**STRING**

In C programming, a string is a sequence of characters terminated with a null character \0. For example:

char c[] = "c string";

When the compiler encounters a sequence of characters enclosed in the double quotation marks, it appends a null character \0 at the end by default.

Memory diagram of strings in C programming

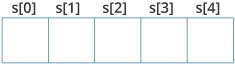
Memory Diagram

**How to declare a string?**

Here's how you can declare strings:

char s[5];

String Declaration in C



Here, we have declared a string of 5 characters.

**How to initialize strings?**

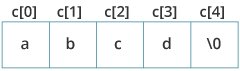
You can initialize strings in a number of ways.

char c[] = "abcd";

char c[50] = "abcd";

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

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



String Initialization in C

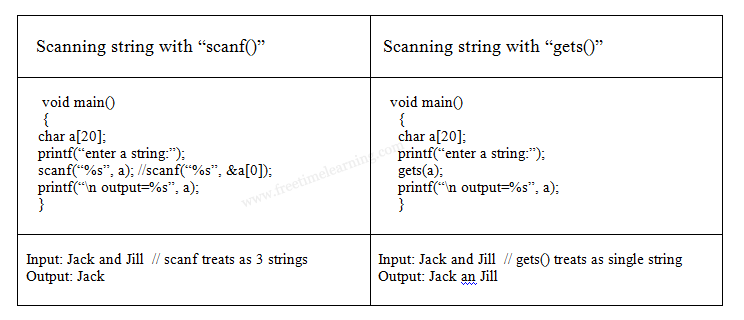
String I/O functions In C Language

The following are the input and output functions of strings in c

**Input functions: scanf(), gets()**

**Output functions: printf(), puts()**

The **scanf()**and **printf()** are generic **i/o** functions that they support all built-in data types such as int, float, long, double, strings,..etc. But **gets()**and **puts()** are specialized to scan and print only string data. There is a little difference between **scanf()** and **gets()**, while reading string from keyboard, the **scanf()**accepts character by character from keyboard until either a new line (‘**\n**’) or blank space is found, which ever comes earlier. Whereas “**gets()**” accepts until a newline is found. That is it accepts white spaces & tab also, these input functions append a null character at end of string, the formatted string %s is used in **printf()**and **scanf()**.for example :



The **scanf()** function consider the jack and Jill as 3 strings, whereas **gets()** considers as single string. In case to scan total string using **scanf()**, then it should be **scanf(“%s%s%s”, a,b,c);** here **a,b,c** are three arrays.

The **printf()** and **puts()** is work in similar way. All **I/O** functions take first byte address (base address of array) as argument and prints the given string using pointer.

  Program : The following program is an example of string I/O functions.

#include<stdio.h>

#include<string.h>

int main()

{

char name[30];

printf(“Enter name: “);

gets(name); //Function to read string from user.

printf(“Name: “);

puts(name); //Function to display string.

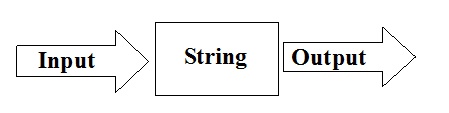
return 0;

}

**Output :**

**Enter name: Free Time Learning  
Name: Free Time Learning**

## C Program Gets() And Puts() Function



Both the functions are used to in the input and output operation of the Strings

The gets() functions are used to read string input from the keyboard and puts() function displays it.

These functions are declared in the ***stdio.h*** header file.

### ****gets() function****

The gets() function is similar to scanf() function but it allows entering some characters by the user in string format with the double quotation. and stored in a character array format.

**Declaration**

Syntax of gets() function

char[] gets(char[]);

**Program 1**

let’s see an example to read a string using gets() function and print it on the screen using printf() function

#include <stdio.h>

#include <stdlib.h>

int main()

{

char ch[30];

printf("Enter the string: ");

gets(ch); //reading string using gets()

printf("you entered string here\n");

printf(ch);//display out put on the screen using printf() function

**return** 0;

}

**When the above code is executed, it produces the following results**

Enter the string: code4coding

you entered string here

code4coding

### ****Puts() function****

The **puts()** function is similar to printf() function. but **puts()** function is used to display only the string after reading by gets() function entered by user(gets similar to scanf() function)

**Declaration**

Syntax of puts() function

int puts(char[])

**Program 2**

let’s see an example to read a string using gets() function and print it on the screen using puts() function

#include <stdio.h>

#include <string.h>

int main()

{

char sentence[50];

printf("Enter your sentence\n");

gets(sentence);//read input from entered by the user

printf("your sentance here\n\n");

puts(sentence); //display the sentence

getch();

**return** 0;

}

**When the above code is executed, it produces the following results**

Enter your sentence

Code4coding is a programming portal

your sentence here

Code4coding is a programming portal

## Read String from the user

You can use the scanf() function to read a string.

The scanf() function reads the sequence of characters until it encounters [whitespace](https://stackoverflow.com/questions/30033582/what-is-the-symbol-for-whitespace-in-c) (space, newline, tab, etc.).

### Difference between scanf() and gets()

The main difference between these two functions is that scanf() stops reading characters when it encounters a space, but gets() reads space as character too.

If you enter name as **Study Tonight** using scanf() it will only read and store **Study** and will leave the part after space. But gets() function will read it completely.

**Note:** The gets() function can also be to take input from the user. However, it is removed from the C standard.  
  
It's because gets() allows you to input any length of characters. Hence, there might be a buffer overflow.

### Example 1: scanf() to read a string

#include <stdio.h>

int main()

{

char name[20];

printf("Enter name: ");

scanf("%s", name);

printf("Your name is %s.", name);

return 0;

}

**Output**

Enter name: Dennis Ritchie

Your name is Dennis.

Even though Dennis Ritchie was entered in the above program, only "Dennis" was stored in the name string. It's because there was a space after Dennis.

### How to read a line of text?

You can use the fgets() function to read a line of string. And, you can use puts() to display the string.

### Example 2: fgets() and puts()

#include <stdio.h>

int main()

{

char name[30];

printf("Enter name: ");

fgets(name, sizeof(name), stdin); // read string

printf("Name: ");

puts(name); // display string

return 0;

}

**Output**

Enter name: Tom Hanks

Name: Tom Hanks

Here, we have used fgets() function to read a string from the user.

fgets(name, sizeof(name), stdlin); // read string

The sizeof(name) results to 30. Hence, we can take a maximum of 30 characters as input which is the size of the name string.

To print the string, we have used puts(name);.

## getchar() & putchar() functions

The getchar() function reads a character from the terminal and returns it as an integer. This function reads only single character at a time. You can use this method in a [loop](https://www.studytonight.com/c/loops-in-c.php) in case you want to read more than one character. The putchar() function displays the character passed to it on the screen and returns the same character. This function too displays only a single character at a time. In case you want to display more than one characters, use putchar() method in a loop.

NOTE:

putchar() function is a file handling function in C programming language which is used to write a character on standard output/screen. getchar() function is used to get/read a character from keyboard input.

|  |  |
| --- | --- |
| **File operation** | **Declaration & Description** |
| putchar() | Declaration: int putchar(int char)  putchar() function is used to write a character on standard output/screen. In a C program, we can use putchar function as below.putchar(char);  where, char is a character variable/value. |
| getchar() | Declaration: int getchar(void)  getchar() function is used to get/read a character from keyboard input. In a C program, we can use getchar function as below. **getchar**(char);  where, char is a character variable/value. |

# ****EXAMPLE PROGRAM FOR PUTCHAR(), GETCHAR() FUNCTIONS IN C PROGRAMMING LANGUAGe****



# 

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | #include <stdio.h>  #include <ctype.h>  int main()  {     char c;     printf("Enter some character. Enter $ to exit...\n");     while (c != '$')     {        c = getchar();        printf("\n Entered character is: ");        putchar(c);        printf("\n")     }     return 0;  } |

#### ****OUTPUT:****

|  |
| --- |
| Enter some character. Enter $ to exit… A Entered character is: A B Entered character is: B $  Entered character is: $ |

# C String Functions(string handling functions)

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

|  |  |  |
| --- | --- | --- |
| **No.** | **Function** | **Description** |
| 1) | [strlen(string\_name)](https://www.javatpoint.com/c-strlen) | returns the length of string name. |
| 2) | [strcpy(destination, source)](https://www.javatpoint.com/c-strcpy) | copies the contents of source string to destination string. |
| 3) | [strcat(first\_string, second\_string)](https://www.javatpoint.com/c-strcat) | concats or joins first string with second string. The result of the string is stored in first string. |
| 4) | [strcmp(first\_string, second\_string)](https://www.javatpoint.com/c-strcmp) | compares the first string with second string. If both strings are same, it returns 0. |
| 5) | [strrev(string)](https://www.javatpoint.com/c-strrev) | returns reverse string. |
| 6) | [strlwr(string)](https://www.javatpoint.com/c-strlwr) | returns string characters in lowercase. |
| 7) | [strupr(string)](https://www.javatpoint.com/c-strupr) | returns string characters in uppercase. |

# C String Length: strlen() function

The strlen() function returns the length of the given string. It doesn't count null character '\0'.

**strlen( ) Function :**

**strlen( )**function is used to find the length of a character string.

**Example:**       **int  n;**

**char st[20] = “Bangalore”;**

**n = strlen(st);**

**•**This will return the length of the string 9 which is assigned to an integer variable n.

**•**Note that the null character “**\0**‟ available at the end of a string is not counted.

example

1. #include<stdio.h>
2. #include <string.h>
3. **int** main()
4. {
5. **char** ch[20]={'j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't', '\0'};
6. printf("Length of string is: %d",strlen(ch));
7. **return** 0;
8. }

Output:

Length of string is: 10

# C Copy String: strcpy()

The strcpy(destination, source) function copies the source string in destination.

**strcpy( ) Function :**

**strcpy( )** function copies contents of one string into another string. Syntax for strcpy function is given below.

**Syntax:  char \* strcpy (char \* destination, const char \* source);**

**Example:**

**strcpy ( str1, str2) – It copies contents of str2 into str1.**

**strcpy ( str2, str1) – It copies contents of str1 into str2.**

If destination string length is less than source string, entire source string value won’t be copied into destination string.

For example, consider destination string length is 20 and source string length is 30. Then, only 20 characters from source string will be copied into destination string and remaining 10 characters won’t be copied and will be truncated.

**Example :   char  city[15];**

**strcpy(city, “BANGALORE”) ;**

 This will assign the string “BANGALORE” to the character variable city.

example

1. #include<stdio.h>
2. #include <string.h>
3. **int** main()
4. {
5. **char** ch[20]={'j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't', '\0'};
6. **char** ch2[20];
7. strcpy(ch2,ch);
8. printf("Value of second string is: %s",ch2);
9. **return** 0;
10. }

Output:

Value of second string is: javatpoint

# C String Concatenation: strcat()

The strcat(first\_string, second\_string) function concatenates two strings and result is returned to first\_string.

**strcat( ) Function :**

**strcat( )**function in C language concatenates two given strings. It concatenates source string at the end of destination string. Syntax for **strcat( )**function is given below.

**Syntax :  char \* strcat ( char \* destination, const char \* source );**

**Example :**

**strcat ( str2, str1 ); - str1 is concatenated at the end of str2.**

**strcat ( str1, str2 ); - str2 is concatenated at the end of str1.**

**•** As you know, each string in C is ended up with null character (‘**\0**′).

**•**In **strcat( )** operation, null character of destination string is overwritten by source string’s first character and null character is added at the end of new destination string which is created after**strcat( )** operation.

Program : The following program is an example of strcat() function

#include <stdio.h>

#include <string.h>

int main( )

{

char source[ ] = “ ftl” ;

char target[ ]= “ welcome to” ;

printf (“\n Source string = %s”, source ) ;

printf ( “\n Target string = %s”, target ) ;

strcat ( target, source ) ;

printf ( “\n Target string after strcat( ) = %s”, target ) ;

}

**Output :**

**Source string = ftl  
Target string = welcome to  
Target string after strcat() = welcome to ftl**

Example 2

1. #include<stdio.h>
2. #include <string.h>
3. **int** main()
4. {
5. **char** ch[10]={'h', 'e', 'l', 'l', 'o', '\0'};
6. **char** ch2[10]={'c', '\0'};
7. strcat(ch,ch2);
8. printf("Value of first string is: %s",ch);
9. **return** 0;
10. }

Output:

Value of first string is: helloc

**Strncat() function :**

**strncat( )** function in C language concatenates (appends) portion of one string at the end of another string.

**Syntax : char \* strncat ( char \* destination, const char \* source, size\_t num );**

**Example :  
strncat ( str2, str1, 3 ); – First 3 characters of str1 is concatenated at the end of str2.  
strncat ( str1, str2, 3 ); - First 3 characters of str2 is concatenated at the end of str1.**

As you know, each string in C is ended up with null character (‘**\0**′).

In **strncat( )** operation, null character of destination string is overwritten by source string’s first character and null character is added at the end of new destination string which is created after **strncat( )** operation.

Program : The following program is an example of **strncat()** function

#include <stdio.h>

#include <string.h>

int main( )

{

char source[ ] =” ftl” ;

char target[ ]= “welcome to” ;

printf ( “\n Source string = %s”, source ) ;

printf ( “\n Target string = %s”, target ) ;

strncat ( target, source, 3 ) ;

printf ( "\n Target string after strncat( ) = %s”, target ) ;

}

**Output :**

**Source string = ftl  
Target string = welcome to  
Target string after strncat()= welcome to ft**

# C Compare String: strcmp()

The strcmp(first\_string, second\_string) function compares two string and returns 0 if both strings are equal.

**strcmp( ) Function :**

**strcmp( )** function in C compares two given strings and returns zero if they are same. If length of string1 < string2, it returns < 0 value. If length of string1 > string2, it returns > 0 value.

**Syntax : int strcmp ( const char \* str1, const char \* str2 );**

**strcmp( )** function is case sensitive. i.e., “A” and “a” are treated as different characters.

**Example :  
char city[20] = “Madras”; d=100  
char town[20] = “Mangalore”; n=110  
strcmp(city, town);**

This will return an integer value “-10‟ which is the difference in the ASCII values of the first mismatching letters “D‟ and “N‟.

\* Note that the integer value obtained as the difference may be assigned to an integer variable as follows:

**int n;  
n = strcmp(city, town);**

Here, we are using gets() function which reads string from the console.

1. #include<stdio.h>
2. #include <string.h>
3. **int** main(){
4. **char** str1[20],str2[20];
5. printf("Enter 1st string: ");
6. gets(str1);//reads string from console
7. printf("Enter 2nd string: ");
8. gets(str2);
9. **if**(strcmp(str1,str2)==0)
10. printf("Strings are equal");
11. **else**
12. printf("Strings are not equal");
13. **return** 0;
14. }

Output:

Enter 1st string: hello

Enter 2nd string: hello

Strings are equal

# C Reverse String: strrev()

The strrev(string) function returns reverse of the given string. Let's see a simple example of strrev() function.

1. #include<stdio.h>
2. #include <string.h>
3. **int** main()
4. {
5. **char** str[20];
6. printf("Enter string: ");
7. gets(str);//reads string from console
8. printf("String is: %s",str);
9. printf("\nReverse String is: %s",strrev(str));
10. **return** 0;
11. }

Output:

Enter string: javatpoint

String is: javatpoint

Reverse String is: tnioptavaj

# C String Lowercase: strlwr()

The strlwr(string) function returns string characters in lowercase. Let's see a simple example of strlwr() function.

1. #include<stdio.h>
2. #include <string.h>
3. **int** main()
4. {
5. **char** str[20];
6. printf("Enter string: ");
7. gets(str);//reads string from console
8. printf("String is: %s",str);
9. printf("\nLower String is: %s",strlwr(str));
10. **return** 0;
11. }

 Output:

Enter string: JAVATpoint

String is: JAVATpoint

Lower String is: javatpoint

# C String uppercase: strupr()

The strupr(string) function returns string characters in uppercase. Let's see a simple example of strupr() function

1. #include<stdio.h>
2. #include <string.h>
3. **int** main(){
4. **char** str[20];
5. printf("Enter string: ");
6. gets(str);//reads string from console
7. printf("String is: %s",str);
8. printf("\nUpper String is: %s",strupr(str));
9. **return** 0;
10. }

Output:

Enter string: javatpoint

String is: javatpoint

Upper String is: JAVATPOINT