SNS COLLEGE OF TECHNOLOGY

Coimbatore – 35

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

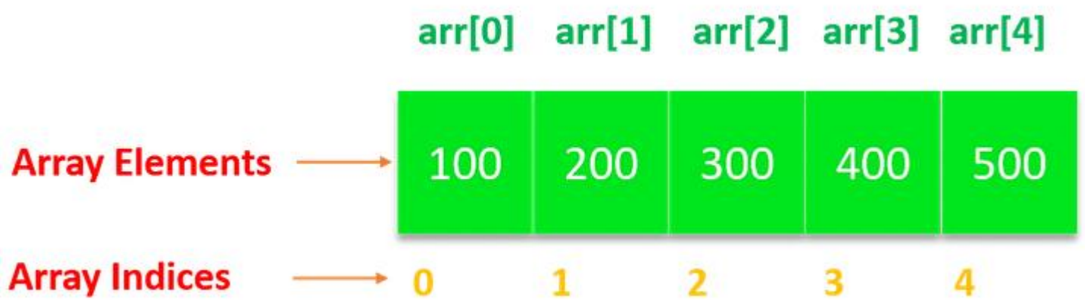
23ITT101 – PROGRAMMING IN C & DS

UNIT III – ARRAYS AND INTRODUCTION TO DATA STRUCTURES

ARRAYS

ARRAYS

- An array is a collection of one or more values of the same data type stored in contiguous memory locations.
- The data type can be user-defined or even any other primitive data-type.
- **Elements of an array can be accessed with the same array name by specifying the index number as the location in memory.**
- The index number of the first element in an array is 0, and the index number of the last element is the size of the array minus 1.
- Arrays are useful when we need to store and manipulate a large amount of data of the same type.



PROPERTIES OF ARRAYS

- Each element of an array is of same data type and carries the same size, i.e., int = 4 bytes.
- Elements of the array are stored at contiguous memory locations where the first element is stored at the smallest memory location.
- Elements of the array can be randomly accessed since we can calculate the address of each element of the array with the given base address and the size of the data element.

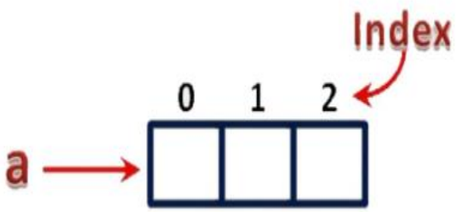
DECLARATION & INITIALIZATION OF ARRAYS

Example Code

```
int a [3] ;
```

Here, the compiler allocates 6 bytes of contiguous memory locations with a single name 'a' and tells the compiler to store three different integer values (each in 2 bytes of memory) into that 6 bytes of memory. For the above declaration, the memory is organized as follows...

In the above memory allocation, all the three memory locations have a common name 'a'. So accessing individual memory location is not possible directly. Hence compiler not only allocates the memory but also assigns a numerical reference value to every individual memory location of an array. This reference number is called "Index" or "subscript" or "indices". Index values for the above example are as follows...

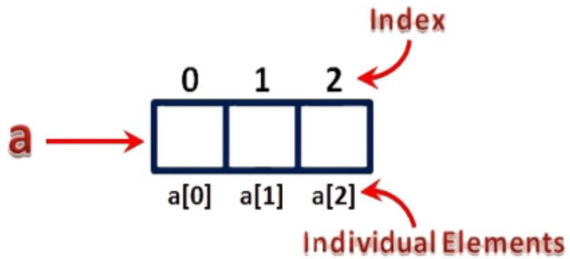


Accessing Individual Elements of an Array

The individual elements of an array are identified using the combination of 'arrayName' and 'indexValue'. We use the following general syntax to access individual elements of an array...

```
arrayName [ indexValue ] ;
```

For the above example the individual elements can be denoted as follows...

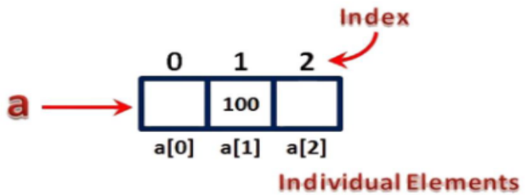


For example, if we want to assign a value to the second memory location of above array 'a', we use the following statement...

Example Code

```
a [1] = 100 ;
```

The result of the above assignment statement is as follows...



DECLARATION OF ARRAYS

There are various ways in which an array can be declared and initialized in various ways. You can declare an array of any data type (i.e. int, float, double, char) in C. The following ways can be used to declare and initialize an array in C.

Array Declaration by Specifying the Size

Arrays can be declared by specifying the size or the number of array elements. The size of the array specifies the maximum number of elements that the array can hold. In the latest version of C, you can either declare an array by simply specifying the size at the time of the declaration or you can provide a user-specified size. The following syntax can be used to declare an array simply by specifying its size.

```
// declare an array by specifying size in [].

int my_array1[20];

char my_array2[5];

// declare an array by specifying user defined size.

int size = 20;

int my_array3[size];
```

DECLARATION OF ARRAYS

Array Declaration by Initializing Elements

An array can be initialized at the time of its declaration. In this method of array declaration, the compiler will allocate an array of size equal to the number of the array elements. The following syntax can be used to declare and initialize an array at the same time.

// initialize an array at the time of declaration.

int my_array[] = {100, 200, 300, 400, 500}

In the above syntax, an array of 5 elements is created and even though the array size has not been specified here, the compiler will allocate a size of 5 integer elements.

DECLARATION OF ARRAYS

Array Declaration by Specifying the Size and Initializing Elements

An array can also be created by specifying the size and assigning array elements at the time of declaration. This method of array creation is different from the previous one. Here, if the number of initialized elements is less than the size of the array specified, then the rest of the elements will automatically be initialized to 0 by the compiler.

// declare an array by specifying size and initializing at the time of declaration

int my_array1[5] = {100, 200, 300, 400, 500}; // my_array1 = {100, 200, 300, 400, 500}

int my_array2[5] = {100, 200, 300}; // my_array2 = {100, 200, 300, 0, 0}

In the above array syntax, my_array1 is an array of size 5 with all five elements initialized. Whereas, my_array2 is an array of size 5 with only three of its elements initialized. The remaining two elements of the second array will be initialized to 0 by the compiler.

DECLARATION OF ARRAYS

Array Initialization Using a Loop

An array can also be initialized using a loop. The loop iterates from 0 to (size - 1) for accessing all indices of the array starting from 0. The following syntax uses a “for loop” to initialize the array elements. This is the most common way to initialize an array in C.

```
int my_array[5];
int i;
for(i = 0; i < 5; i++)
{
   my_array[i] = 2 * i;
}                          // my_array = {0, 2, 4, 6, 8}
```

In the above syntax, an array of size 5 is declared first. The array is then initialized using a for loop that iterates over the array starting from index 0 to (size - 1).

ACCESSING ARRAY ELEMENTS

Since an array is stored contiguously in the memory, it has indices starting from ”0” to “array_size - 1”, also known as zero-based indexing. This indexing represents the position in the array.

The array indices are used to access any element of the array in the following way:

array_name[index]

The index of the element to be accessed is specified within square brackets “[ ]”. The range of the index is-integers in the range [0, size).

Example:

```
int my_array[6];

my_array[0] = 100;        // access 1st element

my_array[2] = 300;        // access 4th element

my_array[5] = 600;        // access last element
```

INPUT AND OUTPUT ARRAY ELEMENTS

Array values can be stored by taking input from the user and storing them in the array. The following example illustrates this:

scanf("%d", &my_array[0]); // input an integer element and store it in 1st position of the array

scanf("%f", &my_array[i-1]); // input a float element and store it in ith position of the array

Similarly, array elements can also be displayed in the output using the printf() method. The index is specified indicating the position of the element to be printed. The following example illustrates this:

printf("%d", my_array[0]); // print the element stored at 1st position or 0th index

printf("%d", my_array[i-1]); // print the element stored at ith position or (i - 1)th index

EXAMPLE

```c
// Program to take 5 values from the user and store them in an array
// Print the elements stored in the array
#include <stdio.h>

int main() {
  int values[5];

  printf("Enter 5 integers: ");

  // taking input and storing it in an array
  for(int i = 0; i < 5; ++i) {
     scanf("%d", &values[i]);
  }

  printf("Displaying integers: ");

  // printing elements of an array
  for(int i = 0; i < 5; ++i) {
     printf("%d\n", values[i]);
  }
  return 0;
}
```

Output

```
Enter 5 integers: 1
-3
34
0
3
Displaying integers: 1
-3
34
0
3
```

CHARACTERISTICS OF ARRAYS

- ✓ An array holds elements that have the same data type.
- ✓ Array elements are stored in subsequent memory locations.
- ✓ Two dimensional array elements are stored row by row in subsequent memory locations.
- ✓ Array name represents the address of the starting element.
- ✓ Array size should be mentioned in the declaration.
- ✓ Array size must be constant expression and not a variable.

ADVANTAGES OF ARRAY

Arrays have a great significance in the C language. They provide several advantages to the programmers while programming. Some of them are:

- ✓ Arrays make the code more optimized and clean since we can store multiple elements in a single array at once, so we do not have to write or initialize them multiple times.
- ✓ Every element can be traversed in an array using a single loop.
- ✓ Arrays make sorting much easier. Elements can be sorted by writing a few lines of code.
- ✓ Any array element can be accessed in any order either from the front or rear in O(1) time.
- ✓ Insertion or deletion of the elements can be done in linear complexity in an array.

DISADVANTAGES OF ARRAY

Every advantageous thing comes with some disadvantages as well. This stands true for arrays as well. Below are some of the disadvantages of the array in C:

Accessing an array out of bounds: The first disadvantage of arrays is that they are statically allocated. This means that once their size is initialized, it can not be increased or decreased.

Homogeneity: We can store only a single type of element in the array i.e., arrays are homogeneous. We can not use it as a template. For example, if the array data type is char, then only characters can be stored in the array. If you try to store integers or any other element of a different data type, it will throw an error.