

AN INTRODUCTION TO PASCAL PROGRAMMING

Pascal is a high level language. It was invented by Niklaus Wirth, a computer scientist at the Institute of Informatics in Zurich. The language was published in 1971 and named in honour of the seventeenth century French Philosopher and Mathematician, Blaise Pascal, who invented the first automatic adding machine. Based on responses from experience users the language was slightly modified and published in a revised form in 1973. His principal objectives for Pascal were for the language to be efficient to implement and run, allow for the development of well structured and well organized programs, and to serve as a vehicle for the teaching of the important concepts of computer programming. The section below illustrates the general format of a typical Pascal program.

Program NameOfProgram (input, output);	{ <i>Heading</i> }
Uses Crt, Dos; [Units]	
CONST	
constantName = literal; comment	
Var	
variableName : Datatype; comment	
	} <i>Declaration Section</i> }
Begin	
statement1	
.;	
.;	
statementN;	
End.	} <i>Executable Statements</i> }

As shown above all Pascal programs must begin with the reserve word *program*, followed by the name given to the program. The words *input* and *output* are used to indicate that the program will accept input and produce output, respectively. **Uses**, identifies predefined programs/units within the compiler that the programmer intends to use in the program. **CONST** is used to indicate constants, that is, values that will remain unchanged throughout the life of the program. For each constant and name must be specified and the literal (value). **Var** is used to indicate that what come after are the variables (known as *identifiers* in Pascal) to be used in the program – all variables to be used in the program must be declared. Equally, all variables must be assigned the appropriate data type. The most common data types used in Pascal are: integer, real, char and string. **Begin** tells the compiler where the executable statements start. **End**, with a full stop (period) indicates where the program terminates. It is important to note that statement beyond this point will not be recognized by the compiler. **Comments** help you to document and maintain your program. Comments are ignored by the compiler.

Integers are positive and negative whole numbers including zero, example, -12, 0, and 5. **Real numbers** are those numbers which have a fractional component, example, -12.4, 0.0, 0.25 and 8.72. **Char** is the short for character; this is fundamentally used to represent a letter of the alphabet. **String** is used to represent a number of characters, example, this can be used to store data for name, and address. You should have already noticed that all executable Pascal statements end with a semicolon (;). Semicolons are not used at the end of statements that contain reserve words. **Reserve words** are words that are set aside by the programming language for special purposes.

CONVERTING PSEUDOCODE TO PASCAL PROGRAMMING

INPUT STATEMENT	
Pseudocode	Pascal
INPUT variableName	Read(variableName);
INPUT (score	Read(score) or ReadLn (score);
ASSIGNMENT STATEMENT	
variableName $\leftarrow$ Expression	variableName := expression
Sum $\leftarrow$ (a + b)	Sum := (a + b);
OUTPUT STATEMENT	
DISPLAY "Statement"	Write('Statement'); or WriteLn('Statement');
DISPLAY "Statement" , variableName	Write('Statement', variableName); or WriteLn('Statement', variableName);
SELECTION: IF-THEN	
IF (x > y) THEN DISPLAY "Sum is", sum ENDIF	if (x > y) then begin Write('Sum is ' , sum); end;
SELECTION: IF-THEN-ELSE	
IF (x > y) THEN DISPLAY "Sum is", sum ELSE DISPLAY "Product is", product ENDIF	if (x > y) then begin Write ('Sum is ' , sum); end else begin Write ('Product is ' , product); end end;
FOR LOOP	
FOR k FROM 1 To 100 DO DISPLAY "Enter Name " INPUT name DISPLAY "Enter Age" INPUT age ENDFOR	For k := 1 to 100 do Begin Write('Enter Name '); Read(name); Write('Enter Age '); Read(age); End;
WHILE LOOP	
INPUT age WHILE (age <> 0) do DISPLAY "Enter Age" INPUT (age); ENDWHILE	Read (age); While (age <> 0) do Begin Write('Enter Age '); ReadLn (age); End;

A SAMPLE PASCAL PROGRAM

/* This program accepts from a user: an item code, unit price and quantity of the item. It then calculates the cost, tax amount and the total to be paid. Finally, the amount to be paid is displayed*\

```
Program Invoice (Input, output);
Uses Crt, Dos;
Const
    GCTRATE = 0.175;
Var
    itemID: string; {Unique code used to represent each item}
    itemName: string; {Name of item}
    unitPrice: real; {Price for a single item}
    quantity: integer; {Number of given item}
    cost: real; {Amount excluding GCT}
    tax: real; {Amount for GCT}
    total: real; {Amount of money to be paid}

Begin
    ClrScr;      {Clears the Screen}
    Write('Enter itemID =>      ');
    ReadLn(itemID);
    Write('Enter Name of Item  =>      ');
    ReadLn(itemName);
    Write('Enter the Unit Price , dollars and cents =>      ');
    ReadLn(unitPrice);
    Write('Enter Number of Items  =>      ');
    ReadLn(quantity);
    {Calculations}
    cost := (unitPrice * quantity);
    tax := (cost * GCTRATE);
    total := (cost + tax);

    WriteLn;
    Write('Total amount to be paid is: $'      , total:8:2 );
    WriteLn;
    WriteLn;
    Write('Press Any Key to Exit.....      ');
    ReadKey;

End.
```

The next two pages give the solution to four (4) sample programs. The main aim of the programs is to give the reader some insight into how the most common programming constructs are implemented in the Pascal programming language. It is left up to the reader to try these program codes in the Pascal compiler. Have fun!

Question 1: Write a program to accept the length and width of a rectangle calculate and display the area and perimeter.

```
Program RectangleAP (input, output);
Uses Crt, DOS;
Var
    length: integer; {The length of the rectangle}
    width: integer; {The width of the rectangle}
    area: integer; {The area of the rectangle}
    perimeter: integer; {The perimeter of rectangle}

Begin
    Clrscr;
    WriteLn;
    Write(' Enter the length of the rectangle => ');
    ReadLn (length);
    Write(' Enter the width of the rectangle => ');
    ReadLn (width);

    area := (length * width);
    perimeter := (length + width) * 2;

    WriteLn;
    Write(' The area of this rectangle is: ', area, ' square units');
    WriteLn;
    Write(' The perimeter of this rectangle is: ', perimeter, ' units');
    Readkey;
End.
```

Question 2: Write a program to accept values in the variable mystery and wild. If mystery is less than wild the program must calculate and print the square of mystery otherwise the cube of wild should be printed.

```
Program MysteryWild (input, output);
Uses Crt, DOS;
Var
    cube, square, mystery, wild: integer;

begin
    Clrscr;
    WriteLn;
    Write('Enter the first number: ');
    ReadLn(mystery);
    Write('Enter the second number :');
    ReadLn(wild);

    if (mystery < wild) then
        begin
            square := (wild * wild);
            WriteLn ('The square of ', wild, ' is ', square);
        end
    else
        begin
            WriteLn;
            cube := (mystery * mystery * mystery);
            WriteLn ('The cube of ', mystery, ' is ', cube);
        end;
    Readkey;
end.
```

Question 3: Write a Pascal program to accept 5 integers and print the number of even numbers and odd numbers.

```
Program EvenOdd (input, output);
Uses Crt, DOS;
Var
    evenCount, oddCount, counter, number: integer;

Begin
    Clrscr;
    For counter := 1 to 5 do
        begin
            Write('Enter an integer:      ');
            Read (number);
            IF (number MOD 2 = 0) then
                begin
                    evenCount := (evenCount + 1);
                end
            else
                begin
                    oddCount := (oddCount + 1);
                end
            end;
        end;

    Write ('Count of even numbers is      ', evenCount);
    Writeln;
    Write ('Count of odd numbers is:      ', oddCount);
    Readkey;
end.
```

Question 4: Write a Pascal program to accept the names and ages of a set of students and print the total of the ages. The program must terminate when zero is entered for an age

```
Program AgeCount (input, output);
Var
    Age, ageTotal: integer;
    Name: string;

Begin
    Write('Enter the student's name ');
    ReadLn(name);
    Write('Enter the student's age ');
    ReadLn(age);

    While (age <> 0) do
        Begin
            ageTotal := (ageTotal + age);
            Write('Enter the student's name ');
            ReadLn(name);
            Write('Enter the student's age ');
            ReadLn (age);
        End
    Write(' The total of the ages is' , ageTotal, 'years');
    Readkey;
End.
```