Iteration: the for statement

Run this program:

```
program ShowFor (output); {Demonstrates the for statement}
…
var LoopCount: integer; Character: char;
begin
        for LoopCount := 1 to 5 do
                write ('Loop ', LoopCount,' ');
        writeln;
        for Character := 'Z' downto 'V' do
                write(Character);
        writeln;
…
end.
```

Write down what you learn about the **for** statement from this program

Example of nested for statements

In order to see what the following piece of code does, you will need to put it into a program with a suitable var declaration. Type in and run the program, then write down what you have learnt about the **for** statement

```
        for Outer := 1 to 3 do
                for Inner := 6 to 7 do
                begin
                        write (Inner ); write (Outer)
                end;
```

The **for** statement is one of Pascal's *loop* or *iterative* structures. Its form is

> **for** *control-variable := initial-value* **to** *final-value* **do**
> *action;*

The reserved word **downto** in place of **to** reverses the counting process.

The control variable must not be of type *real*: It has to be ordinal. It must not have its value altered inside the **for** statement.

After the **for** statement has finished, the value of the control variable is undefined, so do not assume that its value is the final value.

initial-value and *final-value* may be expressions to match the type of *control-variable* .

If the value of *initial-value* is greater than the value of *final-value* then the action in the **for** statement will not be executed at all.

If the values of *initial-value* and *final-value* are equal then the action in the **for** statement will be executed just once.

Assignment to the initial and final values inside the **for** statement have no effect on the **for** statement.

Exercises

21. Write a program to form the sum of the squares of the first n natural numbers, where the user supplies the value of n, then compare with the formula
    n(n+1)(2n+1)/6
and output the message
    "Formula and series agree"
where appropriate

22. Write a program to read in a number *n*, then to read in *n* of real numbers.
Now output the maximum, minimum and average of those *n* real numbers.

23.* Write a program to read in two letters, then write out all the letters between them, excluding the two given letters. Does it matter if the input is upper or lower case?

24.* Write a program to form the sum of the cubes of the first n natural numbers, and compare with the square of the sum of the first n natural numbers. The user is to be able to choose 3 values of n to investigate. Do not use more loops than are necessary.

Output and formatting

Type in and run the following:

```
program Format;
        {Demonstrates simple input and output}
…
var
   Year: integer;

begin
   writeln ('HERE':10, 'IS A':10, 'RHYME':10);
   Year := 1992;
   writeln ('In', Year:1, ', Branson sailed the sky blue');
   writeln ('In', Year:5, ', Branson sailed the sky blue');
   writeln ('In', Year, ', Branson sailed the sky blue');
   writeln('Try printing the same number with different formats');
   writeln (100/8:10:1, 100/8:10:2,100/8:10:5);
end.
```

Write down what do you notice about the output. What is the effect of a number following a colon in the parameter of writeln? And a second number?

Constants

In the previous program we set Year to the value 1992. If Richard Branson's balloon sailed through the sky only in 1992 and no other year we might have declared Year as a constant as below.

```
program Format ;
…
         {Demonstrates simple input and output}
const
   Year= 1992;          {note the = for setting the value of a constant}
begin
   writeln ('HERE':10, 'IS A':10, 'RHYME':10);
   writeln ('In', Year:1, ', Branson sailed the sky blue');
   writeln ('In', Year:5, ', Branson sailed the sky blue');
   writeln ('In', Year, ', Branson sailed the sky blue');
   writeln('Try printing the same number with different formats');
   writeln (100/8:10:1, 100/8:10:2,100/8:10:5);
end.
```

Constant definitions must come just after the program heading.

program heading;
constant definitions;
variable definitions;
statement part.

More examples of constants

```
const
  LastLetter = 'Z';
  MyName = 'Farouk';
  FiveBlanks = '     ';
  Half = 0.5;
  FirstName = MyName; {refers to a previously defined constant}
  Age = 17;
   TaxRate = 0.3;
```

A constant can be any number or string of characters contained in single quotes.

Why use constants?
Using constants helps when we change programs later. Suppose we had a 2000-line program which calculates employees pay and tax. It works fine until the tax rate changes, we don't want to have to go through the program and find every occurrence of the old tax rate but if it is sitting at the top of our program as a constant it is easily changed.

Exercises

25. Write a program with formatted output that will read in two real numbers, x and y and output the messages

    The nearest integer below x is ....
    The nearest integer above y is ....

Remember, there are Pascal functions to help you do this. Look back at earlier notes Test carefully to ensure it works for all values.

26. Write a program to tabulate integers, squares, square roots and reciprocals of numbers 1 to n, where the user can enter the value of n.
27. Write a program to produce the words of this song, paying attention to formatting numbers on output, and using constants where appropriate:

10 green bottles hanging on the wall
10 green bottles hanging on the wall
And if 1 green bottle should accidentally fall
There'd be 9 green bottles hanging on the wall

9 green bottles hanging on the wall
9 green bottles hanging on the wall
And if 1 green bottle should accidentally fall
There'd be 8 green bottles hanging on the wall
...
2 green bottles hanging on the wall
And if 1 green bottle should accidentally fall
There'd be 1 green bottle hanging on the wall
...
And if 1 green bottle should accidentally fall
There'd be no green bottles hanging on the wall

Iteration : the while statement

The **while** statement is a non-deterministic loop or iterative structure. It is used when the number of iterations is not known in advance

Examples

```
program CountLetters;
...

{This program counts characters while character is not '}'}
 var
  Letter : char; Count : integer;
begin
{ initialise variables}
 read(Letter);
 Count := 0;
 while Letter <> '}' do
  begin
   read(Letter);
   Count := Count + 1
  end;
 write('number of letters excluding } =');
 writeln(Count)
end.

program GeomProg;
...
{ to approximate to a GP with initial term 1 }
 const epsilon = 0.001;
 var TotalSoFar, ThisTerm, R : real; TooFarOut : boolean;
begin
 writeln('Enter value of common ratio '); readln(R); write(R);
 if abs(R) > 1 then
  writeln(' this series will not converge')
 else
 begin
        {initialise}
        ThisTerm := 1; TotalSoFar := 0; TooFarOut := true;

        while TooFarOut do
        begin
                TotalSoFar := TotalSoFar + ThisTerm;
                ThisTerm := ThisTerm * R;
                TooFarOut := abs(ThisTerm) > epsilon
        end;
        writeln; write('    approx. value is '); Write(TotalSoFar)
 end
end.
```

General form is

while *boolean expression* **do** *action*

while is a non-deterministic loop or iteration - *action* as long as *boolean expression* holds true. If the condition is false initially, the action will not take place at all. Action will normally consist of a compound statement , so it will be delimited by **begin** and **end.**

- **while** loops can be nested, just like **for** loops.
- **while**, **for** and **if** can be nested
- take care that **while** loops do not become infinite

28. Adapt the GP program given so that the user can specify the initial term.

29. Write a program to read integers until 999 is found. Average the numbers, excluding the 999.
eg if the input is 30 40 10 0 999
then the required average is 20.0 (it should be a real, and format the output).

30. Write a program to read a sentence (ending with a full stop) and count the vowels in it, that is, the number of A,E,I O and U s.
31*. Adapt the answer to 33 so that both upper and lower case vowels are counted together.
32*. A number is divisible by 3 if the sum of its digits is divisible by 3. Write a program to test whether a number of any size (even if larger than the largest allowable integer), is divisible by 3 (Treat the number as a series of characters, and convert them individually to numbers).

By now you should know the meaning of the following terms

identifier	reserved words	iteration
variable	assignment	selection
constant	ordinal	

You should be able to use

Arithmetic operators integer, real, char, boolean
Boolean operators types
formatted output

and these constructions

if for
while

Revision exercises

Write a program which reads an integer value between 1 and 99 and outputs the following messages:
the word 'buzz' if the integer contains the digit 7
otherwise the word 'buzz-buzz' if the integer is a multiple of 7
otherwise the number itself

Read an integer n, if it is less than zero, or greater than 9, print out an error message; if it is between those limits, calculate and print n!

Write a program to calculate the sum of the first n terms of an arithmetic progression by addition. How can you test your program?

Write a program to read values of u and v, and then calculate focal length. This is to be repeated several times : the user should be able to specify how many. (In a Computer Science exam, you would not be required to know this formula, but for this exercise look it up in your Physics text book)

Write a program to read text until a * is found. Count the number of letters before M in the alphabet, the number of M's, and the number of letters after M.