Тема 6

Оператори на езика Pascal. Съставен оператор. Управляващи оператори

При представянето на общия вид на операторите е спазено следното правило:

- Служебните думи и стандартните имена са написани с главни букви и шрифт **Bold – READ;**
- Конкретните данни за даден оператор, които трябва да се напишат в програмата на Pascal са написани с малки букви, **Bold** *Italic*, напр.*: **променлива1, логически израз.***
- ОПЕРАТОР, изписан с главни букви, Regular трябва да се замени с конкретен оператор в програмата на Pascal**.**

1. Оператор за присвояване

Операторът за присвояване служи да зададе стойност на променлива. Той има следния вид:

име на променлива := израз;

Действието му е следното: изчислява се израза и получената стойност се присвоява на променливата. Предишната стойност на променливата се губи. Трябва стойността на израза и променливата да са един и същ тип, както и стойността да е в дефиниционната област на променливата, за да не се предизвика получаване на грешни резултати при изпълнение на програмата.

Y:= a*x+SQR(x);

A:=SIN(x)+COS(x);

Чрез стандартни библиотечни подпрограми на Pascal се осъществява входа към програма на Pascal (въвеждане на данни) и изхода от програма на Pascal (извеждане на данни).

2. Въвеждане на данни от клавиатурата - процедури READLN и READ

READLN(*променлива1, променлива2, ..., променлива n*);
READ(*променлива1, променлива2, ..., променлива n*);

Променливите могат да бъдат от тип: INTERGER, REAL, CHAR, STRING

Действие на тези процедури е следното: програмата спира изпълнението си и чака да се въведат от клавиатурата N-те стойности, разделени с интервал. Натиска се клавиша Enter и стойностите се присвояват на съответните променливи след което продължава изпълнението на програмата. При въвеждане на стойностите те могат да са написани на един или на повече реда. При това трябва да се има предвид, че **READLN** прочита стойностите от целия ред и следващ оператор за въвеждане ще започне да чете от нов ред.

Необходимо е въведените стойности да са съобразени с типа на променливите; в противен случай процедурата дава грешка и програмата се прекъсва. Ако стойностите са

извън дефиниционната област на съответните променливи, възниква препълване и програмата не работи вярно.

Пример: ако

a,b,c: REAL;

d: Integer;

f, p: char;

READ(***a,b,c***);

Когато в програмата се срещне този ред, изпълнението на програмата спира, на екрана се вижда маркера и потребителя трябва да въведе стойности за трите променливи. Ако се въведат следните стойности,

1.2 3.4 4.8 натиска се клавиш Enter

В резултат променливата ***a*** ще получи стойност 1.2, променливата ***b*** ще получи стойност 3.4, променливата ***c*** ще получи стойност 4.8.

READ(*d*);

4 натиска се клавиш Enter

В резултат променливата ***д*** ще получи стойност 4

READ(*f, p*);

RS натиска се клавиш Enter

В резултат променливата ***f*** ще получи стойност R, променливата ***p*** ще получи стойност S.

3. Извеждане на данни към екрана - процедури WRITE и WRITELN

WRITE(*списък на променливи, изрази, символни константи*);
WRITELN(*списък на променливи, изрази, символни константи*);
WRITELN;

Променливите могат да бъдат от тип INTERGER, REAL, CHAR, STRING, BOOLEAN

Тези процедури извеждат стойностите на променливите и изразите една след друга на екрана. Разликата между двете е, че при **WRITE** курсора остава на същия ред на екрана след последната изведена стойност, а при **WRITELN** след извеждане на стойностите курсорът преминава в началото на нов ред. **WRITELN** може да се използва и без параметри. В този случай курсора се премества в началото на нов ред. Този вариант се използва, за да се организира вмъкването на празни редове за по-добро и подреждане на изведените резултати

И при двете процедури програмистът трябва да се погрижи да отдели стойностите на изразите една от друга, иначе те ще се слеят. Това може да се постигне или като между тях се изведе по един символ интервал ' ', или ако към изразите се добави формат за извеждане.

Когато не е зададен формат за извеждане, реалните стойности се извеждат в т.нар. научен формат или с плаваща точка; например 1.5 ще се изведе като 0.150000000E+01.

Форматът за извеждане има вида:

I:n

R:n:m

CH:n

Където*:*
I - целочислен израз (променлива)
R - реален израз (променлива)
CH – символен израз (променлив)
n - задава броя на позициите, в които ще се изведе стойността на израза
m - задава броя на цифрите след десетичната точка (точността на числото).

WRITELN(a:5:2,b:6:2,c:8:3**);**
__1.20___3.40_______4.800

За по-голяма яснота при извеждане на резултатите се препоръчва при извеждането им да се включват и пояснителни текстове. Това се постига като в оператор **WRITE (WRITELN)** се включи подходяща символна константа.

WRITELN('a=', a:5:2);
WRITELN('b=', b:5:2);
WRITELN('c=', c:5:2);

a=_1.20
b=_3.40
c=_4.80

WRITELN(d:3)
WRITELN ('стойността на променливата d e ', d:3)
__4
стойността на променливата d e ___4

WRITELN ('периметърът на триъгълника е ', (a+b+c):8:2, ' см ');

При въвеждане на стойности за променливи величини е добре преди оператор READ да се постави оператор WRITE, който да изведе на екрана съобщение, което показва какви стойности да се въведат.

WRITELN('въведете старна a ');
READ(a);

4. Съставен оператор

Съставеният оператор служи да групира няколко оператора в един. Нарича се още операторни скоби. Той може да бъде поставен на всяко място в програмата, където е допустим само един оператор, а в същото време се налага да се изпълни не един, а няколко оператора (например в оператора **IF** след **THEN** и след **ELSE** може да има само по един оператор). Всъщност тялото на цялата програма е също един съставен оператор. Синтаксисът на съставният оператор е следния:

BEGIN
    ОПЕРАТОР 1;
    ОПЕРАТОР 2;
    ……
    ОПЕРАТОР N
END;

5. Коментар в програмата

На произволни места в текста на програмата може да се вложи свободен текст, наречен коментар, който да дава пояснение за програмиста, който чете програмата. Коментарът не се обработва от и не влиза в кода на изпълнимата програма. Коментарът може да се пише и на кирилица. Задава се по два равностойни начина:

(*коментар*)
{коментар}

6. Програми с линейна структура

Разгледаните до сега оператори са достатъчни за написване на програми с линеен алгоритъм – линейни програми. Те обикновено извършват въвеждане на стойности за входни данни, пресмятане на стойности и присвояването им на зададени променливи (изходни променливи, резултати) и извеждане на получените резултати на екрана. Разгледаните примери са от литература [1] и [3].

За примера за топчето, разгледан при тема Алгоритми, програмата на Pascal е следната:

```
PROGRAM P1;
VAR
v, t, s: REAL;
BEGIN
    WRITELN(' Въведете стойност за скоростта v ');
    READ(v);
    t:= v/9.81;
    s:= v*t – (9.81*SQR(t))/2;
    WRITELN(' Време t= ', t );
    WRITELN(' Разстояние s= ' , s)
END.
```

7. Условен (Логически) оператор IF

Чрез оператор **IF** работата на програмата може да се разклони в зависимост от изпълнението на някакво логическо условие. Той съществува в два варианта: непълен и пълен.

Непълният оператор IF има следния общ вид:

IF ***логически израз*** **THEN** ОПЕРАТОР;

Действието на оператора в този случай е следното: първо се изчислява логическия израз и ако той е истина (**TRUE**) се изпълнява оператора след ключовата дума **THEN** и се продължава нататък; ако изразът е лъжа (**FALSE**), оператора след **THEN** се прескача и се продължава с изпълнението на следващия оператор в програмата.

Пълният логически оператор има следния общ вид:

IF ***логически израз*** **THEN** ОПЕРАТОР1

ELSE ОПЕРАТОР2;

Действието на оператора в този случай е следното: първо се изчислява ***логическия израз***; ако той е истина (**TRUE**) се изпълнява ОПЕРАТОР1 след ключовата дума **THEN** и се продължава с оператора след **IF**, като се прескача ОПЕРАТОР2; ако логическият израз е лъжа (**FALSE),** се изпълнява ОПЕРАТОР2 след ключовата дума **ELSE** и се продължава с оператора след **IF**, като се прескача ОПЕРАТОР1 след **THEN.**

Синтаксисът на оператор **IF** допуска само един оператор след **THEN** и един оператор след **ELSE**. Ето защо в случаите, когато след **THEN** и след **ELSE** се налага да се изпълнят няколко команди, ОПЕРАТОР1 и ОПЕРАТОР2 трябва да бъдат съставни оператори.

8. Оператор за избор (за многостранно разклонение).

С помощта на логическия оператор **IF** и съставен оператор може, според някакво условие, да се разклони алгоритъма на два клона. Често обаче се налага разклоняването на много клонове, според по-сложни условия и/или избор измежду много ситуации (например избор от меню). Вместо да се правят каскади от вложени логически оператори, се използва оператор за избор. А този оператор се организират N+1 разклонения в програмата, в зависимост от стойността на израз (променлива), наречен селектор. Операторът има следния общ вид:

CASE ***селектор*** **OF**
списък от константи1: ОПЕРАТОР1;
списък от константи2: ОПЕРАТОР2;
…
списък от константиN: ОПЕРАТОРN;
ELSE ОПЕРАТОРN+1
END;

ELSE ОПЕРАТОРN+1 не е задължителна част от оператора.

Селектор е израз или променлива от дискретен тип (целочислен, символен, логически, изброен, ограничен).

Списък от константи – последователност от неповтарящи се константи от същия тип като селектора, които са разделени със запетаи.

Действие на оператор **CASE**:

Първо се изчислява стойността на селектора.

Тази стойност се търси последователно в изброените константи в ***списък от константи1, списък от константи2,…. списък от константиN.*** Ако има константа, която съвпада със стойността на селектора, то се изпълнява оператора срещу ***списък от константи***, в която се намира тази константа. След това изпълнението продължава със следващия в програмата оператор.

Ако стойността на селектора не съвпада с нито една от изброените константи в ***списък от константи***, то се изпълнява оператора след **ELSE** и след това изпълнението продължава със следващия в програмата оператор.

Ако стойността на селектора не съвпада с нито една от изброените константи в ***списък от константи*** и в оператора няма **ELSE** изпълнението продължава със следващия в програмата оператор.

9. Оператор за безусловен преход GOTO

Операторът за безусловен преход се използва, когато в програмата се изисква да се наруши естествения ред на изпълнение на операторите и е необходимо да се изпълни не следващия по ред оператор, а някой друг оператор, който е задължително означен с етикет (LABEL).

Общ вид на оператора:

GOTO ***етикет***;

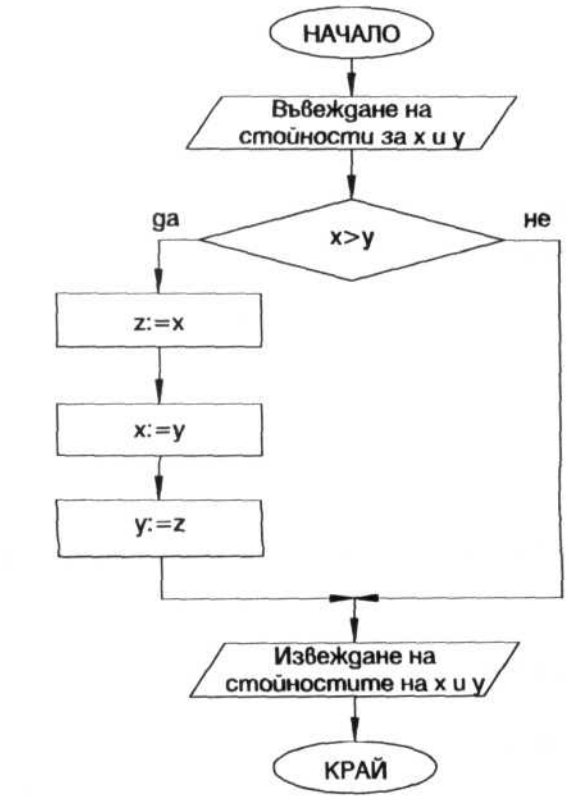
Задължително е в декларативната част на програмата ***етикет*** да се декларира с LABEL.

10. Програми с разклонена структура

Програма за примера с базова алгоритмична структура алтернатива.

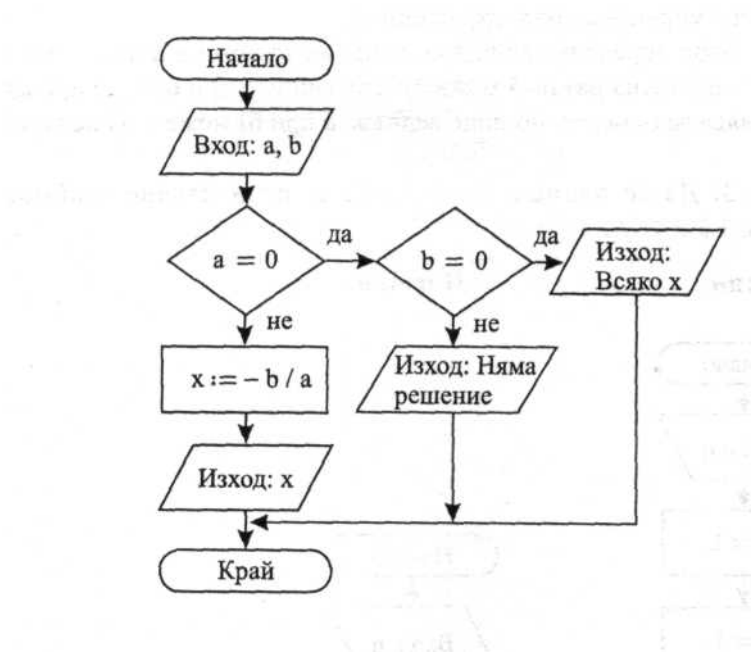
```
PROGRAM P2;
VAR
   R, X, Y: REAL;
BEGIN
   WRITE(' Въведете стойности за  R и X: ');
   READ(R, X);
   IF ABS(X) > R    THEN Y: = 0.0
                         ELSE Y: = SQRT(SQR(R) - SQR(X));
   WRITELN('За X = ', X:9:4,'  Y = ', Y:9:4)
 END.
```

Непълна (кратка) форма на IF и използване на съставен оператор: програма за размяна на стойностите на две числа x и y само ако е изпълнено условието x>y.




```
PROGRAM P3;
(*размяна на стойности*)
VAR
X,Y,Z: REAL;
BEGIN
WRITE(' Въведете стойности за X и Y ');
READLN(X,Y);
IF X>Y  THEN
                BEGIN
                   Z:=X;
                   X:=Y;
                   Y:=Z
                END;
WRITELN (X:8:3, Y:8:3)
END.
```

Вложени оператори IF и необходимост от използване на съставен оператор: програма за решаване на линейно уравнение от вида $ax+b=0$.




```
Program P4;
(* линейно уравнение *)
Var
   a, b, x: Real;
Begin
   Write(' Въведете стойности за a  и  b:');
   Readln(a, b);
   If a = 0  Then   If b = 0    Then Write ('  Всяко  x е  решение на ' )
                                     Else Write (' Уравнението няма решение '  )
               Else
                 Begin
                 x:= - b/a;
                 Write(' Решението на уравнението е X=',x:8:3)
                 End
End.
```

Вложени оператори IF: програма за определяне в кой квадрант лежи точка по зададени координати.

```
PROGRAM P5;
{kvadrant}
VAR
x, y: REAL;
k: INTEGER;
 BEGIN
   WRITE ('X='); READLN(x);
   WRITE('Y='); READLN(y);
    IF x*y>0 THEN IF x>0 THEN k:=l
                                    ELSE k:=3
                 ELSE IF x<0 THEN k:=2
                                    ELSE k:=4;
   WRITELN (' Точка с координати x=' , x:6:2, ' y=' , y:6:2,  '  лежи в квадрант  ', k:2 )
   END.
```

Използване на оператор за многостранно разклонение CASE: програма, която изписва с думи оценка, която е въведена с цифра.

```
PROGRAM P6:
   {изписване на оценка с думи}
VAR
    N: Integer;
BEGIN
    WRITELN;
    WRITE('Въведете оценка ');
    READLN(N);
    CASE N OF
     2:WRITELN(' Слаб ' ) ;
     3:WRITELN(' Среден ' ) ;
     4:WRITELN(' Добър ' ) ;
     5:WRITELN(' Мн. добър ' ) ;
     6:WRITELN(' Отличен ' )
     ELSE
        WRITELN(' не е въведена валидна оценка  ' )
END.
```

Въпроси

1. Представете общия вид на оператора за присвояване, обяснете действието му и изискванията при използването му.
2. Представете общия вид на процедурите за въвеждане на данни, обяснете действието им и изискванията при използването им.
3. Представете общия вид на процедурите за извеждане на данни, обяснете действието им и изискванията при използването им.
4. Представете общия вид и предназначението на съставния оператор.
5. какво е предназначението и ка се представя коментар в програмата?
6. Какво е предназначението на условния оператор? Представете общият му вид и обяснете начина на действие.
7. Какво е предназначението на оператора за избор? Представете общия му вид и обяснете начина на действие.