Practical Work series N° 1 (continued): Subroutines and Arrays

1. Vectors: (one-dimensional arrays):

1.1. Declaration and use:

The declaration and use of vectors in subroutines (functions and procedures) is similar to their use in main programs, unless used as formal parameters, the dimension does not need to be specified.

Example :

➢ Here is a C++ program that reads and displays a vector.

```
#include <iostream>
Using name space std ;

//A module that allows you to fill a vector with n integer values.
Void read_VE ( int tab[ ], int n)
{
int i;
for (i=0;i<=n-1; i++)
cin >> tab[i];
}
//A module that allows you to display the values of a vector of integers.
Void Show_VE ( int tab[], int n)
{
int i;
for (i=0;i<=n-1; i++)
cout <<tab[i];
}
// Main Program
int main()
{
const int max_array_size = 100;
int size;
int v[ max_array_size ];
cout <<"\n Give the number of elements of the vector: ";
cin >>size;
cout <<"\n Give the elements of the vector: \n";
read_VE (v, size); // function call
cout << "\n The elements of the vector are: \n";
Show_ VE ( v, size ); // function call
system( "PAUSE"); //to stop the window from closing
return 0;
}
```

Do not put the vector dimension here

Exercise 1 :

Write a C++ program which allows you to read a vector V1 of real numbers of size N and check whether it is sorted in crossing order or not.

2. **Matrices:** (two-dimensional arrays):

Same thing for matrices, the only difference is that we must specify the second dimension (number of columns) **in the formal parameters .**

Example :

A program that reads and displays a matrix using the functions:

```
#include <iostream>
using namespace std;
void read_matrix ( int mat[ ][ 10 ], int d1 , int d2 )
{
int i,j ;
for(i=0;i<=d1-1;i++)
for(j=0;j<=d2-1;j++)
cin >> mat[i][j];
}
void display_matrix ( int mat[ ][10], int d1, int d2)
{
int i,j ;
for(i=0;i<=d1-1;i++)
{
for(j=0;j<=d2-1;j++)
cost <<mat[i][j]<<” ”;
cout<<”\n”;
}
}
```

You have always specify the number of columns

```
int main () {
int m[20 ][ 10],n1,n2;

cout<<"\ nType the number of rows in the matrix: ";
cin >> n1;

cout <<"\ nType the number of columns in the matrix: ";
cin >>n2;

cout<<"\ nType the elements of the matrix :\ n\n";
read_ matrix ( m,n1,n2);

cout<<"\n The elements of the matrix are :\ n\n";
display_matrix (m ,n1,n2 );

system("PAUSE"); //to stop closing the execution window
return 0;
}
```

procedure call : read_ matrix

Noticed :

- Arrays are always passed by variable, so there is no need to put “*” in the output parameters.

Exercise 2:

Write a C++ program that reads a matrix and calculates its transpose; the rows of the initial matrix become the columns of the transposed matrix, then displays it.

4	***12***	***33***	***7***
0	***7***	***12***	***11***
6	***5***	***4***	***13***

$\longrightarrow$

4	0	6
12	7	5
33	12	4
7	11	13