2 тақырып
Паскаль тіліндегі бағдарламалаудың негізгі құрылымдарының берілуі

- бағдарламалау жүйесі, виртуалды машина, транслятор, компилятор, интерпретатор ұғымы
- Паскаль бағдарламалау тіліне сипаттама, Паскаль тілінің алфавиті
- Паскаль бағдарламасының құрылымы
- деректер, скаляр ұғымы, идентификатор бейнесі
- жай, толық, заттық, логикалық, ауызша түрі
- деректер құрылымы
- индекс, жазу тізімі
- бейне мен операция ұғымы
- арифметикалық операциялар және бейнелер
- логикалық операциялар
- иелену операторы, құрамды оператор, шартты оператор
- write тәртібі, енгізу тәртібі туралы, таңдау операторы, қайталау операторлары, кіріспе шартты цикл операторы, соңғы шартты цикл операторы, параметрлі цикл операторы
- белгілер мен ауысу операторлары
- массивтермен жұмыс істеу

Паскаль тілінде жазылған программалардың жалпы құрылымын төмендегідей жазуға болады:

Program <программаның аты>

<сипаттау бөлімі>

Begin

<операторлар тарауы>

End.

Программаны бастау Begin қызметші сөзінен басталады. Ендігі арада орындалатын әрекеттерге сәйкес операторлық бөлік жазылады да программа End қызметші сөзінен аяқталады.

Turbo Pascal-да берілгендердің типтерін екі үлкен топтарға жіктеуге болады:

- Қарапайым типтер (скалярлық);
- Құрылымдық типтер (структуралық);

Қарапайым (скалярлық) типтердің өзі стандартты және пайдалунушылар типтері болып бөлінеді. Стандартты типтер: бүтін, нақты, логикалық (бульевтік), символдық(литерлік).

Құрылымдық (структуралық) типтер құрамына қарапайым типтер кіреді. Құрылымдық типтерге жолдар, массивтер, жазбалар және файлдар жатады.

Turbo Pascal тілінде атауды ***идентификатор*** деп атайды. Яғни, ***идентификатор*** программаның кез келген элементіне (айнымалылар, тұрақтылар, функциялар, файлдар, жиындар, т.б.) берілген атау. Ол ***стандартты*** және ***пайдаланушылар*** идентификаторлары болып екіге бөлінеді.

Стандартты функциялар, қызметші сөздер, т.с.с. стандартты объектілер атаулары стандартты идантификатор тобына жатады. Олардың көпшілігі алдын ала орындалатын операция немесе белгіленген элемент туралы мағлұмат беріп тұрады. Мысалы, Real (нақты сан), WriteLn (Write Line – жолды шығару), Begin (басы), End (соңы), Program (программа).

Пайдаланушылар идентификаторы ретінде әріптер мен цифрлар тізбегі алынады. Программалаушы идентификатордың төмендегі жазылу ережелерін білуі қажет:

- Идентификатор міндетті түрде әріптен басталатын латын әріптері мен цифрлардан тұрады;
- Оның құрамында орыс алфавитінің әріптері, арнайы символдар, әсіресе, бос орын болмауы қажет;
- Қызметші сөздер пайдаланушылар идентификаторы бола алмайды;
- Идентификатордың максимальды ұзындығы - 127 символ болғанымен, оның тек 63 символы Turbo Pascal-да оқылады;

Идентификатор қысқа, әрі түсінікті болғаны дұрыс. Себебі, оның мағынасы болмағаны қателік туғызбаса да, белгіленген программа элементі туралы мәлімет беруі тиімді екенін ескерген жөн. Мысалы: metka12, Blok, Window1, т.с.с.

Бір программада бірнеше объектіге бір ғана атау беруге болмайды. Бұл шарт орындалмаған жағдайда экранда төмендегідей хабарлама шығады: ***Error4: Duplicate identifier (Қате 4: Қайталанған идентификатор)***

Идентификатор тағайындауда программалаушылар тарапынан жиі жіберілетін қателіктер:

3Dgraph – цифрдан басталған;

Nomer.Doma – құрамында нүкте бар;

Blok# - арнайы символ қойылған;

My Program – құрамында бос орын бар;

Mod – қызметші сөз қолданылған.

Меншіктеу операторы қарапайым операторлар тобына жатады және негізгі оператор деп саналады. Жалпы түрі:

<айнымалы атауы>:= өрнек;

мұнда := меншіктеу белгісінің оң жағындағы өрнек есептелетін, сандық мәні оң жақтағы айнымалыға меншіктеледі.

Символдық мәндерді меншіктеуде апостроф (‘) таңбасын қолданамыз. Мысалы, K:= ‘Информатика’; J:=’*’;

Паскальда бірнеше айнымалы мәні бірдей болса, мән меншіктеуді i:=j:=k:=5 түрінде жазуға болмайды. Ол үшін меншіктеу операторын жеке жазған дұрыс: i:=5; j:=5; k:=5;

Меншіктеу операторы көмегімен логикалық өрнектің нәтижесінде айнымалыға меншіктеуге болады. Мысалы, R1:=(k>0) and (k<10); Мұндай өрнектің нәтижесі ақиқат (true) немесе жалған (false) сөздері, ал R1 айнымалысы логикалық (Boolean) типті болады.

Есептелінген өрнек нәтижесі мен оны меншіктейтін айнымалы типтері сәйкес болуы қажет. Меншіктеуде айнымалы нақты (Real) ал, өрнектің нәтижесі бүтін (Integer) типте және айнымалы типі жолдық (String) ал өрнек символдық (Char) болса, бұл қатеге жатпайды.

Айнымалыларға мәнін екі түрде сипаттауға болады: меншіктеу арқылы, мысалы, x:=5; немесе клавиатурадан еңгізу бұйрығы арқылы. Екінші әдіс жан-жақты программа қылдырады. Бір программаны айнымалынын әртүрлі мәндеріне есептейді. *Енгізу* операторы паскальда айнымалыға клавиатурадан оның мәнін сипаттауға колданылады. Еңгізу операторынын жалпы түрі:

read (<айнымалы_1>,…,<айнымалы_n>);

Оператор кездескенде программаның орындалуы тоқталады. Жүйе мәліметтерді еңгізетін күту режимына көшеді (экран қара, еңгізу курсоры жанып-сөнеді). Қолданушы клавиатурадан мәліметтердің мәндерінін арасынан пробел қалдырып еңгізеді немесе әр мәліметтен кейін <Enter> басып. Нәтижесінде тиісті айнымалыларға тұрақты мәндері беріледі. Бұйрық readln түрі:

readln (<айнымалы_1>,…,<айнымалы_n>);

Айырмашлықтары – readln бұйрығы орындалғанда әр айнымалынын мәні жаңа жолдан енгізілу керек. Осы бұйрықты параметрсіз readln; ТР нәтижелерін қарауға қолданылады. Программаға қайту үшін кезкелген батырманы басу керек. Булеан айнымалылардың мәндерін клавиатурадан еңгізіге ***болмайды.***

Экранға хабарлама немесе программаның нәтижесін *шығару* үшін write немесе writeln бұйрықтары қолданылады. Жалпы түрі:

Write(<өрнек_1>,…,<өрнек_n>);

Шығару тізімінде тұрақтылар, айнымалылар немесе өрнектер болуы мүмкін.

Тұрақтылар, айнымалылардың және өрнектердің мәндері шығару терезесіне шығарылады. TP-да осы терезе **Alt+F5** іске қосылады. Екінші түрі

Writeln(<өрнек_1>,…,<өрнек_n>);

Айырмашылығы – келесі кездесетін бұйрық мәндерді жаңа жолда шығарады. Жаңа жолға курсорды көшіру немесе жолды қалдыру үшін Writeln; қолданылады. Мысалы, келесі бұйрықтар жызалған болса Write(‘p=’, p); Writeln(‘s=’, s); Writeln(‘Программаны құрған Серікбаева Ә.Б.’); және бастапқы мәндер берілсін a=5, b=3.6, c=4.2, онда экранда ппериметрдің, ауданнын мынадай мәндері шығады:

p=1.2800000000E+01s=7.429239530E+00

Программаны құрған Серікбаева Ә.Б.

Шығару операторы арқылы мәліметтерді форматпен шығараламыз. ***Форматтау*** – нәтижелерді қолданышуға ыңғайлы түрде шығару. Ол үшін өрнектен кейін формат қойылады – қос нүкте және сан, өрнектін мәнін шығаратын позициялар. Мысалы, үшбұрыштың бұрыштарынын координаталары берілсін. Медианаңы mb және сырттай сызылған шеңбердін радиусын.

```
Program triangle;
Var x1,x2,x3,y1,y2,y3,a,b,c,mb,r,x,y,p,s:real;
Begin
Write(‘Координаталарды енгіз:’);
Read(x1,x2,x3,y1,y2,y3);
{үшбұрыштың қабырғаларынын ұзындығын есептеу}
a:=sqrt(sqr(x3-x2)+sqr(y3-y2));
b:=sqrt(sqr(x1-x3)+sqr(y1-y3));
c:=sqrt(sqr(x1-x2)+sqr(y1-y2));
{b қабырғасынын ортасынын координатасы}
x:=(x1+x3)/2;
y:=(y1+y3)/2;
{mb медианаңы есептеу}
mb:=sqrt(sqr(x-x2)+sqr(y-y2));
{жартылай периметр}
p:=(a+b+c)/;
{ауданды есептеу}
s:=sqrt(p*(p-a)*(p-b)*(p-c));
{радиус}
r:=a*b*c/(4*s);
{нәтижелерді шығару}
writeln(‘медиана=’, mb:5:2, ‘ радиус=’, r:5:2);
end.
```

Форматты қолданбағанда, онда бүтін және булян түрлерге экранда 15, ал нақты түрге – 18 орын беріледі. Нақты түрдегі мәліметтер келесі түрде шығады:

х.ххххххххххЕтаңбахх,

мұнда х – кезкелген сан.

Паскаль тіліндегі программада операторлар жазылған ретімен орындалады. Осы реттілікті өзгертіп отыратын, яғни программаның кейбір бөліктерін орындамай өтіп кету және кері қайту үшін, *шартсыз көшу* операторы қолданылады. Жалпы түрі:

GOTO таңба;

GOTO операторы орындалғанда, программаның орындалу реті бұзылып таңбамен белгіленген операторға басқару беріледі. Бұл таңбалар label бөлімінде сипатталады.

GOTO операторы программаның логикалық құрылымын күрделендіріп жібереді. Сол себептен бұл операторды жиі қолдануының қажеті жоқ.

Бірнеше операторлардың бірігіунен шыққан операторды *құрама* оператор деп атаймыз. Бұл операторлар begin (басы) және end (соңы) қызметші сөздер арасында жазылады. Әр оператордан кеін (;) таңбасы қойылады. Жалпы түрі:

Begin
Оператор 1;
Оператор 2;
……
оператор N;
End;

Begin (басы) және end (соңы) сөздерін операторлдар жақшасы деп қарастырады. Құрама операторлдың ішінде тағы бір құрама оператор болуы мүмкін. Құрама операторға шартты көшу, таңдау және қайталау операторлары жатады.

Алгоритмдік тілде қойылған шартқа байланысты екі немесе екіден көп тармақтары бар алгоритм – *тармақталған алгоритм* деп аталады. Осындай тармақталған алгоритмді программалауға шартты көшу операторы қолданылады. Жалпы түрі:

IF <шарт> THEN <оператор 1> ELSE <оператор 2>;

Мұнда IF (егер) қызметші сөзінен кейінгі жазылған шарт ақиқат болса THEN (онда) сөзінен кейінгі жазылған <оператор 1> орындалады, шарт сақталмаса ELSE (әйтпесе) сөзінен кейінгі <оператор 2> орындалады. Егер, шартқа байланысты орындалатын бір ғана оператор болса, шартты көші операторы қысқаша түрде жазылады:

IF <шарт> THEN <оператор 1>;

ELSE сөзінін алдындағы операторлың соңына үтір нүкте қойылмайды.

Мысалы, х аргументі бойынша функцияның мәнін есептеудің программасын жазу

$$Y = \begin{cases} x, & x < 0 \\ x^2, & x \geq 0 \end{cases}$$

```
var x,y:real;
begin
wtite(‘x=’); read(x);
if x<0 then y:=x else y:=sqr(x);
writeln (‘x=’,x, ‘ y=’, y);
end.
```

IF операторының қысқартылып отырған түрінде берілген шарт орындалмаса, оператор тасталып кетеді де, басқару келесі операторға өтеді.

Программада қажеттілікке қарай IF операторын бірнеше рет қолдануға болады. Мысалы,

```
IF a<b THEN min:=c ELSE min:=b;
IF c<min THEN min :=c;
```

Егер THEN және ELSE қызметші сөзінен кейін орындалатын операторлар саны екі немесе екіден де көп болатын болса, онда бұл операторларды begin және end операторлар жақшасына аламыз. Жалпы түрі:

IF <шарт> THEN
Begin
<оператор 1>;
<оператор 2>;
…….
<оператор N>
end
ELSE
Begin
<оператор 1>;
<оператор 2>;
…….
<оператор N>
end;

Есептің күрделенуіне байланысты IF операторы құрамына екінші бір IF операторын кірістіруге болады.
Жалпы түрі:

1) IF <шарт 1> THEN
IF <шарт 2> THEN <оператор 1> ELSE <оператор 2>
ELSE <операторт 3>;

2) IF <шарт 1> THEN <оператор 1>
ELSE
IF <шарт 2> THEN <оператор 2>
ELSE <операторт 3>;

3) IF <шарт 1> THEN
IF <шарт 2> THEN <оператор 1>
ELSE <операторт 2>;

Программада берілген шартты жазу бір немесе бірнеше логикалық қатынастардан тұратын, өрнек арқылы жазылады. Логикалық қатынас амалдары екі операторларды бір-бірімен салыстыруды және олардың қайсысы ақиқат және жалған екендігін анықтайды. Негізгі қатынас амалдары:

Амал	Аталуы	Өрнек
=	Тең	A=B
<>	Тең емес	A<>B
>	Үлкен	A>B
<	Кіші	A<B
>=	Үлкен немесе тең	A>=B
<=	Кіші немесе тең	A<=B
in	Құрамына ену	A in B

Егер шарт қарапайым болса, оны бір қатынас таңбасымен өрнектей аламыз. Ал күрделі шартты жазудаарнайы логикалық амалдарды пайдаланып, логикалық өрнек ұйымдастырамыз. Логикалық өрнек күрделі болғанымен, шартты тексергенде, нәтижесі логикалық түр – boolean болады.

Паскальда **CASE** операторды қолданып программаларды құрған кезде келесі бөлімдерді еске сақтаған жөн:

CASE инструкциясы программаның әрі қарай жұмыс істеуін анықтау үшін арналған инструкция;

Инструкцияның алдыңғы тобында көрсетілген константаға айнымалы-селектордың мәні тең болған кезде программаның әрі қарай тізбектеліп орындалуы таңдалады;

Айнымалы-селектор ретінде бүтін типті айнымалылар (**INTEGER**) немесе симолдық (**CHAR**) типтер қолданылады.

CASE инструкциясы жалпы түрде программада былай жазылады:

CASE айнымалы **OF**

1 константа: 1 инструкцияның тізбегі;

2 константа: 2 инструкцияның тізбегі;

3 константа: 3 инструкцияның тізбегі;

ELSE

Егер айнымалының мәні константалар тізімінің еш қайыссысымен сәйкес келмесе қолданылатын инструкция.

END;

мұндағы *айнымалы* - программаның әрі қарай жұмыс істеуін анықтау үшін арналған айнымалы;

константа – үтір арқылы бөлінген константалар. Егер константалар сандар диапазонынан тұрса, онда тізімнің орнына константаның бірінші саны мен соңғы санын екі нүкте арқылы бөліп көрсетуге болады. Мысалы, 1,2,3,4,5,6 деген тізім 1..6 диапазонымен өзгертіліп жазыла алады.

CASE инструкциясы келесі түрде орындалады. Алдымен **CASE** сөзінен кейін орналасқан айнымалының мәні есептеледі, содан кейін алынған мән бірінші константамен салыстырылады. Егер айнымалының мәні константаға тең болса онда 1 инструкцияның тізбегі орындалады да, осымен инструкцияның орындалуы аяқталады. Егер айнымалының мәні бірде-бір константамен сәйкес келмесе, онда **ELSE** сөзінен кейін орналасқан инструкция орындалады. **CASE** инструкциясының синтаксисі **ELSE** сөзін және оның инструкциясын жазбауға мүмкіндік береді.

Егер бірнеше тұрақтыға бір ғана оператор сәйкес келсе тұрақтыларды үтір (,) арқылы жазуға болады. Ал, тұрақтылар диапозонын көрсету үшін (..) таңбасын пайдаланамыз.

Циклдік құрылымды алгоритмді программалау

Берілген есепті шешуде алгоритмнің кейбір бөліктері бірнеше рет қайталанып орындалуы мүмкін. Мұндай құрылымды алгоритмді **қайталанушы алгоритм** немесе циклдік құрылымдық алгоритм деп атаймыз. Turbo Pascal-да циклдік құрылымдық алгоритмді программалауды үш түрлі жолмен ауыстыруға болады.

1. Алдын–ала шартты тексеру арқылы.
2. Келесі шарт бойынша
3. Параметрдің мәніне тәуелді.

While операторы

Алдын –ала берілген шартты тексеру арқылы циклді ұйымдастыруға **While** операторы қолданылады. **While** операторын циклдің қайталану саны белгісіз болғанда пайдаланған ыңғайлы. Қайталанушы процесс қойылған шартты тексеру арқылы жүзеге асады. Яғни, шарт ақиқат болса цикл қайталанады да, шарт жалған болса циклдан шығу орындалады. While оператор екі бөліктен тұрады: циклдің тақырыбынан және циклдің денесінен. Жалпы жазылуы:

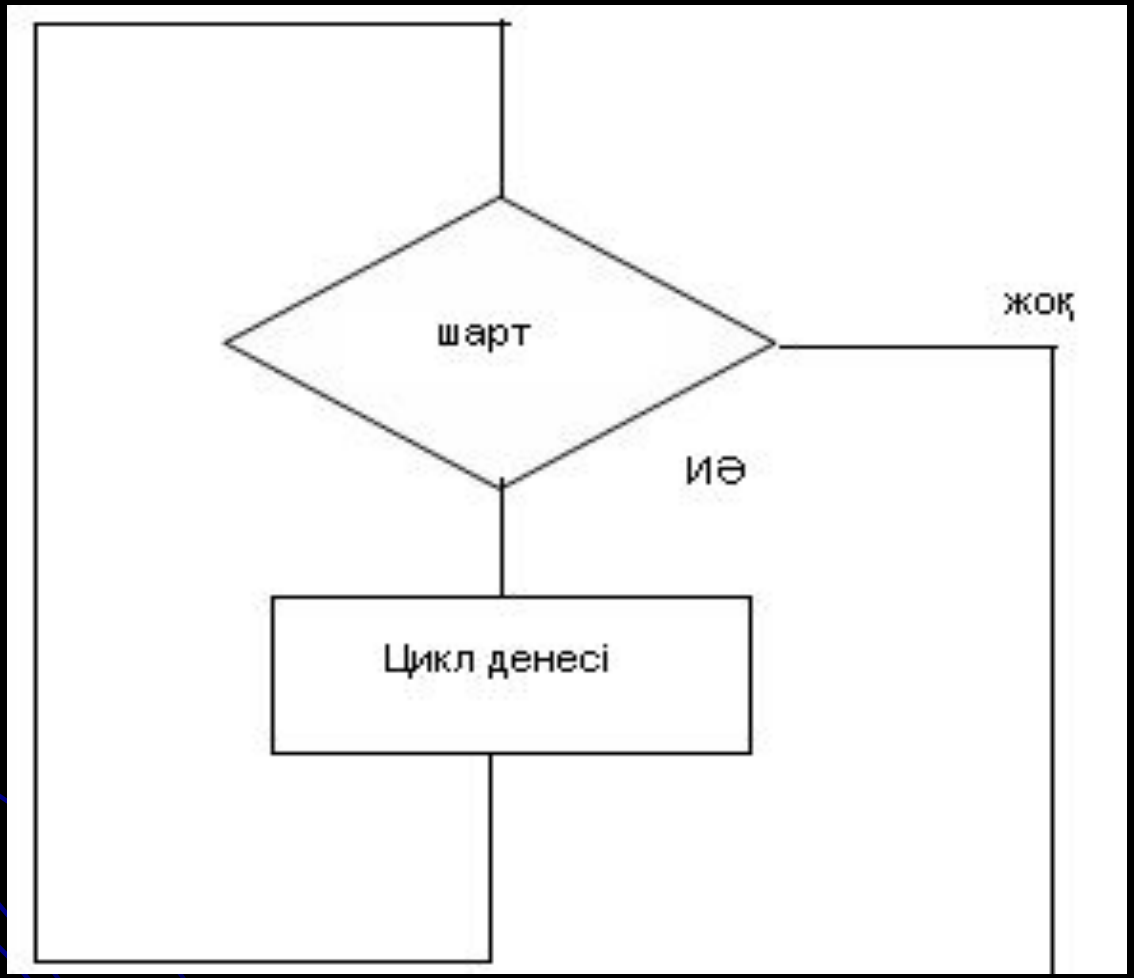
While <шарт> **do**

<циклдің денесі>;

Мұндағы қызметші сөздердің қазақша мағынасы: **While** – “әзір”, **do** – “орында”. Ал, шарт – логикалық өрнек түрінде жазылады. Берілген шартқа тәуелді бірнеше рет қайталанып орындалатын операторды – ***циклдің денесі*** деп атаймыз.

While операторы алгоритмдік тілдегі “әзір” цикл командасына сәйкес. “Әзір” цикл командасының жазылуы және блок схемасы:

әзір <шарт>
цб
    <цикл денесі>
Цс
Бітті


шарт
жоқ
иә
Цикл денесі

Жоғарыдағы блок схема While операторымен ұйымдастырылатын циклді толық сипаттайды. While операторында, әрбір қайталанудың алдында берілген шарт тексеріледі. Шарт ақиқат болса цикл денесі орындалады. Егер шарт орындалмаса цикл денесі бірде-бір рет орындалмайды.

Егер цикл денесі екі немесе екіден де көп операторлардан тұрса, оларды операторлар жақшасының ішіне жазамыз.

While<шарт> **do**
Begin
<циклдің денесі>;
End;

Мысалы: y=x2 функциясының мәнін есептеу керек, мұндағы x=1,2,3,4,5,6. программаның нәтижесінде x аргументінің мәні мен сәйкес функцияның мәні кесте түрінде шығады. X-тің өзгеру қадамы 1-ге тең.

```
Program esep1;
Var y, x: integer;
Begin
X:=1            {x-тің бастапқы мәнін меншіктеу}
While x<=6 do {цикл тәуелді болатын шарт}
Begin
Y:=sqr(x);      {x-тің мәні бойынша н-ті есептеу}
Writeln('x=',x,'_|_y=',y);    {x және y экранға шығару}
X:=x+1;             {x-тің өзгеру қадамы}
End;
End.
```

Келесі шарт бойынша циклді ұйымдастыру

Циклдік процесстерді ұйымдастыруда **Repeat** операторы циклдің қайталану саны белгілі болғанда қолданылады. **Repeat** операторының жалпы жазылуы:

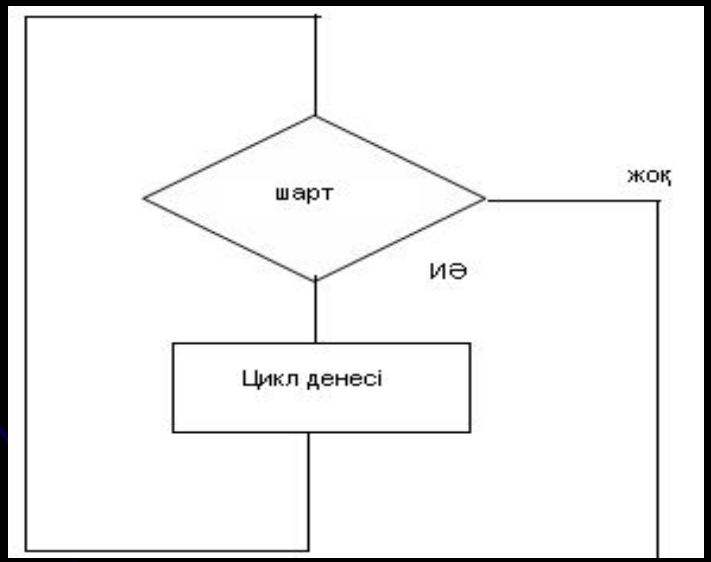
Repeat

<циклдің денесі>;

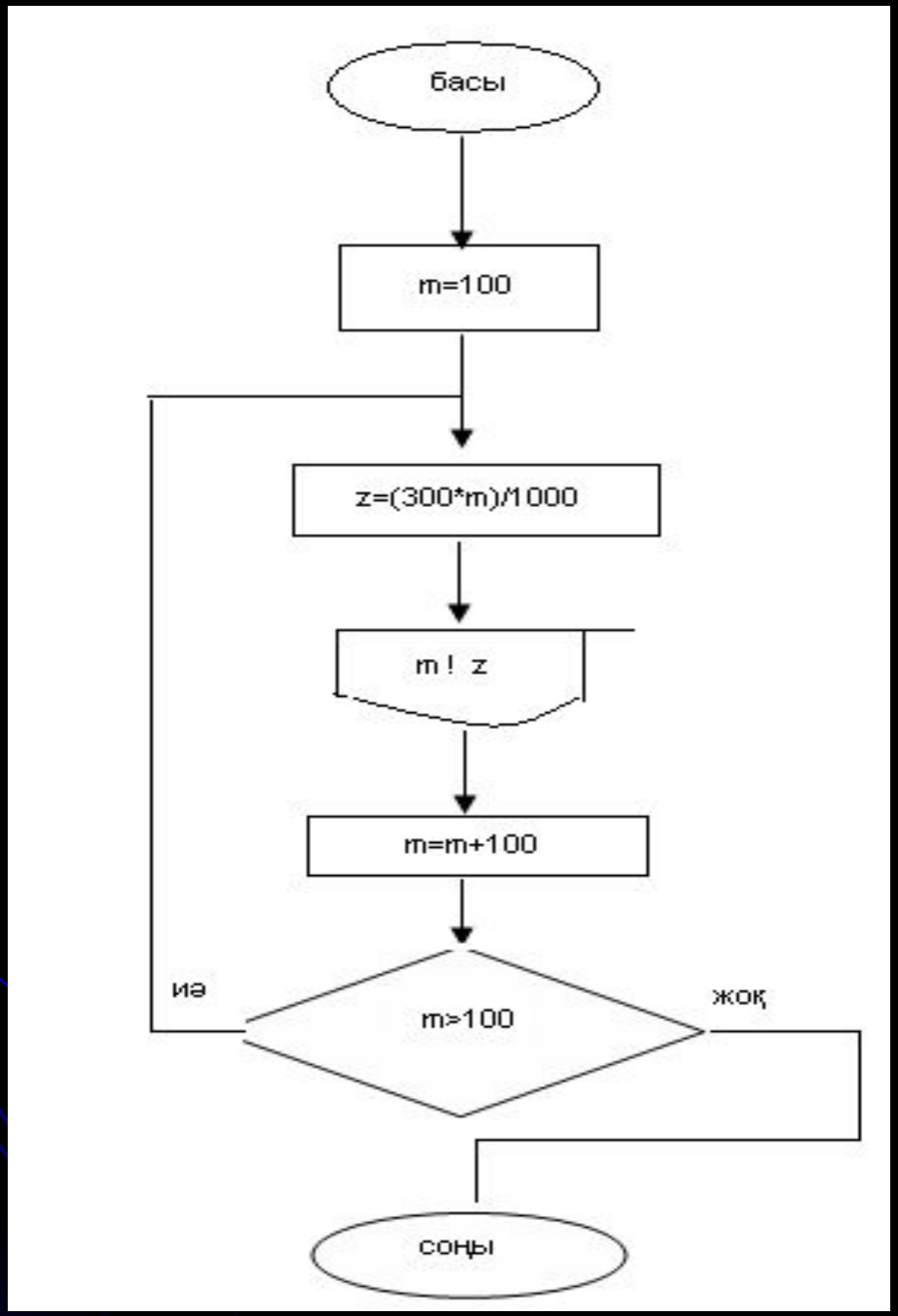
Until <шарт>;

Мұндағы қызметші сөздер **Repeat** –қайтала, **until** – соған дейін деген мағынада қолданылады. Циклдің денесі –қайталанып орындалатын бір немесе бірнеше операторлардан тұрады. Цикл денесін құрайтын операторлар санына шектеу қойылмайды. Шартты тексеру логикалық өрнек арқылы жүргізіледі.

Repeat операторы алгоритмдік тілдегі “дейін” цикл клмандасына ұқсас. “Әзір” цикл командасынан “дейін” циклінің айырмашылығы: қойылған шартқа тәуелсіз бірінші цикл денесі орындалады. Содан кейін, шарт тексеріледі. Демек, шарт ақиқат болмаса цикл денесі кемінде бір рет орындалады. **Repeat** операторының блок схема арқылы сипаттауға болады:



1-мысал: 1кг ірімшік 300теңге тұрады. Ірімшіктің 100,200,300...,1000 грамына төленетін теңгені анықтайтын және есептің жауабын кесте түрінде шығарудың блок схемасын және программасын жазайық.


басы
m=100
z=(300*m)/1000
m ! z
m=m+100
иә
m>100
жоқ
соңы

```
Program esep 2;
Var m:integer; z:real;
Begin
        M:=100;
        Repeat
        Z:=(300*m)/100;

Writeln(m,'_!_',z:4:0);
        M:=m+100;
Readln;
End.
```

For операторы

Циклдік құрылымды алгоритмді программалауда, қайталанушы процесс бір айнымалының мәніне тәуелді болса **For** операторын қолданамыз. Айнымалы тек бір қадамға ғана өзгере отырып, циклді басқарады. Бұл, айнымалы циклдің параметрі делінсе, **For** операторы *параметрлі қайталану операторы* деп аталады.

For операторы алгоритдік тілдегі параметрлі қайталану командасына сәйкес келеді.

і үшін m1 бастап m2 дейін h қадам
цб
серия
цс

For операторы екі түрлі жазылады:

1-нұсқасы:

For X:=M1 **to** M2 **do** S1

Мұндағы қызметші сөздер: **For** (үшін), **to** (дейін) –циклдің қадамы +1-ге өсіп отыратындығын көрсетеді, **do** (орында);

X –скалярлық типтегі айнымалы циклдің параметрі.

M1 –цикл параметрінің бастапқы мәні,

M2 –цикл параметрінің соңғы мәні.

S1 –цикл параметріне тәуелді қайталанып орындалатын оператор. Сондықтан, **S1**-цикл денесі деп аталады.

2-нұсқасы:

For X:=M1 **downto** M2 **do** S1;

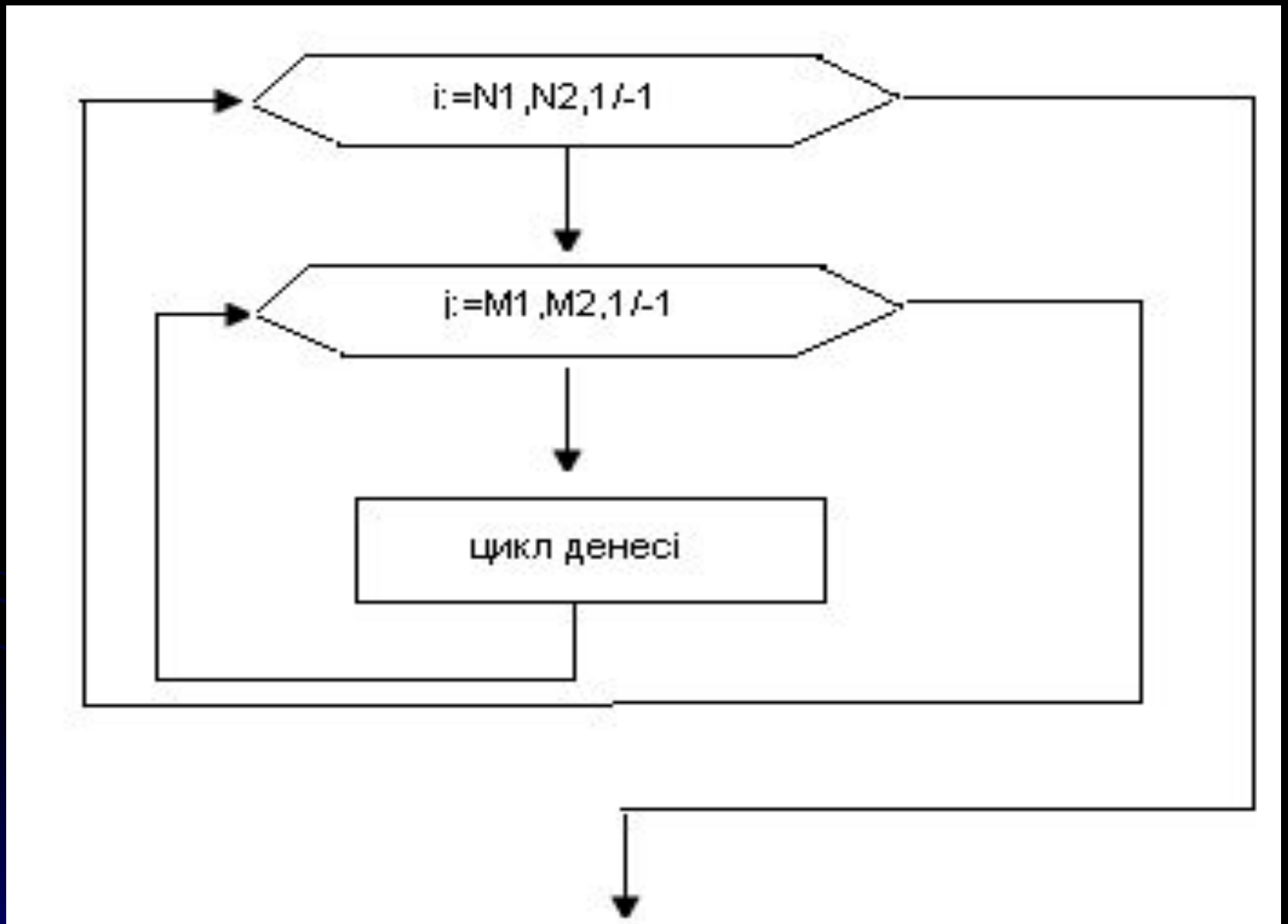
1-нұсқадан өзгешелігі **to** сөзінің орнына **downto** (төменге дейін) қызметші сөзі жазылады. **Downto** циклдің өзгеру қадамы -1 тең екендігін көрсетеді. мұнда X-тің мәні M1-ден M2-ге дейін -1 қадаммен кему үшін M1>M2 шарты орындалуы керек. Егер бұл шарт орындалмаса цикл денесі бірде бір рет орындалмайды.

Егер цикл денесі бірнеше операторлардан тұрса, begin және end операторлар жақшасын пайдаланамыз.

Күрделі циклдер

Күрделі қайталанушы процесстерді ұйымдастыруда бір цикл операторы құрамында екінші бір цикл операторы болуы мүмкін. Бұл жағдайда бірінші цикл операторы – сыртқы цикл, ал, оның құрамындағы екінші оператор – ішкі цикл деп аталады. Сыртқы және ішкі циклді ұйымдастыруда мына шарт орындалады: ішкі циклдің барлық операторы сыртқы цикл денесіне енеді.

Мысалы, суретте екі параметрлі цикл операторлары арқылы жазылған күрделі циклдің жұмысы сипатталынған


i:=N1,N2,1/-1
j:=M1,M2,1/-1
цикл денесі

Массивтің элементтерінің мәнін алуды ұйымдастыру

Берілген массивтің бірінші элементіне қатынас алу үшін массив атауынан кейін бірінші индексті көрсету қажет: A[1]; массивтің бесінші элементіне қатынас алу үшін A[5] деп көрсетіледі

A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]
12	11	5	-2	405	-3	9	-7
Массивтің 1 элементі	Массивтің 2 элементі	Массивтің 3 элементі	Массивтің 4 элементі	Массивтің 5 элементі	Массивтің 6 элементі	Массивтің 7 элементі	Массивтің 8 элементі

Берілген массивтің кез-келген элементтеріне арифметикалық операцияларды, салыстыру және меншіктеу операторларын қолдануға болады.

Массивтің кез-келген элементіне нәтиже беру үшін, меншіктеу операторы қолданылады:

Массив атауы[**индексі**]:=*нәтиже*

Егер программада массив элементтерінің барлық мәндерін экранға шығару керек болса, FOR инструкциясын қолданған өте ыңғайлы. Бұл айнымалы-счетчикті массив элементінің индексі ретінде қолдануға болады.

Мысалы: Апта күндерін атауларын экранға шығару программасын құру.

```
Program days;
Var
    Day:array[1..7] of string[11];
    I:integer;
Begin
    Day[1]:=’Дүйсенбі’;
    Day[2]:=’Сейсенбі’;
    Day[3]:=’Сәрсенбі’;
    Day[4]:=’Бейсенбі’;
    Day[5]:=’Жұма’;
    Day[6]:=’Сенбі’;
    Day[7]:=’Жексенбі’;
    For i:=1 to 7 do
    Writeln(i,’      ‘,day[i]);
End.
```

Массив элементтерінің мәнін енгізу үшін де, For циклдық айнымалысын қолданған өте ыңғайлы. Программаны қолданушы программа қай массив элементін енгізу керектігін білу үшін, енгізердің алдында шығару операторын қолданып қысқашы түсіндірме жазу керек.

Қолданылған әдебиеттер тізімі

1) Марко Кэнту. Delphi 5 для профессионалов. –СПб.:Питер, 2001.
2) Бабушкина И. А., Окулов С.М. Практикум по объектно-ориентированному программированию. М.: БИНОМ, Лаборатория знаний, 2004. – 366 бет.: ил.
3) Хомоненко А.Д. и др. Delhpi 7. – СПБ.: БХВ-Петербург, 2004.- 1216 бет:ил.
4) Фаронов В.В. Delhpi 5: Учебный курс.-М.: Нолидж, 2001.- 605 бет.: ил.
5) Фаронов А.В. TURBO PASCAL /учебник// Изд. «Питер» М.-2001.
6) Культин Н. Turbo Pascal в задачах и примерах. - СПб.: БХВ-Петербург, 2001.-256 бет: ил.
7) Матаев С. Delphi 7. Бағдарлама құру негіздері: Оқу құралы. Қарағанды, 2005. – 271 б.
8) Н.Культин. Основы программирования в Delphi7. – СПБ.: БХВ-Петербург, 2003.