The Pascal Programming Language

The Pascal programming language was created by Niklaus Wirth in 1968. It was named after Blaise Pascal, a famous French Mathematician. Some of benefits of Pascal include:

- It is well-structured
- It is easy to implement
- The syntax is easy to lean and follow
- It encourages the programmer to adopt a disciplined approach to programming

OPERATORS (Symbols used for performing calculations or making comparisons)

Type	Name	Symbol	Example
Arithmetic	**Addition**	+	**5+2=7**
Arithmetic	**Subtraction**	-	**5-2 = 3**
Arithmetic	**Multiplication**	*	**5*2 = 10**
Arithmetic	**Real Division**	/	**5/2 = 2.5**
Arithmetic	**Integer Division**	**div**	**5 div 2 = 2**
Arithmetic	**Modulo**	**mod**	**5 mod 2 = 1**
Relational	**Equal to**	=	**10 = (5*2)**
Relational	**Not Equal to**	<>	**10 <> (5+2)**
Relational	**Less Than**	<	**x < y**
Relational	**Less Than or Equal To**	<=	**x <= y**
Relational	**More Than**	>	**x > y**
Relational	**More Than or Equal To**	>=	**x >= y**
Logical	**And**	**If both expressions evaluate to True the result is True.**	
Logical	**Or**	**If either expression evaluates to True the result is True.**	
Logical	**Not**	**Negates a boolean expression. (True becomes False**	

DATA TYPES AND DECLARATIONS

Data Type	**Examples**
Integer	36
Real	Floating Point Values (e.g. 758.69)
Char	1 character (A)
String	John (Up to 255 characters)
Boolean	True or False

Declaring Variables

Format:

```
var
    Variable Name: Data Type;
```

Examples:

```
var
          sum: integer;
          x,y: real;
          name: string;
          Grade: char;
```

NB. A semicolon (;) is used to separate statements.

Constants

Constants are like variables except that their values cannot change.

Declaring Constants

Format:

```
const
        Constant Name = Value;
```

Examples:

```
const
        Pi = 3.14;
        VAT = 17.5;
```

INPUT AND OUTPUT OF DATA

Input

The two basic Input functions are **Read** and **Readln**, which are used to read data from the keyboard.

Format:

```
Read (item1, item2...)

Readln(item1, item2...)
```

NB.

The Read function reads one or more values into one or more variables. The Readln function does the same thing as the Read function and then skips to the beginning of the next line in the file.

Examples:

```
Read(x,y);

Readln(name, score);
```

Output

The two basic output functions are **Write** and **Writeln**. The write statement leaves the cursor at the end of the current output, whereas the writeln places the cursor at the start of a new line.

Format:

```
Write (item 1, item 2,...);
Writeln (item 1, item 2,...);
```

Example:

```
Write (‘I like’);
Write (‘Information Technology’);
```

Output: I like Information Technology

Example:

```
Writeln ('I like');
Write ('Information Technology');
```

```
Output:          I like
                Information Technology
```

Example:

```
Write ('The answer is:', ' ', ans);
```

```
Output:         The answer is: 15
```

NB. 15 is stored in the variable ans.

The Assignment Statement

Format:

```
Variable := Expression;
```

Example:

```
sum := x + y;
Name := 'John';
```

BASIC PASCAL PROGRAMS

Structure of a Pascal Program

```
Program ProgramName;


Const
        Constant Declaration;

Var
        Variable Declaration;

Begin
        Main Body of Program
End.
```

Reserved Words

The Pascal programming language has several important words in it. These are called keywords or reserved words. These keywords cannot be used as variable names. Examples of keywords are: **program, label, const, type, var, begin, end, and, array, case, div, do, else, file, for , function, goto, if, in, mod, nil , not, of , or, packed, procedure, record, repeat, set, then, to, until, while, with, read, readln, write, writeln.**

NB. Comments are enclosed in curly brackets { }

Sequence Pascal Program Example:

```
Program Square;
{This program finds the square of a number}

var {Variable Declaration}
  number,sq:integer;

  Begin
      write('Enter number: '); {Prompt for input}
      readln(number); {Store data into variable n}
      sq:=number*number; {Calculate the square}
      writeln('The square is: ', sq); {Output result}
      readln;
  end.
```

Conditional Structures

The If Then Else Statement

The if statement allows the conditional execution of one statement, or the choice between execution of two statements.

Format 1:

```
If expression then
   Begin
      Statement(s)
   End;
```

Example:

```
If x > y then
 writeln(x);
```

Format 2:

```
If expression then
   Begin
      Statement(s)
   End
Else
   Begin
      Statement(s)
   End;
```

Example:

```
If x > y then
 writeln(x)
else
 writeln(y);
```

Selection Pascal Program Example

```
Program Larger;
{This programs determines and prints the larger of two numbers}

var
 a,b: integer;

 Begin

   Writeln('Enter two numbers');
   Readln(a,b);

   if a>b then
      writeln(a)
   else
       writeln(b);

   if a = b then
      writeln('Numbers are equal');
   readln;
 end.
```

Loop Structures

Definition: A set of statements which are repeated until some condition is met.

Types of Pascal Loops

For Loop

Format:

```
For control variable := start value to end value do
                Begin
                    Statement(s)
                End
```

Example:

```
For I := 1 to 10 do

                Begin
                    writeln('Enter number');
                    Readln( x);
                    Sq: = x *x;
                    writeln(Sq);

                End;
```

For Loop Pascal Program

```
Program LoopAverageKnown;
{Program finds the average of 3 numbers}

var

 x,i,sum:integer;
 avg:real;

 begin
     sum:=0;

     for I := 1 to 3 do
         begin
              writeln('Enter Number');
               readln(x);
               sum:=sum+x;
         end;

       avg:=sum/3; {calculate the average}

      writeln(avg);  {output the average}

      readln;
 end.
```

While Loop

The while loop executes the statements within the loop as long as the condition is true. The condition is tested at the top of the loop.

Format:

```
While Expression do
 Begin
 Statement(s)
End;
```

Example:

```
begin
   I := 0;
   while I <= 5 do
      begin
         I := I + 1;
         Sq:=i*I;
        Writeln(sq);
      end;
end.
```

While Loop Pascal Program

```
Program AverageUnknown;
{Program finds the average of a set of numbers, the last number is 0}

var
 x,i,sum:integer;
 avg:real;

 begin
     sum:=0;
     i:=0;

       writeln('Enter Number');
       readln(x);

  while i<>999 do

        begin
           i:=i+1;
           sum:=sum+x;
           writeln('Enter Number');
           readln(x);

       end;

      avg:=sum/3;
      writeln(avg);
      readln;
 end.
```

Repeat Until Loop

The repeat until loop is like the while loop except that it tests the condition at the bottom of the loop.

Format:

```
Repeat
 Statement(s);
Until Expression;
```

Example:

```
begin
   I := 0;
   Repeat

         I := I + 1;
         Sq:=i*I;
        Writeln(sq);
    Until I =5;
end.
```

Repeat Loop Pascal Program

```
Program Average;
{Program finds the average of 3 numbers}

var

 x,i,sum:integer;
 avg:real;

 begin
 sum:=0;

  repeat
           i:=i+1;
           writeln('Enter Number');
           readln(x);
           sum:=sum+x;

  until i=3;

      avg:=sum/3;
      writeln(avg);
      readln;
 end.
```

ARRAYS

Definition: A consecutive group of memory locations that have the same name and type. A location is referenced by using the array name and the element's index.

NB The Index type must be ordinal (byte or integer) or an expression that evaluates to these data types.

Declaring Arrays

Format:

var
Arrayname: **Array**[Start Index **..** End Index] **of Arraytype;**

Example:
var
numbers: array[1 .. 3] of integer;

NB Elements of numeric arrays are initialized to 0 by default. Elements of string arrays are initialized to " " by default.

Placing Values into an Array

Format:

Arrayname[index]: = value;

Example:

Numbers[1]: = 10;

Numbers

Index	
1	10
2	
3	

Copying a value from a Location in an array into a variable

Format:

variablename := Arrayname[index];

Example:

x: = Numbers[1];   {x now contains the value 10}

Array Pascal Program

```
Program Search;
{Linear search}

var
            Accounts: array[1 ..5] of integer;
            I, accno: integer;
Begin

        {store Account Numbers}
        for i:= 1 to 5 do
            begin
               Writeln('Enter Account Number');
                     Readln(accounts[i]);
           end;

            Writeln('Enter Account Number');
            Readln(accno);

            I :=1;
            While (accno <> Accounts [I]) and ( I <>5) do
                        I := I +1;

            If accno = Accounts[I] then
                        writeln('Account Found')
            Else
                        writeln('Account not Found');

                readln;
end.
```