Тема 7

Организиране на цикли в Pascal програми

Цикъл в един алгоритъм или програма наричаме група от един или няколко оператора (команди), които се изпълняват многократно.

Цикли със зависим брой изпълнения

При този вид цикли броят на изпълненията на тялото не е известен предварително. Той зависи от резултатите, получавани при всяко изпълнение на тялото на цикъла и изпълнението на зададено условие.

Най-често срещан случай на цикли със зависим брой изпълнения са т.нар. итерационни цикли, при които с увеличаване на броя на изпълненията на тялото на цикъла се получават все по-точни приближения на търсения окончателен резултат. За всяко ново изпълнение на тялото се използват резултатите, получени от предишното изпълнение. Критерий за прекратяване на изчисленията е постигането на предварително зададена точност на резултата

Цикли с независим брой изпълнения

При този вид цикли броят на изпълненията на тялото на цикъла е известен предварително (преди започване на първото изпълнение на цикъла) и не зависи от междинните резултати, получавани в хода на изпълнението на цикъла. Цикълът, управляван от променлива величина (броячен цикъл) е от този вид.

Когато се организират цикли е необходимо да се има предвид структурата на базовата алгоритмична структура цикъл, и да се осигури включване на всички нейни части: променливите, които участват в цикъла да получат начални стойности (инициализация); оператори, които да реализират тялото на цикъла; условие за край на цикъла и подготовка на данните за следващо изпълнение (актуализация).

1. Цикъл с постусловие

При този цикъл условието за край на цикъла (логически израз) е след неговото тяло (повтарящите се команди).

Форматът на оператора за цикъл с постусловие е следния

REPEAT
    ОПЕРАТОР1;
    ОПЕРАТОР2;
    .....
    ОПЕРАТОРN
UNTIL ***логически израз*** ;

където ОПЕРАТОР1, ОПЕРАТОР2, ОПЕРАТОРN образуват тялото на цикъла.

Цикълът действа по следния начин: първо се изпълнява тялото на цикъла; проверява се условието за край и ако то не е изпълнено (логическият израз е **FALSE**), тялото отново се изпълнява и т.н. Цикълът завършва, когато се изпълни условието за е изпълнено край (логическият израз е **TRUE**).

Ако логическият израз остане винаги **FALSE**, цикълът е безкраен.

Цикълът с предусловие се изпълнява поне веднъж.

Пример: Да се изчисли стойността на функцията $y=e^x$, като се използва разлагането на функцията в безкраен ред:

$$y = e^x = 1 + x + x^2/2! + x^3/3! + \ldots + x^n/n! + \ldots$$

$(n! = 1.2.3.4. \ldots .n)$

Въвеждаме следните помощни променливи:

A - променлива, в която ще се получава всеки следващ член на реда;

SUM - променлива, в която ще се натрупва сумата;

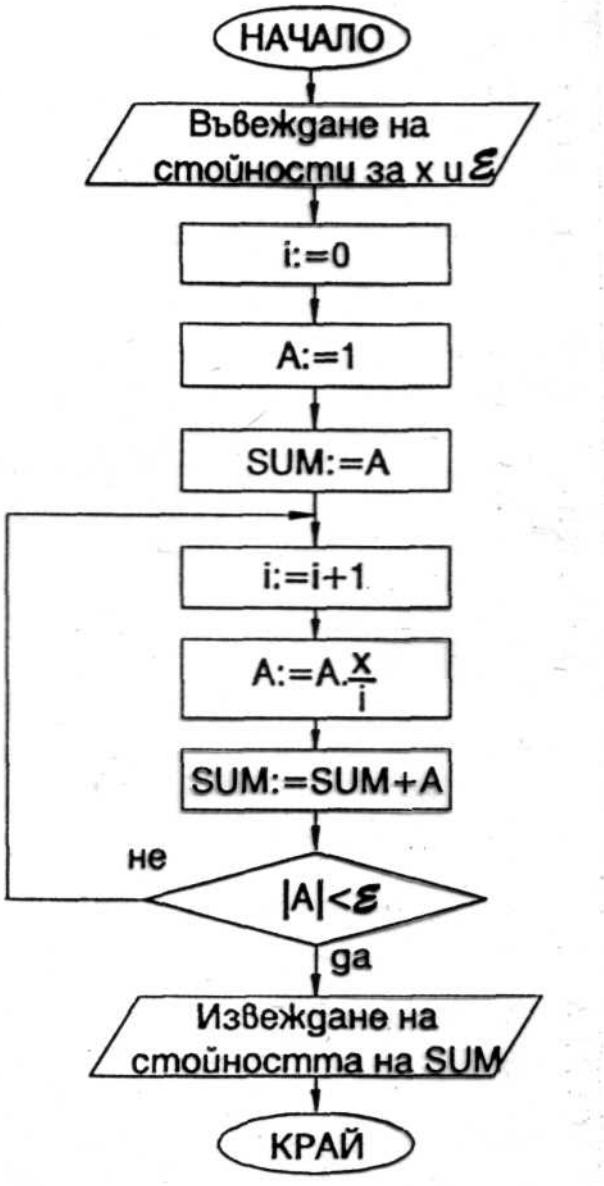
i - спомагателен индекс, който е необходим за изчисляването на стойността на всеки следващ член A.

Всеки следващ член ще се формира чрез умножаване на предходния член с множителя x/i.

Критерий за прекратяване на изчисленията ще бъде условието:

$|A|<\varepsilon$.

Необходимо е да се сравнява именно абсолютната стойност на A с ε защото условието $A<\varepsilon$ ще се окаже изпълнено още при първата проверка за кое да е отрицателно A.



```
PROGRAM EXP;
CONST EPS=0.001; (*точност на изчислението*)
VAR
X: REAL;
A, SUM : REAL;
I: INTEGER;
BEGIN
   WRITE(' Въведете стойност за x ');
   READLN(X);
   I:=0;
   A:=1;
   SUM:=A;
   REPEAT
    I:=I+1;
   A:=A*X/I;
   SUM:=SUM+A
   UNTIL ABS(A)<EPS;
    WRITE(' приближената стойност на функцията е:', SUM:10:3)
  END.
```

2. Цикъл с предусловие

Това е цикъл, който се изпълнява, ако е налице някакво условие за изпълнение (продължаване) на цикъла (логическият израз е **TRUE**).

Форматът на оператора за цикъл с предусловие е следния:

WHILE ***логически израз*** **DO** ОПЕРАТОР;

където ***логически израз*** е условието за изпълнение на цикъла, а ОПЕРАТОР е тялото на цикъла. Когато в тялото на цикъла трябва да се изпълнят няколко команди (оператора), тогава ОПЕРАТОР трябва да бъде съставен оператор.

Цикълът действа по следния начин: първо се проверява условието и ако то е изпълнено, тогава се изпълнява тялото на цикъла, което се състои се от един или няколко оператора; отново се проверява условието и т.н. Цикълът завършва, когато условието повече не е изпълнено (логическият израз стане **FALSE**).

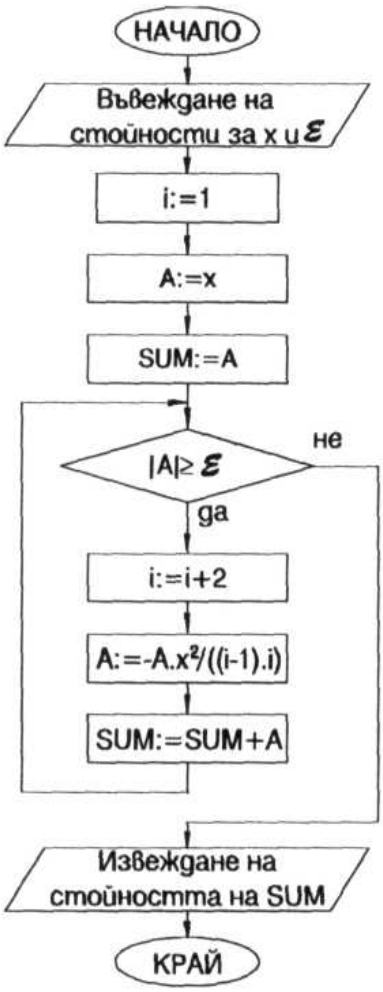
Ако първоначално условието не е изпълнено, тялото се прескача и цикълът не се изпълнява изобщо, а се преминава към следващия оператор в програмата. Това е съществена разлика между цикъл с постусловие и цикъл с предусловие.

Ако при изпълнение на цикъла логическият израз остава постоянно истина (**TRUE**), то цикълът няма да завърши и ще се получи безкраен цикъл. Цикълът е добре организиран когато тялото на цикъла въздейства на логическия израз, така че той в един момент да се промени от **TRUE** на **FALSE** и цикълът да завърши.

Пример: Да се състави програма за изчисляване приближаната стойност на функцията $y = \sin x$, като се използва следната итерационна формула:

$y = \sin x = x - x^3/3! + x^5/5! - \ldots + (-1)^n x^{2n+1}/(2n+1)! + \ldots$

изчислението да се извърши с точност ε.




```
PROGRAM SINUSx;
VAR
 X, EPS: REAL;
 A, SUM: REAL;
I : INTEGER;
BEGIN
    WRITE(' Въведете стойност за x : ');
    READLN(X);
    WRITE(' Въведете стойност за epsilon : ');
    READLN(EPS);
    I := 1;
    A := X;
    SUM := A;
      WHILE ABS(A)> = EPS DO
      BEGIN
           I:=I+2;
           A:=-A*SQR(X)/((I-1)*I);
           SUM:=SUM+A
     END;
WRITELN(' Приближената стойност на SIN x е '  , SUM:8:2);
WRITE(' Стойността на стандартната функция sin(x) е ' , SIN(X):8:2)
END.
```

3. Цикли с брояч (цикли с управляваща променлива)

Цикълът с брояч се нарича още и цикъл с управляваща променлива. Това са цикли, при които предварително знаем колко пъти ще се изпълни тялото на цикъла. При тези цикли се използва целочислена променлива - брояч, която брои изпълненията на цикъла. Операторът за цикъл с брояч е в два варианта – с нарастване и намаляване на брояча.

Форматът на цикъл с нарастване на брояча е следния:

FOR ***променлива-брояч := израз1*** **TO** ***израз2*** **DO** ОПЕРАТОР;

където ***израз1<израз2***

променлива-брояч, израз1, израз2 са от дискретен тип, много често са от тип INTEGER.

израз1 задава началната стойност, която приема брояча преди започването на цикъла;

израз2 задава крайната стойност на брояча, за която цикълът се изпълнява;

ОПЕРАТОР е тялото на цикъла, което се изпълнява многократно. Ако в цикъла трябва да се изпълнят не един, а няколко оператора, тогава ОПЕРАТОР трябва да бъде съставен оператор.

При всяко изпълнение на цикъла броячът нараства с едно. Цикълът завършва, когато стойността на брояча надхвърли стойността на ***израз2***.

Ако ***израз1 > израз2***, тялото на цикъла въобще не се изпълнява, а се преминава към следващата команда след цикъла.

Броячът не може да се променя вътре в тялото на цикъла.

Форматът на цикъл с намаляване на брояча е следния:

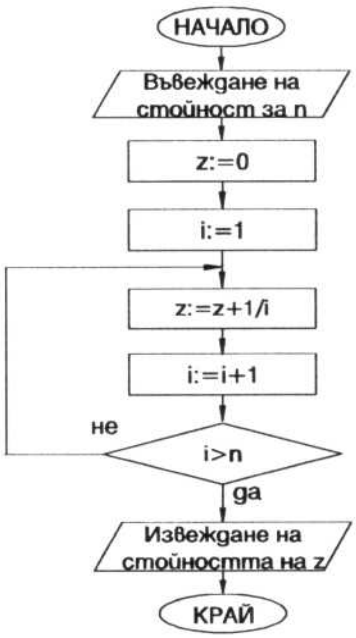
FOR ***променлива-брояч := израз1*** **DOWNTO** ***израз2*** **DO** ОПЕРАТОР;
където ***израз1 > израз2***

Действие на този оператор е сходно с това на предходния оператор с тази разлика, че при всяко изпълнение на цикъла броячът намалява с едно.

Ако ***израз1 < израз2*** тялото на цикъла въобще не се изпълнява, а се преминава към следващата команда след цикъла.

Пример: Да се изчисли стойността на израза

Z = 1 + 1/2 + 1/3 + … + 1/n




```
PROGRAM izr;
VAR
  I, N: INTEGER;
  Z: REAL;
  BEGIN
  READLN(N);
  Z:=0.0;
  FOR I:=1 TO N DO
     Z:=Z+1/I;
  WRITE(' Z = ', Z:10:4)
  END.
```

4. Принудително излизане от цикъл

По-новите версии на Pascal дават възможност да се организира принудително излизане от цикъла преди да е изпълнено условието за край. За тази цел се използва оператор **BREAK**. Ако в тялото на цикъла се включи оператор **BREAK** и се стигне до неговото изпълнение, това води до излизане от цикъла и преминаване към изпълнение на първия оператор след цикъла.

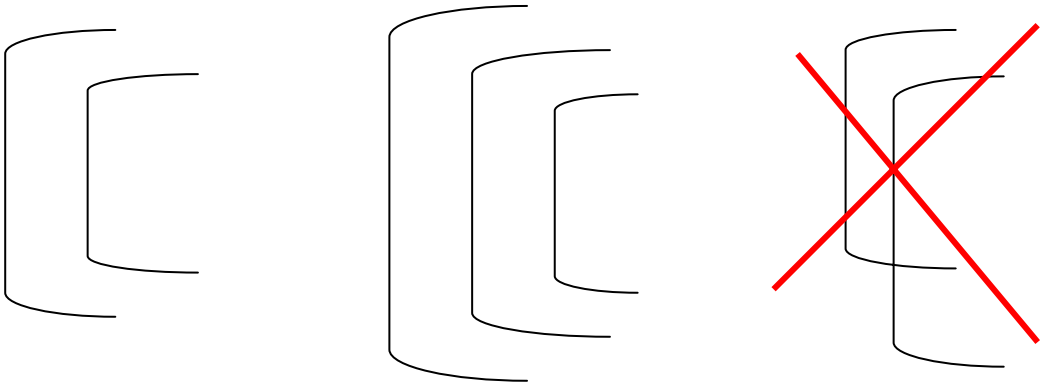
5. Прекъсване на текуща итерация

При нормално изпълнение на цикъла се изпълняват всички оператори от тялото му. По-новите версии на Pascal дават възможност да се прекъсне изпълнението на някоя итерация и да се продължи със следваща. За тази цел се използва оператор **CONTINUE.** При изпълнение на този оператор започва нова итерация без да е завършила текущата.

6. Вложени цикли

Възможно е в тялото на цикъла да се съдържа друг цикъл. Такъв цикъл се нарича вложен или вътрешен цикъл. Цикълът, в тялото на който е вложеният цикъл, се нарича външен цикъл. Може да има няколко нива на влагане на цикли.

Основно правило е, че всеки вложен цикъл трябва да завършва в рамките на външният цикъл. Вложените цикли трябва да имат различни управляващи променливи. От тялото на вътрешния цикъл може принудително да се предаде управлението на оператор от тялото на външния цикъл, но обратното не се допуска.



Пример за вложени цикли: Да се изчисли израза

$$S = \sum_{j=1}^{15} \sum_{k=1}^{10} \sin 2.3k - j\cos.9k$$

```
PROGRAM SUMA1;
VAR
SUM: REAL;
K, J : INTEGER;
BEGIN
SUM:=0;
FOR J:=1 TO 15 DO
FOR K:= 1 TO 10 DO
SUM:=SUM+SIN(2.3*K)-J*COS(1.9*K);
WRITELN((' Сумата е ', SUM:10:3)
END.
```

Въпроси

1. Какви са видовете цикли според броя на изпълнението?
2. Представете общия вид и обяснете начина на действие на оператора за организиране на цикъл с постусловие.
3. Представете общия вид и обяснете начина на действие на оператора за организиране на цикъл с предусловие.
4. Представете общия вид и обяснете начина на действие на оператора за организиране на цикъл с брояч.
5. Кави са правилата за организиране на вложени цикли?