

ARRAYS

An array is a collection of items stored at contiguous memory locations and elements can be accessed randomly using indices of an array. They are used to store similar type of elements as in the data type must be the same for all elements. They can be used to store collection of primitive data types such as int, float, double, char, etc of any particular type. To add to it, an array in C can store derived data types such as the structures, pointers etc.

In C language, arrays are referred to as structured data types. An array is defined as finite ordered collection of homogenous data, stored in contiguous memory locations.

Here the words,

finite means data range must be defined.

ordered means data must be stored in continuous memory addresses.

homogenous means data must be of similar data type.

Why do we need arrays?

We can use normal variables (v1, v2, v3, ..) when we have a small number of objects, but if we want to store a large number of instances, it becomes difficult to manage them with normal variables. The idea of an array is to represent many instances in one variable.

Consider a problem to find the average of marks secured by five students in a course.

```
int main()
{
    int marks1=70, marks2=65, marks3=74, marks4=85, marks5=78;
    int sum;
    float average;
    sum= marks1+marks2+marks3+marks4+marks5;
    average=sum/5.0;
    printf ("Average marks=%d", average);
    return 0;
}
```

Since there are only five students, it is possible to find the average in the above-mentioned manner. Now suppose there are 200 students in a course. For a problem of this scale, it is not feasible to create separate variables for storing the marks and finding the average in the above-mentioned manner. To solve such problems, a method is required that helps in storing and accessing data in a generalized and efficient manner. The C language provides this method in the form of a derived datatype known as array.

Example where arrays are used

- to store list of Employee or Student names,
- to store marks of students,
- or to store list of numbers or characters etc.

Classification of Array

In general, arrays are classified as:

- One dimensional Array
- Multidimensional Array

One dimensional Array

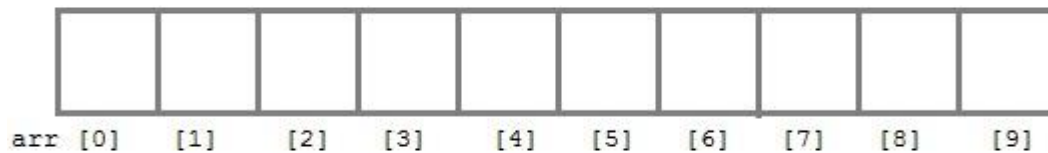
Declaring an Array

Like any other variable, arrays must be declared before they are used. General form of array declaration is,

data-type variable-name[size];

/* Example of array declaration */

int arr[10];



Here **int** is the data type, **arr** is the name of the array and **10** is the size of array. It means array **arr** can only contain **10** elements of **int** type.

Index of an array starts from 0 to size-1 i.e first element of **arr** array will be stored at **arr[0]** address and the last element will occupy **arr[9]**.

/*declare an array of user specified size */

int n = 10;

int arr[n];

int arr[]; /*Incorrect because always specify the number of subscripts when you declare an array*/

Initialization of an Array

After an array is declared it must be initialized. Otherwise, it will contain garbage value (any random value). An array can be initialized at either compile time or at runtime.

Compile time Array initialization

Compile time initialization of array elements is same as ordinary variable initialization. The general form of initialization of array is,

data-type array-name[size] = { list of values };

Array initialize elements by specifying size

- `int mark[5]={ 19, 10, 8, 17,9 }; // integer array initialization`

mark[0] mark[1] mark[2] mark[3] mark[4]

19	10	8	17	9
----	----	---	----	---

Here,

mark[0] is equal to 19
mark[1] is equal to 10
mark[2] is equal to 8
mark[3] is equal to 17
mark[4] is equal to 9

- `float area[5] = { 23.4, 6.8, 5.5,1.7,8.3 }; // float array initialization`
- `char name[6] = {'A','N','S','I','C','\0'}; // character array initialization`
- `int arr[6] = { 10, 20, 30, 40 } ;`



- Compiler creates an array of size 6, initializes first 4 elements as specified by user and rest two elements as 0.
- Above is same as `"int arr[] = {10, 20, 30, 40, 0, 0};"`

- `int marks[4]={ 67, 87, 56, 77, 59 }; // Compile time error`



One important thing to remember is that when you will give more initializer (array elements) than the declared array size than the compiler will give an error.

- `int marks[5]={0};`



- Initializes all array elements as 0.
- Above is same as `"int arr[5] = {0, 0, 0, 0, 0};"`

Array initialize elements by without specifying size

- `int arr[] = { 10, 20, 30, 40 } ;`



- If the array is initialized where it is declared, mentioning the dimension of the array is optional. Compiler creates an array of size 4.
- Above is same as `"int arr[4] = {10, 20, 30, 40};"`

Example 1: Program for Compile time array initialization

```
#include<stdio.h>
int main()
{
    int i;
    int arr[3] = {2, 3, 4};    // Compile time array initialization
    for(i = 0 ; i < 3 ; i++)
    {
        printf("%d\t",arr[i]);
    }
    return 0;
}
```

OUTPUT

2 3 4

Runtime Array initialization

An array can also be initialized at runtime using **scanf()** function. This approach is usually used for initializing large arrays, or to initialize arrays with user specified values. Example,

Example 2: Program for Run time array initialization

```
#include<stdio.h>
int main()
{
    int arr[4];    //array of integer type declaration
    int i, j;
    printf("Enter array element");
    for(i = 0; i < 4; i++)
    {
        scanf("%d", &arr[i]);    // Run time array initialization
    }
    for(j = 0; j < 4; j++)
    {
        printf("%d\n", arr[j]);
    }
    return 0;
}
```

OUTPUT

Enter array elements

5 3 1 7

5

3

1

7

Example 3: Program to take 5 marks of the student from the user and store them in an array, find the average of the marks and Print the marks stored in the array

```
#include <stdio.h>
int main()
{
    int marks[10], i, n, sum = 0.0;
    float average;
    printf("Enter number of elements: "); // Asking user to enter size of the array
    scanf("%d", &n);                      //reading array size from user
    for(i=0; i<n; i++)
    {
        printf("Enter marks %d: ", i+1); // Asking user to enter 5 marks of the student
        scanf("%d", &marks[i]);          //reading array elements
        sum = sum + marks[i];             // adding integers entered by the user to the sum variable
    }
    // Print the marks stored in the array
    for(i=0; i<n; i++)
    {
        printf(" marks %d=%d ", i+1, marks[i]);
    }
    average = sum/n;
    printf("\nAverage marks = %f", average); // Print the average marks
    return 0;
}
```

OUTPUT:

```
Enter number of elements: 5
Enter marks 1: 75
Enter marks 2: 86
Enter marks 3: 93
Enter marks 4: 78
Enter marks 5: 82
marks 1=75 marks 2=86 marks 3=93 marks 4=78 marks 5=82
Average marks = 82.000000
```

Memory representation of One-dimensional Array

```
int val[7]={11, 22, 33, 44, 55, 66, 77};
```

val[0]	val[1]	val[2]	val[3]	val[4]	val[5]	val[6]
11	22	33	44	55	66	77
88820	88824	88828	88832	88836	88840	88844

BeginnersBook.com

All the array elements occupy contiguous space in memory. There is a difference of 4 among the addresses of subsequent neighbours, this is because this array is of integer types and an integer holds 4 bytes of memory.

Memory representation of array

Manipulation of Arrays

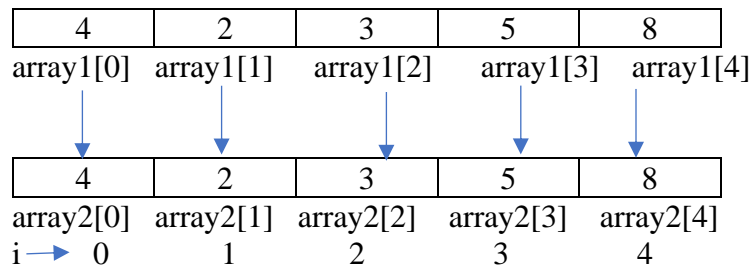
In C it is not possible to perform the arithmetic operations of addition, subtraction, multiplication and division on arrays by array name. These operations must be performed element by element using nested for loops for indexing the elements of the array.

Example:

```
int array1[5]={4,2,3,5,8};
int array2[5];
array2=array1; //Incorrect
```

it can be written as,

```
for(i=0;i<5;i++)
{
    array2[i]=array1[i];
}
```



Example 4: Accept n elements of an array and print them in reverse order

```
#include <stdio.h>
int main()
{
    int a[10], i, n;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter the numbers: ");
    for(i=0; i<n; i++)
    {
        scanf("%d", &a[i]); // Read the numbers
    }
    printf("Elements in reverse order: ");
    for(i=n-1; i>=0; i--)
    {
        printf("%d ", a[i]); // Print the numbers in reverse order
    }
    return 0;
}
```

OUTPUT

```
Enter number of elements: 5
Enter the numbers: 1 2 3 4 5
Elements in reverse order: 5 4 3 2 1
```

Example 5: Accept n elements of an array and copy the elements onto another array and print the resultant array.

```
#include <stdio.h>
int main()
{
    int array1[10],array2[10], i, n;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter the numbers for array1: ");
    for(i=0; i<n; i++)
    {
        scanf("%d", &array1[i]); // Read the numbers of array1
    }
    for(i=0; i<n; i++)
    {
        array2[i]=array1[i]; // copy elements of array1 into array2
    }
    printf("The elements of the array2: ");
    for(i=0; i<n; i++)
    {
        printf("%d ",array2[i]); // print elements of the array2
    }
    return 0;
}
```

OUTPUT

```
Enter number of elements: 4
Enter the numbers for array1: 20 50 40 30
The elements of the array2: 20 50 40 30
```

Example 6: Accept n elements of an array and print the sum of even and odd numbers .

```
#include<stdio.h>
int main()
{
    int arr[50],i,j,n,odd_sum=0,even_sum=0;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter array elements: ");
    for(i = 0; i<n; i++)
    {
        scanf("%d", &arr[i]); // read the array of elements and store it
    }
    for(i=0; i<n; i++)
    {
        if(arr[i]%2==0) //check the even number from the array
            even_sum= even_sum+arr[i]; //Add all those even numbers
        else //check odd number from the array
            odd_sum= odd_sum+arr[i]; //Add all those odd numbers
    }
    printf("Sum of even elements: %d",even_sum);
    printf("\nSum of odd elements: %d",odd_sum);
    return 0;
}
```

OUTPUT

Enter number of elements: 6
Enter array elements: 3 6 7 5 8 12
Sum of even elements: 26
Sum of odd elements: 15

Example 7: Accept n elements of an array and print the maximum and minimum elements of the array.

```
#include <stdio.h>
int main()
{
    int arr[50];
    int i, max, min, n;
    printf("Enter size of the array: ");
    scanf("%d", &n);
    printf("Enter elements in the array: ");
    for(i=0; i<n; i++)
    {
        scanf("%d", &arr[i]);
    }
    //Assume first element as maximum and minimum
    max = arr[0];
    min = arr[0];
    //Finding maximum and minimum elements in an array
    for(i=1; i<n; i++)
    {
        if(arr[i]<min)          //If current element of array is smaller than min
        {
            min = arr[i];
        }
        if(arr[i]>max) // If current element of array is greater than max
        {
            max = arr[i];
        }
    }
    /* Printing maximum and minimum elements of an array */
    printf("Maximum element = %d\n", max);
    printf("Minimum element = %d", min);
    return 0;
}
```

OUTPUT

Enter size of the array: 5
Enter elements in the array: 50 30 70 10 80
Maximum element = 80
Minimum element = 10

Example 8: Accept n elements of an array, reverse the elements on the same array and print the resultant array.

```
#include <stdio.h>
int main()
{
    int array[50], n, i, j, t;
    printf("Enter size of the array: ");
    scanf("%d", &n);
    printf("Enter elements in the array: ");
    for (i= 0; i<n; i++)
    {
        scanf("%d", &array[i]); // Read the array of elements
    }
    //i value is used to traverse from 1st to last and j value is used to traverse from last to 1st

    for (i = 0, j=n-1; i < j; i++, j--)
    {
        t = array[i];          //swap the 1st element to last element then 2nd to last-1.....
        array[i] = array[j];
        array[j] = t;
    }

    printf("Reversed array elements are: ");
    for (i = 0; i < n; i++)
    {
        printf("%d ", array[i]); //print the array after reversing
    }
    return 0;
}
```

OUTPUT

Enter size of the array: 5
Enter elements in the array: 10 20 30 40 50
Reversed array elements are: 50 40 30 20 10

Example 9: C Program to check if an Array is Palindrome or not

```
#include <stdio.h>
int main()
{
    int arr[10], i,j,n,flag=0;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter the numbers: ");
    for(i=0; i<n; i++)
    {
        scanf("%d", &arr[i]);
    }
    //check palindrome

    for(i=0,j=n-1;i<j; i++,j--) //i value is used to traverse from 1st to last and j value is used to
                                traverse from last to 1st
    {
        if(arr[i]!=arr[j])
        {
            flag=1;
            break;
        }
    }
    if(flag==0)
        printf("Palindrome array");
    else
        printf("Not a Palindrome array");
    return 0;
}
```

OUTPUT1:

Enter number of elements: 5
Enter the numbers: 1 2 3 2 1
Palindrome array

OUTPUT2:

Enter number of elements: 5
Enter the numbers: 2 6 7 3 9
Not a Palindrome array

Example 10: Accept n elements in to one array, m elements in to another array and append the elements of the second array to the first array and print the resultant array

```
#include <stdio.h>
int main()
{
    int arr1[50],arr2[50], i,j,n,m;
    printf("Enter size of the 1st array: ");
    scanf("%d", &n);          //Size of array 1
    printf("Enter the numbers of the 1st array: ");
    for(i=0; i<n; i++)
    {
        scanf("%d", &arr1[i]);  //Read the 1st array of elements
    }
    printf("Enter size of the 2nd array: ");
    scanf("%d", &m); //size of array 2
    printf("Enter the numbers of the 2nd array: ");
    for(i=0; i<m; i++)
    {
        scanf("%d", &arr2[i]);  //Read the 2nd array of elements
    }
    for(i=n,j=0;j<m; i++,j++)
    {
        arr1[i]=arr2[j]; //Store the elements of the array 2 into array1
    }
    printf("The resultant array is: ");
    for(i=0;i<n+m; i++) //n+m is the resultant array size
    {
        printf("%d ",arr1[i]); //print the resultant array
    }
    return 0;
}
```

OUTPUT

Enter size of the 1st array: 5
Enter the numbers of the 1st array: 6 7 3 4 5
Enter size of the 2nd array: 6
Enter the numbers of the 2nd array: 9 1 8 2 10 15
The resultant array is: 6 7 3 4 5 9 1 8 2 10 15

Example 11: Insert an element at a required position in to an array

```
#include <stdio.h>
int main()
{
    int arr[50], i,n,val,pos;
    printf("Enter size of the array: ");
    scanf("%d", &n);
    printf("Enter the numbers of the array: ");
    for(i=0; i<n; i++)
    {
        scanf("%d", &arr[i]);
    }
    printf("Enter the new element which you want to insert: ");
    scanf("%d", &val);
    printf("Enter the position where you want to insert: ");
    scanf("%d", &pos);

    for(i=n-1; i>=pos-1; i--)
    {
        arr[i+1]=arr[i]; //shift each element to the right from last to position where new value is to
                        be inserted
    }
    arr[pos-1]=val; //place the value in to the specified position
    printf("The resultant array is: ");
    for(i=0;i<=n; i++)
    {
        printf("%d ",arr[i]); //print the resultant array
    }
    return 0;
}
```

OUTPUT

Enter size of the array: 5
Enter the numbers of the array: 10 30 50 40 80
Enter the new element which you want to insert: 70
Enter the position where you want to insert: 4
The resultant array is: 10 30 50 70 40 80

Example 12: Delete an element at a required position from an array

```
#include <stdio.h>
int main()
{
    int arr[50], i,n,val,pos;
    printf("Enter the size of the array: ");
    scanf("%d", &n);
    printf("Enter the numbers of the array: ");
    for(i=0; i<n; i++)
    {
        scanf("%d", &arr[i]);
    }
    printf("Enter the position where you want to delete: ");
    scanf("%d", &pos);
    for(i=pos-1; i<n; i++)
    {
        arr[i]=arr[i+1]; //shift each element to the left from the position where value is to be
                        deleted to last
    }
    printf("The resultant array is: ");
    for(i=0;i<n-1; i++)
    {
        printf("%d ",arr[i]); //print the resultant array
    }
    return 0;
}
```

OUTPUT

```
Enter the size of the array: 5
Enter the numbers of the array: 10 20 30 40 50
Enter the position where you want to delete: 3
The resultant array is: 10 20 40 50
```

Example 13: Search an element whether it is present or not in the array

```
#include <stdio.h>
int main()
{
    int arr[10], i, val, n, flag=0;
    printf("Enter the size of the array: ");
    scanf("%d", &n);
    printf("Enter the numbers: ");
    for(i=0; i<n; i++)
    {
        scanf("%d", &arr[i]);
    }
    printf("Enter the value want to search: ");
    scanf("%d", &val);
    for(i=0; i<n; i++)
    {
        if(arr[i]==val) //if the array element is match with the search element
        {
            flag=1;
            break;
        }
    }
    if(flag==1)
        printf("search element is present and the position is %d ", i+1);
    else
        printf("search element is not present in the array");
    return 0;
}
```

OUTPUT

```
Enter the size of the array: 5
Enter the numbers: 10 20 30 40 50
Enter the value want to search: 40
search element is present and the position is 4
```

Example 14: Swap Two Array of elements With Using Temp Variable

```
#include <stdio.h>
int main()
{
    int arr1[50],arr2[50], i,j,n,m,t;
    printf("Enter size of the array: ");
    scanf("%d", &n); //size of array 1
    printf("Enter the numbers of the 1st array: ");
    for(i=0; i<n; i++)
    {
        scanf("%d", &arr1[i]); //Read the 1st array of elements
    }
    printf("\nEnter the numbers of the 2nd array: ");
    for(i=0; i<n; i++)
    {
        scanf("%d", &arr2[i]); //Read the 2nd array of elements
    }
    for(i=0,j=0;i<n;i++,j++)
    {
        t= arr1[i];
        arr1[i]=arr2[i]; //swap the elements of array1 and array2
        arr2[i]= t;
    }

    printf("The resultant array1 is: \n");
    for(i=0;i<n; i++)
    {
        printf("%d ",arr1[i]); //print the resultant array1
    }
    printf("\nThe resultant array2 is: ");
    for(i=0;i<n; i++)
    {
        printf("%d ",arr2[i]); //print the resultant array2
    }
    return 0;
}
```

OUTPUT

Enter size of the array: 5
Enter the numbers of the 1st array: 10 20 30 40 50
Enter the numbers of the 2nd array: 11 22 33 44 55

The resultant array1 is: 11 22 33 44 55
The resultant array2 is: 10 20 30 40 50