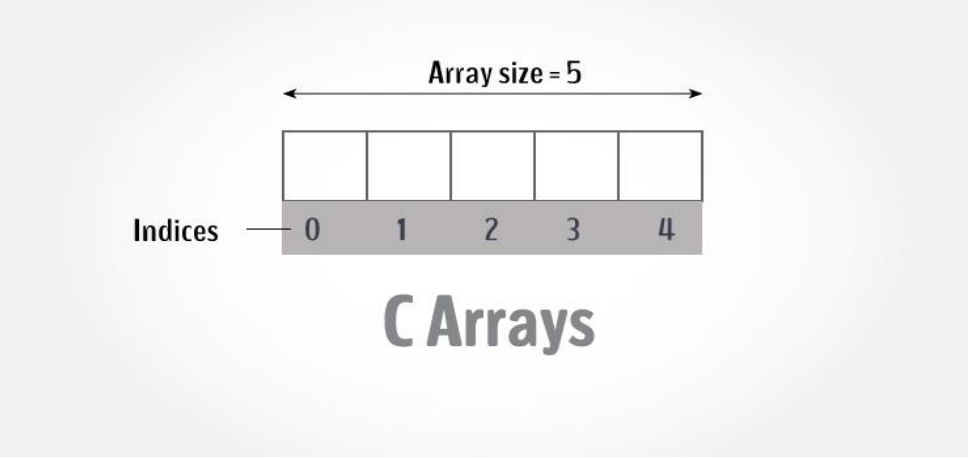
C Fundamental

Arrays



Objectives

- Array Introduction
- Define an array
- Array handling and initialize an array in C
- Two dimensional arrays
- Advantage and Disadvantage of Array

Section 1

ARRAY INTRODUCTION

Advantage and Disadvantage

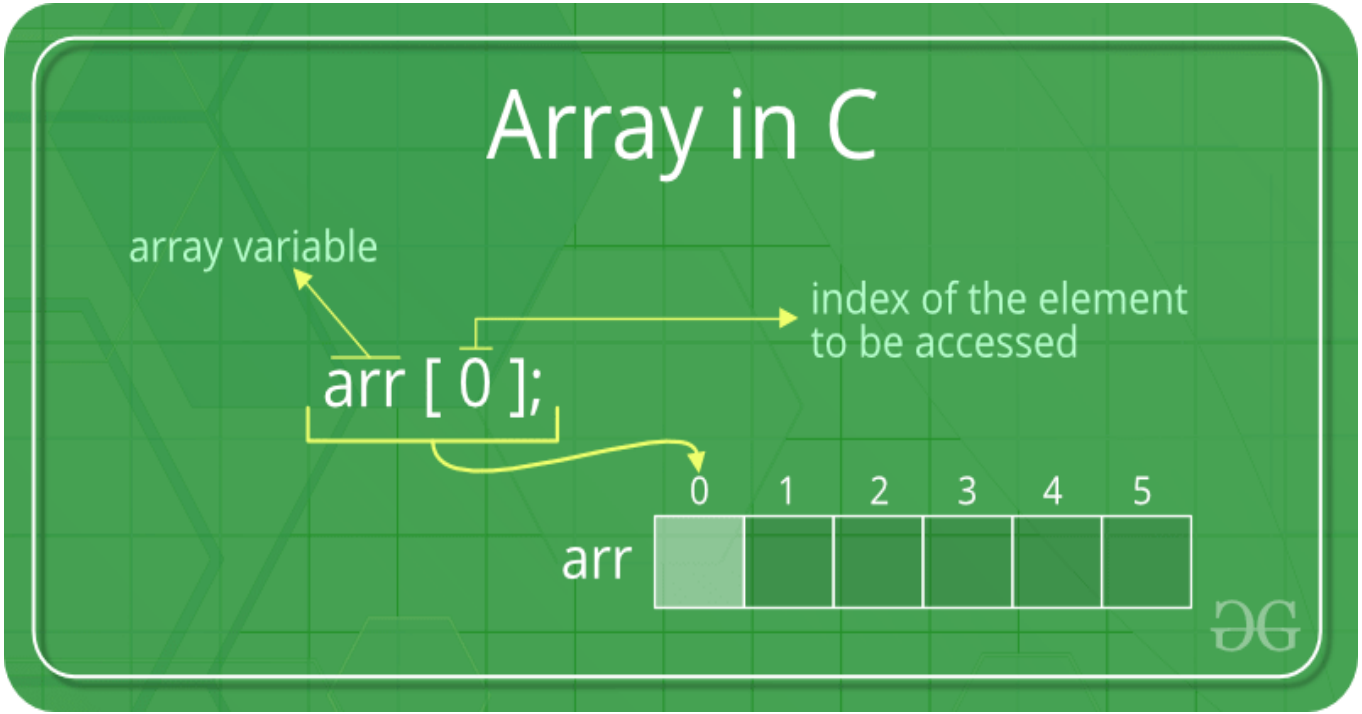
Advantage of C Array

- **Code Optimization**: Less code to the access the data.
- **Ease of traversing**: By using the for loop, we can retrieve the elements of an array easily.
- **Ease of sorting**: To sort the elements of the array, we need a few lines of code only.
- **Random Access**: We can access any element randomly using the array.

Disadvantage of C Array

- **Fixed Size**: Whatever size, we define at the time of declaration of the array, we can't exceed the limit.

Array Elements & Indexes (1)



Array Elements & Indexes (2)

- Each member of an array is identified by unique index or subscript assigned to it
- The dimension of an array is determined by the number of indices needed to uniquely identify each element
- An index is a positive integer enclosed in [ ] placed immediately after the array name
- An index holds integer values starting with zero
- An array with 11 elements will look like:

arr[0], arr[1], arr[2],…. arr[10]

Section 2

DEFINING AN ARRAY

Defining an Array (1)

- An array has some particular characteristics and has to be defined with them
- These characteristics include –

✓ *Storage Class*

✓ *Data Types of the elements in the Array*

✓ *Array Name* (Which indicates the location of the first member of the array)

✓ *Array Size* (A constant value)

Defining an Array (2)

- An array is defined in the same way as a variable is defined.
- The only change is that the array name is followed by one or more expressions, enclosed within square brackets [], specifying the array dimension.
- Storage_Class data_types array_name[size]

```
int player[11];
```

Norms with Arrays

- All elements of an array are of the same type
- Each element of an array can be used wherever a variable is allowed or required
- Each element of an array can be referenced using a variable or an integer expression
- Arrays can have their data types like int, char, float or double

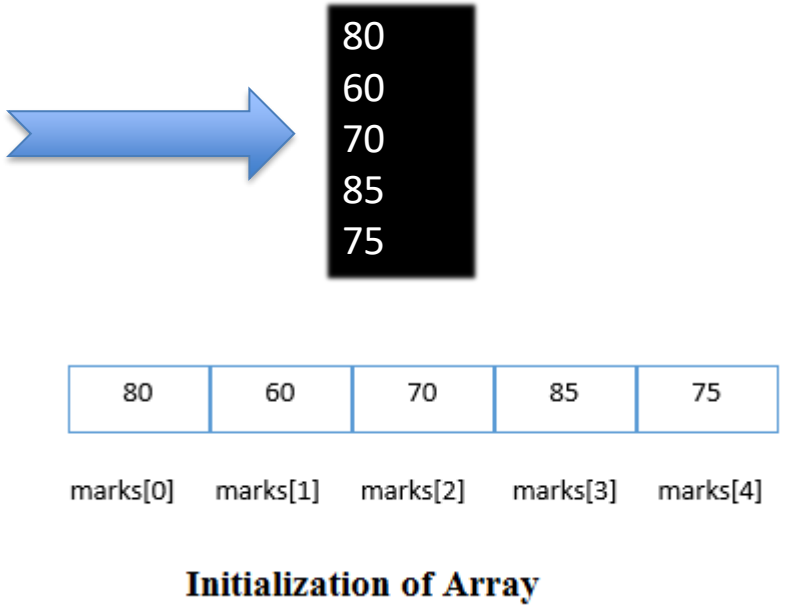
Section 3

ARRAY HANDLING AND INITIALIZATION

Array Initialization

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:

```
#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;
    for (i = 0; i < 5; i++){       //traversal of array
        printf( "%d \n", marks[i]);
    }                              //end of for loop
    return 0;
}
```



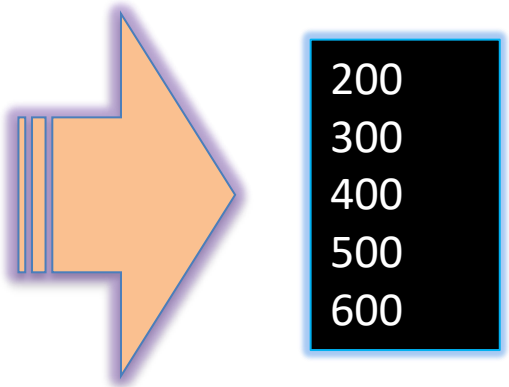
Initialization of Array

Array Initialization (1)

- We can initialize the C array at the time of declaration. Let's see the snippet code:
- int arr[5] = {20, 30, 40, 50, 60};
- In such case, there is **no requirement to define the size**. So it may also be written as the following code:
- int arr[] = {20, 30, 40, 50, 60};

Array Initialization (2)

```
#include<stdio.h>
int main(){
	int i = 0;
	//declaration and initialization of array
	int arr[5] = { 200, 300, 400, 500, 600};
	//traversal of array
	for( i = 0; i < 5; i++){
		printf("%d \n", arr[i]);
	}
	return 0;
}
```



Array Handling in C - 1

- An array is treated differently from a variable in C
- Two arrays, even if they are of the same type and size cannot be tested for equality
- It is not possible to assign one array directly to another
- Values cannot be assigned to an array on the whole, instead values are assigned to the elements of the array

Array Handling in C - 2

```c
#include <stdio.h>
void main()
{
    int arr[3];
    int num1 = 2021, num2 = 0;
    // Enter inputs for the array
    for(int i = 0; i < 3; i++){
       printf("\n Enter value: %d : ", i + 1);
       scanf("%d", &arr[i]);
    }
    // Assigns a value to an element in the array
    arr[0] = num1;
    // Get a value from an element in the array
    num2 = arr[1];
    // Changes the value of an element in an array
    arr[2] = 69;
    arr[1] = arr[2];
    for(int i = 0; i < 3; i++){
       printf("\n Value of element %d is %d", i + 1, arr[i]);
    }
}
```

```
 Value of element 1 is 2021
 Value of element 2 is 69
 Value of element 3 is 69

...Program finished with exit code 0
Press ENTER to exit console.
```

Array Handling in C - 3

Print out an array from end to begin. Example:

```
#include <stdio.h>
void main()
{
    int arr[5];
    int num = 2021;
    // Enter inputs for the array
    for(int i = 0; i < 5; i++){
       printf("\n Enter value: %d : ", i + 1);
       scanf("%d", &arr[i]);
    }
    // Print array from end to begin
    for(int i = 4; i >= 0; i--){
       printf("\n Value of element %d is %d", i + 1, arr[i]);
    }
}
```

Inputs value from 1 to 5 =>

```
Enter value: 1 : 1

Enter value: 2 : 2

Enter value: 3 : 3

Enter value: 4 : 4

Enter value: 5 : 5

Value of element 5 is 5
Value of element 4 is 4
Value of element 3 is 3
Value of element 2 is 2
Value of element 1 is 1

...Program finished with exit code 0
Press ENTER to exit console.
```

Array Handling in C - 4

```
/* Input values are accepted from the user into the array ary[10]*/
#include <stdio.h>
void main()
{
    int ary[10];
    int i, total, high;
    for(i = 0; i < 10; i++){
       printf("\n Enter value: %d : ", i + 1);
       scanf("%d",&ary[i]);
    }
    /* Displays highest of the entered values */
    high = ary[0];
    for(i = 1; i < 10; i++){
       if(ary[i] > high)
         high = ary[i];
    }
    printf("\nHighest value entered was %d", high);
    /*  prints average of values entered for ary[10] */
    for(i = 0,total = 0; i < 10; i++)
       total = total + ary[i];
    printf("\nThe average of the elements of ary is %d",total/i);
}
```

Array Handling in C - 5

Swap elements in array. Example:

```c
#include <stdio.h>
void main()
{
    int arr[5], temp;
    // Enter inputs for the array
    for(int i = 0; i < 5; i++){
       printf("\n Enter value: %d : ", i + 1);
       scanf("%d", &arr[i]);
    }
    // Swap 1st-element with 3rd-element
    temp = arr[0];
    arr[0] = arr[2];
    arr[2] = temp;
    // Print array after change
    for(int i = 0; i < 5; i++){
       printf("\n Value of element %d is %d", i + 1, arr[i]);
    }
}
```

```
 Enter value: 1 : 1

 Enter value: 2 : 2

 Enter value: 3 : 3

 Enter value: 4 : 4

 Enter value: 5 : 5

 Value of element 1 is 3
 Value of element 2 is 2
 Value of element 3 is 1
 Value of element 4 is 4
 Value of element 5 is 5

...Program finished with exit code 0
Press ENTER to exit console.
```

Array Handling in C - 6

Swap elements between two arrays – source code:

```
#include <stdio.h>
void main(){
    int arr1[4], arr2[4], temp;
    // Enter inputs for two arrays
    for(int i = 0; i < 4; i++){
       printf("\n Enter value for 1st-array: %d : ", i + 1); scanf("%d", &arr1[i]);
    }
    for(int i = 0; i < 4; i++){
       printf("\n Enter value for 2nd-array: %d : ", i + 1); scanf("%d", &arr2[i]);
    }
    // Swap two arrays:
    for(int i = 0; i < 4; i++){
        temp = arr1[i]; arr1[i] = arr2[i]; arr2[i] = temp;
    }
    // Print two arrays
    printf("\n Value of elements in 1st-array after swapping:");
    for(int i = 0; i < 4; i++)
        printf(" %d", arr1[i]);
    printf("\n Value of elements in 2nd-array after swapping:");
    for(int i = 0; i < 4; i++)
        printf(" %d", arr2[i]);
}
```

Array Handling in C - 7

Swap elements between two arrays – output:

```
Enter value for 1st-array: 1 : 1

Enter value for 1st-array: 2 : 2

Enter value for 1st-array: 3 : 3

Enter value for 1st-array: 4 : 4

Enter value for 2nd-array: 1 : 5

Enter value for 2nd-array: 2 : 6

Enter value for 2nd-array: 3 : 7

Enter value for 2nd-array: 4 : 8

Value of elements in 1st-array after swapping: 5 6 7 8
Value of elements in 2nd-array after swapping: 1 2 3 4

...Program finished with exit code 0
Press ENTER to exit console.
```

Array Handling in C - 8

Sum of all elements in array. Example:

```
#include <stdio.h>
void main()
{
    int arr[5], sum = 0;
    // Enter inputs for the array
    for(int i = 0; i < 5; i++){
       printf(" Enter value: %d : ", i + 1);
       scanf("%d", &arr[i]);
    }
    // Add all elements
    for(int i = 0; i < 5; i++){
        sum = sum + arr[i];
    }
    printf("\n Sum of elements is: %d", sum);

}
```

```
Enter value: 1 : 1
Enter value: 2 : 2
Enter value: 3 : 3
Enter value: 4 : 4
Enter value: 5 : 5

Sum of elements is: 15

...Program finished with exit code 0
Press ENTER to exit console.
```

Section 4

TWO-DIMENSIONAL ARRAYS

Two-Dimensional Arrays

- The simplest and the most commonly used multi-dimensional array is the two - dimensional array.
- A two-dimensional array can be thought of as an array of two single dimensional arrays.
- A two-dimensional array looks like a railway time-table consisting of rows and columns.

Two-Dimensional Arrays: Declaration

- The syntax to declare the 2D array is given below:

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

- Example:

```
int arr[3][4];
```

Initialization of Two-Dimensional Arrays-1

```
int ary[3][4] =
{1,2,3,4,5,6,7,8,9,10,11,12};
```

The result of the above assignment will be as follows :

ary [0] [0] = 1	ary [0] [1] = 2	ary [0] [2] = 3	ary [0] [3] = 4
ary[1] [0] = 5	ary [1][1] = 6	ary [1] [2] = 7	ary [1][3] = 8
ary[2] [0] = 9	ary [2][1] = 10	ary [2] [2] = 11	ary [2][3] = 12

Initialization of Two-Dimensional Arrays-2

Another type of initialization:

```
int arrays[3][4]=
{
      {1,2,3,4},
      {5,6,7,8},
      {9,10,11,12}
};
```

Initialization of Two-Dimensional Arrays-3

The result of the assignment will be as follows :

ary[0][0] =1	ary[0][1]=2	ary[0][2]=3	ary[0][3]=0
ary[1][0]=4	ary[1][1]=5	ary[1][2]=6	ary[1][3]=0
ary[2][0]=7	ary[2][1]=8	ary[2][2]=3	ary[2][3]=0

A two - dimensional string array is declared in the following manner :

```
char str_ary[25][30];
```

Two-Dimensional Array – Example 1

```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;
}
```

Print out all elements of the Two-Dimensional Array

```
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
```

Two-Dimensional Array – Example 2

Storing elements in a matrix and printing it:

```
#include<stdio.h>
int 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]);
        }
    }
    return 0;
}
```

```
Enter a[0][0]: 1
Enter a[0][1]: 2
Enter a[0][2]: 3
Enter a[1][0]: 4
Enter a[1][1]: 5
Enter a[1][2]: 6
Enter a[2][0]: 7
Enter a[2][1]: 8
Enter a[2][2]: 9


 printing the elements ....

1       2       3
4       5       6
7       8       9

...Program finished with exit code 0
Press ENTER to exit console.
```

Thank you

Q&A