Chapter 5: Arrays and character strings

1. Arrays

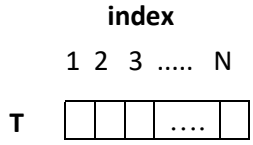
Arrays (or Tables) are data structures that allow manipulating multiple data with the same type.

We distinguish two types of arrays: one-dimensional arrays and two-dimensional arrays.

1.1. One-dimensional arrays

A. Definition

One-dimensional arrays are represented by a set of successive boxes identified by an index.



B. Declaration syntax:

Var

Array_Name : Array [1 .. Array_Size] of Data_Type;

To declare an array you must give it an identifier (name), a size and the type of data it will contain.

Example :

Var
T : Array [1 .. **10**] of **integer**; *// Array T of size 10 which contains integer type values*

C. Arrays in C language :

Syntax :

Type Array_id [Array_Size] ;

Exemple :

int T[10] ;

D. Reading/Writing in one-dimensional arrays

- Read (Table_Id [index]); // allows reading a value from the keyboard and store it in the box indexed by the index value. Exp: **read(T[2]);**

- Write (Id_Tableau [index]); // allows displaying the contents of the box indexed by the value of the index. Exp: **write (T[1]);**

- Id_Table [index] ← expression; // allows assigning the expression evaluation result to the box identified by the index. Exp: **T[3] ← (A+B) div 2;**

E. Manipulating arrays

Generally, we use loops to manipulate arrays.

Exp: the following loop fills the table T with N values entered from the keyboard.

Algorithm	**Program in C**
For (i←1 To N)Do read (T[i]) ;	**For**(i = 0 ; i < N ; i++) Scanf(''%d'', &T[i]);

Exercise1 :
Write an algorithm that allows filling a table with 10 real values, and then calculate the sum and average of the values filled.

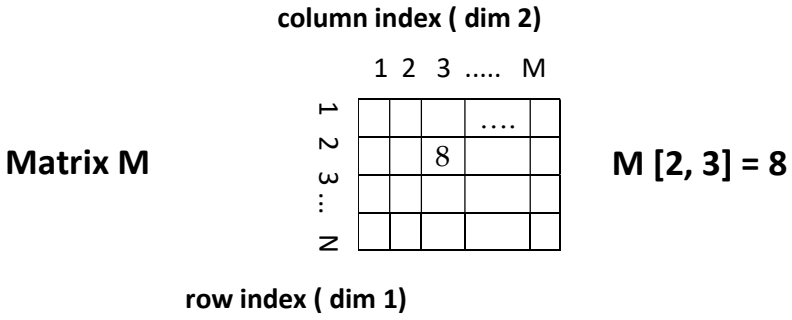
Exercise2 :
Write an algorithm that fills a table with 10 notes, and then calculates the number of notes greater than 10.

1.2. Two-dimensional arrays (matrix)

A. Definition

A Matrix is a data structure represented in the form of two-dimensional arrays, which allow manipulating data of the same type.

In programming, matrix boxes are identified by row numbers (index) and column numbers (index), which represent dimension 1 and dimension 2 respectively.



Note :
If the number of rows is equal to the number of columns (equal to N), we say that the matrix is square of size (or order) N.

B. Declaration

```
Var
   Matrix_name : Array [1 .. row_size, 1 .. column_size] of  Type;
```

Example :

```
VAR
   M : Array [1 .. 6, 1.. 4] of integer ;  // Matrix M of 6 rows and 4 columns with integer values
```

C. Declaration using C language

```
 Type  Array_id [row_size ] [column_size] ;
```

Exemple :

```
  int M[6][4] ;
```

D. Reading/Writing in Matrix

- Read (Matrix_id [row_index, column_index]); // allows to read a value from the keyboard and store it in the box indexed by the values of the row and column.
  Exp: Read(M[2, 3]);

- Write (Matrix_id [row_index, column_index]); // allows to display the contents of the box indexed by the values of the row and column.
  Exp: Write (M[1, 1]);

- Matrix_id [row_index, column_index] ← expression; // allows to assign the expression evaluation result to the box identified by the values of the row and column.
  Exp: M [3, 1] ← (A+B) mod 2;

E. Manipulation of Matrix

In general, we use two nested loops to manipulate a matrix.

Exp: the following loops fills a matrix of dimensions (N x M) with real type values entered on the keyboard.

Algorithm	Langage C

For (i←1 To N) **Do**
  For (j←1 To M) **Do**
  Read (M [i , j]) ;
	For(i = 0 ; i < N ; i++)
	  For(j = 0 ; j < M ; j++)
	  Scanf(''%f'', &M [i] [j]);

Exercise1:
Write an algorithm that fills a matrix with real type values, and then calculates the sum and average of the values in the matrix.

Exercise2:
Write a program that fills an N-dimensional square matrix with integer values. Then fill the first diagonal with 0s.

Exercise 3:
Write an algorithm that allows to fill an integer array T of size N and then search a value X in T.

Exercise 4:
Write an algorithm that allows filling an array with 10 notes, and then calculating the number of notes greater than 10.

Exercise 5:
Write an algorithm that allows entering the notes of 10 students who have 7 modules in a matrix M, then calculate the average of each student and store them in a table T. We assume that all the modules have the same coefficient.

Matrix of notes

Student\Notes	Module 1					Module 7
Student 1						
……						
student 10						

Array of average notes

Av 1	Av 2					Av 10

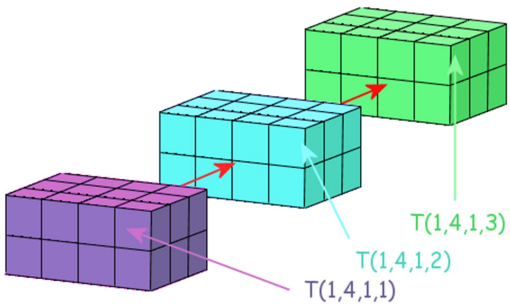
Exercise 6 :
Write an algorithm that finds the number of occurrences of X in a matrix.

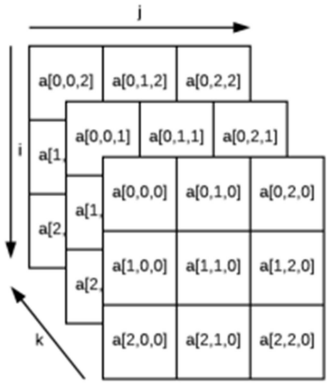
1.3. Multidimensional Arrays

An N-dimensional array is declared by specifying the number and type of values it will contain.

Exemples :
T1 : array [1..100, 1..100, 1…300] of integer;
T2 : array [1..10, 1..47, 1..12, 1..7] of real ;

The first is a three-dimensional array and the second is a four-dimensional array.





The reference of an element in an N-dimensional array is done in the same way as for a one or two-dimensional array, by specifying the index of each dimension.

2. Character strings

2.1. Definition

Strings are finite sequences of characters. The number of characters making the string is the length of this string. A zero-length string does not include any characters which is called an Empty string. In algorithms, a character string is designated between two Quotation mark " ".

Example :

- The string "Algo" constituted of the characters 'A', 'l', 'g', 'o'.
- The string "123" is not the numerical quantity 123.
- The string "" represents an empty string.
- The string " " is a character string containing only one character which is the space (do not confuse with the empty string).

2.2. Declaration

Syntax in algorithms :

var

id : string [length] of characters; **OR** id : string [length];

Syntax in C language:

There is no special type for strings in C language. A string is declared as a one-dimensional array of char, whose end is indicated by the character '\0'. The size of the string is equal to the maximum length of the string plus one so that we can store the character '\0' denoting the end.

Syntax :

char id [length] ;

Example :

Algorithm	Program in C
var ch : string [10] ;	char ch [10]; // maximum string length is 9

2.3. Initialization of a character string

To initialize a string, we declare a charter string identifier and assign its initial values.

Example :

In algorithm:

Var

ch : string [15] ;

ch ← ''Hello'' ;

In C language:

char ch[ ] = { 'H','e','l','l','o', '\0'}; // allocate 6 boxes (6 bytes)

// Using a literal string to initialize an array of unspecified size saves from worrying about the NULL character ('\0') since it is part of the literal string.

char ch[ ] = ''Hello'' ;

// The computer automatically reserves the number of bytes needed for the string, i.e.: the number of characters + 1 (here: 6 bytes).

```
char ch[6] = ''Hello'' ;
```

// If we specify the size of your array, we have to make sure that it has enough space to accommodate the entire string, that is to say the characters that compose it and the NULL character.

2.4. Access to the elements of a String

Access to an element of a string can be done in the same way as accessing an element of an array. Example: For the previous ch string, we have:

```
ch [0] = 'H' ; ch [1] = 'e' ; ch [2] = 'l' ; ch [3] = 'l' ; ch [4] = 'o' ;
```

2.5. Reading and displaying strings

In Algorithms, a character string is read (displayed) globally (at once) and not character by character.

In Algorithms:

```
Read (ch);
Write (ch);
```

In C program:

We can read (display) a character string in two ways:

- First way: scanf ("%s", & ch); printf("%s", ch);
- Second way: gets (ch); puts(ch);

// The difference between scanf and gets is that scanf only brings the text entered before the first blank (space) into the variable to read. However, gets brings all text entered text until to the carriage return (enter command).

Exercise :

Write an algorithm that allows to read a string, then check if the string contains the character 'e'.

Exercise :

Write an algorithm that allows to read two character strings ch1 and ch2, then check if the string ch2 is included in the string ch1.

2.6. Comparison of character strings

In a computer system, each character is associated with a numerical value: its ASCII code (American Standard Code for Information Interchange). All the codes are listed in a table called "ASCII code table". When we store a character in memory (in a variable), we memorize its ASCII code.

To compare two character strings, we compare the characters of the same rank in the two strings starting with the first character of each string (the first character of the first string is compared to the first character of the second string, the second character of the first string is compared to the second character of the second string, and so on…).

Examples: Comparing two strings

• "baobab" < "sport" because the ASCII code of b (98 in base 10) is lower than the ASCII code of 's' (115 in base 10).

• "baobab" > "banana" because the ASCII code of 'o' (111) is greater than the ASCII code of 'n' (110). The comparison cannot be made on the first two characters because they are identical.

• "333" > "1230" because the ASCII code of '3' (51) is greater than the ASCII code of '1' (49). Please note, here it is not numerical values that are compared, but rather characters.

• "333" < "3330" because the second string has a length greater than that of the first (the comparison cannot be made on the first three characters because they are identical).

• "Baobab" < "baobab" because the ASCII code of 'b' (98) is greater than the ASCII code of 'B' (66).

Exercise 1:

Write an algorithm that takes an uppercase string and converts it to lowercase.
Note :
the ASCII code for lowercases alphabetic characters are between 97 (a) and 122 (z).
the ASCII code for uppercases alphabetic characters are between 65 (A) and 90 (Z).

Exercise 2:

Consider a *Length (string)* function which returns the length of a character string.

Exanmple :

ch ← '' Hello '' ;
write (length (ch)) ; // display 5.

Write an algorithm that allows reading two strings of characters and then compare them if they are similar.

Solution :

```
Algorithm comparison;
Var
    ch1, ch2: string [10];
    i, j, k: integer;
Begin
 read (ch1, ch2);
 j ←length(ch1);
 k← length (ch2);
 i←1 ;
 While ( ( ch1[i] = ch2[i] ) AND (i < j) AND (k=j) ) Do
      i++;
 EndWhile ;
 If ( i = j )   Then
    write("ch1 = ch2");
 else
     write("ch1 ≠ ch2");
 endif ;
end.
```

2.7. Character string manipulation functions in C language

The <string.h> library contains several functions for manipulating character strings. Among them :

- The strlen(ch) function: provides the length of a string as a result.
- The strcat(ch1,ch2) function: copies the second string ch2 after the first string ch1.
- The strcmp(ch1, ch2) function: compares two strings and provides a positive integer value if ch1>ch2, zero if ch1=ch2 and negative if ch1<ch2.
- The strcpy(destination, source) function: copies the source string into the destination string address.