Unit-4

Arrays: Introduction – Declaration of Arrays – Accessing elements of the Array – Storing Values in Array– Operations on Arrays – one dimensional, two dimensional and multi dimensional arrays, character handling and strings.

2.1 Introduction:

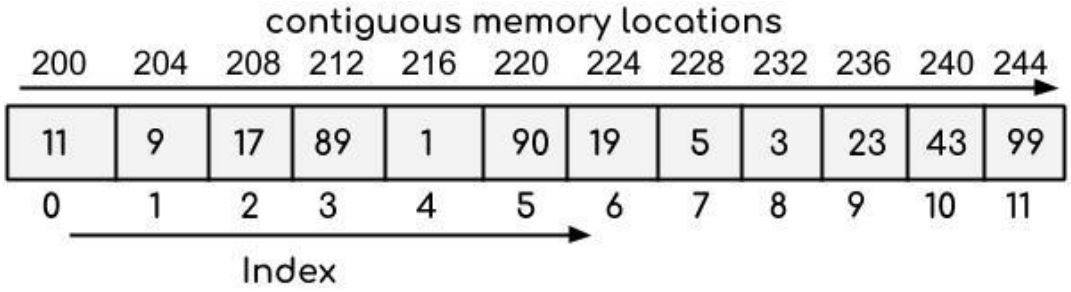
- An array is a collection of variables of the same type that are referenced by a common name.
- In C, all arrays consist of contiguous memory locations.
- The lowest address corresponds to the first element and the highest address to the last element.
- Arrays may have from one to several dimensions. A specific element in an array is accessed by an index.

2.2 Declaration of Arrays

- The general form of single-dimension array declaration is:
  - Type variable name[size];
- Here, type declares the base type of the array, size defines how many elements the array will hold.
- For example, the following declares as integer array named sample that is ten elements long:
  - int a[10];
- Declares an array, named a, consisting of ten elements, each of type int. Simply speaking, an array is a variable that can hold more than one value.
  - Ex: int x[100]; float mark[50]; char name[30];

Memory representation:

The following diagram represents an integer array that has 12 elements. **The index of the array starts with 0**, so the array having 12 elements has indexes from 0 to 11.



2.3 Accessing elements of the Array

- We can access any array element using array name and subscript/index written inside pair of square brackets [].
- *For Example:*
  - Suppose we have an integer array of length 5 whose name is marks. int marks[5] = {5,2,9,1,1}; Now we can access elements of array marks using subscript followed by array name.

- marks[0] = First element of array marks = 5
- marks[1] = Second element of array marks = 2
- marks[2] = Third element of array marks = 9
- marks[3] = Fourth element of array marks = 1
- marks[4] = Last element of array marks = 1
- *Remember array indexing starts from 0. Nth element in array is at index N-1.*

2.4 Storing Values in Array

- you can also initialize the array during declaration like this:
- int arr[5] = {1, 2, 3, 4 ,5};OR (both are same)
- int arr[] = {1, 2, 3, 4, 5};Un-initialized array always contain garbage values.
- Using scanf:
  - For(i=0;i<n;i++)
    - Scanf("%d",&a[i]);

2.5 Operations on Arrays

- Change Value of Array elements
  - int mark[5] = {19, 10, 8, 17, 9} // make the value of the third element to -1
  - mark[2] = -1; // make the value of the fifth element to 0
  - mark[4] = 0;
- Arithematic Operation on array elements.
- Input and Output Array Elements
  - Here's how you can take input from the user and store it in an array element.
  - // take input and store it in the 3rd element
  - scanf("%d", &mark[2]);
- // take input and store it in the ith element scanf("%d", &mark[i-1]);
- Here's how you can print an individual element of an array.
- // print the first element of the array
- printf("%d", mark[0]);
- // print the third element of the array
- printf("%d", mark[2]);
- // print ith element of the array
- printf("%d", mark[i-1]);

Example:

Example 1: Array Input/Output

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

```
}
```

Output:

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

Things you can and can't do

- You can not
  - use = to assign one array variable to another
  - Int a[10];
  - printf("%d",a);
  - use == to directly compare array variables
  - directly scanf or printf arrays
- But you can do these things on array elements.
- You can write functions to do them.

2.6 One dimensional Array

- One dimensional array is an array that has only one subscript specification that is needed to specify a particular element of an array.
- A one-dimensional array is a structured collection of components (often called array elements) that can be accessed individually by specifying the position of a component with a single index value.
- **Syntax:**
- **data-type arr_name[array_size];**
- Example: int a[5]

Initialization

- 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 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 };
/* Here are a few examples */
int marks[4]={ 67, 87, 56, 77 };    // integer array initialization
float area[5]={ 23.4, 6.8, 5.5 };   // float array initialization
int marks[4]={ 67, 87, 56, 77, 59 };    // Compile time error
```

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

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

```
Output:
         2  3  4
```

- **Runtime 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,

```
#include<stdio.h>
void main()
{
   int arr[4];
   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]);
   }
}
```

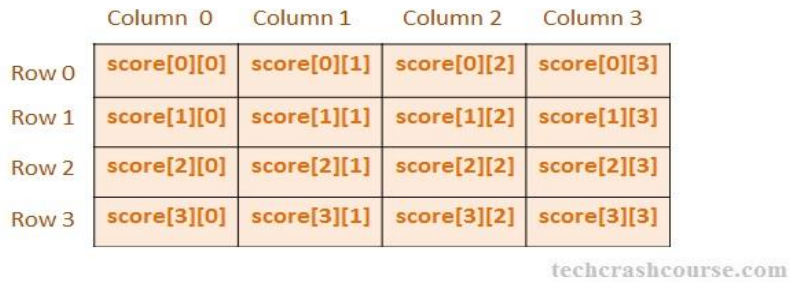
- **Rules for Declaring One Dimensional Array**
  - An array variable must be declared before being used in a program.
  - The declaration must have a data type(int, float, char, double, etc.), variable name, and subscript.
  - The subscript represents the size of the array. If the size is declared as 10, programmers can store 10 elements.
  - An array index always starts from 0. For example, if an array variable is declared as s[10], then it ranges from 0 to 9.
  - Each array element stored in a separate memory location.
- **Points to Remember About Array in C:**
  - An array is a derived data type in C that is constructed from the fundamental data type of C language.
  - An array is a collection of similar types of values in a single variable.
  - In implementation when we required the 'n' number of the variable of a similar data type then we need to go for the array.
  - When we are working with an array always memory will be created in the contiguous memory location, so randomly we can access the data.
  - In an array, all elements will share the same name with a unique identification value called index.

- Always array index will start from 0 and ends with size – 1.
- When we are working with an array compile-time memory management will occur i.e. static memory allocation.
- Always size of the array must be an unsigned integer value which should be greater than zero.

Example Programs:

2.7 Two dimensional and multi dimensional arrays

- Two dimensional arrays are most common type of multi dimensional array. A **two dimensional array** in C language is represented in the form 2D matrix having rows and columns.
  - A two dimensional matrix is an array of one dimensional arrays.
  - Two dimensional array requires two subscript variables or indexes.
  - One subscript represents the row index while other represent column index of an element in matrix.
  - Elements of a two dimensional array are stored in contiguous memory location. First all elements of first row are store then all elements of second row and so on.



- Any element in **two dimensional matrix** is uniquely identified by array_name[row_Index][column_Index], where row_index and column_index are the subscripts that uniquely specifies the position of an element in a **2D matrix**.
- An element in second row and third column of an two dimensional matrix whose name is board is accessed by board[1][2].
- Like single dimensional array, indexing starts from 0 instead of 1.
- We can declare a two dimensional matrix of R rows and C columns as follows:
- data_type array_name[R][C];
- ***For Example:***

A two dimensional integer matrix of 3 rows and 3 columns can be declared as

```
int score[3][3];
```

```
// Declare a two-dimensional array with 3 rows and 2 columns
                int table[3][2]:
// create and initialize  an array
            int table[3][2] = { {10, 22}, {33, 44}, {45, 78} };
                    or
            int table[3][2] = {10, 22, 33, 44, 45, 78 };
                    or
            int table[3][2] = {
                                {10, 22},
                                {33, 44 },
                                {45, 78 }
                              };
```

See the following diagram which represents the above array :

	0	1
0	10	22
1	33	44
2	45	78

In the above array :

table[1][1] contains 22
table[2][0] contains 45
table[2][2] is "array out of bounds"

Given: char ch[4][3]

Ch [0][0]	Ch [0][1]	Ch [0][2]
Ch [1][0]	Ch [1][1]	Ch [1][2]
Ch [2][0]	Ch [2][1]	Ch [2][2]
Ch [3][0]	Ch [3][1]	Ch [3][2]

Example Programs:

Multi dimensional arrays

- C programming language supports multi dimensional Arrays.
- Multi dimensional arrays have more than one subscript variables.
- Multi dimensional array is also called as matrix.
- Multi dimensional arrays are array of arrays.
- Declaration of Multi Dimensional Array
- ***data_type array_name[size1][size2]...[sizeN];***
- Above statement will declare an array of N dimensions of name array_name, where each element of array is of type data_type. The maximum number of elements that can be stored in a multi dimensional array array_name is size1 X size2 X size3...sixeN.
- *For Example:*

  Declaration of two dimensional integer array

  int board[8][8];

  Declaration of three dimensional character array

  char cube[50][60][30];

Advantages of an Array in C:

- Random access of elements using array index.
- Use of less line of code as it creates a single array of multiple elements.
- Easy access to all the elements.
- Traversal through the array becomes easy using a single loop.
- Sorting becomes easy as it can be accomplished by writing less line of code.

Disadvantages of an Array in C:

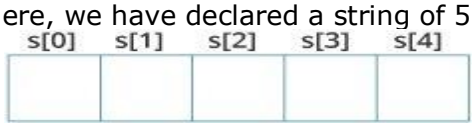
- Allows a fixed number of elements to be entered which is decided at the time of declaration. Unlike a linked list, an array in C is not dynamic.
- Insertion and deletion of elements can be costly since the elements are needed to be managed in accordance with the new memory allocation.

2.8 Character handling

- Characters are the fundamental building blocks of every program.
- A character constant is represented as a character in a single quote. The value of character constant is an integer.
  - For example, a is represented as 'a' which is actually the integer value equal to 97 in ASCII.
  - Similarly, '\n' represents the integer value of newline equal to 10 in ASCII.
- C provides standard character handling library <ctype.h>.
- We should include character handling library i.e. <ctype.h> for using different character handling function.
- This header include various useful functions for performing tests and manipulations of character data in C programming.
- The function in <ctype.h> recieves an unsigned char represented as int or EOF as argument.
- In C programming, character is a one byte integer.
- C programming character library functions with description
- int islower(int ch);
  - It returns a *true* if ch is a *lowercase* letter and *0* otherwise.
- int isupper(int ch);
  - It returns a true value if ch is a *uppercase* and 0 therwise.
- int tolower(int ch);
  - This function converts *uppercase* letter into *lowercase* letter.
- int toupper(int ch);
  - This function converts *lowercase letter* into *uppercase letter*.
- int isdigit(int ch);
  - This function checks whether ch is a *digit* or not.
- int isalpha int ch);
  - This function checks whether ch is *a letter* or not.
- int isalnum(int ch);
  - This function returns a 1 if ch is *a digit* or *a letter* and 0 otherwise.
- int isxdigit(int ch);
  - This function checks if ch is a *hexadecimal digit* or not.
- int isspace(int ch);
  - This function checks if ch is a *whitespace character* or not.
- int isgraph(int ch);
  - This function checks if ch is a *printing character* other than a space or not.
- int iscntrl(int ch);
  - This function checks if ch is a *control character* or not.
- int isprint(int ch);
  - This function checks if ch is a *printing character* including a *space* or not.
- int ispunct(int ch);
  - This function checks if ch is a *printing character* other than *a digit, a space, a letter* or not.

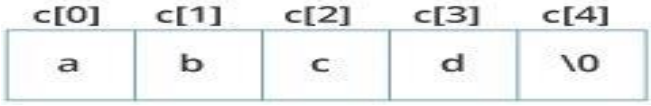
2.9 Strings

- In C programming, a string is a sequence of characters terminated with a null character '\0'.
- How to declare a string?
- Here's how you can declare strings:
  - char s[5]; //Here, we have declared a string of 5 characters.



- How to initialize strings?
- You can initialize strings in a number of ways.
- char c[] = "abcd";
- char c[50] = "abcd";
- char c[] = {'a', 'b', 'c', 'd', '\0'};

char c[5] = {'a', 'b', 'c', 'd', '\0';



Actually, you do not place the *null* character at the end of a string constant.
The C compiler automatically places the '\0' at the end of the string when it initializes the array.
Let us try to print the above mentioned string –

```
#include <stdio.h>
void main ()
{
    char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
    printf("Greeting message: %s\n", greeting );
    getch();
}
```

C supports a wide range of functions that manipulate null-terminated strings –

Sr.No.	Function & Purpose
1	strcpy(s1, s2); Copies string s2 into string s1.
2	strcat(s1, s2); Concatenates string s2 onto the end of string s1.
3	strlen(s1); Returns the length of string s1.
4	strcmp(s1, s2); Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
5	strchr(s1, ch); Returns a pointer to the first occurrence of character ch in string s1.
6	strstr(s1, s2); Returns a pointer to the first occurrence of string s2 in string s1.

The following example uses some of the above-mentioned functions −

```
#include <stdio.h>
 #include <string.h>
void main ()
{
    char str1[12] = "Hello";
    char str2[12] = "World";
    char str3[12]; int len ; /* copy str1 into str3 */
    strcpy(str3, str1);
    printf("strcpy( str3, str1) : %s\n", str3 );
        /* concatenates str1 and str2 */
    strcat( str1, str2);
    printf("strcat( str1, str2): %s\n", str1 );
        /* total lenghth of str1 after concatenation */
    len = strlen(str1);
    printf("strlen(str1) : %d\n", len );
}
```

Question and answers

1. **Define array?**
2. **Discuss how to declare an array with example.**
3. **Explain about array initialization.**
4. **Explain how to access elements of the Array.**
5. **Explain about various types of arrays with example.**
6. **List out some character handling functions.**
7. **Define String and list out some strings functions.**