1. Есепті компьютерге даярлау және шешу кезеңдері.

Ақпарат ғасыры алғашқы компьютерлердің пайда болуымен басталған жоқ., Ол көлемді, күшті ақпараттық магистралы бар, телекоммуникация, теледидар және компьютерлік технологияның бірлесуімен, мәліметтерді сандық өңдеу әдісінің кеңінен қолданылуының нәтижесінде, мультимедиа мен телекомуникацияның пайда болуынан басталады.

Соңғы жылдары ақпараттық және коммуникациялық технологиялар саласында ауқымды ілгерілеу байқалып отыр, оның нәтижесінде бір кезде шыққан компьютерлік жабдықтар, ондан да дамытылған компьютерлермен алмасып және олар басқа да технологияларға қосылып жатады.

Компьютер дайын бағдарламаны орындауға арналған құрал. Ол үшін компьютерге бағдарлама ендіру керек. Бағдарламалау технологиясы бағдарлама құрудың әдістерін үйретеді және бағдарлама мен қамтамасыз етуді жолдарын көрсетіп кетеді.

Бағдарламаны жазу үшін алдын ала оның мақсаты алдағы шығарылатын есептердің түрлері шығару әдістері анық болу керек. Бағдарламаны жасау алдымен алгоритм құрудан басталады. Ол текстi және блок схемалар түрінде құрылады. Құрылған алгоритм белгілі бір бағдарламалау тілінде жазылып компьютерге ендіріледі. Бағдарламалау тілінде алгоритм командалар және операторлар тізбегі ретінде көрсетіледі. Ол тілді компьютер түсініп машиналық кодқа айналдырады. Бағдарлама дегеніміз компьютерге арналған командалардың тізбегі оның көмегімен кез-келген күрделі функцияларды бағдарламалауға болады.

Бағдарлама ұзақ уақыт сақталады қажетті уақытта оны пайдалануға болады және оны жеке сақтап басқа бағдарлама құрамында пайдалануға болады. ЭВМ-де есеп шығару күрделі процес ол үшін ғылымның әр түрлі салаларынан: математикадан, т.б. ғылым салаларынан және бағдарламалаудан, есептеуіш техника негіздерінен білімдеріміз болуы қажет. Кез – келген есепті шығаруда бастапқы мәндерді өңдеу процесін іске асыру

қажет. Бастапқы мәндер өңдеу барысында төмендегідей іс әрекеттерді орындау керек.

1.2 ЕСЕПТІ ЭЕМ – ДЕ ШЫҒАРУДЫҢ НЕГІЗГІ КЕЗЕҢДЕРІ

Есепті ЭЕМ – ді пайдаланып шығару алты кезеңнен тұрады:

- Есептің математикалық жобасын белгілеу;
- Есептің шешу әдісін таңдап алу;
- ЭЕМ – нің ерекшелігін ескеріп, есепті шешу үшін алгоритм таңдау, құрастыру;
- Бағдарламалау;
- Бағдарлама жұмысын ЭЕМ – де тексеру, қалыптастыру;
- Есепті ЭЕМ – де автоматты түрде орындау.

Енді әр кезеңнің орындалуына қысқаша тоқталып өтелік.

1. Есепті математикалық тұрғыдан дұрыс қою сатысына оның мазмұнын анықтайтын барлық айнымалыларға сәйкес математикалық белгілеулер енгізіледі. Сондай – ақ олардың арасындағы байланыстар формальді түрде жазылады. Қолданылатын математикалық аппарат есептің шартынан анықталады.

2. Есепті шешу әдісін таңдап алу сатысында шығатын нәтижемен берілген деректердің арасындағы тәуелділік және таңдап алған әдістің қолданылу мүмкіндігі зерттеледі.

3. Үшінші кезеңде – есепті шешуге арналған әдістің ЭЕМ – ның нақты түріне сәйкес құрылған алгоритімі жасалады.

4. Алгоритмді ЭЕМ – ге түсінікті формальді тілде жазу сатысын бағдарламалау деп атайды. Бағдаралама деп алгоритмнің ЭЕМ – ге түсінікті таңбада жазылуын айтамыз. Бағдарламаны ЭЕМ – де қолданылатын алгоритмдік тілде жазылған осы алгоритмді жүзеге асыратын жеке сөйлемдер тізбегі деп қараумызға болады.Бағдарламаның сәтті құрылып шығуы алгоритмнің жасалу сапасына да тікелей байланысты. Ойластырып жасалған алгоритмге сәйкес құрылған бағдарлама тиімді сәтті шығуы тиіс.

5. Жасалған бағдарламаның қызметін бағдарламалаушы ЭЕМ – да есептеулер жүргізу арқылы тексереді және керек жағдайда

бағдарлама жұмысын бейімдейді (қалыптастырады) немесе өзгертеді.

6. Осылайша тексерілген бағдарламаны нақты есептерді ЭЕМ – да автоматты түрде шешуге пайдалануға болады.

1.3 АЛГОРИТМ ЖӘНЕ БЛОК – СХЕМА

Алгоритм деп – белгілі бір мақсатқа жету жолында рет –ретімен рындалатынып, іс әрекеттер тізбегін айтады. Ол сөз тілінде немесе блок- схемалар түрінде бейнеленіп жазылады

Оған қойылатын талаптар:

1. Дискреттілік жеке-жеке бөліктерге бөліну қасиеттері.
2. Алгоритм нің түсініктілігі
3. Алгоритмнің жалпылығы
4. Алгоритмнің нәтижелілігі

Алгоритімнің негізгі үш түрі бар.

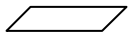
1. Сызықтық алгоритм деп – іс әрекеттің бірінен соң бірі рет-ретімен орындалуын айтады.
2. Тармақталған алгоритм деп – белгілі бір шартқа байланысты орындалатын іс әрекеттер тізбегін айтады.
3. Цикілдік алгоритм деп – белгілі бір процестің қайталанып орындалуын айтады.

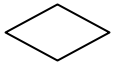
Блок – схема – алгоритмнің орындалуын ұйымдастыру үшін қолданылатын амалдар тізбегінің графиктік кескіні.

Блок – схема келісілген геометриялық фигуралардың көмегімен құрылады және бұл фигураларға келісімге байланысты өзіндік мағыналар беріледі.

Алгоритімнің блок схема түрінде бейнеленуі. Алгоритімнің басталуы элипс 

фигура сы көмегімен көрсетіледі.

 - паралелограмм. Ендіру, шығару деп аталып,

 деректерді жазу үшін қолданылады.

ромбы шартты тексеру.

 - есептеу процесін білдіреді.

 Блок схемадағы байланысты көрсету

қосалқы бағдарламаны шақыру.

Тік төртбұрыш – арифметикалық амалдарды орындаушы блок, ромб – кез келген шартты тексеруші немесе салыстыру процесін орындаушы блок, ал эллипс және параллелограм тәрізді фигураны енгізу және қорытындылау процестерін жүргізуші блоктар ретінде пайдалануға болады. Бұл фигуралар алгоритмнің мазмұнына сәйкес өзара сызықтар арқылы жалғасады. Әрбір фигура ішінде орындалатын амалдар көрсетіледі де, олар амалдар блогы деп аталады. Блок – схеманы жасағанда басқа да геометриялық фигураларды пайдаланылады.

АЛГОРИТМДЕУ-дің қасиеттеріне толық түсінік берейік:

Жалпылық – бұл бір алгоритмді қолданып көптеген есептерді шешу мүмкіндігін сипаттайды. Мысалы : $AX^2 +BX+C=O$ теңдеуін шешу алгоритмін, сол типтес көптеген есептерді коэффициенттердің мәнін өзгерте отырып, шығара аламыз.

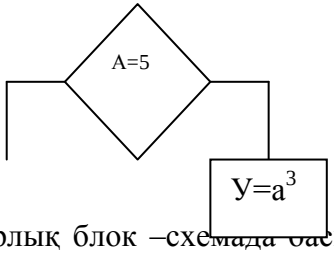
Түсініктілік, берілген есептерді шығара отырып, есептеуші келесі есепте не істеу керек екендігін білу керек.

Нәтижелік, яғни алгоритмді пайдалану арқылы есептердің жауаптарын аламыз.

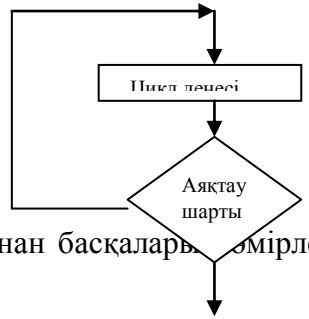
Алгоритмдеу есептері текстік(мәтін) және схемалық түрде болуы мүмкін. Схемалық түрі ең көрнекті болып табылады. Схемалық түрде жазу үшін халықаралық символдар әртүрлі геометриялық фигуралар қолданылады. Ол фигуралар туралы жоғарыда түсінік берілді.

2) Сызықтық, тармақталған, цикілдік алгоритімнің блок схемада бейнеленуі.

а) Тармақталған алгоритм блок схемасы

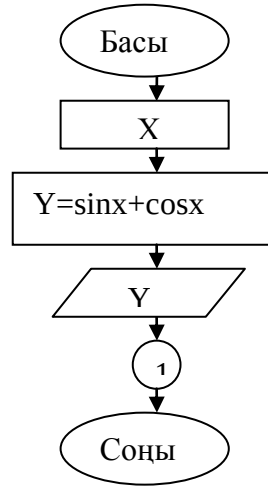


б) Цикілдік алгоритімнің блок схемасы



Барлық блок –схемада бағымен соңынан басқалары өмірленеді, әрі 1 кіріс пен шығыс болады.

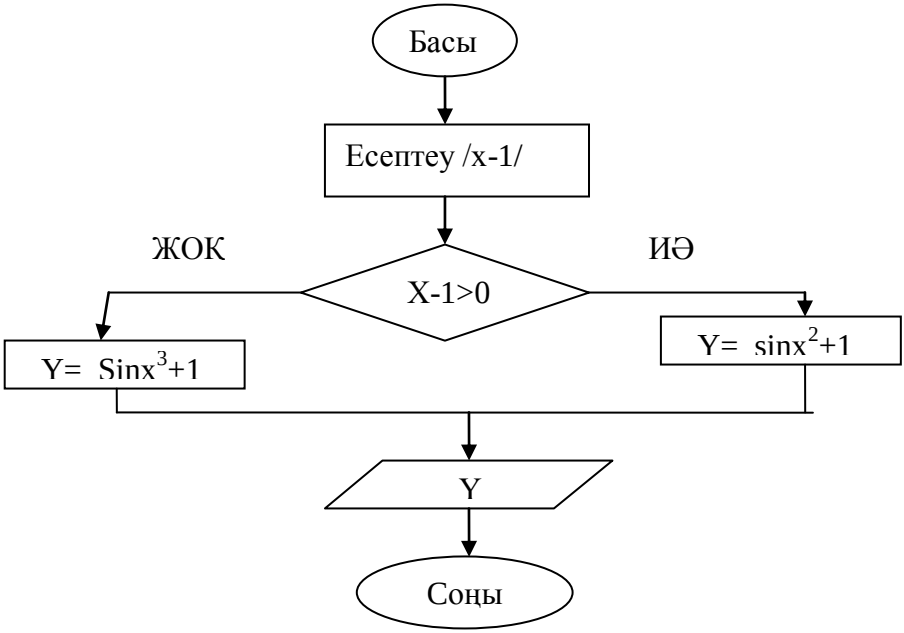
Блоктың “басы” тек кірісті, ол блоктың “соңы” тек шығысты қамтамасыз етеді. Ойлау блогында 1 кіріс, 2 шығыс болады.
Мысалы: Y=Sinx+Cosx функциясын есептеудің сызықтық алгоритмнің блок – схемасын құру.



Алгоритм есептеу процесін жүзеге асыру кезінде бір жолдардан шартқа байланысты басқа жолға өтуін тармақталған алгоритм деп атайды.

1 Мысал: Берілген теңдеулер жүйесін шешудің блок - схемасы:

$$Y=\begin{cases} SinX^3+1 & \text{егер } X-1>0, \\ SinX^2+1 & \text{егер } X-1<0 \end{cases}$$



X-1>0 шарты орындалса, $Sinx^3+1$, ал орындалмаса $Sinx^2+1$-ді есептеулініп нәтижесі шығады.

2. Мысал: Бір өлшемді A(20) массивтің оң элементтерінің қосындысын табу керек.

1-ші блокта 20 элементтен тұратын A массиві енгізіледі, ***2-ші блокта*** оң сандардың қосындысы сақталатын S – айнымалысын нольге теңестіріп аламыз.

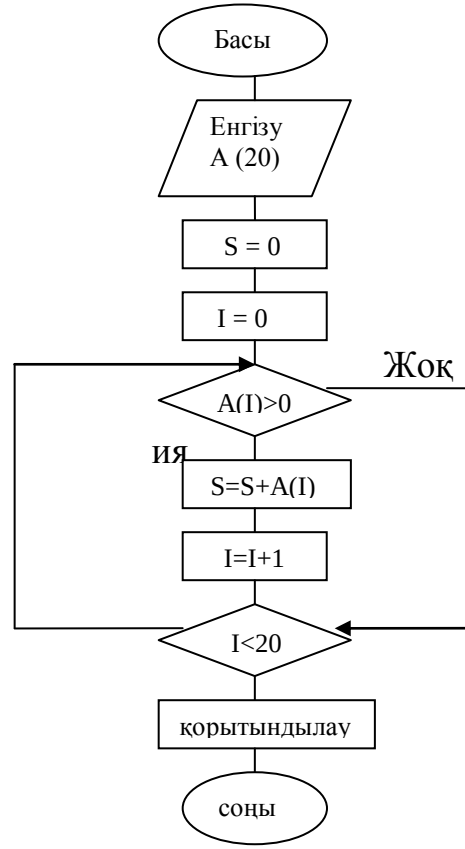
3-ші блокта Цикл құрастыруға қажет айнымалыны I – деп белгілеп, алғашқы мәнін бір деп аламыз.

4-ші блокта A массивінің I – ші элементі оң сан екендігін тексереміз. Егер ол оң сан болмаса, тікелей 6-шы блокқа ауысамыз. Егер, оң сан болса, онда осыған шейінгі оң сандар қосындысына A (I) – элементінің мәні қосылады (5-ші блок).

Келесі 6-шы блокта айнымалы I-дің мәні бірге арттырылады. Ол цикл құрастыруға қажет.

7-ші блокта I – дің массив өлшемінен аспауын тексереміз. Егер I > 20 , болса 8-ші блокқа ауысамыз. Егерде I –дің мәні 20-дан кем болса 7-ші блокқа ораламыз (бұл цикл құрылымы).

8-ші блокта S айнымалысында оң сандардың қосындысы сақталады.



3. Деректердің типтері және оның Паскаль тілінде жазылуы

ПАСКАЛЬ ТІЛІНІҢ НЕГІЗГІ ЭЛЕМЕНТТЕРІ.

60-70-ші жылдарда Н.Вирт ойнап шығарған Паскаль тілі қазіргі кезде дүние жүзінде кең тараған бағдарламалау тіліне айналды. Қазіргі кезде Паскаль тілі қолданбалы бағдарламалар жазу үшін және жүйелі бағдарламалау тілі ретінде де кеңінен қолданылады. Сонымен бірге көптеген мини және макро компьютерлерді қамтамасыз ету осы тілде жазылған. Паскаль тілі бұрын құрылған (Фортран, Алгол және т.б.) бағдарламалау тілдерінен маңызды ерекшелігі ол құрылымдық бағдарламалау идеясын өмірге біртіндеп енгізу. Паскаль тілінің тағы бір ерекшелігі ол деректер құрлымы концепцясының алгоритм түсінігімен қатар бағдарламалаудың негізінде жатқан фундаменталды түсініктер қатарына кіруі.

Бағдарламаның көпшілігі белгілі бір есепті шешу үшін жазылады. Ал есептің шешіміне ақпараттар мен деректерді өңдеу арқылы жетуге болады. Сондықтан бағдарлама құруда төмендегі әрекеттерді орындау қажет :

. ақпараттарды бағдарламаға енгізу;

. ақпараттарды сақтау;

. деректерді өңдеу нұсқауларын беру;

. нәтижелерді шығару.

Бағдарламалаудың негізгі жеті түсінігін анықтауға болады. Олар: енгізу, деректер, операциялар, шығару, шартты орындау, циклдер және қосалқы бағдарламалар.

ЕНГІЗУ. Бұл пернетақтадан келіп түсетін немесе сыртқы есте сақтау құрылғысындағы деректер жиынынан оқылатын ақпарат;

ДЕРЕКТЕР. Бұл тұрақтылар, айнымалылар, массивтер, жазбалар, жиындар, мәтіндер (символдар және жолдар), файлдар;

ОПЕРАЦИЯЛАР. Мәндерді меншіктеуді, өрнектерді есептеуді (қосу, бөлу және т.с.с), мәндерді салыстыруды (теңдік, үлкен, кіші) қамтамасыз етеді.

ШЫҒАРУ. Бұл ақпараттарды дербес компьютердің мониторындағы экранға шғару немесе сыртқы есте сақтау құрылғысындағы деректер жиынына жазуды білдіреді;

ШАРТТЫ ОРЫНДАУ. Бұл бір немесе бірнешенұсқаулардың белгілі бір шарт орындалғанда (ақиқат) орындалуы. Егер шарт орындалмаса, онда бұл нұсқаулар өткізіліп жіберіледі немесе нұсқаулардың басқа жиыны орындалады;

ЦИКЛДЕР. Мұнда нұсқаулардың жиыны қандай да бір шарт ақиқат болған кезде немесе шарт ақиқат болмайынша қайталанып орындалады;

ҚОСАЛҚЫ БАҒДАРЛАМАЛАР. Бағдарламаның кез - келген жерінде аты арқылы шақырып орындауға болатын нұсқаулардың белгілі атпен біріктірілген жиыны.

Паскаль бағдарламалау тілінің құрылымы

Паскаль тілінің алфавиті төмендегідей символдардан тұрады:

1) 26 латын алфавитінің (үлкен/кіші) әріптері;
2) 0-ден 9 –ға дейінгі араб цифрлары;
3) Орыс/қазақ алфавитінің әріптері;
4) Арнайы символдар: + - қосу, - азайту, * - көбейту, /- бөлу, : - қос нүкте, ‘- апостроф, < - кіші, >- үлкен, = - тең, ; - нүктелі үтір, () – жақша, []- тік жақша тағыда басқа символдар;

Арнайы символдар және олардың комбинациялары бағдарламада ерекше роль атқарады.

Паскаль тіліндегі бағдарлама екі бөлімнен тұрады:

- ***қолданылатын деректердің элементтерінің түрлерін сипаттау***;
- ***деректердің элементтерін*** **нұсқауларға** ***байланысты өңдейтін операторлар***.

Қолданылатын деректердің элементтерінің түрлерін сипаттау бөлімі бағдарламада кездесетін барлық деректерді және олардың сипаттамаларын хабарлау үшін қолданылады. Оның құрамына белгілерді, тұрақтыларды, типтерді, айнымалыларды, модульдерді және процедуралар мен функцияларды сипаттау кіреді. Процедуралар мен функцияларды хабарлау жеке бөлім

түрінде болады. Аталған бөлімдердің барлығы да бағдарламада толық болуы міндетті емес.

Операторлар бөлімінде BEGIN және END қызметші сөздерінің аралықтарында орындалатын операторлар тізбектеліп жазылады. Әрбір оператор белгілі бір әрекетті орындайды да, орындалатын операторлар ” **;**” таңбасы арқылы бөлініп жазылады.

Бұл екі бөлімді компилятордың ажырата алуы үшін қызметші сөздер қолданылады. Олар сипаттау шамаларының басталуын және нұсқаулардың басталуы мен аяқталуын көрсету үшін қолданылады да ағылшын тіліндегі сөздер болып келеді. Олардың ерекше қызметтері болғандықтан басқа мақсаттарға қолдануға болмайды.

Паскаль тілінде бағдарлама PROGRAM қызметші сөзінен басталады да онан соң бағдарламаның аты жазылып нүктелі үтірмен аяқталады. **VAR** сөзінен кейін есепте кездесетін айнымалылардың аты типтерімен қоса көрсетіледі. Нұсқаулар беру алдында **BEGIN** қызметші сөзі тұрады. Бағдарламаның аяқталғандығын соңына нүкте қойылған END қызметші сөзі анықтайды. Бағдарламаның атынан соң қойылған “ ; “ таңбасы бағдарлама атының аяқталып, енді деректер элементтерін сипаттау керек екендігін көрсетеді.

Паскаль тілінде әрбір операторлың жаңа жолдан басталып жазылуы міндетті емес, олар “ ; “ таңбасы арқылы да бөлініп бір жолға жазыла береді. Операторларды жаңа жолдан жазу бағдарламаны түсіну, оқу процестерін жеделдетеді.

Бағдарламаның аты белгілі бір атаудан тұрады да онан соң жақшаның ішінде бағдарламаны сыртқы ендіру/шығару құрылғыларымен байланыстратын стандартты INPUT, OUTPUT процедуралары көрсетіліп “ ; “ таңбасымен аяқталады. INPUT-енгізу, OUTPUT шығару файлдарын көрсетеді.

Паскаль тілінде бағдарламаның жалпы құрлымын мына түрде көрсетуге болады:

Program бағдарламаның аты (Input, Output);
Uses - модульдер бөлімі;
Label - белгілер бөлімі;

Const - тұрақтылар бөлімі;
Type - типтер бөлімі;
Var - айнымалылар бөлімі;
Procedure, Function – процедуралар және функциялар бөлімі;

```
Begin
   1 оператор;
   2 оператор;
   3 оператор;
    .............
    п оператор
End.
```

Бір жолдың бойында бірнеше сипаттамалар мен операторларды жазуға болады. Сипаттамалар мен операторларды жаңа жолға тасымалдауға болады, бірақ та жеке сөздерді, тұрақтыларды, құраушы символдарды тасымалдауға болмайды. Бағдарламаны оқу және түсіну жеңіл болуы үшін пробелдер(бос орын), бос жолдар және түсініктеме(комментарий) қолданылады. Түсініктеме дегеніміз – (* “символынан басталып “*)” символымен аяқталады. қос таңбаның арасында кез келген ұлттың әріптерін қолдануға болады. Түсініктемелердің бағдарламаның жұмысына ешбір әсері болмайды, олар мәтінмен жазылып, бағдарламаны түсінуге көмектеседі. Кейде түсініктеме құрастыруға { және } – таңбаларын да қолдануға болады.

Бағдарламаның аталуын, сипаттау бөлімін, операторлар бөлімін айқын көрсету үшін Program, Begin, End сөздерін бір поцизияда жазу керек. Ал оларға тиісті сипаттаулар мен операторлар белгілі бір позицияға ығыстырылып жазылады.

Мысалы, Радиусы 0,2 болатын шардың көлемін табу бағдарламасын жазайық:

```
(*-----------------------------------------------
!        Шардың көлемін есептеу        !
------------------------------------------------ *)
PROGRAM  M1 ( INPUT, OUTPUT);
   CONST PI=3.14;
VAR
       R: REAL;   ( * ШАР РАДИУСЫ  *)
```

```
    V: REAL;   ( * ШАРДЫҢ КӨЛЕМІ *)
BEGIN
   R:=0.2;
   V:=4*PI*R*R*R/3;
   WRITELN(‘ ШАРДЫҢ КӨЛЕМІ=’,V:8:3)
END.
```

Бағдарлама үш бөлімнен: аталуы, шамаларды сипаттау және операторлар бөлімінен тұрады. Бағдарламаның аты M1, стандартты атаулар INPUT, OUTPUT, тұрақты шама PI=3.14.

Айнымалыларды сипаттау бөлімінде R және V айнымалылырының REAL – нақты типке жататындығы көрсетілген. WRITELN(‘ ШАРДЫҢ КӨЛЕМІ=’,V:8:3) жолы экранда ШАРДЫҢ КӨЛЕМІ= және V шамасын 8:3 форматында шығару үшін қолданылған. Мұндағы 8 - барлық сандарды көрсетуге арналған позиция саны, 3 – бөлшек бөлікті көрсетуге бөлінген позиция саны.

Бағдарламаны орындау нәтижесі ШАРДЫҢ КӨЛЕМІ=_ _ _ 0.033 түрінде экранда көрінеді.

3.2. ДЕРЕКТЕРДІҢ ТИПТЕРІ.

Типтер бағдарлама обьектілерінің (константтар, айнымалылар) мәндерінің жиынын және олармен орындалатын операциялардың түрін анықтайды. Мысалы, 1 және 3 бүтін типке жатады да олармен барлық арифметикалық операцияларды орындауға болады. “ Үздік” және “ оқушы” жол түріндегі типке жатады, олармен қию, біріктіру, мәтінді конкатенациялау әрекеттерін ғана орындауға болады. Turbo Pascal бағдарламалау тілінде қолданылатын барлық деректер екі үлкен топқа бөлінеді.

1) **Скалярлық (қарапайым);**
2) **Құрылымдық (күрделі).**

Скалярлық тип өз кезегінде ***стандартты*** және ***қолданушыға арналған*** болып бөлінеді. Стандартты типтерге бүтін, нақты, символдық(литерлік), логикалық (бульдік) деректер кіреді. Құрылымдық тип негізіне: жолдар, массивтер, жиындар, жазбалар, файлдар және сонымен қатар деректердің скалярлық типтері де кіреді.

Деректердің типін анықтау маңызды роль атқарады. Типтерге байланысты компьютер жадында айнымалыларға белгілі өлшемдегі жад ұяшығы бөлінеді. Бұл өлшемдер шектелген , сондықтан әрбір типке белгілі бір мәндердің диапозоны бөлінеді.

Стандартты типтер.

Бүтін және нақты типтер сан түріндегі деректермен жұмыс істеуге арналған.

Бүтін тип Turbo Pascal тілінде бүтін сандар интервалын құрайды. Олар мен орындалатын операциялар берілген деректер мен нәтижелер сол интервал аралығында болғанда ғана орындалады. Бүтін типті айнымалыларды анықтау үшін айнымалыларды сипаттау бөлімінде стандартты идентификатор INTEGER пайдаланылады, мысалы:

VAR i, j, k:INTEGER;

I, j, k – идентификаторы бүтін типті айнымалыларды анықтайды.

Егер бүтін типті операндтар пайдаланылатын болса, онда келесі операциялардың нәтижелері де бүтін типке жатады.

-көбейту, + - қосу, - - азайту, MOD -бөлу, DIV -сандарды бүтін етіп бөлу.

Мысалы: 14 DIV 3 =4; 14 MOD 3 =2; 15 MOD 3=0.

(*, DIV, MOD) операциялары (+, -) операцияларымен салыстырғанда жоғарғы приоритетке жатады. Бірдей приоритетке жататын операциялар солдан оңға қарай орындалады. Төмендегі стандартты функциялардың нәтижелері бүтін типке жатады:

ABS(x) - х-тің абсолют мәнін есептейді;

SQR(x) -х-тің квадаратын есептеуге пайдаланылады;

TRUNC(x) -х нақты сан болса, функцияның мәні х-тің бүтін бөлігіне ие болады;

ROUND(x) -функция мәні нақты санға жақын ең үлкен бүтін санға тең болады

*Ж*уықтауға төмендегідей ереже қолданылады:

ROUND(x) = TRUNC(x+0,5), егер x = > 0

ROUND(x) = TRUNC(x-0,5), егер x < 0,

Мысалы: аргумент мәні	TRUNC(x)	ROUND(x)
x = 8,725	8	9
x = -4,90	-4	-5

Бүтін типті деректердің диапозоны 1-кестеде көрсетілген.

1 кесте

Бүтін типтің аталуы	Мүмкін мәндерінің диапозоны	Жадтағы орны (байт)
Byte (байттық)	0-255	1
Shortint (қысқа бүтін)	-128-127	1
Integer (бүтін)	-32768-32767	2
Word (сөз)	0-65535	2
Longint (ұзын бүтін)	-2147483648-2147483647	4

НАҚТЫ ТИП (REAL) Компьютердің мүмкіншілігімен анықталатын барлық нақты сандардың ішкі жиыны нақты типті мәндер облысы болып белгіленеді. Нақты типті айнымалыларды сипаттау үшін айнымалыларды сипаттау бөлімінде стандартты идентификатор REAL пайдаланылады, мысалы x, y және z айнымалылары нақты типті:

VAR x,y,z:REAL

Егер операндтардың біреуі нақты типке жатса, онда келесі операциялардың нәтижелері де нақты типке жатады:

(+)	(-)	(*)	(/)
қосу	азайту	көбейту	бөлу

бөлу операциясында ғана екі операнд та бүтін типке жатса да нәтижесі әрқашан нақты типке жатады. Көбейту және бөлу операциялары қосу және азайту операцияларымен салыстырғанда жоғарғы приоритетке жатады. Жақшасыз өрнектердің есептеу ережелері бүтін типке қолданатын ережелерге ұқсас. Нақты аргументтерге қолданылған стандартты функциялар ABS(x) және SQR(x) нақты типті нәтижені береді. Нақты немесе бүтін аргументтерге қолданылған: SIN(x), COS(x), EXP(x), ARCTAN(x), SORT(x) функциялардың мәндері де нақты типке жатады.

Нақты типті деректердің диапозоны 2-кестеде көрсетілген.

2 кесте

Нақты типтің аталуы	Мүмкін мәндерінің диапозоны	Мәнді цифрларының саны	Жадта-ғы орны (байт)
Single(бірлік дәлдікте)	1.52.9e-39-1.7e38	7-8	4
Real (нақты)	2.9e-45 –3.4e38	11-12	6
Double (екі еселенген дәлдікте)	5.0e-324-1.7e308	15-16	8
Extended (жоғары дәлдікте)	3.4e-4932-1.1e4932	19-20	10
Comp (күрделі)	-2e63+1-2e63-1	19-20	8

ЛОГИКАЛЫҚ ТИП (BOOLEAN). Логикалық типті айнымалылар тек екі мәнде ғана анықталған: **TRUE** (ақиқат) немесе **FALSE** (жалған). Олар төмендегідей реттелген: FALSE < TRUE. Логикалық типті айнымалыларды баяндау үшін айнымалыларды баяндау бөлімінде стандартты идентификатор **BOOLEAN** пайдаланады.

Мысалы: *VAR a,b,c:BOOLEAN;*

a,b,c – айнымалылары логикалық типке жатады. Логикалық тип операндтарға келесі амалдар анықталған: NOT – кері (терістеу); AND – және (конъюкция); OR – немесе (дизъюнкция). Бұл операцияларды нәтижелері логикалық типке жатады.

NOT амалы (логикалық терістеу). NOT A өрнегінің мәні A мәнін керісінше болады,

мысалы: NOT(TRUE) = FALSE

NOT(FALSE) = TRUE

AND амалы (логикалық көбейту). A AND B өрнегінің логикалық мәні A және B операндаларының мәндері TRUE болғанда ғана TRUE (ақиқат) болады, ал қалған жағдайларда өрнектің логикалық мәні FALSE (жалған) болады:

<TRUE> AND <TRUE> = <TRUE>
<FALSE> AND <TRUE> = <FALSE>
<TRUE> AND <FALSE> = <FALSE>
<FALSE> AND <FALSE> = <FALSE>

OR амалы (логикалық қосу). А және В операндасының біреуінің мәні, TRUE болғанда A OR B өрнегінің логикалық мәні - TRUE (ақиқат) болады, ал қалған жағдайларда нәтиже FALSE (жалған) логикалық мәнін береді:

<TRUE> OR <TRUE> = <TRUE> <FALSE> OR <TRUE> = <TRUE>
<TRUE> OR <FALSE> = <TRUE> <FALSE> OR <FALSE> = <FALSE>

Салыстыру амалдарының (=, <>, <, >, >=) нәтижелері логикалық типке жатады. Аталған амалдардың приоритеті төмендегідей ретпен орындалады: NOT, AND, OR.

Логикалық типке келесі стандартты логикалық функциялар қолданылады: ODD(x), EOLN(x), EOF(x).

Егер х бүтін тақ сан болса ODD(x) логикалық функциясының мәні TRUE (ақиқат), керісінше FALSE (жалған).

Егер х тексті файлдың жолының соңын білдіретін символ болса логикалық функция EOLN(x) = TRUE (ақиқат).

Егер х кез келген файлдың соңын белгілейтін символ болмаған жағдайларда EOF(x) логикалық функциясы FALSE (жалған) мәнін қабылдайды.

Файл жолының соңын білдіретін символдың ASCII – таблица тізбегіндегі нөмірі – 10, ал файл соңын белгілейтін символ – 28.

СИМВОЛДЫҚ ТИП (CHAR). Компьютер мүмкіншілігімен анықталған таңбалар жиынының элементтері таңбалық типті мәндер болып саналады.

Таңбалық типті айнымалыларды анықтау үшін айнымалыларды баяндау бөлімінде стандартты идентификатор CHAR пайдаланылады:

```
VAR ctr1, ctr2:CHAR;
```

Әрбір таңбалық типті айнымалы тек қана бір таңбалық мәнді қабылдай алады. Таңбалық типті мәндер жиынының элементтері

нөмірленген деп саналады. ПАСКАЛЬ тілінің таңбалар жиыны келесі талаптарға сай болуы керек:

- кез келген жиынында барлық таңбалар тұрақталған және реттелген;
- жиынға латын алфавитінің барлық бас және кіші әріптері енгізіліген;

Бұл жиын алфавит бойынша реттелген, бірақ байланысты (жүйелі) болуы міндетті емес;

-жиынға араб цифрлары 0,1,2...9 енгізілген. Бұл жиын өсу реті бойынша реттелген және байланысты (жүйелі) болады;

-жиынға мынандай таңбалар енгізілген: бос орын (пробел), үтір, нүкте т.б. Тырнақша – жақшаға алынған таңба таңбалық типті константа құрайды.

Латын алфавитінің бас әріптерінің төмендегідей орналасуы

“A” < “B” < “C” … < “Y” < “Z”

Олардың реттелгені деп қабылданған. Бұл жиын байланысты болуы міндетті емес. Олай болса әріптердің арасында басқа таңбалар кездесуі мүмкін.

Цифрлардың мынадай орналасуын:

“0” < “1” < “2” < … < “9”

цифрлар жиынының реттелгені дейді. Цифрлардың арасында бос орын болмаған жағдайда олар бір – бірімен байланысты болады.

Таңбалық мәндерге ешқандай стандартты операциялар (қосу, алу, көбейту т.б) анықталмаған, бірақ символдық мәндерін салыстырып оқу, баспаға шығару операцияларына қатыса алады және де меншіктеу нұсқауында да пайдаланылуы мүмкін.

Таңбалар жиынына екі стандартты түрлендіру функциялары қолданылады:

ORD(c) – ASCII – таблицасының тізбегінен “c” таңбасының нөмірін анықтайды (0 мен 255 арасында).

CHR(k) – ASCII – таблицасының тізбегінің k –сыншы таңбасын анықтайды.

Символдық және жол түріндегі тип символдар және олардан құралған жолдар түріндегі деректерді бейнелейді. Олар компьютер жадында сан түріндегі кодтар болып

сақталады.Сандық кодтар экранға шыққан кезде әріптерге түрленеді. Символдық және логикалық деректер типі 3 – кестеде берілген.

3 кесте

Тип	Мәндерінің диапозоны	Жадтағы орны (байт)
CHAR (символдық, литерлік) BOOLEAN(бульдік)	Кодтық кесте симиволы TRUE, FALSE	1 1

Byte , Shortint, Integer, Word, Longint типтерімен сипатталатын айнымалылар бүтін мәндерді ғана қабылдайды. Byte, Word типтерінің таңбалары болмайды.

Single, Real, Double, Extended, Comp типтерімен сипатталатын айнымалылар нақты мәндерді ғана қабылдайды да оң және теріс болады. Қарапайым бағдарламаларда integer және real типтері қолданылады. Comp типі бухгалтерлік есептеулерде қолданылады. Деректер бағдарламаларда константа және айнымалылар түрінде бейнеленеді. Бағдарламаның орындалуы кезінде айнымалылар белгілі бір мәндерге ие болады. Ол мәндер мен айнымалылар белгілібір типке жатуға тиісті.

Бүтін типтегі константа **–** ондық жүйеде нүктесіз жазылған кез келген сан. Ол теріс немесе оң сандар болуы мүмкін. Егер ол оң сан болса “+” таңбасын жазбауға болады. Бүтін константтардың мысалы: 14, -245, 0, 5390. Мәндері ретінде константтарды қабылдайтын айнымалылар бүтін типке (Integer) жатады.Бүтін типті деректермен : +, -, *, / амалдарымен қатар DIV- бөлшек бөлігін алып тастап бөлу, MOD – бөлу кезінде бүтін қалдықты ала отырып бөлу амалдарын орындауға болады. Мысалы A, B, C бүтін айнымалыларының мәндері A=25 , B=2, C=-17 болсын. Төмендегідей операцияларды орындайық:

A+51 нәтижесі: 76,
B-A нәтижесі: -23,
B*C нәтижесі : -34,
A DIV B нәтижесі: 12,
A MOD B нәтижесі: 1.

MOD көпшілік жағдайда бүтін санның 2-ге қалдықсыз бөлінетіндігін анықтау үшін қолданылады. Егер санды 2-ге бөлгенде қалдық 0 –ге тең болса сан жұп, ал қалдық қалатын болса тақ болады.

Нақты типтегі константа паскаль тілінде екі түрде: нүктелік бекітілген және жылжымалы нүктелі болып бейнеленеді.

Нүктелік бекітілген константа ондық жүйедегі бөлшек бөлігі нүктемен ажыратылған сан түрінде бейнеленеді. Мысалы, 27.3, -15.003, 200.59. Паскаль тілінде сандарда 10 – ның дәрежесі түрінде де көрсетуге болады. Оның жазылу форматы mEp. Мұндағы m – мантисса, E – ондық тәртіппен жазу белгісі, p - санның реті. m - жылжымалы нүктелі бүтін немесе нақты сан бола алады. p - шамасы міндетті түрде бүтін сан, ал мантисса құрамында "+", "-" таңбалары бола алады.

Жылжымалы нүктелі тұрақтыға – ондық ретпен жазылған сандар кіреді. Мысалдар:

Математикалық түрде Паскаль тілінде жылжымалы нүктелі константтарды жазу.

68.10^7	68E7
$0,62.10^4$	0,62E+4
$-10,88.10^{12}$	-10,88E12

Бекітілген және жылжымалы нүктелі мәндерді қабылдайтын айнымалыларды ***нақты типті*** (real) айнымалылар деп атайды. Бүтін және нақты типті деректерді ***арфиметикалық деректер*** деп атайды.

Логикалық деректердің типтерін ағылшын математигі Д. Бульдің құметіне ***бульдік*** деп атайды.

Логикалық константтың Паскальде екі түрі бар : TRUE (шын) және FALSE) жалған). Логикалық айнымалылар осы мәндердің бірін қабылдайды да BOOLEAN типті болады. Логикалық деректер белгілі бір шарттардың орындалуын тексеруде, шамаларды салыстыруда қолданылады. Деректерді салыстыруда мынандай қатынас операциялары қолданылады: < - кіші , > - үлкен, > = - кіші немесе тең, >= - үлкен немесе тең және < > - тең емес.

Логикалық деректермен OR- логикалық қосу (ИЛИ), AND – Логикалық көбейту (И) және NOT – логикалық теріске шығару (НЕ) операцияларын орындауға болады. NOT операциясы бір ғана шамамен орындала береді.

Логикалық қосу логикалық шамалардың біреуінің нақты мәні болса нақты нәтиже береді, логикалық көбейтуде екі шамада нақты болса ғана дұрыс шешім береді , логикалық теріске шығару шаманың мәні нақты болса теріс нәтиже, керісінше болғанда оң нәтиже береді.

Символдық тип мәтіндермен, жолдармен әр түрлі операциялар жасауға мүмкіндік береді.

Символдық немесе литерлік константа – кез келген апострофтың ішінде жазылған символ болып табылады да , оны тұтас константа ретінде қарастырады. Ол String типіне жатады. Мысалы: ‘ Trubo pascal бағдарламалау ортасы’, ‘Алматы’.

ТҮГЕНДЕЛЕТІН ТИП. ПАСКАЛЬ тілінде тек бір стандартты түгенделетін тип қана бар – ол логикалық тип. Сонымен қатар, бағдарламаға бағдарламашының өзіне қолайлы түгенделетін типті енгізуіне болады. Түгенделетін тип реттелген идентификатор жиынында (олар түгенделіп берілуге тиісті) анықталады.

Мәндердің кез келген стандартты емес типі бағдарламада типті баяндау бөлімінде анықталуы қажет. Түгенделетін тип типтер бөлімінде былай баяндалады:

TYPE NT = (W1, W2,…Wn),

Мұнда: NT – анықталатын типтің идентификаторы (атауы).

W1, W2,… Wn – константты идентификаторлар, олар NT – типті айнымалы қабылдайтын нақтылы мәндер. Бұл мәндер реттелген, реттелу тәртібі типті баяндайтын идентификаторлардың мынадай орналасуымен анықталады:

W1 < W2 < … <Wn.

Мысал: TYPE Апта = (дүйсенбі, сейсенбі, сәрсенбі, бейсенбі, жұма, сенбі, жексенбі);

Мұнда: Апта типі (дүйсенбі, сейсенбі, сәрсенбі, бейсенбі, жұма, сенбі, жексенбі) константалық идентификаторлар жиыны екені анықталған. Егер салыстыру операндтардың

айнымалылыларының типтері бірдей болса, онда барлық түгенделетін типтерге келесі салыстыру амалдарын қолдануға болады: =, <>, <, <=, >, >=

Барлық скалярлы типті айнымалыларға REAL SUCC(x), PRED(x), ORD(x)

SUCC(x) – функциясы реттелген тізбекте X-ке жалғас келесі элементті анықтайды, мысалы: Алфавит бойынша реттелген әріптердің тізбегі берілсін. Онда SUCC(x) функциясының мәні x - әріпінен кейінгі әріпке тең:

SUCC(B)=C; SUCC(M)=N; SUCC(дүйсенбі)=сейсенбі;

PRED(x) функциясы реттелгентізбектердің X –тің алдындағы элементін анықтайды: PRED(M)=L; PRED (жұма)=бейсенбі;

ORD(x) – функциясы реттелген тізбектегі X –элементінің нөмірін анықтайды: ORD(A)=0; ORD(D)=3; ORD(жұма)=5;

ШЕКТЕЛГЕН ТИПТЕР

Скалярлы типті айнымалыға ол қабылдай алатындай мәндердің әртүрлі ішкі жиынын анықтауға болады. Айнымалының осындай түрін анықтағанда шектелген немесе түгенделетін типтердің көмегімен баяндалады:

VAR A:MIN..MAX;

Мұнда A – айнымалысы, MIN – төменгі және MAX – жоғарғы шектерімен шектелген, ішкі жиында анықталған. Жиынның шектері (шекаралары) екі нүктемен бөлінеді. A – айнымалының негізгі типін анықтайтын жиынды MIN және MAX типіне сәйкес, мысалы: i – айнымалысы 1900 – 1995 жиынның мәндерін қабылдау үшін баяндау бөлімінде келесідей анықталады:

VAR i:1990..1995;

i – айнымалысының негізгі типі INTEGER, яғни жиынның шегі бүтін константалар.

Шектелген типтерді символдық және түгенделетін типтерде де анықтауға болады, мысалы: TYPE Апта = (дүйсенбі, сейсенбі, сәрсенбі, бейсенбі, жұма, сенбі, жексенбі);

VAR

жұмыс_күні : дүйсенбі..жұма;

айнымалы *жұмыс_күні* – Апта типіне сәйкес, дүйсенбіден жұмаға дейінгі күндердің мәндерін қабылдай алады.

Реттелген типтер. Бүтін, символдық, логикалық деректер мәндері жағынан белгілі бір тәртіппен шектелген, сондықтан оларды реттелген типтер деп атайды да компьютерде олар бүтін сан түрінде бейнеленеді. Нақты типтер электрондық есептеу машинасының ішкі форматы арқылы бейнеленеді. Олардың қабылдайтын мәндері өте үлкен болғандықтан компьютерде бүтін сан түрінде бейнелеуге болмайды. Деректердің нақты типі реттелмеген.

Trubo pascal тілінде екі қосымша реттелген қолданушыға арналған тип бар. Олар : *интервалдық* немесе диапозондық және *есептеуші* тип деп аталады. Олар осы типтегі айнымалылардың қабылдайтын мәндерін мүмкіндігінше шектеуге қолданылады. Интервалдық тип реттелген тип негізінде анықталатын минимум және максимум мәндері арқылы беріледі. Мысалы: 1..12 -- ай номерлері, ‘a’..’z’ -- латын алфавитінің әріптері.

Есептелінетін тип өз мәндерін санау түрінде беріледі. Мысалы: жол түріндегі константт: color=(red, blue, green, black). Бұл мысалда стандартты емес *color* типі пайда болады да, ондағы айнымалылар барлығы 4 мәнді: red, blue, green, black қабылдай алады.

2.3. Қолданушы идентификаторлары.

Идентификаторлар тұрақтылардың, типтердің, айнымалылардың, процедуралардың, модульдердің, бағдарламалардың және жазбалардағы өрістердің аттары ретінде пайдаланылады да бағдарламаның ***деректердің элементтерінің түрлерін сипаттау*** бөлімінде алдын ала беріледі. Идентификаторлардың аты ретінде резервтелінген сөздерді пайдалануға тиым салынады.

Паскальда идентификаторларды құрудың бірнеше ережелері бар. Олардың қысқаша сипаттамалары мынандай:

- Барлық идентификаторлар әріптен немесе сызу белгісінен басталуы қажет және оның құрамында бос орын және арнайы символдар болмауы тиіс. Келесі символдар ретінде

әріп, сызу белгісі, сан болуы мумкін. Басқа ұлттар алфавитінің әріптері идентификатор құрамына кірмейді;

- Идентификаторлардын ұзындықтары 127 дейінгі символ болуы мүмкін, бірақ олардың алғашқы 63-і ғана пайдаланылады;
- идентификаторлар арасында бір ғана ажырату белгісі болуы керек;
- идентификаторлардың атын жазуда үлкен және кіші әріптердің айырмашылығы болмайды , оны компилятор ажыратпайды.

Trubo pascal тіліндегі бағдарламаларда әрбір сөздің бірінші әріпін бас әріппен, ал қалғандарын кіші әріптермен жазу әдісі көп қолданылады.Мысалы: TextColor. Бағдарламаның атауы қарапайым түсінікті түрде болуы керек, бір идентификатор бір айнымалыны немесе бір константты таңбалау үшін бір рет қана қолданылуы қажет. Егер идентификатор екі рет қолданылса экранда қате туралы: **Error 4: Duplicate identifier** (4-қате: қосарланған идентификатор) хабарламасы шығады.

Идентификаторларды жазуда қателер жіберу мысалдары:

- 5Summa – цифрдан басталған;
- Nomer. Doma- атаулар арасына **.** қойылған;
- Blok#2 - арнайы символ қолданылған;
- Sum ma-Сөз арасына бос символ қалдырылған;

ІІІ ТАРАУ. ПАСКАЛЬ БАҒДАРЛАМАЛАУ ТІЛІНІҢ ОПЕРАТОРЛАРЫ

Операторлар Turbo Paccal тілінде қарапайым және күрделі болып екі топқа бөлінеді. Қарапайым операторлар құрамына басқа операторлар кірмейді. Күрделі (құрылымды) операторлар құрамына қарапайым операторлар кіреді. Қарапайым операторларға: меншіктеу , көшу, бос оператор, ендіру және шығару операторлары жатады. Ал күрделі операторларға құрамды

оператор, шартты көшу операторы, цикл операторы, таңдау операторы және жазбалармен жұмыс істеу операторлары жатады.

3.1. **Меншіктеу операторы.** Меншіктеу операторы айнымалының мәнін бағдарлама орындалуы кезінде беру үшін қолданылады. Меншіктеу операторы айнымалыны өрнек немесе функция түрінде анықтайды. Оның Pascal тілінде жазылу форматы:

АйнымалыАты : = өрнек;

“: =” таңбасы мәнді меншіктеу деп оқылады. Жеке жағдайларда оң жақ бөліктегі өрнек ***айнымалы*** және ***константт*** түрінде болуы мүмкін. Өрнек ***айнымалының мәнімен немесе нәтижедегі мәнінің типтері сәйкес*** болу керек. Ол сәйкес болмаған жағдлайда: **Type mismatch** (Тип сәйкес емес) түріндегі қате хабарлауы экранда шығады.

Меншіктеу операторының жазылу түрлері төмендегідей болуы мүмкін:

1 мысал. X: = Y+Z;
A: = (I>=1) AND (I<100);
I: = SQR(J)-I*K;

2 мысал. Sort := 1; baga:= 25.14; x:= X+1; y:=x:
natige:= sin(a)+cos(b); name:=’ Астана’;
Y:=SQRT(x); T:=57.45;

Меншіктеу операторы Pascal тілінде негізгі оператор болып саналады. Меншіктеу нұсқауы айнымалының мәнін өзгерту үшін қолданылады. Оң жақ бөліктегі өрнектің мәні формула түрінде есептеліп , сол жақ бөліктегі айнымалыға мешіктеледі.

Өрнектердің үш түрі қолданылады:

- арифметикалық өрнектер –сан түріндегі деректерді өңдеу үшін;

- логикалық өрнектер - әр түрлі деректерді салыстыру және логикалық амалдарды орындау үшін;
- символдық өрнектер – мәтіндерді өңдеу мақсатында.

Мысалы:

```
Var x: integer; y: real;
Begin
x:=5; y:=0.5; y:= y+x; { әзірше қате жоқ}
x:=y; { қате туралы хабарлама пайда болады}
      { себебі оң жақ бөлікте нақты сан сол жақтағы типпен сәйкес емес}
```

Құрамдас операторлар. Құрамдас операторлар олардың элементтері болып табылатын операторлардың орындалу ретін береді. Олар жазылу рентімен орындалуы керек. Pascal тілінің синтаксисінде операторлар **BEGIN, END** қызметші сөздерінің арасында бір-бірінен нүктелі үтір арқылы ажыратылып жазылады. Мысалы:

```
BEGIN  Z:= X; X:=Y; Y:= Z;  END;
```

Сызықтық немесе қарапайым алгоритмдер деп ешқандай шартсыз рет – ретімен бірақ рет орындалатын іс - әрекеттер жиынтығын айтады. Ол электрондық есептеу машиналарында тікелей орындалу режимі деп аталады. Сызықтық алгоритм көмегімен қарапайым есептерді шешуге болады.

Сызықтық алгоритмдерді бағдарламалауда берілген шамаларды ендіру және экран бетінде шығару мақсатында TURBO PASCAL бағдарламалау тілінде WRITE, WRITELN және READ, READLN операторлары қолданылады. Ол операторлардың жазылу форматтары мен экранда информацияны көрсету мүмкіндіктері төмендегідей.

3.2. Деректерді шығару.

Деректерді шығару деп- жедел жадта өңделген деректерді сыртқы құрылғыларға :монитор экранына, принтерге және дискідегі

файлға беру процесін айтады. Ол үшін шығару операторы қолданылады.

WRITE , WRITELN шығару операторлары Паскальда мына форматта жазылады:

WRITE (y1,y2,…yn);
WRITELN;
WRITELN **(**y1,y2,…yn);

Мұндағы, y1,y2,…yn экранға шығарылатын деректер тізімі. Алыңғы екі оператордың орындалкы соңғы операторға пара-пар болады. Шығару операторын қолданудың мынандай ерекшеліктері бар.

- Шығару операторының параметрінің құрамында “ ‘ “ таңбасының ішінде жазылған деректер өзгеріссіз экранға шығарылады. Мысалы. Астана сөзін, A=500 деректерін экран бетіне өзгеріссіз шығару:

WRITE (' Астана сөзін, A=500 ');

- Бүтін және нақты сандарды экранда шығару үшін WRITE операторында шығару форматтарын көрсетуге болады. Формат айнымалыдан соң “ **:** “ таңбасын қойып көрсетіледі. Формат екі шамадан : бірінші – мәнді шығаруға қажеті барлық өрісті, екінші – бөлшек бөлікті шығаруға қажетті өрісті көрсететін сан мәнінен тұрады. Жалпы өрістің құрамына санның теріс таңбасы, оң сандар үшін бос орындар, бүтін бөліктер үшін қажетті цифрлар саны, нүкте және бөлшек бөліктегі цифрларға қажетті орындар саны. Мысалы: **WRITE (Y: 5:2); Бұл Y** шамасын шығаруға жалпы 5 орын, оның ішінде бөлшек бөлігін шығаруға 2 орын бөлінгендігін көрсетеді. **Y= 5.76** болса , экранда **5.76** саны көрінеді. Егер бөлінген формат шығарылатын шамадан артық болса, бүтін бөліктің алдынан бос орындар, ал бөлшек бөліктің соңынан 0-дер қойылады.

Мысалы: **WRITE (‘Y=’,Y : 8:3);** Экранда **Y= ␣ ␣ ␣ 1.760** символы көрінеді.

- Бүтін шамаларды экранда көрсеті үшін бөлшек бөліктің форматы көрсетілмейді.

- Мысалы: Y=179 санын экранға шығару. **WRITE (‘Y=’,Y : 3);** нәтижесі : **Y=179**. Егер 3-тен артық орын бөлінсе: **WRITE (‘Y=’,Y : 5);** нәтижесі : нәтижесі : **Y= ␣ ␣ 179** болады.
- Шығару операторы параметрсіз жазылса :**WRITELN** жаңа жолға көшуді білдіреді. Бұл экранда бос жолдар тастау үшін қолданылады. **WRITELN (a1,a2,…,aN)** шығару операторы алдымен **a1,a2,…,aN** мәндерін экранға шығарады да курсорды жаңа жолға көшіреді.

Экранға шығарылатын деректер бүтін немесе нақгы сан (3,42,-1732.3), символ ('A'- 'Z', J, 'A'- 'Я'), қатар ('сәлем адамзат'), бульдік мән (TRUE) түрінде бола алады. Бұдан басқа ол атаулы тұрақгы (тұрақты аты), айнымалы, көрсеткіштік функцияның шақырылуы болуы мүмкін. Барлық деректер қатарда көрсетілген реті бойынша экранда көрінеді де курсор келесі қатардың басына орналасады. Егер курсор осы қатардағы соңғы деректен кейін қалуы қажет болса, онда: WRITE (y1,y2,…yn); құрылымы қолданылады.

Элементтерді экранға шығарғанда олардың арасында автоматты түрде бос орын қойылмайды.

1- мысал: A:=10; B:=2; C:=100;

```
WRITELN (A,B,C);                 102100
 WRITELN (A:3,B:3,C:3);          10 2  100
WRITELN{A,B:2,C:4);              10  2  100
```

2- мысал. Нақты сандар үшін өрістің кендігін (мөлшерін) көрсеткен кезде сол жақган бастап түзетіледі және экспонентті формада басылып шығарылады.

```
X:=421.53;
Writeln(X);        4.2E+02
Writeln(X::8:1 )   4.2153000000E+02
```

Сондықтан Паскаль ұзындықтың екінші операндысын косуға мүмкіндік береді:

Элемент : ұзындық : цифрлар саны. Екінші сан бекітілген нүктесі бар сандар үшін үтірден кейін қанша сан бар екенін көрсетеді.

```
X:=421.53;                    421.53
Writeln(X:6:2);               421.53
Writeln{x:8:4);               421.5300
```

3.3. Деректерді ендіру.

Орындалатын бағдарламаны құрудың алғашқы әрекеті деректерді ендіру процесінен басталады. Деректерді ендіру деп кіріс деректерді өңдеу мақсатында компьютер жадына беруді айтады. Деректерді ендірудің негізгі құрадары: пернелер тақтасы мен дискілік файл болып табылады.Ендірілгеннен соң айнымалы шамалардың мәндері қажетті есептеулер үшін қолдануға мүмкіндік береді. Паскальда оқуға пайдаланылатын деректерді пернетақтадан енгізудің екі негізгі READ және READLN функциялары бар. READ – оқу, READLN – жолды оқу мағанасын береді. Ендіру операторлары Паскальда мына форматта жазылады:

READ (x1,x2,…xn);
READLN;
READLN **(x**1,x2,…xn);

Мұндағы, x1, x2,…xn ендірілетін деректер тізімі. Айнымалылар integer, real, char, string типті бола алады. Алыңғы екі операторлардың орындалуы соңғы операторға пара-пар болады.

Бағдарламаны орындау кезінде READ операторы кездессе компьютер тоқтап деректерді ендіруді талап етеді. Деректер дұрыс ендірілсе бағдарламаның орындалуы жалғасады.

Мысалы:

```
Var I: integer; a: real; ch:  char;
Begin
READLN (I, a);
READLN (ch);
…
```

Ендіру операторларын пайдаланғанда төмендегі ережелерді білу керек:

- READLN немесе READ операторынан соң бір немесе бірнеше айнымалыларды жазуға болады. Олардың мәндерін ендіру кезінде ажырату үшін әрбір ендірілген шамадан кейін бос орын немесе “ Tab “, “ Enter” пернелерін басуға болады.
- READLN айнымалысыз бағдарламаның соңында жазылады да қолданушының “ Enter” пернесін басуына дейін үзіліс жасауына мүмкіндік береді. Кері жағдайда нәтиже пайда болған экран бағдарлама мәтінін көрсетпей қалады.
- Ендірілетін деректердің типі ендіру бөлімінде көрсетілген айнымалылар типіне сәйкес келуі керек.
- Ендірілген деректедің типі айнымалының типіне сәйкес келмеген жағдайда: **Error 106: Invalid numeric format** (106- қате: Сан форматы дұрыс емес) қатені хабарлауы шығады. Егер бағдарлама операциялық жүйеден қосылса: **Run time error 106** (106-орындау уақытындығы қате) хабарламасы экранда көрінеді.

Есеп шығару мысалдары.

1. *Мысал.* Радиусы R шардың V көлемін есептеу бағдарламасы.

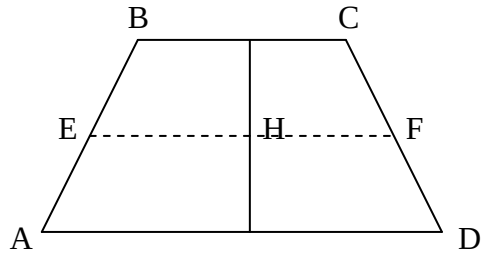
```
  (* Шардың ауданын есептеу *)
Program M2 (Input, OUTPUT);
CONST PI = 3.14;
VAR
    R: REAL;                    (* Шар радиусы *)
    V: REAL;                    (* Шар көлемі *)
BEGIN
   WRITLEN(‘ Радиус мәнін ендір – R:’);
    READ(R);
    V:=4*PI*R*R*R/3;
     WRITELN;
```

```
    WRITELN('Нетижесі:');
    WRITELN('ШАР КӨЛЕМІ=',V:8:3)
END.
```

Экран көрінісі: Радиус мәнін ендір – R:
Ендірлген сан: 0.2
Нәтижесі:
Шар көлемі = 0.033.

2. Мысал. Суретте көрсетілген трапеция ауданын тап.
   Мұндағы S трапецияның ауданы:

S = (BC + AD)H/2 формуласымен анықталады, H – трапецияның биіктігі , BC – жоғары табанының, AD төменгі табанының ұзындығы.



Трапеция ның суреті.

```
(* трапеция ауданын есептеу *);

PROGRAM M3 (INPUT, OUTPUT);
  VAR
     BC:  REAL; (* Жоғары табан ұзындығы*)
     AD:  REAL;  (* Төменгі табан ұзындығы*)
     H   :  REAL; (*Трпецияның биіктігі*)
     S    :  REAL;(* Трапеция ауданы*)
BEGIN
   WRITLEN ('BC, AD, H мәндерін ендір ')
    READ (BC, AD, H);
    S: = (BC + AD)*H/2;
    WRITLEN;
     WRITE('  Трапеция ауданы =', S:7:2)
```

```
END.
```

3. Мысал Тізбектеліп Rтіз = R1+R2 және параллель Rпар=R1.R2/(R1+R2) жалғастырылған кедергілердің мәндерін табыңдар.

```
(* Тібек кедергісін анықтау *)
PROGRAM M4 (INPUT, OUTPUT);
  VAR
       R1, R2 : REAL; (*кедергілердің мәні*)
    RPOS :  REAL;  (* тізбектелініп жалғанған кедергілер мәні*)
    RPAR : REAL;  (*параллель жалғанған кедергілер мәні *)
BEGIN
 WRITLEN (‘ R1, R2 мәндерін енгіз:’);
  READ (R1, R2);
    RPOS:=R1+R2;
    RPAR:=R1*R2/(R1+R2);
 WRITELN;
 WRITELN (‘ тізбектеліп жалғасқан R=’, RPOS:8:2);
 WRITLEN (‘ параллель жалғасқан R=’, RPAR:8:2)
END.
```

Экран көрнісі: R1, R2 мәндерін енгіз:
20 30
тізбектеліп жалғасқан R= 50,00
параллель жалғасқан R= 12,00

4. Мысал. Берілген арифметикалық өрнекті есептеу.

$y = \sqrt{x}$, $R = \sin(x + \pi / 4)$ және бүтін К шамасын бүтін N шамасына бөлгендегі қалдықты табу. π шамасын PI деп таңбалаймыз , қалдық = К/ N деп аламыз.

```
(* Арифметикалық өрнекті есептеу *)
PROGRAM M5 (INPUT, OUTPUT);
   CONST PI = 3.14;
    VAR  Y, R, X     : REAL;
                K, N    : INTEGER;
          ҚАЛДЫҚ  : INTEGER;
BEGIN
```

```
        WRITLEN (‘Мәндерді енгіз X, K, N : ‘);
    READ (X,K,N);
            Y:= SQRT (X);
            R:=SIN (X+PI/4);
             ҚАЛДЫҚ: = K MOD N;
WRITLEN;
WRITLEN (‘Есептеу нәтижесі:’);
WRITLEN (‘Y=’, Y:6:2);
WRITLEN (‘R=’, R:6:2);
WRITLEN (‘ҚАЛДЫҚ=’, :3);
END.
```

3.4. **Арифметикалық өрнектер.**

Бағдарламадағы барлық, есептеулер арифметикалық өрнектер арқылы жүргізіледі. Арифметикалқ өрнектің мәні түтін және нақты сандар түрінде болады. ***Өрнек*** деректер элементімен орындалатын әрекеттердің тәртібін сипаттайды және **операндалардан** (константа, айнымалы, функция) , **жақшалардан** және **операция таңбаларынан** тұрады.

Арифметикалық операциялар. *Операция* операндалармен орындалатынс әрекеттерді көрсетеді. Бағдарламалау тілдерінде міндетті түрде операциялардың таңбалары көрсетілуі тиіс.Мысалы (х+у)*5-10 өрнегінде х,у айнымалылары мен 5. 10 тұрақтылары операндалар деп, қосу мен азайту (+, -) арифметикалқ операциялардың таңбалары деп аталады. Барлық өрнектерде көбейту таңбасының «*» символы міндетті түрде көрсетілуі керек. Өрнекке кіретін барлық элементтер белгілі болса, онда өрнектің мәні анықталғанг деп есепелінеді.

Жақшалар операциялардың орындалу ретін басқар үшін қолданылады.Операциялар бір операцияға (мысалы таңбаны өзгерту –а) байланысты болса ***унарлық*** , екі операцияға (мысалы, а+с) болса ***бинарлық*** деп аталады. Операция таңбаларын қолданғанда екі таңба қатар келсе оның біреуі жақша ішінде жазылады.. Мысалы, а*-с болса, а*(-с) түрінде жазу керек.

Негізгі арифметикалық операциялар және олардың типтері.

Бинарлық операциялардың мынандай типтері бар:

- Қосу, азайту, бөлу көбейту операциялары ***real*** және і***nteger*** типті бола алады;
- Бүтіндей бөлу DIV, бөлуден қалған қалдық MOD, арифметикалық И And, бит бойынша солға Shl және оңға Shr жылжу, арифметикалық ИЛИ Or, арифметикалық 2 модуль бойынша биттік қосу XOR ***integer*** типіне жатады;
- Унарлық операциялар: Таңбаны сақтау - “+”, таңбаны терістеу – “-“ , арифметикалқ жоққа шағару **Not -- Integer** типіне жатады.

DIV, MOD операциялары. ***DIV*** (division) - бүтіндей бөлу деп аталады да бөліндінің бүтін бөлігін алып, бөлшек бөлігін тастап кетеді. Егер бөлінгіш бөлгіштен кем болса нәтиже нөльге тең болады.

Мысалдар:

11div5 =2; 10 div3=3; 2 div3= 0; 123 div4=30; -17 div5=-3.

MOD (modulus- өлшем сөзінен шыққан) – бүтіндей бөлу кезіндегі қалдықты анықтайды.

Мысалдар: 10 mod5= 0; 11mod5=1; 14mod5=4; 17mod-5=-2; -17mod-5=-2.

DIV, MOD операцияларының аргументтері бүтін сан.

Арифметикалық функциялар мен процедуралар.

Арифметикалық өрнектерде төмендегідей стандартты функциялар қолданылады.

Стандартты функция	Орындайтын әрекеті	Аргументінің типі	Нәтижесінің типі
Abs(x)	x - абсолют шаманы есептеу	Real/integer	Real/integer
Sqr(x)	X –ты 2 дәрежеге шығару	Real/integer	Real/integer
Sqrt(x)	$X^{1/2}$- түбір табу	Real/integer	Real/integer
Exp(x)	e^{x}-	Real/integer	Real/integer
Ln(x)	Ln(x)	Real/integer	Real/integer
pi	Пи саны	-	Real
Sin(x)	Sin(x) - есептеу	Real/integer	Real

Cos(x)	Cos(x) - есептеу	Real/integer	Real
Arctan(x)	Arctan(x) - есептеу	Real/integer	Real

Стандартты функциялар бағдарламада қажетті жерінде аттары бойынша шақарылады, олардың аргументтері міндетті түрде () - жақшасы ішінде жазылады.

Функцияларды қолдану кезінде төмендегі жағдайларды ескеру керек:

- Sin, Cos функцияларының аргументі радиан өлшемі бойынша берілуі керек. Бұрыштың градустық өлшемеін радиандық өлшемге айналдыру үшін берілген бұрышты pi/180 өрнегіне көбейту қажет.
- Arctan функциясын есптеу нәтижесі радиан өлшемімен алынады.

Кестедегін басқа да төмендегідей стандартты функциялар мен процедуралар қолданылады:

- ***Random*** (диапозон) – ***0 <= x < диапозон*** шартын қатағаттандыратын кездейсоқ санды шығарады, оның аргументі мен нәтижесінің типі- word. Егер бізге a <= x < b диапозонында кездейсоқ бүтін сан қажет болатын болса, ***Random(b - a)+ a*** өрнегін қолданамыз. Егер диапозон параметрі көрсетілмеген болса, ***Random*** операторы 0 <= x < 1 диапозонындағы нәтижесі real болатын x санын шығарып береді.Бізге басқа диапозондағы нақты санды шығарып алу қажет болса, ***Random* b +a*** өрнегін жазамыз. Random функциясын ең алғаш шақыру алдында ***Randomize*** процедурасын шақыруымыз керек.
- ***Dec(x,n)*** – процедурасы бүтін айнымалы x –тың мәнін ***n*** шамасына азайтады.

Мысалы: x:=10; dec(x,2); нәтижесі – 8. ***n*** параметрі көрсетілмесе процедура dec(x) түрінде жазылып x –тың мәні 1-ге кемиді.

- ***Inc(x,n)*** процедурасы бүтін айнымалы x –тың мәнін ***n*** шамасына арттырады.

Мысалы: x:=10; Inc(x, 3); нәтижесі – 13. ***n*** параметрі көрсетілмесе процедура Inc(x) түрінде жазылып x –тың мәні 1-ге арттырады.

- ***Frac(x)*** функциясы х-тың бөлшек бөлігін есептейді. Аргумент пен нәтиже ***real*** түрінде болады. Мысалы write(frac(0.25*11):4:2); { нәтиже – 0,75};
- ***int(x)*** функциясы х-тың бүтін бөлігін есептейді. Аргумент пен нәтиже ***real*** түрінде болады. Мысалы write(int(422.117) :4:2); { нәтиже – 422.00};
- ***TRUNC(x)*** функциясы нақты х санына тең немесе одан кем бүтін санды х >=0 болғанда, ал х-тен үлкен немесе тең бүтін санды х<=0 (truncate -кесу) болған жағдайда анықтауға мүмкіндік береді. Осы тәртіппен санның бөлшек бөлігін алып тастайды, чғни бүтін сан түріндегі нәтие алу үшін қолданылады. Оның аргументі real, нәтижесі - longint. Мысалы, trunc (6.7); { нәтиже : 6}; trunc (-1,6); { нәтиже : -1}.
- ***ROUND(x)*** функциясы х санының ең жақын жуықталғын бүтін сан түріндегі мәнін анықтайды. Оның аргументі real, нәтижесі - longint.

Мысалы, round (6.7); { нәтиже : 7}; round (-1,6); { нәтиже : -2}.
Жоғарыдағы екі функция типтерді түрлендір үшін қолданылады да ***типті айқын түрлендірушілер*** деп атайды.

Сандарды дәрежелеу. Trubo Packal тілінде сандарды дәрежелеу логарифмдердің қасиеттерін пайдалану арқылы: $\mathbf{c = a^b \Rightarrow ln(c) = bln(a) \Rightarrow c = e^{bln(a)} \Leftrightarrow exp(bln(a))}$ формуласын қолданады. Бұл әдіспен теріс сандарда дәрежелеуге болмайды , ол үшін циклдер қолданылады.

Кейбір есептеулерде қолдануға арналған формулалар:

- Негізі **a** болатын логарифмді есептеу: $\log_a(x) = \ln(x)/\ln(a)$;
- tg(x)=sin(x)/cos(x);
- ctg(x)= cos(x)/sin(x);
- csc(x)=1/sin(x);
- $\arcsin(x) = arctg\left(\frac{x}{\sqrt{1-x^2}}\right)$;

- $\arccos(x) = arctg\left(\frac{\sqrt{1-x^2}}{x}\right) = \frac{\pi}{2} - \arcsin(x);$
- $\text{arctg(x)} = \frac{\pi}{2} - arctg(x).$

IV ТАРАУ. Бағдарламалау тілінің басқарушы операторлары.

Бағдарлама жасаудың негізгі принциптері:

1. Бағдарлама қатаң түрде орындалатын коммандалар тізбегінен тұрады.
2. Бағдарламаны құрайтын командалар орындалатын және орындалмайтын болып екіге бөлінеді.
3. Есептеу машинасы командаларды орындау үшін рет – ретімен бірінен екіншісіне көшеді.
4. Командалардың орындалуы басқарушы командалар көмегімен бұзылуы мүмкін.

Орындалатын оператор – бағдарлама операторларының орындалу ретін бұзбайды.

Басқарушы оператор – бағдарлама операторларының орындалу ретін өзгертетін операторлар.
Кестеде басқарушы операторлар көрсетілген.

Оператор	Орындайтын әрекеті
END.	Бағдарлама орындауды аяқтау
GOTO	Шартсыз көшу
IF – THEN – ELSE	Шартты көшу

FOR – TO	Арифметикалық цикл
WHILE	Итерациондық 1 цикл
RePEAT	Итерациондық 2 цикл
CASE-	Шартты құрылым Ішкі
ON – ERROR	Қатені өңдеу процедурасына көшу

Басқарушы операторларды қолдану ***логикалық өрнектердің түрлері мен қатынас белгілерін*** қолдануды қажет етеді. Шарттар әр түрлі қатынас белгілері қолданылатын логикалық өрнектер түрінде жазылады. Қатынас операциялары екі операнданы салыстырып , қатынастың шын - true немесе жалған екендігін – false анықтайды. Логикалық операциялар таңбаларын пайдаланып күрделі логикалық өрнектер алуға болады.

Логикалық операцияның орындалуы оның жазылу тәртібіне байланысты болады. Trubo Pascal-да логикалық операциялар қатынас операцияларына қарағанда үлкен роль атқаратындықтан логикалық өрнектердегі әрбір қарапайым шарт жақша ішіне алынады.

Мысалы, бүтін m, mmax, n, nmax шамалары үшін m > mmax and n > nmax өрнегі қате туралы хабарлама береді , себебі алдымен ***mmax and n*** операциясы орындалады. Ол дұрыс орындалуы үшін жақшаларды қолданып өрнекті мына түрде жазамыз: (m > mmax) and (n > nmax).

Логикалық өрнектерді жазғанда мынандай заңдылықтарды білуіміз керек:

- Бағдарламада екі жақты 1<x<2 түріндегі теңсіздіктерді жазуға болмайды , ол (x>1) and (x<2) түрінде жазылуы керек.
- x=y=z түрінде теңдік жазуға болмайды ол (x=y) and (x=z) түрінде жазылуы керек.

- Берілген x санының -2 , +2 диапозонда жатпайтындығын көрсету үшін : not((x>-2) and (x < 2)) немесе (x<=-2) or (x=2) өрнегін қолдану қажет.
- Радиусы 1 бірлікке тең дөңгелектің ішінде нүктенің орналасу шарты: sqr(x) +sqr(y)<1;

4.1. Шартсыз көшу операторы GOTO.

Бағдарламада бір оператор орындалып болғаннан соң келесі операторлардан басқа таңбаланған орындағы операторды орындау үшін шартысзы көшу операторы Goto қолданылады. Ол бағдарламаның Label хабарлау бөлімінде көрсетіледі. Оның жазылу форматы:

Goto ТаңбаАты;

Оператор көмегімен бағдарламаның таңбаланған кез келген жолына көшуге мүмкіндік беріледі. Таңба аты әріптер, сандар түрінде белгіленіуі мүмкін. Таңба бағдарламада атынан кейін қос нүкте “ **:**” қою арқылы жазылады.

Мысалы:

```
Label  Tanba1;
…
begin
…
Tanba1: Оператор;
            …
   goto Tanba1;
end.
```

Goto операторын қолданғанда мына ережелерді ескеру керек:

- Таңба бағдарламаның таңбаны хабарлау бөлімінде көрсетілуі керек;
- Көшу бағдарламаның белгілі блогында ғана орындалады;

- Құрылымдық бағдарламаларда бұл операторды мүмкіндігінше аз пайдалану керек.

Goto операторын қолдану мысалы:

```
 Program SumR;
Const Eps = 0.01;
Label 10;
Var S, I, R : Real;
Begin
    S:=0.0;
    I:=0.0;
10 ; I:=I+1;
    R: =1/I;
    S:=S+R;
    IF  (R>Eps) Then Goto 10;
       Writeln (‘қатардың қосындысы=’ , S);
End.
```

2.2. Бос оператор.

Бос оператор құрамында символ болады да ешқандай әрекет жасамайды. Ол таңба арқылы көрсетіледі де, бағдарламадан немесе құрамдас операторлардан шығу үшін қолданылады. Мысалы:

```
Begin
…
goto  metka; ( блоктың соңына көшу)
…
metka: ( таңбаман көрсетілген бос оператор)
end.
```

4. 3. IF –шартты көшу операторы.

Бұл шартты көшу опрераторын, екі түрлі нұсқада жазуға болады.

1-нұсқа:

IF Шарт

```
    THEN
         BEGIN
               {егер шарт орындалса }
               { мұндағы операторлар орындалады}
         END
     ELSE
        BEGIN
              {егер шарт орындалмаса}
              { мұндағы операторлар орындалады}
       END;
```

2- нұсқа:

```
IF   Шарт
    THEN
         BEGIN
               {егер шарт орындалса }
               { мұндағы операторлар орындалады}
         END;
```

Бұл шартты көшу операторының қысқа түрі деп аталады.

Мұндағы Шарт кез – келген тексерілетін өрнек. IF - “ егер”, THEN - “онда” , ELSE - “әйтпесе” кілт сөздерін білдіреді. Шартты көшу операторының орындалуы шартты тексеруден басталады. Егер шарт орындалса , “иә (true)” THEN сөзінен соң орналасқан операторлар орындалады . Шарт орындалмаса , яғни шарт жалған (false) болса ELSE қызметші сөзінен кейінгі оператор орындалады. ELSE қызметші сөзінен соң ешқандай оператор болмаса , шарт орындалмаған жағдайда онан кейінгі тұрған операторлар орындалады.

Мысалы:

```
If a> b then writeln (‘a үлкен  b’);’
          Else   writeln (‘a кіші немесе тең b’);’
```

Немесе:

```
If  (a>=0) and (a<10)  then writeln (‘ бір орынды сан’);’
```

Бір if операторы екінші if операторының құрамына енуі мүмкін, бұл жәағдайды ішкі операторлар деп атайды. Олар үш нұсқада жазылуы мүмкін:

- 1-нұсқа.

```
If  1Шарт  then
    If  2Шарт  then  Оператор1
                Else   Оператор2
          Else Оператор3;
```

- 2-нұсқа.

```
If  1Шарт  then  Оператор1
            Else If  2Шарт  then  Оператор2
                              Else   Операто3;
```

- 3-нұсқа.

```
If  1Шарт  then
    If  2Шарт  then  Оператор1
                Else   Оператор2;
```

Егер шарт бір біріне әсер етпесе, олардың есептеу реті ешқандай роль атқармайды, оларды бағдарламада белгілі бір тәртіппен орналастыруды керек. Орындалу мүмкіндігі жоғары шарт бірінші орында , ал орындалу мүмкіндігі аз шарт екінші орында тұруы бағдарламаның орындалуын жеделдетеді.

Шартты көшу операторларымен жұмыс істегенде мыналарды ескеру керек:

- Ішкі операторларда әр бір ***else*** операторына сәйкес ***then*** операторы болуы керек;
- 2-3 –тен артық ішкі операторларды аз қолдану керек;
- Егер begin және end конструкцисыны арасында бір ғана оператор болса, begin, end сөздерін жазбауға болады;
- Шартты көшу операторларында then сөзінің соңында және else сөзінің алдында “ **;** ! қойылмайды;
- Күрделі логикалық өрнектер and, or, not операцияларымен байланысты бірнеше и/или логикалық айнымалыларының қатынастарынан тұруы мүмкін. Әр бір қатынас жақшалар арқылы жазылады.

Листинг 3,4 3,5 3,6 ,3,9, 3,10

4.4 . CASE – таңдау операторы.

Бағдарлама жазуда if операторының ішкі операторларын көп рет қолдану , оны күрделендіріп жібереді. Сондықтан одан құтылу мақсатында таңдау операторы -- CASE қолданылады. Оның жазылу форматы төмендегідей:

```
 CASE  Өрнек - тест  of
1КонстантТізім: begin
                    { 1Нұсқау}
                     end;
2КонстантТізім: begin
                    { 2Нұсқау}
                     end;
NКонстантТізім: begin
                    { NНұсқау}
                     end
else
     begin
        {Нұсқаулар}
     end;
end;
```

Мұндағы: Өрнек- тест – кез – келген сан, символ түріндегі өрнек. Case операторын орындау ӨрнекТест тексеруден басталады. Case сөзінен соң орналасқан ӨрнекТест константымен өрнектің тізімдегі мәні сәйкес келсе *begin, end* арасындағы нұсқаулар орындалады. Егер сәйкес келмесе else сөзінен кейін орналасқан *begin, end* конструкциясы арасындағы операторлар орындалады. Case операторы *end* операторымен аяқталады.

CASE- ***таңдау операторларымен жұмыс істегенде мыналарды ескеру керек:***

- Case инструкциясы if операторының қортындылаушысы болып табылады. Ол бірнеше нұсқаулардың ішінен қажеттісін таңдап алуға мүмкіндік береді.
- Бғдарлама нұсқауларын таңдау *ӨрнекТест өрнегінің* мәніне таңдау константының мәніне байланысты болады.
- *ӨрнекТест* реттелген тип : көп жағдайда – integer, сирек жағдайларда char, boolean немесе қолданушы типі болады.
- ӨрнекТест -тің мәндері –32768 – ден 32767-ге дейінгі диапозонда өзгереді.
- Таңдалынатын константтар тізімі бір бірінен үтір таңбасымен бөлініп жазылған мәндерден немес диапозондардан тұруы мүмкін. Диапозон шекаралары “ **..** “ шектеуімен екі константарқылы жазылады. Константтиптері *ӨрнекТестің* типпімен сәйкес келуі тиіс.Бір таңдау операторындағы таңдау константтары әр түрлі болуы керек, ондай болмаса бірінші сәйкес тармақ қана орындалады. Әр түрлі операторлардағы таңдау константтары бірдей бола алады.
- Таңдау тізімінің соңғы элементінен кейін “**;** “ таңбасы қойылмайды.

1. Мысал. Апта күндерін анықтау бағдарламасы мен танысайық:

```
(* Апта күндерін анықтау *)
 var d,m,y: integer; n: longint;
begin
    writeln (‘ күнді, айды, жыл датасын “5  08  2003” форматында ендір’);
    readln (d,m,y);
    if(m>=2) then m:=m+1
            else
                begin
                    m:= m=13;    y:=y-1;
                end;
n:= trunc(365.25*y)+ trunc(30.6*m)+ d-621050;   n:= n-trunc( n/7)*7+1;
```

```
case n of
1: write (’дүйсенбі’);
2: write (’сейсенбі’);
3: write (’сәрсебі’);
4: write (’бейсенбі’);
5: write (’жұма’);
6: write (’сенбі’);
7: write (’жексенбі’);
            end;  writeln;
end.
```

4.5. **Қайталау (цикл) операторлары .**

Деректерді өңдеу және есептеулерде қайталанып орындалатын әрекеттер жиі кездеседі. Ол әрекеттерді орындау үшін цикл операторлары қолданылады.

Цикл көп рет қайталанып орындалатын операторлар тізбегінен тұрады. Паскаль бағдарламалау тілінде негізгі циклдің үш түрі қолданылады:

- Алдын-ала шарт қоятын цикл – (**While**);
- Соңына қарай шарт қоятын цикл (**repeat**);
- Есептеуіш цикл (**for**).

Цикл операторларын қолданғанда төмендегі жағдайларды еске сақтау қажет:

. Берілген есепті әр түрлі әдіспен кез келген цикл операторының түрін қолданып шешуге болады;

. Есептің нәтижесі барлық циклдің түрінде бірдей болады;

. While ең әмбебап циклдің түріне жатады;

. repeat - ең қарапайым цикл , оны цикл қызметін алғаш рет қолдануды үйрену үшін пайдалануға болады;

. for – циклі қайталау саны белгілі операцияларды орындауда қолданылады.

. Циклді дұрыс қолданбау компьютердің тұрып қалуына мәжбүр етеді. Trubo Pascal бағдарламалау

ортасында одан шығу үшін < Ctrl> + <Break> пернелер комбинациясын қолданады.

4.6. **REPEAT цикл операторы.**

REPEAT циклі әрекеттерді қайталау саны белгісіз болған жағдайда қолданылады. Операторлар тізбегінің қайталанып орындалуын басқаратын **REPEAT** цикл операторындағы өрнек сол **REPEAT** операторының ішінде орналасады.

Оның жазылу форматы:

Repeat

оператор;

оператор;

.....

Until ***Өрнек***

Мұндағы Repeat (қайталау) , Until (әзірше) қызметші сөздері. Бұл циклдің мынандай ерекше қасиеттеі бар:

. ***Өрнекті*** тексеру цикл денесінің соңында орындалғандықтан цикілдегі

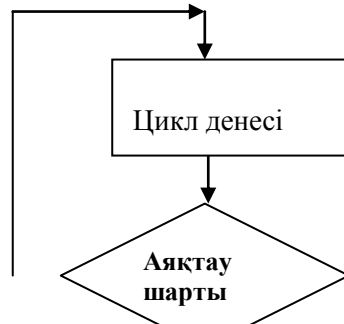
операторлар ең болмаса бір рет орындалады.

. **REPEAT** циклі Ө***рнек*** **TRUE** мәнін қабылдамайынша орындала береді.

. **REPEAT** цикілінде құрамдас оператор емес, жай операторлар тобы.пайдалануы мүмкін. Бұл циклді пайдаланған кезде **BEGIN** және **END** сөздерін пайдаланылмайды.

. Циклдік бөлікте белгілі бір көшетін орында көрсететін Таңба болса, онда цикл соңына жетпей аяқталады.

Циклдің блок –схемасы:



—

Жоқ иә

1 Мысал. У=x^2 функциясының мәндерін х=8, 6, 4, 2 мәндерінде есептеу бағдарламасын жазайық.

```
X:8;
Repeat
Y:=x*x;
Writeln (x:3, y:5);
X:=x-2;
Until x<0;
```

Мұнда алдымен Х шамасының мәні беріледі де, циклдік бөлікте У –тің сәйкес мәні есептелініп монитор экранында Х-тің мәні шығарылады : Одан соң Х-тің мәні 2 –ге азайтылып, У-тің мәні қайта есептелінеді. Бағдарламаның циклдік бөлігі Х-тің мәні 2- ден кіші болғанша орындалады. Өрнектің нәтижесі бульдік типті нәтиже болуы керек. **REPEAT** және **UNTIL-ге азайтылып** кілттік сөздері арасындағы операторлар өрнектің нәтижесі **TRUE** мәнін қабылдамайынша бірінен кейін бірі орындала береді.

2 Мысал. **EPS** дәлдіктен. $\sum_{i=1}^{\infty} 1/\,Sqr(i)$

қатарын **REPEAT** цикл көмегімен есептеудің бағдарламасы:

```
Program Summa;
Const Eps=0.01;
Var
  I : Integer;
```

```
    S, R : Real;
Begin      S : =0. 0;     I  : =0;
    Repeat
        I :=I+1;
        R : =1/Sqr(I);
        S :=S+R;
    Until R<Eps;
     Wrireln (‘Қатардың қосындысы =’ ,S);
End.
```

4.7. **While цикл операторы.**
While - ***әзірше*** **, ДО –** ***орындау*** мағанасын береді.

While цикл операторы мынандай форматта жазылады:

While ***ШарттыӨрнек*** **ДО**

Begin

{ операторлар}

end;

WHILE циклі операторының құрамында операторлардың (бұл құрамдас оператор болуы мүмкін) қайталанып орындалуын басқаратын ***ШарттыӨрнек*** болады.

WHILE циклінде алдымен шарт тексеріледі яғни **While** сөзінен кейінгі ***ШарттыӨрнек*** мәні есептеледі. Егер оның мәні **TRUE** болса, онда **Begin , end** арасындағы операторлар орындалады. Кері жағдайда циклдің орындалуы тоқтатылады.

Оператордың қайталанып орындалуын қамтамасыз ететін ***ШарттыӨрнектің*** типі бульдік болуы керек. Оның мәні ішкі оператор орындалғанға дейін есептеледі. Ішкі оператор ***ШарттыӨрнек*** **TRUE** мәнін қабылдап тұрғанша қайталанып орындала береді. Егер өрнек басынан бастап **FALSE** мәнін

қабылдаса, онда **WHILE** цикл операторының ішінде тұрған оператор бірде – бір рет орындалмайды.

1 Мысал. 1-ден 10-ға джейінгі сандарды қосу бағдарламасын жазайық:

```
S:=0; I:=1;
While I<=10 do { 1-ден 10-ға дейінгі сандар қосындысын табамыз}
    Begin
        S:= S + I;
        I:= I+1; { циклді басқаратын айнымалы мәнін өзгерту}
    end;
```

WHILE ***циклі операторымен жұмыс істегенде төмендегі жағдайларды ескеру қажет:***

- While цикл операторының қайталау саны бағдарлам орындалуы кезінде анықталады.
- While сөзінен соң циклді жалғастырып орындауға мүмкіндік беретін шарт жазылады.
- Шарт - бұл логикалық типті айнымалы: қарапайым ара қатынас өрнек және күрделі ара қатынасөрнек, олар true false мәндерін қабылдауы мүмкін.
- While циклінің дұрыс аяқталуы үшін цикл денесінде цикілді орындау шартына әсер ететін оператор болуы керек.
- Циклді басқаруда succ немесе pred, inc немесе dec функцияларын қолдану маңызды .

1. Мысал:

```
Var I: integer;
Begin
  I:=0;
  While I<10 do
     Begin
```

```
        I:= succ(I); { немесе inc(I); немесе I:=I+1`; }
        write (I, ‘ ‘ );
    end;
  Writeln;
  I:=10;
While I > 0 do
    Begin
       write (I, ‘ ‘ );
        I:= pred(I); { немесе dec(I); немесе I:=I-1`; }
  End;  End.
```

Нәтижесінде: 1 2 3 4 5 6 7 8 9 10

10 9 8 7 6 5 4 3 2 1 жолдары экранда көрінеді.

For цикл операторы .

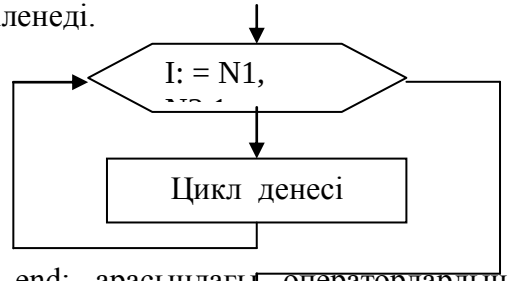
FOR цикл операторы бағдарламаның циклдік бөлігінің қайталанып орындалу саны алдын ала белгілі болған жағдайда қолданылады. Оны есептеуші немесе параметрлі цикл операторы деп те атайды. Онда ***Айнымалы Параметр*** негізгі роль атқарады. Ол циклдің әр бір қадамында өзінің мәнін бір бірлікке өзгертеді, сондықтан оны ***есептеуіш*** деп татайды. Цикл операторы құрамдас оператор болуы да мүмкін. **For** операторын екі тәсілмен орындауға болады.

1- нұсқа.(Есептеуіші артатын цикл):

For Есептеуіш:= *БастапқыМән* **To** *СоңғыМән* **Do**

```
    Begin
    { Операторлар }
    end;
```

For – “үшін”, **do –** “ орында” деген мағына береді. For ... do циклдің атуы, Do –дан кейінгі операторлар цикл денесі деп аталады. Егер цикл денесі бірғана операторлардан тұрса Begin ... end; құрамы жазылмайды. Бұлок – схемеда параметрлі цикл былай белгіленеді.



Begin ... end; арасындағы операторлардың орындалу саны [(*СоңғыМән -БастапқыМән)+1*} өрнегіне байланысты болады. Егер *БастапқыМән* > *СоңғыМән* болса цикл орындалмайды.

Мысалы:

For I:=10 to 14 do write (I: 3);

Экранда 10 11 1 2 13 14 цифрларын көрсетеді.

2-нұсқа. Есептеуіші кемитін цикл.

For Есептеуіш:= *БастапқыМән* **Downto** *СоңғыМән* **Do**

Begin

{ Операторлар }

end;

Мұндағы циклдің орындалу саны [(*БастапқыМән) – (СоңғыМән) + 1*} өрнегіне байланысты болады. Егер *БастапқыМән* < *СоңғыМән* болса цикл бірде бір рет орындалмайды.

Циклдің орындалу мысалы:

For I:=14 to 10 **Downto** write (I: 3);

Экранда 14 13 1 2 11 10 цифрларын көрсетеді.

Егер Айнымалы есептеуіш символдық типті - **char** болса онда оператор экранда әріптер тізбегін шығарады.

Мысалы: for ch:= ‘f’ **to ‘ e’ do** write (ch: 2);

экранда : a, b, c, d, e әріптері шығарылады.

for ch:= ‘e’ **Downto ‘ a’ do** write (ch: 2);

экранда : e, d , c, b, a әріптері шығарылады.

FOR операторы орындала бастаған кезде бастапқы және соңғы мәндер бір рет анықталады және бұл мәндер **FOR** операторының орындалу барында сақталады. **Басқарушы айнымалының** типі реттік болуы керек.

FOR циклін бағдарламадағы деректерді шығару үшін қолдануға болады. If, reaadln операторларын қосымша қолдану арқылы жеке –жеке беттер түрінде деректер шығаруға болады . Мысалы , төмендегі бағдарламаны орындағанда экран беті цифрлар мен толғаннан соң бағдарлама < ENTER> пернесін басқанға дейін тоқтатылады.

```
For I:=1 to 50 do
  Begin
    Writeln(I);
    If I mod 24=23 then readln;
End;
```

Цикл операторын қолданғанда төменгі жағдайларды ескеру керек:

- For орператоры циклдің орындалу саны алдын ала белгілі болған және бағдарламада анықталған жағдайда қолданылады;
- Циклдің қайталану саны Айнымалы есептеуіштің бастапқа және соңғы мәндерініе байланысты болады;
- Айнымалы есептеуіш реттелген типті: көпшілік жағдайда - integer, кейбір жағдайларда ө char, Boolean типті болуы мүмкін. Нақты типтерді қолдануға болмайды;
- Параметрдің бастапқа және соңғы мәндері бір типке жататын константтар, айнымалылар, өрнектер бола алады;

- For цикілінің параметріәрбір орындалу кезінде бір –ге ғана өзгере алады, ал өзгеру қадамын өзгерту үшін while, repeat цикдерін қолдану керек.

2. Мысал, *цикл қадамын 2 –ге өзгеру бағдарламасы:*

```
For I:=1 to 10 do
    Begin
      Writeln(I:4);I:=I+1
    End;
```

Экранда 1 3 5 7 9 сандарын көреміз.

- Циклден мезгілінен бұрын шығу үшін goto, break операторларын қолдануға болады. *Continue* процедурасы кез келген циклді тоқтатып, басқаруды аталуына беріп қйтадан орындата алады. Циклден ерте шығу мысалы:

```
For I:=1 to 45 do
    Begin
      F:=F+ I;
I (f>100) or  (i=39); then break;
    End;
```

3. Мысал. 100 сан берілген,оның қосындысын есептеп шығару керек.

$S=\sum x_i$ мұндағы i = 1,100

Егер $S_0=0$, онда $S_1= S_0 + X = X_1$ $S_2 = S_1 + X_2$

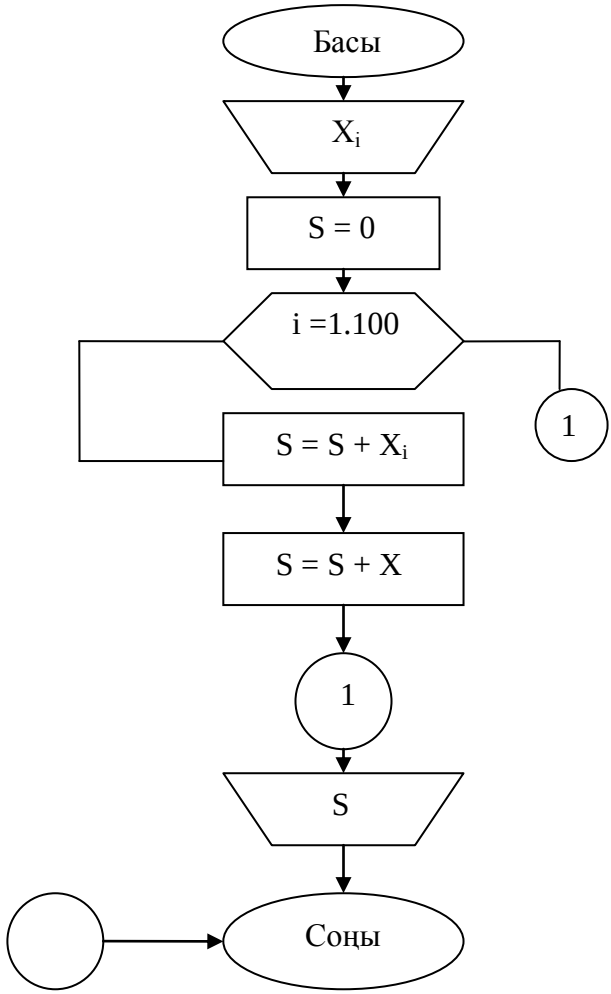
$S_3 = S_2 + X_3$ $S_{100} = S_{99} + X_{100}$

Рекурентті формуланы қосынды үшін мына түрде жазады:

$S = S + X_1$

Берілген есептердің қосу операциясы 100 рет қайталанады, яғни циклдік процесс жүреді.

Алгоритм қосындысын мына түрде жазуға болады.



Есеп шығару мысалдары

1. Мысал. $ax^2 + bx+c=0$ түрінде берілген квадрат теңдеуді шешейік. Теңдеудің түбірлері

$x_{1.2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ формуласымен $b^2 - 4ac>0$ болған жағдайда анықталады. Ал кері жағдайда

$e = -\frac{b}{2a}$ және $f = \frac{\sqrt{|b^2 - 4ac|}}{2a}$ анықталады. Нақты шешімді табу үшін $b^2 - 4ac>0$ шартының орындалуы жеткілікті. $a=0$ жағдайда теңдеудің түбірі $x = -\frac{c}{b}$ формуласымен анықталады.

```
{* Квадрат теңдеуді шешу бағдарламасы*}

RPOGRAM CORNI (INPUT, OUTPUT);
      LABEL 20;
      VAR A,B,C,D,E,F,X,X1, X2, Z: REAL;
BEGIN
 READ (A, B, C);
 IF A = 0 THEN BEGIN
                      X: = B / C;
                       WRITELN (X)
                       GOTO 20
                  END
             ELSE BEGIN
                           D: = B*  B-4. 0 * A*C;
                           Z:= 2.0*A;
                           E: = - B/Z;
                           F: = SQRT (X1,X2)
                      END;
           IF D> = 0 THEN BEGIN
                                X1: = E + F;
                                X2: = E – F;
                                 WRITELN (X1, X2)
```

```
                END
            ELSE WRITELN (E,F);
        20:
        END.
```

2 мысал. Берілген квадрантқа жататын нүктенің квадранттар номері бойынша орнын анықтау.
{* CASE операторын қолдану*}

```
PROGRAM KVADR (INPUT, OUTPUT);
VAR N: INTEGER;
BEGIN
      READ (N);
      CASE N OF
      1: WRITELN (‘ КООРДИНАТАЛАРДЫҢ МӘНІ X ЖӘНЕ
Y>0’);
      2: WRITELN (‘КООРДИНАТАЛАРДЫҢ МӘНІ X<0 ЖӘНЕ
Y>=0’);
      3: WRITELN (‘КООРДИНАТАЛАРДЫҢ МӘНІ X<0 ЖӘНЕ
Y< 0’);
      4: WRITELN (‘КООРДИНАТАЛАРДЫҢ МӘНІ X> = 0
ЖӘНЕ Y<0’)
   END
END.
```

3 Мысал. $y = \dfrac{x^3 - 4x + 1}{|x| + 1}$ функциясының x –тің мәні XN нен XK –ға дейін HX қадаммен өзгергендегі мәндерін анықтау.
{* шартты көшу операторын қолдану*}

```
RPOGRAM FUNYI (INPUT, OUTPUT);
    LABEL 50;
    VAR X, Y, XN, XK, HX: REAL;
     BEGIN
```

```
            READ (XN, XK, HX);
            X:= XN;
          50: Y: = (X*X*X-4*X+1) / (ABS (X) + 1);
              WRITELN (X: 4: 2,  ‘ ‘ , Y);
              X : = X + HX;
              IF X< = XK THEN GOTO 50
END.
```

4. Мысал: FOR I: = m_1 TO m_2 DO s;

Немесе

FOR I: m_1 DOWNTO m_2 DO s; цикл операторын қолданып

n = $\left\lceil \frac{x_k - x_h}{h_x} \right\rceil + 1$ өрнегін есептеу.

{* өрнекті есептеу*}

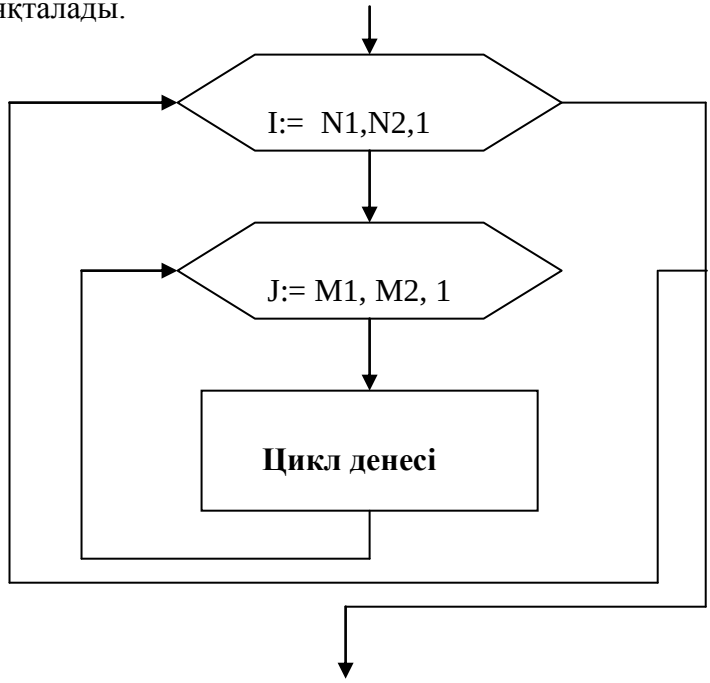
```
PROGRAM FUNYD (INPUT, OUTPUT);
      VAR N, I: INTEGER;
            X, Y, XN, XK, HX: REAL;
       BEGIN
            READ (XN, XK, HX);
             N: TRUNC ((XK – XN )/ HX) +1;
          X: XN;
          FOR I: = 1 TO N DO
           BEGIN
          Y: = (X*X*X-4*X+1) / (ABS (X) + 1);
           WRITELN (X:4:2,  :4,Y);
  END
END.
```

4.10. Ішкі циклдер

Цикл операторларының ішінде циклдің басқа операторлары да кездесі мүмкін бұл жағдайда ішінде цикл орналасқан циклді ішкі циклдер деп атайды. Ішкі циклдерді бағдарламалау кезінде ішкі циклдің барлық операторлары толығымен сыртқы циклдің денесінде орналасуы керек. Көбейтк кестесін экран бетінде шығару бағдарламасын қарастырайық:

```
Var i, j : byte;
Begin
  For i:= 1 to 10 do
     For j:= 1 to 10 do
        Writeln(i, '*', j, ' = ', i*j);
End.
```

Блок-схемада ішкі циклдің орындалу тәртібі көрсетілген: Ішкі цикл сыртқы циклдің бір мәнінде бір рет толық орындалып шығады, онан соң сыртқы циклдің келесі мәнінде қайтадан орындалады. Осы тәртіппен ішкі циклдің орындалуы сырқы цикл мәндері аяқталғанша жалғасады да ең соңында сыртқы цикл аяқталады.



Экранда Пифагор кестесін жасау бағдарламасы:

```
Var
 I , j : integer;  { жол және баған номерлері}
   Begin
      For i:= 1 to 10 do;
                Begin
                    J:= 1 to 10 do  write (i*j :5);
                    Writeln;
                 End;
End.
```

5. 1. Процедуралар мен функциялар

Процедуралар деп – негізгі бағдарлама кезінде шақырып алып қолдануға болатын бағдарламаларды айтады.

Ондай бағдарламалар процедура және функция болып екіге бөлінеді. Оларды кейде жалпы атпен қосалқы бағдарламалар деп те, атайды.

Процедура –бағдарламада белгілі бір қайталанатан іс әрекеттерді атауы бойынша бөлек жазып, қайтадан атауы арқылы бір немесе бірнеше рет іске қосуға болатын бағдарламаның бөлігі. Қосалқы бағдарламаны қолдану арқылы негізгі бағдарламанырды ықшамдауға болады. Turbo Pascal бағдарламалау тілінде қосалқы программаның 2 түрі бар:

Процедура мен Функцияны Turbo Pascal тілінде арнайы бағдарлама ретінде жеке жазуға болады, сондықтан оны көмекші(қосалқы) бағдарламалар деп атайды. Мысалы:

1. **Circle** – шеңбер салу процедурасы.
2. **Rectangle** – ағымдағы түс және түрі бойынша төртбұрышты бейнелейді.
3. **Win_color** әртүрлі көлем бойынша берілген түстегі терезелерді бейнелеу бағдарламасын құруда қолданылады.
4. **line To** – ағымдағы орыннан бастап берілген координатаға дейін түзу сызу процедурасы.

Процедура кәдімгі негізгі бағдаралама сияқты атынан хабарлау бөліміне және операторлар тобынан тұрады. Процедура құрылымы мынандай түрде жазылады.

Procedure Аты (параметрлер)
Хабарлама бөлімі
Begin операторлар бөлімі
End.

Операторлар бөлімінде процедура орналасатын болса **End** операторынан кейін «**;**» қойылады. Процедура аты бойынша шақырылады. Процедураның параметрлері арқылы деректер процедурадан бағдарламаға немесе бағдарламадан процедураға беріледі. Процедура құрамындағы параметрді формальді параметр деп атайды. Ол өзінің типімен бірге көрсетіледі. Ал оған сәйкес келетін нақты параметр типсіз жазылады. Формальді параметр мен нақты параметрдің сандары сәйкес болуы керек. Егер формальді параметрлер бірнешеу болып типтері бірдей болса, параметрлер үтірлер арқылы бөліп жазып, онан соң типін көрсетуге болады. Процедураны мынандай тәсілмен шақыруға болады.

Summa (5;M;7); процедуралық параметрлер **мәндік** параметрлер және **айнымалы** параметрлер болып бөлінеді.

Мәндік параметрлер процедураның кіріс параметрлерінің ролын атқарады да олар нақты параметрлердің мәндерін қабылдай алады. Ол өз мәндерін параметрлерге бере аламайды.

Айнымалы параметрлер әрі кіріс әрі шығыс параметрдің ролын атқарады. Олар нақты параметрлерде бөліп көрсету үшін формальді параметрлердің алдына VAR сөзді қолданылады.

```
Procedure Express (A,B,C: Real; var X,Y: Real);
Var Z: Real;
Begin
   Z:=A+B+C;
    X:=Sqr (Z);
    Y:=Sqrt (Z);
End;
```

Бұл процедураны мынандай атпен шақырып алуға болады.
Express (25.4, 44.6, 30, X_1 Y_1) бұл процедурада A,B,C айнымалылар көрсетілген мәнді қабылдайды да $\mathbf{X_1}$, $\mathbf{Y_1}$ нақты параметрлері тұрақтылар айнымалылар және өрнектер түрінде болуы мүмкін.

Есеп. $\sum_{i=1}^{\infty} 1/sgr(i)$

```
PROGRAM SUMMA;
CONST EPS =0.01;
VAR
I: INTEGER;
S,R; REAL;
BEGIN
S:=0.0;
I:=1;
WHILE R> EPS
BEGIN
S:=S+R;
I:=I+1;
R:=1/Sqr (I);
WRITELN (‘ҚАТАРДЫҢ ҚОСНДЫСЫ=’,S:10:3);
END;
END
END.
```

5.2. ФУНКЦИЯЛАР. Қосымша бағдарламалардың 2-ші түрі функциялар деп аталады. Функцияларды аты бойынша қажетті уақытында шақырып алып қолдануға болады. Сондықтан әртүрлі мақсаттарда қолданылатын функциялар алдын ала жасалынып белгілі бір атпен сақталынып қойылады. Функцияның процедурамен салыстырғандағы ерекшелігі көптеген кіріс параметрлері болса да оны орындаудың бірақ нәтижесі болады. Нәтиже функцияның атымен аталады да сол аты бойынша негізгі бағдарламаны білдіреді.

Function Аты (формальді параметрлер) типі;

Сипаттау бөлімі
Begin
.....
End.
Функция нақты параметрлерін көрсету арқылы аты бойынша шақырылады. Функция өрнектер ішінде де шақырып алуға болады. Мыс. N,K сандарының факториалдарын айнымасын табуға арналған функцияны жазу факториал деп атайды.
N!; 5! 1*2*3*4*5=5!
```
Function Fact (N:integer): integer;
Var P,I : integer;
Begin
  P:=1;
For I:=2 to ndo
P:=P*I;
Fact :=P;
End.
```
Мұндағы, ***Fact*** функцияның аты, функция нәтижесінің типі нақты сан болады. Нақты және формальді параметр ***n*** -бүтін типті. Функция N,K нақты параметрлері атымен бағдарламада шақырылып алынады. P=fact (n)-fact(k) хабарланған атаулар негізгі бағдарламаны операторлар бөліміне және процедурамен функция бөлімдерінде әсер етеді. Сондықтан ондай атаулар глобальді деп аталады.
Есеп.
```
Program f1 (I,O);
Var F, M,K:integer;
Function Fact (N:integer): integer;
Var P,I : integer;
Begin
  P:=1;
For I:=2 to ndo
P:=P*I;
Fact :=P;
End;
Begin
```

```
Write (‘M,K мәнін ендір’);
Readln (M,K);
F:=Fact (M)-fact (k);
Writeln (‘F=’, F:5);
End.
```

5.3. CRT МОДУЛІ. ЭКРАНДЫ БАСҚАРУ ПРОЦЕДУРАЛАРЫ МЕН ФУНКЦИЯЛАРЫ.

Күрделі бағдарламалармен жұмыс істеген кезде Turbo Pascal тілінде жазылған бағдарлама көбінесе мәтіндік режимді қолданады.

CRT МОДУЛІ мәтіндік режимде экран бетінде әр түрлі операцияларды орындауға мүмкіндік беріп, басқарып отырады. Ол курсорды экран бетінде қозғалтуға, экранның және символдың (яғни экран бетіндегі белгілердің) түсін өзгертуге, сол экранда терезе құруға, дыбыс шығарып басқаруға және т.б. әрекеттерді орындауға мүмкіндік береді. Оның негізгі міндеті – әр түрлі терезелерді құру немесе жаңарту, басқа бір диалогтық бағдарламаның атрибуттарын және менюін құру.

Мәтінді режимде ең аз бейнелену бірліктері тек қана пиксел (нүкте) емес, символ толығымен жатады, кей – кезде оны орын белгісі деп те атайды. Әр символ бір пикселден тұрады, бірақ әр символдың сурет салу поцессін өзіне операциялық жүйе алады.

Мәтінді режимде бейне жадының көрінісі келесі әдістермен сақталады. Әр орын белгісі үшін 2 байттан бөлінеді. Бірінші байт символдың кодын сақтайды, ол осы берілеген орын белгісінен орын алады, ал екінші – былай айтқанда символдың атрибуттық түсі. Атрибуттық түстің байты өзіне символдың түсін (төрт кіші бит), фонның түсін (үш үлкен бит), тағы арнайы жарықтылық битін кіргізеді. Олар көбінесе тек қана хабарлама жіберуде қолданылады, ол өзіне қолданушының назарын аудару үшін керек. Әр биттің мазмұны атрибуттық түстің байты 1 суретінде көрсетілген.

Бұл бірлік биттегі берілген түстің анықтамасын береді. Суреттен көрініп тұрғандай, барлық түс үш маңызды түстен құралатындығын: көк, жасыл және қызыл көреміз.

Мәтіндік режимде символдар үшін 16 түсті қолдануға болады (әр түрлі төрт түстің қосылысы 16 әр түрлі комбинация 0000 ден 1111 ге дейін береді). Олар 0 ден 15 ке дейінгі цифрлармен немесе модул **crt** табылған константармен беріледі. Ең соңғы сегіз түсі алдыңғыларынан түстің ашықтылығымен ерекшеленеді (интенсивностью). Мысалы, сары – бұл ашық қоңыр. Фонның түсіне үш бит бөлінгендіктен, оған тек қана 8 түс ендіруге болады.

Жарықтылық биті	**Қызыл**	**Жасыл**	**Көк**	**Жарық бейне көрсететін бит**	**Қызыл**	**Жасыл**	**Көк**
Фонның түсі				**Символдың түсі**			

1. сурет

Мәтінді режим 2 маңызды парамаетрлерімен сипатталады: жолдағы максималды символдар санымен және экран бетіндегі жол санымен. Бұл параметрлердің стандартты берілгені – 25 жол әр қайсысында 80 символдан. Бірақ textmode процедурасының көмегімен басқа да элементтерді енгізуге болады:

Textmode (mode);

mode – word типінің бүтін константасы (табл. 8.2).

Мысалы:

- Textmode (BW40); немесе Textmode (0); - ақ – қара активизация мәтінінің режимі 25 жолдан 40 символдан тұрады;
- Textmode (259); немесе Textmode (CO80+Font8x8); - түстік активизация мәтіндік режим 50 жолдан 80 символдан тұрады.

1. Кесте. Түстің константасы

Түсі	Константаның аты	Константаның мәні
Қара	Black	0
Көк	Blue	1
Жасыл	Green	2
Бирюзовый	Cyan	3
Қызыл	Red	4
Күлгін	Magenta	5
Қоңыр	Brown	6
Ашық – сұр	Light Gray	7
Қанық – сұр	Dark Gray	8
Ашық – көгілдір	Light Blue	9
Ашық – жасыл	Light	10
Ашық – бирюзовый	Light Cyan	11
Ашық – қызыл	Light Red	12
Ашық – күлгін	Light Magenta	13
Сары	Yellow	14
Ақ	White	15

2.

Мәтінді режим

Mode константа-сының аты	Mode константа-сының мәні	Жолдың саны, символдар саны	Адаптердің типі	Шығудың түрі
BW40	0	25x40	Түрлі түсті	Ақ - қара
CO40	1	25x40	Түрлі түсті Түрлі түсті	Түрлі түсті
BW80	2	25x80	Түрлі түсті Моно	Ақ - қара
CO80	3	25x80	Түрлі түсті	Түрлі түсті
MONO	7	25x80		Ақ - қара
Font8x8	+256	50 строк		Түрлі түсті

Екі координат арқылы толық экран мәтінінің орын белгісін анықтауға болады, сонымен қатар бұл сол жақ бұрыштық жоғарғы экранның координатасы (1.1). X координатасы жол символының

позициясы болады, ал У координатасы жолдың номерін, жоғарыдан төменге дейін бағыттағыш деп саналады.
Turbo Pascal мәтінді информациясын экран бетіне шығарған кезде ол әр символдың атрибуттарын басқаруға мүмкіндік береді. Ол үшін модул crt стандартты процедурасы қолданылады (3 кесте).

3 Кесте Түсті басқару процедурасы

Процедура	Іс әрекеті
Low Video	Экран бетіне шығарылатын анық символдарды анықтайды.
High Video	Ең үлкен жарық қиылысын анықтау рижимі.
Text Back Ground (Color)	Түстің түрін көрсетеді. (егер,аудан түсі,шығаратын символды айналдырады.). Color – 0 ден 7-ге дейін диапозонды анықтау, сегіз константың біреусі түсі болады, crt модуль арқылы анықтайды.
Text Color (Color)	Символдардың шығару түсін анықтайды. Мұнда Color – бүтін типтің анықтамасы,0 ден 15 ке дейін диапазонда орналасқан, ол crt модуліне сәйкес бір константболады.

Textattr ауыспалысын қолданып, бір уақытта барлық символдардың атрибуттық түсін тапсыру өте ыңғайлы. Айырманың мазмұнын 16-лық константтүрінде берген дұрыс, бұл барлық атрибуттардың байтын көрсетеді. Бірінші 16-лық сан көріністің түсіне сәйкес, ал екінші сан шығарылған символдың түріне сәйкес.
Мысалы:
Textattr: = $1E; {көк көріністе сары символдар}
Textattr: = 7; {қара көріністе ақ символдар}
Ең соңғы жағдайда мәнді қара түске 0 (Black) түсті константа, сол үшін 16 лық $07 мәннің орнына ондық мәні 7 тура келеді.
1 бағдарлама мазмұны жай бағдарламадан тұрады, бұл істің түсін орнатуды қарастырады және экранды тазалау, айқындама курсорынан тұрады.

1. Бағдарлама : Экранды басқару функциялары

```
Program K1;
Uses CRT;
Begin
textcolor (lightcyan); {немесе textcolor(13) немесе textcolor($D)}
textbackground (green); {немесе textbackground (2)}
{осылардың барлығының орнына мынаны жазуға болады
textattr:=$2D}
clrscr; {экран бетін жасыл түспен тазалау}
gotoxy (35, 13); writeln ('Сәлеметсізбе!'); readln;
end.
```

5.3.1. ТЕРЕЗЕМЕН ЖҰМЫС ІСТЕУ.

Терезе – бұл экранның шекаралы тік бұрышты ауданы, бұл да толық экран сияқты жұмыс істейді. Терезені анықтау үшін мына процедура қолданылады.

Window (x1, y1, x2, y2);

Экран бетіне жаңа активті мәтінді терезені анықтайды. Терезені анықтау кезінде x1, y1 координаты сол жақ бұрыш терезесі, ал x2, y2 төменгі оң жақ бұрыш терезесі болады, сол жақ жоғарғы экранның координаттары жоқ, ал минимальдік терезенің өлшемі 1 жол және 1 қатардан тұрады.

Назар аударыңыз - x1, y1 немесе x2, y2 параметрлерін дұрыс емес берген жағдайда (мысалы, x1=85 немесе y2=30) Window процедурасы алып тасталынады. Бұл жағдайда қате туралы хабарлама шықпайды.

Білуге тиіс:

- Анықтама процедурасы таңдалғаннан кейін терезеде ешқандай өзгеріс болмайды, сондықтан өзгерісті процедура орындалғаннан кейін білесің.
- Толық бет емес сол жақ жоғарғы бұрыш арқылы активтік терезе сәйкес координаттар арқылы анықталады.
- Барлық шешім тек активтілік терезесі арқылы орындалады.

4. Берілген кесте бойынша функция мен процедуралар терезеге керекті болып есептеледі. Ескере кету керек, егер терезе Window процедура көмекшісі арқылы анықталмаса, онда терезе толық

экран, Window (1, 1, 80, 25); процедурасын шақыруына сәйкес болады. Мұндай шақырылым активтілік терезені алып тастауға қолданылады.

4 кестесі процедура және функциядан тұрады, активтілік терезені басқаруға мүмкіндік береді (толық экран, егер терезе анықталмаса).

4. Кесте. Активті терезеге шығуды басқаратын процедуралар мен функциялар.

Процедура және функция	Іс әрекеті
ClrScr	Активті терезені тазалайды және курсорды сол жақ бұрышқа орналастырады.
ClrEo1	Активті терезенің жолын курсордың бағыты бойынша барлығын соңына дейін тазалайды.
GotoXY (x,y)	Курсорды X, Y координаталарына орналастырады.
WhereX	X координатасын алдыңғы қалпына әкеліп қоятын функция.
WhereY	У координатасын алдыңғы қалпына әкеліп қоятын функция.

2 бағдарлама мазмұнында келтірілген бағдарлама экран бетінде 7 жолды (терезе), әр түрлі көріністің түсіне сәйкес келеді (қарасыз). Осы әдіспен экран бетінде, түрлі түсті үшбұрыштардан тұратын оюларды көрсетуге болады.

2. Бағдарлама: Түрлі түсті көрініс және window процедурасының қызметін демонстрациялау

```
Program K2;
uses CRT;
Var k, x: integer;
begin
   textbackground (0);
   clrscr;
   x:=2;
```

```
   for k:=1 to 7 do begin
   textbackground (k);
   window (x, 1, x+11, 25); clrscr;
   x:=x+11;
   end; readln
end.
```

5.3.2. БАҒДАРЛАМАНЫ ОРЫНДАҒАН КЕЗДЕ ТОҚТАП ҚАЛУЫ. Кейде компьютердің жұмысын жасанды түрде тоқтату керек. Ол үшін CRT модулінің тағы бір процедурасы қолданылады.

Delay (time); Time – тоқтап қалу уақыты миллисекундада.
Бұл процедура көбінесе экран бетіне әр түрлі өзгерістерді шығаруда қолданылады. Crt модуліндегі атақты қателерге осы процедура қатысты болады.
Мысалдағы (3. Бағдарлама мазмұны) delay процедурасы экранның түрлі түсті жұлдызшалармен толып кетпеуін жайлап тоқтатады. Delay процедурасы басқа мысалдардағы программа көрінісі төмендегідей көрсетіледі.

3. Бағдарлама: Түрлі түсті жұлдызшалардан спираль суреті н салу

```
Program K3;
uses CRT;
type direction = (right, down, left, up);
Var c, x, y, h, k:integer;
procedure go (d: direction; n: integer);
begin
   for k:=1 to n do begin
     case d of
       right: x:=x+2;
       down:  y:=y+1;
       left:  x:=x-2;
       up:    y:=y-1;
     end;
       c:=random (15)+1; textcolor (c);
```

```
      gotoxy (x,y); write ('*'); delay (30);
  end;
end;
 begin
    clrscr; x:=39; y:=13; h:=1;
    gotoxy (x,y); write ('*');
    repeat
      go (right, h); go (down, h); h:=h+1;
      go (left, h);  go (up, h);   h:=h+1;
  until h>24;
end.
```

. ПЕРНЕ ТАҚТАНЫ БАСҚАРУ.

Стандартты перне тақта үш түрлі пернесі болады:

- Символдық (әріп, сан);
- Басқарушы (функционалдық перне, қою, алып тастау, т.с.с.)
- <Ctrl>, <Alt>, <Shift>, <Numlock>, <Capslock> және т.б.

Кез келген бір пернені басқан кезде код пайда болады, ол код сканирование (скан - код) деп аталады. Әр бір пернеде өз код сканированиясы болады, оның өзінің перне тақтада өз орны болады және де ол пернедегі суретке байланысты болмайды. Скан – код пернелердің сандық қатары болады (мысалы, <Esc> пернесі сол жақ жоғарғы бұрышта орналасқан және оның номері 01).

Бірақ бағдарламаға басылған перненің сандық қатары керегі жоқ, керісінше осы пернеде көрсетілген код символы – ASCII коды керек. Бұл код скан – кодқа байланысты болмайды, өйткені бір пернеде бірнеше символдар бола алады. Мысалы, <1> символы бар перне тағы <Shift>пернесімен бірге басылса <!> мынандай белгі береді.

ASCII кодында скан – кодты қайта құруда операциялық жүйе орындалады, қайта құруды орындайды және алынған ASCII – кодын оперативті жадтың *перне тақтаның буфері* деп аталатын жеріне орналастырады. Сол жерден оны Turbo Pascal бағдарламасы немесе бағдарламаның басқа да тілдері ала алады.

I.5. ДЫБЫСТЫ БАСҚАРУ.

Бұны орындау үшін екі процедура ғана қажет:

- Sound (Hz) – ішкі динамикті қосатын процедура. Hz динамиктен шығатын сигналдың мөлшерін герц есебімен береді. nosound процедурасы өшірілмегенше дыбыстық сигнал істеп тұрады (егер сіз бұл процедураның шақырылуын бағдарламада жазуға ұмытып кетсеңіз, бағдарлама жұмысын бітірсе де дыбыс өшірілмейді);
- Nosound – ішкі динамикті өшіретін процедура.

Мысалы:

Sound (5000); delay (100); nosound;

5. және 6. Бағдарлама мазмұнында келтірілген бағдарлама Turbo Pascal дағы дыбысты басқарудың мүмкіндіктерін бейнелейді.

6. Бағдарлама: Гамманы бірінші, екінші және үшінші октавада *шығару*

```
Program K5;
uses CRT;
{бірінші октавадағы нотаға сәйкес келетін жиілік, келесі октаваға жиілік екі еселенеді}
const octava_1: array [1..7] of integer=(262, 294, 330, 349, 392, 440, 493);
Var n, octava, koef: integer;
begin
  koef:=1;
    for octava:=1 to 3 do {үш октава}
    begin
      for n:=1 to 7 do {әр октавада 7 нотадан}
      begin
```

```
        sound (octava_1[n]*koef);
        delay (300); nosound;
      end;
       koef:=koef*2; {келесі октаваға өтеміз}
    end;
  sound (octava_1[1]*koef); delay (300); nosound; {төртінші октаваға дейін}
end.
```

7. Бағдарлама мазмұны. @ символын экран бетінде дыбыспен қозғау

```
Program K7;
uses CRT;
const
{перненің коды – символды басқаруға мүмкіндік беретін бағыттаушы сызық}
 left=#75;right=#77; up=#72;down=#80;
 ch ='@';  {орын ауыстыруға арналған символ}
Var c:char;  curx, cury: byte; badkey: boolean;
procedure beep (badkey: boolean);
begin
   if badkey then {қатенің сигналы}
   begin
      sound (500); delay (100);
      sound (440); delay (200);
      nosound;
```

```
    end
      else
begin {0.02 секундта шертпе қозғалысының ұзындығы 360 Гц}
        sound (360); delay (20); nosound;
      end;
   end;
    begin
      clrscr; curx:=40; cury:=12;
      repeat
      gotoxy (curx,cury); write (ch);
      badkey:= false; c:= readkey;
      if c=#0 then {кеңейтілген код?} c:= readkey;
       case c of
          #27 :;  {esc – ештеңе істемейді}
          right: if curx < 80 then inc (curx);
          left:  if curx > 1 then dec (curx);
          up:    if cury > 1 then dec (cury);
          down:  if cury < 25 then inc (cury);
       else badkey:= true;
       end;
       beep (badkey);
       until c=#27
       end.
```

6. МАССИВТЕР

Массив бағдарламада қолданылатын ең қажетті түсінік. Массивте бірдей типке жататын шамалар, информациялар,

сақталып өңделеді. Арнайы бірдей типті шамаларды сақтау тәсілін массивтер деп атайды. Массив бірдей типті элементтерден тұрады да, ол элементтер ортақ индетификатор арқылы таңбаланады. Индекс бойынша қажетті элементті табуға болады.

Массив элементтерінің номерлері ***индекстер*** деп, ал массивтің элементтері ***индекстелген айнымалылар*** деп аталады.

Массивтегі деректер бағдарлама аяқталғанша ғана сақталады. Ал бағдарлама аяқталғаннан соң бағдарламаны ұзақ сақтау үшін дискіге жазу керек.

Массивтің негізгі сипаттамалары:

. **типі -** барлық массив элементтерінің типі ортақ ;

. **өлшемі (ранг)** – массив индекстерінің саны;

. **индекстердің өзгеру диапозоны –** массивтегі элементтер санын анықтайды.

Вектор (бір өлшемді массив) – элементтері индекстермен номерленген массивтің мысалы бола алады. Егер массивте кесте сақталатын болса , оны ***екі өлшемді массив*** – ***матрица*** деп атайды. Оның элементтері екі индекспен номерленеді. Екіден жоғары өлшемде массивтер практикада сирек кездеседі.

Компьютер жадында массивтің барлық элементтері белгілі бір үздіксіз аймақта ғана орналасады. Осыдан массив сөзі шыққан. Массив элементінің индексі ретінде ***реттелген типті*** өрнек қолданылады. Олар бүтін, константнемесе integer типті айнымалы, сирек жағдайларда char, boolean типті болуы мүмкін. Массив элементін шақыру үшін индекстер массив атауынан соң тік жақша [] ішінде жазылады. Мысалы, a [3], c[2,3] . Массивтерді өңдеу оның элементтерінің индексін өзгерту арқылы орындалады. Мысалы, циклде массив элементтерін іздеу үшін төмендегіөрнектерді қолдануға болады:

. **a**[i] - барлық массив элементтері;

. **a**[2*i] - жұп орындардағы массив элементтері**;**

. **a**[2*I-1] - тақ орындардағы массив элементтері**.**

Массивтерді сипаттау.

Бағдарлама деректер массивімен жұмыс істеуі үшін, ең алдымен массивтерге жадтан орын бөлінуі

қажет. Массивтің әр элементі үшін жадтан жеке ұяшықтар бөлінеді. Ол ұяшықтардың барлығының өлшемдері, массивте сақталатын мәліметтердің типтеріне қарай бірдей өлшемде болады.
Массиатерді сипаттау барлық айнымалылар сияқты, Var сипаттау бөлімінде жазылады.
Айнымалылар типі – массив элементін қабылдайтын мәннің типі.
Массив атауы барлық элеметтер үшін бірыңғай болады. Көпшілік жағдайда массивтің төменгі индекісі ретінде 1 саны қолданылады.

Массивтің жоғарғы және төменгі индекстерінің айырмасына 1-дің қосындысы осы индекс бойынша, массив элеметтерінің максималы мүмкін санын анықтайды.
Массив элеметтері, кезкелген типті (real, integer, char,...), сонымен қатар құрамына типтерді (массив, жазба,....) қабылдай алады.

Берілген массивке бөлінген жад көлемі- бір элементке типіне байланысты бөлінген орының массив элеметтерінің жалпы санына көбейтіндісіне тең, яғни максималды мүмкін элеметтерінің, барлық индекстерінің өзара көбейтіндісімен анықталады.

Массивтер саны бекітілген бір типтес компонеттерден тұрады. **Массивке төмендегі қасиеттер тән:**

- Массивтің әр компоненті айқын түрде белгілене алады және оған тікелей қатынас жасауға болады.
- Массив элементтерінің саны оны сипаттаған кезде анықталып, одан соң өзгермейді.

Массив элементтерін белгілеу үшін айнымалы массивтің аты және қажетті элементке сілтеп тұрған индекс пайдаланылады. Индекстің типі реттелген болу керек.

Массив элемнеттеріне қатынасу үшін, массивтің идентификаторын жақшаға алынған бір немесе бірнеше индекспен көрсету керек.

Индекстік өрнектер массивтің өлшеміне сәйкес элемнеттерді белгілейді. Өрнектің саны массивті хабарлағандағы индекстік типінің санынан артық болмауы керек.

Сондай- ақ әр өрнектің типі сәйкес индекстік типпен меншіктеу бойынша сәйкес болуы керек.

Массивтерді VAR бөлімінде хабарлау.

Массивтерді VAR бөлімінде хабарлау ***array*** (массив) қызметші сөзін қолдау арқылы орындалады.

Оның жазылу форматы :

Бір өлшемді массив үшін: **var** ***МассивАты*** : **array** [*ТөменгіШекара.. ЖоғарғыШекара*]

Of ; *ЭлементтерТипі;*

Мысалы:

Var a: array [1..100] of integer; { бүтін 100 элементтен тұратын массив}

b: : array [1..50] of char; { 50 элементтен тұратын символ}

c: array [-3.. 4] of boolean { логикалық мәнді8 элемент тұратын массив}.

Екі өлшемд массивтің жазылу фарматы:

a:array[ТөменгіШекара индекісі1, ЖоғарғыШекара индекісі1]

[Төменгі шекара индекісі2. Жоғарғы шекара индекісі2] of : элеметтер типі.

Екі өлшемді массив кестелік **мәндер (матрица)** деп аталады. Ол екі индекспен белгіленіп, оның біреуі жолдың, екіншісі бағанның нөмерін көрсетеді. Компьютер жадында массивтің бар элеметтері үздік бір аймақты алып жатады. Осыған байланысты массив ұғымы шыққан. Екі өлшемді массив компьютер жадында жол бойынша алдымен бір жолдың элеметтері,

одан соң екінші жолдың элементтері ретпен орындалады. Массив элеметтерінің индекстері ретінде, реттелген типті өрнектер алынады. Осыған көпшілік жағдайда индекс ретінде Integer типті тұрақты немесе айнымалысы ал сирек жағдайда Char немесе Boolean типтері орындалады.

Екі өлшемді массивті (матрица) енгізу:

```
For i:=1 to n do
For j:=1 to m do
Readln(a[i,j]);
```

Массив түрлері.

Бір өлшемді массив (вектор). Массив элементі оның позициясына (орнына) қарай анықталады. Қарапайым тілмен - бұл санау массивтің басынан басталады. Бір өлшемді массивті мысалы ретінде сынып журналдағы оқушылар тізімін алуға болады. Мысалы:

1	Абдуғожа Е
2	Тілеуов М
3	Билібаева М.
4	Қазбекова М.
5	Сыдық А

Оқушылардың фамилялары ол массив элементі, ал нөмірлері осы элементтің массивтегі тұрған орнын білдіреді. Берілген жағдайда массивке “Оқушы” деген ат беруге болады. Сонда бірінші элемент мәні “Оқушы [1]” - “ Абдуғожа Е ”, ал ал төртінші элемент мәні “Оқушы [4]” - “ Қазбекова М. ” болады.

Бірақ элементінің тұрған позициясы (орны) координаттар жүйесінде бірнеше өлшемді болуы мүмкін, яғни элемент бірнеше индекстермен анықталады. Мысалы: A(4,3,8). Элементті анықтайтын

индекстер саны массив өлшемі немесе айқындалатындай ранг деп атадады.

Екі өлшемді массив (матрица). Екі өлшемді массив ретінде шахмат тақтасын қарастыруға болады, шаршылардың орналасуын жолдар мен бағандардың мәндері білдіреді, ал элеметтерінң мәні сол шаршыдағы фигураның бар жоғымен анықталады.

Үш өлшемді масив, n -өлшемді массив.
Бір және екі өлшемді массивтерден басқа үш өлшемді массив және n өлшемді массивтерде кездеседі. Егер үш өлшемді массивті үш өлшемді массивті жазықтықта бейнелеуге болса, ал n өлшемді массивті жазықтықта бейнелеу қиынрақ. Ол біздің үш өлшемді кеңістікте өмір сүріп жүргенімізге байланысты. Осыған қарамастан көп өлшемді массивтер ғылыми және эканомикалық есептерде жиі қолданылады.

Екі өлшемді массив үшін:
var ***МассивАты*** : **array** [*ТөменгіШекараИндекс1*.. *ЖоғарғыШекараИндекс1*]
[*ТөменгіШекараИндекс2*.. *ЖоғарғыШекараИндекс2*]
Of ; *ЭлементтерТипі;*

Мысалы, компьютер жадында 1 2 3 4
5 6 7 8
9 10 11 12
әрбір саны бүтін типті - integer болатын кесте орналассын. Кесте элементтерінадрестеу үшін екі индекс – жол номерлері мен баған номерлері қажет. 1- Индекс 1-3 дейінгі, 2- Индекс – 1 –ден 4- ке дейін өзгеруі керек. Индекстің төменгі шекарасы жоғарғы шекарадан "**:**" таңбасы арқылы ажыратылып жазылады.Төменгі шекара индекстің ең аз мәнін , ал жоғарғы шекара индекстің ең үлкен мәнін анықтайды. Төменгі шекара жоғарғыдан артпауы тиіс. Жоғарыдағы кестені массивте мына түрде көрсетеміз: **var y: array [1..3, 1..4] of integer;**

Мысалы, латын алфавиті әріптерінің берілген сөйлемде қанша рет қайталанатынын анықтау бағдарламасы:
Var count: array [‘a’..’z’] of integer;
Массивті хабарлау үшін компиляторға жадтан орын бөлуге тура келеді. Мысалы, 100 элементті нақты тип үшін жадтан 100 ұяшық 6 байттан барлығы 600 байт бөлінеді. 20 жол , 30 бағаннан тұратын 600 элементі бар , элементтері 255-тен аспайтын бүтін оң сан болатын екі өлшемді ма ссивті 600 байт ұяшыққа орналастыруға болады.

Trubo Pascal бағдарламалау ортасында айнымалылар үшін жадта 64Кбайт орын бөлінген . бұл күрделі бағдарламалар құруды қиындатады. Сондықтан одан құтылу үшін арнайы әдістер қолданады. Мысалы: Var huge : array [1..105, 1 .. 105] of real; бағдарламасы

Error 22: Structure too large (22 –қате,құрылым өте үлкен) қатесін тудырады. Жадты үнемдеу үшін byte, shortint типтерін қолдану тимді.

Бағдарламаның денесінде массивті атауы арқылы пайдаланады. Массивтерді баяндаудың екі түрі бар:

а) алдымен массив типі, типтер тарауында анықталады:

TYPE tm = ARRAY[T1] OF T2

мұнда tm – массив типінің идентификаторы; T1 – индекстің типі; T2 – элементтерінің типі; сонан кейін, tm массив типі,VAR – тарауында айнымалыларды (мысалы; «а» аталуы) анықтауға қолданылады:

VAR a:tm;

ә) массив типті айнымалыларды тікелей VAR – тарауында анықтауғада болады:

VAR a:ARRAY[T1] OF T2;

Массивтерді баяндаудың мысалдарын қарастырайық:

```
CONST n=10;  m=20;
TYPE масс1 = ARRAY[1..10] OF REAL;
       масс2= ARRAY[1..n, 1..m] OF BOOLEAN;
 VAR a,b,c: масс1;
       d,e: масс2;
```

Бірөлшемді (вектор) массивтер, индекстік типтер саны бірге тең болғанда құрылады:

```
TYPE =ай(қаңтар, ақпан, наурыз, көкек, мамыр, маусым, шілде,
тамыз, қыркүйек, қазан, қараша, желтоқсан);
VAR ай_күндері :ARRAY[ай] OF 28..31;
ай_аты:ай ;
BEGIN
  FOR ай_аты:=қаңтар TO желтоқсан DO
 CASE ай_аты OF
   Қаңтар, наурыз, мамыр, шілде, тамыз, қазан, желтоқсан:
ай_күндері[ай_аты]:=31; көкек,маусым, қыркүйек, қараша:
ай_күндері[ай_аты]:=30; ақпан:
ай_күндері[ай_аты]:=28;
END;        {CASE нұсқауының соңы}
END;        {BEGIN нұсқауының соңы}
```

Бұл бағдарлама сынығында бір жылдағы 12 ай аттарын «ай» типімен белгілеп, «ай-аты» (типі – «ай») айнымалысымен «ай-күндері» атауы – массив (негізгі типі шектелген сан жиыны) анықталған. FOR қайталану нұсқауымен «ай-аты» алғашқы мәні қаңтардан бастап желтоқсанға дейін түгенделетін «ай» типінің мәндерін біртіндеп қабылдайды; цикл денесіндегі таңдау нұсқауы (CASE) әр ай_атына байланысты ай_күндері массивіне тиісті күндер санын меншіктейді (мысалы: ай_күндері[қаңтар] :=31;..., ай_күндері[маусым] :=30;), CASE нұсқауының өрнегі(таңдау индексі) «ай_аты» айнымалысымен анықталып, таңдау константаларының үш тобы сұрыптау нұсқауларының қайсысы қолданылатынын айқындайды. 12 элементтен тұратын «ай_күндері» массивінің мәні: 31,30 немесе 28 сандарын қабылдайды.

Массивтің индекстік типтер саны екіге тең болса – екі өлшемді массив (матрица) деп аталады:

```
 TYPE қатар = ARRAY[1..8] OF CHAR;
VAR Тақта = ARRAY[1..8] OF қатар;
немесе бірден – «тақтаны» VAR тарауында:
  VAR Тақта = ARRAY[1..8,1..8] OF CHAR;
```

«Тақта» массивінің орта шенін жұлдызшаға “*” тең екенін белгілеу керек болса:

тақта[4,4] := “*”;

6. 2. Массивтермен орындалатын әрекеттер.

1 . Мысал. Матрицаның оң элементтерін экранға шығару. Оң элементі жоқ жолдарды бос қалдыру.

```
PROGRAM MATR (INPUT, OUTPUT);
      CONST  IMAX = 10; JMAX = 15;
       VAR I, J: INTEGER ;
             A : ARRAY [1…IMAX, 1…JMAX] OF REAL;
BEGIN
        FOR I: = 1 TO  IMA X  DO
            BEGIN
        FOR J: = 1 TO  JMA X  DO
            BEGIN
             READ (A [I, J]);
              IF A [I, J]> 0  THEN WRITE (A[I, J]:5 :2)
END;
  WRITELN
    END
END.
```

2. Мысал . $X(x_1, x_2, x_3 \ldots x_{100})$ массивінің оң элементтерінен Y массивін, теріс элементтерінен Z массивін құрастыру.

```
{* Оң және теріс массив элементтерін құру*}
ROGRAM SORT (INPUT, OUTPUT);
        CONST NMAX = 10;
         VAR I, N, K: INTEGER;
               X, Y, Z: ARRAY [1..NMAX] OF REAL;
BEGIN
    FOR I: = 1        TO NMAX DO
          READ (X[I]);
          K: = 0;
          N: = 0;
FOR I: = 1  TO NMAX DO
          IF X [I] > 0 THE
```

```
 BEGIN
          K: K + 1;
         Y [K]: = X [I]
END
ELSE
BEGIN
     N : = N + 1
      Z [N] : = X [I]
END;
       FOR I: = 1        TO K DO WRITE (Y [I]: 6:2);
       FOR I: =1         TO N DO WRITE (Z[I]: 6:2)
END.
```

7. Жиындар

Жиындар деп - бірдей типке жататын элементтер тобын айтады. Жиын элементтері жеке – жеке нөмірленбеген оны жеке табу мүмкін емес. Сондықтан жиындарды элементтерінің орналасқан реті міндетті емес болған жағдайда қолданылады. Мысалы: дауысты дыбыстың немесе дауыссыз дыбыстар жиыны, шахмат фигуралары мен жүрістер жиыны т.б жатады.

Жиын элементінің типі – жиынның базалық типі деп аталады. Жиын элементі типінің мәндері мүмкін болатын базалық типтің ішкі жиындарынан тұрады.

Turbo Pascal – да базалық типке шектеу қойылған. Онда реттелген тип үшін мәндерінің саны 256 – дан аспауы керек. Жай типтерде (char, byte, Boolean) саналмаған типпен диапазондарды қолдануға рұқсат етілмейді. Олардың элементтерінің саны да 256 – дан аспауы керек. Жиын элементтері мәнін «[ ]»қолдану арқылы жазылады.

Мысалы: [1,2,3,4]
[‘a’, ‘b’, ‘c’] ;

Егержиын элементі жоқ болса онда оны бос жиын деп атайды да [ ] тік жақшаны пайдаланады. Оның ішіне ешқандай символ жэазылмайды. Жиын түріндегі типті хабарлау үшін Set of – сөз тіркесі қолданылады. Оның жазылуы мына түрде болады:

Type Типтің аты = Set of жиын элементі мәнінің типі
Var Айнымалының Аты, ...: Типтің аты;

Жиын айнымалыларының типін көрсетпей – ақ қысқа түрде жазуға болады:
Var Айнымалының Аты, ...: Set of жиын элементі мәнінің типі
Егер тұрақты жиын типтері берілсе, ол мына түрде болады:
Const = [жиын элементтері];
Егер типтелген Const болса, ол мына түрде болады:
Const КонстАты: жиын элемент типі = [жиын мәнінің типі];

Жиындармен орындалатын операциялар

Жиындармен барлық салыстыру >,<, = біріктіру ‘ +’ ,қиып алу ‘ * ’, жиындар айырмасын табу ‘ - ’, және элементтерінің жиынға жататынын анықтау (in) операциялары орындалады.

Теңдік операциясы (=) – егер а және в жиындары бірдей элементтен тұрған жағдайда орындалады.
Мысалы: A [1,2,3,4]; A[1…4];
B [4,3,2,1]; B[4,3,2,1];

Тең емес (< >) – а және в жиындардағы элемент саны бірдей емес бол,ан кезде қолданылады.
Мысалы: A [1…4]; A< >B
B [4,3];
Үлкен немесе тең – бір жиынның 2 – ші жиын ішінде жататындығын анықтау үшін қолданылады да, егер ол сол жиында жататын болса
Мысалы: A > B true

Кіші немесе тең - алдыңғыға сәйкес
Мысалы: A< B true
‘in’ операциясы
Мысалы: 2 in A[1…4] бұл операция шартты операторларда қолданылады.

Жиындарды біріктіру (+) – нәтижесінде жаңа жиын пайда болады.

Мысалы: A[1,2]
B [3,4]
C [1,2,3,4]

Жиындардың қиылысуы (*) - нәтижесінде үшінші жиын пайда болады

Мысалы: A [1,2]
B [3,4]
C [3,4] – қиылысады.

Жиынның айырмасы (-) – нәтижесінде

Мысалы: A – B = C [1,2] true

Мысалы: Натурал санның алғашқы жүзінен барлық жай сандарды табу.

```
Program Nat;
Const N = 100;
Type Set Of 1…N;
Var
 N1, N2: Integer;
 S1, Pr: Set;
Begin
 S1:= [2… N];
 Pr := [1];
 N2 := 2;
While S1 < > [ ] do
Begin
 S1 := S1 – [ N1];
 N1 := N1 + N2;
End;
 Pr := Pr  + [N2];
Repeat
 N2 := N2 + 1;
Until (N2 IN S1) or (N2>N);
End;
 For N1 : = 1 to N Do
```

```
    If N1 IN PR then
 WRITE (N1….4);
Writeln;
End .
```

8. ЖАЗБАЛАР (RECORD)

Жазба, массив тәрізді, мәліметтердің күрделі типін құрастыруға – бір заттың (нәрсенің) әртүрлі қасиеті мәліметтерін біріктіріп қолдануға пайдаланылады. Жазбаның массивтерден айырмашылығы:

- жазбаны құрастырушы элементтер (өрістер) біріңғай типке жатпауы да мүкін;
- жазба, құрастырушы элементтерінің (өрістерінің) атауларымен ғана (өріс идентификаторлары) тікелей анықталады.

ТҰРАҚТЫ ӨРІСТІ ЖАЗБАЛАР

Бағдарлама құрылымында, жазба типі (RECORD) – массив типі секілді сипатталады:

```
TYPE za_id = RECORD
     id11, id12, ...,idN1: Tun_1;
     id21, id22, ..., idN2:Tun _2;
     idK1, idK2, ..., idKNk: Tun_K;
END;                {жазба типінің соңы}
```

Мұнда: TYPE, RECORD, END – Паскаль тілінің түйінді сөздері;

id11, id12, …, idKNk - өрістердің (элементтерінің) идентификаторлары;

Тип_1, Тип _2,..., Тип_K - өрістердің типтері;

za_id – жазба атауы.

Мысал: Паскаль тілінде комплекстік сандар үшін стандартты тип белгіленбеген. Бағдарлама құрылымында комплекстік сандарды қолдану үшін жазба типін (RECORD) пайдаланған ыңғайлы болады (жазба типі нақты тип (REAL) өрістен тұрады):

```
   TYPE Comp1 = RECORD
        Re, Im:REAL;
END;
```

Сонда VAR тарауында жаңа - Comp1 типіне жататын X,Y айнымалыларын анықтауға болады:

VAR X,Y:Comp1;

Бұл мысалда: Comp1 – жазба атауы; Re, Im өрістердің идентификаторлары X,Y - Comp1 типті айнымалылар (жазба).

Мысал: Уақыт жазбасы, шектелген типпен анықталған үш өрістен (күні, айы, жылы) құрылған болса:

```
TYPE Уақыт= RECORD
        күн:1..31;
        ай:1..12;
        жыл:1900..1995
END;
VAR D: Уақыт;
```

Типі – D айнымалысы (жазба), болған уақиғаның: күнін, айын, жылын біріктіріп қолдануға оңтайлы.

Жазба айнымалысының өрісін бағдарлама денесінде пайдаланғанда жазба айнымалысының идентификаторы мен нүктемен бөліктелген өрістің идентификаторы көрсетіледі, мысалы:

```
X.RE:=2.5;        X.TM:=3.4;
D.күн:=16;              D.жыл :=1984;
```

Паскаль тілінде – жазба (RECORD) типін басқадай күрделі типтерді ұйымдастыру үшін пайдалануға болады. Мысалы, жазба типіне сәйес массивті былай анықтауға болады:

```
TYPE Семья = (әке, шеше, бала1, бала2);
VAR Туысқан:ARRAY [Семья] OF Уақыт;
```

Мұнда: Уақыт жазбасы – жоғарыда баяндалған жазба типі; Туысқан массиві (бірөлшемді -4 элементті), семья мүшелерінің туған уақытын белгілеуге ыңғайлы, жазбаларынан тұрады. Бірінші баланың туған уақытын анықтау үшін, келесі нұсқаулар пайдаланылады:

```
Туысқан[бала1].күн:=18;        {1-ші бала 18 күні туған}
Туысқан[бала1].ай:=4;          {туған айы 4-ші (апрель)}
Туысқан[бала1].жыл:=1982;      {туған жылы 1982}
```

Жазба өрісінің типін – RECORD типін баяндауда тікелей анықтауға немесе бұрынырақ баяндалған типтің атауымен көрсетуге болады.

Мысал: Бес группаның (топтың) қысқы сынақ қорытындысына (Rezultat) келесі мәліметтер кіруге тисті:

- сынақ болған пәннің аты Disc (төрт пәннің аттары);
- топтағы 15 студенттердің фамилиясы (Name);
- әр студенттің сынақтан алған бағасы (Baga).

Қорытындыға (Rezultat) кіруге тиісті мәліметтердің құрамы массивтен құралған, оның әр элементі жазба типімен, екі өрістен (Disc, Stud), анықталған. Екінші Stud өрісі массив түрінде анықталып, онында әр элементі жазба типімен (Name, Baga - өрістерінен) баяндалған:

```
    TYPE
 Rezultat=ARRAY[1..4] OF RECORD
            Disc(Matem, Progr, Phizic,Graf);
            Stud:ARRAY[1..15] OF RECORD
  Name:ARRAY[1..20] OF CHAR;
    Baga:2..5
END                    {Stud жазбасының соңы}
END;                   {Rezultat жазбасының соңы}
VAR
       Exam:Rezultat;
```

Айнымалылар тарауындағы Exam айнымалысы Rezultat типімен анықталады.

WITH НҰСҚАУЫ

Жазбаның өрістерін бағдарламада қолдануы үшін құрамды атты пайдаланады, мысалы:

```
Exam[2].Disc:=Graf;          Exam[2].Stud[13].Baga :=5;
```

Exam массивінің 2-ші элементінің Stud өрісті массивтің 13-ші жазба элементін Baga өрісінің мәні 5-ке тең. Жазба типімен анықталған айнымалыларды қолдану үшін, жазба құрамына сәйкес өрістердің атаулары толық түгенделуі керек. Құрамды аттардың жазуын ықшамдау үшін (жазбаның ішкі өрістерін

қолданылуын жеңілдету үшін) - WITH нұсқауын қолданған қолайлы. Оның жалпы түрі

WITH A DO S;

Мұнда: A – жазба (RECORD) типті айнымалының атауы;

WITH, DO – Паскаль тілінің түйінді сөзі; S – паскаль тілінің нұсқауы.

WITH нұсқауын қолданғаннан кейін, S – нұсқауының денесінде жазбаның өрістерін қолданғанда, өріс идентификаторларының алдында идентификаторы жазылмайды, мысалы: жоғарыда келтірілген мысалдарды, WITH нұсқауы, келесі түрде ықшамдап жазуға мүмкіндік береді:

```
  WITH туысқан [бала1] DO
BEGIN
   күн:=18;
   ай:=4;
   жыл:=1982;
END;
WITH Exam[2] DO
  BEGIN
  Disc:=Graf;
Stud[13].Baga:=S;
END;
```

9. ЖОЛ ТҮРІНДЕГІ - STRING- ТИПІ ЖӘНЕ ОҒАН ҚОЛДАНЫЛАТЫН СТАНДАРТТЫ ФУНКЦИЯЛАР

Turbo – Pascal жүйесінде, алдын ала ұзындығы анықталмаған жолдар үшін STRING типі пайдаланылады. Егерде, Паскаль тілінің негізгі нұсқасында символдардан тұратын массивтердің ұзындығы, алдын ала белгіленген болса, мысалы:

```
  Var Avator:ARRAY[1..18] OF CHAR;
STRING
VAR
        Fam:STRING[20]; {Fam:ARRAY[1..20] OF CHAR;}
        Adr:STRING[25];  {Adr:ARRAY[1..25] OF CHAR;}
        Maman:STRING;   {Maman:ARRAY[1..?] OF CHAR;}
```

Мысалда көрсетілген, Fam айнымалысы-ұзындығы 20 символға дейінгі, ал Adr айнымалысы – 25 символға дейінгі, таңбалардан тұратын айнымалыларды анықтауға пайдаланылады. Ал, Maman айнымалысының ұзындығы – 255 символға дейін жетуі мүмкін.

Анықтауға болады:

Fam:= ‘Құнанбай-ұлы’; Maman = ‘Ақын’;

Adr:= ‘Семей’;

Бағдарлама құрылымында, бұл типке қарасты айнымалылар, READLN процедурасы арқылы ғана енгізілуі керек, мысалға:

…

READLN (Fam);

…

STRING типтегі айнымалыларға келесі стандартты функциялар анықталған:

1. LENGTH (SRTOKA);

Бұл функцияның мәні – бүтін сан, STRING айнымалысының нақты ұзындығына тең. Мысалы, FAM айнымалысының мәні - ‘Құнанбай-ұлы’ болған жағдайда: к:= LENGTH(Fam); к – айнымалысы, Fam тіркесінің ұзындығына, 12-ге тең.

2. CONCAT (STR1, STR2, …,STRN);

Функция, жақша ішіндегі айнымалыларды бір – біріне қосақтау (тіркестіру) үшін қажет (функция мәні - STRING (тіркес)). Мысалы:

Stroka:=CONCAT(‘көк’,’шолақ’);

Stroka айнымалысы, ‘көкшолақ’ мәнін қабылдайды.

3. POS (STR1, STR2);

Бұл функцияның мәні – бүтін сан, STR1 жолының STR2 жолындағы кездесетін – позициясына тең. Егер, STR1 жолы STR2 жолының ішінде жатпайтын болса, функция мәні – нольге тең болады. Мысалы:

k:=POS(‘шол’, ‘көкшолақ’); {k=4}

n:=POS(‘дәу’, ‘ұзын’); {n=0}

k – айнымалысы 4-ке тең, өйткені STR2 (‘көкшолақ’-та), STR1 (‘шол’) жолы, 4-ші таңбадан бастап кездеседі. Ал, n – айнымалысының мәні 0-ге тең себеі: «Дәу» жолы, «ұзын» жолының ішінде кездеспейді.

1. COPY (STR, I, N);

COPY – функциясы, берілген Str жолының I-ші позициясынан басталатын,ұзындығы N –ге тең символдар тіркесін анықтауға қолданылады, мысалы:

s:=COPY(‘Көкшолақ’,4,3); {s= ‘шол’}

p:=COPY(‘Ақтаңлақ’,3,3); {p= ‘таң’}

COPY – функциясының мәні – STRING (тіркесі).

2. DELETE (Str, I, N);

DELETE – функциясы, берілген Str жолының I-ші позициясынан бастап, ұзындығы N –ге тең, тіркесі «өшіріледі».Мысалы:

s:=DELETE (‘Бесбармақ’,4,3); {s= ‘Басмақ’}

DELETE – функциясының мәні – STRING (тіркес).

3. INSERT (Str1, Str2, T);

Бұл функция, Str1 жолын Str2 жолының I-ші позициясынан бастап орналастыруға пайдаланылады, мысалы:

жаңа: = INSERT(‘қала’,’айда’,3); {жаңа= ‘айқалада’}

INSERT– функциясының мәні – STRING (тіркес).

4. STR (Stroka,N);

STR – функциясы бүтін сан типіне жататын N – айнымалысын Stroka тіркесіне (сандар символына) түрлендіреді:

STR(San, 175); {San=’175’}

String типіндегі San айнымалысы ’175’ тіркесіне тең болады.

Келесі бағдарлама, String типін қолданып, берілген файлдағы 20-сөздің қайсысы және қаншасы «в» әріпінен аяқталатынын анықтайды:

```
Program Buckba_”B”;
    Const
 n=20;              {Сөздердің саны, 20-ға тең}
 Var
      Slova:Array[1..n] Of String;      {Сөздер массиві}
```

```
    k, i, l: Integer;          {к – «в»-ға аяқталатын сөздер саны}
  res, dat: Text;    {Берілген деректермен, нәтижеге}
                     {арналған текстік файлдар}
Begin                                    {бағдарлама денесі}

Assign(res, 'C:\TP55\DAN4.TXT);
Assign(dat, 'C:\TP55\DAN4.TXT);
Reset(dat);                              {Деректер файлы - dan4.dat}
RewriteE(res);                           {Нәтиже файлы – dan4.txt}
Writeln(res, '':10, 'Берілген сөздер');
For i:=1 TO n DO
  Readln(dat, Slova[i] );
            {Деректерді (сөздер массивін), файлдан енгізу}
For i:=1 TO (n Div 2) DO
   Writeln(res, Slova[i]:20, '--', Slova[(i+(n Div2))]);
            {Сөздерді екі қатармен, файлға шығару}
Writeln(res);
Writeln(res, ' «в» - әріпіне аяқталатын сөздер:');
K:=0;                  {Басында 0-ге тең деп санаймыз}
   For i:=1 TO n DO
   Begin
            {Циклде, «в» -ге аяқталатын сөздерді іздейміз}
I:=Length(Slova[i]);
            {функция, кезекті сөздің ұзындығын табады}
    If Copy(Slova[i],l,1)= 'B' then
Begin
            {«в» -ға аяқталған сөздерді, нәтиже файлына жазу }
K:=k+1;              {олардың сандарын санау}
Writeln(res, k:3, ;: Slova[i]);
End;
          {If Copy Slova…нұсқауының соңы}
End;              {For – циклінің соңы}
Writeln(res);
Writeln(res, ' "B" - әріпіне аяқталатын сөздер саны =' ,k);
Close(res);                          {Нәтиже файлын жабу}
                          {Buckba_ "B" – бағдарламасының соңы}
```

Buckba_ “B” – бағдарламасында, STRING типті сөздер массивіне (Slova), Length пен Copy функциялары қолданылған. Барлық нұсқаулардың жұмыс істеу тәртібі түсініктемелер арқылы баяндалған.

10. Файлдар. Файлдармен орындалатын операциялар

1. Файлдардың түрлері

Осыған дейін, мәліметтерді пернетақтадан енгізіп, оларды экранға шығардық, бағдарлама орындалуы біткеннен соң ол мәліметтер сақталмай жоғалып кететін. Жұмыс нәтежесін компьютердің оперативті жадында сақтап, есептеулер біткеннен кейін қайтадан пайдалана алу үшін, мысалы, оларды есептеулерге пайдалану немесе жаңа мәліметтеді алу үшін, оларды есте сақтап қалу керек. Қазіргі кезде дербес компютерлердің барлығы ақпаратты сақтауға арналған арнайы құрылғылармен жабдықталған. Қызыметтеріне қарай оларды: магниттік ақпарат тасымалдаушылар немесе жазылатын CD-дискілері деп атайды. Ақпаратты сақтау осы тасымалдаушыларда ұйымдастырылады, бұл жағадайда біз алған нәтежиелерді солардың жадында сақтаймыз. Барлық мәліметтер осы тасымалдаушыларда файлдар түрінде сақталады.

Файл дегеніміз- мәліметтерді сақтауға арналған сыртқы тасымалдаушыдағы жадының ат қойылған аймағы (бөлігі). **File** (ағылшын тілінен аударғанда) бума, іс қағаздар жыйнағы сонымен қатар ақпаратты сақтау деген мағынаны білдіреді. Әр түрлі формада берілген деректер файлдарда сақталады. Ол бағдарлама, құжат, бейне және дыбыс түрінде болуы мүмкін.

Файл деп дискідегі немесе сыртқы еске сақтау құралындағы белгілі атпен аталатын жад аймағын айтады. Дискілік операциялық жүйесінде файл 2 түрге бөлінеді:

1) мәтіндік
2) екілік файлдар.

Мәтіндік файлдар адамдардың оқу үшін жазылады да, онда бағдарламалардың мәтіндері және Dos операциялық жүйесінің

командалық файлдары сақталады. Мәтіндік файлға жатпайтын файлдары екілік файлдар (кодтар) деп атайды. Dos жүйесінде және Pascalдағы файлдар физикалық файылдар деп атайды. Деректерді ендіру мақсатында файлдарды қолдануға болады. Ол үлкен кәсіптік бағдарламалар жасауда кең түрде қолданылады.

Деректерді ендіру дегеніміз – жад ұяшықтарын файлдардан алған шамалармен толтыру деген сөз. Ал шығару операциясы дегеніміз – жад аймағынан деректерді файлға беру деген сөз. Ендіру және шығару операциясын орындайтын жад аймағын буфер деп атайды. Dos операциялық жүйесінде файлдарды орналастыру кестесін FAT деп атайды. Dos жүйесінде файлдың аты мен заты латын әрпімен 8:3 форматында жазылады. Файлға қойылатын шамалар:

1) кез – келген файлды жазу кезінде өзіне тән ат қойылады.
2) Файл элементтері бір ғана типтен тұрады.
3) Файл элементтерінің санын файлдың ұзындығы деп аталады.

Файл жазу кезінде оның ұзындығы алдын – ала берілмейді. Ол сыртқы есте сақтау құрамының сыйымдылығына байланысты шектеледі. Ешқандай элементі жоқ файлды бос файл деп атайды да, оның ұзындығы 0 – ге тең болады. Массивтердің файлдардан айырмашылығы массивтерде деректер бағдарлама орындалып жатқан кезде ғана сақталады да, бағдарлама тоқталған кезде жоқ болып кетеді. Ал файлдар да деректер бар уақытта сақталып тұрады. Компьютерде бір мезгілде көптеген файлдар сақталады. Қажетті файлды жылдам тауып алу үшін жеке каталогтарға бөліп сақтайды.

Каталог дегеніміз – дискінің белгілі бір атпен аталатын аймағы. Каталогтар файлдың аты мен қоса мынандай сипаттамалар сақталады: байт есебімен файлдың өлшемі, файлды жасаған немесе өзгетркен соңғы уақыт, файл атрибуттары («тек қана оқу үшін», «жасырын», «жүйелік және архивтік»), дискідегі файлдың физикалық орны.

Borland Pascal- да файлдарды екі негізгі белгісі бойынша топтастыруға болады:

А) файлдың типі бойнша (оның логикалық құрылымына)

Б) файл элементтеріне қатынау тәсілі бойынша.

Типтері бойынша файлдар шартты түрде үш түрге бөлінеді:

- ***мәтіндік файл;***
- ***типтік файл;***
- ***типтелмеген файл***.

Біз көбінесе мәтіндік және типтік файлдармен жұмыс істейміз.

Қатынау тәсілі бойынша файлдар келесідей бөлінеді:

- ***Тікелей қатынас жасау файлдары (параллель)***
- ***Кезекпен қатынас жасау файлдары()тізбектелген)..***

Олардың айырмашылығы тікелей қатынау файлдарында арнайы процедуралар және функцияларды пайдалана отырып, мәліметтерді, олардың файлдағы орнына тәуелсіз, оқуға немесе жазуға болады, ал кезекпен қатынау файлында оның ортасында немесе соңында тұрған мәліметтерді барлығын оқуға тура келеді. Мәтіндік файлдар кезекпен қатынау файлдарына жатады, ал типтік файлдар тікелей қатынау файлдарына жатады.

2. Құрылғылар.

Компьютерде жеке компоненттерін мынандай мақсаттарға байланысты бөліп көрсетуге болады: ендіру, шығару және деректерді сақтау. Бұл мақсатта қолданылатын құралдарды сыртқы немесе периферийлік. Сыртқы құрылғылармен файлдарды шатыстырмас үшін сыртқы құрылғылардағы файлдарды физикалық файлдар деп атайды. Құрылғылармен жұмыс істеу үшін қолдануға болмайды – логикалық қондырғының аты қойылады. Ондай логикалық құрылғылардың атына мынадай атаулар жатады:

1) CON – инфармация пернетақтадан ендіріліп консольдің комегімен дисплей экранына жіберіледі. Клавиатурадан символдарды ендіру кезінде ол буфер жолдарынан алынады да ENTER пернесін басу кезінде бағдарламаға беріледі. Ал Ctr + E басу кезінде файлдың аяқталғандығы туралы белгі беріледі.

2) PRN – бұл принтердің атауы егер компьютерге бірнеше принтер қосылса олардың логикалық атаулары (RPT) RPT2 тағы сол сияқты аталады.
3) COM1, COM2 – тізбектелген фортқа жалғанған құрылғылардың атауы. Олар компьютерді бір – бірімен қосуға және «тышқан» тәріздес тетікті қосуға арналған.

3. Файлдардың типтері.

Pascal тілінде файлдардың 3 типі бар:
1. мәтін түріндегі файлдар (text)
2. типтелген файлдар (file of)
3. типтелмеген файлдар (file)

Кез – келген бағдарлама файлды қолданатын болса міндетті түрде мынандай қадамдардан тұруы керек:
1. файлды ашу.
2. файлды өңдеу (оқу немесе жазу).
3. файлды жабу.

Файлдық тип типтерді хабарлау бөлімінде мына форматта жазылады:
Type ТипАты = file of БазалықТип;
Var АйнымалыАты : file of БазалықТип;

4. Файлдар мен орындалатын операциялар

Файлдардан қажетті элементті табу үшін оның алдыңғы элементтерін рет – ретімен қарап шығу керек. Бұл әдісті файлды тізбектеп оқу деп аталады. Файлды өңдеу үшін алдымен оны ашып онан соң қажетті әрекеттерді орындап соңында файлды жабуымыз керек.

Файлдармен жұмыс істеу үшін систем модель төмендегідей процедуралары мен функциялары қолданылады:

Assign (ФайлАты, СыртАты) – файлдық айнымалыны сыртқы құрылғыдағы файл мен байланыстырады.

Reset (ФайлАты) – файлды оқу үшін ашады.

Read (ФайлАты, АйнымалыАты) – файлдағы деректерді оқып, оперативтік жадқа береді.

Close (ФайлАты) – файлды жабу.

Rewrite (ФайлАты) – файлды оған қосымша деректерді жазу үшін ашады.

Write (ФайлАты, АйнымалыАты) – деректерді файлға жазады.

Eof (ФайлАты) – файлдың соңын анықтайтын логикалық функция. Оның мәні true болса, файлдың соңы анықталады.

Бұл функциялар мен процедураларда мынандай белгілеулер қолданылады:

ФайлАты – айнымалыларды хабарлау бөліміндегі файлға берілген ат.

СыртқыАты – сыртқы құрылғыдағы файлдың аты. Ол апостроф ішінде жазылады. Мысалы: d : \ Label. pas.

4. Мәтіндік файлдар.

Мәтіндік файлдар жеке жолға бөлінген символдар тізбегі ретінде жазылады. Әрбір жолдың соңында # 13, #26 символдар кездеседі.

1 – символ келесі жолға көшуді, екіншісі файлдың соңын көрсетеді. Мәтіндік файлдарда деректерді ендіру мен шығару стандарты Read және write процедуралар арқылы орындалады. Реттелген символдардың тобы автоматты түрде файл операцияларда қолданылатын айнымалылар типінің мәніне түрленеді. Мысалы: read (f, n) j файылынан ... типті айнымалыларды оқып n шамасына меншіктейді. Мұндағы n айнымалысының типі n – Bute. Мәтін түрінде файл жазу үшін Var бөлімінде text түріндегі айнымалы хабарланады:

Var АйнымалыАты : text; ол бағдарламада файлды көрсету үшін қолданылады. Онан соң файлдық айнымалы Assign процедурасы көмегімен физикалық файлмен байланысады да файл ашылады. Ондай әдіспен нақты файлға сыртқы құрылғылардығы логикалық файлдық айнымалы сәйкес келеді. Мысалы: а дискісінде мәтін түрінде қарапайым файл жазайық:

```
Var  f : text;
Begin
Assign (f, ‘a : \ B. dat’);  a – ғы  B.dat файлы.
Rewrite (f); жазу үшін файлды құру және ашу.
Writeln (f, ‘файлға беру’); файлға жаңа жол қосамыз.
Close (f); файлды жабамыз.
```

Pascalда файлдың 2 айнымалы: Input және Output қатарына жазылады. Олар CON құрылысы мен автоматты түрде байланысады да стандартты ендіру шығару қызметін атқарады. Input пернетақтамен байланысқан Output дисплей экранына байланысқан файлдық айнымалылар. Бұл файлда алдын ала ашылған және ендіру шығару операцияларын қолдануға болады. Текстік файлдар үшін, мәндерді оқуға және жазуға мүмкіндік беретін оқу және Read және Write жазу операцияларының арнайы түрі бар. Мұндай мәндер автоматты түрде символдық түрге және керісінше көшіріледі. Мысалы; Read (F,I), интерпретациясы ондық сан болатын сандар тізбегін оқиды, мұндағы F текстік файл, I бүтін типті айнымалы.

Мәтіндік (текст) типтің стандартты INPUT және OUTPUT екі айнымалысы бар. Стандартты INPUT файлдық айнымалысы – бұл операциялық жүйенің стандартты енгізу файлымен байланысқан оқуға ғана мүмкін файл (әдетте бұл пернелік), ал стандартты OUTPUT файлдық айнымалысы – бұл операциялық жүйенің стандартты шығару файлымен байланысқан жазуға ғана мүмкін файл (әдетте бұл дисплей). Бағдарлама орындалу алдында INPUT және OUTPUT файлдары автоматты түрде ашылады да, бағдарлама орындалып біткеннен кейін бұл файлдар автоматты түрде жабылады.

Read - Текстік файлдан бір немесе бірнеше мәнді бір немесе бірнеше айнымалыға оқиды.

READLN- READ жасайтын әрекеттерді орындап, келесі қатардың басына дейін өткізу жасайды.

WRITE - Текстік файлға бір немесе бірнеше мәнді жазады.

WRITELN - WRITE жасайтын әрекеттерді орындап, файлға қатар соңы маркерін қосады.

EOLN - Файл үшін ENO-OF-LINE қатар соңын анықтайды.

Төменде көрсетілген бағдарлама F1 текстi файлынан 1-ші курс студенттерінің тізімі енгізіледі және ол алфавит реті бойынша сұрыпталып, экранына шығарылады. F1 файлының компоненттері студенттің фамилиясын (20 символдан тұратын) және 4 пән бойынша (математикалық анализ, физика, тарих және алгебра) бағалары білдіреді. F1 файлына сәйкес келетін бастапқы деректер жиыны кез келген текст редакторының көмегімен дайындала алады, себебі фамилиялар, бағалар бос орынмен бөлінген символдар тізбегін білдіреді.

```
Program Fil_Text;
Const M=20;
Type
   Predmet=(Matan, Fiz, Ist, Alg);
  Ozenki = Array [Predmet] Of Integer;
Var
   F1    :Text;
  St      : Array [1..100] Of String[M];
  R       : String [M];
  Or      : Array[1..100] Of Ozenki;
  Rt      : Ozenki;
  I,N,K  : Integer;
  J       : Predmet;
  FileName :  String[20];
Begin
  If ParamCount=1 Then
17  FileName:=ParamStr(1)
  Else
     FileName:='predmet.dat';
  Assign (F1, FileName);
  Reset (F1);
  N:=0;
  While Not Eof (F1) Do
       Begin
```

```
            N:=N+1;
            Read (F1,St[N]);
            For J:=Matan To Alg Do
               Read (F1, Oz [N,J);
           Readln(F1);
End;
Close(F1);
For I:=1 To N-1 Do
   For K:=I+1 To N  Do
     If St [I]>St[K] Then
          Begin
             R:=St[K];
             St [K]:=St[I];
             St[I]:=R;
             Rr:=Oz[K];
              Oz[K]:=Oz[I];
              Oz[I]:=Rr;
        End;
  Writeln (‘1-ші курс студенттері:’);
  Ror I:=1 To N Do
       Begin
          Write (I:3, ‘ : ‘);
          For K:=1 To M Do
             Write (St[I,K]);
          For J:=Matan To Alg Do
              Write (Oz[I, J]:2);
          Writeln;
       End;
End.
```

Айнымалыларды сипаттаудың Var операторында F1 файлы текстік деп анықталған және компоненттер саны 100-ден аспайды деп жорамалданып, ST және OZ массивтері анықталған. Файл компоненттерінің нақты саны EORфункциясын пайдаланып WHILE цикілінің орындалу процесі кезінде анықталады.

16, 17 жолдарда PARAMCOUNT және PARAMSTR стандартты функциялары пайдаланылған. PARAMCOUNT функциясы

командалық жол немесе альтернативті түрде бағдарламалық ортаның RUN/PARAMETERS пункті арқылы берілетін параметрлердің жалпы санын анықтайды. PARAMSTR(K) функциясы К-шы параметр болатын жолды анықтайды. Сонымен 16-19 жолдарда FILENAME айнымалысына командалық жол арқылы берілетін сыртқы файлдың аты меншіктеледі (осы жағдайда PARAMCOUNT функциясының мәні 1-ге тең). Кері жағдайда FILENAME айнымалысына сыртқы 'predmet.dat' файлының аты меншіктеледі.

20-шы жолда F1 айнымалысы аты FILENAME айнымалысы арқылы сипатталатын сыртқы файлмен байланыстырылады. RESET (F1) функциясы F1файлын ашады және F1 файылының бірінші компонентін оқуға мүмкіндік береді.

23 –30 жолдарда тұрған цикл операторлары сыртқы файлдағы ақпараттарды ST[I] (I –ші студенттің фамилиясын алфавит реті бойынша сұрыптайды. Сұрыптаудың нәтижесі 43 –52 жолдардағы операторлардың көмегімен экранға беріледі.

Output файлдық айнымалы көмегімен файлды принтерге және экранға шығару.

Мысалы.

```
Begin
Assign (output, 'PEN');
Rewrite (output);
Writeln (0, '*принтер');
Close (output)
End.
```

Файлды экранға шығару.

```
Begin
Assign (f, ' – ');
Rewrite (f);
Writeln (f, '*экран*');
Close (f)
End.
```

Мәтіндік файлдар мен ғана жұмыс істейтін процедуралар мен функциялар.

Set Text Bug (var f : text;

Var Buf [:size : word]) f – файлы үшін алдын ала буферден байт есебімен белгілі орын бөлінеді.

Flush (var f : text) – f файлының міндетті түрде физикалық файлға жазылады.

Append (var f : text) – ашылған f файлының соңына деректер қосуға мүмкіндік береді.

EoLn (var f : text) – жолдың немесе файлдың соңын анықтайды (#13, #26)

Seak EoLn (var f : text) – бұл да жолдың соңын немесе файлдың соңын анықтайды, сонымен қатар жол соңының алдында бос орын мен табуляцияның бар жоқ екендігін тексереді. (#32, #9)

Мысалы: Компьютердің маркасы оның қатты дискісінің және оперативтік жадтың көлемі санымен қатар компьютердің жылдамдығы туралы мәліметтер жинап экранда көрсететін файлды a дискісінде жазу керек.

```
program Comp1;
 uses crt;
 type comp = record
     marka : string [15];
     hdd, ram : real;
     speed : integer;
 end;
     myfile = file of comp;
 var   f1 : myfile;
       i, n : integer;
       c1 : comp;
 begin
 clrscr;
   writeln (' Компьютер саны ');
   readln (n);
   assign (f1, ' a : \ computer ');
   rewrite (f1);
```

```
   for i := 1 to n do
begin
writeln (' маркосын ендір: ');
readln (c1. marka);
writeln ('HDD және RAM: көлемін ендір:');
readln (c1.hdd,c1.ram);
writeln ('жылдамдығын ендір:');
readln (c1.speed);
write( f1,c1);
end;
close (f1);
writeln ('марка винчестер көлемі RAM жылдамдығы:');
reset (f1);
for i := 1 to n do
begin
read (f1, c1);
write (c1.marka:15,c1.hdd:10,c1.ram:7,c1.speed:8);
writeln;
end;
readln
end.
```

5. Типтелген файлдар.

Бір типті информацияларды мысалы сан түріндегі деректерді мәтіндік файлда сақтау тиімді емес. Компьютер символдық типті 2 – і түрге және керісінше айналдыру үшін өз рессурсын көп жұмсауына тура келеді.

Типтелген файл бір типті және ұзындықтары бірдей тізбектелген элементтерден тұрады. Типтелген файлдың берілу кезінде оның өлшем шектемейді және файлдың әр бір элементінің өз номері болады. Оның бірінші элементі (0) ноль деп таңбаланады. әр бір уақыт мезетінде бір ғана ағымдағы элемент жұмыс істеуге мүмкіндік береді. Ол файл көрсеткішінде белгіленеді. Берілген элемент пен оқу және жазу жұмысы аяқталғаннан соң файл көрсеткіші автоматты түрде келесі

элементке көшіп оқу және жазу әрекетін орындайды. Файл элементтерінің ұзындығы бірдей болғандықтан әр бір элементтің орнын жылдам анықтауға болады. Сондықтан файл көрсеткіші кез – келген элементпен тікелей жұмыс істеуді қамтамасыз ете алады. Мәтіндік файлмен салыстырғанда типтік файлдың ерекшелігі ол жолдарға бөлінбейді. Ондағы деректер компьютердің оперативтік жадында 2 – к код түрінде сақталады. Мәтін түріндегі файлдан басқа барлық файлдар 2 – к код түрінде қарастырылады. 2 – к код түріндегі файлдарды Pascal редакторында тікелей көріп, мазмұнын анықтай алмаймыз. Ал оның мазмұнын анықтау үшін деректердің типтелген файлдағы жазылу форматын анықтап Pascal бағдарламасы арқылы өңдеуге болады. Типтелген файлдарды хабарлау файл File of type – файлдың айнымалы арқылы беріледі. Типтелген файлдарды хабарлау type және var бөлімінде мына түрде жазылады:

```
Type    ТипАты := file of Тип;
Var     АйнымалыАты : ТипАты;
Var     АйнымалыАты  : file of Тип;
```

6. Типтелген файлдармен жұмыс істейтін процедуралар мен конструкциялар.

Типтелген файлдарды resert және rewrite әдістерімен ашуға болады. Resert процедурасы бұрыннан бар, ал Rewrite процедурасы жаңадан файл жазып оны ашу үшін қолданылады. Типтелген файлдар мен жұмыс істеу кезінде мыналарды білуіміз керек:

1. типтелген және типтелмеген файлдарға resert және rewrite процедураларын қолдана беруге болады.
2. типтелген файлды жазу және оқу үшін тек қана write және read процедуралары қолданылады. Ал writeln, readln прцедураларын қолдануға болмайды және файл элементтерін типі бірдей болу керек.

Типтелген файлдар мен жұмыс істейтін функциялармен жұмыс істейді.

File Ros (f) – файлдағы элеметтің ағымдағы орнын көрсетіледі. Ол бүтін сонды түрінде болады. Мұндағы f файлдық айнымалы.

File Size (f) – ашылған файлдағы жазбаның нақты санын көрсетеді. Оны файлдағы байт есебімен көрсетуге болады. Егер файл бос болса 0 мән көрсетеді.

Seek (f, n) – файл көрсеткішін n орнына жылжытады.

TrunCate (f) – f файылындағы ағымдағы көрсеткішінің орнынан бастап файлдың соңына дейінгі жазбаларды өшіреді.

Мәтін түріндегі файлдың ең тиімді қасиеті - әр түрлі типті деректерді сақтау, ал типтелген файлдың ең тиімді қасиеті файл элементтеріне оқу және жазу үшін тікелей қатынас жасау болып табылады.

Мысалы:

1. құраушылары бүтін сан болатын файл жазып оның жұп орындарында тұрған элементтерін көрсететін онан соң тақ орындарда турған элементтерін көрсететін файл жазу:

```
var f: file  of integer;
I, j, n : integer;
Begin
Assign (f, ‘a : \ 10. 10. dat’);
Rewrite (f);
Randomize;
N := 1+ Randomize (10);
For i:= 1 to n * 20 do
Begin
If odd ((i - 1) div 10) then j := I else j := i;
Write (f, i);
Write (i : 4)
End;
Writeln (‘файлдағы элементтер f :’, File Size (f));
Seek (f, 1);
While Not E of (f) do
Begin
Read (f, i), if  odd (file Ros (F))
```

```
End;
Close (f);
Readln
End.
```

Мысалы:

Мәтіндік файлға төрт таңбалы 50 сан жазу керек. Әрбір сан жеке жолдарға жазылатын болса нәтежедегі файлдың өлшемі қандай болады?

Оны жуықтап бағалап көрелік. Әрбір таңба 1 байт орын алады. Ондай таңбалардың саны 50х4=200. Мәтіндік файлдарға берген анықтама бойынша. Әрбір жолдың соңында жол соңының белгісі, ал файлдың соңында файл соңы белгісі болғандықтан, файл өлшемі 250 байттан асып кетеді . Бұл мәліметтерді кіші өлшемдегі файлдарда сақтауға бола ма?

Бағдарлама мәліметтерді өңдей алу үшін, біз оған алдын ала оперативті жадында белгілі бір орын бөлуіміз керек. Жадта бөлінетін орынның өлшемі мәліметтердің типіне байланысты болғандықтан, бір элементтерге 2 байт byte типті бір элеменке 1 байт орын алады.

Мұндай амалдарды мәліметтерді файлда сақтау кезінде қолдануға мүмкіндік беретін **типтелген файл** деп аталатын арнайы файлдық тип бар.

Типтелген файл – барлық элементтері бір типті мәліметтер болып келетін файл түрі. Типтелген файл элементтері файлдық типтен кез-келген тип бола алады.

Әр элементті файлға жазу үшін, мәліметтердің типіне байланысты, міндеті белгіленген мөлшерде орын бөлінеді. Типті файлдарды сипаттау бөлімінде алу үшін, мәтіндік файлды сипаттағандай, файл атауын беру керек және берілген файлдағы мәліметтерді сипаттайтын тип жазылады.

< файл атауы>:file of <мәліметтер типі>

Мысалы :

File Out: File of integer;

Бұл жазу , берілген файлдағы мәліметтер 32768-ден 32767 аралығанда жататын бүтін сандар екенін көрсетеді.

Сонымен қатар, типті файлдармен жұмыс істегенде. **Assign()** процедурасының көмегімен нақтылы файл аты мен байланыстыратын атауды көрсетеміз.

Assign (<файл атауы>,< файлдың нақты атауы >)

Мысалы:

Assign (File Out,'may. Dat');

Типті файлдармен жұмыс жасағанда мәтіндік файлдардағы секілді оларды жазуға және оқуға ашуға болады. Ол үшін стандартты функциялар қолданады:

Rewrite (<файл атауы>)- процедурасы көмегімен файл жабылады.

Типтелген файлдармен жұмыс істейтін процедуралар мен функциялар.

Типтелген файлдар үшін форматтары келесі түрле болатын оқудың **READ** және жазудың **WRITE** процедуралары бар :

```
Read (F1,  X);
Write (F1, X);
```

мұндағы **X** айнымалысы **F1** файлының компнентіне сәйкес типте болуы керек.

Типтелген файл бағдарламаның көмегімен ғана жасала алады. Сол себепті төменде текстік файлды типтелген файлға түрлендіруге мүмкіндік беретін бағдарламаның мысалы келтірілген.

```
Proram Fil _ Type ;
Const  M = 20;
Type
    Stud = Record
       Fam: String [ M] ;
          Matan , Fiz, Ist, Aig : Integer ;
     End;
Var
  F1 : Text;
  F2 : File of  Stub;
  R : Stub;
  I :   Integer;
```

```
Begin
  Assign (F1, ‘ F1. txt’);
  Reset (1);
  Assing (F2, ‘F2.ttt’);
  Rewrite (F2);
  While Not Eof (F1) Do
     Begin
        For I : = 1To M Do
           Read ( F1, R. Fam[I]);
        Read ( F1, R. Matan);
        Read ( F1, R. Fiz);
        Read ( F1, R. Ist);
        Read ( F1, R. Alg);
        Write (F2,R);

     End;
  Close ( F1);
  Close (F2);
End.
```

PROGRAM операторында **F1** және **F2** екі файлы сипатталған **REWRITE(F2)** процедурасының көмегімен **F2** файлы жазуға ашылады. **WHILE циклінде F1** файлынан студенттер жайлы ақпарат жүйелі түрде **R** (фамилиясы және бағасы) жазуының сәйкес өрістеріне оқылады. Кейін **WRITE (F2, R)** операторының көмегімен **R** жазуы **F2** файлына жазылады. Процесс бастапқы файл үшін **END – OF - FILE** жағдайы туғанша қайталанады.

Алынған файл студенттер тізімін алфавит бойынша сұраптауға мүмкіндік беретін келесін бағдарламаның бастапқы файл ретінде пайдалана алады. Program Fil _ Tyr _ Sort;

```
Type
    Stud = Record
        Fam : Array [1..10] of Char;
        Matan, Fiz, Ist, Ald : Integer;
      End;
     Var
       F2          : File of Stud ;
```

```
        R          : Stud;
        St         : Array [1..100] of Stud;
        N, I, J    : Integer;
    Bedin
       Reset (F2);
       N : = 0;
       While not Eof (F2)  Do
           Begin
              N : = N + 1;
               Read (F2, St [N]) ;
           End ;
   For I : = 1   To N –1 Do
      For J : = I+1   To    N  Do
If St [I] . Fam > St [J] .Fam Then
     Begin
       R: = St [J] ;
       St [J] : = St [J];
       St [J] : = R ;
      End;
   Writeln (’1  - ші курстың студенттері: ’);
   For I : = 1 To N Do
    With St [I] Do
       Begin
         Write (I, ‘ : ‘ , Fam, ‘  ‘ , Matan: 2);
          Writeln (fiz: 2, Ist : 2, Alg : 2);
    End;
End.
```

11. Мәтіндік және графикалық режімдер. Графикалық модулдер

Қазіргі дербес компьютерлердің графикалық информацияларды енгізу, шығару және өңдеуге арналған техникалық құрылғылары бар. Графикалық информацияны енгізу үшін – сканер, ал шығару үшін дисплей экраны және плоттер пайдаланылады. Дисплей экраны нүктелер жиыны болып табылатын төртбұрышты аймақ

болып табылады. Ол графикалық және мәтіндік режимдерде жұмыс атқара алады. Графикалық режимде экранның әрбір нүктесін түрлі түске бояп, сол түстер арқылы сызық, мәтін және әр түрлі бейнелер кескіндеуге болады. **Турбо Паскаль** жүйесінде графикалық информациялармен жұмыс үшін **GRAPH. TPU** модулі пайдаланылады. Бұл модульде графикалық процедуралар мен функциялардың бағдарламасы машиналық тілде жазылған. Графикалық бағдарламалау процесі тиімділігін арттыру мақсатында Borland International фирмасы **GRAPH** арнайы бағдарламалар кітапханасын жасап шығарған болатын, онда қазіргі кездегі мониторлардың барлық типтерімен жұмыс істейтін, экранға түрлі мөлшерлердегі қаріптер шығара алатын драйверлер жиыны бар.

Графикалық режимнің төмендегідей сипаттамалары бар: мүмкіндігі (разрешение), палитра (бояу түрлері), фонның түсі, шығарылатын графикалық нүктенің түсі, оперативті жадыдағы графикалық экрандық беттер мөлшері.

Бұл сипаттамалар, ең алдымен дисплейдің техникалық типімен жүйеден графикалық режимге бөлінген видеожадының көлеміне тәуелді. Дисплей экранының жұмысын басқаратын **техникалық құрылғы** деп аталады. Казіргі уақытта кеңінен таралған адаптерлердің **EGA, VGA** және **SVGA** жатады.

Жалпы алғанда, компьютерде негізгі екі экран режнмінің жұмысы - символдық және графикалық экран режимдері пайдаланылады Компьютерді қосып, **Турбо Паскалъ** жүйесін шақырғанда текст режимде жұмыс істейді. Графикалық режимді алу үшін **GRAPH.TPU** модулін (Uses GRAPH - пен бірге) қосып, керекті графикалык режим **INITGRAPH** процедурасымен инициализация жасалуы қажет. Режимді инициализациялау дегеніміз - дисплей адаптерінің жұмысын берілген графикалық режимнің күйіне келтіру, яғни физикалық экранды осы режнмнің жұмысына көшіру. Осы режимді тағайындағаннан кейін GRAPH модулінің барлық командаларын пайдалануға болады. Графикалық режимнен шығу үшін **Closegraph** процедурасы пайлаланылады.

Turbo Pascal – ортасында бейне адаптер негізінен мәтіндік режимінде жұмыс істейтіндіктен сурет салу үшін графикалық режимде көшуге болады. Ол үшін InitGraph процедурасы қолданылыды. Оның жазылу форматы InitGraph (драйвер, режим, драйвер орны).

Мұндағы драйвер – бүтін типті айнымалы, ол графикалық драйвердің түрін анықтайды.

Режим – бүтін типті айнымалы адаптердің жұмыс режимі анықтайды.

Драйвер – жол түріндегі шама драйвер файлының қайда екенін канықтайды. Бұл Egavega bgi – файлында көрсетіледі. Егер жоғарыдағы 2-файл бар болған жағдайда гшрафикалық драйвер оперативтік жвдқа графикалық драйвер жүктеледі. Бейне режимінде төмендегідей проц-н қолданылатын проце-р функциялар.

InitGraph (Драйвер, Режим, Драйвер орны)- графикалық режимді орнату.

Graph Resul – графикалық режимдегі функцияны ауыстырып қосады да, ол режимдегі қатенің кодын бүтін сан түрінде көрсетеді. Ол үшін GROK=0 қолданылады.

Get Graph Mode – графикалық режимнің ағымдағы коды.

Set Graph Mode – жаңа графикалық режимге көшу.

Restore CR Mode – мәтіндік режимге көшу.

Close Graph – графикалық режимді жабу.

Clear Graph – экрандық бетті тазалау.

Графикалық экранның координаталар жүйесі. Графикалық функциялар мен процедуралар.

Экранда суретті бейнелейтін жеке эелементті Пиксель деп атайды. Ол Х және У координатасының қиылысу нүктесінен тұрады да, ол Пиксельді әртүрлі түспен белгілеуге болады. Экран бетіндегі сол жақ ең жоғарғы нүкте (0,0) координатасы мен беріледі де, л нүктеден оңға қараған ось У деп таңбаланады. Экран бетіндегі Х осьінің ең үлкен мәні 639, У осьінікі 479. сондықтан монитор экранның айқындалуы

640*480 координата осьінен жұмыс істейтін мынандай функциялар бар GetMax X – X координатасының ең үлкен мәні.

GetMax X – y координатасының ең үлкен мәні

Get X – x координатасының ағымдағы мәні.

Get У – y координатасының ағымдағы мәні.

Get Pi[el (x,y) – координатысы xy б/н нүктесінің түсі.

Графикалық режимдегі элементтер графикалық фигураны бейнелейтін процедуралар бар. Олардың графикалық тримитив деп аталады.

Графикалық режимді алғашқы рет іске қосу үшін InitGraph процедурасы қолданылады, оның жазылуы:

InitGraph (var Driver, Mode: integer; ‘ ’);

Егер дисплей типін көрсеткіміз келмесе немесе оны білмесек, стандартты Detect тұрақтысын қолданамыз. Сонда InitGraph процедурасы көмегімен драйвер автоматты түрде таңдап алынады да, оның тиімді режімі де өзінен - өзі орнатылады. Graph модулінің маршрутын, яғни жолын көрсету үшін апостроф немесе параметр қойылады. Бағдарламаның бас жағында мынадай жолдар арқылы жазылады:

```
uses Crt, Graph;
var
Driver, Mode: integer;
  Driver := Detect;
  InitGraph (Driver, Mode, ‘ ‘);
```

Мұндағы: Driver – гарфикалық адаптердің типі;

Mode – оның жұмыс режимі;

‘ ‘ – графикалық адаптердің орналасқан орны

Driver:=Detect - адаптердің автоматты түрде таңдап алу үшін көрсетіледі.

ClrScr – экранды немесе терезені тазалап, курсорды экранның сол жақ жоғарғы бұрышына көшіреді, тек мәтіндік режимде жұмыс атқарады.

Circle(x,y:integer; r:word) процедурасы - ағымдағы түспен, көрсетілген центр және радиус бойынша шеңбер сызады.

GraphResult - графикалық режимге көшу кезіндегі қатені анықтайды.

GetPalette(var Palette: PaletteType) процедурасы - ағымдағы бояудың сипаттамасын береді.

GetPaletteSize функииясы - ағымдағы графикалық режимнің бояуындағы түстердін мөлшерін, типін береді.

GetDefaultPalette (var Palette:PaletteType) процедурасы - ескертусіз тағайындалған графикалық режимнің бояулардың сипаттамасын береді.

GotoXY(X,Y: Byte) - курсорды координаталары болып келетін экран нүктесіне көшіреді. X сол жақ шеттен оң жаққа қарай, ал Y жоғарыдан төмен қарай берілген қашықтықтар бірлігін көрсетеді.

GetMaxX - x-тің координатасының ең үлкен мәнін көрсету.

GetMaxY - y координатасының ең үлкен мәнін көрсету.

GetMaxColor функциясы - ағымдағы шығарылатын түстің номерін береді.

GetColor функциясы - ағымдағы графикалық режимдегі түстің ең үлкен номерін береді.

GetBkColor функциясы-шығарылатын ағымдағы фонның түсін береді.

GetMaxColor функциясы ағымдағы графиктік режимде шығатын түстің ең жоғарғы номерін береді.

Line(x1,y1,x2,y2) - x1, y1 координатасымен x2, y2 координатасын қосатын түзу сызық.

LineTo(X,Y:integer) процедурасы - кесіндіндіні ағымдағы курсор тұрған нүктеден координатасы көрсетілген нүктеге дейін сызады.

MoveTo(X,Y:integer) процедурасы курсорды көрсетілген координатаға орналастырады.

OutText(TextString:String) процедурасы - ағымдағы түспен, графикалық курсордың ағымдағы позициясынан бастап, тағайындалған бағытта текстi (жолды) шығарады.

OutTextXY(X,Y: integer; TextString: String) процедурасы көрсетілген

позициядан бастап экранға текст шығарады.

Rectangle(x1,y1,x2,y2) - сол жақ жоғарғы (x1,y1) және оң жақ төменгі төбелерінің координаталарын (x2,y2) бойынша төртбұрыш сызады.

SetPalette(ColorNum, Color:word) процедурасы - берілген бояудағы түсті реттік номері үшін стандартты бояудағы түсті өзгертуге мүмкіндік береді.

SetAllPalette(var Palette) процедурасы - алдыңғы процедурадағыдай, бірақ бояудағы түстің барлық реттік номерін өзгертеді.

SetColor(Color:word) процедурасы - графикалық немесе текстік информацияны графикалық экранға шығаратын ағымдағы түсті тағайындайды;

SetBkColor(Color:word) процедурасы - орыңдалғанда графикалық экранның ағымдағы фонның түсі берілген түске өзгереді.

SetLineStyle(LineStile, Pattern,Thickness: word) - сызық қалыңдығы мен стилін орнату.

SetTextStyle(Fount, Direction, CharSize: Word) - шығарылатын мәтіннің стилін орнатады, мұндағы Font – шрифт нөмері, Direction – шығарылатын шрифтің бағытын, CharSize – шығарылатын символдың өлшемін көрсетеді.

TextColor(Color: Byte) - экранға шығарылатын символдың түсін өзгертеді.

TextBackGround(Color: Byte) - экранның түсін өзгертеді.

TextColor(Color: Byte) - экранға шығарылатын символдың түсін өзгертеді.

Window(x1,y1,x2,y2: Byte) - экранда терезе құру процедурасы, x1,y1,x2,y2 - терезе төртбұрышының сол жақ жоғарғы және оң жақ төменгі төбелерінің координаталары.

CloseGraph - графикалық режимнен шығу.

Модульдер – деп әр түрлі әрекеттерді орындай алатын операторлар тобынан функциялардан және процедуралардан құралған жүйені айтады.

Crt – модулі оның құрамына экранның мәтіндік режимде жұмыс істеуін және пернетақтамен басқару әрекеттердің әрекетін қамтитын процедуралар мен функциялар орналасқан.

Graph – модулі экранның графикалық режимде жұмыс істеуін қамтамасыз ететін const– тан, функциядан, процедурадан тұрады. Оның көмегімен әртүрлі деңгейде суреттер алуға болады.

Графикалық экран үшін түсті тағайындау командаларын қарастырайық:

Палитра (бояулар) дегеніміз - бұл графикалық режимге арналған
түстер жиынтығы. Режимге арналған бірнеше палитралар бар, бірақ әдетте стандартты 16 түс пайдаланылады.

Black = 0 – қара	DarkGray = 8 – сұр
Blue = 1 – көк	LigthBlue = 9 – көкшіл
Green = 2 – жасыл	LightGreen = 10 – ақшыл жасыл
Gegan = 3 – көгілдір	LightCyan = 11 – ақшыл көк
Red = 4 – қызыл	LigthRed = 12 – қызғылт
Magenta = 5- күлгін	LightMagenta = 13 – ашық күлгін
Brown = 6 - қоңыр	Yellow = 14 – сары
LightGray = 7 – боз	White = 15 – ақ

Кестеде келтірілген түстердің ағылшынша атауларының номерлері GRAPH модулі үшін тұрақты (константа) болып есептеледі. Ағашқы 8 түсті (0..7) фонның түсі үшін де, шығару түсі үшін де пайдалануға болады, ал қалған түстер (8..15) тек графикалық бейнелерді шығару түсі үшін ғана пайдаланылады. Мысалы: экранға түрлі түсті палитралар мен сызықтар шығарады.

```
begin
d := Detect;
Initgraph(d, r, '  ' );
  E := GraphResult;
  if  e <> grok then
  Writeln (GraphErrorMsg (e));
begin
  SetLineStyle (Solidln, 0, Thickwidth);
  GetPalette (Palette);
  for color :=0 to Palette.Size-1 do
begin
  SetColor (color);
  Line (GetMaxX div 3, Color * 10, 2 * GetMaxX div 3, Color * 0)
end;
  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.
```

2 – ші мысал экранға бір уақытта палитраның түстері ауысады.

```
begin
  d := detect; initgraph(d, r, '  ');
  e := GraphResult;
  if e <> grok then
  Writeln (GraphErrorMsg (e));
begin
  SetLineStyle (Solidln, 0, Thickwidth);
   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;
Repeat
```

```
  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.
```

Кез – келген түс бойынша кез – келген координатталарда сызылатын түзулерді бейнелеу. Мысалы:

```
begin
   Gd := Detect;
   InitGraph (Gd, Gm,  ' ' );
   If GraphResult <> grOk then
   Halt (1);
   Randomize;
   Repeat
   SetColor (Random (GetMaxColor)+1);
   LineTo (Random (GetMaxX), Random (GetMaxY));
   Until  KeyPressed;
   End.
```

Келесі бағдарламада түрлі түсті шеңберлер экранға біртіндеп толтырылады.

```
begin
  d:= detect; initgraph(d, r, '  ' );
  e := graphresult;
  if  e <> grOK then
writeln (grapherrormsg (e))
else
 begin
   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);
 repeat
   setcolor (succ (random (white)));
```

```
  setlinestyle (0, 0, 2 * random (2) + 1);
  x := random (getmaxX);
  y := random (getmaxY);
  circle (x, y, random (getmaxX div 4));
  until keypressed;
  if readkey =# 0 then x := ord (readkey);
  readln;
  closegraph;
end
end.
```

кез – келген координаталарда сызылған түзулер экранға біртіндеп толтырылады:

```
Begin
  Gd := Detect;
  InitGraph (Gd, Gm, '  ' );
  Randomize;
  if GraphResult  <> grok then
  Halt (1);
  GetPalette (Palette);
  Repeat
  if Palette.Size <> 1 then
  SetBkColor (Random (Palette.Size));
  LineTo (Random (GetMaxX), Random(GetMaxY));
  Until KeyPressed;
End.
```