# ARRAYS

## Need of Arrays

- In C programming, one of the frequently arising problems is to handle similar types of data.
- ***For example:*** If the user wants to store marks of 100 students. This can be done by creating 100 variables individually but, this process is rather tedious and impracticable. This type of problem can be handled in C programming using arrays.
- An array variable can store more than one value of same data type, where a individual variable can store only value of any data type defined.

## Definition

- Array is the derived data type.
- Array is the homogeneous collection of data items.
- The elements of an array are of same data type and each item can be accessed using the same name.
- All the data items of an array are stored in consecutive memory locations in RAM.

***Note:*** *An array is a derived data type because it cannot be defined on its own; it is a collection of basic data types such as char, integer, double, and float.*

## Classification of arrays

- ✓ Single Dimension Array
- ✓ Multi Dimension Array

## Single Dimension Array (1D - Array)

- A single-dimensional array is a linear list containing the data items of the same type and stored in continuous memory location.

## Declaration of 1D Array

- Syntax: **Data_typeArray_name[ Array_size];**

  - ✓ **Data_type** –general data types like - int, char, float, double.
  - ✓ **Array_name** – valid identifier.
  - ✓ **Array_size -** maximum number of elements.

  **Example:**1. int arr [20] ;
  2. float arr1 [30] ;
  3. double arr2 [30] ;

## Pictorial Representation of Single Dimensional array

- Representation of an array **int arr[5]**, with five integers array elements and by assuming the memory location starting from 1024 –

| | | | | | |
|---|---|---|---|---|---|
| Memory Location | **1024** | **1026** | **1028** | **1030** | **1032** |
| Array Elements | **10** | **5** | **2** | **15** | **20** |
| Array Index | **arr[0]** | **arr[1]** | **arr[2]** | **arr[3]** | **arr[4]** |

We can note the following points from the above representation-

- "arr" is an array which can store maximum of five integer values.
- The array elements are arranged in continuous memory location starting from 1024 (assumed memory) has an incrementing of 2 bytes for integer values.
- The **memory allocated = size of data type * array_size.**
- For the above representation total memory is 10 bytes (2 bytes *5 elements).
- The array Index/ Subscript starts from **0** and the maximum Index / Subscript will be **Array_size -1**.
- Elements of the array is represented using array name with Subscript / Index as follows:

| ***Representation*** | ***Array Element*** |
|---|---|
| a[0] | 10 |
| a[1] | 5 |
| a[2] | 2 |
| a[3] | 15 |
| a[4] | 20 |

***Note:*** *The array values may be of any data type but the size and index should be integer only.*

***Example:*** *1. int arr[5] – valid*
*2. int arr[10.5] - invalid*

## Initialization of Arrays

The different types of initializing arrays:

1. At Compile time
   (i) Complete array initialization.
   (ii) Partial array initialization.
   (iii) Initialization without size.
   (iv) String initialization.
2. At Run Time

## Compile Time Initialization

- Assigning / providing the required value to the variable during declaration is termed as initialization.
- We can initialize the elements of arrays in the same way as the ordinary variables when they are declared.
- During Compile time - Arrays can be initialized at the time of declaration when their initial values are known in advance.
- The general form of initialization of arrays is:

**Data_typeArray_name [Array_size]={list of values to be assigned};**

### I. Complete Array Initialization:-

- ✓ In the complete array initialization **all the locations** of an array is assigned with some value during declaration.

**Ex:-**int arr[5]={10,15,1,3,20};

- ✓ During compilation, 5 contiguous memory locations are reserved by the compiler for the variable “arr” and all these locations are initialized as shown in figure.

| | | | | | |
|---|---|---|---|---|---|
| **arr[5]** | **10** | **15** | **1** | **3** | **20** |
| Array Index | **arr[0]** | **arr[1]** | **arr[2]** | **arr[3]** | **arr[4]** |

### II. Partial array initialization:-

- ✓ In partial array initialization **few locations** from starting location of an array is assigned with some value during declaration.
- ✓ If the number of values to be initialized is less than the size of the array, then the remaining locations will be initialized to zero automatically.

**Ex-** int a[5]={10,15};

- ✓ Even though compiler allocates 5 memory locations, using this declaration statement; the compiler initializes first two locations with 10 and 15, the next set of memory locations are automatically initialized to 0's by compiler as shown in figure.

| | | | | | |
|---|---|---|---|---|---|
| **arr[5]** | **10** | **15** | **0** | **0** | **0** |
| Array Index | **arr[0]** | **arr[1]** | **arr[2]** | **arr[3]** | **arr[4]** |

## III. Initialization without size:-

- ✓ In initialization without size of an array the size is not specified during the declaration but the values are specified.
- ✓ Consider the declaration along with the initialization.

  **Ex:-**int arr [ ]={10, 15, 1, 3, 20};

- ✓ In the above declaration, even though we have not specified exact number of elements to be used in array, the array size will be set depending on the total number of initial values specified. So, the array size will be set to 5 automatically by the compiler.

## IV. Array initialization with a string: -

- ✓ Here the string is initialized to the array and the data type will be character only.
- ✓ Consider the declaration with string initialization.

  **Ex:-** char str[ 6 ]="GRAPH";

- ✓ The array "str" is initialized as shown in figure.

| | | | | | | |
|---|---|---|---|---|---|---|
| **str[6]** | **'G'** | **'R'** | **'A'** | **'P'** | **'H'** | \0 |
| Array Index | **str[0]** | **str[1]** | **str[2]** | **str[3]** | **str[4]** | **str[5]** |

- ✓ In the above figure Array is initialized with a String Eventhough the string "GRAPH" contains 5 characters it need 6 locations, because string always ends with null character. So, the array_size is 6 (i.e., string length is 5 + 1 byte for null character).

  **Ex:-**

  char str1[9]="COMPUTER"; // correct

  char str1[8]="COMPUTER"; // wrong

- ✓ We can also observe that the individual characters of the string are occupying the single location in the array (G - str[0],R - str[1] and so on).

## Run Time Initialization

- ✓ In Run time initialization array is initialized explicitly at run time (During execution via output screen).
- ✓ This approach is usually applied for initializing large arrays and when initially values for an array are unknown.
- ✓ scanf() can be used to initialize an array at run time.

**Ex:**
```
int x[3];
scanf("%d%d%d",&x[0],&x[1],&x[2]);
```

- ✓ The above statements will initialize array elements with the values entered through the key board for specified locations of the array.

(Or)

- ✓ For the arrays with larger size,for( ) loop can be used for initialization of an array.

**Ex:** int x[10];

**Case1:***To initialize all thelocations of an array*
```
for(i=0;i<10;i++)
        scanf("%d", &x[i]);
```
- ➢ All 10 locations of an array "x" are initialized.

**Case2:***To initialize first "n" locations of an array (n – variable array_size<10)*
```
printf("Enter the array size\n");
scanf("%d",&n);
for(i=0;i<n;i++)
        scanf("%d", &x[i]);
```
- ➢ First "n" locations of an array "x" are initialized.

***Note:*** *The array values can be initialized for the individual location using assignment operator.*

***Ex:*** *int arr[3];*
```
        arr[0]=10;
        arr[1]=20;
        arr[2]=40;
```

## Displaying Array Elements

- ✓ The elements stored in an array can be displayed in the output screen using printf( ).

**Ex:**
```
int x[3];
printf("%d\t%d\t%d\n",x[0],x[1],x[2]);
```

- ✓ The above statements will displaythe elements stored under specified locations of the array.

(Or)

- ✓ For the arrays with larger size, for( ) loop can be used for displayingarray elements.

**Ex:** int x[10];

**Case1:***Todisplay all theelements stored in all the locations of an array*
```
for(i=0;i<10;i++)
        printf("%d\n", x[i]);
```
- ➢ All the elements in 10 locations of an array "x" are displayed.

**Case2:***Todisplay first "n" locations of an array (n – variable array_size<10)*

```
for(i=0;i<n;i++)
        printf("%d\n", x[i]);
```

- First "n"  elements at "n" locations of an array "x" are displayed.

## Exercise Programs:

1. **Write a C- Program to read a 1D - array with n elements and to print the same.**
2. **Write a C- Program to find the sum and average of all the elements in the array.**
3. **Write a C- Program to find the largest element and its position in the array.**
4. **Write a C- Program to generate the Fibonacci series using arrays.**
5. **Write a C- Program to find the sum of odd elements and even elements stored in an array.**
6. **Write a C- Program to print the twin primes up to specified limit.**
7. **Write a C- Program to generate 100 random integers in the range 1-100, store them in an array and print the average.**
8. **Write a C- Program to print the average of the given numbers and also the numbers greater than the average.**
9. **Write a C- Program to convert the decimal value to binary value.**
10. **Write a C- Program to convert the binary value to decimal value.**
11. **Write a C- Program to find the smallest and largest elementin the array.**
12. **Write a C- Program to evaluate the Polynomial using $a_0x^0+a_1x^1+a_2x^2+\dots+a_nx^n$Horner's method.**
13. **Write a C- Program to swap the content of two arrays.**
14. **Write a C- Program to compute the following:**

    **C[0]=a[0]+b[9]**
    **C[1]=a[1]+b[8]**
    **C[2]=a[2]+b[7]**
    **C[3]=a[3]+b[6]**
    **C[4]=a[4]+b[5]**
    **C[5]=a[5]+b[4]**
    **C[6]=a[6]+b[3]**
    **C[7]=a[7]+b[2]**
    **C[8]=a[8]+b[1]**
    **C[9]=a[9]+b[0]**

15. **Write a C- Program to delete the duplicate elements from an array.**

## Operations on Arrays

- ✓ Inserting an element to an array.
- ✓ Deleting an element from an array.
- ✓ Searching an element from an array.
- ✓ Sorting the elements of an array.

## Inserting an Element to an array:

- ✓ This operation helps in adding a new element in to an array.
- ✓ For adding the element in to the array two thing are necessary-
  1. Position of the array.
  2. Value to be inserted.

**Ex:**

- ✓ Let us consider we have linear array "a" as below-

| | | | | | |
|---|---|---|---|---|---|
| a[10] | 10 | 15 | 1 | 3 | 20 |
| Array Index | a[0] | a[1] | a[2] | a[3] | a[4] |

- ✓ Now consider the new element to be inserted is 50 at 3 location.
- ✓ To insert 50 at 3rdPosition, make third position as empty by shifting the elements by one place towards right till 3rdposition encounter.

| Position | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| a[10] | 10 | 15 | 1 | 3 | 20 | 0 |
| Array Index | a[0] | a[1] | a[2] | a[3] | a[4] | a[5] |
| | | | | | $\longrightarrow$ | |

| Position | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| a[10] | 10 | 15 | 1 | 3 | 20 | 20 |
| Array Index | a[0] | a[1] | a[2] | a[3] | a[4] | a[5] |
| | | | | $\longrightarrow$ | | |

| Position | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| a[10] | 10 | 15 | 1 | 3 | 3 | 20 |
| Array Index | a[0] | a[1] | a[2] | a[3] | a[4] | a[5] |

$\longrightarrow$

| Position | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| a[10] | 10 | 15 | 1 | 1 | 3 | 20 |
| Array Index | a[0] | a[1] | a[2] | a[3] | a[4] | a[5] |

*

- ✓ Once we reached 3rd position, insert 50 to array by using a[pos-1] = 50.

**Logic:**

```
if(pos<=n)
{
 for(i = n-1;i>=pos-1;i--)
      {
      a[i+1] = a[i];
      }
      a[pos-1] = value;

      n=n+1;
}
```

**Program to insert an element in to the array at specified location**

```
#include <stdio.h>
#include <conio.h>

void main()
{
int array[100], pos, i, n, ele;

printf("Enter number of elements in array\n");
scanf("%d",&n);

printf("Enter %d elements in to array\n", n);

for (i = 0; i< n; i++)
scanf("%d", &array[i]);

printf("Enter the location to insert an element\n");
scanf("%d", &pos);

printf("Enter the Element to insert\n");
scanf("%d", &ele);

if(pos<=n)
{
 for (i = n - 1; i>= pos - 1; i--)
        {
               array[i+1] = array[i];
        }
 array[pos-1] = ele;

 n=n+1;

 printf("Resultant array is\n");

 for (i = 0; i< n; i++)
 printf("%d\n", array[i]);
}
else
Printf("Insertion is not Possible\n");


getch();
}
```

## Deleting an element from an array

- ✓ This operation helps in deleting an element from an array.
- ✓ For deleting the element from the array, we need the element position to be deleted.
- ✓ Array is to be traversed to find the required position, once the position is found the element is deleted. Then the element deleted along with the position is displayed.

**Ex:**

- ✓ Let us consider we have linear array "a" as below-

| a[10] | 10 | 15 | 1 | 3 | 20 |
|---|---|---|---|---|---|
| Array Index | a[0] | a[1] | a[2] | a[3] | a[4] |

- ✓ Now consider the element stored at position 2 to be deleted.
- ✓ To delete 15 at 2ndPosition shift the elements from 3rd to 5th position by one place towards left.

| Position | 1 | 2 ← | 3 | 4 | 5 |
|---|---|---|---|---|---|
| a[10] | 10 | 15 | 1 | 3 | 20 |
| Array Index | a[0] | a[1] | a[2] | a[3] | a[4] |

|  |  |  | ← |  |  |
|---|---|---|---|---|---|
| Position | 1 | 2 | 3 | 4 | 5 |
| a[10] | 10 | 1 | 1 | 3 | 20 |
| Array Index | a[0] | a[1] | a[2] | a[3] | a[4] |

|  |  |  |  | ← |  |
|---|---|---|---|---|---|
| Position | 1 | 2 | 3 | 4 | 5 |
| a[10] | 10 | 1 | 3 | 3 | 20 |
| Array Index | a[0] | a[1] | a[2] | a[3] | a[4] |

| Position | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| a[10] | 10 | 1 | 3 | 20 | 20 |
| Array Index | a[0] | a[1] | a[2] | a[3] | a[4] |

✓ Once the element is deleted print the element deleted and its position.

**Program:**

```
#include <stdio.h>
#include<conio.h>

Void main()
{
int array[100],pos, i, n, ele;

printf("Enter number of elements in array\n");
scanf("%d",&n);

printf("Enter %d elements\n", n);

for( i=0; i< n ; i++)
scanf("%d",&array[i]);

printf("Enter the location where you wish to delete element\n");
scanf("%d",&pos);

if(pos>n)
printf("Deletion not possible.\n");
else
{
ele=array[pos];
printf("The Element deleted is %d at %d position",ele,pos);
for( i=pos-1; i< n -1;i++)
array[i]= array[i+1];
}
n=n-1;
printf("Resultant array is\n");
for( i =0; i< n ;i++)
printf("%d\n", array[i]);

getch();
}
```