Array

Definition of Array

Why Array is used

Types of Array

One Dimensional Array

Declaration of 1-D Array

Accessing 1-D Array Elements

Processing 1-D Array

Initialization of 1-D Array

1. D Array and Functions

Passing Individual Array Element to Function

Passing whole 1-D Array to a Function

Two Dimensional Array

Declaration and Accessing Individual Elements of 2-DArray

Processing of 2-D Array

Initialization of 2-D Arrays

Arrays with more than Two Dimensions

Multidimensional Array and Functions

**ARRAY**

An Array is a data structure containing a number of data values all of which are same type.Data structure is a format for organizing and storing data.

Array is data structure which you can visualize as follows:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

imagine an array as a large chunk of memory divided into smaller block of memory and each block is capable of storing a data value of some type.

**Why Array is Used**

The variables that we have used till now are capable of storing only one value at a time. Consider a situation when we want to store and display the age of 100 employees. We can do the following

* Declare 100 different variables to store the age of employees.
* Assign a value to each variable.
* Display the value of each variable.

We can do the above steps, but it would be difficult to handle so many variables in the program and the program would become very lengthy. The concept of arrays is used in these types of situations where we can group similar type of data items.

**TYPES OF ARRAY**

There are Three types of Array. They are

* One Dimensional Array
* Two Dimensional Array
* Multi Dimensional Array

**ONE DIMENSIONAL ARRAY**

**Declaration of 1-D Array**

The simplest form of array one can imagine is One Dimensional array. Like other simple variables, arrays should also be declared before they are used in the program. The syntax for declaration of an array is-

data\_type array\_name[size];

Here array\_name denotes the name of the array and it can be any valid C identifier, data\_type is the data type of the elements of array. The size of the array specifies the number of elements that can be stored in the array. Here are some examples of array declarations

int age[100];

float salary [15];

char grade[20];

Here age is an array of type int, which can store 100 elements of type int. The array salary is a float type array of size 15, can hold values of type float and third one is a character type array of size 20, can hold characters.

**Accessing 1-D Array Elements**

The elements of an array can be accessed by specifying the array name followed by subscript in brackets. In C the array subscript start from 0. Hence if there is an array of size, the valid subscripts is one less than the size of the array. Let us take an array.

int arr[5]; /\*size of array arr is 5, can hold five integer elements\*/

The elements of this array are

arr[0], arr[1], arr[2], arr[3]

Here 0 is the lower bound and 4 is the upper bound of the array.

**Processing 1-D Array**

For processing arrays we generally use a for loop and the loop variable is used at the place of subscript. The initial value of loop variable is taken 0 since array subscripts start from zero. The loop variable is increased by 1 each time so that we can access and process the next element in the array.

* Reading values in arr

for(i=0; i<10; i++)

scanf(“%d”,&arr[i]);

* Displaying values of arr

for(i=0; i<10; i++)

Printf(“%d”,arr[i]);

* Adding all the elements of arr

sum=0;

for(i=0; i<10; i++)

Sum+=arr[i];

**Example 1:**

Program to input values into an array and display them

#include <stdio.h>

int main()

{

    int a[1000],i,n;

     printf("Enter size of array: ");

    scanf("%d",&n);

     printf("Enter %d elements in the array : ", n);

    for(i=0;i<n;i++)

    {

        scanf("%d", &a[i]);

    }

    printf("\nElements in array are: ");

    for(i=0;i<n;i++)

    {

        printf("%d  ", a[i]);

    }

    return 0;

}

**Output:**

Enter size of array: 5

Enter 5 elements in the array: 1

2

3

4

5

Elements in array are: 1  2  3  4  5

**Initialization of 1-D Array**

After declaration, the elements of a local array have garbage value while the elements of global and static arrays are automatically initialized to zero. We can explicitly initialize arrays at the time of declaration. The syntax for initialization of an Array is

data\_type array\_name[size]={value1, value2, ....valueN};

Here array\_name is the name of the array variable, size is the size of array and value1,value2....valueN are the constant value known as initializers, which are assigned to the array elements one after another.

Example:

int a[5]={1, 11, 6, 3, 9};

**1-D Arrays and Functions**

**Passing Individual Array Elements to a Function**

In C programming, you can pass an entire array to functions. Before we learn that, let's see how you can pass individual elements of an array to functions.

We know that an array element is treated as any other simple variable in the program. So like other simple variables, we can pass individual array elements as arguments to a function.

**Example 2:**

Program to pass array elements to a function

#include <stdio.h>

void check(int num);

int main()

{

int arr[10],i;

printf("Enter the Array elements ");

for(i=0;i<10;i++)

{

scanf("%d",&arr[i]);

check(arr[i]);

}

return 0;

}

void check(int num)

{

if(num%2==0)

printf("%d is even\n",num);

else

printf("%d is odd\n",num);

}

Output:

Enter the Array elements

5

5 is odd

6

6 is even

**Passing whole 1-D Array to a Function**

We can pass whole array as an actual argument to a function. The corresponding formal argument should be declared as an array variable of the same data type.