бір программалау тілі қолданысқа кіріп, өмірден өз орнын таба алмайды.

Сонымен, жүйелік программалау маманы Харлан Миллс айтқан: "Ғарыш кемелерінің ұшуының өзі де құрылымдық программалауды пайдаланбай жүзеге асқан деп ойламаймын", – деген сөзінде шындық бар.

Pascal тілі де құрылымдық программалау тәсілдерін үйретететін оқыту құралы ғана болуға тиіс еді. Бірақ бірсыпыра себептерге байланысты ол бұл аядан шығып, әлемдегі ең негізгі тілдердің біріне айналды.

Құрылымдық программалау тілдері табиғи тілдің сөздерін және грамматикасын кеңінен пайдалану жолымен дамыды, бірақ оның негізінің түп тамыры болатын программалаудың процедуралық стиліне тиіскен жоқ. Осы *процедуралық стиль* сөзі компьютердің командаларды атқарушы рөлінде болатынын

және оған нәтиже алуға қажетті барлық әрекеттер (процедуралар) тізбегін толық әрі дәлме-дәл сипаттап беру керек екендігін білдіреді. Оған қоса, компьютердің фон-Нейман архитектурасының өзі де оларды қуатты, бірақ ақылсыздау робот ретінде қарастыруды ұсынады.

1972 жылы шотланд ғалымы Р.Ковальский осындай мүмкіндігі бар тіл ретінде тұжырымдамалар және олармен орындалатын операцияларды өңдей алатын математикалық логика тілін жетілдіріп пайдалануды ұсынды. Бұл идеяны Франция, Португалия, Ресей, т.б. елдердің ғалымдары жалпы түрде қарастырып, практикалық түрде жүзеге асырмақшы болды.

Осындай мақсатта жасалған программалау тілі PROLOG (французша – *PROgramming et LOGicque)* деп аталды. Оны жасауға және дамытуға жаңа ғылыми бағыттың нақты түрдегі қорытындысы (прологы) ретінде пайда болған **логикалық программалау** ғылымы үлес қосып келеді.

Логикалық программалау компьютер есепті адам сияқты жолмен шығаруы керек деген ойды негізге алады. Логикалық программалаумен қатар пайда болып, функцияның математикалық ұғымдарына негізделген екінші бір сала **функционалдық программалау** деп аталады.

Функционалдық программалаудың алғашқы тілі 1960 жылдардың басында шыққан LISP тілі болып табылады. LISP сөзі – ағылшынның *LISt Processing* деген сөз тіркесінің қысқартылған нұсқасы. Бұл да кең таралған тілдердің бірі. Ол жасанды зерде мәселелерін шешу жолында, әсіресе АҚШ-та көп қолданылады.

Қазіргі кезде компьютерлердің көбісіне арналған LISP тілінің бірнеше нұсқалары бар. Бірақ бұл тіл функционалдық программалаудың жалғыз ғана құралы емес. Оған ұқсас тілдерге Miranda, Hope, FP тілдері жатады.

Өткен ғасырдың 80-жылдары программалауда тағы да бір жаңа ағым пайда болды. Енді компьютерге бастапқы мәліметтер мен нәтиже арасында байланыс орнату ісі жүктелді. Біз компьютерге ол жұмыс істеуге тиіс объектілер сипаттамасын енгіземіз, ал компьютердің өзі сол объектілердің қалай байланысқанын анықтауы керек болды.

Объектілер қасиеттерін сипаттаудың осы түрі **объектіге бағытталған программалау** деп аталды. Объектіге бағытталған программалау тілдеріне C++ және Smalltalk тілдері жатады.

Бұлардың бәрі бүгінгі күн тұрғысынан туындаған көзқарастар болатын. Ал тарихи тұрғыдан объектіге бағытталған программалау компьютердің мүмкіндіктерінің жетілдірілуі мен ол шығара алатын есептер аймағының кеңуі арқасында пайда болды.

Алғашқы программалау тілдері тек бір санды ғана өңдей алатын еді. Ал сан жиымдары цикл арқылы өңделетін.

Кейіннен машина бір оператор арқылы жүздеген және мыңдаған сандарды қоса алатын, жиым ішінен максимумын табатын, оларды реттейтін, т.с.с. мүмкіндіктерге ие болды Сандар жиымы, программалаушы тұрғысынан қарағанда, әртүрлі амалдар орындауға болатын бір ғана объект түрінде қарастырылатын кезге жеттік.

Бірақ нақты өмірде біз тек сандық ақпаратпен ғана жұмыс істемейміз. Мысалы, студентті тек оның жасы емес, аты, фамилиясы, тобы, үй адресі, т.б. сипаттайды. Әртүрлі типтегі мәліметтерді біріктіріп, **мәліметтер құрылымын** алуға мүмкіндік (қажеттілік) туды. Осындай әрбір құрылымды *біртұтас күйде* өңдеуге болатын болды.

Осындай құрылымдар олармен орындауға болатын әрекеттермен бірге сипаттауға арналған жасанды тіл объектілері мен олардың қасиеттері атауларынан тұрады. Мұндай қасиеттер компьютерге осы объектілерді қалай түрлендіруге болатындығы жайлы сигнал береді. Бұл тілде программалау объектілерді сипаттау және бастапқы мәліметтер мен нәтижелерді осындай объектілерге құрылымдау ісіне келіп тіреледі.

Құрылымдар мен объектілердің күрделілігін, практикалық тұрғыдан алғанда, шектеусіз арттыру ісі (мысалы, құрылымдар объектісі болатын жаңа объект жасау ісі) ақпараттың өте үлкен көлемімен жұмыс істеуге мүмкіндік береді. Сол себепті компьютерлер де күннен-күнге күрделіленіп келе жатқан есептерді шығаруға пайдаланылып келеді.

Бақылау сұрақтары

1. *Программалауды оңайлататын жаңа тіл ойлап шығаруды алғаш рет кім ұсынды?*
2. *Фортран тілі қай жылы шықты және оның ерекшелігі қандай болды? Алгол, PL/1, Кобол тілдері ше?*
3. *Программалау тілдерін жасауды стандарттау қажеттілігі неге байланысты туындады?*
4. *Грейс Хоппердің программалау саласындағы еңбектері.*
5. *Программалау тілдерінің саны неге көп болып кетті?*
6. *Құрылымдық программалау тілдерінің ерекшелігі неде?*
7. *Программалау теоретигі Эдсгер Дийкстра қандай бағытты ұстану қажеттілігі туралы айтты?*
8. *Объектіге бағытталған программалау тілдері қандай болады?*

ҚОЛДАНЫЛҒАН ТЕРМИНДЕРДІҢ ТҮСІНДІРМЕЛІК СӨЗДІГІ

ASCII-кодтар (American Standart Code for Information Interchange) – символдар кодтарының негізі болып табылатын американдық кодтар стандарты

Айнымалы – программа орындалуы барысында өз мәнін өзгерте алатын объект

Айнымалылар – идентификаторлармен белгіленіп, программаның орындалу барысында әртүрлі мәндерді қабылдай алатын шамалар

Айнымалының қасиеттері:

1) кез келген сәтте айнымалының белгілі бір мәні болады немесе ол анықталмаған болып есептеледі;
2) айнымалыға соңғы берілген мән оның алдыңғы мәнін өшіріп жібереді.
3) айнымалыны таңдау (оқу) және оны пайдалану айнымалының мәнін өзгертпейді

Айнымалылардың әрекет ету аймағы – программаның ағымдағы әрекеттеріне қандай мәліметтердің қатынасуға болатынын анықтау аймағы

Айыру белгілері – бос орын, барлық басқару символдары, ENTER (келесі жолға көшіру) пернесін басу белгісі және түсініктемелер

Алгоритм – алға қойған мақсатқа жету немесе берілген есепті шешу бағытында атқарушыға біртіндеп қандай әрекеттер жасау қажеттігін әрі түсінікті әрі дәл етіп көрсететін нұсқаулар жиыны

Алгоритм – берілген есептің шығару жолын реттелген амалдар тізбегі түріне келтіру

Алгоритм – берілген мәндерді пайдаланып қажетті нәтижеге жетуді жүзеге асыратын әрекеттердің орындалу ережесі

Алгоритм атқарушы – көрсетілген іс-әрекеттер тізбегін бұлжытпай орындай отырып, керекті нәтиже алуды жүзеге асыратын машина, құрылғы немесе адам

Алгоритмді өрнектеу түрлері:

1) табиғи тіл арқылы жазу;
2) белгілі бір түйінді сөздер – терминдер (псевдокодтар – жалған кодтар) арқылы қысқаша тізбекті түрде жазу (қарапайым алгоритмдік тілде);
3) график жолымен (блок-схема арқылы) жазу;
4) программалау тілдерінде жазу

Алгоритмдік тіл немесе программалау тілі – жазу ережелері қарапайым болып келетін жасанды тіл

Алгоритмнің қасиеттері:

1) алгоритмнің айқын, дәл өрнектелу қасиеті;
2) алгоритмнің үздіктілік қасиеті;
3) алгоритмнің нәтижелілік қасиеті;
4) Алгоритмнің жалпылық немесе ортақтық қасиеті

Алгоритмнің формалды орындалуы – орындаушы алгоритм командаларын орындай отырып, өзі атқарып отырған істің мән-жайына көңіл аудармай, әрекеттерді ретімен атқара отырып, белгілі бір нәтиже алуы

Атқарушының командалар жүйесі – алгоритм атқарушы орындай алатын командалар жиыны

Ауыстырғыш оператор (switch) – берілген өрнек мәні бойынша программада кездесетін бірнеше нұсқаның бірін таңдап алуға мүмкіндік беретін шартты оператор

Басқару операторлары – шартсыз және шартты көшу операторлары, цикл (қайталау) ұйымдастыру операторлары

Басқару тізбектері – мәліметтер енгізу мен шығаруда қолданылатын арнайы символдар тіркесі (C/C++ тілінде)

Бейнемонитор (немесе дисплей) – электрондық сәулелік түтікшеден немесе сұйық кристалды тақтадан тұратын мәтін және графикалық бейнелер шығарылатын компьютер құрылғысы

Бос оператор – ешқандай да әрекеттің орындалмайтынын көрсететін оператор

Вирт синтаксистік диаграммасы – тіл ережелерін графикалық түрде бейнелейтін синтаксистік диаграмма (Н. Вирт көп қолданған)

Драйверлер – дербес компьютерлердің техникалық құрылғыларын басқаратын арнайы программа

Енгізу/шығару операторы – адам мен компьютер арасында мәлімет алмасу мақсатында алдын ала анықталған *printf(), scanf()* функцияларын (C тілінде) қолданып, берілген мәндерді пернетақтадан енгізіп, алынған нәтижені дисплей экранына шығару операторлары

Жиым (массив) – бір атаумен белгіленіп, реттеліп біріктірілген бір типтегі мәліметтер жиыны

Идентификатор – міндетті түрде әріптен басталатын сандар мен әріптер (астын сызу белгісінің де) тізбегі

Көшу (немесе шартсыз көшу) *операторы* – операторлардың рет-ретімен орналасуын бұзып, келесі атқарылуды белгісі бар жолға беру операторы

Қабатталған циклдер – бір цикл тұлғасы ретінде (цикл ішінде) басқа циклдік құрылым тұратын күрделі циклдерді ұйымдастыру тәсілі

Қолданушы функциясы – белгілі бір ат қойылып, жеке программа түрінде бөлек жазылған, қажет кезінде оны қайталап пайдаланып отыруға болатын негізгі программаның арнайы бөлігі. Осындай алдына ала ат қойылған командалар тобын программаның кез келген жерінен оның атын көрсету арқылы шақырып орындауға болады

Құрама оператор – бір-бірінен нүктелі үтір арқылы бөлінген бірнеше операторларды { және } таңбаларымен (C тілінде) шектей отырып, оларды бір оператор тәрізді орындалатын етіп біріктіру жолы

Құрылымдар – С тіліндегі өзара логикалық байланысқан әртүрлі типті мәліметтерді байланыстыру

Құрылымдық операторлар – басқа операторларды белгілі бір ережелер бойынша біріктіру жолымен құрастырылған операторлар тобы. Олар үш топқа – құрама, шартты және қайталау операторларына жіктеледі

Логикалық амалдар:

AND - және (логикалық көбейту);
OR - немесе (логикалық қосу);
NOT - емес (терістеу немесе жоққа шығару);
XOR - арифметикалық *немесе* амалы;

Машинаға бағытталған тілдер компьютердің ерекшеліктерін есепке ала отырып әріптерді де пайдаланатын тілдер

Мәлімет енгізу – бастапқы мәндерді пернетақтадан, дискілерден немесе енгізу-шығару порттарынан оқу арқылы жүзеге асыру

Мәліметтер – белгілі бір процесс көмегімен тасымалдап, өңдеуге болатын, формальды түрде бейнеленген фактілер мен идеялар

Мәліметтер – сан мәндерін, мәтін ретіндегі сөз тіркестерін де мән ретінде қабылдай алатын тұрақтылар (константалар), айнымалылар, тағы осылар тәрізді құрылымдар немесе солардың адрестері

Меншіктеу операторы – жазылған өрнектердің мәнін есептеп, оны айнымалыға (функцияға) беру операторы

Нақты сандар – аралас сандардың бүтіні мен бөлшегін нүкте (немесе Е) арқылы бөліп жазу тәсілі

Нәтиже алу (шығару) – аралық немесе қорытынды мәліметтерді экранға, дискіге немесе енгізу-шығару порттарына жазу

Нұсқауыш – С тіліндегі мәліметтердің адресін сақтайтын айнымалы, яғни адрестің символдық түрде кескінделуі

Оператор – операциялар мен мәндерді көрсететін немесе солардың элементтерінің қай жерде орналасқанын білдіретін символдар жиыны

Оператор – программалау тілінің белгілі бір іс-әрекетті орындай алатын тиянақты мағынасы бар ең қарапайым сөйлемі

Операциялар немесе амал-әрекеттер – берілген, есептелген мәндерді меншіктеу, соларды өңдеу, салыстыру істерін орындау

Өрнек – арифметикалық немесе логикалық амалдар таңбасымен біріктірілген айнымалылар, атаулар, функциялар, жиымдар тағы да басқа мағынасы бар сөздер тізбегі

Параметрлі цикл операторы (for) – параметр өзінің алғашқы мәнінен соңғы мәніне дейін берілген қадаммен өзгеру барысында циклге кіретін бір немесе бірнеше әрекеттерді қайталап орындау ісін атқаратын оператор

Паскаль тілі – 1968-1971 жылдары Швейцарияда профессор Никлаус Вирт жасап шығарған оқып үйренуге қолайлы программалау тілі

Пиксель – экранның адрестелетін ең кіші элементі

Проблемаға бағытталған тілдер шығарылатын есептердің ерекшеліктерін еске ала отырып, есептің математикада жазылу тіліне жақындастырылған тілдер

Программа – алгоритмді машинаға түсінікті нұсқаулар тізімі ретінде жазу

Программалау тілінің түйінді сөздері – бір-бірімен айыру белгілерімен бөлініп, программада алдын ала анықталған белгілі бір мағынасы бар сөз тіркестері

Программаны орындау ортасы – программалау тілдерінде программа кодын құрастырып, оны орындауға мүмкіндік беретін біріктірілген орта

Процедура – программадағы бірнеше операторларды олардың бастапқы параметрлерін өзгерте отырып, бірнеше рет қайталап орындауды жүзеге асыратын қосалқы программалар түрі

Рекурсия – процедура немесе функция операторларының орындалуы барысында оның өзін-өзі, яғни осы қосалқы программаның өзін өзінің ішінде пайдалану тәсілі

С – программалық жабдықтамалар жасауға және әртүрлі есептер шығаруға арналған құрылымдық программалау тілі

Стандартты функциялар – жиі кездесетін математикалық және басқа да функциялардың программалау тілінің ішкі объектісі ретінде берілуі

Сызықтық құрылым – бірінен кейін бірі орындалып тізбектеле орналасқан бірнеше әрекеттер (операторлар) жиыны

Таңдау операторы (switch) – бірнеше тармақты операторлардың ішінен берілген өрнектің мәніне байланысты біреуін таңдауды жүзеге асыратын оператор

Тармақты құрылым – шартқа байланысты екі оператордың бірінің орындалуы

Тип (мәліметтердің немесе *шамалардың типі)* – программалау оюъектілерінің қабылдай алатын мәндерінің және солармен орындауға болатын амалдардың жиыны, яғни шамалардың қабылдайтын мәндеріне берілетін сипаттама

Транслятор – программалау тілін машина тіліне аударатын программалар тобы

Тұрақты **(**немесе *константа***)** – программаның орындалу барысында мәндері өзгеріссіз қалатын шамалар

Тілдің әліпбиі – программаның элементтерін құруда қолдануға болатын символдар жиыны

Тілдің қарапайым объектілері – сан, идентификатор, константа, айнымалы, функция, өрнек ұғымдары

Тіркестік (жолдық) константа – С тіліндегі қос тырнақшаға алынған символдар жиыны

Тіркестік айнымалы – мән ретінде апострофтармен қоршалған сөз тіркестерін қабылдайтын айнымалы

Тіркестік өрнек – амал белгілері, тіркестік тұрақтылар, айнымалылар және функция атауларынан құралған, ұзындығы 256 символдан артпайтын сөз тіркестерінен құралған тізбек

Файл – сыртқы есте сақтау құрылғыларында (магниттік дискілерде) орналастырылған және мәлімет өңдеу, тасымалдау кездерінде біртұтас күйде қарастырылатын мәліметтер жиыны

Файл (file) – магниттік дискілерде тізбектеле орналасқан жеке элементтерден тұратын белгілі бір ат қойылған ақпараттар тобы.

Функция – жұмыс нәтижесінде бір ғана мәнді анықтайтын қосалқы программалар (операторлар тізбегі) түрі

Цикл – белгілі бір шарт орындалса (кейде орындалмаса), көрсетілген командалардың бірнеше рет қайталанып атқарылуы немесе шарт көрсетілмейақ алдын ала олардың неше рет қайталанатынын бүтін санмен беру

Цикл – операторлар бөлігінің бірнеше рет қайталана орындалуы

Шартты оператор (if) – тармақталу процестері бар алгоритмдерді ұйымдастыратын, белгілі бір шарттың орындалуы немесе орындалмауына байланысты екі мүмкіндіктің бірін таңдау операторы

Шартты операция (? :) – берілген шартқа байланысты әртүрлі мән қабылдай алатын *шартты өрнектер* құратын үшорынды операция (C/C++ тілінде).

Шартын алдын ала тексеретін цикл операторы (while) – орындалар алдында қайталану шартының мәні есептеліп, соның ақиқат/жалған болуына байланысты циклді ұйымдастыру операторы.

Шартын соңынан тексеретін цикл операторы (do..while) – цикл тұлғасы орындалғаннан кейін қайталану шартының мәні есептеліп, соның ақиқат/жалған болуына байланысты циклді ұйымдастыру операторы.

ҚОЛДАНЫЛҒАН АТАУЛАРДЫҢ ҚЫСҚАША ОРЫСША-ҚАЗАҚША СӨЗДІГІ

А

Аргумент заданный по умолчанию	Келісім бойынша берілген аргумент
Арифметические операции	Арифметикалық амалдар
Арифметика указателей	Нұсқауыштар арифметикасы

Б

Библиотека	Кітапхана

В

Ввод	Енгізу
Ввод данных	Мәліметтерді енгізу
Ввод текста	Мәтінді енгізу
Вывод	Шығару
Вывод данных	Мәліметтерді шығару
Вывод текста	Мәтінді шығару
Выражение сравнения	Салыстыру өрнегі

Д

Данные	Мәліметтер
Двоичный поиск	Екілік түрде іздеу
Двоичное число	Екілік сан
Динамическая память	Динамикалық жады
Динамическое распределение памяти	Жадты динамикалық түрде бөлу
Документация	Құжаттама

З

Знаки	Таңбалар

И

Инициализация	Бастапқы мән беру, инициалдау
Интегрированная среда	Біріктірілген (интегралданған) орта
Именование переменной	Айнымалы атауы

К

Классы памяти	Жады кластары
Ключ	Кілт (кілтипан), түйін
Ключевое слово	Түйінді (өзекті) сөз
Ключевые понятия	Өзекті ұғымдар (түсініктер)
Конкатенация строк	Сөз тіркестерін біріктіру
Константы с плавающей точкой	Жылжымалы нүктелі тұрақтылар
Константы символьные	Символдық тұрақтылар (константалар)
Константы целочисленные	Бүтін тұрақтылар

Л

Логика	Логика, қисын
Логические выражения	Логикалық өрнектер

М

Массив	Жиым
Массив двумерный	Екіөлшемді жиым
Массив динамический	Динамикалық жиым
Метод	Тәсіл, әдіс

Н

Наследование	Мұралау

О

Оборудование	Жабдық
Область имен	Атаулар аймағы
Объединение	Біріктіру
Объект	Объект, нысан
Объектно-ориентированное программирование	Объектіге бағытталған программалау
Операция декремента (--)	Декремент 1-ге кеміту) операциясы (--)
Операция инкремента (++)	Инкремент (1-ге арттыру) операциясы (++)
Операция приведения типов	Типтерді келтіру операциясы
Оператор присваивания	Меншіктеу(тағайындау) операторы
Оператор составной	Құрама оператор
Оператор цикла	Цикл операторы
Оператор цикла с предусловием	Алғы шартты цикл операторы
Оператор цикла с постусловием	Соңғы шартты цикл операторы
Опертор выбора	Таңдау операторы
Оператор условный	Шартты оператор
Освобождение памяти	Жадты босату
Основание	Негіз
Основание системы счисления	Санау жүйесінің негізі
Основной тип	Негізгі тип
Отношение	Қатынас

П

Память автоматическая	Автоматты жады
Память статическая	Статикалық жады
распределение памяти	жадты бөлу (үлестіру)
Переменная автоматическая	Автоматты айнымалы
Переменная внешняя	Сыртқы айнымалы
Переменная временная	Уақытша айнымалы
Переменная глобальная	Ауқымды (жалпылама) айнымалы
Переменная локальная	Жергілікті айнымалы
Переменная ссылочная	Сілтеме айнымалысы
Перестановки	Орын ауыстыру (алмастыру)
Перечисления	Тізбелер

Последовательность	Тізбек
Преобразование	Түрлендіру
Преобразование типов данных	Мәліметтер типтерін түрлендіру
Приведение типов	Типтерді келтіру
Приоритет операции	Амалдар үстемділігі (приоритеті)
Присваивание	Меншіктеу, тағайындау
Программа	Программа (бағдарлама)
Программа вложенная	Қабаттасқан (кіріктірілген) программа
Программа линейная	Сызықтық программа
Программа разветвленная	Тармақталған программа
Программа циклическая	Циклдік (қайталау) программа
Программирование	Программалау
Программное обеспечение	Программалық жабдықтама
Программная систем	Программалық жүйе
Р	
Рекурсия	Рекурсия, функцияның өзін-өзі шақыруы
С	
Случайные	Кездейсоқ
Символ новой строки (\n)	Жаңа жолға көшу символы
Символ константы	Тұрақты символы
Символические	Символдық
Сстема счисления	Санау жүйесі
Сортировка	Сұрыптау, іріктеу, реттеу
Составной тип	Құрама тип
Сравнение строк	Тіркестерді салыстыру
Ссылка	Сілтеме, сілтеу
Строка	Тіркес, сөз тіркесі
Строковая переменная	Тіркестік айнымалы
Структура	Құрылым
Т	
Таблица	Кесте
Тип данных	Мәліметтер типтері
Тип данных без знака	Таңбасыз мәліметтер типі
Тип данных основной	Негізгі мәліметтер типі
Тип данных с плавающей точкой	Жылжымалы нүктелі мәліметтер типі
Тип данных символьный	Символдық мәліметтер типі
Тип данных целочисленный	Бүтін мәліметтер типі

У

Указатель	Нұсқауыш
Указатель на функцию	Функцияға нұсқауыш
Управляющие последовательности языка С	С тілінің басқару тізбектері
Установка	Орнату (тағайындау)
Установочные параметры	Тағайындау параметрлері

Ф

Форматирование	Пішімдеу, форматтау
Функция определенная пользователем	Тұтынушы анықтаған функция
Функция со многими аргументами	Көп аргументті функция
Функция сравнения	Салыстыру функциясы

Ц

Целочисленная константа	Бүтін мәнді тұрақты
Цикл	Цикл, қайталау
Цикл вложенный	Қабаттасқан (кіріктірілген) цикл
Цикл с параметром	Параметрлі цикл

Ч

Число с плавающей точкой	Жылжымалы нүктелі сан
Число с фиксированной точкой	Бекітілген (тұрақты) нүктелі сан
Число шестнадцатеричное	Он алтылық сан
Число восьмеричное	Сегіздік сан

Ш

Шаг цикла	Цикл қадамы, адымы

Я

Язык программирования	Программалау тілі

С ТІЛІНДЕ ПРОГРАММАЛАУДАН ТЕСТ СҰРАҚТАРЫ

1. Идентификатор дегеніміз не?

1) программадағы объектінің аты

2) динамикалық жады

3) жиыннан тұратын массив

4) программаның берілу жолы

5) компиляторға арналған сөздер

2. Символдық типтегі шамаларға компилятор жадында қанша байт орын бөлінеді?

1) 1

2) 2

3) 10

4) 8

5) 27

3. Жиым (массив) дегеніміз не?

1) ұяшықтарда орналасқан мәліметтер

2) бір атаумен аталған бір типті мәндер тізбегі

3) өлшемді анықтайтын тип

4) динамикалық жадыны пайдаланатын көрсеткіш

5) сілтемені анықтайтын типтер жиынтығы

4. Қандай да бір әрекеттер тізбегін орындайтын операциялар мен сипаттамалардың айқындалған тізбегін ... деп атайды.

1) процедура

2) рекурсия

3) функция

4) дұрыс жауап жоқ

5) 1,2

5. Функцияның сипаттамасы басындағы параметрлер қалай аталады?

1) формальды

2) фактілі

3) аргументті

4) 2,3 жауап дұрыс

5) дұрыс жауап жоқ

6. С тілінде кез келген программаның орындалуы қай функциядан басталады?

1) random

2) main()

3) randomize()

4) clrscr()

5) Әр программада әртүрлі

7. Өзін-өзі шақыратын функцияны не деп атаймыз?

1) процедура

2) функция

3) рекурсия

4) 1,2

5) дұрыс жауап жоқ

8. Кітапханалық функциялар қайда орналасқан?

1) кітапханалық файлдарда

2) С ішінде

3) компьютерде

4) дұрыс жауап жоқ

5) интернетте

9. С әліпбиі:

1) араб цифрі: 0-9

2) арнайы белгілер

3) түйінді сөздер

4) операциялар таңбалары

5) барлығы дұрыс

10. Латын әріптерінен, цифрлардан тұратын тек әріптерден басталуы тиіс таңбалар тізбегін ... деп атайды.

1) тұрақты

2) айнымалы

3) идентификатор

4) дұрысы жоқ

5) функция

11. a[3] = {1,3,5} жиымы берілсін делік. Осы жиымды экранға шығаратын программаны көрсетіңіз...

1) int i, a[3] = {1,3,5};
for(i=0;i<3;i++)
printf(" %d",a[i]);

2) int a[3] = {1,3,5}, i;
for(i=0; i<3;i++)
printf(" %f",a[i]);

3) int a[i] = {1,3,5}, i;
for(i=0;i<3,i++)
printf (" %k",a[i]);

4) int a[3], i;
for(i=0;i<3;i++)
printf(" %i",a[i+1]);

5) int a[3], i;

```
for(i=0;i<3;i++)
printf(" %c",a[i]);
```

12. С тілінде тұрақтыларды сипаттау үшін қандай түйінді сөз қолданылады?

1) int
2) var
3) const
4) procedure
5) дұрысы жоқ

13. return (өрнек) операциясының қызметі:

1) функцияны шақырған операторға өрнектің мәнін қайтарады
2) С тілінде ондай оператор қолданылмайды
3) функция мән қайтармаған кезде қолданылады
4) программаға әсемдік береді
5) қызметі анықталмаған, заң бойынша программа соңында тұруы керек

14. rand() функциясының тақырыптық файлда орналасқан прототипі қандай?

1) <math>
2) <string>
3) <stdlib>
4) <cmath>
5) rand

15. Функцияның ішінде ғана белгілі, соның ішінде анықталатын айнымалы қалай аталады

1. ауқымды (глобальді) айнымалы
2. жергілікті айнымалы
3. белгісіз айнымалы
4. белгілі айнымалы
5. рекурсивті айнымалы

16. Егер float пен double типтері араласса, нәтижесі ... болады;

1. float;
2. long;
3. **double**;
4. int;
5. short;

17. Кез келген блоктан немесе функциядан тыс хабарланған айнымалы...

1. ауқымды (глобальді) айнымалы
2. жергілікті айнымалы
3. программалық айнымалы
4. базалық айнымалы
5. унарлық айнымалы

18. ... компиляторға функция арқылы берілетін аргументтердің санын, типін, ретін анықтауға көмектеседі.

1. функцияның идентификаторы
2. функцияның прототипі
3. файлдың прототипі
4. жергілікті прототип
5. аумақтық идентификатор

19. Функцияның тақырыбында void түйінді сөзі не үшін қолданылады?

1. функция мән қайтаратынын көрсету үшін
2. функцияның тақырыбында void түйінді сөзі қолданылмайды
3. дұрыс жауабы жоқ
4. функцияның ешқандай мән қайтармайтынын көрсету үшін
5. функция мән сақтайтынын көрсету үшін

20. Қандай айнымалылар өзі сипатталған функциядан шыққаннан кейін де өз мәнін сақтайды?

1. static секілді хабарланған жергілікті айнымалылар
2. аумақты айнымалылар
3. extern секілді хабарланған жергілікті айнымалылар
4. static секілді хабарланған ауқымды айнымалылар
5. жергілікті айнымалылар

21. Блок дегеніміз не?

1. функцияның екінші аты
2. айнымалыларды сипаттауы бар құрама оператор
3. кездейсоқ оператор
4. қатені анықтайтын оператор
5. дұрыс жауап жоқ

22. Айнымалы дегеніміз не?

1. белгілі бір атауы бар жады аймағы, онда анықталған типтің мәліметтері сақталады
2. анықталмаған тип мәліметтерінің жадыда сақталуы
3. дұрыс жауабы жоқ
4. жады кластарының бірнеше бөліктерге бөлінуі
5. операндадан, операция белгісінен, жақшадан тұратын, анықталған типтің мағынасын білдіруі

23. Шартты операция форматын көрсетіңіз:

1. операнд_1? операнд_2 : операнд_3
2. операнд_1 := операнд_2
3. 1, 2
4. if (өрнек)оператор_1:[else оператор_2;]
5. if (тұрақты өрнек)1:[оператор тізбегі]

24. Жады кластарын беру үшін қандай спецификатор қолданылады?

1. auto
2. extern
3. static
4. register
5. **барлығы дұрыс**

25. С тілінде қайталау операторының қандай түрлері бар?

1. while (өрнек) оператор;
2. do ... while (өрнек)
3. дұрыс жауап жоқ
4. **1, 2, 5**
5. for(инициализациялау; өрнек; модификация) оператор;

26. Тармақталу операторы дегеніміз не?

1. **Алгоритмнің белгілі бір шарттың орындалуына немесе орындалмауына байланысты тармақталып, бірнеше жолдарға бөлінуі**
2. дұрыс жауап жоқ
3. бір ғана шарттың орындалуын тексеретін оператор
4. шарттардың орындалмауын тексеретін оператор
5. операторлардың тармақталуын тексеретін оператор

27. #include директивасын қолданған кезде бұрыштық ("<",">") жақшаның орнына тырнақшаларды қолдануға болады ма, болса оның мәні неде?

1. болмайды
2. болады, бірақ одан ештеңе өзгермейді
3. болады, бұл уақытта файлды іздеу әрекеті бастапқы файлы бар каталогта жүргізіледі, сосын стандартты каталогтан іздейді
4. **болады, онда файлды стандартты каталогтан іздейді**
5. болады, бұл жағдайда файлды стандартты каталогтан іздейді, сосын бастапқы файлы бар каталогты қарастырады

28. Бастапқы ".h-файлдың" құрамына не кіреді?

1. типтің, тұрақтылардың, кіргізілген функциялардың, шаблондардың, тізімдердің анықталуы
2. функцияның, мәліметтің, шаблонның, аттың сипатталуы
3. атау кеңістігі
4. процедуралардың директивалары
5. **көрсетілген жауаптың бәрі дұрыс**

29. #define директивасы не үшін қолданылады?

1. **тұрақтының мәнін беру үшін**
2. макростар үшін
3. шартты компиляторды басқаруға арналған символ үшін
4. символдық константа үшін
5. берілген жауаптың бәрі дұрыс

30. Құрамында функциялары немесе анықталған мәліметтері бар бастапқы (тақырыптық) файлдардың кеңейтілуі қандай және ол файлдар қалай аталады?

1. ".htp", ".htp-файлдары"
2. **".h", ".h-файлдары"**
3. ".c" ".c-файлдары"
4. ".f" ".f- файлдары"
5. ".exe", ".exe- файлдары"

31. Сөз тіркесінің (жолдың) ұзындығы қай функцияның көмегімен анықталады?

1. **strlen**
2. src
3. return
4. template
5. main()

32. 10 нақты саннан тұратын массивті сипаттау қай нұсқада дұрыс берілген?

1. int a[10];
2. float a(10);
3. **float a[10];**
4. float a[1..10];
5. int a[1..10];

33. Операндтардан, операция таңбаларынан, жақшалардан тұратын .... мәндерін есептеу үшін қолданылады. Көп нүктенің орнындағы сөзді табыңдар.

1. **өрнектер**
2. айнымалылар
3. тұрақтылар
4. логикалық типтер
5. жады класы

34. Айнымалы дегеніміз не?

1. программа орындалуы барысында әртүрлі мәндер қабылдайтын шама

2. жады класының бірнеше аймақтарға бөлінуі
3. программалар тізбегі
4. алгоритмнің белгілі бір шартының орындалуы
5. дұрыс жауабы жоқ

35. Таңдау операторы (нұсқауы) қай сөз арқылы беріледі?

1) **switch**
2) return
3) for
4) if
5) class

36. Егер long пен float типтері араласса, нәтижесі ... болады.

1) char;
2) short;
3) int ;
4) **float;**
5) double.

37. Бөлгендегі қалдық табу операциясы қалай беріледі?

1. %
2. /
3. *
4. <<
5. &

38. Блок ішінде анықталған айнымалы қалай аталады?

1. ауқымды
2. тұрақты
3. жергілікті
4. тұрақсыз
5. сілтеуіш

39. Блоктың ішінде сипатталған идентификатор қалай аталады?

1. глобальды көрініске ие
2. локальді идентификатор
3. оператордың белгісі
4. функцияның прототипінде көрсетілген параметр
5. көрсетілген типтің айнымалысы құрылғаннан бастап, ол жойылғанша өмір сүреді

40. *a* атты бүтін айнымалының дұрыс сипатталуы:

1. **int *a*;**
2. *a* int;
3. integer *a*;
4. float *a*;
5. char *a*.

41. Блоктан тыс анықталған айнымалы қалай аталады?

1. ауқымды (глобальды)
2. жергілікті
3. символды
4. тұрақты
5. жергіліксіз

42. Қай нұсқада sizeof жазылуы дұрыс?

1. sizeof тип
2. sizeof [тип]
3. sizeof (тип)
4. sizeof: тип
5. sizeof_ тип

43. Логикалық терістеу қай нұсқада дұрыс берілген?

1. -
2. ~
3. =
4. !
5. --

44. extern спецификаторы нені білдіреді?

1. айнымалы программа ішінде анықталғанын
2. айнымалы программадан тыс анықталғанын
3. айнымалы анықталмағанын
4. айнымалы автоматты түрде анықталғанын
5. дұрыс жауабы жоқ

45. 12 бүтін саннан тұратын *a* массивінің дұрыс сипатталуы:

1. float *a*[12];
2. int *a*[12];
3. float *a*[10];
4. char *a*[12];
5. int *a*(12);

46. Функция тұлғасы қандай таңбалармен қоршалып тұрады?

1. ()
2. / /
3. { }
4. * *
5. " "

47. 0...255 сандар диапазоны қай типке жатады?

1. double
2. unsigned char
3. bool
4. float
5. signed char

48. Goto нұсқауының дұрыс жазылуы:

1. Goto 65
2. Goto m65
3. Goto белгі
4. Goto ?65
5. Goto printf

49. Келесі операциялардың қайсысы тренарлы (үш операндты)?

1. !
2. &
3. <=
4. !=
5. ?:

50. С тілінде комментарийді қандай таңбалар ішінде жазады?

1. // //
2. (* *)
3. **/* */**
4. < >
5. { }

51. Программаның барлық объектілері үшін ортақ мәліметтерді сақтау үшін қандай класс қолданылады?

1. статикалық
2. тұрақты
3. рекурсивті
4. айнымалы
5. **ауқымды (глобальді)**

52. Егер short пен int типтері араласса, нәтижесі – ... болады

1. char;
2. short;
3. **int;**
4. float;
5. double.

53. Нұсқауыштарды сипаттау форматы қай нұсқада дұрыс көрсетілген?

1. аты *типі
2. *аты типі
3. **типі *аты**
4. & аты типі
5. тип_аты

54. float *a*[10]; нені білдіреді?

1. 1-ден 10-ға дейінгі бүтін сандар тізбегін сипаттау
2. 1-ден 10-ға дейінгі нақты сандар тізбегін сипаттау
3. **10 нақты саннан тұратын *a* массивін сипаттау**
4. 10 мәнді қабылдай алатын *a* жиымын сипаттау
5. 10x10 өлшемді *a* матрицасын сипаттау

55. switch операциясынан шығу үшін қай нұсқау қолданылады?

1. do
2. **break**
3. case
4. return
5. default

56. Типтерді түрлендіру мысалы. Программа орындалғанда экранға не шығады:

```
main()
{ char ch;
```

```
int i; float fl;
fl=i=ch='A';
printf(" %c %d %6.2f\n",ch,i,fl);}
```

1. **A 65 65.00 ;**
2. B 66 65.000;
3. B 63 63.000;
4. B 62 62.000;
5. A A 65.00.

57. *= операциясы нені білдіреді?

1. көбейту
2. меншіктеу
3. әр элементін көбейту
4. **көбейтіп барып меншіктеу**
5. комментарий

58. Лексем дегеніміз не?

1. сөз тіркестері
2. сөйлемдер
3. **тілдің өзіндік мағынасы бар ең кіші бірлігі**
4. программамның тұлғасы
5. бетті ауыстыру

59. " int n;" жазуы нені білдіреді?

1. n класы берілгенін
2. n спецификаторы берілгенін
3. n айнымалысы нақты мән қабылдайды
4. **n айнымалысы бүтін мән қабылдайды**
5. дұрыс жауабы жоқ

60. string.h – бұл мынадай функциялардың кітапханасы

1. **сөз тіркесі функциялары**
2. стандартты функциялар
3. енгізу/шығару базалық кітапханасы
4. графикалық функциялар
5. буфермен жұмыс істеу функциялары

61. graphics.h - бұл мынадай функциялардың кітапханасы

1. сөз тіркесі функцияларының
2. стандартты функциялардың
3. енгізу/шығару базалық кітапханасының
4. **графикалық функциялардың**
5. буфермен жұмыс істеу функцияларының

62. С тілінде айнымалылар программаның мынадай аймағында сипатталады

1. **басында**
2. аяғында
3. ортасында

4. кез келген жерінде
5. еш жерінде сипатталмайды

63. С тілінде бір таңбамен белгіленген бас және кіші әріптер мәні...

1. әртүрлі болып саналады
2. бірдей болып саналады
3. компиляторға байланысты өзгереді
4. тілдің нұсқасына байланысты өзгереді
5. барлығы да дұрыс

64. Айнымалыларға, тұрақтыларға, мәліметтер типіне және функцияларға қойылған атау былай аталады...

1. идентификаторлар
2. түйінді сөздер
3. директивалар
4. нұсқауыштар
5. дұрыс жауап көрсетілмеген

65. Келесі сөздер ішіндегі нақты сандар типтерін анықтайтын түйінді сөздерді көрсету керек:

1) char, 2) int, 3) float, 4) double, 5) long, 6) long int, 7) short, 8) signed, 9) unsigned

1. 1,4,6,7
2. 2,3,4,8
3. 1,7,8,9
4. 3,4,6

5. 3,4

66. Келесі сөздер ішіндегі бүтін сандар типтерін анықтайтын түйінді сөздерді көрсету керек:

1) char, 2) int, 3) float, 4) double, 5) long, 6) long double, 7) short, 8) signed, 9) unsigned

1. 1,2,5,7,8,9
2. 5,6,7,9
3. 1,2,3,6,9

4. 2,5,7,9

5. 2,7,8,9

67. Директиваның дұрыс жазылған жолын көрсетіңіз:

1. #define PI = 3.1415
2. #define PI == 3.1415
3. #DEFINE = 3.1415
4. #define PI = 3.1415;

5. #define PI 3.1415

68. sizeof() операциясы мынаны анықтайды:

1. литерлік өрнекті
2. константалық тұрақты өрнекті

3. операнд типіне бөлініп берілетін байттар санын
4. операнд орналасатын байттар санын
5. логикалық өрнекті

69. Программа нәтижесі нешеге тең болады?

```
void main()
{int i = 5;
 switch (i++) {case 5: printf("%d %d", 1 , i); break;
          case 6: printf("%d %d", 2 , i); break;
         default: printf("%d %d", 3 , i); }}
```

1. 1 5
2. 2 6
3. 2 5
4. 3 6
5. 1 6

70. Программа нәтижесі нешеге тең болады?

```
void main()
{int i = 5;
 switch (++i) {case 5: printf("%d%d", 1 , i); break;
          case 6: printf("%d%d", 2 , i); break;
         default: printf("%d%d", 3 , i); }}
```

1. 1 5
2. 2 6
3. 2 5
4. 3 6
5. 1 6

71. int a[5] массивінде қандай элемент жоқ?

1. a[1]
2. a[2]
3. a[3]
4. a[4]
5. a[5]

72. Төмендегі қай функция файлға мәлімет жазады?

1. fread()
2. fscanf()
3. gets()
4. fprintf()
5. seek()

73. Төмендегі қай функция файлдан мәлімет оқиды?

1. fprintf()
2. fscanf()
3. fread()

4. fwrite()
5. seek()

74. Файлды одан мәлімет оқу үшін ашу (режимін көрсету керек)

1. r
2. w
3. a
4. r+
5. w+

75. Мәлімет жазатын файл ашу қажет, егер файл бұрыннан бар болса, ондағы мәлімет жойылады (режимін көрсету керек).

1. r
2. w
3. a
4. r+
5. w+

76. Мәлімет қосу: файл соңына мәлімет жазу үшін оны ашу қажет (режимін көрсету керек).

1. r
2. w
3. a
4. r+
5. w+

77. Файлдағы мәліметті оқып, оған мәлімет жазу: оқу және жазу (режимін көрсету керек).

1. r
2. w
3. a
4. r+
5.w+

78. Файлды жаңарту үшін оны ашу, егер файл бұрыннан болса, оның мәліметі жойылады (режимін көрсету керек)

1. r
2. w
3. a
4. r+
5. w+

79. Мәлімет қосу: файлды ашып, оны жаңартады, мәлімет бұрнғының соңына жазылады (режимін көрсету керек).

1. r
2. w
3. a
4. r+
5. a+

80. Төмендегі функция файлдан сөз тіркесін оқиды

1. fread()
2. fgets()
3. fgetc()
4. fputs()
5. fputc()

81. Төмендегі функция файлға сөз тіркесін жазады

1. fwrite()
2. fgets()
3. fgetc()
4. fputs()
5. fputc()

82. printf() функциясының сөз тіркесін шығару кезіндегі форматы қандай?

1. %d
2. %c
3. %s
4. %f
5. %x

83. printf() функциясының нақты сан шығару кезіндегі форматы қандай?

1.%d
2.%c
3.%s
4.%f
5.%x

84. printf() функциясының таңбасыз ондық бүтін сан шығару кезіндегі форматы:

1. %d
2. %u
3. %o
4. %x
5. %f

85. printf() функциясының таңбасыз сегіздік бүтін сан шығару кезіндегі форматы қандай?

1. %d
2. %o
3. %x
4. %c
5. %f

86. printf() функциясының таңбасыз он алтылық бүтін сан шығару кезіндегі форматы қандай?

1. %d
2. %o
3. %x
4. %c
5. %f

87. printf() функциясының экспоненциал нақты сан шығару кезіндегі форматы қандай?

1. %d
2. %o
3. %e
4. %c
5. %f

88. Егер char мен short типтері араласса, нәтижесі қандай болады?

1. char;
2. **short ;**
3. float;
4. long ;
5. int;

89. Мәліметтердің бір еселік дәлдікпен берілетін нақты санды типі қалай сипатталады?

1. float
2. double
3. char
4. int
5. struct

90. Мәліметтердің екі еселік дәлдікпен берілетін нақты санды типі қалай сипатталады?

1. float
2. double
3. char
4. int
5. struct

91. Мәліметтердің символдық типі қалай сипатталады?

1. float
2. double
3. char
4. int
5. struct

92. Тақырып файлдары нұсқаулары қай жерде орналасады?

1. main() функциясы алдында
2. main() функциясы артында

3. main() функциясы ішінде (тұлғасында)
4. жеке файлда
5. керекті функция шақырылатын блокта

93. < және > символдарымен қоршалып тұратын тақырып файл аттары компиляторға бұл файл қандай каталогта орналасқанын білдіреді?

1. LIB
2. BIN
3. INCLUDE
4. BGI
5. TVISION

94. " және " символдарымен қоршалып тұратын тақырып файл аттары компиляторға бұл файл қандай каталогта орналасқанын білдіреді?

1. LIB
2. BIN
3. INCLUDE
4. жұмыс істеп тұрған каталогта
5.TVISION

95. С программасы мәтіндері қандай типті файлдарда сақталады?

1. .txt, .doc
2. .cpp, .c
3. .pas, .bas
4. .h, .htm
5. .html, .xtml

96. Программалардың объектілік кодтары қандай типті файлдарда сақталады?

1. .exe
2. .obj
3. .cpp, .c
4. .com
5. .txt

97. Программалардың атқарылатын екілік кодтары қандай типті файлдарда сақталады?

1. .exe
2. .com
3. .obj
4. .cpp, .c
5. .txt

98. Түйінді сөздер (ключевые слова) дегеніміз не?

1. қарапайым идентификаторлар
2. арнайы қорға енгізілген сөздер

3. операциялар
4. функциялар
5. макрокомандалар

99. С тілінің int типті бүтін саны 16 разрядты процессорда қандай орын алады?

1. 2 байт;
2. 4 байт;
3. 8 байт;
4. 6 байт;
5. 10 байт;

100. С тілінің int типті бүтін саны 32 разрядты процессорда қандай орын алады?

1. 2 байт;
2. 4 байт;
3. 8 байт;
4. 6 байт;
5. 10 байт;

Қолданылған әдебиеттер

1. *Павловская Т.А. С/С++*. Программирование на языке высокого уровня. -СПб.: Питер, 2011. -461 с.
2. *Павловская Т.А. Щупак Ю.А.* С/С++. Структурное и объектно-ориентированное программирование: Практикум.-СПб.: Питер, 2011. -352 с.
3. *Сэмюел П. Харбисон, Гай Л. Стил.* Язык программирования Си. Пер. с англ. –М.: ООО "Бином-Пресс", 2004. -528 с.
4. *Липпман С., Лажойе Ж.* Язык программирования С++. Вводный курс, 3-е изд. Пер с англ. –СПб. –М.: Невский диалект – ДМК Пресс, 2003. -1104 с.
5. *Шиманович Е.Л.* С/С++ в примерах и задачах.-Мн.: Новое знание, 2004. -528с.
6. *Керниган Б., Ритчи Д., Фьюэр А.* Язык программирования Си. -М. : Финансы и статистика, 2004. -271с.
7. *Страуструп Б.* Язык программирования С++, спец. изд. Пер. с англ. –М.: БИНОМ; -СПб.: Невский диалект "БИНОМ", 2008. -288 с.
8. *Вирт Н.* Алгоритмы и структуры данных: Пер. с англ. 2-е изд., испр. -СПб.: Невский диалект,2008. -352с.
9. *Абрамов С.А., Гнездилова Г.Г., Капустина Е.И., Селюн М.И.* Задачи по программированию. -М.: Наука, Гл. ред. физ.-мат. лит. 1988. -224 с.
10. *Культин Н.Б.* С/С++ в задачах и примерах. 2-е изд., перераб. и доп. -СПб.: БХВ-Петербург, 2009. -368 с.
11. *Березин Б.И., Березин С.Б.* Начальный курс С и С++ . -М: ДИАЛОГ-МИФИ, 2004. -288с.
12. *Подбельский В.В., Фомин С.С.* Программирование на языке Си. -М.: ФиС, 2004. –600 с.
13. Подбельский В. В. Язык С++. -М.: ФиС, 2001. -559с.
14. *Костюкова Н.И., Калинина Н.А.* Язык Си и особенности работы с ним. –М.: "Интернет-университет информационных технологий - ИНТУИТ. ру", 2006. -208 с. // www.intuit.ru
15. *Анисимов А.Е., Пупышев В.В.* Сборник заданий по основаниям программирования. 2007. // www.intuit.ru
16. *Бөрібаев Б., Дүйсебекова К.* Си тілінде программалау: Оқу-әдістемелік құрал. –Алматы: Қазақ университеті, 2007. -208 б.
17. *Бөрібаев Б.* Программалау тілдеріне кіріспе: Жоғары оқу орындарына арналған оқулық . –Алматы: АЭСА, 2008. -376 б.
18. *Құралбаев З.Қ.* Алгоритмдеу және программалау тілдері. –Алматы: "TST-company" баспасы, 2008. -354 б.
19. *Бөрібаев Б.* Компьютердің арифметикалық негіздері:Оқу құралы. -Алматы: Қазақ университеті, 2009. -80 б.
20. *Балапанов Е., Бөрібаев Б., Бекбаев А.,*т.б. Информатика терминдерінің қазақша-ағылшынша-орысша, орысша-қазақша-ағылшынша, ағылшынша-орысша-қазақша сөздігі. -Алматы: Сөздік-Словарь,1998. -176 б.

Стандартты математикалық функциялар

`Math.h` тақырып файлында стандартты математикалық функциялардың анықтаулары берілген, оларды программа құру кезінде дұрыс жаза білу керек (Қ1.1-кестені қ.).

Қ1.1-кесте

Функция прототипі	**Қайтаратын мәні**
`double acos(double x)`	arccos x
`double asin(double x)`	arcsin x
`double atan(double x)`	arctg x
`double atan2(double x,double y)`	arctg (y/x)
`double ceil(double x)`	"жоғары" қарай дөңгелектеу
`double cos(double x)`	cos x
`double cosh(double x)`	ch x
`double exp(double x)`	e^x
`double fabs(double x)`	| x |
`double floor(double x)`	"төмен" қарай дөңгелектеу
`double log(double x)`	ln x
`double log10(double x)`	lg x
`max(a,b)`	максимум (a,b) типі үлкен аргумент типімен бірдей болады
`min(a,b)`	минимум (a,b) типі кіші аргумент типімен бірдей болады
`double pow(double x, double y)`	x^y
`double pow10(int p)`	10^p
`double sin(double x)`	sin x
`double sinh(double x)`	sh x
`double sqrt(double x)`	x-тің квадрат түбірі
`double tan(double x)`	tg x
`double tanh(double x)`	th x
`double hypot((double x, double y)`	x^2+y^2 квадрат түбірі
`double poly(double x, int n, double *a)`	полином мәні
`double ldexp(double x, int n)`	$x*2^n$

Бұл кестедегі max және min функцияларының прототиптері, негізінде, функция емес, өздеріне сәйкес макроанықтаулар түрінде stdlib.h тақырып файлында берілген.

Аргументтері типі де, өз типі де double болып келген функциялардың long double типіндегі аналогтары бар. Олардың аттары жоғарғы кестеде көрсетілген функция аттарынан l әрпін қосу арқылы алынады, мысалы, (fabs, fabsl), (acos, acosl), т.с.с.

Ceil(x) және floor(x) функцияларының айырмашылығы – алғашқысы берілген нақты санды жоғарғы жақтағы бүтін санға, ал екіншісі – төменгі жақтағы бүтін санға қарай дөңгелектейді. Екі функция да double форматындағы бүтін санды қайтаратынын есте сақтаған жөн. Мысалы:

```
ceil(0.1) =1.0                floor(0.1) = 0.0
ceil(0.5) =1.0                floor(0.5) = 0.0
ceil(0.9) =1.0                floor(0.9) = 0.0
ceil(-0.9)=0.0                floor(-0.9)=-1.0
ceil(-0.5)=0.0                floor(-0.5)=-1.0
ceil(-0.1)=0.0                floor(-0.1)=-1.0
```

Көбінесе программалаушылар дөңгелектеу үшін floor(x+0.5) функциясын пайдаланады. Бірақ ол кейде математикадағы ұғымнан алшақ кетіп жатады, мысалы,

floor(-0.5+0.5)=0.

Әрине, C/C++ тілдерінде Паскаль тіліндегідей round функциясының баламасы жоқ, дегенмен ондай функцияны қолдан жасап алуға да болады:

```
int round(double x)
{ int res;
 res=(x<0)? x-0.5 : x+0.5;
 return res;
}
```

Егер нақты санның бірнеше бөлшек таңбаларын (k) алып дөңгелектеу керек болса, онда ол санды 10^k-ға көбейтіп алып, бүтін санға дейін дөңгелектеп, қайтадан 10^k-ға бөлу керек.

Math.h тақырып файлында стандартты константалардың да анықтаулары берілген, оларды да программа құру кезінде дұрыс жазып пайдалана білу керек (Қ1.2-кестені қ.).

Қ1.2-кесте

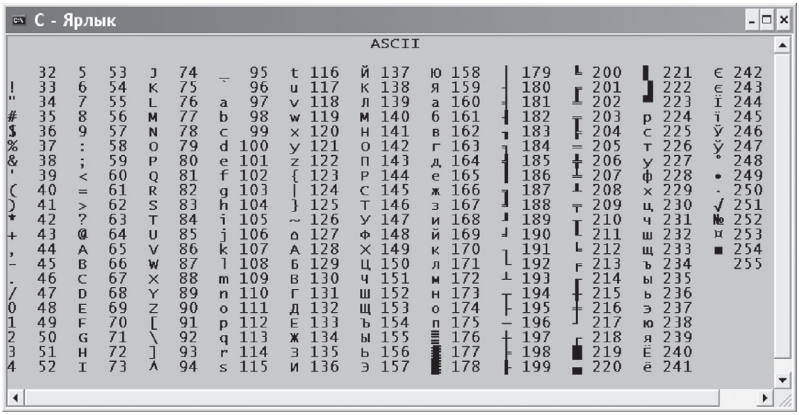
Константа	Мәні	Константа	Мәні
M_PI	π	M_E	e=2.71828…
M_PI_2	$\pi/2$	M_LOG2E	$\log_2 e$
M_PI_4	$\pi/4$	M_LOG10E	log e
M_2_PI	$2/\pi$	M_LN2	ln 2
M_1_SQRTPI	$1/\sqrt{\pi}$	M_LN10	ln 10
M_2_SQRTPI	$2/\sqrt{\pi}$	M_SQRT2	$\sqrt{2}$=1.414…
		M_SQRT_2	$\sqrt{2}/2$

Символдық мәліметтер және оларды компьютерде бейнелеу

Жеке берілген символдық мәліметтер (константалар мен айнымалылар) компьютер жадында сол символдардың ASCII-кодтарына сәйкес бүтін сандық мәндер жазылған бір-бір байт орын алады. Соларды экранға шығару үшін келесі программаны орындауға болады:

```
#include <stdio.h>
#include <conio.h>
void main()
{
 int i,j;
 gotoxy(37,1);
 printf("ASCII");
 for(i=32; i<=52; i++)
  { gotoxy(1,i-29);
    for(j=i; j<=255; j+=21)
     printf("%c %3d ",j,j);
  }
 getch();
}
```

Бұл программа жұмысының нәтижесі:



Қ1.2-сурет. Жеке символдық мәліметтердің `ASCII-кодтық кестесі` (code page 866)

Кестедегі символдар "босорын" таңбасы кодынан – 32-ден (0x20) басталады, оған дейінгі символдар басқару кодтарына сәйкес келеді. Орыс әліпбиінің бас әріптері 128 (0x80) кодынан басталады ('Ё' әрпі басқа орында тұр), ал кіші әріптері 160-тан 175-ке (а-дан п-ға) дейін, сонан соң 224-тен 239-ға (р-ден я-ға) дейін орналасқан. Олардың арасында кестелер салу кезінде қолданылатын графикалық символдар орналасқан. Қазақ әріптерін пайдалану үшін Unicode кестесіне көшу керек, сондықтан C/C++ тілдерінде экранға қазақ әріптері шығарылмайды.

Бірбайттық символдық константалардың мәндері жалқы тырнақшаға (' – апострофқа) алынып жазылады да, үш түрлі тәсілмен беріледі. Олардың алғашқысы экранда бейнеленетін ASCII-код кестесінің символдарына (Қ1.2-суретті қ.) қатысты айтылады, бұлар апострофқа алынып жазылады:

'F', '%', '$', 'ы', '5', '+', '"', 'q', 'Я', ' '

Соңғы константа "босорын" таңбасы. Апостроф таңбасын мұндай тәсілмен жаза алмаймыз. Сондықтан басқа тәсілді қолданамыз.

Екінші тәсіл кері қиғаш сызық (слеш) таңбасынан кейін керекті символдың оналтылық кодын жазу арқылы беріледі:

'\x27' ("апостроф" символының коды, $39_{10} = 27_{16}$)

'\x5C' (кері қиғаш сызық \ символының коды, $92_{10} = 5C_{16}$)

Он алтылық кодтың алдыңғы нөлсіз жазылатынына назар аударыңдар (оналтылық сандар түрінде берілген константаларды жазу кезінде \0x болып бейнеленеді). Осы тәсілді ASCII-код кестесінің бас жағында (1-31 кодтары) орналасқан басқару кодтарының кез келгенін жазу үшін қолдануға болады. C/C++ тілдеріндегі Escape-тізбектері болып табылатын басқару кодтары (4.6-кестені қ.) да осы тәсілмен жазылады. Мысалы:

'\'' – "апостроф" символы

'\\' – "кері слеш" символы

Үшінші тәсіл "кері слеш" таңбасынан соң, символдың сегіздік кодын жазу болып табылады:

'\47' ("апостроф" символының коды, $39_{10} = 47_{8}$)

'\134' (\символының коды, $92_{10} = 134_{8}$)

Мұндағы сегіздік кодтың да алдыңғы нөлсіз жазылатынына назар аударыңдар (сегіздік сандар түрінде берілген константалар-

ды жазу кезінде олардың алдында \0 таңбасы жазылады). Апостроф ішіндегі сегіздік код [0, 377] аралығында болуы тиіс.

Символдық айнымалылар олардың типін көрсететін `char` немесе `unsigned char` спецификаторы (түйінді сөзі) арқылы жарияланады. Оларға бірден мән беруге, яғни айтылған тәсілдердің біреуімен инициалдауға болады:

```
char ch1='Я',ch2='\x9F',ch3='\237',
ch4= 0x9F,ch5=0237,ch6=159;
```

Осылай сан арқылы мән тағайындау тәсілі DOS-қосымшаларында да және Windows ортасында да символдардың дұрыс бейнеленуіне кепілдік бере алады. Windows ортасында консолдық программалар құру кезінде символдық мәндер тағайындау тәсілі `ASCII`-код кестесінің тек бірінші жартысында орналасқан символдар үшін ғана дұрыс жұмыс істей алады.

Сөз тіркестерінен тұратын мәліметтермен операциялар орындау

Бірөлшемді символдық жиымдар (массивтер) түрінде берілетін сөз тіркестерін өңдеу үшін C/C++ тілдерінің жүйелік функциялар кітапханасында бірсыпыра амалдар (операциялар) қарастырылған. Олардың прототиптері `string.h` тақырыптық файлына жинақталып, көбісінің атаулары `str` (ағылш. `string` – тіркес сөзінің алғашқы әріптері) таңбаларынан басталады. Тіркестерді өңдейтін функцияларды қарастыру үшін, қолданылатын аргументтер мен олардың типтері жайлы ортақ белгілеулер туралы бірсыпыра түсініктер бере кетейік:

`S, S1,S2` – символдық жиымға нұсқауыш (көбінесе жиым аты);

`CS` – `const char *` типіндегі нұсқауыш (яғни өзгермейтін жиым немесе бастапқы мәліметтер түрінде берілген тіркестік константа);

`ch` – символ коды, көбінесе `int` типіндегі сандық мән;

`k` – символдар саны.

Қ1.3-кесте

Функция	Атқарылатын операция
Сөз тіркесінің (тіркестің) ұзындығын анықтау	
`strlen(CS)`	`S` тіркесіндегі символдар санын береді
Тіркестерді қалыптастыру	
`strcpy(S1,CS2)`	`CS2` тіркесін `S1`-ге көшіреді, нұсқауышты `S1`-ге қайтарады
`strncpy(S1,CS2,k)`	`CS2` тіркесінің алғашқы `k` символын `S1`-ге көшіреді, нұсқауышты `S1`-ге қайтарады
`strset(S,ch)`	`S` тіркесін `ch` символдарымен толтырады, нұсқауышты `S`-ке қайтарады
`strnset(S,ch,k)`	`S` тіркесінде `ch` символын `k` рет қайталайды, нұсқауышты `S`-ке қайтарады
Тіркестерді біріктіру (конкатенация)	
`strcat(S1,CS2)`	`CS2` тіркесін `S1` тіркесінің соңына қосып жазады, нұсқауышты `S1`-ге қайтарады (`S1` тіркесінің ұзындығы екі тіркес сиятындай көлемді болуы тиіс)

`strncat(S1,CS2,k)`	`CS2` тіркесінің алғашқы `k` символын `S1`тіркесіне қосып жазады да, нұсқауышты `S1`–ге қайтарады
Регистрді ауыстыру	
`strlwr(S)`	`S` тіркесі символдарын кіші әріптерге ауыстыру, тек латын әліпбиі әріптеріне әсер етеді
`strupr(S)`	`S` тіркесі символдарын бас әріптерге ауыстыру, тек латын әліпбиі әріптеріне әсер етеді
Тіркесті керісінше жазып шығу	
`strrev(S)`	`S` тіркесіндегі символдарды кері бағытта жазып шығады
Сандық мәліметке түрлендіру	
`strtol(CS,ptr,r)`	`CS` тіркесінде символдық түрде бейнеленіп, негізі `r` болып келетін санау жүйесінде жазылған сан `long` типіндегі машиналық форматтағы санға түрлендіріледі. `ptr` нұсқауышына түрлендіруді тоқтатқан (үзген) символ адресі жазылады. Қайтарылатын мән – түрлендіру нәтижесі
`strtoul(CS,ptr,r)`	Жоғарыдағыдай түрлендіру ісі екі еселенген таңбасыз бүтін сан үшін атқарылады
`strtod(CS,ptr)`	Нақты санды символдық түрден `double` типіндегі машиналық форматқа түрлендіру
Тіркестерді салыстыру	
`strcmp(CS1,CS2)`	Егер `CS1=CS2` болса, функция 0 мәнін қайтарады, егер `CS1>CS2` болса, функция 0-ден артық мән, ал егер `CS1<CS2` болса, функция 0-ден кіші мән қайтарады
`strncmp(CS1,CS2,k)`	`CS1` және `CS2` тіркестерінің алғашқы `k` символдары ғана салыстырылады
`stricmp(CS1,CS2)`	Салыстыру кезінде бас әріптер мен кіші әріптер кодтарының әртүрлілігі есепке алынбайды
`strcmpi(CS1,CS2)`	Жоғарыдағы операция сияқты, тек функция аты ғана басқаша

`strnicmp(CS1,CS2,k)`	Екі тіркестің алғашқы `k` символдарын бас әріп пен кіші әріп кодтарының айырмашылығын есепке алмай салыстырады
`strncmpi(CS1,CS2,k)`	Жоғарыдағы операция сияқты, тек функция аты ғана басқаша
Символды іздеу	
`strchr(CS,ch)`	`CS` тіркесі солдан оңға қарай `ch` символы табылғанша біртіндеп қарастырылады. Егер ол табылса, нұсқауыш `CS` тіркесіндегі осы символды (позициясын) көрсетіп тұрады, егер ондай символ табылмаса, онда нұсқауыш `null` (яғни 0) мәнін қайтарады
`strrchr(CS,ch)`	Жоғарыдағы операция сияқты, тек символды іздеу `CS` тіркесінің соңынан басына қарай жүргізіледі
Ішкі тіркесті іздеу	
`strstr(CS1,CS2)`	`CS2` тіркесін `CS1` тіркесінің құрамынан іздеу (алғашқы кіруін анықтау). Егер ол табылса, нұсқауыш табылған тіркестің алғашқы символын көрсетеді, табылмаса, ол `null` мәнін береді
Арнайы іздеу	
`strpbrk(CS1,CS2)`	`CS2` символының `CS1` тіркесіндегі алғашқы кездесуі ізделеді. Табылған символға нұсқауыш немесе null қайтарылады
`strspn(CS1,CS2)`	Толығынан `CS2` символдарынан тұратын `CS1` тіркесінің бастапқы фрагментінің ұзындығы анықталады (символдар реттілігі ешқандай рөл атқармайды)
`strcspn(CS1,CS2)`	`CS2` символдарының бір де бірі құрамына кірмейтін `CS1` тіркесінің бастапқы фрагментінің ұзындығы анықталады
`strtok(S1,CS2)`	`S1` тіркесі ішінен `CS2` символдарымен бөлінген лексемдерді іздеу

Қ1.3-кестесінде көрсетілген функцияларға аздаған қосымша түсініктер бере кетейік.

Санның символдық бейнеленуін соған сәйкес машиналық форматқа түрлендіретін `strtol` және `strtoul` функцияла-

рында оның негізін r=0 деп те беруге болады. Мұндайда санау жүйесінің негізі санның символдық жазылуымен анықталады. Егер тіркес '0' символынан басталып, ары қарай 7-ден аспайтын цифрлар орналасатын болса, онда ол сегіздік сан болып саналады. Ал егер сөз тіркесі '0x' немесе '0X' символдарынан басталып, ары қарай оналтылық цифрлар орналасатын болса, онда r=16 болып есептеледі.

Strtok функциясында лексем болып ажыратқыш символдарының бірімен (босорын, үтір, нүкте, т.с.с.) аяқталатын символдар тіркесі саналады. Ол функцияны алғашқы рет пайдалану кезінде S1 тіркесінде бастапқы лексем орналасып, қайтарылатын мән оның бірінші символына нұсқауыш болып саналады. Осымен қатар S1 тіркесіндегі анықталған ажыратқыш символ орнына нөлдік байт жазылады. Бұл табылған лексеммен кейіннен сөз тіркесі ретінде жұмыс істеуге мүмкіндік береді. Мұнан кейінгі strtok функциясын пайдалану кезінде келесі лексемдерді іздеу үшін бірінші аргумент орнына нөлдік аргумент жазуға болады. Сонда функция енгізілген нөлдік байттан оң жақта орналасқан келесі лексемді іздеп табуға кіріседі. Осы тәсілмен S1 тіркесіне кіретін барлық лексемдерді біртіндеп тауып алуға болады. Түсіндіру үшін бір мысал келтірейік:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{ char *ptr;
 ptr=strtok("FEB.14,2009",".,-/");
 while(ptr!=NULL)
 { printf("ptr=%s\n",ptr);
  ptr=strtok(NULL, ".,-/");
 }
 getch();
}
```

Бұл программа нәтижесі

```
ptr=FEB
ptr=14
ptr=2009
```

МАЗМҰНЫ

Б. Бөрібаев

ПРОГРАММАЛАУ ТЕХНОЛОГИЯЛАРЫ

Оқулық

Басуға 21.12.11. қол қойылды. Қағазы офсеттік.
Қаріп түрі “Таймс” Пішімі 60х90/16. Баспа табағы 22.
Таралымы 2100 дана. Тапсырыс 1660.

Тапсырыс берушінің дайын файлдарынан басылып шықты.

ЖШС РПБК «Дәуір», 050009,
Алматы қаласы, Гагарин д-лы, 93а.
E-mail: rpik-dauir81@mail.ru