Lesson 6: Logical Operators, Relational Operators and Boolean Expressions

In a nutshell, this lesson will cover:

- What are logical operators?
- The Logical Operators
  1. AND
  2. OR
  3. NOT

- Examples of logical operators:
  1. AND
  2. OR
  3. NOT

- Relational operators
- Boolean Expressions

What are Logical Operators?

Logical operators are used in conditional expressions which will eventually be evaluated and return *true* or *false*. True and false are boolean values. A Boolean expression involves logical operators which are used to combine conditional expressions together. They are also used in assignments (an example of this would be shown later). Such operators consist of simple logical operators, such as '**Not**' or '**And**'. They are used in between conditional expressions.

For example:

```
If (x = 0) AND (a = 2) Then ...
```

The Logical Operators

There are three types of logical operators, each of which are concerned with conditional expressions. These are:

- **AND**
- **OR**
- **NOT**

These logical operators have a different effect on the conditional expressions. Let's see how each of the logical operator behaves on the following conditional expressions...

The 'AND' Logical Operator

```
If (admin = 'admin') AND (password = 'pass') Then
        Writeln('Login accepted. Welcome Administrator!');
```

Expression 1	*Expression 2*	*AND (result)*
true	true	***true***
false	true	false
true	false	false
false	false	false

You can see very clearly from this table that if expression 1 **and** expression 2 are both true (i.e. the user inputs 'admin' and 'pass' into variables 'admin' and 'password' respectively), the message will be displayed. Above is a table showing the possible combinations. So, from the above table, one can conclude that for a logical operation such as AND, to give out a true result, both conditional expressions should be true.

The 'OR' Logical Operator

```
If (month = 'July') OR (month = 'August') Then
        Writeln('Month is either July or August.');
```

Expression 1	*Expression 2*	*OR (result)*
true	true	***true***
false	true	***true***
true	false	***true***
false	false	false

Either expression 1 **or** expression 2 should be true to display the message. If for example expression 1 is true and any other conditional expressions are false, the result is true! Above is the *truth table* showing all the possible combinations. So, from the above table, one can conclude that for a logical operation such as OR, to give out a true result, only one of the conditional expressions should be true.

The 'NOT' Logical Operator

Not is different from the two logical operators. It only accepts one input and is known as the 'inverter'. If for example the result of two conditional expressions is true, the 'not' operator will invert the result to false! So, the purpose of the logical operator, 'not', is to invert the input. The simple truth table for the not operator is as follows:

Input	*NOT (result)*
true	*false*
false	*true*

Example of the 'AND' Operator

```
Program Lesson6_Program1;
Uses Crt;
Var n1, n2 : string;

Begin
          Writeln('Enter two numbers: (''0'' & ''0'' to exit)');
          Repeat
                    Write('No.1: ');
                    Readln(n1);
                    Write('No.2: ');
                    Readln(n2);
                    If (n1 = '0') AND (n2 = '0') Then Halt(0);
          Until (n1 = '0') AND (n2 = '0');
End.
```

Example of the 'OR' Operator

```
Program Lesson6_Program2;
Uses Crt;
Var n1, n2 : String;

Begin
          Writeln('Enter two numbers: (''1'' & ''2'' to exit)');
          Repeat
                    Write('No.1: ');
                    Readln(n1);
                    Write('No.2: ');
                    Readln(n2);
                    If (n1 = '1') OR (n2 = '2') Then Halt;
          Until (n1 = '1') OR (n2 = '2');
End.
```

Example of the 'NOT' Operator

```
Program Lesson6_Program3;
Uses Crt;
Var n1 : String;

Begin
          Writeln('Enter two numbers: (any number except 0 to exit)');
          Repeat
                    Write('No.1: ');
                    Readln(n1);
                    If not(n1 = '0') Then Halt; { can also be ... if n1 <> '0' Then Halt; }
          Until not(n1 = '0'); { similar to n1 <> '0' }
End.
```

Relational Operators

Relational operators are used to compare values in terms of equality or inequality. We have already encountered programs where the equals '=' operator have been used to test for equality within If-Statements. In other instances, we might have a variable for which we would like to know if its greater than 100 (> 100), or less than or equal to 0 (<= 0). In such cases we would want to use the 'greater than' and the 'less than or equal to' operators.

Consider the following example:

```
Var age : Integer;

Begin
          Repeat
                    Write('Enter age (1 - 100): ');
                    Readln(age);
                    If (age < 1) Then
                              Writeln('Age cannot be less than 1...')
                    Else If (age > 100) Then
                              Writeln('Age cannot be greater than 100...');
          Until (age > 0) AND (age <= 100); { loop ends when  }
End.
```

Boolean Expressions

Boolean expressions are expressions which evaluate to either '**true**' or '**false**'. The data type of a boolean variable in Pascal is called a **Boolean** and stores either *true* or *false*. A boolean value is a value that can also be stored just like any other data type.

```
Var
          bool : Boolean;
          A, B : Integer;

Begin
          A := 10;
          B := 20;
          bool := False;
          bool := (A = 10) OR (B = 10);
          Writeln(bool); { outputs TRUE }
          bool := (A = 10) AND (B = 10);
          Writeln(bool); { outputs FALSE }
End.
```

Example use of a boolean variable:

```
Program Lesson6_Program4;
Var quit : Boolean;
    a    : String;

Begin
          Repeat
                    Writeln('Type ''exit'' to quit:');
                    Readln(a);
                    If a = 'exit' Then
                              quit := True;
          Until quit = True;
End.
```