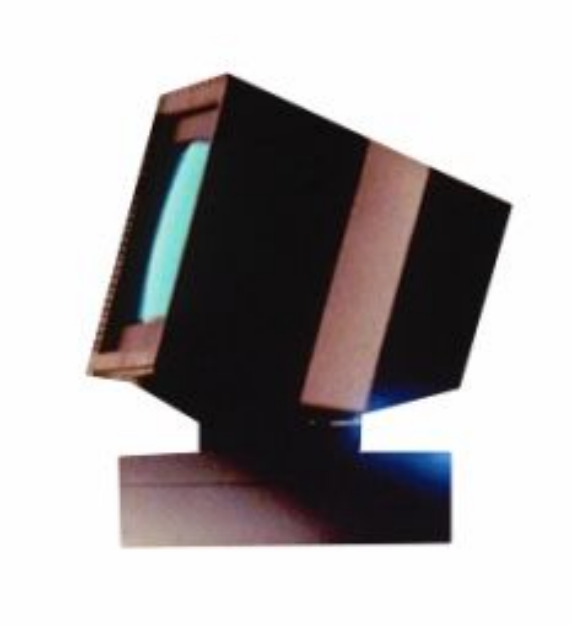
Т.Б.Нұрпейісова

БАҒДАРЛАМАЛАУ ТІЛДЕРІНІҢ НЕГІЗІ



Алматы 2003

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Қ.И. Сәтбаев атындағы

ҚАЗАҚ ҰЛТТЫҚ ТЕХНИКАЛЫҚ УНИВЕРСИТЕТІ

Т.Б.Нұрпейісова

БАҒДАРЛАМАЛАУ ТІЛДЕРІНІҢ НЕГІЗІ

Республикалық оқу әдістемелік бірлестігі оқу құралы ретінде ұсынған

Алматы, 2003

ЖОК 681.322.181.4(075)
ББК32.973-018я7

Нұрпейісова Т.Б. Бағдарламалау тілдерінің негізі:. Оқу құралы.-Алматы: ҚазҰТУ, 2003, 84 б.
ISBN 9965-673-27-3

“Бағдарламалау тілдерінің негізі” атты оқу құралы - технологиялық мамандықтардың алғашқы курсында оқылатын “Ақпараттану” пәнінің мемлекеттік стандартына сай құрастырылып, оның жұмыс бағдарламасының 4-ші модуліне сай жазылған. Ол студенттерді оқу процесінде өтетін тақырыптарды өздігінше оқып дағдылануға бағыттайды. Оқу материалы жүйеленіп, дәйекті түрде берілген. Әр алгоритмнің ерекшелігі, оның теориялық негіздері, бағдарлама құру, өңдеу тәсілдері толық талданған. Оқу құралы студенттердің білім деңгейін арттырып, тақырыпты оқып-үйрену негізділігін, оқу-мақсаттық мәселелері мен олардың өзара байланыстылығын қарастырады.

Оқу құралы барлық технологиялық мамадықтардың студенттеріне “Ақпараттану” пәнін оқып-үйрену үшін керек.

ББК32.973-018я7

Сурет - 14. Кесте - 8. Әдебиеттер тізімі - 5 атау.

Пікір жазғандар: *Б.Д.Хисаров*, техн. ғыл. канд.,
профессор, “АЭжБИ”;
Е.Ж.Айтхожаева, техн. ғыл. канд.,
профессор, ҚазҰТУ;

Қазақстан Республикасы Білім және ғылым министрлігінің 2003 жылғы жоспары бойынша басылады.

$$H\frac{4310020000}{00(05)-03}$$

ISBN 9965-673-27-6
©ҚазҰТУ, 2003

АЛҒЫ СӨЗ

Бұл оқу құралындағы материал Қ.И.Сәтбаев атындағы ҚазҰТУ-дың барлық технологиялық мамандықтарындағы студенттерге арналған “Ақпараттану” пәнінің бір тарауында оқылады.

Автордың оқу құралын жазу мақсаты – соңғы кезеңдегі есептеу техникасының көмегімен бағдарлама құру, оны құру кезіндегі әдістемелік және оқу құралының мазмұндық біртектілігін, берілген материалды игеру жеңілділігі мен түсініктілігін қамтамасыздандыру.

Оқу құралында есепті ЭЕМ-да шешуге арналған барлық дайындық кезеңдері, есепті шешуден санау нәтижесін алу мен өңдеуге дейін комплексті және әртүрлі мысалдар келтіру негізінде қарастырылған. Қазіргі кездегі ЭЕМ бағдарламалық қамтамасыздандыру жүйелерінің мүмкіншіліктерін толығымен пайдалануға назар аударған.

Оқу құралының педагогикалық тұжырымдамасында барлық технологиялық мамандықтардағы “Ақпараттану” пәнінде автордың дәрістегі оқыған – алгоритм құру, бағдарламалау, есепті ЭЕМ-да шешудегі әдістемесі мен материалы алынған. Компьютерде бағдарлама құруға қажетті – компьютер негізі, оны жұмысқа қосу, басқару, кез келген бағдарламамен жұмыс істеу негізі, автордың авторлар тобымен бірігіп шығарған “Компьютерлік технологиялар және бағдарламалау” оқу құралында қарастырылған.

“Бағдарламалау тілдерінің негізі” оқу құралында автордың алдына қойған мақсаты – есепті ЭЕМ-да шешу үшін берілген есептің алгоритмі мен бағдарламасын құруда студенттердің іс-тәжірибелік дағдысын дамытып, лабораторлық жұмыстарды орындауды жеңілдету болып табылады.

1. БАҒДАРЛАМАЛАУ ТІЛДЕРІНІҢ НЕГІЗДЕРІ

Бағдарламалау тілдері - формальды электронды ЕМ түсінетін, адамдардың машинамен қатынасын туғызу үшін жасалған тіл.

Траслятор – пайдаланушылардың кіріс тілінде жазылған бағдарламаларды машина тіліне аударатын бағдарлама немесе құрылғы. Қызмет ету бағдарламаласы - машина жүйесін пайдаланушыларға көмектесетін барлық басқарылмайтын бағдарламалар қосындысының жалпы аталымы.

1.1. ЭЕМ-де есептерді дайындау мен оларды шешу әдістемесі

ДЭМ-де есепті шығаруға төмендегі негізгі кезеңдер жатады, бірақта олардың кейбір бөліктері компьютерсіз орындалады:

1. Есепті қою және оның математикалық моделін құру;
2. Есепті шығарудың сандық әдісін таңдау;
3. Алгоритмнің мәлімет құрылымын құру;
4. Алгоритімді бағдарламалау тілінде жазу;
5. Бағдарламаны тексеру және түзету;
6. ЭЕМ-де есеп шығару;
7. Нәтижелерді өңдеу мен безендіру.

1.1.1. Есептің қойылымы

Бұл кезеңде есеп зерттеулерінен: анықтауға келетін, айнымалы, шектеулер жазылады және есептің математикалық моделін құрайтын айнымалылар арасындағы байланыс таңдалады. Нәтижесінде инженерлік есеп, формаланған математикалық есепке айналады.

1.1.2. Сандық әдіспен шығаруды таңдау

Қойылған математикалық есепті сандық әдіспен шығару керек, өйткені, ол арифметикалық және логикалық операцияларды тізбектей шығаруға мүмкіндік береді. Мұндай әдістерді құру және зерттеумен математиканың сандық анализі деп аталатын бөлімі шұғылданады. Сандық әдістерге

мысал ретінде, анықтауыш интегралды – тік бұрыштар әдісі арқылы шығаруға болады. Бұл әдісте интеграл соңғы қосындымен анықталады. Бұл әдіс арқылы анықтауыш интегралдың шығарылуы – интеграл асты функция мәндерін, олардың қосындысын және оларды интеграл қадам шамасына көбейту, тізбекті түрде өткізіледі.

Егер есептің бастапқы шарттары - берілген жай дифференциалды теңдеулер жүйесін шешуден тұрса (Коши есебі), онда оған Эйлер, Рунге-Кутта, болжам-тексеру және басқа да әдістерді пайдалануға болады. Әдісті таңдау кезінде есепке қойылатын талаптар мен оның ЭЕМ-де орындалу мүмкіндіктерін ескеру керек: есеп дәлділігі, нәтижені жедел алу, бастапқы және аралық нәтижелерді сақтауға қажетті жедел жад көлемінде қойылатын талаптар (көп көлемді есеп үшін), бағдарламаның орындалу күрделілігін, бағдарламаны дайындау мен есепті шығару бағасы. Есеп күрделі болмаған жағдайда, оны бағдарламалық түрде орындау үшін қарапайым немесе стандартты кітапханалық қосымша бағдарламасы бар әдіс қарастырылады. Бұл кітапханалық бағынышты бағдарламалар әдетте каталогтар арқылы сипатталған. Оларды қолдану - бағдарламаларды құруды жеңілдетеді.

1.1.3. Алгоритм мен мәлімет құрылымын құру

Есепті шығару үшін таңдалған әдіс – стандартты кітапханалық бағдарлама түрінде орындалса, онда алгоритм бастапқы берілгенді енгізу – сипаттаудан, стандартты бағынышты бағдарламаны шақыру мен нәтижелерді экранға немесе баспаға шығарудан құралады. Кейде бағынышты бағдарламалар есептің кейбір бөлігін ғана шығарады (есептейді). Мұндай жағдайда, есеп шығару кезеңдерінің тізбегін анықтау қажет. Өйткені, олардың кейбіреулері стандартты бағынышты бағдарламалар арқылы орындалып, қалғандарын өз бетінше бағдарламалау қажет.

Алгоритм құрылымын көрсету үшін әртүрлі бағдарламалау тілдері қолданылады. Сондықтан, алгоритм логикасын құру кезеңінде-ақ оның ерекшеліктерін ЭЕМ-ге кіру тілінде орындалуын ескеруге тырысады. Мұндай бағыт – алгоритм логикасы шектен тыс күрделі болмаған жағдайда ғана тиімді. Егер есеп күрделі болса, алгоритмді құру

барысында барлық қиыншылықтарды шешуге асықпаңыз. Қазіргі таңда күрделі бағдарламаларды құру бағыты жоғарыдан төмен жобалау принциптерінен, модульдік және құрылымдық бағдарламалау тізбегін қолданудан тұрады. Айталық, есептің анық құрылымдылығы, бағынышты есептер тізбегіне бөлінуі, бағынышты есептердің жеке модульдермен орындалуы, алгоритм логикасының біртіндеп қарастырылуы, типтік-логикалық конструкцияларды қолдануда өте жақсы құрал. Олар негізінде қысқа мерзімде күрделі есептерді шешуге арналған жұмысқа жарамды бағдарламалар құруға мүмкіндік жасайды. Бұл принциптерді білу және қолдану тек маман - бағдарламалаушыларға ғана емес, сонымен қатар ЭЕМ-де кез келкен өз есептерін шығаратын инженер мен зерттеушілерге де қажет. Мәліметтер - құрама таңдау жолы мен ұйымдастыру тәсілдері алгоритмінің негізгі құрама бөлімі: бастапқы, аралық нәтижелер. Мәлімет құрылымын, түрін, құрамын таңдау, әдетте алгоритмнің ЭЕМ-ге кіру тілінің орындалуына байланысты. Графикалық блок-схема арқылы алгоритмнің құрылуы көрсетіліп аяқ-талады. Мұндай көрсетілудің мақсаты – оның бағдарламалау кезеңіне дейін құрылып жатқан алгоритм логикасының ақиқаттылығын тексеру.

1.1.4. Алгоритмді бағдарламалау тілінде жазу

Сандық әдіспен жобалау алгоритмін таңдау кезінде, белгілі алгоритмдік тілге бағдарлама болмаған. Ендігі кезек, бағдарламалау – алгоритмін ЭЕМ-нің кіру тілінде жазу. Бейсик, Паскаль тілдерінің әмбебаптығына және белгілі машинаға тәуелсіздігіне қарамастан, әр ЭЕМ кейбір шектеулерді рұқсат тілдерінің конструкцияларына, мәлімет түрлеріне, форматына, т.б. қояды.

1.1.5. Бағдарламаны тексеру және түзету

ДЭМ-да есеп шығару кезінде жүйемен байланыс процедурасы жеңілдетіледі. Бағдарламалау тілдерінің бірін пайдаланғанда, бағдарламаны жедел жадқа пернелер тақтасы арқылы қатар-қатарымен енгізеді. Бағдарламаны енгізу барысында синтаксистік бақылау жүргізіледі. ДЭЕМ-ның

ОЖ-де жұмысын басқару үшін ортаның менюі немесе арнайы пернелер қолданылады, оларды басқару барысында сәйкесті іс-қимыл орындалады:

- бағдарламаны есепке жіберу;
- бағдарлама мәтінін экранға шығару;
- бағдарлама қатарын түзету режиміне көшу,т.б.

Бағдарламаны бір бұйрық көмегімен иілгіш дискіден жедел жадқа жүктеуге немесе оны иілгіш дискіге жазып және ақпаратты монитор экранына не баспаға шығаруды дайындауға болады. Мәліметтер мен бағдарламаларды пернелер тақтасы арқылы ЭЕМ-ге енгізу кезеңінде қате кетуі мүмкін. Сондықтан бағдарламаны тексеру және түзету кезінде қателерді анықтау, жою амалдары орындалады. Түзету кезеңінің маныздылығы мен күрделілігіне байланысты әр дамыған бағдарламалау жүйелерінің қатені анықтауға, жоюға көмектесетін арнайы құралдары бар. Айталық, алгоритмді құру кезеңінде құрылған бағдарламаға кіретін қарапайым бақылау құралдарын қарастыру қажет: енгізілген мәліметтерді компьютер (эхо-печать) жадына орналастырғаннан кейін экранға немесе баспаға шығару (печать), түйінді нүктелерде аралық нәтижелерді баспаға немесе экранға шығару. Бағынышты бағдарламалар немесе прцедуралар модуліне бөлінудің арқасында, бағдарламаның құрылымын максимальды қарапайымдатуға (жеңілдетуге) және неғұрлым жай бағдарламалаушыға үйреншікті тілдер конструкциясын пайдалану – ең басты жұмыс болып табылады.

Бағдарламаны тестілеу мақсаты: дәл сол есепке ұсынылғаны тексеріліп, қойылған әртүрлі шарттарға дұрыс жауап беру. Бағдарламалар тестілеуді әдетте арнайы алынған бақылау мысалдары арқылы орындайды. Тексеру көлемі, бағдарламаның жиі қолданылуына және пайдалану ұзақтығына байланысты болады. Бір-екі рет пайдаланылатын бағдарламалар үшін мұндай көп көлемді сынақ керек емес, ол тек қана көп уақыт бойы, жұмыс істейтін бағдарламалар үшін қажет. Егер бағдарламаны құруда модульдік принципі пайдаланылса, тестілеу процесі қапапайымдатылады. Бұл жағдайда, модульдер қатаң түрде тестіленеді де, содан кейін модульдер арсындағы мәлімет алмасу дұрыстығын бақылауға бағытталған комплексті тексеру орындалады.

1.1.6. ЭЕМ-да есеп шығару

Бұл кезеңде ЭЕМ-сы бағдарламамен қарастырылған есептеулерді орындайды және оның нәтижесін экранға немесе баспаға шығарады. Келесі есеп нәтижелерін өңдеуді жеңілдету үшін, бағдарламаларды жобалауда нәтижелерді түсініктемелерімен бірге шығаруды қарастыру керек, шығарылған сандар қандай айнымалыларға сәйкес келеді: нәтижелер кестенің басында тақырыпша ретінде болуы тиіс. Мәлімет топтарын бір-бірінен бос қатармен бөліп, табуляциялау кезінде аргумент мәндерін көрсеткен дұрыс. Егер олар параметрлер арқылы байланысса, онда мәндерді шығарар алдында параметр мәндерін көрсету керек. Шығарылатын инженерлік есеп вариантын көрсетіп, шын мәнінде математикалық модельдің кейбір тәжірибесінің нәтижесі болса, онда нәтижелерді баспаға бермес бұрын, тәжірибе шарттарын көрсететін мәліметтерді қарастырған жөн. ЭЕМ-де нәтижелерді өңдеу, есептеу және безендіру сайып келгенде, ЭЕМ-нен алынған мәліметтерді талдауда және оларда қарастырылған сала аймағының ізделген жоба шешімдерін бағалау мен пайдалану болып табылады.

1.2. Алгоритмнің қасиеттері мен блок-схема, бағдарлама туралы түсініктер

Енді алгоритм, блок-схема және бағдарлама туралы түсініктерге тоқталайық. Есепті ЭЕМ-ға тапсырмас бұрын, әуелі есептеу процесін белгілі бір жүйемен сипаттап жазу керек. Мәселен, орындалатын амалдардың ретін немесе тізбегін дайындап алу керек.

1.2.1. Алгоритм және оны өрнектеу тәсілдері

ЦЕМ-да есептер алдын ала жасалған алгоритмге сай шығарылады. *Алгоритм* - математика мен есептеуіш техниканың негізгі ұғымының бірі. ***Алгоритм*** *деп - белгілі бір есептер тобына кіретін, кез келген есепті, бір мағыналық шешімге алып келетін формальды ережелер жүйесін айтады.*

Алгоритм теориясында - есеп машиналарының бағдарламалау теориясына негіз болатын алгоритмдік жүйелер,

алгоритімдік тілдер және цифрлы-автоматтардағы бейнелерді кескіндеу дәлдігі жатады.

Алгоритмнің негізгі жазылу тәсілдеріне:

- сөз және формула;
- құрлымды немесе блок-схема;
- граф-схема;
- Петри торлар жатады.

Есептерді шешу үстінде арифметикалық амалдарға алып келетін алгоритмді - *сандық алгоритмдер* деп атайды.

Егер қойылған есеп - логикалық амалдарға байланысты болса, онда ондай алгоритмдерді - *логикалық алгоритмдер* деп атайды. Логикалық алгоритмдерге: графтағы қысқа жолды табу, сандар тобындағы ең үлкен, ең кішісін табу сияқты алгоритмдер жатады.

Алгоритмдерді формальді түрде жазуға да болады.

Ал *оператор* деп - ақпаратты өңдеу мен алгоритм актісін формальді түрде көрсетуді айтады. Алгоритмдердің үш операторлық түрі болады: *алгоритмдердің сызба схемасы, алгоритмдердің матрицалық схемасы, алгоритмдердің логикалық схемасы.*

Алгоритмнің қасиеттері:

1. Есепті шешу жолын, бір мағыналы түрде анықтау.
2. Мүмкіндігінше есептің әртүрлі вариантарын шешуге жағдай туғызу.
3. Белгілі бір заңдылық бойынша өзінің қолданылу аймағында, міндетті түрде есептеу нәтижесіне жету.

Есепті шешу үшін оның алгоритмін жазып алу қажет. Есеп алгоритмін арнаулы символдар немесе өзімізге белгілі математикалық формулалар арқылы сипаттап жазуға болады. Сонымен, алгоритм құру - есептің бастапқы берілген күйінен ақырғы нәтижесіне дейінгі орындалатын амалдар тізбегін немесе сипатын жазу. Есепті шешу үшін алгоритмді жеке тексеріп, зерттеу - ең жауапты іс. Амалдар сипатын оқырман қауымға түсінікті жеткізу үшін, алгоритмнің көрнекті блок-схемасын жасаған жөн.

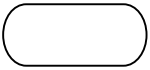
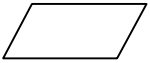
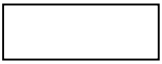
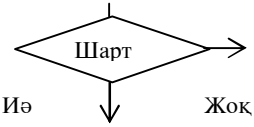
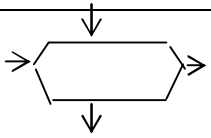
Блок-схема - *алгоритмнің орындалуын ұйымдастыру үшін қолданылатын амалдар тізбегінің графиктік кескіні.* Ол белгілі бір геометрикалық бейнелердің көмегімен жасалады және бұл бейнелерге алдын ала келісілген шартқа байланысты өзіндік мағыналар беріледі. Мысалы, тік төрт бұрыш – арифметикалық амалдарды орындаушы, ал ромб - кез келген шартты

тексеру немесе салыстыру процесін орындаушы блок, т.с.с. Бұл бейнелер алгоритмнің мазмұнына сай өзара бағыттауыш арқылы жалғасады. Әрбір бейне ішінде орындалатын амалдар көрсетіліп, олар - *амалдар блогы* деп аталады.

1.2.2. Алгоритмді бейнелеу блоктары

Блок-схеманың белгілері төмендегі кестеде берілген (1-кесте). Сонымен, бағдарламаны қалыптастырмас бұрын әуелі есептеу алгоритмін, сонан соң есептеу алгоритмнің блок-схемасын көрнекті етіп жасап алған дұрыс.

1-кесте

Іс-әрекеттің аты	**Блок бейнесі**	**Атқаратын қызметі**
Басы, соңы		Алгоритмнің басы, соңы
Енгізу		Айнымалыларды енгізу
Процесс		Есептеу амалдарының жиыны
Таңдау		Шарт тексеру
Модификация		Циклдің басы
Құжат		Қорытындылау, нәтижені баспаға шығару
Қосалқы бағдарлама		Бағынышты бағдарламаларды шақыру
Түсініктеме		Анықтама, ішкі бағдарламалардағы формулалардың түсініктемесі

Бағдарлама дегеніміз - алгоритмнің ЭЕМ-ға түсінікті тілде жазылуы. Бағдарламаны ЭЕМ тілінде жазылған бір мәнді талданатын, осы алгоритмді жүзеге асырушы жеке сөйлемдер тізбегі деп қарауымыз керек. Бағдарламаның сәтті құрылып шығуы - алгоритмнің жасалуына да тікелей байланысты. Ойластырып жасалған алгоритмге сай құрылған бағдарлама тиімді, әрі сәтті шығуы тиіс.

Есепті ЭЕМ-да шешу дегеніміз - берілген ақпаратты белгілі бір ережелереге, нұсқауларға сүйеніп, құрылған бағдарлама бойынша өңдеу.

Сонымен, әуелі зерттелетін объектіні, оның ерекшеліктерін, қажетті параметрлерін анықтаймыз. Сонан соң есепті математикалық тұрғыдан сипаттап, оның математикалық моделін құрамыз да, модельді - математикалық әдістерді пайдаланып талдаймыз, зерттейміз, қажетті критерийлерді анықтаймыз. Міне, осы жұмыстар орындалғанда ғана құбылысты сипаттаушы математикалық теңдеулер жүйесі пайда болады. Бұл - есептің математикалық қойылуы. Үшінші сатыда - математикалық есепті шешуге арналған әдістің ЭЕМ-ның нақты түріне сәйкес құрылған алгоритмін жасау. Сонан соң қажет болған жағдайда, алгоритмге сәйкес блок-схема жасалады. Төртінші сатыда - алгоритмге сәйкес бағдарлама жасалады. Жасалған бағдарламаның қызметін бағдарлама құрушы ЭЕМ-ге қарапайым есептеулер жүргізу арқылы тексереді, бағдарлама қызметін қалыптастырады. Міне, осылайша тексерілген бағдарламаны, нақты есептерді ЭЕМ-да автоматты түрде шешуге, пайдалануға болады.

1.2.3. Бағдарламалық тілдердің синтаксисі мен семантикасы

Бағдарламалық тілдердің де табиғи тілдер сияқты өзінің синтаксисі мен семантикасы бар.

Синтаксис. Бағдарламалық тілдің *синтаксисі* деп - алфавиттер бойынша алгоритм қадамдарын құрастыру ережелерін айтады.

Семантика. Бағдарламалық тілдің *семантикасы* деп - құрылымдарының мағынасын түсіндіретін ережелерді айтады.

1.3. Бағдарламалық тілдердің ең қарапайым құрылымдары

Бағдарламалық тілдердің ең қарапайым құрылымдары - *алфавит, тұрақтылар, айнымалылар, массивтер, өрнектер болып табылады*. Енді осыларға қысқаша сипаттама берелік.

1.3.1. Мәндер мен типтер

Алгоритм бойынша өңделетін шамалардың (деректердің) табиғаты әртүрлі (сандар, мәтіндер, логикалық немесе екілік разрядтар, т.с.с.) болып келуі мүмкін. Осыған орай мәндердің табиғатындағы айырмашылықтарды ажырату үшін тип (тегі) ұғымы қолданылады. Барлық мәндер - берілген мәндер және нәтижелер (қорытындылар) болып бөлінеді.

Алгоритмнің атқарылу барысында мәні өзгермейтін шама - *тұрақты,* ал мәні өзгеретін шама - *айнымалы* деп аталады. Математика мен физика курсын оқығанда тұрақты шамалардың бүтін, нақты, комплексті типтері бар екенін білеміз. Ақпараттануда бұлардан басқа, мәні логикалық және символдық тізбекті болып келетін шамалар да кездеседі.

Мәні символдар тізбегінен тұратын тұрақты шамалар – *мәтін шама* деп аталады. Мәтін бір ғана символдан тұратын болса - оны, *литерлік шама* дейді.

"Ақиқат" немесе "Жалған" деген екі мәннің бірін қабылдайтын шаманы - *логикалық шама* деп атайды. Сөйтіп, бағдарламалау тілінде тұрақты шамалардың **бүтін, нақты, комплексті, литерлі, мәтінді, логикалық типтері** кездеседі.

1.3.2. Айнымалылар

Ақпараттануда айнымалы ұғымы, математикадан өзгешелеу. Мысалы, $y = ax^2 + bx + c$ квадрат үш мүшелігінде a,b,c - тұрақты мән қабылдайтын коэффициент деп, ал x пен y - көп мән қабылтайтын айнымалы деп түсінеміз.

Ал бағдарламалау тілінде a,b,c,x,y - әрқайсысын айнымалы шама деп түсінеміз. *Айнымалының символдық белгісі - оның идентификаторы деп аталады.*

Сонымен, әр айнымалы шаманың **аты**, **тұрақты типі** және **айнымалы мәндері** болады. Бағдарламалау тілінде айнымалы **жай айнымалы** және **индексті айнымалы** болып бөлінеді. Жоғарыда біз жай айнымалылыға сипаттама бердік. Индексті айнымалы массив ұғымымен байланысты. Бірдей аталған біртекті шамалар тобы - ***массив*** деп аталады. Топқа кіретін әр шама массивтің ***элементі*** деп аталады. Математикадағы векторлар, анықтауыштар, матрицалар, тізбектер - массивтің аналогтары болып табылады. Массив элементтерін пайдалану үшін, массивтың атын ғана емес, оның топтағы реттік нөмірін де көрсету керек. A(3)- A массивінің үшінші элементі. Элементтің нөмірі оның индексі, ал элементтің индексімен қоса көрсетілген аты - *индексті айнымалы* деп аталады. Бағдарламалау тілдерінде айнымалының индексі өзімен қатар, жақша ішіне алынып жазылады.

Алгоритмдегі әр айнымалының мәнін ЭЕМ өзінің еске сақтау құрылысының бір ұяшығында, ал массив элементтерінің мәндерін, бір-бірімен тіркесіп, қатар орналасқан бірнеше ұяшықтарда сақтайды. Ал идентификатор - ұяшықтың аты болып есептеледі.

1.3.3. Оператор

Оператор – кез келген бағдарламалау тілінің негізгі ұғымдарының бірі. Ал алгоритм қадамындағы іс-әрекетті сипаттауға қолданылып, арнайы ережелермен құрастырылған символдар жиынын - *оператор* деп түсінеміз. Оператор алфавит, тұрақты айнымалы шамалар, өрнектер және арнаулы қызметші сөздер арқылы құрастырылады. Әр алгоритмдік тілдің өзіне тән операторлар жиынтығы болады.

1.3.4. Бағдарлама

Берілген алгоритмнің бағдаламалау тілінде операторлар тізбегі арқылы сипатталған түрін - *бағдарлама* дейміз.

1.3.5. Транслятор

Аудармашы бағдарлама. ЭЕМ - тек екілік жүйеде жазылған ақпаратты ғана өңдей алатындай етіп жасалғандықтан, ол жазылған бағдарламаны бірден атқара алмайды. Алгоритмдік тілде жазылған бағдарлама алдымен екілік жүйеге аударылуы керек. Мұндай аударманы *транслятор* деп аталатын арнаулы бағдарламаның көмегімен ЭЕМ өзі атқарады. Бұл аудармашы бағдарламада алгоритмдік тілдің барлық ережелері мен аудару тәсілдері қарастырылған. Сондықтан, арнайы бағдарлама - бағдарламадағы грамматикалық ережелерді сақтамаудан туған қателерді тауып, ол қателер жайлы хабар беріп отырады. Стандартты функцияларды есептеу үшін жиі кездесетін математикалық формулардың мәндерін есептейтін арнаулы бағдарламалар, транслятордың құрамына кіреді. Олар - *стандартты* немесе *ішкі формулалар* деп аталады

1.3.6. Бағдарламалау тілі

Біз алгоритмді сипаттаудың 2 әдісін қарастырдық: *табиғи тіл* және *схема арқылы*. Алгоритмді сипаттауға қойылатын негізгі талаптар: алгоритмнің көрнекті, ықшамды, түсінікті болуы. Бірақ, күрделі есептердің алгоритмдерін сипаттағанда алгоритмнің көрнектілігі, ықшамдылығы жойылады. Бұл жағдайда алгоритмді схема түрінде сипаттау қолайлы. Схеманың басқа әдістерден артықшылығы - оның көрнектілігінде. Оны схеманы бағдарламалаудың алғашқы кезеңі ретінде пайдалану ыңғайлы. Қарастырылған әдістердің екеуі де, алгоритм атқарушысының адам болуына бағытталған. Егер алгоритмді орындаушы ЭЕМ болса, оны алгоритмдік тілде сипаттау қажет.

Бағдарламалау тілі дегеніміз - алгоритмді автоматты құрылғылар атқаратындай етіп сипаттауға арналған жасанды тіл. Қазіргі кезде дүниежүзінде бірнеше жүздеген бағдарламалау тілдері бар. Олар ЭЕМ-ның ерекшелігін мен шығарылатын есептің саласына байланысты жасалады. Мысалы, Алгол, Фортран, QБейсик тілдері математикалық есеп алгоритмдерін, КОБОЛ – экономикалық есептердің

алгоритмдерін сипаттауға, ал ПЛ/1, ПАСКАЛЬ универсалды тілдер болып есептелінеді.

Аталған тілдер машина құрылғыларынан тәуелсіз, жоғары деңгейлі бағдарламалық тілдер. Бағдарламалық тілдердің түрлері көп болғанымен, олардың ортақ ерекшеліктері де бар. Біріншіден, бағдарламалық тілдердің сөйлемдері олардың мағынасының тұрақты болуын қамтамасыз ететін қатаң ережелермен құрастырылады. Екіншіден, бағдарламалық тілдерде табиғи тіліміздің кейбір сөздері мен математикалық символдары қолданылады. Бұл бағдарламалық тілдің конструкцияларын түсінуді жеңілдетеді. Бұдан былай біз QБейсик бағдарламалау ортасында жұмыс жасаймыз. Оны жұмысқа қосу тәртібі мен менюі қосымшада сипатталған.

1.3.7. Бейсик тілінің таңбалары

Бейсик тілінде латын алфавитінің 26 бас әріпі, сандары (0-ден 9-ға дейін) мен арнайы белгілер қолданылады. Ал орыс немесе қазақ, яғни ұлттық алфавит әріптері тек түсініктеме ретінде немесе қос тырнақшаның ішіндегі сөз тіркесі түрінде ғана кездеседі. Ұлттық алфавит әріптерімен айнымалыны, функцияны, шама атауларын белгілеуге рұқсат етілмейді.

Енді Бейсик тілінің арнайы белгілерін қарастырайық:

Айыру белгілері;

а) _ Бос орын;
ә) . нүкте;
б) , үтір;
в) ; нүктелі үтір;
г) ' апостроф "тырнақша";
ғ) (ашу жақшасы;
д)) жабу жақшасы.

Арифметикалық амалдар:

а) + қосу немесе сөз тіркесін біріктіру;
ә) - алу;
б) * көбейту;
в) / бөлу;
г) ^ дәрежелеу.

Қатынас таңбалары:
а) < кіші;
ә) > үлкен;
б) <= кіші немесе тең;
в) >= үлкен немесе тең;
г) = тең;
ғ) <> тең емес.

Қосалқы белгілер:
а) _ астын сызу;
ә) ? сұрақ белгісі;
б) & логикалық "және" амалының белгісі;
в) $ коммерциялық белгі.

Айнымалылар түрлерінің таңбалары:
а) $ ақша бірлігі;
ә) % процент.

1.3.8. Бейсик тіліндегі сандардың жазылуы

Бейсик тілінде тұрақты сандар жиі қолданылады. Тұрақты сандар - *бүтін* және *нақты* болып екіге бөлінеді. Бүтін сандар - *оң* және *теріс* таңбалы болады, мысалы: -5, +100, 25, -15,0.

Нақты сандар деп – бүтін және бөлшек бөлігі бар сандарды айтамыз. Олар - *табиғи* және *экспоненциалды* болып бөлінеді. Табиғи нақты санның бүтін және бөлшек бөлігі нүктемен бөлінеді, мысалы; -0.5, +20.25, 30.45, -130.5, 125.0

Егер санның бүтін бөлігі нөлге тең болса, оны жазбауға болады. Экспоненциалды нақты сан табиғи нақты санды 10-ның дәрежесіне көбейткен күйде жазылады. Бұл тұрғыда өте үлкен немесе өте кіші сандар кескінділеді. Ондық дәреженің орнына Е таңбасы жазылып, көбейту таңбасы қойылмайды, мысалы:

10^5 орнына 1.35E+05;
0.25 10^−9 орнына 25 E-09.
Көрсеткіші тек екі таңбамен жазылады.

1.3.9. Тұрақты шамалардың жазылуы

Тұрақты деп – бағдарламаның орындалу барысында өз мәнін өзгертпейтін шамаларды айтамыз. Олар тұрақты сандар және тұрақты сөз тіркестері болып екіге бөлінеді. Тұрақты сандар жоғарыда айтылғандай, *бүтін* және *нақты* сандардан тұрады.

Тұрақты сөз тіркестері немесе символды тұрақтылар, Бейсик тілінде қос тырнақшаға алынған әртүрлі таңбалар жиынан тұрады. Олар мәтіндік ақпаратты сипаттау үшін қажет. Символды тұрақтылардың ұзындығы 255 таңбадан аспауы керек.

1.3.10. Бейсик тіліндегі айнымалылар

Ақпаратты өңдеу барысында әртүрлі мәнді қабылдайтын шамаларды - айнымалылар деп атаймыз. Айнымалыларды белгілеу үшін әртүрлі атаулар – индентификаторлар қолданылады. Индентификатор ретінде латын әріптерінен басталатын сөз тізбегі алынады, ең соңғы таңбасы болып, арнайы белгілер (%,$) тұруы мүмкін. Ол белгі айнымалының түрін көрсетеді. Индентификатор ұзындығы 8 таңбадан аспай және Бейсик тіліндегі түйінді сөздерден өзгеше болуы керек.

Сонымен айнымалылар - *бүтін*, *нақты* және *символды* болып бөлінеді. Егер индентификатор % белгісімен аяқталса, ол бүтін мәнді айнымалы; $ белгісімен аяқталса, ол қос тырнақшаға алынған сөз тіркестерінен тұратын символды айнымалы болғаны. Егер индентификатордың соңында арнайы белгі тұрмаса, ол нақты айнымалы болып есептелінеді.

Мысалы:

ALMA – нақты мән қабылдайтын айнымалы;

B10% - бүтін мән қабылдайтын айнымалы;

CUM$ - мән ретінде сөз тіркесін қабылдайтын символды айнымалы.

1.3.11. Массивтер

Бірыңғай шамалар тізбегінен тұратын, бір индентификатормен (атаумен) белгіленген айнымалылар тобын - ***массив***

(жиын) деп атаймыз. Массивтің түрлері де ***бүтін, нақты, символды*** болады. Массивтер бір немесе бірнеше индексті болады да, индексі жай жақшаға алынып, үтірмен бөлініп жазылуы тиіс.

Мысалы: а − A(10); в − B(I); с − C(2,5)

Массивтің ең үлкен индексінің мәні алынып DIM − операторы арқылы бағдарламаның басында сипатталуы керек, ол ЭЕМ жадында массив элементтері үшін орын сақтайды.

Мысалы:

```
10 DIM A(10),B(20)
20 DIM C(2,5)
```

1.3.12. Стандартты функциялардың жазылуы

Математикада жиі қолданатын функциялар, Бейсик тілінде стандартты функциялар түрінде жазылады (2-кесте). Ол үш латын әріпімен белгіленіп, аргументі жақшаға алынады.

Стандартты функциялардың Бейсик тілінде жазылуы

2-кесте

Математикалық жазылуы	Бейсик тілінде жазылуы
sin x	SIN(X)
cos x	COS(X)
tg x	TAN(X)
arctg x	ATN(X)
ex	EXP(X)
ln x	LOG(X)
|x|	ABS(X)
$\sqrt{x}$	SQR(X)
X санының бүтін бөлігі	INT(X)
X санын ең жақын бүтін санға дейін жеткізіп дөңгелектеу	CINT(X)
0-ден 1-ге дейінгі псевдо − кездейсоқ сан	RND(X)

Басқа функцияларды (arcsinx, arccosx, ctgx, т.б.) жоғарыда көрсетілген функциялар арқылы жазады, мысалы:

ctgx=1/tgx т.б.

$$\log_a b = \frac{\ln b}{\ln a} \qquad \sqrt[n]{x} = x^{\frac{1}{n}}$$

1.3.13. Арифметикалық өрнектер

Бейсик бағдарламасында жазылатын өрнектер кәдімгі математика формуларының жазылуына ұқсас келеді, бірақ өрнектің символдары бір қатарда жазылуы тиіс, яғни бөлшектің алымы мен бөлімі, дәреже көрсеткіштері мен индекстері бір жолда жазылады. Амалдардың орындалу реттігі төмендегідей:

- жақша ішіндегі өрнек есептелінеді;
- функциялардың мәндері есептелінеді;
- дәрежелеу амалдары атқарылады;
- көбейту және бөлу амалдары атқарылады;
- қосу және алу амалдары атқарылады.

Орындалу реті бірдей амалдар, солдан оңға қарай орындалады.

Мысалы:

Математикалық жазылуы	Бейсик тілінде жазылуы
Ax^2+bx+c	a*x^2+b*x+c
$\sin x^2+x$	SIN(X^2)+SQR(X)
$e^x+\ln x^2$	EXP(X)+LOG(X^2)
$\sqrt[3]{x^2}-ax$	(X)^(2/3)-a*x

Бағдарлама операторлар тізбегінен тұрады. Оператор дегеніміз - ақпаратты өңдеуге арналған нұсқаулар. Нұсқаулар үлкен латын әріптерімен жазылған сөз тіркестерінен тұрады да, осы тілдің ең қарапайым сөйлемі болып есептелінеді. Сонымен, бағдарлама қатарлар тізбегінен тұрады. Әр қатардың реттік нөмірі және ақпараттық бөлігі болады. Қатар нөмірі 10, 20, 30, 40.... тәрізде бүтін сандардан тұрады, ол оператордың орындалу ретін көрсетеді.

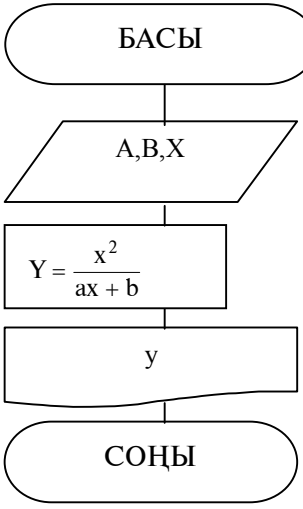
Нөмірлерді ондық сандар арқылы нөмірлеу – ұмыт кеткен қатарларды енгізуге мүмкіндік береді. Ақпараттық бөлік бір немесе бірнеше операторлан тұрады. Қатарда бірнеше оператор орналасқан болса, олар қос нүктемен (:) ажыратылып жазылады. Әрбір қатар апострофпен бөлінген қатардың соңында жазылған түсініктемемен аяқталуы мүмкін.

Бейсик тілі ағылшын тілінің түйінді және қызметші сөздерінен тұрады. Олардың мағынасы - оператордың не командының іс-әрекетін сипаттайды.

Бағдарламаны орындау үшін операторлан бөлек командалар да қолданылады. Командалар - *жүйелік* (системдік) немесе *бұйрық* (директива) *командалар* болып бөлінеді. Операторлар нөмірленген белгісі бойынша орындалса, бұйрық - нөмірсіз теріліп, бірден орындалуы тиіс.

Мысалы: x=2.5, ал a=3.5 және b=3.5 болғандағы $y = \frac{x^2}{ax + b}$ функциясының бағдарламасын жазу керек болсын.

Ең алғаш бұл есептің блок-схемасын құрайық (1.3-сурет).



1.3-сурет

1 қадам: A,B,X айнымалылардың мәнін енгіземіз;
2 қадам: У – функциясының мәнін табамыз;
3 қадам: У – мәнін қорытындылап, баспаға аламыз.
4 қадам: соңы;
Бағдарламасы төмендегідей:

```
10 A=0.5: B=3.5: X=2.5
20 Y=X^2/(A*X+B)
30 PRINT "Y=";Y
40 END
```

Орындау бұйрығы берілген соң У функциясы есептелініп, оның нәтижесі экранға шығады. Бағдарламаның әрбір жолын пернелер тақтасынан теріп алғаннан кейін міндетті түрде **"енгізу"** пернесін басу керек, егер ол перне басылмаса, терілеген таңбалар ЭЕМ − жадына енгізілмейді.

1.3.14. Тұрақты сөз тіркестері

Тұрақты сөз тіркестері немесе символды тұрақтылар Бейсик тілінде екі қос тырнақшаға алынған әртүрлі таңбалар жиынынан тұрады. Олар мәтіндік ақпаратты сипаттау үшін қажет. Символды тұрақтылар 255 таңбадан аспауы керек.

Индентификатор ретінде латын әріптерінен басталатын сөз тізбегі алынады, ең соңғы таңбасы болып, арнайы белгілер (%,$) тұруы мүмкін. Ол белгі айнымалының түрін көрсетеді. Индентификатор ұзындығы 8 таңбадан аспауы керек және Бейсик тіліндегі түйінді сөздерден өзгеше болуы керек.

Сонымен айнымалылар − *бүтін*, *нақты* және *символды* болып бөлінеді. Егер индентификатор % белгісімен аяқталса, ол бүтін мәнді айнымалы; $ белгісімен аяқталса, ол қос тырнақшаға алынған сөз тіркестерінен тұратын символды айнымалы болғаны. Егер индентификатордың соңында арнайы белгі тұрмаса, ол нақты айнымалы болып есептелінеді.

Мысалы:

A$=”Алма” - A$ мәні қабылдайтын тұрақты;

B$=”ты” - B$ мәні қабылдайтын тұрақты;

C$=A$+B$ - мән ретінде “Алматы”сөз тіркесін қабылдайтын символды айнымалы.

1.4. Енгізу, қорытындылау операторлары

Жалпы операторлар - *атқарылатын* және *атқарылмайтын* болып екіге бөлінеді. Атқарылмайтын операторлар бағдарламада қажет мәліметтерді береді де, өздері оның орындалуына қатыспайды, оларға **REM**, **DIM, DATA** операторлары жатады.

Атқарылатын операторлар - бағдарламада белгілі бір іс-әрекет атқарады. Бұларға - *меншіктеу*, *көшу*, *қайталану*, *енгізу*, *қорытындылау* операторлары жатады.

***Енгізу операторлары*. INPUT, READ, DATA**

INPUT – енгізу операторы, ол бағдарламаға сәйкес ЭЕМ – ның сұрауы бойынша пернелер тақтасынан деректерді енгізу қызметін атқарады. Бұл оператор, бағдарлама RUN арқылы атқарылуға берілген соң, экрандағы сұрау белгісінен басталады. Сұрау белгісі алдында қандай айнымалылардың мәнін енгізу керектігі жайлы түсініктеме болуы мүмкін (егер ол түсініктеме бағдарламада жазылса).

INPUT операторының жазылу үлгісі:

nc INPUT [;] ["түсініктеме";] айнымалы [,айнымалы]

түсініктеме – экранға түсініктеме беретін сөз тіркесі, оны қазақша – орысша жазуға болады. **Айнымалы** – мәні пернелер тақтасынан енгізілетін айнымалылар тізбегі, олар үтір арқылы бөлініп теріледі. Сонымен, пернелер тақтасынан енгізілетін мәндер INPUT операторындағы айнымалыларға меншіктеледі. Енгізілген мәндер айнымалылар санына сәйкес болуы керек.

Мысалы:

```
10 INPUT A
20 INPUT A,B,C
```

Экранда сұрау белгісі пайда болғаннан кейін 10-шы оператор бойынша А айнымалының мәнін және 20 оператор бойынша А, В,С мәндерін үтір арқылы енгізуді күтіп тұрады.

Ескерту: Егер пернелер тақтасынан енгізілген мәндер саны тізімдегі айнымалыларға тең болмаса немесе айнымалылар мен мәндердің типі сәйкес болмаса, онда экранға: "? REDO FROM START"" – енгізуді қайталаймыз деген хабар шығады. Бұл жағдайда айнымалылардың мәндерін басынан бастап қайтадан енгізу қажет.

READ және DATA – операторлары қосалқы қолданылады. READ операторы DATA операторының мәндер тізбегіндегі сандарды оқып және сол мәндерді өзінің тізімінде тұрған айнымалыларға меншіктеу қызметін атқарады. Жазылу үлгісі:

nc READ айнымалы [,айнымалы…]

мұндағы **айнымалы** − айнымалының немесе массив элементтерінің атауы.

READ − операторындағы айнымалылар мен олар меншіктейтін DATA операторындағы сәйкес тұрақты мәндердің түрлері бірдей болуы тиіс.

DATA операторы - орындалмайтын оператор, ол бағдарламаның кез келген жерінде орналасады және бағдарламаның орындалуына еш әсерін тигізбейді. DATA операторы бастапқы мәндерді белгілі ретпен қарапайым айнымалыларға немесе массив элементтеріне қабылдату үшін қолданылады. Жазылу үлгісі:

nc DATA тұрақты шама [,тұрақты шама…]

мұндағы тұрақты шама − кез келген тұрақты сан немесе символды тұрақты.

Мысалы:

```
20 DATA 5,1.5,-7.5,2.5,-2
```

Бұл қатар сандардан тұратын мәндер блогын құрайды. Егер бағдарламада бірнеше READ операторы болса, онда кезекті READ операторы DATA тізіміндегі тұрақты мәндерді басынан бастап меншіктемей, өзінің алдындағы READ операторынан қалған мәндерден бастап меншіктейді.

Мысалы:

```
10 DATA 3, -5, 6, -0.5, 6E-8
20 READ A,B
30 X=X+B
40 READ C,D
50 Y=C^2-D
60 READ E
```

Бұл операторлар орындалған кезде айнымалылардың мәндері төмендегідей болады: A=3;B=-5; C=6; D=-0.5; E=6*10^-8

Ескерту: егер DATA операторындағы айнымалылар мәнінің саны READ операторына қажетті деректер санынан кем болса, онда "OUT OF DATA" − "деректер жоқ" деген хабар беріледі.

PRINT − баспаға мәлімет шығару операторы.

PRINT – операторы өзінің тізімінде тұрған айнымалылар мен, өрнектердің мәндерін экранға (принтерге) шығару үшін қолданылады. Жазылу үлгісі:

nc PRINT [тізім][,] немесе ?[тізім] [;]

мұндағы тізім – өрнектердің, айнымалылардың, тұрақтылардың үтірмен, бос орынмен, нүктелі үтірмен ажыратылған тізімі. Символды тұрақтылар қос тырнақшаға "Р=" алынады. Егер оператор тек PRINT сөзінен тұрса, онда экранға бос жол шығады. PRINT операторының орнына **"?"** – белгісін қоюға болады. PRINT тізіміндегі әрбір элементтің экранға шығу реті сол элементтің алдында тұрған тыныс белгісіне байланысты.

Егер PRINT операторының тізіміндегі элементінен кейін үтір тұрса, оған кейінгі элемент келесі зонадан бастап экранға шығады, ол дегеніміз - экрандағы әрбір қатар жалғаса орналасқан 14 орыннан тұратын зоналарға бөлінген және экранның әр қатарында 5 зона бар деп есептелінеді. Яғни, олар 1 – зона 1-14; 2 – зона (15-28); 3 – зона (29-42); 4 – зона (43-56); 5 – зона (57-80) орындарда орналасқан. Ал тыныс белгісі нүктелі үтір немесе бос орынмен берілсе, онда элемент тізім алдындағы элементіне жалғаса, экранға шығады. Олардың аралары бір немесе екі бос орынмен бөлінеді.

Мысалы:

Оператор	Нәтиже
1. PRINT 2.5,3.5	2.5 3.5
2. PRINT "A=0.5";"B=3"	A=0.5 B=3
3. A=0.5: B=2.5: C=2.0 PRINT A; A+B,C	0.5 3.0 2.0

Нәтижені қағазға шығару үшін PRINT операторының алдына L әріпін жазып, LPRINT деген операторын қолдансақ жеткілікті. Егер PRINT операторының ең соңына нүктелі үтір немесе үтір қойсақ, онда келесі кездесетін PRINT операторының тізімі де осы қатарға жалғаса экранға, алдыңғы таңбаға байланысты, қатарласып немесе зонаға бөлініп орналасады.

STOP және END –операторлары.

STOP – аялдау операторы. Ол бағдарламаның кез келген жерінде қолданылып – есептерді тоқтату қызметін атқарады. STOP – операторынан кейін бағдарлама жұмысын уақытша тоқтатады, экранға "STOP AT LINE NN" – деген хабар пайда болады. Жұмысты жалғастыру үшін CONT (жалғастыру) бұйрығын орындау керек. END (соңы) – оператор бағдарламаның ең соңын көрсетеді және бұл оператор орындалғаннан кейін, бағдарлама жұмысын аяқтайды.

1.5. Сызықты құрылымды алгоритмдер

Егер бағдарламадағы операторлар жазылу ретімен орындалса, ондай бағдарламаны - сызықты деп атайды. Бұл бағдарламада бір уақытта бір ғана оператор атқарылады және ол - *белсенді оператор* деп аталады.

1.5.1. Меншіктеу операторы

Меншіктеу операторы – бір шаманы немесе белгілі бір өрнектің мәнін есептеп, оны бір айнымалыға меншіктейді.

Оператордың жазылу үлгісі:

[LET] айнымалы = <өрнек>

мұндағы nc – оператор нөмірі; айнымалы – кез келген айнымалы немесе массив элементі; өрнек - өрнек, айнымалы немесе тұрақты шама.

LET – (айталық) оператордың аты, бұл сөзді жазбауға да болады. *Ескерту*: егер оператордың жазылу үлгісінде квадрат жақша ішінде мәлімет жазылса, оны бағдарлама құрушы қалауына байланысты жазбауға да болады.

Меншіктеу операторының орындалу барысында теңдік (меншіктеу) белгісінің оң жағындағы өрнектің сандық мәні есептелінеді де, есептеліп табылған мән - теңдіктің сол жағындағы айнымалыға беріліп, меншіктелінеді. Айнымалы ретінде белгілі бір мағынасы бар сөз тіркестері де қолданылады.

Әр қатардың соңында " ' " (апостроф) белгісінен кейін ұлттық әріптер мен түсініктеме беруге болады.

Меншіктеу операторының мысалдары.

```
10 LET A=10
20 X=A^2
```

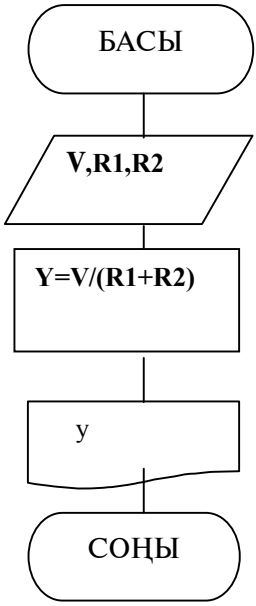
```
30 B(3)=X
40 Y=A*X^2+B*X+C
```

1.5.2. Сызықты бағдарламаны қолдану мысалы

Мысалы тізбектегі ток күшін мына формула бойынша есептеу керек болсын, I=V/(R1+R2). Есептің алгоритмі төмендегідей болады.

1қ. Басы;
2қ. V,R1,R2 − айнымалыларының мәндерін енгізу керек;
3қ. I айнымалысына I=V/(R1+R2) деген мән меншіктеу керек;
4қ. I айнымалының мәнін баспаға алу керек;
5қ. Соңы.

Бағдарламасы:




```
10 PRINT "Тізбектегі ток күшін
анықтау "
20 PRINT "R1,R2,V − мәндерін
енгіземіз"
30 INPUT R1,R2,V
40 LET I=V/(R1+R2)
50 PRINT "I=" ;I
60 END
```

1.5-сурет. Мысалдың блок-схемасы

Немесе бағдарламаны мына түрінде жазуға болады.

```
10 PRINT "тізбектегі ток күшін анықтау"
20 DATA 10, 1.5, 7.2
30 READ V,R1,R2
40 I=V/(R1+R2)
50 PRINT "I=";I
60 END
```

1.6. Тармақталған алгоритмдер

Берілген есепті шығару үшін қолданылатын алгоритм қадамдарының жазылуы қатар болғанымен, атқарылу реті қатар бола бермейді. Себебі, алгоритм қадамдарының атқару реті берілген немесе аралық шарттың нәтижесіне байланысты. Берілген шартты тексеру нәтижесінде алгоритмнің келесі орындалатын қадамы анықталады.

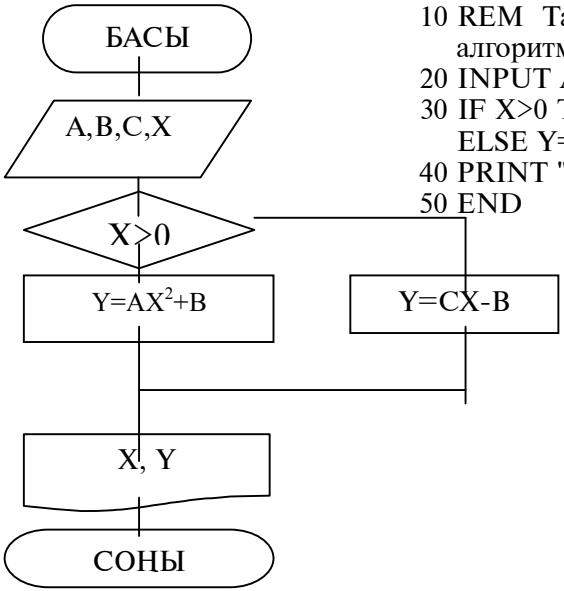
Шартқа байланысты атқарылатын әрекет немесе әрекеттер тобы - тармақ деп, шарт тексеріліп тұрған қадам - таңдау қадамы деп аталады.

Есептің берілуіне байланысты алгоритмде тармақ саны бір немесе бірнешеу болуы мүмкін.

Мысалы: $У=\begin{cases} Ax^2+b & \text{егер } x<0 \\ cx-b & \text{егер } x>0 \end{cases}$

1. Енді есеп шешімін математикалық түрде сипаттайық:
   егер x>0 болса, онда y=cx-b;
   егер x>0 болса, онда $y=ax^2+b$;
2. Алгоритм қадамдарын құрастырайық:

1 қадам. Алгоритм басы;
2 қадам. Енгізу: a,в,c және x мәндерін енгізу;
3 қадам. Шарт қою: егер x>0 болса онда y=cx-b, әйтпесе $y=ax^2+b$;
4 қадам. Қорытындылау: x және y мәнін баспаға алу;
5 қадам. Соңы.




```
10 REM  Тармақталу
   алгоритмі
20 INPUT A,B,C,X
30 IF X>0 THEN Y=C*X-B
   ELSE Y=A*X^2+B
40 PRINT "X=";X,"Y=";Y
50 END
```

1.6-сурет. Мысалдың блок-схемасы

Бұл жерде біз тармақталу құрылымының ең қарапайым түрі, бір тармақтан тұратын мысал қарастырайық. Бұл мысалдағы шарттың орындалу реті: шарт ақиқат болса "иә" жолындағы тармақ, ал жалған болса, "жоқ" жолындағы тармақ орындалады. Сонымен, тармақталу алгоритміне міндетті түрде шартты тексеру блогы кіреді. Ол ромб түрінде кескінделіп, басқа блоктарымен бір кіріс және екі шығыс сызығы арқылы байланысады. Көбінесе тармақталу алгоритмдері екі түрде кездеседі, олар "**аттап өту**" және "**таңдау**" мүмкіндіктерін іске асыруға көмектеседі. Тармақталу алгоритмдерін бағдарламалау үшін **GOTO** – *көшу* және **IF** – *егер* операторларын білуіміз керек.

1.6.1. GOTO шартсыз көшу операторы

GOTO шартсыз көшу операторының жазылу үлгісі:

GOTO m

мұндағы: m – бағдарламадағы кез келген операторлық таңбасы.

Мысалы:

20 X=X+H: N=N+1: GOTO 100

деп алсақ, басқару 100 операторға беріледі бұл жерде GOTO жолдағы соңғы оператор, оны келесі жолда жалғыз жазуға да болады. m – таңбалы оператор GOTO операторының алдында да, соңында да тұруы мүмкін.

1.6.2. Шартты көшу операторы – IF

Тармақталу кезінде шартты тексеру блогы орындалу барысында, алгоритмнің екі мүмкіндігінің тек біреуі таңдап алынып жүзеге асырылады да, ал екіншісі – таңдап алынбаған тармақ, біріктіру нүктесіне дейін орындалмай қалады. Енді осы алгоритмдерді оператор ретінде жазу жолдарын қарастыратын болсақ, шартты тексеру блогы - IF операторы арқылы жазылады. Шартты көшу операторының шартсыз көшу операторынан айырмашылығы - басқа операторға басқаруды, белгілі бір шарттың тексерілу нәтижесіне байланысты береді.

Шартты операторлық қысқа және толық түрлері болады. Оның жазылу түрлері:

1- түрі. IF <шарт> GOTO m [ELSE <оператор>]

2-түрі. IF <шарт> THEN <оператор 1> [(ELSE <оператор2>)]

мұндағы IF – шарт операторы; <шарт> - арифметикалық немесе логикалық өрнек; GOTO – көшу операторы; m – оператор таңбасы; <оператор> - бұл жерде бір оператор немесе қос нүктемен бөлінген операторлар тізбегі; THEN – Бейсик тілінің түйінді сөзі, ол "**онда**" деген сөзді білдіреді, ал ELSE – "**әйтпесе**" деген сөзді білдіреді.

Квадрат жақшаның ішіндегі сөздер қажетсіз болса, жазбауға да болады, онда ол операторлық қысқа түрін білдіреді, әйтпесе операторлық толық түрі көрсетілген деп есептейміз.

Шарт операторының орындалу реті:

Егер шарт ақиқат болса, онда THEN сөзінен кейінгі тұрған оператор орындалып, бірден келесі қатарға көшеміз. Егер шарт орындалмаса, онда ELSE сөзінен кейінгі тұрған оператор орындалып, бірден келесі қатарға көшеміз. Сонымен оператордың қысқа түрі - "аттап өтуді", ал толық түрі - "таңдауды" жүзеге асырады.

Енді нұсқауда қарастырған мысалдың бағдарламасын жазайық. Мысалдағы X айнымалысының таңбасына байланысты жоғарғы не төменгі формуланы таңдап, сол арқылы Y функциясының мәнін табамыз.

Бағдарлама мына түрде жазылады:

```
10 REM  Тармақталу алгоритмі
20 INPUT A,B,C
30 INPUT "X=";X
40 IF X<0 THEN  60
50 Y=C*X-B: GOTO 70
60 Y=A*X^2+B
70 PRINT "X=";X,"Y=";Y
80 END
```

Енді осы бағдарламаның ықшамдалған түрін келтірейік:

```
10 REM  Тармақталу алгоритмі
20 INPUT A,B,C,X
30 IF X>0 THEN Y=C*X-B: GOTO 70
40 Y=A*X^2+B
70 PRINT "X=";X,"Y=";Y
80 END
```

Енді осы бағдарламада IF − шарт операторының толық түрін келтірейік:

```
10 REM  Тармақталу алгоритмі
20 INPUT A,B,C,X
30 IF X>0 THEN Y=C*X-B ELSE Y=A*X^2+B
70 PRINT "X=";X,"Y=";Y
80 END
```

1.7. Циклді алгоритмдер

Көптеген есептерді шығару әдісіне байланысты алгоритмдердің кейбір қадамдарын бірнеше рет қайталап атқаруға тура келеді. Сондықтан, қайталанып орындалатын қадамдарды сипаттау үшін цикл құрылымы қолданылады.

Цикл құрылымы – цикл денесі, параметрі, параметрдің алғашқы және соңғы мәндері, қадамы деген ұғымда тығыз байланысты.

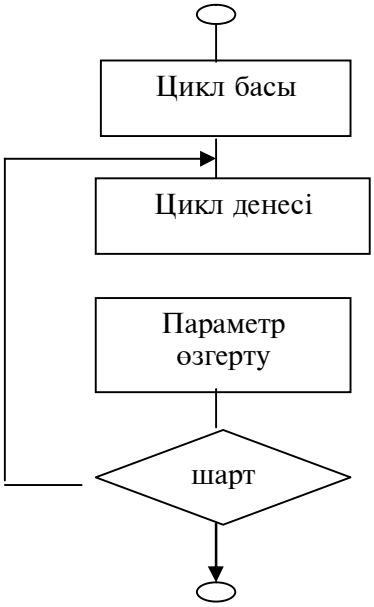
Цикл денесі деп – есептің алгоритміне байланысты кез келген нәтижеге жету үшін қайталанып, орындалуға тиісті қадамдарының жиынын айтамыз. Цикл денесінің қайталанып атқаруын ұйымдастыруға қатысушы айнымалыны - басқарушы айнымалы немесе *циклдің параметрі* дейміз. Циклдік алгоритмдерді пайдалану – бағдарламаны цикл операторы түрінде қысқартып жазу мүмкіндігін береді. Циклдің қайталану саны - *белгілі* және *белгісіз* болып екі топқа бөлінеді. Қайталану саны алдын ала белгілі болып келетін циклді - *арифметикалық цикл* деп, ал орындалу саны белгісізі - *итерациялық цикл* деп есептелінеді.

Қайталану саны алдын ала белгілі циклдерді ұйымдастырудың үш түрі бар. Олар: 1 – *"Дейін"*, 2 – *" Әзір"*, 3 – *"Параметрлі цикл"*.

"*Дейін*" және *"Әзір"* циклдерін Бейсикте IF және GOTO операторлары арқылы жазуға болады, ал "Параметрлі циклді" сипаттау үшін, арнайы FOR - *үшін* және NEXT – *келесі* операторлар қолданылады.

1.7.1. "Дейін" үлгісінің құрамы

Мына блок-схемада (1.7-сурет) "Дейін" үлгісі көрсетілген. Ол төрт бөліктен тұрады. Циклдің басы немесе циклді дайындау бөліміндегі параметрге оның алғашқы мәні беріледі. Бұл бөлімде параметрден басқа да айнымалыларға мәндер берілуі мүмкін. Цикл денесі есептің алгоритміне байланысты қайталанып, орындалуға тиісті амалдар енеді.



Үшінші бөлімде - параметр белгілі бір шамаға өзгереді. Сөйтіп, цикл денесінің қайталанып орындалуына дайындық жүргізіледі.

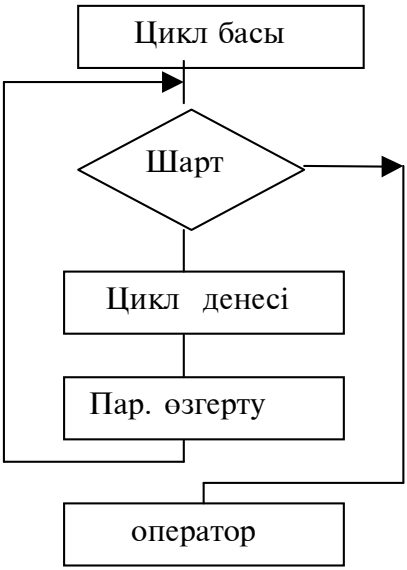
1.7.1-сурет. Есептің блок-схемасы

Төртінші бөлімде - параметрдің ағымдағы мәніне байланысты шартты тексеру жолымен цикл денесін қайталау немесе қайталамау керектігі анықталады.

"*Дейін"* үлгісінің ерекшелігі - циклді аяқтау шарты цикл денесінен кейін тексерілгендіктен, цикл денесі ең болмаса бір рет қайталанады.

1.7.2. "Әзір" үлгісінің құрамы

"Әзір"циклінің құрамы "Дейін" циклінің құрамындай, тек бөлімдердің орналасу реті өзгеше. Циклден шығу шарты бірінші тексергенде орындалмаса, цикл денесі бірде-бір рет атқарылмай қалуы мүмкін. "Әзір" циклінде параметрді өзгерткеннен кейін, басқаруды шарт тексеру бөліміне беретін қосымша оператор енгізу керек.



1.7.2-сурет. Есептің блок-схемасы

Сонымен, цикл құру үшін алдымен оның айнымалы аргументіне, яғни параметріне алғашқы мәнін меншіктеу керек. Сонан соң әрбір қайталану кезеңіне цикл параметрі белгілі бір адыммен (қадамға) өзгере отырып, ол алдын ала берілген параметрдің ең соңғы мәніне жетуі қажет. Мысалы, параметрді x деп алып, оның алғашқы мәнін X_A, соңғы мәні X_C және қадамы h - қа тең десек, онда цикл бойынша X төмендегідей өзгереді.

$$X_A,\ X_{A+h},\ X_{A+2h},\ \ldots\ldots\ldots\ldots,\ X_C$$

Циклдің қайталану санын n − деп белгілесек, оны мына формула арқылы есептеуге болады:

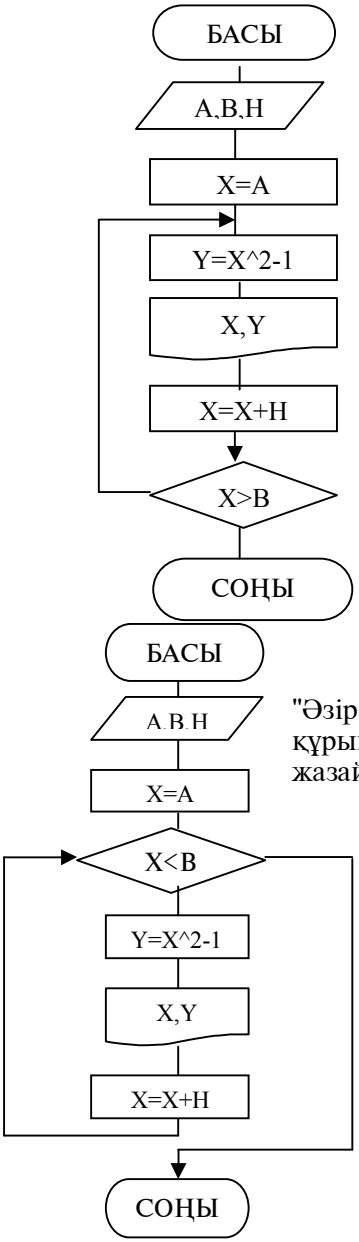
$$n =] X_C - X_A / h [-1,$$

мұндағы n − тек бүтін сан, егер ол бөлшек сан болса, оның тек бүтін бөлігі алынады. Циклдің "*Дейін*" және *"Әзір*" үлгісіне бір мысал қарастырайық. Мысалы: Бір аргументті функция мәндерін есептеген бағдарлама жазу керек болсын:

Y=X^2-1,

мұндағы A − X-тың алғашқы, B − соңғы мәндері, h - өзгеру қадамы, яғни X өзгеру аймағы A<X<B.

"Дейін" үлгісі бойынша блок-схемасын құрып (1.7.3-сурет) бағдарламасын жазайық.



Бағдарламасы:

```
10 A=-1:B=10:H=0.5
 20  X=A
 30 Y=X^2-1
 40 PRINT "X=";X,"Y=";Y
 50 X=X+H
 60 IF X<=B GOTO 30
 70 END
```

1.7.3-сурет. Есептің блок схемасы

"Әзір" үлгісі бойынша блок-схемасын құрып (1.7.4-сурет), бағдарламасын жазайық.

Бағдарламасы:

```
10   A=-1:B=10:H=1
20   X=A
  30   IF X>=B GOTO 70
40  Y=X^2-1
50  PRINT "X=";X, "Y=";Y
 60  X=X+H :GOTO 30
 70  END
```

1.7.4-сурет. Есептің блок-схемасы

1.7.3. " Параметрлі цикл " үлгісінің құрамы

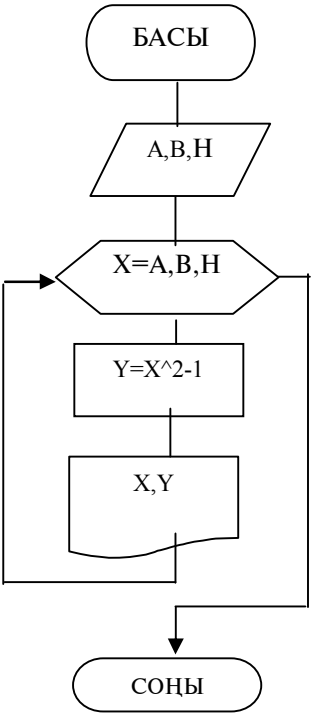
Цикл құрылымының блок схемасын салып және бағдарламасын жазуды жеңілдету үшін, алгоритмдердің ықшамдалған түрі - "*Модификатор*" немесе "*Параметрлі цикл*" блогын қолданамыз. Ол алты бұрыш тәрізді геометриялық фигурадан тұрады және оның міндетті түрде екі кіріс және екі шығыс сызығы болуы тиіс. Сонымен, циклді ықшамдап жазу үшін параметрлі циклдің FOR "үшін" және NEXT − "келесі" операторларын қолданамыз.

Циклдің басында FOR операторы, одан кейін қайталанатын операторлар тізбегі, ал циклдің соңында NEXT операторы жазылады. Параметрлі цикл операторының жазылу үлгісі:

nc FOR X=M TO N STEP K
цикл денесі
NEXT X

мұндағы X − цикл параметрі, айнымалы;
M,N − цикл параметрінің алғашқы және соңғы мәндері;
K − цикл параметрінің өзгеру қадамы;
FOR - үшін; NEXT −келесі; TO - дейін;
STEP - қадам деген ұғымдар.

Сонымен, FOR, NEXT, TO, STEP, NEXT − параметрлі цикл операторының өзекті сөздері болып табылады. Егер қадам бірге тең болса, онда оператор соңындағы STEP сөзі мен K қадамын жазбауға да болады. "*Дейін*" және *"Әзір*" үлгілерінде қарастырған мысалды алып, оны параметрлі цикл операторымен сипаттап жазайық. Оның блок-схемасы мен бағдарламасы төмендегідей.



Бағдарламасы:

```
10  A=-1:B=10:H=1
20 FOR X=A TO B STEP H
30  Y=X^2-1
40 PRINT "X=";X,"Y=";Y
50 NEXT X
 60  END
```

1.7.5-сурет.Есептің блок схемасы

Сонымен, FOR және NEXT операторлары тек қосалқы тізбекте қолданылады. FOR - цикл басы, ал NEXT – циклдің соңы. FOR және NEXT операторындағы айнымалы (X) сәйкес болуы керек.

1.8. Массив элементтерінің алгоритмдері

Массив элементтерінің қарапайым алгоритмдерін сипаттау.

ЭЕМ-ның жадында сандардың бір-бірімен байланысқан көптеген мәндерін сақтау үшін *индексті айнымалылар*, яғни *массивтер* қолданылады. Сонымен, *бірдей аталған (индентификаторлары бірдей) біртекті шамалардың тобы* -

массив деп аталады. Топқа кірген әр шаманы - *массив элементі* деп атайды. Массивтің басына қатысты алынған әр элементтің реттік нөмірі - оның *индексі* деп аталады. Массив элементінің индексімен қоса көрсетілген атын – *индексті айнымалы* дейді. Индексті айнымалының бір немесе бірнеше индекстері, сандармен немесе әріптермен жақша ішінде жазылады, мысалы: A(5), C(J), T(K). Екі, үш өлшемді массивтердің индекстері жақшаның ішінде үтірмен бөлініп жазылады. Мысалы: A(5,4), C(I,J), ALMA(N,M), т.с.с. Сонымен, массивтер бір өлшемді, екі өлшемді, т.б. болып ажыратылады. Массив элементтерімен түрлі операцияларды атқаруды - *массив элементтерін өңдеу* дейміз. Оны сипаттауға жай циклдер, тармақталған, қабаттасқан циклдер құрылымы қолданылады.

Бейсик тілінде жазылған бағдарламада жай айнымалы мен массивтердің атаулары бірдей болуы мүмкін. Екі өлшемді массивтің элементтері ЭЕМ-ның жадында жатық жолдар бойынша орналасады. Жадта орын сақтау үшін массивтің ең үлкен өлшемін көрсету қажет. Массив өлшемін сипаттау – арнаулы DIM - өзекті сөзі ағылшынның DIMENTION - өлшем деген сөзінен шыққан, егер есептеу барысында A1, A2 ,... , A20 массив элементтері кездессе, онда оны бағдарлама басында DIM A(20) түрінде жазып сипаттаймыз. Мұндағы: A – массивтің аты, ал 20 – оның индексінің ең үлкен қабылдайтын мәні.

Егер DIM C(10,15) деп алсақ, онда бұл екі өлшемді массив, яғни матрица түрінде қабылданады, бұл 10 қатары, 15 бағанасы бар C әрпімімен белгіленген матрицаның сипатталуы.

Индекстің ең кіші мәні нөлге тең деп есептелінеді. Индекстің ең кіші мәнін бірге теңеу үшін DIM операторларынан бұрын:

OPTION BASE 1 деген операторды жазуымыз керек.

Егер массив элементі алдын ала сипатталмай, оның элементін бағдарламада қолданатын болса, онда ол массив айқын сипатталмаған, бір өлшемді, 10 элементі бар деп есептелінеді. Бір DIM операторлармен бірнеше массивтерді сипаттауға да болады, мысалы:

10 DIM A(10), C(5,4), K(20)

мұндағы A, C, K − массив аттары, ал 10, 5, 4, 20 − олардың индекстерінің ең үлкен мәндері.

1.8.1. Массивтерді енгізу - қорытындылау

Массив элементтерін енгізу − INPUT, READ, DATA операторларын қолдану арқылы орындалады. READ,DATA операторларын қолданып, массив элементтерін еңгізуді екі әдіспен орындауға болады.

1-әдіс. Индексті айнымалылар енгізу тізімінде теріліп беріледі.

Мысалы:

```
10 DIM X(5)
20 DATA 1,20,15,40,50,30
30 READ X(0), X(1), X(2), X(3), X(4), X(5)
```

индексті айнымалылардың мәндері жай айнымалылардың мәндерін меншіктегендей беріледі. Бұл әдіс аз мөлшерлі массивтерді өңдеу үшін қолайлы.

2-әдіс. Үлкен мөлшерлі массивтерді өңдеу үшін енгізу - қорытындылау операторын, цикл денесінің ішіне орналастырған қолайлы. Цикл орындалған сайын, массивтің бір элементтінің мәні енгізіліп - қорытындыланады.

```
10 DIM X(5)
20 DATA 1,2,3,4,5
30 FOR I=1 TO 5
40 READ X(I)
50 NEXT I
60 FOR I=1 TO 5
70 PRINT X(I)
80 NEXT I
```

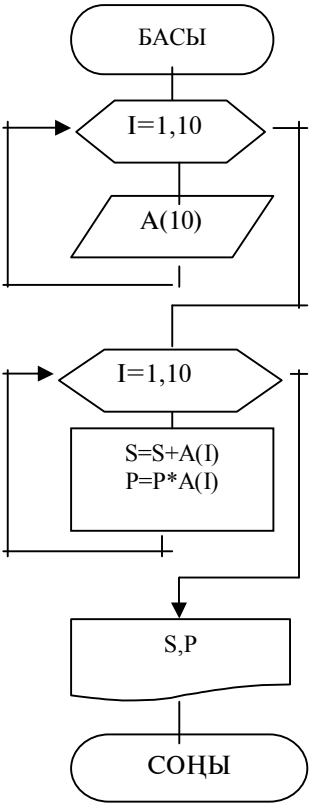
Сонымен қатар INPUT операторын қолданып, массив элементтерін енгізетін болсақ, онда бағдарламада оны төмендегідей сипаттауға болады:

```
10 DIM X(10)
20 FOR I=0 TO 9
30 INPUT X(I)
```

40 NEXT I

Енді массив элементтерін өңдеуді, бірнеше есептерді шығара отырып қарастырайық:

1-есеп. Берілгені: A(10) массиві, A(10) массивінің қосындысы мен көбейтіндісін табу. Ол үшін қосындыны S деп алып, оның алғашқы мәнін нөлге теңейміз, ал көбейтіндісін P деп алып, алғашқы мәнін 1–ге теңейміз. Қосындысын S=S+A(I), көбейтіндісін P=P*A(I) өрнектер бойынша табамыз. Есептің блок-схемасын сызамыз (1.8.1-сурет).

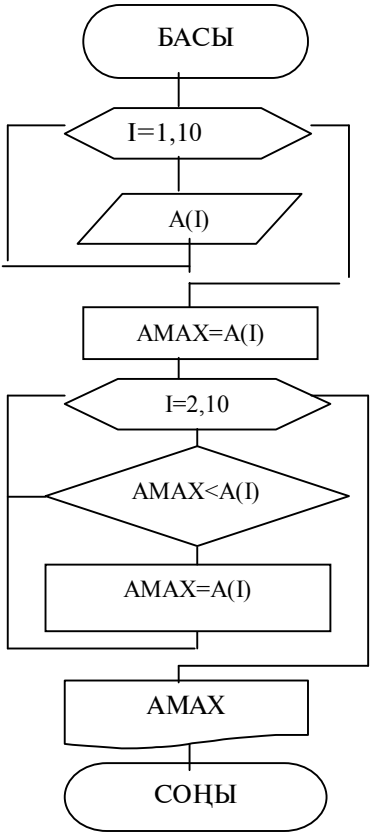


Бағдарламасы:

```
10 DIM A(10)
20 FOR I=1 TO 10
30 INPUT A(I): NEXT I
 40 S=0:P=1
 50 FOR I=1 TO 10
 60 S=S+A(I):P=P*A(I)
 70 NEXT I
 80 PRINT "S=";S, "P=";P
 90 END
```

1.8.1-сурет. Есептің блок-схемасы

2-есеп. А(10) массивінің ең үлкен элементін табу. Ол үшін массивтің ең алғашқы элементін ең үлкен элемент деп алып, цикл ішіндегі басқа аэлементпен салыстыра отырып, ең үлкен элементін табамыз. Есептің блок-схемасын сызайық (1.8.2-сурет).



1.8.2-сурет. Есептің блок-схемасы

Бағдарламасы:

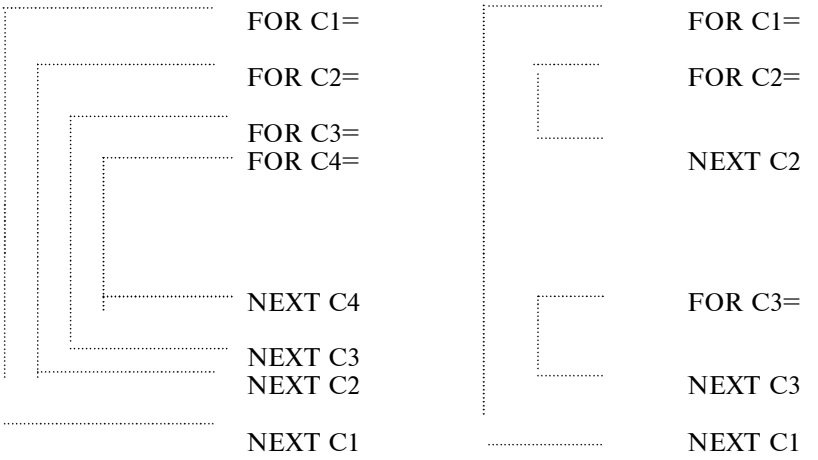
```
10 DIM A(10)
20 FOR I=1 TO 10
30 INPUT A(I):NEXT I
40 AMAX=A(1)
50 FOR I=2 TO 10
60 IF AMAX>A(I) THEN 80
 70 AMAX=A(I)
 80 NEXT I
 90 PRINT "AMAX=";AMAX
 100 END
```

1.8.2. Қабаттасқан цикл құрылымды алгоритмдерді қолданып, матрицаларды өңдеу

Қабаттасқан цикл құрылымы деп - бір немесе бірнеше цикл денелері орналасқан құрылымды айтамыз. Цикл

денесінде бір немесе бірнеше циклі бар құрылым - *сыртқы цикл*, ал цикл денесінде орналасқан цикл - *ішкі* деп аталады.

Ал қабаттасқан циклге қойылатын негізгі талап - ішкі цикл, сыртқы циклдің тек ішінде орналасуы керек. Қабаттасқан циклдегі параметрлердің өзгеру реті бірдей емес. Оның өзгеруі - қойылған есептің шартына байланысты. Төменде қабаттасқан циклдің дұрыс құрастырылған құрылым түрлерін қарастырамыз.

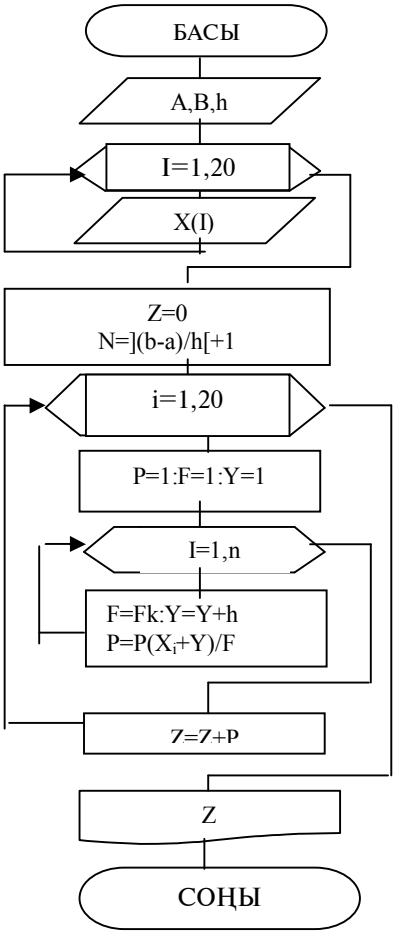


Есепті қарастырайық. Мысалы:
Есептеу керек:

$$Z = \sum_{i=1}^{20} \prod_{k=1}^{n} \frac{Xi + Yr}{k!}$$

Мұндағы X_i – (X1,X2,...,X20) массив элементтері, У – тің өзгеру аймағы - a<y<b, қадамы - h. Бұл есепті шығару барысында сыртқы циклде қосынды табылады, ал ішкі циклде – көбейтінді екі рет есептелінеді. Ол көбейтіндімен *факториалды* есептейді және өзгеру аймағына байланысты У айнымалының мәнін есептеу циклін қарастырады. Блок-схемасын қарастырсақ, оны төмендегідей сипаттауға болады (1.8.3-сурет).

Бұл схемада – 3-блокта Z қосындысының алғашқы мәнін нөлге теңейді, ішкі циклдің қайталану санын анықтайды,4-блокта i-параметрі өзгеретін сыртқы циклді құрады. 5-блокта көбейтіндінің (P=1), *факториалдың* (F=1) және айнымалының (y=a) алғашқы мәндері меншіктеледі. 6-блок R параметрі өзгеретін ішкі циклді құрайды. 7-блок F - *факториалды*, P - көбейтінді мен ағымдағы Y айнымалының мәнін есептейді. 8-блок сыртқы циклда қосындыны табады.



Бағдарламасы мына түрінде жазылады:

```
10 DIM X(20)
20 PRINT "МАСС. ЭЛ.ЕНГІЗУ"
30 FOR I=1 TO 20
40 PRINT I: INPUT X(I):NEXT I
50 INPUT "A,B,H";A,B,H
60 Z=0:N=INT((B-A)/H)+1
70 FOR I=1 TO 20
80 P=1: F=1: Y=A
90 FOR K=1 TO N
100 F=F*K:P=P*(X(I)+)/F
110 Y=Y+H: NEXT K
120 Z=Z+P: NEXT I
130 PRINT "Z=";Z: END
```

1.8.3-сурет.Есептің блок-схемасы

1.8.3. Матрицаларды өңдеу

Матрицалармен жұмыс істегенде, бұрынғы қарастырылған бағдарламалау әдістері қолданылады. Алғашқы есептелетін және қызметші матрицалар DIM операторы арқылы сиппатталуы керек. Матрицаларының элементтері READ,INPUT, PRINT операторлары арқылы әр элемент жекеше енгізіліп, қорытындыланады. Бағдарламаны қысқарту үшін, мүмкіндік болса матрица элементтерін енгізумен қатар, оны өңдеуді бір цикл операторында қолданған қолайлы. Мысалы, берілген екі матрицаның көбейтіндісін табу бағдарламасын құру керек. A(3,4) және B(4,2) берілген матрицалар. C(i,j) – матрицасының элементтері мына формула бойынша есептелінеді деп қарастырайық:

$$C_{i,j} = \sum_{k=1}^{4} A_{i,k} B_{k,j} \quad ,$$

мұндағы i=1,2,3 және j=1,2.

Бұл есепті шығару үшін қабаттасқан цикл құруымыз керек. Сыртқы циклде есептелінетін C матрицасының i – параметрінің өзгеретін жолын теру. Бұл циклде i-жолында j-параметірінің өзгеруіне қарасты орташа цикл атқарылады. i мен j–дің әрбір бекітілген мәніне K – параметріне байланысты ішкі цикл құрастырылады. Бұл ішкі циклде қосынды табылады және C(i,j) матрицасының элементтері есептеледі. Ал бағдарламасы мына түрде болады:

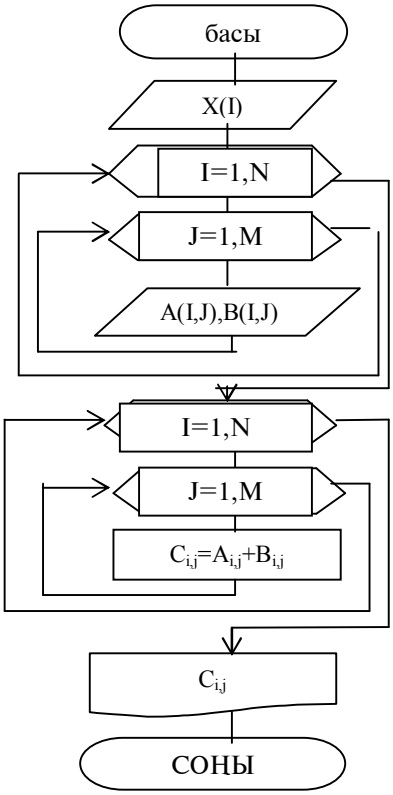
```
10 REM "Матрицалар көбейтіндісі"
20 DIM A(3,4), B(4,2),C(3,2)
30 FOR I=1 TO 3
40 INPUT A(I,1), A(I,2), A(I,3), A(I,4):NEXT I
50 FOR I=1 TO 4
60 INPUT B(I,1), B(I,2): NEXT I
70 FOR I=1 TO 3
80 FOR J=1 TO 3
90 S=0
100 FOR K=1 TO 4
110 S=S+A(I,J)*B(K,J)
```

```
120 NEXT K
130 C(I,J)=S
140 NEXT J:NEXT I
150 FOR I=1 TO 3
160 PRINT C(I,1), C(I,2): NEXT I: END
```

Тағы бір мысалды қарастырайық: екі матрицаны қосу керек. Бізге N жатық жолы және M тік жолы бар A(N,M), B(N,M) матрицалары берілсін: A және B матрицаларының қосындысы деп элементтері

$C_{i,j}=A_{i,j}+B_{i,j}$,

i=1,N, j=1,M өрнегімен анықталатын C=A+B матрицасын айтады. Берілген екі матрицаның қосындысын табу үшін, бірінің ішінде бірі орналасқан қос цикл қолданылады. Осы есепті шығару жолының блок-схемасы мен бағдарламасын төменде келтіреміз.




```
10 INPUT "N,M"; N,M
15DIM A(N,M),B(N,M),C(N,M)
20 FOR I=1 TO N
30 FOR J=1 TO M
40 INPUT A(I,J),B(I,J)
 50 NEXT J: NEXT I
 60 FOR I=1 TO N
 70 FOR J=1 TO M
 80 C(I,J)=A(I,J)+B(I,J)
 90 NEXT J: NEXT I
 100 FOR I=1 TO N
 110 FOR J=1 TO M
 120 PRINT C(I,J)
 130 NEXT J:PRINT
 140 NEXT I: END
```

1.8.4-сурет. Есептің блок схемасы

Бірінші операторда екі өлшемді массивтер DIM операторымен жарияланған. 20-50 операторларда массив элементтері үшін сыртқы (i) және ішкі (j) цикл құрылымы A(i,j), B(i,j) массивінің элементтерін енгізу үшін орындалады. Соңында ішкі және сыртқы циклдер аяқталады. Одан кейін массив элементтерін қосу үшін қайта қос цикл құрылып, екі массивтің қосындысы табылады (60-90 операторлар). Келесі қос цикл - массив элементерінің қосындысын қорытындылайды(100-140).

Үлкен көлемді матрицаларды өңдеу үшін, оның элементтерін *кездейсоқ сандар* тізбегімен толтырған ыңғайлы. Кездейсоқ сандармен жұмыс істеу жолдарына тоқталайық. Кездейсоқ сандар тізбегі әртүрлі процестерді модельдеу кезіндегі есептерде және компьютер жүйелерінде қолданылады. Оны сипаттау үшін **RANDOMIZE** операторы қолданылады. Оператордың жазылу үлгісі:

RANDOMIZE [База**]**

мұндағы База – жұмысқа қосылатын сандардың аумағы.

Бұл оператор кездейсоқ сандардың генерациясын іске қосады. Егер база көрсетілмеген болса, онда бағдарлама жұмысы тоқтатылып, экранға –32768-ден +32767-ге дейінгі база генерациясына қажетті аумақты белгілеуді сұрайтын сұраныс пайда болады. **RANDOMIZE** операторын жазбауға да болады, бірақ бағдарлама бірнеше рет орындалғанда, үнемі бірдей кездейсоқ сандар шығады.

Генерацияланған сандар тізбегінен кездейсоқ сандарды алу үшін **RND** стандартты функциясын қолдануымыз керек. **RND** – 0 мен 1 аралығындағы кездейсоқ сандарды береді (0.00000-ден 0.999999-ға дейін).

Үлестірімді заң бойынша **[**а,в**]** аралығындағы бүтін сандарды алу үшін төмендегі формула қолданылады.

X=int((B-A+1)*RND)+A,

мұндағы int – аргументтен кіші не оған тең максимальды бүтін. Мысалы: int (3.99)=3

Int(-3.99)=-4

Енді бүтін кездейсоқ санды алу мысалдарын қарастырайық.

```
CLS
RANDOMIZE TIMER
? ”N1 1–ден 10-ға дейінгі бүтін сан”
N1 = INT (10*RND)+1
? N1
? ”N2 -5–тен +5-ке дейінгі бүтін сан”
N2 = INT (11*RND)-5
? N2
```

Есеп қарастырайық. A(10,10) матрицасының элемент мәндерін **[**-10,200**]** аралықтағы кездейсоқ сандармен толтырып, матрицаның диагональ элементтерінің көбейтіндісін табу керек. Енді бағдарламасын құрайық:

```
CLS
RANDOMIZE
INPUT N
DIM A(N,N)
P=1
FOR I=1 TO N
FOR J=1 TO N
A(I,J)= INT(211*RND)-10
? “A(“;I;”,”;J;”)=”;A(I,J)
IF I=J THEN P=P*A(I,J)
NEXT J,I
? “P=”;P
END
```

Берілген матрицаның негізгі диагоналымен оның астындағы қосындысын табу үшін, алдыңғы есептегі IF операторын былай өзгертеміз:

```
IF I<=J THEN P=P*A(I,J)
```

1.8.4. Матрицалық операторлар

Кейбір Бейсик тілінің түрлерінде матрица мен векторларды өңдеу үшін қолданылатын арнаулы бекітілген

қызметші операторлар бар, олар - *матрицалық операторлар* деп аталады.

Сызықты алгебраның ережесіне сай матрицалық операторлар сандық массивтермен (матрица және векторлармен) әр түрлі амалдар орындайды. Сонымен бірге сандық массивтермен (матрица және векторлармен) әртүрлі амалдар орындайды және сандық, символдық айнымалылар мәнін енгізіп, қорытындылайды. Матрицалық операторларды қолданып, бағдарламаны анағұрлым жеңілдетуге болады. Бағдарламада қолданылатын матрицалық операциялардың белгілері, атқаратын қызметтері мен жазу үлгілері төменде келтірілген (3-кесте).

3-кесте

Операция белгісі	**Атқаратын қызметі**	**Жазу үлгісі**
Мат PRINT a,b,c...	Массивтің жолдық және сандық мәліметін баспаға алу	MAT PRINT A
Мат INPUT a,b,c...	Массивтің жолдық және сандық мәліметін баспадан енгізу	MAT INPUT X,Y
Мат READ a,b,c...	Сандық және жолдық массивтердің элементтерін DATA блогынан оқу	MAT READ A,B
Мат a=b	B массив элементтерін A массив элементтеріне сәйкестеп меншіктеу	MAT A=B
Мат c=a+b	Бірдей мөлшерлі екі сандық массивті қосу	MAT C=A+B
Мат c=a-b	Бірдей мөлшерлі екі сандық массивтің алымын табу	MAT C=A-B
Мат c=(R)*a	Массивтің әр элементтерін скаляр шамаға көбейту	MAT C=(3)*A
Мат c=a+b	A және B матрицаларын сызықты алгебра ережесі бойынша көбейту	MAT C=A*B
Мат c=INV(a)	A матрицасына кері C матрицаны есептеу	MAT C=INV(A)
Мат c=CON	C матрицасының әр элементіне 1-ді меншіктеу	MAT C=CON

3-кестенің жалғасы

Мат c=ZEP	С матрицасын нөлдендіру	MAT C=ZEP
Мат c=IDN	С бірліктің матрицасын құру	MAT C=IDN(A)
Мат c=TRN(a)	А орыналмастырылған матрица	MAT C=TRN(A)

1.8.5. Матрицалық операторлардың кейбір ерекшеліктері

Енді матрицалық операторлардың кейбір ерекшеліктерін атап өтейік:

1) Әрбір матрицалық оператор міндетті түрде MAT - қызметші сөзінен басталады;

2) Матрицалық операторды жазғанда массивтердің, дөңгелек жақшасыз, тек аты ғана жазылады;

Мысалы, А мен В меншіктеу матрицалық операторында MAT A=B деп айнымалы берілген болса, мұндағы А мен В жай айнымалы емес, сандық А мен В массивін негіздейді.

3) Баспаға мәлімет алу − матрицалық операторы MAT PRINT-тен кейін массив аты жазылып, одан кейін үтір қойылса, онда массив элементтері зоналық форматпен қорытындыланады. Ал массив атынан кейін нүктелі үтір қойылса, массив элементтері ықшамды түрде баспаға беріледі;

4) MAT INPUT матрицалық операторы бойынша массив элементтері тізбектелген жолдар ретінде, аралары бос орынмен бөлініп енгізіледі, ал символдық элементтер - қос тырнақшаға алынады;

5) Матрицаларда орын алмастыру мен көбейту операторлары да бір матрицаның атын, өрнектің екі жағында да қолдануға рұқсат етілмейді. Мысалы:

```
100 MAT C=C*B
110 MAT C=TRN(C) деп жазуға болмайды.
```

Осының алдындағы қарастырылған есеп бағдарламасын А(3,4) және В(4,2) матрицаларының көбейтіндісін табуды матрицалық операторларды қолданып жазып көрелік.

```
10 DIM A(3,4), B(4,2), C(3,2)
20 MAT INPUT A
30 MAT INPUT B
40 'A мен B матрицаларының көбейтіндісі'
50 MAT C=A*B
60 PRINT' "көбейтіндінің матрицасы"
70 MAT PRINT C:END
```

Бұрыңғы бағдарламалармен салыстырғанда, матрицалық операторларды қолданғанда, бағдарламаның анағұрлым ықшамды болғанын байқауға болады.

1.9. Қолданушылар анықтайтын функциялар

Функция мен процедураларды қолданып құрылған көмекші бағдарламалар. Тәжірибедегі көп есептерді шығару кезінде, бір алгоритм бойынша, айнымалылардың әртүрлі мәндері үшін параметрлердің көптеген есептеуін табу қажет болады. Бұл есептеулерді бағдарламалық модульдер (бағынышты бағдарламалар және процедуралар) арқылы сипаттаған кезде бағдарлама ықшамды болады.

Сонымен, амалдар тобын сипаттайтын операторларды бағдарламада қайталап жаза бермеу үшін, көмекші бағдарламалар қолданған қолайлы. Бағдарламаның әр жерінде арнаулы әдіспен пайдалануға болатын ерекше өрнектелген операторлар тобын - *көмекші бағдарлама* дейді.

Белгілі бір тағайындалған аты бар операторлар тобын - *көмекші бағдарламаның сипаттамасы* деп атайды. Берілген есептің толық алгоритмін сипаттауға арналған операторлар негізгі бағдарламаны құрайды. Көмекші бағдарламаны жұмысқа қосу үшін оған әсер ету керек, яғни негізгі бағдарламаның қажетті жерінде көмекші бағдарламаны жұмысқа қосатын арнаулы оператор жазылады. Оны - *көмекші бағдарламаны шақыру* деп атайды. Көмекші бағдарламаның сипаттамасында көрсетілетін параметрлер - есепті шығаруға қажетті операциялардың қандай шамалармен жүргізілетінін ғана көрсетеді. Ал олардың нақты мәндері көмекші бағдарламаны шақыру кезінде беріледі. Сондықтан көмекші бағдарламаның сипаттамаларында көрсетілген

параметрлер - *формальды параметрлер* деп, ал көмекші бағдарламаны шақыру кезінде көрсетілетін параметрлерді - *нақты параметрлер* деп атайды.

Формальды параметрлер мен нақты параметрлер арасында типі, саны, ұзындығы, орналасу реті бойынша сәйкестік болуы керек. Бейсик – жүйесінде қолданушылар анықтайтын процедураның екі түрін қолданады. *Олар өрнекті функциялар және бағынышты бағдарламалар*. Оларды бағдарламада қолданудан бұрын анықтау керек. Сондықтан барлық функциялар мен бағынышты бағдарламаларды бағдарламаның басында сипаттаған дұрыс. Сонымен, қорытынды ретінде бағдарламалардың келесі құрылымын қарастырған жөн:

массивтерді сипаттау;
өрнекті функцияны сипаттау;
бағынышты бағдарламаны сипаттау;
негізгі бағдарламаны сипаттау.

1.9.1. Қолданушылар анықтайтын өрнекті функциялар

Сонымен, есептерді шығарған кезде, бір арифметикалық немесе символды өрнектерді бағдарламаның бірнеше жерінде әртүрлі шамалар үшін есептеу мүмкін дедік. Бұл өрнектерді, есептеу процедурасын қолданушылар құратын стандартты емес функциялар арқылы сипаттауға болады. Қолданушы функцияға меншікі ат беріліп, ол функцияда қажетті өрнек сипатталады.

Функцияны бір рет анықтап алып, стандартты функциялармен бірге оны бағдарламада бірнеше рет қолдануға болады. Мұндай функцияларды шақыруды, тек сол функция анықталған бағдарламада ғана қолдануға болады.

1.9.2. Функцияның сипатталуы

Функция DEF орындалмайтын оператор арқылы сипатталады. Оның жазылу үлгісі:

пс DEF FN<*аты*> (g1, g2, g3, ..., gn)=өрнек

мұндағы, *аты* – FN префиксімен бірге функцияның атын анықтайды. Функцияның аты айнымалының аты сияқты анықталып, функцияны шақыру үшін қолданылады. g1, g2,

g3,...gn бүтін, нақты, символды формальді параметрлердің тізімі; өрнек – арифметикалық немесе символды типті өрнек. Бұл өрнек формальді параметрлер тізіміне кірмейтін айнымалылар болуы мүмкін, бірақ бұл айнымалылардың мәндері міндетті түрде функцияны шақырғанға дейін анықталуы керек.

Колданушы анықтайтын өрнекті функцияны шақырудың жазылу үлгісі: FN<аты> (b1, b2, ... bn), мұндағы b1, b2, ... bn-нақты параметрлер. Функцияны шақырғанда, барлық нақты параметрлер анықталған болуы тиіс. Функцияның орындалу реті:

Негізгі бағдарламада функцияның аты кездескенде басқару аты аталған көмекші бағдарламаға беріледі. Барлық формальды параметрлер нақты параметрлермен ауыстырылады да, олардың мәндері бойынша бағдарламада көрсетілген есептеулер орындалады. Шыққан нәтиже функция көрсеткішінің типі мен ұзындығына түрлендіріліп, өрнектегі есептеулерге қатысады.

Мысалы:

$$Z = \frac{\log_2 x + \log_b y}{2\log_{b+2}(x+y)}$$

функциясы берілген, есептеу бағдарламасын құру керек. Кез келген негіздегі логарифмді есептеу үшін DEF оператор функциясын қолданамыз. Бұл есептің кез келген алгоритімін $\log_a P$ = LnP/LnA ретінде анықтаймыз. Бұл формуладағы немесе өрнекті оператор-функцияда кез келген логарифмді негізі бар есепті шығару үшін логарифмнің а негізі мен ол есептелінетін р санын, екі параметрді белгілеуміз керек. Бұл функцияның атын FNB деп атасақ, онда бағдарлама келесі түрғыда сипатталады:

```
10 DEF FNB (A,P) = LOG (P) / LOG (A)
20 PRINT "x,y,b мәндерін енгізу"
30 INPUT x,y,b
40 z = (FNB(2,x)+FNB(B,y))/FNB(B+2,X+Y)*0.5
50 PRINT "Z=";Z
60 END
```

Тағы бір мысалды карастырайық:

$$w = \sqrt{x^2 + y^2 + \sin^2 xy} + \sqrt{y^2 + z^2 + bin^2 zy} + \sqrt{z^2 + x^2 + \sin^2 xz}$$

мәнін табу керек, мұндағы x=21.87, y=35.13, z= 9.87. Есепті шығару барысында әртүрлі аргументтерді квадрат түбірден шығару - үш рет орындалады, сондықтан FNB аты бар $\sqrt{a^2 + b^2 + \sin^2 ab}$ түбірді табу функциясын қолданайық. Мұндағы a, b – формальды параметрлер. Бағдарламада есептегенде, функцияны үш рет шақырамыз. Формальды параметрлер нақты x, y, z параметріне ауысады. Бағдарламаның жазылу үлгісі:

```
10 DEF FNB (A,B)=SQR(A*A+B+SIN(A*B)^2)
20 DATA 21.87, 35.13, 9.87
30 READ x,y,z
40 w = FNB (X,Y)+FNB(Y,Z)+FNB(Z,X)
50 PRINT "W="; W: END
```

1.9.3. Бағынышты бағдарламалардың сипаты

Бағынышты бағдарламалар GOSUB және RETURN операторлары. Қолданушы анықтайтын функция есептелінетін нәтиже біреу ғана болғанда қолданылады. Ал бағынышты бағдарлама - функциялық бағдарлама бірнеше операторлардан тұратын жеке бағдарламалық модуль болып табылады.

Бейсик тілінде бағынышты бағдарламаның арнайы аты болмағандықтан, бағдарламаны іске қосу үшін оны GOSUB операторымен шақырамыз. Оператордың жазылу үлгісі:

nc GOSUB m

мұндағы m – бағынышты бағдарламаның бірінші операторының нөмірі.

GOSUB операторы m - жолынан басталатын ішкі бағдарламаны шақырады. Бағынышты бағдарлама RETURN операторымен аяқталатын, операторлар тізбегінен тұрады. RETURN операторы кездескен кезде бағынышты бағдарлама

аяқталып, басқару GOSUB операторынан кейінгі операторға беріледі.

Бағынышты бағдарламаға параметрлердің мәнін беру және қорытындыны еске сақтау - қолданылған негізгі бағдарламадағы меншіктеу операторымен орындалады. Енді бағынышты бағдарлама құру методикасының мысалдарын қарастырамыз.

Мысалы:

$$U=e^{z1+y1}-e^{z2+y2}$$

функциясының мәндерін есептейтін бағдарлама құру керек.

Мұндағы z1 мен z2 және y1 мен y2 төмендегі теңдеулердің шешімі.

```
5z^2+20z-1.5=0,
2y^2-y-7.3=0.
```

Теңдеудің шешімдерін бағынышты бағдарлама ретінде сипаттайық. Егер теңдеудің шешімдері жорамал болса, онда олар 0-ге тең деп есептейміз. Бағынышты бағдарламада ax^2+bx+c=0 теңдеуінің x1 және x2 шешімін табу деп сипаттайық. Онда бағдарламаны былай жазамыз:

```
10 REM
20 A=5: b=20: c=-1.5
30 GOSUB 200
40 z1=x1: z2=x2
50 REM
60 A=2: b=-1: c=-7.3
70 GOSUB 200
80 y1=x1: y2=x2
90 U=exp(z1+y1)-exp(z2+y2)
100 PRINT "теңдеудің шешімі"
110 PRINT z1, z2, y1, y2
120 PRINT "функцияның мәні";U
130 end
200 D=b*b-4*a*c
210 IF D<0 THEN X1=0: RETURN
215 D1=SQR(D)
220 X1=(-B+D1)/(2A)
```

```
230 X2=(-B-D1)/(2A)
240 RETURN
```

Тағы бір мысал :

$$C_n^m = \frac{n!}{m!(n-m)!}$$

өрнекті есептеу бағдарламасын құрыңыз. Факториялды бағынышты бағдарлама арқылы есептеңіз. Бұл мысалда факториялды есептеуге, яғни N!, M! және (N-M)! факториалдарын есептеуге тура келеді. Сондықтан факториялды есептейтін операторлар тобын үш рет қолдану керек, оны қайталамас үшін, факториялды бағынышты бағдарламада есептейміз.

```
10 INPUT "N,M"; N,M
20 L=N: GOSUB 80:C1=P
30 L=M: GOSUB 80: C2=P
40 L=N-M: GOSUB 80: C3=P
50 C=C1/(C2*C3)
60 PRINT "C=";C
70 END
80 REM Факториалды есептеу
90 P=1
100 FOR I=1 TO L
110 P=P*I
120 NEXT I
130 RETURN
```

Бұл бағдарламаның 80-130-жолдарында бағынышты бағдарлама орналасқан. Оның көмегімен 20 жолда - N!, 30 жолда - M!, 40 жолда − N-M! факториялдары есептелінеді.

1.10. Графикалық тәртіп орнату ережелері

Бейсик тілінде дисплей экранында екі тәртіппен жұмыс істеуге болады, олар: символдар мен графиктерді бейнелеу, сонымен қатар экранды ақ-қара немесе түрлі-түсті ету. Символдар 25-қатардағы 80-орында бейнелене алады, ал график - *пиксел* деп аталатын (pixel − сурет салу) кішкене нүктелерден тұрып, 200-қатарда орналасқан 320 немесе 640

нүктелер арқылы салынады. Экрандағы бейне символдар тұратын шағын тік төрт бұрышты не кішкене нүктелерді сәулелендіру көмегімен тұрғызылады. Экранға символдар мен графиктердің салынуына байланысты оның жұмыс тәртіптерінің үш түрі бар (4-кесте).

1.10.1. Символдық тәртіп

Экран 40 не 80 символ сиятын 25 қатарға бөлініп, олар солдан оңға және жоғарыдан темен қарай бірден бастап нөмірленеді. Әр символдың орны қатар мен орын нөмірлерінің қиылысуымен анықталады. 1-ші қатардан 24-ші қатарға дейін мәлімет орналастыруға болады, ал 25-қатар – функционалдық пернелер қызметін көрсетіп тұрады да, оны пайдалану үшін қосымша әрекеттер жасау кажет. Символдар түсі мен экран реңін әртүрлі түске бояуға болады.

4-кесте

Рет нөмірі	Бейне түрі	Экран құрылымы
0	Ақ-қара не түрлі-түсті символдар	40 не 80 символадан тұратын 25 қатар
1	Ақ-қара не түрлі-түсті графиктер	320x200 нүктелер; 1-қатарда 40 символ
2	Ақ-қара түсті графиктер	640x200 нүктелер; 1-қатарда 80 символ

1.10.2. Графикалық тәртіп

Экран 320 нүктеден тұратын 200 қатарға бөлінеді, тәртіп орта деңгейлі болып саналады, ал бір қатарда 640 нүкте орналастыра алатын қосымша мүмкіндік – жоғары деңгейлі екінші графикалық тәртіп деп есептеледі. Нүкте қатарларын жоғарыдан төмен қарай, ал бір қатарда орналасқан нүктелерді - солдан оңға қарай нөлден бастап нөмірлеу келісілген. Әр нүкте көлденең есептелетін X координатасы мен тік бағаналар нөміріне сәйкес келетін Y координаталарының қиылысумен анықталады. Айта кететін бір жайт, келісілген нөмірлеу бойынша OX осі солдан оңға қарай, ал OY осі жоғарыдан төмен қарай бағытталған деп есептеледі.

Орта деңгейдегі графикалық тәртіп 1 нөмірімен, жоғары деңгей 2 нөмірімен белгіленеді. Экранға нүкте, түзу сызық, шеңбер (эллипс), тұйық сызықтар салуға және оларды әртүрлі түспен бояуға болады.

Бір қатарда 1-тәртіпте 40, ал 2-тәртіпте 80 символ бейнелеуге болады. Жоғары деңгей мүмкіндігі тек ақ-қара түспен жұмыс істей алады.

1.10.3. SCREEN, WIDTH, CLS операторлары

Бейсик жүйесі жүктелгенде бірден экранда 0-нөмірлі символдық тәртіп орнайды, басқа тәртіпке көшу SCREEN операторымен жүргізіледі. Оның форматы:

SCREEN N[, M],

мұндағы N − тәртіп нөмірі (0, 1, 2); M − әртүрлі түске бояу мүмкіндігі (0, 1). Символдық тәртіпте M=0 экранның түрлі-түсті болмайтынын білдіреді, ал бірінші графикалық тәртіпте M 0-ге тең болса, ол керісінше экранның әртүрлі түске боялатынын көрсетеді. Екінші графикалық тәртіп тек ақ-қара түсті ғана пайдаланады.

WIDTH операторы экран қатарының енін анықтайды, мысалы:

WIDTH 80' қатарда 80 символ орналасады;

WIDTH 40' қатарда 40 символ орналасады;

1 − графикалық тәртіпте экран енін 80 символ сиятындай болса, онда дисплей автоматты түрде 2-графикалық тәртіпке көшеді. Экранды тазалау үшін CLS операторы қолданылады. Ол тек CLS сөзінен ғана тұрады, мысалы,

10 CLS

1.10.4. Таңбалар түсін өзгерту

Таңбалар мен экран реңінің түсін өзгерту үшін COLOR операторы қолданылады. Оның символдық тәртіптегі форматы:

COLOR [<таңба-түсі>] [,<рең-түсі>]

<таңба-түсі> - 0 мен 31 арасындағы бүтін сан, ол таңбаның түсін анықтайды;

<рең-түсі> - 0 мен 7 арасындағы бүтін сан, ол экранның түсін анықтайды. Әр түстің өзіне тең нөмірі болады. PM PC

тәрізді компьютерлерге арналған Бейсик жүйесінде түстердің нөмірлері төмендегідей:

0 - қара 8 – сұр (боз)
1 – көк 9 – көкшіл
2 – жасыл 10 – ашық жасыл
3 – көгілдір 11 – аспан көк
4 - қызыл 12 - қызғылт
5 – күлгін 13 - қызғыш
6 – қоңыр 14 - сары
7 – ақ 15 – әппақ

Сол жақтағы бағана негізгі түстерді, ал оң жақтағысы негізгі түстердің ақшылдау болып келетін қосалқы түрлерін айқындайды. Кейбір экрандарда бұл екі бағана түстерінің нөмірлері (0 және 8, 1 және 9 т.с.с.) өзгеше болғанмен, оларды көзбен ажырату қиын.

COLOR 7, 3' – көгілдір экранда ақ таңбалар бейнеленеді;

COLOR 14, 1' – көк экранда сары символдар бейнеленеді.

Түстердің 16-дан 31-ге дейінгі нөмірлері алғашқы 16 (0-15) түстермен жазылған символдардың өздеріне назар аудартып, жыпылықтап жанып-сөніп тұратын тәртібін орнатады. Ақ-қара түсті экранда көрсетілген бұл нөмірлердің көбі қолданылмайды, олардың 5-кестеде берілген мүмкіндіктері пайдаланылады.

5-кесте

Атқаратын жұмысы	Таңба түсі	Рең түсі
Қара ренді экранда ақ таңбалар	7	0
Символдардың астын сызу	1	0
Ақ реңді қара таңбалар	0	7
Таңбаларды көрсептеу	0	0

Орта деңгейлік графикалық тәртіпте (SCREEN 1) түсті өзгерту үшін COLOR операторының жазылу форматы басқаша болады.

COLOR [<рең-түсі>] [,<палитра-нөмірі>]

мұндағы <рең-түсі> - реңнің түсін анықтайды (0, 1,..., 15); <палитра-нөмірі> - түстер палитрасының нөмірін анықтайды, жұп сан – 0 палитра, тақ сан – 1 палитра. Егер оператордың екі параметрінің біреуі көрсетілмесе, онда оның мәні өзгермейді. Оператордың екінші параметрі түстің емес, жалпы түстер жиыны – палитра нөмірін анықтайды. Палитраның екі түрі бар, олар 0 және 1 сандарымен белгіленеді. Палитраның өзгеруі - экрандағы келесі сызылатын сызық түстерін өзгертеді. Әр палитра, түстердің төрт мүмкіндігінің бірін таңдай алады. Бұдан былай сызылатын график сызықтары тек 0, 1, 2, 3 нөмірлеріне сәйкес түстерге боялады. Ал түс нөмірі бұл оператордан кейін орындалатын сызық, шеңбер сызатын операторларда беріледі. Палитраға байланысты бұл нөмірлер 6-кестеде көрсетілген түстерді бейнелейді.

1.10.5. Экранға график тәртібін орнату мысалы

6-кесте

Түс нөмірі	**0 – палитра**	**1 – палитра**
0	Рең түсіндей	рең түсіндей
1	Жасыл	көгілдір
2	қызыл	күлгін(қызғыш көк)
3	қоңыр	ақ

Егер COLOR операторда параметрлер керсетілмей кетсе, онда палитра нөмірі – 1, ал сызық нөмірі – 3 болып орнайтыны келісілген. Мысалы:

```
10 SCREEN 1 ‘экранның 1 – графикалық тәртібі
   орнайды.
20 COLOR 2.0 ‘жасыл экранда 0 – палитраға сәйкес
   түстер
30 FOR x1=0 TO 150 ‘таңдап сызық арқылы сурет
40 X2 = 60+50*SIN (X1/30)  ‘салынады
50 Y2 = 90+100*COS (X/30)
60 LINE (X1,100) – (X2,Y2)
70 NEXT X1
80 END
```

Экранға салынатын сызық түстері палитра нөміріне сай төмен көрсетілген операторлар арқылы тағайындалады.

1.10.6. Экранға график салу

Дисплей экранына график салу үшін, оның нүктелерінің координаталарын орта деңгейлі тәртіпте X = 0 −319, Y = 0 - 199, ал жоғары деңгейде X = 0-639, Y = 0-199 аумағында көрсету қажет. Экранда X осі солдан оңға қарай, ал Y осі жоғарыдан төмен қарай бағытталған. Дисплей экранның бетіне /кейін қағазға/ нүкте, түзу немесе қисық сызық, шеңбер, эллипс және кез келген тұйық сызық сызып шығаруға болады. Тұйық сызықтардың ішін әртүрлі түске бояу мүмкіндіктері де бар. Сызықтарды жылжыту, айналдыру және басқа орынға көшіру арқылы да көрнекті бейнелер жасауға болады. Енді осы айтылған жұмыстарды атқаратын операторларды қарастырайық.

1.10.7. Нүкте салу операторы

Экраннның кез келген жеріне нүкте салуға болады, оның форматы:

PSET (x,y) [,<түсі>],

мұндағы /x, y/ - салынатын нүкте коодинаталары; <түсі> − нүктенің түсі (0, 1, 2, 3). Егер нүкте түсі көрсетілмесе, ол палитраға сейкес 3 − нөмірлі түске боялуы тиіс. Мысалы:

```
10 SCREEN 1: CLS: COLOR 7,0 ' ақ экранда;
20 PSET (100, 100), 1          ' жасыл нүкте салынады.
```

1.10.8. Салынған нүктені өшіру операторы

Салынған нүктені өшіру үшін PRESET операторын қолданады, оның форматы:

PRESET (x, y),

мұндығы x және y өшірілетін нүктенің координаталары.

Төмендегі бағдарламада X =100 түзуінің бойына біртіндеп ақ түсті нүкте салынады да, шамалы уақыттан соң нүкте өшіріліп, оның төмен жылжыған бейнесі қайта салынады, т.с.с.

```
10 SCREEN 1: COLOR 1,1 ' 1- графикалық тәртіп, көк
                         экран;
```

```
20 FOR Y = 0 TO 199      ' цикл басы, 200 рет нүкте
                           салынады;
30 PSET (100, Y)         ' ақ нүкте салу;
40 FOR K = 0 TO 100      ' нүктенің көрініп тұратын
                           уақыты;
45 PRESET (100, Y)       ' нүктені өшіру;
50 NEXT K
60 NEXT Y                ' цикл соңы.
```

1.10.9. Түзу сызық салу операторы

Түзу сызық салу операторы - экранда екі шетінің координаталары берілген кесінді, тік төрт бұрыш және боялған тік төрт бұрыш салу үшін қолданылады. Форматы:

LINE [(x1, y1)− (x2, y2)[,<түсі>][,B[F]]],

мұндағы (x1, y1), (x2, y2) − түзудің бастапқы және соңғы нүктелерінің координаталары, бастапқы нүкте көрсетілмесе, онда курсор тұрған соңғы нүкте түзудің басы болып есептеледі; <түсі> - сызылатың сызық түсінің нөмірі (0, 1, 2, 3), қойылмаса, 3 болып саналады; B − диагоналы осы түзу болып келген тік төрт бұрыш салу белгісі; BF − тік төрт бұрыштың іші сызық түсіне боялатынын көрсетеді. Мысалдар:

```
LINE (10,10)-(100,50) '  координаталары көрсетілген
                         нүктелер арасына кесінді
                         жүргізіледі;
LINE (10,10)-(100,50) 2, B 2 − нөмірлі тік төрт бұрыш
                          тұрғызылатынын көрсетеді;
LINE (10,10)-(100,50) 2, BF ' F параметрі тік төрт
                          бұрыштың іші сызық түсіне
                          боялатыннын көрсетеді.
```

Төмендегі екі оператор 2 − графикалық тәртіп орнатып, экранның диагоналы бойынша сызық жүргізеді.

```
10 SCREEN 2
20 LINE (0,199)-(639,0)
```

Ал төмендегі бағдарлама LINE операторының бір параметрлі мүмкіндігін пайдалана отырып, экранда тік төрт бұрыш сызады.

```
10 SCREEN 1: COLOR 1,0:CLS
20 PSET (10,10), 2
30 LINE – (100,10),2
30 LINE – (100,50)
30 LINE – (10,50)
30 LINE – (10,10)
```

Енді берілген фукция гафиктерін салу жолдарына тоқталайық. Кез келген функция графигін салу үшін, оның координат осьтерін дисплей экранының ортасына салу керек. Ол үшін LINE операторын қолданамыз. Енді мысал қарастырамыз.

1-есеп. Айталық **Y=SIN(X)** функциясының графигін салу керек. Мұндағы х-тің өзгеру аумағы 0-ден 2π аралығында. Бағдарламасын құрастырамыз:

```
10 SCREEN 2
11 locate (2),(1):print "       Вариант 1"
12  locate (3),(1):print "   sin(x)  [0;2*pi]"
14 locate (22),(1):print "     Кафедра КТПиП"
20 liNE (300,0)-(300,200)
30 LINE (0,100)-(600,100)
31 line (300,0)-(297,3)
32 line (300,0)-(303,3)
33 line (600,100)-(597,103)
34 line (600,100)-(597,97)
35 locate (14),(76):print "X"
36 locate (1),(36):print "Y"
45 PI=3.14159265#
50 X=0
60 PSET(300+X*60,100-x*SIN(X)*60)
65 X=X+.01
70 IF X<2*pi GOTO 60
71 if inkey$="" then 71
75 screen 0
80 END
```

2-есеп. Y=SIN(X)+COS(X) - 1 функциясының графигін салу бағдарламасын құру керек. Бұл бағдарламада координат осьтерін салу үшін WINDOW операторын колданайық. Ол

координат осьтерін берілген шектеулер мәндерін қолданып, функция графигін салу кезінде x,y нүктелерінің координаттарын өзі есептейді.

```
10  REM  Y=SIN X +COS X -1 функциясының графигін салу
30 CLS
40 SCREEN 2
50 WINDOW (-2,-2)-(2,2)
60 LINE (-2,0)-(2,0)
70 LINE (0,-2)-(0,2)
80 FOR X=-3.1415 TO 3.1415 STEP 0.1
90 Y=SIN(X)+COS(X)-1
100 PSET (X,Y)
110 NEXT X
120 END
```

1.10.10. Шеңбер сызу операторы

Шеңбер сызу операторы - осьтері координата сызықтарына параллель болып келген эллипс, шеңбер, доға, радиустер тұрғызады. Форматы:

CIRCLE (x, y), R[,<түсі>][,<басы>],[<соңы>][,U],
мұндағы (x, y) – шеңбер центрінің координаталары; R- шеңбердің радиусі немесе эллипстің үлкен осі; <түсі> - палитраға сәйкес сызық түрі (1, 2, 3) көрсетілмесе, 3 болып есептеледі; <басы>, <соңы> - радианмен берілген 0 мен 2π арасындағы бұрыштардың басы мен соңы, осы бұрышқа тірелетін доғаның сызылатынын көрсетеді. Егер бұрыш мәні теріс сан болса, онда доғаның шеті центрмен қосылады; U - эллипс салуға арналған шеңберді қысу коэффициенті. Егер U>1 болса, OY осіне қысылған эллипс, ал U<1 болса, OX осіне қысылған эллипс тұрғызылады. Егер U көрсетілмесе немесе U=1 болса, онда шеңбер сызылады. Мысалдар:

1) 100 CIRCLE (160,100), 50 ‘экран ортасында шеңбер сызу;
2) 10 SCREEN 1: COLOR 2,0 ‘ жасыл рең, 0 – палитра;
   20 FOR R=20 TO 100 ‘ R 20-дан 100 – ге дейін өзгереді;
   30 CIRCLE (100,160), R,2 ‘ ортада қызыл шеңберлер сызу;

```
   40 NEXT R                    ‘ цикл соңы.
3) 10 PI=3.1415: SCREEN 1:CLS
   20 CIRCLE (160,100),50,1,PI/2,PI,
```

мұнда 90-нан 180-ге дейін доға сызылады.

```
4) 10 PI=3.1415
   20 CIRCLE (160,100),50,1,-PI/2,-PI
```

Доғаның шеттері радиус арқылы центрмен қосылады.

1.10.11. Түйық сызықтар ішін бояу операторы

PAINT(бояу) операторы - түйық сызықпен аймақтарды бояу үшін қажет. Бұл операторда аймақтың қоршау сызығының түсі және оның ішін қандай түспен бояйтыны көрсетіледі, оның форматы:

PAINT (x, y), <бояу-түсі>, <сызық-түсі>, мұндағы (x, y) – бояуды бастайтын нүктеміздің координаталары. Мысалы:

```
10 SCREEN 1: COLOR 2,0: CLS
20 CIRCLE (100,100),20,3 ‘ шеңбер ақ сызықпен сызылады;
30 PAINT (100,100),2,3 ‘ ақ дөңгелек іші қызылмен боялады.
```

Егер түйық сызықтың бір жері байқаусызда "үзілген" болса, экран түгел бояу түсін қабылдайды.

Келесі мысалда жасыл экранда тесігі болғандықтан, бүкіл экран қызыл түске боялып кетеді.

```
10 SCREEN 1: COLOR 2,0: CLS
20 CIRCLE (100,100),20,3  ‘ шеңбер ақ сызықпен сызылады;
25 LINE (100,100)-(140,150),0 ‘ сызық түсі реңмен бірдей;
30 PAINT (100,100),2,3      ‘ ақ дөңгелек іші қызылмен
```

боялады да, артынан экран түгел боялып кетеді.

1.10.12. Әртүрлі сызық сызу операторы

Экранда қағаз бетінде сурет салған тәрізді әртүрлі сызықтар сызуға болады. Қарындашты көтеріп, қалаған нүктеге жылжытатынымыз тәрізді, экран бетінде де кез келген жерден бастап сызық салу мүмкіндігі бар. Осындай әректтер DRAW (сызу) оператормен орындалады, оның форматы:

DRAW “<сызу-командалары>",

мұндағы DRAW - оператор аты (сызу); <сызу-командалары> - қос тырнақшаға алынған символдар тіркесі, олардан әртүрлі командарлар құрастыруға болады. Бұл оператордың командалары Бейсик тілінің басқа график салу мүмкіндігін пайдаланбай-ақ, түрлі масштабпен сызықтар сызып, оларды жылжыту, бұрау жұмыстарын жүргізе алады. Тұйық сызықтар сызып, олардың ішін әртүрлі түспен бояй аламыз. Кез келген команданың жазылуы:

DRAW “...Yznn...”

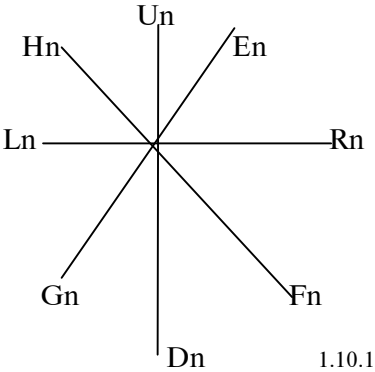
Y - міндетті түрде жазылмайтын команда толықтырғышы; Z- команданың өзі; nn - бүтін сан, ол кейде қатар орналасқан екі бүтін саннан тұруы да мүмкін.

Нүктені кез келген бағытта жылжыту командасы Mx, y тіркесінен тұрады, ол курсор тұрған орыннан бастап, x, y координаталарымен берілген нүктеге дейін кесінді сызады. Егер, DRAW “M100,100”командасын орындасақ, онда курсор тұрған орыннан, координаталары x=100, y=100 нүктесіне дейін, сызық жүргізіледі. Жұмыс істеу барысында қате жібермес үшін, курсордың қай орында екенін қадағалап отыру қажет. Есте сақтайтын бір жайт, бірінші графикалық тәртіпте CLS операторын орындаған соң, курсор экран ортасына (x= 160, y=100) келіп орналасады.

M командасындағы x, y мәндері оң немесе теріс таңбалы болуы мүмкін, мұндай жағдайда жаңа координаталар абсолютті емес салыстырмалы деп есептеліп, оның жаңа мәні бұған дейін курсор тұрған нүкте координаталарына осы көрсетілген сандарды қосу не алу арқылы айқындалады. Жоғарыда көрсетілген мысалды аздап өзгертейік.

DRAW “M+100,100”

Енді сызықтың соңғы нүктесі курсор тұрған орыннан 100 адым (пиксель) оңға ығысады, ал OY осі бойынша бұрынғыша абсолюттік координата көрсетілген, яғни y=100. Нүктені көрсетілген бағытта жылжыту командалары: оңға, солға, жоғары, төмен және қиғаш бағытталған сызықтар салу үшін қолданылады. Бұл командалар схема түрінде 1-суретте көрсетілген. *Ескерту: Көрсетілген бағытта нүктелермен берілген қашықтықты* масштаб командасы Sn арқылы өзгертуге болады.



1.10.1-сурет. Командалардың орындалу бағыты

Команда атаулары көрсетілген бағытты мегзейтін ағылшын сөздерінің алғашқы әріптерінен алынған, олар: up (жоғары), down (төмен), right (оңға), left (солға). Ал қалған қиғаш бағыт атаулары сағат тілімен орналасқан латын алфавитінің E-ден кейінгі әріптері.

Жылжыту командаларының толықтырғыштары кез келген команда атының алдына орналаса алады. Олар B немесе N әріптерінен тұрады: B − нүкте жаңа орынға жылжитынын, бірақ сызық салынбайтынын білдіреді, ал N - жылжу барысында сызық салынып, соңында курсор шыққан нүктесіне қайтып оралатынын көрсетеді. Енді толықтырғышты пайдаланып, экранда жұлдызша салып шығайық:

```
10 SCREEN 1:COLOR 0,0 ' 1 − графикалық тәртіп, 0-палитра;
20 CLS:KEY OFF '          экранды тазалу, курсор ортада;
30 DRAW "NU10 NE10 NR10 NF10 ND10 NG10 NL10 NH10"
```

Жұлдызшаны экранның ортасына емес, кез келген жерге орналастыру үшін 30-қатарда M командасын B толықтырғышмен қосып жазамыз:

30 DRAW "BM60,60 NU10 NE10 NR10 NF10 ND10 NG10 NL10 NH10".

Жалпы график салуды BM x, y командасын пайдаланып, курсорды сызып алмай, экранның қажетті нүктесіне орналастырып алған соң бастаған жөн.

Сызық түсін өзгерту командасы Cn өзінен кейін орналасқан командалар арқылы салынатын сызық түсін палитраға сәйкес 0, 1, 2, 3 нөмірлі түстердің біріне бояйды. Жоғарыда көрсетілгген жұлдызшаны қызыл түспен салу үшін 30-қатар былай жазылуы тиіс:

30 DRAW "BM60,60 C2 NU10 NE10 NR20 NF10 ND10 NG10 NL10 NH10"

Бұрын n-нөмірлі түспен салынған тұйық сызықтар ішін m-нөмірлі түспен бояу жұмысы курсор тұрған нүктеден басталады. Осы команданы қолданып, экранда түрлі-түсті кішкене үй бейнесін салайық:

```
10 SCREEN 1: COLOR 0,1
20 KEY OFF: CLS            ' key off 25-қатарды тазартады;
30 DRAW "C3 U20 E20 F20 D20 L39"
40 DRAW "BM170,90 H2,3"
```

Соңғы екі қатарды бір DRAW операторымен жазуға да болады.

1.10.13. Экранда салынған сызықтарды бұрау командалары

Экранда салынған сызықтарды бұрау командалары - An мен Tan. Алғашқы An командасы бұдан кейін сызылатын сызықтардың бәрін n 90^0 бұрышқа бұрады. 0 градусқа горизонталь бағытта оңға бағытталған вектор сәйкес келеді де, бұрыштар мәні осы вектордан бастап, сағат тілінің бағытына қарсы есептеледі.

Ескерте кететін тағы бір жайт, кез келген команда параметрі ретінде тек бүтін санды ғана емес, айнымалыны да

алуға болады. Бұл жағдайда "теңдік" белгісінен кейін айнымалы жазылып, сонан соң нүктелі үтір қойылуы тиіс, мысалы, DRAW "C2" операторын N=2:DRAW "C=N;" түрінде де жазуға болады. Осы мүмкіндікті пайдаланып, бір-бірінен тік бұрышқа бұрыла отырып, айнала жылжу қозғалысын көрсететін төрт тілсызық (стрелка) салу бағдарламасын An командасы арқылы жасайық:

```
10 SCREEN 1:COLOR 0,0: CLS: KEY OFF
20 FOR I=0 TO 3
25 CLS
30 DRAW "A=1;C=1;  U10 R50 U5 F15 G15 U5 L50 U10'
40 NEXT I
```

Екінші бұру командасы – TA n келесі сызылатын сызықтарды –360-тан +360-қа дейінгі кез келген бұрышқа бұра алады. Егер бұрыш мәні теріс таңбалы болса, бұру - сағат тілінің бағыты бойынша жүргізіледі.

Осы команданы пайдаланып, алдыңғы мысалды аздап өзгертейік. Бұл жолы сағат тілінің секунд тілсызығының айналу қозғалысын бейнелейтін бағдарлама жасаймыз:

```
10 SCREEN 1: COLOR 0,0: CLS: KEY OFF
20 FOR I=0 TO 360 STEP 3
30 DRAW "C0 NU70 TA=I;" ' тілсызықты өшіру;
40 DRAW "C1 NU70"       ' тілсызықты салу;
50 NEXT I
```

1.11. Файлдармен жұмыс істеу

Ұзақ сақтауға арналған мәліметтер сыртқы сақтау құрылымында файл түрінде орналасады. *Файл* дегеніміз – сыртқы сақтау құрылымына белгілі бір атпен аталып, үздіксіз жазылатын біртекті мәліметтер жиыны. Көпшілік жағдайда, мәліметтерді сақтаудың ынғайлы түрі ретінде файлды пайдалану жолдарын атап өтейік. Мәліметтер жиыны тым көп болған жағдайда, олардың бәрін жедел жадта орналастыруға болмайтындықтан, оны файлда сақтайды. Файл құрып, бағдарлама жұмысын аяқтағаннан кейін ол сақталып қалады.

Файлда жазылған мәліметтерді бірнеше бағдарламада пайдалануға болады. Файл біртекті компонент (мәліметтер, сөз тіркестері немесе логиқалық жазулар) жиыннан құралады. Файл компоненттерін қарастырудың екі түрі бар: біртіндеп қарастыру, белгілі бір компонентке тікелей шығу немесе реті арқылы өту. Біртіндеп мәліметтерді қарастыратын файлды-*тізбек арқылы қатынас жасайтын файл* деп атаймыз, ал мәліметтерді белгілі компонент немесе өзіндік нөмірі арқылы оқу немесе өзгертуге болатын файлды - *тікелей қатынас жасайтын файл* дейміз.

Файлға орналастырылған мәліметтерді оқу, өзгерту немесе жазу үшін, файлдың аты-жөні (спецификациясы) болуы қажет. Онда, файлдың толық аты мен файл жазылған құрылғының аты болады. Файлдың толық аты - оның өзіндік аты мен атаудың тіркеуінен тұрады, оның жазылуы:

“аты”.“заты”

FANAT.BAS
PROGR.TXT

1.11.1. Тізбек арқылы қатынас жасайтын файлдар

Бейсик тілінде бағдарламаға мәліметтерді енгізіп-шығару кезінде, файл атауының орнына белгілі нөмір қойып пайдалануға болады. *Файл нөмірі* дегеніміз - бағдарламада қолданылатын бүтін сан. Бағдарламаларда файлды пайдалану үшін келесі амалдарды орындау керек:

1. Файлды ашу;
2. Мәліметтерді файлға жазу;
3. Файлдан мәліметтерді оқу;
4. Ашылған файлды жабу.

Файлды ашу. Файлды ашу операторы OPEN - файлды өңдеу тәртібін (оқу немесе жазу) және оған белгі бекіту мүмкіндігін ұйымдастырады. Оның жазылуы:

OPEN <”файлдың аты”> FOR <өңдеу тәртібі> as #N

Мұндағы OPEN- файлды ашу операторы; құрылғы белгісі - “файлдың аты” және файлдың толық аты көрсетілуі мүмкін. Құрылғылардың белгілі стандартық белгісі: A, B, C немесе 1,

2, 3 - мәліметтерді дискіге шығару немесе оқу ретінде қолданылады; <өндеу тәртібі> - мәліметтерді оқу немесе жазу тәртібін көрсетеді, егер ол INPUT – болса, онда дискідегі файлдан жедел жадтағы бағдарламаға мәліметтер жиынын енгізеді; OUTPUT- болса, жедел жадтағы бағдарламадан дискідегі ұйымдастырылған файлға мәліметтер жиынын шығару болады; APPEND – жазылған жазулар тізбегінің соңына, толықтыру тәртібін орнатады; N – файлдың нөмірі 1-ден 15-ке дейін бүтін сан, шығару-енгізу операторында файлдағы жазулар тізбегіне сұраныс жасау немесе жазу үшін пайдаланылады. Мысалы:

OPEN “A:\NIK.DAT” FOR OUTPUT AS #1

операторы тізбекті түрде қатынас жасайтын жаңа файлды ашады (құрады), жазулар тізімін жедел жадтағы бағдарламадан A: дискісіне жазу (шығару) үшін, бірақ онда жаңа жазуларды жазу кезінде ескі жазулар сақталмайды. Осы бағдарламадағы файлға 1-ші нөмір арқылы сұраныс жасалады.

1. OPEN “C:\RIK.TXT” FOR INPUT AS #2

операторы құрастырылған файлды ашып, жазулар тізімін оқу (енгізу). C: дискісінен жедел жадтағы орналасқан бағдарламадағы 2-ші нөмірлі басқа ашылған файлдарда өзгертулер жасайды.

2. OPEN “A:\TIR.REZ” FOR APPEND AS #3

операторы құрастырылған файлды ашады, бірақ файлдың соңын көрсететін белгіні файлдың соңына қояды. Сонымен, ескі жазулар өшірілмей қалады да, жаңа жазулар файлдың соңына жазылады.

Файлды жабу. Файлды пайдаланған соң, оны міндетті түрде жабу қажет. Файлды жабу операторының жазылуы:

CLOSE #N

мұндағы CLOSE – жабу операторының аты: N – жабылуға тиіс файлдың нөмірі. Мысалы:

CLOSE #1 - нөмірі бірінші файлды жабу;
CLOSE #5 - нөмірі бесінші файлды жабу;
CLOSE - барлық ашылған файлды жабу.

Мәліметтерді жазу. PRINT #, PRINT # USING, WRITE # операторлары мәліметтерді жедел жадтағы бағдарламадан дискідегі файлға шығарады (жазады). PRINT # - операторы файлға жазылатын сандық мәндерді бос орнымен

бөліп жазады. Сөз тіркестерін (логикалық жазулар) немесе символ тізімін бір-біріне тиістіріп орналастырады. Оператордың жазылуы:

PRINT # файл нөмірі [, айнымалы тізімі]

Мысалы:

A=5.2 : B= 6.7: F$= ’’РАКЕТА’’

PRINT #2, A, B, F$, “РАКЕТА-”. “5-МАРС”

мұндағы 2-ші нөмірлі файлға 5.2, 6.7, “РАКЕТА-5-МАРС” сөз тіркесін жазады.

PRINT # USING мәліметтерді жедел жадтағы бағдарламадан көрсетілген нұсқау арқылы дискіге жазады.

Оператордың жазылуы:

PRINT # файл нөмірі [,айнымалы тізімі]

Мысалы:

```
OPEN ’’A:\NIK.DAT’’  FOR OUTPUT   AS  #1
FOR Z=1 TO 50
PRINT #3, USING ‘‘ ##.##’’; SQR(Z)
NEXT Z
CLOSE #2
```

3-ші нөмірлі файлға PRINT #3, USING операторы түбір асты х-тың мәнін ##.## нұсқау арқылы үтірмен бөліп жазады.

Көпшілік жағдайда сандық мәндерді дискіге жазу үшін WRITE # операторы қолданылады. Оператордың жазылуы:

WRITE # файл нөмірі, айнамалы тізімі:

Ол әрбір мәнді үтірмен бөліп жазады, кездесетін сөз тіркестерін қос тырнақшаға алып отырады. Мысалы:

```
OPEN ’’A:\DIDAR.TXT’’  FOR OUTPUT   AS  #1
FOR I=1 TO 10
PRINT ’’ Введите ФИО, должность, оклад’’
INPUT F$,P$,OKLAD
WRITE #3, F$,P$,OKLAD
NEXT I
CLOSE #3
```

Мәліметтерді шығару. Жедел жадтағы бағдарламаға дискідегі файлдан мәліметтерді оқу INPUT# және LINE INPUT# операторлары арқылы жүргізіледі. Олардың форматы қарапайым INPUT және LINE INPUT# операторлары мен

бірдей, мәліметтерді пернелер тақтасынан енгізгенде былай болады:

```
10  rem  файлды түзету;
20   rem  мәліметтер файлын оқу үшін оны ашу керек;
30  open "zarp" for input as #1
40   rem x арқылы цикл;
50   for x=1 to k
60  input #1, T1(x), T2$(x), T3$(x), T4$(x), T5(x), T6(x)
70 next x
80 rem файлды жабу.
```

LINE INPUT# операторы тек сөз тіркестерін ғана шығарады. Файлға енгізілетін мәліметтер бір-бірінен айыру таңбаларымен бөлінуы тиіс, олар төменде аталған операторлар тізімі арқылы орындалады:

CR -қатар соның тексеретін оператор;

(_) - бос орын қалдыратын сызықша;

EOF (файл нөмірі) − end of file “файл соңы” функциясы. Оның файл сонын тексеретін көрсеткіші болады; егер файлдың соңғы сөз тіркестерін шығару кезінде файл көрсеткіші ең сонында тұрса, EOF логикалық функцияның мәні true болады, онда бірді (1) қайтарады, ал егер файл соңына жетпесек, онда нөлді (0)- false қайтарады. Бағдарламаның келесі жолдарын жоспарлау үшін IF операторында EOF операторының мазмұны бағаланады.

Мәліметтерді түзету. Файлдың бір немесе бірнеше ескі жазуларын жою немесе өзгерту, орнына жаңа жазулар жазу үшін, екі файлды бірдей ашу керек: ескі файлды жанартып, жаңа файлды ескі файлдың өңделген жазуларымен толықтырамыз.

Қосымша жазу тәртібі. APPEND тәртібі OPEN операторында жаңа жазуларды немесе жаңа мәліметтерді файлға жазу үшін файл таңбасын файлдың соңына тасымалдайды. Сондықтан, PRINT# , PRINT# USING, WRITE# операторлар жаңа жазуларды файлдың соңына жазады, сонда ескі жазулар сақталып қалады. Мысалы: Берілген құрлым арқылы қызметкерлердің анкеталық мәліметтерін дайындау .

Берілген құрлымы арқылы, мекеме қызметкерлерінің аты-жөні мен айлықтарын көрсететін ақпаратты дайындау керек. Қызметкерлердің әрқайсысына арналған: табельдік нөмірі, қызметкерлердің аты-жөні, олардың жұмыс бөлімнің аты, атқаратын қызметі, алатын айлығы, атқарылған жұмыс күндерінің жалпы санына арналған өрістері арқылы бір жазуды құру қажет. Сонымен, Бейсик тілінде файлды құру - берілген мәліметтерді файлға жазу және осы файлды өңдеу бағдарламасына жазу. Ұсынуға берілген құжатты төменде келтірілген түрде баспаға шығару керек (7-кесте) :

Орындау кезеңі.

7-кесте

№	Аты-жөні	бөлімі	қызметі	айлығы	жұмыс күндерінің жалпы саны
1	2	3	4	5	6

Экран бетінің жоғарғы жағында кестенің басы орналасу керек, ал төменгі жағында - сұраныс пен көмекші қатар орналасуы тиіс.

Дискідегі файлдың белгісі - ZARP делік, жазулар тізімі шектеулі болуы қажет. Мысалы, ең үлкен саны − 30, ал ең кішісі − 1 болсын.

Сонымен, бір жазудың өрістері:

1-- − табель нөмірі;

2-Аты-жөні;

3-Бөлім;

4-Қызметі;

5-Атқарылған күндерінің жалпы саны.

Құжатты өңдеу ретін таңдау үшін, меню жүйесін жобалау керек. Меню жүйесінің негізгі қадамдары:

Мәліметтерді енгізу;

Түзету, өңдеу тәртібі;

Баспаға беру;

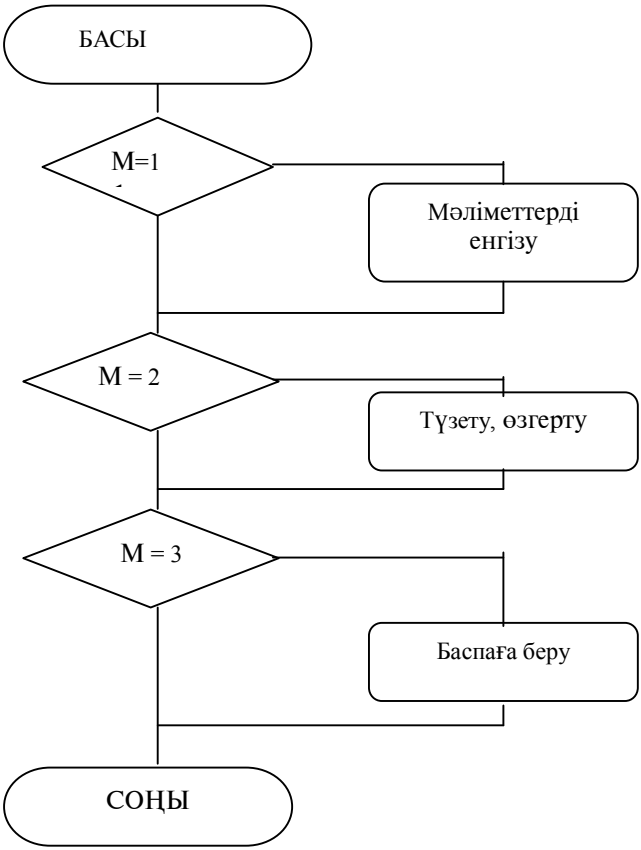
Шығу.

Бағдарлама жұмысын басқару үшін таңдалған тәртіптің нөмірі енгізіліп, жұмысы осы меню арқылы аяқталсын.

Өзгерген айлығын түзету жұмысын орындау үшін, ескі файлды ашып, керекті аты-жөнін енгізген соң, экранға ескі

айлығы шығуы керек, оның орнына жаңа айлық мөлшері енгізіледі.

Құрастырылған алгоритмнің блок-схемасы төменде сипатталған (1.11.1-сурет). Келтірілген блок-схема – меню тәртібін көрсетеді.



1.11-сурет

Бейсик тілінде жазылған бағдарлама

```
10  rem бағдарлама
20  rem
```

```
30  dim T1(30), T2$(30), T3$(30), T4$(30), T5(30), T6(30)
35  rem – экранды тазарту
36  cls
37  print "- - - - - -  - - - - - - - - - -  - - - - - - - - - - - "
38  input k
40  rem - - - - - - -  - - - -
50  print "- - - - :"
60  print "1 - - - - -  - - - - - - "
70  print "2 - - - - - - - - - - - - - - "
80  print "3 - - - - - - -  - - - - - - - - - "
85  print "4 - - - - - -  - - - - - - "
90  input answ
95  if answ = 4 then end
100 on answ gosub 1000, 2000, 3000
110 goto 40

1000 rem - - - -  - - - - - -
1030 rem - - - -  - -  x
1040 for x=1 to k
1050 print "- - - - - - -  - - - - - -  - - ";x;"- - - - - - - - - - :"
1070 input "- - - - - - - - - -  - - - - - :";T1(x)
1090 input "- - - - - - - :";T2$(x)
1110 input "- - - - - :";T3$(x)
1130 input "- - - - - - - - - :";T4$(x)
1160 input "- - - - - :";T5(x)
1180 input "- - - - - - - - - - - - -  - - - :";T6(x)
1183 next x
1185 rem - - - - - - - - -  - - - -  - - - - - - -  - - -  - - - - - -
1186 open "zarp" for output as #1
1190 rem - - - - - - - - -  - - - - - -  -  - - - - -
1195 for x=1 to k
1200 write #1, T1(x), T2$(x), T3$(x), T4$(x), T5(x), T6(x)
1210 next x
1220 rem - - - - - - - - - -  - - - -
1230 close #1
1240 return

2000 rem - - - - - - - - - - - - -
2010 rem - - - - - - - - -  - - - -  - - - - - - -  - - -  - - - - - -
2020 open "zarp" for input as #1
```

```
2030 rem - - - -  - -  x
2040 for x=1 to k
2050 input #1, T1(x), T2$(x), T3$(x), T4$(x), T5(x), T6(x)
2060 next x
2070 rem - - - - - - - - - -  - - - -
2080 close #1
2090 rem - - - - - - - -  - - - - - - - - - - - - - -
2100 rem - - - - - - - - - -  - - - -  - - -  - - - - - -
2110 open "zarp" for output as #1
2120      input      "- - - - - - -      - - - - - - -      - - -
- - - - - - - - - - - - - - ";fam$
2140 rem - - - -  - - - - - - - -  - - - - - - - -  T2$
2145 rem flag=0 - - -  - - - - - - - - , 1-- - - - - - -
2146  rem  N-- - - - -   - - - - - - - - - - -   -   - - - - - - - - - -
- - - - - - - -
2148 flag=0
2150 for x=1 to k
2160 if T2$(x) = fam$ then flag=1:N=x:goto 2180
2170 next x
2180 if flag=0 then print "- - - - - - -  - -  - - - - - - - ":goto
2230
2190 rem - - - - - - - - - - - - -
2195 print "- - - - - -  - - - - -  =";T5(N)
2210 input "- - - - - - -  - - - - -  - - - - - :";okl
2220 T5(N)=okl
2230 print "- - - - - -  - - -  - - - - - - - - - - - - - - -  (1-- - , 0-
- - - )? ";
2240 input answ
2250 if answ=1 then goto 2120
2260  rem  - - - - - - - - - -   - - - - - - - - - - -   - - - - - -   -
- - - - -
2270 for x=1 to k
2280 write #1, T1(x), T2$(x), T3$(x), T4$(x), T5(x), T6(x)
2290 next x
2300 rem - - - - - - - - - -  - - - -
2310 close #1
2320 return

3000 rem - - - - - -  - - - - - - - - - -
3010 rem - - - - - - - - - -  - - - -  - - -  - - - - - -
```

```
3020 open "zarp" for input as #1
3030 rem - - - -  - -  x
3040 for x=1 to k
3050 input #1, T1(x), T2$(x), T3$(x), T4$(x), T5(x), T6(x)
3060 next x
3070 rem - - - - - - - - -  - - - -
3080 close #1
3090 rem - - - - - - - -  - - - - - - - - -
3100 print tab(28); "- - - - - - - - - -  -  - - - - - - - - "
3105 rem - - - - - - - - - - - - -
3107 gosub 3500
3110 print "|- - - - - |";
3120 print tab(14);"- - - - - - - ";
3130 print tab(27);"|";tab(31);"- - - - - ";tab(48);"|";
3140 print tab(51);"- - - - - - - - - ";
3150 print tab(67);"|- - - - -   |- - - "
3154 rem - - - - - - - - - - - - -
3155 gosub 3500
3160 rem - - - - - - -  - - - - - -
3170 for x=1 to k
3188 print "|";tab(3);T1(x);tab(7);"|";
3190 print T2$(x);tab(27);"|";
3200 print T3$(x);tab(48);"|";
3210 print T4$(x);tab(67);"|";
3220 print T5(x);tab(76);"|";
3230 print T6(x)
3240 next x
3250 rem - - - - - - - - - - - - -
3260 gosub 3500
3270 return

3500 rem - - - - - - - - -  - - - - -
3505 for x=1 to 79
3510 print "-";
3520 next x
3530 print
3540 return
```

Дайын құжат.

Баспаға шығарылатын құжат келесі (8-кесте) түрінде болуы керек.

8-кесте

Нөмірі	**аты-жөні**	**бөлім**	**қызметі**	**айлығы**	**жұмыс күндерінің жалпы саны**
1	Б.Омаров	Кадр	Бастық	15000	25
2	А.Наум	Радист	Радист	8000	25

1.11.2. Тікелей қатынас жасайтын файл

Егер жауап алу және бір жазуды өңдеу уақытын (жазуды қосу, жою, түзету, іздестіру) қысқарту керек болса, онда тікелей қатынас жасайтын файлды пайдалану ұсынылады. Ондай файлды ұйымдастыру үшін тікелей қатынас жасайтын құрылғыны пайдаланады. Оларға – қатты диск (винчестер) немесе иілгіш дискеталар жатады.

Тікелей катынас жасайтын файл жазулар тізімнен тұрады және оларды әртүрлі ретпен өзіндік нөмірлері арқылы оқуға, түзетуге болады. Тізбек түріндегі жазулар бірден бастап нөмірленеді. Логикаға байланысты мәліметтер бір жазуда орналасады. Сонымен, дискіде жазуларды сақтау және оқу үшін оның нөмірі пайдаланылады.

1.11.3. Тікелей қатынас жасайтын файлды ашу

Файлды ашу операторы OPEN. Ол арқылы файлдың аты белгіленеді және онда белгіні бекіту мен жазулардың ұзындығы анықталады. Оператордың жазылуы:

OPEN <”файлдың аты”> AS # <файл нөмірі> LEN = <жазулар ұзындығы>

Барлық жазулардың ұзындық өлшемі байт арқылы көрсетіледі.

1.11.4. Файл буферінде айнымалылар тізімін анықтау

Барлық мәліметтерге FIELD операторы файлдың буферінде оның нөмірімен белгіленген, тізбек түрінде айнымалылар ұзындығына тең орын бөледі. Айнымалы белгісі (идентификаторы) символдық айнымалы немесе символдық массивтің элементі болуы керек. FIELD операторының жалпы ұзындығы - LEN операторымен анықталған жазудың жалпы ұзындығына тең болуы керек. Буферден мәліметтерді оқу үшін LSET және RSET операторлары пайдалынады.

Мысалы:

```
10 FIELD #1, 10 AS DSTN2$, 5 AS TIM$, 10 AS AIRC$, 3 AS FPL$
20 LSET DSTN1$ = DS1$
```

Тікелей қатынас жасайтын файлды құру бағдарламасы:

```
10 CLS
20 OPEN "RACES.dat" FOR RANDOM AS #1 LEN = 28
21 OPEN "DESTAN.DAT" FOR RANDOM AS #2 LEN = 10
22 FIELD #1, 10 AS DSTN2$, 5 AS TIM$, 10 AS AIRC$, 3 AS FPL$
23 FIELD #2, 10 AS DSTN1$
29 RACES = 10: DESTAN = 20
30 REM Menu
40 CLS : PRINT "1. - - - - - - - - - - - - - - "
50  PRINT  "2. - - - - -  - - - - - - - - - -  -  - - - - - -  - - - - - - - - "
60 PRINT "3. - - - - - - - - - - - - - - "
65 PRINT "4. - - - - - - - - - - - - - - - - - - - - - - "
70 PRINT "5. - - - - - - - - - - - - - - - - - - "
80 INPUT A
90 IF A = 1 OR A = 2 OR A = 3 OR A = 4 OR A = 5 THEN GOTO 100 ELSE GOTO 40
100 ON A GOTO 300, 500, 700, 900, 200
200 CLOSE #1: CLOSE #2: END
300 FOR I = 1 TO DESTAN
305 PRINT "- - - - - - - "; I; "--  - - - - -  - - - - - - - - - - - "; : INPUT DS1$
310 IF LEN(DS1$) > 10 THEN GOTO 305
315 LSET DSTN1$ = DS1$
320 PUT #2, I
```

```
     325 NEXT I
     330 FOR I = 1 TO RACES
     331 CLS
     335 PRINT "- - - - -  - - - - -  - "; I
     340 FOR S = 1 TO DESTAN
     345 GET #2, S
     350 DS1$ = DSTN1$
     351 PRINT S; ". "; DS1$
     355 NEXT S
     360 INPUT "- - - - - - - -  - - - - -  - - - - - - - - - - "; D
     365 IF D < 1 OR D > DESTAN THEN GOTO 360
     370 INPUT "- - - - - - -  - - - - -  - - - - - - "; TI$
     375 IF LEN(TI$) > 5 THEN GOTO 370
     380 INPUT "- - - - - - -  - - -  - - "; AI$
     385 IF LEN(AI$) > 10 THEN GOTO 380
     390  INPUT  "- - - - - - -  - - - - - - - - -  - - - - - - - - -
- - - - "; FP$
     395 IF LEN(FP$) > 3 THEN GOTO 390
     400 GET #2, D
     405 DS1$ = DSTN1$
     410 LSET DSTN2$ = DS1$
     415 LSET TIM$ = TI$
     420 LSET AIRC$ = AI$
     425 LSET FPL$ = FP$
     430 PUT #1, I
     435 NEXT I
     440 GOTO 40

     500 CLS
     505    INPUT    "- - - - - - -  - - - - - - - -  - - - - - -
- - - - - - - - - - "; DS$
     510 IF LEN(DS1$) > 10 THEN GOTO 505
     511 CLS
     515  PRINT  "- - - - -  - - - - - - - - - -  -  - - - - -
- - - - -  - - - - - -  - - -  - -  - - . - - - - - "
     520 PRINT "----------------  -------  ------------  ------
---------"
     521 Q = 2
     525 FOR I = 1 TO RACES
     530 GET #1, I
```

```
535 IF RTRIM$(DSTN2$) = DS$ THEN Q = Q + 1: DS1$ = DSTN2$: TI$ = TIM$: AI$ = AIRC$: FP$ = FPL$: W = I: E = Q: GOSUB 590
540 NEXT I
550 IF INKEY$ = "" THEN GOTO 550 ELSE GOTO 40

590 LOCATE E, 1: PRINT DS1$
591 LOCATE E, 24: PRINT W
592 LOCATE E, 33: PRINT TI$
593 LOCATE E, 45: PRINT AI$
594 LOCATE E, 57: PRINT FP$
595 RETURN

700 INPUT "- - - - - - -  - - - - - - - - - - "; D
705 IF D < 1 OR D > 10 THEN GOTO 700
706 CLS
710 PRINT "- - - - -   - - - - - - - - - - -        -    - - - - - - - - - -  - - - - - -    - - -  - -    - - . - - - - - "
715 PRINT "----------------   -------   ------------   ---------------"
720 GET #1, D
725 DS1$ = DSTN2$: TI$ = TIM$: AI$ = AIRC$: FP$ = FPL$: W = D:   E = 3: GOSUB 590
730 IF INKEY$ = "" THEN GOTO 550 ELSE GOTO 40

900 CLS
905 INPUT "- - - - - - -  - - - - - - - - - - - "; R
910 IF R < 1 OR R > RACES THEN GOTO 900
915 INPUT "- - - - - - -  - - - - -  - - - - -  - - - - - -  - - - - - - - - "; TI$
920 IF LEN(TI$) > 5 THEN GOTO 915
925 LSET TIM$ = TI$
930 PUT #1, R
935 GOTO 40
```

Бейсик тілінің түйінді сөздері

1-ҚОСЫМША

Ағылшынша	Қазақша
AND	және
AUTO	автоматты түрде
CLOSE	жабу
CLS	экранды тазалау
CIRCLE	шеңбер
CONinue	жалғастыру
DATA	берілген мәні, мәлімет
DEFine FuNction	функцияны анықтау
DELETE	өшіру, жою
ELSE	әйтпесе
END	соңы
FILES	файлдар
FOR	үшін
GOTO	көшу
GOSUB	бағынышты бағдарламаға көшу
IF	егер
INPUT	енгізу
KEY	перне
KEYBOARD	пернелер тақтасы
KILL	жою, құрту
LET	айталық
LINE	қатар, жол, сызық
LIST	тізім
LOAD	жүктеу
LOCATE	орналастыру
NEW	жаңа
NEXT	келесі
OPEN	ашу
OR	немесе

Ағылшынша	Қазақша
PRINT	басып алу
PRINT USINC	нұсқамен басып алу
READ	оқу
REM	түсінік беру
RESTORE	қалпына келтіру
RETURN	кері қайту
RENUBer	қайта нөмірлеу
RUN	орындау
SAVE	сақтау
STEP	қадам, адым
STOP	тоқтау
STOP AT LINE N	N қатарында тоқтау
STRING	Сөз тіркесі
THEN	Онда
TO	Дейін, шейін,
WHILE	әзірше

2-ҚОСЫМША

Qbasic ортасында бағдарлама құру, өңдеу жолдары

Енді **Qbasic** бағдарлама құру ортасымен танысамыз. Оны іске қосу үшін қатты дискіден **Qbasic.exe** бағдарламасын тауып, іске қосамыз. Qbasic ортасы жүктелген кезде, экранға оның терезесі шығады. Терезе көрінісіне тоқталайық. Бірінші қатарда бағдарлама аты жазылады, екінші қатарда оның менюі орналасқан, үшінші қатардан бастап, тік төрт бұрышты, бағдарлама теруге арналған жұмыс аймағы орналасады. Оның астыңғы қатарында ***Immtdiate*** (шұғыл орындау) терезесі бар. Ол терезеге көшу үшін, *F6* функционалды пернесін басу керек. Терезеге енгізілген командалар ***Enter*** пернесін басқан кезде атқарылады (интерактивті режим).

Курсор бағдарлама теруге арналған жұмыс аймағында орналасқан. Жаңа файыл аты Untitled (аты жоқ) – деп аталады. Бағдарлама теру жұмысын бастау үшін **Esc** пернесін басу керек.

Енді **Qbasic** меню құрылымын қарастырайық. Бұл меню – *жоғарғы меню* - деп аталады. Оны іске қосу үшін ALT пернесін басамыз. Жоғарғы меню келесі пункттерден тұрады:

- **File** – файылмен жұмыс істеуге арналған командалар тобы:

New –жаңа бағдарлама құру;
Open... –құрылған бағдарламаны ашу;
Save – бағдарламаны дискідегі файылға сақтау;
Save as – бағдарламаны дискідегі файылға басқа атпен сақтау;
Print... -бағдарламаны баспаға шығару;
Exit – бағдарламадан шығу (жұмысты аяқтау);

- Edit
Cut – белгіленген фрагментті буферге қиып алу;
Copy – белгіленген фрагментті буферге көшіру;
Paste –буфердегі фрагментті қою;
Clear – белгіленген фрагментті жою (Delete пернесімен орындауға болады);

- Viev
Output Screen – нәтиже орналасатын терезені көрсету, оны (F4 пернесімен орындауға болады);

- **Starch**
Find – фрагмантті табу;
Change – табу және ауыстыру;

- **Run**
Start – бағдарламаны орындау (Shift+F5 пернелерін басу арқылы орындауға болады);
Continue – бағдарламаны орындауды жалғастыру (F5 пернесімен);
- Debug – бағдарламаны орындау кезінде трассировка тәртібі мен бақылау нүктелерін орналастыру
- Options – экран параметрлерін орнату т.б.

Төменде файылдармен жұмыс істеу тәсілдерін қарастырамыз.
1. Жаңа файл құру үшін менюден: File-New таңдаймыз. Мысал: Тік бұрышты үшбұрыштың ауданын есептеу бағдарламасын құру керек болсын.

Бағдарламасы:

REM үшбұрыштың ауданын есептеу

```
CLS
INPUT « a, b катеттерінің ұзындығың;a,b
S=a*b/2
print  «Үшбұрыш ауданы S=ң;S
END
```

Жауабын алу үшін төменде аталған іс-әрекеттерді орындаймыз:

- Бағдарлама мәтінін пернелер тақтасынан енгіземіз;
- Бағдарламаны файылға сақтаймыз;
- Бағдарламаны орындаймыз.

2. Бағдарламаны сақтау үшін менюден: File-Save таңдаймыз.

3. Бағдарламаны орындау үшін менюден: File-Run-Start таңдап немесе F5 пернесін басамыз.

4. Жұмысты аяқтау үшін менюден: File-Exit таңдаймыз.

ӘДЕБИЕТТЕР ТІЗІМІ

1. *Петров В.А., Алексеев В.Е.*, и *др.* Вычислительная техника и програмирование.-М.:Высшая школа, 1990.

2. *Алексеев В.Е.*, и *др.* Вычислительная техника и програмирование. -М.: Высшая школа, 1991.

3. *Светозарова Г.И., Мельников А.А.*, и *др.* Практикум по программированию на языке Бейсик.-М.:Наука.1988.

4. *Айдосов Г.А.* Есептеу техникасы және ЭЕМ-де программалау. -Алматы: Мектеп, 1995.

5. *Симонович С., Евсеев Г.* Практическая информатика.-М,: АСТпресс, 2001.

МАЗМҰНЫ

Оқулық басылым

Нұрпейісова Төлеужан Байболқызы

Бағдарламалау тілдерінің негізі

Оқу құралы

РБ меңгерушісі *З.А. Ғұбайдулина*
Редакторы ***К. Мүптекеқызы***
Техн.редакторы *Ж.К.Еланова*
Компьютерде беттеген ***А.Н.Оразалиева***

Басуға қол қойылды “ “ 2003 ж.

Таралымы дана. Пішімі 60x84 1/16. N 1 баспаханалық қағаз. Көлемі 5 баспа-табақ. Тапсырыс N . Бағасы келісімді.

□азҰТУ-дың Баспа орталығы, Ладыгин көшесі, 32