Unit 6. Arrays

6.1 Introduction to Array,

6.2 Types of Array (Single Dimensional and Multidimensional),

6.3 Declaration and Memory Representation of Array,

6.4 Initialization of array,

6.5 Character Array and Strings,

6.6 Reading and Writing Strings,

6.7 Null Character,

6.8 String Library Functions(string length, string copy, string concatenation, string compare)

Introduction

- An array is collection of similar types of data under the same name.
- A single variable of any data type can hold only one value of that type at a time but in array we can store any number of data of defined type in to a single variable name.
- Each array elements are accessed by using the name of array with array index e. g. a[0] represents first element in array a.
- When an array is declared, the index of element in that array starts from 0 and maximum index is the maxsize-1. e.g. if an array of size 10 is declared then indices ranges from 0 to 9

Declaration of Array

- An array in C can be declared as below

  <data type><space><arrayname[SIZE]> ;

  e.g. ***int a[10];*** declares an array a of size 10.

  – The data type is any valid C data type

  – Array name is valid identifier

  – SIZE is the integer constant either literal or symbolic.

- The memory allocation of array will be the contiguous block of memory for the defined numbers of items (SIZE) of the perticular data type. For example in above array a, memory is allocated for 10 integers in contiguous memory blocks with increasing indices from 0 to 9
- char ch[20]; declares an array of 20 characters starting from ch[0] ….. ch[19]. So memory is allocated to store the 20 characters in contiguous memory blocks.

Declaration of Array

- The size of array can be given as literal integer as well as symbolic constant as below:
  - **Using Literal integer constant**

```
int   arr[100];          /*size is 100 */
float num[50];       /*size is 50 */
```

  - **Using Symbolic Constant integer**

```
/* defining Symbolic constant */
        #define SIZE 100
/* Declaring array of size SIZE */
        int   arr[SIZE];

/*defining constant MAX */
   const int MAX =50;

/* Declaring array of size SIZE */
  float num[MAX];
```

Types of array

- Based on the size of the array, it can be
  - Single Dimensional
  - Multi-Dimensional
- In single dimensional array only one size is declared which represents the no of elements allowed to that array to store.
- In two dimensional array two sizes **ROWSIZE** and **COLUMNSIZE** are used to declare the array.
- The declaration of array in 2D is

  data_type arrayname[RSIZE][COLSIZE];

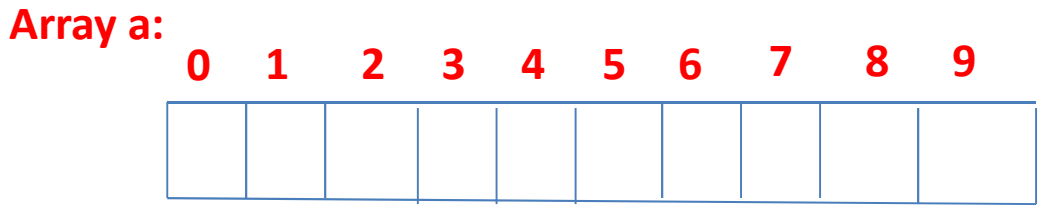
  - **e.g., int a[3][4]; declares a 2D array with 3 rows and 4 columns (4 items at each row).**

1D array

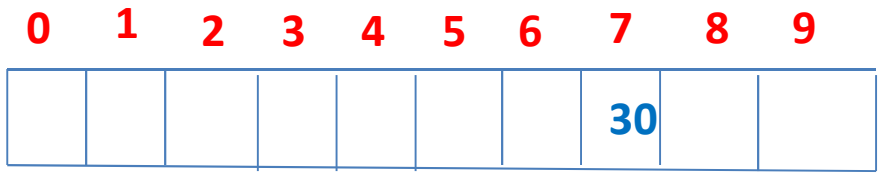
– **e.g.**

`int a[10];` **declares a 1D array with 10 integers.**

– **The memory allocation for this array will be like this(contiguous block of memory for each items)**



- The array element can be accessed by using the array name with index value as a[6] – represents element at 7th position.
- e.g. a[6]=30 assigns the value 30 to 7th element in array a.



1D array

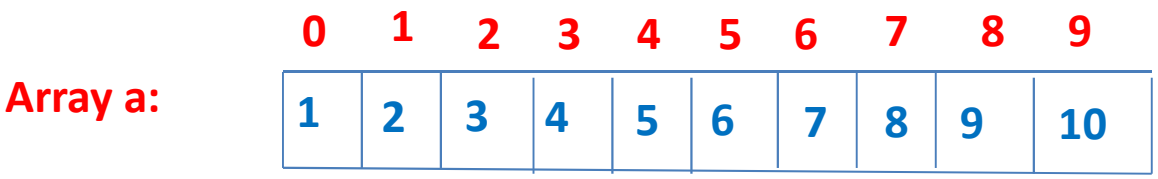
- To store the all element value in memory declared by array, we can use the loop e.g.

  `int a[10];` declares a 1D array with 10 integers.

- Assigning value to each element of array using loop

```
for(i=0;i<10;i++) /*i to be declared */
   a[i]=i+1;
```

- The above C-code assigns values 1 to 10 to element a[0] to a[9] as below



- To input values from keyboard, we can do as:

```
for(i=0;i<10;i++) /*i to be declared */
   scanf("%d",&a[i]);
```

Example 1D array: Program to input 10 integers in array and display

```
#include<stdio.h>
main()
{
        int a[10];
        printf("\nEnter 10 integers:");
        for(i=0;i<10;i++)
        {
                scanf("%d",&a[i]);
        }
        printf("The array is:\n");
        for(i=0;i<10;i++)
                printf("%d  ",a[i]);
}
```

2D array

– **e.g.**

– **int a[3][4]; declares a 2D array with 3 rows and 4 columns (4 items at each row).**

Columns

rows	0	1	2	3
0	00	01	02	03
1	10	11	12	13
2	20	21	22	23

- The array element can be accessed by using the array name with row/col indices as a[2][3] – represents element at 3rd row and 4th column.

Characteristics of Array

- An array holds elements that have the same data type.
- Array elements are stored in subsequent memory locations.
- Array element can be accessed using array name with index where index range is 0 to maxsize-1;
- 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 a constant expression and not a variable.
-

Initialization of Array -1D

- An array can be initialized as :
  - **DataType array_name[]={**value1,value2,….,valueN**};**
- In the above example, the size of array is automatically taken as the number of values used in initialization. E.g.
  - **int arr[]={10,20,30,40,50,60,70,80,90,100} ;**
  - This example declares an array named arr with size 10 since there are 10 values in the array initialization.
  - **char crr[]={ 'a' , 'p', 'p', 'l', 'e'};** declares an array of characters , the size becomes 5
- After initialization, array element can be accessed similarly by using array name and corresponding index

Initialization of Array – 2D

- A 2D array can be initialized as :
  - **DataType array_name[][COLSIZE]={value1,value2,….,valueN};**
- In the above example, the size of array is automatically taken as the number of values used in initialization. E.g.
  - **int arr[][3]={10,20,30,40,50,60,70,80,90} ;**
  - **Or we can write as:**
  - **int a[][3]={{1,2,3},{4,5,6},{7,8,9}};**
  - This example declares an array named arr with col_size=3 with 3 rows since there are 9 element in the initialization list.
  - **The column size in the array declaration and initialization is must since it can not automatically identify the no of element in one row.**

- After initialization, array element can be accessed similarly by using array name and corresponding index

Array initialization example-1D

```
#include<stdio.h>
main()
{
        int a[][3]={1,2,3,4,5,6,7,8,9};
        int i,j;
        char str[]={'a','p','p','l','e','\0'};
        for(i=0;i<3;i++)
        {
                for(j=0;j<3;j++)
                        printf("\na[%d][%d]=%d",i,j,a[i][j]);
        }

        printf("\n The string is : %s",str);

}
```

2D Array initialization example

```
#include<stdio.h>
main()
{
        int a[][3]={1,2,3,4,5,6,7,8,9};
        int i,j;
        for(i=0;i<3;i++)
        {
                for(j=0;j<3;j++)

        printf("\na[%d][%d]=%d",i,j,a[i][j]);
        }
}
```

2D array Example

```
#include <stdio.h>
int main()
 {
        int i,j,k;
        int A[3] [3];
        printf("Enter integer elements of A[3][3] array:\n");

 /*Input the data element for 2D array  3x3, total 9 data.*/
        for(i=0; i<3; i++)
            for (j=0; j<3; j++)
                scanf("%d",&A[i][j]);
        printf("\nThe Array is:\n");
        for( i=0;i<3;i++)
        {
                for( j=0;j<3;j++)
                {
                        printf ("%d ",A[i][j]);
                }
        printf ("\n");
        }
        return 0;
}
```

Output of above program

Enter integer elements of A[3][3] array:

1 2 3 4 5 6 7 8 9

The Array is:

1 2 3

4 5 6

7 8 9

3D array Example

```
#include <stdio.h>
int main()  {
 int i,j,k;
 int A[2] [3] [3];
 printf("Enter integer elements of A[2] [3][3] array:\n");

 /*Input the data element for 3D array  2x3x3, total 18 data.*/
 for(i=0; i<2; i++)
    for (j=0; j<3; j++)
        for (k = 0; k<3; k++)
             scanf ("%d", &A[i] [j] [k]);

 for( i=0;i<2;i++)
{
     printf ("\n");
     for( j=0;j<3;j++)
     {
          printf ("\n");
          for(k = 0; k<3; k++)
               printf ("%d ", A[i][j][k] );
    }
  }
  printf ("\n");
  return 0;
}
```

Output of above program

Enter integer elements of A[2] [3][3] array:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

1 2 3
4 5 6
7 8 9

10 11 12
13 14 15
16 17 18

Character Array - Strings

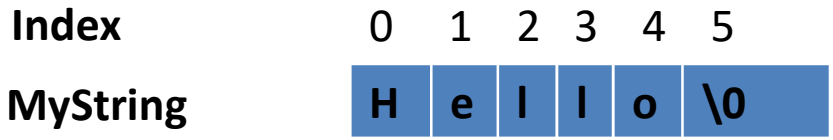
- Strings are sequence of characters i.e. one-dimensional array of characters terminated by a **null** character '\0'.
- Thus a null-terminated string contains the characters that comprise the string followed by a **null**.
- The following declaration and initialization create a string consisting of the word “Hello".

  char MyStr[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
- This can be done alternatively as :

  char Mystr[] = “Hello”;
- In later case the null character is automatically added for string termination
- To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello.“- see above example.

Sotrage allocation for string

- For the string declared above as: char Mystr[]="Hello"; the storage allocation for the Mystr variable is like below.




```
#include<stdio.h>
main()
{
        char MyStr[]="Hello";
        int I;
        printf("The string is %s", MyStr);
        for(i=0;i<=6;i++)
            printf("\na[%d]=%c",MyStr[i]);
}
```

Output:
The string is: Hello
a[0]=H
a[1]=e
a[2]=l
a[3]=l
a[4]=o
a[5]=

Example: length of string, copy string

```
#include<stdio.h>
main()
{
        char str[100],cstr[100],rstr[100];
        int i,j,len;
        printf("input a string:");
        scanf("%s",&str);
        len=0;
        for(i=0;str[i]!='\0';i++)
                len++;
        printf("Length of Str= %d",len);
        for(i=0;i<=len;i++)
                cstr[i]=str[i];
        printf("\n str=%s, cstr=%s",str,cstr);
        for(j=0,i=len-1;i>=0;i--)
                rstr[j++]=str[i];
        str[i]='\0';
        printf("\nStr=%s, rstr=%s",str,rstr);
}
```

Output:
input a string:Hello
Length of Str= 5
str=Hello, cstr=Hello
Str=Hello, rstr=olleH

String Library Functions

- There are so many String library functions which can be used for string manipulation.
- For string library functions, use header file **<string.h>**
- Some of these string functions are:
  - String length : **strlen(s)** gives string length of string s.
  - String copy : **strcpy(s1,s2)** – Copies str2 into str1.
  - String concatenation : **strcat(s1,s2)** –String compare : **strcmp(s1,s2)**- Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
  - **strrev(s1):** reverse the string s1.

```
#include<stdio.h>
#include<string.h>
int main()
{
  char str1[] = "Hello";
  char str2[] = "World";
  char str3[12];
  int  len ;
              /* copy str1 into str3 */
  strcpy(str3, str1);
  printf("strcpy(str3,str1): %s\n", str3 );
  if(strcmp(str1,str3)==0)
              printf("\nStr1 and Str3 are identical");
  else printf("\nstr1 and str3 are not identical");
               /* concatenates str1 and str2 */
  strcat( str1, str2);
  printf("\nstrcat( str1, str2):%s\n", str1 );
              /* total lenghth of str1 after concatenation */
  len = strlen(str1);
  printf("\nstrlen(str1) :%d\n", len );
  if(strcmp(str1,str3)==0)
                          printf("\nStr1 and Str3 are identical");
  else printf("\nstr1 and str3 are not identical");
  return 0;
}
```