4. ПРОГРАМИ С ЦИКЛИ

Цикълът е алгоритмична структура, при която се реализира многократно повтаряне на група оператори. Такова повтаряне още се нарича **итерация**. Той се състои от две основни компоненти - тяло на цикъла и условие за излизане от цикъла.

Тялото на цикъла се състои от операторите, които трябва да се повторят многократно, за да се реши поставената задача и някои оператори, необходими за организацията на самия цикъл.

Условието за излизане от цикъла определя кога да се прекрати повтарянето на операторите от тялото на цикъла.

Към цикъла обикновено се отнасят и групата оператори, които подготвят влизането в цикъла, например присвояване на начални стойности на променливи, участващи в тялото на цикъла и др. Тази група оператори образува така наречената *подготовка (инициализация)* на цикъла.

За описване на цикли Паскал предлага *оператор за цикъл с предусловие, оператор за цикъл с постусловие и оператор за цикъл с параметър*. При всеки от тях инициализацията предхожда оператора за цикъл.

4.1. Оператор за цикъл с предусловие

Той има вида

While *Логически израз* **do** *Оператор;*

където *Оператор* може да бъде и съставен оператор. Изпълнението на оператор за цикъл с предусловие се заключава в следното:

а) Изчислява се *логическия израз*;

б) Проверява се намерената логическа стойност, т.е.

- ако тя е true се изпълнява *Оператор* и се преминава отново към операция а);
- ако тя е false се излиза от цикъла, т.е. изпълнява се следващия оператор след оператора за цикъл.

За този цикъл е характерно следното:

- в тялото на цикъла трябва да се променя стойността поне на една от променливите участващи в *Логическия израз*, иначе е невъзможно да настъпи момент, в който стойността му да е станала false.
- може да се излезе от цикъла без нито едно изпълнение на тялото му.

Пример 4.1. Програма, която изчислява стойностите на функцията y=sin x , за x=0.5, 0.6,...,1.5 rad и извежда в две колони стойностите на x и y.

```
Var X, Y:real;
Begin
 X := 0.5;
 While X <= 1.5 do
   begin
     Y := sin(X);
     Writeln ('X = ',X:3:1,' ':3,'Y = ',Y:8:5);
     X := X + 0.1
   end
End.
```

Пример 4.2. Програма за намиране на средно-аритметичното на последователно задавани цели числа до задаването на определено контролно число A.

```
Var
 A, n, K:integer;
 S, C:real;
Begin
   S:=0; n:=0;
   Write('Въведете контролното число: '); Readln(A);
   Write('Въведете първото число:  '); Readln(K);
   While K<>A do
     begin
       S:=S+K; {Сумиране на въвежданите числа}
       n:=n+1;  {Преброяване на въвежданите числа }
       Write('Въведете следващо число:  '); Readln(K);
     end;
   If n>0
     then begin
       C:=S/n;
       Writeln('Средно аритметичното е ', C )
     end
     else Writeln('Не са въведени числа, различни от контролното число.')
End.
```

Променливата S не е от целочислен тип, защото в процеса на натрупване може да надхвърли максимално допустимата целочислена стойност.

Пример 4.3. Програма за обръщане реда на цифрите в дадено число.

Нека да започнем с отделянето на цифрите на едно конкретно число, например 25387. При последователно целочислено деление на 10 ще получим следната картина за частните и за остатъците:

5387	→	538	→	53	→	5	→	0
↓		↓		↓		↓		
7		8		3		5		

Както се вижда от примера, при целочислено деление на 5387 на 10 се получава като остатък последната цифра (7) и като частно числото 538. При целочислено деление на 538 на 10 се получава като остатък предпоследната цифра (8) и като частно числото 53. При целочислено деление на 53 на 10 се получава като остатък втората цифра (3) и като частно числото 5. При целочислено деление на 5 на 10 се получава като остатък първата цифра (5) и като частно числото 0.

Обърнатото число се получава както следва:

1. Умножаваме цифрата 7 по 10 и прибавяме цифрата 8.
2. Полученото число умножаваме по 10 и прибавяме 3.
3. Полученото число умножаваме по 10 и прибавяме 5.

Програмата за обръщане реда на цифрите в дадено число има вида:

```
Var
 M,N,L:LongInt; C:byte;
Begin
 Write('Задайте числото: ');Readln(M);
 L:=M;
 N:=0;
 While L > 0 do
   begin
     C:=L mod 10;
     N:=N*10+C;
     L:=L div 10
```

```
  end;
 Writeln('Обърнатото число е ',N);
 Readln
End.
```

Пример 4.4. Програма за намиране броя на дните между зададена начална дата Dn,Mn,Gn и зададена крайна дата Dk,Mk,Gk.

Броят на дните между зададена начална дата Dn,Mn,Gn и зададена крайна дата Dk,Mk,Gk се намира както следва:

- сумират се последователно дните на всички месеци от месеца Mn, присъстващ в началната дата до месеца Mk-1, предшестващ месеца, който присъства в крайната дата;
- към намерената сума се добавя Dk и се изважда Dn.

Броят на дните в месец със зададен номер и година се определя както е показано в пример 3.4.

```
Var
 Dn,Mn,Gn,Dk,Mk,Gk,BrDni,BrDnMes,M,G:integer;
Begin
 Writeln('Въведете началната дата:');
 Write('  - ден: '); Readln(Dn);
 Write('  - месец: '); Readln(Mn);
 Write('  - година: '); Readln(Gn);
 Writeln('Въведете крайната дата: ');
 Write('  - ден: '); Readln(Dk);
 Write('  - месец: '); Readln(Mk);
 Write('  - година: '); Readln(Gk);
 G:=Gn; M:=Mn; BrDni:=0;
 While (G<Gk) or (M<Mk) do
   begin
     Case M of
       1,3,5,7,8,10,12:        BrDnMes:=31;
       4,6,9,11:               BrDnMes:=30;
       2:      If (G mod 400=0) or (G mod 4=0) and (G mod 100<>0)
                 then BrDnMes:=29
                 else BrDnMes:=28
     end;
     BrDni:=BrDni+BrDnMes;
     M:=M+1;
     If M=13
       then begin G:=G+1;M:=1 end
   end;
 BrDni:=BrDni+Dk-Dn;
 Writeln('От ',Dn,'.',Mn,'.',Gn,' до ',Dk,'.',Mk,'.',Gk,' има ',BrDni,' дни.');
 Readln
End.
```

4.2. Оператор за цикъл с постусловие

Той има вида:

Repeat

Оператор 1;

Оператор 2;

. . . . . .

Оператор N

until *Логически израз;*

Всеки от операторите може да бъде съставен оператор. Изпълнението на оператора за цикъл с постусловие се заключава в следното:

а) Изпълнява се тялото на цикъла (*Оператор1; Оператор2; ... ОператорN*);

б) Изчислява се стойността на *логическия израз*;

в) Според стойността на логическия израз се извършва следното:

- при стойност false - преминаване към точка а) с цел повтаряне на цикъла;

- при стойност true - излизане от цикъла.

За този цикъл е характерно следното:

- в тялото на цикъла трябва да се променя стойността на поне една от променливите, участващи в *Логическия израз*;

- не може да се излезе от цикъла, без да се изпълни поне един път тялото му.

Пример 4.5. Програма, която изчислява стойностите на функцията y=sin(x), за x=0.5, 0.6,...,1.5 rad и извежда в две колони стойностите на x и y.

```
Var
 X, Y:real;
Begin
 X := 0.5;
 Repeat
   Y := sin(X);
   Writeln ('X = ',X:3:1,' ':3,'Y = ',Y:8:5);
   X := X + 0.1
 until X>1.5
End.
```

Пример 4.6. Програма за последователно въвеждане и обработване на цели числа.

Когато се въведат N ненулеви или три нулеви числа, въвеждането се прекратява и се извежда средно аритметичното на нечетните и броят на четните от въведените числа.

```
Var
 x,N,BrNul,BrNeNul,BrChet,BrNeChet, SumNeChet:integer;
 C:real;
Begin
 Write('N=');Readln(N);
 BrNul:=0;          {Брой на нулевите числа}
 BrNeNul:=0;        {Брой на ненулевите числа}
 BrChet:=0;         {Брой на четните числа}
 BrNeChet:=0;       {Брой на нечетните числа}
 SumNeChet:=0;      {Сума на нечетните числа}
 Repeat
   Write('Въведете число: '); Readln(X);
   If x=0
     then BrNul:=BrNul+1
     else begin
       BrNeNul:=BrNeNul+1;
       if x<>0
         then if odd(x)
                 then begin
```

```
                SumNeChet:=SumNeChet+x;
                BrNeChet:=BrNeChet+1
              end
          else BrChet:=BrChet+1
    end
  until (BrNeNul=N) or (BrNul=3);
  If BrNeChet>0
    then begin
        C:=SumNeChet/BrNeChet;
        Writeln('Средно-аритметичното на нечетните числа е ', C:7:3)
      end
    else Writeln('Не са въведени нечетни числа.');
  Writeln('Броят на четните числа е ', BrChet);
  Readln
  End.
```

Както се вижда от примерите, повечето цикли могат да се опишат както с помощта на оператора за цикъл с предусловие, така и чрез оператора за цикъл с постусловие, но с различна степен на удобство и прегледност. Основното е, че когато трябва да е възможно да се мине през оператора за цикъл без нито едно изпълнение на тялото му, то трябва да се избере цикъл с предусловие. Когато е наложително поне едно изпълнение на тялото, то трябва да се избере цикъл с постусловие.

4.3. Оператор за цикъл с параметър

С разгледаните дотук оператор за цикъл с предусловие и оператор за цикъл с постусловие може да се опише всеки цикличен участък на една алгоритмична структура. Въпреки това за случаите, когато броят на изпълнението на операторите от тялото на цикъла е предварително известен, се предлага специален оператор за цикъл, който е по-удобен и по-бърз. Той е в два варианта: с нарастващ параметър и с намаляващ параметър.

4.3.1. Оператор за цикъл с нарастващ параметър

Той има вида:

For *Параметър :=Начална стойност* **to** *Крайна стойност* **do** *Оператор;*

където:

Параметър е променлива от дискретен тип (цял, символен, логически, изброен или ограничен тип);

Начална стойност и *Крайна стойност* на параметъра са изрази от същия тип, от който е *Параметър*.

Оператор представлява тялото на цикъла и може да бъде прост или съставен оператор.

Операторът се изпълнява в следната последователност:

а) Изчисляват се *Начална стойност на параметъра* и *Крайна стойност на параметъра*;

б) Сравняват се Начална стойност на параметъра и Крайна стойност на параметъра и:

- ако *Начална стойност>Крайна стойност*, следва излизане от цикъла;
- ако *Начална стойност<=Крайна стойност*, на *Параметър* се присвоява *Начална стойност*;

в) Изпълнява се *Оператор*;

г) Сравнява се стойността на *Параметър* с *Крайна стойност* и:

- ако *Параметър <> Крайна стойност*, то *Параметър* приема следващата дискретна стойност за съответния тип (i := succ(i)) и следва връщане към т. в);
- ако *Параметър=Крайна стойност*, то следва излизане от цикъла.

Пример 4.7. Програма, която изчислява стойностите на функцията y=sin(x), за x=0.5, 0.6,...,1.5 rad и извежда в две колони стойностите на x и y.

```
Var
 i:integer; X, Y:real;
Begin
 For i:=1 to 11 do
   begin
     X:=0.5 +(i-1)*0.1;
     Y := sin(X);
     Writeln ('X = ', X:3:1,' ':3, 'Y = ', Y:8:5);
   end
End.
```

Пример 4.8. Програма за намиране средния успех на студентите от една група по дадена дисциплина.

Входни данни са:

- брой на студентите в групата;
- оценките на студентите по дисциплината.

Най-напред се въвежда броят BrStud на студентите. След това се въвеждат една по една оценките и се прибавят, към някаква променлива Suma, на която предварително е присвоена нулева стойност (Защо?). Когато приключи въвеждането на оценки, вече е намерена сумата им като стойност на променливата Suma. Остава Suma да се раздели на BrStud, за да се получи средният успех. Въвеждането и сумирането на оценките трябва да стане чрез цикъл, тъй като тези две операции се повтарят за всеки студент. Целесъобразно е да се използва цикъл с параметър, защото е известен броят на повторението на операторите от тялото на цикъла.

```
Var
 i, BrStud, Suma, Ocenka : integer;
 SredenUspeh:real;
Begin
 Write('Въведете броя на студентите :'); Readln(BrStud);
 Suma:=0;
 For i:=1 to BrStud do
   begin
     Write('Въведете оценка :'); Readln(Ocenka);
     Suma:=Suma+Ocenka;
   end;
 SredenUspeh:=Suma/BrStud;
 Writeln('Средният успех е :', SredenUspeh:4:2)
End.
```

4.3.2. Оператор за цикъл с намаляващ параметър

Той има вида:

For *Параметър :=Начална стойност* **downto** *Крайна стойност* **do** *Оператор;*

Тук за *Параметър, Начална стойност, Крайна стойност* и *Оператор* са в сила същите изисквания както по-горе.

Операторът се изпълнява в следната последователност:

а) Изчисляват се Начална стойност на параметъра и Крайна стойност на параметъра;

б) Сравняват се *Начална стойност* и *Крайна стойност* и:

- ако *Начална стойност* < *Крайна стойност*, следва излизане от цикъла;
- ако *Начална стойност* >= *Крайна стойност*, на параметъра се присвоява *Начална стойност*;

в) Изпълнява се *Оператор*;

г) Сравнява се стойността на *Параметър* с *Крайна стойност* и:

- ако *Параметър* <> *Крайна стойност*, то *Параметър* приема предходната дискретна стойност за съответния тип (I:= pred(I)) и следва връщане към инструкция в);
- ако *Параметър* = *Крайна стойност*, то следва излизане от цикъла.

Пример 4.9. Програма, която изчислява стойностите на функцията y=sin(x), за x=1.5, 1.4, ... , 0.5 rad и извежда в две колони стойностите на x и y в намаляващ ред на стойностите на x.

```
Var
 i:integer; X, Y:real;
Begin
 For i:=11 downto 1 do
   begin
     X:=0.5 +(i-1)*0.1;
     Y := sin(X);
     Writeln ('X = ', X:3:1,' ':3, 'Y = ', Y:8:5);
   end
End.
```

4.4. Принудително излизане от цикъл

Нека да си припомним, че от цикъл с предусловие се излиза, когато условието за оставане в цикъла се окаже неудовлетворено, от цикъл с постусловие се излиза, когато условието за излизане от цикъла се окаже удовлетворено и от цикъл с параметър се излиза, когато параметърът достигне крайната стойност.

Последните версии на Турбо Паскал дават възможност да се излезе от всеки от тези цикли, т.е. да се прекратят итерациите (изпълненията на операторите от тялото на цикъла) преди да е удовлетворено съответното условие. Това може да се направи чрез оператора **Break.** Ако в тялото на цикъла има оператор **Break** и се стигне до неговото изпълнение, то следва излизане от цикъла и преминаване към изпълнение на първия оператор след цикъла.

Пример 4.10. Програма за намиране на средно-аритметичното на последователно задавани цели числа до задаването на определено контролно число A или до въвеждане на m числа, използваща оператора **Break**.

```
Var
 i, n, m:integer;
 A,K, S, C:real;
Begin
 Write('Въведете контролното число: '); Readln(A);
```

```
Write('Въведете броя на числата: '); Readln(m);
S:=0; n:=0;
For i:=1 to m do
  begin
    Write('Въведете поредното число:  '); Readln(K);
    If K=A
       then Break;   { При K=A се излиза от цикъла}
    S:=S+K; n:=n+1;
  end;
  If n>0
  then begin
    C:=S/n;
    Writeln('Средно-аритметичното е ', C:10:5)
  end
  else Writeln('Не са въведени числа, различни от контролното.');
 Readln
End.
```

В горната програма при А=К се излиза от цикъла и се продължава с първия оператор след цикъла.

4.5. Прекъсване на текущата итерация от изпълняван цикъл

Нека да си припомним, че при всяка итерация от изпълнението на даден цикъл се изпълняват всички оператори от неговото тяло. Последните версии на Турбо Паскал дават възможност някои от итерациите да бъдат прекъсвани на определено място и да се преминава към следваща итерация. Това се постига чрез оператора **Continue.** Ако в тялото на цикъла има оператор **Continue** и се стигне до неговото изпълнение, то ще последва започване на нова итерация без да е завършена текущата итерация.

Пример 4.11. Програма за намиране на средно-аритметичното на положителните от последователно задавани m цели числа, като се използва операторът **Continue**.

```
Var
 i, n, m:integer;
 K, S, C:real;
Begin
 Write('Въведете броя на числата: '); Readln(m);
 S:=0;   {S - сума на положителните числа}
 n:=0;   { n - брой на положителните числа}
 For i:=1 to m do
   begin
     Write('Въведете поредното число:  '); Readln(K);
     If K<=0 then Continue;    {При K<=0 се прекъсва текущата итерация}
     S:=S+K; n:=n+1;
   end;
   If n>0
   then begin
     C:=S/n;
     Writeln('Средно-аритметичното е ', C:10:5)
   end
   else Writeln('Не са въведени положителни числа.');
 Readln
End.
```

В горната програма при K<=0 се прекъсва текущата итерация, т.е. пропуска се изпълнението на продължението на цикъла (в случая това са операторите S:=S+K и n:=n+1) и започва следваща итерация.

4.6. Вложени цикли

В тялото на всеки от разгледаните три типа цикли може да се съдържа цикъл. Такъв цикъл наричаме вложен цикъл. Той също може да бъде цикъл с предусловие, цикъл с постусловие или цикъл с параметър. Вложеният цикъл обикновено се нарича вътрешен цикъл, а този, в който е вложен - външен цикъл. Вътрешният цикъл може да се явява външен по отношение на друг, вложен в него и т.н.

Пример 4.12. Програма, която извежда всички числа в интервала от M до N, чиито цифри образуват строго нарастваща последователност.

Програмата е с два цикъла:

- външен цикъл, който предлага за проверка всички числа от M до N;
- вътрешен (вложен) цикъл - проверява всяко число, като му отделя цифрите и ги сравнява.

```
Var
  i,K,M,N,c1,c2:integer;
Begin
  Write('Задайте началното число: '); Readln(M);
  Write('Задайте крайното число: '); Readln(N);
  For i:=M to N do          {начало на външния цикъл}
    begin
      K:=i;
      c1:=K mod 10;
      Repeat                        {начало на вътрешния цикъл}
        c2:=c1;
        K:=K div 10;
        c1:=K mod 10;
      until (K=0) or (c1>=c2);      {край на вътрешния цикъл}
      If K=0
        then Writeln(i);
    end;                        {край на външния цикъл}
  Readln
End.
```