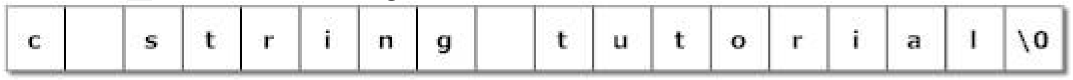
# Strings in C

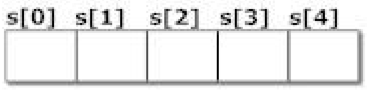
Array of characters is called a string. A string is terminated by a null character /0. For example:
"c string tutorial"
Here, "c string tutorial" is a string. When, compiler encounter strings, it appends a null character /0 at the end of string.



## Declaration of strings

Before we actually work with strings, we need to declare them first.
Strings are declared in a similar manner as arrays. Only difference is that, strings are of char type.
Using arrays

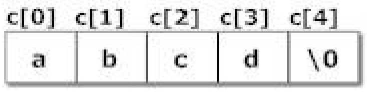
```
char s[5];
```



## Initialization of strings

In C, string can be initialized in a number of different ways.
For convenience and ease, both initialization and declaration are done in the same step.
Using arrays

```
char c[] = "abcd";
    OR,
char c[50] = "abcd";
    OR,
char c[] = {'a', 'b', 'c', 'd', '\0'};
    OR,
char c[5] = {'a', 'b', 'c', 'd', '\0'};
```



The given string is initialized and stored in the form of arrays as above.

## Reading Strings from user

You can use the scanf() function to read a string like any other data types.
However, the scanf() function only takes the first entered word. The function terminates when it encounters a white space (or just space).
Reading words from user

```
char c[20];
scanf("%s", c);
```

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

**Write a C program to illustrate how to read string from terminal.**

```
#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: BCA SECOND SEM
Your name is ?.
```

Here, program ignores SECOND SEM because, scanf() function takes only a single string before the white space, i.e. BCA.

Reading a line of text

An approach to reading a full line of text is to read and store each character one by one.

Example #2: Using getchar() to read a line of text

**1. C program to read line of text character by character.**

```c
#include <stdio.h>
int main()
{
    char name[30], ch;
    int i = 0;
    printf("Enter name: ");
    while(ch != '\n')    // terminates if user hit enter
    {
        ch = getchar();
        name[i] = ch;
        i++;
    }
    name[i] = '\0';// inserting null character at end
    printf("Name: %s", name);
    return 0;
}
```

In the program above, using the function getchar(), ch gets a single character from the user each time.

This process is repeated until the user enters return (enter key). Finally, the null character is inserted at the end to make it a string.

This process to take string is tedious.

Example #3: Using standard library function to read a line of text

**2. C program to read line of text using gets() and puts()**

To make life easier, there are predefined functions gets() and puts in C language to read and display string respectively.

```c
#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;
}
```

Both programs have the same output below:

```
Output
Enter name: TISTA BHADURI
Name: TISTA BHADURI
```

Passing Strings to Functions

Strings are just char arrays. So, they can be passed to a function in a similar manner as arrays.

```
#include <stdio.h>
void displayString(char str[]);

int main()
{
    char str[50];
    printf("Enter string: ");
    gets(str);
    displayString(str);  // Passing string c to function.
    return 0;
}
void displayString(char str[]){
    printf("String Output: ");
    puts(str);
}
```

Find the Frequency of Characters in String

```
#include <stdio.h>

int main()
{
   char str[1000], ch;
   int i, frequency = 0;

   printf("Enter a string: ");
   gets(str);

 printf("Enter a character to find the frequency: ");
   scanf("%c",&ch);

   for(i = 0; str[i] != '\0'; ++i)
   {
       if(ch == str[i])
           ++frequency;
   }

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

   return 0;
}
```

**Output**

```
Enter a string: You Guys are awesome.
Enter a character to find the frequency: e
Frequency of e = 3
```

Program to count vowels, consonants etc.

```
#include <stdio.h>

int main()
{
   char line[150];
   int i, vowels, consonants, digits, spaces;

   vowels =  consonants = digits = spaces = 0;

   printf("Enter a line of string: ");
   scanf("%[^\n]", line);

   for(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'))
      {
         ++consonants;
      }
      else if(line[i]>='0' && line[i]<='9')
      {z
         ++digits;
      }
      else if (line[i]==' ')
      {
         ++spaces;
      }
   }

   printf("Vowels: %d",vowels);
   printf("\nConsonants: %d",consonants);
   printf("\nDigits: %d",digits);
   printf("\nWhite spaces: %d", spaces);

   return 0;
}
```

**Output**

```
Enter a line of string: adfslkj34 34lkj343 34lk
Vowels: 1
Consonants: 11
Digits: 9
White spaces: 2
```

Remove Characters in String Except Alphabets

```c
#include<stdio.h>

int main()
{
   char line[150];
   int i, j;
   printf("Enter a string: ");
   gets(line);

   for(i = 0; line[i] != '\0'; ++i)
   {
      while (!( (line[i] >= 'a' && line[i] <= 'z') || (line[i] >= 'A' && line[i] <= 'Z') || line[i] == '\0') )
      {
         for(j = i; line[j] != '\0'; ++j)
         {
            line[j] = line[j+1];
         }
         line[j] = '\0';
      }
   }
   printf("Output String: ");
   puts(line);
   return 0;
}
```

**Output**

```
Enter a string: m2'c-a@first84sem./
Output String: bcasecondsem
```

This program takes a string from the user and stored in the variable line.
The, within the for loop, each character in the string is checked if it's an alphabet or not.
If any character inside a string is not a alphabet, all characters after it including the null character is shifted by 1 position to the left.
Assignments:
Read a Line From a File and Display it
Find the Frequency of Characters in a String
Check Whether a Character is an Alphabet or not
count the number of vowels, consonants and so on
Find the Length of a String

Copy String Without Using strcpy()
Check Whether a Character is Vowel or Consonant
Display Characters from A to Z Using Loop

String Functions

## Compare String: strcmp()

The strcmp(first_string, second_string) function compares two string and returns 0 if both strings are equal.

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

```
#include<stdio.h>
#include <string.h>
int main(){
  char str1[20],str2[20];
  printf("Enter 1st string: ");    // Hello
  gets(str1);//reads string from console
  printf("Enter 2nd string: ");    // hello
  gets(str2);
  if(strcmp(str1,str2)==0)
      printf("Strings are equal");
  else
      printf("Strings are not equal");
 return 0;
}
```

Output:

```
Enter 1st string: hello
Enter 2nd string:  hello
Strings are equal
```

## String Length: strlen() function

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

```
#include<stdio.h>
#include <string.h>
int main(){
char ch[20]= {'b', 'c', 'a', 's', 'e', 'c', 'o', 'n', 'd', 's',  'e', 'm', '\0'};
  printf("Length of string is: %d", strlen(ch));
 return 0;
}
```

Output:

```
Length of string is: ?
```

## Copy String: strcpy()

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

```
#include<stdio.h>
#include <string.h>
int main(){
 char ch[20]= {'b', 'c', 'a', 's', 'e', 'c', 'o', 'n', 'd', 's',  'e', 'm', '\0'};
```

```
   char ch2[20];
   strcpy(ch2,ch);
   printf("Value of second string is: %s",ch2);
 return 0;
}
```

Output:

```
Value of second string is: ?
```

## String Concatenation: strcat()

The strcat(first_string, second_string) function concatenates two strings and result is returned to first_string.

```
#include<stdio.h>
#include <string.h>
int main(){
  char ch[10]={'h', 'e', 'l', 'l', 'o', '\0'};
   char ch2[10]={'a', 'a', 's', 'u','\0'};
   strcat(ch,ch2);
   printf("Value of first string is: %s",ch);
 return 0;
}
```

Output:

```
Value of first string is: helloaasu
```

## Reverse String: strrev()

The strrev(string) function returns reverse of the given string.

```
#include<stdio.h>
#include <string.h>
int main(){
  char str[20];
  printf("Enter string: ");
  gets(str);//reads string from console
  printf("String is: %s",str);
  printf("\nReverse String is: %s",strrev(str));
 return 0;
}
```

Output:

```
Enter string: youguys
String is: youguys
Reverse String is: syuguoy
```

## String Lowercase: strlwr()

The strlwr(string) function returns string characters in lowercase.

```
#include<stdio.h>
#include <string.h>
int main(){
  char str[20];
  printf("Enter string: ");
  gets(str);//reads string from console
  printf("String is: %s",str);
  printf("\nLower String is: %s",strlwr(str));
 return 0;
}
```
Output:

```
Enter string: MCAstudent
String is: MCAstudent
Lower String is: mcastudent
```

String Uppercase: strupr()

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

```
#include<stdio.h>
#include <string.h>
int main(){
  char str[20];
  printf("Enter string: ");
  gets(str);//reads string from console
  printf("String is: %s",str);
  printf("\nUpper String is: %s",strupr(str));
 return 0;
}
```
Output:

```
Enter string: mcastudent
String is: mcastudent
Upper String is: MCASTUDENT
```

C String strstr()

The strstr() function returns pointer to the first occurrence of the matched string in the given string. It is used to return substring from first match till the last character.

**Syntax:**

```
char *strstr(const char *string, const char *match)
```

String strstr() parameters

**string:** It represents the full string from where substring will be searched.

**match:** It represents the substring to be searched in the full string.

String strstr() example

```
#include<stdio.h>
#include <string.h>
int main(){
  char str[100]="this is javatpoint with c and java";
  char *sub;
  sub=strstr(str,"java");
  printf("\nSubstring is: %s",sub);
 return 0;
}
```
Output:
```
javatpoint with c and java
```

## String Test 1

---

1) Which of the function is more appropriate for reading a multi-word string?
puts()
gets()
printf()
scanf()
2) Which library function can change an unsigned long integer to a string?
system()
ltoa()
ultoa()
unsigned long can't be change into a string
3) What is the value return by strcmp() function when two strings are the same?
2
1
0
Error
4) What is built in library function for comparing the two strings?
strcmp()
equals()
str_compare()
string_cmp()
5) What will be the output of the below program?
```
#include<stdio.h>
int main()
{
    char a[] = "%d\n";
    a[1] = 'b';
    printf(a, 65);
    return 0;
}
```
b
a
A
65

Ans –

B
**Explanation:**The function gets() is used for collecting a string of characters terminated by new line from the standard input stream **stdin.**
Therefore gets() is more appropriate for reading a multi-word string.
C
**Explanation:**The function ultoa() is used for converting an unsigned long integer to a string.
C
Explanation:
C library function strcmp() compares the two strings with each other and the value is return accordingly.

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

Comparison occurs between a first string (str1) with a second string (str2).
On comparing the two string, the values return by a function strcmp() are:
If, str1 is equal to str2 then Return value = 0
If, str1 is greater than str2 then Return value > 0
If, str1 is less than str2 then Return value < 0
A
**Explanation:**The strcmp() is a built-in function available in "string.h" header file. It is used for comparing the two strings. It returns 0 if both are same strings. It returns positive value greater than 0 if first string is greater than second string, otherwise it returns negative value.
C
**Explanation:**
**Step 1:** char a[] = "%d\n"; The variable 'a' is declared as an array of characters and initialized with string "%d".
**Step 2:** a[1] = 'b'; Here, we overwrite the second element of array ?a? by 'b'. Hence the array ?a? becomes "%c".
**Step 3:** printf(a, 65); becomes printf("%c", 65);
Therefore it will print the ASCII value of 65. Hence the output is 'A'.