

Chapter 6

IF.....THEN.....ELSE

- 6.1 Conditional Statements
- 6.2 IfThen.....Else
- 6.3 Boolean Operators

6.1 Conditional Statements

“If it rains I’ll go to the cinema.”

“If I earn over Lm 100.00 this month I’ll buy you a nice present.”

“If you have a University degree you can apply for the job.”

We associate sentences like these with the human faculty of reasoning. The sentence starts with some sort of condition and what follows this condition will happen only if the condition is true. Structures like these exist in programming languages and it is because of them that sometimes, wrongly of course, computers are said to have an intelligence like that of human beings. Such structures are known as **CONDITIONAL** statements.

Before we consider some examples of conditional statements let us have a look at the operators which can be used to form conditions. You will recall the arithmetic operators: **+** **-** ***** **/** **div** **mod**

Operators that are used to form conditions are known as **relational operators**.

Relational Operator	Meaning
=	is equal to
<	is less than
>	is greater than
<=	is less than or equal to
>=	is greater than or equal to
<>	is not equal to

Conditional statements are formed by combining the standard data types, integer, real, string, char and Boolean with the above relational operators. The following are examples of simple conditional or ‘**Boolean**’ expressions:

IF x > 50 THEN	IF maximum < 100 THEN
IF letter <> 'q' THEN	IF x >= 75 THEN
IF count > (x * 24) THEN	IF (x + y) = (p - q) THEN

In Pascal these **Boolean** expressions are used in the following way:

```
IF x > y THEN  
BEGIN  
  instruction 1;  
  instruction 2;  
  instruction 3;  
  instruction 4;  
END;
```

{The block of instructions between the **BEGIN** and **END** will only be executed if the condition that **x > y** is true. If the condition is not true the block is missed and the control of the program passes to the instructions which follow the block}

PROG 61: Specification

A program which asks the user to enter a number between 1 and 100, which detects whether the number is odd or even and which outputs a statement to this effect. The program gives the user a chance to use it as many times as he likes without having to re-run it.

```
PROGRAM Conditions;           { Introduces the conditional statement }
USES CRT;
LABEL 100;                    {The label declaration comes after USES CRT}
VAR
  Num   : INTEGER;
  Ans   : CHAR;
BEGIN
  100: CLRSCR;                 {100 is the label to which control returns with the GOTO statement}
  WRITELN ('Enter a number between 1 and 100');
  READLN (Num);
  IF Num MOD 2 > 0 THEN       {if a number is divided by 2 and the remainder > 0 then the number
                              must be odd}
  BEGIN
    WRITELN ('The number you entered is odd');
  END;
  IF Num MOD 2 = 0 THEN
  BEGIN
    WRITELN ('The number is even');
  END;
  WRITELN ('Would you like to try again? write y or n');
  READLN (Ans);
  IF Ans = 'y' THEN GOTO 100;
END.
```

Notes of PROG61

- The first thing you will notice about this program is the new Pascal word **LABEL** and the **GOTO** statement right at the end of the program. Normally program instructions are **executed** - obeyed - by the computer in the order in which they are written, i.e. from top to bottom. There may be occasions, however, when the programmer needs to disrupt this normal flow and jump to some other part of the program. Now in order to do this he must tell the computer where to resume the flow of the instructions and hence the need for a **label**. A label then is a storage location which holds in it a number or a string which marks a point for a **GOTO** statement to refer to in the program. In **PROG 61** the label is a number - 100 - but it could equally be a string such as 'StartAgain'.
- Notice the simple technique used in the last three lines of the program. These instructions enable the user to re-run the program as many times as he likes without having to leave it.

Note that a label must be declared after the USES CRT declaration and before CONST and VAR .

GOTO statements are to be avoided like the plague. Too many of them result in what is known as spaghetti programming, which will be very difficult to understand. Fortunately Pascal provides the means to avoid GOTO's as we shall see later.

Now load **PROG 61** and run it a few times with different sets of data.

Exercise 6.1

Write a program which asks the user to enter his age in years - as an integer. If the age entered is ≥ 50 the program will output the message: "You are too old for the job." If the age is < 50 the message will be: "You will be considered for the job." The program asks the user if he wants another go.

6.2 IfThen.....Else

"If it rains I'll go to the cinema otherwise I'll go for a walk"

This situation is catered for by the **if..... then.....else** structure

```
IF x > y THEN
BEGIN
  instruction 1;      {If condition is true execute the instructions in this block}
  instruction 2;      {and ignore the instructions in the block that follows the }
  instruction 3;      {Pascal word ELSE}
END ELSE
BEGIN
  instruction          {Otherwise skip first block and execute the instructions in the
  instruction          second block before continuing with the program}
  instruction
END;
```

Notice that there is no ; when ELSE follows END

Note the importance of the block structure in conditional statements. The condition only applies to the statements within the block.