C Array

So far we have used fundamental or primary data types or variables namely int, float, double, char. These types of variables are very useful but they can store only one value at a time. Consider following example

```
#include<stdio.h>
Int main()
{
Int a=2;
a=5;
printf("%d",a);
return 0;
}
```

In above program value of a printed is 4. Initially 2 is assigned to a and after that new value 4 is assigned to a which replaces the old value. This fact is same for all types of variables which stores only one latest value. In many applications we need to handle large set of values of same type at a time for example roll numbers of 100 students, marks of 100 students, list of employees in organisation. Declaration and handling large number of variables for individual value is increasing the complexity and size of the program. To overcome this drawback of primary variables and to handle large set of similar values c has provision of derived or secondary variable called as array.

What is array?

An array is defined as the collection of similar type of data items stored at contiguous memory locations. Arrays are the derived or secondary data type in C programming language which can store the multiple values of primary type of data such as int, char, double, float, etc. The array is the simplest data structure where each data element can be randomly accessed by using its index number. C array is beneficial if you have to store similar elements. For example, if we want to store the marks of a 100 students then we don't need to define different variables for the marks of each student. Instead of that, we can define an array which can store the marks all students at the contiguous memory locations. By using the array, we can access the elements easily. Only a few lines of code are required to access the elements of the array.

What are the properties of Array?

The array contains the following properties.

- 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.

What are the advantage of C Array

1) Code Optimization: Less code to the access the data.

2) Ease of traversing: By using the for loop, we can retrieve the elements of an array easily.

3) Ease of sorting: To sort the elements of the array, we need a few lines of code only.

4) Random Access: We can access any element randomly using the array.

What are the disadvantages of C Array?

1) Fixed Size: Whatever size, we define at the time of declaration of the array, we can't exceed the limit. So, it doesn't grow the size dynamically.

2) Similar data: Array can store only same type of data.

Declaration of C Array

Like any other variable array must be declared before they are used. The general form of array declaration is.

data_type array_name[array_size];

Data type specifies the types of elements that array contains such as int, float, char and size indicates the maximum number of elements that can be stored in array.

Example

```
int marks[5];
```

Here, int is the *data_type*, marks are the *array_name*, and 5 is the *array_size*.

float average[6];

Initialization of C Array

The simplest way to initialize an array is by using the index of each element. We can initialize each element of the array by using the index. Consider the following example.

```
marks[0]=80;//initialization of array
marks[1]=60;
marks[2]=70;
marks[3]=85;
marks[4]=75;
```

marks[0]	marks[1]	marks[2]	marks[3]	marks[4]
80	60	70	85	75

C Array: Declaration with Initialization

Similar to primary variable array can be initialised at the time of declaration. The syntax for Declaration with Initialization is as below,

data_type array_name[array_size]={list of elements};

Let's see the code.

```
int marks[5]={20,30,40,50,60};
float per[4]={70.6,35.6,58.35,66.4};
```

In such case, there is **no requirement to define the size**. So it may also be written as the following code.

```
int marks[]={20,30,40,50,60};
```

Array Example	**Array declaration with initialization.**
#include<stdio.h> int main() { int i=0; int marks[5]; //declaration of array marks[0]=80; //initialization of array marks[1]=60; marks[2]=70; marks[3]=85; marks[4]=75; //Display of array for(i=0;i<5;i++) { printf("%d \n",marks[i]); }//end of for loop } /* Output 80 60 70 85 75	#include<stdio.h> int main() { int i=0; int marks[5]={20,30,40,50,60} for(i=0;i<5;i++) { printf("%d \n",marks[i]); } return 0; } /*Output 20 30 40 50 60

Accessing or handling of array elements-

Handling of array elements requires- input, arithmetic operations and output. The input values to the array elements can be given by two ways, one is by initializing the values of elements but this is not convenient way to initialize the large no of elements.

Second way is to use input statements similar to primary variables. This method is convenient because once the array is declared, individual element of array can be accessed by its subscript in the bracket of array. This number specifies the position of the element in the array. All elements of the array are numbered starting with the zero. Thus marks[2] is not second element it is third element. To access the elements of array index variable is used as a subscript or position of the element in array or element number. Thus, the different elements can be accessed by its index or subscript. By the use of loop control statements this method is very useful for input, arithmetic operations and output. Following example shows the all these operations.

//Input to array elements for(i=0;i<10;i++) { scanf("%d ",marks[i]); }	The loop repeats the scanf() statement for entering10 elements of array. We can give input for marks[0] to marks[9].
//accessing individual element for(i=0;i<10;i++) { Sum=sum+marks[i]; }	Here individual element is added in sum using for loop which will give sum of all elements.
for(i=0;i<10;i++) { printf("%d \n",marks[i]); }	Here this loop will display marks of one by one lement.
/*average marks obtained by class of 30 students*/ #include<stdio.h> int main() { int i,sum=0;	/* Find smallest element in a 1-D array */ #include<stdio.h> int main() { int a[30], i, num, smallest; printf("\nEnter no of elements :");

```
float average;
int marks[30];
for (i=0;i<=29;i++)
  {
printf("Enter the marks of %d th student",i);
          scanf("%d",&marks[i]);
    }
          for (i=0;i<=29;i++)
          {
                    sum=sum+marks[i];
          }
average=sum/30.0;
printf("Average marks=%f",average);
return 0;
}
```

```
   scanf("%d", &num);
   printf("\nEnter elements of array :");
   for (i = 0; i < num; i++)
      scanf("%d", &a[i]);
   smallest = a[0];
   for (i = 0; i < num; i++)
   {
      if (a[i] < smallest)
     {
         smallest = a[i];
      }
   }
 printf("\nSmallest Element : %d", smallest);
 return (0);
}
```

```
/*Find sum of odd numbers in 1-D array */
#include<stdio.h>
main()
{
   int a[10],i,sum=0;
   printf("Enter elements of array: ");
   for(i=0; i<5; i++)
      scanf("%d",&a[i]);
   for(i=0; i<5; i++)
   {
      if(a[i]%2==1)
         sum=sum+a[i];
   }
   printf("Sum of Odd values= %d ",sum);
}
```

```
/* Fibonacci series up to number n using array */
#include<stdio.h>
int main()
{
   int i,n,fib[25];
   printf("Enter the number of terms");
   scanf("%d",&n);
   fib[0] = 0;
   fib[1] = 1;
   for (i = 2;i<n;i++)
         {
      fib[i]=fib[i-1]+fib[i-2];
   }
   printf("The fibonacci series is as follows
");
     for (i=0;i<n;i++)
         {
```

```
        printf("%d ", fib[i]);
    }
    return 0;
}
```

/* Implement C program to merge two 1-D arrays into third 1-D array.*/

```
#include<stdio.h>
int main()
{
    int arr1[50], arr2[50], size1, size2, i, k, merge[100];
    printf("Enter Array 1 Size: ");
    scanf("%d", &size1);
    printf("Enter Array 1 Elements: ");
    for(i=0; i<size1; i++)
    {
        scanf("%d", &arr1[i]);
        merge[i] = arr1[i];
    }
    k = i;
    printf("\nEnter Array 2 Size: ");
    scanf("%d", &size2);
    printf("Enter Array 2 Elements: ");
    for(i=0; i<size2; i++)
    {
        scanf("%d", &arr2[i]);
        merge[k] = arr2[i];
        k++;
    }
    printf("\nThe new array after merging is:\n");
    for(i=0; i<k; i++)
        printf("%d ", merge[i]);
    getch();
    return 0;
}
```

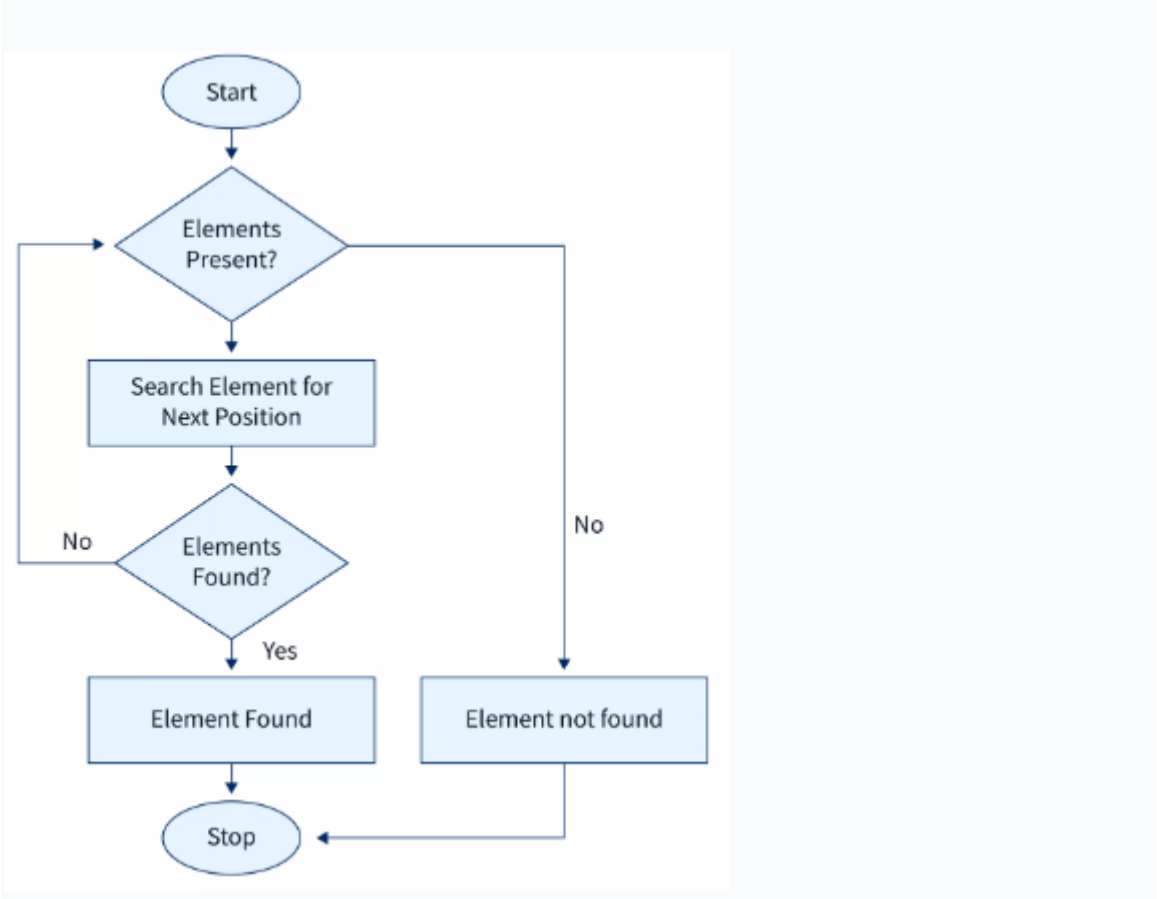
Linear Search in C

Linear Search is the most basic method of searching an element of array. The Linear Search algorithm in C **sequentially** checks each element of the list until the key element is found or the entire list has been traversed. It is also known as sequential search, as it sequentially checks each element of the list until the **key element** is found or the entire list has been traversed. It uses conditional statements and relational operators to find whether the given element is present in the **list or not**. It is easy to learn and implement.

Approach to Implement Linear Search Algorithm in C

- Take input of the element that is going to be searched. It can be referred to as a key element.
- Compare every element of the list with the key element starting from the leftmost end of the list.
- If any element of the list matches with the key, **return the index** of that element.
- If the entire list has been traversed and none of the elements matched with the key, then return -1, which specifies the key element is not present in the list.

Flow Chart of the Linear Search Algorithm



/* To search an element in a given 1-D array.*/

```
#include<stdio.h>
int main()
{
  int arr[20], size, key, i, index;
  printf("Number of elements in the array: ");
  scanf("%d", &size);
  printf("Enter elements of the list: ");
  for (i = 0; i < size; i++)
   scanf("%d", &arr[i]);
  printf("Enter element to search: ");
  scanf("%d", &key);
  for (index = 0; index < size; index++)
    if (arr[index] == key)
            break;
  if (index < size)
   printf("Key element is at index %d", index);
  else
    printf("Key element not found");
  return 0;
}
/*Output
Number of elements in the array: 5
Enter elements of the list:
8       9       7       6       4
Enter element to search: 7
Key element is at index 2
```

```
#include<stdio.h>
int main()
{
  int arr[5], key, i, index;
    printf("Enter elements of the list: ");
  for (i = 0; i < 5; i++)
   scanf("%d", &arr[i]);
  printf("Enter element to search: ");
  scanf("%d", &key);
  for (index = 0; index < 5; index++)
    if (arr[index] == key)
            break;
  if (index < 5)
    printf("key element is at index %d", index);
  else
    printf("Key element not found");
  return 0;
}
/*Output
Enter elements of the list:
5       6       8       9       10
Enter element to search: 6
key element is at index 1
```

```
/*Linear Search in C for Multiple Occurrences*/
#include<stdio.h>
int main()
{
  int arr[20], size, key, i, index;
  int countKey = 0;
  printf("Number of elements in the list: ");
  scanf("%d", &size);
  printf("Enter elements of the list: ");
  for (i = 0; i < size; i++)
    scanf("%d", &arr[i]);
  printf("Enter the element to search i.e. key element: ");
  scanf("%d", &key);
  for (index = 0; index < size; index++)
  {
    if (arr[index] == key)
          {
      printf("Key element found at index %d\n", index);
      countKey++;
    }
  }
  if (countKey == 0)
    printf("Key element not found");
  else
    printf("Key element is present %d times in the array.\n", countKey);
  return 0;
}
/*Output
Number of elements in the list: 5
Enter elements of the list:
8       7       7       9       4
Enter the element to search i.e. key element: 7
Key element found at index 1
Key element found at index 2
Key element is present 2 times in the array.
```

Passing Array to Function in C. (How to pass array to user defined function?)

In C, there are various general problems which requires passing more than one variable of the same type to a function. For example, consider a function which sorts the 10 elements in ascending order. Such a function requires 10 numbers to be passed as the actual parameters from the main function. Here, instead of declaring 10 different numbers and then passing into the function, we can declare and initialize an array and pass that into the function. This will resolve all the complexity since the function will now work for any number of values.

The array name itself is the address of first element of the array. For example if array name is arr then you can say that **arr** is equivalent to the **&arr[0].** We need to pass only the name of the array in the function which is intended to accept an array. The array defined as the formal parameter will automatically receive the array specified by the array name defined as an actual parameter. Syntax for declaration of function to handle the array is as below.

1. **return_type function_name(data_type[])** ;

In above syntax in argument data type of array elements with blank square bracket is written. So the formal argument is array name with square bracket only. In actual argument base address i.e. name of the array is used.

void Sort(int[]); //Function declaration

General structure of program on passing the array to function

```
return_type Function_Name(data_type[ ]);   // function declaration with argument as array
int main()
{
   ----------
   variable=Function_Name (array_Name); //function call with actual argument as array name,
   --------
    return 0;
}
return_type Function_Name(data_type array_name[ ]) //function definition to receive array.
{
        ----------
        ----------
        return (value or expression);
}
```

2. **return_type function_name(data_type arrayname[], int size);**

In above syntax depending upon value returned by function return type is given, then function name. In argument data type is types of array elements and second argument is size of array which is int type. In function call two actual arguments are name of the array and size of the array.

e.g.

```
int minarray(int arr[],int size)
```

```
return_type Function_Name(data_type[ ],int);
                                    // function declaration with arguments array & size
int main()
{
   ----------
   variable= Function_Name (array_Name, no);
                                //function call with actual arguments as array name & size,
   -------
    return 0;
}
return_type Function_Name(data_type array_name[ ],int size)
                                    //function definition to receive array & its size.
{
        ----------
        ----------
       return (value or expression);
}
```

```
#include<stdio.h>
int minarray(int arr[],int);                    // Declaration of function to pass arr[ ]
int main()
{
        int i,min;
        int numbers[6];
        printf("Enter the array elements");
        for(i=0;i<6;i++)
        {
                scanf("%d",&numbers[i]);
        }
        min=minarray(numbers,6);                //function call passing array with size
        printf("minimum number is %d \n",min);
        return 0;
}
int minarray(int arr[],int size)                //function definition to  receive actual arguments
{
```

```
        int min=arr[0];
        int i=0;
        for(i=1;i<size;i++)
        {
        if(min>arr[i])
                {
                min=arr[i];
                }
        }
return min;
}
```

```
/*C function to sort the array*/
#include<stdio.h>
void Sort(int[]);  //Function declaration
int main ()
{
   int arr[10] = { 10, 9, 7, 101, 23, 44, 12, 78, 34, 23};
   Sort(arr);                          //Function call with arr name or base address of arr.
         return 0;
}
void Sort(int a[])                      //array a[] points to arr.
{
int i, j,temp;
   for(i = 0; i<10; i++)
   {
      for(j = i+1; j<10; j++)
      {
         if(a[j] < a[i])
         {
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
         }
      }
   }
   printf("Printing Sorted Element List ...\n");
   for(i = 0; i<10; i++)
   {
      printf("%d\n",a[i]);
   }
}
```

Two Dimensional Array in C- The two-dimensional array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as the collection of rows and columns.

Declaration of two-dimensional Array in C- The syntax to declare the 2D array is given below.

```
data_type array_name[rows][columns];
```

example - int twodimen[4][3]; - Here, 4 is the number of rows, and 3 is the number of columns.

Example of arr2d[3][3] is as shown in following table.

	Column1	**Column2**	**Column3**
Row1	arr2d[0][0]	arr2d[0][1]	arr2d[0][2]
Row2	arr2d[1][0]	arr2d[1][1]	arr2d[1][2]
Row3	arr2d[2][0]	arr2d[2][1]	arr2d[2][2]

Initialization of 2D Array in C

The two-dimensional array can be declared and initialised in the following way.

```
int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
```

Example of 2D array

```
#include<stdio.h>
int main()
{
int i=0,j=0;
int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
for(i=0;i<4;i++)
{
  for(j=0;j<3;j++)
   {
    printf("arr[%d][%d] = %d \t",i,j,arr[i][j]);
   }
printf("\n");
}
return 0;
}
/*Output
arr[0][0] = 1,  arr[0][1] = 2,  arr[0][2] = 3,
arr[1][0] = 2,  arr[1][1] = 3,  arr[1][2] = 4,
arr[2][0] = 3,  arr[2][1] = 4,  arr[2][2] = 5,
arr[3][0] = 4,  arr[3][1] = 5,  arr[3][2] = 6,
```

Accessing or handling of 2D array elements-

```
#include<stdio.h>
int main()
{
int i,j;
int arr[4][3];                          //Declaration of array with 4 rows 3 columns
printf("Enter the elements of array");
for(i=0;i<4;i++)
{
  for(j=0;j<3;j++)
   {
    scanf("%d",&arr[i][j]);
        }
}
printf("Display the array elements\n");
for(i=0;i<4;i++)
{
  for(j=0;j<3;j++)
   {
    printf("arr[%d][%d] = %d,\t",i,j,arr[i][j]);
        }
        printf("\n");
}
return 0;
}
```

Loop for input array elements

Loop for output of array elements

/*Find the sum of all elements in a 2-D array */	**/* Display even and odd numbers from given 2-D array*/**
#include<stdio.h> int main() { int i,j,sum=0; int arr[4][3]; printf("Enter the elements of array");	#include<stdio.h> int main() { int i,j; int arr[3][3]; printf("Enter the elements of array"); for(i=0;i<3;i++)

```
for(i=0;i<3;i++)
{
  for(j=0;j<3;j++)
   {
    scanf("%d",&arr[i][j]);
   }
}
//calculate sum of all elements
for(i=0;i<3;i++)
{
  for(j=0;j<3;j++)
   {
    sum=sum+arr[i][j];
   }
}
printf("\n Sum of all elements=%d",sum);
return 0;
}
/*Output
Enter the elements of array
2      5      6
3      1      3
2      5      7
 Sum of all elements=34
```

```
 {
    for(j=0;j<3;j++)
    {
     scanf("%d",&arr[i][j]);
    }
 }
printf("Even numbers\n");
for(i=0;i<3;i++)
 {
  for(j=0;j<3;j++)
    {
     if(arr[i][j]%2==0)
     printf("%d\t",arr[i][j]);
    }
 }
printf("\n Odd numbers\n");
for(i=0;i<3;i++)
 {
    for(j=0;j<3;j++)
     {
      if(arr[i][j]%2!=0)
      printf("%d\t",arr[i][j]);
     }
  }
return 0;
}
/*Output
Enter the elements of array
23      25      46
12      15      24
17      18      9
Even numbers
46      12      24      18
 Odd numbers
23      25      15      17      9
```

/*Find smallest element in a 2-D array*/

```
#include<stdio.h>
int main()
{
int i,j,smallest;
int a[3][3];
printf("Enter the elements of array");
for(i=0;i<3;i++)
  {
    for(j=0;j<3;j++)
     {
      scanf("%d",&a[i][j]);
     }
  }
smallest=a[0][0];
for(i=0;i<3;i++)
  {
   for(j=0;j<3;j++)
      {
       if(a[i][j]<smallest)
           smallest=a[i][j];
       }
   }
printf("\n smallest no =%d",smallest);
return 0;
}
/*Output
Enter the elements of array
8      9      7
5      6      4
2      5      6

 smallest no =2
```

/* Search an element in a given 2-D array*/

```
#include<stdio.h>
int main()
{
  int item,count=0,array[3][3];
  printf("Enter elements of array:");
  for(int i=0;i<3;i++)
  {
    for(int j=0;j<3;j++)
          {
     scanf("%d",&array[i][j]);
    }
  }
  printf("Enter the item to find:");
  scanf("%d",&item);
  for(int i=0;i<3;i++)
  {
    for(int j=0;j<3;j++)
          {
     if(array[i][j]==item)
           {
       printf("Item found at[%d,%d]\n",i,j);
       count++;
     }
    }
  }
  if(count==0)
    printf("Item Not found");
  return 0;
} /*Output
Enter elements of array:
5      6      7
```

	8 9 5 4 3 2 Enter the item to find:6 Item found at[0,1]
/* Addition of 3*3 Matrix*/ #include<stdio.h> int main() { int i,j,sum=0; int a[3][3],b[3][3],c[3][3]; printf("Enter the elements of array a"); for(i=0;i<3;i++) { for(j=0;j<3;j++) { scanf("%d",&a[i][j]); } } printf("Enter the elements of array b"); for(i=0;i<3;i++) { for(j=0;j<3;j++) { scanf("%d",&b[i][j]); } } //addtion of matrix for(i=0;i<3;i++) { for(j=0;j<3;j++) { c[i][j]=a[i][j]+b[i][j]; } } printf("\n addition of a and b\n"); for(i=0;i<3;i++) { for(j=0;j<3;j++) { printf("%d\t",c[i][j]); } printf("\n"); } return 0; } /*Output Enter the elements of array a 2 5 6	

4 6 3 1 2 3 Enter the elements of array b 8 7 9 6 5 4 5 6 3 addition of a and b 10 12 15 10 11 7 6 8 6	

Working with the strings

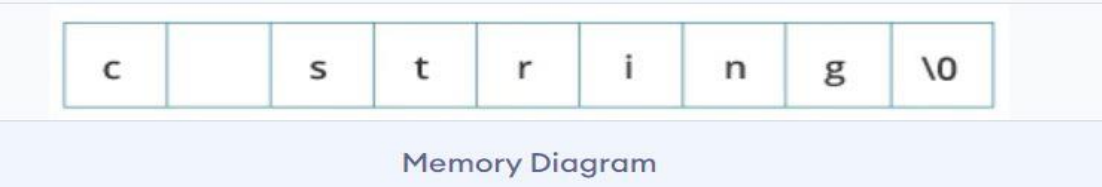
1. What is a string?

In C language the group of characters, digits and symbols enclosed within quotation marks are called as string. The string is always declared as a character array. In other words, character array is called strings. To manipulate, test such as words and sentences normally strings are used. Every string is terminated with null character '\0'. The null character is a byte with all bits at logic zero. Hence its decimal value is zero.

For example

char name[ ]={'I','N','D','I','A','\0'};

Each character of the string occupies 1 byte of memory. The last character is always '\0'. it is not compulsory to write '\0' in string. The compiler automatically puts '\0' at the end of the character array or string. The characters of the string are stored in the contiguous (neighboring) memory locations. Following table shows the storing of string elements in contiguous memory locations or memory map. char c[] = "c string";



Memory Diagram

2. How to declare and initialize the string?

Syntax for declaration of string.

char str_name[size];

Where,

str_name= is any name given to the string variable.

size= is used to define the length of the string, i.e the number of characters, the strings will store.

There is an extra terminating character which is the Null character (‘\0’) used to indicate the termination of string which differs strings from normal arrays.

e.g

```
char c[] = "c string";
```

When the compiler encounters a sequence of characters enclosed in the double quotation marks, it inserts a null character \0 at the end of the string. So initialization of null character is not essential.

initialization of string : A string can be initialized in different ways

1. char str[] = "Welcome To RIT";
2. char str[50] = "Welcome To RIT";
3. char str[] = {'W','e','l','c','o','m','e','T','o','R','I','T'};
4. char str[14] = {'W','e','l','c','o','m','e','T','o','R','I','T' };

```
/*Program to demonstrate initialization of string*/
#include<stdio.h>
#include<string.h>
int main()
{
        char Name1[6]={'S','A','N','J','A','Y'};
        char Name2[]={'S','A','N','J','A','Y'};
        char Name3[7]="SANJAY";
        printf("Name1= %s\n",Name1);
        printf("Name2 = %s\n",Name2);
        printf("Name3 = %s",Name3);
        return 0;
}
/*Output
Name1= SANJAY
Name2 = SANJAY
Name3 = SANJAY
```

Accepting string as an input: -

1._scanf() to read a string:- The scanf() function reads the sequence of characters until it encounters whitespace (space, newline, tab, etc.).

2. gets( ):- function is used to read a string of characters including white spaces. gets() function overcomes the drawback of scanf() function for receiving a multiword string containing spaces.

Syntax :- gets(str);

/* Demo of scanf( ) function for string*/	**/*Program for scanning a string with spaces using gets function:**
#include<stdio.h>	#include<stdio.h>
int main()	int main()
{	{
char name[50];	char name[50];
printf("Enter your full name\n");	printf("Enter your full name\n");
scanf("%s", name);	gets(name);
printf("Full Name : %s",name);	printf("Full Name : %s",name);
return 0;	}
}	**/*Output:**
Output:	Enter your full name
Enter your full name	Rohit Sharma
Rohit Sharma	Full Name : Rohit Sharma
Full Name : Rohit	

3. Traversing a string.

Traversing string is somewhat different from the traversing an integer array. We need to know the length of the array to traverse an integer array, whereas we may use the null character in the case of string to identify the end the string and terminate the loop.

Hence, there are two ways to traverse a string.

- **By using the length of string**
- **By using the NULL character.**

/*Demo of Traversing a string by using the length of string	**/* Demo of Traversing a string by using the NULL character**
#include<stdio.h>	#include<stdio.h>
int main()	int main()
{	{
char Text[]="HAVE A NICE DAY";	char Text[]="HAVE A NICE DAY";

```
    int i=0;
    while(i<=15)
     {
          printf("%c",Text[i]);
          i++;
     }
    return 0;
}
/*Output
HAVE A NICE DAY*/
```

Explanation – while loop is used to print the character up to length 15. Thereafter loop terminates.

```
 int i=0;
 while(Text[i]!='\0')
  {
        printf("%c",Text[i]);
        i++;
  }
 return 0;
}
/*Output
HAVE A NICE DAY*/
```

Explanation- Characters are displayed without knowing length of string by using NULL character in while loop.

/* Find string length without using a function in C*/

```
#include<stdio.h>
int main()
{
  char Text[50];
  int i=0;
  gets(Text);
  while(Text[i]!='\0')
   {
         i++;
   }
   printf("Length of string %d",i);
  return 0;
}
/*Output
Rajarambapu Institute of Technology
Length of string 35
```

/*Copy one string to another without using function*/

```
#include<stdio.h>
#include<string.h>
int main()
{
        char ori[20],dup[20];
        int i;
        printf("Enter original string\n");
        gets(ori);
        for(i=0;i<20;i++)
        {
                dup[i]=ori[i];
        }
        printf("Duplicate string \n%s",dup);
}
/*output
Enter original string
Suryakumar Yadav
Duplicate string
Suryakumar Yadav
```

/*Concatenate one string to another without using function*/

```
#include<stdio.h>
#include<string.h>
int main()
{
char Name[50],Fname[20],Lname[20];
        int i,j,k;
        printf("Enter first name\n");
        gets(Fname);
        printf("Enter last name\n");
        gets(Lname);
        for(i=0;Fname[i]!='\0';i++)
          Name[i]=Fname[i];
        Name[i]=' ';
        for(j=0;Lname[j]!='\0';j++)
              Name[i+j+1]=Lname[j];
        printf("Name \n%s",Name);
}
/*Output
Enter first name
Rajaram
Enter last name
Shinde
Name
Rajaram Shinde
```

/*Count character occurrence in a string.*/

```
#include <stdio.h>
int main()
{
   char str[50];
   char toSearch;
   int i=0, count=0;
   printf("Enter any string: ");
   gets(str);
   printf("Enter any character to search: ");
   toSearch = getchar();
   while(str[i] != '\0')
   {
      if(str[i] == toSearch)
      {
         count++;
      }
      i++;
   }
   printf("Total occurrence of '%c' = %d",
toSearch, count);
   return 0;
}
```

/* Count the number of vowels, consonants, digits and white space in a given string.*/

```
#include<stdio.h>
int main ()
{
   char s[]="100 Characters are displayed without knowing length of string by using";
   int i = 0;
   int vowels=0, consonants=0, digits=0, spaces=0;
```

```
   while(s[i] != '\0')
   {
      if(s[i]=='a'||s[i] =='e'||s[i]=='i'||s[i]=='u'||s[i]=='o')
      {
         vowels ++;
      }
         else if((s[i]>='a'&&s[i]<='z')||(s[i]>='A'&&s[i]<='Z'))
         {
            consonants++;
         }
         else if(s[i]>='0'&&s[i]<='9')
         {
             digits++;
         }
         else if(s[i]=' ')
         {
            spaces++;
         }
      i++;
   }
         printf("The number of vowels %d\n",vowels);
         printf("The number of consonants %d\n",consonants);
         printf("The number of digits %d\n",digits);
         printf("The number of spaces %d\n",spaces);
}
/*Output
The number of vowels 18
The number of consonants 39
The number of digits 3
The number of spaces 10*/
```

Pre- defined C String functions.

There are many important string functions defined in "string.h" library.

Sr. No.	Function	Description
1.	**strlen(string_name);**	Returns the length of string name. Count no of characters in the string.
2.	**strcpy(destination, source);**	copies the contents of source string to destination string.
3.	**strcat(first_string, second_string);**	concats or joins first string with second string. The result of the string is stored in first string.
4.	**strcmp(first_string, second_string);**	compares the first string with second string. If both strings are same, it returns 0.
5.	**strrev(string);**	returns reverse string.
6.	**strlwr(string);**	returns string characters in lowercase.
7.	**strupr(string);**	returns string characters in uppercase.

/* To find string length without using strlen( ) function*/

```
#include<stdio.h>
#include<string.h>
int main()
{
  char Text[50];
  int length;
  printf("Enter the text\n");
  gets(Text);
  length=strlen(Text);
  printf("Length of string %d",length);
  return 0;
}
/*Output
```

/*Copy one string to another using strcpy()*/

```
#include<stdio.h>
#include<string.h>
int main()
{
        char ori[20],dup[20];
        int i;
        printf("Enter original string\n");
        gets(ori);
        strcpy(dup,ori);
        printf("Duplicate string \n%s",dup);
}
/*Output
Enter original string
Rohit Sharma
```

Enter the text Rajarambapu Institute of Technology Length of string 35	Duplicate string Rohit Sharma
/*Concatenate one string to another using strcat( )*/ #include<stdio.h> #include<string.h> int main() { char name[20],Lname[20]; printf("Enter first name\n"); gets(name); printf("Enter last name\n"); gets(Lname); strcat(name," "); strcat(name,Lname); printf("Name \n%s",name); } /*Output Enter first name Rohit Enter last name Sharma Name Rohit Sharma	**/*Compare two strings strcmp( )*/** #include<stdio.h> #include<string.h> int main() { char text1[20],text2[20]; int diff; printf("Enter first text\n"); gets(text1); printf("Enter second text\n"); gets(text2); diff=strcmp(text1,text2); if(diff==0) printf("Two strings are identical"); else printf("Two strings are different"); } /*Output Enter first text Rohit Enter second text Virat Two strings are different
/*Reverse the string strrve( )*/ #include<stdio.h> #include<string.h> int main() { char text1[20]; printf("Enter the text\n");	/*Convert upper case to lower case strlwr( )*/ #include<stdio.h> #include<string.h> int main() { char text1[20]; printf("Enter the text\n");

```
        gets(text1);
        strrev(text1);
        printf("Reverse string\n");
        puts(text1);
}
/*Output
Enter the text
Rohit
Reverse string
tihoR
```

```
        gets(text1);
        strlwr(text1);
        printf("Lower case string\n");
        puts(text1);
}
/*Output
Enter the text
SACHIN
Lower case string
sachin
```

/*Convert lower case to upper case strlupr( )*/

```
#include<stdio.h>
#include<string.h>
int main()
{
        char text1[20];
        printf("Enter the text\n");
        gets(text1);
        strupr(text1);
        printf("Upper case string\n");
        puts(text1);
}/*Output
Enter the text
suryakumar yadav
Upper case string
SURYAKUMAR YADAV
```

/*check weather string is Palindrome or not (If the original string is equal to reverse of that string, then the string is said to be a palindrome.)*/

```
#include<stdio.h>
#include<string.h>
int main()
{
   char s[100];
   int i,n,c=0;
   printf("Enter  the string : ");
   gets(s);
   n=strlen(s);
    for(i=0;i<n;i++)
    {
        if(s[i]==s[n-i-1])
        c++;
         }
        if(c==i)
            printf("string is palindrome");
   else
      printf("string is not palindrome");
 }
```