جامعة
القاهرة الجديدة
التكنولوجية

NEW CAIRO
TECHNOLOGICAL
UNIVERSITY

TUE – The Technological Universities in Egypt

NCTU – New Cairo Technological University

Faculty of Industry and Energy Technology

ICT Department

First-Year

Course: C Programming

Lecture 7

Presented by

Dr. Ghada Maher

C Array

An array is defined as the collection of similar type of data items stored at contiguous memory locations. Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc.

It also has the capability to store the collection of derived data types, such as pointers, structure, etc. The array is the simplest data structure where each data element can be randomly accessed by using its index number.

Properties of Array

Advantage of C Array

1) Code Optimization: Less code to the access the data.

2) Ease of traversing: By using the for loop, we can retrieve the elements of an array easily.

3) Ease of sorting: To sort the elements of the array, we need a few lines of code only.

4) Random Access: We can access any element randomly using the array.

Disadvantage of C Array

1) Fixed Size: Whatever size, we define at the time of declaration of the array, we can't exceed the limit. So, it doesn't grow the size dynamically like LinkedList which we will learn later.

Declaration of C Array

We can declare an array in the c language in the following way.

```
data_type array_name[array_size];
```

```
int marks[5];
```

Here, int is the *data_type*, marks are the *array_name*, and 5 is the *array_size*.

Initialization of C Array

The simplest way to initialize an array is by using the index of each element. We can initialize each element of the array by using the index. Consider the following example.

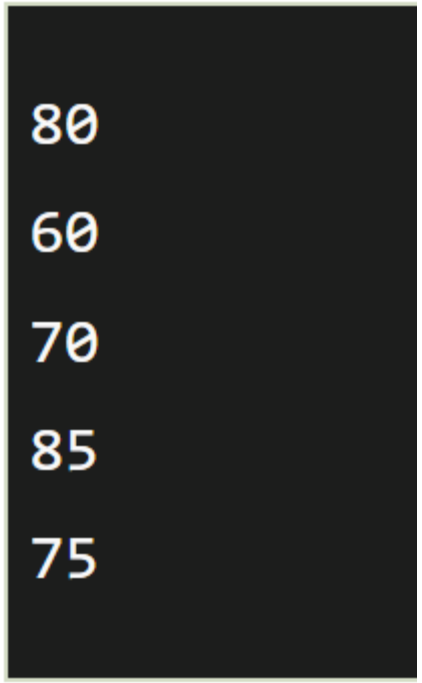
```
marks[0]=80;//initialization of array
marks[1]=60;
marks[2]=70;
marks[3]=85;
marks[4]=75;
```

Initialization of Array

C array example

```
#include<stdio.h>
int main(){
int i=0;
int marks[5];//declaration of array
marks[0]=80;//initialization of array
marks[1]=60;
marks[2]=70;
marks[3]=85;
marks[4]=75;
//traversal of array
for(i=0;i<5;i++){
printf("%d \n",marks[i]);
}//end of for loop
return 0;
}
```

Output



C Array: Declaration with Initialization

We can initialize the c array at the time of declaration. In such case, there is no requirement to define the size. So it may also be written as the following code.

```
int marks[5]={20,30,40,50,60};
```

Example.

```c
#include<stdio.h>
int main(){
int i=0;
int marks[5]={20,30,40,50,60};//declaration and initialization of array
 //traversal of array
for(i=0;i<5;i++){
printf("%d \n",marks[i]);
}
return 0;
}
```

Output

```
20
30
40
50
60
```

C Array Example: Sorting an array

```c
#include<stdio.h>
void main ()
{
    int i, j,temp;
    int a[10] = { 10, 9, 7, 101, 23, 44, 12, 78, 34, 23};
    for(i = 0; i<10; i++)
    {
        for(j = i+1; j<10; j++)
        {
            if(a[j] > a[i])
            {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }
    printf("Printing Sorted Element List ...\n");
    for(i = 0; i<10; i++)
    {
        printf("%d\n",a[i]);
    }    }
```

Output

```
Printing Sorted Element List ...
101
78
44
34
23
23
12
10
9
7
```

Program to print the largest and second largest element of the array.

```c
#include<stdio.h>
void main ()
{
    int arr[100],i,n,largest,sec_largest;
    printf("Enter the size of the array?");
    scanf("%d",&n);
    printf("Enter the elements of the array?");
    for(i = 0; i<n; i++)
    {
        scanf("%d",&arr[i]);
    }
    largest = arr[0];
    sec_largest = arr[1];
    for(i=0;i<n;i++)
    {
        if(arr[i]>largest)
        {
            sec_largest = largest;
            largest = arr[i];
        }
        else if (arr[i]>sec_largest && arr[i]!=largest)
        {
            sec_largest=arr[i];
        }
    }
    printf("largest = %d, second largest = %d",largest,sec_largest);
}
```

Output

```
Enter the size of the array?5
Enter the elements of the array?5 6 3 2 10
largest = 10, second largest = 6
```

Two Dimensional Array in C

The two-dimensional array can be defined as an array of arrays.

The 2D array is organized as matrices which can be represented as the collection of rows and columns. However, 2D arrays are created to implement a relational database lookalike data structure.

Declaration of two dimensional Array in C

The syntax to declare the 2D array is given below.

```
data_type array_name[rows][columns];
```

```
int twodimen[4][3];
```

Here, 4 is the number of rows, and 3 is the number of columns.

Initialization of 2D Array in C

In the 1D array, we don't need to specify the size of the array if the declaration and initialization are being done simultaneously. However, this will not work with 2D arrays.

We will have to define at least the second dimension of the array.

The two-dimensional array can be declared and defined in the following way.

```
int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
```

Two-dimensional array example in C

```c
#include<stdio.h>
int main(){
int i=0,j=0;
int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
//traversing 2D array
for(i=0;i<4;i++){
 for(j=0;j<3;j++){
   printf("arr[%d] [%d] = %d \n",i,j,arr[i][j]);
 }//end of j
}//end of i
return 0;
}
```

Output

```
arr[0][0] = 1
arr[0][1] = 2
arr[0][2] = 3
arr[1][0] = 2
arr[1][1] = 3
arr[1][2] = 4
arr[2][0] = 3
arr[2][1] = 4
arr[2][2] = 5
arr[3][0] = 4
arr[3][1] = 5
arr[3][2] = 6
```

C 2D array example: Storing elements in a matrix and printing it.

```c
#include <stdio.h>
void main ()
{
   int arr[3][3],i,j;
   for (i=0;i<3;i++)
   {
      for (j=0;j<3;j++)
      {
         printf("Enter a[%d][%d]: ",i,j);
         scanf("%d",&arr[i][j]);
      }
   }
   printf("\n printing the elements ....\n");
   for(i=0;i<3;i++)
   {
      printf("\n");
      for (j=0;j<3;j++)
      {
         printf("%d\t",arr[i][j]);
      }
   }
}
```

Output

```
Enter a[0][0]: 56
Enter a[0][1]: 10
Enter a[0][2]: 30
Enter a[1][0]: 34
Enter a[1][1]: 21
Enter a[1][2]: 34

Enter a[2][0]: 45
Enter a[2][1]: 56
Enter a[2][2]: 78

 printing the elements ....

56      10      30
34      21      34
45      56      78
```