ҚАЗАҚСТАН РЕСПУБЛИКАСЫ
БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Б.Б.Бөрібаев, А.М.Махметова

АЛГОРИТМДЕУ ЖӘНЕ ПРОГРАММАЛАУ ТІЛДЕРІ

Оқулық

Қазақстан Республикасы
Білім және ғылым министрлігі бекіткен

Алматы, 2011

ӘОЖ 004(07)
КБЖ 32.973я7
Б 79

Пікір жазғандар:

техника ғылымдарының докторы, профессор **А. Ж. Сейкетов**
физика-математика ғылымдарының кандидаты **С. Б. Беркімбаева**
педагогика ғылымдарының кандидаты **Р. М. Дузбаева**

Бөрібаев Б. Б., Махметова А. М.

Б 79 **Алгоритмдеу және программалау тілдері:** Оқулық. – Алматы: ЖШС РПБК «Дәуір», 2011. – 328 бет.

ISBN 978-601-217-274-4

Ұсынылып отырған оқулық негізгі мақсаты – студенттерді алгоритмдерді құрастыру мен қазіргі кезде оқу орындарында кең таралған Паскаль программалау тілін үйрету болып табылады. Оқу құралы алгоритмді құрастырудың графикалық тәсілдерін қарастырып, программалау тіліндегі енгізу/шығару, басқару операторларын және процедуралар мен функцияларды толық қамтиды. Сонымен бірге программаны енгізу, түзету және нәтижесін алу мүмкіндіктерін толық ашатын ортаменен тиянақты түрде таныстырады. Бұл оқулықта студенттердің программалауды меңгеру ісін жүзеге асыратын мүмкіндіктердің бәрі қарастырылған.

Оқулық жоғары оқу орындарының студенттеріне арналып жазылған, дегенмен оны колледждер мен мектеп оқушылары, өз бетімен ЭЕМ-да жұмыс істеп үйренушілер де пайдалана алады.

ӘОЖ 004(07)
КБЖ 32.973я7

ISBN 978-601-217-274-4

1. АЛГОРИТМДЕУ НЕГІЗДЕРІ

1.1. Негізгі ұғымдар мен түсініктер

ЭЕМ-ді пайдалану істерін қарастырмас бұрын оның жұмысымен тығыз байланысты алгоритм, программа ұғымдарын білуіміз қажет. Әрбір ЭЕМ алдын ала берілген алгоритммен, яғни жоспармен жұмыс істейді. Алгоритмді заңдылық, реттелген амалдар жиыны, кезекпен орындалатын операциялар тізімі деп ұғынған жөн. Бұл ұғым қазіргі кезде кеңінен қолданылып жүр. Оның көптеген анықтамалары да бар. Соның бірін келтіре кетейік.

Алгоритм – *берілген есептің шығару жолын реттелген амалдар тізбегі түріне келтіру*. Кез келген есепті қарапайым амалдарды тізбектей орындау арқылы шығаруға болады. Алгоритмді ЭЕМ-де орындау үшін оны программа түрінде жазып шығу керек.

Сонымен, алгоритм оны атқарушы ЭЕМ-ге жұмыс тәртібін түсіндіретін ережелер мен нұсқаулар тізбегінен тұрады. Алгоритмді атқарушының рөлін негізінен адам немесе автоматтандырылған аспап, яғни ЭЕМ, робот, т.б. атқарады. Мысалы, $y=(ax+b)(cx-d)$ функциясын есептеу төменгі іс-әрекеттерден тұрады:

1) a-ны x-ке көбейту, оны R_1 деп белгілеу;

2) оған b-ны қосу, нәтижесін R_2 деп белгілеу;

3) c-ны x-ке көбейту, оны R_3 деп белгілеу;

4) одан d-ны алу, оны R_4 деп белгілеу;

5) R_2-ні R_4-ке көбейту, оны y деп белгілеу.

Алгоритмнің орындалу кезінде оны орындаушыға келесі жолы қандай нұсқау бойынша орындалатыны белгілі болуы қажет. Ал орындаушының жүзеге асыра алатын командалар жиыны – командалар жүйесін құрайды.

Алгоритм мен программаға байланысты ЭЕМ-нің мынадай жұмыс ерекшеліктері болады:

1) есепті шығару жолы алгоритм түрінде өрнектелуі қажет;

2) алгоритм программаға айналдырылуы тиіс;

3) программа машина жадына енгізіліп, ретімен орындалуы керек. Алгоритм күнделікті тұрмыста да кеңінен қолданылады. Мысалы, студент болу үшін алгоритмнің мынадай қадамдарын орындау керек.

1. Орта мектепті бітіріп, аттестат алу.
2. Керекті құжаттарды аттестаттың түпнұсқасымен бірге белгілі бір оқу орнына өткізу.
3. Конкурстан өту.

Бұл көрсетілген пункттердің орнын ауыстыруға болмайды. Олар көрсетілген ретпен кезектесіп орындалуы тиіс. Сонда ғана керекті нәтижеге (студент болу) қолымыз жетеді.

Алгоритм информатиканың іргелі ұғымдарының бірі. Квадрат теңдеудің түбірін табу ережесі, үшбұрыштың ауданын есептеу жолдары алгоритмдердің мысалдары болып табылады.

Алгоритмдеу – *есепті шығару алгоритмін құрастыру процесі*, мұның нәтижесінде мәліметтерді өңдеу процесінің кезеңдері айқындалады да, кезеңдер мазмұны формальды (жасанды) түрде жазылып, солардың орындалу реттілігі анықталады.

Алгоритмдік тіл – алгоритмдерді жазуға арналған символдар мен сол символдардан тұратын конструкцияларды құрастыру және түсіндіру ережелерінің жиыны.

Программалау тілі ЭЕМ-дерде программаларды орындау ісін атқарады.

Программа – *алгоритмді машинаға түсінікті нұсқаулар тізімі ретінде жазу.* Программада берілген мәліметтердің сипаттамаларымен бірге оларды өңдейтін командалар болады. Осы командалар тізбегі орындалу барысында есептің нәтижесі шығады. Командалар қандай мәліметтер қандай операцияларға қатынасатынын, олар қандай реттілікпен орындалатынын және нәтиженің қандай түрде шығарылатынын көрсетеді. Бұлар операторлар арқылы жүзеге асырылады.

Әрбір ЭЕМ алдын ала жазылған программамен істейді. Процессор программаның құрамындағы командаларды кезекпен орындап отырады. Командалар тізбегін программа деп қарастыруға болады. Команда бір ғана қарапайым амалды орындау үшін берілген бұйрық ретінде беріледі. Командалар: арифметикалық немесе логикалық амал; ақпаратты тасымалдау командасы;

берілген сандарды салыстыру командасы; келесі командаларға көшу тәртібін орындау, т.с.с.

Алгоритм және программа ұғымдары ұқсас екені көрініп тұр, алгоритм есептің шығару жолын қарапайым әрекеттер тізбегімен өрнектесе, программа сол өрнекті машинаға түсінікті тілмен жазып береді. Сондықтан программа жазу тәсілдері солардың қандай ережені пайдаланып, қандай компьютерлерге арналып жазылғанына байланысты алгоритмдік тілдер немесе программалау тілдері ұғымына келіп тіреледі.

Мәліметтер – белгілі бір процесс көмегімен тасымалдап, өңдеуге болатын, формальды түрде бейнеленген фактілер мен идеялар.

Оператор – операциялар мен мәндерді көрсететін немесе солардың элементтерінің қай жерде орналасқанын білдіретін символдар жиыны. Мысалы:

```
A:= B+C;    {A, B, C – айнымалы;}
K:= 2;      IF T<0 THEN ...
```

Айнымалы – программа орындалуы барысында өз мәнін өзгерте алатын объект.

Айнымалы қасиеттері:

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

Бұл пәннің заттық негізі болып (ЭЕМ-де шығару мақсатында) алгоритмдер мен программаларды құрастыру тәсілдері мен құралдары саналады. Программалар құру үшін программалау жүйелері пайдаланылады.

Программалау жүйесі – программалауды автоматтандыру құралдары. Олар программалау тілінен, осы тілдің трансляторы-

нан, құжаттамаларынан және де программаларды дайындау, әрі орындау құралдарынан тұрады.

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

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

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

1.2. Алгоритм қасиеттері

Алгоритм ұғымының мәнін ашатын негізгі қасиеттерінен немесе оған қойылатын талаптардан қысқаша мағлұматтар келтірейік. ЭЕМ-де орындалуға тиіс алгоритмдерге мынадай талаптар қойылады:

1) ол анық әрі дәл өрнектелуі тиіс – детерминділік (анықтылық, бір мәнділік) қасиеті, яғни алгоритмді басқаша түсінуге жол бермей, тек қана көрсетілген әрекеттерді айқын түрде орындауға арналған нұсқаулар дәлдігі;

2) алгоритм шектелген уақыттан соң нәтиже беруі тиіс – нәтижелілік қасиеті, мұнда белгілі бір әрекеттер саны атқарылған соң, процестің қажетті нәтижесін алып, оны аяқтау мүмкіндігінің болуы немесе есептеу процесін ары қарай жалғастыруға болмайтындығы жайлы мәлімет алуымыз қажет;

3) бір тектес есептерге жалпы бір ғана алгоритм қолданылуы тиіс – жалпылық қасиеті, ол алгоритмнің осы сияқты көптеген басқа да есептерге қолданылу мүмкіндігінің болуын көрсетеді;

4) алгоритмді кішкене бөліктерге бөлу мүмкіндігі болуы қажет – модульдік (дискреттілік, бөліктік) қасиет – есептеу процесін жекеленген қарапайым операцияларға бөлу мүмкіндігінің болуы, яғни күрделі есепті атқарылуына күдік келтіруге болмайтын шағын бөліктерге жіктеу орындалуы тиіс.

Біріншіден, алгоритм анық, әрі *дәл өрнектелуі* қажет. Онда қандай қадамдар көрсетілсе, тек соны ғана орындау керек. Есеп шығару жолына керектің бәрі біржақты анықталуы және орындаушыға түсінікті, әрі нақты болуы тиіс. Екіншіден, алгоритм *нәтижелі болуы* керек. Әрекеттердің шектелген санынан

кейін белгілі бір уақыт ішінде қорытынды нәтиже алуымыз қажет. Әрбір алгоритм біршама бастапқы мәліметтердің болуын талап етеді және іздеген нәтижені алуға жеткізеді. Мысалы, сандарды қосу алгоритмі үшін бастапқы мәліметтерге қосылғыштар мәні жатады, ал нәтижесі қосынды болады. Үшіншіден, алгоритмнің *жалпылық қасиеті* болады, яғни бастапқы мәліметтер мәнінің бір жиыны бір ғана нәтиже береді. Егер берілген мәліметтер өзгерсе, нәтиже де өзгереді. Басқаша айтқанда, бір алгоритм бір типтес есептердің әр түрлі алғашқы мәліметтері үшін әр түрлі нәтижелер беруі тиіс. Мысалы, квадрат теңдеуді шешу алгоритмі кез келген *a*, *b*, *c* мәндері үшін оның түбірін дұрыс табуы керек. Төртіншіден, алгоритмнің *үзік-үзік модульдерге бөліну* қасиеті болуы тиіс, яғни үлкен алгоритмді бірнеше кішкене алгоритмдерге жіктеуге әрқашанда мүмкіншілік болуы керек. Сондықтан алгоритмді екі-үш бөлікке бөліп, оларды өзінше құра алатын дәрежеде жұмыс істелуі қажет. Олар тек бірінің қорытындысын келесі жолы керекті мәлімет ретінде қолдануы тиіс.

1.3. Алгоритмдерді бейнелеу жолдары

Алгоритмдерді ЭЕМ-де орындау үшін оларды алдын ала жазып алу керек, яғни ол белгілі бір заңдылықпен өрнектелуі тиіс. Жалпы алгоритмді жазып өрнектеу, яғни оларды бейнелеу түрлеріне төмендегі тәсілдер жатады:

- табиғи тіл сөздері арқылы;
- формулалық-сөздік тәсіл арқылы;
- графикалық түрде бейнелейтін блок-схемалар арқылы;
- псевдокодтар арқылы;
- құрылымдық диаграммалар арқылы;
- программалау тілі арқылы.

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

Осы тәсілмен келесі мысалдың алгоритмін жазып шығайық. Сандар жиымы (массиві) берілген делік. Осы жиым сандарының көрсетілген аралықта, яғни интервалда толығынан жататынын/жатпайтынын тексеру керек. Интервал өзінің шекаралық А және В мәндерімен берілген.

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

Бұл тәсілде көрнекілік жоқ, яғни толық формальдау мүмкіндігі жоқ. Жалпы алгоритмді табиғи тілде өрнектеу ЭЕМ-дерде қолданылмайды, өйткені онда дәлдік, нақтылық болмайды.

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

Мысалы, үшбұрыш ауданын оның үш қабырғасының ұзындығы арқылы есептеу алгоритмін құру керек болсын делік.

1. – үшбұрыштың жарты периметрін есептеу

$$p=(a+b+c)/2$$

2. – үшбұрыштың ауданын есептеу

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

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

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

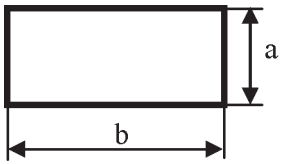
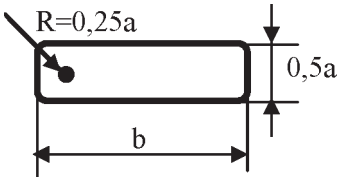
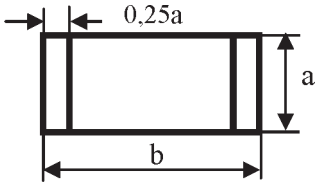
Алгоритмді **графикалық түрде блок-схемалар арқылы** көрсету – оның логикалық құрылымын графикалық түрде бейнелеу болып саналады. Бұл – алгоритмдерді өрнектеудің ең көп тараған түрі. Мұнда мәліметтерді өңдеудің әрбір кезеңі атқарылатын операцияға сәйкес әр түрлі геометриялық фигуралар (блоктар) түрінде көрсетіледі.

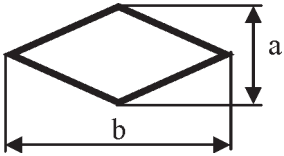
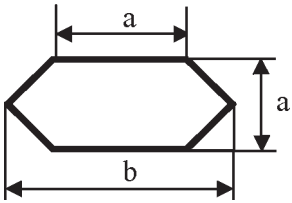
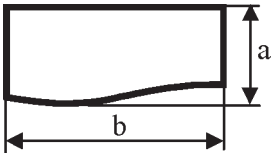
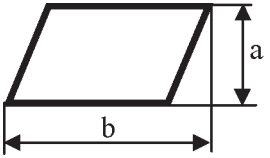
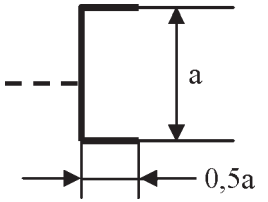
Әр блоктың ішіне орындалатын іс-әрекеттің (амалдың) мазмұны жазылады. Символдардың (блоктардың) бір кіру және бір шығу сызықтары болуға тиіс.

Графикалық жолмен алгоритмдерді жазу үшін мемлекеттік стандарт белгіленген, онда кез келген амал белгілі бір геометриялық фигурамен өрнектеледі. Ол фигуралар немесе блоктар амалдар символы деп те аталады. Блоктар бағытталған сызықтармен байланысып, бірінен соң бірі орналасады. Жиі қолданылатын амалдар, яғни мәліметтерді ЭЕМ-ге енгізу, формуламен есептеу, шарттардың орындалуын тексеру, нәтижені қағазға басу символдары 1-кестеде көрсетілген. Осы суреттегі көрсетілген блоктардан (символдардан) алгоритм схемалары құрастырылады. Алгоритмдер схемасымен ақпаратты өңдеудің әрбір сатысы немесе орындалатын операциялар реті анықталады. Кейде алгоритмдер схемасын оның блок-схемасы деп те атайды.

1-кесте

Алгоритмдерді бейнелеу блоктары

Іс-әрекеттің аты	**Блоктың пішімі**	**Атқаратын жұмысы**
Процесс		Математикалық өрнектерді есептеу
Басы – соңы		Алгоритмдерді бастау, аяқтау
Алдын ала анықталған процесс (подпрограмма)		Қосалқы программаларға кіру және шығу

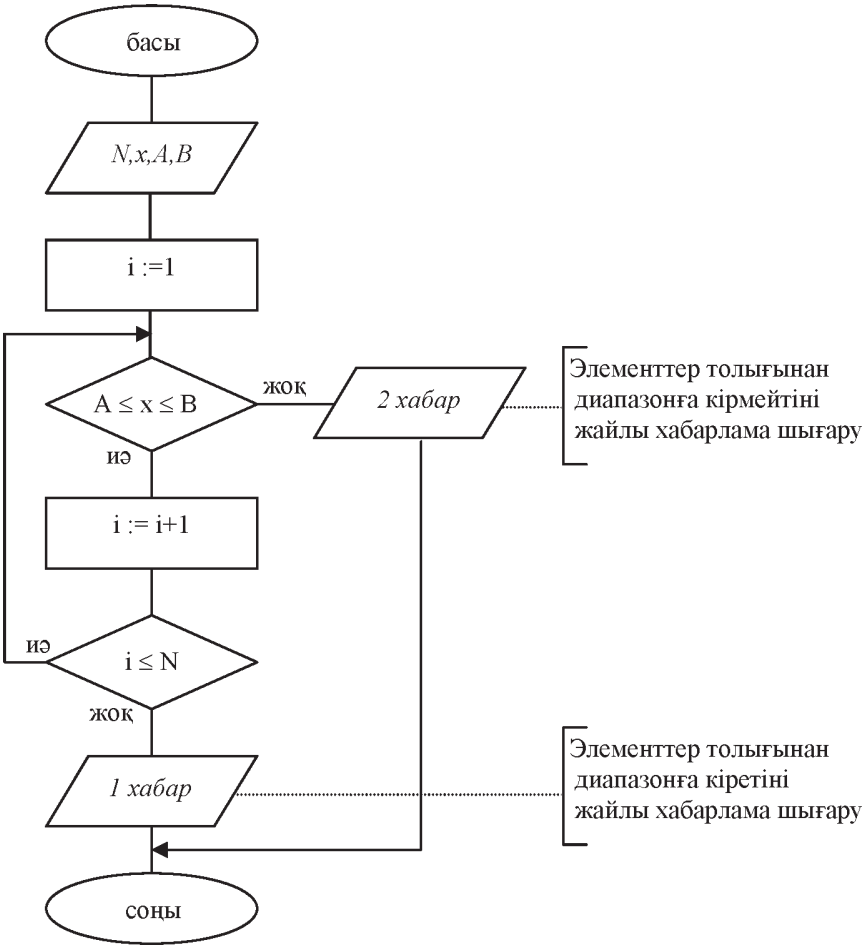
Шешім		Есеп шығару жолын таңдау
Модификация		Цикл басы
Құжат		Нәтижені баспаға (қағазға) шығару
Енгізу-шығару		Мәліметтерді енгізу және шығару
Түсініктеме		Схеманы, формулаларды түсіндіру

Сонымен алгоритм блоктармен немесе геометриялық көпбұрыштар түріндегі фигуралармен өрнектеледі.

Блок-схемалар дәстүрлі (кәдімгі) және құрылымды болып екіге бөлінеді.

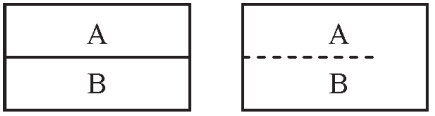
Табиғи тіл сөздері арқылы көрсетілген алгоритмдегі мысалды блок-схемалар арқылы бейнелеу төмендегі суретте көрсетілген. Онда берілген жиым сандарының көрсетілген аралықта,

яғни интервалда толығынан жататынын/жатпайтынын тексеру керек болатын. Интервал өзінің шекаралық А және В мәндерімен берілген.

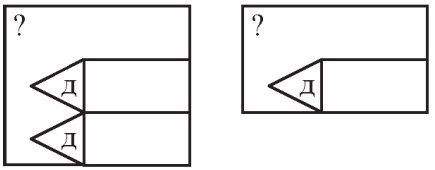


Құрылымдық диаграммалар – модульдер арасындағы байланысты көрсететін, мәліметтер құрылымын, программаларды және мәліметтерді өңдеу жүйелерін бейнелейтін құрылымдық блок-схемалар ретінде пайдаланылады. Құрылымдық диаграммалардың бірнеше түрлері болады: Насси-Шнейдерман диаграммасы, Варнье диаграммасы, Джексон диаграммасы, МЭСИД және т.б. диаграммалары.

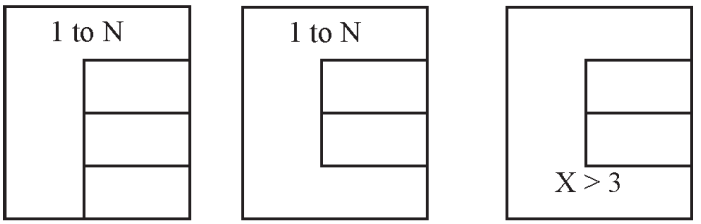
МЭСИД диаграммасының негізгі элементтері



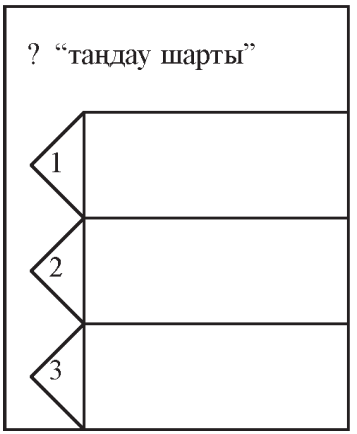
– сызықтық алгоритм



– тармақты алгоритм



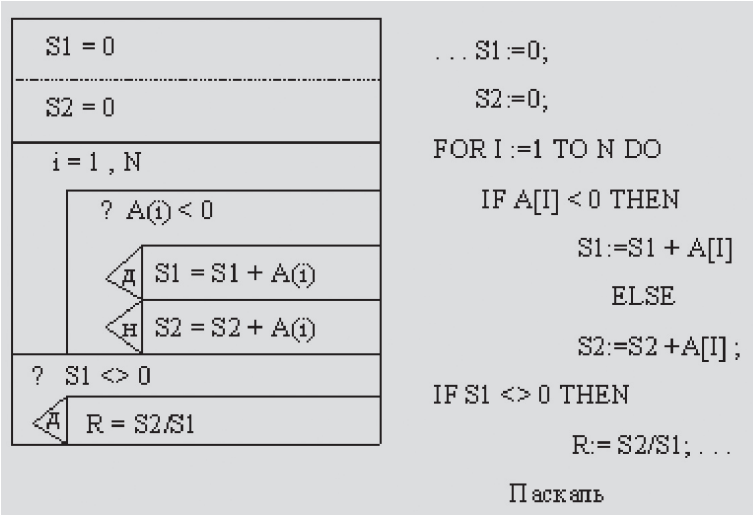
– қайталау



– таңдау

МЭСИД диаграммасын пайдалануға бір мысал келтірейік.

Оң және теріс сандардан тұратын бір өлшемді жиым берілген. Осы жиымдағы оң элементтердің қосындысын оның теріс элементтерінің қосындысына бөлгендегі бөліндіні анықтау қажет. Төменде осы есептің диаграммасы мен соған сәйкес Паскаль тілінің операторлары оның оң жақ бөлігінде келтірілген.



Программалау тілдері – программаларды ЭЕМ-де тікелей орындауға арналған алгоритмдерді жазу тәсілі. Программа – алгоритмнің ЭЕМ-ге түсінікті түрде жазылуы.

Әрбір машинаның өз тілі (машиналық тіл) болады және ол тек осы тілдегі программаларды, яғни командалар тізбегін орындай алады. Программаларды машиналық тілде жазу өте күрделі, әрі адамды шаршататын жұмыс болып табылады. Программалаушылардың жұмыс өнімділігін арттыру мақсатында жасанды тілдер, яғни программалау тілдері қолданылады. Мұндайда жасанды тілде жазылған программа машиналық тілге аударылуы тиіс. Аудару жұмысын, яғни программаны бір тілден екінші тілге түрлендіру ісін транслятор атқарады. Жиі қолданылатын, тікелей интерпретация жасайтын транслятор түріне Бейсик тілінің трансляторы жатады, ол командаларды аударады да, оны бірден орындайды. Мұндай транслятор жұмысының қорытындысы қажетті нәтижелерді алу болып табылады.

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

Алгоритмдік, яғни программалау тілдері есептерді шығару жолын баяндау-өрнектеу үлгісі, белгілі бір проблеманы шешу негізінде орындалатын әрекеттерге басшылық, ой еңбегін үнемдеуге мүмкіндік беретін әдіс, есеп шешімін табуды автоматтандыруға қажетті іс-әрекет, жаңа проблеманы шешу кезінде қолданылатын тәсілдер, күрделі процестерді өрнектеу және математикалық дәлдікпен анық етіп жазу құралы бола алады.

1.4. ЭЕМ-де есеп шығару кезеңдері

ЭЕМ-де есеп шығару күрделі процесс болып есептеледі, ол төмендегі кезеңдерден тұрады:

1. Берілген есепті математикалық түрде өрнектеу, яғни есепті мәселе ретінде қоя білу.

2. Есепті шығарудың ЭЕМ-ге ыңғайлы сандық тәсілдерін анықтау.

3. Есепті шығару жолын алгоритм түрінде бейнелеу.

4. Есепті ЭЕМ-де шығару программасын жасау және оның қателерін түзету.

5. Есепке керекті мәліметтер дайындау.

6. ЭЕМ-де есепті шығару және шыққан нәтижені іс жүзінде қолдану.

Берілген есепті математикалық түрде өрнектеу дегеніміз – есептің берілген мәндерін математикалық таңбаларды қолданып жаза білу және керекті математикалық формулаларды, өрнектерді анықтау болып саналады.

Күрделі формулаларды, теңдеулерді арифметикалық амалдар тізбегіне айналдыру есепті шығарудың сандық тәсілдерін табу не анықтау жолы болып есептеледі. Қазіргі кезде барлық есептердің шығару жолының сандық тәсілдері белгілі десе де болады, тек солардың ішінен өзімізге тиімді жолын таңдап алуымыз керек. Бұл мақсатта есепті шығару дәлдігін, нәтижені жылдам табу мүмкіндігін, мәліметтерді дайындау мен есепті шығарудың бағасын салыстыра отырып қарастыру қажет.

Есептің алгоритмін жасағанда, оның шығару жолы тізбектелген іс-әрекеттер ретінде схема түрінде өрнектеледі.

Программа жасағанда қазірде кеңінен тараған программалау тілінің бірінде алгоритм нақты түрде жазылады. Бізде кең

тараған тілдерге - Паскаль, Дельфи, Си жатады. Жазылған программаның қатесін түзету ЭЕМ-нің көмегімен шешіледі, өйткені жіберілген қателерді тек ЭЕМ ғана жылдам аңғарып, түзету мүмкіндігін береді.

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

1.5. Алгоритмдерді график түрінде жазу

Алгоритмдер блоктардың өзара байланысуына қарай үш түрлі бірыңғай құрылымға – сызықтық, тармақтық және циклдік болып үш топқа бөлінеді. Енді соларды қарастырайық.

1.6. Алгоритмдердің бірыңғай құрылымы

Күрделі алгоритмдерді құру үшін қарапайым бірыңғайланған алгоритмдік құрылымдар қолданылады. Олар сызықтық, тармақталу және цикл құрылымдарынан тұрады (2-кесте).

Программалау теориясында кез келген күрделі программаны үш түрлі құрылымнан құрастыруға болатыны дәлелденген, олар: *сызықтық*, *тармақты* және *циклдік* құрылымдар. Осы үшеуі құрылымдық программалаудың негізгі конструкциялары, яғни құраушылары болып саналады.

Сызықтық құрылым бірінен кейін бірі орындалып тізбектеле орналасқан бірнеше операторлардан тұрады.

Тармақты – шартқа байланысты екі оператордың бірінің орындалуы

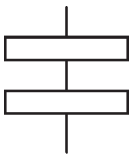
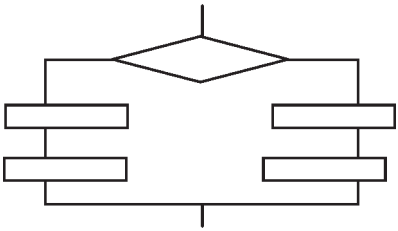
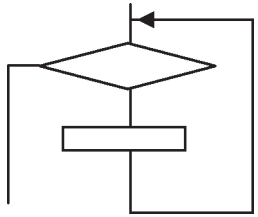
Цикл – операторлар бөлігінің бірнеше рет қайталана орындалуы.

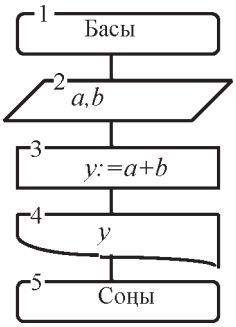
Оператор – тілдің қарапайым сөйлемі, ол белгілі бір әрекет немесе амал орындап, «**;**» таңбасымен аяқталады.

Негізгі конструкцияларды пайдалану мақсаты – қарапайым құрылымды программалау болып саналады. Мұндай программалар оңай оқылады, түзетіледі және керек болса, оңай өзгертіледі. Құрылымдық программалауда `goto` операторын қолдануға болмайды, өйткені ол программа логикасын түсінуді қиындатады. Бірақ кейде `goto` операторын қолдану қажет болатын кездер болады.

2-кесте

Алгоритмдердің бірыңғай құрылымдары

Сызықтық құрылым	Тармақты құрылым	Циклдік құрылым
		



1.1-сурет. Алгоритм схемасы

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

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

1. Сызықтық құрылымды алгоритм немесе қарапайым сызықтық алгоритм іс-әрекеттердің орындалу ретіне қарай тізбектеле орналасқан блоктардан тұрады. Амалдардың бұлай бірінен соң бірі реттеліп орындалу тәртібін табиғи атқарылу дейді.

Мысалы, $y= a+b$ формуласы бойынша есептеу тіктөртбұрыш арқылы кескінделетін есептеу блогы (3-блок) арқылы өрнектеледі. Ал нәтижені қағазға басу үшін көпбұрышты құжат алу блогын (4-блок) пайдаланып, оның ішіне нәтиженің атауларын жазамыз. Жоғарыда көрсетілген $y=a+b$ формуласымен есептеу үшін a және b-ның сандық мәндерін ЭЕМ-ге енгізіп (2-блок), содан кейін қосу амалын орындап, ақырында y-ті қағазға басып шығарып, жұмысты тоқтатамыз. Осы алгоритмнің схемасы 1.1-суретте көрсетілген.

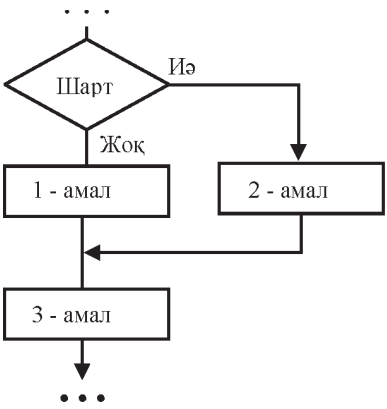
2. Тармақталу алгоритмдері. Тұрмыста кездесетін алгоритмдер әр түрлі болып келеді. Олардың жиі кездесетін түріне алгоритмнің белгілі бір шарттың орындалуына не орындалмауына байланысты тармақталып бірнеше жолдарға бөлінуі жатады.

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

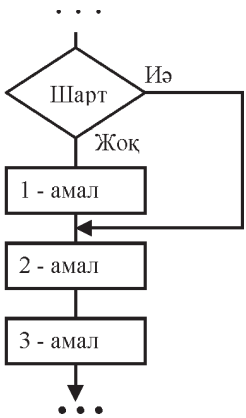
«Таңдау» жолымен тармақталуда берілген шарт тексеріледі (1.2-сурет), егер ол шарт орындалса (орындалуы ақиқат болса), онда 2-амал жүзеге асырылып, содан кейін келесі 3-амалға көшеміз. Ал, егер де шарт орындалмаса, яғни оның орындалу мүмкіндігі жалған болса, онда 1-амал атқарылып, содан кейін 3-амал атқарылады. Сонымен, шарттың ақиқат немесе жалған болуына байланысты 1-амал немесе 2-амал орындалады.

«Аттап өту» (1.3-сурет) алгоритмінде шарт орындалса, 1-амалды аттап өтіп, бірден 2-амалды, содан кейін 3-амалды орындаймыз. Ал шарт жалған болса, онда 1-амал міндетті түрде орындалып, одан кейін 2- және 3-амалдар жүзеге асырылады. Тармақталу кезеңінде шартты тексеру блогы орындалуы барысында, алгоритмнің екі мүмкіндігінің тек біреуі ғана таңдап

алынып жүзеге асырылады да, ал екінші таңдап алынбаған тармақ біріктіру нүктесіне дейін орындалмай қалады. Енді осыған нақты мысалдар келтірейік.



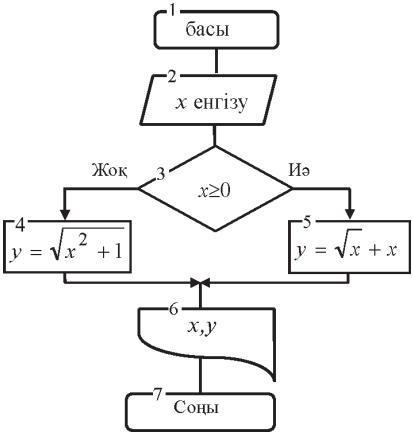
1.2-сурет. «Таңдау» алгоритмі



1.3-сурет. «Аттап өту» алгоритмі

1-мысал. y функциясын төмендегі формула бойынша есептеп шығару керек.

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



1.4-сурет. Тармақталу алгоритмі

Мұнда x айнымалысының таңбасына (оң, теріс) байланысты не жоғары, не төменгі формуланы таңдап алып, сол арқылы y функциясының мәнін табамыз (1.4-сурет). 2-блоктың орындалу барысында x айнымалысына белгілі бір мән беріледі де, ол мән енгізу операторлары арқылы программаға енгізілуі тиіс. Бұдан кейін енгізілген мәннің оң немесе теріс екендігі үшінші шартты тексеру блогы арқылы айқындалады. Шарттың

“ақиқат” (иә) немесе “жалған” (жоқ) болуына байланысты 4- не 5-блоктардың бірі ғана орындалып, “таңдау” орындалады. 6-блок *x* айнымалысының және *y* функциясының сандық мәндерін экранға немесе қағазға басып шығарады.

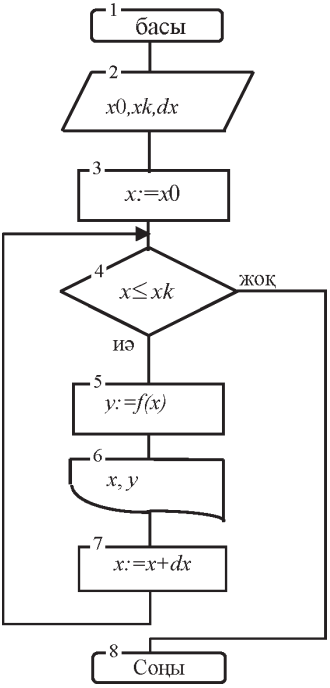
3. Циклдік алгоритмдер. Математикада, экономикада көптеген есептерді шығару кезеңінде бір теңдеуді пайдаланып, ондағы айнымалының өзгеруіне байланысты оны бірнеше рет қайталап есептеуге тура келетін сәттер де жиі кездеседі. Осындай қайталап орындалатын есептеу процесінің белгілі бір бөліктерін цикл деп атайды. Осы бірнеше рет қайталанатын бөлігі бар алгоритмдер тобы циклдік алгоритмдерге жатады. Циклдік алгоритмдерді пайдалану оларды кейіннен программаларда цикл операторы түрінде қысқартып жазу мүмкіндігін береді. Циклдер қайталану санының алдын ала белгілі және белгісіз болуына байланысты екі топқа бөлінеді. Қайталану сандары алдын ала белгілі болып келетін циклдер тобы арифметикалық цикл болып есептеледі, ал орындалу саны белгісіз циклдер – қадамдық (итерациялық) цикл болып аталады.

Практикада белгілі бір айнымалының сандық мәніне байланысты орындалатын арифметикалық циклдер жиі кездеседі. Мұнда арифметикалық прогрессияға ұқсас болып келетін циклдер ең қарапайым арифметикалық цикл болып табылады. Оны басқару қайталану кезеңінде прогрессияның заңына сәйкес тұрақты шамаға өзгеріп отыратын цикл параметрінің сандық мәнімен байланысты болуы тиіс.

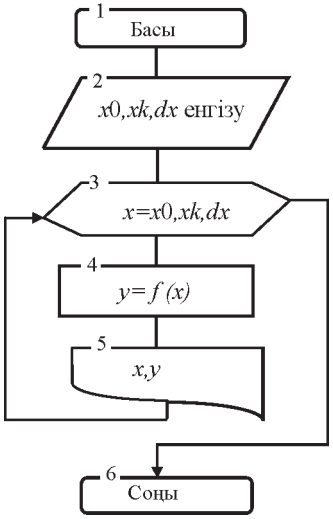
Цикл орындалуы алдында оның айнымалы аргументі – параметрі алғашқы мәнге ие болуы керек, сонан кейін қайталану кезеңінде цикл параметрі белгілі бір шамаға (қадамға) өзгере отырып, ол алдын ала берілген ең соңғы мәнге дейін жетуі қажет.

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

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



1.5-сурет. Қарапайым циклдік алгоритм

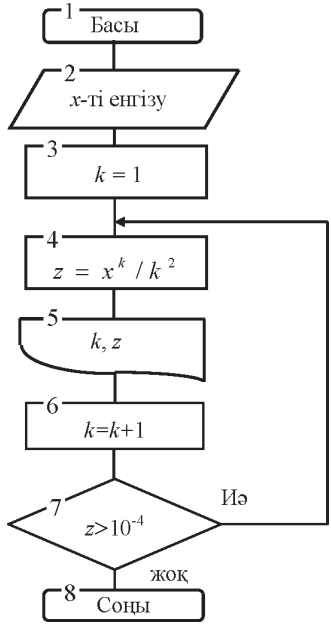


1.6-сурет. Модификаторлы циклдік алгоритм

мұнда [...] – өрнектің бүтін бөлігі алынатынын көрсетеді. *n* әрқашанда бүтін сан болуы тиіс, егер ол аралас сан болса, онда оның бөлшегі алынып тасталады, өйткені циклдің қайталану саны бүтін натуралдық сан болуы тиіс. Арифметикалық цикл үшін *y=f(x)* функциясының есептелу жолы алгоритм ретінде 1.5-суретте көрсетілген. Мұндағы 3-ші, 4-ші, 7-блоктар циклді ұйымдастыру үшін қажет. Олар цикл параметрінің алғашқы мәнін, өзгеру қадамын белгілеп және оның ең соңғы мәніне жеткен-жетпегенін тексереді. Ал 5- және 6-блоктар бірнеше рет қайталанып циклдің өзін құрайды. 4-блок шартты тексеріп қайталану процесін ұйымдастырады.

Алгоритм схемасын салуды және программаны жазуды жеңілдету үшін цикл алгоритмдері ықшамдалған түрде “модификатор” немесе “цикл басы” блогын пайдалану арқылы жазылады. Онда 1.5-суретте көрсетілген 3-ші, 4-ші, 7-блоктардың орнына “цикл басы” блогы орналасады. Ол алты-бұрыш тәрізді геометриялық фигурадан тұрады және оның міндетті түрде екі кіру және екі шығу сызығы болуға тиіс. Осы блокты пайдалану арқылы жоғарыда келтірілген алгоритм 1.6-суретте көрсетілген түрде кескінделеді. Параметрдің алғашқы *x* мәні оның соңғы *x* мәнінен кем болса, онда оның қадамы *dx* оң сан болады. Керісінше, параметрдің алғашқы мәні оның соңғы мәнінен артық болса, онда қадам теріс сан болады.

4. Қадамдық циклдер. Циклді орындаудың алдында, оның қайталану саны белгісіз болған жағдайда қадамдық циклдер пайдаланылады. Мұнда циклді жазу үшін тек қана "шартты тексеру" блогын қолдану қажет, ол циклді аяқтау үшін белгілі бір шартты тексереді. Қадамдық циклдердің схемасын сызғанда модификаторды (алтыбұрышты) қолдана алмаймыз, себебі алдын ала циклдің неше рет қайталанатыны бізге белгісіз. Енді осындай циклдер жұмысына мысал келтірейік.



1.7-сурет. Қадамдық цикл алгоритмі

3-мысал. $Z = \dfrac{x^k}{k^2}$ функциясының мәндерін $k = 1, 2, 3, \ldots$ және Z-ті 0.0001-ден артық болған жағдайда есептейік, мұндағы $0 \le x \le 1$. Бұл мысалда алдын ала цикл неше рет қайталанатынын айта алмаймыз, өйткені бізде тек k параметрінің алғашқы мәні мен қадамы ғана белгілі. Сонымен қатар Z функциясының 0.0001-ден артық болуы циклді қайталау шарты болып есептеледі ($Z > 0.001$). 1.7-суретте осы есептің алгоритм схемасы көрсетілген.

1.7. Программалау тілдері

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

Программа жазуды жеңілдету үшін математикалық формулаларды кеңінен қолданатын, ағылшын тілінің негізінде жасалған

алгоритмдік тілдер Бейсик, Паскаль, Фортран, СИ, т.б. кеңінен қолданылады. Алгоритмдік немесе программалау тілі – жазу ережелері қарапайым жасанды тіл. Оның машина тілдерінен айырмашылығы – табиғи ағылшын тілі негізге алынып, кең тараған математикалық таңбалармен толықтырылып жасалған. Сондықтан алгоритмдік тілдерде программа жасау адамдарға әрі жеңіл, әрі ыңғайлы болып келеді. Алгоритмдік тілдер автоматты түрде ЭЕМ-нің көмегімен аудармашы программалар арқылы машиналық тілге көшіріледі. Алгоритмдік тілді машина тіліне тікелей аударатын үлкен программаларды транслятор деп атайды. Алгоритмдік тілдерді пайдалану программалауды жеңілдете отырып, ЭЕМ-де есеп шығару процесін оңайлатады, алайда онда есеп шығару уақыты аздап көбейеді.

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

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

Негізінде ЭЕМ-де кез келген программалау тілінде жазылған есеп машина тіліне аударылып барып орындалады. Есептің орындалу кезеңдерін 1.8-суретте көрсетілген схема түрінде көрсетуге болады.

1.8-сурет. Есепті шығару кезеңдері

Қазіргі кезде бес жүзге жуық алгоритмдік тілдер тараған. Олардың әрқайсысы белгілі бір мақсаттарда қолданылады. Мысалы, Фортран – ғылыми-техникалық (инженерлік) есептерді шығару үшін, Паскаль – оқып үйренуде, ал Бейсик – дербес компьютерлерде диалог режимінде жұмыс істеуде қолданылады.

Бақылау сұрақтары

1. *Алгоритм және программа дегеніміз не, олардың қандай ұқсастықтары мен айырмашылықтары бар?*
2. *ЭЕМ-де орындалатын алгоритмдердің қандай қасиеттері болады?*
3. *Алгоритмдерді өрнектеу жолдары.*
4. *ЭЕМ-де есеп шығару кезеңдері.*
5. *Алгоритм схемаларының әр түрлі блоктары, олардың бейнеленуі, байланыстары.*
6. *Сызықтық, тармақталу және циклдік алгоритмдер ұғымдары.*
7. *Қадамдық циклдер және олардың ерекшеліктері.*
8. *Алгоритм командасы дегенді қалай түсінуге болады?*
9. *Автоматты құрылғылар алгоритм атқарушы бола ала ма?*

Тапсырмалар

1. *Өздеріңізге белгілі алгоритмдерден бірнеше мысал келтіріңдер.*
2. *Екі нақты оң сандар берілген. Осы сандардың қосындысын, айырмасын, арифметикалық ортасын және көбейтіндісін табатын алгоритм құрыңыз.*
3. *Тікбұрышты үшбұрыштың катеттері берілген. Үшбұрыштың гипотенузасы мен ауданын табатын алгоритм құру керек.*
4. *Тікбұрышты үшбұрыштың ауданын Герон формуласы бойынша есептейтін алгоритм құрастыру қажет.*

2. ТУРБО ПАСКАЛЬ ПРОГРАММАЛАУ ОРТАСЫ

Турбо Паскаль программалау жүйесі – Паскаль тілінің (француздың атақты математигі және философы Блез Паскальдің (1623 – 1662) құрметіне осылай аталған) компиляторы және программа құру мүмкіндіктерін арттыруға бағытталған құралдары бар программалық қоршаудан тұрады. Алдағы уақытта қысқарақ жазылуы үшін Паскаль тілінің компиляторын *Турбо Паскаль тілі,* ал программалық қоршау арқылы орындалатын мүмкіндіктердің барлығын *Турбо Паскаль ортасы* деп атайтын боламыз.

2.1 Турбо Паскаль ортасымен жұмысты бастау

Турбо Паскаль жүйесінің көлемі үлкен, ол бірнеше дистрибутивтік дискеттерге жазылып, қатты дискіге орнатылады. Жүйені қатты дискіге орнатқан кезде *TP* (немесе *PAS, TURBOPAS, PASCAL*) деген атпен жеке каталог ашылады да, оның ішіне дистрибутивті дискідегі барлық файлдар көшіріліп жазылады. Турбо Паскальді іске қосу үшін ДК файлдарының бұтақ тәріздес құрылымының ішінен осы *TP* каталогын, оның ішінен *TURBO.EXE* файлын тауып алу керек. Бұл файлда жұмыс істеуге дайын тұрған Турбо Паскальда программалаудың сұқбаттасу жүйесі бар. Файл Турбо Паскальдың ең негізгі минимальды құрамынан (мәтіндік редактор, компилятор, программа жинақтауышы және жүктеуіш) тұрады. Сонымен бірге осы ортада қалыпты жұмыс жасауға қажет *TURBO.TPL* файлында жазылған негізгі кітапхана мен анықтамалық мәлімет (*TURBO.HLP*) керек болады. Кітапта келтірілген көптеген мысалдарды теріп жазып, компиляциядан өткізіп, орындау үшін осы файлдар жеткілікті болып табылады.

Егер аталған файлдар *D:* дискісінің *TP* каталогында орналасқан болса, онда MS DOS ортасында Турбо Паскальді іске қосу үшін төмендегі команданы орындау керек:

D:\TP\TURBO

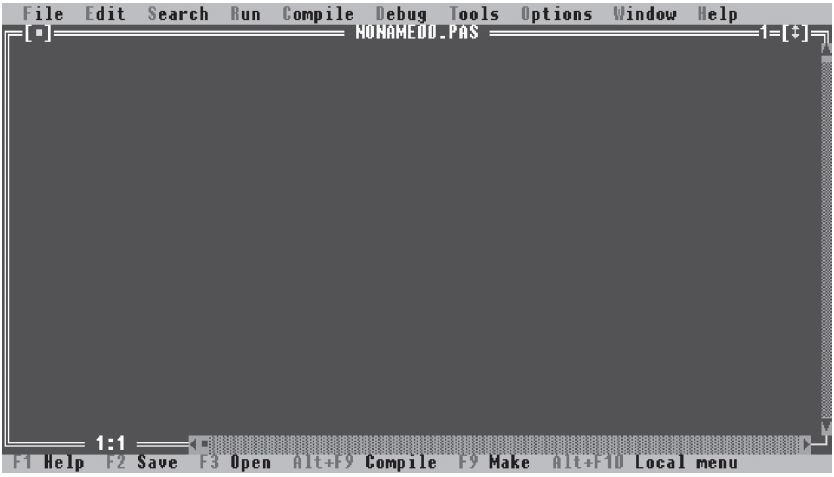
Осы команда бойынша *MS DOS* операциялық жүйесі *TURBO.EXE* файлын орындауға жібереді: программаны компьютер жедел жадына жүктеп, басқаруды осы программаға береді. Жүйемен жұмыс жасағанда *TP* каталогын үнсіз келісім бойынша таға-

йындалатын ағымдағы жұмыс каталогы ретінде пайдалануға болмайды. Себебі байқаусызда каталог ішіндегі басқа файлды өшіріп алып, программалау жүйесін істен шығарып алуымыз мүмкін, оған қоса, біраз жұмыс істегеннен кейін бұл каталог Турбо Паскальға тікелей қатысы жоқ жұмыс файлдарына толып кетеді. Бұдан былай *TP* каталогы жүйелік каталог деп атаймыз.

Жүйелік каталогты ағымдық жұмыс каталогы ретінде қолданбаудың тағы бір себебі, Турбо Паскаль жүйесі баптау параметрлерін *TURBO.TP* және *TURBO.PCK* файлдарында сақтап отырады. Жүйе іске қосылған кезде осы файлдарды ол ағымдағы каталогтан іздейді. Егер ағымдағы каталог сіздің жеке каталогыңыз болса, онда жүйе іске қосылған кезде сіздің талаптарыңызға сай параметрлерді өзі таңдап алады. Бұл файлдар сіздің каталогыңыздан табылмаған жағдайда, олар жүйелік каталогтан ізделеді, файлдар ол жерден де табылмаса, стандартты параметрлер автоматты түрде таңдалып іске қосылады. Жұмыс соңында керекті параметрлерді сақтайтын файлдарды өз каталогыңызда сақтасаңыз, жүйе іске қосылған сайын баптау параметрлерін таңдауды орындамайсыз.

Турбо Паскаль жүйесі ойдағыдай іске қосылғаннан кейін экран беті 2.1 суретте көрсетілгендей түрде болады.

Турбо Паскаль жүйесінен шығу үшін *Alt+X* пернелерін қатар басу керек.



2.1 сурет. Турбо Паскаль іске қосылғаннан кейінгі экран түрі

Турбо Паскаль біріктірілген ортасын сыртқы түрі бойынша үш бөлікке бөлуге болады.

Бірінші, ортаңғы бөлік – көп терезелі мәтіндік редактор аймағы. Мұнда программа мәтіні (немесе басқа мәтін) теріліп, редакцияланады.

Екінші, экранның жоғарғы бөлігі – біріктірілген орта ресурстарын басқаруға арналған негізгі меню қатары. Негізгі меню Турбо Паскаль программалау жүйесінің барлық функционалдық мүмкіндіктерін басқаратын болғандықтан, оның көмегімен біріктірілген ортаны тез игеріп кетуге болады.

Үшінші, экранның төменгі бөлігі – түсініктеме қатары. Бұл жолда жүйенің қалып-күйі және осы мезетте орындауға болатын іс-әрекеттер жайлы мәліметтер беріліп отырады.

Оң жақ жоғарғы бұрышта терезе нөмірі жазылады. Турбо Паскаль жүйесімен жұмыс жасағанда бір мезетте бірнеше терезе ашуға болады. Терезелердің әрқайсысын жеке-жеке ашуға, жабуға, екпінді етуге, көлемін және экрандағы орнын өзгертуге болады. Қанша терезе ашсаңыз да, оның тек біреуі ғана екпінді күйде болады. Екпінді терезе дегеніміз – екі ақ түсті жақтаулармен көмкерілген терезе. Екпінді терезе ішіндегі курсорды бағыттауыш пернелер (←,↑,→,↓) арқылы басқаруға болады. Жақтау бойында терезені басқаруға арналған арнайы символдар орналасқан. Жақтаудың жоғарғы жақ ортасында редакциялау үшін ашылған файл аты жазылады. Файл аты ретіне қарай ашылған терезелер санына байланысты автоматты түрде NONAME00.PAS немесе NONAME01.PAS, NONAME02.PAS, т.с.с. болып жазылып тұрады. NONAME – аты жоқ файл деген мағынаны білдіреді. Оларды сақтау кезінде қалауымызша өзгерте аламыз.

Турбо Паскаль жүйесінде мәтіндік редактор терезесінен басқа: программаны жөндеуіш терезе, программа нәтижесін шығару терезесі, анықтама терезесі, стектер және регистрлер терезелері қолданылады. Бұл терезелерді қалауыңыз бойынша кезек-кезек немесе бір мезетте қатар ашып қоюыңызға болады.

2.2 Функциональдық пернелер қызметі

Функциональдық пернелер Турбо Паскаль ортасын басқару үшін қолданылады. Олар *F1, F2, ..., F12* деп белгіленіп, пернетақтаның жоғарғы бөлігінде орналасады. Әр перне Турбо

Паскаль ортасының негізгі менюінің бір командасымен байланысқан. Функциональдық пернелер қызметін: *Alt* (*ALTernative – қосымша*), *Ctrl* (*ConTRoL – басқару*) және *Shift* (*SHIFT – жылжыту*) пернелерінің бірімен қатар басып түрлендіруге болады.

F1 – ішкі анықтамалық жүйеден мәлімет алу (Help – көмек алу);

F2 – редакцияланып жатқан файлды дискіге жазу;

F3 – дискідегі файлды редактор терезесінде ашып оқу;

F4 – программаны жөндеу режимінде қолданылады: программаның орындалуын бастау немесе тоқтату және программаның атқарылуын курсор тұрған жолда тоқтату;

F5 – екпінді терезені бүкіл экранды толық алатындай етіп ұлғайту;

F6 – келесі терезені екпінді ету;

F7 – программаны жөндеу режимінде қолданылады: программаның келесі жолын орындау; егер келесі жолда процедураны (функцияны) орындау керек болса, онда сол процедураға кіріп оның бірінші операторын орындау алдында аялдау;

F8 – программаны жөндеу режимінде қолданылады: программаның келесі жолын орындау; егер келесі жолда процедураны (функцияны) орындау керек болса, онда сол процедураны (функцияны) орындап, жұмысын тексеру;

F9 – программаны компиляциядан өткізу, орындауға жібермеу;

F10– негізгі меню көмегімен сұқбаттасу режиміне көшу;

Ctrl– F9 – программаны атқару: редактордағы программаны компиляциядан өткізіп, оны жедел жадыға жүктеу және программаны орындап Турбо Паскаль ортасына қайтып оралу;

Alt – F5 – редактор терезесін программа нәтижесі шығатын тұтынушы терезесіне ауыстыру.

Осы айтылғандардың ішінен біздің жиі қолданатынымыз: *Ctrl – F9* программаны орындау және *Alt – X* Турбо Паскаль ортасынан шығу. *F2* және *F3* пернелері файлмен жұмыс істеу үшін қажет. Ал *ALT – F5* пернелерінің көмегімен кез келген уақытта орындаған программа нәтижесін көре аламыз.

2.3 Мәтіндік редактор

Турбо Паскаль ортасының мәтіндік редакторы программа мәтінін жазуға және редакциялауға арналған ыңғайлы құралдардан тұрады. Турбо Паскаль ортасының редакциялау режимінде тұрғандығын мәтіндік редактор терезесіндегі курсорға (жыпылықтағын кішкене тіктөртбұрыш) қарап білуге болады. Редакциялау режимі Турбо Паскаль іске қосылғаннан кейін бірден автоматты түрде орнатылады. Редакциялау режимінен Турбо Паскаль ортасының кез келген басқа режиміне функциональдық перне немесе негізгі меню арқылы көшуге болады. Егер орта негізгі меню жолынан таңдау режимінде тұрса, онда курсор көрінбейді, ал меню жолында көк түсті көрсеткіш –тіктөртбұрыш пайда болады. Негізгі меню жолынан таңдау режимінен редакциялау режиміне қайта оралу үшін *Esc* пернесін басу керек. Ал негізгі меню жолынан таңдау режиміне өту үшін *F10* пернесін басу қажет.

Терезенің оң жақтағы және төменгі бөлігінде айналдыру жолақшалары орналасқан (полоса прокрутки). Әр жолақшаның мәтінге байланысты терезе курсорының қай жерде орналасқанын көрсететін жеке “курсоры” бар. Айналдыру жолақшасына қарап, Сіз мәтіннің қай бөлігінде тұрғаныңызды жылдам анықтай аласыз.

Программа мәтінін пернетақта көмегімен теремiз. Жаңа жолға көшу *Enter* пернесі арқылы орындалады.

Терезеде орналасқан жалпы символдар саны 64535-тен, ал программаның бір жолындағы символдар саны 126-дан аспауы керек.

Курсорды бағыттауыш пернелер (←, ↑, →, ↓) арқылы басқаруға болады. Егер Сіз кезекті символды қате терсеңіз, оны *Backspace* пернесінің көмегімен өшіруіңізге болады. *Del* (*DELete – өшіру*) пернесі курсор тұрған орындағы символды өшіреді, ал *Ctrl – Y* курсор тұрған жолды толығымен өшіреді.

Турбо Паскаль ортасында әр жолдың соңына экранда көрінбейтін жол соңы “белгісі” – бөлу символы қойылады. Бұл символ *Enter* пернесінің көмегімен қойылып, *Backspace* немесе *Del* пернесінің көмегімен өшіріледі. Егер бір жолды бірнеше жолға бөлу керек болса, онда курсорды бөлетін жерге орналастырып, *Enter* пернесін басу керек. Ал бірнеше жолды біріктіру үшін

курсорды бірінші жолдың соңына орналастырып, *Del* пернесін немесе екінші жолдың басына орналастырып, *Backspace* пернесін басу керек.

Мәтіндік редактордың қалыпты жұмыс жасау режимі – ығыстырып енгізу, яғни кірістіру режимі, әр енгізілген символ экрандағы мәтінді оңға жылжытып отырып жазылады. Қалып кеткен символды немесе жолды тек осы режимде ғана қоюға болады. Редактор сонымен бірге жаңа символды бұрын жазылған мәтінді өшіріп, оның үстіне жазу, яғни алмастыру режимінде де жұмыс жасай алады. Бұл режимде әр жаңадан енгізілген символ курсор орналасқан жердегі символ орнына жазылады. Экрандағы мәтін оңға жылжымайды. Символды режимінен символдарды *алмастыру* режиміне көшу үшін *Ins* пернесін басу керек. *Ins* пернесін қайта басу символды *кірістіру* режиміне қайта ауыстырады. Қай режимде жұмыс жасап отырғаныңызды курсорға қарап білуге болады: символды кірістіру режимінде курсор жыпылықтаған сызық, ал символдарды ауыстыру режимінде – биіктігі символмен бірдей жыпылықтаған тіктөртбұрыш болып өзгереді.

Редактордың айта кетерлік тағы бір мүмкіндігі – *автошегініс режимі*. Бұл режимде алдыңғы жол қай позициядан басталса, келесі жол да автоматты түрде сол позициядан басталады. Автошегініс режимі программа мәтінін безендірудің жақсы стилін береді: сол жақ шеттен басталған шегініс арқылы шартты немесе құрама операторлар тобын ерекшелеп бөліп орналастыруға болады. Автошегініс режимін алып тастау үшін *Ctrl* пернесін басып тұрып, алдымен латын алфавитінің *O* пернесін басып, сонан кейін *O* пернесін жіберіп, *I* пернесін басу керек. Осы пернелерді қайталап басу жолымен автошегініс режиміне ауысуға болады.

Турбо Паскаль мәтіндік редакторының жиі қолданылатын командалары.

Курсорды жылжыту

PgUp – бір парақ жоғары;
PgDn – бір парақ төмен;
Home – жолдың басына;
End – жолдың соңына;
Ctrl – PgUp – мәтіннің басына;
Ctrl – PgDn – мәтіннің соңына.

Редакциялау командалары

Backspace – курсордың сол жағындағы символды өшіру;

Del – курсор тұрған жердегі символды өшіру;

Ctrl – Y – курсор орналасқан жолды толығымен өшіру;

Enter – жолды курсор бойынша екіге бөлу;

Ctrl – Q L – ағымдағы жолды қалпына келтіру (курсор осы жолдан кетпей тұрғанда ғана орындалады).

Мәтін блоктарымен (фрагментімен) орындалатын әрекеттер

Ctrl – K B – фрагмент басын белгілеу;

Ctrl – K K – фрагмент соңын белгілеу;

Ctrl – K P – фрагментті баспаға жіберу.

Ctrl – Ins – фрагмент көшірмесін буферге алу;

Shift – Del – фрагментті буферге қиып алу;

Shift – Ins – буфердегі мәліметті курсор позициясына енгізу;

Ctrl – Del – фрагментті өшіру;

Alt – Backspace – соңғы әрекетті болдырмай тастау.

Ескерту. Фрагментті тышқанның сол жақ батырмасын басулы күйде ұстап, әдеттегідей түрде белгілеуге де болады.

2.4 Турбо Паскаль ортасының негізгі мүмкіндіктері

Файлдармен жұмыс істеу. Жоғарыда айтылғандай, Турбо Паскаль ортасы іске қосылғаннан кейін автоматты түрде мәтінді редакциялау режимі іске қосылады. Бұл жерде жаңа программа теруге немесе терілген программаны түзетуге болады.

Программа мәтінін ортадан тыс сақтаудың негізгі түрі файл болып табылады. Турбо Паскаль ортасымен жұмысты аяқтағаннан кейін, программа мәтінін дискіге файл ретінде жазып сақтап, кейіннен оны пайдалануға болады. Дискідегі файлдармен жұмыс жасау үшін *F2* (файлды жазу) және *F3* (файлды оқу) пернелерін қолданамыз. Егер жаңа программа жазылса, Турбо Паскаль ортасы оған автоматты түрде стандартты *NONAME00.PAS* (*NO NAME* – аты жоқ) деген ат береді. Программа мәтінін файлға сақтау үшін *F2* пернесін басамыз. Сонда экранда *Save file as... (...деген файлда сақтау)* деген жазуы бар сұқбат терезесі ашылады. Файлға ат беру өрісіне өзіңіз қалауыңызша ат енгізіп *Enter* пернесін басыңыз.

Егер программа мәтінін сақтамай, Турбо Паскаль ортасымен жұмысты аяқтасаңыз экранда:

NONAMEOO.PAS has been modified. Save?

(NONAMEOO.PAS файлына өзгерту енгізілді. Сақтау қажет пе?) деген мәліметі бар сұқбат терезесі ашылады. Егер программа мәтінін сақтау керек болса, *Y* (Yes – иә), кері жағдайда *N* (No – жоқ) пернесін басыңыз.

Программаны орындау арқылы тексеріп түзету. Программа мәтіні дайын болғаннан кейін оны орындауға жіберуге болады, олар мынадай әрекеттерден тұрады: программаны компиляциядан өткізу, оны стандартты процедуралар және функциялар кітапханасымен байланыстыру (қажет болған жағдайда), жедел жадыға жүктеп, басқаруды программаға беру. Осы іс-әрекеттер тізбегі *программаны орындау* немесе *атқару* деп аталады және ол *Ctrl – F9* пернелері көмегімен де жүзеге асады.

Программада синтаксистік қате жоқ болса, программаның әр жолы рет-ретімен орындалып, экранда неше жолдың компиляциядан өткендігі және программаға бөлінген жедел жады көлемі жайлы мәлімет шығады. Басқаруды жүктелген программаға бермес бұрын Турбо Паскаль ортасы экранды тазалайды, дәлірек айтсақ экранға программны орындау терезесін шығарады. Программа орындалып біткеннен кейін, компьютер басқаруды қайтадан өз қолына алып, экранға редактордағы программа мәтінін шығарады.

Егер программа орындалу барысында синтаксистік қате табылса, онда Турбо Паскаль ортасы жұмысын тоқтатып, экранға редактордағы программа мәтінін шығарады да, курсорды қате табылған жолға орналастырады. Ал, экранның жоғарғы бөлігіне табылған қате жайлы болжау мәліметі шығады. Бұл қатені түзетіп, программаны тез орындауға көмектеседі.

Табылған қате синтаксистік қате болмаған жағдайда, оның табылған орнын көрсету қажетті мәліметті бере алмайды. Қате мәндерді дұрыс дайындамау салдарынан шығуы мүмкін. Мысалы, егер қате теріс саннан түбір алу кезінде шыққан болса, онда осы түбір алу операторы орналасқан жол көрсетіледі. Ал қатені бұл жерден емес, ертерек, осы айнымалыға теріс мәнді меншіктеген жерден іздеу керек. Мұндай жағдайда программаны *F4, F7* және *F8* пернелерінің көмегімен біртіндеп, жол-жолмен адымдап орындау керек. Программаны түзетіп, орындаудан тәжірибеңіз аз болса, тек *F7* пернесін қолдануыңызға болады. *F7* пернесін бір басқаннан кейін Турбо Паскаль ортасы програм-

маны компиляциядан өткізіп, біріктіріп, жедел жадыға жүктеп, бірінші операторды орындаудың алдында тоқтайды. Орындалатын оператор экранда ерекшеленіп тұрады. Енді *F7* пернесін әр басқан сайын программаның бір жолы орындалатын болады. Программаның күдік тудырған жерінде айнымалы немесе өрнектің мәндерін қарап шығуыңызға болады. Ол үшін курсорды қажет айнымалы немесе өрнек жазылған жерге орналастырып, *Ctrl – F4* пенелерін қатар басу керек. Экранда үш жолдан тұратын сұқбат терезесі ашылады. Жоғарғы жолда айнымалының аты көрсетіледі. Ортаңғы жолда осы айнымалының ағымдағы мәнін шығару үшін курсорды жоғарғы жолға қойып, *Enter* пернесін басу керек. Егер *Ctrl – F4* пенелерін басар кезде курсор бос жолда немесе басқа айнымалыда тұрса, онда ашылған сұқбат терезесінің жоғарғы жолы да бос немесе басқа айнымалыны көрсететін болады. Бұл жағдайда сұқбат терезесінің жоғарғы жолына өзініңізге керек айнымалының атын пернетақтадан енгізіп, *Enter* пернесін басу керек. Төменгі жолда көрсетілген айнымалының ағымдағы мәні шығады. Осы тәсілмен тек айнымалының ғана емес, осы айнымалы бар өрнектің де мәнін көруге болады.

Турбо Паскаль ортасының анықтамалық жүйесі. Турбо Паскаль ортасының құрама бөлігінің бірі – оның анықтамалық қызметі. Егер Сіз ағылшын тілін жақсы білетін болсаңыз, Турбо Паскальмен жұмыс жасағанда ешқандай қиындық болмайды. *F1* пернесін бассаңыз болды, экранда қажет анықтама пайда болады. Бұл анықтама Турбо Паскаль ортасының ағымдағы қалып-күйіне байланысты (бұндай анықтама *мәтінге тәуелді* деп аталады) болады. Мысалы, *F1* пернесін Турбо Паскаль ортасы программа қатесін тапқан кезде бассаңыз, осы қатенің қандай жағдайда туындайтынына және оны қалай түзетуге болатындығы жайлы мәлімет аласыз.

Редактор терезесінен анықтамалық қызмет көрсету терезесіне төрт тәсілмен кіруге болады:

F1 – анықтама алу;

Shift – F1 – анықтамалық мәліметтер тізімінің ішінен керекті анықтаманы таңдап алу;

Ctrl – F1 – стандартты прцедура, функция, тұрақты немесе айнымалы жайлы анықтама алу;

Alt – F1 – алдыңғы анықтамаға көшу.

Shift–F1 командасын қолданғанда, экранда анықтама алуға болатын, алфавит бойынша реттелген стандартты процедуралар, функциялар, тұрақтылар және айнымалылар тізімінен тұратын терезе пайда болады. Анықтама алу үшін курсорды бағыттауыш пернелер (←,↑, →,↓) арқылы жылжыта отырып, қажет жолға келгенде *Enter* пернесін басыңыз.

Осы анықтаманы басқа жолмен де алуға болады: экранға қажет процедура (функция, тұрақты немесе айнымалы) атын жазып немесе курсорды программадағы қажет процедура атына қойып *Ctrl–F1* пернесін басыңыз. Турбо Паскаль ортасы курсор қай жерде тұрғанын сараптап, стандартты процедура атын ерекшелеп, қажетті анықтаманы экранға шығарды.

Көп жағдайларда анықтама Турбо Паскаль мүмкіндіктерін көрсететін кішігірім программа түрінде беріледі. Анықтаманы "қиып" алып редактор экранына орналастыруға болады. Ол үшін *Alt* пернесін басып жібермей тұрып, латын алфавитінің *E* әрпін басу керек. Экранда *Edit* меню жолының ішкі менюі ашылады. Бағыттауыш пернелер көмегімен курсорды *Copy examples* (мысалды көшіру) жолына қойып, *Enter* пернесін басу керек. Сонда анықтама мәтіні редактордың ішкі буферіне көшіріледі. Мысалды буфердегі мәліметті экрандағы курсор тұрған орынға қою үшін, *Esc* пернесін басып анықтама жүйесінен шығып, курсорды редактор экранының қажетті жеріне орналастырып, *Shift – Ins* (буфердегі мәліметті программа мәтініне жеке фрагмент ретінде көшіру), сонан соң *Ctrl – K H* (фрагменттің ерекшеленуін алып тастау) пернелерін басу керек.

Бақылау сұрақтары

1. *Турбо Паскаль ортасы деп нені айтады?*
2. *Турбо Паскаль жүйесі MS DOS ортасында қалай іске қосылады?*
3. *Турбо Паскаль жүйесінен шығу үшін не істеу керек?*
4. *Турбо Паскаль біріктірілген ортасы қандай бөліктерден тұрады?*
5. *Турбо Паскаль ортасындағы функциональдық пернелер қызметі.*
6. *Alt, Ctrl және Shift пернелерінің қызметі.*
7. *Турбо Паскаль ортасының мәтіндік редакторы ерекшеліктері.*
8. *Мәтіндік редактордағы автошегініс режимі дегеніміз не?*
9. *Мәтін бөліктерін қалай өшіруге болады?*
10. *Курсорды жылдам жылжыту пернелерінің қызметі.*
11. *Мәтіндік редактордағы түзету (редакциялау) командалары.*

12. *Мәтін блоктарымен (фрагментімен) орындалатын әрекеттер.*
13. *Программа мәтіні файлға қалай жазылып, соңынан қалай оқылады?*
14. *Файл атын қалай өзгертуге болады?*
15. *Программаны орындау қалай атқарылады?*
16. *Программаны компиляциядан өткізу дегеніміз не?*
17. *Программаның синтаксистік қателерін қалай анықтайды?*
18. *Программа нәтижесін қалай көреміз?*
19. *Турбо Паскаль ортасында қалай анықтама алуға болады?*
20. *Редактор терезесінен анықтамалық қызмет терезесіне қандай тәсілдермен кіруге болады?*
21. *Анықтамалық мәтіндерді буферге көшіріп алып, программа ішіне орналастыру тәсілдері.*

Тапсырмалар

1. *Төменде келтірілген EX_1 программасының мәтінін теріңіз.*
2. *Программа мәтінін толық белгілеңіз.*
3. *Белгіленген программа мәтінін буферге көшіріңіз.*
4. *Edit - Show clipboard командаларын орындап, Clipboard (буфер) терезесін ашып онда EX_1 программасының бар екендігіне көз жеткізіңіздер.*
5. *Жаңа терезе ашыңыз.*
6. *Жаңа терезенің 10-шы жолынан бастап EX_1 программасының мәтінін орналастырыңыз.*
7. *Программаны орындауға жіберіңіз.*

- -

Программа мәтіні

```
PROGRAM EX_1;
USES CRT;
BEGIN
  ClrScr;
Writeln (‘Құттықтаймыз! Программа көшірмесін жасау ойдағыдай өтті.’);
Writeln(‘  Ары қарай жалғастыру үшін  Enter пернесін басыңыз’);
Readln;
END.
```

- -

Төмендегі әрекеттерді орындау керек.

1. Программа мәтінінің кез келген жолын өзіңізге белгілі тәсілмен өшіріңіз.

2. Өшірілген фрагментті “өзгерісті қалпына келтіру” (Edit-Undo) операциясының көмегімен қалпына келтіріңіз.

3. “Өзгерісті қалпына келтіру” операциясына кері операцияны (Edit-Redo) орындаңыз.

3. ТУРБО ПАСКАЛЬ ТІЛІНЕ КІРІСПЕ

3.1 Паскаль тілінің жалпы сипаттамалары

Паскаль тілін 1968-71 жылдары Швейцарияда профессор Никлаус Вирт оқып үйренуге қолайлы программалау тілі ретінде ұсынған болатын. Қазіргі кезде барлық дербес компьютерлер осы тілде жұмыс атқара алады. Паскаль тілінде жазылған программаның дұрыстығы компьютерде жеңіл тексеріледі және жіберілген қате тез түзетіледі.

Бұл тілде жазылған программаны компьютерде орындау кезінде ол алдымен трансляция сатысынан өтіп (машина тіліне аударылып), объектілік программа түріне ауысады да, сонан кейін барып орындалады. Осы сәтте компьютерде программаның екі нұсқасы болады, оның біріншісі – алгоритмдік тілдегі алғашқы жазылған нұсқасы, ал екіншісі – объектілік кодтағы машина кодында жазылған программа. Есептің нәтижесін тек машиналық кодта жазылған программа арқылы аламыз, ал программаны түзету қажет болғанда оның алғашқы нұсқасы өңделіп, оны қайта түрлендіру сатысы жүзеге асырылады.

Біз мысалдар келтіріп, оқу процесінде қарастырып отырған тіл – Турбо Паскаль деп аталатын дербес ЭЕМ-дерге арналған Паскаль программалау тілінің бір нұсқасы, осы тәрізді Турбо Паскаль нұсқасы да жиі қолданылады. Бұлардың бір-бірінен айырмашылығы онша көп емес. Оқу орындарында қолданылатын компиляторларға байланысты осы екеуі де пайдаланыла береді. Төменде келтіріліген мысалдар осы екі нұсқада да орындала береді.

Қазіргі кезде Паскаль тілі кез келген күрделі есептерді шығара алатын, кең таралған стандартты оқып үйрену тіліне айналды. Оған бірнеше себептер бар.

Біріншіден, Паскаль тілі өз идеологиясы бойынша қазіргі программалау технологиясына өте жақын, оның негізгі операторлары құрылымдық программалау талаптарын толық орындай алады.

Екіншіден, бұл тіл программалардағы жоғарыдан төмен қарай құру технологиясын (қадамдар бойынша жіктей отырып талдау мүмкіндігі) толық жүзеге асыра алады.

Үшіншіден, мұнда мәліметтер құрылымының көптеген түрлері бар, сондықтан алгоритмдерді қарапайым етіп құра аламыз, ол программа жасаудағы еңбек өнімділігін арттырудың негізгі кепілі бола алады.

Алгоритмдік тілдердің машиналық тілдерден негізгі айырмашылықтары:

• алгоритмдік тіл алфавиті машина тілі алфавитінен әлдеқайда көлемді, сол себепті программа мәтіні көрнекі түрде жазылады да, оны оқу, түсіну жеңіл болады;

• пайдаланылатын операциялар жиыны машиналық операциялардан тәуелсіз, олар кез келген есептің алгоритмін жазу ыңғайлылығына қарай математикалық амалдарға байланысты таңдалып алынады;

• операцияларда қолданылатын мәліметтерге – операндтарға қайталанбайтын қарапайым атаулар беріледі, сонан кейін операндтар тек өз аттары бойынша шақырылып пайдаланылады;

• программалау тілінде мәліметтердің машиналық типтеріне қарағанда кең ауқымды көптеген типтер түрлері қолданылады.

Осыдан алгоритмдік тілдің машинадан тәуелсіз, математикаға жақындау, ыңғайлы тіл екенін аңғаруға болады.

Алгоритмдік тілдің синтаксисін жазу үшін арнайы *метатіл* қолданылады, ол алгоритмдік тілдің нақты конструкцияларының барлық ерекшеліктерін жинақы түрде көрсете алады. Біз осы мақсатта Джон Бэкус пен Питер Наур ұсынған қысқаша жазу тәсілі – *Бэкус-Наурдың металингвистикалық формулаларын* пайдаланатын боламыз. Бұл жазу тәсілі Backus Naur Form немесе қысқаша BNF (БНФ тілі) деп аталып жүр. Ол кезінде АЛГОЛ тілінің синтаксисін көрсету үшін кең қолданылды.

Тілдің синтаксисін жазуда оның кейбір түсініктері пайдаланылады: олардың ең қарапайым мүмкіндіктерін анықтап алып, солар арқылы ең басты болып саналатын – программа түсінігіне дейінгі барлық күрделі ұғымдарды анықтап шығамыз. Синтаксис тұрғысынан алғанда, әрбір анықталатын ұғым (негізгі символ емес) БНФ тілінің метаайнымалысы болып табылады, ал оның

мәні ретінде белгілі бір берілген шекті элементтерден тұратын кез келген конструкцияны қарастыра аламыз (яғни негізгі символдар тізбегі).

Тілдің әрбір ұғымы үшін тек бір ғана метаформула болуы тиіс, оның сол жағында анықталатын ұғым (БНФ тілінің метаайнымалысы) тұрады, ал формуланың оң жағында осы метаайнымалының мүмкін болатын барлық мәндері (осы ұғымға кіретін барлық мүмкін болатын конструкциялар) орналасады.

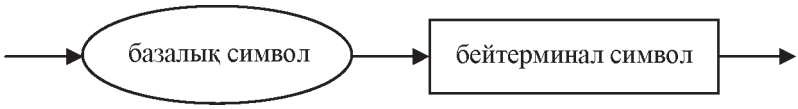
Барлық метаайнымалылар бұрыштық жақшаларға <...> алынып жазылады, олар анықталатын тілдің алфавитіне кірмейді, яғни метасимволдар болып табылады, мысалы, <өрнек>, <сан> және т.с.с. Ал тілдің негізгі символдары тікелей көрсетіледі. Метаформуланың сол жағы мен оң жағы мынадай арнайы таңбамен **::=** бөлініп жазылады. Бұл таңбаның мағынасы «анықтама бойынша мынаған тең» деген түсінікті береді. Әдетте метаайнымалы мәні ретінде мүмкін болатын бірнеше конструкциялардың кез келген бірі қабылдана алады. Мүмкін болатын барлық конструкциялар формуланың оң жағында мынадай «|» метасимволмен бөліне отырып жазылады, оның мағынасын «немесе» сөзі деп түсіну керек. Метаформуланың оң жағында метаайнымалының барлық мүмкін мәндерінен басқа сол мәндердің құрылу ережесі де көрсетіле алады.

Тағы да, біз қарастыратын екінші тәсіл – тіл ережелерін графикалық түрде бейнелейтін синтаксистік диаграмма болып табылады. Мұндай диаграммаларды Паскаль тілін жасаушы Н.Вирт көп қолданған болатын, сондықтан оны *Вирт синтаксистік диаграммасы* деп айтып жүр.

Синтаксистік диаграммаларда екі геометриялық фигура – тіктөртбұрыш және эллипс (кейде дөңгелек немесе овал) кең пайдаланылады. Тіктөртбұрыш ішінде тілдің анықталатын элементтері (бейтерминалды символдар), ал эллипс ішінде терминалдық символдар, яғни анықтауды қажет етпейтін таңбалар жазылады (3.1-сурет).

Диаграмма құрылымы бойынша жылжу, яғни оны оқу бағыты бағыттауыш тілсызық (стрелка) арқылы көрсетіледі.

Синтаксистік диаграммаларды пайдаланып, тілдің дұрыс конструкциясын түсіну үшін көрсетілген бағыт бойынша бір



3.1-сурет. Синтаксистік диаграммалар құрылымы

фигурадан екінші фигураға қарай жылжу қажет. Бір емес, тармақталған бірнеше бағыт көрсетілген жағдайда, олардың кез келгенін таңдап алуға болады. Егер сол бағытта басқа диаграммаға сілтеме тұрса, онда сол жаңа диаграммаға кіріп, сондағы көрсетілген бағыттар бойынша жылжи отырып одан шығып, бастапқы диаграммаға қайтып оралу қажет. Егер жылжу бағытында нүкте кездессе, онда бұл тармақтың тек Турбо Паскаль тіліне ғана қатысты екенінің белгісі болады, яғни бұл мүмкіндікті стандартты Паскаль тілінің кеңейтілуі деп ұққан жөн.

Программалау тілдерінің синтаксистік диаграммалары мен BNF тәсілінің мүмкіндіктері тіл ережелін көрсету үшін бірдей деп есептеледі.

3.2 Паскаль тілінің алфавиті

Паскаль тілі латын (ағылшын) алфавитінің бас әріптері мен кіші әріптерінен, астын сызу таңбасынан, араб цифрларынан және ажырату символдарынан (шектеуіштерден) тұрады.

<алфавит> **::** = <әріптер> | <цифрлар> | <шектеуіштер>
<әріптер> **::** = **A** | **B** | **…**| **Z** | **a** | **b** | **…**| **z** | <астын сызу таңбасы>
<цифрлар> **::** = **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**
<астын сызу таңбасы> **::** = _
<шектеуіштер>**::=**<операциялар таңбалары>|<жақшалар> |
<қордағы сөздер>| <ажыратқыштар>
<операциялар таңбалары>**::=** <арифметикалық>|<қатынас>|
<логикалық>|<сөз тіркестік>|<жиындық>
<арифметикалық операциялар таңбалары>**::=** + | – | * | / | **div** | **mod**
<қатынас операциялары таңбалары>**::=** = | <> | > | < | >= | <=
<логикалық операциялар таңбалары>**::= not** | **and** | **or** | **xor**
<тіркестік операциялар таңбалары>**::=** + |<қатынас операциялары>
<жиындық операциялар таңбалары>**::=** * | + | – | = | <> | <= | >= | **in**
<жақшалар>**::=** <индекстер үшін>|<өрнектер мен функциялар үшін>|
<операторлық>| <сөзтіркестік>

<индекстер үшін жақшалар>**::= [ ] | (. .)**
<өрнектер мен функциялар үшін жақшалар>**::= ( )**
<операторлық жақшалар>**::= begin end**
<сөз тіркестері үшін жақшалар>**::= ' '**
<ажыратқыштар>**::= := |. |, | : | ; | ^ | $ | #**
<қордағы сөздер>**::= and | asm | array | begin |**
case | const| constructor | destructor | div |
do | downto | else | end | exports | file | for |
function | goto | if | implementation | in |
inherited | inline | interface | label | library |
mod | nil | not | object | of | or | packed |
procedure | program | record | repeat | set |
shl | shr | string | then | to | type | unit | until |
uses | var | while | with | xor

«Босорын» таңбасы да ажыратқыштарға жатады. Ол программаның түсініктілігін жақсарту үшін қолданылады. «Босорын» идентификатор ішінде, сандар цифрлары арасында және қордағы сөздер ішінде болмауы тиіс.

Комментарий, яғни түсініктеме – **{ }** немесе **(* *)** жақшалары арасындағы мәтін. Олардың ішінде әріптер (орыс, қазақ әріптері), цифрлар, арнайы таңбалар бола береді. Түсініктемелер программа жұмысын, командаларды түсіндіру үшін қолданылады. Олар кез келген орында жазыла береді.

3.3 Программа құрылымы

Программалар белгілі бір мәселені, есепті шешу үшін пайдаланылады. Есеп шығару барысында компьютерге керекті бастапқы мәліметтер енгізіледі, оларды қалай өңдеу керектігі көрсетіледі және нәтиже қандай түрде, қай құрылғыға шығарылатыны айтылады.

Паскаль тіліндегі программа жеке-жеке жолдардан тұрады. Оларды теру, түзету арнайы мәтіндік редакторлар арқылы атқарылады. Программа қатарларының алдындағы азат жол немесе бос орындар саны өз қалауымызша алынады. Бір қатарға бірнеше командалар немесе операторлар орналаса алады, олар бір-бірімен нүктелі үтір (;) арқылы ажыратылып жазылады, бірақ бір жолда бір ғана оператор тұрғаны дұрыс, әрі түзетуге жеңіл, әрі ыңғайлы болып саналады.

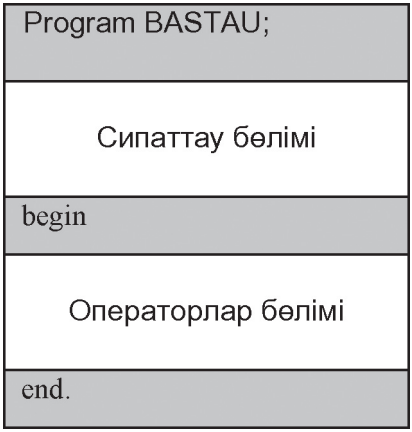
Кез келген программаның алғашқы жолы PROGRAM сөзінен басталатын оның тақырыбынан, яғни атынан тұрады. Одан кейін программаның ішкі объектілерінің сипатталу бөлімі орналасады. Бұл бөлім программадағы айнымалылар, тұрақтылар тәрізді объектілердің жалпы қасиеттерін алдын ала бекітіп анықтап алуға көмектеседі. Сипаттау бөлімі бірнеше бөліктерден тұрады, бірақ программа күрделілігіне байланысты көбінесе ол бір немесе екі ғана бөліктен тұруы мүмкін.

Программаның соңғы және негізгі бөлімі операторлар бөлімі болып табылады. Орындалатын іс-әрекеттер, командалар осы бөлімде орналасады. Ол *begin* түйінді сөзінен басталып, барлық атқарылатын операторлар (командалар) тізбегі жеке-жеке жолдарға жазылып біткен соң, *end* түйінді сөзімен аяқталады.

Программа құрылымын төменгі 3.2-суреттегідей етіп бейнелеуге болады. Операторлар бөлімінде командалар реттеліп орналасады, олардың кейбірі шартқа байланысты атқарылса, ал кейбірі қайталанып цикл немесе қосымша программа (подпрограмма, процедура) түрінде де орындалуы мүмкін.

Мәліметтер – сан мәндерін, мәтін ретіндегі сөз тіркестерін де мән ретінде қабылдай алатын тұрақтылар (константалар), айнымалылар, т.б. осылар тәрізді құрылымдар немесе солардың адрестері.

Мәлімет енгізу – бастапқы мәндерді пернетақтадан, дискілерден немесе енгізу-шығару порттарынан оқу арқылы жүзеге асырылады.



3.2-сурет. Паскаль тіліндегі программа бөлімдері

Операциялар немесе амал-әрекеттер – берілген, есептелген мәндерді меншіктеу, соларды өңдеу, салыстыру істерін орындайды.

Нәтиже алу (шығару) – аралық немесе қорытынды мәліметтерді экранға, дискіге немесе енгізу-шығару порттарына жазу.

Шартты атқарылу – белгілі бір көрсетілген шарт орындалса (ақиқат болса), онда командалар

жиыны атқарылады, әйтпесе олар атталып өтіледі немесе басқа командалар жиыны жүзеге асырылады.

Цикл – белгілі бір шарт орындалса (кейде орындалмаса), көрсетілген командалар жиыны бірнеше рет қайталанып атқарылады немесе шарт көрсетілмей-ақ алдын ала олардың неше рет қайталанатыны бүтін санмен беріледі.

Қосымша программа – алдына ала ат қойылған командалар тобы программаның кез келген жерінен оның атын көрсету арқылы шақырылып атқарыла береді.

Турбо Паскаль тілінің біріктірілген ортасы – кез келген программа жазып, оны орындауға қажетті құралдарды толық бере алады. Тіл элементтерімен танысу үшін «Мен Паскальда программалай аламын» деген мәтінді экранға шығаратын бір қысқаша программа жазайық. Оның алғашқы нұсқасы төменде келтірілген:

```
Program My_First_Program;
const
   Text = 'Мен Турбо Паскальда программалай
          аламын';
begin
   WriteLn(Text);
end.
```

Программада алты жол бар. Оның әрқайсысы белгілі бір мағынасы бар мәтін фрагментінен тұрады, жолдарға бөлу тілдің ережесіне байланысты емес, программалаушының қалауына байланысты жүргізіледі. Осы программаны былай да жазуға болатын еді:

```
Program My_First_Program;
const
   Text = 'Мен Турбо Паскальда программалай аламын';
begin
 WriteLn(Text);
end.
```

Паскальдада босорын таңбасы тіл элементтерін бір-бірінен бөліп жазу үшін қолданылады, сондықтан, мысалы, программаны былай етіп жазу қате болып есептеледі.

```
PROGRAM MY_First_Program; const Text='Мен Турбо
Паскальда программалай аламын';
BEGIN WriteLn(Text);end.
```

Паскальда егер ол мәтіндік константа болмаса, бас әріп пен кіші әріп бірдей болып есептеледі. Сондықтан программаның бірінші жолын мынадай түрде де жазуға болатын еді:

```
program my_first_program;
```

Енді әрбір жол мағыналарын қарастырып шығайық. Алғашқы жол

```
Program My_First_Program;
```

Мұндағы **Program** сөзі программа атын жариялау үшін ғана қолданылады. **Program** сөзі Паскальдің *түйінді сөзі* құрамына кіреді, ол программа атын беруден басқа еш жерде қолданылмауы тиіс. Паскальдің бірсыпыра түйінді сөздері бар, олардың тізімі жоғарыда келтірілді. Паскаль ортасының редакторы түйінді сөздерді басқа түспен ерекшелеп отырады. Осыған орай программа мәтіндерінде түйінді сөздер көбінесе қарайтылған түрде белгіленіп беріледі.

Паскаль тілінің түйінді сөздері – бір-бірімен айыру белгілерімен бөлініп, программада алдын ала анықталған белгілі бір мағынасы бар сөз тіркестері. Паскаль тілінің түйінді сөздерін үш топқа бөлуге болады, олар: қордағы (резервтегі) сөздер, стандартты атаулар немесе идентификаторлар және бейстандарт идентификаторлар. Тілдің операторларын, яғни қарапайым сөйлемдерін жазу үшін мағынасы мен қолдану тәсілі біржола анықталып қойылған символдар тіркесінен тұратын қордағы (резервтегі) түйінді сөздер пайдаланылады. Олардың жалпы саны 80-нен асып жығылады.

Енді программамызға оралайық. **Program** атауы кейін еш жерде қолданылмайтындықтан, оны жариялайтын бірінші жолды (**Program** My_First_Program;) жазбай кетуге де рұқсат етілген.

Жоғарыдағы My_First_Program атауы – ағылшынша «Менің Алғашқы Программам» деген сөз, бірақ ол босорынсыз жазылған – босорын сөздерді ажыратқыш таңба болып саналатындықтан, оны кез келген жерге қоя беруге болмайды (идентификатордағы сөздерді бөлу үшін астын сызу таңбасын қолдану ережесі қалыптасқан).

Бірінші жол нүктелі үтір (;) таңбасымен аяқталған. Паскаль тілінде бұл символ операторлың немесе сипаттау сөйлемінің соңын білдіреді. Осындай ажыратқыш таңба бір жолға бірнеше оператор жазу мүмкіндігін береді.

Екінші жол

const

түйінді сөзінен ғана тұрады, ол ары қарай бір немесе бірнеше константалар *(CONSTants – константы)* сипатталатынын білдіреді. Константа деп тілдің өз мәнін өзгертпейтін объектілерін айтады. Паскальдағы константалар ғылыми инженерлік есептердегі тәрізді жиі қолданылатын тұрақты мәндерді белгілі бір сөзбен немесе символмен белгілейді. Мысалы, біз тұрақты сан болып есептелетін $\pi=3,14159265$ константасын мектептен біледіміз. Программаны орындау кезінде компилятор *pi* константасын оның мәнімен алмастырады.

Паскаль тіліндегі константаны сипаттау дегеніміз – оның аты мен мәнін көрсету. Үшінші жолда мынадай мәтін бар:

```
Text = 'Мен Турбо Паскальда программалай аламын';
```

Мұнда **Text** атты константаға оның мәні ретінде «Мен Турбо Паскальда программалай аламын» деген сөз тіркесі меншіктеледі.

Паскальда константалардың әр түрлі типтері – бүтін немесе нақты сандар, символдар, сөз тіркестері, жиымдар (массивтер), т.б. қолданылады. Text атауының символдар тіркесінен тұратын константа екенін осы жолды қоршап тұрған екі апостроф (жоғарғы үтір) таңбасынан білуге болады, бірақ апострофтардың өздері сөз тіркесіне кірмейді. Олар тек сөз тіркесін бір ғана мән – мәтіндік константа ретінде қабылдау керек екендігін білдіріп тұр. Егер апострофтың өзін де мәтіндік константаға кіргізу керек болса, онда ол қатарынан екі рет жазылады. Мысалы:

```
Text = ''Турбо Паскаль'';
```

тіркесі 'Турбо Паскаль' константасын көрсетеді.

Программаның алғашқы үш жолы нақты бір әрекет атқармайды, олар тек компиляторға қолданылатын программа жайлы және онда қолданылатын объектілер туралы мәлімет береді. Сондықтан программаның осы жолдары *сипаттау бөлімі* деп аталады. Төртінші жолдағы **begin** түйінді сөзі программаның

келесі *операторлар бөлімінің* басталғанын білдіреді. Біздің мысалда бұл бөлімде

```
WriteLn(Text);
```

операторы тұр, ол экранға мәлімет шығарады.

Программаны нүктемен аяқталатын **end** түйінді сөзі аяқтайды. Нүкте компиляторға программаның біткенін хабарлайды. Осы **end.** сөзінен кейін жазылған мәтінді компилятор қабылдамайды.

Программаны аударып (компиляциялап) орындамас бұрын осындағы жалғыз атқарылатын операторды `WriteLn(Text);` қарастырайық.

Жалпы Паскальда, оның ішінде Турбо Паскальда да, арнайы енгізу-шығару операторлары жоқ. Программа сыртқы ортамен мәлімет алмасу үшін арнайы стандартты процедураларды пайдаланады. `WriteLn(Text);` – тіл құрамындағы мәлімет шығару процедурасын пайдалану операторы болып табылады (оның аты *WRITE LiNe – жолды жазу* сөзінен алынған).

Процедура түсінігі (8-ші тарауды қара) – тілдің негізгі ұғымдарының бірі. Процедура – бірсыпыра операторлар тізбегі, олардың бәріне бір ат қою арқылы толық орындап шығуға болады. Программада поцедура аты кездессе, соған қатысты операторлар тізбегі бір рет орындалады.

WriteLn процедурасы Паскаль тілінің стандартты, яғни ішкі процедурасы болып табылады. Стандартты процедураны алдын ала сипаттау қажет емес, оның атын кез келген программада көрсетіп, оператор сияқты орындай беруге береді.

WriteLn процедурасының бірнеше параметрлері болады. Параметрлер процедура атынан кейінгі жақша ішінде көрсетілетін тізім түрінде беріледі. Біздің мысалымызда процедурада тек бір ғана параметр – `Text` константасы көрсетіліп отыр. **WriteLn** процедурасының алғашқы параметрі ретінде мәлімет шығарылатын құрылғы немесе файл аты да көрсетіле алады (5-ші тарауды қара), яғни программалаушы қайда мәлімет шығаратынын өзі таңдайды. Егер құрылғы немесе файл аты көрсетілмесе, онда мәлімет тікелей экранға шығарылады.

Сонымен, программада қолданылған төрт сөз (**Program, const, begin** және **end**) қорға енгізілген түйінді сөздер болып саналады. WriteLn сөзі стандартты процедура аты, ол

қордағы сөздер санатына кірмегенмен, тілдің түйінді сөздері қатарына жатады, яғни оны да басқа мағынада қолдануға болмайды. Программадағы екі сөз – `My_First_Program` және `Text` *объектілер аттары – идентификаторлар* (атау) ретінде қолданылып отыр. Программалаушы идентификатор ретінде кез келген сөз тіркесін пайдалана алады, оларға мынадай талаптар қойылады:

- идентификатор латын алфавиті әріптерінен, астын сызу таңбасынан және цифрлардан құралады;
- басқа символдарды идентификаторды жазу үшін қолдануға болмайды;
- идентификатор цифрдан басталмайды;
- идентификатор бірде бір қордағы сөздермен немесе түйінді сөздермен сәйкес болмауы керек;
- идентификатор ұзындығы шектелмейді, бірақ программа оның алғашқы 63 символын ғана түсінеді.

Идентификаторларда бас әріп пен кіші әріп бірдей болып саналады, яғни мына сөздерді Text, text, TEXT компилятор бір сөз деп ұғады.

Енді программаны орындайық. Ол үшін программа мәтінін теріп болған соң, *Ctrl – F9* пернелерін қатар басу керек. Егер мәтін тергенде қате жіберілмесе, экрандағы сөздер өшіріліп, оған программа нәтижесі шығарылады. Нәтиже шығарылатын экран *тұтынушы экраны* деп аталады. Нәтиже толық шығып болысымен, экранға қайтадан программа мәтіні шығады. Егер нәтижені көре алмай қалған болсаңыздар, оны қайтару үшін *Alt – F5* пернелерін қатар басу керек. Нәтижені оқып болған соң, кез келген пернені басу экранда программа мәтінін көрсететін Турбо Паскаль редакторын қайта шығарады.

Енді Турбо Паскаль редакторы мүмкіндіктерінің бірімен танысайық. Редактордың бас менюіне көшу үшін *F10* пернесін басып, тышқан сілтемесін (курсорын) *Debug* (тексеру – отладка) опциясына, яғни командасына жеткізіп *Enter* пернесі басылса, экранда осы опцияның келесі деңгей менюі ашылады. Жаңа меню жоғары қатардан суырылып шығатын болғандықтан, оны суырмалы немесе қалқымалы меню деп атайды. Осы менюдегі *Output* (программаны шығару) опциясын таңдап, *Enter*-ді тағы

басайық. Экранға нәтижені көрсететін тұтынушы терезесі шығады да, енді ол пернелерді басқанмен жабылып қалмай, тұрақты түрде көрініп тұрады. Енді экранда екі терезе де (редактор және нәтиже терезелері) көрініп тұруы үшін, қайтадан *F10* пернесін басып, *Window* сөзін таңдап, *Enter*-ді басайық та, курсорды *Tile* (черепица) опциясына қойып, *Enter*-ді тағы да басайық. Егер бәрі дұрыс атқарылса, экран 3.3-суреттегідей болып шығады.

Программа терезесінің айналасындағы қос сызықты жақтау осы терезенің екпінді (активті) күйде жұмыс істеп тұрғанын көрсетеді. Редактор терезесін екпінді ету үшін: *Alt* пернесін басып, оны жібермей, 1 цифрын тереміз (редактор терезесінің нөмірі – 1, ал тұтынушы терезесінің нөмірі – 2, олар 3.3-суреттегі терезенің оң жақ жоғарғы бұрышында көрініп тұрады). Енді ары қарай жұмыс істей беруге жол ашық.

Енді экрандағы программа мәтінін өзгертіп, үшінші жол соңындағы нүктелі үтірді алып тастайық та, апостроф ішіндегі сөзді де басқаша жазайық:

```
Text = 'Турбо Паскальда программалауды үйренемін'
```

Енді *Ctrl – F9* пернелерін басып, программаны орындасақ, компилятор қате бар екенін хабарлап, мынадай мәлімет береді:

Error 85: «;» expected. (85 қате: «;» болмай тұр.)**,**

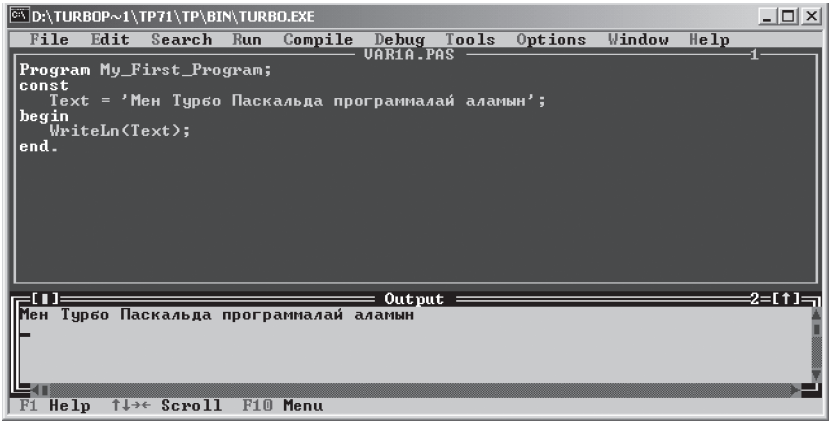
ал редактор курсорды **begin** сөзінің бірінші символына орналастырады да, осы маңайда қате бар екенін көрсетеді (компилятор «;» ажыратқышын оператор соңынан таппай, келесі сөзді – **begin** көрсетіп тұр). Программаны түзетіп – үшінші жол соңына «;» таңбасын қойып, есепті қайта іске қосалық. Бұл жолы бәрі де дұрыс болып, тұтынушы экранына төмендегідей мәтін шығады:

Турбо Паскальда программалауды үйренемін

Бұл мәтін константа мәніне толық сәйкес келеді, есеп дұрыс шығарылды.

Паскаль тілінің программасы блоктардан құралады. Бір блок ішіне басқа да кішігірім блоктар кіре береді. Блоктар екі бөлімнен тұрады, олардың алғашқысы – мәліметтерді сипаттау бөлімі, ал екіншісі – сол мәліметтерді пайдалана отырып, әр түрлі іс-әрекеттерді (операцияларды, амалдарды) атқару бөлімі.

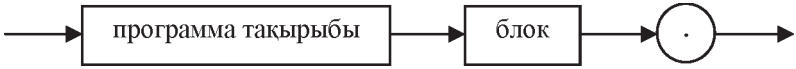
Мәліметтерді сипаттау бөлімі болмауы да мүмкін, ал екінші бөлім – негізігі бөлім, ол міндетті түрде болуы тиіс. Басқа блокқа



3.3-сурет. Турбо Паскаль біріктірілген ортасындағы редактор және тұтынушы экраны

кірмейтін блок басты (глобальды) блок болып саналады. Ал, блок ішіндегі блок – жергілікті, локальды блок деп аталады.

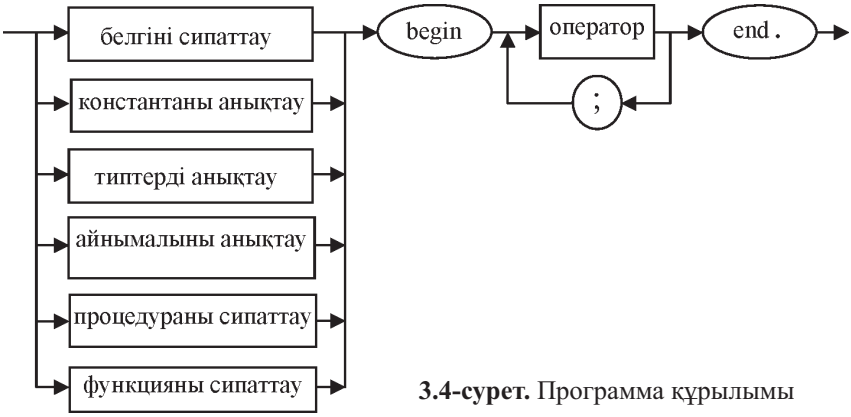
Сонымен, кез келген программа синтаксистік ереже бойынша тақырып пен блоктан тұрады.



Басты блок – негізгі программа блогы, сондықтан ол әрқашанда болуы тиіс. Оның құрамына жергілікті блоктар – процедуралар мен функциялар кіреді, олар кейбір программаларда болмауы да мүмкін.

Программалардың негізгі объектілері болып саналатын айнымалы, константа және олардың типтері де орналасқан блогына байланысты басты немесе жергілікті деп аталады. Программа объектілерінің де жұмыс істеу, әсер ету аймағы сол өздері орналасқан блокпен шектеледі. Блоктық құрылым программа мәтінін тиянақты түрде қатесіз құрастыруға көмектеседі. Программалық блоктың синтаксистік диаграммасын былай етіп көрсетуге болады:

Турбо Паскаль программасының тақырыбын жазбай да кетуге рұқсат етілген, бірақ жалпы Паскаль тілінің стандарты бойынша тақырып болуы қажет.



3.4-сурет. Программа құрылымы

Сонымен, кез келген программаның тақырыбы, онан кейін сипаттау бөлімі және `begin` және `end` сөздерімен қоршалған операторлар бөлімі болуы керек. Сипаттау бөлімі, толық болған жағдайда, 7 бөліктен тұрады, олар:

1. программамен байланысатын кітапханалық модульдер атауларының тізімі (ол uses түйінді сөзімен анықталады);
2. белгілерді (label) сипаттау;
3. константаларды (const) сипаттау;
4. мәліметтер типтерін (type) анықтау;
5. айнымалыларды (var) сипаттау;
6. процедураларды (procedure) сипаттау;
7. функцияларды (function) сипаттау.

3.4 Тілдің қарапайым конструкциялары

Негізгі символдардан тілдің бөлінбейтін ең кіші (минимальды) синтаксистік бірліктері болып табылатын қарапайым конструкциялары құрылады. Оларға: идентификаторлар, константалар, айнымалылар, стандартты функциялар көрсеткіштері жатады.

Программаның әр түрлі объектілерінің атаулары болуы тиіс. Атаулар ретінде идентификаторлар қолданылады. **Идентификатор** деп әріптерден, цифрлардан және астын сызу белгісінен құрастырылып, міндетті түрде әріптен басталатын символдар тізбегін айтады. Идентификатор ұзындығы 63 символдан артпауы тиіс. Идентификаторларды мағынасы бар сөз тіркестерінен

құрастырған дұрыс деп саналады, мысалы, MAX, MIN, BAGA, SUMMA және т.с.с. Қордағы сөздер мен түйінді сөздерді идентификатор ретінде пайдалануға болмайды.

Константалар – программаның жұмыс істеу барысында өз мәндерін өзгертпейтін мәліметтер. Константаларға атау беруге (типін көрсетіп немесе көрсетпей де) немесе тікелей олардың өздерін пайдалануға болады.

Арифметикалық константалар *бүтін* және *нақты сандық мәліметтерді* бейнелеу үшін қажет.

Бүтін сандар: +4, -100, 15743, 0 және т.с.с. Разрядтылығы 16 биттен тұратын дербес ЭЕМ-дер үшін қолданылатын бүтін сандар (ағылшынша INTEGER) -32768 бен +32767 аралығында ғана жазылады, бұдан үлкен сандар нақты сандарға айналдырылады.

Паскаль тілінде ондық және он алтылық бүтін сандар пайдаланылады. Он алтылық сандардың алдына $ белгісі қойылады. Мысалы, $ABC немесе $8B2.

Нақты константаларды жазудың *бекітілген нүкте арқылы* және *экспоненциальды* (жылжымалы нүкте арқылы) түрлері бар. Бекітілген нүкте арқылы жазылған санның бүтіні мен бөлшегі нүкте арқылы бөлініп тұрады, ал экспоненциал сан орталарында E әрпі орналасқан мантисса мен дәрежеден тұрады.

<мантисса>E{±}<дәреже>

Бекітілген нүкте арқылы нақты сандар кәдімгі табиғи аралас сандар тәрізді санның бүтіні мен бөлшегін нүкте арқылы бөлген күйде жазылады. Мысалы: 2.65, 0.5, -0.862, -6.0. Ал өте үлкен немесе өте кіші нақты сандар көрсеткіші бар экспоненциал сандар ретінде жазылады да, олардың диапазоны әлдеқайда кең болады. Мысалы:

Кәдімгі жазылуы	*Паскаль тілінде жазылуы*
145	145
1F	$1F
147,125	147.125
-6,045	-6.045
12×10^{14}	12E+14
$0{,}52\times10^{4}$	0.52E4
$5{,}2\times10^{-12}$	5.2E-12
-45×10^{6}	- 45E6

Турбо Паскаль тіліндегі константалар типтері мен олардың өзгеру диапазондары, компьютер жадында алатын орындары 3.1-кестеде көрсетілген.

Паскаль ортасында бірнеше атаулары бекітілген константалар бар, оларды жарияламай-ақ пайдалана беруге болады, олар:

MAXINT=32767, MAXLONGINT=2147483647.

Логикалық константалар екі мәннің бірін қабылдай алады: **`True`** (ақиқат) немесе **`False`** (жалған). Мұнда келесідей константалардың логикалық типтері бар: Boolean (1 байт), Bytebool (1 байт), Wordbool (2 байт), Longbool (4 байт).

Константалардың символдық типі `Char` ASCII (ақпарат алмасудың американдық стандартты коды) кодтарының бір символын ғана бейнелей алады. Символ компьютер жадында 1 байт орын алады. Символдық константа апострофқа ‘’ алынып жазылады. Символдар өз кодтарына байланысты реттеліп орналаса алады. Цифрлар мен латын алфавиті әріптерінің кодтары мынадай қатынастарға сәйкес реттеліп орналасады:

`'0'< …<'9'<'A'< …<'Z'<'a'< …<'z'.`

3.1-кесте

Константа типі	Диапазоны (ауқымы)	Жады көлемі	Ескертулер
Shortint	-128..127	1 байт	Таңбасы бар
Byte	0..255	1 байт	Таңбасыз
Integer	-32768..32767	2 байт	Таңбасы бар
Word	0..65535	2 байт	Таңбасыз
Longint	-2147483648..2147483647	4 байт	Таңбасы бар
Single	1.5E-45..3.4E38	4 байт	7-8 цифрлы, таңбасы бар
Real	2.9E-39..1.7E38	6 байт	11-12 цифрлы, таңбасы бар
Double	5.0E-324..1.7E308	8 байт	15-16 цифрлы, таңбасы бар
Extended	3.4E-4932..1.1E4932	10 байт	19-20 цифрлы, таңбасы бар
Comp	9.2E18..9.2E18	8 байт	19-20 цифрлы, таңбасы бар

Константалардың символдар тізбегінен тұратын тіркестік (жолдық) **типі `String`** апострофтарға алынған сөз тіркесінен тұрады. Тіркестің немесе жолдың ұзындығы деп оның құрамындағы символдар санын айтады. Егер тіркестік константа символдарының ішінде апострофтың өзі кездессе, ол қатар орналасқан екі апостроф түрінде жазылады. Тіркестің ұзындығы 0-ден 255 символға дейін болады.

Айнымалы – бұл белгілі бір мәнге берілген атау. Ол айнымалы идентификаторы арқылы белгіленеді. Стандартты типтегі айнымалылар соған сәйкес константалар үшін көрсетілген мәндер диапазонын қабылдайды.

3.5 Мәліметтер типтері

Паскаль тіліндегі программада әрбір айнымалы мен константаның берілген бір өзіндік типі болады. Тип сол объектіге қолдануға болатын операциялар жиынын анықтайды және сол операциялардың орындалуынан шыққан нәтиженің де типін айқындайды. Типтердің стандартты және тұтынушы анықтаған түрлері болады.

Программада пайдаланылатын барлық айнымалылар соларды жариялайтын **`VAR`** *сипаттау бөлімінде* төмендегідей болып көрсетілуі тиіс.

```
VAR <идентификатор> [,<идентификатор>,…]: <типі>;
 [<идентификатор> [,<идентификатор>,…]: <типі>;…]
```

мұнда тік жақша ішінде тұрған элементтерді жазбай кетуге рұқсат етілген, яғни оларды пайдалну міндетті болып саналмайды.

Мысалы,

```
VAR  A : INTEGER;
     B,C: REAL;
```

Бұл жерде бүтін типтегі A айнымалысы және нақты типтегі B, C айнымалылары сипатталған.

Тип, өз кезегінде, алдын ала типтерді сипаттай алатын **`TYPE`** бөлігінде төмендегідей түрде анықталуы мүмкін.

```
TYPE <тип идентификаторы> = <типі>;
```

Мысалы,

```
TYPE  I = INTEGER;
      R = REAL;
```

Мұндай типтерді жариялау жолдарынан кейін A, B және C айнымалыларын төмендегідей етіп сипаттауға болады:

```
VAR A: I;
    B,C: R;
```

Реттелген стандартты тип мәндердің шектелген жиындары тізбегін белгілейді. Оларға әдетте бүтін сандар типі, байттық, символдық және логикалық типтер жатады.

Саналатын тип – белгілі бір мәндердің реттелген жиынын анықтайды, сол мәндердің идентификаторлары нақты түрде тұрақты сияқты жақша ішінде ретімен тізіліп көрсетіледі. Осындай әрбір элемент үшін компьютер жадынан бір байт орын бөлінеді. Реттелген мәндер жиым (массив) индекстері тәрізді нөлден бастап нөмірленеді.

TYPE <тип идентификаторы> **= (**<идентификатор> [,<идентификатор>,…]**);**

Интервалдық (аралық) **тип** берілген айнымалы қабылдай алатын мәндер жиынын анықтайды. Мұнда реттелген типтің ең кіші және ең үлкен мәндері көрсетіледі. Компьютер жадынан әрбір элемент үшін бір байт орын бөлінеді.

TYPE <тип идентификаторы>= <константа>..<константа>;

Константалар ретінде нақты сандық типтен басқа кез келген қарапайым тип қолданыла алады.

Мысалы,

```
TYPE  GR =(DS101, DS102, DS201, DS202, DS301, DS302);
      SPEC = DS101..DS302;
      DIGIT = 0..9;
VAR   A: DIGIT;
      B: SPEC;
      D: 100..200;
```

Тіркестік (жолдық) **тип** ұзындығы 0 мен 255 символдар аралығындағы сөз тіркестерін сипаттау үшін қолданылады. Сөз тіркесінің мүмкін болатын ең үлкен (максимальді) ұзындығы тік жақша ішінде көрсетіледі. Егер ол көрсетілмесе, тіркестің максимальды ұзындығы 255 болып саналады. Компьютер жадындағы тіркестік айнымалылар көлемі, тіркестік константалар сияқты, сол сөз тіркесінің байтпен берілген максимальді ұзындығына 1 байт (нөлдік байт) қосқанға тең болады. Нөлдік байт сол

тіркестік айнымалының ұзындығын есте сақтау үшін керек. Турбо Паскальдің маңызды бір ерекшелігі – берілген сөз тіркесінің әрбір символын соның реттік нөмірі арқылы жекелеп өңдеуге болады.

TYPE <тип идентификаторы>**=String**[<максимальды ұзындығы>];

Мысалы,

```
TYPE        TSTRING = STRING[100];
            TS = STRING;
VAR         S,S1: TSTRING;
            S2: STRING[20];
            SS: TS;
```

Жиым (массив) – қасиеттері ұқсас болып келетін бір типтегі айнымалылардың реттелген жиыны. Элементтер арасындағы олардың реттілігі жиым индекстері арқылы анықталады. Жиымның әрбір элементіне бір немесе бірнеше индекс сәйкес келеді. Егер әрбір элементке бір индекс сәйкес келетін болса, онда ол – бірөлшемді жиым (вектор). Жиым элементі екі индекспен анықталатын болса – ол екіөлшемді жиым (матрица), мұндайдағы бірінші индекс – жол нөмірі, ал екіншісі – сол элемент орналасқан бағана нөмірі.

```
TYPE <тип идентификаторы> = ARRAY [<индекстер
                    типтері тізімі>] OF <типі>;
        <индекстер типі > :: = <қарапайым тип>
        <қарапайым тип>:: = <тип
        идентификаторы>|<идентификатор>
  [,<идентификатор>]|<константа>..<константа>
```

Индекс типі ретінде нақты сандық типтен басқа кез келген қарапайым тип қолданыла алады. Көбінесе индекстерде бүтін тип арқылы жазылған интервалдық типтер пайдаланылады.

Мысалы,

```
TYPE T1 = ARRAY [-10..20,1..30] OF BYTE;
     T2 = ARRAY [0..50] OF BOOLEAN;
     T3 = ARRAY [BYTE] OF INTEGER;
VAR  A,B: T1;
     C: T2;
     Z: ARRAY[1..100] OF REAL;
     MAS: T3;
```

Мұнда типтерді сипаттау бөлігінде жиымдардың үш түрлі типі келтірілген. T1 – екіөлшемді жиым, оның жолдарының нөмірлері –10-нан 20-ға дейін, ал бағаналар нөмірлері 1-ден 30-ға дейін өзгере алады. T1 типті жиым элементтері 0 мен 255 аралығындағы таңбасыз бүтін сан мәндерін қабылдай алады. T2 типі – логикалық типтегі бірөлшемді жиым элементтерін анықтайды, оның элементтерінің нөмірлері 0 мен 50 аралығында өзгереді. T3 типі – таңбасы бар бүтін сандар типіндегі элементтерден тұратын бірөлшемді жиымды анықтайды, оның индекстерінің өзгеру диапазоны 0 мен 255 аралығында болады. Айнымалыларды сипаттау бөлігінде T1 типіндегі A және B айнымалылары анықталады. C айнымалысының типі – T2, ал MAS айнымалысының типі – T3 болып сипатталған. Кейбір типтерді алдын ала сипаттамайақ, мысалы, Z жиымы тәрізді, бірден айнымалыларды сипаттау бөлігінде оның атауын да, индексінің өзгеруін де, типін де қатар көрсетуге болады.

Жиым элементтерін пайдалану индексті айнымалылар арқылы жүзеге асырылады. Индексті айнымалының индекстерінің саны сол жиымның элементтері санына тең болады. Индекстер мәндері бүтін сандар арқылы, қарапайым айнымалылар көмегімен немесе арифметикалық өрнектер мәндерін есептеу жолымен анықтала береді. Егер берілген екі жиымның сипаттамалары бірдей болса, онда олардың мәндерін бірінен біріне мынадай меншіктеу **B:=A** арқылы көшіруге болады.

Мысалы,

```
 …
S:= S+Z[I];
P:= P*A[I][J];
C[6]:= TRUE;
P:= P*A[I,J];
R:= B[I+5,J];
MAS[I]:= MAS[I-1]*MAS[I];
 …
```

Жиымдарды сипаттаудың әр түрлі тәсілдерін қарастырайық.

Элементтері бүтін сан болып келген, 10 жолдан және 50 бағанадан тұратын A матрицасын сипаттау керек болсын делік. Осы мысалды үш тәсілмен орындайық.

```
1)  CONST  N = 10;
           M = 50;
  TYPE TMATR = ARRAY[1..N,1..M] OF INTEGER;
  VAR  A: TMATR;

2)  TYPE TSTR = ARRAY[1..50] OF INTEGER;
          TMATR = ARRAY[1..10] OF TSTR;
  VAR  A: TMATR;

3)  VAR  A: ARRAY[1..10,1..50] OF INTEGER;
    VAR  A: ARRAY[1..10] OF ARRAY[1..50]
                 OF INTEGER;
```

Айнымалылар – программа орындалуы барысында өз мәндерін өзгерте алатын белгілі бір атаумен белгіленген мәндер. Оларды жариялау да программаның сипаттау бөлігінде орналасады, айнымалыны сипаттау кезінде оның идентификаторымен қатар типін де көрсету қажет (3.7-сурет). Айнымалыларды пайдалану да олардың идентификаторларын, яғни аттарын көрсету арқылы орындалады.

Айнымалы типі оның мүмкін болатын мәндерін, компьютер жадындағы алатын орнын және осы айнымалымен орындалатын операциялар жиыны анықтайды

3.5-суретте Паскаль тілінің Турбо Паскаль нұсқасындағы айнымалылар типтерінің жіктелуі, яғни классификациясы көрсетілген. Соған сәйкес айнымалылар типтері қарапайым және құрылымды болып екіге бөлінеді екен.

Қарапайым немесе скалярлық типтер мәндердің реттелген жиындарын сипаттайды. Олар реттелген және нақты болып екіге бөлінеді.

Реттелген типтер тобы мәндер жиыны шектелген айнымалылар типтерін біріктірсе, нақты типтер тобы – мәндер жиыны шартты түрде шексіз болып саналатын айнымалылар типтерін біріктіреді.

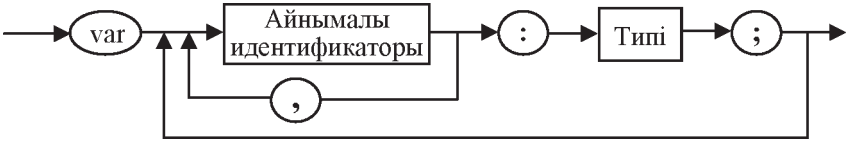
Айнымалылардың реттелген типтері стандартты, саналатын және кесінді болып бөлінеді. Стандартты түрде төмендегі типтер бекітілген:

- бүтін типтер – 3.2-кестені қараңыз;

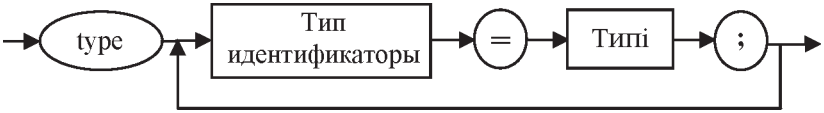
3.5-сурет. Типтердің жіктелуі

• Boolean бульдік типі екі-ақ мән қабылдайды – false (0) және true (1), бірақ компьютер жадында ол толық бір байт орын алады;

• Char символдық типі ASCII кестесінде (2 қосымшаны қараңыз) көрсетілген символдар жиынын анықтайды. Кестеде барлығы 255 код көрсетілген, олардың көбісінің символдық бейнесі тағайындалған. Символдар, мысалы, латын, орыс, қазақ әріптері, цифрлар және нүкте, үтір, т.б. арнайы таңбалар.



3.6-сурет. Айнымалыларды сипаттаудың синтаксистік диаграммасы



3.7-сурет. Типтерді жариялаудың синтаксистік диаграммасы

Стандартты емес (бейстандартты) реттелген типтерді айнымалыларды сипаттау (3.6 сурет) кезінде көрсету керек.

3.2-кесте

Аты	Белгіленуі	Мәндер диапазоны	Ішкі бейнелену ұзындығы, байт
Бүтін сан	Integer	-32768 .. 32767	2 (таңбасы бар)
Қысқа бүтін сан	ShortInt	-128 .. 127	1 (таңбасы бар)
Ұзын бүтін сан	LongInt	-2^{31}.. $2^{31}-1$	4 (таңбасы бар)
Байт	Byte	0 .. 255	1 (таңбасыз)
Машиналық сөз	Word	0 .. 65535	2 (таңбасыз)

Саналатын тип типтерді жариялау кезінде программалаушы анықтаған мәндер бойынша қалыптасады. Мәндер тізімін жай жақша ішінде үтір арқылы бөліп жазады, мысалы:

```
Var D:(Mon,The,Wed,Thu,Fri,Set,Sun);... { D айнымалы
      көрсетілген мәндерді ғана қабылдай алады }
```

Ескерту. Компьютер жадында саналатын тип мәндері нөлден басталған бүтін сандармен кодталады. Мысалы, `Mon` идентификаторына 0 сәйкес келеді, `The` – 1 және т.с.с.

Саналатын типті жариялау кезінде алдымен жаңа типті анықтап алып, сонан кейін осы типтегі айнымалыны сипаттауға болады, мысалы:

```
Type Day = (Mon,The,Wed,Thu,Fri,Set,Sun); {жаңа
типті жариялау}
Var D: Day;... {осы типтегі айнымалыны сипаттау}
```

Кесінді түріндегі айнымалы типі бұрын анықталған тип мәндерінің диапазоны ретінде көрсетіледі. Оны сипаттау кезінде де типтерді жариялау конструкциясын пайдалануға болады, мысалы:

```
Type Data=1..31;{бүтін типтердің бірінің диапазоны}
Var DataN: Data;...
```

немесе типті жеке сипаттамай-ақ, сол айнымалыны мынадай түрде жарялауға болады:

```
Var DataN: 1..31;...
```

Нақты типтер бөлшегі бар аралас сандарды бейнелеу үшін қолданылады. Компьютер жадында (ішкі көрсетілімде) нақты сандардың мантиссасы мен дәрежесі бөлек сақталады, ондағы мантисса мен дәрежеге бөлінетін разрядтар көлемі сол санның типімен анықталады. Соған сәйкес компьютердегі нақты сандарды өңдеу жұмысы мантиссаға бөлінген екілік разрядтар санына байланысты белгілі бір дәлдікпен ғана орындалады. Санның дәрежесін жазуға бөлінген разрядтар көлемі сол типке байланысты сандардың бейнелену диапазонын анықтайды. 3.3-кестеде тілдің Турбо Паскаль нұсқасындағы нақты сандар типінің сипаттамалары келтірілген.

Ескерту. Мыналарды есте сақтаған жөн:

• Real типінен басқа нақты сандар типімен жұмыс істеу кезінде компиляциялаудың айрықша режимін орнату керек ({$N+} директивасын енгізу немесе компилятордың осыған сәйкес опцияларын енгізу);

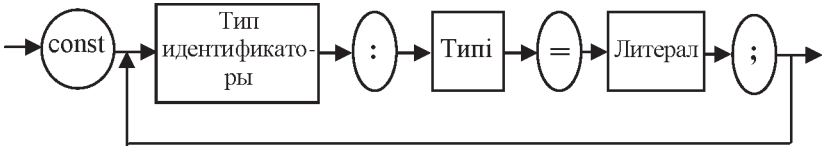
• Real типі үшін ең жай істейтін арифметика қолданылады.

Мәліметтердің құрылымдық типтері кейін қарастырылады.

Инициалданған айнымалылар. Турбо Паскаль тілінде бастапқы мәндері берілген айнымалыларды сипаттау мүмкіндігі бар. Мұндай айнымалылар инициалданған деп аталады және оларды арнайы `const` конструкциясы арқылы жариялайды (3.8-сурет).

3.3-кесте

Аты	Белгіленуі	Ондық цифрлар саны	Дәреженің өзгеру диапазоны	Ішкі бейнелену ұзындығы, байт
Нақты	Real	11..12	-39..+38	6
Бір еселі дәлдікпен	Single	7..8	-45..+38	4
Екі еселі дәлдікпен	Double	15..16	-324..+308	8
Кеңейтілген	Extended	19..20	-4951..+4932	10
«Үлкен бүтін»	Comp	19..20	$-2^{63} + 1.. 2^{63} - 1$	8



3.8-сурет. Инициалданған айнымалыларды жариялаудың синтаксистік диаграммасы

Ескерту. Тілдің идеологиясы бойынша инициалданған айнымалыларды `const` нұсқауында жариялау дұрыс болып саналмайды. Сол себепті тілдің кейінгі нұсқаларында бұл сәйкессіздік дұрысталды.

Программадағы инициалданған айнымалыларды жай айнымалылар тәрізді өзгертуге болады, мысалы:

```
Const   a: real=5.6;...
        a:=(n-1)/k;...
```

Қабаттасқан айнымалылар. Кейде компьютер жадының нақты физикалық адрестерінде орналасқан айнымалыларды программаның басқа айнымалылары тұрған жерде де қатар жариялауға тура келеді. Осындай қабаттастыра жарияланатын айнымалыларды сипаттау да *absolute* түйінді сөзін қосу арқылы *var* конструкциясы көмегімен жүргізіледі. 3.9-суретте айнымалыларды қабаттастыра орналастырудың да екі нұсқасы болатынын көрсететін толық синтаксистік диаграмма келтірілген.

1. Абсолюттік адрес бойынша қабаттастыру. Мұнда *absolute* сөзінен соң қос нүктемен ажыратылған *word* типіндегі екі сан орналасады. Бірінші сан сегмент адресін, ал екінші сан – оның ығыстырылуын көрсетеді (7.1-ді қара). Мұндай жариялау айнымалыны және компьютер жадының осы көрсетілген адреске сәйкес аймағын физикалық түрде байланыстыру болып табылады.

Мысалы:

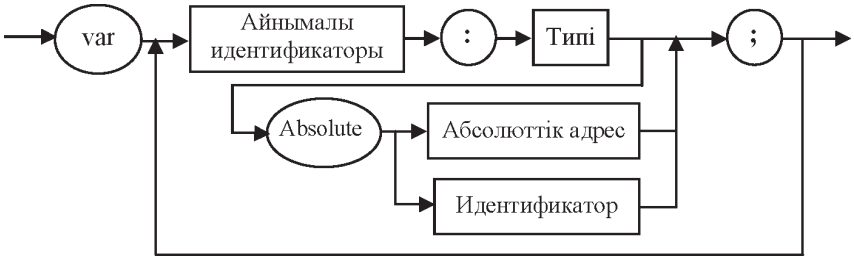
```
Var A: word absolute $0000:$00FF;
L:array[1..2] of char absolute 128:0;...
```

Осы нұсқа, мысалы, операциялық жүйе кестелерін пайдалану үшін қолданылады.

1. Бұрын анықталған айнымалымен қабаттастыру. Мұнда *absolute* сөзінен соң бұрын анықталған айнымалы идентификаторы

орналасады. Осы арқылы *absolute* сөзінде сипатталған айнымалыға одан кейін тұрған идентификаторға сәйкес айнымалы адресі меншіктеледі. Сонымен, компьютер жадында әр түрлі атаулары бар (олар әр түрлі типте де болуы мүмкін) мәліметтерді біріктіру ісі жүзеге асырылады. Мысалы:

```
Var c:byte;
a:real absolute c;...
```



3.9-сурет. Айнымалыны жариялаудың толық синтаксистік диаграммасы

Осылай қабаттастыру нәтижесінде кез келген сәттегі бір айнымалыны өзгерту оның екіншісінің де мәнін өзгертуге себепші болады. Absolute арқылы байланысқан айнымалылардың компьютер жадындағы ішкі орналасу аймағының көлемдері (ішкі орналасуы) жоғарыдағы мысалда көрсетілгендей бірдей болмай қалса, нәтиженің дұрыстығы тексерілмейді. Осылай қабаттасу әрекетіне 5.5 параграфта мысал келтірілген.

3.6 Арифметикалық және логикалық өрнектер

Меншіктеу операторы барлық тілдерде пайдаланылатын негізгі оператор болып табылады. Математикадағы қарапайым теңдеу тәрізді айнымалыларға сандық (символдық та болуы мүмкін) мән беру өрнегін мұнда меншіктеу операторы деп атайды.

Меншіктеу операторы жазылған өрнектердің мәнін есептеп, оны айнымалыға тағайындау үшін қолданылады. Өрнек мәнінің типі айнымалының типіне міндетті түрде сәйкес келуі тиіс. Кейде нақты сан түріндегі айнымалыға бүтін сан мәнін меншіктеуге болады, ондайда бүтін сан нақты санға айналып кетеді. Меншіктеу операторының жазылу ережесі (форматы) төмендегідей болады:

<айнымалы атауы>:=<өрнек>;

мұндағы <айнымалы атауы> – айнымалы идентификаторы, := – меншіктеу белгісі, яғни айнымалының мәні өрнектің есептелген сан мәнін қабылдайды; <өрнек> – арифметикалық өрнек, яғни формула немесе сан. *Өрнек деп арифметикалық операциялардың, яғни амалдардың таңбаларымен біріктірілген айнымалылардың, функциялардың, тұрақтылардың жиынын айтады.* Өрнектердің есептелу барысында амалдардың орындалу реті жақшалардың көмегімен өзгертіледі. *Өрнек операция таңбаларымен және жақшалармен біріктірілген операндтардан тұрады.* Операндтар рөлін айнымалылар, тұрақтылар (константалар) және функциялар атқарады. Өрнек операторлар құрамына кіретін негізгі объект болып табылады.

Жалпы арифметикалық өрнек:

$$E = \begin{cases} \text{тұрақты,} \\ \text{айнымалы атауы,} \\ \text{функция,} \\ \text{өрнек,} \end{cases}$$

түрлерінің бірінде берілуі мүмкін. Жалпы өрнек тек айнымалылардан, тұрақтылардан немесе функциялардан тұруы мүмкін. Мысалдар:

*(5+7*x)/1.8 , (sin(x)+5*cos(2+x))/ln(x), .m.c.c.*

Төмендегі өрнектердің математикада және программалау тілінде жазылулары қатар келтірілген.

$\frac{a+b}{a\text{-}b} \rightarrow (a+b)/(a-b)$ $\qquad e^{x+1} \rightarrow exp(x+1)$

$a^b \rightarrow exp(b*ln(a))$ $\qquad \sqrt{1+\sqrt{x}} \rightarrow sqrt(1+sqrt(x))$

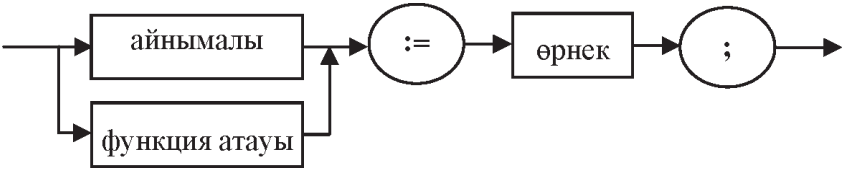
$\frac{a+b}{xy} \rightarrow (a+b)/(x*y)$ $\qquad \sqrt[3]{x} \rightarrow \exp(\ln(x)/3)$

$a^{b^{x+1}} \rightarrow \exp(\ln(a)*\exp((x+1)*\ln(b)))$

Меншіктеу операторы екі міндет атқарады:

1. Айнымалылардың белгілі мәндері бойынша арифметикалық өрнектің сандық мәнін есептейді;

2. Табылған мән айнымалы атауына меншіктеледі (телініп жазылады), яғни анықталған мән сол айнымалыға сәйкес компьютер жады ұяшығына орналасады. Мұнда әдеттегі теңдік "=" белгісімен программалау тіліндегі меншіктеу ":=" белгісін шатастырмау қажет. Олар тек түр жағынан ғана емес мағынасы жағынан да өзгеше. Мысалы, *x*=5 өрнегі *x*-тің мәні 5-ке тең дегенді білдіреді де, *x*=*x*+3 өрнегінің дұрыс мағынасы жоқ. Ал *x*:=5 өрнегі *x* айнымалысына арналған ұяшыққа 5 санын жазамыз дегенді білдіреді. Ал, енді *x*:=*x*+3 өрнегі де дұрыс, өйткені бұл бұрынғы *x* ұяшығында тұрған санға 3 санын қосып *x* ұяшығына қайта орналастыру дегенді білдіреді. Төменде меншіктеу операторының синтаксистік диаграммасы келтірілген (3.10-сурет).



3.10-сурет. Меншіктеу операторының синтаксистік диаграммасы

Меншіктеу операторына мысалдар:

```
i:= i+1;
x:=5.35;
x1:= (-b+sqrt(b*b-4*a*c))/(2*a);
a[2,3]:=m-n;
fun:=false; {fun айнымалысының типі-boolean}
c:=2*pi*r;
R:=19.36;
M:='завод';
Y:=sqrt(sqr(x)+1);
Y1:=3.5+sin(x);
```

Меншіктеу операторы тек арифметикалық өрнектер үшін ғана емес логикалық және символдық өрнектер үшін де пайдаланылады. Мысалы, егер К := А AND В, мұнда А-ақиқат, ал В-жалған болса, онда К жалған мән қабылдайды, К-ақиқат мән қабылдау үшін А және В мәндері бірдей ақиқат болуы қажет, өйткені AND сөзі ЖӘНЕ деген ұғымды білдіреді.

Символдық мән әрқашанда апостроф ‘ ішіне алынып жазылады. Мысалы: B := ‘ T ‘; B5 := ‘9’.

Паскальдағы операциялар орындалуының реттілігі (приоритеті) олардың басымдылық деңгейлерінің төмендеуі бойынша төмендегідей болады:

- бірорындық минус;
- **NOT (ЕМЕС)** логикалық операциясы;
- көбейту тәрізді операциялар тобы ;
- қосу тәрізді операциялар тобы;
- салыстыру (қатынас) операциялары.

Бірорындық минус арифметикалық түрдегі операндтарға қолданылады. **NOT** операциясы – логикалық типтегі операндтар үшін қажет. Егер өрнекте приоритеттері бірдей бірнеше операция тұрса, олар солдан оңға қарай орындалады. Олардың приоритеттерін жай жақшалар арқылы өзгерте аламыз. Логикалық өрнектерде приоритет типтеріне байланысты түсінбеушілік болдырмас үшін жақшалар қойылып отырады.

Мысалы, егер мынадай өрнекте **… (X > 5) AND (Y > 10) …** жақшалар қойылмаса, онда синтаксистік қате шығады, өйткені **AND** операциясының приоритеті қатынас операциясынан (>) жоғары болады.

<көбейту тәрізді операциялар> **:: = * | / | div | mod | and**

<қосу тәрізді операциялар> **:: = + | – | or | xor**

<қатынас операциялары> **:: = = | <> | < | > | <= | >= | in**

Қатынас операциялары барлық стандартты қарапайым типтерге қолданыла береді. Оның нәтижесі әрқашанда логикалық тип болады.

Мысалы, **(5 + 6) < (5 – 6) = TRUE** орындалу нәтижесі – **FALSE**, ал **NOT(8.5 < 4)** нәтижесі – **TRUE**.

Сөз тіркестерін салыстыру солдан оңға қарай әрбір символ бойынша жүргізіледі. Салыстыру кезінде қысқалау сөз тіркесінің оң жағына босорын таңбалары қойылады.

Әдетте өрнек ішінде бірсыпыра операциялар атқарылады, олар да төмендегі сияқты өз приоритеттілігі бойынша орындалады (3.4-кесте):

• арифметикалық операциялар: + (қосу), – (азайту), * (көбейту), / (нақты сандарды бөлу), **div** (бүтін сандарды бөлу), **mod** (қалдық

табу) – бұлар нақты және бүтін сандарға қолданылады, нәтижелері де – сан болады;

• қатынас операциялары: > (үлкен), < (кіші), = (тең), <> (тең емес), >= (кіші емес), <= (үлкен емес) – бұлар сандарға, символдарға, символдардан құралған сөз тіркестеріне және де басқа мәліметтер типтеріне қолданылады, нәтижесі – логикалық типтегі мән;

• логикалық операциялар: **and** (және), **or** (немесе), **xor** (арифметикалық немесе), **not** (емес) – бұлар с логикалық айнымалылар мен константаларымен жұмыс істейді, нәтижесі – логикалық тип;

• разрядтық операциялар: **and** (және), **or** (немесе), **xor** (арифметикалық немесе), not (емес), **shr** (оңға ығыстыру), **shl** (солға ығыстыру) – бұлар бүтін сандармен ғана жұмыс істей алады, нәтижесі – бүтін сан;

• тіркестік (жолдық) операция: + (сөз тіркестерін біріктіру) – символдар және сөз тіркестерімен жұмыс істейді, нәтижесі – сөз тіркесі (4.5 параграфты қара);

• жиындармен атқарылатын операциялар: + (біріктіру), – (толықтыру), * (қиылыстыру), нәтижесі – жиын; **in** (элементтің жиынға кіретінін анықтау), нәтижесі – логикалық типтегі мән (4.7 параграфты қара);

• сілтеуіштермен (указатели) атқарылатын операциялар: @ (программалық объектінің адресін анықтау), нәтижесі – адрес (7.1 параграфты қара).

3.4-кесте

Операциялар	Приоритеті
@, not	1
*, /, div, mod, and, shr, shl	2
+, -, or, xor	3
>, <, <>, =, <=, >=, in	4

3.4-кестеде жоғарыдағы операциялардың приоритеттері (орындалу реттілігі) көрсетілген.

Өрнектерде стандартты (1 қосымшаны қара) және программалаушы анықтаған функциялар (5 тарауды қара) да қолданылады, олардың приоритеті – ең жоғарғы болып саналады.

Арифметикалық операциялар. Арифметикалық операциялармен біріктірілген өрнектер солдан оңға қарай «жолдарға» орналастырылып, олардағы операциялардың атқарылуы жақшалар бойынша немесе өз приориттеріне сәйкес орындалады. Әр түрлі приоритеттегі операцияларды орындайтын программалар жазғанда мұқият болу керек. Мысалы:

• a+b/c орындалғанда, алдымен бөлу амалы соңынан қосу амалы атқарылады;

• (a+b)/c*d өрнегі a+b қосындысының c-ға бөлініп, шыққан нәтижесі d-ға көбейтілетінін білдіреді.

Арифметикалық өрнектерді программалағанда, келесі операцияларды орындау ережелерін де есте сақтау керек.

1. «Бүтін бөлу» және «бөлгендегі қалдықты табу» операциялары тек бүтін сан типіндегі операндтарға ғана қолданылады, мысалы: 6 div 4 = 1, ал 6 div 4 = 2. Егер айнымалыларға осы операциялар қолданылатын болса, онда олар бүтін сан типінде жариялануы тиіс, мысалы:

```
Var     i, n: Integer;...
        n mod 2;...
```

2. Әр түрлі типтегі сандар қатысатын арифметикалық операцияларды орындағанда, автоматты түрде типтерді түрлендіру жұмысы атқарылады:

• егер бір операнд бүтін сан типінде, ал екіншісі – нақты сан типінде болса, онда бүтін типтегі айнымалы нақты типке түрлендіріледі де, операция нәтижесі – нақты сан типіндегі мән болады;

• егер операндтар ретінде әр түрлі нақты сандар мен бүтін сандар пайдаланылса, онда олардың мәндері солардың ішінде кездесетін ең үлкен разрядты типке түрлендіріледі. Мысалы, өрнекте **double**, **extended** және **real** типіндегі айнымалылар болса, онда олардың барлығы да **extended** типіне ауыстырылады, нәтижесі де осы типте болады.

Қатынас операциялары нақты және бүтін сандар үшін, логикалық мәндер, символдар кодтары, сөз тіркестері және жиымдар үшін анықталған. Олардың нәтижесі – егер қатынас ақиқат болса – **true**, ал егер жалған болса – **false** болады.

Сандардың бейнелену разрядтылығы шектеулі болғандықтан, нақты сандар дәл өрнектелмейді, сондықтан нақты сандардың

теңдігін/теңсіздігін тексеру сандарды аздап іріленділеріп дөңгелектеу арқылы атқарылуы тиіс, мысалы:

```
(x-y)>1e-10 { x <> y орнына }
(x-y)<1e-10 { x = y орнына }
```

Егер осылай дөңгелектеу көрсетілмесе, онда ол санның бөлшегіндегі таңбаларға байланысты (3.3-кестені қара) автоматты түрде анықталады да, дәлдігі ойлаған жерден шықпай қалуы мүмкін (2.7 параграфты қара).

Логикалық операциялар boolean типіндегі мәндермен орындалады. Егер логикалық операцияларда операндтар ретінде қатынас операциясы қарастырылса, олардың приоритеті төмен болғандықтан, жақшалар қолданылуы керек. Мысалы, *x* [a, b] интервалына кіретінін анықтағанда, ақиқаттық мәні қолданылатын логикалық өрнек былай болып жазылуы тиіс:

```
(x >= a) and (x <=b.
```

Разрядтары бойынша атқарылатын логикалық операциялар мен ығыстыру операциялары бүтін сандармен орындалады. Алынатын нәтижесі де бүтін сан болады. Ал ығыстыру операциясының екінші операнды биттер түрінде берілген бірінші операндты неше разрядқа ығыстыру керек екендігін анықтайды :

5 *shl* 4 – екілік биттер түрінде бейнеленген 5 саны (00000101) солға қарай 4 разрядқа ығыстырылады да (01010000), ол сол санды 2^4=16-ға көбейткенге сәйкес келіп, нәтижесі – 80 болып шығады ($01010000_2 = 2^6+2^4 =80$).

Қалған операциялар соларға арналған тарауларда қарастырылған.

Сонымен меншіктеу операторы айнымалылардың мәнін өзгертеді екен. Бұл оператор орындалуы барысында меншіктеу белгісінен оң жақта тұрған өрнек мәні есептеліп, оның нәтижесі сол жақтағы айнымалы мәні болып саналады. Оператор соңына нүктелі үтір қойылады. Мысалы:

```
a.   Var  a, b, c:real;
     Begin...
          c:=(a*a-sin(b))/(a+25.1);...
b.   Var  v:boolean; a:integer; b:real;
     Begin
```

```
a:=8; b:=1.5;
v:=(a>5)and(b>=8); {v false мәнін
қабылдайды}
...
```

Меншіктеу операциясының нәтижесі дұрыс болуы үшін, оның құрамындағы өрнек пен айнымалы бірдей типте немесе бір-біріне түрлендірілетін сәйкес типтерде болуы керек.

Сәйкес келетін типтерге мыналар жатады:

- барлық бүтін сандар типтері;
- барлық нақты сандар типтері;
- белгілі бір негізгі (базалық) типке кіретін кесінді мен базалық тип;
- бір базалық тип кесінділері;
- символ және сөз тіркесі.

Меншіктеу операциясының сол жағы мен оң жағында орналасқан мәндер типі әр түрлі болса, өрнек нәтижесі айнымалы типіне келтіріледі. Мысалы:

```
Var
  L:longint; {айнымалы типі longint}
  E,x:extended; {айнымалылар типтері extended}
  I:integer; {айнымалы типі integer}
  R:real; {айнымалы типі real}
Begin...
  R:=I*E/(x+L);... {оң жақтағы өрнек
  нәтижесінің типі – extended, бірақ R
  айнымалысы типі – real, сондықтан өрнек мәні
  де осы типке түрленеді}
End.
```

Егер меншіктеу операциясының сол жағы мен оң жағында орналасқан мәндер типі бір біріне сәйкес келмесе, онда оларды *тікелей түрлендіру* керек.

Тікелей түрлендіру төмендегі арнайы функциялар арқылы орындалады:

- ***Trunc(x)*** – нақты санның бөлшегінің разрядтарын алып тастау арқылы нақты сан типін бүтін сан типіне түрлендіреді;
- ***Round(x)*** – аралас санды дөңгелектеу арқылы нақты сан типінен бүтін сан типіне түрлендіреді;

- ***Ord(x)*** – реттелген типтегі мәнді оның нөміріне түрлендіреді;
- ***Chr(x)*** – ASCII кестесіндегі символ нөмірін сол символдың өзіне түрлендіреді.

Мысалы:

```
a.    Var a,b,c:read; n:integer;
      Begin
      Read(a,b); {сандар бір жолдан немесе екі
      жол арқылы да енгізіле береді }
      ReadLn(c,n);... { сандар алдыңғы
      сандар енгізілген жолдан да енгізілуі
      мүмкін }
b.    Var a:real; c:char;
      Begin...
      Read(a);...
      Write ('Жалғастыру керек пе? (y/n)');
      Read(c); {енгізу кезінде программа
      күтпейді, ол алдыңғы <Enter> пернесін
      басқан кездегі соның кодын буферден алады}
```

Бұларға қоса тікелей түрлендіру үшін стандартты немесе тұтынушы анықтаған функцияларды пайдалануға болады. Бұндай түрлендіру тәсілін кейде *автоанықтау* деп те атайды, мысалы:

```
   Var                                    h:char;
... h:=Char(65);... {h 'A' мәнін қабылдайды}
```

Осындай түрлендіруден кейін мән емес, тек оның типі өзгереді. Бірақ мұның нәтижесінде мәннің көлемі ұлғаюы немесе кішіреюі мүмкін. Көлемі кішрейетін кездерде түрлендірілетін санның таңбасы да өзгеруі мүмкін. Ал көлемі үлкейетін сәттерде таңба өзгертілмейді.

Мысалы:

```
Type
Month=(Jan,Fab,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct
,Nov,Dec);
Var M:Month;
A,B:integer;
C:char;
L:longint;
```

```
Begin
A:=10; C:='E';
B:=Integer(C); {E символының коды 69 –
ұзындығы 2 байт}
M:=Month(A-2); {мәні Aug}
L:=LongInt(M); { мәні 7}
```

Бақылау сұрақтары

1. *Өздеріңізге белгілі бүтін типтерді диапазондарының өсу реті бойынша атап шығыңыздар.*
2. *Өздеріңізге белгілі нақты типтерді диапазондарының өсу реті бойынша атап шығыңыздар.*
3. *Бірнеше бүтін және бірнеше нақты типтерді қолданудың қажеттілігін түсіндіріңіз.*
3. *Өзіңізге белгілі арифметикалық операциялардың жазылуы мен орындалуын түсіндіріңіз?*
4. *Өзіңізге белгілі стандартты арифметикалық функцияларды атаңыз.*
5. *Реттік типтегі шамаларға арналған функцияларды атаңыз.*
6. *Мәндер типін түрлендіруге арналған функцияларды атаңыз.*
7. *Бүтін сандардың берілуі жайлы айтыңыз.*
8. *Нақты сандардың берілуі жайлы айтыңыз.*
9. *Форматпен шығару не үшін қолданылады?*
10. *Тип-диапазон дегеніміз не?*
11. *Төмендегі символдар тіркестерінің қайсылары идентификатор бола алады?*
    a3, a^3, a·3, x_3, l00s, s100, min, π, alfa, sin, sinx, sin(x), sin5sin, MyVariable, My_Variable, Stop, Lab12, 72_3, MyVariable#, My-Variable, 1_2_3, Center, Month–Week, Year, Monday@Sunday, This_a_Bad_Identifier, 1Color, _3, #5, __.
12. *Төмендегі мысалдарды Паскальдағы өрнектерді құру ережелеріне сәйкес жазып шығыңдар:*
    Келесі өрнекті кәдімгі математикалық түрде жазу керек:
    *(-b - sqrt((sqr(b) -4*a*c))/(2*a).*
13. *Мына өрнектің (sin(sqr(x) – 1) + 2* abs(y))/ cos(2 + y) мәнін айнымалылар: x = 1, y = -2 болғанда, есептеу керек.*
14. *Мына өрнекті a / b * c / d * e /f* h төмендегі бөлшекке сәйкес келетіндей етіп жақшалармен толықтырып шығыңдар*

$$\frac{a}{b \cdot \frac{c}{d \cdot \frac{e}{f \cdot h}}}$$

15. *Келесі өрнектерді Паскаль тілінде жазу керек:*

$$а)\ln\left(\sqrt{e^{(x-y)}}+X^{|y|}+Z\right);\quad б)\frac{Y^{x+1}}{\sqrt{|Y-2|+3}}+\frac{X+\frac{Y}{2}}{2|X+Y|};$$

$$в)Y+\frac{X}{Y+\frac{X-2}{Y+X^3}};\ г)(1+Y)\frac{X+\frac{Y}{X^2-1}}{Y^{x-2}+\frac{1}{X^2+4}};$$

$$д)\frac{\sqrt{X-1}-\ln(|Y|)}{1+\frac{X^2}{2}+\frac{X^3}{3}};\ е)\frac{A-B}{1-\frac{1}{X}}.$$

1. *Келесі операторлар тізбегін орындағанда, экранға қандай мәндер шығарылады?*
   *x:=(sin(sqr(l)-1)+2*abs(-2))/cos(2-2);*
   y:=x(sqr(2));*
   write(x,y);
2. *Егер енгізілетін мәліметтер тізбегі мынадай екі саннан: 0.5 және 0.16 тұрса, келесі операторлар тізбегі экранға қандай мән шығарады?*
   read(x,y);
   *z:=sqr(sqr(x))*sqrt(y);*
   write(z);
3. *Егер енгізілетін мәліметтер тізбегі мынадай екі сан: 5.2 және 18.7 болса, төмендегі операторларды орындағанда, x және y айнымалыларының мәндері қандай болады?*
   read(x,y);t:=x;x:=y;y:=t;

1. *Егер бастапқы мәліметтер мынадай сандардан:*
   а) 3.5 және 2.4;
   б) 6.4 және -10.1 тұратын болса, төмендегі операторлар орындалған соң, экранға не шығады?
   read(x,y); x:=x+y; y:=x-y;
   x:=x-y; write(x,y);
5. *Берілген екі санды енгізіп, солардың арифметикалық және геометриялық орталарын есептеп (оң сандар енгізіледі деп саналады), нәтижелерін экранға шығаратын операторлар тізбегін жазып шығыңдар.*

4. ТУРБО ПАСКАЛЬДІҢ СТАНДАРТТЫ МОДУЛЬДЕРІ

4.1 Математикалық функциялар

Функция аты	Функция қызметі	Нәтиже типі
Abs(X)	Аргументтің абсолюттік шамасы (модулі) Abs(-3.5)=3.5	X типімен сәйкес келеді
ArcTan(X)	Аргументтің радианмен алынған арктангенсі ArcTan(1)=7.8539816340E-01	Real
Cos(X)	Аргументтің радианмен алынған косинусы Cos(PI/3)=5.0000000000E-01	Real
Exp(X)	Аргументтің экспонентасы (E-нің X дәрежесі) Exp(1)=2.7182818285E+00	Real
Ln(X)	Натуралдық логарифм Ln(10)=2.3025850930E+00	Real
PI	Pi санының мәні PI=3.1415926536E+00 (дәлірек 3.1415926535897932385)	Real
Random	0 мен 1 арасындағы кезейсоқ сан	Real
Random(X)	0 мен X арасындағы кезейсоқ сан	Word
Sin(X)	Аргументтің радианмен алынған синусы Sin(PI/3)=8.6602540378E-01	Real
Sqr(X)	Аргументтің квадраты Sqr(-12)=144	X типімен сәйкес келеді
Sqrt(X)	Аргументтің квадрат түбірі Sqrt(841)=2.9000000000E+01	Real

Турбо Паскаль тілінде стандартты функция түрінде жазылмаған математикалық функцияларды есептеу үшін, оларды стандартты функциялар арқылы өрнектеу керек. Мысалы:

tg(X)=**Sin(X)/Cos(X)**
lg(X)=**Ln(X)/Ln(10)**
X^n=**Exp(n*Ln(X))**

Random немесе **Random(X)** функцияларын қолданар алдында, осы функциялар генерациялайтын кездейсоқ сандар тізбегінің

сәйкес келмеуін қадағалайтын, **Randomize** процедурасын (параметрсіз процедура) іске қосу керек.

4.2 Дөңгелектеу функциялары және типтерді түрлендіру

Функция аты	Аргументтің типі	Нәтиже типі	Функцияның қызметі
Chr(X)	Бүтін Chr(66)=’B’ Chr(Ord(‘M’)=’M’	Char	ASCII-кодты символға түрлендіру (0-255)
Frac(X)	Real Frac(-12.34)=-.34	Real	Нақты X санының бөлшегін алу
Int(X)	Real Int(-12.34)=-12	Real	Нақты санның бүтін бөлігін алу
High(X)	Реттік жиым, жол, ашық жиым	Аргументпен сәйкес келеді	Элемент нөмірінің ең үлкен мәнін алу
Low(X)	Реттік жиым, жол, ашық жиым	Аргументпен сәйкес келеді	Элемент нөмірінің ең кіші мәнін алу
Ord(X)	Реттік Ord(‘A’)=65 Ord(Chr(86))=86	LongInt	X символына сәйкес келетін ASCII-кестесіндегі оның кодын анықтау
Round(X)	Real Round(-1.2)=-1 Round(-1.5)=-2 Round(1.2)=1 Round(1.5)=2	LongInt	X-ті ең жақын бүтінге дейін дөңгелектеу
Trunc(X)	Real Trunc(-1.2)=-1 Trunc(-1,5)=-1 Trunc(1.2)=1 Trunc(1.5)=1	LongInt	Санның бөлшегін алып тастау

4.3 Реттік типтегі процедуралар және функциялар

Функция аты	Функция қызметі
Odd(X)	Аргументтің тақ сан екендігін тексереді Odd(0)=false; Odd(1)=true; Odd(2)=false; Odd(-1)=true;
Pred(X)	Аргументтің алдыңғы мәнін береді Pred(10)=9; Pred(-10)=-11
Succ(X)	Аргументтің келесі мәнін береді Succ(10)=11; Succ(-10)=-9

Процедура аты	Процедураның қызметі
Dec(X [,dx])	X айнымалысының мәнін dx-ке кемітеді (егер dx параметрі берілмесе, онда ол –1-ге кемітеді) k:=5; Dec(k)=4; Dec(k,2)=3; Dec(k,-2)=7
Inc(X [,dx])	X айнымалысының мәнін dx-ке өсіреді (егер dx параметрі берілмесе, онда +1-ге өсіреді) k:=5; Inc (k)=6; Inc (k,2)=7; Inc (k,-2)=3

4.4 Сөз тіркестерімен жұмыс істейтін процедуралар мен функциялар

Функция аты	Функция қызметі
Concat(<1 тіркес>,<2 тіркес>**,..)**	Сөз тіркестерін біріктіру Concat('A','BC','_1')='ABC_1'
Copy(<тіркес>,<позиция>,<саны>**)**	Сөз тіркесінің берілген бөлігін көрсетілген позициядан бастап көшіру Copy ('INFORMATION',3,5) = 'FORMA'
Length(<тіркес>**)**	Ағымдағы сөз тіркесінің ұзындығын анықтау Length('Астана')=6
Pos(<ішкі тіркес>,<толық сөз тіркесі>**)**	Сөз тіркесіне оның ішкі бөлігінің кіру позициясын анықтау Pos('т','Анықтама')= 5 Pos('к','Анықтама')= 0

Conca функциясы үшін сөз тіркесінің жалпы ұзындығы 256 байттан аспауы керек. Сөз тіркестері үшін орындалатын «+» белгісі оның символдары үшін конкатенация (біріктіру) операциясының атқарылатынын білдіреді.

Copy функциясы үшін, көрсетілген позиция сөз тіркесінің ұзындығынан үлкен болса, функция нәтижесі бос жол болады. Егер <позиция>+ <саны> сөз тіркесінің ұзындығынан үлкен болса, онда тек соңғы символдар көшіріледі. Егер <позиция> нөмірі [1,255] аралығына жатпаса, онда программа орындалуында қате болады.

Процедура аты	Процедураның қызметі
Delete(<тіркес>,<позиция>,<саны>**)**	Берілген позициядан бастап сөз тіркесінің бөлігін алып тастау 1) S:=’abcdefgh’; Delete(S,2,4); Нәтиже: S=’afgh’ 2) S:=’abcdefgh’; Delete(S,2,10); Нәтиже: S=’a’
Insert(<ішкі тіркес>,<толық тіркес>, <позиция>**)**	Берілген позициядан бастап бір сөз тіркесін екінші сөз тіркесіне кірістіру S:=’abcdefgh’; Insert(‘XXL’,S,3); Нәтиже: S=’abXXLcdefgh’
Str(<сан>,<символдар тіркесі**>)**	Сандық мәнді символдар тіркесіне түрлендіру 1) Str(567,A); Нәтиже: A=’567’ 2))B:=567; {B:integer} Str(B:5,A); Нәтиже: A=’_ _567’ 3) B:=5.67E+3; {B:real} Str(B:8:0,A); Нәтиже: A=’_ _ _ _5670’
Val (<сөз тіркесі>,<сан>,<код**>)**	Сөз тіркесін сандық мәнге түрлендіру (егер қате жоқ болса, онда <код>=0) 1)A:=’135’; Val(A,R,Code); Нәтиже: R=135; Code=0 2)A:=’_135’; Val(A,R,Code); Нәтиже: R=анықталмаған; Code=1 3)A:=’2.5E+4’; Val(A,R,Code); Нәтиже: R=25000; Code=0

Delete процедурасы үшін, егер <позиция> символдар тіркесінің ұзындығынан үлкен болса, онда ол өзгермейді. Егер <по-

зиция> + <саны> тіркес ұзындығынан үлкен болса, онда көрсетілген позициядан бастап сөз тіркесінің соңы өшіріледі. Егер <позиция> нөмірі [1,255] аралығына жатпайтын болса, онда программа орындалуында қате пайда болады.

Insert процедурасында, егер бір символдар тіркесін кірістіру нәтижесінде бастапқы берілген сөз тіркесінің максимальды ұзындығынан артық тіркес шықса, онда кірістірілген сөз тіркесінің соңғы символдары қойылмайды. Егер <позиция> нөмірі бастапқы тіркестің *Length* функциясымен анықталған нақты ұзындығынан артық болса, онда нәтиже жалғастырылған (сцепленная) сөз тіркесі болып саналады.

Val процедурасы үшін сөз тіркесіне түрлендірілетін сан алдында және соңында бос орындар болмауы тиіс.

4.5 Басқа процедуралар мен функциялар

Функция аты	Модулі	Процедура немесе функцияның қызметі
Keypressed	Crt	Функция. Пернетақтадан бір перне басылса, True мәнін береді, кері жағдайда False мәнін береді
ReadKey	Crt	Функция. Кез келген перне басылғанша, программаның орындалуын тоқтата тұрады.
SizeOf(X)	System	Функция. Аргументтің ұзындығына сәйкес байт санын береді
WhereX	Crt	Функция. Ағымдағы терезеге байланысты, курсор тұрған позицияның горизонталь координатасын (нөмірін) береді
WhereY	Crt	Функция. Ағымдағы терезеге байланысты, курсор тұрған позицияның вертикаль координатасын береді
ClrScr	Crt	Процедура. Экранды тазалайды
Delay (X)	Crt	Процедура. Программаның орындалуын X миллисекундке тоқтатады

Exit	System	Процедура. Процедураның, функцияның немесе негізгі программаның уақытынан бұрын аяқталуын іске асырады
FillChar(X,COUNT,Value**)**	System	Процедура. Х айнымалысының тізбек байттарының берілген COUNT санын Value мәнімен толтырады
GetDate(<жыл>,<ай>, <күн>,<апта күні**>)**	Dos	Процедура. Ағымдағы датаның мәнін береді
GotoXY(X,Y**)**	Crt	Процедура. Курсорды экранның координатасы көрсетілген жеріне жылжытады
Window(X1,Y1,X2,Y2**)**	Crt	Процедура. Экрандағы мәтіндік терезе көлемін анықтайды (X1,Y1- сол жақ жоғарғы бұрыш координатасы; X2,Y2- оң жақ төменгі бұрыш координатасы)

4.6 Енгізу-шығару процедурасы

Турбо Паскаль тілінде мәндерді енгізу стандартты **READ** немесе **READLN** процедуралары (операторлары), ал мәндерді шығару **WRITE** немесе **WRITELN** процедуралары (операторлары) арқылы орындалады. **READ** және **READLN** процедуралары символдарды (мәндер типі **CHAR**), сөз тіркестерін (мәндер типі **STRING**) немесе сандық мәндерді (мәндер типі **INTEGER**, **BYTE**, **REAL** және т.б.) енгізу үшін қолданылады.

READ процедурасын шақыру:

READ ([<файл аты**>,]**<айнымалылар тізімі**>);**

READLN процедурасы үшін сәйкесінше:

READLN ([<файл аты **>,]**< айнымалылар тізімі **>);**

Егер <файл аты> көрсетілмесе, онда мәндерді оқу стандартты **INPUT** файлынан жүргізіледі, бұл жағдайда стндартты құрылғы **INPUT** файлымен байланысқан пернетақта немесе дисплей болып саналады.

Әр енгізу операторының тізімінде көрсетілген айнымалыларға меншіктелетін тұрақты мәндері бар өз мәндер жиыны бар. Енгізу

тізіміне мәндерді меншіктеу айнымалылардың орналасу ретіне байланысты, солдан оңға қарай жүргізіледі.

Есте сақтаңыздар:

- Айнымалылар және тұрақтылар типі сәйкес болу керек (тек **REAL** типті мәндерін енгізерде, **INTEGER** типті айнымалылар мен тұрақтыларды көрсетуге болады).
- Енгізілетін *сандық мәндер* бір немесе бірнеше бос орынмен ажыратылуы керек; санның таңбасын және цифрларын бос орын арқылы бөлуге болмайды.
- Егер сөз тіркесі, яғни жол енгізілетін болса (мәндер типі **STRING**), онда **READ** операторы **VAR** сипатталуында көрсетілген максимальді ұзындықтан аспайтын символдар тізбегін ғана оқиды.
- Символдар тізбегі (мәндер типі **CHAR** немесе **STRING**) олар енгізілмеген жағдайда, бос орын ретінде қабылданады.

Мысал.

А) Сандық мәндерді енгізу:

```
VAR B,A,D: REAL;
    K:INTEGER;        Енгізілетін мәндер:
        ...           2.5 –4.95  20  1.25E2
   READ(A,D);         Енгізгеннен кейін:
   READ(K,B);         A=2.5; D=-4.95; K=20; B=125
```

Ә) Сандық және тіркестік (жолдық) мәндерді енгізу.

```
VAR A: REAL;
    B:INTEGER;
    C1,C2,C3: CHAR;              Енгізілетін мәндер:
    D: STRING[5];                2.5 10 ЖОЛБАРЫС
    . . .                        Енгізгеннен кейін:
    READ(A,B,C1,C2,C3,D);        A=2.5; B=10; C1=' ';
                                 C2='K';C3='L';
                                 D=' ЛБАРЫ'
```

Мысалдан көріп отырғандай, мәндерді араластырып (сандық және жолдық) енгізу дұрыс орындалмайды. Егер енгізілетін мәндер тізбегінен кейін бос орын қалдырылмаса, онда енгізуде қате (ERROR 106) орын алады. Сондықтан сандық мәндер мен тіркестік немесе символдық мәндерден жеке-жеке бөле отырып енгізу керек.

READ операторын орындаған кезде енгізу жолының соңы (<Enter> пернесін басу), мәндер элементтерін ажырататын бос орын енгізумен бірдей болып саналады, сондықтан енгізудің келесі жолына көшу орындалмайды. Ал енгізілетін мәндер тізбек бойынша сәйкес айнымалыға меншіктеледі

Мысалы, ұқсас операторлар үшін енгізілетін мәндер тізбегі әр түрлі болып келуі мүмкін:

```
READ(A,B,C);        Енгізілетін мәндер:  2 9 5 3 7  немесе
READ(D,E);          Енгізілетін мәндер:  1 жол: 2 9 5
                                         2 жол: 3 7
```

READ оераторының **READLN** операторынан ерекшелігі, **READLN** операторында соңғы айнымалы оқылғаннан кейін сөз тіркесінің қалған бөлігі қабылданбайды. Бұдан кейінгі **READ** немесе **READLN** операторы мәндерді жаңа жолдың басынан бастап оқиды, демек **READLN** операторы жолдың соңын (<Enter> пернесін басу) қабылдамайды. Жолдың соңына жетіп, енгізілетін мәндердің келесі жолына ауысуда **READLN** операторын параметрсіз қолдануға болады; мұндайда енгізу мәндерінің келесі жолына шартсыз көшу орындалады.

Мысалы, екі өлшемді жиымның элементтерін енгізудің әр түрлі тәсілдерін қарастырайық:

```
1)FOR I:=1 TO 2 DO
        BEGIN
          FOR J:=1 TO 3 DO
             READ (A[I,J]);
               READLN
  END;
```

Енгізілетін мәндер: 3 5 1
-4 7 9

```
2)FOR I:=1 TO 2 DO
   FOR J:=1 TO 3 DO
     READ (A[I,J]);
```

Енгізілетін мәндер: 3 5 1
-4 7 9

```
3)FOR I:=1 TO 2 DO
   FOR J:=1 TO 3
     READLN(A[I,J]);
```

Енгізілетін мәндер: 3
5
1
-4
7
9

Келесі мысалда **READLN** операторын қолданғанда, жолдың соңындағы мәндердің қабылданбайтындығын анық көруге болады.

```
VAR A,B,C,D:INTEGER;

          ...
     READLN(A,B,C);
       READLN(D);
```

Енгізілетін мәндер:
1 жол: 10 20 30 40 50
2 жол: 60

Нәтиже:
A=10;B=20;C=30;D=60

Мәндерді шығару процедурасы

WRITE процедурасы (операторы) келесі типтегі өрнектерді шығаруға арналған: Integer, Byte, Real, Char, String, Boolean және т.б.

WRITE ([< файл немесе құрылғы аты **>,] <**өрнектер тізімі**>);**

Егер <файл аты> көрсетілмесе, онда шығару стандартты **OUTPUT** файлына (дисплей экранына) орындалады. Егер <файл аты> көрсетілсе, онда бұл файл сипатталуы немесе алдын ала дайындалуы қажет.

Баспадан шығару үшін **LST** логикалық құрылғысы қолданылады. Мұндайда стандартты **PRINTER** модулі іске қосылуы керек (демек, программа басында **Uses Printer** сөздері тұруы керек).

WRITE операторы тізімдегі өрнектердің мәндерін ағымдағы жолға ол толғанша шығарады. Егер бұдан кейін тағы да шығару операторы тұрса, ал ағымдағы жол толмаған жағдайда, мәндерді осы жолға жалғастырып шығара береді.

Мысалы:

```
X:=5; Y:=10;
WRITE ('X=', X);
WRITE (' Y=',Y);
```

Экранда: X=5 Y=10

Турбо Паскаль тілінде стандартты типтегі шамаларды баспаға шығару үшін белгілі бір сандар позициясы беріледі. Шығарылатын мәндер үшін өріс енін (позиция санын) формат арқылы анықтауға болады.

Форматы бар шығару операторының жазылуы:

WRITE ([< файл немесе құрылғы аты >,]R_1:N_1,R_2:N_2,…, R_m:N_m);

мұндағы - R_1,R_2,…,R_m – шығарылатын айнымалылар атаулары;

N_1,N_2,…,N_m – өріс ендері.

Егер өріс ені мәннен үлкен болса, онда мәндердің сол жағында бос орын қалдырылады. Ал, шығарылатын мән өріс еніне сыймаса, онда формат қабылданбайды да, баспаға нақты мәндер толық шығарылады.

Нақты мәндерді шығару ерекшеліктері

Егер нақты типтегі айнымалы сипатталса, онда оны шығарудың төмендегідей нұсқаларын пайдалануға болады:

2. Write(R); Шығару қалыпты түрде іске асырылады (экспоненциальды формада):

{ | } d.ddddddddd E{ +| } dd

3. Write(R:N); Бүтін сан шығарудың қалыпты түрі, өріс ұзындығы N, оң жақ шекарасы бойынша тураланған. N-нің ең кіші мәні 8-ге тең. Өріс ені бұдан кіші болған жағдайда, компилятор мәндер разрядын 8-ге дейін автоматты түрде өсіреді.

4. Write(R:N:M); Бекітілген нүкте арқылы шығару, нүктеден кейін M (0<=M<=24) ондық таңба көрсетіледі, мұнда N>M+2 болуы тиіс (2 орын – сан таңбасы мен ондық нүкте үшін).

Мысалы,

```
VAR B,D:REAL;
    C:INTEGER;
    A:STRING[10];
                . . .
            A:='КІТАП';
            B:=1253E-5;
            C:=12;
            D:=1253E2;
  WRITE (LST,'B=',B:10:3,' C=',C:8,'A=',A:7,'
B1=',B:8,'D=',D:6);
```

Баспаға (мұндағы _ экранда көрінбейтін бос орын символын білдіреді):

B=_____0.013_C=_____12_A=__кітап_B1=_1.3E-02_D=_1.3E+05

түрінде шығарылады.

WRITELN процедурасының шығарылу форматы да осыған ұқсас:

WRITELN ([<файл немесе құрылғы аты **>,]**<өрнектер тізбегі**>);**

Бұл процедура шақырылғанда, файлдың ағымдағы жолының құрастырылуы аяқталады. Келесі **WRITE** немесе **WRITELN** операторы жаңа жол құрастырады. **WRITELN** операторын параметрсіз қолдануға да болады.

Мысалы, **WRITE** және **WRITELN** операторларын қатар қолданғанда, мәндерді жол-жолмен реттеп шығаруға болады:

```
VAR A,B,C,D,E:INTEGER;
BEGIN
    A:=1;B:=2;C:=3;D:=4;E:=5;
    WRITELN ('A=',A,'B=',B);
    WRITE('C=',C);
    WRITELN('D=',D,'E=',E);
END.
```

Нәтиже экранда екі жолға шығады:

```
_A=1_B=2
_C=3_D=4_E=5
```

A (M,N) бүтін сандар матрицасын экранда тікбұрышты кесте түрінде шығаруды төмендегідей түрде іске асыруға болады:

```
        . . .
FOR I:=1 TO M DO
BEGIN
    FOR J:=1 TO N DO
       WRITE(A[I,J]:5);
       WRITELN
END;
```

A(M,N) нақты сандар матрицасын, үтірден кейін бір разрядпен, кесте түрінде принтерге шығару программасы:

```
USES PRINTER;
VAR A:ARRAY[1..10,1..10]OF REAL;
    M,N:INTEGER;
BEGIN
  READLN(M,N);
         FOR I:=1 TO M DO
         FOR J:=1 TO N DO
       READ(A[I,J]);
    FOR I:=1 TO M DO
    BEGIN
      FOR J:=1 TO N DO
        WRITE(LST,A[I,J]:6:1);
        WRITELN(LST)
    END;
     READKEY
END.
```

Бақылау сұрақтар

1. *Turbo Pascal тілінде жазылған программа қандай бөлімдерден тұрады?*
2. *Программаның қандай бөлімдерін міндетті түрде жазу керек?*
3. *Идентификатор дегеніміз не?*
4. *Идентификаторға қандай шектеулер қойылады?*
5. *Өзіңізге белгілі арифметикалық операциялар мен функцияларды атаңыз.*
6. *Бүтін және нақты айнымалылар қалай хабарланады?*
7. *Айнымалының мәні қалай анықталады?*
8. *Мәндерді енгізу операторын сипаттаңыз.*
9. *Мәндерді шығару операторын сипаттаңыз.*
10. *Меншіктеу операторын және оны қолдану ережелерін сипаттаңыз.*

Тапсырмалар

1. *Екі нақты оң сандар берілген. Осы сандардың қосындысын, айырмасын, арифметикалық орташа мәнін және көбейтіндісін табыңыз.*

2. *Тікбұрышты үшбұрыштың катеттері берілген. Үшбұрыштың гипотенузасы мен ауданын табыңыз.*
3. *Тікбұрышты үшбұрыштың ауданын Герон формуласы бойынша есептеңіз.*
4. *Физикалық дененің t уақытының ішінде жүріп өткен арақашықтығын есептеңіз. Дененің бастапқы кездегі жылдамдығы v0 және ол бірқалыпты үдеумен қозғалады.*
5. *Дененің H биіктіктен еркін құлау уақытын анықтаңыз.*
6. *Биіктігі h, табандары a, b трапецияның ауданын есептеңіз.*
7. *Жақтаулары a және b тікбұрыштың периметрін және ауданын есептеңіз.*
8. *Салмағы m сұйықтықты t1-ден t2 температураға дейін қыздыруға қажет жылу мөлшерін анықтаңыз.*
9. *Жақтаулары a, b және c болып келген параллелепипедтің көлемін есептеңіз.*
10. *Экранға өз атыңызды шығарыңыз.*
11. *A және B айнымалыларының мәндерін алмастырыңыз. Мәндерді уақытша сақтау үшін r айнымалысын қолданыңыз.*

5. ТУРБО ПАСКАЛЬ ТІЛІНІҢ БАСҚАРУ ОПЕРАТОРЛАРЫ

5.1 Шартсыз көшу операторы. Белгілер. Бос оператор. Құрама оператор

GOTO шартсыз көшу операторы операторлардың рет-ретімен орындалуын бұзып, келесі атқарылуды осында көрсетілген белгісі бар операторға ауыстыру ісін орындайды. Бір оператор бірнеше белгімен белгіленуі мүмкін. Оператордың жазылуы:

GOTO <белгі>**;**

<белгі> – бұл программаның белгіні сипаттау (**LABEL**) бөлігінде міндетті түрде көрсетілген таңбасыз бүтін сан немесе идентификатор.

Операторды белгілеу үшін оның алдына белгі жазылып, қос нүкте қойылады.

< белгі**> : [<**белгі**>: …]** <оператор> **;**

Программа жазған кезде **GOTO** операторын қолданбауға тырысу керек. Себебі, бұл оператор программаны түзетуді және тестіден өткізуді қиындатады. Кез келген алгоритмді іске асыру үшін, тілдің бұдан өзге операторлары да жеткілікті.

Бос оператор программада таңбаланбайды және ол ешқандай іс-әрекет атқармайды. Бос оператор программада қосымша нүктелі үтір түрінде жазылады.

Егер транслятор бірнеше операторлар тобын бір оператор ретінде қарастыруы керек болса, онда бұндай операторларды **BEGIN** және **END** операторлық жақшаларының ішіне алу керек. Осындай операторлар тобы **құрама оператор** деп аталады. Құрама оператор программаның кез келген жерінде қолданылуы мүмкін.

Программалау теориясы тұрғысынан алғанда, құрылымсыз болып табылатын басқаруды басқа жерге беретін оператор мен процедураларды пайдалану “артық”, себебі кез келген алгоритмді бұларсыз-ақ құрылымды түрге келтіріп орындауға болады. Алайда, жылдам құрастырылған алгоритмдер көбінесе құрылымсыз болып шығады да, оларды жүзеге асыру кезінде де құрылымсыз

болып келетін басқаруды беру нұсқаларын қолдануға тура келеді. Енді құрылымсыз алгоритмдерді жүзеге асырудың артықшылығы мен кемшілігіне тоқтала кетейік.

Құрылымсыз басқаруды беру түрін ұйымдастыру үшін *goto* шартсыз көшу операторы және арнайы процедуралар қолданылады.

Шартсыз көшу операторы. Бұл оператор басқаруды арнайы белгімен анықталған нүктеге береді (5.1-сурет).

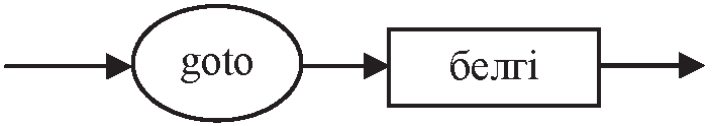
Программадағы барлық белгілер программаның *label* – белгілерді сипаттау бөлімінде (5.2-сурет) жариялануы керек. Белгі программаның кез келген орындалатын операторының алдына қойылады және бір оператордың алдына бірнеше белгі қоюға болады.

Құрылымсыз басқаруды беруді келесі процедуралар көмегімен де орындауға болады:

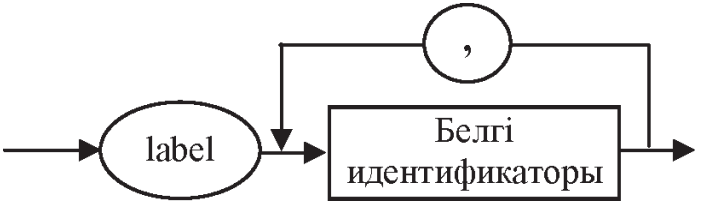
• ***Break*** – кез келген типтегі цикл түрінен шығуды іске асырады;

• ***Continue*** – цикл тұлғасының қалған операторларын орындамай-ақ, оның келесі сатысына көшуді атқарады;

• ***Halt*** (<аяқтау коды>) – программадан шығып, операциялық жүйеге берілген аяқталу кодын қайтару ісін орындайды. Егер аяқталу коды нөлге тең болса, программа ойдағыдай аяқталды деп саналады. Ал аяқталу коды нөлден өзгеше болса, програм-



5.1-сурет. Шартсыз көшу операторының синтаксистік диаграммасы

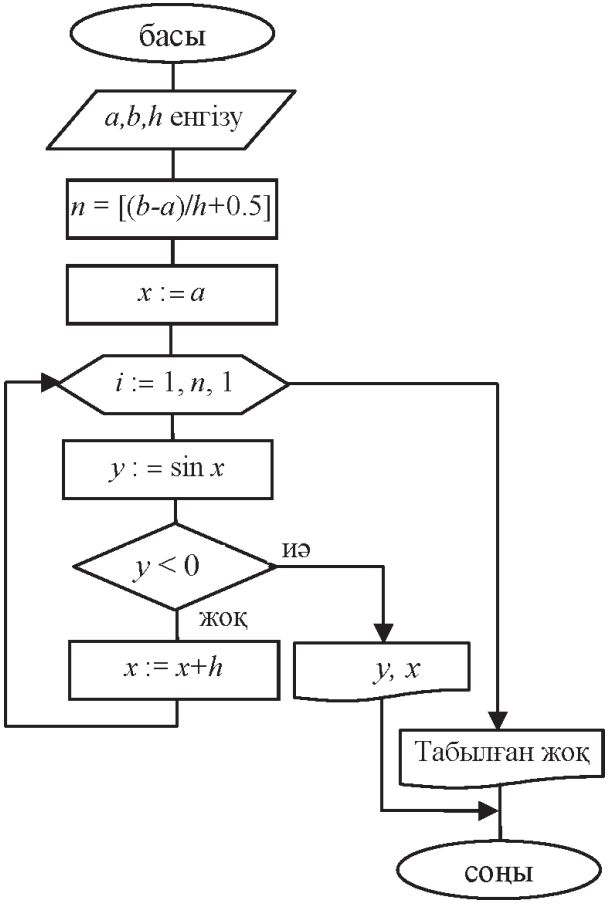


5.2-сурет. Белгіні сипаттау синтаксистік диаграммасы

ма кез келген бір қатенің туындауына байланысты аяқталды деп есептеледі. Аяқталу кодын программалаушының өзі тағайындайды және сол код жайлы мәлімет программа құжаттамасына жазылады;

• ***Exit*** – программадан шығуды іске асырады. Егер ол негізгі программада қолданылса, *Halt* процедурасы тәрізді орындалады.

Құрылымды алгоритмдер нұсқасын құрастыру мәселесі көбінесе іздеу циклдерімен жұмыс жасаған кезде туындайды. Алдыңғы тарауларда (1 тарауда) айтылғандай мұндай циклдерде



5.3-сурет. Тізбектегі алғашқы теріс элементті анықтау алгоритмінің схемасы

берілген сипаттамаға сай элемент табылғанша элементтер тізбегі қайталанып тексеріліп отырады. Іздеу циклінің ерекшелігі, ізделіп отырған элемент тізбек ішінде болмауы да мүмкін. Сондықтан циклден шығудың екі нұсқасын қарастыру керек: ізделініп отырған элемент табылған жағдайда цикл соңына жетпей, одан ерте шығу және тізбектегі барлық элементтерді қарастырып барып, циклді аяқтап шығу.

5.1-мысал. $\sin(x_n)$ функциясы мәндерінің тізбегі ішінен алғашқы теріс элементті анықтайтын программа құру керек. Мұнда n беріліп, x $[a,b]$ аралығында h қадаммен өзгереді.

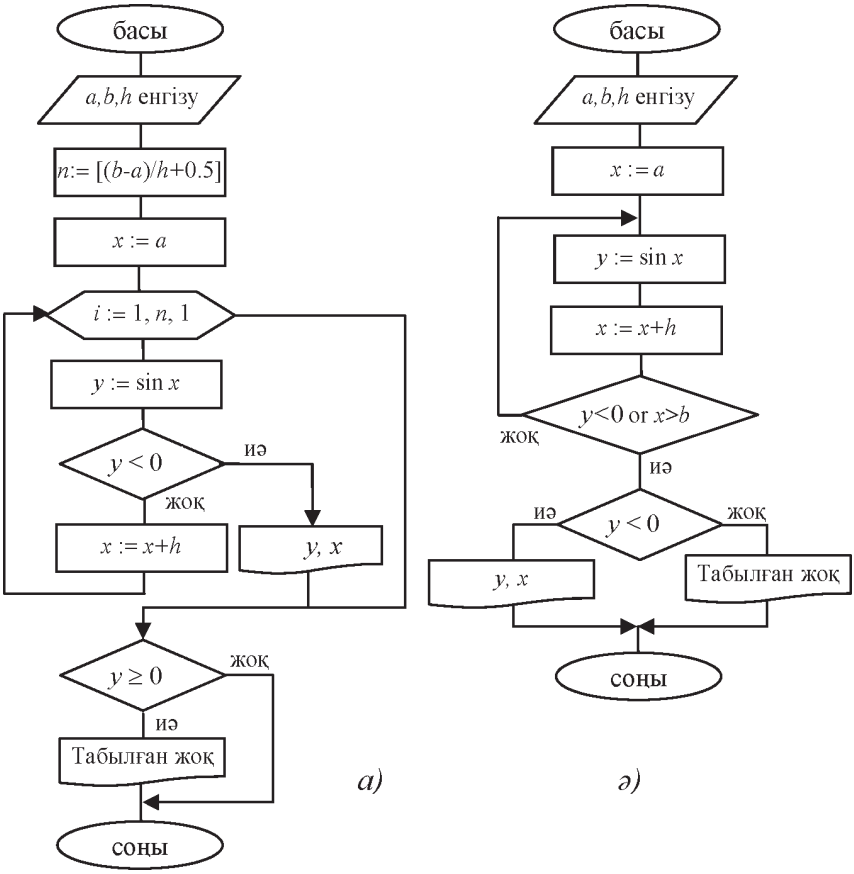
Айтылған элементті табу үшін функция мәнін есептеп оның таңбасын тексеріп отыру керек. Осы аралықта берілген қадам бойынша тізбекте неше элемент болатындығын есептеуге болады. Сондықтан функция мәнін анықтауды санауыш циклі арқылы жүргіземіз. Бастапқы берілген нақты мәндердің кейбір нұсқасы үшін тізбекте мұндай элемент болмауы да мүмкін. Демек, цикл аяқталғаннан кейін экранға осыған сәйкес мәлімет те шығарылуы тиіс.

Есепті құрылымсыз алгоритм бойынша шешу нұсқасы 5.3 суретте көрсетілген.

Бұл нұсқаны түрлендірусіз тек қана *goto* операторының көмегімен іске асыруға болады. Себебі бұл оператор басқаруды программаның кез келген жеріне бере алады, ал *break* процедурасы тек қана цикл соңынан кейін орналасқан операторға береді. Сондықтан *break* процедурасы арқылы құрылған нұсқа ізделіп отырған элементтің табылмағандығы жайлы экранға мәлімет шығару үшін оны қосымша тексеруді талап етеді (5.4, а сурет).

Құрылымды алгоритм нұсқасын құрастыру үшін цикл түрін өзгертеміз: санауыш цикл орнына программадан шығудың екі шартын біріктіретін, күрделі шарты бар "цикл-дейін" циклін қолданамыз. Программа шарттың қай түрі бойынша циклден шыққандығы белгісіз болғандықтан, бұл алгоритмде де ізделініп отырған элементтің табылған, не табылмағандығын қосымша тексеру керек (5.4, *ә* сурет). Алгоритмдердің сәйкес программаларын қарастырайық.

Goto операторының көмегімен орындалған нұсқа:



5.4-сурет. Іздеу циклін ұйымдастыру нұсқалары:
а – break процедурасы көмегімен; ә – құрылымды нұсқа.

```
   Program ex;
Var i,n:integer; x,y,a,b,h:real;
Label konec; {белгіні сипаттаймыз}
Begin
  Write(' a,b,h енгізіңіз:');
  ReadLn(a,b,h);
  n:=round((b-a)/h+0.5); {элементтер санын
бүтін
                              санмен анықтаймыз}
  x:=a;
```

```
  for i:=1 to n do
    begin
     y:=sin(x);
     if y<0 then
     begin
      WriteLn('y=',y:8:6,' x=',x:6:3);
      goto konec; {экранға іздеген элемент
     табылғандығы жайлы мәлімет шығарғаннан
     кейін басқаруды программа соңына береміз}
     end;
     x:=x+h;
    end;
  WriteLn('Элемент табылған жоқ.');
  konec: {басқару берілетін орын}
End.
```

Break процедурасы көмегімен орындалған нұсқа:

```
  Program ex;
  var i,n:integer; x,y,a,b,h:real;
Begin
  Write('a,b,h енгіз:');
  ReadLn(a,b,h);
  n:=round((b-a)/h+0.5); {элементтер санын бүтін
                          санмен анықтаймыз}
  x:=a;
  for i:=1 to n do
    begin
      y:=sin(x);
      if y<0 then
        begin
          WriteLn('y=',y:8:6,' x=',x:8:6);
          break; {циклден ерте шығуды орындай-
мыз }
        end;
      x:=x+h;
    end;
    {break процедурасы орындалғанда басқару
```

```
        берілетін орын}
  if y>=0 then WriteLn('Элемент табылған жоқ.');
End.
```

Құрылымды нұсқа:

```
   Program ex;
var  x,y,a,b,h:real;
Begin
  Write(' a,b,h енгіз:');
  ReadLn(a,b,h);
     x:=a;
  repeat
    y:=sin(x);
       x:=x+h;
  until (x>b) or (y<0); {циклден шығудың
                         біріктірілген шарты}
  if y<0 then WriteLn('y=',y:8:6,' x=',x:6:3)
         else WriteLn('Элемент табылған жоқ.');
End.
```

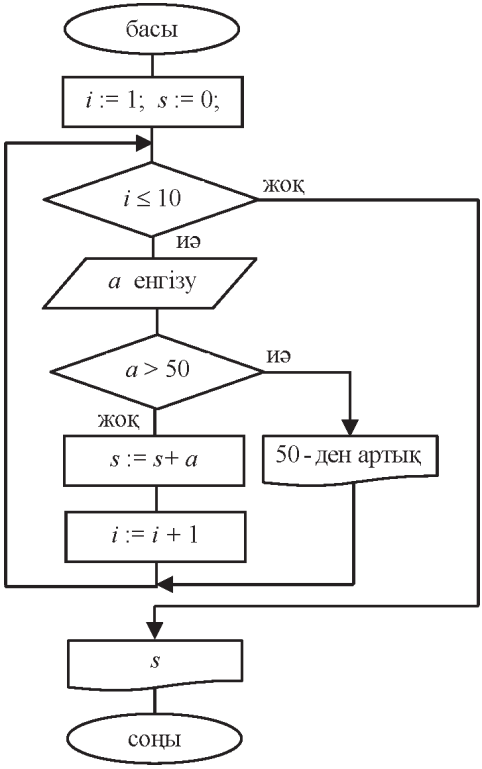
Циклдің құрылымды нұсқасын жасай отырып, біз циклден шығу шартын және ізделіп отырған элементтің табылмағандығы жайлы шешім шығаратын нақты сипаттаманы анықтадық. Ал, *goto* немесе *break* процедурасы арқылы құрастырылған алгоритм, құрылымсыз басқаруды беру кезіндегі кездесетін барлық жадайды ескеруді талап етеді. Сонымен бірге құрылымды түрде жазылған программа ең қарапайым және орындалу барысында қателер саны да аз кездесетін нұсқа болып саналады.

Continue операторын қолданатын программа мысалын қарастырайық.

5.2-мысал. 50-ден аспайтын 10 санның қосындысын анықтайтын алгоритм программасын жазу керек. Теріс сан енгізілген жағдайда программа оны қабылдамай, экранға хабарлама шығарып, оң мән енгізгенді күтуі тиіс.

Бұл мысалда циклдің қайталану саны белгісіз болғандықтан, санауышты циклді қолдану мүмкін емес, сондықтан итерациялық циклді, мысалы, “цикл-әзірше” түрін қолданамыз. Осы есепті шғару алгоритмі 5.5-суретте келтірілген. Бұл алгоритмді *goto* опе-

раторы, *continue* процедурасы немесе тармақталу операторының көмегімен іске асыруға болады. Соңғы тармақталу операторын пайдалану тәсілі құрылымды нұсқа болып табылады.



5.5-сурет. 50-ден аспайтын 10 санның қосындысын анықтайтын алгоритмнің схемасы

Айтылған нұсқаларды қарастырайық.

Goto операторының көмегімен шығарылған нұсқа:

```
   Program ex;
Var i,s,a:integer;
Label cycl;
Begin
 i:=1; s:=0;
 while i<=10 do
  begin
```

```
   Read(a);
   if a>50 then
    begin
     WriteLn('Енгізілген сан 50-ден аспауы ке-
рек');
     goto cycl; {басқаруды цикл соңына беру}
    end;
   s:=s+a; {айналып өтетін операторлар тобы}
   i:=i+1;
  cycl: end;
 WriteLn(s);
End.
```

Continue процедурасының көмегімен шығарылған нұсқа:

```
   Program ex;
Uses crt;
Var i,s,a:integer;
Begin
  i:=1; s:=0;
  while i<=10 do
   begin
     Read(a);
     if a>50 then
      begin
       WriteLn('Енгізілген сан 50-ден аспауы
                керек');
      continue;{басқаруды келесі қадамға
               береміз}
      end;
     s:=s+a;
     i:=i+1;
   end;
  WriteLn(s);
End.
```

Құрылымды нұсқа:

```
   Program ex;
```

```
Uses crt;
Var i,s,a:integer;
Begin
 i:=1;
 while i<=10 do
  begin
    Read(a);
    if a>50 then WriteLn('Енгізілген сан 50-ден
                                  аспауы керек')
            else
             begin
              s:=s+a;
              i:=i+1;
             end
  end;
 WriteLn(s);
End.
```

Бұл мысалда құрылымды алгоритм ең қарапайым түр болып саналады. Мұнда *goto* операторы мен *continue* процедурасын қолдану тиімді емес. *Continue* процедурасын қолдану тек айналып өтетін операторлар саны көп болған жағдайда ғана тиімді.

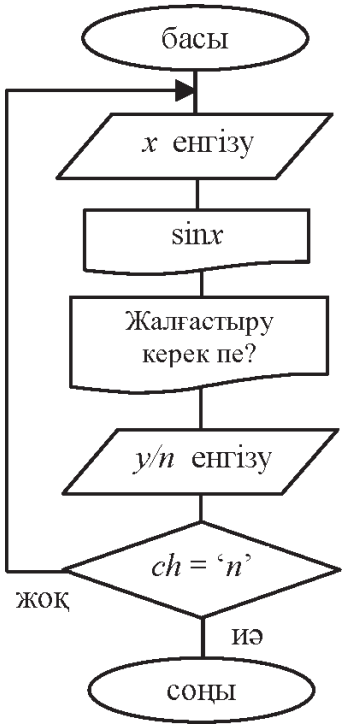
Практикалық есептерді шығаруда тәжірибесі аз программалаушылар *goto* операторын жиі қолданатын алгоритм түрін төмендегі мысал арқылы көрсетейік.

5.3-мысал. Тұтынушы енгізген аргумент мәні бойынша функция мәнін бірнеше рет есептейтін программа құру қажет. Нәтиже алынғаннан кейін программа "Жалғастыру керек пе?"-деп сұрайтын болсын. Егер жұмысты жалғастыру керек болса, тұтынушы пернетақтадан "Y", кері жағдайда "N" әрпін енгізуі керек. Сәйкесінше бірінші жағдайда программа жұмысын жалғастырады, ал екінші жағдайда – тоқтатады.

Программа негізінде циклдік процесс жатыр. Көп жағдайда мұны "цикл-дейін" (5.6-сурет) операторы және *goto* операторы көмегімен жазуға тырысады. Бұл қатені қайталамас үшін, мына жағдайды есте сақтау керек: егер алгоритм схемасында орындалып кеткен программа фрагментіне басқаруды қайта беру керек болса, онда жеке цикл қарастырып, бұл циклден шығу шартын анықтау керек.

Төменде осы алгоритмді “цикл-дейін” көмегімен шығару программасы келтірілген.

```
   Program ex;
Var x:real; ch:char;
Begin
repeat
 Write(‘ x-ті енгізіңіз:’);
 ReadLn(x); {еш уақытта Read қолданылмайды,
себебі
                  ары қарай символ енгізіледі}
 WriteLn(‘Нәтиже ‘,sin(x):8:4);
 WriteLn(‘Жалғастыру керек пе? (y/n)’);
 ReadLn(ch);
until ch=’n’;
End.
```



5.6-сурет. Функция мәнін есептеу алгоритмінің схемасы

Жоғарыда айтылғандардың барлығынан мынадай қорытынды шығаруға болады.

1. Goto шартсыз көшу операторын жалпы Pascal программаларында мүлдем қолданбаған жөн. Егер *goto* операторын қолдану барысында алгоритм құрылымсыз болып шықса, онда оны құрылымды алгоритмге түрлендіруге тырысу қажет. Бұндай шектеу *goto* операторын қолданған жерде көп қате кездесетін болғандықтан, қойылып отыр.

2. Break және ***continue*** құрылымсыз басқаруды беру процедураларын қолдануға болады, бірақ бұларда басқарылудың қай жерге берілетіндігін нақты анықтап алу керек. Циклдің осы процедуралар қолданылған жерінде қате жиі кездеседі, сондықтан программаны арнайы тестіден өткізіп отыру қажет.

Halt және *Exit* процедуралары программаны немесе ішкі программаны арнайы тоқтату үшін қолданылады. Сондықтан бұл процедуралар программа құрылымына онша әсерін тигізбейді.

Бақылау сұрақтары

1. *Шартсыз көшу операторын қолданудың келеңсіз жағдайларын атаңыз.*
2. *Белгі дегеніміз не? Ол қалай сипатталады және қолданылады?*
3. *Бос оператор дегеніміз не?*

Тапсырмалар

1. *Таңдау бойынша негізгі төрт арифметикалық амалдың бірін орындайтын «калькулятор» программасын жазыңыз. Программа жұмыстың аяқталған-аяқталмағандығы жайлы сұрайтын болсын.*
2. *Пернетақтадан енгізілген үш санның ең үлкенін таңдап алатын программа құрыңыз. Программа жұмыстың аяқталған-аяқталмағандығы жайлы сұрайтын болсын.*
3. *Пернетақтадан енгізілген санның квадрат түбірін есептейтін программа құрыңыз. Программа жұмыстың аяқталған аяқталмағандығы жайлы сұрайтын болсын.*
4. *Пернетақтадан енгізілген апта күнінің нөміріне байланысты сәйкес күннің атын экранға шығаратын программа құрыңыз. Программа жұмыстың аяқталған-аяқталмағандығы жайлы сұрайтын болсын.*
5. *Пернетақтадан енгізілген нақты санды 2-ден 9-ға дейін дәрежелейтін программа құрыңыз. Программа жұмыстың аяқталған-аяқталмағандығы жайлы сұрайтын болсын.*

6. *a-дан b-ға дейінгі бүтін сандардың квадратын экранға шығаратын программа құрыңыз.*

7. $y = x^2 - 16x + 32$ *функциясының мәнін 1-ден 10-ға дейінгі аралықта 0,25 қадаммен есептейтін программа құрыңыз.*

8. $y = \begin{cases} x^3 & \text{егер } x \succ 0 \\ x^2 & \text{егер } -2 \geq x \geq 0 \\ x & \text{басқа жағдайда} \end{cases}$ *функциясының мәнін –5-тен 5-ке дейінгі аралықта 0,5 қадаммен есептейтін программа құрыңыз.*

9. $f(x) = \dfrac{4x^2 + 16(x+2)}{2x}$ *функциясының x a-ден b-ға дейін d қадаммен өзгергендегі мәнін есептейтін программа құрыңыз.*

10. *Апталардың реттік нөмірі бойынша жыл мезгілін анықтайтын программа құрыңыз. Программа жұмыстың аяқталған-аяқталмағандығы жайлы сұрайтын болсын.*

5.2 Шартты оператор

Шартты оператор программадағы шарт логикалық өрнек ретінде берілгенде тармақталу әрекетін атқару үшін қолданылады. Оның жазылуы:

IF <логикалық өрнек> **THEN** <1-ші оператор>
[ELSE <**2**-ші оператор>**];**
<келесі оператор>**;**

Операторлардың орындалу ережесі: егер логикалық өрнек нәтижесі **АҚИҚАТ (TRUE)** болса, онда <1-ші оператор> орындалады, одан кейін <келесі оператор> орындалады; егер – **ЖАЛҒАН (FALSE)** болса, онда <2-ші оператор> орындалады да, содан кейін барып <келесі оператор> орындалады. 1-ші және 2-ші операторлар жәй немесе құрама оператор болуы мүмкін. Егер операторлардың **ELSE-**ден басталатын бөлігі жоқ болса, онда логикалық өрнек нәтижесі **ЖАЛҒАН (FALSE)** болған жағдайда, бірден <келесі оператор> орындалады. Шартты операторлар қабаттасып келіп, **ELSE** сөзі бірнеше рет кездесетін болса, оның әрқайсысы өзінің алдындағы **IF** операторында жазылған шартқа байланысты атқарылады. Шартты операторлардың қабаттасқан түрінің сатылары санын өте көбейтпеген дұрыс, өйткені мұндайда программа көрнектілігі жоғалып, қате кетуі жиіленеді.

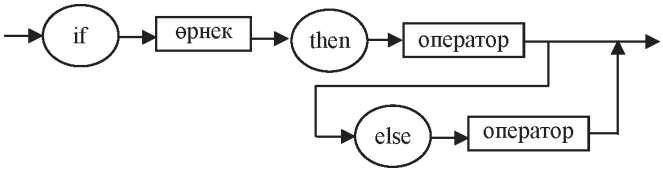
Мысалы,

```
...IF A>0 THEN P:=P+1
          ELSE
            IF A<0 THEN  L:=L+1
                   ELSE  N:=N+1; ...
...IF A>0 THEN
           BEGIN
             S:=S+A; K:=K+1
           END;...
```

Шартты көшу операторы (5.7-сурет) тармақталуды, яғни белгілі бір шарт бойынша түрлі іс-әрекеттер орындауды программалау үшін қолданылады. Шарт логикалық өрнек түрінде жазылады да, өрнек нәтижесіне байланысты: нәтиже ақиқат (true) болса **then** түйінді сөзінен кейінгі оператор, кері жағдайда **else** түйінді сөзінен кейінгі оператор орындалады.

Әр тармақта бір оператор (сонымен бірге басқа if операторын) немесе құрама оператор жазуға болады.

Құрама оператор дегеніміз – **begin...end** операторлық жақшалары ішіне алынған операторлар тізбегі. Тізбектегі операторлар бір-бірінен нүктелі үтір «;» арқылы ажыратылады. *Else* сөзінің алдына нүктелі үтір ***еш уақытта*** қойылмайды, себебі мұндайда шартты операторды жазылуы ары қарай жалғастырылады.



5.7-сурет. Шартты көшу операторының синтаксистік диаграммасы

Бұл операторды else тармағын қолданбауға да болады. Кейде шартты операторды қысқа түрлерін қабаттастыра қолдану есептің айқындылығын көрсете алмайды. Мысалы, мына программа фрагменті алгоритм схемасының (5.8-сурет) екі нұсқасының қайсына сәйкес екендігі түсініксіз:

```
if <1 шарт> then
   if <2 шарт>  then
```

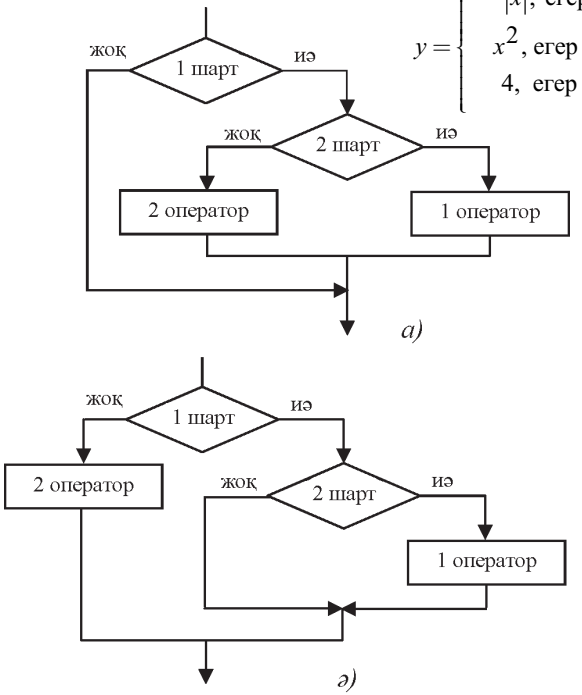
```
                    <1 оператор>
               else
    <2 оператор>;
```

Мұндайда “кірістіру ережесі” қолданылады: *else* тармағы әр уақытта соңғы *if* түйінді сөзінде жазылған шартқа тиісті болады. Бұл 5.8, *а*-суретіндегі алгоритмге сәйкес келеді. Егер 5.8, *ә*-суреттегі алгоритмді іске асыру керек болса, онда операторлық жақшаларды қолдану керек:

```
 if <1 шарт> then
begin
 if <2 шарт> then
             <1 оператор>
end
else <2 оператор >;
```

5.4-мысал. Төменде көрсетілген функция мәнін есептейтін программа жазу қажет:

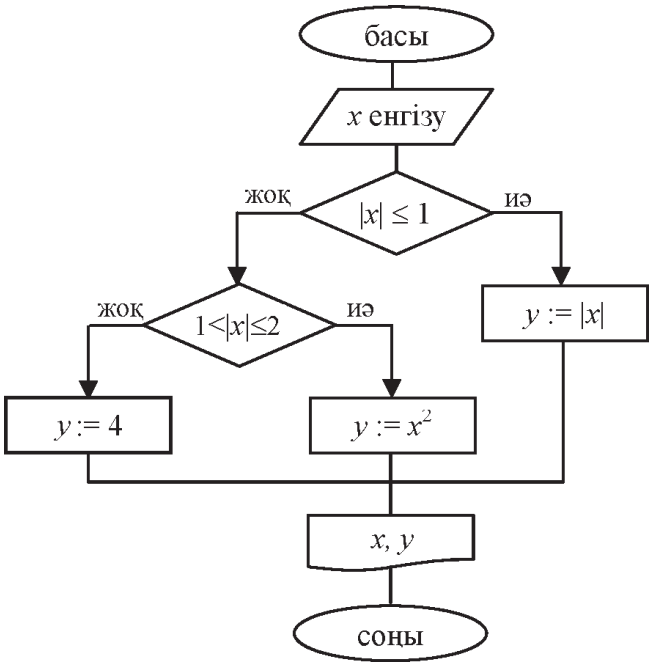
$$y = \begin{cases} |x|, \text{ егер } |x| \le 1; \\ x^2, \text{ егер } 1 < |x| \le 2; \\ 4, \text{ егер } |x| > 2. \end{cases}$$



5.8-сурет. Алгоритмдер фрагменттері

Бұл программаны аргумент мәнін енгізуден бастау керек. Енгізілген мәннің қандай аралықта жатқанына байланысты функция мәнін берілген өрнектің бірі бойынша есептейміз. Бұл есепті шешу алгоритмі 5.9-суретте көрсетілген. Программа мәтіні төменде берілген.

```
   Program ex;
Var x,y:real;
Begin
 WriteLn('Аргумент мәнін енгіз:');
 ReadLn(x);
 if abs(x)<=1 then y:=abs(x) {бірінші аралық}
    else
     if (abs(x)>1) and (abs(x)<=2)
        then y:=sqr(x) {Екінші аралық}
        esle y:=4; {Үшінші аралық}
 WriteLn(' x=',x:8:5,' болғанда y=',y:8:5)
  End.
```



5.9-сурет. Берілген аралықта функция мәнін есептеу алгоритмінің схемасы

Бақылау сұрақтары

1. *Шартты оператор не үшін керек?*
2. *Шартты операторлардың жазылуының екі формасының айырмашылығы неде?*
3. *Шартты операторлардың ішінде басқа шартты операторды жазуға бола ма?*
4. *Құрама оператор қайда және қалай қолданылады?*
5. *Шартты операторлардың блок-схемасының орындалуын түсіндіріңіз.*

Тапсырмалар

1. *M(x,y) нүтесі орналасқан квадранттың нөмірін анықтап, экранға шығарыңыз.*
2. *X, Y нақты сандары берілген. Сандардың кішісін олардың жарты қосындысымен, ал үлкенін олардың екі еселенген көбейтіндісімен алмастырыңыз. Есептеулер нәтижесін экранға шығарыңыз.*
3. *Үш нақты сан берілген. Осы сандардың ішіндегі оң сандарды квадраттап, экранға шығарыңыз.*
4. $$y=\begin{cases} x*x & \text{егер} \quad 0 \prec x \prec 2 \\ x+4 & \text{егер} \quad -2 \prec x \le 0 \\ 0 & \text{басқа жағдайларда} \end{cases}$$ *функциясының мәнін есептейтін программа жазыңыз.*
5. *Үш санның ең үлкенін анықтайтын программа құрыңыз.*
6. *x, y сандары берілген. Егер x және y теріс сан болса, онда оларды модульдерімен алмастырыңыз; егер тек біреуі ғана теріс сан болса, олардың екеуін де 0.5-ке өсіріңіз; егер екеуі де оң сан болса, онда оларды 10 есе өсіріңіз.*
7. *M(x,y) нүктесі радиусы r, центрі a,b нүктесінде орналасқан дөңгелектің ішінде жататындығын анықтаңыз.*
8. *M(x,y) нүктесі центрі координаталар жазықтығының басында орналасқан, сыртқы радиусы R1, ішкі радиусы R2 шеңберге жататындығын анықтайтын программа құрыңыз.*
9. *a, b, c үш саны берілген. a<b<c теңсіздігін тексеріп, нәтижені «Дұрыс» немесе «Дұрыс емес» деген түрде экранға шығарыңыз.*
10. *Ең жақын бүтін санға дейін дөңгелектелген a және b сандарының бөліндісі жұп сан екенін анықтаңыз.*

5.3 Таңдау операторы

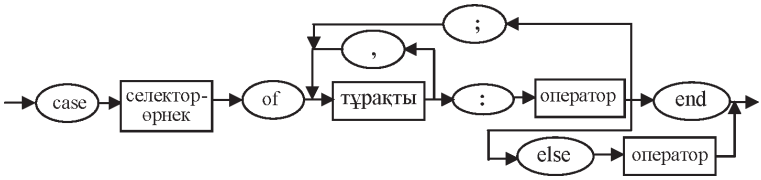
Жоғарыда біз тармақталу операторында белгілі бір шарттың орындалу немесе орындалмауына байланысты алгоритмнің 1-ші немесе 2-ші операторы орындалатынын қарастырдық. Алайда, көбінесе мүмкін болатын жағдай екеу емес, одан да көп бола береді, яғни процесс көп тармақты болады.

CASE таңдау операторы алгоритм құрамындағы екіден көп тармақтың бірін таңдау керек болған кезде шартты оператордың орнына қолданылады.

Тармақталу берілген логикалық мәнге ие болатын өрнек түріндегі шартты тексеруден басталады, өрнек (селектор) мәндері таңдап алынған типтердің бірі болатын бүтін, символдық, логикалық типтердің бірінде жазылады. Бұл өрнек оператор орындалар кезде белгілі бір мәнге ие болуы тиіс. Сол мән оған байланысты орындалатын оператордың белгісі рөлін атқарады. Егер өрнектің есептелген мәні көрсетілген мәннің біріне сәйкес келсе, сол қатардағы оператор орындалады.

CASE таңдау операторы көп тармақты алгоритмдерді барынша қарапайым және көрнекі түрде жазуға мүмкіндік береді. Оның құрамында селектор (*selector* – таңдау) деп аталатын өрнек және параметрлер тізімі болуы тиіс, әрбір параметрге сәйкес таңдау тұрақтылары болуы қажет. CASE операторының синтаксистік диаграммасы (5.10-сурет) былай кескінделеді:

Ал, оның жалпы жазылу түрі:



5.10-сурет. Таңдау операторының синтаксистік диаграммасы

CASE < селекторлық өрнек > **OF**

<белгі> [,<белгі> ,<белгі>..<белгі>] :<1 оператор>;

<белгі> [,<белгі> ,<белгі>..<белгі>] :<2 оператор>;

<белгі> [,<белгі> ,<белгі>..<белгі>] :<n-ші оператор>;

[**ELSE** <оператор>]

END

Селекторлық өрнек (селектор, ауыстырғыш) пен белгі түріндегі тұрақты (белгі нұсқалар, таңдау белгісі) бір типті болуы керек. Белгі-тұрақты программадағы белгілер тәрізді белгілер бөлімінде сипаттауды талап етпейді. Белгі-тұрақтыларға **GOTO** операторында сілтеме жасауға болмайды. Белгілер саналатын немесе интервалды түрде берілуі тиіс.

Диаграммаға сәйкес таңдау операторы төмендегідей ықшамды түрде де жазылады:

```
case   <селектор-өрнек >  of
   <1-белгі>:  <1-оператор>
   <2-белгі >:  <2-оператор>
   . . .
   <n - белгі >:  <n-оператор>
   else  <n+1-оператор>
end;
```

Таңдау операторы төмендегідей ереже бойынша орындалады: алдымен селекторлық өрнек есептеледі; сонан кейін белгі нұсқасы селектордың ағымдағы мәніне сәйкес оператор орындалады; бұдан кейін **CASE** операторынан кейін орналасқан операторға көшу атқарылады. Егер селектор мәні белгі нұсқаларының ешқайсысымен сәйкес келмесе, онда **ELSE** түйінді сөзінен кейін орналасқан оператор орындалады. ELSE тармағы жоқ болған жағдайда, **CASE** операторынан кейін орналасқан оператор атқарылады.

Мысалы, төмендегі функцияны есептеу қажет болсын делік:

$$Z=\begin{cases}\cos x, & егер\ k=3\\ \sin x, & егер\ k=2\\ e^{x}, & егер\ k=1\\ \ln x, & егер\ k=0\\ 0, \text{ басқа жағдайда}\end{cases}$$

мұндағы k мәніне сәйкес белгілер жазып, соларға төмендегідей түрде көшу әрекеттерін орындаймыз:

```
… CASE K OF
0:  Z:= LN(X);
1:  Z:= EXP(X);
```

```
2:  Z:= SIN(X);
3:  Z:= COS(X)
  ELSE
    Z:= 0
END; …
```

Бұл мысалда *k* параметрінің мәніне байланысты стандартты функциялардың бірі есептеледі. *k* параметрінің мәні таңдау операторларына дейін белгілі болуы тиіс.

Келесі мысалда **JAUAP** айнымалысы **V** символдық айнымалысының енгізілген мәніне байланысты YES немесе NO мәндерінің біріне ие болады. Бұл жерде белгілер үтір арқылы бөлініп, тізбек түрінде берілген.

```
… VAR V: CHAR;
  JAUAP: STRING;
        …
  CASE  V  OF
   'D', 'd', 'Д', 'д': JAUAP:='YES';
    'N', 'n', 'H', 'н': JAUAP:='NO'
  ELSE
     JAUAP:=' '
  END; …
```

Келесі мысалда белгі нұсқалары интервалмен берілген.

```
… VAR V:CHAR;
    JAUAP:STRING;
      …
    CASE  V  OF
      'A'..'Z','a'..'z':JAUAP:='әріп';
         '0'..'9':JAUAP:='цифр'
    ELSE
        JAUAP:='арнайы символ'
    END; …
```

Таңдау операторы бірнеше нұсқалар ішінен қандай да бір параметрдің мәніне тең нұсқаны іске асыру үшін қолданылады.

5.5-мысал. Көрсетілген нүктеде берілген функция мәнін есептейтін программа жазу керек.

Тұтынушыға функцияны қарапайым менюден таңдауға мүмкіндік берейік. Әр функцияға қандай да бір сан (код) сәйкес келеді:

Функция кодын енгізіңіз:

1 – y=sin(x)

2 – y=cos(x)

3 – y=exp(x)

Енгізілген кодқа сәйкес функция таңдалып алынады. 5.11-суретте программа алгоритмінің схемасы берілген.

Төменде құрастырылған алгоритмді іске асыратын программа мәтіні берілген.

```
   Program ex;
Var x,y:real; Kod:byte; key:boolean;
Begin
 WriteLn(' Функция кодын енгіз:');
 WriteLn('1 - sin(x)');
 WriteLn('2 - cos(x)');
 WriteLn('3 - exp(x)');
 ReadLn(Kod);
    Write('Аргумент мәнін енгіз');
 ReadLn(x);
 Key:=true; {код дұрыстығының белгісі}
 case Kod of
  1: y:=sin(x);
  2: y:=cos(x);
  3: y:=exp(x);
  else Key:=false; {код функцияға сәйкес емес}
 end;
 if Key then
        WriteLn(' x=',x:12:6,'y=',y:12:6)
       else
        WriteLn('Функция коды дұрыс
                 енгізілмеген.');
End.
```

5.6-мысал. Апта күндерінің реттік нөмірі бойынша олардың аттарын анықтайтын программа құру керек.

```
PROGRAM DAY;
   VAR kyn: integer;          {Бұл программада бірден}
BEGIN                         {жетіге дейінгі кез кел-
 write('күннің реттік нөмірі:'); ген}
     readln(kyn);             {сан енгізіліп, егер ол}
     case kyn of
 1: writeln ('дүйсенбі');     {1-ге тең
                              болса,»дүйсенбі»}
 2: writeln ('сейсенбі');     {2-ге тең
                              болса,»сейсенбі»}
 3: writeln ('сәрсенбі');     {3-ке тең
                              болса,»сәрсенбі»}
 4: writeln ('бейсенбі');     {4-ке тең
                              болса,»бейсенбі»}
 5: writeln ('жұма');         {5-ке тең болса,»жұма»}
 6: writeln ('сенбі');        {6-ға тең болса,»сенбі»}
 7: writeln ('жексенбі');     {7-ге тең
    end                       болса,»жексенбі»}
 else writeln('1..7аралы-     {деген сөз жазылып
ғындағы сан енгізу керек')    шығады.}
END.
```

5.7-мысал. Кез келген жыл мерзімін енгізіп, сол жылдың шығыс календары (қазақша жыл санау) бойынша қай жануардың атына сәйкес келетінін анықтау программасы төмендегідей болады. Алгоритм негізіне 12-ге қалдықсыз бөлінетін жыл мешін жылы болатыны алынған.

```
PROGRAM GYL1;
VAR
       gyl: integer;
BEGIN
       write ( ' Керекті жылды енгіз: ' ) ;
       readln (gyl);
       write (gyl:4, ' жыл   ' );
       case (gyl mod 12) of
0: write (' мешін ' );
1: write (' тауық' ) ;
2: write (' ит ' ) ;
3: write (' доңыз ' ) ;
4: write (' тышқан ' ) ;
5: write (' сиыр ' ) ;
```

```
6: write (' барыс ' ) ;
7: write (' қоян ' ) ;
8: write (' ұлу ' ) ;
9: write (' жылан ' ) ;
10: write (' жылқы ' ) ;
11: write (' қой ' )
end;
writeln(‘ жылы болды ‘ )
END.
```

5.8-мысал. Кез келген айдың бірінші жұлдызы аптаның қай күні екені белгілі болғанда, сол айдың енгізілген кез келген күнінің аптаның қандай күні болатынын анықтау программасын құру қажет.

```
     PROGRAM case1;
      Label  10;
      const k=1; {айдың бірі дүйсенбі болған}
                 {егер айдың бірі сейсенбі
                  болса,k==2, егер айдың бірі
                 сәрсенбі болса k=3   т.с.с. }
    VAR den, n: integer; t:  char;
BEGIN
10: write('айдың күнін енгіз:');readln (den);
  n:= den mod 7+k-1;  if n>7 then n:=n-7;
     write ('айдың', den: 3,'күні');
     CASE  n of
        1: write ('дүйсенбі');
        2: write ('сейсенбі');
        3: write ('сәрсенбі');
        4: write ('бейсенбі');
        5: write ('жұма');
        6: write ('сенбі');
        7: write ('жексенбі')
     end;
writeln('болады');
writeln('тағы енгізесің бе? (иә="y", жоқ="n")');
readln (t);
if (t='Y') or (t='y')  then goto 10
   END.
```

Бақылау сұрақтары

1. *Таңдау операторы не үшін керек?*
2. *Таңау операторының бірнеше тармағы бңр мезетте орындалуы мүмкін бе?*
3. *Айнымалы мәнінен кейін неше оператор жазуға болады?*
4. *Айнымалы мәні ретінде бірнеше тұрақтыны көрсетуге бола ма?*

Тапсырмалар

1. *Апта күнінің нөмірін және оған сәйкес күннің атын орыс, ағылшын тілдерінде экранға шығаратын программа құрыңыз.*
2. *Ай нөмірін және оған сәйкес айдың атын орыс, ағылшын тілдерінде экранға шығаратын программа құрыңыз.*
3. *Айдың нөмірін енгізіңіз. Осы айға сәйкес жыл мезгілінің атын экранға шығарыңыз.*
4. *Уақытты енгізіңіз (тек сағатты). Енгізілген уақытқа сәйкес экранға: «Қайырлы таң», «Қайырлы күн», «Қайырлы кеш», «Қайырлы түн» деген мәліметтердің бірін шығаратын программа құрыңыз.*
5. *Отыратын орын санын енгізіп, осыған сәйкес транспорт атын экранға шығарыңыз: «велосипед», «мотоцикл», «жеңіл автомашина», «микроавтобус», «автобус».*
6. *Енгізілген санға сәйкес келесі операциялар атын экранға шығаратын программа құрыңыз: 1 - қосынды; 2 - айырма; 3 – көбейтінді; 4 – бөлінді;*
7. *Енгізілген санға сәйкес келесі функциялар атын экранға шығаратын программа құрыңыз: 1 - квадрат; 2 – квадрат түбір; 3 - синус; 4 – косинус.*
8. *Енгізілген санға сәйкес келесі мәліметтерді экранға шығаратын программа құрыңыз: 1 – фамилия; 2 – аты; 3 – тегі; 4 – туған жылы.*
9. *Пернетақтадан енгізілген санның дәрежесін есептейтін программа құрыңыз. Дәреже 0 .. 9 аралығында.*
10. *Бүтін оң х санының символдарының санын анықтайтын программа құрыңыз.*

5.4 Қайталану саны белгілі цикл операторы

Цикл операторлары құрамына кіретін операторлар тобын бірнеше рет қайталау үшін қолданылады. Турбо Паскаль тілінде арифметикалық прогресия түріндегі (**FOR –** санауышы бар цикл операторы**)** *параметрлік* және итерациялық түрдегі *қадамдық* цикл операторлары (**WHILE** және **REPEAT**) бар.

Арифметикалық прогресия түріндегі цикл операторы циклдің қайталану саны алдын ала белгілі болған жағдайда қолданылады. Мұнда цикл параметрінің өзгеру қадамы +1 немесе –1 болуы мүмкін.

Оператордың жазылуы:

FOR <цикл параметрі>:= <1 өрнек> { **TO** / **DOWNTO** } <2 өрнек> DO <оператор>;

<цикл параметрі> – бұл кез келген типтегі айнымалы (бүтін, символдық, санауыш, интервалды);

TO – цикл параметрінің өзгеру қадамы +1 болған кезде қолданылатын түйінді сөз;

DOWNTO – цикл параметрінің өзгеру қадамы -1 болған кезде қолданылатын түйінді сөз;

<1 өрнек> – цикл параметрінің бастапқы мәні, өрнек типі цикл параметрінің типімен бірдей;

<2 өрнек> – цикл параметрінің соңғы мәні, өрнек типі цикл параметрінің типімен бірдей;

<оператор> – цикл тұлғасы – жәй немесе құрама оператор.

Параметрлі цикл операторының орындалу схемасы төмендегі суретте көрсетілген.

FOR операторы төмендегі ереже бойынша орындалады:

- <1 өрнек> есептеліп, цикл параметріне меншіктеледі;
- цикл соңы шарты тексеріледі: <цикл параметрі> үлкен < 2 өрнек> болса **TO** сөзі қолданылады және <цикл параметрі> кем < 2 өрнек> болса **DOWNTO** сөзі қолданылады;
- цикл тұлғасы орындалады;
- цикл параметрі бірге өседі (**TO**) немесе кемиді (**DOWNTO**);
- циклдің бірінші сатысынан басқасы қайталанады.

Бұл операторды пайдаланғанда, мына жағдайларды есте сақтау керек:

- **FOR** циклінің ішінде цикл параметрінің бастапқы, соңғы және ағымдағы мәндерін өзгертуге болмайды.
- Қадамы +1 болып келетін нұсқада цикл параметрінің бастапқы мәні соңғы мәнінен үлкен болса, цикл бір де бір рет орындалмайды. Сол сияқты, цикл қадамы –1 болғанда да, бастапқы мән соңғы мәннен кем болса, ол бір де бір рет атқарылмайды.

• Циклден шығу **GOTO** операторы немесе **BREAK** процедурасы көмегімен орындалған кездерден басқа жағдайлардың барлығында цикл параметрінің мәні анықталмаған болып саналады.

• Цикл тұлғасы ретінде басқа оператор қолданылуы мүмкін.

Мысалы, **F=N!** факториалының мәнін есептеу үшін төмендегі операторларды қолдануға болады:

```
a) … F:=1;                    b) … F:=1;
FOR I:=1 TO N DO          FOR I:=N DOWNTO 1 DO
F:=F*I; …                 F:=F*I; …
```

Келесі мысалда цикл 26 рет орындалады және SIM айнымалысы мән терінде ’A’-дан ‘Z’-ке дейінгі латын әріптерін қабылдайды.

```
...
FOR  SIM:='A' TO 'Z' DO
       WRITELN(SIM);
...
```

Егер цикл ішінде тағы да басқа цикл болса, онда бұндай цикл *қабатталған цикл* немесе *күрделі цикл* деп аталады. Ішінде циклі бар цикл *сыртқы цикл* деп, ал сыртқы цикл ішіндегі цикл *ішкі цикл* деп аталады. Ішкі және сыртқы циклдерде жоғарыда айтылған үш циклдің кез келгені болуы мүмкін**: FOR**, **WHILE** немесе **REPEAT**. Қабаттасқан цикл құрастырған кезде, ішкі циклдің барлық операторлары сыртқы цикл тұлғасының ішінде болуы керек. Циклдерді бір-біріне қабаттастыру саны компьютердің жады көлемімен шектеледі. Алдымен ішкі цикл орындалады, сонан кейін сыртқы цикл параметрі өзгеріп, тағы да ішкі цикл орындалады, т.с.с.

Мысал. Y мәнін төмендегі формаула бойынша есептеу керек болсын делік:

$$Y = \sum_{i=1}^{N} \prod_{j=1}^{M} A_{ij}$$

Мұның программасы мәтіні төмендегідей болады.

```
PROGRAM SP;
    CONST N=10;
          M=15;
```

```
VAR A: ARRAY [1..N,1..M] OF REAL;
       I,J: INTEGER;
       P,Y: REAL;
BEGIN
       FOR I:=1 TO N DO
       FOR J:=1 TO M DO
           READLN(A[I,J]);
       Y:=0;
       FOR I:=1 TO N DO
         BEGIN
           P:=1;
           FOR J:=1 TO M DO
               P:=P*A[I,J];
           Y:=Y+P
         END;
       WRITELN('Y=',Y)
 END.
```

Бақылау сұрақтары

1. *Параметрлі цикл операторы неше рет орындалады?*
2. *Параметрлі цикл операторының неше нұсқасын білесіз?*
3. *Цикл параметрі дегеніміз не? Цикл параметірінің бтастапқы және соңғы мәні дегеніміз не?*
4. *Параметрлі цикл денесі дегеніміз не?*
5. *Параметрлі циклдың орындалу ретін түсіндіріңіз.*
6. *Параметрлі цикл блок-схемасының жұмысын түсіндіріңіз.*
7. *Break және Continue процедураларының қызметін түсіндіріңіз.*
8. *Параметрлі циклды қолдануға мысал келтіріңіз.*

Тапсырмалар

1. *Пернетақтадан енгізілген n үшін қатарының қосындысын есептеңіз.* $s = 1 + \frac{1}{x} + \frac{1}{2x} + \frac{1}{3x} + \cdots + \frac{1}{nx}$
2. $y = 0.4x^2 - \frac{1}{x}$ *функциясының x [2;10] аралығында 0,5 қадаммен өзгергендегі мәнін есептеңіз.*
3. *Пернетақтадан енгізілген n үшін n! мәнін есептеңіз.*
4. *Пернетақтадан енгізілген n үшін* $y = 1! + 2! + 3! + \cdots + n!$ *функциясының мәнін есептеңіз.*

5. *Пернетақтадан енгізілген n үшін* $\frac{1}{2}+\frac{2}{3}+\frac{3}{4}+\frac{4}{5}+\frac{5}{6}+\frac{6}{7}$ *қатарының алғашқы n мүшесінің қосындысын.*
6. *Пернетақтадан енгізілген a және n үшін* $a\cdot(a+1)\cdot(a+2)\cdot\dots\cdot(a+n)$*-ді есептеңіз.*
7. *Әр түсті және әр түрлі өрнектермен өрнектелген, көлемдері бірдей параллепипедтерден құрастырылған, көлемі 10x10 көлбеу тор салыңыз.*
8. *Әр түсті және әр түрлі өрнектермен өрнектелген, көлемдері бірдей дөңгелектерден құрастырылған, көлемі 10x10 тор салыңыз.*
9. *Әр түсті және әр түрлі өрнектермен өрнектелген, көлемдері бірдей тікбұрыштардан құрастырылған, көлемі 10x10 пирамида салыңыз. Пирамиданың ең төменгі қатары 20 тікбұрыштан тұрады.*
10. *Әр түсті және әр түрлі өрнектермен өрнектелген, көлемдері бірдей ромбтардан құрастырылған, көлемі 15x15 тор салыңыз.*

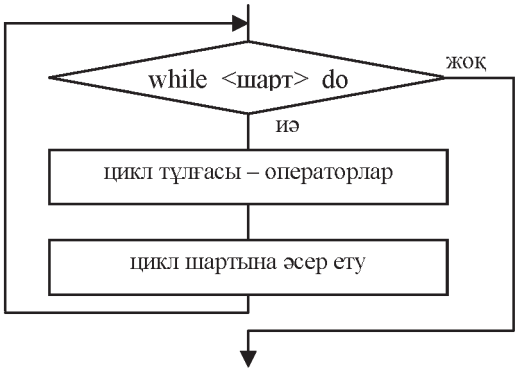
5.5 Шарты алдын ала берілген цикл операторы

Итерациялық түрдегі цикл операторлары циклдің қайталану саны алын ала белгісіз болғанда немесе цикл параметрінің өзгеру қадамы +1 немесе –1-ден өзгеше болған жағдайда қолданылады.

Шарты алдын ала берілген цикл операторының жазылуы:

WHILE <логикалық өрнек> **DO** <оператор>**;**

Логикалық өрнек, яғни шарт цикл тұлғасының әр орындалу



5.12-сурет. Шарты алдын ала тексерілетін цикл алгоритмінің орындалу схемасы

алдында есептелді. Егер логикалық өрнек **АҚИҚАТ (TRUE)** мәнін қабылдаса, онда цикл тұлғасы орындалады, егер мән **ЖАЛҒАН (FALSE)** болса циклден шығу жүзеге асырылады. Логикалық өрнек бірден жалған болған жағдайда цикл тұлғасы бір де бір рет орындалмайды. Цикл тұлғасы жәй немесе құрама оператор болуы мүмкін. Циклдің орындалу схемасы 5.12-суретте көрсетілген. Мұнда цикл операторлары (тұлғасы) ішінде шартқа кіретін айнымалыларды өзгертетін өрнектер міндетті түрде болуы тиіс, әйтпесе цикл шексіз болып кетуі мүмкін.

FOR операторы көмегімен жазылған кез келген алгоритмді **WHILE** операторы көмегімен жазуға болады. Мысалы, **F=N!** факториалының мәнін есептеу программасы:

```
…    F:=1;
      I:=1;
      WHILE I<=N DO
       BEGIN
        F:=F*I;
        I:=I+1;
       END;    …
```

Келесі мысалда Sin(x) мәнін функцияны қатарға жіктеу арқылы есептеу керек:

$$Y = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots = \sum_{i=1}^{\infty} U_i$$

Қосындыға тек төмендегі шартты қанағаттандыратын қатар мүшелерін ғана қосу керек:

$$|U_i| > \varepsilon$$

демек, қатардың әрбір мүшесі берілген ε дәлдіктен үлкен болуы тиіс.

```
 PROGRAM SINX;
VAR X,Y,E,U,Z : REAL;
              K: INTEGER;
BEGIN
    READLN(X,E);
```

```
    K:=0;
    Y:=0;
    U:=X;
    Z:=SQR(X);
  WHILE  ABS(U)>E DO
    BEGIN
      Y:=Y+U;
      K:=K+2;
      U:= -U* Z/(K*(K+1));
    END;
  WRITELN('SIN(X)=',SIN(X), 'Y=',Y);
END.
```

Программаның дұрыстығын тексеру үшін шығару операторына синусты стандартты функция көмегімен есептегенде алынған нәтижені қатар көрсету қарастырылған. Егер есептеу нәтижесінің стандартты функция көмегімен есептеу арқылы алынған нәтижеден айырмашылығы берілген дәлдіктен артық болмаса, программа дұрыс жұмыс жасайды деп есептеуге болады.

Итерациялық формула бойынша X санынан түбір алу мысалын қарастырайық.

$$Y_{i+1}=(Y_i + X/Y_i)/2, \quad Y_{i+1} - Y_i \ <= \varepsilon \text{ дәлдікпен}$$

Y_0=A алғашқы жуықтама параметр болып саналады.

```
PROGRAM SQRTX;
VAR X:REAL;  {аргумент         }
  EPS:REAL;  {есептеу дәлдігі }
   Y0:REAL;  {алғашқы жуықтау }
   Y1:REAL;  {келесі жуықтау  }
    A:REAL;  {бастапқы жуықтау}
BEGIN
  READLN( A,EPS,X);
  IF  X>0 THEN
    BEGIN
```

```
      Y0:=A;
      Y1:=(Y0+X/Y0)/2;
      WHILE ABS(Y1-Y0)>EPS DO
      BEGIN
       Y0:=Y1;
       Y1:=(Y0+X/Y0)/2
      END;
     WRITELN('Y1=',Y1,'  X=',X)
     END
  ELSE
     WRITELN(, X, 'саны нөлден кіші');
END.
```

Бақылау сұрақтары

1. *Қандай есептеулерді цикл деп атайды?*
2. *Шарты алдын ала берілген цикл операторы неше рет орындалады?*
3. *Шарты алдын ала берілген циклдан шығу қалай іске асырылады?*
4. *Шарты алдын ала берілген цикл операторының құрылымын түсіндіріңіз.*
5. *Шарты алдын ала берілген цикл операторының орындалу ретін түсіндіріңіз.*
6. *Шарты алдын ала берілген цикл операторының блок-схемасын түсіндіріңіз.*
7. *Шарты алдын ала берілген цикл операторына мысал келтіріңіз.*

Тапсырмалар

1. *15 әр түсті қилысқан дөңгелектерден тұратын көлденең сызық сызыңыз.*
2. *15 әр түсті қилысқан эллипстерден тұратын экран «диагоналін» сызыңыз.*
3. *Радиустары бірдей секторлардың тігінен орналасқан тізбегін салыңыз. Соңғы сектор – дөңгелек.*
4. *40 әр түсті сызықтан тұратын араның суретін салыңыз.*
5. *50 әр түсті сызықтан тұратын «веер» суретін салыңыз.*
6. *36 әр түсті квадраттардан тұратын дөңделек суретін салыңыз.*
7. *20 әр түсті дөңгелектерден тұратын квадрат суретін салыңыз.*
8. *100 әр түсті шеңберден тұратын синусоида фрагментін салыңыз.*
9. *Пернетақтадан енгізілген n үшін n! мәнін есептеңіз.*
10. *N натурал саны берілген. Осы санның цифрларының қосындысын есептеңіз.*

5.6 Шарты соңынан тексерілетін цикл операторы

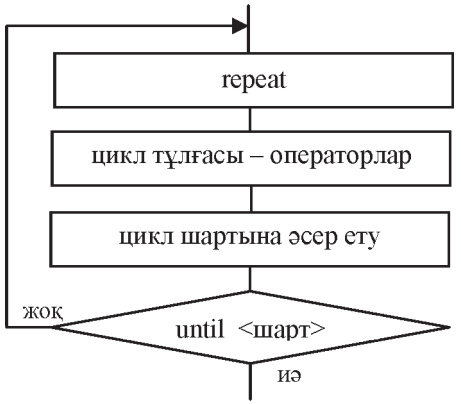
REPEAT
< 1 оператор > **[;**
< 2 оператор >**] [;**
…
< n оператор >**]**
UNTIL <логикалық өрнек>**;**

Бұл цикл қайталану саны белгісіз, бірақ циклден шығу шарты белгілі болған жағдайда қолданылады. Циклді басқаратын логикалық өрнек циклден шығу шарты болып саналады. Егер шарт АҚИҚАТ (**TRUE)** мәнін қабылдаса, онда цикл тоқтатылады. **REPEAT** операторын қолданғанда, цикл тұлғасы кем дегенде бір рет орындалады. Цикл операторларының басқа түріне қарағанда, бұл оператор **BEGIN – END** операторлық жақшасын керек етпейді, себебі операторлық жақша қызметін **REPEAT – UNTIL** түйінді сөздері атқарады.

Бұл операторлардың орындалу схемасы 5.13-суретте көрсетілген.

F=N! факториалын есептеу **REPEAT – UNTIL** операторларының көмегімен төмендегідей түрде жазылады:

```
. . .
F:=1;
I:=1;
```



5.13-сурет. Шарты соңынан тексерілетін цикл алгоритмінің орындалу схемасы

```
REPEAT
      F:=F*I;
      I:=I+1;
UNTIL  I>N;
 . . .
```

Осы оператордың көмегімен жазылған басқа мысалды қарастырайық. Берілген сандар тізбегінің ішінен 7-ге бөлінетіндер санын анықтау.

```
    PROGRAM REP;
VAR  A,K: INTEGER;
     C: CHAR;
BEGIN
     K:=0;
     REPEAT
       WRITELN(' Келесі санды енгізіңіз ');
       READLN( A);
       IF A MOD 7=0 THEN K:=K+1;
    WRITELN('Циклден шыққыңыз келе ме? Д/Y');
       READLN( C ) ;
     UNTIL ( C='Д' ) OR ( C='Y') ;
     WRITELN('KOL=',K);
END.
```

Бұл мысалда циклден шығу шарты – қойылған сауалға жауап бергенде енгізілген **Д** немесе **Y** символы болып саналады. Егер осы айтылған символдың бірі енгізілсе, **UNTIL** сөзінен кейін жазылған логикалық өрнек **АҚИҚАТ (TRUE)** мәнін қабылдайды да, циклден шығу орындалады.

Келесі мысалда A(M,N) матрицасының жұп жолдарының ең үлкен және ең кіші элементтерін тауып, солардың орнын ауыстыру керек.

```
PROGRAM OBMEN;
VAR A: ARRAY[1..30,1..30] OF INTEGER;
    I,J : INTEGER;
    K1,L1,K2,L2 : INTEGER;
    T,M,N: INTEGER;
BEGIN
```

```
  READLN(M,N);
  For I:=1 to M do
    FOR J:=1 TO N DO
        READLN( A[I,J]);
K1:=1;L1:=1;{ең үлкен элемент координатасы}
 K2:=1;L2:=1;{ең кіші элемент координатасы}
  I:=2;
  WHILE I<=M DO
       BEGIN
         FOR J:=1 TO N DO
             IF A[K1, L1]< A[I,J] THEN
                BEGIN
                   K1:=I;L1:=J;
                END
               ELSE
                 IF A[K2, L2]> A[I,J] THEN
                    BEGIN
                       K2:=I;L2:=J;
                    END;
         I:=I+2;
       END;
  T:= A[K1, L1];
  A[K1, L1]:= A[K2, L2];
  A[K2, L2]:=T;
  FOR I:=1 TO M DO
      BEGIN
        FOR J:=1 TO N DO
            WRITE ( A[I,J] :6);
        WRITELN ;
      END;
END.
```

Бізге тек жұп жолдар керек болғандықтан, жол индекстерін өзгерту үшін **WHILE** циклы қолданылады. Мұны **FOR** циклының көмегімен орындай алмаймыз.

Турбо Паскаль тілінде **BREAK** немесе **CONTINUE** процедураларын **FOR, WHILE** және **REPEAT** циклдерының ішінде қолдануға болады. **BREAK** процедурасы циклді тоқтатып,

басқаруды циклден кейін орналасқан операторға береді. **CONTINUE** процедурасы өзінен кейін орналасқан операторларды орындамай, циклдің келесі қадамына көшеді.

Мысалы, бірөлшемді массивтің 3-ке аяқталатын бірінші элементін анықтау керек.

```
PROGRAM KON;
VAR A:ARRAY [1.. 30] OF INTEGER;
    FL: BOOLEAN;
    I,N: INTEGER;
Begin
  READLN(N);
  FOR I:=1 TO N DO
    READLN( A[I] );
  FL:=FALSE;
  FOR I:=1 TO N DO
    BEGIN
     IF A[I] MOD 10 <> 3 THEN CONTINUE;
     WRITELN('3-ке аяқталатын бірінші сан
              нөмірі ',I);
     FL:=TRUE;
     BREAK;
    END;
  IF NOT FL THEN WRITELN('3-ке аяқталатын сан
                          жоқ');
END.
```

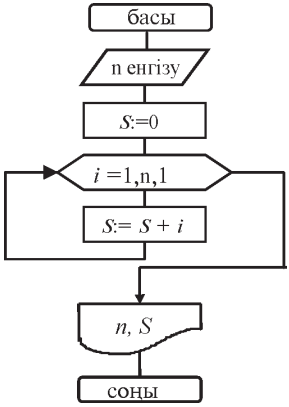
Егер 3-ке аяқталатын сан табылса, экранға мәлімет шығады. Жалауша – **FL** айнымалысы **АҚИҚАТ (TRUE)** мәнін қабылдап, программа өз жұмысын тоқтатады. Сан табылмаған жағдайда программа соңына дейін орындалады. **FL** айнымалысының мәні **ЖАЛҒАН(FALSE)** болып қалады. Экранға сан табылмағаны жайлы мәлімет шығады.

5.9-мысал. Алғашқы *n* натурал сандардың қосындысын есептейтін программа құру керек.

Қосынды жинақтау тәсілімен анықталады. Қосылатын сандар саны белгілі болғандықтан, қайталану саны белгілі цикл операторын қолданамыз. Әр сан қосылған сайын цикл параметрі 1-ге өсіп отырады. Циклға кірер алдында қосындыға нөлді меншіктеу

керек. 5.14-суретте программа алгоритмінің схемасы келтірілген. Төменде программа мәтіні берілген.

```
   Program ex;
Var
  i,n,S:integer;
Begin
  WriteLn(' n-ді енгізіңіз');
  ReadLn(n);
  S:=0;
  for i:=1 to n do
  S:=S+i;
  WriteLn(n,' санның
   қосындысы ',S, '-ке тең');
End.
```



5.14-сурет. n санды қосу алгоритмі

5.10-мысал. $1 - 1/x + 1/x^2 - 1/x^3 + ...$ қатарының қосындысын берілген ε (эпсилон) дәлдікпен анықтайтын программа жазу керек.

Математика курсынан білетініміздей тізбек түріндегі қатар қосындысы дегеніміз берілген мүшелердің қосындысы ұмтылатын шек болып саналады. Егер мұндай шек бар болса, қатар жинақталған болып, кері жағдайда жинақталмаған болып саналады. Таңбасы ауысып отыратын қатар $|r_n| < |r_{n+1}|$ болған жағдайда, жинақталған болып саналады. Мұндағы r_n және r_{n+1} – сәйкесінше қатардың n-ші және n+1-ші мүшелері. Сонымен бірге $|S-S_n| \le |r_{n+1}|$ екендігі дәлелденген. Мұнда S – жалпы қатар қосындысы, ал S_n – қатардың n мүшесінің қосындысы.

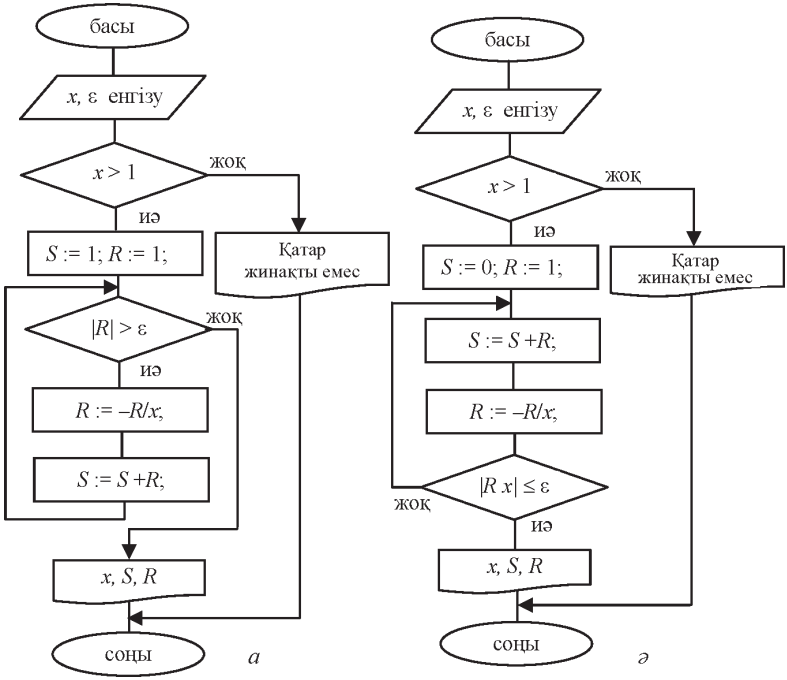
Демек, қажетті қосындыны табу үшін қатар элементтерінің жеке қосындысын тізбектің кезекті мүшесі берілген $|r_n| < \varepsilon$ қателіктен кіші болғанша, жинақтап отырамыз.

Қайталану саны белгісіз болғандықтан, цикл-әзірше нұсқасын қолданамыз. Ол үшін S=S+R операциясын екі рет қайталау керек: біреуін циклға дейін, екіншісін цикл соңына орналастырамыз. Циклға дейін жазылған S=0 және S=S+R операторларын R=1 операторымен алмастырамыз. Циклден шығу шартын да кері шартқа алмастырамыз. Цикл-әзірше нұсқасы көмегімен жазылған алгоритмнің түрі 5.15 *а*-суретіндегідей, ал программасы төмендегідей болады:

```
   Program ex;
Var S,R,X,eps:real;
Begin
  WriteLn('x және эпсилонды енгіз:');
  ReadLn(X,eps);
  if x>1 then {егер x>1 болса, онда қатардың
                  қосындысын есептейміз}
      begin
     S:=1;
     R:=1;
     while abs(R)>eps do
       begin
        R:=-R/X;
        S:=S+R;
       end;
      WriteLn('x=',x:6:2,' S=',S:8:2,',ал R=',
                R:8:6)
    end
  else WriteLn('Қатар жинақты емес');
End.
```

Осы алгоритмді цикл-дейін нұсқасы көмегімен де жазуға болады (5.15, *ә*-сурет).

```
   Program ex;
var S,R,X,eps:real;
Begin
 WriteLn('x және эпсилонды енгіз:');
 ReadLn(X,eps);
 if x>1 then
  begin
   S:=0;
   R:=1;
   repeat
    S:=S+R;
    R:=-R/X
   until abs(R*x)<=eps;
   WriteLn('x=',x:6:2,' S=',S:8:2,', ал
R=',R:8:6)
  end
 else WriteLn('Қатар жинақты емес');
End.
```



5.15-сурет. Қосындыны берілген дәлдікпен есептеу алгоритмі:
а – цикл-әзірше нұсқасы бойынша; ә – цикл-дейін нұсқасы бойынша.

Бақылау сұрақтары

1. *Шарты соңынан тексерілетін цикл дегеніміз не?*
2. *Шарты соңынан тексерілетін цикл операторы неше рет орындалады?*
3. *Шарты соңынан тексерілетін циклдан шығу қалай іске асырылады?*
4. *Шарты соңынан тексерілетін цикл операторының құрылымын түсіндіріңіз.*
5. *Шарты соңынан тексерілетін цикл операторының орындалу ретін түсіндіріңіз.*
6. *Шарты соңынан тексерілетін цикл операторының блок-схемасын түсіндіріңіз.*
7. *Шарты соңынан тексерілетін цикл операторына мысал келтіріңіз.*

Тапсырмалар

1. $y = x^3 - x^2 + 16x - 43$ *функциясының x [–4;4] аралығында 0,5 қадаммен өзгергендегі мәнін есептеңіз.*

$$y=\begin{cases} x-4, & x \succ 2 \\ x^2+14, & x \prec -2 \\ \dfrac{x}{4}, & \textit{басқа жағдайда} \end{cases}$$

2. *функциясының x [–5;3] аралығында 0,25 қадаммен өзгергендегі мәнін есептеңіз.*
3. *Пернетақтадан енгізілген n үшін* $s=\sin 1+\sin 2+\cdots+\sin n$ *қатарының мәнін есептеңіз.*
4. *Пернетақтадан енгізілген x және n үшін*

$$s=\cos x+\cos x\cdot\cos x+\cdots+\underbrace{\cos x\cdot\cos x\cdot\cdots\cdot\cos x}_{n\ рет}$$

қатарының мәнін есептеңіз.5. Кемпірқосақ доғасының фрагментін салыңыз.

6. *Пернетақтадан енгізілген n, r және c үшін радиусы r, түсі c n дөңгелектер тізбегін салатын программа құрыңыз.*
7. *15 әр түсті үшбұрыштардан тұратын экран «диагональын» сызыңыз.*
8. *Көлемдері бірдей 40 әр түсті тікбұрыштардан тұратын ромб суретін салыңыз.*
9. *Пернетақтадан енгізілген n үшін, әр түсті қилысқан дөңгелектерден тұратын көлемі nxn тор суретін салыңыз.*
10. *Көлемдері өсіп отыратын, 36 әр түсті дөңгелектерден тұратын шеңбер салыңыз.*

6. БАЗАЛЫҚ ҚҰРЫЛЫМДАРДЫ СИПАТТАУ

6.1 Жиымдарды сипаттау. Жиым элементтерін пайдалану

Паскаль тілінде әр айнымалы мен тұрақтының өз типтері болады. Мәндер типіне қарап, осы айнымалы немесе тұрақтыға қолданылатын операциялар жиыны және нәтиже типі анықталады. Паскаль тілінде стандартты типтер және тұтынушы анықтайтын типтер бар.

Программада қолданылатын барлық айнымалылар, оларды сипаттайтын **VAR** бөлімінде жарияланауы тиіс.

VAR <идентификатор> [,<идентификатор>,…]**:** <тип>**;**
[<идентификатор> [,<идентификатор>,…]**:** <тип>**;**…]

Мысалы,

```
VAR A:INTEGER;
    B,C:REAL;
```

Бұл мысалда А айнымалысы бүтін, ал В және С айнымалылары нақты тип ретінде сипатталған.

Тип алдын ала типтерді сипаттайтын **TYPE** бөлімінде жариялануы да мүмкін.

TYPE <идентификатор типі> **=** <тип**> ;**

Мысалы,

```
TYPE I = INTEGER;
     R = REAL;
```

Бұдан кейін А, В және С айнымалыларын төмендегідей түрде сипатталады:

```
VAR A: I;
   B,C: R;
```

Шектелген бір текті мәндер жиыны реттелген стандартты типке (бүтін, байттық, символдық және логикалық) жатады.

Саналатын тип – мәндерді тұрақты ретінде өрнектейтін идентификаторлардың реттелген жиынын анықтайды. Әрбір осындай элементке компьютер жадының бір байты бөлінеді. Ол жиымның

(массивтің) индексі ретінде қолданылады да, оның реттік нөмірі нөлден басталады. Саналатын типтің жазылуы:

TYPE <идентификатор типі> **= (**<идентификатор>
[<идентификатор>,…]**);**

Аралық (интервалды) тип – белгілі бір айнымалы қабылдай алатын оның мәндері жиынының қажетті бөлігін анықтайды. Аралық типтің екі шеткі – ең кіші және ең үлкен мәні беріледі. Әрбір элементке ЭЕМ жадының бір байты бөлінеді. Аралық типтің жазылуы:

TYPE <идентификатор типі> **=** <тұрақты>**. .**<тұрақты**>;**

Тұрақты ретінде нақты типтен басқа барлық қарапайым типтердің мәні қолданыла береді.

Мысалы,

```
TYPE GR = (DS101, DS102, DS201, DS202, DS301,
DS302);
       SPEC = DS101..DS302;
       DIGIT = 0..9;
       VAR A: DIGIT;
       B: SPEC;
       D: 100..200;
```

Жолдық немесе тіркестік тип ұзындығы 0-ден 255 символға дейінгі жолдардан тұратын сөз тіркестерін сипаттау үшін қолданылады. Жолдың, яғни сөз тіркестерінің максимальды ұзындығы тік жақша ішіне жазылады. Егер максимальды ұзындық берілмеген болса, ол 255-ке тең деп есептеледі. Тіркестік айнымалылар тіркестік тұрақтылар тәрізді компьютер жадында сөз тіркестерінің максимальды ұзындығынан қосымша 1 байтқа (нөлдік) артық орын алады. Турбо Паскальдің ерекшелігі – тіркестік символдардың әрбір элементіне оның нөмірі арқылы қол жеткізуге болады.

TYPE <идентификатор типі> **= String** [<максимальды ұзындық>];

Мысалы,

```
TYPE TSTRING = STRING[100];
           TS = STRING;
```

```
 VAR  S,S1: TSTRING;
S2: STRING[20];
SS: TS;
```

Жиым (массив) – қасиеттері бірдей, бір типті айнымалылардың реттелген жиыны. Жиым элементтері ішіндегі реттік қатынас индекс арқылы беріледі. Әр элементке бір немесе бірнеше индекс тағайындалады. Егер әр элементке бір индекстен берілсе, онда бұл бір өлшемді жиым (вектор) болып саналады. Ал, екі индекс берілсе – екі өлшемді жиым (матрица) болады да, мұнда бірінші индекс элемент орналасқан жол нөмірін, ал екіншісі бағана нөмірін көрсетеді.

TYPE <идентификатор типі> = **ARRAY** [<индекстер типі тізімі>]
OF <тип>;

<индекс типі> **::** = <қарапайым тип>

<қарапайым тип>**::** = <идентификатор типі>|<идентификатор> [,<идентификатор>] | <тұрақты>**. .**<тұрақты>

Индекс ретінде нақты типтен басқа кез келген қарапайым тип қолданыла алады. Мұнда көбінесе бүтін типтің аралық мәндері жиі қолданылады.

Мысалы,

```
TYPE T1 = ARRAY [-10..20,1..30] OF
BYTE;
     T2 = ARRAY [0..50] OF BOOLEAN;
     T3 = ARRAY [BYTE] OF INTEGER;
VAR A,B: T1;
    C: T2;
    Z: ARRAY[1..100] OF REAL;
   MAS: T3;
```

Бұл мысалдың типтерді сипаттау бөлімінде жиымның үш түрі келтірілген. T1 – жол нөмірі – 10-нан 20-ға дейін, ал бағана нөмірі 1-ден 30-ға дейін өзгеретін екі өлшемді жиым. T1 жиымы элементтерінің типі – 0-ден 255-ке дейінгі таңбасыз бүтін сандар. T2 – бір өлшемді жиым, элемент нөмірі 0-ден 50-ге дейін өзгеретін логикалық тип. T3 – бір өлшемді жиым, элементтері 0-ден 255-ке дейінгі таңбалы бүтін сандар.

Айнымалыларды сипаттау бөлімінде A және B айнымалысы T1 типті, C – T2 типті, MAS – T3 типті ретінде сипатталған. Типті алдын ала сипаттаудан басқа, жиымды Z айнымалысы сияқты айнымалыларды сипаттау бөлімінде де жариялауға болады.

Жиым элементтерін олардың индекстері арқылы пайдалануға болады. Айнымалыдағы индекстер саны элементтер санына тең. Индекстер бүтін сан, қарапайым айнымалы, арифметикалық өрнек ретінде берілуі мүмкін. Егер екі жиымның сипаттамалары бірдей болса, оларды **B:=A** меншіктеуі арқылы теңестіріп, көшірмесін алуға мүмкіндік бар.

Мысалы,

```
...
S:= S + Z[I];
P:= P * A[I][J];
C[6]:= TRUE;
P:= P * A[I,J];
R:= B[I+5,J];
MAS[I]:= MAS[I-1] * MAS[I];
...
```

Жиымды сипаттаудың түрлі тәсілдерін қарастырайық.

10 жол және 50 бағанадан тұратын бүтін типті А матрицасын сипаттау керек.

1)
```
CONST N = 10;
      M = 50;
TYPE   TMATR = ARRAY[1..N, 1..M] OF INTEGER;
VAR   A: TMATR;
```

2)
```
TYPE  TSTR = ARRAY [1..50] OF INTEGER;
     TMATR = ARRAY [1..10] OF TSTR;
VAR A: TMATR;
```

3)
```
VAR A:ARRAY[1..10,1..50] OF INTEGER;
```

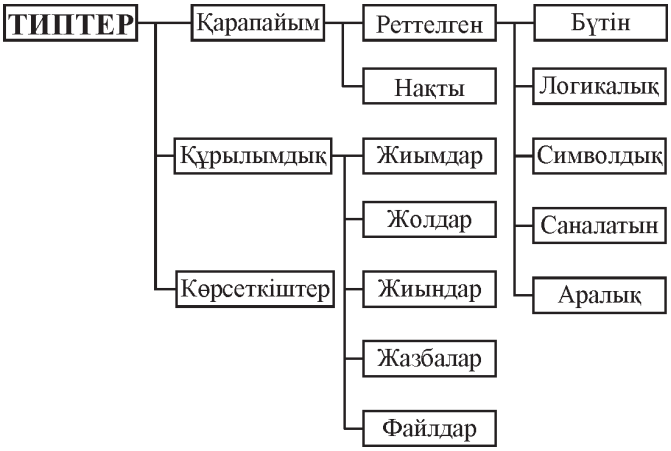
4)
```
VAR  A:ARRAY[1..10]OF ARRAY[1..50]OF INTEGER;
```

Көп жағдайларда кесте, мәтін, жиым және т.с.с. біртипті мәліметтерді өңдеуге тура келеді. Мұндай мәліметтерді сипаттау үшін құрылымдық мәндер типі қолданылады. Турбо Паскальда төмендегідей құрылымдық мәндер типтері (6.1 сурет) бар:

- жиымдар — біртипті немесе кестелік мәліметтерді сипаттау үшін;
- жолдар – сиволдық (мәтіндік) мәліметтерді сипаттау үшін;
- жиындар – абстрактілі математикалық жиындарды сипаттау үшін;

• жазбалар – әр түрлі типтегі кестелік мәліметтерді сипаттау үшін;

• файлдар – бір типте тізбектеле жазылған мәліметтер жиыны.



6.1-сурет. Типтер құрамы

Жиымдар

Индекс типі оның мүмкін мәнін анықтайды. Индекс типі ретінде *longint* және оның туынды титерінен басқа кез келген реттелген тип *(boolean, char, integer,* саналатын тип, сонымен бірге осы типтер диапазоны) көрсетілуі мүмкін.

Индекс типінің санына байланысты бір, екі және n өлшемді жиымдар болады. Екі өлшемді жымдар *матрица* деп аталады және бірінші индекс жол нөмірін, екінші индекс бағана нөмірін көрсетеді.

6.2-сурет. Жиымдар типін анықтаудың синтаксистік диаграммасы

Паскаль тіліндегі жиым элементтерінің типі файлдан басқа кез келген тип бола алады. Жиым түріндегі айнымалыны сипаттау екі тәсілмен іске асырылады:

- айнымалыларды сипаттау бөлігінде, мысалы:
  ```
  Var a:array[1..10] of integer; { 10 бүтін саннан тұратын жиым}
    b:array[byte] of char; {256 символдан тұратын жиым, жиым индексі 0-ден 255-ке дейін өзгереді}
    c:array['A'..'C',-5..-3] of byte; {9 саннан тұратын матрица}
    d:array['A'..'C'] of array [-5..-3] of byte; {құрылымы бойынша алдыңғы матрицаға эквивалентті 9 саннан тұратын матрица}
  ```
- типін алдын ала сипаттау, мысалы:
  ```
  Type mas=array[1..10] of integer; { типті сипаттаймыз}
  Var a:mas; {айнымалыны сипаттаймыз}
  ```

Турбо Паскальда индекс санына шектеу жоқ. Алайда бір программада қолданылатын индекстердің барлығының саны 65537 байттан аспауы керек.

Программадағы жиым элементтерін үш тәсілмен сипаттауға болады. Біріншіден, жиым типтік тұрақтылар немесе элементтерге мән беру арқылы инициализациялануы мүмкін. Екіншіден, жиым элементтері пернетақтадан немесе файлдан енгізуге болады. Үшіншіден, жиым элементтерін программада былай анықтауға:

- кездейсоқ сандар генераторы арқылы;
- берілген бойынша және басқа жиымнан көшіруге болады.

Жиымдарды инициализациялау. Инициализацияланатын (мәндер берілетін) жиымдарды сипаттау үшін Турбо Паскальда типтелген тұрақтылар қолданылады. Сәйкес мәндер жақша ішінде, үтір арқылы ажыратылып көрсетіледі. Көп өлшемді жиым элементтері солдан оңға қарай индексінің өсу реті бойынша жазылады. Әр ішкі жиым жеке жақшаға алынады. Матрица мәндері бір жолда реттеліп жазылады. Мысалы:

```
Const a:array[1..5] of real = (0,-
3.6,7.8,3.789,5.0);
b:array[boolean, 1..5,] of real = ((0,-3.6,7.8,
3.789,5.0),(6.1,0,-4.56,8.9,3.0));
```

{жиым төмендегідей инициализацияланады:

$b_{false,1}$=0, $b_{false,2}$=-3.6, $b_{false,3}$=7.8, ..., $b_{true,1}$=6.1,т.с.с.}

```
c:array[1..3,0..1,-2..1] of byte = (((3,6,9,6)
,(0,4,3,9)),((5,7,3,1),(45,8,0,2)),((5,9,2,3),
(1,5,8,4)));...
```

Жиымдармен орындалатын операциялар. Жиымдармен тек меншіктеу операциясын орындау ғана анықталған.

Меншіктеу – бір жиым элементтерін екінші жиымға көшіру. Бұл операцияны тек бір типтес жиымдармен орындауға блады.

Егер жиымдар бір жолда үтір арқылы сипатталса, олар біртипті болып саналады. Мысалы:

```
Var a,b:array[boolean] of real;
... a:=b; ...
```

немесе, егер бастапқыда жиым типі анықталып, сонан кейін осы типтегі жиымдар анықталса:

```
Type mas=array[boolean] of real;
Const a:mas=(3.6,-5.1);
Var b:mas;
... b:=a; ...
```

Жиым элементтерін пайдалану – жиым элементтерімен орындалатын іс-әрекеттерді орындау болып саналады. Жиымның белгілі бір элементін пайдалану үшін жиым атын және тік жақша ішіне үтір арқылы ажырата отырып, жиым элементтерінің индекстерін көрсету керек, мысалы:

```
Var a:array[char,boolean] of real;
```
{матрицаны жариялаймыз}

```
... a['A',true]:=5.1; ...
```
{мәнді $a_{A,true}$ элементіне меншіктейміз}

Индекс мәнінің нақты нөмірін тікелей, мысалы а[3] немесе жанамалы түрде, айнымалының мәні орналасқан идентификаторды көрсету арқылы, мысалы, а[i] деп жазуға болады.

Индексті жанамалы түрде көрсету – жиым элементтерін тізбектеп өңдеуге мүмкіндік береді. Индекстің өзгеру аралығы жиымды жариялау кезінде белгілі болғандықтан, қайталану саны белгілі циклді қолдануға болады. Цикл параметрі ретінде жиымның жанамалы түрде берілген индекстік айнымалысы қолданылады. Мысалы:

```
   Var a:array[1..6] of integer; ...
for i:=1 to 6 do a[i]:=i; ... { i=1 болғанда a1-ге 1
```
меншіктеледі, i=2 болғанда a_2-ге 2 меншіктеледі, i=3 болғанда a_3-ке 3 меншіктеледі және т.с.с.}

Жиымдардың жанамалы адресіне қажет айнымалылар саны жиым мөлшеріне сәйкес келеді. Матрицамен жұмыс жасағанда индекстерді сақтау үшін, екі айнымалы қажет: бірі – жол нөмірін сақтау үшін, екіншісі – бағана нөмірі үшін.

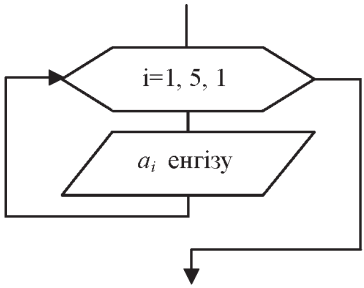
Ескерту. Көп өлшемді жиымдардан оның ішкі бір жолын, яғни *ішкі бөлігін* ерекшелеп алуға болады. Ол үшін жиымның оң жақ шеткі индекстерін алып тастап, тек жолды – ішкі жиымды анықтайтын сол жақтағы индексті қалдыру керек. Осылай матрицаның тек бір жолын бөліп алуға болады. Мысалы:

```
Type mas=array[boolean] of real; {екі нақты
                                  саннан тұратын жиым}
Const
a:array[1..2] of mas=((2.6,-5.1),(7.0,-4.2));
{төрт нақты саннан тұратын матрица}
Var b:mas;
Begin b:=a{1}; ... {b жиымына a матрицасының бірінші
```
жолы көшіріледі}

Жиымдарды енгізу-шығару. Жиымдарды енгізу-шығару қайталану саны белгілі параметрлі цикл көмегімен жеке-жеке элементтерді енгізу-шығару арқылы орындалады. Мысалы:

```
Var a:array[1..5] of real$
Begin
for i:=1 to 5 do Read(a[i]); {жиымды енгіземіз }
ReadLn; { мәндер жаңа жолдан енгізілу үшін буферді
```
тазалаймыз }

Жиым элементтерінің мәні цикл арқылы рет-ретімен енгізіледі. Мысалы, жоғарыда көрсетілген цикл үшін жиым элементтері: a_1, a_2, a_3, a_4, a_5 түрінде енгізіледі (6.3-сурет). Бұл мәндерді бос орын арқылы бір-бірінен бөле отырып бір жолға жазып немесе әр элемент мәнінен кейін Enter пернесін басып енгізуге болады.



6.3-сурет. Жиым элементтерін циклде енгізу

Көлемі үлкен матрицаны немесе бір өлшемді жиымдар элементтерін жеке-жеке жолдармен енгізіп шығарған ыңғайлы.

Мысалы:

```
Var a:array[1..5, 1..7] of real; {5 жол, әр жолда 7
                                  элементі бар матрица}
Begin
 for i:=1 to 5 do {a1, a2, a3, a4, a5 жиымының жолдарын
                   енгізетін цикл}
   for j:=1 to 7 do {i -ші жолдың элементтерін
                     енгізетін цикл:}
     Read(a[i,j]); {ai,1, ai,2, ai,3, ai,4, ai,5, ai,6, ai,7}
   ReadLn; {енгізу буферін тазалау}
end; ...
```

6.1-мысал. A(5) жиымының ең үлкен элементін және оның нөмірін анықтайтын программа жазыңыз.

Алдымен жиым элементтерін енгізу керек. Бұл операцияны орындау үшін қайталау саны белгілі циклді қолданамыз.

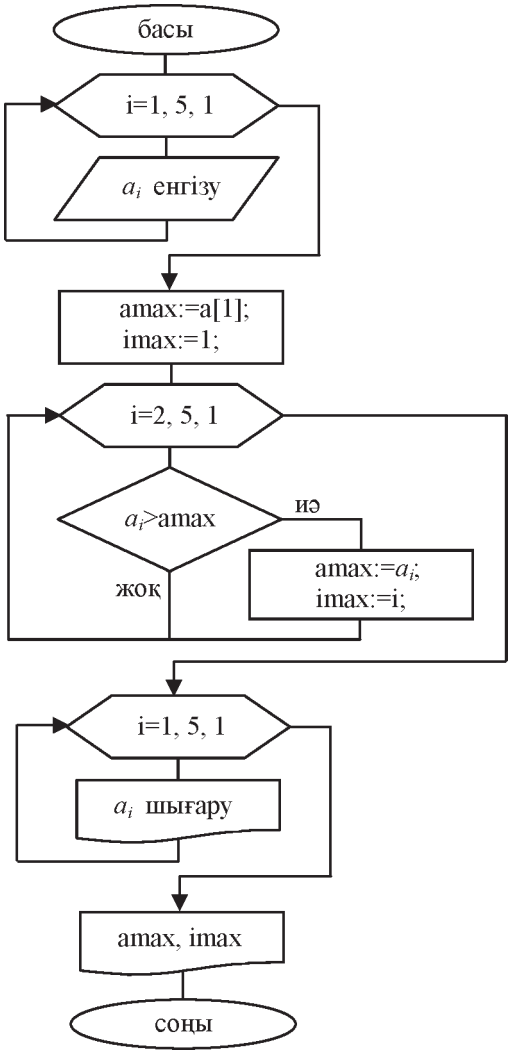
Ең үлкен элементті табу үшін: алдымен жиымның бірінші элементін ең үлкен элемент ретінде *amax* айнымалысына меншіктейміз де, осы элементтің нөмірін *imax* деп белгілейміз. Содан кейін жиым элементтерін жеке-жеке *amax* айнымалысының мәнімен салыстырып шығамыз. Егер салыстырып отырған элемент *amax* айнымалысының мәнінен үлкен болса, оны ең үлкен элемент ретінде *amax* айнымалысында меншіктейміз. Элемент нөмірін *imax* айнымалысына жазамыз (6.4-сурет).

Жиым элементтерін тізбектеп қарап шығу үшін қайталану саны белгілі циклді пайдаланамыз. Цикл параметрі 2-ден 5-ке дейін өзгереді, себебі бірінші элементті біз ең үлкен элемент ретінде алдық. Барлық элементтерді қарап шығып, біз жиымның ең үлкен элементін және оның нөмірін анықтаймыз. Бұдан кейін экранға бастапқы жиымды және оның ең үлкен элементі мен оның нөмірін шығару керек.

6.4-суретте программа алгоритмінің схемасы берілген. Жиым элементтерін енгізу-шығару операциялары ұқсас екендігі алгоритм схемасында параметрлі цикл (модификатор) блогынан байқалып тұр.

Төменде программа мәтіні келтірілген.

```
Program ex;
Var a:array[1..5] of real;
  amax:real; i,imax:byte;
Begin {Жиым элементтерін енгізуге сұраныс}
 WriteLn('5 сан енгізіңіз:'); {жиым элементтерін енгізу}
 for i:=1 to 5 do Read(a[i]);
  ReadLn; {жиымның ең үлкен элементін іздеу}
 amax:=a[1]; imax:=1;
  for i:=2 to 5 do
   if a[i]>amax then
    begin
     amax:=a[i];
     imax:=i;
    end;
  {жиымды шығару}
  WriteLn('Бастапқы мәндер:');
  for i:=1 to 5 do Write(a[i]:5:2);
  WriteLn;
  {нәтижені шығару}
  WriteLn('Ең үлкен элемент= ',amax:5:2,',
оның
       нөмірі= ',imax);
End.
```



6.4-сурет. Жиым элементтерінің ең үлкенін табу

Бақылау сұрақтар

1. *Қандай мәндер типтерін білесіз?*
2. *Мәндердің қарапайым типтерін атап, оларға мысал келтіріңіз.*
3. *Мәндердің күрделі типтерін атаңыз.*
4. *Жиым дегеніміз не?*
5. *Жиым элементі деп нені атайды?*

6. *Жиым элементінің индексі дегеніміз не?*
7. *Жиымда неше элемент болуы мүмкін?*
8. *Жиымның бір элементінде неше индекс болуы мүмкін?*
9. *Жиым элементтерінің типі қандай болуы мүмкін?*
10. *Жиым элементтерінің индекстерінің типі қандай болуы керек?*

Тапсырмалар

1. *Он элементтен тұратын бірөлшемді жиым берілген. Алдымен жиым элементтерін пернетақтадан енгізіп, олардың оң элементтерінің және теріс элементтерінің қосындысын жеке-жеке есептеп экранға шығаратын программа құрыңыз.*
2. *25 элементтен тұратын бірөлшемді жиым берілген. Пернетақтадан енгізілген а санының осы жиымда бар екендігін тексеріп, экранға мәлімет шығаратын программа құрыңыз.*
3. *25 элементтен тұратын бірөлшемді жиым берілген. Жиымның оң элементтерінің санын анықтайтын программа құрыңыз.*
4. *20 элементтен тұратын бірөлшемді жиым берілген. Жиымдағы оң және теріс элемнттердің санын анықтап, экранға мәлімет шығаратын программа құрыңыз.*
5. *15 элементтен тұратын бірөлшемді жиым берілген. Жиым элементтерінің арифметикалық ортасын анықтайтын программа құрыңыз.*
6. *20 элементтен тұратын бірөлшемді жиым берілген. Жиымның ең үлкен элементін және оның индексін анықтайтын программа құрыңыз.*
7. *30 элементтен тұратын бірөлшемді жиым берілген. Жиым элементтерін кері ретпен индекстерін көрсетіп экранға шығаратын программа құрыңыз.*
8. *20 элементтен тұратын бірөлшемді жиым берілген. Жиымның жұп және тақ элементтерінің арифметикалық ортасын есептеп, экранға шығаратын программа құрыңыз.*
9. *20 элементтен тұратын бірөлшемді жиым берілген. Жиымдағы оң топтар санын (оң топ дегеніміз, кем дегенде екі қатар орналасқан оң сан) анықтайтын программа құрыңыз.*

6.2 Жиын типін анықтау. Жиындардың қасиеттері.

Жиындармен орындалатын операциялар

Қарапайым типтегі (нақты типтен басқасы) біртиптес мәліметтерді **жиынға** (set) біріктіруге болады.

Жалпы алғанда, жиынды төмендегідей етіп сипаттауға болады:

TYPE <идентификатор типі>= **SET OF** <компонент типі>**;**

Жиындағы компонент типі (базалық тип) – аралық немесе саналатын тип болып табылады. Жиын типіндегі айнымалы мәндері оның компоненттерін үтірмен бөле отырып, тік жақша ішінде жазылады.

Мысалы,

```
TYPE INTERVAL= 5..10;
     MN=SET OF INTERVAL;
VAR  PR:MN;
```

PR айнымалысы төмендегі мәндерді қабылдай алады:

[5,6,7,8,9,10], [5], [6],…, [5,6], [5,7],…, [6,7,8],…, [ ],

мұндағы [ ] – бос жиын, себебі бұнда базалық типті көрсететін өрнек жоқ. Бос жиын жиындардың барлық типімен үйлесе береді.

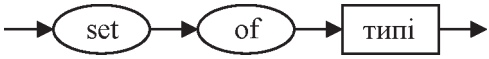
Турбо Паскаль тілінде жиындарға төмендегідей шектеулер қойылады:

- Жиын элементтерінің саны 256-дан аспауы керек.
- Жиын элементтері тек қарапайым типтегі (нақты типтен басқа) мәндер бола алады.
- Жиын құрамына кіретін элементтер алдын ала анықталуы тиіс.
- Жиын элементтері кез келген ретпен жазылуы мүмкін.

Жиын – қазіргі математиканың негізгі ұғымдарының бірі. Ол қайталанбайтын объектілердің реттелмеген жиыны деген мағынаны береді. Жалпы жағдайда, жиында бірде бір элемент болмауы да мүмкін. Мұндай жиын *бос жиын* деп аталады.

Турбо Паскаль тілінде жиындарды сипаттауға арналған құрылымдық тип бар. Жиын типіндегі мәліметтер, өзара байланысқан біртипті элементтер тобынан тұрады. Элементтер арасындағы байланысты тек программа жасаушы ғана анықтай алады және ол бақыланбайды.

Жиындық тип кез келген бір базалық типтегі элементтер тобы ретінде жарияланады (6.5-сурет). Ол элементтерінің саны 0-ден 255-ке дейінгі аралығынан аспайтын, шекті жиындар түрінде сипатталады.



6.5-сурет. Жиындық типтің синтаксистік диаграммасы

Базалық типке – мәндерінің саны 255-тан асатын *integer* және *longint* типтерінен басқа кез келген реттік тип жатады. Базалық тип ретінде оның мәндерінің белгілі бір аралықтары (диапазондары) ғана пайдаланыла алады.

Жиындағы элементтердің орналасу реті қадағаланбайды. Бұны математикадағы жиын ұғымына сәйкес деп айтуға болады.

Жаңа жиындық типті әдетте алдын ала жариялайды да, сонан кейін айнымалылар мен тұрақтыларды сипаттау бөлімінде пайдаланады. Мысалы:

```
Type
 Digits = set og 1..100; {«1-ден 100-ге дейінгі бүтін
                          сандар жиыны» типі}
 Setchar = set of char; {«ASCII кестесінің
                          символдары жиыны» типі}
 letter = set of 'a'..'z'; {«латын алфавитінің кіші
                  әріптерінің жиыны» типі}
 logic = set of boolean; {« логикалық мәндер
                          жиыны « типі}
Var
 mychar:setchar; {айнымалы - ASCII кестесінің
                          символдары жиыны }
 bool:logic; { айнымалы - логикалық мәндер жиыны }
 mydig:Digits; { айнымалы - 1-ден 100-ге дейінгі бүтін
                  сандар жиыны }
simst:letter; { айнымалы - латын алфавитінің кіші
              әріптері жиыны }
```

Жиын типін айнымалылар мен тұрақтыларды сипаттаған кезде де анықтауға болады, мысалы:

```
Var
 number:set of 1..31; { айнымалы - 1-ден 31-ге
                  дейінгі бүтін сандар жиыны}
 cif:set of 0..9; { айнымалы – сандар жиыны }
 kods:set of #0..#255; { айнымалы - ASCII
                          кестесінің
                 символдары жиыны }
 workweek, week:set of dayn; { айнымалы – апта
                          күндері жиыны }
```

Жиын типіндегі айнымалы мәндері де жиын болады. Осы типтегі айнымалылар мен тұрақтылардың нақты мәндері тік жақшаға алынып, үтірмен ажыратылып жазылған элементтер тізімі түріндегі жиын құраушысы (конструкторы) көмегімен анықталады. Жиын элементтері базалық типтегі тұрақты, айнымалы және өрнек түрінде берілуі мүмкін. Сонымен бірге жиынға кіретін элементтер мәнінің аралығын – интервалын да көрсетуге болады, мысалы:

[ ] – бос жиын;

`[2,3,5,7,11]` – тек бірнеше бүтін сандардан тұратын жиын;

`['a','d','f','h']` – бірнеше латын әріптерінен тұратын жиын;

`[1,k]` - 1 бүтін санынан және k айнымалысының ағымдағы мәнінен тұратын жиын;

`[k..2*k]` – бүтін сан және k айнымалысының 2*k өрнегінің нәтижесіне дейінгі мәнінен тұратын жиын;

`[2..100]` - 2-ден 100-ге дейінгі бүтін сандар жиыны;

`[1,2,3..7]` - 1,2,3,4,5,6,7 сандарынан тұратын бүтін сандар жиыны;

`[red,yellow,green]` – саналатын типтегі үш элементтен тұратын жиын

Жиындарды инициализациялау. Жиын типтегі айнымалыларды типтік тұрақтыларды қолдану арқылы инициализациялауға болады. Мысалы:

```
Type setnum=set of byte;
Const  S:setnum=[1..10]; {инициализацияланған айнымалы, оның программадағы бастапқы мәні 1-ден 10-ға дейінгі бүтін сандардан тұратын жиын}
```

Жиындармен орындалатын операциялар. Жиындық типтегі мәндермен жұмыс жасау үшін арнайы операциялар қарастырылған. Олар жиындар теориясында анықталған: бірігу, қиылысу және жиындарды толықтыру операциялары. Бұл екі орынды операциялар, операнд ретінде жиын типіндегі мәндерді қабылдайтын жиын және өрнек қолданылады. Операндтың екеуі де бір жиындық типке жатуы керек. Бұл операндтардың Турбо Паскальдағы белгіленуі мен геометриялық бейнеленуі 6.1-кестеде келтірілген.

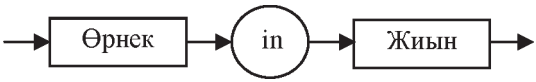
6.1-кесте

Математикалық жазылуы	Турбо Паскальдағы операция	Геометриялық бейнеленуі	Операция нәтижесі
A ∪ B	A+B		А және В жиындарының бірігуі – А және В жиындары элементтерінен тұратын жиын
A ∩ B	A*B		А және В жиындарының қилысуы – А жиынына да, В жиынына да кіретін элементтерден тұратын жиын
A\B	A-B		А және В жиындарын толықтырушы А жиынының В жиынына кірмейтін элементтерінен тұратын жиын

Мысалы:

```
[1,2] + [3,4] = [1,2,3,4];
[1..10] * [3,8,9,15,23,45] = [3,8,9];
[1..15] - [3,8,9,15,23,45] =
[1,2,4..7,10..14];
[red,blue,green,black] * [blue,magenta,yellow]
= [blue].
```

Қатынас операциялары. Жоғарыда айтылған операциялардан басқа жиындарға қатынас операциялары қолданылады. Қатынас әрекеттерінің операндтары ретінде жиын типіндегі айнымалылар мен өрнектер қолданылады. 6.2-кестеде математикалық жиындарды салыстыру операциялары мен Турбо Паскальдағы қатынас операциялары арасындағы сәйкестік келтірілген.



6.6-сурет. Элементтің жиынға кіретінін тексеретін синтаксистік диаграмма

Элементтің жиынға кіретіндігін тексеру операциясы. ***In*** сөзімен белгіленген элементтің жиынға кіретіндігін тексеру опе-

рациясының синтаксистік диаграммасы 6.6-суретте көрсетілген. Қатынас операцияларына қарағанда *in* операциясында бірінші операнд жиындық мәннің базалық типіне жатуы тиіс. *In* операциясының нәтижесі логикалық мән болып табылады.

Төменде қатынас операциялары мен элементтің әр түрлі жиынға кіретіндігін тексеру операцияларының мысалдары келтірілген.

6.2-кесте

Математикада жазылуы	Турбо Паскаль-дағы операция	Операция нәтижесі	
		TRUE	**FALSE**
A = B	A = B	A және B жиыны өзара тең	A және B жиыны өзара тең емес
A ≠ B	A <> B	A және B жиыны өзара тең емес	A және B жиыны өзара тең
A≤B	A <= B	A жиынының барлық элементтері B жиынына кіреді	A жиынының кейбір элементтері B жиынына кіреді
A≥B	A >= B	B жиынының барлық элементтері A жиынына кіреді	B жиынының кейбір элементтері A жиынына кіреді

['a','b'] = ['b','a'] – операция нәтижесі TRUE;
[4,5,6] = [4..6] – операция нәтижесі TRUE;
['c','b'] = ['c','b','d'] – операция нәижесі FALSE;
[2,3,5,7] <= [1..9] – операция нәтижесі TRUE;
[3,6..8] <= [2..7,9] – операция нәтижесі TRUE;
[5..8,9..12] >= [6,8,11] – операция нәтижесі TRUE;
10 in [2,4,6,8,10,12,14] – операция нәтижесі TRUE;
k=1,3,5 болғанда k in [1,3,5,7,9] – операция нәтижесі TRUE және k=2,4,6 болғанда – FALSE.

Жиындық типтегі мәндерді енгізуге және шығаруға болмайды. Алайда жиын элементтерінің мәнін енгізіп және оларды біріктіру операциясының көмегімен жиынға қосуға болады, мысалы:

```
S:=[ ]; {бастапқы жиын бос}
Read(n);
while not Eof do
 begin
```

```
  S:=S+[n]; {бастапқы жиын мен енгізілген элементтерден
             тұратын жиынды біріктіру}
end; ...
```

Жиын элементтерін шығару үшін: цикл ішінде базалық типтегі барлық элементтердің жиынға кіретіндігін тексеріп, кіретіндерін шығарады. Мысалы:

```
for i:='a' to 'z' do
if i in S then Write(i:3);
```

6.2-мысал. Енгізілген сөздің әріптен немесе сызу белгісінен және сөздің ішіне арнайы символдардың кіретіндігін анықтайтын программа жазу керек.

Латын алфавитінің бас әріптері мен кіші әріптері және сызу символдарынан тұратын жиын құрастырамыз:

['A'..'Z','a'..'z','_'].

Сонымен бірге, сөздің екінші символынан бастап кездесуі мүмкін символдар жиынын құрастырамыз:

['A'..'Z','a'..'z','_','0'..'9'].

Программаға сөз тіркесін енгізіп, бірінші символын, сонан соң цикл ішінде қалған символдарын да тексеру керек.

```
Program ex;
Var sr:string;
  key:boolean;
  i:integer;
Begin
  WriteLn('Жолды енгізіңіз');
  ReadLn(st);
  if st[1] in ['A'..'Z','a'..'z','_'] then
     {бірінші символды тексеру}
   begin
    i:=2;
    key:=true;
    while (i<=length(st)) and key do {қалған
              символдарды тексеру}
    if st[i] in ['A'..'Z','a'..
```

```
'z','_','0'..'9']
     then inc(i)
     else key:=false;
    if key then WriteLn('st,' жолы -
        идентификатор.')
     else WriteLn(st,' жолында рұқсат
                  етілмеген символдар бар.');
  end
  else
   WriteLn(st,' жолы рұқсат етілмеген
символдан
      басталады.');
End.
```

6.3-мысал. Берілген натурал санның жазылуындағы цифрлар санын анықтайтын программа құрастыру керек.

Есепті шығару үшін енгізілген сан модулін символдық жолға түрлендіріп, осы символдардан жиын құрастырамыз. Енді жиындағы символдардың 0-ден 9-ға дейінгі сандар жиынына кіретіндігін тексеріп, экранға мәлімет шығарамыз.

```
Program ex;
Var n:longint;
  st:string;
  mnoj:set of '0'..'9';
  i:integer; j:char;
Begin
  WriteLn('Сан енгізіңіз:');
  ReadLn(n);
  Str(abs(n),st);
  mnoj:=[]; {бастапқыда жиын бос}
  for i:=1 to length(st) do
   mnoj:=mnoj+[st[i]]; {жиын құрастырамыз}
  WriteLn(n,' санының жазылуына келесі сандар
      кіреді:');
  for j:='0' to '9' do {жиынға кіретін
     цифрларды экранға шығарамыз}
   if j in mnoj then Write(j+' ');
End.
```

6.4-мысал. Пернетақтадан енгізілген символдар тіркесінен тұратын және бос орын арқылы ажыратылған бірнеше сөздерден құрастырылған жолдың ішіндегі:

- жолдың әр сөзінде кезесетін;
- жолдың тек бір сөзінде кездесетін;
- жолдың кем дегенде бір сөзінде кездесетін;
- жолдың екі және одан артық сөздерінде кездесетін

дауысты әріптер санын анықтайтын программа құру керек.

Есепті шығару үшін «ASCII кодтағы символдар жиыны» типін анықтаймыз. Қазақ (орыс) алфавитінің дауысты әріптерін жиындық типтелген тұрақты ретінде береміз. Соңғы және аралық нәтижелерді сақтау үшін жиындық типтегі айнымалыларды анықтаймыз:

- *res1* – әр сөзге кіретін дауысты әріптер жиыны;
- *res2* – тек бір сөзге кіретін дауысты әріптер жиыны;
- *res3* – бір сөйлем ішіндегі дауысты әріптер жиыны;
- *res4* – бірнеше сөзге кіретін дауысты әріптер жиыны;
- *mns1* – ағымдағы сөзге кіретін дауысты әріптер жиыны.

Программаға бір жолды енгізіп, ондағы сөздерді жеке-жеке бөліп аламыз. Әр сөзге кіретін дауысты әріптер жиынын – *mns1* анықтаймыз.

Егер енгізілген жолда тек бір ғана сөз бар болса, онда

res1 = res2 = res3 = mns1,

болады да, *res4* бос жиын болып шығады.

Егер енгізілген жолдағы сөздер саны бірден артық болса, онда әрбір жаңа сөз нәтижелік жиынды төмендегідей етіп өзгертетін болады:

1. әр сөзге кіретін дауысты әріптер жиыны, осыған дейін табылған сөздерге кіретін дауысты әріптер жиыны мен **res1 З mns1** сөзінің дауысты әріптер жиынының қиылысуына тең болады;

2. бірнеше сөзге кіретін дауысты әріптер жиыны – *res4* жаңа сөзде қайталанатын дауысты әріптер санына өседі (бірігу):

res4 және (res3 # mns1);

3. бір сөйлемдегі дауысты әріптер жиыны – *res3* сөздегі дауысты әріптер санына өседі (бірігу):

res3 З mns1

4. тек бір сөзге кіретін *res2* – дауысты әріптер жиынын, барлық сөздерді өңдегеннен кейін, бір сөйлемдегі дауысты әріптеры жиыны мен бірнеше сөзге кіретін дауысты әріптер жиынының айырмасы ретінде анықтаймыз:

res1 / res4

```
Program ex;
Type setchar=set of char;
Const G:setchar=['а','я','у','ю','э','е','о
','ё','и','ы']; { «дауысты дыбыстар жиыны «
типтік тұрақты}
Var res1, {әр сөзге кіретін дауысты әріптер
жиыны}
res2, {тек бір сөзе кіретін дауысты әріптер
жиыны}
res3, {сөйлемнің дауысты әріптер жиыны}
res4, {бірнеше сөзге кіретін дауысты әріптер
жиыны}
mns1:setchar; {ағымдағы сөздің дауысты әріптер
жиыны}
st,slovo:string; ch:char;
i,k:integer; first:boolean;
Begin
WriteLn('Бастапқы жолды енгізіңіз:');
ReadLn(st); {бастапқы жолды оқимыз}
st:=st+' '; {өңдеу жеңіл болу үшін жолдың
соғына бос орын қосамыз}
first:=true; {«бірінші жол» белгісі}
while st<>'' do {сөзді қиып алу және өңдеу ци-
клы}
begin
k:=pos(' ',st);
slovo:=copy(st,1,k-1); {қиып алынған сөз}
Delete(st,1,k); {сөзді жолдан алып тастаймыз}
{сөзге кіретін дауысты әріптер жиынын
анықтаймыз}
mns1:=[]; {бастапқы күй «бос жиын»}
for i:=1 to k-1 do
```

```
if slovo[i] in G then {егер дауысты дыбыс бол-
са, онда}
mns1:=mns1+[slovo[i]]; {жиынға қосамыз}
{нәтижелік жиындарды құрастырамыз}
if first then {егер бірінші сөз болса, онда}
begin
res1:=mns1; {әр сөзге кіреді}
res2:=mns1; {тек бір сөзге кіреді}
res3:=mns1; {кездескен дауысты әріптер}
res4:=[]; {бірнеше сөзге кіреді}
first:=false; {«бірінші сөз» белгісін алып та-
стаймыз}
end
else {егер бірінші сөз болмаса, онда}
begin
res1:=res1*mns1; {әр сөзге кіретіндер}
res4:=res4+res3*mns1; {бірнеше сөзге
кіретіндер}
res3:=res3+mns1; {кездескен дауысты әріптер}
end
end;
res2:=res3-res4; {тек бір сөзге кіретіндер}
{сқйлемді өңдеу нәтижелерін шығарамыз}
WriteLn('Әр сөзге кіретін дауысты әріптер жиы-
ны:');
for ch:=#0 to #255 do if ch in res1 then
Write(ch:2);
WriteLn;
WriteLn('Тек бір сөзге кіретін дауысты әріптер
жиыны:');
for ch:=#0 to #255 do if ch in res2 then
Write(ch:2);
WriteLn;
WriteLn('кем дегенде бір сөзге кіретін дауысты
әріптер жиыны:');
for ch:=#0 to #255 do if ch in res3 then
Write(ch:2);
```

```
WriteLn;
WriteLn('Бірнеше сөзге кіретіндер дауысты
әріптер жиыны:');
for ch:=#0 to #255 do if ch in res4 then
Write(ch:2);
WriteLn;
End.
```

Бақылау сұрақтар

1 *Жиын дегеніміз не?*
2. *Жиындармен орындалатын қандай операцияларды білесіз?*
3. *Turbo Pascal программалау жүйесінде жиындармен орындалатын операциялар қалай жазылады?*
4. *Жиын құрастырушы дегеніміз не және ол қандай жағдайда қолданылады?*
5. *Жиын типті айнымалылар қалай хабарланады?*
6. *Бос жиын дегеніміз не және ол қалай беріледі?*
7. *жиын элементтерін шығаруды қалай ұйымдастырамыз?*

Тапсырмалар

1. *1..100 бүтін сандар жиынынан 444 қалдықсыз бөлінетін сандар жиынын бөліп алыңыз. Жиынды экранға шығарыңыз.*
2. *латын әріптерінен құралған жиыннан өз атыңызға кірмейтін әріптерден тұратын жиынды құрастырыңыз. Жиынды экранға шығарыңыз.*
3. *Пернетақтадан енгізілген фамилияға да, атқа да және текке де кіретін символдардан тұратын жиын құрастырыңыз. Жиынды экранға шығарыңыз.*
4. *Пернетақтадан енгізілген фамилияға, атқа және текке кіретін символдардан тұратын жиын құрастырыңыз. Жиынды экранға.*
5. *'0'..'9' диапозонына кіретін және пернетақтадан енгізілген туған жыл датасына кірмейтін символдардан тұратын жиын құрастырып, экранға шығарыңыз.*
6. *1..100 бүтін сандар жиынынан, бүтін сан квадраты болатын сандардан жиын құрастырыңыз. Жиынды экранға шығарыңыз.*
7. *Пернетақтадан 10 элементтен тұратын сызықтық кестеге енгізілген бүтін сандар жиынынан осы кестенің ең кіші элементіне бөлінетін сандар жиын құрастырыңыз.*
8. *Сәйкесінше n және m символдан тұратын g және f символдық файлдары берілген. g файлына да, f файлына да кіретін символдар жиынын құрастырыңыз.*

6.3 Жазбаларды хабарлау. Жазба элементтерімен жұмыс істеу.

Түрлі нұсқадағы жазбалар

Жазбаны сипаттау (RECORD)

Жазба – өріс деп аталатын, саны нақты компоненттерден тұратын мәндер құрылымы. Әр өрістің өз идентификаторы және типі бар. Жазба компоненттерінің әрқайсысына тікелей қол жеткізуге болады және оларды іріктеп жаңартуға болады. Жазбадағы идентификатор бірегей (уникальным) болуы керек. Жазбаның жеке өрісіне қол жеткізу үшін, оның: жазба, нүкте және өріс идентификаторынан тұратын құрама атауын көрсету керек. Жазбаны процедура немесе функция параметрі ретінде көрсетуге болады, алайда функция жазба мәні бола алмайды.

Жазба типінің жалпы түрдегі сипаттамасы:

TYPE <идентификатор типі>= **RECORD**
<11идентификатор >**[,**< 12идентификатор >**,…]:** <1 тип >**;**
< 21идентификатор >**[,**< 22идентификатор >**,…]:** <2 тип >**;**
. . .
END;

Мысалы:

```
TYPE  TA= RECORD
            P1: REAL;
            P2: CHAR;
            P3: BYTE
          END;
       VAR A: ARRAY[1..10] OF TA;
```

Бұл мысалда әр элементі **TA** типті жазба болып келген бір өлшемді жиым сипатталған.

Жазбаны программаның айнымалыларды сипаттау бөлімінде де жариялауға болады.

```
VAR C : RECORD
            P1: REAL;
            P2: CHAR;
            P3: BYTE
        END;
```

Мысал. Құрылымы төмендегідей жазба жиымы берілген:

- топ шифры;
- сынақ кітапшасының нөмірі;
- пән коды;
- баға.

ИС101 тобы студенттерінің орташа бағасын анықтау керек. Жиымды енгізгенде ең соңғы жазба «99999» топ шифры.

```
PROGRAM SRBALL;
TYPE ZAP=RECORD
     SHG:STRING[5];
     NZK:INTEGER;
     KD:1..100;
     OC:2..5
   END;
VAR MAS:ARRAY[1..100] OF ZAP;
    K,N,I:BYTE;
    SUM:REAL;
BEGIN
   I:=0;
   REPEAT
    INC(I);
   READLN (MAS[I].SHG, MAS[I].NZK, MAS[I].KD,
                                   MAS[I].OC)
   UNTIL MAS[I].SHG='99999';
   N:=I; SUM:=0; K:=0;
   FOR I:=1 TO N DO
     IF MAS[I].SHG='ИС101' THEN
       BEGIN
         SUM:=SUM+MAS[I].OC;
         INC(K)
       END;
     IF K<>0 THEN SUM:=SUM/K;
  WRITELN ('ИС-101 тобының орташа баллы=',SUM)
END.
```

Біріктіру операторы

Жазба компоненттерімен жұмыс жасағанда, құрама атауы қолданылады. Атауды қысқартып, жазбамен жұмыс жасауды ыңғайлы ету үшін **WITH** – біріктіру операторы қолданылады.

WITH < RECORD типті айнымалының идентификаторы **> DO**

< оператор>;

WITH – біріктіру операторын қолдансақ, жазба компонентіне сілтеме жасағанда айнымалы атын жазбауға болады.

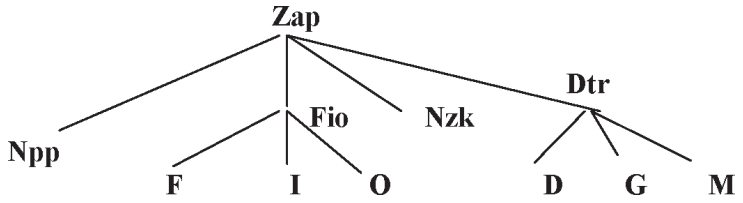
Жоғарыдағы мысалды біріктіру операторының көмегімен жазсақ төмендегідей болып шығады:

```
      . . .
I:=0;
REPEAT
        INC(I);
        WITH MAS[I] DO
        READLN(SHG,NZK,KD,OC)
      UNTIL MAS[I].SHG='99999';
      N:=I; SUM:=0; K:=0;
      FOR I:=1 TO N DO
      WITH MAS[I] DO
        IF SHG='ИС101' THEN
          BEGIN
            SUM:=SUM+OC;
            INC(K)
          END;
            . . .
```

Жазбаны біріктірілген түрде сипаттап, **WITH** операторын қолдануға болады. Студенттер жайлы жазба келесі өрістерден тұрады делік:

- реттік нөмір;
- ФАТ (бұл өріс – фамилия, аты, тегі өрістерінен тұрады),
- сынақ кітапшасының нөмірі;
- туған жылы (бұл өріс – жыл, ай, күн өрістерінен тұрады).

Мұндағы жазба құрылымын граф түрінде төмендегідей бейнелеуге болады:



Жазбаны енгізу және санау программасының құрылымы келесідей түрде болады:

```
USES CRT;
TYPE ZAP=RECORD
 NPP:BYTE;
 FIO:RECORD
 F,I,O:STRING[15];
 END;
  NZK:WORD;
 DTR:RECORD
  G:1970..2000;
 M:STRING[3];
   D:1..31
   END;
  END;
VAR A:ZAP;
  K,N:BYTE;
BEGIN CLRSCR;
 K:=0;
  WITH A DO
  WITH FIO DO
     WITH DTR DO
REPEAT
 INC(K);
 WRITELN('ВВОД ');
 READLN(NPP);
 READLN(F);
 READLN(I);
 READLN(O);
READLN(NZK);
```

НЕМЕСЕ

```
USES CRT;
TYPE ZAP=RECORD
  NPP:BYTE;
  FIO:RECORD
F,I,O:STRING[15];
   END;
    NZK:WORD;
    DTR:RECORD
   G:1970..2000;
    M:STRING[3];
        D:1..31
 END;
   END;
  VAR A:ZAP;
     K,N:BYTE;
 BEGIN CLRSCR;
      K:=0;
WITH A,FIO,DTR DO
 REPEAT
          INC(K);
WRITELN('ВВОД ');
     READLN(NPP);
       READLN(F);
      READLN(I);
       READLN(O);
    READLN(NZK);
       READLN(G);
      READLN(M);
```

```
 READLN(G);                          READLN(D);
 READLN(M);                        UNTIL D=99;
 READLN(D);                         WRITELN(K);
UNTIL D=99;                          READKEY
WRITELN(K);                       END.
READKEY
END.
```

Бірнеше нұсқалы жазбалар

Жазба құрамы мен құрылымы өзінің қандай-да бір *белгі-өріс* деп аталатын өрісінің мәніне байланысты динамикалық түрде өзгеруі.

Жалпы түрде алғанда нұсқалы жазбалар келесі түрде жазылады:

TYPE <идентификатор типі >= **RECORD**
<1 өріс идентификаторы>**:** <1 тип >**;**
<2 өріс идентификаторы >**:** <2 тип >**;**
. . .
CASE <селектор>**:**< селектор типі > **OF**
<1 нұсқа белгісі >**:(**< 11 нұсқа өрісі>**:**< 11 тип >
[;<12 нұсқа өрісі >**:**<тип 12>**;**< 13 нұсқа өрісі >**:**< 13 тип >**;. . .]);**
< 2 нұсқа белгісі >**:(**< 21 нұсқа өрісі >**:**< 21 тип >
[;<22 нұсқа өрісі >**:**<тип 22>**;**< 23 нұсқа өрісі >**:**< 23 тип >**;. . .]);**
< k нұсқа белгісі >**:(**< k1 нұсқа өрісі >**:**< k1 тип >
[;<k2 нұсқа өрісі >**:**< k2 тип >**;**< k3 нұсқа өрісі >**:**< k3 тип >**;. . .]);**
. . .
< m нұсқаның белгісі>**:( )**
END;

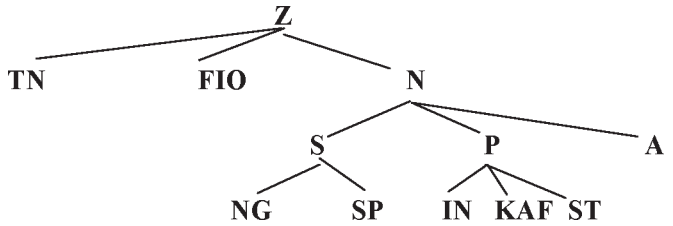
Бұл сипаттамада нұсқа бөлімі 1, 2 ..., өрістері жататын тұрақты бөлімнен кейін жазылады. Белгі типі селектор типіне сәйкес болу керек. Егер қандай-да бір нұсқа белгісіне өріс сәйкес келмесе *m* нұсқасының белгісіндей бос дөңгелек жақша жазылады.

Келесі құрылымдағы жазбаны сипаттау керек делік. Әр жазбада: табельдік нөмір және фамилия жазылған өрістер бар. Жазбаның кімге тиісті екендігіне байланысты басқа өрістер құрылымы өзгеріп отырады:

- студенттер үшін: топ нөмірі және мамандық;

• мұғалімдер үшін: институт, кафедра, жұмыс стажы;
• қызметкерлер үшін қосымша өріс жоқ.

Жазба құрылымының граф түрінде бейнеленуі:



Сәйкес жазба құрылымы мәндерінің сипаттамасы:

```
TYPE TZ=RECORD
       TN:BYTE;
       FIO:STRING;
       CASE N:CHAR OF
          'P': ( IN:BYTE; KAF:STRING; ST:BYTE );
          'S': ( NG:BYTE; SP:INTEGER );
          'A': ( )
       END;
VAR Z:TZ;
```

Жазба – өріс деп аталатын, саны нақты компоненттерден тұратын мәндер құрылымы. Жазбалар әр түрлі, бірақ логикалық байланысқан мәліметтер үшін қолданылады. Жазбаның әр өрісінің жазбаны сипаттау кезінде берілетін өзіндік аты болады.

Турбо Паскальда жазбалардың екі типі бар: бекітілген өрістер және нұсқалы жазбалар.

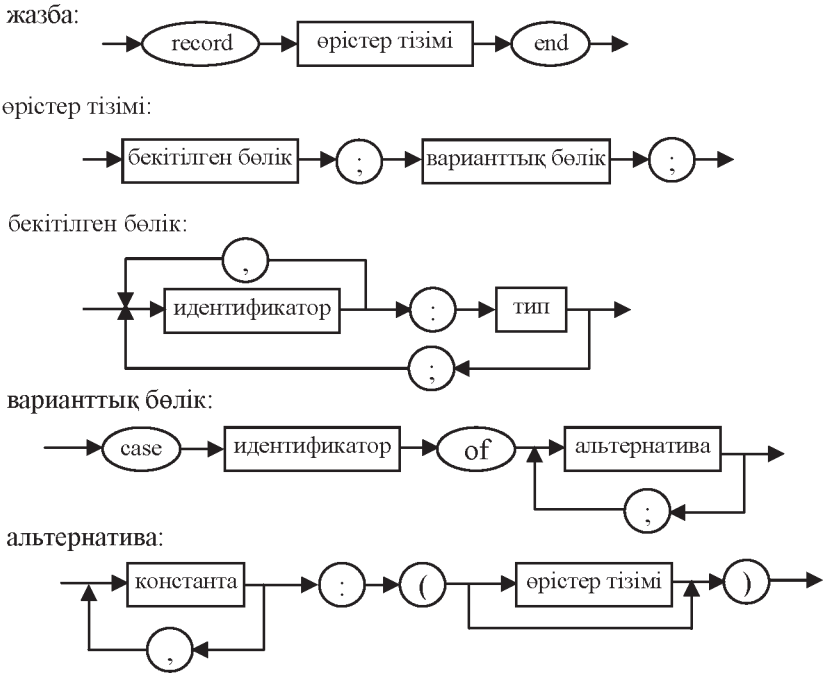
Бекітілген өрістері бар жазбалар. Бекітілген өрістері бар жазбаның диаграммасы 6.7-суретте келтірілген.

Тілдің кез келген мәндерінің типі тәрізді жазбаны екі тәсілмен анықтауға болады:

• айнымалыларды сипаттағанда, мысалы:

```
Var Zap1,Zap2:record {5 өрістен тұратын
екі жазба}
F,S:real; {нақты типтегі екі өріс}
A,B:integer; {бүтін типті екі өріс}
C:char; {символдық типтегі өріс}
end;
```

```
Zap3:record {3 өрістен тұратын жазба}
S:string[80]; {ұзындығы 80 байт символдық
жол}
A:array[1..20] of real; {20 нақты эле-
менттен тұратын бірөлшемді жиым}
Flag:boolean; {логикалық типтегі өріс}
end; ...
```



6.7-сурет. Жазбаларды жариялаудың синтаксистік диаграммасы

- жазба типі алдын ала хабарланады, мысалы:

```
Type Zt1=record {« 5 өрістен тұратын жаз-
ба» типі}
F,S:real; {нақты типтегі екі өріс}
A,B:integer; {бүтін типті екі өріс}
C:char; {символдық типтегі өріс}
end;
Zt2=record {« 3 өрістен тұратын жазба «
типі}
```

```
    S:string[80]; {ұзындығы 80 байт символдық
    жол}
    A:array [1..20] of real; {20 нақты эле-
    менттен тұратын бірөлшемді жиым }
    Flag:boolean; {логикалық типтегі өріс}
    end;
    Var Zap1,Zap2:Zt1; {Zt1 типті екі айныма-
    лы}
    Zap3:Zt2; ... {Zt2 типті айнымалы}
```

Жазба өрісі ретінде, бастапқыда анықталған немесе жазба ішінде анықталған басқа жазба да қолданылуы мүмкін. Мысалы:

```
Type Human=record {қызметкер жайлы жазба}
Fio:record {өріс типі « 3 өрістен тұратын жаз-
ба»}
Fam, {фамилия}
Name, {аты}
Otch:string; {және қызметкер тегі}
end; {Fio}
BirthDay:record {өріс типі «3 өрістен тұратын
жазба »}
Day:1..31; {күн}
Month:1..12; {ай}
Year:word; {туған жылы}
end; {Birthday}
end; ... {Human}
немесе
Type Data=record
Day:1..31;
Month:1..12;
Year:word;
end;
Famio=record
Fam,Name,Otch:string;
end;
Human=record {типі «қызметкер жайлы жазба»}
Fio:Famio; {өріс типі Famio}
BirthDay:Data {өріс типі Data}
end; ...
```

Программаның жеке айнымалысы және жазба жиымы ретінде де сипаттауға болады. Мысалы:

```
Var Sotr:Human;
Otdel:array[1..20] of Human;
```

Жазбаларды инициализациялау. Қандай да бір нақты жазбаға бастапқы мәнді типтік тұрақты арқылы меншіктеуге болады. Жазба өрістерінің бастапқы мәндері жақша ішінде үтір арқылы жазылады. Әр өріс үшін оның аты және қос нүктеден кейін мәні жазылады. Мысалы:

```
   Const
BirthDay:Data=(Year:1973; Month:6; Day:30); ...
```

Жазбалармен орындалатын операциялар. Жазбалармен төмендегідей операцияларды орындауға болады.

Жазба өрістерін пайдалану. Мысалы, жоғарыда сипатталған *Human* типті, *Sotr* айнымалысының өрістеріне келесі жолмен қол жеткізуге болады:

```
   Sotr.BirthDay.Year:=25;
m:=Sotr.BirthDay.Year; ...
```

Жазба өрістерін бірнеше рет пайдалану керек болса *with* операторын қолданған жөн. Мысалы, *Human* типті, *Sotr* айнымалысының *Day* өрісіне *with* операторы арқылы қол жеткізудің келесідей нұсқалары бар:

```
   1. with Sotr do BirthDay.Day:=30;
   2. with Sotr.BirthDay do Day:=24;
   3. with Sotr, BirthDay do Dat:=31;
   4. with Sotr do with BirthDay do Day:=7; ...
```

Жазбаларды меншіктеу. Бұл операция жазба типтері сәйкес болғанда және тізбек бойынша орындалады. Мысалы:

```
   Otdel[i]:=Sotr; ...
```

Жазбаны пернетақтадан енгізу және оларды экранға шығару осы типке сәйкес айнымалыларды енгізу және шығару ережелері бойынша орындалады.

6.5-мысал. Оқу тобы студенттері жайлы (фамилия және туған жылы) мәліметтерден тұратын жазба жиымын енгізетін программа құру керек. Фамилиясы пернетақтадан енгізілген студент жайлы мәліметтер іздеуді ұйымдастыру керек.

```
Program ex;
Type
data=record {“дата жайлы мәліметтер жазбасы ”
типі}
year:word; {жыл}
month:1..12; {ай}
day:1..31; {күн}
end;
zap=record {“студент жайлы жазба” типі}
fam:string[16]; {фамилия}
birthday:data; {туған жылы}
end;
Var fb:array[1..25] of zap; {топ студенттері
жайлы мәндер жиымы}
fff:string; {фамилия енгізуге арналған жол}
i,j,m,n:byte;
key:boolean; {іздеу кілті, егер фамилия табыл-
са - true}
Begin
WriteLn('Студенттер саны жайлы мәлімет
енгізіңіз n<=25');
ReadLn(n);
m:=0;
{бастапқы мәндерді пернетақтадан өрістен кейін
өрісті енгізу}
repeat m:=m+1;
Write(Фамилияны енгізіңіз: '); Readln(fb[m].
fam);
Write(Туған жылын енгізіңіз: '); Readln(fb[m].
birthday.year);
Write(' ай: '); Readln(fb[m].birthday.month);
Write(' күн: '); Readln(fb[m].birthday.day);
until n=m;
WriteLn;
{бастапқы мәндерді экранға with операторының
көмегімен шығару }
Writeln('Топ студенттерінің тізімі'); WriteLn;
```

```
for i:=1 to m do
with fb[i] do
begin
Write(i:2,fam:17);
with birthday do
WriteLn(year:6,month:4,day:4);
end;
WriteLn;
{мәндерді жазба жиымының ішінен іздеу}
WriteLn('Фамилияны енгізіңіз');
Readln(fff);
i:=0;
key:=false; {«мәндер табылмады « белгісі}
repeat i:=i+1;
if fb[i].fam=fff then key:=true
until key or (i=m);
{вывод результата}
if key then {егер студент табылса, ол жайлы
мәліметтер экранға шығарылады}
with fb[i] do
begin
WriteLn('Студент жайлы мәліметтер:');
Write(fam:18,' ');
with birthday do
WriteLn(day:2,':',month:2,':',year:5,' года');
end
else WriteLn(fff:18,'Студент жайлы мәлімет
жоқ');
End.
```

Нұсқалы жазбалар. Кейде әртүрлі типті мәліметтерді, кейбір өрістерінің мәніне қарай, бір жазбаның түрлі нұсқалары ретінде қарауға тура келеді. Осындай жағдайларды іске асыру үшін жазба бекітілген өрістер тізімінен басқа, өзермелі бөлімдерден тұруы мүмкін.

Диаграммадан көріп отырғандай өзгермелі бөлім бірнеше нұсқадан тұруы мүмкін. Оның әр-қайсысында бір нұсқаға тиісті

өрістер тізімі беріледі. Алдын ала әр нұсқаның сәйкестігін тексеретін тұрақтысы беріледі

Нұсқалы жазбаларды анықтайтын бірнеше мысалдар қарастырайық:

- типті алдын ала сипатталмаған нұсқа:

```
Type M:record
case {}
byte of {}
0: (by: array[0..3] of byte); {}
1: (wo: array[0..1] of word); {}
2: (lo: longint); {}
end; ...
```

- типті алдын ала сипаттау арқылы:

```
Type
Figure=(Square,Triangle,Circle);
{саналушы тип}
Paramf=record {тип –нұсқалы жазба}
X,Y:real; {бекітілген өріс}
case {өзгермелі бөлім}
Fig:Figure of {тұрақтының сәйкестігін
анықтайтын тип және айнымалы}
Square: (Side:real);
Triangle: (Side1,Side2,Side3:real);
Circle: (Radius:real);
end;
Var Param:Paramf; {айнымалыны сипаттау}
```

«Нұсқалы жазба» типі айнымалысының мәні қарапайым жазбадағыдай пернетақтадан енгізу, литерлік тұрақты және типтік тұрақты көмегімен меншіктеледі. Алайда инициалданған айнымалыны жариялаған кезде өзгермелі бөлім үшін тек бір нұсқа беріледі.

Мысалы, жазбада адам немесе кеме жайлы мәліметтер бар делік (Абылай хан – адам және кеме аты). Бұл үшін келесі нұсқалы жазбаны сипаттаймыз.

```
Type
Forma=record {адамның және кеменің тұрақты
орнын анықтайтын жазба}
```

```
case {өзгермелі бөлім}
boolean of {нұсқаның сәйкестігін тексеретін
тұрақты типі}
{адам үшін нұсқа}
True: (BirthPlace:string[40]; {мекен-жайы}
{кеме үшін нұсқа}
False: (Country:string[20]; {елі}
EntryPort: string[20]; {порт}
EntryDate: array[1..3] of word; {дата}
Count:word) {сиымдылығы}
end; ...
```

Екі жазбаның бастапқы мәндерін меншіктейтін типтік тұрақты: Object2 – адам жайлы жазба, Object 1 – кеме жайлы төмендегідей анықталады:

```
  Const
Object1: Forma = (Country: 'Қазақстан';
EntryPort: 'Ақтау';
EntryDate: (16,3,89);
Count:12);
Object2: Forma = (BirthPlace: 'Астана'); ...
```

Барлық нұсқаларда өріс идентификаторлары әр түрлі болуы керек және бекітілген өріс бөлімі атымен сәйкес келмеуі керек. Нұсқаның сәйкестігін анықтайтын кейбір тұрақты мәндері үшін, жазбаның өзгермелі бөлімінде өріс болмауы мүмкін. Мұндай жағдайда қос нүктеден кейін бос тізім «( )» қоюға болады.

Сонымен бірге, нұсқалы жазбаларды пайдаланғанда кейбір ерекшеліктерді есте сақтау керек:

- Жазба типті айнымалыны орналастыруға жадыдан, әр уақытта нұсқаның ең үлкен көлеміне сәйкес бекітілген көлемді орны бөлінеді. Демек, әр түрлі нұсқалар жадының бір бөлігінде, "бірінің орнына бірі" жазылады.

- Турбо Паскаль тілі трансляторының нұсқалы жазбалардың дұрыс өңделгендігін қадағалайтын ешқандай құралы жоқ. Демек, кез келген уақытта барлық нұсқалардың кез келген өрісіне қол жеткізуге болады.

- Сақталатын ақпараттардың сәйкестігін және оларға қол жеткізу нұсқаларын программалаушы өзі қадағалауы керек. Алайда, нұсқалы жазбалардың осы ерекшелігі мәндер типін айқындалмаған өңдеу үшін қолданылады.

Мысал:

```
Type
Perem=record {нұсқалы жазба}
case byte of
0: Wo:word; {word типті айнымалы}
1: Lo:longint; {longint типті айнымалы}
2: Re:real; {real типті айнымалы}
end;
Var C:Perem;
Begin ...
C.Lo:=0; {ауданды тазалаймыз}
C.Wo:=10; {нұсқалы өріске шаблон бойынша
таңбасыз бүтін 10 санын жазамыз}
WriteLn(C.Lo:10); ... {нұсқалы өріс мәні ша-
блон бойынша басылып шығады}
```

Бұл мысалда өріске жадыдан 6 байт орын бөлінеді. Тұтынушы осы өріспен, жазба өрістерінің сәйкес идентификаторын қолданып, кез келген үлгі (шаблон) бойынша жұмыс істей алады.

Бақылау сұрақтар

1. *Жазба түріндегі мәндер типі дегеніміз не?*
2. *Жазба қалай хабарланады?*
4. *Жазба элементтерімен (өрісімен) қандай операциялар орындауға болады?*
5. *Жазба мен жиымның айырмашылығы неде?*
6. *Жазбалармен қандай операциялар орындауға болады?*
7. *Біріктіру операторының қызметі қандай?*
8. *Нұсқалы жазба дегеніміз не?*
9. *Жазбалар қалай қолданылады?*

Тапсырмалар

1. *Файлда өз жолдастарыңыз (кем дегенде 5) жайлы жазбаларды сақтайтын программа құрастырыңыз. Мәліметтерді*

шығаруды пернетақтадан енгізілген жазба нөмірі бойынша ұйымдастырыңыз. Мысалы:
{ Жазба нөмірі: 3
Фамилия: Сәрсенбаев
Аты: Алтынбек
Тегі: Асанұлы
Туған жылы: 1967
Туған айы: 12
Туған күні: 18 }

2. *Файлда емтихан нәтижелері жайлы бірнеше (кем дегенде 10) жазбаны сақтайтын программа құрыңыз. Әр жазба үш өрістен тұрады: жазба нөмірі, фамилия, баға. Жазбаларды келесідей формада шығаруды ұйымдастырыңыз:*
{ 1 Сәрсенбаев 3 }
3. *Файлда күндізгі температура жайлы бірнеше (кем дегенде 10) жазбаны сақтайтын программа құрыңыз. Әр жазба төрт өрістен тұрады: жазба нөмірі, күн, ай, температура. Жазбалардың барлығын және ең төменгі, ең жоғары температураны экранға шығарыңыз.*
4. *Кітаптарда жалпы мәліметтер жазылған файл құрастырыңыз. Әр кітап жайлы мәлімет: автор фамилиясы, кітап аты, парақтар саны және шыққан жылы. Берілген автордың көрсетілген жылдағы кітабын табыңыз. Автор фамилиясы мен кітаптың шыққан жылы пернетақтадан енгізіледі.*
5. *Әртүрлі даталар енгізілген файл құрастырыңыз. Әр дата – күн, ай және жылдан тұрады. Перненің басылуы бойынша сәйкес мәліметтерді шығарыңыз:*
F1 – ең кіші жыл;
F2 – барлық даталар;
F3 – ең соңғы дата;
F10 – программадан шығу.
6. *Мекеме қызметкерлерінің телефон нөмірлері жазылған файл құрастырыңыз. Жазбада: қызметкер фамилиясы, аты және телефон нөмірі көрсетілген. Қызметкер фамилиясы бойынша оның телефон нөмірін іздеуді ұйымдастырыңыз.*
7. *Экспортқа шығарылатын тауарлар жайлы мәліметтер жазылған файл құрастырыңыз. Жазбада: тауар аты, тауар импорттаатын ел және экспортық партия көлемі көрсетілген. Тауар экспорттаатын елдер тізімі мен экспорттың жалпы көлемін анықтаңыз.*

7. ТУРБО ПАСКАЛЬ ГРАФИКАСЫ

7.1 Graph модулінің жалпы сипаттамасы

Турбо Паскаль компиляторының 4.0 нұсқасынан бастап, оның құрамына **Graph** – графикалық ішкі программалар кітапханасы қосылған болатын. Кітапханада графикалық экранды басқаруға арналған 50-ден астам әртүрлі процедуралар мен функциялар бар. Кітапханамен танысу жеңіл болуы үшін мұндағы процедуралар мен функциялар функционалдық қызметтері бойынша топтастырылған.

Графикалық режимге көшу және мәтіндік режимге қайтып оралу

Компьютер іске қосылғаннан кейінге стандартты және Турбо Паскаль ортасынан программа жіберуге дейінгі қалыпкүйі экранның мәтіндік режиміне сәйкес келеді. Сондықтан, компьютердің графикалық құралдарын қолданатын кез келген программа, алдымен дисплей адаптерінің графикалық режим жұмысын іске қосуы керек. Программа жұмысын аяқтағаннан кейін компьютер мәтіндік режимге қайтып оралады.

Дисплей адаптерлерінің графикалық режим жұмыстарының қысқаша сипаттамасы

Графикалық процедураларды нақты адаптермен жұмыс істеуге лайықтап орындау үшін, қажетті графикалық драйверді іске қосу керек. Драйвер – компьютердің белгілі бір техникалық құрылғысының жұмысын басқаруға арналған арнайы программа. Графикалық драйвер – графикалық режимде дисплей адаптерінің жұмысын басқарады. Турбо Паскальда барлық типтегі адаптерлер үшін графикалық драйверлер бар. Олар программалар ішіндегі жеке BGI ішкі каталогында орналасады, кеңейтілімі BGI (ағылшынның: Borland Graphics Interface – Borland фирмасының графикалық интерфейсі) болып келеді. Мысалы, CGA.BGI – CGA

адаптерге арналған драйвер, EGAVGA .BGI –EGA және VGA адаптерлеріне арналған драйвер және т.с.с.

Осы кезде шығарылып жатқан компьютерлер IBM фирмасының немесе соған сәйкес келетін адаптерлермен жабдықталған. Қазіргі кезде MDA адаптерінен басқа адаптерлердің барлығының да графикалық режимде жұмыс істеу мүмкіндігі бар. Бұл режимде дисплей экраны өте жиі орналасқан нүктелер – пикселдер жиыны ретінде қарастырылады. Пикселдердің санын программа арқылы басқаруға болады.

Нақты адаптердің графикалық мүмкіндіктері экранға, ондағы пикселдер мен бояу түстері санына байланысты болады. Сонымен бірге кейбір адаптерлер бірнеше графикалық парақтармен жұмыс істей алады. Графикалық парақ дегеніміз жедел жадының экран "картасын" жасауға, дәлірек айтқанда, әр пикселдің көрсетілуі (түсі) жайлы ақпарат жазылған аймағы. Төменде кең таралған адаптерлердің графикалық жұмыс режимдерінің қысқаша сипаттамасы келтірілген.

CGA адаптері (Color Graphics Adapter – түрлі түсті графикалық адаптер) 5 графикалық режимде жұмыс істейді. Оның төрт режимі – мүмкіндігі төмен және тек түстер жиынымен (палитра) ерекшеленетін экрандарға (көлденеңінен 320 пиксел, тігінен 200 пиксел, демек 320x200) арналған. Әр палитра үш түстен, ал экрандағы қалыпты қара түсті қосқанда төрт түстен тұрады: 0 палитра (ашық – жасыл, қызғылт, сары), 1 палитра (ашық – ақшыл көк, ақ), 2 палитра (жасыл, қызыл, қоңыр) және 3 палитра (ақшыл көк, күлгін, ашық – сұр). Бесінші ржим – мүмкіндігі жоғары 640x200 нүктелі экран үшін, бірақ мұнда әр пиксел алдын ала белгіленген бір түске ғана бояла алады немесе мүлде боялмайды. Бұл палитра тек түрлі түстерден тұрады. Графикалық режимде CGA адаптері тек бір парақты қолданады.

EGA адаптері (Enhanced Graphics Adapter – кеңейтілген графикалық адаптер) CGA адаптерінің барлық графикалық режимдерін қайталай алады. Сонымен бірге мұнда өте төменгі (640x200, 16 түс, 4 парақ) және өте жоғарғы (640x350, 16 түс, 1 парақ) режимдерде жұмыс істеуге болады. Кейбір модификацияларда монохромды (640x350, 1 парақ, 2 түс) режимде де жұмыс істеуге болады.

MCGA адаптері (Multi – Color Graphics Adapter – көптүсті графикалық адаптер) CGA адаптерімен үйлесімді және 640x480, 2түсті, 1 парақ режимі бар. Мұндай адаптерлермен IBM фирмасының PS/2 дербес компьютерлерінің кіші модельдері жабдықталған. Осы серияның жоғарғы модельдері жақсартылған VGA адаптерімен (Video Graphics Array – графикалық бейнежиым) жабдықталған. VGA адаптері CGA және EGA адаптерлерінің жұмысын толық қайталайды және оларды мүмкіндігі өте жоғары (640x480, 16 түс, 1 парақ) режиммен толықтырады.

Қазіргі кезде супер-VGA адаптерлері (SVGA) кең тараған. Бұл адаптерлер мүмкіндігі – 800x600 нүктелі, 256 түсті және одан да көп түстерді пайдалана алады. Алайда Graph кітапханасында олар үшін драйвер жоқ. SVGA адаптері VGA адаптерімен үйлесімді болғандықтан, қазіргі графикалық адаптерлерді басқару үшін EGAVGA.BGI драйверлерін қолдануға тура келеді.

Жоғарыда айтылған адаптерлермен бірге Hercules фирмасының адаптерлері де көп қолданысқа ие болған. HGC адаптерінің мүмкіндгі 720x348, пикселдер бір түске ғана боялады (әдетте ашық қоңыр) немесе монохромды болып келеді. HGC+ адаптерінің HGC адаптерінен айырмашылығы аз, ал HICC (Hercules In Color Card) адаптері – HGC + адаптерінің 16 түсті кеңейтілген түрі.

7.2 Графикалық режимді инициализациялау және мәтіндік режимге көшу

InitGraph процедурасы графикалық режимді іске қосады. Процедура тақырыбының жазылуы:

Procedure InitGraph (**var** Driver, Mode: Integer; Path:**String);**

Мұндағы Driver – Integer типті айнымалы, графикалық драйвер типін анықтайды; Mode – осы типтес айнымалы, графикалық адаптердің жұмыс режимін анықтайды; Path – драйвер файлының аты және оны іздеу жолы көрсетілген **String** типті өрнек.

Процедура шақырылардың алдында, дискідегі ақпарат ішінде графикалық драйвер жазылған файл болуы керек. Процедура осы драйверді компьютер жедел жадына жүктейді де, адаптерді графикалық режимге ауыстырады. Драйвер типі графикалық адаптер типіне сәйкес болуы керек. Драйвер типін көрсету үшін

модульде келесі алдын ала анықталған тұрақтылар қолданылады:

```
const
  Detect = 0; {типті автоматты анықтау режимі}
  CGA = 1;
  MCGA = 2;
  EGA - 3;
  EGA 64 = 4;
  EGAMono = 5;
  IBM 8514 = 6;
  HercMono = 7;
  ATT 400 - 8;
  VGA = 9;
  PC 3270 = 10;
```

Адаптерлердің көбі әр түрлі режимде жұмыс істей алады. Адаптерге қажет режимді көрсету үшін Mode айнымалысы қолданылады. Процедураны шақырарда айнымалының мәні төмендегі тұрақтылардың бірі бола алады:

```
const
  { CGA адаптері :}
  CGACO =0; {Төменгі мүмкіндік, 0 палитра }
  CGAC 1 = 1; { Төменгі мүмкіндік, 1 палитра }
  CGAC 2 = 2; { Төменгі мүмкіндік, 2 палитра }
  CGAC 3 = 3; { Төменгі мүмкіндік, 3 палитра }
  CGAHi = 4; { жоғарғы мүмкіндік }
  { MCGA адаптері: }
  MCGACO = 0; { CGACO эмуляциясы }
  MCGAC 1 = 1; { CGAC 1 эмуляциясы }
  MCGAC 2 = 2; { CGAC 2 эмуляциясы }
  MCGAC 3 = 3; { CGAC 3 эмуляциясы)
  MCGAMed = 4; { CGAHi эмуляциясы }
  MCGAHi = 5; {640x480}
  { EGA адаптері: }
  EGALo = 0; {640x200, 16 түс}
  EGAHi = 1; {640x350, 16 түс}
  EGAMonoHi = 3; {640x350, 2 түс }
  {HGC и HGC+ Адаптері: }
```

```
  HercMonoHi = 0; {720 x 348}
  { ATT400 адаптері: }
  ATT400CO = 0; { CGACO режимінің аналогі}
  ATT400C1 = 1; { CGAC 1 режимінің аналогі }
  ATT400C2 = 2; { CGAC 2 режимінің аналогі }
  ATT400C3 = 3; { CGAC 3 режимінің аналогі }
  ATT 400 Med = 4; { CGAHi режимінің аналогі }
  ATT400H1 = 5; {640 x 400, 2 түс}
  { VGA адаптері: }
  VGALo = 0; {640 x 200}
  VGAMed = 1; {640 x 350}
  VGAHi = 2; {640 x 480}
  PC 3270 H 1 = 0; (HercMonoHi аналогі }
  { 1BM8514 адаптері }
  IBM 8514 LO = 0; {640x480, 256 түс}
  IBM 8514 H 1 = 1; {1024x768, 256 түс}
```

Мысалы, драйвер CCA.BGI С дискісінің TP\BGI каталогында және 320x200 2 палитра жұмыс режимін орнату керек. Онда процедураны іске қосу төмендегідей түрде болады:

```
Uses Graph;
var
  Driver, Mode : Integer;
begin
  Driver := CGA; {Драйвер }
  Mode := CGAC2; {Жұмыс режимі }
  InitGraph(Driver, Mode, 'C:\TP\BGI');
.....
```

Егер компьютер адаптері белгісіз және программа кез келген адаптермен жұмыс жасауға бейімделген болса, онда процедураны драйвер типін автоматты анықтау түрінде іске қосу керек:

```
  Driver := Detect;
InitGraph (Driver, Mode, 'C:\TP\BGI');
```

Процедура осы тәсілмен іске қосылса, экран графикалық режимге қосылады, ал процедурадан шығар кезде Driver және Mode айнымалылары, драйвердің типін және жұмыс режимін анықтайтын, бүтін санды мәндерге ие болады. Бірнеше режимде жұмыс істей алатын мүмкіндігі бар адаптерлер үшін ең жоғарғы

режим алынады. Мәселен, CGA адаптері үшін Driver= Detect мәні Detect айнымалысына 1 (CCA) мәнін қайтарады және Mode мәні – 4 (CGAHi), ал VGA адаптері үшін Driver = 9 (VGA) және Mode = 2 (VGAHi) болуы тиіс.

GraphResult функциясы графикалық процедуралардың соңғы іске қосылуының нәтижесі жазылған Integer типтегі мәнді қайтарады. Егер қате болмаса, функция мәні – нөл, кері жағдайда – төмендегідей мәні бар теріс сан болады:

```
const
  grOk =0; {Қате жоқ }
  grInitGraph =-1; {Графикалық режим іске қосылған жоқ }
  grNotDetected =-2; {Драйвер типі анықталмаған }
  grFileNotFind =-3; {Графикалық драйвер табылмады }
  grInvalidDriver =-4;{Драйвер типі дұрыс емес}
  grNoLoadMem =-5; { Жадыда драйверді орналастыруға орын
жоқ }
  grNoScanMem =-6; { Жадыда аумақтарды қарауға орын жоқ }
  grNoFloodMem =-7; { Жадыда аумақтарды бояуға орын жоқ }
  grFontNotFound =-8; {Қаріп жазылған файл табылмады }
  grNoFontMem =-9; { Жадыда қаріпті орналастуға орын жоқ }
  grInvalidMode =-10; {Графикалық режим дұрыс емес }
  grError =-11; {Жалпы қате }
  grIOError =-12;{Енгізу-шығару қатесі}
  grInvalidFont =-13;{Қаріп форматы дұрыс емес}
  grInvalidFontNum =-14; {Қаріп нөмірі дұрыс емес}
```

GraphResult функциясы іске қосылған соң, қате белгісі алынып тасталады, қайта іске қосылғанда функция мәні нөлге тең болады.

GraphErrorMsg функциясы көрсетілген қате коды бойынша соған сәйкес мәтіндік мәлімет жазылған **String** типті мәнді береді. Функция тақырыбының жазылуы:

Function GraphErrorMsg(Code: Integer):**String**;

мұндағы Code – GraphResult функциясына қайтарылатын қате коды.

Мысалы, драйвер типін автоматты түрде анықтап, ең жоғарғы мүмкіндікті режимді іске қосуға арналған операторлар тізбегінің түрі:

```
var
```

```
  Driver, Mode, Error : Integer;
begin   Driver := Detect ; {Драйверді автоматты анықтау }
  InitGraph(Driver, Mode, ‘’); {графиканы іске қосу }
  Error := GraphResult; {Нәтиже алу }
  if Error <> grOk then { Қатені тексеру}
      begin {Іске қосу процедурасындағы қате}
         WriteLn (GraphErrorMsg(Error) ); {Мәлімет шығарамыз }
          ......
      end
      else {Қате жоқ}
  ......
```

InitCraph процедурасын іске қосқанда, қате көбіне графикалық адаптер драйвері жазылған файлдың қайда орналасқандығын дұрыс көрсетпегендіктен болады (мысалы, CGA.BCI файлы CGA адаптері үшін). Драйверді баптау ісі InitGraph процедурасын шақырғанда, драйвер атында жазылған қажет файлды іздеу маршрутын көрсетуден басталады. Мысалы, егер драйвер D дискісінің PASCAL каталогының DRIVERS ішкі каталогында орналасқан болса, шақыру мынадай түрде жазылу керек:

InitGraph(Driver, Mode, ‘d:\Pascal\Drivers’);

Ескерту. Бұдан кейінгі мысалдардың барлығында InitGraph процедурасының Driver параметрі бос жол түрінде шақырылады. Мұндай шақыру формасы тек қажет графикалық файл ағымдағы каталогта орналасқанда ғана дұрыс болып саналады. Мысалдардың қайталануын жеңілдету үшін өз компьютеріңіздің адаптеріне сәйкес файлды ағымдағы каталогқа көшіріп қоюыңыз қажет.

CloseGraph процедурасы адаптердің графикалық режимдегі жұмысын аяқтап, мәтіндік режим жұмысын қалпына келтіреді. Процедура тақырыбы:

Procedure CloseGraph;

RestoreCRTMode процедурасы қысқа мерзімге мәтіндік режимге ауысуды іске асырады. Оның CloseGraph процедурасынан ерекшелігі – графикалық режимнің орнатылған параметрлерін алып тастап, графикалық драйверді орналастыруға бөлінген жады көлемін босатпайды. Процедура тақырыбы:

Procedure RestoreCRTMode;

GetGraphMode функциясы графикалық адаптер жұмысы ре-

жимінің коды жазылған integer типті мәнді қайтарады. Функция тақырыбы:

Function GetGraphMode: Integer;

SetGraphMode процедурасы адаптердің жаңа графикалық жұмыс режимін тағайындайды. Процедура тақырыбы:

Procedure SetGraphMode(Mode: Integer);

Мұндағы Mode – орнатылатын режим коды.

Келесі программа графикалық режимнен мәтіндік жұмыс режиміне және керісінше ауысуды жүзеге асырады:

```
Uses Graph;
var
   Driver, Mode, Error: Integer;
begin
   {Графикалық режимді іске қосамыз}
   Driver := Detect;
   InitGraph(Driver, Mode, '');
   Error:= GraphResult; {Нәтижені есте
сақтаймыз}
   if Error<>grOk then {Қатені тексереміз}
      WriteLn (GraphErrorMsg(Error)) {Қате
бар}
   else
      begin {Қате жоқ}
         WriteLn('Бұл графикалық режим');
         WriteLn('»Enter»пернесін
басыңыз...':20);
         ReadLn;
         {Мәтіндік режимге ауысамыз}
         RestoreCRTMode;
         WriteLn ('Бұл мәтіндік режим,...');
         ReadLn;
         {Графикалық режимге қайтып ораламыз}
         SetGraphMode (GetGraphMode);
         WriteLn('Бұл тағы да графикалық ре-
жим...');
         ReadLn;
         CloseGraph
      end
end.
```

Бұл мысалда мәліметтерді шығару үшін графикалық режимде де, мәтіндік режимде де стандартты WriteLn процедурасы қолданылады. Егер сіздің компьютеріңіз қазақ (орыс) алфавитін қолдамайтын CGA адаптерімен жабдықталған болса, онда графикалық режимде оларды шығару мүмкін емес. Сондықтан мәліметтердің барлығын латын әріптерімен жазу керек.

DetectGraph процедурасы драйвер типі мен жұмыс режимін қайтарады. Процедура тақырыбы:

Procedure DetectGraph(var Driver,Mode: Integer);

Мұндағы Driver –драйвер типі; Mode – жұмыс режимі.

CetGraphMode функциясынан ерекшелігі – бұл процедура Mode айнымалысына қолданылып отырған адаптер графикалық режимінің ең үлкен мәнін қайтарады.

GetDriverName функциясы жүктелген драйвер аты жазылған String типті мәнді қайтарады. Функция тақырыбы:

Function GetDriverName: String;

GetMaxMode функциясы адаптердің жұмыс режимдерінің саны жазылған Integer типті мәнді қайтарады. Функция тақырыбы:

Function GetMaxMode: Integer;

GetModeName функциясы нөмірі бойынша адаптердің жұмыс режимі және экран мүмкіндіктері жазылған String типті мәнді қайтарады. Функция тақырыбы:

Function GetModName(ModNumber: Integer): String;

мұндағы ModNumber – режим нөмірі.

Келесі программа графикалық режим іске қосылғаннан кейін, экранға жүктелген драйвер аты және оның жұмыс режимдері жайлы мәліметтер шығарады.

```
Uses Graph;
var
   a,b: Integer;
begin
   a := Detect;
   InitGraph (a, b, '');
   WriteLn (GetDriverName);
   for a := 0 to GetMaxMode do
      WriteLn(GetModeName(a):10);
   ReadLn;
```

```
   CloseGraph
end.
```

GetModeRange процедурасы. Берілген графикалық адаптердің жұмыс режимдерінің диапазонын қайтарады. Процедура тақырыбы:

Procedure GetModeRange(Drv: Integer; **var** Min, Max: Integer); мұндағы Drv – адаптер типі; Min – ең төменгі режим нөмірі қайтарылатын, Integer типті айнымалы; Max – ең жоғарғы режим нөмірі қайтарылатын integer типті айнымалы.

Егер Drv параметрінің мәні дұрыс көрсетілмесе, онда процедура екі айнымалыға да –1 мәнін қайтарады. Процедураны шақырар алдында экранның графикалық жұмыс режимін орнатпаса да болады. Келесі программа экранға барлық адаптерлердің атын және жұмыс режимдерінің нөмірін шғарады.

```
Uses Graph;
var
   D,L,H: Integer;
const
   N:array [1..11] of String [8]=
      ('CGA', 'MCGA', 'EGA',
      'EGA64', 'EGAMono', 'IBM8514',
      'HercMono', 'ATT400', 'VGA',
      'PC 3270', 'Қате');
begin
   WriteLn('Адаптер Мин. Макс.');
   for D:=1 to 11 do
      begin
         GetModeRange(D, L, H);
         WriteLn(N[D], L:7, H:10)
      end
end.
```

7.3 Сызықтармен, нүктелермен және фигуралармен жұмыс істеу

Көптеген графикалық процедуралар мен функциялар экран курсорының ағымдағы позициясын қолданады. Графикалық режимнің экран курсорының мәтіндік редактор курсорынан

ерекшелігі, ол экранда көрінбейді. Экран көрсеткішінің позициясы кез келген экрандағы координата тәрізді сол жақ жоғарғы бұрышқа байланысты беріледі. Сол жақ жоғарғы бұрыштың координатасы – 0,0. Сонымен, экранның көлденең координатасы солдан оңға, ал тік координатасы – жоғарыдан төмен қарай өседі.

GetMaxX және **GetMaxY** функциялары экранның ағымдағы режимнің көлденең және тік максимальды координаталары жазылған Word типті мәнді қайтарады. Мысалы:

```
Uses Graph;
var
   a,b: Integer;
begin
   a:= Detect;
   InitGraph(a, b,'');
   WnteLn(GetMaxX, GetMaxY:5);
   ReadLn;
   CloseGraph
end.
```

GetX және **GetY** функциялары экран курсорының ағымдағы көлденең және тік координаталары жазылған integer типті мәнді қайтарады. Координаталар терезенің сол жақ жоғарғы бұрышына байланысты анықталады. Егер мәлімет шығарылатын терезе орнатылмаса, онда координаталар экранның сол жақ жоғарғы бұрышынан бастап есептеледі.

SetViewPort процедурасы графикалық экранда төртбұрышты терезе орнатады. Процедура тақырыбы:

Procedure SetViewPort(XI,Y1,X2,Y2: Integer; ClipOn: Boolean);

мұндағы X1... Y2 – терезенің сол жақ жоғарғы (X1, Y1) және оң жақ төменгі (X2, Y2) бұрыштарының координаталары; ClipOn – экрандағы бейненің кішіреймейтін элементінің "қиылатын" бөлігін анықтайтын Boolean типті өрнек.

Терезе координатасы әр уақытта экранның сол жақ жоғарғы бұрышына байланысты анықталады. Егер ClipOn параметрінің мәні True болса, бейненің терезе шекарасына симайтын элементтері қиып тасталынады, кері жағдайда терезе шекарасы ескерілмейді. Бұл параметрді басқару үшін, модульде арнайы тұрақтыларды қолдануға болады:

const

ClipOn = True ; {Қиып тастауды қосу}

ClipOff = False ; { Қиып тастауды қоспау }

Келесі мысал ClipOn параметрінің қызметін көрсетеді. Программа ClipOn параметрінің мәні әртүрлі, екі тікбұрышты терезе және олардың ішіне бірнеше дөңгелек салады. Көрнекі болуы үшін терезелер жақтауларға алынады (7.1-сурет).

```
Uses Graph,CRT;
var
   x,y,e: Integer;
   x11, y11, x12, y12, {1-ші терезе координа-
тасы}
   x21, x22, {2-ші терезенің сол жақ жоғарғы
бұрышы}
   R, {Бастапқы радиус} k: Integer;
begin
   DirectVideo:=False; {CRT модулінде
бейнежадыға тікелей қол жеткізуді алып тастай-
мыз}
   {Графикалық режимді іске қосамыз}
   x:= Detect;
   InitGraph(x, y, ' ');
   {Нәтижені тексереміз}
   e:= GraphResult;
   if e <> grOk then
      WnteLn (GraphErrorMsg (e)) {Қате}
   else
      begin {Қате жоқ}
{Экран мүмкіндіктеріне байланысты координата-
ларды есептейміз}
         x11:= GetMaxX div 60;
         x12:= GetMaxX div 3;
         y11:= GetMaxY div 4;
         y12:= 2*y11;
         R:= (x12-x11) div 4;
         x21:= x12*2;
         x22:= x21+x12-x11;
{Тезезе саламыз}
```

```
         WriteLn('ClipOn:':10,'ClipOff:':40);
         Rectangle(x11, y11, x12, y12);
         Rectangle(x21, y11, x22, y12);
{1-терезені белгілеп, оның ішіне төрт дөңгелек
саламыз}
         SetViewPort{x11, y11, x12, y12,
ClipOn);
         for k := 1 to 4 do
            Circle(0,y11,R*k) ;
{2-терезені белгілеп, оның ішіне дөңгелек са-
ламыз}
         SetViewPort(x21, y11, x22, y12,
ClipOff);
         for k := 1 to 4 do
            Circle(0,y11,R*k);
{кез келген перненің басылуын күтеміз}
         if ReadKey=#0 then k := ord(ReadKey);
         CloseGraph
      end
end.
```

GetViewSettings процедурасы ағымдағы графикалық терезенің координатасы мен қиып тастау белгісін қайтарады. Процедура тақырыбы:

Procedure GetViewSettings(**var** ViewInfo: ViewPortType);

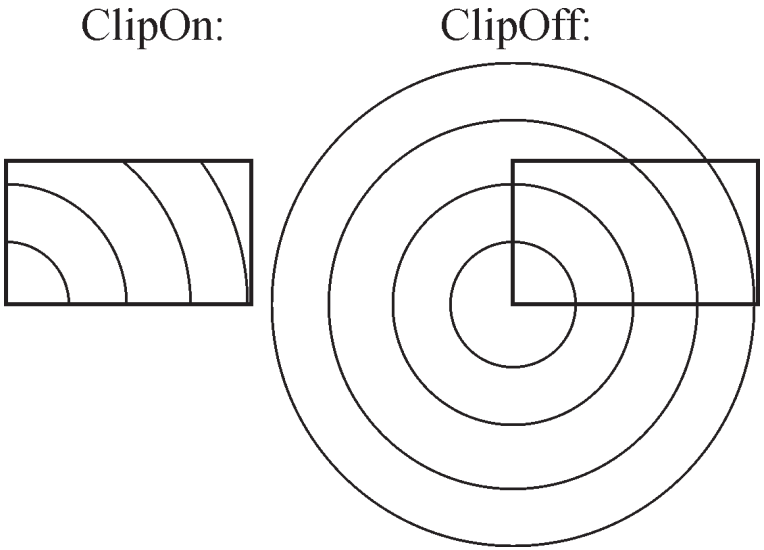
мұндағы Viewlnfo – ViewPortType типті айнымалы. Бұл тип Graph модулінде төмендегідей анықталған:

```
type
   ViewPortType = record
      xl,yl,x2,y2: Integer; {Терезе координа-
тасы}
      Clip: Boolean {Қиып тастау белгісі}
   end;
```

MoveTo процедурасы курсордың жаңа орнын тағайындайды. Процедура тақырыбы:

Procedure MoveTo(X,Y: Integer);

мұндағы X, Y – курсордың, сәйкесінше, жаңа көлденең және тік координаталары.



7.1-сурет. Терезедегі бейнелердің қиындылары

MoveRel процедурасы курсордың жаңа орнын салыстырмалы координаталар арқылы тағайындайды. Процедура тақырыбы:

Procedure MoveRel(DX, DY: Integer);

мұндағы DX, DY – курсордың, сәйкесінше, жаңа көлденең және тік координаталарының өсімшесі.

Өсімше курсордың процедураны шақыруға дейін тұрған орнына байланысты беріледі.

ClearDevice процедурасы графикалық экранды тазалайды. Процедура шақырылғаннан кейін курсор экранның сол жақ бұрышына орналасады, ал экранның өзі SetBkColor процедурасында анықталған фон түсіне боялады. Процедура тақырыбы:

Procedure ClearDevice;

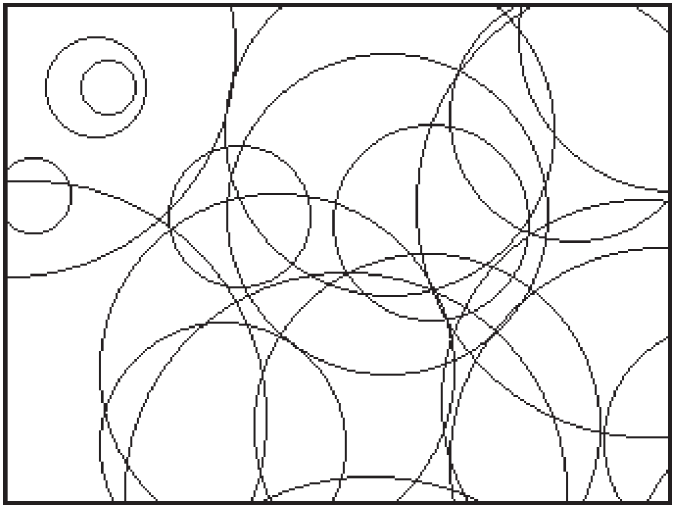
ClearViewPort процедурасы графикалық терезені тазалайды, ал егер осы уақытқа дейін терезе анықталмаса, онда экранды тазалайды. Терезе тазаланғанда, ол ағымдағы палитраның 0 нөміріне боялады. Курсор сол жақ жоғарғы бұрышқа орналасады. Процедура тақырыбы:

Procedure ClearViewPort;

Келесі мысалда экранда терезе салынады да, ол кездейсоқ дөңгелектермен толтырылады (7.2-сурет). Кез келген пернені

басқан кезде терезе тазаланады. Программадан шығу үшін Enter пернесін басыңыз.

```
Uses CRT,Graph;
var
   x1,y1,x2,y2,Err: Integer;
begin
   {Графикалық режимді іске қосамыз}
   xl := Detect;
   InitGraph(xl,x2,'');
   Err:= GraphResult;
   if ErrOgrOk then
      WriteLn (GraphErrorMsg(Err))
   else
      begin
{Экран мүмкіндігіне байланысты терезе координатасын анықтаймыз}
         x1:= GetMaxX div 4;
         y1:= GetMaxY div 4;
         x2:= 3*x1;
         y2:= 3*y1;
{Терезе саламыз}
         Rectangle(x1,y1,x2,y2);
         SetViewPort(xl+l,yl+l,x2-l,y2-
1,ClipOn);
{Терезені кездейсоқ дөңгелектермен толтырамыз}
         repeat
      Circle (Random (GetMaxX)), Random
(GetMaxX),
            Random (GetMaxX div 5));
         until KeyPressed;
{Терезені тазалап, Enter пернесінің басылуын күтеміз}
         ClearViewPort;
         OutTextXY (0,0, 'Press Enter …' )
         ReadLn
         CloseGraph
      end
end.
```



7.2-сурет. Кездейсоқ шеңберлер жиыны

GetAspectRatio процедурасы экран жақтауларының қатынасын бағалауға мүмкіндік беретін екі санды қайтарады. Процедура тақырыбы:

Procedure GetAspectRatio(var X,Y: Word);

мұндағы X, Y – Word типті айнымалылар. Бұл айнымалыларға қайтарылатын мәндер, графикалық экран жақтауларының қатынасын пикселмен есептеуге мүмкіндік береді. Осы айнымалылар арқылы табылған коэффициентті дөңгелек, квадрат, т.с.с. дұрыс геометриялық фигураларды салуда қолдануға болады. Мысалы, егер тік жақтауы L пикселге тең квадрат салу керек болса, мынадай операторларды қолдану қажет:

GetAspectRatio (Xasp, Yasp);

Rectangle(x1, y1, x1+L*round (Yasp/Xasp), y1+L);

Ал егер L квадраттың көлденең ұзындығын анықтайтын болса, онда мынадай операторды қолданамыз:

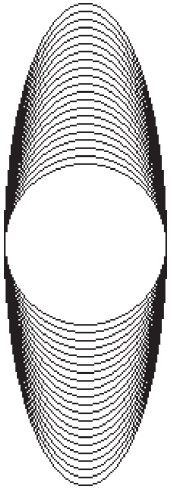
Rectangle(x1,y1,x1+L,y1+L*round(Xasp/Yasp));

SetAspectRatio процедурасы графикалық экран жақтауларының масштабтық қатынас коэффициентін анықтайды. Процедура тақырыбы:

Procedure SetAspectRatio(X,Y: Word);

мұндағы X, Y – жақтаулардың тағайындалатын қатынасы.

Келесі программа, экран жақтауларының қатынасы әртүрлі 20 дөңгелек салады (7.3-сурет).



7.3-сурет. Экран жақтауларының қатынасы әртүрлі дөңгелектер

```
Uses Graph,CRT;
const
   R=.50;
   dx=1000;
var
   d,m,e,k: Integer;
   Xasp,Yasp: Word;
begin
   d:=Detect;
   InitGraph(d, m,.'');
   e:=GraphResult;
if e <> grOk then
      WriteLn(GraphErrorMsg(e))
   else
      begin
         GetAspectRatio(Xasp, Yasp);
         for k:=0 to 20 do
            begin
               SetAspectRatio(Xasp+k*dx,Yasp);
```

```
            Circle(GetMaxX div 2,GetMaxY div 2,R)
               end;
            if ReadKey=#0 then k:= ord(ReadKey);
            CloseGraph
         end
end.
```

SetActivePage процедурасы бейнежадының көрсетілген парағын екпінді етеді. Процедура тақырыбы:

Procedure SetActivePage(PageNum: Word);

Мұндағы PageNum – парақ нөмірі.

Процедура көппарақты жұмысты қолдайтын адаптерлермен (EGA, VGA және т.с.с.) жұмыс істейді. Нақты түрде, процедура графикалық шығарылымды бейнежадының басқа ауданына жібереді, алайда мәтінді Write/WriteLn операторларының көмегімен тек осы мезетте көрініп тұрған парақта шығаруға болады (екпінді терезе көрінбеуі мүмкін). Парақтар нөмірі нөлден басталады.

SetVisualPage процедурасы нөмірі берілген парақты көрінетін етеді. Процедура тақырыбы:

Procedure SetVisualPage(PageNum: Word);

мұндағы PageNum – парақ нөмірі.

Процедура көппарақты жұмысты қолдайтын адаптерлермен (EGA, VGA және т.с.с.) жұмыс істейді. Парақтар нөмірі нөлден басталады.

Келесі программа алдымен көрінетін парақта квадрат салады, көрінбейтін парақта дөңгелек салады. Enter пернесін басқаннан кейін көрінетін парақ ауысады.

```
Uses Graph;
var
   d,m,e: Integer;
   s: String;
begin
   d:= Detect;InitGraph(d, m, '');
   e:= GraphResult;
if e <> grOk then
   WriteLn (GraphErrorMsg(e))
```

```
else {Қате жоқ. Драйвердің бейнежадымен
көппарақты жұмыс істейтіндігін тексереміз:}
   if d in [HercMono,EGA,EGA64,MCGA,VGA] then
      begin{Көппарақты режимді қолданамыз}
         if d<>HercMono then
            SetGraphMode(m-1);
         {Көрінетін парақты толтырамыз}
      Rectangle(10,10,GetMaxX div 2,GetMaxY
div 2);
         OutTextXY(0,0,'Page 0. Press En-
ter...');
         {Көрінбейтін парақты толтырамыз}
         SetActivePage (1);
         Circle(GetMaxX div 2, GetMaxY div 2,
100);
  OutTextXY(0,GetMaxY-10,'Page 1. Press En-
ter...');
         {Парақтарды көрсетемі}
         ReadLn;
         SetVisualPage(1);
         ReadLn;
         SetVisualPage (0);
         ReadLn;
         CloseGraph
      end
   else
      begin {Драйвер көппарақты режимде жұмыс
істемейді}
         s:= GetDriverName;CloseGraph;
         WriteLn('Адаптер ',s,' тек 1 парақты
режимді қолдайды');
      end
end.
```

If d <> HercMono **then** SetGraphMode (m-1); операторына көңіл аударыңыз. Бұл оператодың көмегімен EGA, MCGA, VGA адаптерлерінде көппарақты режимді кепілді түрде орнатуға

болады. Жоғарыда айтылғандай, графикалық режим Driver = Detect арқылы іске қосылғаннан кейін, адаптердің ең жоғары нөмірлі жұмыс режимі орнатылады. Аталған адаптерлер тек бір графикалық парақпен жұмыс жасай алады. Ал екі парақпен жұмыс істеуді қамтамасыз ету үшін, режим нөмірін кішірейту керек.

Сызықтар және нүктелер

PutPixel процедурасы көрсетілген координаталарда берілген түспен нүкте салады. Процедура тақырыбы:

Procedure PutPixel(X / У : Integer/ Color: Word)/

мұндағы X, Y – нүкте координаталары; Color – нүкте түсі.

Координаталар терезенің немесе экранның сол жақ жоғарғы бұрышқа байланысты беріледі.

Келесі программа экранға "жұлдызды аспанды" шығарады және оны өшіреді. Программадан шығу үшін кез келген пернені басыңыз.

```
Uses CRT, Graph;
type
   Pixeltype = record
   x,y:Integer;end;
const
   N=5000;{ "Жұлдыз" саны}
var
   d,r,e,k: Integer;
   x1,y1,x2,y2: Integer;
   a: array [1..N] of
Pixeltype;{Координаталар}
begin
   {Графиканы іске қосамыз}
   d:= Detect;InitGraph(d, r,'');
   e:= GraphResult;
   if e<>grOk then
      WriteLn(GraphErrorMsg(e))
   else
      begin
         {Экран ортасында терезе саламыз}
```

```
          x1:= GetMaxX div 4;
          y1:= GetMaxY div 4;
          x2:= 3*x1;
          y2:= 3*y1;
          Rectangle(x1,y1,x2,y2);
          SetViewPort(x1+1,y1+1,x2-1,y2-
1,ClipOn);
{"Жұлдыз" салып, олардың координаталарын есте
сақтаймыз}
          for k:=1 to N do with a[k] do begin
             x:=Random(x2-x1);
             y:=Random(y2-y1)
          end;
{Шығару циклы}
          repeat
             for k:=1 to N do
                 with a[k] do {"Жұлдызды"
жағамыз}
                    PutPixel(x,y,white);
             if not KeyPressed then
              for k:=N downto 1 do with a[k] do
{"Жұлдызды" өшіреміз}
              PutPixel(x,y,black)
          until KeyPressed;
          while KeyPressed do k :=
ord(ReadKey);
          CloseGraph
       end;
end.
```

GetPixel функциясы координатасы берілген пикселдің түсі көрсетілген Word типті мәнді қайтарады. Функция тақырыбы :

Function GetPixel(X,Y: Integer): Word;

мұндағы X, Y – пиксел координатасы

Line процедурасы бастапқы және соңғы коодинаталары берілген түзу сызады. Процедура тақырыбы:

Procedure Line (X1,Y1,X2,Y2: Integer);

Мұндағы XI ... Yl – түзудің басының (XI, Y1) және соңының (X2, Y2) координаталары.

Түзу ағымдағы стиль және түспен салынады. Келесі программада экран ортасында терезе салынып, терезе кездейсоқ түзулермен толтырылады. Программадан шығу үшін кез келген пернені басу керек.

```
Uses CRT, Graph;
var
   d,r,e: Integer;
   x1,y1,x2,y2: Integer;
begin
Графиканы іске қосамыз}
   d:=Detect;InitGraph(d, r, '');
   e:=GraphResult;
   if e <> grOk then
      WriteLn(GraphErrorMsg(e))
   else
      begin
{Экран ортасында терезе саламыз}
         x1:=GetMaxX div 4;
         y1:=GetMaxY div 4;
         x2:=3*x1;
         y2:=3*y1;
         Rectangle(x1,y1,x2,y2);
         SetViewPort(x1+1,y1+1,x2-1,y2-
1,ClipOn);
{Кездейсоқ түзулер циклы}
         repeat
         SetColor(succ(Random(16)));{Кездейсоқ
түс}
            Line(Random(x2-x1), Random(y2-y1),
            Random(x2-x1), Random(y2-y1))
         until KeyPressed;
         if ReadKey=#0 then d:= ord(ReadKey);
         CloseGraph
      end
end.
```

LineTo процедурасы курсордың ағымдағы позициясынан бастап, берілген жаңа координатасына дейін түзу сызады. Процедура тақырыбы:

Procedure LineTo(X,Y: Integer);

мұндағы X, Y – курсордың жаңа координаталары, ол түзудің соңғы координатасы болып табылады.

LineRel процедурасы курсордың ағымдағы позициясынан, оның координаталарының берілген өсімшесі сәйкес түзу сызады. Процедура тақырыбы:

Procedure LineKel (DX, DY;integer);

мұндағы DX, DY – курсордың жаңа координатасының өсімшелері. LineTo және LineRel процедураларында түзу ағымдағы стиль және түспен салынады.

SetLineStyle процедурасы салынатын түзудің жаңа стилін анықтайды. Процедура тақырыбы:

Procedure SetLineStyle(type,Pattern,Thick: Word)

мұндағы type, Pattern, Thick – сәйкесінше, түзудің типі, үлгісі және қалыңдығы.

Түзу типін төмендегі тұрақтылардың көмегімен беруге болады:

```
const
  SolidLn= 0;{Біркелкі түзу}
  dottedLn= 1;{Нүктелі түзу}
  CenterLn= 2;{Штрих-пунктирлі түзу}
  DashedLn= 3;{Пунктирлі түзу}
  UserBitLn= 4;{Түзу түрін тұтынушы анықтайды}
```

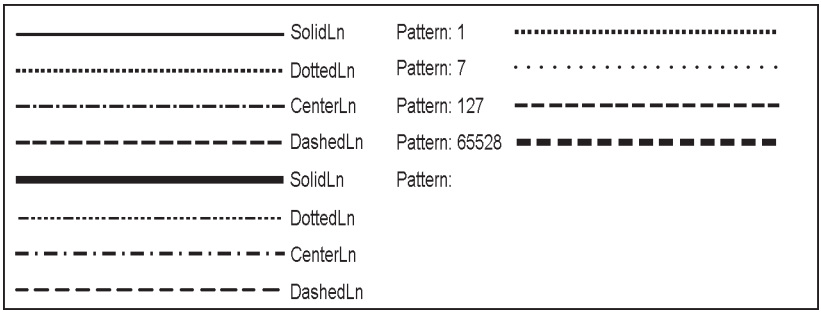
Pattern параметрі түрін тұтынушы анықтайтын түзулер үшін ғана қолданылады (тек Type = UserBitLn болған жағдайда). Pattern параметрінің екі байты түзу үлгісін анықтайды: сөздің әр бірге тең биті, түзудің жанып көрсетілетін пикселіне сәйкес келеді, нөлге тең бит – жанбайтын пикселге сәйкес болады. Сонымен, Pattern параметрі ұзындығы 16 пикселге тең кесінді салады. Осы үлгі түзудің ұзындығы бойымен қайталанып отырады.

Thick параметрі келесі екі мәннің бірін қабылдайды:

```
const
  NormWidth = 1;{Қалыңдығы бір пиксел}
  ThickWidth = 3;{ Қалыңдығы үш пиксел}
```

мұндағы процедура орнатқан түзу стилі (ағымдағы стиль) тікбұрыш, көпбұрыш және басқада фигураларды салғанда қолданылады.

Келесі мысалда барлық стандартты стилдермен түзулер салынады, сонан кейін экранда үлгі аты шығады да, экран осы үлгідегі түзумен толтырылады (7.5-сурет). Программадан шығу үшін нөлді енгізіңіз.



7.5-сурет. Түзу сызықтар түрлері

```
Uses CRT, Graph;
const
   style:array [0..4]of String[9]=('SolidLn',
'dottedLn','CenterLn','DashedLn','UserBitLn');
var
   d,r,e,i,j,dx,dy: Integer;
   p: Word;
begin
{Графиканы іске қосамыз}
   d:=Detect;InitGraph(d, r, '');
   e:=GraphResult;
   if e <> grOk then
      WriteLn (GraphErrorMsg(e))
   else
      begin
{Түзу ығысуын есептейміз}
         dx:=GetMaxX div 6;
         dy:=GetMaxY div 10;
         {стандартты түзулерді шығарамыз}
```

```
          for j:=0 to 1 do {Екі қалыңдық үшін}
             begin
           for i:=0 to 3 do {Түзутің төрт типі}
                   begin
            SetLineStyle(i, 0, j*2+1);
          Line(0,(i+j*4+1)*dy,dx,(i+j*4+1)*dy);
        OutTextXY(dx+10,(i+j*4+1)*dy,style [i])
                   end
              end;
{Үлгіні енгізіп, түзу саламыз}
          j:=0;
          dy:=(GetMaxY+1) div 25;
          repeat
             OutTextXY(320,j*dy,'Pattern:');
             GotoXY(50,j+1);
             ReadLn(p);
             if p<> 0 then
                begin
           SetLineStyle(UserBitLn,p,NormWidth);
                 Line(440,j*dy+4, 600, j*dy+4);
                   inc(j)
                end
          until p=0;
          CloseGraph;
       end
end.
```

GetLineSettings процедурасы. Түзудің ағымдағы стилін қайтарады. Процедура тақырыбы:

Procedure GetLineSettings(var Stylelnfo: LineSettingstype)

Мұндағы Stylelnfo – ағымдағы стилді қайтаратын, LineSettingstype типті айнымалы.

LineSettingstype типі Graph модулінде төмендегідей анықталған:

```
type
  LineSettingstype = record
  LineStyle: Word;{Түзу типі}
  Pattern : Word;{Үлгі}
```

Thickness: Word {Қалыңдығы}
end;

SetWriteMode процедурасы. Жағадан салынатын түзу мен экранда бұрын салынған түзулер арасындағы қатынас тәсілін орнатады. Процедура тақырыбы:

Procedure SetWriteMode(Mode);

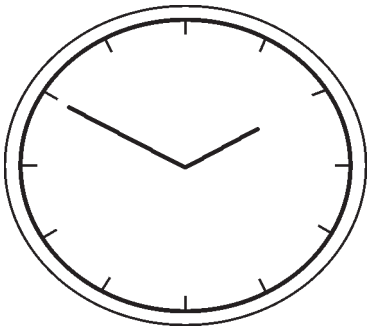
Мұндағы Mode – салынатын бейнелі түзулер арасындағы қатынас тәсілін анықтайтын, Integer типті өрнек.

Егер Mode параметрінің мәні 0 болса, онда салынатын түзу экранда бар түзудің үстіне салынады (орталық процессордың MOV нұсқауы). Егер мән 1 болса, онда жаңа түзу салу XOR логикалық операциясының көмегімен орындалады (ерекше ИЛИ): жаңа түзу мен экрандағы түзудің қилысу нүктелерінде пикселдің жануы инверсияланады, сондықтан бірінен соң бірі салынатын екі түзу экрандағы түзудің түсін өзгертпейді.

SetWriteMode процедурасы орнатқан режим, Drawpoly, Line, LineRel, LineTo және Rectangle процедураларына да таралады. Mode параметрін беру үшін модульде анықталған төмендегі тұрақтыларды қолдануға болады:

const
CopyPut = 0;{ MOV операциясымен салу}
XORPut = 1;{ XOR операциясымен салу }

Келесі мысалда, экранда сағат циферблаты салынады (7.6-сурет). Көрнекі болу үшін сағат жүрісі 600 есе жылдамдатылған (Delay (100) операторын қараңыз). Қалауыңызша программаны жүйелік сағатқа қосып және секунд стрелкасын салып,



7.6-сурет. Сағат циферблаты

күрделетуіңізге болады. Программадан шығу үшін кез келген пернені басыңыз.

```
Uses Graph, CRT;
var
   d,r,r1,r2,rr,k,
   x1,y1,x2,y2,x01,y01: Integer;
   Xasp,Yasp : Word;
begin
   {Графиканы іске қосамыз}
   d:=detect;InitGraph(d, r,'');
   k:=GraphResult;
   if k <> grOK then
      WriteLn(GraphErrorMSG(k))
   else
      begin
{Экран көлемі мен жақтаулар қатынасын
анықтаймыз}
         x1:=GetMaxX div 2;
         y1:=GetMaxY div 2;
         GetAspectRatio(Xasp, Yasp);
         {Радиусты есептейміз:}
         r:=round(3*GetMaxY*Yasp/8/Xasp);
         r1:=round(0.9*r);{Сағат бөліктері}
         r2:=round(0.95*r);{Минут бөліктері}
{Циферблатты бейнелейміз}
 Circle(x1,y1,r);{Бірінші сыртқы дөңгелек}
 Circle(x1,y1,round(1.02*r) );{Екінші
дөңгелек}
    for k := 0 to 59 do {Циферблат бөліктері}
        begin
          if k mod 5=0 then
          rr := r1 {Сағат бөліктері}
           else
           rr : = r2;{Минут бөліктері}
{Бөліктер соңының координатасын анықтаймыз}
     x01:=x1+Round(rr*sin(2*pi*k/60));
```

```
     y01:=y1-Round(rr*Xasp*cos(2*pi*k/60)/
Yasp);
     x2:=x1+Round(r*sin(2*pi*k/60));
     y2:=y1-Round(r*Xasp*cos(2*pi*k/60)/Yasp);
     Line(x01,y01,x2,y2) {Бөліктерді шығарамыз}
            end;
{Стрелкаларды шығаруға дайындаймыз}
      SetWriteMode(XORPut);
      SetLineStyle(SolidLn,0,ThickWidth);
{Бір сағаттағы минут санауышы}
{k = минуты}
      r:=0;
{Стрелкаларды шығару циклы}
      repeat
        for k:=0 to 59 do
          if not KeyPressed then begin
{Сағат стрелкаларының координаталары}
 x2:=x1+Round(0.85*r1*sin(2*pi*r/60/12));
 y2:=y1-Round(0.85*r1*Xasp*cos(2*pi*r/60/12)/
Yasp);
{Минут стрелкаларының координаталары }
          x01:=x1+Round(r2*sin(2*pi*k/60));
      y01:=y1-Round(r2*Xasp*cos(2*pi*k/60)/
Yasp);
{Стрелкаларды бейнелейміз}
       Line(x1,y1,x2,y2);
       Line(x1,y1,x01,y01);
       Delay(100);{Нақты жылдамдықты көрсету
үшін кешігу 60000 болуы керек}
{Стрелкаларды өшіру үшін оларды тағы да
шығарамыз!}
         Line(x1,y1,x01,y01);
         Line(x1,y1,x2,y2);
{Сағаттағы минут санауышын түзетіп, өсіреміз}
             inc(r);
             if r=12*60 then r:=0
               end
```

```
         until KeyPressed;
         if ReadKey=#0 then k:=ord(ReadKey);
      CloseGraph
   end
end.
```

Көпбұрыштар

Rectangle процедурасы. Бұрыштарының координатасы бойынша тікбұрыш салпды. Процедура тақырыбы:

Procedure Rectangle(X1,Y1,X2,Y2: Integer);

мұндағы X1... Y2 – тікбұрыштың сол жақ жоғарғы бұрышының (X1, Y1) және оң жақ төменгі бұрышының координатасы (X2, Y2). Тікбұрыш ағымдағы түс және түзу стилімен салынады.

Келесі мысалда экранда бірінің ішіне бірі кірістірілген 10 тікбұрыш салынады.

```
Uses Graph, CRT;
var
   d,r,e,xl,yl, x2,y2,dx,dy: Integer;
begin
 {Графиканы іске қосамыз}
   d:=Detect; InitGraph(d, r, ' ');
   e:=GraphResult;
   if e <> grOK then
      WriteLn(GraphErrorMsg(e))
   else
      begin
{Жақтаулардың өсімшесін анықтаймыз}
         dx:=GetMaxX div 20;
         dy:=GetMaxY div 20;
{Кіріктірілген тікбұрыштар саламыз}
         for d:=0 to 9 do
    Rectangle(d*dx,d*dy,GetMaxX-d*dx,GetMaxY-
d*dy);
       if ReadKey=#0 then d:=ord(ReadKey);
         CloseGraph
      end
end.
```

DrawPoly процедурасы сыну нүктесінің координаталары бойынша, еркін сынық сызық салады.

Procedure DrawPoly(N: Word; var Points)

мұндағы N – екі шеткі нүктемен бірге алғандағы сыну нүктелерінің саны; Points – сыну нүктелерінің координатасы көрсетілген, Pointtype типті айнымалы.

Сыну нүктелерінің координатасы Word типті екі мән арқылы беріледі: біріншісі көлденең, екіншісі тік координатасы. Олар үшін модульде анықталған келесі типті қолдануға болады:

```
type
   Pointtype=record
   x, y: Word
end;
```

Сызу кезінде ағымдағы түс пен түзу стилі қолданылады. Осы процедураның көмегімен экранға синустың графигін саламыз:

```
Uses Graph;
const
   N=100;{График нүктелерінің саны}
var
   d, r, e: Integer;
   m: array [O..N+1] of Pointtype;k : Word;
begin
 {Графиканы іске қосамыз}
   d:=Detect;InitGraph(d, r, '');
   e:=GraphResult;
   if e <> grOk then
      WriteLn(GraphErrorMsg(e))
   else
      begin
         {График координаталарын есептейміз}
         for k:=0 to N do with m[k] do
            begin
               x:=trunc(k*GetMaxX/N);
y:=trunc(GetMaxY*(-sin(2*Pi*k/N)+1)/2);
            end;
{Графикті түзу сызықпен аяқтаймыз}
```

```
            m[succ(N)].x:=m[0].x;
            m[succ(n)].y:=m[0].y;
            DrawPoly(N + 2, m);
            ReadLn;
            CloseGraph
         end
end.
```

Бұл мысалда, көлденең сызық сызу үшін сынық сызықтың бастапқы және соңғы нүктелерін біріктіреміз. Сыну нүктелерінің саны N Word типті өрнек болғанымен, процедура ішінде бұл параметрге қолданылатын буфер жадысының көлеміне байланысты шектеулер қойылады. Сіз бұны алдыңғы мысалдағы N өзгерту арқылы байқауыңызға болады: егер N=678 болса, онда график экранға шықпайды, ал GraphResult функциясына – 6 мәні қайтарылады (аудандарды көру үшін жады көлемі жетпейді). Сонымен, бұл программа үшін сыну нүктелерінің саны 679 аспауы кеек. Алайда, төмендегі программа үшін сыну нүктелерінің саны 510. Бұл программада сынық сызық бірінің үстіне бірі бірнеше рет сызылатын диагональ түзулер түрінде берілген.

```
Uses Graph;
const
   N=510;{Экрандағы диагональ түзулер көрініп
тұратын шекті мән }
var
   d,k: Integer;
   Coo: array [1..N] of Pointtype;
begin
   d:=Detect;InitGraph(d,k,' ') ;
   for k:=1 to N do with Coo[k] do
      if odd(k) then
         begin
            X:=0;
            Y:=0
         end
      else
         begin
```

```
            X:=GetMaxX;
            Y:=GetMaxY
         end;
   DrawPoly(N,Coo);
   ReadLn;
   CloseGraph
end.
```

Доғалар, дөңгелектер, эллипстер

Circle процедурасы дөңгелек сызады. Тақырыбы:

Procedure Circle(X,Y: Integer;R: Word);

мұндағы X, Y – центрдің координатасы; R – пикселмен берілген радиус.

Дөңгелек ағымдағы түспен сызылады. Сызықтың қалыңдығы ағымдағы стилмен анықталады, ал түрі әрқашан SolidLn (біркелкі) болып келеді. Процедура, радиустың графикалық экран жақтауларына бағытталуына байланысты сызықтық көлемінің өзгеруін, демек, GetAspectRatio коэффициентін есепке ала отырып, дұрыс дөңгелек сызады. Сондықтан R параметрі көлденең бағыттағы пикселдер санын анықтайды.

Келесі мысалда экран ортасында терезе салынып, ол біртіндеп кездейсоқ дөңгелектермен толтырылады. Программадан шығу үшін кез келген пернеге басыңыз.

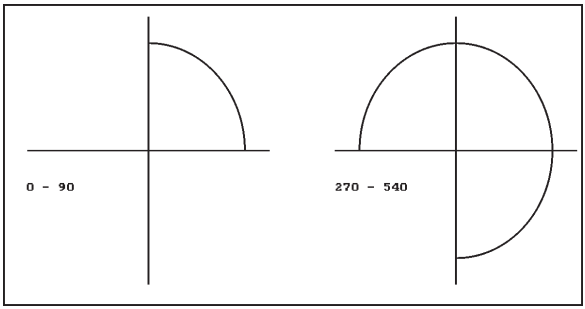
```
   Uses Graph, CRT;
var
   d,r,e,x,y: Integer;
begin.
{Графиканы іске қосамыз}
   d:=Detect;InitGraph(d,r,'');
   e:=GraphResult;
   if e <> grOK then
      WriteLn(GraphErrorMsg(e))
   else
      begin
{Экран ортасында терезе саламыз}
         x:=GetMaxX div 4;
```

```
          y:=GetMaxY div 4;
          Rectangle(x,y,3*x,3*y);
          SetViewPort(x+1,y+1,3*x-1,3*y-
1,ClipOn);
{кездейсоқ дөңгелектер сызу циклы}
          repeat
     SetColor(succ(Random(white)));{Кездейсоқ
түс}
 SetLineStyle(0,0,2*Random(2)+1);{және сызық
стилі}
   x:=Random(GetMaxX);{Кездейсоқ орналасу}
   y:=Random(GetMaxY);{Дөңгелек центрі}
     Circle(x,y,Random(GetMaxY div 4));
          until KeyPressed;
          if ReadKey=#0 then x:=ord(ReadKey);
          CloseGraph
      end
end.
```

Arc процедурасы доға сызады. Тақырыбы:

Procedure Arc (x,y: integer; BegA, endA,R: word);

мұндағы X, Y – центр координатасы; BegA, endA – сәйкесінше, доғаның бастапқы және соңғы бұрыштары; R – радиус.



7.7-сурет. Arc процедурасын пайдалану

Бұрыштар сағат жүрісіне қарсы саналып, градуспен көрсетіледі. Нөл бұрыш вектордың көлденең, солдан оңға қарайғы бағытына сәйкес келеді. Егер бастапқы бұрышты 0 және соңғы бұрышты – 359 градус деп берсе, онда толық дөңгелек сызылады.

Дөңгелек доғасын сызғанда, сызықтар мен радиус үшін Circle процедурасындағы қатынастар қолданылады.

Бріншісінің бұрыштары 0 және 90, екіншісінің – 270 және 540 градус доғалардың көрнісі (7.7-сурет):

Келесі программа осы бейнені экранға шығарады:

```
Uses Graph, CRT;
var
   d,r,e: Integer;
   Xasp,Yasp: Word;
begin
{Графиканы іске қосамыз}
   d:=Detect;
   InitGraphtd, r, '');
   e:=GraphResult;
   if e <> grOK then
      WriteLn(GraphErrorMsg(e))
   else
      begin
         GetAspectRatio(Xasp,Yasp);
{R = 1/5 экранның тік көлемінен}
         r:=round(Yasp*GetMaxY/5/XAsp);
         d:=GetMaxX div 2;{Екінші графиктің
ығысуы}
         e:= GetMaxY div 2;{Көлденең осьтің
орны}
{Сол жақ графикті саламыз}
     Line(0,e,5*r div 2,e);{ Көлденең ось}
     Line(5*r div 4,e div 2,5*r div 4,3*e div 2) ;
      Arc (5*r div 4,e,0,90,R);{Доға}
      OutTextXY(0,e+e div 8,'0 - 90');{Жазу}
{Оң жақ график}
    Line(d,e,d+5*r div 2,e);
   Line(d+5*r div 4,e div 2,d+5*rdiv4,3*e div 2);
         Arc (d+5*r div 4,e,270,540,R);
         OutTextXY(d,e+e div 8,'270 - 540');
{Кез келген перненің басылуын күтеміз}
```

```
          if ReadKey=#0 then d:=ord(ReadKey);
             CloseGraph
       end
end.
```

GetArcCoords процедурасы доғаның бастапқы, соңғы және центрінің координаталарын қайтарады. Тақырыбы:

Procedure GetArcCoords(**var** Coords: ArcCoordstype);

мұндағы Coords – процедура доғаның бастапқы, соңғы және центрінің координаталарын қайтаратын, ArcCoordstype типті айнымалы.

ArcCoordstype типі Graph модулінде төмендегідей анықталған:

```
type
 ArcCoordstype=record
    X,Y: Integer;{Центр координатасы}
    Xstart,Ystart: Integer;{Доғаның басы}
    Xend,Yend: Integer;{Доғаның соңы}
 end;
```

Arc және GetArcCoords процедураларын бірге қолдану, екі түзудің доға арқылы түйіндесуін сызуға мүмкіндік береді. Келесі бұрыштары дөңгеленген тікбұрыш сызатын мысалда, радиус ұзындығының түзетілуіне назар аударыңыз.

```
Uses Graph,CRT;
const
   RadX=50;{Көлденең радиус}
   lx=400;{Ені}
   ly=100;{Биіктігі}
var
   d,r,e: Integer;
   coo: ArcCoordstype;
   x1,y1: Integer;
   xa,ya: Word;
   RadY: Integer;{Тік радиус}
begin
{Графиканы іске қосамыз}
   d:=Detect;InitGraph(d,r,'') ;
```

```
   e:=GraphResult;
   if e <> grOK then
      WriteLn(GraphErrorMsg(e))
   else
      begin
 GetAspectRatio(xa,ya);{Жақтаулар қатынасын
аламыз}
{Тік радиус пен экран жақтауларына қатысты
фигураның орналасуын есептейміз}
       RadY:=round (RadX *(xa /ya));
       x1:=(GetMaxX-lx) div 2;
       y1:=(GetMaxY-2*RadY-ly) div 2;
{Фигураны сызамыз}
  Line(x1,y1,x1+lx,y1);{Жоғарғы көлденең түзу}
   Arc (x1+lx,y1+RadY,0,90,RadX);{Дөңгелектеу}
         GetArcCoords(coo);
         with coo do
            begin
         Line(Xstart,Ystart,Xstart,Ystart+ly);
{Оң жақ тік түзу}
             Arc(Xstart-
RadX,Ystart+ly,270,0,RadX);
               GetArcCoords (coo);
        Line(Xstart,Ystart,Xstart-lx,Ystart);
{Төменгі көлденең түзу}
        Arc(Xstart-lx,Ystart-
RadY,180,270,RadX);
        GetArcCoords(coo);
        Line(Xstart,Ystart,Xstart,Ystart-ly);
        Arc(Xstart+RadX,Ystart-ly,90,180,RadX)
           end;
         if ReadKey=#0 then d:=ord(ReadKey);
         CloseGraph
      end
end.
```

Ellipse процедурасы эллипстік доға сызады. Тақырыбы:

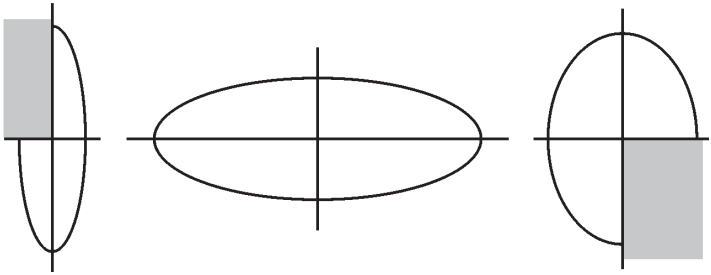
Procedure Ellipse(X,Y: Integer;BegA,endA,RX,RY: Word);

мұндағы X, Y – центр координатасы; BegA, endA – сәйкесінше, доғаның бастапқы және соңғы бұрыштары; RX, RY – эллипстің пикселмен берілген көлденең және тік радиусы.

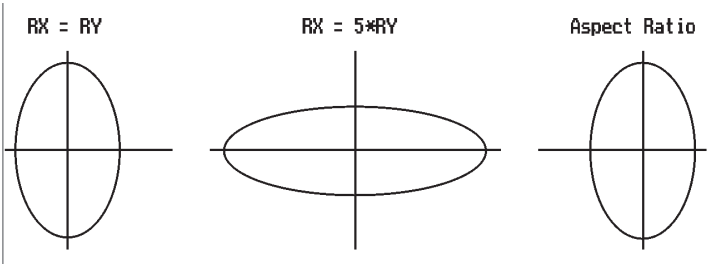
Эллипстік доғаны сызғанда, Circle процедурасындағы түзуге қатысты келісімдер мен Arc процедурасындағы бұрышқа қатысты келісімдер қолданылады. Егер радиустарды CetAspectRatio масштабтық коэффициентті ескере отырып алсақ, дұрыс дөңгелек сызылады.

Келесі программада радиус қатынастары әр түрлі үш эллипстік доға сызылады (7.8-сурет). Экран мүмкіндігі неғұрлым жоғары болса, соғұрлым жақтаулар арасындағы қатынас бірге жақын және бірінші графиктің үшінші графиктен айырмашылығы аз болады.

```
Uses Graph, CRT;
```



7.8-сурет. Эллипстік доғалар




```
var
   d,r,e: Integer;
   xa,ya: Word;
begin
{Графиканы іске қосамыз}
```

```
   d:=Detect;InitGraph(d,r,'');
   e:=GraphResult;
   if e <> grOK then
      WriteLn(GraphErrorMsg(e))
   else
      begin
{Бірінші график}
         OutTextXY(5 0,4 0,'RX = RY');{Жазу}
         Line (0,100,160,100);{Ось X}
         Line (80,55,80,145);{Ось Y}
         Ellipse (80,100,180,90,40,40);
         {Екінші график}
         OutTextXY(260,40,'RX = 5*RY');
         Line (190,100,410,100);
         Line (300,55,300,145);
         Ellipse (300,100,0,359,100,20);
         {Үшінші график}
         OutTextXY(465,40,'Aspect Ratio');
         Line (440,100,600,100);
         Line (520,55,520,145);
         GetAspectRatio(xa, ya);
    Ellipse (520,100,0,270,40,round(40*(xa/
ya)));
         if ReadKey=#0 then
            d:=ord(ReadKey);
         CloseGraph
      end
end.
```

7.4 Түстерді басқару

SetColor процедурасы шығарылатын түзулер мен символдардың ағымдағы түсін тағайындайды. Тақырыбы:

Procedure SetColor(Color: Word);

мұндағы Color – ағымдағы түс.

Graph модулінде, түстерді беру үшін дәл CRT модуліндегідей тұрақтылар анықталған.

GetColor функциясы ағымдағы түс коды жазылған Word типті мәнді қайтарады. Тақырыбы:

Function GetColor: Word ;

GetMaxColor функциясы SetColor процедурасында қолдануға болатын ең жоғарғы түс коды жазылған Word типті мәнді қайтарады. Тақырыбы:

Function GetMaxColor: Word;

SetBkColor процедурасы. Фон түсін орнатады. Тақырыбы:

Procedure SetBkColor(Color: Word)

мұндағы Color – фон түсі.

Фоны тек қара түсті бола алатын мәтіндік режимге қарағанда, графикалық режимде фон кез келген түсте болуы мүмкін. Жаңа фон түсін орнату, бірден графикалық экран түсін өзгертеді. Бұл бір бейненің екі бөлігінің фон түсі екі түрлі болмайтындығын білдіреді. CGA – адаптері үшін, жоғары мүмкіндікті режимде фон түсін орнату, екпінді пикселдер түсін өзгертеді. Сіз фон түсін 0 (*Black*) өзгеше кез келген түске ауыстырғаннан кейін, 0 түсті қара түс ретінде қолдана алмайсыз. Себебі, Graph модулі 0 түсті фон түсі ретінде қабылдайды да, 0 түсті фон түсіне алмастырылады. Сонымен, егер сіз фон түсін қарадан басқа түске ауыстырсаңыз, бастапқы қара түсті қайтара алмайсыз!

Егер сіздің компьютеріңіз түсті экранмен жабдықталған болса, онда келесі программа SetBkColor процедурасының жұмысын көрнекі түрде көрсетеді. Программа бірінің ішіне бірі салынған он тікбұрыштар салады да, фонның түсін ауыстырады. Программадан шығу үшін кез келген пернеге басыңыз.

```
uses Graph, CRT;
const
   NC:array [0..15]of String[12]=
('Black','Blue','Green','Cyan','Red','Magenta',
  'Brown','LightGray','DarkGray','LightBlue',
  'LightGreen1,'LightCyan1,'LightRed',
  'LightMagenta','Yellow','White');
var
   d,r,e,k,color,dx,dy: Integer;
begin
{Графиканы іске қосамыз}
   d:=Detect; InitGraph(d,r,'');
```

```
   e:=GraphResult;
   if e <> grOK then
      WriteLn(GraphErrorMsg(e))
   else
      begin
 {Экран ортасына мәтін шығарамыз}
   OutTextXY(200,GetMaxY div 2,'BACKGROUND
COLOR');
   dx:=GetMaxX div 30; {Ұзындықтың өсімшесі}
   dy:=GetMaxY div 25; {Биіктіктің өсімшесі}
     for k:=0 to 9 do{ 10 тікбұрыш шығарамыз}
Rectangle(k*dx,k*dy,GetMaxX-k*dx,GetMaxY-
k*dy);
      color:=black; {Фонның бастапқы түсі}
         repeat {Фонды өзгерту циклі}
      SetBkColor(color);
      SetBkColor(color);
      SetFillStyle(0,Color);
      SetBkColor(color);
      Bar(345,GetMaxY div 2,440,GetMaxY div
2+8);
      SetBkColor(color);
      OutTextXY(345,GetMaxY div 2,NC[color]);
      SetBkColor(color);
      delay(1000);
      SetBkColor(color);
      inc(color);
      if color > White then
      color:=Black
      until KeyPressed;
      if ReadKey=#0 then
         k:=ord(ReadKey);
         CloseGraph
      end
end.
```

GetBkColor функциясы ағымдағы фон түсі жазылған, Word типті мәнді қайтарады. Тақырыбы:

Function GetBkColor: Word;

SetPalette процедурасы палитраның бір түсін жаңа түске ауыстырады. Тақырыбы:

Procedure SetPalette (N: Word; Color: Shortlnt);

Мұндағы N – палитрадағы түс нөмірі; Color – жаңа орнатылатын түс нөмірі.

Бұл процедура тек *EGA* немесе *VGA* адаптерлерімен жұмыс жасай алады. Ол IBM 8514 немесе 256 – түсті *VGA* адаптерінде қолданылмайды, бұл адаптерлер үшін ерекше *SetRGBPatette* процедурасы бар. EGA/VGA адаптерлеріне палитраның түстерінің бастапқы орналасуы олардың тұрақтылармен сипатталу ретіне сәйкес келеді: 0 – қара, 1 – көк, 2 – жасыл және т.с.с. Процедура іске қосылғаннан кейін бейненің, палитра түстерінің N индексті түсімен салынған фрагменттері Color түсіне ие болады. Мысалы, SetPalette (2, White) операторы орындалғаннан кейін, 2 индексті түс (бастапқыда бұл – күлгін түс, Cyan) ақ түске ауыстырылады. Нөмірі 0 түс, фон түсі болып саналады және ол да кез келген түс сияқты өзгере алады.

Келесі программа экранға түрлі түсті түзулерді шығарып, олардың түсін кездейсоқ түске ауыстырады

```
uses Graph, CRT;
var
   d,r,e,N,k,color: Integer;
   Palette: PaletteTyper;
begin
{Графиканы іске қосамыз}
   d:=Detect; InitGraph(d,r,'');
   e:=GraphResult;
   if e <> grOK then
      WriteLn(GraphErrorMsg(e))
   else
      begin
{Қалың, біркелкі сызықтарды таңдаймыз}
       SetLineStyle(SolidLn, 0, ThickWidth);
       GetPalette(Palette); {Ағымдағы палитра}
       for Color:=0 to Palette.Size-1 do
           begin
            SetColor(Color);
```

```
       Line(GetMaxX div 3,Color*10,2*GetMaxX
div 3,Color*10)
            end;
  {Палитраны өзгертіп, тұтынушы әрекетін
күтеміз}
         while not KeyPressed do
            for e:=0 to Palette.Size-1 do
              SetPalette(e,Random(Palette.
Size));
         if ReadKey=#0 then d := ord(ReadKey);
         CloseGraph
      end
end.
```

GetPalette процедурасы ағымдағы палитраның көлемі мен түсін қайтарады. Тақырыбы:

Procedure GetPalette (**var** Palettelnfo: PaletteType)

мұндағы Palettelnfo – палитраның көлемі мен түсін қайтаратын, PaletteType типті айнымалы.

Graph модулінде келесі тұрақты анықталған.

const

MaxColors =15;

және оның типі де анықталған:

type

PaletteType = record

Size : Word; *{Палитрадағы түстер саны}*

Colors : array [0..MaxColors] of Shortlnt

{Палитраға кіретін түстер нөмірі }

end;

Келесі программа көмегімен экранға ағымдағы палитраның барлық түстерінің нөмірін шығаруға болады.

```
uses Graph;
var
   Palette: PaletteType;
   d,r,e,k: Integer;
begin
{Графиканы іске қосамыз}
```

```
   d:=Detect; InitGraph(d,r,'');
   e:=GraphResult;
   if e <> grOk then
      WriteLn(GraphErrorMsg(e))
   else
      begin
         GetPalette(Palette); {Палитраны
аламыз}
         CloseGraph; {Мәтіндік режимге
ораламыз}
         with Palette do {Түстер нөмірін
шығарамыз}
            for k:=0 to pred(Size) do
               Write(Colors[k]:5);
      end
end.
```

SetAllPalette процедурасы палитраның бірнеше түстерін бір мезгілде өзгертеді. Процедура тақырыбы:

Procedure SetAllPalette (**var** Palette);

Palette параметрі процедура тақырыбында типтік емес параметр ретінде сипатталған. Бұл параметрдің бірінші байтында N палитра ұзындығы, қалған N байттар – (–1) мен *MaxColors* аралығындағы жаңадан орнатылатын түстер нөмірі. (–1) коды бастапқы палитраның сәйкес түсі өзгермейтіндігін білдіреді.

Келесі программада палитраның барлық түстері бір мезгілде өзгереді.

```
uses Graph, CRT;
var
   Palette: array [0..MaxColors] of Shortint;
   d,r,e,k: Integer;
begin
{Графиканы іске қосамыз}
   d:=Detect; InitGraph(d,r,'');
   e:=GraphResult;
   if e <> grOk then
      WriteLn(GraphErrorMsg(e))
```

```
   else
      begin
{Қалың, біркелкі сызықтарды таңдаймыз}
         SetLineStyle(SolidLn, 0, ThickWidth);
{Сызықтарды қол жеткізуге болатын барлық
түстермен шығарамыз}
     for k:=1 to GetMaxColor do
      begin
      SetColor(k);
     Line(GetMaxX div 3,k*10,2*GetMaxX div
3,k*10)
           end;
      Palette[0]:=MaxColors; {Палитра көлемі}
      repeat {Палитраны ауыстыру цикл}
       for k:=1 to MaxColors do
       Palette[k]:=Random(succ(MaxCoLors));
            SetAllPalette(Palette)
         until KeyPressed;
         if ReadKey=#0 then k:=ord(ReadKey);
         CloseGraph
      end
end.
```

GetPaletteSize функциясы палитра көлемі (қол жеткізуге болатын түстердің максимальді саны) жаылған Integer типті мәнді қайтарады. Тақырыбы:

Function GetPaletteSize: Integer;

GetDefaultPalette процедурасы үнсіз келісім бойынша орнатылатын палитра құрылымын қайтарады (автобаптау режимінде). Тақырыбы:

Procedure GetDefaultPalette(**var** Palette: PaletteType);

мұндағы *Palette* –палитра көлемі мен түстері қайтарылатын *PaletteType* типті айнымалы.

SjetFillStyle процедурасы толтыру стилін орнатады (тип және түс) заполнения. Тақырыбы:

Procedure SetFillStyle (Fill,Color: Word);

мұндағы *Fill* – толтыру типі; *Color* – толтыру түсі.

Толтыру арқылы бейненің қандай да бір фрагментін өрнектермен қайталап салып отыруға болады. Толтыру типін көрсеті үшін келесі алдын ала анықталған тұрақтылар қолданылады:

const

EmptyFill = 0; *{Фон түсімен толтыру (өрнек жоқ)}*
SolidFill = 1; *{біркелкі толтыру}*
LineFill = 2; *{ ------- толтыру}*
LtSlashFill = 3; *{ /////// толтыру}*
SlashFill = 4; *{Қалыңдалған /// толтыру}*
BkSlashFill = 5; *{ Қалыңдалған \\\ толтыру}*
LtBkSlashFill = 6; *{ \\\\\\\ толтыру}*
HatchFill = 7; *{ +++++++ толтыру}*
XHatchFill = 8; *{ xxxxxxx толтыру}*
InterleaveFill= 9; *{Тікбұрышты клеткалармен толтыру}*
WideDotFill = 10; *{Сирек нүктелермен толтыру}*
CloseDotFill = 11; *{ Жиі нүктелермен толтыру }*
UserFill = 12; *{Өрнек түрін тұтынушы анықтайды}*

Келесі мысал программасы сіздерге стандартты толтыру типтерін көрсетеді.

```
uses Graph, CRT;
var
   d,r,e,k,j,x,y: Integer;
begin
{Графиканы іске қосамыз}
   d:=Detect; InitGraph(d,r,'');
   e:=GraphResult;
   if e <> grOk then
      WriteLn(GraphErrorMsg(e))
   else
      begin
        x:=GetMaxX div 6;{Графиктің экрандағы
орны}
         y:=GetMaxY div 5;
         for j:=0 to 2 do{Екі қатар}
            for k:=0 to 3 do{төрт квадраттан}
               begin
   Rectangle((k+1)*x,(j+1)*y,(k+2)*x,(j+2)*y);
       SetFillStyle(k+j*4,j+1);
```

```
      Bar((k+1)*x+1,(j+1)*y+1,(k+2)*x-
1,(j+2)*y-1);
                end;
          if ReadKey=#0 then k:=ord(ReadKey);
          CloseGraph
      end
end.
```

Егер *Fill* параметрінің мәні 12 *(UserFill*) болса, онда өрнек суретін программист *SetFillPattern* процедурасын шақыру арқылы өзі анықтайды.

SetFillPattern процедурасы сурет үлгісін және штрихтау түсін анықтайды. Тақырыбы:

Procedure SetFillPattern(Pattern: FillPatternType; Color: Word);

мұндағы *Pattern – SetFillStyle* процедурасында *Fill = UserFill* үшін сурет үлгісін анықтайтын *FillPatternType* типті өрнек*; Color* – толтыру түсі.

Сурет үлгісі 8х8 пикселден тұратын матрица түрінде беріледі және төмендегідей типтегі 8 байт жиым түрінде де берілуі мүмкін:

```
type
  FillPatternType = array [1..8] of Byte;
```

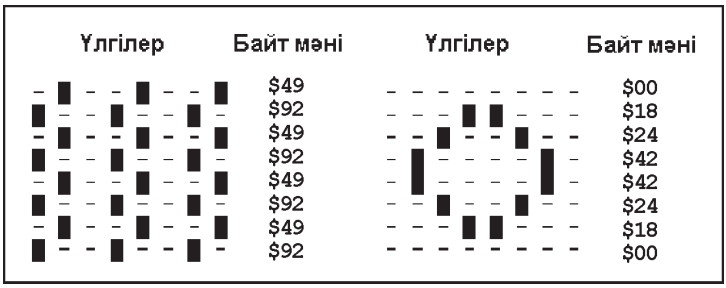
Осы байттардың кез келген разрядтары пикселдің жануын басқарады. Бірінші байт экранның бірінші жолының 8 пикселін, екінші байт – екінші жолдың 8 пикселін және т.с.с.

7.9-суретте толтырудың екі үлгісінің мысалы келтірілген. Суретте сызықшамен жанбайтын пиксел, ал тіктөртбұрышпен жанатын пиксел белгіленген. Әр 8 пикселге сәйкес байттың оналтылық коды келтіріледі.

Келесі программа осы үлгімен экранның екі тікбұрышты ауданын толтырады.

```
uses Graph, CRT;
const
pattl: FillPattern-
Type=($49,$92,$49,$92,$49,$92,$49,$92);
patt2: FillPattern-
Type=($00,$18,$24,$42,$42,$24,$18,$00);
var
```

```
   d,r,e: Integer;
begin
{Графиканы іске қосамыз}
   d:=Detect; InitGraph(d,r,'');
   e:=GraphResult;
   if e <> grOk then
      WriteLn(GraphErrorMsg(e))
   else
      begin
         if d=CGA then SetGraphMode (0);
{CGA адаптері үшін түстерді орнатамыз}
         SetFillStyle(UserFill,White);
{Сол жақ жоғарғы квадрат}
        SetFillPattern(Patt1,1);
        Bar(0,0,GetMaxX div 2, GetMaxY div 2);
{Оң жақ төменгі квадрат}
         SetFillPattern(Patt2,2);
Bar(GetMaxX div 2,GetMaxY div
2,GetMaxX,GetMaxY);
         if ReadKey=#0 then d:=ord(ReadKey);
         CloseGraph
      end
end.
```



7.9-сурет. Толтыру үлгілері және олардың кодтары

Егер процедураны шақырғанда түс коды дұрыс көрсетілмесе, онда процедура іске қосылмайды да, осыған дейін орнатылған толтыру үлгісі сақталады. Алдыңғы мысалда, *CGA* –адаптерінің жұмыс режимін орнататын

if d=CGA **then** SetGraphMode (o);

алып тастасақ, осы адаптермен жабдықталған компьютер экранына бірдей екі тікбұрыш салынады. Себебі,

SetFUlPattern (patt2,2);

операторында ағымдағы режим үшін мүмкін емес түс коды жазылған, сондықтан оператор орындалмайды. Бұл айтылғандар *SetFillStyle* процедурасына қатысты емес. Бұл процедураның *Fill* параметрі 0 мен 11 диапазонындағы мәндерді қабылдайды: программа *CGA* – адаптерінің жоғарғы режимінде де дұрыс жұмыс істейтін болады, фон түсінен басқа палитра түстері ақ түске ауыстырылады.

GetFillPattern процедурасы *SetFillPattern* процедурасы орнатқан толтыру үлгісін қайтарады. Тақырыбы:

Procedure GetFillPattern(**var** Pattern: FillPatternType);

мұндағы *Pattern* – толтыру үлгісі қайтарылатын, *FillPattemType* типті айнымалы.

Егер программа үлгіні *SetFillPattern* процедурасының көмегімен орнатпаса, Pattern жиымы мәні 255 ($FF) байттармен толтырылады.

GetFillSettinqs процедурасы ағымдағы толтыру стилін қайтарады. Тақырыбы:

Procedure GetFillSettinga(**var** Pattlnfo: FillSettingsType);

мұндағы *Pattlnfo* – ағымдағы толтыру стилі қайтарылатын, *FillSettingsType* типті айнымалы.

Graph модулінде келесі тип анықталған:

```
type
  FillSettingsType = record
  Pattern: Word; {Үлгі}
  Color : Word {Түс}
end;
```

Бұл жазбадағы Pattern және Color өрістерінің мәні, *SetFillStyle* процедурасын шақырғандағы сәйкес параметрлердегідей.

SetRGBPalette процедурасы *IBM 8514 VGA* адаптерімен жұмыс жасайтын түстер гаммасын тағайындайды. Тақырыбы:

Procedure SetRGBPalatte(ColNum,RedVal, GraenVal,BlueVal: Integer);

мұндағы *ColNum* – түс нөмірі; *RedVal, GreenVal, BlueVal* –

сәйкесінше, қызыл, жасыл және көк түстердің қоюлығын анықтайтын Integer типті өрнек.

Келесі программада экран ортасына ақ түсті тікбұрыш салынады да, ол *SetRGBPalette* процедурасының көмегімен кездейсоқ түстерге боялады. Программадан шығу үшін кез келген пернеге басыңыз.

```
uses Graph,CRT;
var
   Driver,Mode,Err,xl,yl: Integer;
begin
{Графиканы іске қосамыз}
   Driver:=Detect;
   InitGraph(Driver,Mode,'');
   Err:=GraphResult;
   if ErroO then
      WriteLn(GraphErrorMsg(Err))
   else if Driver in [IBM8514, VGA] then
        begin
 {Экран ортасына тікбұрыш саламыз}
          x1:=GetMaxX div 4;
          y1:=GetMaxY div 4;
          SetColor(lS);
          Bar(x1,y1,3*x1,3*y1);
 {Ақ түсті кездейсоқ түске ауыстырамыз}
           while not KeyPressed do
          SetRGBPalette(15,Random(256),Random(
256),Random(256));
            CloseGraph
         end
      else
         begin
        CloseGraph;.
      WriteLn('Адаптер түстерді
басқарудың','RGB-режим қолдамайды');
    end
end.
```

FloodFill процедурасы ағымдағы толтыру стилін (өрнек және түс) қолданып, тұйық фигураның ішін бояйды. Тақырыбы:

Procedure FloodFill(X,Y: Integer; Border: Word);

мұндағы *X, Y* – тұйық фигураның ішіндегі кез келген нүкте координатасы; *Border* – шекара сызығының түсі.

Егер фигура тұйық болмаса, онда бүкіл экран боялады.

Процедурадағы тұйық фигураның шекарасын қарау алгоритмінің жетік емес екендігін есте сақтау керек. Егер қатарынан екі бос жол шығарылатын болса, онда бояу тоқтатылады. Мұндай жағдай көбіне *LtSlashFill* типін қолданып, кішігірім фигураны бояғанда пайда болады. Турбо Паскаль тілінің фирмалық нұсқауларында *FloodFill* процедурасының орнына, мүмкіндігінше, *FillPoly* процедурасын қолдануға кеңес берілген.

Келесі программа кездейсоқ дөңгелектер ішін бояуды көрсетеді. Алдымен экранда терезе салынып, оның ішінде тікбұрышты бояйды. Тікбұрыштың жартысы боялмай қалады, себебі программа жұмысын тоқтатып, Enter пернесінің басылуын күтеді. Сонан кейін, экранға кездейсоқ дөңгелектер салынып, кез келген перне басылғанша боялады. Егер *LtSlashFill* типінің орнына *SlashFill* типі қолданылса, тікбұрыш толығымен боялады. Егер программа өте ұзақ жұмыс істейтін болса, ол тұрып қалуы мүмкін, бұл көрсетілген алгоритмнің жетік еместігін дәлелдейді.

```
uses Graph, CRT;
var
   d,r,e,x,y,c: Integer;
begin
 {Графиканы іске қосамыз}
   d:=Detect; InitGraph(d,r,'');
   e:=GraphResult;
   if e <> grOk then WriteLn(GraphErrorMsg(e))
   else
      begin
 {Терезе саламыз}
         x:=GetMaxX div 4;
         y:=GetMaxY div 4;
         Rectangle(x,y,3*x,3*y);
         SetViewPort(x+1,y+1, 3*x-1,3*y-
1,ClipOn);
         {Кішкене тікбұрыштың боялуын
көрсетеміз }
```

```
        SetPillStyle(LtSlashFill,GetMaxColor);
         Rectangle(0,0,8,20);
         FloodFill(1,1,GetMaxColor);
         OutTextXY(10,25,'Press Enter...');
      ReadLn; { Enter пернесінің басылуын
күтеміз}
 {Кез келген перне басылғанша дөңгелектер са-
ламыз}
         repeat
 {Кездейсоқ бояу стилін анықтаймыз}
    SetFillStyle(Random(12),Random(GetMaxCol
or+1));
{Дөңгелектің центрінің координатасы мен түсін
береміз}
       x:=Random (GetMaxX div 2);
        y:=Random (GetMaxY div 2);
        c:=Random (succ(GetMaxColor));
         SetColor(c);
  {Дөңгелекті шығарып, бояймыз}
       Circle(x, y,Random(GetMaxY div 5));
       FloodFill (x,y,c);
       until KeyPressed;
       if ReadKey=#0 then
       x:=ord(ReadKey);
       CloseGraph;
      end
end.
```

Bar процедурасы экранның тікбұрышты аймағын бояйды. Тақырыбы:

Procedure Bar (X1,Y1,X2,Y2: Integer);

мұндағы *X1...Y2* – боялатын ауданның сол жақ жоғарғы (*X1, Y1*) және оң жақ төменгі (*X2, Y2*) бұрыштарының координатасы.

Процедура тікбұрышты *SetFillStyle* процедурасында тағайындалған ағымдағы өрнек үлгісімен және түспен бояйды (шекарасын сызбайды).

Келесі программа әдемі түстер эффектін көрсетеді (кездейсоқ тікбұрыштарды бояу).

```
uses Graph, CRT;
var
  d,r,e: Integer;
begin
{Графиканы іске қосамыз}
  d:=Detect; InitGraph(d,r,'');
  e:=GraphResult;
  if e <> grOk then
    WriteLn(GraphErrorMsg(e))
  else
    begin
{Экран ортасында терезе саламыз}
      d:=GetMaxX div 4;
      r:=GetMaxY div 4; Rectangle(d,r,3*d,3*r);
      SetViewPort(d+1,r+1,3*d-1,3*r-1,ClipOn);
{Кездейсоқ тікбұрыштарды салып, бояу циклы}
      repeat
SetFillStyle(Random(12),Random(succ(GetMaxColor)));
Bar(Random(Ge tMaxX),Random(Ge tMaxY),
        Random(Ge tMaxX),Random(Ge tMaxY));
   until KeyPressed;
     if ReadKey=#0 then d:=ord(ReadKey);
    CloseGraph
   end
end.
```

Bar3D процедурасы. Кеңістікте параллепипед салып, оның алдыңғы жағын бояйды. Тақырыбы:

Procedure Bar3D (XI,Y1,X2,Y2,Depth: Integer; Top: Boolean);

мұндағы *X1...Y2* – параллепипедтің алдыңғы жағының сол жақ жоғарғы (*X1, Y1*) және оң жақ төменгі (*X2, Y2*) бұрыштарының координатасы; *Depth* – кеңістіктегі бейненің пикселмен берілген үшінші өлшемі («тереңдігі»); *Top* – жоғарғы жағының бейнелену тәсілі.

Егер *Top* параметрінің мәні *True* болса, онда параллелепипедтің жоғарғы беті сызылады, кері жағдайда сызылмайды (бұл жағдай бірінің үстіне бірі қойылған параллелепипедтер үшін қолданылады). Бұл параметрдің мәні ретінде *Graph* модулінде анықталған келесі тұрақтылардың бірін қолдануға болады:

const
  TopOn = True;
  TopOff = False;

Сызу кезінде ағымдағы түзу стилі (*SetLineStyle*) және түс (*SetColor*) қолданылады. Алдыңғы жақ ағымдағы толтыру стилімен боялады (*SetFillStyle*).

Процедура көбіне бағаналық диаграмма салғанда қолданылады. Параллелепипедтің боялмаған жақтарынан бейненің басқа элементтері көрініп тұрады.

Келесі программа *Bar3D* процедурасын қолданудың түрлі мүмкіндіктерін көрсетеді.

```
uses Graph,CRT;
var
   d,r,e: Integer;
begin
{Графиканы іске қосамыз}
   d:=Detect;
   Ini-tGraph(d,r,'');
   e:=GraphResult;
   if e <> grOk then
      WriteLn(GraphErrorMsg(e));
   else
      begin
{Жоғарғы жағы бар бағана:}
         Bar3D (80, 100, 120, 180, 15, TopOn);
{Жоғарғы жағы жоқ бағана:}
         Bar3D (150, 150, 190, 180, 15,
TopOff);
{Бұл бағана келесі бағана үстінде тұр, ол
түссіз:}
         Bar3D (230, 50, 250, 150, 15, TopOn);
         Bar3D (220, 150, 260, 180, 15, TopOn);
{Бұл бағананың жоғарғы жағы жоқ, сондықтан ол
үстіне қойылған бағанаға кедергі жасамайды:}
         Bar3D (300, 150, 340, 180, 15, TopOff);
         SetLineStyle(3,0,1);
         SetColor(Yellow);
```

```
         SetFillStyle(LtSlashFill,Yellow);
         Bar3D (300, 50, 340, 150, 15, TopOn);
         if ReadKey=#0 then d:=ord(ReadKey);
         CloseGraph;
      end
end.
```

FillPoly процедурасы тұйық көпбұрыштың шекарасын сызып, бояйды. Тақырыбы:

Procedure FillPoly (N: Word; **var** Coords);

мұндағы *N* – тұйық көпбұрыштың төбелерінің саны; *Coords* – төбелер координатасы жазылған, *PointType* типті айнымалы.

Төбелер координатасы *Integer* типті екі мән арқылы беріледі: біріншісі көлденең, екіншісі тік координатасы. Олар үшін модульде анықталған келесі типтерді қолдануға болады:

```
   type
    PointType = record
    x,y: Integer
end;
```

Шекара сызығының стилі және түсі *SetLineStyle* және *SetColor* процедураларында беріледі, бояу типі және түсі – *SetFillStyle* процедурасымен.

Келесі мысалда экранға кездейсоқ боялған көпбұрыштар салынады.

```
uses Graph, CRT;
var
   d,r,e: Integer;
   p: array [1..6] of PointType; n,k: Word;
begin
{Графиканы іске қосамыз}
   d:=Detect; InitGraph(d,r,'');
   e:=GraphResult;
   if e <> grOk then
      WriteLn(GraphErrorMsg(e))
   else
      begin
{Экран ортасында терезе саламыз}
         d:=GetMaxX div 4;
```

```
          r:=GetMaxY div 4;
          Rectangle(d,r,3*d,3*r);
          SetViewPort(d+l,r+l,3*d-l,3*r-
l,ClipOn);
{Боялған кездейсоқ көпбұрыштарды шығару циклі}
          repeat
{Кездейсоқ түс пен өрнекті таңдаймыз}
SetFillStyle(Random(12),Random(succ(GetMaxCol
or)));
             SetColor
(Random(succ(GetMaxColor)));
{Кездейсоқ координаталарды белгілейміз}
             n:=Random (4) + 3;
             for k:=1 to n do with p[k] do
                begin
                   x:=Random (GetMaxX div 2);
                   y:=Random (GetMaxY div 2)
                end;
             FillPoly (n, p)
 {Шығарып бояймыз}
          until KeyPressed;
          if ReadKey=#0 then k:=ord(ReadKey);
          CloseGraph
       end
end.
```

FillEllipse процедурасы эллипс шекарасын салып бояйды. Тақырыбы:

Procedure FillEllipse(X, У,RX,RY: Integer);

мұндағы *X, Y* – центр координатасы; *RX, RY* – эллипстің пикселмен берілген көлденең және тік радиусы.

Эллипс *SetLineStyle* және *SetColor* процедураларында берілген сызықпен сызылып, *SetFillStyle* процедурасында орнатылған параметрлер көмегімен боялады.

Sector процедурасы эллипстің секторын сызып, ішін бояйды. Тақырыбы:

Procedure Sector (X,Y: Integer; BegA,EndA,RX,RY: Word);

мұндағы *BegA, EndA* – сәйкесінше эллипстік сектордың бас-

тапқы және соңғы бұрыштары. Басқа параметрлер *FillEllipse* процедурасының параметрлеріне сәйкес.

Келесі программада кездейсоқ боялған эллипстер мен секторлар салады. Программадан шығу үшін кез келген пернені басыңыз.

```
   uses Graph, CRT;
var
   d,r,e: Integer;
begin
{Графиканы іске қосамыз}
   d:=Detect; InitGraph(d,r,'');
   e:=GraphResult;
   if e <> grOk then
      WriteLn(GraphErrorMsg(e));
   else
      begin
{Экран ортасында терезе саламыз}
         d:=GetMaxX div 4;
         r:=GetMaxY div 4;
         Rectangle(d,r,3*d,3*r);
         SetViewPort(d+1,r+1,3*d-1,3*r-
1,ClipOn);
{Шығару циклы}
         repeat
SetFillStyle(Random(12),Random(succ(GetMaxCol
or)));
SetColor(Random(succ(GetMaxColor)));
Sector(Random(GetMaxX div),Random(GetMaxY div 2),
       Random(360),Random(360),Random(GetMaxX
    div 5),Random(GetMaxY div 5));
      FillEl.lipse (Random (GetMaxX div 2),
      Random(GetMaxY div 2),Random(GetMaxX
div 5),
            Random(GetMaxY div 5))
         until KeyPressed;
```

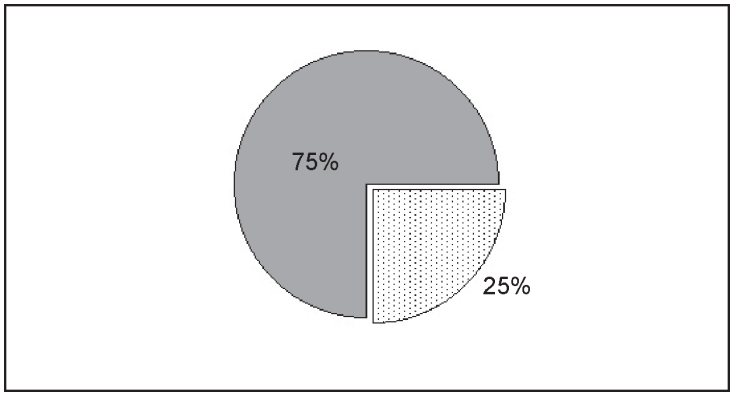
```
        if ReadKey=#0 then d:=ord(ReadKey);
        CloseGraph
    end
end.
```

PieSlice процедурасы дөңгелек секторын салып, оның ішін бояйды. Тақырыбы:

Procedure PieSlice(X,Y: Integer; BegA,EndA,R: Word);

Sector процедурасынан ерекшелігі тек бір көлденең радиус *R* көрсетіледі, қалған параметрлер *Sector* процедурасының параметрлеріне сәйкес.

Сектор *SetLineStyle* және *SetColor* процедураларында көрсетілген сызықпен сызылады да, *SetFillStyle* процедурасында орнатылған параметрлер бойынша боялады. Процедураны дөңгелек диаграмма салғанда қолданған ыңғайлы, мысалы, келесі программадағыдай (7.10-сурет).



7.10-сурет. PieSlice процедурасының көрінісі

```
uses Graph, CRT;
var
   d,r,e: Integer;
begin
   {Графиканы іске қосамыз}
   d := Detect;
   InitGraph(d, r, '');
   e := GraphResult;
```

```
   if e <> grOk then
      WriteLn(GraphErrorMsg(e))
   else
      begin
         {Кішкене сектор саламыз}
         SetFillStyle(WideDotFill, White);
         PieSlice(GetMaxX div 2+5,GetMaxY div
2+4,270,360,100);
         { Үлкен сектор саламыз }
         SetFillStyle (SolidFill, Red);
         PieSlice (GetMaxX div 2,GetMaxY div
2, 0,270,100).;
         {Жазулар шығарамыз}
         OutTextXY (GetMaxX div 2+90,GetMaxY
div 2+70, '25%');
         OutTextXY(GetMaxX div 2-50,GetMaxY
div 2-20, '75%');
         {Кез келген перненің басылуын
күтеміз}
         if ReadKey=#0 then d := ord(ReadKey);
      CloseGraph
   end
end.
```

7.5 Графикалық режимде мәтін шығару

Төменде сипатталатын стандарт процедуралар мен функциялар графикалық режимде мәтін шығаруды қолдайды. Бұл Write немесе WriteLn процедураларын қолданудан ерекше. Себебі, графикалық режимде түрлі қаріптермен, көлденеңінен немесе тігінен, қаріп мөлшерін өзгерте отырып, экранға мәтін шығаратын арнайы процедуралар бар. Бірақ Borland фирмасының графикалық режимде экранға шығарылатын стандартты қаріптерінің ішінде орыс қаріптері жоқ. Сол себепті, орысша мәтінді экранға шығару мүмкін емес.

Төменде Graph модулінің мәтін шығаруға арналған стандартты құралдары сипатталады.

QutText процедурасы курсордың ағымдағы орнынан бастап мәтіндік жолды шығарады. Тақырыбы:

Procedure OutText(Txt: String);

мұндағы Txt – шығарылатын жол.

Мәтін көлденеңінен шығарылғаннан кейін көрсеткіш мәтін соңында орналасады, ал егер мәтін тігінен шығарылса, көрсеткіш бастапқы орнында қалады. Жол алдын ала орнатылған стиль және туралау параметрлері бойынша шығарылады. Егер мәтін экран шекарасынан шығып кетсе, штрихтық қаріптерді қолданғанда шекарадан шыққан қаріптер кесіп тасталынады, ал стандартты қаріптер мүлде шығарылмайды.

OutTextXY процедурасы көрсетілген орыннан бастап мәтін шығарады. Тақырыбы:

Procedure OutTextXY (X,Y: Integer; Txt: String);

мұндағы X, Y – мәтін шығару нүктесінің координатасы; Txt – шығарылатын жол. OutText процедурасынан тек шығару нүктесінің координатасымен ғана ерекшеленеді. Көрсеткіш өз орнын өзгертпейді.

SetTextStyle процедурасы экранға шығарылатын мәтін стилін орнатады. Тақырыбы:

Procedure SetTextStyle(Font,Direct,Size: Word);

мұндағы Font – қаріп коды (нөмірі); Direct – бағытталу коды; Size – қаріп мөлшерінің коды.

Қаріп кодын көрсету үшін төмендегідей алдын ала анықталған тұрақтыларды қолдануға болады:

```
const
  DefaultFont = 0;{Нүктелік қаріп 8x8}
  TriplexFont = 1;{Үш еселенген қаріп TRIP.CHR}
  SmallFont = 2;{Кішірейтілген қаріп LITT.CHR}
  SansSerifFont = 3;{Тік қаріп SANS.CHR}
  GothicFont = 4;{Готикалық қаріп GOTH.CHR}
```

Бұл тұрақтылар 4.0, 5,0, 5.5 және 6.0 версиялар үшін барлық қаріптерді анықтайтындығын еске сала кетеміз. 7.0 версияда қаріптер саны әлде қайда кеңейген, алайда жаңа қаріптер үшін мнемоникалық тұрақтылар қарастырылмаған. Бұл версияда, SetTextStyle процедурасында келесі қаріптер нөмірін қолдануыңызға болады:

Нөмір	Файл	Қысқаша сипаттамасы
5	scri.chr	«Қолжазба» қаріп
6	simp.chr	Courier типті бірштрихты қаріп
7	tscr.chr	Times Italic типті қисайтылған қаріп
8	Icom.chr	Times Roman типті қаріп
9	euro. chr	Courier типті үлкейтілген қаріп
10	bold.chr	Үлкен екіштрихты қаріп

DefaultFont қаріпі Graph модуліне кіреді және оған кез келген уақытта қол жеткізуге болады. Бұл жалғыз матрицалық қаріп, оның символдары 8x8 пиксел матрица арқылы салынады. Қалған қаріпердің барлығы векторлық: олардың элементтері бағыты және көлемі арқылы сипатталатын векторлар жиынығы арқылы құрастырылады. Векторлық қаріптер бейнелену мүмкіндіктерінің кеңдігімен ерекшеленеді. Ал, олардың басты ерекшелігі қаріп мөлшерінің бейнелену сапасын төмендетпей, өте жеңіл өзгеруінде. Қаріптердің әрқайсысы жеке дискілік файлда орналасады. Егер Сіз қандай да бір векторлық қаріпті қолданатын болсаңыз, ол қаріп алдын ала сіздің каталогыңызға жазылып қойылуы тиіс. Таңдап алынған қаріп каталогта жоқ болса, оның орнына стандартты қаріп шығарылады.

DefaultFont қаріпі графика іске қосылған кезде шығарылатын мәтін сарапталып, графикалық драйвермен жасалынады. Сондықтан, егер сіздің компьютеріңіз мәтіндік режимде орыс әріптерін шығара алатын болса, онда осы қаріптің көмегімен графикалық режимде де орысша мәтін шығара аласыз. Басқа қаріптерді тек модификациялаудан кейін ғана шығаруға болады.

Шығарылатын мәтін бағытын беру үшін келесі тұрақтыларды қолдануға болады:

```
const
  HorizDir = 0;{солдан ңға қарай}
  VertDir = 1;{Төменнен жоғары қарай}
```

Стандартты OutText және OutTextXY процедуралары мәтінді тек екі бағытта ғана шығара алады. Векторлық қаріптердің құрылымын біле отырып, мәтінді кез келген бағытта шығаратын

жеке шығару процедураларын құрастыру қиын емес (F_GrText модулінің OutString процедурасы).

Әр қаріп өз өлшемін он есе өзгерте алады. Шығарылатын символ көлемі 1-ден 10-ға дейінгі аралықта өзгеретін Size параметрімен беріледі (нүктелік қаріп – 1 мен 32 диапазонында). Егер параметр мәні 0 болса, нөмірі 1 көлем орнатылады, егер 10 үлкен болса – 10 - нөмірлі көлем орнатылады. Экрандағы барлық мүмкіндіктерді көруді жүзеге асыратын қаріптің ең кіші көлемі - 4 (нүктелік қаріп үшін –1) болады.

Келесі программада әртүрлі қаріптер шығарылады (7.11-сурет). Олардың көлемдері жолдың биіктіктері бірдей болатындай етіп таңдалып алынған. Программаны орындауға жіберер алдында .CHR кеңейтілімі бар барлық қаріп файлдарын ағымдағы каталогқа көшіріп алу керек.

```
uses Graph, CRT;
const
   FontNames: array [1..10] of String[4] =
      ('TRIP','LITT','SANS','GOTH', 'SCRI',
'SIMP', 'TSCR', 'LOOM', 'EURO','BOLD');
   Tabl=50;
```

№	Name	Size	Symbols
1	TRIP	5	abcdefghijklmnopqrtuvwyz
2	LITT	10	abcdefghijklmnopqrstuvwxyz
3	SANS	4	abcdefghijklmnopqrstuvwxyz
4	GOTH	4	abcdefghijklmnopqrstuvwxyz
5	SCRI	4	abcdefghijklmnopqrstuvwxyz
6	SIMP	4	abcdefghijklmnopqrstuvwxyz
7	TSCR	4	abcdefghijklmnopqrstuvwxyz
8	LCOM	4	abcdefghijklmnopqrstuvwxyz
9	EURO	3	abcdefghijklmnopqrstuvwxyz
10	BOLD	2	abcdefghijklmnopqrstuvwxyz

7.11-сурет.

```
   Tab2=150;
   Tab3=220;
var
   d,r,Err, {Графиканы іске қосатын айнымалы-
лар}
   Y,dY,{Шығару ординатасы және оның өсімшесі}
   Size,{Символдар өлшемі}
   MaxFont,{Ең үлкен қаріп нөмірі}
   k: Integer;{Қаріп нөмірі}
   NT, SizeT, SynibT: String;{Шығарылатын жол}
   c: Char;
{-------------------}
Procedure OutTextWithTab ( S1, S2, S3, S4:
String);
{Tab1..Tab3 табуляция позициясын ескере оты-
рып, S1..S4 жолдарын шығарады}
begin
   MoveTo((Tab1-TextWidth(Sl)) div2,Y);
   OutText (S1);
   MoveTo(Tabl+(Tab2-Tabl-TextWidth(S2))
div2,Y);
   utText (S2);
   MoveTo(Tab2+(Tab3-Tab2-TextWidth(S3)) div
2,Y);
   OutText(S3);
   if S4='Symbols' then
{Symbols бағанының тақырыбы}
      MoveTo((Tab3+GetMaxX-TextWidth(S4)) div
2,Y)
   else {Қалған жолдар}
      MoveTo(Tab3+3,Y);
   OutText(S4)
end;
{------------}
begin
   {Графиканы іске қосамыз}
   InitGraph(d,r,'');
```

```
   Err:=GraphResult;
   if ErrogrOk then
      WriteLn(GraphErrorMsg(Err))
   else
      begin
 {Қаріптер санын анықтаймыз:}
         {$IFDEF VER70'}
         MaxFont:=10;
         {$ELSE}
         MaxFont:=4;
         {$ENDIF}
         SetTextStyle(l,0,4);
         Y:=0;
        OutTextWithTab('N','Name',Size','Symbo
ls');
{Тақырыптың Y сызығының биіктігін анықтаймыз}
         Y:=4*TextHeight('Z') div3;
         Line(0,Y,GetMaxX,Y);
{Кестенің Y сызығының басын және әр жолдың dY
биіктігін анықтаймыз}
         Y:=3*TextHeight('Z') div 2;
         dY:=(GetMaxY-Y) div (MaxFont);
{Символдар жолын дайындаймыз}
         SymbT:='';
         for c:='a' to 'z' do
            SymbT:=SymbT+c;
{Кесте жолдарын шығару цикл}
        for k:=1 to MaxFont do
            begin
               Size:=0;
 {Жол биіктігі dY тең болғанша өлшемді
ұлкейтеміз}
               repeat
                  inc(Size);
                  SetTextStyle(k,0,Size+1);
               until (TextHeight('Z')>=dY) or
```

```
(Size=10) or (TextWidth(FontNames[k])>(Tab2-
Tab1));
{Қаріптің NT нөмірін және и SizeT өлшемін дай-
ындаймыз}
                Str(k,NT);
                Str(Size,SizeT);
{Кесте жолын шығарамыз}
       SetTextStyle(k,HorizDir,Size);
       OutTextWithTab(NT,FontNames[k],SizeT,Sy
mbT);
                inc(Y,dY)
             end;
{Рамка сызықтарын саламыз}
         Rectangle(0,0,GetMaxX,GetMaxY);
         Line(Tab1,0,Tabl,GetMaxY);
         Line(Tab2,0,Tab2,GetMaxY);
         Line(Tab3,0,TaЬ3,GetMaxY);
{Тұтынушы әрекетін күтеміз}
         ReadLn;
         CloseGraph
      end
end.
```

SetTextJustlfy процедурасы енгізілетін мәтінді курсордың ағымдағы орнына қарай немесе берілген координаталарға байланысты тураласуды тағайындайды. Тақырыбы:

Procedure SetTextJustify(Huri2,Vert: Word);

мұндағы Horiz – көлденең туралау; Vert – тігінен туралау.

Туралау мәтіннің қалай орналасатындығын – көрсетілген орыннан солға немесе оңға қарай, жоғары, төмен не ортаға тураланатындығын анықтайды. Мұнда келесі тұрақтыларды қолдануға болады:

const

LeftText = 0; {Көрсеткіш мәтіннің сол жағында }

CenterText= 1; { Симметриялы солға және оңға, жоғары және төмен}

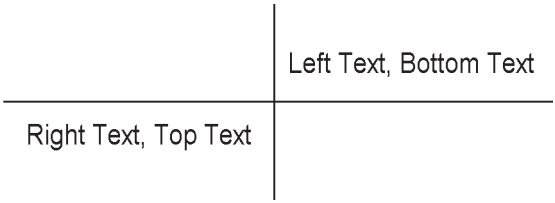
RightText = 2; { Көрсеткіш мәтіннің оң жағында }

BottomText= 0; { Көрсеткіш мәтіннің төменгі жағында }

TopText = 2; { Көрсеткіш мәтіннің жоғарғы жағында }

Келесі программа мәтінді графикалық экранның центріне байланысты әртүрлі орналастыру тәсілдерін бейнелейді (7.12-сурет).

```
uses Graph, CRT;
var
   d,r,e: Integer;
begin
{Графиканы іске қосамыз}
   d:=Detect; InitGraph(d,, r, ' ');
```



7.12-сурет.

```
   e:=GraphResult;
   if e <> grOk then
      WriteLn(GraphErrorMsg(e))
   else
      begin
{Экран ортасында қилысқан сызықтар саламыз}
  Line(0,GetMaxY div 2,GetMaxX,GetMaxY div 2);
  Line(GetMaxX div 2,0,GetMaxX div 2,GetMaxY);
Мәтінді центрдің оң және жоғарғы жағына орна-
ластырамыз}
         SetTextStyle(TriplexFont,HorizDir,3);
         SetTextJustify(LeftText,BottomText);
         OutTextXY (GetMaxX div 2, GetMaxY div
2, 'LeftText,BottomText');
{Мәтінді сол және төменгі бөлікке орналастыра-
мыз}
  SetTextJustify (RightText, TopText);
  OutTextXY (GetMaxX div 2, GetMaxY div 2,
'RightText, TopText');
         if ReadKey=#0 then d := ord(ReadKey);
```

```
         CloseGraph
      end
end.
```

SetUserCharSize процедурасы берілген пропорциялар бойынша енгізілетін символдар мөлшерін өзгертеді. Тақырыбы:

Procedure SetUserCharSize(XI,X2,Y1,Y2: Word);

мұндағы XI...Y2 – көлденең және тік пропорцияларды анықтайтын Word типті өрнек.

Процедура тек векторлық қаріптерге қолданылады. Пропорциялар қаріптің ені мен биіктігінің стандартты мәннен неше есе өсетіндігін көрсететін масштабтық коэффициентті береді. Көлденең коэффициент X1-дің X2-ге қатынасы ретінде беріледі, тігінен –Y1-дің Y2-ге қатынасы ретінде беріледі. Мысалы, символдардың енін екі есе өсіру үшін X1=2 және X2=1 деп беру керек. Символдардың стандартты мөлшері SetUserCharSize процедурасының алдыңғы параметрлерін алып тастайтын SetTextStyle процедурасы арқылы беріледі.

Келесі мысалда кішірейтілген қаріптің пропорцияларының өзгеруін көреміз.

```
uses Graph, CRT;
var
   d, r, e: Integer;
begin
 {Графиканы іске қосамыз}
   d:=Detect; .InitGraph (d, r, '');
   e:=GraphResult;
   if e <> grOk then
      WriteLn(GraphErrorMsg(e))
   else
      begin
         MoveTo (0, GetMaxY div 2);
        SetTextStyle (SmallFont, HorizDir, 5);
         SetTextJustify (LeftText,
BottomText);
 {Стандартты 5 биіктікпен мәлімет шығарамыз}
         OutText ('Normal Width,');
         {Қаріптің енін екі еселейміз}
```

```
          SetUserCharSize (2, 1, 1, 1);
          OutText (' Double Width, ');
 {Биіктігін екі есе өсіріп, стандартты енді
береміз}
          SetUserCharSize (I, 1, 2, 1);
          OutText ('Double Height,');
          SetUserCharSize (2, 1, 2, 1);
          OutText (' Double Width and Height');
          if ReadKey=#0 then d := ord(ReadKey);
          CloseGraph
      end
end.
```

TextWidth функциясы енгізілетін жолдың пикселмен берілген ұзындығын қайтарады. Тақырыбы:

Function TextWidth(Txt: String): Word;

Енгізудің ағымдағы стилі мен сәйкесінше SetTextStyle және SetUserCharSize процедураларымен берілген символдардың мөлшерінің өзгеру коэффициентін ескереді.

TextHeight функциясы енгізілетін жолдың пикселмен берілген биіктігін қайтарады. Тақырыбы:

Function TextHeight(Txt: String): Word;

GetTextSettings процедурасы мәтіннің ағымдағы стилі мен туралануын қайтарады. Тақырыбы:

Procedure GetTextSettins(**var** Textlnfo: TextSettingsType);

мұндағы Textlnfo – Graph модулінде төмендегідей анықталған TextSettingsType типті айнымалы:

```
type
  TextSettingsType = record
  Font : Word; {Қаріп нөмірі}
  Direction: Word; {Бағыты}
  CharSize : Word; {Мөлшер коды}
  Horiz : Word; {Көлденең туралау}
  Vert : Word; {Тігінен туралау }
end;
```

InstallUserFont функциясы программаға стандартты емес векторлық қаріпті қолдануға мүмкіндік береді. Тақырыбы:

Function InstallUserFont(FileName: String): Integer;
мұндағы FileName – векторлық қаріп жазылған файл аты.

InstallUserDriver функциясы BGI – драйверлер жүйесіне стандартты емес графикалық драйверді қосады. Тақырыбы:

Function InstallUserDriver(FileName: **String;** AutoDetectPtr: Pointer): Integer;

мұндағы FileName – драйвер программасы жазылған файл аты; AutoDetectPtr – дисплей түрін автоматты түрде анықтайтын арнайы процедураға кіру нүктесінің адресі, басқа процедуралар сияқты драйвер құрамында болуы тиіс.

Бұл функция стандартты графикалық драйверлер тобын кеңейтеді және көбіне аппараттық құралдарды программалау үшін қажет.

Көп есептердің нәтижесін график түрінде берген дұрыс. График тұрғызу есептерінің бастапқы берілуі алуан түрлі. Мәселен, бір немесе бірнеше функция графигі бір немесе бірнеше масштабта керек болуы мүмкін. Функциялар аналитикалық тәсілмен немесе кесте түрінде берілгенде функцияны іштей буындау (интерполяция) қажет болуы мүмкін. Графикті қарапайым түрде немесе логарифмдік масштабпен, координаталық өстерді қосып немесе қоспай салуға болады. Алайда график тұрғызудың негізгі принциптері өзгеріссіз қалады.

Функция графигін салу үшін, экранда графиктің әр нүктесіне сәйкес нүкте қою керек. Экрандағы нүктелер координатасы график ауданына қатысты, масштабты және өстің экранда төмен бағытталғандығын ескере отырып анықталады. *X* және *Y* өстерінің масштабы экрандағы график ауданына және мәндер интервалына байланысты алынады.

График нүктелерінің координатасы төмендегідей түрде анықталады:

$$kx_i = int(x_{ima} - x_i) \cdot m_x n + kx_n,$$
$$ky_i = int(y_{max} - y_i) \cdot m_y n + ky_n,$$

мұндағы *int* – санның бүтін бөлігін алу дегенді білдіреді.

График барлық есептелген нүктелер арқылы өтетін сызықтар немесе сынық сызықтар жиынтығы ретінде салынады.

Координаталық сетканы салу координаталары дәл осылай анықталады.

7.1-мысал. y=cos(x+2)/2 функциясының графигін салатын программа жазыңыз. Алдымен берілген интервалдағы нүктелер саны анықталады және графикке координаталық сетка салынады.

Төменде, осы есептің программа мәтіні берілген.

```
Program Gr;
Uses Crt,Graph;
Const
n=5; {сан жазылатын позиция саны }
m=2; {мән шығардағы мантисса мөлшері}
k=100; {нүктелер саны}
nx=5; ny=5; {x және y бойынша сеткадағы
түзулер саны}
kxn=60; kxk=600;
kyn=45; kyk=350; {терезе параметрі}
Type arr=array[1..100] of real;
ari=array[1..100] of integer;
Var
gd,gm,i:integer; {адаптер типі және режимі}
x,y:arr; {аргумент және функция мәні үшін
жиымдар}
kx,ky:ari; {x және y координатаық нүктелері
үшін жиым}
ymin,ymax:real; {y экстремалды мәні}
dx,dy:real; {графиктегі x және y бойынша сетка
қадамы}
dkx,dky:integer; {экрандағы x және y бойынша
сетка қадамы}
mx,my:real; {масштаб коэффициенті}
st:string[5]; {жұмыс жолы}
h,xn,xk:real; {қадам, x өсі бойынша интервал}
Begin
ClrScr;
Write('Интервалдың бастапқы және соңғы мәнін
енгізіңіз:');
ReadLn(xn,xk);
h:=(xk-xn)/(k-1);{x өсі бойынша қадамды
анықтаймыз}
```

```
x[1]:=xn;
ymin:=1e30;
ymax:=-1e10;
for i:=1 to k do {функцияны табуляциялап, экс-
тремумды іздейміз}
begin
y[i]:=cos(x[i]+2)/2;
if y[i]>ymax then ymax:=y[i];
if y[i]<ymin then ymin:=y[i];
if i<>100 the x[i+1]:=x[i]+h;
end;
mx:=(kxk-kxn)/(x[k]-x[1]); {x өсі бойынша мас-
штабты анықтаймыз}
my:=(kyk-kyn)/abs(ymax-ymin); {y өсі бойынша
масштабты анықтаймыз}
for i:=1 to k do {нүктелер координатасын
анықтаймыз}
begin
kx[i]:=round((x[i]-x[1])*mx)+kxn;
ky[i]:=round((ymax-y[i])*my)+kyn;
end;
gd:=detect;
InitGraph(gd,gm,''); {графикалық режимді іске
қосамыз}
SetColor(4); {ағымдағы түс - қызыл}
OutTextXY(180,20,'Y=cos(x+2)/2'); {тақырыпты
шығарамыз}
SetColor(17); {салу түсі – көгілдір}
SetBKColor(7); {фон түсі – сұр}
Rectangle(kxn,kyn,kxk,kyk); {графикті шығару
үшін тікбұрыш саламыз}
SetColor(4); {ағымдағы түс - қызыл}
for i:=1 to k-1 do {графикті шығарамыз}
Line(kx[i],ky[i],kx[i+1],ky[i+1]);
dkx:=round((kxk-kxn)/nx); {x өсі бойынша сетка
қадамын анықтаймыз}
dky:=round((kyk-kyn)/ny); {y өсі бойынша сетка
қадамын анықтаймыз}
```

```
SetColor(17); {ағымдағы түс – ақшыл көк}
for i:=1 to nx do { x өсіне параллель торды
саламыз}
Line(kxn+dkx*i,kyn,kxn+dkx*i,kyk);
for i:=1 to ny do {y өсіне параллель торды
саламыз }
Line(kxn,kyk-dky*i,kxk,kyk-dky*i);
dx:=(x[k]-x[1])/nx; {x өсі бойынша тор қадамын
анықтаймыз}
dy:=(ymax-ymin)/ny; {y өсі бойынша тор қадамын
анықтаймыз}
SetTextJustify(1,2); {"төмен қарай орталыққа"
туралау}
for i:= 1 to nx+1 do {аргумент мәнін
шығарамыз}
begin
Str((x[1]+dx*(i-1)):n:m,st); {санды жолға ай-
налдырамыз}
OutTextXY(kxn+dkx*(i-1), kyk+6,st); {тор
сызығының астына мәндерді шығарамыз}
end;
SetTextJustify(2,1); {«сол жақтан орталыққа»
туралау}
for i:=1 to ny+1 do {функция мәнін шығарамыз}
begin
Str((ymin+dy*(i-1)):n:m,st); {санды жолға ай-
налдырамыз }
OutTextXY(kxn-6,kyk-dky*(i-1),st); {y өсінің
сол жағына шығару}
end;
ReadLn; {ENTER-ді басуды күтеміз}
Closegraph;
End.
```

Программа жұмысының нәтижесі берілген интервалдағы функция графигі.

Нәтижені графиктен басқа диаграмма немесе түрлі гистограммалар ретінде де алуға болады. Тағы бір мысал қарастырайық.

7.2-мысал. 12-ден көп емес мәндер бойынша дөңгелек диаграмма салатын программа жазыңыз. Әр мән дөңгелектің бір секторын алып тұрады.

Дөңгелек диаграмма боялған дөңгелек секторларының жиынтығы ретінде салынады. Сектор бұрышы жалпы мәндер қосындысының ішіндегі сәйкес мәннің үлесіне пропорционал. Жазу сектор бұрышының биссектрисының қарсысына жазылу керек. Төменде программа мәтіні келтірілген.

```
Program diagramma;
Uses Graph;
Const
kmax=12; {функцияның максимальді мәндер саны}
r=150; {диаграмма радиусы}
n=5; {шығару өрісінің ені}
m=2; {санның бөлшек бөлігінің мәлшері}
Type
mas=array[1..kmax+1] of integer;
mas1=array[1..kmax] of real;
Var
f:mas1; {функция мәндерінің жиымы}
alfa:mas; {диаграмма бұрыштары мәнінің жиымы}
driver,err, {адаптер типі және жұмыс режимі}
k, {функцияның нақты мәндерінің саны}
bet, {диаграмма секторының радиусы және бис-
сектрисасы құраған бұрыштың шамасы}
y,x, {сектор доғасының центрі болатын нүктенің
координатасы}
x1,y1, {жазуды шығарудың бастапқы нүктесінің
координатасы}
xn,yn, {диаграмма центрінің координатасы }
i:integer;
st:string[5]; {алдын ала шығарылатын жазу
жолы}
s:real; {функция мәндерінің қосындысы}
Begin
WriteLn('Нүктелер санын енгізіңіз
(1-ден,'kmax:3-ке дейін')');
```

```
ReadLn(k);
{функция мәндерін шығарып, олардың қосындысын
анықтаймыз}
s:=0;
for i:=1 to k do
begin
WriteLn('Функцияның',i:3,'-ші мәнін
енгізіңіз');
ReadLn(f[i]); while f[i]<0 do
begin
WriteLn('Беруге болмайтын мән, енгізуді
қайталаңыз');
WriteLn('Функцияның',i:3,'- ші мәнін
енгізіңіз');
ReadLn{f[i]);
end;
s:=s+f[i];
end;
if s=0 then
begin
WriteLn('Барлық мәндер нөлге тең.');
ReadLn; halt(1); {қате бойынша шығу}
end;
{графикалық режимді іске қосамыз}
driver:=detect;
InitGraph(driver,err,'');
Setbkcolor(15); {фон түсі ақ}
SetPalette(1,0);
SetColor(1); {салу түсі қара}
{диаграмма центрінің координатасын есептейміз }
xn:=GetMaxX div 2;
yn:=GetMaxY div 2;
 {Диаграмма секторының бұрыштарының мәнін
есептейміз }
alfa[1]:=0;
for i:=2 to k+1 do
```

```
begin
if i<>k+1 then alfa[i]:=alfa[i-1]+round(f[i-
1]/s*360)
else alfa[k+1]:=360;
SetFillStyle(i mod 10,i); {секторды бояудың
типі мен түсін орнатамыз}
Pieslice(xn,yn,alfa[i-1],alfa[i],r); {сектор
саламыз}
 {Жазу шығарудың бастапқы координаталарын
есептейміз }
bet:=(alfa[i-1]+alfa[i]) div 2;
x:=xn+round(r*cos(bet*pi/180));
y:=yn-round(r*sin(bet*pi/180));
if ((bet>=0) and (bet<=90)) or ((bet>=270) and
(bet<=360)) then x1:=x+10 else x1:=x-8*n-10;
if (bet>=0) and (bet<=180) the y1:=y-15 else
y1:=y+7;
Str(f[i-1]:n:m,st); {мәнді жолға айналдырамыз}
OutTextXY(x1,y1,st); {жазу шығарамыз}
end;
ReadLn;
CloseGraph;
End.
```

Программа жұмысының нәтижесі дөңгелек диаграмма болады.

Бақылау сұрақтар

1. *Графикалық процедуралар мен функциялар кітапханасы қалай аталады?*
2. *Графикалық режимді қалай іске қосамыз?*
3. *Графикалық режимнен қалай шығамыз?*
4. *Кесінді сызу параметрлері қалай өзгереді?*
5. *Кесінді сызу процедурасын сипаттаңыз.*
6. *Дөңгелек, дөңгелек доғасын және эллипс доғасын салу процедураларын сипаттаңыз.*
7. *Тіктөртбұрыш салу процедурасын сипаттаңыз.*

8. *Ағымдағы түсті және фон түсін өзгерту процедурасын сипаттаңыз.*
9. *Экрандағы нүкте түсін қалай өзгертуге болады.*

Тапсырмалар

1. *Кез келген тақырыпқа «пейзаж» салыңыз.*
2. *Автомобиль суретін салыңыз.*
3. *Роботтың суретін салыңыз.*
4. *Дөңгелек және эллипс салу процедураларының көмегімен түрлі түсті ромашка гүлін салыңыз.*
5. *Ішіне үшбұрыш, квадрат, жұлдыз салынған үш шеңбер салыңыз. Фигуралардың түстері әр түрлі болсын.*
6. *Олимпиада жалауын салыңыз.*
7. *Қисық сызықты жазу дәптерінің фрагментін салыңыз.*
8. *Туған жылыңыздың датасын тікбұрыштардың көмегімен жазыңыз.*
9. *Фамилияңыздың, атыңыздың және тегіңіздің бас әріптерін түрлітүсті тікбұрыштар көмегімен жазыңыз.*
10. *Ішінде өз атыңыз жазылған тікбұрыш контурын салыңыз.*
11. *Ақ түсті экранға өзіңіздің пошталық индексіңізді конверттегідей етіп жазыңыз.*
12. *Түрлі-түсті көбелек суретін салыңыз.*

8. МОДУЛЬДІК ПРОГРАММАЛАУ

8.1 Программа құрылымы

Турбо Паскаль тіліндегі программа екі бөлімнен тұрады: программа тақырыбы және программа блогы.

Тақырыптың жалпы түрі:

PROGRAM <программа аты>**[(**<файлдар тізімі>**)];**

Программа тақырыбын жазбай кетуге де болады. Стандартты **INPUT** (енгізу) және **OUTPUT** (шығару) сөздерін де жазбай кетуге болады, себебі олар (үнсіз) келісім бойынша бәрібір алынады.

Программа блогы сипаттау және орындау бөлімдерінен тұрады. Сипаттау бөлімі міндетті түрде орындау бөлімінің алдында жазылады және онда келесідей бөліктер болуы мүмкін:

LABEL <белгілерді сипаттау>**;** – белгілерді сипаттау бөлігі,

CONST <тұрақтыларды сипаттау>**;** – тұрақтыларды сипаттау бөлігі,

TYPE <типтерді сипаттау>**;** – типтерді сипаттау бөлігі,

VAR <айнымалыларды сипаттау>**;** – айнымалыларды сипаттау бөлігі,

PROCEDURE < процедураны сипаттау**> ;** – процедураны сипаттау бөлігі,

FUNCTION <функцияны сипаттау>**;** – функцияны сипаттау бөлігі,

BEGIN

<программаның орындалатын бөлімі> – операторлар бөлімі

END.

Программа мәтіні ұзындығы 127 символдан аспайтын жолдар түрінде жазылады. Турбо Паскальда сипаттау бөліктерінің жазылу реттілігі жоғарыдай болуы міндетті емес, олар кез келген түрде және әр бөлік бірнеше рет қайталануы немесе мүлде болмауы да мүмкін. Операторлар бөлімі (**BEGIN** сөзінен басталып, **END** сөзімен аяқталады) өзара нүктелі үтір арқылы

бөлінген операторлардан тұрады. Программаның орындалатан бөлімі міндетті түрде болуы керек деп саналады.

Компилятор жұмысын директивалар арқылы басқаруға болады. Оларды бастапқы мәтінге арнайы синтаксисі бар комментарий ретінде қосады. Егер процедура немесе функция жеке файл түрінде сақталса, онда олардың сипаттамасын бастапқы мәтінге қосу үшін, компиляторға **INCLUDE** директивасы төмендегідей түрде процедура және функцияның сипаттау бөлігінде жазылуы тиіс:

{$I <файл аты>**}.**

Мысалы,

```
        PROGRAM A1;
        VAR ...
{$I  B1.PAS}
BEGIN
            ...
END.
```

B1.PAS файлының түрі төмендегідей болуы мүмкін:

```
PROCEDURE PP;
VAR ...
BEGIN
      ...
END;
```

8.2 Процедураны сипаттау және шақыру

Есептеулердің бірнеше рет қайталанатын бөліктерін және программаның модульділігін қамтамасыз ету үшін Турбо Паскаль тілінде процедура және функция қолдану мүмкіндіктері қарастырылған.

Процедура – арнайы ат берілген, қандай да бір аргументтер санына байланысты нәтижені есептеуге арналған операторлар жиынтығы.

Процедура немесе **функция** (жалпы аты – ішкі программа) негізгі программаның немесе басқа процедураның (функцияның) сипаттау бөлімінде анықталады. Процедураның (функция) құрылымы негізгі программа құрылымына ұқсас: тақырып бөлігі, сипаттау бөлігі және орындалу бөлігінен тұрады.

Процедура тақырыбының синтаксисі:

PROCEDURE <процедура аты> **[(** <формальді параметрлер тізімі >**)];**

Мысалы:

PROCEDURE PR1 (A,B,C : INTEGER; VAR S: REAL);

мұндағы **PR1** – процедура аты, ал **A,B,C,S** – параметр болып табылатын айнымалылар аты.

Процедураның негізгі программадан айырмашылығы, процедурада тақырып міндетті түрде жазылады және нүктелі үтірмен аяқталады. Процедура сипаттамасы формальді параметрлермен орындалады.

Процедура операторы процедураны негізгі программадан немесе басқа процедурадан (функциядан) шақыруды іске асырды.

Шақыру төмендегі форма бойынша іске асырылады:

<процедура аты > **[(**<нақты параметрлер тізімі>**)]**;

Жоғарыда келтірілген процедура аты үшін төмендегідей шақыру операторын жазуға болады:

PR1 (A,B,C,S);

Процедура тақырыбындағы тізім форматының процедураны шақырудағыдан айырмашылығы бар. Шақыру кезінде айнымалылар, тұрақтылар және өрнектер үтір арқылы жазылады, ал тақырыпта, айнымалылардың жазылуы айнымалыларды сипаттау бөліміндегі айнымалыларды жариялауға ұқсас келеді. Тізімдегі барлық элементтер үшін мәндер типі көрсетілу керек. Бір типке жататын айнымалылар үтір арқылы жазылады, ал әр түрлі типтегі мәндер тобы нүктелі үтір арқылы ажыратылады. Процедура орындалғаннан кейін, басқару процедураны шақыру операторынан кейін ораласқан жолға беріледі.

8.3 Функцияны сипаттау

Функция бір мәнді есептеуге арналған және өрнектерде стандартты функцияға тәрізді қолданылады.

Функция тақырыбының синтаксисі:

FUNCTION <функция аты >**[(**<формальді параметрлер тізімі >**)]:**
<нәтиже типі>;

Мысал:

FUNCTION PRF (A,B,C: INTEGER) : REAL;

Функцияны сипаттаудың процедурадан айырмашылығы:

- функция орындалуының нәтижесі тек бір ғана мән болуы тиіс;
- нәтиже идентификаторы формальді параметрлер тізімінде көрсетілмейді;
- функцияның орындалатын бөлігінде, функция атына кем дегенде бір рет нәтиже мәні меншіктелуі керек (көбіне функциядан шығарда);
- формальді параметрлер тізімінен кейін нәтиже мәні көрсетіледі;
- функция шақырылғаннан кейін басқару өрнектің функциядан кейінгі операциясына беріледі.

Функцияны шақыру үшін функция көрсеткішін қолданамыз (нақтылы параметрлер көрсетілген функция аты). Функция көрсеткіші есептелінетін қандай да бір өрнекте (меншіктелу операторының оң жағында, енгізу операторының мәндер тізімінде, шартты операторлардың логикалық өрнегінде, т.с.с.) көрсетілуі керек. Жоғарыда көрсетілген функцияны келесі тәсілдердің бірімен шақыруға болады:

```
S:=PRF ( A,B,C);
Writeln ( PRF ( A,B,C));
If  PRF ( A,B,C)>20 then K=K+1;
```

8.4 Формальді және нақтылы параметрлер

Процедураны (функцияны) сипаттағанда оның тақырыбында келесі параметрлер түрі көрсетілуі мүмкін:

- параметр-мән;
- параметр-айнымалы;
- параметр-тұрақты;
- параметр-процедура;
- параметр-функция.

Параметрлерді жазғанда мыналарды есте сақтау керек:

- формальді және нақтылы параметрлер саны бірдей болуы керек;
- нақтылы параметрлердің жазылу реті мен типі сәйкес формальді параметрлердің жазылу реті мен типіне сәйкес болуы керек;
- формальді және нақтылы параметр идентификаторлары бірдей болуы мүмкін;

- формальді параметрлер Турбо Паскаль тілінде тақырып бөлігінде сипаттамалармен бірге болады және оларды процедураның (функцияның) сипаттау бөлігінде хабарлаудың қажеті жоқ;
- формальді параметрлер қарапайым немесе ертеде анықталған типті болуы керек.

Ішкі программаға жиымды берер кезде оны алдын ала **TYPE** типтерді сипаттау бөлімінде сипаттау керек.

Мысал.

```
TYPE  TV=ARRAY [1..30] OF  INTEGER;
        TM=ARRAY [1..20,1..20]  OF  REAL;
...
PROCEDURE TOP ( A:TM; VAR  B: TV ; N: INTEGER);
...
```

Мұнда жиымның екі типі сипатталған. **TV** – бір өлшемді жиым үшін және **TM** – екі өлшемді жиым үшін. Формальді параметрлер тізімінде **A** және **B** айнымалылары үшін, сәйкесінше, матрица мен векторды алдын ала анықтаған типтер қолданылады.

Процедура немесе функция тақырыбында берілген параметрлер тізімі ішкі программаны шақырушы программамен байланыстырады. Осы параметрлер арқылы ішкі программаға бастапқы мәндер беріледі және нәтиже қайтарылады (процедурада). Турбо Паскаль тілінде параметрлерді берудің екі түрі бар: мәні бойынша және сілтеме бойынша.

Параметр-мән

Параметрлердің мәнін, ішкі программаның жергілікті (ішкі) айнымалыларына жадыдан орын бөлінетін стекке жіберген кезде, сәйкес нақты параметрлердің мәндері жазылатын қосымша орын бөлінеді. Шақырушы программада параметр-мән үшін ішкі программаның аргументі ретінде тек айнымалы емес өрнекті де қолдануға болады. Ішкі программаның жұмысы аяқталғаннан кейін жадыдан бұл параметрге бөлінген орынға қол жеткізуге болмайды. Сондықтан, параметрді мән бойынша бергенде жадыдағы орынды нәтиже алу үшін пайдалануға болмайды.

Параметр-айнымалы

Сілтеме бойынша шақырғанда, ішкі программада берілетін айнымалылар үшін жадыдан орын бөлінбейді. Ішкі программаға

айнымалының мәні емес, сәйкес нақты параметрдің жадыдағы орнына сілтеме беріледі. Осы айнымалымен есептеулер жүргізетін ішкі программа шын мәнінде сәйкес нақты параметрмен жұмыс істейді. Сондықтан, процедура орындалғанда айнымалымен орындалған өзгерістер сақталады. Параметр-айнымалыны жазған кезде формальді параметрлер тізімінде **VAR** қызметші сөзі көрсетіледі. Есептелетін нәтижелер үшін тек параметр-айнымалылар қолдануға болады. Формальді параметр-айнымалылар үшін нақтылы мән ретінде тұрақты немесе өрнек қолдануға болмайды. Себебі, олардың адресі жоқ.

Параметр-айнымалы ретінде көлемі көрсетілмеген ашық типті жиым және жолдарды қолдануға болады. Ашық жиым ішкі программаның базалық элементтер типін анықтайтын, бірақ көлемі мен шекарасын анықтамайтын, формальді параметрі бола алады. Бұл жағдайда элементтердің индексі нөлден басталады. Ашық жиымның жоғарғы шекарасы **HIGH** функциясының көмегімен қайтарылады. Мұндай сипаттама тек бір өлшемді жиым үшін ғана қолданылады. Ашық жиым үшін стекте оның көшірмесі жасалады. Бұл стектің толып кетуіне әкелуі мүмкін.

Ашық жиымды қолдануға мысал қарастырайық. Бір өлшемді жиым элементтерінің қосындысын есептеу керек.

```
  FUNCTION SUM (VAR A: ARRAY OF INTEGER):INTEGER;
VAR  S,I : INTEGER;
BEGIN
    S:=0;
    FOR I:=0 TO HIGH(A)  DO
    S:=S+A[I];
    SUM:=S;
END;
```

Негізгі программада мұндай жиымды **Var A: array [-2 .. 3] of integer;** түрінде сипаттауға болады. Бұл жерде ең маңыздысы жиымның нақты шекарасы емес оның элементтерінің саны, ол 6-ға тең.

Ашық жолды компилятордың **{$P+}** директивасын қолданып, стандартты **OPENSTRING** және **STRING** типтерінің көмегімен беруге болады.

Мысалы,

```
 PROCEDURE ZAP ( VAR ST : OPENSTRING; R: INTEGER );
```

немесе

```
{$P+}
PROCEDURE ZAP ( VAR ST : STRING; R: INTEGER );
```

Турбо Паскаль тілінде жолдың формальді және нақтылы параметрлердің ұзындықтарының сәйкестігін қадағалауды алып тастайтын, **{$V-}** компиляция режимін орнатуға болады. Көлемі қысқа жол берілген кезде, формальді параметр ұзындығы осыған сәйкестелінеді, ал көлемі ұзын жол берілгенде, формальді параметр ұзындығы максимальді ұзындыққа дейін қысқартылады. Бақылау тек параметр-айнымалы үшін ғана қосылады, параметр-мән үшін бақылау қажет емес.

Процедура және функция қолданылатын мысал қарастырайық. M бағана және N жолдан тұратын матрицаның барлық элементтері өсу немесе кему бойынша реттелген және жай сан болып келген бағана нөмірлерінен тұратын вектор құрастыратын процедура жазу керек. Бастапқы мәндерді енгізу, процедураны шақыру және нәтижені шығару негізгі программада орындалады.

```
USES CRT;
TYPE TMAS=ARRAY[1..100,1..100] OF WORD;
       TVECT=ARRAY[1..100] OF WORD;
VAR A:TMAS;
      V:TVECT;
  N,M,K:BYTE;
     I,J:BYTE;
PROCEDURE FORM(VAR X:TMAS;   {матрица}
  N,M:BYTE;     {жол және бағана саны }
                VAR R:TVECT;  {нәтиже-вектор}
  VAR K:BYTE);  {алынған вектор ұзындығы}
VAR I,J,Z,S:BYTE;
             F:BOOLEAN;
FUNCTION PROS(B:WORD):BOOLEAN;
{жай санды тексеру функциясы}
VAR I:WORD;
BEGIN
      IF B<>1 THEN PROS:=TRUE
                      ELSE PROS:=FALSE;
      FOR I:=2 TO B DIV 2 DO
```

```
            IF B MOD I = 0 THEN PROS:=FALSE;
END;
BEGIN
      K:=0;
      FOR J:=1 TO M DO
      BEGIN
            Z:=0; S:=0; F:=TRUE;
   FOR I:=1 TO N-1 DO
            BEGIN
                  IF X[I,J]>X[I+1,J] THEN Z:=Z+1;
                  IF X[I,J]<X[I+1,J] THEN S:=S+1
   END;
            IF (Z = N-1) OR (S = N-1) THEN
                  BEGIN
                FOR I:=1 TO N DO
              IF NOT(PROS(X[I,J])) THEN F:=FALSE;
         IF F THEN
         BEGIN
               K:=K+1; R[K]:=J
         END;
   END;
    END;
END;
BEGIN
      WRITELN(‘ N және M-ді енгіз:’);
      READLN(N,M);
      WRITELN(‘Матрицаны енгіз:');
      FOR I:=1 TO N DO
            FOR J:=1 TO M DO
                  READLN(A[I,J]);
      FORM(A,N,M,V,K);
      WRITELN(‘Нәтиже:’);
      FOR I:=1 TO K DO
            WRITE(V[I],’ ‘);
      READKEY
END.
```

Бұл мысалда процедураға бастапқы мәндер беріледі: екі өлшемді жиым және оның көлемі. Жиым, процедурада оның

көшірмесіне орын бөлінбес үшін параметр-айнымалы ретінде беріледі. Нәтиже: вектор және оның көлемі міндетті түрде параметр-айнымалылар ретінде беріледі. Жай санды тексеру процедураның ішкі функциясы және оған негізгі программадан қол жеткізуге болмайды.

Параметр-тұрақтылар

Процедураға және функцияға берілетін аргументтер стекте сақталатын болғандықтан, көлемі үлкен жиым берілген кезде стек толып кетуі мүмкін. Турбо Паскаль тілінде ішкі программаның формальды параметрлерін сиаттауға болатын **CONST** сипаттаушысы бар. Осындай параметрге сәйкес аргумент, **Var** сипттаушысының параметрі тәрізді, сілтеме бойынша беріледі. Бірақ, процедураның (функцияның) өзінде оған жаңа мән меншіктеуге болмайды.

PROCEDURE < процедура аты> **(CONST** <тұрақты аты>: <типі>; **...);**

FUNCTION <функция аты**> (CONST** <тұрақты аты> : <типі> ; ...**):** <нәтиже типі> **;**

Параметр-тұрақтыны басқа ішкі программаға параметр ретінде беруге болмайды.

Параметр-процедура және параметр-функция

Процедуралық типті хабарлау үшін, процедура (функция) аты жазылмаған ішкі программа тақырыбы қолданылады.

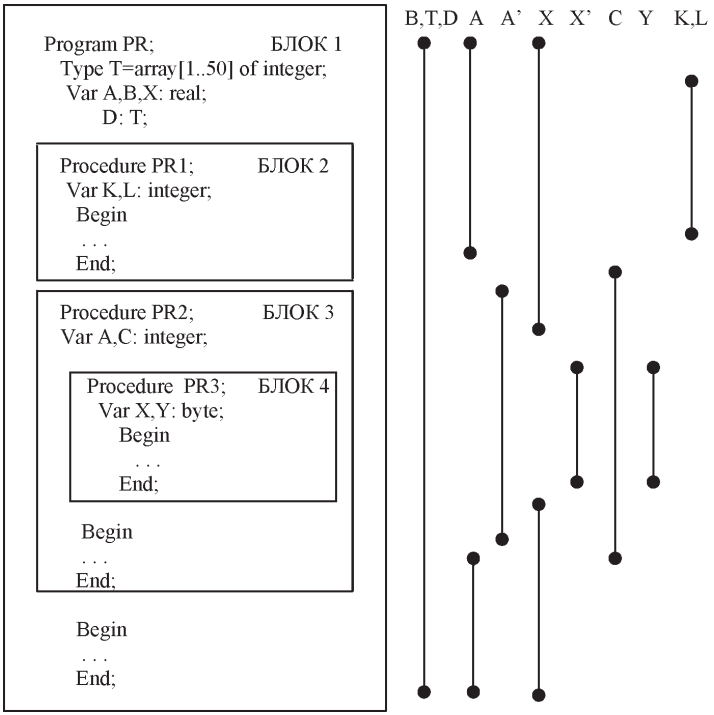
Мысалы:

```
TYPE
TPR1= PROCEDURE( X,Y : REAL; VAR Z : REAL);
TPR2= PROCEDURE ;
TF1= FUNCTION: STRING;
        TF2=FUNCTION ( VAR S: STRING) : REAL;
```

8.5 Атаулардың әсер ету аймағы

Кез келген ішкі программа өзінің сипаттау аймағы бар блок болып саналады. Осы блоктың ішінде басқа процедура және функцияны сипаттауға, шақыруы мүмкін. Қандай да бір блоктың ішінде сипатталған объектілер, осы блоктың **жергілікті** объектісі

болып саналады және оған басқа сыртқы блоктан қол жеткізуге болмайды. Оларға тек сипатталған блоктың ішінде ғана сілтеме жасауға болады. Объект – тұрақты аты, типтер, айнымалылар, процедуралар, функциялар. Сыртқы блокта сипатталған, бірақ ішкі блокта сипатталмаған объектілер **ауқымды** объекті деп аталады және оларды ішкі де, сыртқы да блоктан шақыруға болады. Жергілікті және ауқымды айнымалылардың аттары бірдей болса, жергілікті айнымалылар өз аумағында ауқымды айнымалылардың қызметін тоқтатады.



Суретте жеке идентификаторлардың қызмет аумағы схема түрінде көрсетілген:

- Y – 4 блоктың жергілікті айнымалысы, оған 1,2,3 блоктан қол жеткізуге болмайды.
- K, L – 2 блоктың жергілікті айнымалысы, оған 1,3,4 блоктардан қол жеткізуге болмайды.
- C – 3 блоктың ауқымды айнымалысы, 1 және 2 блоктан қол жеткізуге болмайды.

- B, D - 2,3,4 блоктардың ауқымды айнымалылары, 1,2,3,4 блоктардан қол жеткізуге болады.
- T - жалпы тип.

A идентификаторы әр түрлі екі айнымалыны білдіреді: A – қызмет аумағы 1 және 2 блоктар және A' айнымалысы – қызмет аумағы 3 және 4 блоктар. X атауы да дәл осы сияқты: X айнымалысы – қызмет аумағы 1, 2 және 3 блоктар және X' айнымалысы қызмет аумағы тек 4 блок.

Егер ішкі программада параметрлер бар болса, онда формальді параметрлер тізіміндегі идентификаторлар осы процедура (функция) үшін жергілікті параметр және процедураның (функция) ішіндегі әр ішкі программа (бар болған жағдайда) үшін аумақты параметр болып саналады.

8.6 Рекурсивті процедуралар және функциялар

Рекурсия - процедура немесе функция өз құрамына кіретін операторларды орындау барысында, өзін өзі немесе басқа процедура және функция арқылы өзін шақыратын есептеу процесін ұйымдастыру тәсілі.

Рекурсия **тікелей** немесе **жанама** болуы мүмкін. Тікелей рекурсия кезінде ішкі программаны шақыру операторы оның орындалу бөлімінде орналасады. Кез келген тікелей рекурсивті процедураны (функцияны) рекурсивсіз етуге болады. Рекурсивті сипаттама әдетте қысқы және көрнекті, бірақ көп машиналық уақытты (қайталап шақыру керек болғандықтан) және жадыны (айнымалылардың бірнеше рет қайталануынан) қажет етеді.

Рекурсивті ішкі программа бір рет сырттан шақырылады. Рекурсивті процедура немесе функцияның жұмысының аяқталау шарты оның өзінде жазылады.

Мысал қарастырайық. **F=M!** мәнін есептеу керек. Факториал мәнінің рекурсивті анықтамасын төмендегідей түрде жазуға болады:

M=1 болғанда F=1

M> 1 болғанда F= M!= M*(M-1)!, демек M! (M-1)! арқылы анықталады.

а) Рекурсивті функция

```
PROGRAM MAIN;
VAR   M: INTEGER;
```

```
        F:REAL;
FUNCTION FACT (N:INTEGER): REAL;
BEGIN
    IF  N=1 THEN
        FACT:=1
            ELSE
        FACT:= N* FACT(N-1);
END;
BEGIN
    READLN(M);
    F:= FACT ( M );
    WRITELN (‘ M!=’, F);
END.
```

b) Рекурсивті процедура

```
PROGRAM MAIN;
VAR   M: INTEGER;
        F:REAL;
PROCEDURE FACT(N:INTEGER; VAR F: REAL);
VAR Q : REAL;
BEGIN
    IF  N=1 THEN Q:=1
            ELSE  FACT(N-1,Q);
    F:=N*Q;
END;
BEGIN
    READLN(M);
    FACT ( M, F );
    WRITELN (‘ M!=’, F);
END.
```

Есептің а) нұсқасында M=4 болғанда келесі іс-әрекет орындалады:

FACT:=4*FACT(3); FACT:=3*FACT(2); FACT:=2*FACT(1); FACT(1):=1.

Функция бірінші рет шақырылғанда жадыдан параметр-мәнге сәйкес жергілікті айнымалылар үшін орын бөлінеді. Оған нақтылы параметрдің (M) мәні меншіктеледі. Функцияға әр кірген сайын жаңа жергілікті айнымалылар құрылады. FACT(1)=1

болғандықтан функцияның орындалуы тоқтатылады. Бұдан кейін келесі іс-әрекеттер орындалады:

FACT(1):=1; FACT:=2*FACT(1);
FACT(2):=2; FACT:=3*FACT(2);
FACT(3):=3*2=6; FACT:=4*FACT(3); т.е. FACT=24.

8.7 Модуль құрылымы. Модульді іске қосу

Процедуралар мен функцияларды жеке *модульге* топтастыруға болады. *Модуль* (**unit**) – мәтіні компиляциядан жеке (негізгі программаға байланыссыз) өтетін программалық бірлік. Егер модуль компиляциядан нақты режимде өтсе, онда оның кеңейтілуі TPU болады. Қорғалған режим үшін – TPP.

Модуль құрылымы Турбо Паскаль тіліндегі программа құрылымынан өзгеше. Модуль төрт негізгі бөлімнен тұрады: тақырып - **UNIT** қызметші сөзінен кейін жазылады; сипаттаушы бөлім (интерфейс) - **INTERFACE** қызметші сөзінен кейін жазылады (бұл жерде айнымалы, процедура, функция, тұрақты және осы модуль қолданатын басқа программалық модульдер қол жеткізе алатын мәндер типі сипатталады); орындалатын бөлім (ішкі) – **IMPLEMENTATION** қызметші сөзімен асталады (мұнда программа мәтіні және тек осы модуль ішінде қол жеткізуге болатын жергілікті объектілер жазылды); міндетті емес бөлім (инициализациялау секциясы) – орындалатын бөлімнен кейін **BEGIN** және **END** сөздерінің арасында жазылады (егер модульді инициализациялау қажет емес болса, онда секцияға тек **END** қызметші сөзі жазылады).

Модульдің ішкі программаларын сипаттаған кезде қысқартылған тақырыптарды қолданған дұрыс (параметрсіз және функция үшін нәтиже типін көрсетпей), мысалы, FORWARD директивасын қолданғанда. Модуль UNIT қызметші сөзінен және модуль атынан тұратын тақырыптан басталады. Модуль аты міндетті түрде өзі тұрған файл атымен (PAS кеңейтілімі бар) сәйкес келу керек. Модуль құрылымы төмендегідей болып келеді:

UNIT <модуль аты>**;**
INTERFACE
USES <қосылатын модульдер тізімі>**;**
TYPE <осы модульде анықталып, басқа модульдерден қол жеткізуге болатын типтердің сипаттамасы >**;**

CONST < осы модульде анықталып, басқа модульдерден қол жеткізуге болатын тұрақтылардың сипаттамасы >**;**

VAR < осы модульде анықталып, басқа модульдерден қол жеткізуге болатын айнымалылардың сипаттамасы >**;**

PROCEDURE <осы модульде анықталып, басқа модульдерден қол жеткізуге болатын процедуралар тақырыбы **>;**

FUNCTION < осы модульде анықталып, басқа модульдерден қол жеткізуге болатын функциялар тақырыбы >**;**

IMPLEMENTATION

USES < қосылатын модульдер тізімі >**;**

TYPE < осы модульде анықталып, басқа модульдерден қол жеткізуге болмайтын типтердің сипаттамасы **>;**

CONST < осы модульде анықталып, басқа модульдерден қол жеткізуге болмайтын тұрақтылардың сипаттамасы >**;**

VAR < осы модульде анықталып, басқа модульдерден қол жеткізуге болмайтын айнымалылардың сипаттамасы **>;**

PROCEDURE < осы модульде анықталып, басқа модульдерден қол жеткізуге болатын процедураларды іске асыру **>;**

FUNCTION < осы модульде анықталып, басқа модульдерден қол жеткізуге болатын функцияларды іске асыру **>;**

PROCEDURE < осы модульде анықталып, басқа модульдерден қол жеткізуге болмайтын процедуралардың тақырыбы және іске асыру **>;**

FUNCTION < осы модульде анықталып, басқа модульдерден қол жеткізуге болмайтын функциялардың тақырыбы және іске асыру **>;**

BEGIN <бұл түйінді сөз инициализациялау секциясында операторлар бар болғанда ғана қажет >

<Модульдің міндетті емес бөлімі>

END.

Интерфейстік және орындалу бөлімдері бос болуы мүмкін, міндетті түрде модуль құрамында болуы керек. Модуль іске қосылғанда алдымен инициализациялау секциясының операторлары (бар болса) орындалады, сонан кейін осы модуль қосылған бас программаның негізгі блогының операторлары орындалады.

Мысал қарастырайық. Вектор көлемі және оның элементтері енгізіліп, бүтін сандардан тұратын бір өлшемді жиымның элементтерін өсу реті бойынша сұрыптайтын процедураны шақы-

ратын негізгі программа жазу керек. Жиым элементтерінің саны 100-ден аспауы тиіс. Процедураны модуль түрінде ұйымдастырайық.

```
USES CRT,MODSORT;
VAR A:MAS;
      I:BYTE;
      N:BYTE;
 BEGIN
      WRITELN('ВВОД ИСХОДНЫХ ДАННЫХ:');
      READLN(N);
      FOR I:=1 TO N DO
           READLN(A[I]);
      SORT(A,N);
      FOR I:=1 TO N DO
           WRITELN(A[I]);
      READKEY
END.
```

Программаның бірінші сөйлемі **Uses.** Мұнда стандартты **Crt** модулі және сұрыптау процедурасы жазылған **Modsort** модулі көрсетілген. Жиымды сипаттайтын тип негізгі программада емес модульде көрсетілген.

```
UNIT MODSORT;
INTERFACE
TYPE MAS=ARRAY[1..100] OF INTEGER;
PROCEDURE SORT(VAR A:MAS; N:BYTE);
IMPLEMENTATION PROCEDURE SORT;
VAR I,J:BYTE;
     X:INTEGER;
BEGIN
     FOR J:=1 TO N-1 DO
          FOR I:=1 TO N-J DO
               IF A[I]>A[I+1] THEN
               BEGIN
                    X:=A[I]; A[I]:=A[I+1]; A[I+1]:=X
               END;
END;
END.
```

Модульдің интерфейстік бөлімінде **mas** типі сипатталған және сұрыптау процедурасының тақырыбы көрсетілген. **Uses** сөзінің көмегімен модуль іске қосылған кезде процедура мен типтерге кез келген программадан қол жеткізуге болады. Бұл негізгі программада көрсетілген.

Бақылау сұрақтары

1. *Ішкі программа дегеніміз не?*
2. *Параметрсіз процедуралар қалай сипатталады?*
3. *Ішкі программаның Pascal программасынан негізгі айырмашылығын атаңыз.*
4. *Параметрсіз процедуралар қалай шақырылады?*
5. *Кіріктірілген ішкі программалар дегеніміз не?*
6. *Ауқымды және жергілікті айнымалылар дегеніміз не?*
7. *Ішкі программаны қолданудың артықшылықтарын атаңыз.*
8. *Процедура мен функцияны сипаттаудың негізгі айырмашылықтарын атаңыз.*
9. *Программада функцияны шақыру қалай іске асырылады?*
10. *Қандай жағдайларда параметрсіз процедураны қолданған ыңғайлы?*
11. *Параметрлі процедуралардың артықшылығы неде?*
12. *Нақтылы және формальды параметрлердің айырмашылығы неде?*
13. *Формальды параметрлер тобы қалай сипатталады? Нақтылы параметрлер тобы ше?*
14. *Параметр-мәндерді берудің ережесі қандай?*
15. *Параметр-айнымалыларды берудің ережесі қандай?*
16. *Параметр-тұрақтыларды берудің ережесі қандай?*
17. *Бір ішкі программада әр типті бірнеше параметр-айнымалыны қолдануға бола ма?*

Тапсырмалар

1. *Экранға биіктігін 8 жол етіп өз атыңызды шығарыңыз. Әр әріптің бейнесі жеке процедура көмегімен жазылуы керек және түстері әртүрлі болсын.*
2. *Түрлі стильдегі, түсі және қалыңдығы әртүрлі сызықтарды қолданып экранға өзіңіздің фамилияңыздың, атыңыздың және тегіңіздің бас әріптерін шығарыңыз. Әр әріптің бейнесі жеке процедура көмегімен жазылуы керек.*
3. *Экранға көлемдері бірдей әр түсті 7 жұлдыз тізбегін салыңыз.*
4. *Калькулятор программасын жазыңыз. «+», «-», «*», «/» операцияларының орындалуын жеке процедура ретінде құрастырыңыз.*

5. *10 бүтін саннан тұратын жиымды өңдейтін программа жазыңыз. Жиымның элементтерінің қосындысын, арифметикалық ортасын және ең үлкен элементін анықтаңыз. Программаның әр логикалық блогын процедура ретінде құрастырыңыз.*
6. *15 әр түсті дұрыс сегізбұрыштардан тұратын экран «диагоналін» сызыңыз.*
7. *36 әр түсті дұрыс бесбұрыштардан тұратын дөңгелек суретін салыңыз. Кез келген пернені басқанда программа өз жұмысын аяқтайтын болсын.*
8. *Әр түсті және әр түрлі өрнектермен өрнектелген, көлемдері бірдей алтыбұрыштардан құрастырылған пирамида салыңыз. Пирамиданың ең төменгі қатары 20 алтыбұрыштан тұрады.*
9. *Үшбұрыштардан құрастырылған бірнеше шыршалар қатарынан тұратын орман суретін салыңыз.*
10. *Олимпиада жалауын салыңыз. Жалаудың әр дөңгелегі 36 шеңберден құрастырылсын.*
11. *Көлемі 20х20 болатын әртүсті тікбұрышты үшбұрыштардан құрастырылған тор суретін салыңыз.*

Төмендегі тапсырмалардың барлығында параметрлі процедураны қолданыңыздар.

1. *Әртүсті 15 алтыбұрыштан тұратын экран "диоганалін" салыңыз.*
2. *Әртүсті 10 ромбтан тұратын тізбек салыңыз.*
3. *Енгізілген сөз тіркесіндегі бос орындар санын анықтайтын программа құрыңыз.*
4. *Бүтін сандағы цифрлар санын анықтайтын программа құрыңыз.*
5. *Сағат, минут және секундпен берілген уақытты толығымен секундқа айналдыратын программа құрыңыз.*
6. *Пернетақтадан енгізілген катеті бойынша үшбұрыштың гипотенузасын, ауданын, периметрін есептейтін программа құрыңыз.*
7. *Іштері боялған әртүрлі сегізбұрыштардан құрастырылған, көлемі 8х8 тор суретін салыңыз.*
8. *Іштері боялған әртүрлі кіші ромбтардан үлкен ромб құрастырыңыз. Үлкен ромб диагоналі 10 кіші ромбтан тұрады.*
9. *Ішінде өз атыңыз жазылған тікбұрыш салыңыз. Координаталар мен түсті процедура параметрі ретінде беріңіз. Осы процедураны параметрлерді өзгертіп бірнеше рет шақыру керек.*
10. *Іші боялған жұлдызшалармен синусоида салыңыз.*
11. *Экранға іші боялған дұрыс көпбұрыш салыңыз. Көпбұрыштың бұрыштарының саны, түсі және толтыру стилі пернетақтадан енгізілсін.*

12. Келесі пернелерді басқанда сәйкес есептеулерді орындайтын программа құрыңыз. F1 – шеңбердің ұзындығын есептеу;
F2 – дөңгелектің ауданын есептеу;
F3 – шардың көлемін есептеу;
F5 - программадан шығу.

Төмендегі тапсырмалардың барлығында функцияны қолданыңыздар.

1. *a,b,c,d нақты сандары үшін m(a,b,c)+m(b,c,d)+m(c,d,a) өрнегінің мәнін анықтаңыз. Мұндағы m(x,y,z) – үш санның ең кішісін анықтайтын функция.*
2. *x,y нақты сандары берілген. u = min(x,y), v = min(xy,x+y), z = min(u+v², 3.14) анықтаңыз.*
3. *a, b, c нақты сандары берілген.* $\frac{\max(a,a+b)+\max(a,b+c)}{1+\max(a+bc,1.15)}$ *есептеңіз.*
4. *Нақты s, t сандары берілген.* $g(1.2,-s)+g(t,s)-g(2s-1,st)$ *есептеңіз.*

Мұндағы $g(a,b)=\frac{a^2+b^2}{a^2+3ab+3b^2+4}$

5. *Нақты s, t сандары берілген.* $f(t,-2s,1.7)+f(2.2,t,s-t)$ *есептеңіз.*

Мұндағы $f(a,b,c)=\frac{2a-b-\sin(c)}{5+|c|}$

6. *Нақты a,b,c берілген.* $g(a,c)*\frac{g(a,b)}{g(b,c)}$ *есептеңіз.*

Мұндағы $g(x,y)=\frac{\cos(x+y)}{\sin(x+y)}*\frac{\cos(x-y)}{\sin(x-y)}$

7. *Бүтін типті 10 элементтен тұратын жиым берілген.* $\frac{t(mas)+98}{t(mas)-98}*t(mas)$ *есептеңіз. Мұндағы* $t(mas)$ *- жиым элементтерінің арифметикалық орташасы.*
8. *a, b сөз тіркестері берілген.* $\max(f(a),5)*\max(f(b),10)$ *есептеңіз. Мұндағы* $f(s)$ *- сөз тіркесіндегі бос орындар саны.*
9. *Градусты радианға айналдыратын функция жазыңыз.*
10. *Тәулікпен, сағат, минут, секундпен берілген уақытты секундқа айналдыратын функция жазыңыз.*

9. ФАЙЛДАРМЕН ЖҰМЫС ІСТЕУ

9.1 Файлдар жайлы жалпы мәліметтер

Мәліметтердің көлемді жиынын, мысалы, жұмысшылар туралы деректерді немесе бір қызметкерге қатысты ақпараттар жиынын (жалақы, стаж, адрес, отбасы құрамы т.б.) тұрақты пайдаланып отыру үшін магниттік дискіге жазып қойған тиімді. Магниттік диск немесе таспа (лента) компьютердің жедел жадына симайтын ақпараттың өте көлемді түрлерін сақтай алады. Алдын ала жазылып қойылған мәліметтер жиыны оны пайдаланатын программамен жұмыс істеу барысында компьютерге қосылған магниттік дискіден біртіндеп оқыла бастайды. Сондай-ақ, программа жұмысының нәтижесін де дискіге жазып қойып, қажет кезінде баспаға немесе дисплейге шығаруға болады.

Паскальда компьютерде мәлімет өңдеу кезінде ақпараттар ЭЕМ-нің сыртқы жадына (дискіге) жазылып сақталады. Дискідегі ақпараттармен жұмыс істеу үшін арнайы объектілер – файлдар қолданылады. *Файл деп тізбектеле орналасқан жеке компоненттерден тұратын дискідегі біртектес мәліметтердің реттелген жиынын айтады.* Файл – белгілі бір мақсатқа жету жолында өзіндік (біркелкі) ұйымдастыру жолымен бір-бірімен логикалық байланыста құрастырылған мәліметтердің (жазбалардың) атаулы жиыны. Осы уақытқа дейін программада тек INPUT және OUTPUT тәрізді стандартты файлдар пайдаланылды. Файлдағы компоненттер саны алдын ала берілмейді және олардың жиымдар (массивтер) сияқты индексі болмайды. Файлдың ортасындағы компонентті оқу үшін алдыңғы компоненттерді біртіндеп ретімен оқи отырып жетуге болады. Жалпы жағдайда файл программадағы айнымалы түрінде беріледі. Файлды әрбір пайдалану кезінде программаға тек бір компонент оқылады немесе жазылады.

Физикалық жазба – сыртқы жадымен бір рет қатынас құру (пайдалану) кезінде оған жазылатын немесе одан оқылатын мәліметтер жиыны, яғни сыртқы жады мен жедел жады арасында тасымалданатын мәліметтердің ең кіші (минимальді) көлемі. Физикалық жазба бірнеше логикалық жазбалардан тұрады.

Логикалық жазба – файлдардан мәлімет оқу немесе жазу операторларында пайдаланылатын мәліметтер бірлігі. Логикалық жазбалар сыртқы құрылғыға жиі-жиі мәлімет жазбас (оқымас) үшін көлемдірек физикалық жазбаларға біріктіріледі.

Сыртқы құрылғыдағы (магниттік дискідегі) файл жазбаларын пайдалану үшін логикалық файл ұғымы қолданылады. *Логикалық файл* немесе программадағы файл – бұл белгілі бір мақсатқа жету жолында біріктірілген логикалық жазбалардан тұратын мәліметтер жиыны.

Файлдарды сипаттау немесе жариялау олардың айнымалы түрінде берілген атынан соң **file of** түйінді сөздерінен тұрады. Түйінді сөздерден кейін оның компоненттерінің типі беріледі. Мысалы:

```
Type
     stroka:  file of char;
     number: file of integer;
```

`Assign` түйінді сөзі программадағы файл мен сыртқы құрылғыдағы файлды байланыстыру үшін қолданылады, мысалы:

```
ASSIGN (F, 'ALMA.TXT')
```

Бұл процедура дискідегі `ALMA.TXT` файлын программадағы `F` атты файлмен байланыстырады, программада ары қарай бұл файлды пайдалану үшін жай ғана `F` деп көрсету жеткілікті.

Файлдағы жазбалар саны әр түрлі бола береді, бірақ кез келген сәтте тек бір ғана жазбаны пайдалана аламыз. *Файл ұзындығы* деп оған жазылған компоненттер санын айтамыз. Ішінде жазбасы жоқ файл – *бос файл* болып табылады.

Файлдық типтегі әрбір айнымалы программаның **VAR** сипаттау бөлігінде жариялануы тиіс. Мұндай айнымалыны өрнектерде және меншіктеу операторларында пайдалануға болмайды. Файл компоненттері типі файлдық типтен басқа кез келген тип бола алады.

Турбо Паскальда алдын ала анықталған төмендегі стандартты тип бар:

```
TYPE TEXT = FILE OF CHAR;
```

Жалпы Паскаль программалау жүйесінде файлдардың 3 түрі қолданылады:

- жазбалар типіндегі файлдар (типтелген файлдар);

- ұзындықтары белгісіз сөз тіркестерінен тұратын мәтіндік файлдар;
- жазбалар блоктары түріндегі мәліметтерді тасымалдау (жазу немесе оқу) үшін қолданылатын типсіз файлдар.

Файлдармен жұмыс істеу барысында төмендегі ережелерді есте сақтаған жөн:

- қолданылатын барлық файл аттары программа тақырыбында көрсетілуі мүмкін;
- мәтіндік файлдар **TEXT** типімен сипатталуы тиіс**;**
- программадағы әрбір файл сыртқы құрылғыда орналасқан нақты бір файлмен сәйкес келуі керек, олардың сәйкестігі **ASSIGN** процедурасы арқылы тағайындалуы тиіс**;**
- белгілі бір файлда бұрын жазылған мәліметті оқу, түзету немесе толықтыру үшін **RESET** процедурасы қолданылады, мәлімет жазу үшін жаңадан файл ашу ісі – **REWRITE** процедурасы арқылы орындалады;
- файлмен жұмыс істеу аяқталған соң, оны **CLOSE** процедурасымен жабу қажет.

Файл деп көбінесе компьютердің сыртқы жадында – дискеттерде, винчестерлерде, CD-лерде немесе олардан басқа есте сақтау құрылғыларында және де енгізу-шығару құрылғыларында орналасқан белгілі бір ат қойылған мәліметтер тізбегін (файл компоненттерін) айтады. Файлдарда мәтін, программа, сандық мәліметтер, графикалық бейнелер және т.с.с. ақпараттар түрлері сақталады. Файлдарды пайдадалану ісін ұйымдастыру барысында Паскаль программалары MS DOS операциялық жүйесімен бірігіп отырып жұмыс жасайды.

MS DOS файлдық жүйесі

Мәтіндер мен программаларды тұрақты сақтауға арналған дискідегі мәліметтер жиыны файл деп аталады. Операциялық жүйе жұмысы үшін керекті мәліметтер рөлін әртүрлі типтегі файлдар атқарады. Дискіде аттары әртүрлі көптеген файлдар сақталады.

Файлдар аттары MS DOS жүйесінде төмендегі талаптарға сәйкес келуі тиіс:

- файл аты – бірден сегіз таңбаға дейінгі латын әрпінен басталатын сандар мен латын әріптері жиыны;
- файл атының үш таңбадан аспайтын екінші бөлігі типі болып саналады, ол файл атынан нүктемен бөлінеді. Типті файл атының кеңейтілуі деп те айтады, тип кейде болмауы да мүмкін;
- файл аты мен типін жазу үшін латынның бас және кіші әріптері, араб цифрлары, кейбір арнайы символдар, мысалы, астын сызу таңбасы «_» немесе доллар белгісі «$» пайдаланылуы мүмкін;
- файл аты ретінде MS DOS операциялық жүйесінде кейбір құрылғылар аттары ретінде пайдаланылатын сөз тіркестерін пайдалануға болмайды, олар: PRN, CON, NUL, COM1, COM2, AUX, LPT1, LPT2, LPT3.

Windows типтес операциялық жүйелерде бұл заңдылықтар сақталмайды, мысалы, файл аты 255 символға дейін ұзартылған және орыс, қазақ әріптері де қолданыла береді. Бірақ Паскаль программалау тілінде MS DOS ережелерін толық сақтаған жөн.

Әдетте, файл аты оның мазмұнымен сәйкес етіп құрылады, ал оның 3 таңбадан аспайтын кеңейтілуі сақталатын мәліметтер типін (Windows-те суреті, белгішесі) көрсетеді. Файлдарды кеңейтілуі бойынша айырып тануға мүмкіндік беретін MS DOS операциялық жүйесінің мына төмендегідей стандартты типтері бар:

.COM, .EXE - тікелей атқарылуға арналған командалық файлдар;

.DAT - мәліметтер файлы;

.DOC - Word программасында дайындалған құжаттық файл;

.PAS .BAS .CPP – Паскаль, Бейсик, C++ тілдерінде жазылған программалық файлдар;

.TXT - Блокнот программасында дайындалған мәтіндік (текстік) файл.

Файлдың толық атының мысалдары:

COMMAND.COM NORTON.EXE
TANIA7.DOC Arman2.txt
START Game.BAS AIGUL.CPP MARAT.doc TEXT1

Файлды дискіге жазғанда немесе өзгерткенде оның көлемі, жазылған уақыты, мерзімі де тіркеледі. Файлдың *аты, типі, байтпен берілген көлемі*, операциялық жүйенің календары мен сағатынан алынып жазылған *күні, айы, сағаты* **файлдың атрибуттары** (көрсеткіштері) деп аталады. Олар файл жазылғанда, өзгертілгенде немесе көшірілгенде тіркеліп отырады.

Кейбір командаларды орындау барысында файлдардың атын толық көрсетпеуге болады, мұндайда MS DOS жүйесі файлдар тобын іздейді.

Кейде файлдардың біреуін ғана емес, қатар орналасқан бірнешеуін топтауға тура келеді. Мұны файлдың атына шаблондарды немесе нұсқаларды, яғни «*» және «?» белгілерін қою арқылы істеуге болады.

* - файлдың атындағы кез келген сөз тіркестерін білдіреді;

? - файлдың атындағы болуы мүмкін кез келген таңбаны немесе оның болмауын да көрсетеді.

Мысалы:

1 KOL* - аты KOL әріптерінен басталатын барлық файлдар тобын көрсетеді (kolia.txt, kolem.doc, kol7.bas, KOL.PAS, KOLKA, т с.с.);
2 *.EXE - типі EXE болып келген барлық файлдар тобы (alma.exe, talon.exe, kolia.e, a141.exe, beta.exe ...);
3 *.* - жұмыс істеп отырған ағымдағы (үстіміздегі) каталогтағы барлық файлдарды көрсетеді;
4 * - типі көрсетілмеген файлдар тобын көрсетеді (yakitsha, alia, ...);
5 ???.BAS - типіBAS болып келген аты бір, екі және үш таңбадан тұратын файлдар тобы (ali.bas, a.bas, t1.bas, tor.bas, olia.bas ...);
6 A?.* - аты A таңбасынан тұратын және A әрпінен басталып екі таңбадан тұратын типі кез келген сөз болатын файлдар тобы (a.txt, a.com, as.doc, a2.bas, ad.pas ...).

MS DOS дискетке (винчестерге де) мәліметтер жазардан бұрын ол арнайы тәсілмені белгіленуі (форматталуы) керек. Белгілеу операциялық жүйенің арнайы командасымен орындалады. Бұрын пайдаланылған диск форматталса, ондағы барлық мәлімет толығымен өшіріледі де, кейіннен қалпына келтірілмейді.

Дискілерде көптеген файлдар сақталады, мысалы, винчестерде мыңдаған файлдар болады. Оларды кітапханадағы тәрізді каталогтарға (бумаларға) бөліп, бірінің ішіне бірі орналасқан каталогтар **бұтақ тәрізді көп сатылы (иерархиялы) файлдар жүйесін** құрайды. Дискіні форматтағанда, онда мәліметтер жазылатын **басты** немесе **түпкі** каталогы ашылады, оның аты болмайды. Келесі деңгей каталогы міндетті түрде белгілі бір атаумен белгіленеді.

Бір каталогқа аттары бірдей екі файлды орналастыруға болмайды, оның соңғысы жазыларда алдыңғысы бірден жойылып кетеді, бірақ аттас файлдар әртүрлі каталогтарда орналаса береді.

Бір каталог екінші бір каталогтың ішінде орналаса береді, мұндайда ол бағынышты атанып, сатылы түрде бұл каталогқа басқа бір каталог бағынышты болып орналасуы мүмкін. Осындай түрде бұтақ тәрізді файлдар жүйесі құралады. Каталогтарды құру және өшіру арнайы командалар арқылы орындалады.

Файлдар бұтақ тәрізді күрделі түрде құрылса, онда оның тек өз атын ғана емес, сонымен қатар орналасқан каталогын, оның түпкі каталогпен байланысын, диск атын көрсету қажет. Мұндай файлдың нақты тұрағын (адресін) көрсететін каталогтар тізбегін - **маршрут** немесе **жол** деп те атайды. Сонымен, файл толық түрде мынадай элементтермен беріледі:

1) дискінің аты, ол бірақ кейде көрсетілмей де кетеді;

2) бірінің ішіне бірі кіретін каталогтар тізбегінің аттарынан тұратын маршрут (ол да кейде көрсетілмей кетеді);

3) тізбекті аяқтайтын файлдың өз аты (файл тізімдегі ең соңғы каталогта орналасады).

Уақыттың әрбір мезетінде бір диск ағымдағы екпінді күйде болады, оның айғағы MS DOS шақыруы, яғни жұмыс істеп тұрған диск мен файлдар маршруты атының көрсетілуі.

Бағынышты каталогтар тізбегі бір-бірінен «\» таңбасымен бөлініп жазылады, мысалы,

C:\DEMO> - DEMO C: дискісіндегі 1-деңгейлі каталог;

C:\DEMO\SYSTEM> - SYSTEM C: дискісіндегі 2-деңгейдегі каталог. Әрбір мезетте бір каталог жұмыс атқарады да, ол екпінді немесе активті (ағымдағы) деп аталады. Компьютер іске қосылғанда мұндай каталог рөлін түпкі каталог атқарады неме-

се компьютер өшірілер кездегі екпінді болған каталог атқарады, программалаушы өз қалауынша командалар көмегімен басқа каталогты екпінді ете алады.

Мысалы:

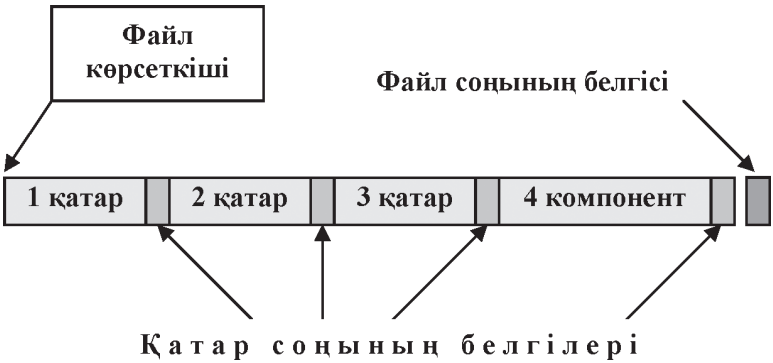
A:> немесе C:\>

C:\DEMO\VICTOR> немесе C:\SYS\AZAT \ALMA>

Сонымен файлдың маршрутын немесе адресін толық көрсету үшін дискінің атын, каталог аттарының тізбегін, ең соңында файлдың атын және типін толық көрсету қажет. Жалпы ереже бойынша файлдың толық аты төмендегідей болады.

[дискінің аты:\] [маршрут\] файлдың аты[.типі]

Тік (квадрат) жақшаның ішіндегі сөздер көрсетілмеуі де мүмкін.



9.1-сурет. Файл құрылымынан мысал

9.2 Файлдармен жұмыс істеуге арналған процедуралар мен функциялар

Файлдармен жұмыс істеу үшін төменде көрсетілген стандартты функциялар мен процедуралар қолданылады.

ASSIGN (<файл аты>**,**<сыртқы құрылғыдағы файл аты>**) –** бұл процедура файлдық айнымалы мен сыртқы құрылғыдағы файл атын байланыстырады. Мұндағы <файл аты> – бұл файлдық айнымалы, яғни программада файлдық типте жарияланған дұрыс идентификатор. <Сыртқы құрылғыдағы файл аты> – файл аты болып табылатын мәтіндік өрнек немесе логикалық құрылғы

аты. Құрылғыдағы файл аты алдына сол файлдың тұрған орнын бейнелейтін жолы, яғни маршруты – диск аты және (немесе) ағымдағы каталог аты және оның жоғарғы деңгейіндегі каталогтар аттары көрсетілуі мүмкін.

Мысалы, **`assign(F,'A:\GF')`** процедурасы – A: дискісіндегі `GF` файлын программадағы файл аты болып саналатын `F` айнымалысымен байланыстырады, программада ары қарай бұл файлды пайдалану үшін жай ғана **`F`** деп көрсетсе жеткілікті.

RESET(<файл аты>**)** – *тізбекті қатынасу кезінде* мәлімет оқу үшін бұрыннан бар файлды ашу процедурасы, ал *тікелей қатынасу кезінде* бұрыннан бар файлдан мәлімет оқу, әрі оған мәлімет жазу процедурасы. Мұнда файл көрсеткіші бірінші жазбаға орналасады (оның нөмірі – 0). Басқаша айтқанда, мысалы, RESET(F) – файлды бастапқы жағдайға келтіру болып табылады. Бұл процедура F айнымалысына сәйкес келетін файл көрсеткішін оның ең басына алып келеді. Файл бұдан бұрын ашылуы тиіс.

REWRITE(<файл аты>**)** – мәлімет жазылатын жаңа файл ашу процедурасы. Егер осындай атты файл бұрыннан бар болса, онда ол өшіріледі. Файл көрсеткіші бірінші жазба алдына орналасады. Мысалы, rewrite(F) процедурасы бойынша дискіде F деген атпен құрамында ешбір компоненті болмайтын жаңа бос файл ашылады да, файл көрсеткіші оның алғашқы компоненті болатын орынды көрсетіп тұрады.

READ(<файл аты>, <айнымалы>) – типі файл компоненттерінің типімен сәйкес келетін айнымалыға файлдың келесі компоненттерін оқу процедурасы. Мұнда файл көрсеткіші оқылған компоненттер санына байланысты ары қарай ығысады. Мысалы, READ(F, v_1, v_2, …, v_n) – мұндағы F – файлдық айнымалының аты, v_i – файл компоненттерінің типіне сәйкес келетін айнымалылар атауларының тізімі. Әрбір айнымалының мәні F файлының соған сәйкес компонентіне тең болады да, әрбір мәнді оқығаннан кейін көрсеткіш келесі компонентке ауысады.

WRITE(<файл аты>, <айнымалы>) – айнымалы мәндерін файлдың ішкі көрсеткішіне сәйкес аты көрсетілген файлға жазу процедурасы. Мұнда да файл көрсеткіші жазылған компоненттер санына байланысты ары қарай ығысады. Мысалы, write (F, v_1, v_2, …, v_n) арқылы файлға жазу кезінде әрбір айнымалының мәндері

(v_1, v_2, …, v_n) файлдың компоненті түрінде тізбектеле жазылады, одан кейін файлдың көрсеткіші келесі компонентке ауысады.

SEEK(<файл аты>**,** < компонент нөмірі>**)** – файлдың ағымдағы ішкі көрсеткішін керекті файл компонентіне орналастыру процедурасы. Файл жазбаларымен тікелей қатынасу ісін ұйымдастыру кезінде қолданылады. Мысалы, SEEK(F, N) – F файлының ішіндегі N-нөмірлі компонентін іздеп табу. Файл көрсеткіші F файлының N-нөмірлі компонентіне ауысады, мұндағы N – бүтін сан түріндегі өрнек. Бұл процедура мәтіндік файлдар үшін қолданылмайды.

CLOSE(<файл аты>**)** – файлды жабу процедурасы. Жаңа файл ашылған соң, міндетті түрде пайдаланылуы тиіс, әйтпесе мәліметтер жоғалып кетуі мүмкін. Мысалы, CLOSE(F) процедурасы F файлымен программаның мәлімет алмасу мүмкіндігін тоқтатады.

ERASE(<файл аты>**)** – файлды өшіру процедурасы. Ашылған файл бұған дейін жабылуы тиіс. Мысалы, ERASE(F) процедурасы дискідегі F файлын өшіреді. Өшірместен бұрын ол файлды жабу керек.

RENAME(<ескі файл аты>**,** <жаңа файл аты>**)** – файл атын өзгерту процедурасы. Файл жабылған соң қолданылады.

IORESULT – соңғы енгізу-шығару операциясының шартты белгісін қайтару функциясы. Егер операция дұрыс аяқталса, функция нөл мәнін қайтарады. Бұл функция енгізу-шығару қателерін автоматты түрде бақылауды алып тастағанда ғана жұмыс істейді. Компилятордың {$I-} директивасы оны алып тастайды, ал {$I+} – қателерді автобақылау амалын іске қосады. Егер автобақылау іске қосылмай тұрғанда, енгізу-шығару операциясы қате шығуына себепші болса, онда қате жалаушасы орнатылады да, одан кейінгі барлық енгізу-шығару операциялары **IORESULT** функциясы шақырылмағанша, орындалмайды.

EOF(<файл аты>**)** – end of file «файлдың соңы» функциясы, ол аты көрсетілген файлдың (мысалы, F) ең соңына жеткеніміздің көрсеткіші болады. Егер файл көрсеткіші оның ең соңында тұрса, яғни ең соңғы компонент пайдаланылған болса, EOF логикалық функциясының мәні – true, ал файл соңына жетпесек, false болады.

EOLN(<файл аты>) – мәтіндік файлдағы бір жолдың, яғни қатардың соңын анықтайтын функция. Жолдың соңына жетсек, оның мәні – **TRUE**, әйтпесе **FALSE** болады.

FILEPOS(<файл аты>) – файлдың ағымдағы жазбасының (компонентінің) нөмірін анықтау функциясы. Мысалы, FILEPOS(F) – F файлының көрсеткішінің нөмірін анықтайды.

FILESIZE(<файл аты>) – файлдың көлемін, яғни ондағы компоненттер санын анықтайтын функция. FILESIZE(F) – ағымдағы F файлының көлемін, яғни ондағы компоненттер санын анықтайды.

Жұмыс істеу барысында файлды қолданбастан бұрын ASSIGN процедурасы арқылы оған белгілі бір айнымалы түрінде ат беріледі. Файлға мәлімет жазбастан немесе одан оқымастан бұрын міндетті түрде оны REWRITE немесе RESET процедуралары арқылы құру немесе ашу қажет. Бұл процедуралармен жұмыс істеудің нәтижесінде файлдың көрсеткіші оның бірінші компонентіне орналасады, яғни FILEPOS(F)=0. Дискілік файлдың көлемін тек оның соңына компонент қосу жолымен ғана кеңейтуге болады. Көрсеткішті файлдың соңына қарай SEEK(F, FILESIZE(F)) процедурасының көмегімен жылжытуға болады. Файлмен жұмыс істеп болғаннан кейін оны міндетті түрде CLOSE процедурасымен жабу керек.

Турбо Паскаль файлдары

Турбо Паскаль тілінде файл бір типке жататын компоненттер тізбегі ретінде анықталады, олар: жазбалар файлы, бүтін сандар файлы, сөз тіркестері файлы және т.с.с. Файлдың мәліметтердің басқа құрылымдық типтерімен салыстырып қарағандағы ерекшелігі – оның кез келген сәтте тек бір ғана компонентіне қол жеткізуге болатындығы жатады. Файл компоненттерінің саны алдын ала анықталмайды. Компьютердің сыртқы жадында орналасқан файл көлемінің ең үлкен (максимальды) шамасы тек есептеу жүйесінің техникалық мүмкіндіктерімен ғана шектеледі.

Жалпы дискілік файлдар және логикалық құрылғылар пайдаланылады.

Дискілік файл компьютердің сыртқы жадының, мысалы, дискеттің немесе винчестердің *белгілі бір ат қойылған аймағы* болып табылады. Файлдармен орындалатын енгізу-шығару опе-

рациялары, физикалық тұрғыдан алғанда, арнайы буферді пайдаланады. Мысалы, файлға жазылатын мәліметтер алдымен буферге орналасады да, ол толған кезде барып файлға жазылады. Ал файлдан оқылған жазбалар да алдымен буферге орналасады да, кейіннен сол буферден алынып пайдаланылады. Буферді пайдалану – файлдармен орындалатын енгізу-шығару операцияларының жылдамдығын әжептеуір арттыруға мүмкіндік береді, өйткені жай жұмыс істейтін дискімен бір рет мәлімет алмасу орнына буфермен оншақты рет мәлімет алмасу (оқу/жазу) операциясын орындауға болады. Дискілік файлдағы мәліметтерді тек тізбекті түрде біртіндеп оқымай, керекті мәліметті оның көрсетілген нөмірі бойынша кез келген жерінен ала беруге болады.

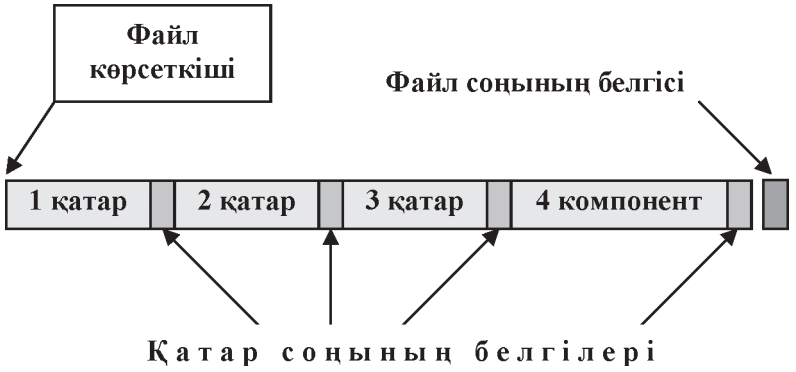
Дисплей, пернетақта, т.б. негізгі енгізу-шығару құрылғыларымен ақпарат алмасу кезінде логикалық құрылғылар пайдаланылады. Логикалық құрылғылардың стандартты аттары бар, олар:

- **CON** – консоль: мәлімет шығару кезінде – экран, ал мәлімет енгізуде – пернетақта;
- **PRN** – принтер;
- **NUL** – «бос құрылғы», программа қателері түзетілгеннен кейін оның жұмысын тексеру кезінде мәлімет шығару құрылғысы ретінде пайдаланылады.

Дискілік файлдардағыдай емес, логикалық құрылғылар арқылы енгізу-шығару операцияларын орындау тек тізбекті түрде жүргізіледі, өйткені файлға жазу кезінде мәлімет құрылғыға біртіндеп, бір компоненттен соң келесі компонент тізбекті түрде беріліп отырады және файлдан мәлімет аларда да оның компоненттері одан бір-бірлеп оқылады.

Файл компонентін анықтау (пайдалану) файл көрсеткіші бойынша жүргізіледі. Оқу немесе жазу операцияларын орындау кезінде көрсеткіш автоматты түрде келесі компонентке жылжытылып отырады (9.2-сурет).

Турбо Паскаль тілінде файлдарды бір-бірінен ажырату (идентификациялау) үшін файлдық айнымалылар пайдаланылады. Информацияны бейнелеу тәсіліне байланысты файлдардың үш типі болады, соған сәйкес файлдық айнымалыларды сипаттау тәсілдері де болады (9.3-сурет).

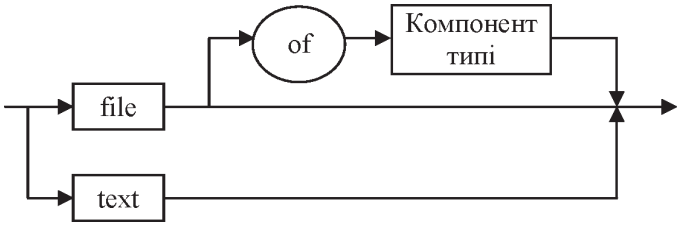


9.2-сурет. Файлды ұйымдастыру

Типтелген файлдар. Типтелген файлдың файлдық айнымалысы былай сипатталады

Type < файлдық айнымалы идентификаторы > = **file of** < компонент типі >; ...

мұндағы < компонент типі > – файлдық типтен басқа кез келген мәліметтер типі.



9.3-сурет. Файлдық типтің синтаксистік диаграммасы

Файлда сақталатын немесе құрылғыдан құрылғыға тасымалданатын ұзындықтары бірдей компоненттер тізбегін (сандарды, жазбаларды, т.б.) өңдеу кезінде типтелген файлдар пайдаланылады.

Файлдық айнымалы типі болып табылатын мәтіндік файлдар былай сипатталады:**Type** <Файлдық айнымалы идентификаторы> = **text; ...**

Мәтіндік файлдар ұзындықтары әр түрлі болып келетін сөз тіркестері түріндегі мәтіндермен жұмыс істеу кезінде қолданылады.

Типтелмеген файлдар сипатталуы:

Type <Файлдық айнымалы идентификаторы> - **file; ...**

Сыртқы есте сақтау құрылғысы мен жедел жады арасында ұзындықтары көрсетілген физикалық жазбаларды жоғары жылдамдықта тасымалдау кезінде типтелмеген файлдар пайдаланылады, олар тасымалдау кезінде түрлендірілмейді.

Файлдық айнымалылар Pascal тілінің кез келген айнымалысы тәрізді сипаттау бөлімінде де жариялана береді, мысалы:

```
Var F1:file of real;
F2:file;
F3:text; ...
```

немесе типтерді алдын ала сипаттау арқылы да көрсетіледі:

```
Type FF = file of integer;
Var F1:FF; ...
```

Қажет болса, файлдық айнымалыны ішкі программаға параметрлер арқылы беруге болады. Бірақ бұл мақсатта тек параметр-айнымалыларды ғана пайдалануға болады, мысалы:

```
Type FF = file of integer;
Procedure Print(Var F1:FF); ...
```

Файлдармен жұмыс істеу кезінде мынадай әрекеттер атқарылады:

- файлдық айнымалыны инициалдау, яғни файлдық айнымалының файлмен байланысын орнату;
- файлды ашу - файлды енгізу/шығару операцияларын атқаруға дайындау;
- файл компоненттерін өңдеу – енгізу/шығару операцияларын атқару;
- файлды жабу (қайталап ашылған файл автоматты түрде жабылады).

Файлдық айнымалыны инициалдау. Физикалық құрылғы (дискіжетек немесе сыртқы құрылғы) мен файлдық айнымалы арасындағы байланыс арнайы процедура арқылы орнатылады.

Assign (Var f; st:string) процедурасы f файлдық айнымалысын st тіркесімен анықталатын файлмен немесе логикалық құрылғымен байланыстыра отырып, оны инициалдайды.

Егер файл ағымдағы каталогта болса, онда тек файл мен оның

типін көрсету жеткілікті, әйтпесе файлдың толық атын көрсету керек, мысалы:

```
Type FI1 = file of integer;
Var f1,f2,f3:FI1;
...
Assign (f1,'F1.dat'); {файлдық айнымалыны
ағымдағы каталог ішіндегі файлмен
байланыстыру}
Assign (f2,'d:\iva\a.dat'); {файлдық
айнымалыны көрсетілген каталогтағы файлмен
байланыстыру }
Assign (f3,'CON'); {файлдық айнымалыны
консольмен байланыстыру}
```

Файлды ашу кезінде мәліметті тасымалдау бағыты көрсетіледі, мұнда файл мәлімет оқу немесе жазу үшін ашылады. Мәтіндік файл сөз тіркестерін қосу үшін де ашылуы мүмкін. Мәлімет оқу үшін ашылған типтелген файлға жаңа жазбалар қосуға немесе ескі жазбалар үстіне жаңаларын жазуға болады.

1. **ReSet(Var f)** процедурасы - f файлдық айнымалысымен анықталған файлды мәлімет оқу үшін ашады. Бұл процедура орындалар кезде файл көрсеткіші бірінші компонентке орналасады (жазбалардың бірінші блогы буферге оқылады). Мұнда логикалық құрылғы енгізу операциясын орындауға даярланады. Егер бұрын болмаған файлды ашу көрсетілсе, онда қате болғаны тіркеліп, Word типтес lOResult функциясы 0-ден басқа мән қайтарады (төмендегі функция сипатталуын қараңыз). Енгізу-шығару операцияларын бақылауды алып тастап, lOResult функциясын пайдалана отырып, аты көрсетілген файлдың бұрын дискіде болғанын-болмағанын тексеруге болады:

```
Var f:file of char;
Begin
Assign(f, 'a.dat'); {файлдық айнымалыны
инициалдау}
{$ I-} {енгізу-шығару қателерін бақылауды алып
тастау}
ReSet(f); {файлды оқу үшін ашу}
```

```
{$ I+} {қателерді бақылауды іске қосу}
If IOResult <>0then WriteLn('Файл жоқ');
else WriteLn('Файл бар'); ...
```

2. **ReWrite(Var f)** процедурасы f файлдық айнымалысымен анықталған файлды оған мәлімет жазу үшін ашады. Бұрын бар файлды ашу көрсетілсе, ондағы мәлімет өшіріледі және ол туралы мәлімет берілмейді. Ал бұрын болмаған жаңа файлды ашу көрсетілсе, ол жаңадан пайда болып, мәлімет жазуға дайын болады (физикалық түрде буфер тазаланады). Мұнда логикалық құрылғы мәлімет қабылдауға дайындалады.
3. **AppEnd(Var f:text)** процедурасы f файлдық айнымалысымен анықталған мәтіндік файлды оған сөз тіркестерін қосып жазу үшін ашады. Мұнда файл көрсеткіші файлдағы мәліметтер соңына орналасады да, енгізілетін мәліметтер бұрынғылардың соңғына жазылады.

Кез келген программада файлдық айнымалыны жарияламайақ стандартты екі файлды пайдалануға болады, олардың файлдық айнымалылары:

- **INPUT** - стандартты енгізу құрылғысынан мәлімет оқу;
- **OUTPUT** - стандартты шығару құрылғысына мәлімет жазу.

Бұлар қарапайым енгізу-шығару операцияларында қолданылатын мәтіндік файлдар. Енгізу-шығару операторларында бұлар әдетте көрсетілмейді, ал қалған файлдарды белгілі бір файлдық айнымалылармен немесе логикалық құрылғылармен байланыстырып алып барып қолдану керек.

MS DOS жүйесінде стандартты енгізу құрылғысы – пернетақта, ал стандартты шығару құрылғысы – дисплей экраны.

Ескерту. Керек болған жағдайда бұларды операциялық жүйе мүмкіндіктері арқылы басқаша етіп қайта тағайындауға болады. Мұндайда, пернетақтадан мәлімет енгізу орнына – MS DOS командалық жолында программа атынан соң «<» символын және файл атын, ал экран орнына файлға мәлімет шығару керек болғанда, –«>» символы мен файл атын көрсету қажет. Бұлардың тек мәлімет енгізуін немесе тек мәлімет шығаруын ғана көрсетуге немесе екеуін де қатарластыра қолдана беруге болады.

Мысалы:

A:\>example.exe <a.dat > a.res – a.dat файлынан мәлімет енгізу, ал a.rez файлына мәлімет шығару.

Мұндай қайта тағайындаулар енгізу-шығару операцияларын тікелей құрылғылар арқылы атқаратын Crt модулін пайдаланбаған кезде ғана орындалады.

Файл компоненттерін өңдеу кезінде орындалатын негізгі операциялар – жазу және оқу операциялары. Осылардың негізінде бұлардан күрделірек мынадай әрекеттер атқарылады:

- файл құру (жаңадан ашу) – файлға керекті жазбалар енгізу;
- файлды толықтыру (модификациялау) – барлық жазбаларды немесе олардың бірнешеуін өзгерту, жазбаларды өшіру және қосу;
- файлдағы керекті мәліметті іздеу.

Осы операцияларды орындау әрбір типтегі файлдар үшін әр түрлі болып атқарылады.

Файлды жабу (оны оқу әрі жазу кезінде де) мынадай процедура арқылы орындалады

```
Close(Var f);
```

Мұнда жаңадан құрылған файл каталогта тіркеледі. Дискімен мәлімет алмасу буфер арқылы орындалатындықтан, файлға жазылатын мәліметтің бір бөлігі буферде қалып кетуі мүмкін, сондықтан файлды жабу кезінде олар да файлға жазылып қойылады. Файл жабылғанмен, файлдық айнымалының сол файлмен байланысы жоғалмайды, сол файлды қайта пайдалану керек болғанда, Assign процедурасын тағы да жазу қажет емес.

9.3 Типтелген файлдарды өңдеу ерекшеліктері

Белгілі бір типтегі файл (типтелген файл) ұзындықтары бірдей және ішкі форматтары да біркелкі жазбалар жиынынан тұрады. Жазбалар бірінен соң бірі тізбектеле орналасады. Файлдың бірінші секторының алғашқы 4 байты жазбалардың саны мен ұзындықтары жайлы мәлімет сақтайды. Осындай құрылымды файлдардағы мәліметтерді тізбекті түрде біртіндеп және таңдай отырып, кез келген жерінен (тікелей қатынасу арқылы) алуға болады.

Тізбекті қатынасу кезінде жазбалар бірінен соң бірі оқылады, яғни I+1-жазбаны оқу/жазу тек I-компоненттен кейін ғана орындалады.

Тікелей қатынасу кезінде жазбалар нөлден бастап нөмірленіп орналасқаны есепке алынып, керектілері нөмірлері бойынша тауып алынады. Жазбалар нөміріне сәйкес файл көрсеткіші орнын көрсете отырып, керекті мәліметті бірден **SEEK** процедурасы арқылы тауып алуға болады. Жалпы файлды пайдалану кезінде мынадай сипаттамалар қолданылады:

TYPE <тип идентификаторы> **= FILE OF** <компонент типі>;

Мысалы,

```
1) TYPE T = FILE OF REAL;
   VAR F: T;
2) VAR F: FILE OF REAL;
3) TYPE ST = RECORD
                A: STRING[10];
   B: INTEGER;
     C: REAL;
     D: BYTE
   END;
   VAR DAN: FILE OF ST;
```

Бірінші нұсқада файл типі типтерді сипаттау бөлімінде көрсетіліп, айнымалыларды сипаттау бөлімінде сол типті пайдалануға болады. Екінші нұсқада тип алдын ала сипатталмаған. Үшінші нұсқада алдын ала файл жазбалары типі сипатталады, ал айнымалыларды сипаттау бөлімінде сол тип жеке жазба типін көрсету үшін қолданылады.

Типтелген файлдарды жазу/оқу процедуралары мен функциялары: **READ**, **WRITE**, **ASSIGN, RESET, REWRITE, SEEK, CLOSE, FILEPOS, FILESIZE, EOF**. **TRUNCATE** процедурасы файл көрсеткіші тұрған позициядан бастап файлды қысқартып тастайды.

Мәліметтері пернелерден енгізілетін жазбалардан тұратын файл жасау керек болсын делік. Файл құрылған соң, оның ішкі мәліметтері экранға шығарылуы тиіс.

Файл жазбалары құрылымы:

- фамилиясы;
- табельдік нөмірі;
- айлығы.

```
TYPE TZ=RECORD
  FIO:STRING[10];
  TN:INTEGER;
  ZP:REAL
END;
VAR  ZAP:TZ;
     FOUT:FILE OF TZ;
     FL:BOOLEAN;
     NAME:STRING;
BEGIN
     REPEAT
      WRITELN('ФАЙЛ АТЫ ');
  READLN(NAME);
  ASSIGN (FOUT,NAME);
 {$I-} RESET(FOUT); {$I+}
  IF IORESULT=0 THEN
    BEGIN
   WRITELN(NAME,' ФАЙЛЫ', ' АШЫЛҒАН');
   CLOSE(FOUT);
   FL:=FALSE
    END
              ELSE
    BEGIN
        REWRITE(FOUT);
   FL:=TRUE
    END
     UNTIL FL;
     WITH ZAP DO
     REPEAT
  WRITELN(' FIO,TN,ZP ЕНГІЗУ ');
  READLN(INPUT,FIO,TN,ZP);
  WRITE(FOUT,ZAP);
     UNTIL EOF(INPUT);
     CLOSE(FOUT);
```

```
      RESET(FOUT);
      WITH ZAP DO
           REPEAT
   READ(FOUT,ZAP);
              WRITELN(FIO:15,TN:9,ZP:8:2);
           UNTIL EOF(FOUT);
           CLOSE(FOUT)
         END.
```

Программа басында бұрын болмаған файл аты кездескенше файл аттары енгізіледі, әйтпесе бұрынғы болған файл мәліметтерімен толық өшіріліп кетеді. Жаңа файл аты енгізілген соң, **FL** жалаушасы мәні **TRUE** болады да, файл атын енгізу тоқталады. Бұдан кейін мәліметтерді тікелей пернелерден енгізу **INPUT** файлы арқылы орындалады. Енгізуді аяқтау белгісі – **Ctrl+Z**. **INPUT** стандартты файл атын пернелерден енгізерде, **READLN** файлдан мәлімет оқу операторында және енгізу соңын тексеретін **EOF** функциясында да көрсетпеуге болады. Файл ашылып, мәліметтер енгізіліп болған соң ол **CLOSE** процедурасымен жабылады. Содан кейін құрылған файл мәлімет оқу үшін ашылады да, оның мәліметтері кесте түрінде экранға шығарылады.

Файл соңына кейіннен қосымша мәліметтер енгізіле алады. Ол үшін файл көрсеткішін орналастыру процедурасы қолданылады:

SEEK(<файл аты**>, FILESIZE(<**файл аты**>)).**

Келесі программада алдыңғы құрылған файл жазбаларымен тікелей қатынасу арқылы табельдік нөмірлері пернелерден енгізілетін жұмысшылардың жалақыларының қосындысын табу керек (файл құрылған кезде пайдаланылған табельдік нөмірлер 101-999 сандары аралығында болып саналады, сол себепті 101 нөмірлі жазба файлда алғашқы орында тұрады, келесісі – 102, т.с.с.).

```
TYPE TZ=RECORD
FIO:STRING[10];
TN:INTEGER;
ZP:REAL
     END;
VAR  ZAP:TZ;
     FOUT:FILE OF TZ;
```

```
      TN1,TN2,N:INTEGER;
      S:REAL;
      NAME:STRING;
BEGIN
    WRITELN('ФАЙЛ АТЫ ');
    READLN(NAME);
          ASSIGN (FOUT,NAME);
          RESET(FOUT);
           S:=0;
   REPEAT
     READLN(TN2);
  TN1:=TN2-101; {жазбалар көрсеткішін құру}
SEEK(FOUT,TN1);
READ(FOUT,ZAP);
S:=S+ZAP.ZP;
            UNTIL EOF;
            WRITELN('S= ',S);
            CLOSE(FOUT)
         END.
```

Бұл программадағы мәліметтер өңдеу процесі табельдік нөмірлерді енгізу тоқталғанда, яғни пернелерден Ctrl+Z енгізілгенде аяқталады.

Типтелген файл – бұл файлдық айнымалыны жариялау кезінде берілген барлық компоненттері бір типте болатын файл. Файл компоненттері дискіде ішкі екілік форматта сақталады. Егер осындай файл мәліметтерін кез келген мәтіндік редактормен қарайтын болсақ, тек символдық мәліметтер көрінеді де, файлдағы сандар орнына бос орын таңбалары немесе псевдографика символдары тұрады.

Сонымен, типтелген файлдармен жұмыс істеу кезінде төмендегідей арнайы процедурлар мен функциялар қолданылады.

1. **Read(Var f; c1, c2, ...,cn)** процедурасы типтелген файл компоненттерін оқиды. Файлдың соңына жеткенде, бұл процедура енгізу-шығару қатесін көрсетеді.
2. **Write(Var f; c1, c2, ..., cn)** процедурасы типтелген файлға мәліметтер жазады.

3. **Seek(Var f; numcomp:word)** процедурасы файл көрсеткішін numcomp нөмірлі файл компонентіне орналастырады.
4. **FileSize(Var f):longint** функциясы файлдық айнымалы көрсетіп тұрған файл компоненттері санын береді. SeekQ функциясымен бірге файл соңын анықтау үшін қолданылады.
5. **Seektf, FileSize(f)); ...**
6. **FilePos(Var f):longint** функциясы келесі енгізу-шығару операциясы өңдейтін компонент нөмірін береді.

Мәлімет оқу немесе жазу үшін файл ашылған соң, файл көрсеткіші оның басындағы 0 нөмірлі бірінші компонентте орналасады. Әрбір компонент оқылған/жазылған соң, көрсеткіш келесі компонентке ығысады. Әрбір компонент ұзындығы тұрақты болғандықтан, мәліметтерді тікелей нөмірлері арқылы да тауып пайдалануға болады.

Файл соңына жазбалар қосу мәліметтерді оқу арқылы орындалады. Мұндайда файл көрсеткіші оның соңына орналасады да, сонан кейін керекті мәліметтер жалғастырылып жазылады.

Компоненттерді файл ортасына жазу кезінде оның нөмірі арқылы орнын анықтап алған соң, одан кейінгі компоненттер уақытша файлға жазылады да, жаңа компоненттер жазылып, оған жалғаса уақытша файлдағы мәлімет қайтадан негізгі файлға көшіріліп жазылады.

Компоненттерді өшіру файлды қайта жазу арқылы орындалады.

6.3-мысал. Компоненттері пернелерден енгізілетін символдардан тұратын файл құру керек. Файлдағы символдарды өзгертіп, ондағы мәліметтерді олардың басынан және соңынан кезектестіре оқып, соңында көрсетілген символды тауып оны өшіретін программа жазып шығу керек.

```
Program ex;
Var f,f1:file of char; {екі файлдық айнымалы}
ch,i:char;
j:longint;
name:string[8];
Begin
```

```
WriteLn('Файл атын енгізіңіз:'); ReadLn(name);
{ файл жасау және ашу}
Assign(f,name+'.dat'); {файлды файлдық айныма-
лымен байланыстырамыз}
ReWrite(f); {файлды мәлімет жазу үшін ашу
(құру)}
WriteLn('Символдар немесе CTRL-Z енгізіңіз:');
{файлға жазбалар енгізу}
while not EOF(f) do {пернелерден CTRL-Z
енгізілгенше}
begin
ReadLn(ch); {пернеден символ енгіземіз}
Write(f,ch); {символды файлға жазамыз}
end;
WriteLn;
{ файлдан біртіндеп жазбаларды оқу}
Reset(f); { файлды оқу үшін ашу}
while not EOF(f) do { файл соңына жеткенше}
begin
Read(f,i); {файлдан символ оқу}
Write(i,' '); {символды экранға шығару}
end;
WriteLn;
{файлдағы жазбаларды өзгерту}
Reset(f); {файлды оқу үшін ашу}
while not EOF(f) do {файл соңына жеткенше}
begin
Read(f,i); {файлдан символ оқу}
Write(i,' '); {символды экранға шығару}
i:=chr(ord(i)+10); {символды өзгерту}
WriteLn(i); {өзгертілген символды экранға
шығару}
Seek(f,FilePos(f)-1); {бір компонентке кері
қайту}
Write(f,i); {символды қайта жазу}
end;
WriteLn;
```

```
{жазбаларды файл басынан және соңынан
кезектестіре оқу}
ReSet(f); {файлды оқу үшін ашу}
j:=0; {компонент нөмірін 0-ге орналастыру}
while not EOF(f) do {файл соңына жеткенше}
begin
Read(f,i); {файл басынан символ оқу}
Write(i); { символды экранға шығару }
Seek(f,FileSize(f)-FilePos(f)); {файл соңынан
оқу үшін көрсеткішті орналастыру}
Read(f,i); { файл соңынан символ оқу }
Write(i); { символды экранға шығару }
j:=j+1; {компонент көлемін үлкейту}
Seek(f,j); { файл басынан келесі компонентке
көрсеткішті орналастыру }
end;
WriteLn; WriteLn('Өшірілетін символды
енгізіңіз'); ReadLn(ch);
{жазбаларды өшіруге даярлау; бастапқы файл
атын өзгерту және сол атпен жаңа файл ашу}
Close(f); {файлды жабу}
ReName(f,name+'.bak'); { файл атын өзгерту}
ReSet(f); { файлды оқу үшін ашу}
Assign(f1,name+'.dat'); {жаңа файлды айнымалы-
мен байланыстыру}
ReWrite(f1); { жазу үшін жаңа файл ашу}
{жазбаларды өшіру – қалған жазбаларды басқа
файлға жазу}
while not EOF(f) do { }
begin
Read(f,i); { файлдан символ оқу }
if i<>ch then Write(f1,i); {егер символ
өшірілмеуі керек болса, оны жаңа файлға жазу}
end;
Erase(f); {ескі файлды жою, ол жабылған соң
онда ешнәрсе өзгермейді, сол себепті оны
қайтадан жабу қажет емес}
```

```
{жаңа файлдан жазбаларды біртіндеп оқу}
ReSet(f1); {жаңа файлды оқу үшін ашу}
while not EOF(f) do
begin
Read(f1,ch); { файлдан оқу }
Write(ch,' ');
end;
WriteLn; End.
```

6.4-мысал. Адамдардың фамилиялары мен туылған күндері тізім түрінде жазылған файл жасайтын программа құру керек. Осы файлдағы мәліметтерден берілген фамилиясы бойынша адамның туылған күнін анықтау керек.

```
Program ex;
Type fam=record { «қызметкерлер туралы
мәліметтер» типіндегі жазба}
ff:strin[20]; {фамилиясы}
year:word; {туған жылы}
month:1..12; { туған айы}
day:1..31; { туған күні}
end;
Var f:file of fam; { «қызметкерлер файлы»
файлдық айнымалысы}
fb:fam;
n,i:integer;
fff:string;
key:boolean;
Begin
Assign(f,'a.dat'); { файлдық айнымалыны файл-
мен байланыстыру}
ReWrite(f); { жазу үшін файл ашу}
WriteLn('Мәлімет немесе CTRL-Z енгізіңіз');
while not EOF do { CTRL-Z енгізілгенше цикл}
begin
ReadLn(fb.ff,fb.year,fb.month,fb.day);
{мәліметтерді өрістер бойынша енгізу, фами-
лияны бөлек жолда енгізу, жолды енгізу Enter
арқылы аяқталады}
```

```
Write(f,fb); { жазбаны файлға бір компонент
түрінде жазу}
end;
Close(f); {файлды жабу}
WriteLn('Фамилияны енгізіңіз');
ReadLn(fff);
key:=false; {«жазба табылмады» белгісін орна-
ту}
ReSet(f); {файлды оқу үшін ашу}
while (not EOF(f)) and (not key) do {файл
соңына жетпей және жазба табылмағанша }
begin
Read(f,fb); { файлдан жазба оқу}
if fb.ff=fff then {егер фамилия сәйкес келсе,
онда}
begin {мәлімет шығару}
WriteLn('Дата: ',fb.year,fb.month,fb.day:3);
key:=true; {«жазба табылды» белгісін орнату}
end;
end;
if not key then {егер белгі орнатылмаса }
WriteLn('Мәліметтер жоқ'); {онда « Мәліметтер
жоқ'» деп жазамыз}
Close(f); {файлды жабу}
End.
```

Кез келген мәтіндік файл CHAR типті компоненттері бар типтелген файл ретінде оқылады. Мұнда мәтіндік файл соңы маркері екі символдар тіркесінен #13 және #10 түрінде болады.

6.5-мысал. CHAR типті компоненттерден тұратын типтелген файл түріндегі мәтіндік файл ашатын және одан мәліметтерді бір-бір символдан оқитын программа жазу керек.

```
Program char_text_file;
Type ff=file of char; {жаңа тип - символдық
файл}
Var
f:ff; { символдар файлы типіндегі файлдық ай-
нымалы }
```

```
a:char;
n,i:integer;
fname,st:string[30];
Begin
WriteLn('Файл атын енгізіңіз'); ReadLn(fname);
Assign(f,fname); {файлдық айнымалыны файлмен
байланыстыру}
ReSet(f); {мәтіндік файлды типтелген файл
ретінде оқу үшін ашу}
while not EOF(f) do { файл соңына жеткенше}
begin
st:='';
Read(f,a); {символды оқу}
while (a<>#13) and not EOF(f) do {жол соңы
маркеріне немесе файл соңына дейін}
begin
st:=st+a; {оқылған символды жолға қосу}
Read(f,a); {келесі символды оқу}
end;
if not EOF(f) then Read(f,a); { #10 символын
өткізіп жіберу}
WriteLn(st); {құрастырылған жолды шығару}
end;
Close(f);
End.
```

9.4 Мәтіндік файлдармен жұмыс істеу

Мәтіндік файлдар символдардан құралған қатарлардан, яғни жолдардан тұрады, әрбір қатардың соңына оның аяқталғанын белгілеп отыратын шартты белгі (CR) қойылады. Әрбір файл оның соңын білдіретін арнайы символмен аяқталады. Мәтіндік файлдар стандартты TEXT түйінді сөзімен сипатталады және олардың компоненттерін тек бірінен кейін бірін тізбектей отырып қана оқуға болады. Мұндай файлдармен жұмыс істеу кезінде арнайы қосымша процедуралар мен функцияларды пайдаланлады.

Айта кететін бір жайт, мәтін түріндегі файлға не файлдық құрылғыларға түсетін символдар тізбегі енгізілгенде жолдар

соңында міндетті түрде орналасатын CR – carriage return шартты белгісі «енгізу» (↵) пернесін басу арқылы қойылады, ал мәтін толық теріліп біткенде, оның ең соңына Ctrl+Z қос пернесін басу арқылы «файл соңы» белгісі – EOF орналасуы тиіс. Сонымен, файл компоненттері (сандар, сөздер) бір-бірінен «бос орын» немесе CR арқылы бөлініп жазылып, олардың ең соңында CTRL+Z шартты белгісі міндетті түрде қойылады.

Мәтіндік файлдардағы мәліметтер тек тізбекті түрде ғана алынады. Олармен әр түрлі мәтіндік редакторлар жұмыс істей алады. Мәтіндік файлдардың типі – стандартты **TEXT** типі.

VAR <файл аты**>: TEXT;**

Символдық енгізу-шығару операциялары **READ және WRITE** процедураларымен, ал жолдар – **READLN** және **WRITELN** процедураларымен атқарылады. Бұдан басқа да **ASSIGN, RESET, REWRITE, CLOSE, EOF, EOLN** сияқты арнайы процедуралар қолданылады. **APPEND** процедурасы бұрыннан бар мәтіндік файлға жазбалар қосу үшін қажет. Мәтіндік файлдар үшін **SEEK, FILEPOS, FILESIZE** процедуралары мен функциялары қолданылмайды**,** өйткені мұндағы элементтер ұзындығы әр түрлі болып келеді.

INPUT және **OUTPUT** стандартты процедуралары мәліметтерді пернелерден енгізіп, экранға шығару үшін қажет болады.

1-мысал. Дискіде бүтін сандардан тұратын f1.dat файлы бар делік. Осы файлдағы сандарды біртіндеп оқи отырып, экранға солардың ішіндегі тек жұп сандарды шығару қажет болсын.

```
program (fil, output);
 var fil:text;
 i:integer;
begin
 assign (fil,'f1.dat');
 reset(fil); writeln;
 while not eof(fil) do
  begin
   read(fil,i);
   if i mod 2=0 then write(i,' ':4)
  end;  close(fil)
end.
```

2-мысал. Пернелерден n сан енгізіп, оны int.dat файлына жазу керек.

```
program rem(int, input);
 const n=15;
var  int:text;
    i,n:integer;
begin
 assign(int, 'ff.dat'); rewrite(int);
 for i:=1 to n do
  begin
  write('сан енгізіңіз:'); read(n);
  write(int,k,'':2)
 end;
  writeln(int);
 close(int)
   end.
```

2-мысалдың басқаша шығарылу жолын қарастырайық.

```
   program rr;
   const n=15;
   type v=file of integer;
    var i,k:integer;
     int:v;
   begin
     assign(int, 'fff.dat');
     rewite(int);
     for i:=1 to n do
    begin
    write('сан енгізіңіз:');
    read(k);
     write(int,k)
    end
   end.
```

3-мысал. Мәліметтері бар F1 мәтіндік файлы берілген делік. Осы файлдың барлық жазбаларындағы ИС-101 тобы кодын ИС-201 кодына ауыстыру керек. Өзгертілген жазбаларды F2 файлына жазу қажет.

```
      VAR F1,F2: TEXT;
```

```
        POLE:STRING;
        NAME:STRING[12];
        PZ: INTEGER;
BEGIN WRITE('МӘЛІМЕТ ОҚЫЛАТЫН ФАЙЛ АТЫ:');
        READLN(NAME);
        ASSIGN(F1,NAME);
  WRITE('МӘЛІМЕТ ШЫҒАРЫЛАТЫН ФАЙЛ АТЫ:');
        READLN(NAME);
        ASSIGN(F2,NAME);
        RESET(F1);  REWRITE(F2);
        WHILE NOT EOF(F1) DO
          BEGIN
            READLN(F1,POLE);
   WHILE POS('ИС-101', POLE) <> 0 DO
     BEGIN
      PZ:= POS('ИС-101', POLE);
      DELETE(POLE,PZ+3,1);
      INSERT('2',POLE,PZ+3);
     END;
   WRITELN(F2,POLE)
  END;
        CLOSE(F1);
 CLOSE(F2);
END.
```

Мұнда бұрын мәлімет жазылған файлдан жолдар біртіндеп оқылады да, әрбір жолдағы топ нөміріндегі 1 символы 2-ге өзгертіледі. Өзгертілген жолдар жаңа файлға жазылады.

Мәтіндік файл – бұл компонентері – ұзындықтары әр түрлі жолдарды құрайтын сөз тіркестерінен тұратын файл, әрбір жол арнайы жол соңы белгісімен аяқталады.

Ескерту. Жол соңы маркері – бұл ASCII-кодтар кестесіндегі арнайы екі символдан – «#13, #10» тұратын тіркес. 13 коды курсорды жол басына орналастыру командасы болып, ал 10 коды – келесі жолға көшу командасы болып саналады. Бұл кодтар комбинациясы ENTER пернесін басқанда қалыптасады.

Мәтіндік файл мәлімет жазу, оқу және файл соңына жазбалар қосу үшін ашылады. Мәлімет жазу үшін ашылған файлдан

мәлімет оқуға болмайды және керісінше, оқу үшін ашылған файлға мәлімет жазылмайды. Жазбалар қосу үшін ашылған файл одан мәлімет оқу немесе жазу мақсатында ашылмағаны тексеріледі, егер сондай болған жағдайда алдымен ол жабылады да, сонан кейін жазбалар қосу үшін қайта ашылады.

Мәтіндік файлдармен жұмыс істеу кезінде төмендегі арнайы процедуралар мен функциялар қолданылады.

1. **EOLn([Var f])** функциясы: типі **boolean** – егер мәтіндік файлдағы жол соңы анықталса, TRUE мәнін қайтарады.
2. **Read([Var f:text;] v1, v2,... vn**) процедурасы символдарды, жолдарды және сандарды енгізеді. Енгізу тізбегі CHAR, STRING, бүтін және нақты сандар типіндегі бір немесе бірнеше айнымалылардан тұрады. Егер оқу кезінде файл көрсеткіші жол соңына барса, нәтиже #13 символы, ал файл соңына барса, #26 символы болады. STRING типті айнымалы енгізілгенде осы процедурамен оқылып, бір жолға орналасатын сөз тіркестерінің ұзындығы жолдың максимальды ұзындығына тең болады (егер оған дейін жол символдарына қосылмайтын жол соңы немесе файл соңы маркері кездесіп қалмаса). Жолдың максимальды ұзындығынан тыс символдар алынып тасталынады. Жаңадан Read процедурасымен мәлімет оқу бос жол қайтарады. Бұл Read процедурасының бірнеше жолды оқымай, тек бір жолды ғана дұрыс қабылдап, қалғандарын бос қалдыратынын білдіреді. Сандарды енгізгенде бұл процедура алғашқы символға дейінгі барлық босорын, табуляция белгілерін алып тастайды да, сөз тіркесін келесі босорын не табуляция белгілеріне дейін оқиды. Осылай қалыптасқан сөз тіркесі типке сәйкес символдық түрден ішкі бейнелеу форматына түрлендіріледі де, тізімдегі келесі айнымалыға меншіктеледі. Егер формат бұзылса, онда қате тіркеледі, ал файл соңы маркері кездессе, айнымалыға 0 мәні меншіктеледі де, ол жайлы ешқандай хабар берілмейді. Read және ReadLn процедуралары арқылы логикалық константалар енгізілмейді. Пер-

нелерден енгізілген мәліметтер буферге орналасады да, олар ENTER басылғанда процедураға беріледі, олардың ұзындығы 127 байттан аспауы тиіс. Одан артық символдар бір процедура арқылы енгізілмейді.

3. **ReadLn([Var f;] v1,v2, ...,vn)** процедурасы да символдарды, жолдарды және сандарды енгізеді. Бұл процедура да Read тәрізді жұмыс істейді, бірақ соңғы айнымалы оқылғаннан кейін жолдың соңына дейінгі қалған бөлігі алынып тасталады да, келесі ReadLn немесе Read жаңа жолдың алғашқы символынан бастап енгізеді. Процедура параметрсіз тұрса, онда ол жол соңына дейінгі барлық символдарды оқымайды. Read процедурасынан кейін ReadLn процедурасын параметрсіз пайдалану енгізу буферін тазалайды. Параметрсіз ReadLn процедурасының алдында Read болмаса, ол енгізу әрекетіне көшіп, программа орындалуын ENTER басылғанша, тоқтата тұрады да, тұтынушы экранын оқуға мүмкіндік береді.
4. **Write([Var f;] v1,v2, ...,vn)** процедурасы мәліметтерді мәтіндік файлға немесе логикалық құрылғыға шығарады. Оның шығару тізімі CHAR, STRING, BOOLEAN және де бүтін немесе нақты сандар типіндегі бір немесе бірнеше өрнектен тұрады. Сандарды шығару кезінде олар символдық түрге айналады. Файлдық айнымалы көрсетілмесе, мәлімет стандартты OUTPUT файлы арқылы экранға шығарылады. Тізімдегі кез келген параметр мынадай форматта болуы мүмкін:

<параметр> [: <бүтін1> [: <бүтін2>]],

мұндағы <бүтін1> мен < бүтін2> шығарылатын мәннің ұзындығын көрсетеді.

5. **WriteLn([Var f;] v1,v2, ...,vn)** процедурасы да мәліметтерді мәтіндік файлға немесе логикалық құрылғыға шығарады. Файлдық айнымалы болмаған жағдайда, мәлімет стандартты OUTPUT файлына, яғни экранға шығарылады. Бұл процедура Write процедурасымен бірдей, тек шығарылатын мәліметтер жолы #13 және #10 символдарымен аяқталады. Параметрсіз WriteLn файлға жол соңы маркерін (экранға мәлімет шығарғанда, келесі жолға көшу) жазады.

6. **SeekEOLn([Var f]):boolean** функциясы алғашқы символ таңбасына дейінгі жол соңы және табуляция маркерлерін немесе барлық босорын таңбаларын алып тастайды да, жол соңы маркерін кездестіргенде, TRUE мәнін қайтарады. Егер файлдық айнымалы болмаса, ол стандартты INPUT файлымен жұмыс істейді.
7. Функция **SeekEOF([Var f]):boolean** функциясы файл соңы таңбасына дейінгі немесе алғашқы символға дейінгі жол соңы және табуляция маркерлерін немесе барлық босорын таңбаларын алып тастайды да, файл соңы маркерін кездестіргенде, TRUE мәнін қайтарады. Егер файлдық айнымалы болмаса, ол стандартты INPUT файлын тексереді.

Бірнеше мысал қарастырайық.

6.1-мысал. 26 жолдан тұратын мәтіндік файл құру керек, оның әрбір жолына латын алфавитінің бас әріптерінің кез келген санын төмендегідей түрде жазып шығу қажет:

AAAAA
BBBBB
C
DDDDDDDDDDDDDDDDDDDDD
EEEEEEEEEEEEE и т.д.

```
Program form_text_file;
Var
f:text; {мәтіндік файлға арналған файлдық ай-
нымалы}
f:char; n,i:integer; fname,st:string[30];
Begin
WriteLn('Файл атын енгізіңіз'); ReadLn(fname);
Assign(f,fname); {файлдық айнымалыны
анықтаймыз}
ReWrite(f); {жазбалар үшін файл ашу}
Randomize; {кездейсоқ сандар алуды дайындау}
for a:='A' to 'Z' do {жолдарды құрастыру}
begin
st:='';
n:=Random(30)+1;
for i:=1 to n do st:=st+a;
```

```
WriteLn(f,st); {мәтіндік файлға жол жазу}
WriteLn(st); {бақылау үшін оны экранға шығару}
end;
Close(f); {файлды жабу}
End.
```

Мәтіндік файл ұзындығы әр түрлі болғандықтан, оларды тек тізбекті түрде біртіндеп өңдейді (жазу, оқу және іздеу). Файлды толықтырудың кез келген түрі (оның соңына жазба қосудан басқа) оның мәліметтерін басқа файлға жазу арқылы орындалады.

6.2-мысал. Мәтіндік файлдан барлық «бос» жолдарды – символдары жоқ жолдар және босорын мен табуляция белгілері ғана бар жолдарды – алып тастайтын программа жасау керек.

Өңдеу барысында мәтіндік файлдан жолдардың белгілі бір бөлігі алынып тасталатындықтан, файлдың бос емес жолдарын сақтайтын арнайы файл жасау керек болады.

```
Program ex;
Var f1,f2:text; {мәтіндік файлдың файлдық ай-
нымалылары}
st,name:string;
Begin
WriteLn('Файл атын енгізіңіз:'); ReadLn(name);
Assign(f1,name); {файлдық айнымалыны
тағайындау}
{$I-} {файлдың бар екенін тексеру}
Reset(f1);
{$I+}
if IOResult=0 then {егер осындай атты файл бар
болса}
begin
Assign(f2,'temp.dat'); {жаңа файл
тағайындаймыз}
ReWrite(f2); {жазбалар үшін жаңа файл ашу}
while not EOF(f1) do {файл соңына жеткенше}
begin
if SeekEOLn(f1) then ReadLn(f1,st) {егер жол
бос болса,оны алып тастаймыз}
else
```

```
begin
ReadLn(f1,st); {жолды оқу}
WriteLn(f2,st); {оны жаңа файлға жазу}
end;
end;
Close(f1); {ескі файлды жабу}
Close(f2); {жаңа файлды жабу}
Erase(f1); {ескі файлды өшіру}
ReName(f2,name); {жаңа файлдың атын өзгерту}
end
else WriteLn('Мұндай аты бар файл табылмады.');
End.
```

9.5 Типсіз файлдар

Кез келген файлды ASCII код символдарының тізбегі түрінде де қарастыруға болады. Турбо Паскальда файл 128 байттық блоктардан тұратын кез келген түрде ұйымдастырылған құрылым деп саналады.

Типсіз файлдар файлдарды көшіру кезінде қолданылады, мұндайда оның жазбаларының ішкі құрылымдық ерекшеліктері қарастырылмайды. Егер дискідегі сегмент ұзындығы 1024 байт болса, онда блок көлемі 128 байттан болғанда, топтағы блоктар саны 8-ге тең.

Мәлімет алмасу тікелей программа мен файл арасында буферсіз атқарылады. Блоктарды адрестеу олардың нөмірі бойынша жүргізіледі. Мұнда блоктар файл компоненттері болып саналады. Типсіз файлдарды пайдалану компьютер жадын тиімді түрде үнемдеуге мүмкіндік береді.

Типсіз файлдармен жұмыс істеу кезінде 128 символдық блоктар тобымен мәлімет алмасатын арнайы процедуралар қарастырылған.

BLOCKREAD(<файл аты>,<айнымалы>,<компоненттер саны>
[,<нақты саны>**]);**

бұл процедура файлдан блок оқу үшін қолданылады.

BLOCKWRITE(<файл аты>,<айнымалы>,<компоненттер саны>
[,<нақты саны>**]);**

– файлға блок жазу үшін қажет.
Мұндағы:
<файл аты> - типсіз файл аты;
<айнымалы> - оқу немесе жазу үшін қолданылатын айнымалы аты;
<компоненттер саны> - бір мәлімет алмасу кезінде тасымалданатын компоненттер саны;
<нақты саны > - нақты түрде тасымалданған ұзындығы 128 байт жазбалар саны.

Блоктар арқылы енгізу-шығаруға арналған файл **FILE** типімен сипатталады. Типсіз файлдар үшін **READ** және **WRITE** процедуралары қолданылмайды.

VAR <файл аты**>: FILE;**

Типсіз файл ашқанда, файл жазбаларының ұзындығын байтпен көрсетуге болады. Ол **RESET** немесе **REWRITE** процедураларын пайдаланғанда, **WORD** типіндегі екінші параметр түрінде жазылады. Егер жазба ұзындығы көрсетілмесе, ол 128 байтқа тең болып саналады.

Блоктар арқылы енгізу-шығарудан бір мысал келтірейік. **FROMF** файлы мәліметтерін **TOF** файлына көшіру керек болсын делік.

```
VAR
FROMF, TOF: FILE;
NR, NWR: WORD;
NAME:STRING[12];
BUF: ARRAY[1..2048] OF CHAR;
BEGIN
     WRITE('ОҚЫЛАТЫН ФАЙЛ АТЫ');
     READLN(NAME);
ASSIGN(FROMF, NAME);
     WRITE('ЖАЗЫЛАТЫН ФАЙЛ Аты')$
     READLN(NAME);
ASSIGN(TOF,NAME);
RESET(FROMF, 1);
REWRITE(TOF, 1);
REPEAT
   BLOCKREAD(FROMF, BUF, SIZEOF(BUF), NR);
```

```
    BLOCKWRITE(TOF, BUF, NR, NWR);
UNTIL (NR = 0) OR (NWR <> NR);
CLOSE(FROMF);
CLOSE(TOF);
END.
```

Программада типсіз файлды ашу кезінде **RESET** процедурасында жазба ұзындығы 1 болып көрсетілген. Мұнда көшіру кезінде жаңа файлға артық символдар жазылмайды.

Бақылау сұрақтары

1. *Файл қайда орналасады және қалай белгіленеді?*
2. *Файлдармен қандай стандартты функциялар арқылы жұмыс істейміз?*
3. *Файлдармен қандай стандартты процедуралар арқылы жұмыс істейміз?*
4. *Мәтіндік файлдар қандай болады?*
5. *Файлдарды пайдаланатын құрылғылар программада қалай белгіленеді?*
6. *Файлдан мәлімет оқу үшін не істеу керек?*
7. *Файлға мәлімет жазу үшін не істеу керек?*
8. *Бейстандарт типтерді қалай және не үшін қолдануға болады?*
9. *Физикалық файл дегеніміз не?*
10. *Логикалық файл дегеніміз не?*
11. *Логикалық файлфизикалық файлмен қалай байланысады?*
12. *Файл не үшін және қалай ашылады?*
13. *Файл қалай жабылады?*
14. *Ағымдағы көрсеткіш дегеніміз не?*
15. *Файлдарды енгізу-шығару операциясын сипаттаңыз.*
16. *Типтелген файлдар арасындағы алмасуларды сипаттаңыз.*

Тапсырмалар

1. *N компоненттен тұратын бүтін сандар файлын құрастырыңыз. Файлдың тақ индексті компоненттерінің қосындысын табыңыз.*
2. *N компоненттен тұратын нақты сандар файлын құрастырыңыз. Файлдың барлық оң компоненттерін олардың квадраттарымен алмастырыңыз. Бастапқы және өңделген файлды экранға шығарыңыз.*
3. *N компоненттен тұратын бүтін сандар файлын құрастырыңыз. Файлдың ең үлкен санын анықтап, экранға шығарыңыз.*
4. *N компоненттен тұратын бүтін сандар файлын құрастырыңыз.*

Файл компоненттерін өсу реті бойынша сұрыптаңыз. Бастапқы және өңделген файлды экранға шығарыңыз.

5. *N компоненттен тұратын бүтін сандар файлын құрастырыңыз. Файлдан барлық теріс компоненттерді алып тастаңыз. Бастапқы және өңделген файлды экранға шығарыңыз.*
6. *N компоненттен тұратын f бүтін сандар файлын құрастырыңыз. Тағыда g және h файлдарын құрастырыңыз. g файлына f файлының барлық жұп сандарын, ал h файлына барлық тақ сандарды көшіріңіз. Экранға f, g және h файлдарын шығарыңыз.*
7. *N компоненттен тұратын f символдар файлын құрастырыңыз. Тағы да g файлын құрастырып, оған f файлын кері ретпен көшіріңіз. Файлдарды экранға шығарыңыз.*
8. *N компоненттен тұратын символдар файлын құрастырыңыз. Файлда жиі кездесетін символды анықтаңыз. Экранға осы символды және оның санын шығарыңыз.*
9. *N компоненттен тұратын бүтін сандар файлын құрастырыңыз. Файлдың -10-нан 10-ға дейінгі сандарын кері санға алмастырыңыз. Бастапқы және өңделген файлды экранға шығарыңыз.*
10. *N компоненттен тұратын бүтін сандар файлын құрастырыңыз. Файлдағы барлық 3-ке қалдықсыз бөлінетін сандарды олардың үш еселенген көбейтіндісімен алмастырыңыз. Бастапқы және өңделген файлды экранға шығарыңыз.*

10. МӘЛІМЕТТЕРДІҢ ДИНАМИКАЛЫҚ ҚҰРЫЛЫМЫ

10.1 Статикалық және динамикалық жады түрлері жайлы жалпы түсінік

Алдыңғы тарауларда, компиляция барысында компьютер жадынан тұрақты орын бөлінетін айнымалыларды қарастырдық. Жадының бұл аудандар (**VAR** бөлімінде сипатталған айнымалылар үшін), айнымалы керек болмаса да, программа жұмысын толық аяқтағанша сақталады. Бұл жағдайда жады тиімсіз қолданылады. Мәселен, жиымның нақты элементтер саны бойынша баптауын еске алайық немесе статикалық жады көлемі үлкен бірнеше жиым сипатталса, ал нақты бір уақытта олардың барлығы қолданылмаса.

Жағдайды, жадыны бөлудің арнайы механизімін қолданып дұрыстауға болады. Турбо Паскаль тілі, жадыны программаны орындау барысында динамикалы түрде бөліп және босатып отыруға мүмкіндік береді.

Динамикалық жадының келесі артықшылықтарын атап өтуге болады:

- үнемділігі және тиімділігі;
- байланысқан құрылымдарда элементтер санын динамикалық түрде өзгерту, мысалы, тізімдерде (статикалық жадыда элементтер саны әр компиляция үшін тұрақты);
- статикалық айнымалылар, өздері сипатталған блок жұмыс жасағанда ғана сақталады, ал динамикалық жадыда – блоктан шыққаннан кейін де, программа жұмысын аяқтағанша сақталады. Динамикалы орналасқан айнымалылар **VAR** бөлімінде сипатталмайды және олардың программада аттары болмайды. Компилятор бұндай айнымалылар үшін жадыдан орын бөлуді жоспарламайды.

10.2 Динамикалық айнымалыларды сипаттау және оларды қолдану

Программада динамикалық жады бөлігіне қол жеткізу, көрсеткіш (сілтеме) деп аталатын, арнайы сілтемелік айнымалы арқылы іске асырылады.

“Көрсеткіш” типті айнымалыда, онымен байланысқан динамикалық жады бөлімінің адресі жазылады. Компилятор “көрсеткіш” типті айнымалыға статикалық жадыдан төрт байт орнын бөледі. Әдетте, белгілі бір типтегі мәліметтермен байланысқан көрсеткіш типтелген деп аталады. Бірақ ол типсіз, кез келген типтегі мәліметтер көрсеткішімен байланысқан болуы мүмкін. Бұл жағдайда көрсеткіш бос (байлаусыз) деп аталады.

“Көрсеткіш” типінің сипатталу форматы:

TYPE <көрсеткіш идентификаторы>=^<тип>;

“Көрсеткіш” типі мен “көрсеткіш” типтегі айнымалыларды сипаттау мысалдары.

TYPE { типтерді дұрыс хабарлау }

P1=^WORD; { p1 – **WORD** типті мәндердің “көрсеткіш” типінің идентификаторы}

P2=^CHAR; { p2 - **CHAR** типті мәндердің “көрсеткіш” типінің идентификаторы}

P4=ARRAY[1..10] OF ^REAL; {p4 - **REAL** типті мәндерге сілтеме жасайтын, көрсеткіш жиымдарының “көрсеткіш” типінің идентификаторы.}

{ типтерді қате хабарлау **}**

```
   P5=^ARRAY[1..10] OF REAL;
P6=^STRING[25];
P7=^RECORD
FIELD1: STRING [15];
    FIELD2: REAL;
END;
```

“Көрсеткіш” типінің сипатталу форматында тип идентификаторы көрсетілу керек, сондықтан стандартты идентификаторларды (**INTEGER, REAL** және т.б.) бірден “көрсеткіш” типінің сипатталуында жазуға болады. **P5, P6** және **P7** типтерінің сипатталуындағы қателерді компилятор, бұндай жағдайда алдымен тип идентификаторын сипаттап, сонан кейін оны басқа сипаттауларда қолдану керек болғандақтан көрсетеді.

Келесі сипаттаулар дұрыс болады:

```
      TYPE
...
MAS=ARRAY[1..10] OF REAL;
ST=STRING[25];
```

```
REC=RECORD
FIELD1: STRING [15];
FIELD2: REAL;
END;
VAR
P5: ^MAS;
P6: ^ST;
P7: ^REC;
...
```

Көрсеткіш келесі үш қалып-күйдің бірінде болуы мүмкін:

1. әлі инициализацияланбаған;
2. орналастыру адресі бар;
3. алдын ала анықталған **NIL** тұрақтысының мәні жазылған; мұндай көрсеткіш бос көрсеткіш деп аталады, демек ешқандай айнымалыны көрсетпейді. **NIL** мәнде көрсеткіштің әр төрт байттында 0 жазылады.

Көрсеткіштерді басқа көрсеткіштермен салыстыруға (=, <>), оларға адрес беруге немесе басқа көрсеткіштің мәнін меншіктеуге, параметр ретінде беруге болады. Көрсеткішті баспадан шығаруға және экранда бейнелеуге болмайды.

Бөлінген динамикалық жадыға қол жеткізу төмендегідей кодталады:

<көрсеткіш идентификаторы >^

Динамикалық жадыда орналасқан айнымалыларға қол жеткізу мысалын қарастырайық:

```
TYPE
SYM=^CHAR;
ZAP=RECORD
FIELD1, FIELD2: REAL;
   END;
M=ARRAY[0..9] OF WORD;
VAR
CH: SYM;
REC: ^ZAP;
MAS: ^M;
...
```

CH^:='*'; {**CHAR** типті динамикалық айнымалыға қол жеткізіп, ол ауданға жұлдызша символын жазу}

...

READLN (REC^.FIELD1); {динамикалық жазбаның **FIELD1** өрісіне қол жеткізу, оған пернетақтадан мәндер енгізу}

...

WRITELN (MAS[5]^); {динамикалық жиымның **MAS[5]** элементіне қол жеткізу, көрсетілген элемент мәнін экранға шығару}

...

CH^, REC^.FIELD1 және **MAS[5]^** адрестері программада соларға сәйкес **CH, REC** және **MAS** көрсеткіштерінде жазылған динамикалық объектілер рөлін атқарады деуге болады.

POINTER типті айнымалыларға қол жеткізу (көрсеткіші ешқандай нақты типті көрсетпейтін және кез келген типтегі көрсеткіштермен үйлесімді) қатеге алып келетіндігін айта кету керек.

Например.

```
VAR
P:POINTER;
...
P^:=1; {қате!}
```

10.3 Динамикалық жадымен жұмыс істеуге арналған процедуралар мен функциялар

Динамикалық жадыға орналастыруға және босатуға арналған стандартты процедуралар

Программа орындалу барысында, динамикалық жадыны қолданатын кездер болады, мысалы: жадыны керекті түрде бөлу, ол жерге қандай да бір мәндерді орналастыру, олармен жұмыс істеу және ол мәндер керек болмаған жағдайда жадыны босату, т.с.с.

Динамикалық жадыны екі тәсілмен бөлуге болады:

1. NEW процедурасының көмегімен:

New (P);

мұндағы **P** - «типтелген көрсеткіш» типті айнымалы.

Бұл процедура жаңа динамикалық айнымалы жасайды (ол үшін жадыдан орын бөледі) және оған **P** көрсеткішін орналастырады (**P**-да жадының бөлінген ауданының адресі жазылады). Бөлінетін ауданның мөлшері мен құрылымы, **P** көрсеткіші байланысқан

типтегі мәндерге берілетін жады көлеміне байланысты. Жасалған айнымалы мәніне **P^** көмегімен қол жеткізуге болады.

2. GETMEM процедурасының көмегімен**:**

GetMem (P,size);

мұндағы **P** – қажет типтегі "көрсеткіш" типті айнымалы.

size – сұралған жады мөлшерін байтпен көрсететін бүтін санды өрнек.

Бұл процедура қажет қасиеттері және мөлшері бар жаңа динамикалық айнымалы жасайды. Айнымалының адресін "көрсеткіш" типті **P** айнымалысына орналастырады. Жаңа жасалған айнымалы мәніне қол жеткізу **P^** көмегімен іске асырылады.

Мысалы:

```
TYPE
REC=RECORD
FIELD1:STRING[30];
FIELD2:INTEGER;
  END;
PTR_REC= ^REC;
VAR
     P:PTR_REC;
BEGIN
```

GETMEM(P, SIZEOF (REC)); {жады бөлу, бөлінген аудан адресі **P-**да жазылады; бұл жадының байтпен берілген мөлшерін сипатталған типтегі мәндерге қолданылатын, стандартты **SizeOf** функциясы анықтап, қайтарады; алайда, қолданылатын өрістердің ішкі берілу мөлшерін біле отырып, жады мөлшерін қолмен есептеп, **SizeOf (Rec)** орнына тұрақты ретінде жазуға болады}

...

{жадыны пайдалану}

...

FREEMEM(P, SIZEOF(REC)); {қажет емес жадыны босату}

...

Динамикалық жадыны төрт тәсілмен босатуға болады.

1. Автоматты түрде, барлық программа аяқталғаннан кейін.
2. Стандартты **DISPOSE** процедурасының көмегімен**.**

Dispose (P);

мұндағы **P** – "көрсеткіш" типті айнымалы (типтелген).

DISPOSE(P) процедурасының жұмысының нәтижесінде, **P** көрсеткішімен байланысқан жады ауданы, алдағы уақытта қолдануға

болатын бос аудан ретінде белгіленеді. Бірақ **P** көрсеткіші және онымен байланысқан жады физикалық түрде тазаланбайды, жазба экземплярын өшіріп тастап та оның өрістерінің мәнін алуға болады, алайда бұл жағдайды қолдануға кеңес берілмейді.

DISPOSE *процедурасының қолдану тәсілдері әр түрлі боғандықтан, бұл процедураны* ***MARK*** *және* ***RELEASE*** *процедураларымен бірге қолдануға болмайды.*

2. FREEMEM процедурасының көмегімен.

FreeMem (P, size);

мұндағы **P** - "көрсеткіш" типті айнымалы,

size – босатылатын жады мөлшерін байтпен көрсететін бүтін санды өрнек.

Бұл процедура жадыны, **P** көрсеткішімен байланысқан, **SIZE** өрнегінің мәнінен тең мөлшермен, бос жады ретінде белгілейді (**GETMEM** мысалын қараңыз)**.**

4. Стандартты **MARK** және **RELEASE** процедуралары көмегімен.

Mark (P);

Release (P);

мұндағы **P** - "көрсеткіш" типті айнымалы;

MARK - **P** көрсеткіш-айнымалысында динамикалық ауданның қалып-күйін есте сақтайды;

RELEASE – **MARK** процедурасы **P** көрсеткішінің ағымдағы мәнін есте сақтағаннан кейін **NEW** немесе **GETMEM** процедуралары бөлген, барлық динамикалық жадыны босатады.

Қолдану тәсілдері әр түрлі болғандықтан, **MARK** және **RELEASE** процедураларына қол жеткізуді **DISPOSE** және **FREEMEM** процедураларына қол жеткізумен кезектестіруге болмайды.

Мысалы:

```
VAR
P:POINTER;
P1,P2,P3:^INTEGER;
BEGIN
NEW(P1);
P1^:=10;
MARK(P); {динамикалық ауданды белгілеу}
NEW(P2);
```

```
P2^:=25;
NEW(P3);
P3^:=P2^+P1^;
WRITELN (P3^);
RELEASE(P); {P2^ және P3^ байланысқан жады бо-
сатылды, ал  P1^ қолданылуы мүмкін}
END.
```

Динамикалық жадыны өңдеуге арналған стандартты функциялар

Программа орындалу барысында динамикалық жады қалыпкүйін бақылау қажет болуы мүмкін. Мұндай бақылау мақсаты – келесі бөлінетін, қажет мөлшердегі динамикалық ауданның мүмкіндіктерін бағалау. Мұндай мақсаттар үшін Турбо Паскаль екі функция бар (параметрсіз).

MaxAvail;

Бұл функция динамикалық ауданның дәл осы мезетте бос ең үлкен бөлігінің байтпен берілген мөлшерін қайтарады. Осы мөлшер бойынша динамикалық жадының бөлінетін ең үлкен мөлшері жайлы айтуға болады.

Қайтарылатын тип мәні - **longint**.

```
TYPE ZAP=RECORD
       FIELD1: STRING [20];
       FIELD2: REAL;
         END;
VAR P: POINTER;
BEGIN
...
IF MAXAVAIL <SIZEOF(ZAP)
THEN
WRITELN ('НЕ ХВАТАЕТ ПАМЯТИ!')
ELSE
GETMEM(P, SIZEOF(ZAP));
...
```

Екінші функция:

MemAvail;

Бұл функция динамикалық жадының бос байттарының жалпы санын қайтарады, демек, барлық бос бөліктердің мөлшері және бос динамикалық аудан мөлшері қосылады. Қайтарылатын тип мәні - **longint**.

```
...
WRITELN( 'бос', MEMAVAIL, ' байт' );
WRITELN('ең үлкен бос бөлік=', MAXAVAIL, 'байт' );
...
```

Бұл шешім келесі жағдайға негізделген. Динамикалық аудан, "түйдек" (**HEAP**) деген аты бар арнайы бөлінген ауданда орналасады. Түйдек, программа іске қосылғаннан кейінгі бос жадыны толығымен немесе оның бір бөлігін алады. Түйдек мөлшері компилятордың **$M** директивасы арқылы беруге болады:

{$M <стек>, <түйдектің ең кішісі>, <түйдектің ең үлкені>**}**

мұндағы <стек> - стек сегментінің байтпен берілген мөлшерін көрсетеді. Үнсіз келісім бойынша стек мөлшері 16 384 байт, ал стектің ең үлкен мөлшері 65 538 байт;

<түйдектің ең кішісі> - түйдектің байтпен берілген ең кіші мөлшерін көрсетеді; үнсіз келісім бойынша ең кіші мөлшер 0 байт;

<түйдектің ең үлкені> - түйдектің байтпен берілген ең кіші мөлшерін көрсетеді; үнсіз келісім бойынша ең үлкен мөлшер 655 360 байт, көп жағдайда түйдекке барлық бос жадыны береді; бұл мән түйдектің ең кіші мөлшерінен кем болмауы керек.

Барлық мәндер ондық немесе он алтылық формада беріледі. Мысалы, келесі екі директива эквиваленті:

```
{$M 16384,0,655360}
{$M $4000, $0, $A000}
```

Егер жадының көрсетілген ең кіші көлемі бос болмаса, онда программа орындалмайды.

Динамикалық жадыға орналастыру ***System*** модулінің басқару программаларының бірі болып табылатын топ администраторы арқылы іске асырылады.

Бақылау сұрақтары

1. *Сілтемелік типтер мен айнымалылар қалай хабарланады?*
2. *Жадыны динамикалық айнымалыларға бөлу қандай процедуралар көмегімен іске асырылады?*
4. *Сілтемелік айнмалы қандай мәндер қабылдай алады?*
5. *nil сәлтемелік айнымалысының мәні мен анықталмаған мәннің аырмашылғы неде?*

6. *new және dispose процедураларының қызметі қандай?*
7. *Көрсеткішті алу операциясы қалай орындалады?*
8. *Көрсеткіштермен қандай операциялар орындауға болады?*
9. *Статикалық айнымалының динамикалық айнымалыдан айырмашылығы неде?*

Тапсырмалар

1. *Пернетақтадан енгізілген төрт санның ең үлкенін таңдап алу программасын құрастырыңыз.*
2. *15 бүтін элементтен тұратын жиым мәндері бойынша графика бейнесін алатын программа құрыңыз.*
3. *Келесі өрістерден тұратын жазба жазылған файл құрастырыңыз: 1) нөмірі, 2) тегі, 3) телефоны. Пернетақтадан енгізілген тегіне сәйкес телефон нөмірін экранға шығарыңыз.*
4. *Көлемі 4x5 жиымды толтыратын және бағанасы бойынша сұрыптайтын программа құрыңыз. Бастапқы және өңделген жиымды экранға шығарыңыз.*
5. *Көлемі 4x5 жиымның барлық теріс элементтерін олардың абсолют шамасымен алмастыратын программа құрыңыз. Бастапқы және өңделген жиымды экранға шығарыңыз.*
6. *N бүтін саннан тұратын файл құрастырыңыз. Файлдың ең үлкен, ең кіші компонентін және олардың позицияларның нөмірін экранға шығарыңыз.*
7. *Пернетақтадан енгізілген кезейсоқ текке де, атқа да кіретін символдар жиынын экранға шығарыңыз.*

РЕДАКТОР

Редактормен жұмыс жасаудың негізгі жолдары бірінші тарауда айтылған. Төменде редактор командаларыының толық сипаттамасы берілген. Барлық командаларды курсорды жылжыту, өшіру/кірістіру, блоктармен жұмыс жасау командалары деп бірнеше топтарға бөлуге болады.

Сипаттау барысында курсорды басқару пернелеріне келесідей белгілеулер қолданылады:

ВЛ курсорды солға жылжыту; ВВ курсорды жоғары жылжыту;
ВП курсорды оңға жылжыту; ВН курсорды төмен жылжыту.

П1.4.1. **Курсорды жылжыту командалары**

Ctrl-S немесе *ВЛ* - бір символ символға;
Ctrl-D немесе *ВП* - бір символ оңға;
Ctrl-A немесе *Ctrl-ВЛ* – бір сөз солға;
Ctrl-F немесе *Ctrl-ВП* - бір сөз оңға;
Ctrl-E немесе *ВВ* – бір жол жоғары;
Ctrl-X немесе *ВН* - бір жол төмен;
Ctrl-W – курсормен бірге бір жол төмен жылжыту;
Ctrl-Z - курсормен бірге бір жол жоғары жылжыту;
Ctrl-R немесе *PgUp* – бір парақ жоғары;
Ctrl-C немесе *PgDn* – бір парақ төмен;
Ctrl-Q S немесе *HOME* – жолдың басына;
Ctrl-Q D немесе *END* – жолдың соңына;
Ctrl-Q E немесе *Ctrl-HOME* – экран басына;
Ctrl-Q X немесе *Ctrl-END* – экран соңына;
Ctrl-Q R немесе *Ctrl-PgUp* – файлдың басына;
Ctrl-Q C немесе *Ctrl-PgDn* – файлдың соңына;
Ctrl-Q B – блоктың басына;
Ctrl-Q K- блоктың соңына;

Ctrl-Q P – ең соңғы позицияға (іздеуден немесе іздеу/алмастырудан кейін қолданылады);

Ctrl-Q W – ең соңғы қатеге.

П1.4.2**. Өшіру/қою командалары**

Ctrl- V немесе *INS* – кірістіру режимін қосу/алып тастау;
Ctrl-N – бос жол кірістіру;
Ctrl- Y – жолды өшіру;
Ctrl-H немесе *Backspace* – курсордың сол жағындағы символды өшіру;
Ctrl-G немесе *DEL* – курсор тұрған жердегі символды өшіру;
Ctrl-T- курсордың оң жағындағы сөзді өшіру;

Ctrl-Q Y – курсордың оң жағындағы жол қалдығын өшіру.

П1.4.3. **Блоктармен жұмыс жасауға арналған командалар**

Программа мәтінін даярлаған кезде мәтін фрагменттерін басқа орынға көшіру немесе мүлдем өшіріп тастау керек жағдайлар болады. Осындай операцияларды орындау үшін блоктарды – мәтін фрагментін қолданған ыңғайлы. Блоктың ұзындығы бірнеше экран бетін алуы мүмкін (64 Кбайтқа дейін). Редактор экранында бір мезгілде бір ғана блокты белгілеуге болады. Терезелер арасында блоктармен алмасу үшін редактор буферін қолданамыз (негізгі менюдің EDIT опциясы).

Ctrl-K B – Блок басын белгілеу;

Ctrl-K K – Блок соңын белгілеу;

Ctrl-K T – Блок ретінде курсордың сол жағындағы сөзді белгілеу;

Ctrl-K P – блок баспадан шығару;

Ctrl-K C – блокты курсор тұрған жерден бастап көшіру;

Ctrl-K K – блоктың орнын ауыстыру;

Ctrl-K H – блоктың бояуын алып тастау. Осы пернелерді қайта басу блокты қайтадан бояу;

Ctrl-K Y – блокты өшіру;

Ctrl-K R – дискілік файлдан блокты оқу;

Ctrl-K W – блокты дискіге жасу;

Ctrl-K I - блокты оңға жылжыту;

Ctrl-K U - блокты солға жылжыту.

П1.4.4. **Басқа командалар**

Ctrl-Q F – үлгі бойынша іздеу;

Ctrl-L – іздеуді жалғастыру;

Ctrl-Q A – үлгі бойынша іздеп тауып алмастыру;

Ctrl-U – іздеуді тоқтату;

Ctrl-K n – маркерді орнату; n = 0..9 (төмендегіні қара);

Ctrl-Q n - маркерді іздеу;

Ctrl-Q W – қатені іздеу;

Ctrl-Q [- қос жақшаның оң жақ жұбын іздеу; (төмендегіні қара);

Ctrl-Q] - қос жақшаның сол жақ жұбын іздеу; (төмендегіні қара);

Ctrl-Q L – бүлінген жолды қалпына келтіру;

Ctrl-Q T немесе *Ctrl-O T* – табуляцияны іске қосу/алып тастау;

Ctrl-O F – табуляция толтыруын ауыстыру;

Ctrl-O I немесе *Ctrl-Q I* – автошегіністі іске қосу/алып тастау;

Ctrl-O O – компилятор баптауларын файл басына орналастыру (төмендегіні қара).

Командалардың көпшілігінің қызметі түсінікті. Алайда кейбір командалардың қызметіне түсініктеме беру керек.

Ctrl-K n. Нөмірі n = 0..9 маркердің ағымдағы орнын анықтайды. Маркер экранда көрінбейді және ол программаның орындалуына еш кедергі жасамайды. Үлкен программалар жазған кезде, қажет фрагменттерді жылдам тауып алу үшін, команда *Ctrl-Q n* командасымен бірге орындалады. *Ctrl-K n* пернелер комбинациясының терілуіне назар аударыңыз – алдымен Ctrl пернесін басыңыз, пернені басулы күйде ұстап тұрып К пернесін басыңыз; бұдан кейін пернелерді жіберіп *n* цифрын басыңыз. *Ctrl-Q n* командасы да дәл осылай теріледі.

Ctrl-Q] және *Ctrl-Q [*. Бұл командалар жақшаларды іздеу үшін қолданылады. Командалар (және), {және}, [және] қос жақшалардың жұбын іздеуге мүмкіндік береді. Курсорды жақшаға орналастырып сәйкес команданы берсеңіз – редактор сол жақшаның екіншісін тауып береді.

Ctrl-O O. Бұл команда, біріктірілген ортаның ағымдағы баптаулары компилятор дерективалары, мысалы:

```
{$A+,B-,D+,E+,F+,G+,I+,L+,N+,0-,R + ,S+,V+,X+}
{$M 16384,0,655360}
```

`{$DEFINE single}` түрінде жазылған, файл басына көшуге мүмкіндік береді.

П1.4.5. **Біріктірілген ортаға редактордан берілетін командалар**

Кейбір жиі қолданылатын командаларлды біріктірілген ортаға бірден редакциялау режимінен беруге болады. Олардың барлығы жоғарыда менюлер жүйесінде қарастырылды. Түсінікті болу үшін оларды тағы бір қайталаймыз.

F1 – анықтама алу;

F2 – редактор терезесіндегі файлды дискіге жазу;

F3 – дисктегі файлда редактор терезесінде оқу;

F4 – курсорға дейін орындау (RUN/GO TO CURSOR командасы);

F5 – терезені бүкіл экранға орындау немесе терезені бастапқы қалпына келтіру;

F6 – келесі терезені екпінді ету;

F7 – процедураны қадағалау (RUN/TRACE INTO командасы);

F8 – процедураны орындамай жіберу (RUN/STEP OVER командасы);

F9 – программаны компиляциядан өткізу (COMPILE/MAKE командасы);

F10 – негізгі менюге көшу;

Ctrl-F1 – жанама анықтама алу;

Ctrl-F2 – түзету режимін алып тастау;

Ctrl-F3 – программалық стек терезесін екпінді ету;

Ctrl-F4 - өрнекті есептеу немесе айнымалына көрсету/өзгерту (DEBUG/EVALUATE/MODIFY командасы);

Ctrl-F5 – терезенің орны мен көлемін өзгерту режиміне ауысу;

Ctrl-F7 – түзету терезесіне өрнек қосу (DEBUG/WATCHES/ADD WATCH командасы);

Ctrl-F8 – бақылау нүктесіналмастыру;

Ctrl-F9 – программаны компиляциядан өткізіп, орындау;

Ctrl-Del - редактор буферін тазалау;

Ctrl-Ins – блокты редактор буферіне көшіру;

Alt-C - COMPILE компиляциялау менюін шақыру;

Alt-D - DEBUG түзету менюін шақыру;

Alt-E - EDIT редакциялау менюін шақыру;

Alt-F - FILE файлдық қызмет менюін шақыру;

Alt-H - HELP анықтама қызметінің менюін шақыру;

Alt-O - OPTIONS баптаулар менюін шақыру;

Alt-R - RUN программаны орындау менюін шақыру;

Alt-S - SEARCH іздеу қызметінің менюін шақыру;

Alt-W- WINDOW терезелер менюін шақыру;

Alt-X- Турбо Паскальдан шығу;

Alt-0 – ашық терезелер тізімін алу;

Alt-F1 – соңғы анықтаманы алу;

Alt-F3 – екпінді терезені жабу;

Alt-F5 – программа терезесін көрсету;

Alt-F9 - COMPILE/COMPILE командасын орындау;

Shift-F1 – анықтама қызметінің сілтемелер тізімін алу;

Shift-F6 – алдыңғы терезені екпінді ету;

Shift-Del – редактор терезесіндегі блокты буферге тасмалдау;

Shift-Ins – буфердегі фрагментті редактор терезесіне көшіру.

ТУРБО-ПАСКАЛЬДІ ШАҚЫРУ

Турбо Паскальды іске қосатын DOS командасының толық форматы:
[PATH]turbo [Options] [FileName]
Мұндағы PATH – жүйелік каталогқа апаратын жол;
Optoins – опциялар тізімі;
FileName- PAS-файлдың аты.
Тік жақшада команданың міндетті емес параметрлері көрсетілген.

FileName параметрі - экранда оқылатын файл атын көрсетеді. Егер бұл параметр көрсетілсе, онда аты көрсетілген файл, Турбо Паскаль іске қосылғанда автоматты түрде редактор терезесінде ашылады. Егер файл атында кеңейтілім көрсетілмесе, біріктірілген орта стандартты .PAS кеңейтілімін қосады.

Options көмегімен біріктірілген орта баптауларын өзгертуге болады. Әр параметр /Z форматында беріледі, мұндағы Z – біріктірілген ортаның баптауларын анықтайтын әріп (төмендегіні қараңыз); «+» (баптауды іске қосу) немесе «-» (баптауларды алып тастау) белгілері; не бірнеше параметрлер бір-бірінен бос орын арқылы ажыратылып жазылады.

Options тізімінде келесідей басқарушы параметрлерді қолдануға болады:

/C<файл аты> - қажет конфигурациядағы файлды жүктеу (біріктірілген ортаның OPTIONS/OPEN командасына сәйкес); < файл аты > -конфигурациялық файл аты, мысалы: /CMy. tp;

/D+ - қосымша мониторды қолдану (OPTIONS/ENVIRONMENT/STARTUP/DUAL MONITOR SUPPORT командасына сәйкес);

/E<көлем> - экран буфері үшін қажет көлемдегі жады бөлу (OPTIONS/ENVIRONMENT/STARTUP/EDITOR HEAP SIZE командасына сәйкес); <көлем> - бөлінетін жады көлемі (килобайт); мысалы: /E15 – экран буфері үшін 15 Кбайт жады бөлу;

/G+ - графикалық экран көшірмесін сақтау (OPTIONS/ENVIRONMENT/STARTUP/GRAPHICS SCREENSAVE командасына сәйкес);

/L+ - ДК сұйықкристалды дисплеймен жабдықталған (OPTIONS/ENVIRONMENT/STARTUP/LCD COLOR SET командасына сәйкес);

/N+ - ДК CGA типті адаптермен жабдықталған (OPTIONS/ENVIRONMENT/ STARTUP/CGA SNOW CHECKING командасына сәйкес);

/O<размер> - Турбо Паскаль жүйесінің оверлейлік модульдерін сақтауға қажет жады көлемін орнату (OPTIONS/ENVIRONMENT/STARTUP/ OVERLAY HEAP SIZE командасына сәйкес);

/P+ - экранның түстер палитрасын сақтау (OPTIONS/ ENVIRONMENT/ STARTUP/EGA/VGA PALETTE SA УЕ командасына сәйкес);

/S<путь> - «жылдам» дискіге жолды анықтау (OPTIONS/ENVIRONMENT/ STARTUP/SWAP FILE DIRECTORY командасына сәйкес);

/T+ - TURBO. TPL файлынан жадыға SYSTEM. TPU кітапханасын жүктеу (OPTIONS/ENVIRONMENT/STARTUP/LOAD TURBO. TPL командасына сәйкес);

/W<көлем> - Турбо Паскаль терезелерін сақтауға қажет жады көлемін орнату (OPTIONS/ENVIRONMENT/STARTUP/WINDOW HEAP SIZE командасына сәйкес);

/X+ - EMS-жадыны қолдану (OPTIONS/ENVIRONMENT/ STARTUP/USE EXPANDED MEMORY командасына сәйкес).

ДК ТАҢБА ГЕНЕРАТОРЛАРЫНЫҢ КОДТАУ НҰСҚАЛАРЫ

IBM PC ДК стандартты таңба генераторы символдарды Қ2.1-суретте көрсетілгендей кодтайды. Кодтары 0-ден 127-ге дейінгі таңба генераторының бірінші жартысын құрайтын символдар (Қ2.1, а-сурет), ASCII стандарты бойынша құрастырылған және барлық IBM-үйлесімді ДК үшін бірдей. Екінші символдар бөлігі (128...255 дейінгі кодтар) әр типті ДК үшін әр түрлі болуы мүмкін. IBM фирмасының стандартты таңба генераторында псевдографика символдары қатар орналасқан үш бағанада (176...223 кодтары, Қ2.1, ә-сурет) берілген. 128-ден 175-ке дейінгі және 224-тен 239-ға дейінгі кодтар жазылған бағаналар кейбір ұлттық символдарды жазу үшін қолданылады. Ал соңғы бағана (240...255 кодтар) – арнайы белгілерді жазу үшін қолданылады. Шет елдік программалардың көбі символдардың осы орналасуын негізге ала отырып құрастырылады.

	000	016	032	048	064	080	096	112	
00		►		0	@	P	`	p	00
01	☺	◄	!	1	A	Q	a	q	01
02	☻	↕	"	2	B	R	b	r	02
03	♥	‼	#	3	C	S	c	s	03
04	♦	¶	$	4	D	T	d	t	04
05	♣	§	%	5	E	U	e	u	05
06	♠	▬	&	6	F	V	f	v	06
07	•	↨	'	7	G	W	g	w	07
08	◘	↑	(	8	H	X	h	x	08
09	○	↓	)	9	I	Y	i	y	09
10	◙	→	*	:	J	Z	j	z	10
11	♂	←	+	;	K	[	k	{	11
12	♀	∟	,	<	L	\	l	¦	12
13	♪	↔	-	=	M	]	m	}	13
14	♫	▲	.	>	N	^	n	~	14
15	*	▼	/	?	O	_	o	⌂	15

а)

	128	144	160	176	192	208	224	240	
00	Ç	É	á	░	└	╨	α	≡	00
01	ú	æ	í	▒	┴	╤	β	±	01
02	é	Æ	ó	▓	┬	╥	Γ	≥	02
03	â	ô	ú	│	├	╙	π	≤	03
04	ä	ö	ñ	┤	─	╘	Σ	⌠	04
05	à	ò	Ñ	╡	┼	╒	σ	⌡	05
06	å	û	ª	╢	╞	╓	µ	÷	06
07	ç	ù	º	╖	╟	╫	γ	≈	07
08	ê	ÿ	¿	╕	╚	╪	¤	°	08
09	ë	Ö	⌐	╣	╔	┘	Θ	∙	09
10	è	Ü	¬	║	╩	┌	Ω	·	10
11	ï	¢	½	╗	╦	█	δ	√	11
12	î	£	¼	╝	╠	▄	∞	ⁿ	12
13	ì	¥	¡	╜	═	▌	φ	²	13
14	Ä	Pt	«	╛	╬	▐	ε	■	14
15	Å	f	»	┐	╧	▀	∩		15

ә)

Қ2.1- сурет. IBM фирмасының стандартты кодтары: а) 0...127 кодтары үшін; ә) 128...255 кодтары үшін

Біздің елде қолданылатын стандартты таңбагенераторы Халықаралық телеграфия және телефония (МККТТ) бойынша кеңес беретін комитеттің ұсыныстары бойынша жасалған. Таңба генератор кестесінің екінші жартысындағы символдардың орналасуы (Қ2.2, а-сурет) IBM фирмасының таңбагенераторынан мүлдем өзгеше. Бұл ерекшелік біздің елдің ДК-де шет ел программалық жабдықтамаларын қолдануда біраз қиындықтар тудырады. Сондықтан стандартты (ГОСТ) кодтау нұсқалары альтернативті (Қ2.2, ә-сурет) кодтармен алмастырылады, оның басты артықшылығы - псевдографика символдарының IBM таңба генераторымен сәйкес орналасуы. Ал кемшілігі – орыс алфавитінің символдары үздіксіз тізбек құрамайды. Қ2.2, ә-суретінде көрсетілген нұсқа, қазіргі кезде біздің елдің ДК-де кеңінен тараған. Біздің елдің сұранысы бойынша экспортқа компьютерлер даярлайтын шет ел фирмалары үшін осы кесте стандарт болып алынады. Сонымен бірге, біздің барлық программалық жабдықтамаларымыз да осыған негізделген.

	128	144	160	176	192	208	224	240	
00	╨	╔	┌	А	Р	а	р	Ё	00
01	╩	╕	┐	Б	С	б	с	ё	01
02	╥	╖	└	В	Т	в	т	⁄	02
03	┼	╙	┴	Г	У	г	у	ˋ	03
04	╫	═	─	Д	Ф	д	ф	ˊ	04
05	╖	╟	├	Е	Х	е	х	ˎ	05
06	╕	╤	┬	Ж	Ц	ж	ц	→	06
07	╤	╢	┤	З	Ч	з	ч	←	07
08	╘	╧	┴	И	Ш	и	ш	↓	08
09	╙	╞	├	Й	Щ	й	щ	↑	09
10	╒	╪	┼	К	Ъ	к	ъ	÷	10
11	╗	▒	▀	Л	Ы	л	ы	±	11
12	╛	▒	▄	М	Ь	м	ь	№	12
13	╞	▓	▌	Н	Э	н	э	¤	13
14	╪	╪	▌	О	Ю	о	ю	■	14
15	╒	┼	▄	П	Я	п	я		15

а)

	128	144	160	176	192	208	224	240	
00	А	Р	а	▒	└	╨	р	≡	00
01	Б	С	б	▒	┴	╥	с	±	01
02	В	Т	в	▓	┬	╓	т	≥	02
03	Г	У	г	│	├	╙	у	≤	03
04	Д	Ф	д	┤	─	┘	ф	⌠	04
05	Е	Х	е	╡	┼	┌	х	⌡	05
06	Ж	Ц	ж	╢	╞	╓	ц	÷	06
07	З	Ч	з	╖	╟	╫	ч	≈	07
08	И	Ш	и	╕	╘	╪	ш	°	08
09	Й	Щ	й	╣	╒	┘	щ	∙	09
10	К	Ъ	к	║	╧	┌	ъ	·	10
11	Л	Ы	л	╖	╤	▀	ы	√	11
12	М	Ь	м	╗	╟	▄	ь	ⁿ	12
13	Н	Э	н	╗	═	▌	э	²	13
14	О	Ю	о	┘	╪	▌	ю	■	14
15	П	Я	п	┐	┴	▄	я		15

ә)

Қ2.2-сурет. 128...255 кодтары үшін кодтау нұсқалары:
а) МККТТ ұсынысы бойынша; ә) кең тараған

ҚАТЕЛЕР КОДЫ ЖӘНЕ МӘЛІМЕТТЕР

1. Компиляциядан өткізу кезіндегі қателер жайлы мәліметтер

Турбо Паскаль интегралданған ортасы программа компиляторлардан өткен кезде пайда болған қателер жайлы толық мәлімет береді. Қате кездескен кезде интегралданған орта автоматты түрде бастапқы программа мәтінін экранға шығарып, курсорды қате табылған орынға орналастырады және редактордың жоғарғы жолында диогностикалық мәлімет пайда болады. F1 пернесінен басқа кез келген пернеге бассаңыз жоғарғы жол бастапқы қалпына келіп, интегралданған орта редакциялау режиміне ауысады. Егер қате жайлы мәлімет шыққаннан кейін F1 пернесіне бассаңыз, экранда сол қатені түзету жайлы нұсқаулар жазылған анықтама қызметінің терезесі пайда болады. Кейбір қателер бірден емес, программа мәтінін талдау барысында анықталады. Мысалы, меншіктеу операторындағы типтердің сәйкессіздігін осы оператордың оң жағындағы өрнекті толығымен есептегенше анықтау мүмкін емес.

Қате №	Аты
1	*Out of memory (Жады шекарасынан асып кету).*
2	*Identifier expected (Идентификатор көрсетілмеген).*
3	*Unknown identifier (Белгісіз идентификатор).*
4	*Duplicate identifier (Идентификатор екі рет қайталанған).*
5	*Syntax error (Синтаксистік қате).*
6	Error in real constant (*Нақты типтегі тұрақтыда қате бар*).
7	*Error in integer constant (Бүтін типтегі тұрақтыда қате бар).*
8	*String constant exceeds line (Жолдық тұрақты берілген аймақтан тыс жатыр).*
9	*Too many nested files (Кіріктірілген файлдар саны тым көп).*
10	*Unexpected end of file (Файл соңы табылмады).*

11	*Line too long (Тым ұзын жол)*
12	*Type identifier expected (Бұл жерде тип идентификаторы қажет).*
13	*Too many open files (Ашылған файлдар саны көп).*
14	*Invalid file name (Файл аты дұрыс емес).*
15	*File not found (Файл табылмады).*
16	*Disk full (Диск толып кеткен).*
17	*Invalid compiler directive (Компилятор дерективасы дұрыс емес).*
18	*Too many files (Файлдар саны тым көп).*
19	*Undefined type in pointer definition (Көрсеткіш сипаттауында тип көрсетілмеген).*
20	*Variable identifier expected (Айнымалының иденфикаторы жоқ).*
21	*Error in type (Типті сипаттауда қате жіберілген).*
22	*Structure too large (Тым үлкен құрылым).*
23	*Set base type of range (Жиынның базалық типі шекарадан тыс жатыр).*
24	*File components may not be files (Файл компоненті ретінде файлды қолдануға болмайды) .*
25	*Invalid string length (Жол ұзындығы дұрыс емес).*
26	*Type mismatch (Типтер сәйкестігі).*
27	*Invalid subrange base type (Тип-диапозон үшін базалық тип дұрыс көрсетілмеген).*
28	*Lower bound greater than upper bound (Төменгі шекара жоғарғы шекарадан үлкен).*
29	*Ordinal type expected (Реттік тип қажет).*
30	*Integer constant expected (Бүтін тұрақты қажет).*
31	*Constant expected (Тұрақты қажет).*
32	*Integer or real constant expected (Бүтін немесе нақты тұрақты қажет) .*
33	*Type identifier expected (Тип идентификаторы қажет).*
34	*Invalid function result type (Функция нәтижесінің типі дұрыс көрсетілмеген).*
35	*Label identifier expected (Белгі идентификаторы қажет).*
36	*BEGIN expected (BEGIN жазылмаған).*
37	*END expected (END жазылмаған).*
38	*Integer expression expected (INTEGER типті өрнек керек).*
39	*Ordinal expression expected (Саналатын типтегі өрнек керек).*
40	*Boolean expression expected (BOOLEAN типіндегі өрнек керек).*

41	*Operand types do not match operator (Операндтар типі орындалатын операцияға сәйкес емес).*
42	*Error in expression (Өрнек жазылуында қате бар).*
43	*Illegal assignment (Меншіктеу дұрыс емес).*
44	*Field identifier expected (Өріс идентификаторы қажет) .*
45	*Object file too large (Объекті файлы тым үлкен).*
46	*Undefined external (Сыртқы процедура анықталмаған).*
47	*Invalid object file record (Объекті файлы дұрыс жазылмаған).*
48	*Code segment too large (Сегмент коды тым үлкен).*
49	*Data segment too large (Мәндер сегменті тым үлкен).*
50	*DO expected (DO операторы қажет).*
51	*Invalid PUBLIC definition (PUBLIC-анықтама дұрыс емес).*
52	*Invalid EXTRN definition (EXTRN-анықтама дұрыс емес).*
53	*Too many EXTRN definition (EXTRN-анықтамалар саны көп).*
54	*OF expected (OF қажет).*
55	*INTERFACE expected (Интерфейстік секция қажет).*
56	*Invalid relocatable reference (Ауыспалы сілтеме дұрыс емес).*
57	*THEN expected (THEN қажет).*
58	*TO or DOWNTO expected (TO немесе DOWNTO қажет).*
59	*Undefined forward (Озбалы (опережающее) сипаттама анықталмаған).*
60	*Too many procedures (Процедуралар саны көп).*
61	*Invalid typecast (Типті түрлендіру дұрыс емес).*
62	*Division by zero (Нөлге бөлу).*
63	*Invalid file type (Файлдық тип дұрыс емес).*
64	*Cannot Read or Write variables of this type (Берілген типтегі мәліметтерді санау немесе жазу мүмкін емес).*
65	*Pointer variable expected (Айнымалы-көрсеткіш қолдану керек).*
66	*String variable expected (Жолдық айнымалы қажет).*
67	*String expression expected (Жолдық типтегі өрнек қажет).*
68	*Circular unit reference (Модульдердің қиылысқан сілтемесі).*
69	*Unit name mismatch (Программалық модульдер аты сәйкес емес).*
70	*Unit version mismatch (Модульдер версиясы сәйкес емес).*
71	*Duplicate unit name (Программалық модуль аты қайталанған).*
72	*Unit file format error (Модуль файлының форматы қате).*

73	*IMPLEMENTATION expected (Модульдің орындалатын бөлігі жоқ).*
74	*Constant and case types do not match (Тұрақты типі мен CASE операторындағы өрнек типі бір біріне сәйкес емес).*
75	*Record variable expected (Жазба типті айнымалы қажет).*
76	*Constant out of range (Тұрақты шекарадан тыс жатыр).*
77	*File variable expected (Файлдық айнымалы қажет).*
78	*Pointer expression expected (Көрсеткіш типті өрнек керек).*
79	*Integer or real expression expected (Нақты немесе бүтін типті өрнек керек).*
80	*Label not within current block (Белгі ағымдағы блок ішінен тыс жатыр).*
81	*Label already defined (Белгі анықталған).*
82	*Undefined label in processing statement part (Алдыңғы операторлар бөлігінде белгі анықталмаған).*
83	*Invalid @ argument (@ операциясының аргументі дұрыс емес).*
84	*Unit expected (UNIT кодтық сөзі керек).*
85	*”;” expected (”;” қою керек).*
86	*”:” expected (”:” қою керек).*
87	*”,”expected (”,” қою керек).*
88	*”(” expected (”(” қою керек).*
89	*”)” expected (”)” қою керек).*
90	*”=” expected (”=” қою керек).*
91	*”:=” expected (”:=” қою керек).*
92	*”[” or ”(.”expected (”[” немесе ”(.” қою керек).*
93	*”]” or ”.)” expected (”]” немесе ”.)” қою керек).*
94	*”.” expected (”.” қою керек).*
95	*”..” expected (”..” қою керек).*
96	*Too many variables (Айнымалылар саны көп).*
97	*Invalid FOR control variable (FOR қайталану операторының параметрі дұрыс емес).*
98	*Integer variable expected (Бүтін типті айнымалы қажет).*
99	*File and procedure types are not allowed here (Бұл жерде файлдық немесе процедуралық типтер қолдануға болмайды).*
100	*String length mismatch (Жол ұзындығы сәйкес емес).*
101	*Invalid ordering of fields (Өрістер реті дұрыс емес).*
102	*String constant expected (Жолдық типтегі тұрақты керек).*
103	*Integer or real variable expected (INTEGER немесе REAL типті айнымалы қажет).*

104	*Ordinal variable expected (Реттік типтегі айнымалы керек).*
105	*INLINE error (INLINE операторында қате бар).*
106	*Character expression expected (Алдыңғы өрнек символдық типті болу керек).*
107	*Too many relocation items (Ауыспалы элементтер саны көп).*
108	*Overflow in arithmetic operator (Арифметикалық операторды орындау кезінде нәтиже берілген шекарадан шығып кетті).*
109	*No enclosing FOR, WHILE or REPEAT statement (FOR, WHILE немесе REPEAT операторларын аяқтайтын операторлар жоқ).*
110	*Debug information table overflow (Қалыпқа келтіру (отладка) информациялық кестесі берілген аймақтан шығып кеткен).*
111	N/A
112	*CASE constant out of range (CASE тұрақтысы берілген шекарадан тыс) .*
113	*Error in statement (Операторда қате бар).*
114	*Cannot call an interrupt procedure (Үзіліс процедурасын шақыру мүмкін емес).*
115	N/A
116	*Must be in 8087 mode to compile this (Компиляция үшін 8087 режимі керек).*
117	*Target address not found (Көрсетілген адрес табылмады).*
118	*118 Include files are not allowed here (Бұл жерде қосылатын (включаемые) файлды қолдануға болмайды).*
119	*No inherited methods are accessible here (Программаның көрсетілген жерінде мұрагерлік (унаследованных) тәсіл жоқ).*
120	N/A
121	*Invalid qualifier (Квалификатор дұрыс жазылмаған).*
122	*Invalid variable reference (Айнымалыға жасалған сілтеме нақты емес).*
123	*Too many symbols (символдар саны көп).*
124	*Statement part too large (Операторлар бөлімі тым үлкен).*
125	N/A
126	*Files must be var parameters (Файлдар параметр-айнымалы түрінде тасмалдануы керек).*
127	*Too many conditional symbols (Шартты символдар саны көп).*

128	*Misplaced conditional directive (Шартты деректива қалып кеткен).*
129	*ENDIF directive missing (ENDIF дерективасы қалып кеткен).*
130	*Error in initial conditional defines (Шартты анықтамаларда қате бар).*
131	*Header does not match previous definition (Тақырып алдыңғы анықтамаға сәйкес келмейді).*
132	*Critical disk error (Дисктің сынағыштық қатесі).*
133	*Cannot evaluate this expression (Берілген өрнекті есептеуге болмайды).*
134	*Expression incorrectly germinated (Өрнек дұрыс аяқталмаған).*
135	*Invalid format specifier (Формат спецификаторы дұрыс емес).*
136	*Invalid indirect reference (Ретсіз жанама сілтеме).*
137	*Structured variable are not allowed here (Бұл жерде құрылымдық типтегі айнымалыны пайдалануға болмайды).*
138	*Cannot evaluate without System unit (Өрнекті SYSTEM модулінсіз есептеуге болмайды).*
139	*Cannot access this symbol (Берілген символға қол жеткізу мүмкін емес).*
140	*Invalid floating-point operation (Жылжымалы-үтірмен орындалатын операция ретсіз).*
141	*Cannot compile overlay to memory (Оверлейлік модуль компиляциясын жадыда орындау мүмкін емес).*
142	*Procedure or function variable expected (Процедуралық типтегі айнымалы қолданылу керек).*
143	*Invalid procedure or function reference (Процедураға немесе функцияға ретсіз сілтеме жасалған) .*
144	*Cannot overlay this unit (Бұл модульді оверлейлік ретінде қолдануға болмайды).*
145	*Too many nested scopes (Кіріктірмелер саны көп).*
146	*File access denied (Файлға қол жеткізу мүмкін емес).*
147	*Object type expected (Бұл жерде OBJECT типі болу керек)*
148	*object types are not allowed (Жергілікті объектілерді хабарлауға болмайды).*
149	*VIRTUAL expected (VIRTUAL сөзі қалып кеткен).*
150	*Method identifier expected (Инкапсуляциялық ереже идентификаторы қалып кеткен).*

151	*Virtual constructor are not allowed (Шебер (конструктор) виртуалды бола алмайды).*
152	*N/A*
153	*Destructor identifier expected (Құрылым бұзушы (деструктор) идентификаторы қалып кеткен).*
154	*Fail only allowed within constructor (Стандартты FAIL процедурасын тек шеберде шақыруға болады).*
155	*Invalid combination of opcode and operands (Командалар және операндтар кодының комбинациясы дұрыс емес).*
156	*Memory reference expected (Адрес қалып кеткен).*
157	*Cannot add or subtract relocatable symbols (Жылжымалы символдарды қосуға немесе азайтуға болмайды).*
158	*Invalid register combination (Регистрлер комбинациясы ретсіз).*
159	*286/287 instructions are not enabled (286/287 микропроцессорлар командасына қол жеткізу мүмкін емес).*
160	*Invalid symbol reference (Символға ретсіз сілтеме жасалған).*
161	*Code generation error (Кодтар генерациясында қате бар).*
162	*ASM expected (ASM резервтегі сөзі қалып кеткен).*

2. Программа орындалу кезінде шығатын қателер

Программа орындалу кезінде анықталатын кейбір қателер экранда Runtime error nnn at xxxx:yyyy (xxxx:yyyy адресі бойынша nnn кезеңін орындаудағы қате) мәлімдемесінің шығуына әкеледі, мұнда nnn — қате нөмірі; xxxx:yyyy — адрес (сегмент немесе жылжу). Бұл мәлімдемеден кейін программа өз жұмысын тоқтатады. Программа орындалу кезінде шығатын қателер төртке бөлінеді: СОЖ (сұхбатты операциялық жады) анықтайтын қателер (1-ден 99-ға дейінгі қателер), енгізу-шығару қателері (100-ден 149-ға дейінгі қателер), дағдарысты қателер (критические ошибки) (150-ден 199-ға дейін) және фаталды қателер (200-ден 255-ке дейінгі қателер).

3. Операциялық жады анықтайтын қателер

Қате №	***Аты***
1	*Invalid function number (Функция нөмірі дұрыс жазылмаған).*
2	*File not found (Файл табылмады).*

3	*Path not found (Жол табылмады).*
4	*Too many open files (Ашылған файлдар саны көп).*
5	*File access defined (Файлға қол жеткізуге рұқсат жоқ)*
6	*Invalid file handle (Ретсіз файлдық канал).*
12	*Invalid file access code (Файлға қол жеткізу коды нақты емес).*
15	*Invalid drive number (Дискенгізгіш нөмірі ретсіз).*
16	*Cannot remove current directory (Ағымдағы каталогты өшіруге болмайды).*
17	*Cannot rename across drives (Атты ауыстырғанда әр түрлі дискенгізгіш аттарын көрсетуге болмайды).*

4. Енгізу-шығару қателері

Егер операторлардың біреуі {$1+} директивасымен компиляциядан өтсе, онда енгізу-шығару қатесі программаның орындалуын тоқтатады. {$1—} қалпында программа орындалуын жалғастырып, қате IORESULT функциясымен қайтарылады.

Қате №	***Аты***
100	*Disk read error (Дисктен оқу кезінде қате кетті).*
101	*Disk write error (Дискке жазу кезінде қате кетті).*
102	*File not assigned (Файлға ат берілмеген).*
103	*File not open (Файл ашылмаған).*
104	*File not open for input (Файл мәндер енгізу үшін ашылмаған).*
105	*File not open for output (Файл мәндер шығару үшін ашылмаған).*
106	*Invalid numeric format (Сандық формат дұрыс емес).*

5. Дағдарысты қателер

Қате №	***Аты***
150	*Disk is write protected (Диск жазудан сақталған).*
151	*Unknown unit (Белгісіз модуль).*
152	*Drive not ready (Дискенгізгіш «дайын емес» қалыпкүйінде).*
153	*Unknown command (Бейтаныс команда).*
154	*CRC error in data (Бастапқы мәндерде қате бар).*
155	*Bad drive request structure length (Дисктен мәлімет аларда құрылым ұзындығы дұрыс көрсетілмеген).*

156	*Disk seek error (Дисктен оқитын құрылғы бастарын дискке орнату операциясы кезінде қате кетті).*
157	*Unknown media type (Тасмалдауыш типі белгісіз).*
158	*Sector not found (Сектор табылмады).*
159	*Printer out of paper (принтерде қағаз біттi).*
160	*Device write fault (Құрылғыға жазу кезінде қате кетті).*
161	*Device read fault (Құрылғыдан оқу кезінде қате кетті).*
162	*Hardware failure (Аппарат жұмыс істемей тұр).*

6. Фаталды қателер

Бұл қателер программа жұмысының лезде тоқтауына әкеліп соғады.

Қате №	***Аты***
200	*Division by zero (Нөлге бөлу).*
201	*Range check error (Шекараларды тексеру кезінде қате табылды).*
202	*Stack overflow error (Стек толып кетті).*
203	*Heap overflow error (Мәлімет толып кетті).*
204	*Invalid pointer operation (Көрсеткішпен орындалатын белгісіз операция).*
205	*Floating point overflow (Жылжымалы нүктелі сандармен жұмыс жасау кезінде мәндер берілген аймақтан шығып кетті).*
206	*Floating point underflow (Жылжымалы үтірмен жұмыс жасау кезінде реті жоғалып кетті).*
207	*Invalid floating point operation (Жылжымалы нүтірмен жұмыс жасауға болмайтын операция) .*
208	*Overlay manager not installed (Оверлейді басқарудың ішкі жүйесі орнатылмаған).*
209	*Overlay file read error (Оверлейлік файлды оқу кезінде қате жіберілген).*
210	*Object not initialized (Объект инициалданбаған).*
211	*Call to abstract method (Абстракты ережені шақыру).*
212	*Stream registration error (Тіркеуге алу ағынында қате бар).*
213	*Collection index out of range (Терілген индекс диапазон шекарасынан тыс жатыр).*
214	*Collection overflow error (Коллекция толып кеткен).*

Паскаль тіліндегі программалар мысалы

1 тапсырма: қосу, азайту, көбейту және бөлу арифметикалық амалдарын орындайтын “КАЛЬКУЛЯТОР” программасын жазыңыз.

Енгізілетін мәндер форматы: 1-ші сан, орындалатын амал символы (+,-,*,/), 2-ші сан.

Бөлу амалын орындау кезінде 2-ші санның нөлге тең емес екендігін тексеру керек.

Программа мәтіні

```
Program calc;
  uses crt;
  var x,y,z:real; op,ch:char;
begin
 repeat
  clrscr;
   writeln('КАЛЬКУЛЯТОР программасы');
   writeln(' + – * / амалдарын орындайды');
   write(' 1-ші санды енгізіңіз:');
   readln(x);
   repeat
write('Орындалатын амал символын енгізіңіз:');
readln(op);
   until (op='+') or (op='-') or (op='*') or (op='/');
   write(' 2-ші санды енгізіңіз:');
   readln(y);
   case op of
'+':z:=x+y;
'-':z:=x-y;
'*':z:=x*y;
'/':if y<>0
       then z:=x/y
       else
        begin
          writeln(' 0-ге бөлуге болмайды!');
          z:=0;
        end;
   end;
```

```
    writeln(x,op,y,'=',z:10:3);
        write(' КАЛЬКУЛЯТОР программасымен жұмысты аяқтайсыз ба? (Yes/No)?');
    readln(ch);
    until upcase(ch)='y';
end.
```

2 тапсырма: $ax^2 + bx + c = 0$ квадрат теңдеуді шешуге арналған программа жазыңыз.

Программа мәтіні

```
Program kvur;
  var a,b,c,d,x1,x2:real;
begin
writeln(' Квадрат теңдеуді шешу' );
write(' a,b,c коэффициенттерін енгізіңіз:');
readln(a,b,c);
d:=b*b-4*a*c;
if d<0 then
     writeln(' Шешім жоқ ')
   else
     begin
       writeln(' Шешім бар ');
       x1:=(-b+sqrt(d))/(2*a);
       x2:=(-b-sqrt(d))/(2*a);
       writeln('x1=',x1:5:5,' x2=',x2:5:5);
     end;
 writeln(' ENTER батырмасын басыңыз ');
 readln;
 end.
```

3 тапсырма: Берілген ақшаның құндылығы 10 000, 5000, 2000, 1000, 500, 200, 100, 50, 20, 10 теңге банкноттарының ең аз санымен өрнектеу керек.

Программа мәтіні

```
Program dengi (input, output);
  var s, (*summa deneg*)
  sk (*obchee kol-vo banknot *) : longint;
  b (*kol-vo banknot odnogo vida*) : integer;
begin
```

```
writeln ( ‘vvedite summu deneg v banknotax: ‘); read (s);
sk:= 0; b:= s div 10000;
if b > 0 then
begin
sk:= sk + b; writeln (10000, ‘: ‘, b); s:= s mod 10000
end;
b := s div 5000;
if b> 0 then
begin
sk := sk + b; writeln (5000, ‘ :’, b); s := s mod 5000
end;
b := s div 2000;
if b> 0 then
begin
sk:= sk + b; writeln (2000, ‘ : ‘, b); s := s mod 2000
end;
b:= s div 1000;
if b> 0 then
begin
sk := sk + b; writeln (1000, ‘ : ‘, b); s := s mod 1000
end;
b := s div 500;
if b> 0 then
begin
sk := sk + b; writeln (500, ‘ : ‘, b); s := s mod 500
end;
b := s div 200;
if b> 0 then
begin
sk:= sk + b; writeln (200, ‘ : ‘, b); s := s mod 200
end;
b := s div 100;
if b> 0 then
begin
sk:= sk + b; writeln (100, ‘:’, b); s:= s mod 100
end;
b := s div 50;
if b> 0 then
begin
```

```
 sk:= sk + b; writeln (50, ':', b); s:= s mod 50
 end;
 b := s div 20;
 if b> 0 then
 begin
 sk:= sk + b; writeln (20, ':', b); s:= s mod 20
 end;
 b := s div 10;
 if b> 0 then
 begin
 sk:= sk + b; writeln (10, ':', b); s:= s mod 10
 end;
 b := s div 5;
 if b> 0 then
 begin
 sk:= sk + b; writeln (5, ':', b); s:= s mod 5
 end;
 b := s div 1;
 if b> 0 then
 begin
 sk:= sk + b; writeln (1, ':', b); s:= s mod 1
 end;
 writeln ('vsego ', sk, ' banknot ')
end.
```

Программа көлемін кішірейту үшін есепті процедураны қолданып шығарамыз.

Программа мәтіні

```
Program banknot (input, output);
  var s, sk : longint;
     bk : integer;
 procedure bb (v : integer; var ss,ssk: longint);
  (* v – cennoct banknot *)
  (*ss – summa deneg *)
  (*ssk – kol-vo banknot *)
  var b : integer;
begin
  b := ss div v;
  if b> 0 then
```

```
      begin
       ssk := ssk + b; writeln (v, ' tengelik : ', b, ':2); ss := ss mod v;
      end
  end;
  begin (*nachalo programmy*)
   writeln ('summa deneg: ');
   read (s);
   sk := 0; bk:=0;
   (* obrachenia k proc bb*)
   bb (10000, s, sk);
   bb (5000, s, sk);
   bb (2000, s, sk);
   bb (1000, s, sk);
   bb (500, s, sk);
   bb (200, s, sk);
   bb (100, s, sk);
   bb (50, s, sk);
   bb (20, s, sk);
   bb (10, s, sk);
   bb (5, s, sk);
   bb (1, s, sk);
   writeln ('vsego ', sk, ' banknot.')
  end.
```

4 тапсырма: $y = 0.5x^2 + 4x - 3$ функциясының графигін $x \in [-1.5;5]$ аралығында $h = 0.1$ қадамымен тұрғызатын программа жазыңыз.

Программа мәтіні

```
Uses Graph;
var
  x, dx: real;     x1,x2: real;
  y: real;
  mx,my: integer;
  x0,y0: integer;
  px,py: integer;
  grDriver: integer;
  grMode: integer;
  ErrCode: integer;
  i: integer;
```

```
Begin
  grDriver:=VGA;
  grMode:=VGAHi;
InitGraph (grDriver, grMode, 'C:\BP\BGI');
    ErrCode:=GraphResult;
    if ErrCode <> grOK then begin
     writelen ('Графикалық режим қосылмады');
     writelen ('Жұмысты аяқтау үшін <ENTER>
пернесіне басыңыз');
        readlen;
        Halt (1);
    end;
   x0:=640; y0:=480;
   mx:=20;  my:=20;
   line(10,y0,630,y0);
   line(x0,10,x0,470);
x1:=-15;
     x2:=5;
    dx:=0.1;      x:=x1;
     while (x<x2) do begin
     y:=0.5*x*x+x*4-3;
     px:=x0+round(x*mx);
     py:=y0-round(y*my);
     PutPixel (px,py,White);
     x:=x+dx;
   end;
  readln;
End.
```

Паскаль тілінің дербес компьютерге арналған нұсқасының қордағы сөздері

Ағылшынша	*Қазақша*	*Ағылшынша*	*Қазақша*
absolute	абсолюттік	label	белгі (тамға)
and	логикалық ЖӘНЕ	library	кітапхана
array	жиым (массив)	mod	бөліндінің қалдығы
asm	ассемблер	nil	болмау (бос болу)
begin	блок басы	not	логикалық ЕМЕС
case	вариант	or	логикалық НЕМЕСЕ
const	тұрақты (константа)	of	одан (-дан, -ден, -тан, -тен)
constructor	конструктор	object	объект
div	бүтін бөлу	packed	тығыздалған
go to	ауысу (көшу)	procedure	процедура
do	орындау, атқару	program	программа
downto	кеміту	record	жазба
destructor	деструктор (бұзушы)	repeat	қайталау
else	әйтпесе	set	жиын
end	блок соңы	shl	биттердісолға ығыстыру
exports	экспорт (жіберу)	shr	биттерді оңға ығыстыру
external	сыртқы	string	қатар (жол)
file	файл	then	онда
for	үшін	to	үлкейту
forward	алдындағы	type	тип (түр)
function	функция	unit	модуль
if	егер	until	дейін (шейін)
implementation	жүзеге (іске) асыру	uses	пайдалану
in	ішіндегі (ішіне ену)	var	айнымалы
inline	Негізгі	while	әзірше, болмайынша
interrupt	кідірту (үзу)	with	-дан, -ден, -тан, -тен
interface	интерфейс	xor	алып тастау НЕМЕСЕ
inherited	мұралау (қаблдау)		

ASCII-КОДТАР КЕСТЕСІ

0	32	64 - @	96 - `	128 - А	160 - а	192 - └	224 -р
1 - ☺	33 - !	65 – A	97 - a	129 - Б	161 - б	193 - ┴	225 -с
2 - ☻	34 - «	66 – B	98 - Ь	130 - В	162 - в	194 - ┬	226 -т
3 - ♥	35 - #	67 – C	99 - c	131 - Г	163 - г	195 - ├	227 - у
4 - ♦	36 - $	68 – D	100 - d	132 - Д	164 - д	196 - ─	228 - ф
5 - ♣	37 - %	69 – E	101 - e	133 - Е	165 - е	197 - ┼	229 - х
6 - ♠	38 - &	70 – F	102 - f	134 - Ж	166 - ж	198 - ╞	230 - ц
7 - •	39 - '	71 - G	103 - g	135 - З	167 - з	199 - ╟	231 - ч
8 - ◘	40 - (	72 - H	104 - h	136 - И	168 - и	200 - ╚	232 - ш
9 - o	41 -)	73 - I	105 - i	137 - Й	169 - й	201- ╔	233 - щ
10 - ◙	42 -*	74 - J	106 - j	138 - К	170 - к	202 - ╩	234 - ъ
11 -♂	43 - +	75 - K	107 - k	139 - Л	171 - л	203- ╦	235 - ы
12 - ♀	44 - ,	76 - L	108 - l	140 - М	172 - м	204 - ╠	236 - ь
13 - ♪	45 - -	77 - M	109 - m	141 - Н	173 - н	205 - ═	237 - э
14 - ♫	46 - .	78 - N	110 - n	142 - О	174 - о	206 - ╬	238 - ю
15 - ☼	47 - /	79 - 0	111 - o	143 - П	175 - n	207 - ╧	239 - я
16 - ►	48 - 0	80 - P	112 - p	144 - Р	176 - ░	208 - ╨	240 - Ё
17 - ◄	49 - 1	81 - Q	113 - q	145 - С	177 - ▒	209 - ╤	241 - ё
18 - ↕	50 - 2	82 - R	114 - r	146 - Т	178 - ▓	210- ╥	242 - Є
19 - !!	51 - 3	83 - S	115 - s	147 - У	179 - ≥	211 - ╙	243 - є
20 - ¶	52 - 4	84 - T	116 - t	148 - Ф	180 - ┤	212 - ╘	244 - Ї
21 - §	53 - 5	85 - U	117 - u	149 - Х	181 - ╡	213 - ╒	245 - ї
22 - ■	54 - 6	86 - V	118 - v	150 - Ц	182 - ╢	214- ╓	246 - ÷
23 - ↨	55 - 7	87 - W	119 - w	151 - Ч	183- ╖	215 - ╫	247 - ≈
24 - ↑	56 - 8	88 - X	120 - x	152 - Ш	184 - ╕	216 - ╪	248 - °
25 - ↓	57 - 9	89 - Y	121 - y	153 - Щ	185 - ╣	217 - ┘	249 - •
26 - →	58 - :	90 - Z	122 - z	154 - Ъ	186 - ║	218 - ┌	250 - ·
27 - ←	59 - ;	91 - [	123 - {	155 - Ы	187 - ╗	219 - █	251 - √
28 - └	60 - <	92 - \	124 - |	156 - Ь	188 - ╝	220 - ▄	252 - n
29 - ↔	61 - =	93 -]	125 - }	157 - Э	189 - ╜	221 - ▌	253 - ²
30 - ▲	62 - >	94 - ^	126 - ~	158 - Ю	190 - ╛	222 - ▐	254 - ■
31 - ▼	63 - ?	95 - _	127 - □	159 - Я	191 - ┐	223 - ▀	255 - □

ҚОЛДАНЫЛҒАН ӘДЕБИЕТТЕР ТІЗІМІ

1. *Абрамов С.А.* и др. Задачи по программированию. М., Наука, ГРФМЛ, 1988.
2. *Вирт Н.* Алгоритмы и структуры данных. М., Мир, 1989.
3. *Грибанов В.П., Калмыкова О.В., Сорока Р.И.* Основы алгоритмизации и программирование. Уч.пос., М., МЭСИ,2001.
4. *Дайитбегов Д.М., Черноусов Е.А.* Основы алгоритмизации и алгоритмические языки. Учебник. М., Финансы и статистика, 1992.
5. *Йенсен К., Вирт Н. Паскаль.* Руководство для пользователя, М., Финансы и статистика, 1989.
6. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.. МЦНМО, 1999.
7. *Фаронов В.В.* Turbo Pascal 7.0. Начальный курс. Учебное пособие. М., Нолидж,1998.
8. *Шень А.* Программирование: теоремы и задачи. М., МЦНМО, 1995.
9. *Ван Тассел Д.* Стиль, разработка, эффективность, отладка и испытание программы. М., Мир,1981.
10. *Вирт Н.* Алгоритмы + структуры данных = программы. М., Мир,1985.
11. *Джонс Ж.,Харроу К.* Решение задач в системе Турбо Паскаль. М., Финансы и статистика, 1989.
12. *Епанешников А.М., Епанешников В.А.* Программирование в среде Turbo Pascal 7.0. М., Диалог-МИФИ, 1995.
13. *Лэнгсам Й., Огенстайн М., Таненбаум А.* Структуры данных для персональных ЭВМ, М., Мир, 1989.
14. *Майерс Г.* Искусство тестирования программ. М., Финансы и статистика, 1982
15. *Поляков Д.Б., Круглов И.Ю.* Программирование в среде Турбо-Паскаль (версия 5.5), М., МАИ, 1992
16. *Рубенкинг Н. Турбо* Паскаль для Windows. В 2 т. М.,СК Ферлаг Интернешнл,1994
17. *Семашко Г.Л., Салтыков А.И.* Программирование на языке Паскаль. М., Наука, 1988
18. Турбо Паскаль 7.0. К., Торгово-издательское бюро BHV, 1996
19. *Фаронов В.В.* Turbo Pascal 7.0. Практика программирования. Учебное пособие. М., Нолидж,1998

МАЗМҰНЫ

Б.Б. Бөрібаев, А.М. Махметова

АЛГОРИТМДЕУ ЖӘНЕ
ПРОГРАММАЛАУ ТІЛДЕРІ

Оқулық

Басуға 21.12.11. қол қойылды. Қағазы офсеттік.
Қаріп түрі “Таймс” Пішімі 60х90/16. Баспа табағы 20.5.
Таралымы 1900 дана. Тапсырыс 1618.

Тапсырыс берушінің дайын файлдарынан басылып шықты.

ДӘУІР
ЖШС РПБК «Дәуір», 050009,
Алматы қаласы, Гагарин д-лы, 93а.
E-mail: rpik-dauir81@mail.ru