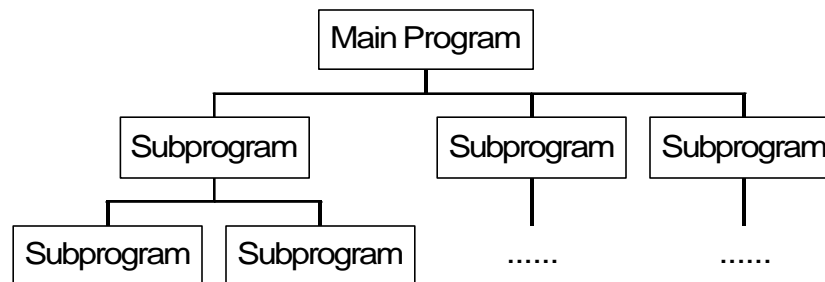


**FORM 5 PASCAL PROGRAMMING****Unit 8: Subprogram (I) - Procedures**

March, 01

**8.1 STRUCTURED PROGRAMMING (結構化程式編寫)**

- **Structured Programming** is applied to make the program more \_\_\_\_\_ and \_\_\_\_\_;
- consists of 3 elements:
  - ◇ Top-down development
  - ◇ Modular design
  - ◇ Structured coding
- **Top-Down Development** refers to the method of breaking down a “big” problem into “smaller” subprograms (mini-programs) by **Stepwise Refinement Method**. The result can be illustrated as follows:



- PASCAL provides 2 different types of subprogram (子程序) (or called modules, subroutines):
  - ◇ **Procedures** (過程)
  - ◇ **Functions** (函數)

**8.2 DECLARATION OF PROCEDURES**

- the structure of PROCEDURE is very similar to that of a PROGRAM;

```

procedure identifier (formal parameter list);
const
var
  ...           {local variables}

begin
  :
  :
End;
  
```

- In FORMAL PARAMETER (形式參數) list, there are 2 kinds of parameters:
  - ◇ *Value Parameters (or Input Parameters)* 值參數  
used only to pass data to the procedure and its value is not to be changed in the procedure
  - ◇ *Variable Parameters (or Output Variables)* 變量參數  
must be preceded by VAR;  
used to return results to the calling programs
- e.g. Procedure RectArea (Width, Length : real; VAR Area : real);

where Width and Length are value parameters  
Area is variable parameter.

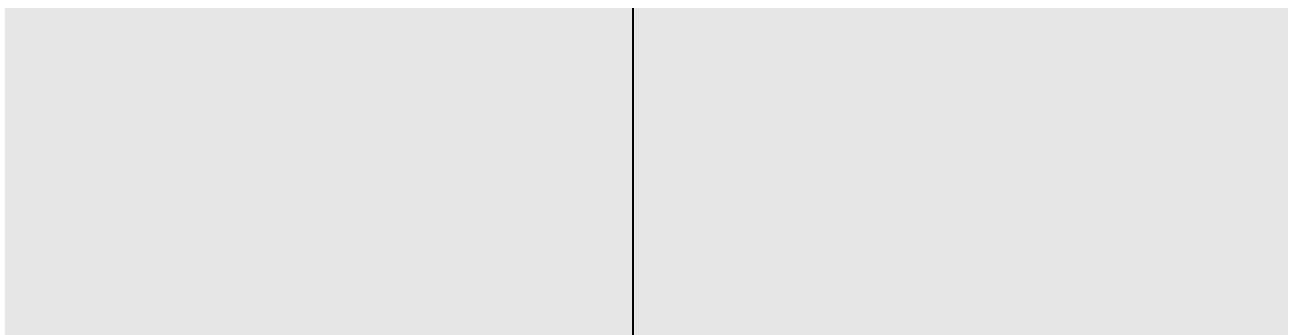
**RULE OF THUMB:**

*Unless it is necessary to change the values of the parameters, always use VALUE PARAMETERS!!!*

- e.g. To illustrate the PROCEDURE declaration

Without using procedure	Using Procedure
<pre> program main; var   k : integer;  begin {main program}   k := 20;   write('The original value is ', k);   k := k+1;   writeln('Now it becomes ', k);   k := k+1;   writeln('Now it becomes ', k);   k := k+1;   writeln('Now it becomes ', k) end. </pre>	<pre> program main; var   k : integer;    procedure Add_One;   begin {Add_One}     k := k + 1;     writeln('Now it becomes ', k);   end;  begin {main program}   k := 20;   write('The original value is ', k);   Add_One;   Add_One;   Add_One end. </pre>

The output of both programs are:



- e.g. To illustrate the difference among of Value Parameter, Variable Parameter, and Local Variable.

```
program Main;

var
  x, y : integer;

procedure Swap1;
var
  temp : integer;
begin
  Temp := x;
  x := y;
  y := temp;
end;

procedure Swap2 (A, B : integer );
var
  temp : integer;
begin
  Temp := A;
  A := B;
  B := temp;
end;

procedure Swap3 (var A, B : integer );
var
  temp : integer;
begin
  Temp := A;
  A := B;
  B := temp;
end;

begin {Main Program}
  x := 10;
  y := 20;
  write ( 'Original values :');
  writeln('x= ':5, x, 'y= ':7, y); writeln;

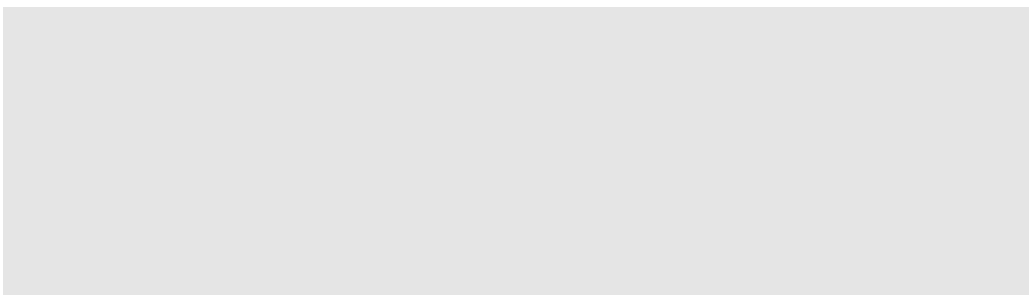
  Swap1;
  write('Values after calling procedure Swap1 :');
  writeln('x= ':5, x, 'y= ':7, y); writeln;

  Swap2;
  write('Values after calling procedure Swap2 :');
  writeln('x= ':5, x, 'y= ':7, y); writeln;

  Swap3;
  write('Values after calling procedure Swap3 :');
  writeln('x= ':5, x, 'y= ':7, y); writeln;

end.
```

The output is:



## ■ Notes:

Procedure Swap1	Procedure Swap2	Procedure Swap3
◇ No formal parameter list;	◇ Formal parameter list is present;	◇ Formal parameter list is present;
◇ Since x and y are <b>Global Variables</b> , any modification within that procedure will affect the values of x and y directly;	◇ Without the reserved word VAR in the formal parameter list, a and b are <b>value parameters</b> only;	◇ With the reserved word VAR before a and b, they are considered <b>variable parameters</b> ;
	◇ Changes of <b>formal parameters</b> (a and b) have no effect on the <b>actual parameters</b> (x and y). Thus no expected swapping occurs.	◇ Changes of formal parameters (a and b) will affect the values of the <b>actual parameters</b> (x and y)

- e.g. The payroll program in previous units can be modified with the use of procedures.

```

program Payroll;
var HourlyRate, Wage : real;
    NoOfHours : integer;

procedure GetData( var NoOfHours : integer;
                  var HourlyRate : real );
{ Get and return the number of hours worked,
  NoOfHours and the hourly rate, HourlyRate }
begin
  write( 'Enter Number of Hours worked: ' );
  readln( NoOfHours );
  write( 'Enter Hourly Rate: ' );
  readln( HourlyRate );
end;

procedure CalculateWage( NoOfHours : integer;
                        HourlyRate : real;
                        var Total : real );
{ Input   : NoOfHours, HourlyRate
  Process : Base on NoOfHours, calculate the allowance
  Output  : Total }
var Allowance, RegularWage : real;
    Overtime : integer;
begin
  RegularWage := HourlyRate * NoOfHours;
  Overtime := NoOfHours - 7;
  if Overtime > 0
  then Allowance := HourlyRate * 0.40 * Overtime
  else Allowance := 0;
  Total := RegularWage + Allowance;
end;

procedure OutputWage( Total : real );
{ Input   : Total wage
  Process : Display wage }
begin
  writeln( 'The wage is $', Total:1:2 );
end;

begin
  GetData( NoOfHours, HourlyRate );
  CalculateWage( NoOfHours, HourlyRate, Wage );
  OutputWage( Wage );
end.

```

end of unit 8