

CS 210: Data Structures



Dr. Balu L. Parne
Assistant Professor,
CSE, SVNIT Surat.



Introduction to Strings



Introduction to String

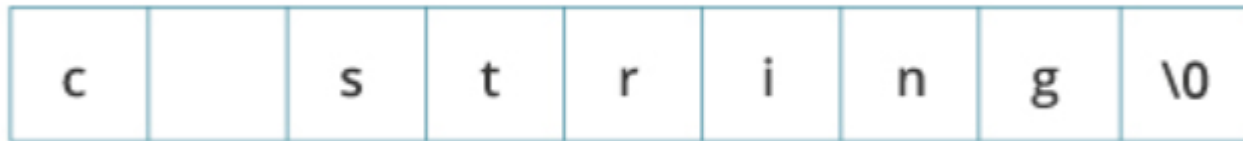
- Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character `'\0'`.
- String
 - A sequence of characters that is treated as a single data item
 - Have you used strings so far ???
 - What about `printf("Hello");` statement?
 - "Hello" is a string !!!

Introduction to Strings:

- 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.
- In C programming, a string is a sequence of characters terminated with a null character `\0`.

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

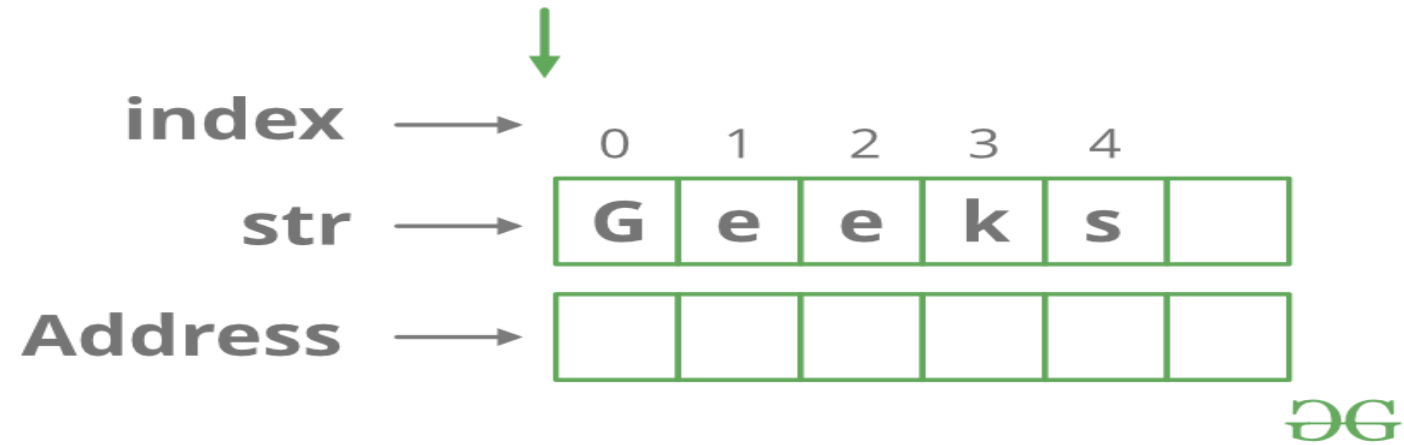
- When the compiler encounters a sequence of characters enclosed in the double quotation marks, it appends a null character `\0` at the end by default.



Introduction to Strings:

String in C

`char str[] = "Geeks"`

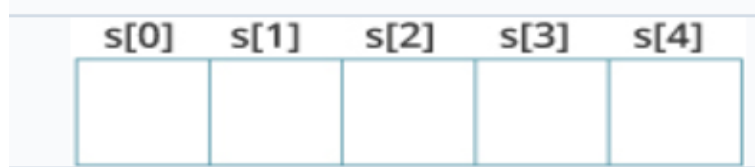


Declaring and initializing string variables:

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

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

- Here, we have declared a string of 5 characters.



```
char str_name[size];
```

In this example, syntax **str_name** is any name given to the string variable and **size** is used to define the length of the string, i.e the number of characters strings will store.

Declaring and initializing string variables:

- The following declaration and initialization create a string consisting of the word "Hello".

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

- 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".
- We can declare the above string in other way as follows:

```
char greeting[] = "Hello";
```

Declaring and initializing string variables:

What will be the output of following Code?

```
#include <stdio.h>
int main ()
{
    char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
    printf("Greeting message: %s\n", greeting );
    return 0;
}
```

```
Greeting message: Hello
```


Declaring and initializing string variables:

- Initialization of a strings is possible in a different ways as follows:

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

```
char c[50] = "abcd";
```

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

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

```
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.

c[0]	c[1]	c[2]	c[3]	c[4]
a	b	c	d	\0

Read String from the user :

- The **scanf** function is used to read the string.
- The **scanf()** function reads the sequence of characters until it encounters whitespace (space, newline, tab etc.).
- Using scanf function

```
char addr[20];  
scanf("%s", addr);
```
- scanf terminates as soon as the first white space is found !!!
- What happens if you want to input "NEW YORK" in addr ???
- Only "NEW" will be stored in addr !!!

Read String from the user :

- 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;
}
```

C:\Users\balup\Desktop\C Programs\Strings\StringInputScanf.exe

Enter name: Balu PArne

Your name is Balu.

Process exited after 5.523 seconds with return value 0

Press any key to continue . . .

Read String from the user :

- Example 2: scanf() to read a string

```
#include<stdio.h>
int main()
{
    char str[50]; // declaring string
    scanf("%s",str); // reading string
    printf("%s",str); // print string
    return 0;
}
```


There is possibility of a doubt that why we have not used the '&' sign with string name 'str' in scanf statement!

We know that the '&' sign is used to provide the address of the variable to the scanf() function to store the value read in memory. As str[] is a character array so using str without braces '[' and ']' will give the base address of this string. That's why we have not used '&' in this case as we are already providing the base address of the string to scanf.

Read String from the user :

- How to read a line of text?
- Example 2: gets() and puts()

```
#include <stdio.h>
int main()
{
    char name[30];
    printf("Enter name: ");
    gets(name); // read string
    printf("Name: ");
    puts(name); // display string
    return 0;
}
```

 C:\Users\balup\Desktop\C Programs\Strings\StringInputgets.exe

```
Enter name: Balu Parne Department of CoED SVNIT Surat
Name: Balu Parne Department of CoED SVNIT Surat
```

```
-----
Process exited after 40.58 seconds with return value 3221225477
Press any key to continue . . .
```

Read String from the user :

- How to read a line of text? Example 3: 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;
}
```

C:\Users\balup\Desktop\C Programs\Strings\StringInputFgets.exe

```
Enter name: Balu Parne Department of CoED SVNIT Surat
Name: Balu Parne Department of CoED
```

```
-----
Process exited after 68.76 seconds with return value 0
Press any key to continue . . .
```

Read String from the user :

- we have used `fgets()` function to read a string from the user.

`fgets(name, sizeof(name), stdin); // 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 used 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.*

Commonly used String Functions : `strlen()`

- The `strlen()` function calculates the length of a given string.
- It is defined in the `<string.h>` header file.
- The `strlen()` function takes a string as an argument and **returns its length**. The returned value is of type long int.
- **Note that the `strlen()` function doesn't count the null character `\0` while calculating the length.**

Commonly used String Functions : strlen()

```
#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;
```

C:\Users\balup\Desktop\C Programs\Strings\stringLengthFun.exe

Length of string a = 7

Length of string b = 7

Process exited after 1.177 seconds with return value 0
Press any key to continue . . .

Difference : strlen() and sizeof()

```
#include <stdio.h>
#include <string.h>
int main()
{
    char a[20]="Program";
    char b[25]={'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));
    printf("Size/length of string a = %ld \n",sizeof(a));
    printf("Size/length of string b = %ld \n",sizeof(b));
    return 0;
}
```

C:\Users\balup\Desktop\C Programs\Strings\stringLengthFun.exe

Length of string a = 7

Length of string b = 7

Size/length of string a = 20

Size/length of string b = 25

Process exited after 0.8288 seconds with return value 0

Press any key to continue . . .

Commonly used String Functions : `strcmp()`


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

Return Value from strcmp():

Return Value	Remarks
0	if both strings are identical (equal)
negative	if the ASCII value of the first unmatched character is less than second.
positive integer	if the ASCII value of the first unmatched character is greater than second.

Commonly used String Functions : strcmp()

```
#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;
}
```

 C:\Users\balup\Desktop\C Programs\Strings\StringCmpFun.exe

```
strcmp(str1, str2) = 0
strcmp(str1, str3) = -1
```

```
-----
Process exited after 0.3999 seconds with return value 0
Press any key to continue . . .
```

Commonly used String Functions : **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.

Commonly used String Functions : strncmp

```
#include <string.h>
int main()
{
    char s1[30] = "CoED SVNIT Surat ";
    char s2[30] = "CoED SVNIT Surat Gujrat";
    /* 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;
}
```

string 1 and string 2 are equal

Process exited after 0.4023 seconds with return value 0
Press any key to continue . . .

Commonly used String Functions : strncmp

```
#include <string.h>
int main()
{
    char s1[30] = "CoED SVNIT Surat ";
    char s2[30] = "CoED SVNIT Surat Gujrat";
    /* below it is comparing first 8 characters of s1 and s2*/
    if (strncmp(s1, s2, 20) == 0)
    {
        printf("string 1 and string 2 are equal");
    }else
    {
        printf("string 1 and 2 are different");
    }
    return 0;
}
```

string 1 and 2 are different

Process exited after 0.4821 seconds with return value 0
Press any key to continue . . .

Commonly used String Functions : `strcat()`

- In C programming, the *strcat()* function concatenates (joins) two strings.

- The function definition of *strcat()* is:


*char *strcat(char *destination, const char *source)*

- The `strcat()` function takes two arguments:
 - destination - destination string
 - source - source string
- The *strcat()* function concatenates the destination string and the source string, and **the result is stored in the destination string.**
- It is defined in the *string.h* header file.

Commonly used String Functions : strcat()

What will be the output of the following program?

```
#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;
```

 C:\Users\balup\Desktop\C Programs\Strings\StringCatFun.exe

```
Output string after concatenation: HelloWorld
-----
Process exited after 0.724 seconds with return value 0
Press any key to continue . . .
```

Commonly used String Functions : strcat()

What will be the output of the following program?

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[15] = "SVNIT", str2[] = "Surat";
    // concatenates str1 and str2
    // the resultant string is stored in str1.
    strcat(str1, str2);
    puts(str1);
    puts(str2);
    return 0;
}
```

When we use strcat(), the size of the destination string should be large enough to store the resultant string. If not, we will get the segmentation fault error.

C:\Users\balup\Desktop\C Programs\Strings\StringCatFun1.exe

```
SVNITSurat
Surat
```

```
-----
Process exited after 0.4864 seconds with return value 0
Press any key to continue . . .
```

Commonly used String Functions : `strcpy()`


- In C programming, the `strcpy()` function is used to copy strings
- The function prototype of `strcpy()` is:

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

- The `strcpy()` function copies the string pointed by source (including the null character) to the destination.
- The `strcpy()` function also returns the copied string.
- The `strcpy()` function is defined in the `string.h` header file.

Commonly used String Functions : strcpy()

```
#include <stdio.h>
#include <string.h>
int main()
{
    char s1[30] = "VNIT Nagpur";
    char s2[30] = "SVNIT Surat";
    /* this function has copied s2 into s1 */
    strcpy(s1, s2);
    printf("String s1 is: %s", s1);
    return 0;
}
```

 C:\Users\balup\Desktop\C Programs\Strings\StringCpyFun.exe

String s1 is: SVNIT Surat

Process exited after 0.2487 seconds with return value 0

Press any key to continue . . .

Commonly used String Functions : strcpy()

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20] = "C programming";
    char str2[20];
    // copying str1 to str2
    strcpy(str2, str1);

    puts(str2); // C programming

    return 0;
}
```

C:\Users\balup\Desktop\C Programs\Strings\StringCpyFun2.exe

C programming

Process exited after 0.6536 seconds with return value 0
Press any key to continue . . .

C – String functions Summary:

strlen - Finds out the length of a string
strlwr - It converts a string to lowercase
strupr - It converts a string to uppercase
strcat - It appends one string at the end of another
strncat - It appends first n characters of a string at the end of another.
strcpy - Use it for Copying a string into another
strncpy - It copies first n characters of one string into another
strcmp - It compares two strings
strncmp - It compares first n characters of two strings
strcmpi - It compares two strings without regard to case ("i" denotes that this function ignores case)
stricmp - It compares two strings without regard to case (identical to strcmpi)
strnicmp - It compares first n characters of two strings, Its not case sensitive
strdup - Used for Duplicating a string
strchr - Finds out first occurrence of a given character in a string
strrchr - Finds out last occurrence of a given character in a string
strstr - Finds first occurrence of a given string in another string
strset - It sets all characters of string to a given character
strnset - It sets first n characters of a string to a given character
strrev - It Reverses a string

Home Work Problems:

- Write a program to find length of a string.
- Write a program to copy one string into another and count the number of characters copied.
- Write a program that checks whether the given string is a palindrome or not.
- Write a program to entered username and password from user and check whether it matches with predefined username and password string or not. If it matched print "Access Granted" Otherwise print "Access Denied".

Home Work Problems:

- Write a program to convert lowercase string to uppercase string.
-

Lowercase to uppercase using **strupr** :

```
#include <stdio.h>
#include <string.h>
#define MAX_SIZE 100 // Maximum string size
int main()
{
    char str[MAX_SIZE];
    /* Input string from user */
    printf("Enter your text : ");
    gets(str);
    strupr(str); // Convert to uppercase
    printf("Uppercase string : %s", str);
    return 0;
}
```

Lowercase to uppercase :

```
#include <stdio.h>
#define MAX_SIZE 100 // Maximum string size
int main()
{
    char str[MAX_SIZE];
    int i;
    /* Input string from user */
    printf("Enter your text : ");
    gets(str);
    for(i=0; str[i]!='\0'; i++)
    {
        if(str[i]>='a' && str[i]<='z')
        {
            str[i] = str[i] - 32;
        }
    }
    printf("Uppercase string : %s",str);
    return 0;
}
```

Uppercase to Lowercase using **strlwr**

```
#include <stdio.h>
#include <string.h>
#define MAX_SIZE 100 // Maximum string size
int main()
{
    char str[MAX_SIZE];
    /* Input string from user */
    printf("Enter any string: ");
    gets(str);
    strlwr(str); // Convert to Lowercase
    printf("Lowercase string: %s", str);
    return 0;
}
```

Uppercase to Lowercase:

```
#include <stdio.h>
#define MAX_SIZE 100 // Maximum string size
int main()
{
    char str[MAX_SIZE];
    int i;
    /* Input string from user */
    printf("Enter any string: ");
    gets(str);
    // Iterate loop till last character of string
    for(i=0; str[i]!='\0'; i++)
    {
        if(str[i]>='A' && str[i]<='Z')
        {
            str[i] = str[i] + 32;
        }
    }
    printf("Lower case string: %s", str);
    return 0;
}
```

Practice Problems:

- Write a C program to find total number of alphabets, digits or special character in a string.
- Write a C program to count total number of vowels and consonants in a string.
- Write a C program to find reverse of a string.
- Write a C program to check whether a string is palindrome or not.
- Write a C program to search all occurrences of a character in given string.



Thank You...!!! 😊