**Arrays in C**

An array is defined as the collection of similar type of data items stored at contiguous memory locations. Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc. It also has the capability to store the collection of derived data types, such as pointers, structure, 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 student in 6 subjects, then we don't need to define different variables for the marks in the different subject. Instead of that, we can define an array which can store the marks in each subject 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.

**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.

**Advantages 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.

**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 like LinkedList which we will learn later.

**Declaration of C Array**

We can declare an array in the c language in the following way.

**data\_type array\_name[array\_size];**

Now, let us see the example to declare the array.

*int marks[5];*

Here, int is the data\_type, marks are the array\_name, and 5 is the array\_size.

**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;

**initialization of array in c language**

**C array example**

#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;

//**traversal of array**

for(i=0;i<5;i++){

printf("%d \n",marks[i]);

}//end of for loop return 0;

}

**Output**

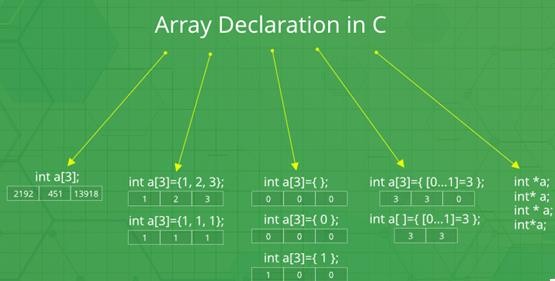
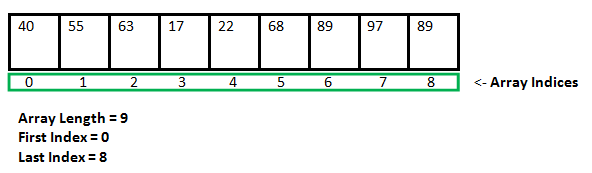
80

60

70

85

75



There are various ways in which we can declare an array. It can be done by specifying its type and size, by initializing it or both.

1. **Array declaration by specifying size**

// Array declaration by specifying size

int arr1[10];

// With recent C/C++ versions, we can also

// declare an array of user specified size int n = 10;

int arr2[n];

2. **Array declaration by initializing elements**

// Array declaration by initializing elements int arr[] = { 10, 20, 30, 40 };

// Compiler creates an array of size 4.

// above is same as "int arr[4] = {10, 20, 30, 40};"

3. **Array declaration by specifying size and initializing elements**

// Array declaration by specifying size and initializing

// elements

int arr[6] = { 10, 20, 30, 40 }

// Compiler creates an array of size 6, initializes first

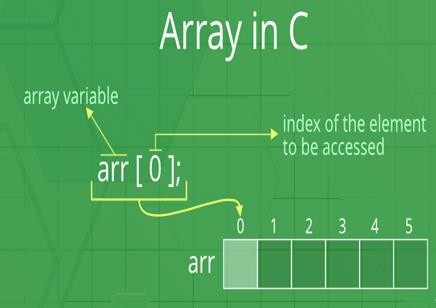
// 4 elements as specified by user and rest two elements as 0.

// above is same as "int arr[] = {10, 20, 30, 40, 0, 0};"

**Accessing Array Elements**

Array elements are accessed by using an integer index. Array index starts with 0 and goes till size

of array minus 1.



**Example:**

#include <stdio.h>

int main()

{

int arr[5]; arr[0] = 5; arr[1] = -10; arr[2] = 2; arr[3] = arr[0];

printf("%d %d %d %d", arr[0], arr[1], arr[2], arr[3]);

return 0;

}

**Output:**

5 -10 2 5

**The elements are stored at contiguous memory locations**

**Example:**

// C program to demonstrate that array elements are stored

// contiguous locations

#include <stdio.h>

int main()

{

// an array of 10 integers. If arr[0] is stored at

// address x, then arr[1] is stored at x + sizeof(int)

// arr[2] is stored at x + sizeof(int) + sizeof(int)

// and so on. int arr[5], i;

printf("Size of integer in this compiler is %lu\n", sizeof(int));

for (i = 0; i < 5; i++)

// The use of '&' before a variable name, yields

// address of variable.

printf("Address arr[%d] is %p\n", i, &arr[i]);

return 0;

}

**Output:**

Size of integer in this compiler is 4

Address arr[0] is 0x7ffd636b4260

Address arr[1] is 0x7ffd636b4264

Address arr[2] is 0x7ffd636b4268

Address arr[3] is 0x7ffd636b426c

Address arr[4] is 0x7ffd636b4270

**Some Programs on Arrays**

*C program to declare and initialize the array.*

#include<stdio.h>

**int** main(){

**int** i=0;

**int** marks[5]={20,30,40,50,60};//declaration and initialization of array

//traversal of array **for**(i=0;i<5;i++){ printf("%d \n",marks[i]);

}

**return** 0;

}

**Output**

20

30

40

50

60

*C Programs for Sorting an array*

#include<stdio.h>

**void** main ()

{

**int** i, j,temp;

**int** a[10] = { 10, 9, 7, 101, 23, 44, 12, 78, 34, 23};

**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]);

}

}

*C* Program to print the largest and second largest element of the array.

#include<stdio.h>

**void** main ()

{

**int** arr[100],i,n,largest,sec\_largest; printf("Enter the size of the array?"); scanf("%d",&n);

printf("Enter the elements of the array?");

**for**(i = 0; i<n; i++)

{

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

}

largest = arr[0]; sec\_largest = arr[1]; **for**(i=0;i<n;i++)

{

**if**(arr[i]>largest)

{

sec\_largest = largest;

largest = arr[i];

}

**else if** (arr[i]>sec\_largest && arr[i]!=largest)

{

sec\_largest=arr[i];

}

}

printf("largest = %d, second largest = %d",largest,sec\_largest);

}

*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 | | |

*// Program to find the average of n numbers using arrays*

#include <stdio.h>

int main()

{

int marks[10], i, n, sum = 0, average;

printf("Enter number of elements: ");

|  |  |  |
| --- | --- | --- |
| scanf("%d", &n);  for(i=0; i<n; ++i)  {  printf("Enter number%d: ",i+1);  scanf("%d", &marks[i]);  // adding integers entered by the user to the sum variable sum += marks[i];  }  average = sum/n;  printf("Average = %d", average);  return 0;  } | | |
|  | **Output** |  |
| Enter n: 5  Enter number1: 45  Enter number2: 35  Enter number3: 38  Enter number4: 31  Enter number5: 49  Average = 39 | | |

*There are seven numbers stored in an array. The following program prints all the numbers of that*

*array as well as prints the numbers in backward.*

#include<stdio.h>

int main()

{

int M[10];

int j;

/\* store seven numbers in array M \*/

M[0] = 2;

M[1] = 4;

M[2] = 6;

M[3] = 8;

M[4] = 10;

M[5] = 12;

M[6] = 14;

/\* print numbers in M \*/

printf("Print all the Numbers : \n");

for (j = 0; j < 7; ++j)

printf("M[%d] = %d\n",j,M[j]);

/\* print numbers in M backwards \*/

printf("\nFrom End to Beginning : \n");

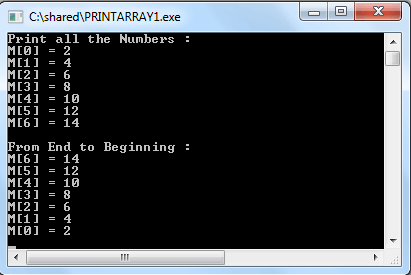
for (j = 6; j >= 0; --j)

{

printf("M[%d] = %d\n",j,M[j]);

}

} Output:



**Copying arrays:**

We have two arrays list1 and list2

int list1[6] = {2, 4, 6, 8, 10, 12};

int list2[6];

and we want to copy the contents of list1 to list2. For general variables (e.g. int x=3, y=5) we use simple assignment statement (x=y or y=x). But for arrays the following statement is wrong.

list2 = list1;

We must copy between arrays element by element and the two arrays must have the same size. In the following example, we use a for loop which makes this easy.

#include <stdio.h>

main()

int list1[6] = {2, 4, 6, 8, 10, 12};

int list2[6];

for (int ctr = 0; ctr<6; ctr++)

{

list2[ctr] = list1[ctr];

}

printf("Elements of list2 :\n");

for (int ctr = 0; ctr<6; ctr++)

{

printf("%d ",list2[ctr]);

}

} Copy

Output:

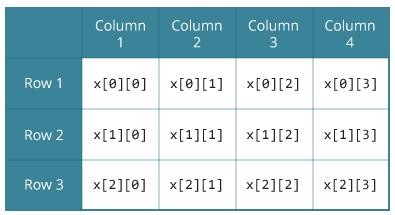
Elements of list2 :

2 4 6 8 10 12

**C Multidimensional Arrays**

In C programming, you can create an array of arrays. These arrays are known as multidimensional

arrays. For example,



float x[3][4];

Here, x is a two-dimensional (2d) array. The array can hold 12 elements. You can think the array

as a table with 3 rows and each row has 4 columns.

Similarly, you can declare a three-dimensional (3d) array. For example,

float y[2][4][3];

Here, the array y can hold 24 elements.

**Initializing a multidimensional array**

Here is how you can initialize two-dimensional and three-dimensional arrays:

**Initialization of a 2d array**

// Different ways to initialize two-dimensional array

int c[2][3] = {{1, 3, 0}, {-1, 5, 9}};

int c[][3] = {{1, 3, 0}, {-1, 5, 9}};

int c[2][3] = {1, 3, 0, -1, 5, 9};

**Initialization of a 3d array**

You can initialize a three-dimensional array in a similar way like a two-dimensional array. Here's

an example,

int test[2][3][4] = {

{{3, 4, 2, 3}, {0, -3, 9, 11}, {23, 12, 23, 2}},

{{13, 4, 56, 3}, {5, 9, 3, 5}, {3, 1, 4, 9}}};

Example 1: Two-dimensional array to store and print values

// C program to store temperature of two cities of a week and display it.

#include <stdio.h> const int CITY = 2; const int WEEK = 7; int main()

{

int temperature[CITY][WEEK];

// Using nested loop to store values in a 2d array for (int i = 0; i < CITY; ++i)

{

for (int j = 0; j < WEEK; ++j)

{

printf("City %d, Day %d: ", i + 1, j + 1);

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

}

}

printf("\nDisplaying values: \n\n");

// Using nested loop to display vlues of a 2d array for (int i = 0; i < CITY; ++i)

{

for (int j = 0; j < WEEK; ++j)

{

|  |  |  |
| --- | --- | --- |
| printf("City %d, Day %d = %d\n", i + 1, j + 1, temperature[i][j]);  }  }  return 0;  } | | |
|  | **Output** |  |
| City 1, Day 1: 33  City 1, Day 2: 34  City 1, Day 3: 35  City 1, Day 4: 33  City 1, Day 5: 32  City 1, Day 6: 31  City 1, Day 7: 30  City 2, Day 1: 23  City 2, Day 2: 22  City 2, Day 3: 21  City 2, Day 4: 24  City 2, Day 5: 22  City 2, Day 6: 25  City 2, Day 7: 26  Displaying values: City 1, Day 1 = 33  City 1, Day 2 = 34  City 1, Day 3 = 35  City 1, Day 4 = 33  City 1, Day 5 = 32  City 1, Day 6 = 31  City 1, Day 7 = 30  City 2, Day 1 = 23  City 2, Day 2 = 22  City 2, Day 3 = 21  City 2, Day 4 = 24  City 2, Day 5 = 22  City 2, Day 6 = 25  City 2, Day 7 = 26 | | |

Example 2: Sum of two matrices

// C program to find the sum of two matrices of order 2\*2

#include <stdio.h>

int main()

{

float a[2][2], b[2][2], result[2][2];

// Taking input using nested for loop printf("Enter elements of 1st matrix\n"); for (int i = 0; i < 2; ++i)

for (int j = 0; j < 2; ++j)

{

printf("Enter a%d%d: ", i + 1, j + 1);

scanf("%f", &a[i][j]);

}

// Taking input using nested for loop printf("Enter elements of 2nd matrix\n"); for (int i = 0; i < 2; ++i)

for (int j = 0; j < 2; ++j)

{

printf("Enter b%d%d: ", i + 1, j + 1);

scanf("%f", &b[i][j]);

}

// adding corresponding elements of two arrays for (int i = 0; i < 2; ++i)

for (int j = 0; j < 2; ++j)

{

result[i][j] = a[i][j] + b[i][j];

}

// Displaying the sum printf("\nSum Of Matrix:");

for (int i = 0; i < 2; ++i)

for (int j = 0; j < 2; ++j)

{

printf("%.1f\t", result[i][j]);

if (j == 1)

printf("\n");

|  |  |  |
| --- | --- | --- |
| }  return 0;  } | | |
|  | **Output** |  |
| Enter elements of 1st matrix  Enter a11: 2; Enter a12: 0.5; Enter a21: -1.1; Enter a22: 2;  Enter elements of 2nd matrix  Enter b11: 0.2; Enter b12: 0; Enter b21: 0.23; Enter b22: 23;  Sum Of Matrix:  2.2 0.5  -0.9 25.0 | | |
|  | Example 3: Three-dimensional array |  |

// C Program to store and print 12 values entered by the user

#include <stdio.h>

int main()

{

int test[2][3][2];

printf("Enter 12 values: \n");

for (int i = 0; i < 2; ++i)

{

for (int j = 0; j < 3; ++j)

{

for (int k = 0; k < 2; ++k)

{

scanf("%d", &test[i][j][k]);

}

}

}

|  |  |  |
| --- | --- | --- |
| // Printing values with proper index.  printf("\nDisplaying values:\n");  for (int i = 0; i < 2; ++i)  {  for (int j = 0; j < 3; ++j)  {  for (int k = 0; k < 2; ++k)  {  printf("test[%d][%d][%d] = %d\n", i, j, k, test[i][j][k]);  }  }  }  return 0;  } | | |
|  | **Output** |  |
| Enter 12 values:  1  2  3  4  5  6  7  8  9  10  11  12  Displaying Values: test[0][0][0] = 1 test[0][0][1] = 2 test[0][1][0] = 3 test[0][1][1] = 4 test[0][2][0] = 5 test[0][2][1] = 6 test[1][0][0] = 7 test[1][0][1] = 8 test[1][1][0] = 9 test[1][1][1] = 10 | | |

test[1][2][0] = 11 test[1][2][1] = 12

C program demonstrates how to store the elements entered by user in a 2d array and how to display the elements of a two dimensional array.

#include<stdio.h>

int main(){

/\* 2D array declaration\*/

int disp[2][3];

/\*Counter variables for the loop\*/

int i, j;

for(i=0; i<2; i++) {

for(j=0;j<3;j++) {

printf("Enter value for disp[%d][%d]:", i, j);

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

}

}

//Displaying array elements

printf("Two Dimensional array elements:\n");

for(i=0; i<2; i++) {

for(j=0;j<3;j++) { printf("%d ", disp[i][j]); if(j==2){

printf("\n");

}

}

}

return 0;

}

Output:

Enter value for disp[0][0]:1

Enter value for disp[0][1]:2

Enter value for disp[0][2]:3

Enter value for disp[1][0]:4

Enter value for disp[1][1]:5

Enter value for disp[1][2]:6

Two Dimensional array elements:

1 2 3

4 5 6

*Things that you must consider while initializing a 2D array*

We already know, when we initialize a normal [array](https://beginnersbook.com/2014/01/c-arrays-example/) (or you can say one dimensional array) during declaration, we need not to specify the size of it. However that’s not the case with 2D array, you must always specify the second dimension even if you are specifying elements during the declaration. Let’s understand this with the help of few examples –

/\* Valid declaration\*/

int abc[2][2] = {1, 2, 3 ,4 }

/\* Valid declaration\*/

int abc[][2] = {1, 2, 3 ,4 }

/\* Invalid declaration – you must specify second dimension\*/

int abc[][] = {1, 2, 3 ,4 }

/\* Invalid because of the same reason mentioned above\*/

int abc[2][] = {1, 2, 3 ,4 }

How to store user input data into 2D array

We can calculate how many elements a two dimensional array can have by using this formula: The array arr[n1][n2] can have n1\*n2 elements. The array that we have in the example below is having the dimensions 5 and 4. These dimensions are known as subscripts. So this array has **first subscript** value as 5 and **second subscript** value as 4. So the array abc[5][4] can have 5\*4 = 20 elements.

To store the elements entered by user we are using two for loops, one of them is a nested loop. The outer loop runs from 0 to the (first subscript -1) and the inner for loops runs from 0 to the (second subscript -1). This way the the order in which user enters the elements would be abc[0][0], abc[0][1], abc[0][2]…so on.

#include<stdio.h>

int main(){

/\* 2D array declaration\*/

int abc[5][4];

/\*Counter variables for the loop\*/

int i, j;

for(i=0; i<5; i++) {

for(j=0;j<4;j++) {

printf("Enter value for abc[%d][%d]:", i, j);

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

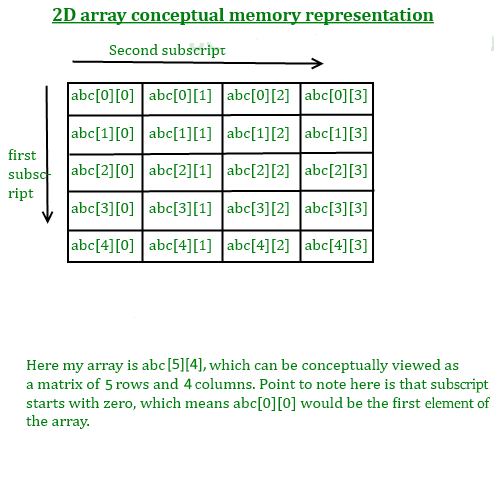
}

}

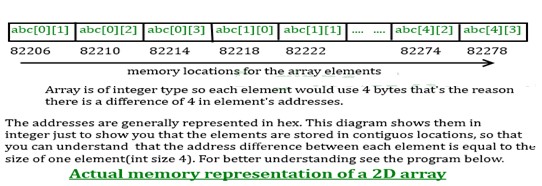
return 0;

}

In above example, We have a 2D array abc of integer type. Conceptually we can visualize the above array like this:



However the actual representation of this array in memory would be something like this:



**C Character Array and Strings**

The string in C programming language is actually a one-dimensional array of characters which is 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". 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".

char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};

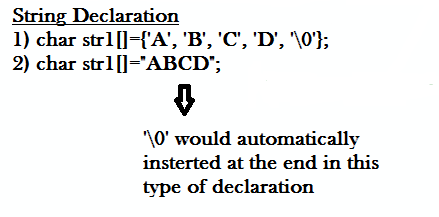
If We follow the rule of array initialization then we can write the above statement as follows:

char greeting[] = "Hello";

Following is the memory presentation of above-defined string in C.



String Declaration



**Method 1:**

char address[]={'T', 'E', 'X', 'A', 'S', '\0'};

**Method 2: The above string can also be defined as** –

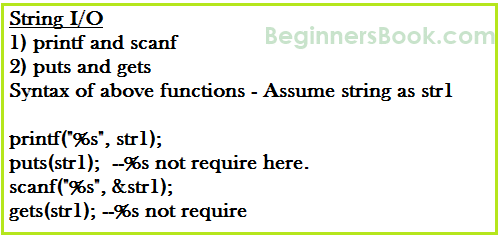
char address[]="TEXAS";

In the above declaration NULL character (\0) will automatically be inserted at the end of the string.

**What is NULL Char “\0”?**

'\0' represents the end of the string. It is also referred as String terminator & Null Character.

String I/O in C programming



**Read & write Strings in C using Printf() and Scanf() functions**

#include <stdio.h>

#include <string.h>

int main()

{

/\* String Declaration\*/

char nickname[20];

printf("Enter your Nick name:");

/\* I am reading the input string and storing it in nickname

\* Array name alone works as a base address of array so

\* we can use nickname instead of &nickname here

\*/

scanf("%s", nickname);

/\*Displaying String\*/

printf("%s",nickname);

return 0;

}

Output:

Enter your Nick name:Negan

Negan

**Note:** %s format specifier is used for strings input/output

Read & Write Strings in C using gets() and puts() functions

#include <stdio.h>

#include <string.h>

int main()

{

/\* String Declaration\*/

char nickname[20];

/\* Console display using puts \*/

puts("Enter your Nick name:");

/\*Input using gets\*/

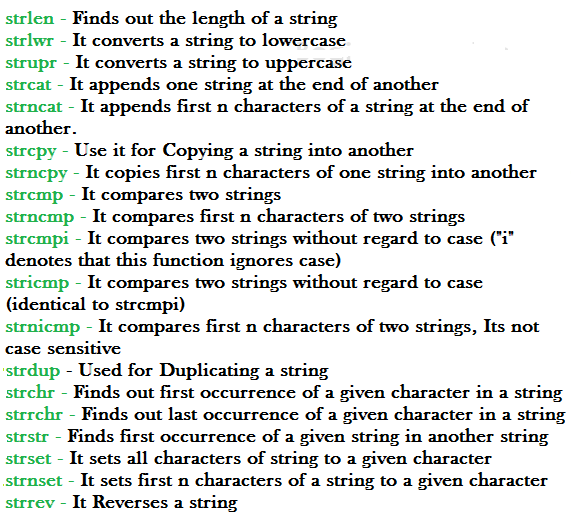
gets(nickname);

puts(nickname);

return 0;

}

**C – String functions**



**C String function – strlen**

Syntax:

size\_t strlen(const char \*str)

**size\_t** represents unsigned short

It returns the length of the string without including end character **(terminating char ‘\0’)**.

**Example of strlen:**

#include <stdio.h>

#include <string.h>

int main()

{

char str1[20] = "BeginnersBook"; printf("Length of string str1: %d", strlen(str1)); return 0;

}

Output:

Length of string str1: 13

**strlen vs sizeof**

strlen returns you the length of the string stored in array, however sizeof returns the total

allocated size assigned to the array. So if I consider the above example again then the following statements would return the below values.

strlen(str1) returned value 13.

sizeof(str1) would return value 20 as the array size is 20 (see the first statement in main function).

**C String function – strnlen**

Syntax:

size\_t strnlen(const char \*str, size\_t maxlen)

size\_t represents unsigned short

It returns length of the string if it is less than the value specified for maxlen (maximum length)

otherwise it returns maxlen value.

**Example of strnlen:**

#include <stdio.h>

#include <string.h>

int main()

{

char str1[20] = "BeginnersBook";

printf("Length of string str1 when maxlen is 30: %d", strnlen(str1, 30)); printf("Length of string str1 when maxlen is 10: %d", strnlen(str1, 10)); return 0;

}

Output:

Length of string str1 when maxlen is 30: 13

Length of string str1 when maxlen is 10: 10

Have you noticed the output of second printf statement, even though the string length was 13 it returned only 10 because the maxlen was 10.

**C String function – strcmp**

int strcmp(const char \*str1, const char \*str2)

It compares the two strings and returns an integer value. If both the strings are same (equal) then this function would return 0 otherwise it may return a negative or positive value based on the comparison.

**If string1 < string2 OR string1 is a substring of string2** then it would result in a negative value. If string1 > string2 then it would return positive value.

**If string1 == string2** then you would get 0(zero) when you use this function for compare strings.

**Example of strcmp:**

#include <stdio.h>

#include <string.h>

int main()

{

char s1[20] = "BeginnersBook";

char s2[20] = "BeginnersBook.COM";

if (strcmp(s1, s2) ==0)

{

printf("string 1 and string 2 are equal");

}else

{

printf("string 1 and 2 are different");

}

return 0;

}

**Output:**

string 1 and 2 are different

**C String function – strncmp**

int strncmp(const char \*str1, const char \*str2, size\_t n)

size\_t is for unassigned short

It compares both the string till n characters or in other words it compares first n characters of both the strings.

**Example of strncmp:**

#include <stdio.h>

#include <string.h>

int main()

{

char s1[20] = "BeginnersBook";

char s2[20] = "BeginnersBook.COM";

/\* below it is comparing first 8 characters of s1 and s2\*/

if (strncmp(s1, s2, 8) ==0)

{

printf("string 1 and string 2 are equal");

}else

{

printf("string 1 and 2 are different");

}

return 0;

}

**Output:**

string1 and string 2 are equal

**C String function – strcat**

char \*strcat(char \*str1, char \*str2)

It concatenates two strings and returns the concatenated string.

**Example of strcat:**

#include <stdio.h>

#include <string.h>

int main()

{

char s1[10] = "Hello"; char s2[10] = "World"; strcat(s1,s2);

printf("Output string after concatenation: %s", s1);

return 0;

}

**Output:**

Output string after concatenation: HelloWorld

**C String function – strncat**

char \*strncat(char \*str1, char \*str2, int n)

It concatenates n characters of str2 to string str1. A terminator char (‘\0’) will always be

appended at the end of the concatenated string.

**Example of strncat:**

#include <stdio.h>

#include <string.h>

int main()

{

char s1[10] = "Hello"; char s2[10] = "World"; strncat(s1,s2, 3);

printf("Concatenation using strncat: %s", s1);

return 0;

}

**Output:**

Concatenation using strncat: HelloWor

**C String function – strcpy**

char \*strcpy( char \*str1, char \*str2)

It copies the string str2 into string str1, including the end character (terminator char ‘\0’).

**Example of strcpy:**

#include <stdio.h>

#include <string.h>

int main()

{

char s1[30] = "string 1";

char s2[30] = "string 2 : I’m gonna copied into s1";

/\* this function has copied s2 into s1\*/

strcpy(s1,s2);

printf("String s1 is: %s", s1);

return 0;

}

**Output:**

String s1 is: string 2: I’m gonna copied into s1

**C String function – strncpy**

char \*strncpy( char \*str1, char \*str2, size\_t n)

size\_t is unassigned short and n is a number.

**Case1:** If length of str2 > n then it just copies first n characters of str2 into str1.

**Case2:** If length of str2 < n then it copies all the characters of str2 into str1 and appends several

terminator chars(‘\0’) to accumulate the length of str1 to make it n.

**Example of strncpy:**

#include <stdio.h>

#include <string.h>

int main()

{

char first[30] = "string 1";

char second[30] = "string 2: I’m using strncpy now";

/\* this function has copied first 10 chars of s2 into s1\*/

strncpy(s1,s2, 12); printf("String s1 is: %s", s1); return 0;

}

**Output:**

String s1 is: string 2: I’m

**C String function – strchr**

char \*strchr(char \*str, int ch)

It searches string str for character ch (you may be wondering that in above definition I have given data type of ch as int, don’t worry I didn’t make any mistake it should be int only. The thing is when we give any character while using strchr then it internally gets converted into integer for better searching.

**Example of strchr:**

#include <stdio.h>

#include <string.h>

int main()

{

char mystr[30] = "I’m an example of function strchr";

printf ("%s", strchr(mystr, 'f'));

return 0;

}

**Output:**

f function strchr

**C String function – Strrchr**

char \*strrchr(char \*str, int ch)

It is similar to the function strchr, the only difference is that it searches the string in reverse

order, now you would have understood why we have extra r in strrchr, yes you guessed it correct, it is for reverse only.

Now let’s take the same above example:

#include <stdio.h>

#include <string.h>

int main()

{

char mystr[30] = "I’m an example of function strchr";

printf ("%s", strrchr(mystr, 'f'));

return 0;

}

**Output:**

function strchr

**Why output is different than strchr?** It is because it started searching from the end of the

string and found the first ‘f’ in function instead of ‘of’.

**C String function – strstr**

char \*strstr(char \*str, char \*srch\_term)

It is similar to strchr, except that it searches for string srch\_term instead of a single char.

**Example of strstr:**

#include <stdio.h>

#include <string.h>

int main()

{

char inputstr[70] = "String Function in C at BeginnersBook.COM";

printf ("Output string is: %s", strstr(inputstr, 'Begi'));

return 0;

}

**Output:**

Output string is: BeginnersBook.COM

**Some C Programs on String**

1. Write a program in C to find the length of a string without using library function.

#include <stdio.h>

#include <stdlib.h>

void main()

{

char str[100]; /\* Declares a string of size 100 \*/

int l= 0;

printf("\n\nFind the length of a string :\n"); printf("---------------------------------\n"); printf("Input the string : ");

fgets(str, sizeof str, stdin);

while(str[l]!='\0')

{

l++;

}

printf("Length of the string is : %d\n\n", l-1);

}

Output:

Find the length of a string :

---------------------------------

Input the string : w3resource.com

Length of the string is : 15

Flowchart:

2. Write a program in C to separate the individual characters from a string.

#include <stdio.h>

#include <stdlib.h>

void main()

{

char str[100]; /\* Declares a string of size 100 \*/

int l= 0;

printf("\n\nSeparate the individual characters from a string :\n");

printf("------------------------------------------------------\n");

printf("Input the string : ");

fgets(str, sizeof str, stdin);

printf("The characters of the string are : \n");

while(str[l]!='\0')

{

printf("%c ", str[l]);

l++;

}

printf("\n");

}

Sample Output:

Separate the individual characters from a string :

------------------------------------------------------ Input the string : w3resource.com

The characters of the string are :

w 3 r e s o u r c e . c o m

3. Write a program in C to print individual characters of string in reverse order.

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

void main()

{

char str[100]; /\* Declares a string of size 100 \*/

int l,i;

printf("\n\nPrint individual characters of string in reverse order :\n");

printf("------------------------------------------------------\n");

printf("Input the string : ");

fgets(str, sizeof str, stdin);

l=strlen(str);

printf("The characters of the string in reverse are : \n");

for(i=l;i>=0;i--)

{

printf("%c ", str[i]);

}

printf("\n");

}

***The program can also be written as below:***

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

void main()

{

char str[100]; /\* Declares a string of size 100 \*/

int l=0;

printf("\n\nPrint individual characters of string in reverse order :\n");

printf("------------------------------------------------------\n");

printf("Input the string : ");

fgets(str, sizeof str, stdin);

l=strlen(str);

printf("The characters of the string in reverse are : \n");

for(str[l]='\0';l>=0;l--)

{

printf("%c ", str[l]);

}

printf("\n");

}

**Sample Output:**

Print individual characters of string in reverse order :

----------------------------------------------------------- Input the string : w3resource.com

The characters of the string in reverse are :

m o c . e c r u o s e r 3 w

4. Write a program in C to count the total number of words in a string.

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define str\_size 100 //Declare the maximum size of the string void main()

{

char str[str\_size];

int i, wrd;

printf("\n\nCount the total number of words in a string :\n"); printf("------------------------------------------------------\n"); printf("Input the string : ");

fgets(str, sizeof str, stdin);

i = 0;

wrd = 1;

/\* loop till end of string \*/

while(str[i]!='\0')

{

/\* check whether the current character is white space or new line or tab character\*/

if(str[i]==' ' || str[i]=='\n' || str[i]=='\t')

{

wrd++;

}

i++;

}

printf("Total number of words in the string is : %d\n", wrd-1);

}

**Sample Output:**

Count the total number of words in a string :

------------------------------------------------------ Input the string : This is w3resource.com Total number of words in the string is : 3

5. Write a program in C to convert a string to lowercase.

#include<stdio.h>

#include<ctype.h>

int main()

{

int ctr=0;

char str\_char;

char str[100];

printf("\n Convert a string to lowercase :\n");

printf("----------------------------------"); printf("\n Input a string in UPPERCASE : "); fgets(str, sizeof str, stdin);

printf(" Here is the above string in lowercase :\n ");

while (str[ctr])

{

str\_char=str[ctr];

putchar (tolower(str\_char));

ctr++;

}

return 0;

}

**Sample Output:**

Convert a string to lowercase :

----------------------------------

Input a string in UPPERCASE : THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG. Here is the above string in lowercase :

the quick brown fox jumps over the lazy dog.