FORM 4 PASCAL PROGRAMMING
Unit 6: Arrays and Strings

March, 01

6.1 WHAT IS AN ARRAY（陣列）?

- is a structured data type in which we store a collection of data items of the same type;
- Declaration:

```
const
  max = 40;
var
  alphabet     : array[1..26] of char;
  mark         : array[1..max] of integer;
  school_mark  : array[1..32, 1..6] of integer;
  names        : array[1..max] of string;
```

- Or:

```
const
  max = 40;

type
  namearray    = array[1..max] of string;
  scorearray   = array[1..max] of integer

var
  mark         : namearray;
  names        : scorearray;
```

6.2 USING ARRAYS

The following example illustrates how the use of arrays help processing a collection of marks.

Without using arrays

```
program ProcessMarks;
const NoOfScores = 10;
var Score1, Score2, Score3, Score4, Score5,
    Score6, Score7, Score8, Score9, Score10 :
integer;
    Average : real;
begin
  { Read the scores }
  writeln( 'Enter Scores:' );
  readln( Score1 );
  readln( Score2 );
  readln( Score3 );
  readln( Score4 );
  readln( Score5 );
  readln( Score6 );
  readln( Score7 );
  readln( Score8 );
  readln( Score9 );
  readln( Score10 );

  { Calculate the average }
  Average := ( Score1 + Score2 + Score3 + Score4 +
               Score5 + Score6 + Score7 + Score8 +
               Score9 + Score10 ) / NoOfScores;
  writeln( 'The average is ', Average:1:2 );

  { Print out scores above average }
  writeln( 'Scores above average are:' );
  if Score1 > Average
    then writeln( Score1 );
  if Score2 > Average
    then writeln( Score2 );
  if Score3 > Average
    then writeln( Score3 );
  if Score4 > Average
    then writeln( Score4 );
  if Score5 > Average
    then writeln( Score5 );
  if Score6 > Average
    then writeln( Score6 );
  if Score7 > Average
    then writeln( Score7 );
  if Score8 > Average
    then writeln( Score8 );
  if Score9 > Average
    then writeln( Score9 );
  if Score10 > Average
    then writeln( Score10 )
end.
```

Using arrays

```
program ProcessMarks;
const NoOfScores = 10;
type ScoreArray = array[1..NoOfScores] of integer;
var Score : ScoreArray;
    Average : real;
    i, Total : integer;
begin
  { Read the scores }
  writeln( 'Enter Scores:' );
  for i := 1 to NoOfScores do
    readln( Score[i] );

  { Calculate the average }
  Total := 0;
  for i := 1 to NoOfScores do
    Total := Total + Score[i];
  Average := Total / NoOfScores;
  writeln( 'The average is ', Average:1:2 );

  { Print out scores above average }
  writeln( 'Scores above average are: ' );
  for i := 1 to NoOfScores do
    if Score[i] > Average
      then writeln( Score[i] )
end.
```

6.3 TECHNIQUES INVOLVING ARRAYS

6.3.1 Initialize （起始化）all the elements of an array

```
var
  marks : array [1..max] of integer;
  i : integer;
begin
  for i := 1 to Max do
  marks[i] := 0;
```

6.3.2 Searching of an element of an array

```
program SearchingArray;
const MaxSize = 10;

type NameType = string;
     NameArray = array[1..MaxSize] of NameType;

Var
  Position : integer;
  Target : NameType;
  Name : NameArray;

begin
  Position := 0;

  for i := 1 to MaxSize do
    if Target = Name[i]
      then Position := i

  :
  :
```

6.3.3 Input/output of elements of an array

```
program InputOutput_Array;
const
  MaxSize = 10;
var
  NameArray = array[1..MaxSize] of String[30];

begin
  for i := 1 to MaxSize do
begin
  write('Enter name ', i, ' : ');
  readln(NameArray[i]);
end;

for i := 1 to MaxSize do
begin
  write('The name ', i, ' is ');
  writeln(NameArray[i]);
end;
```

6.3.4 Sum of all elements of an array

```
program ProcessTestScores;
const
    Max = 40;
type
    MarkArray = array[1..Max] of integer;
var
    Marks : MarkArray;
    Total : integer;
    Average : real;

begin
{ Finding the sum of each row }
  :
  :
  for Student := 1 to Max do
    begin
      Total := Total + Mark[Student]
    end;

{ Finding the Average }
  average := Total / Max;
  :
  :
```

6.3.5 Find the largest/smallest element

```
program Largest;
const MaxSize = 10;
type ScoreType = integer;
     ScoreArray = array[1..MaxSize] of ScoreType;

Var  NoOfScores : integer;
     Score : ScoreArray;
     HighScore : ScoreType;
     i : integer;

begin
  HighScore := Score[1];
  for i := 2 to NoOfScores do
    if Score[i] > HighScore
      then HighScore := Score[i]


end.
```

6.3.6 Copying the whole array

```
program CopyArray;

const
  MaxSize = 10;

var
  Mark_5D, Mark_5C : array[1..MaxSize] of integer;
  i : integer;

begin
  :
  :
  for i := 1 to MaxSize do
     Mark_5D[i] := Mark_5C[i]
  :
  :
end.
```

6.4 WHAT IS A STRING（字串）?

- is a finite sequence (array) of characters enclosed in apostrophes (’ ’)
- e.g. 'Hi!', 'F.4', 'Benevolent', 'How are you my friend?'

- Declaration:

```
var
  DemoStr1 : string ;    {default length 255 characters}
  DemoStr2 : string[30]; {preset length 30 characters}
```

- to access any character in a string, put the string variable name followed by an integer index in square bracket:

  e.g. `LastName:=‘Lai’;`

  The statement

  `write(LastName[2])`

  causes the letter ‘ a’ to be printed

6.5 STRING OPERATIONS

6.5.1 Comparison of strings

- In PASCAL, all characters are ordered according to their ASCII codes
  e.g. ‘ G’ <‘ T’
  ‘ g’ >‘ G’
- Strings of more than one character are compared character by character beginning at the left
  e.g. ‘ AIA’ >‘ AI’
  ‘ 1234’ =‘ 1234’
  ‘ IBM’ < ‘ Ibm’

6.5.2 Input/Output of strings

- e.g.

```
program testrw;
type
  TestType=string[30];
var
  name, address : TestType;
begin
  write('Enter your name (then press <ENTER>) :');
  readln(name);
  write('Enter your address (then press <ENTER>) :');
  readln(address);
  writeln('Hello! ', name);
  writeln('You live in: ');
  writeln(address)
end.
```

Suppose you enter the following data when you run (or dry run) the program:

Your name: Tsang Cho Choi
Your address: 5/F Kowloon Road Wong Tai Sin

What is the screen output? (write it down in the box below)

6.6 COMMON STRING FUNCTIONS AND PROCEDURES

- PASCAL provides some functions for string manipulation (manipulation of character string) rather than numerical data. We are here to describe these features and illustrate their applications.

6.6.1 CONCAT(s_1, s_2, ..., s_n)

- This function returns the result of joining s1, s2, ..., sn together. In TURBO PASCAL the operator "+" is also used to concatenate strings : s1 + s2 + ... + sn.

- Example : Join 3 strings : Day, Month, Year together to form another string Today.

```
program DATE;
var
  Day, Month, Year, Today : string;
begin
  Day := '31';
  Month := 'DECEMBER';
  Year := '1992';
  Today := CONCAT( Day, ' ', Month, ',', Year );
  writeln ( Today )
end.
```

6.6.2 LENGTH(s)

- LENGTH function is used to return the number of characters in the string s. The returned value is a non-negative integer, but not a string.

6.6.3 COPY(str, i, n)

- COPY function returns a string that is a substring of str starting from the ith character of s and consisting n characters. s must be a string constant or variable, i and n must be type INTEGER. If n = 0 or i > LENGTH(s), the function returns a null string.

6.6.4 ORD(s)

- This function returns an integer value which is the ASCII code of the string s. s must be an ordinal data such as 1, 2, 'A', 'G', etc. and cannot be a null string.

*6.6.5 CHR(***n***)*

- This function returns the character with ordinal number or ASCII code corresponding to n which can be an integer or a numeric expression.

6.6.6 VAL(s, n, code)

- VAL is a procedure which converts the string s into its corresponding numeric value and stores it in the variable n. If no errors are detected during execution of the procedure, code remains at zero. Otherwise, code will contain the position of the string where error occurs. VAL is the complement procedure of STR.

*6.6.7 STR(***n, s***)*

- STR is also used as a type conversion procedure which converts the number n into a string and stores it in the variable s. It is the complement procedure of VAL.

e n d o f u n i t 6