

FORM 5 PASCAL PROGRAMMING

Unit 9: Subprogram (II) - Function

March, 01

9.1 FUNCTIONS VS. PROCEDURES

- **Function** is the other type of subprogram, other than procedure. Structurally both function and procedure are similar. However, user-defined function have the following distinct features:
 - ◇ A data type of a function result must be included in the function header;
 - ◇ At least one statement in the function must assign a value to the function name;
 - ◇ function is used when only ONE VALUE is to be returned.

9.2 TYPES OF FUNCTION

- In PASCAL, functions are divided into 2 main categories :

Built-in (or Standard) Functions 內置函數

are system developed which can be called directly in any PASCAL program.

User-Defined Functions 用戶自訂函數

are declared in the program similar to that of a procedure and can only be referenced within the program itself.

9.3 COMMON BUILT-IN FUNCTIONS

9.3.1 *SQR(x)*

- *SQR* function is used to calculate the square of a given number *x*.
- Example : Find the difference of the squares of two numbers where the numbers are input from the keyboard.

```

program DIFF2SQR (input, output);
var
  Num1, Num2 : real;
begin
  write ( 'Please input two numbers : ' );
  readln ( Num1, Num2 );
  writeln ( 'The difference of the squares of two number is : ' );
  write ( SQR(Num1) - SQR(Num2):7:2 )
end.

```

9.3.2 *SQRT(x)*

- *SQRT* function returns the square root of a given number *x*, which must be greater than or equal to 0.
- Example :Find the value of a number raised to the power 1.5 which is input from the keyboard.

```
program POWER ( input, output );
var
  Num, Num15 : real;
begin
  write ( 'Please input a number : ' );
  readln ( Num );
  Num15 := SQRT( Num * Num * Num );
  write ( 'The number raised to the power 1.5 ');
  writeln ( 'is : ', Num15:10:2)
end.
```

9.3.3 *TRUNC(x)*

- *TRUNC* function is used to truncate the decimal part of *x* and return a whole number which is less than or equal to *x*.
- Example :Program segment that truncates the number 3.88 to 3 and prints it out.

```
writeln ( TRUNC(3.88) ) ;
```

9.3.4 *ROUND(x)*

- *ROUND* function is used to round off the argument *x* to the nearest integer.
- Example :Program segment that round the number 3.88 and prints it out.

```
writeln ( round(3.88) ) ;
```

9.3.5 ABS(x)

- ABS function returns the absolute value of the numeric expression x.

- Example :

```
writeln (3 - 18);  
writeln (ABS(3 - 18));
```

9.3.6 SIN(x)

- SIN function computes the sine of an angle in radians. x can be a numeric constant or a numeric expression. When x is in degree, we can convert x into radian by multiplying $\pi/180$.
- Example :Program segment which prints different values of sine.

```
X := 0.18;  
Y := 1.414;  
writeln ( SIN( 1.5 ) );  
writeln ( SIN( 90 * 3.1416 / 180 ):5:2 );  
writeln ( SIN( X + Y * 2 ) );
```

9.3.7 COS(x)

- COS function computes the cosine of an angle in radians. x can be a numeric constant or a numeric expression. When x is in degree, we can convert x into radian by multiplying $\pi/180$.
- Example :Program segment which prints different values of cosine.

```
var  
  Degree, Radian : real;  
begin  
  writeln ( 2 * COS( 0.4 ) );  
  Degree := 180;  
  Radian := Degree * 3.141593 / 180;  
  write ( COS(Radian):5:2 )  
end.
```

9.3.8 LN(x)

- LN computes the natural logarithm of x (in base e, where $e = 2.718\dots$).x must be a positive number which can be a numeric constant or a numeric expression.
- Example :Program segment that prints different values of natural logarithm.

```
writeln ( LN( 2.71828 ):5:2 );
writeln ( LN( 99 ) );
writeln ( LN( 1 ):5:2 );
writeln ( LN( 4 + 2 / 3 ) );
```


9.3.9 RANDOM

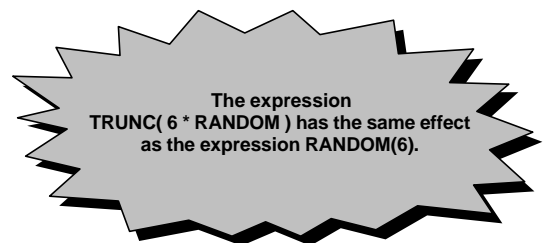
- **RANDOM** function returns a random number (隨意數字)greater than or equal to zero and less than 1. (i.e. $0 \leq \text{random number} < 1$);
- **RANDOM(x)** returns an integer greater than or equal to zero and less than x (i.e. $0 \leq \text{random number} < x$);
- The standard procedure **randomize** should be called before calling the functions.
- Example :Illustration of the functions **RANDOM(x)** and **RANDOM**.

```
program RandomFunction;
var
  I : integer;
begin
  randomize;

  for I := 1 to 3 do
    write ( RANDOM:10:6 );
  writeln;

  for I := 1 to 3 do
    write ( TRUNC(6*RANDOM)+1:10 );
  writeln;

  for I := 1 to 3 do
    write ( RANDOM(6)+1:10 )
end.
```



0.750375	0.964247	0.144816
3	6	1
3	3	4

9.3.10 PRED(y)

- PRED function returns the predecessor of y (if it exist).

9.3.11 SUCC(y)

- SUCC function returns the successor of y (if it exist).
- Example :To illustrate the functions PRED and SUCC.

```

Writeln (PRED('b'):5, SUCC('b'):5);
writeln (PRED('B'):5, SUCC('B'):5);
writeln (PRED(100):5, SUCC(100):5);
writeln (PRED(-1):5, SUCC(-1):5);

```

**9.4 USER-DEFINED FUNCTIONS**

- Often, the function provided in PASCAL are not sufficient for solving a particular problem. We may hope to write our own functions;
- Syntax:

```

Function Identifier (Formal Parameter List) : Return Data-Type;

var
    :                               {local declaration section}

begin

    Statements

end;

```

- ☞ Function header, declarations of variables which are local to the function.
- ☞ Function executable statements which are preceded by the reserved word BEGIN and followed by the reserved word END.
- ☞ A function should be ended with a semicolon.

- Example :An user-defined function TAN which computes the tangent of an angle X in radian.

```
FUNCTION TAN( X : real ) : real;

begin
  TAN := SIN(X) / COS(X)
end;
```

- Example :An user-defined function AS which computes the summation of a A.P. to n terms using the formula $AS = \frac{N}{2}(2a + (N - 1)d)$.

```
FUNCTION AS(a,d:real; N:integer ) : real;

begin
  AS := (2*a+(N-1)*d)*N/2
end;
```

- Example :A program contains a function CalculateCube which computes the cube of an integer X.

```
program CalculateCube;
var X : integer;

function CubeOf( X : integer ) : integer;
{ Input  : an integer X
  Return : the cube of X via CubeOf }
begin
  CubeOf := X * X * X
end;

begin
  X := 3;
  writeln( 'The cube of X is ', CubeOf( X ) )
end.
```

end of unit 9