**VISION INSTITUTE OF MANAGEMENT**

**C PROGRAMMING**

**BCA 1st YEAR/2nd SEM**

**UNIT-3(STRING)**

**String-**

Strings are actually 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.**

**OR**

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.

**Declaration of strings**:

 Declaring a string is as simple as declaring a one dimensional array. Below is the basic syntax for declaring a string:

 char str\_name[size];

In the above syntax str\_name is any name given to the string variable and size is used define the length of the string, i.e the number of characters strings will store. Please keep in mind that there is an extra terminating character which is the Null character (‘\0’) used to indicate termination of string which differs strings from normal character arrays.

**For Example:**

 

**Initializing a String**:

 A string can be initialized in different ways. We will explain this with the help of an example. Below is an example to declare a string with name as **c** and initialize it with “abcd”-

char c[] = "abcd";

char c[50] = "abcd";

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

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



Let's take another example:

char c[5] = "abcde";

Here, we are trying to assign 6 characters (the last character is '\0') to a char array having 5 characters. This is bad and you should never do this.

**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.).

### 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 "Ritchie" 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);.

**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.

**Standard Library function:**

You need to often manipulate [strings](https://www.programiz.com/c-programming/c-strings) according to the need of a problem. Most, if not all, of the time string manipulation can be done manually but, this makes programming complex and large.

To solve this, C supports a large number of string handling functions in the [standard library](https://www.programiz.com/c-programming/library-function) "string.h"

**Few commonly used string handling functions are discussed below:**

| Function |  | Work of Function |
| --- | --- | --- |
| [strlen()](https://www.programiz.com/c-programming/library-function/strlen) |  | computes string's length |
| [strcpy()](https://www.programiz.com/c-programming/library-function/strcpy) |  | copies a string to another |
| [strcat()](https://www.programiz.com/c-programming/library-function/strcat) |  | concatenates(joins) two strings |
| [strcmp()](https://www.programiz.com/c-programming/library-function/strcmp) |  | compares two strings |
| strlwr() |  | converts string to lowercase |
| strupr() |  | converts string to uppercase |
|  |

Strings handling functions are defined under "string.h" header file.

#include <string.h>

**Note:** You have to include the code below to run string handling functions.

# C strlen()-

#### The strlen() function calculates the length of a given string.

The strlen() function takes a string as an argument and returns its length. The returned value is of type long int.

It is defined in the [<string.h>](https://www.programiz.com/c-programming/library-function/string.h) header file.

## Example: C strlen() function:

#include <stdio.h>

#include <string.h>

int main()

{

 char a[20]="Program";

 char b[20]={'P','r','o','g','r','a','m','\0'};

 printf("Length of string a = %ld \n",strlen(a));

 printf("Length of string b = %ld \n",strlen(b));

 return 0;

}

**Output:**

Length of string a = 7

Length of string b = 7

Note that the strlen() function doesn't count the null character \0 while calculating the length.

# C strcpy()-

#### The strcpy() function copies the string to the another character array.

## strcpy() Function Syntax:

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

The strcpy() function copies the string pointed by source (including the null character) to the character array destination.

The function also returns the copied array.

The strcpy() function is defined in the string.h header file.

### Example: C strcpy():

#include <stdio.h>

#include <string.h>

int main()

{

 char str1[10]= "awesome";

 char str2[10];

 char str3[10];

 strcpy(str2, str1);

 strcpy(str3, "well");

 puts(str2);

 puts(str3);

 return 0;

}

**Output:**

awesome

well

It is important to note that the destination array should be large enough to copy the array. Otherwise, it may result in undefined behavior.

# C strcat()-

#### The function strcat() concatenates two strings.

In C programming, strcat() concatenates (joins) two strings.

The strcat() function is defined in [<string.h>](https://www.programiz.com/c-programming/library-function/string.h) header file.

## C strcat() syntax:

char \*strcat(char \*dest, const char \*src)

It takes two arguments, i.e, two strings or character arrays, and stores the resultant concatenated string in the first string specified in the argument.

The pointer to the resultant string is passed as a return value.

### Example: C strcat() function:

#include <stdio.h>

#include <string.h>

int main()

{

 char str1[] = "This is ", str2[] = "programiz.com";

 //concatenates str1 and str2 and resultant string is stored in str1.

 strcat(str1,str2);

 puts(str1);

 puts(str2);

 return 0;

}

**Output:**

This is programiz.com

programiz.com

# C strcmp()-

#### The strcmp() function compares two strings and returns 0 if both strings are identical.

## C strcmp() Prototype:

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

The strcmp() function takes two strings and returns an integer.

The strcmp() compares two strings character by character.

If the first character of two strings is equal, the next character of two strings are compared. This continues until the corresponding characters of two strings are different or a null character '\0' is reached.

It is defined in the string.h header file.

### Example: C strcmp() function:

#include <stdio.h>

#include <string.h>

int main()

{

 char str1[] = "abcd", str2[] = "abCd", str3[] = "abcd";

 int result;

 // comparing strings str1 and str2

 result = strcmp(str1, str2);

 printf("strcmp(str1, str2) = %d\n", result);

 // comparing strings str1 and str3

 result = strcmp(str1, str3);

 printf("strcmp(str1, str3) = %d\n", result);

 return 0;

}

**Output:**

strcmp(str1, str2) = 32

strcmp(str1, str3) = 0

The first unmatched character between string str1 and str2 is third character. The ASCII value of 'c' is 99 and the ASCII value of 'C' is 67. Hence, when strings str1 and str2 are compared, the return value is 32.

When strings str1 and str3 are compared, the result is 0 because both strings are identical.

### gets() and puts():

Functions gets() and puts() are two string functions to take string input from the user and display it respectively as mentioned in the [previous chapter](https://www.programiz.com/c-programming/c-strings).

#include<stdio.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;

}

**Note:**Though, gets() and puts() function handle strings, both these functions are defined in "stdio.h" header file.

## Passing Strings to Functions:

Strings can be passed to a function in a similar way as arrays. Learn more about [passing arrays to a function](https://www.programiz.com/c-programming/c-arrays-functions).

### Example: Passing string to a Function

#include <stdio.h>

void displayString(char str[]);

int main()

{

 char str[50];

 printf("Enter string: ");

 fgets(str, sizeof(str), stdin);

 displayString(str); // Passing string to a function.

 return 0;

}

void displayString(char str[])

{

 printf("String Output: ");

 puts(str);

}

## Strings and Pointers:

Similar like arrays, string names are "decayed" to pointers. Hence, you can use pointers to manipulate elements of the string.

### Example 4: Strings and Pointers:

#include <stdio.h>

int main(void) {

 char name[] = "Harry Potter";

 printf("%c", \*name); // Output: H

 printf("%c", \*(name+1)); // Output: a

 printf("%c", \*(name+7)); // Output: o

 char \*namePtr;

 namePtr = name;

 printf("%c", \*namePtr); // Output: H

 printf("%c", \*(namePtr+1)); // Output: a

 printf("%c", \*(namePtr+7)); // Output: o

}

**Examples of Strings:**

**Example 1. Find the Frequency of a Character-**

#include <stdio.h>

int main() {

 char str[1000], ch;

 int count = 0;

 printf("Enter a string: ");

 fgets(str, sizeof(str), stdin);

 printf("Enter a character to find its frequency: ");

 scanf("%c", &ch);

 for (int i = 0; str[i] != '\0'; ++i) {

 if (ch == str[i])

 ++count;

 }

 printf("Frequency of %c = %d", ch, count);

 return 0;

}

**Output-**

Enter a string: This website is awesome.

Enter a character to find its frequency: e

Frequency of e = 4

**Example 2. Find the number of vowels, consonants, digits and white spaces-**

#include <stdio.h>

int main()

{

 char line[150];

 int vowels, consonant, digit, space;

 vowels = consonant = digit = space = 0;

 printf("Enter a line of string: ");

 fgets(line, sizeof(line), stdin);

 for (int i = 0; line[i] != '\0'; ++i)

 {

 if (line[i] == 'a' || line[i] == 'e' || line[i] == 'i' ||

 line[i] == 'o' || line[i] == 'u' || line[i] == 'A' ||

 line[i] == 'E' || line[i] == 'I' || line[i] == 'O' ||

 line[i] == 'U')

 {

 ++vowels;

 }

 else if ((line[i] >= 'a' && line[i] <= 'z') || (line[i] >= 'A' && line[i] <= 'Z'))

 {

 ++consonant;

 }

 else if (line[i] >= '0' && line[i] <= '9')

 {

 ++digit;

 }

 else if (line[i] == ' ')

 {

 ++space;

 }

 }

 printf("Vowels: %d", vowels);

 printf("\nConsonants: %d", consonant);

 printf("\nDigits: %d", digit);

 printf("\nWhite spaces: %d", space);

 return 0;

}

**Output:**

Enter a line of string: adfslkj34 34lkj343 34lk

Vowels: 1

Consonants: 11

Digits: 9

White spaces: 2