Practical Work No. 5
One-dimensional arrays

Goals

- Definition and classic use of tables.
- Know how to code repetitions for tables in C language
- Change the order of elements in an array (adding, deleting, displaying elements).

I. Declaration

Syntax:

<simple type><Array name>[<dimension>];

Examples*:*

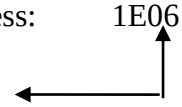
```
int T [20] ;
float T1 [100] ;
char T2[20] ;
```

II. Memorization

In C, the name of an array is the representative of the address of the first element of the array. The addresses of the other components are calculated (automatically) relative to this address.

Example : intT[5] = {100,200,300,400,500};

……	**100**	**200**	**300**	**400**	**500**	……
Address:	1E06	1E08	1E0A	1E0C	1E0E	1E10



T

If the size is not explicitly specified during initialization, the computer automatically reserves the necessary number of bytes.

Examples

int A [] = {10, 20, 30, 40, 50};
==>reservation of 5*sizeof(int) bytes (in our case: 10 bytes)
float B [] = {-1.05, 3.33, 87e-5, -12.3E4};
==>reservation of 4*sizeof(float) bytes (in our case: 16 bytes)
char C [] = {'a', 'b', 'c', 'd', 'e'};
==>reservation of 5*sizeof(char) bytes (in our case: 5 bytes)

III. Access to the components of an array

Consider an array T of dimension N:
Algorithmically,
- access to the first element of the array is done by **T[1]**
- access to the last element of the array is done by **T[N]**

In C,
- access to the first element of the array is done by **T[0]**
- access to the last element of the array is done by **T[N-1]**

Example : int T[5] = {100,200,300,400,500};

Name: T	**100**	**200**	**300**	**400**	**500**
index :	**0**	**1**	**2**	**3**	**4**
Content :	**T[0]**	**T[1]**	**T[2]**	**T[3]**	**T[4]**

IV. Filling an array

```
#include <stdio.h>
void main(){
        int T[5];
        int i; /* Counter */
        for (i=0; i<5; i++)  {
                printf("T[%d]:", i);
                scanf("%d", &T[i]);
         }
}
```

V. Displaying an array

```
#include <stdio.h>
void main(){
        int T[5], i;
        for (i=0; i<5; i++)
                printf("%d \t :",T[i]);
}
```

VI. Work requested

Exercise 1:
Write a program that reads the dimension N of an array T of type int (maximum dimension: 50 components), fills the array with values entered on the keyboard and displays the array.
Then calculate and display the sum, the product, and the average of the elements of the array.

Exercise 2:

Write a C program that reads the dimension **N** of an integer array **T** (maximum size: 50 elements).
The program should:

1. Fill the array with values entered by the user,
2. Display all the elements of the array,
3. After reading an integer **x**, count and display how many times **x** appears in the array.

Exercise 3:
Write a program that reads the dimension N of an array T of type int (maximum dimension: 50 components), fills the array with values entered on the keyboard and displays the array.
Then arrange the elements of array T in reverse order without using a helper array. Display the resulting array.
Idea: Swap array elements using two indices that traverse the array starting at the beginning and end of the array respectively and meeting in the middle.

Exercise 4: Searching an element
Let T be an array containing n integers. Write a C program that reads an integer x and determines the index of the first occurrence of x in T. If x is not present, the program should display -1.

Exercise 5: Filling with distinct elements

Write a C program that fills an array T with n integers such that all elements are distinct (no integer is repeated).

Exercise 6: Counting distinct elements

Write a C program that reads an integer **n** ($10 < n \leq 100$) and then reads **n** integers into an array **T**. The program calculates and displays the number of distinct elements in **T**.

Exercise 7: Frequency of elements

Write a C program that counts and displays the frequency of each element in an array **T**. The elements are integers between 1 and 100.

Exercise 8: removing the zero occurrences

Write a program that reads the dimension N of an array T of type int (maximum dimension: 50 components), fills the array with values entered on the keyboard and displays the array.
Then delete all occurrences of the value 0 in the array T and pack the remaining elements. Display the resulting array.

Exercise 9: Common Elements
Write a C program that:

1. Fills a first array **T1** with **n1** values entered by the user,
2. Fills a second array **T2** with **n2** values entered by the user,
3. Displays all the elements of the first array,
4. Displays all the elements of the second array,
5. Determines and displays the number of elements they have in common.

Exercise 9: Ordered Elements
Write a C program that:

1. Fills a first array **T1** with **n1** values entered by the user,
2. Fills a second array **T2** with **n2** values entered by the user (**n2 ≤ n1**),
3. Displays all the elements of the first array,
4. Displays all the elements of the second array,
5. Checks whether all the elements of **T2** appear in the same order within **T1**.