C Fundamental

Strings

Objectives

- String in C
- String and Pointer
- Commonly Functions
- Backslash Escapes

Section 1

STRING IN C

String in C - 1

- String is a collection of characters
- In C programming, the collection of characters is stored in the form of arrays. Hence it's called C-strings.
- C-strings are arrays of type char terminated with null character, that is **‘\0’** (ASCII value of null character is 0).

Define a C-string:

```
char str[] = "aString";
```

- In the above code, str is a string and it holds 4 characters.
- Although, "C++" has 3 character, the null character \0 is added to the end of the string automatically.

String in C - 2

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

```
char str[] = "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.

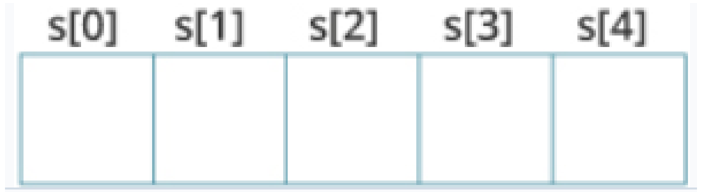
c		s	t	r	i	n	g	\0

String in C - 3

- Declare a string:

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

- And we have declared a string of 5 characters.

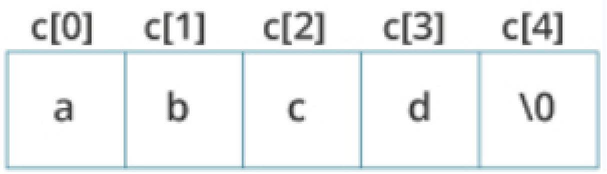


String in C - 4

Initialize strings:

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

char c[50] = "abcd";

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

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

And we have initialized a string of 5 characters.



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

String in C - 5

- Assigning Values to Strings

✓ Arrays and strings are second-class citizens in C, they do not support the assignment operator once it is declared. For example:

```
char c[100];
c = "C programming";  // Error! array type is not assignable.
```

String in C - 6

Read a string:

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

```c
#include <stdio.h>
int main()
{
    char name[20];
    printf("Enter name: ");
    scanf("%s", name);
    printf("Your name is %s.", name);
    return 0;
}
```

```
Enter name: Joe Biden
Your name is Joe.
```

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

String in C - 6

To read a line of text:

Use the **fgets()** function to read a line of string. And use **puts()** to display the string.

```
#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;
}
```

```
Enter name: Joe Biden
Name: Joe Biden
```

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

String in C - 6

- In the above example about how to get a line of text. We have used **fgets()** function to read a string from the user:

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

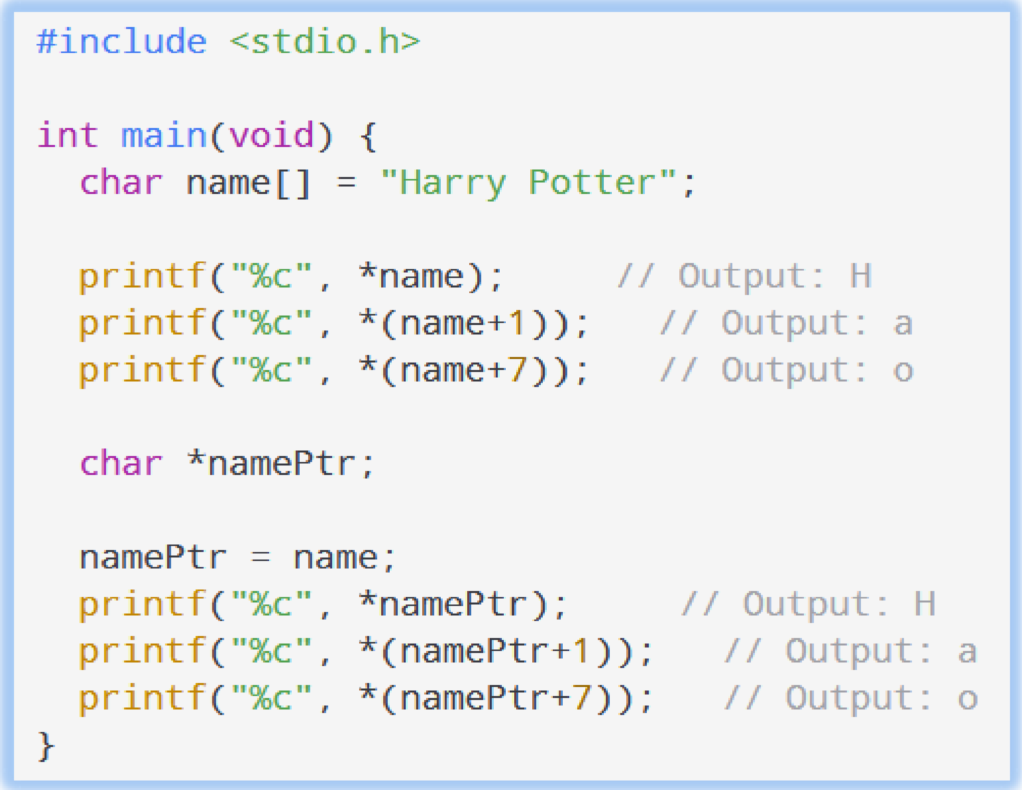
- The **sizeof(name)** results to 30. Hence, we can take a maximum of 30 characters as input which is the size of the name string.
- To print the string, we have used **puts(name);**

Section 2

STRING AND POINTER

String and Pointer

Like pointer and array, each character in a string is equivalent to each element in an array

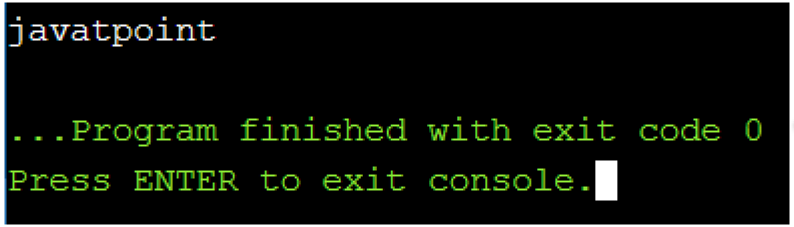


```c
#include <stdio.h>

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
}
```

String and Pointer - 1

There are various advantages of using pointers to point strings. Following example to access the string via the pointer:

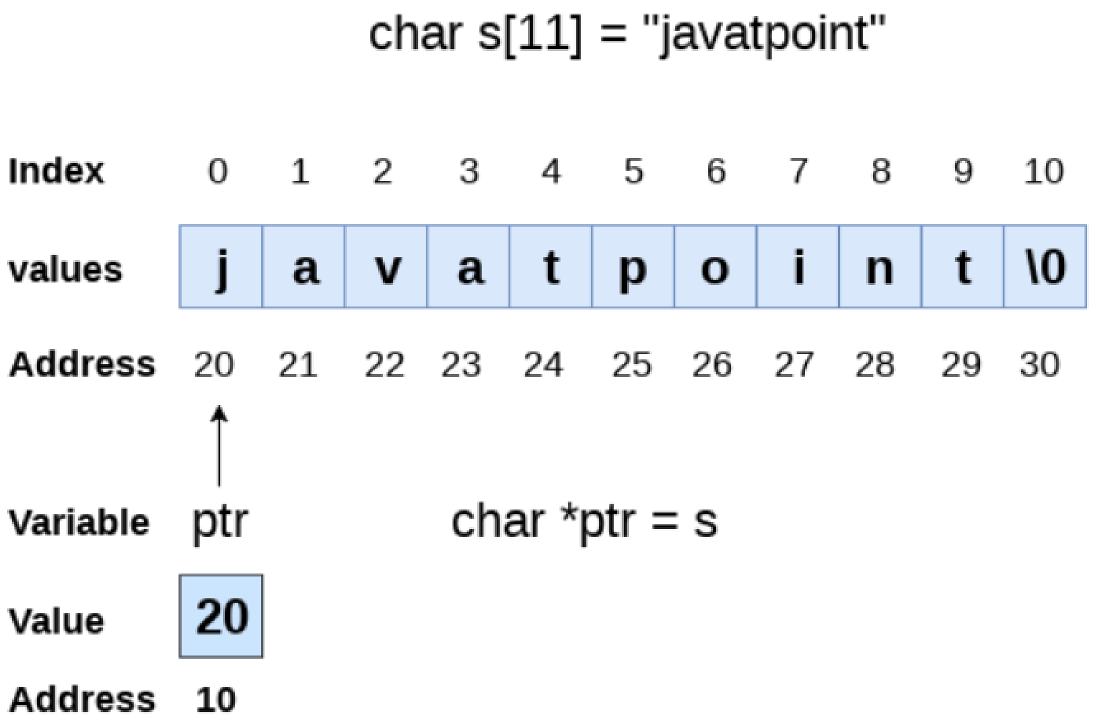
```
#include<stdio.h>
void main ()
{
    char s[11] = "javatpoint";
    char *p = s; // pointer p is pointing to string s.
    printf("%s",p); // the string javatpoint is printed if we print p.
}
```



```
javatpoint

...Program finished with exit code 0
Press ENTER to exit console.
```

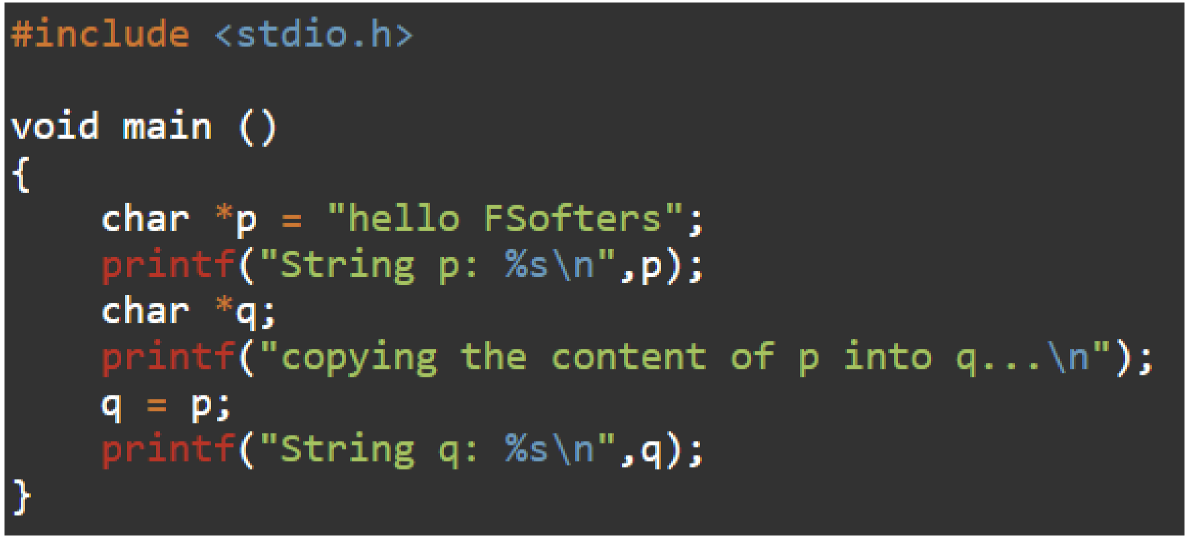
String and Pointer - 2

Look at the figure for the details of the above example

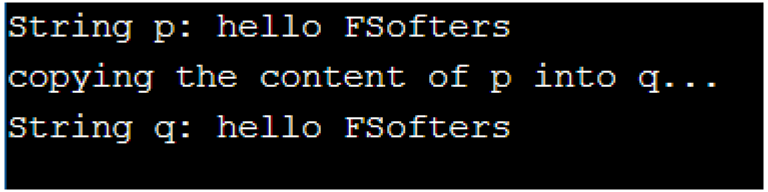

char s[11] = "javatpoint"
Index 0 1 2 3 4 5 6 7 8 9 10
values j a v a t p o i n t \0
Address 20 21 22 23 24 25 26 27 28 29 30
Variable ptr
char *ptr = s
Value 20
Address 10

String and Pointer - 3

Copy content of string by using pointer:



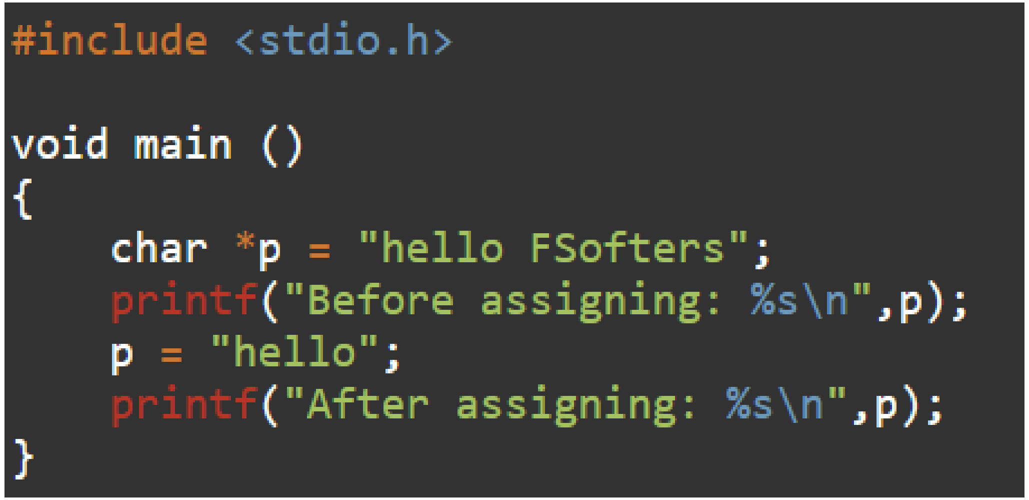
```c
#include <stdio.h>

void main ()
{
    char *p = "hello FSofters";
    printf("String p: %s\n",p);
    char *q;
    printf("copying the content of p into q...\n");
    q = p;
    printf("String q: %s\n",q);
}
```



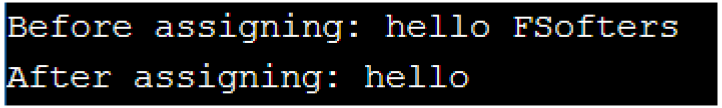
```
String p: hello FSofters
copying the content of p into q...
String q: hello FSofters
```

String and Pointer - 4

Once a string is defined, it cannot be reassigned to another set of characters. However, using pointers, we can assign the set of characters to the string. Consider the following example:



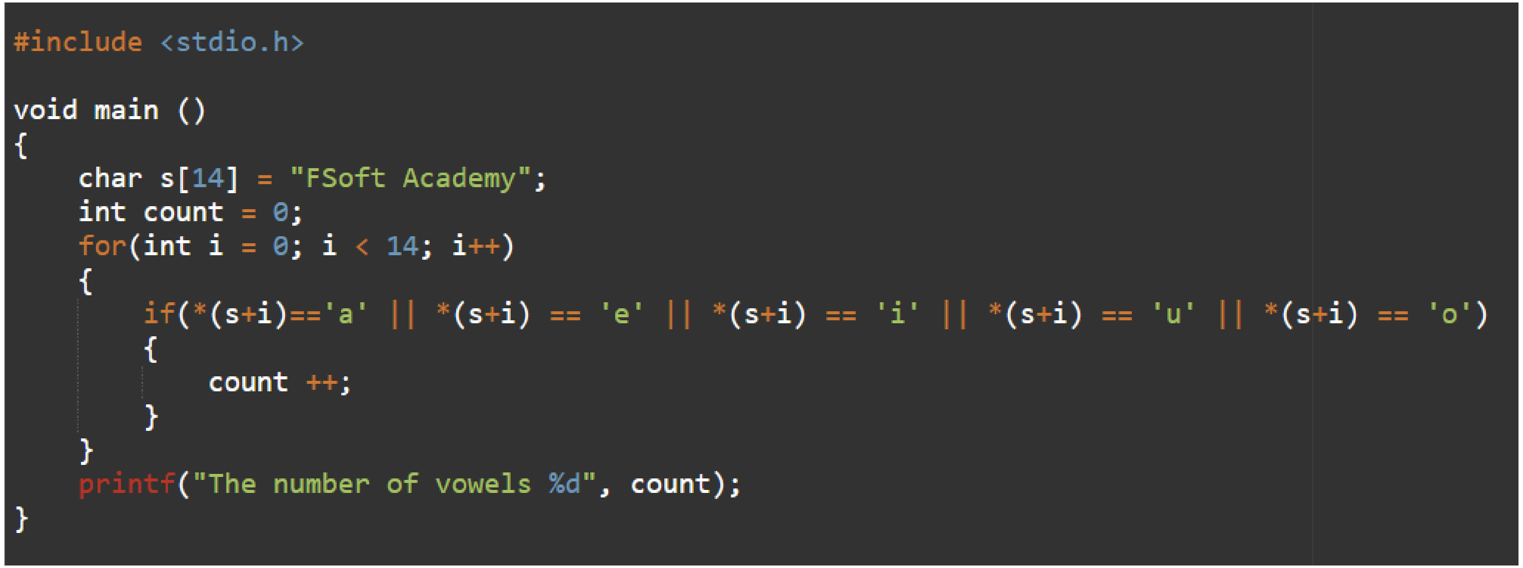
```
#include <stdio.h>

void main ()
{
    char *p = "hello FSofters";
    printf("Before assigning: %s\n",p);
    p = "hello";
    printf("After assigning: %s\n",p);
}
```



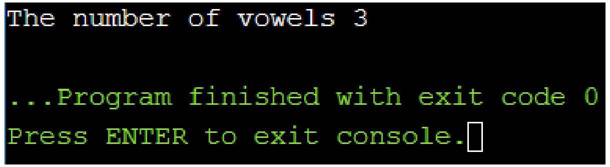
```
Before assigning: hello FSofters
After assigning: hello
```

String and Pointer - 5

Using the length of string. Let's see an example of counting the number of vowels in a string:



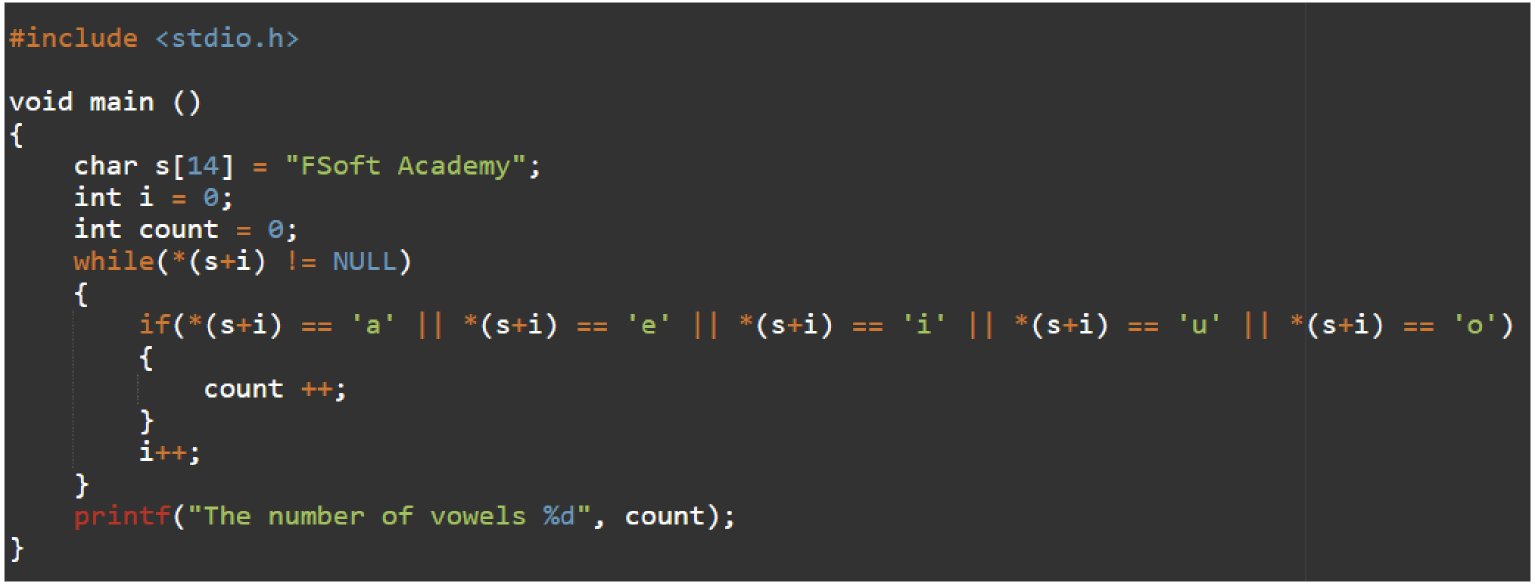
```
#include <stdio.h>

void main ()
{
    char s[14] = "FSoft Academy";
    int count = 0;
    for(int i = 0; i < 14; i++)
    {
        if(*(s+i)=='a' || *(s+i) == 'e' || *(s+i) == 'i' || *(s+i) == 'u' || *(s+i) == 'o')
        {
            count ++;
        }
    }
    printf("The number of vowels %d", count);
}
```



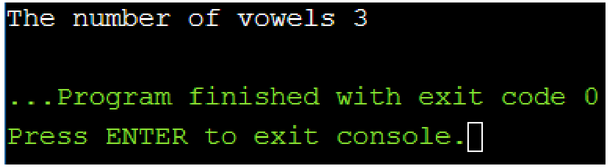
```
The number of vowels 3

...Program finished with exit code 0
Press ENTER to exit console.
```

String and Pointer - 6

And using the null character. Counting the number of vowels in a string:



```
#include <stdio.h>

void main ()
{
    char s[14] = "FSoft Academy";
    int i = 0;
    int count = 0;
    while(*(s+i) != NULL)
    {
        if(*(s+i) == 'a' || *(s+i) == 'e' || *(s+i) == 'i' || *(s+i) == 'u' || *(s+i) == 'o')
        {
            count ++;
        }
        i++;
    }
    printf("The number of vowels %d", count);
}
```



```
The number of vowels 3

...Program finished with exit code 0
Press ENTER to exit console.
```

Section 3

COMMONLY FUNCTIONS

String in C: Commonly Functions

No.	Function	Description
1)	strlen(string_name)	returns the length of string name.
2)	strcpy(destination, source)	copies the contents of source string to destination string.
3)	strcat(first_string, second_string)	concats or joins first string with second string. The result of the string is stored in first string.
4)	strcmp(first_string, second_string)	compares the first string with second string. If both strings are same, it returns 0.

String in C: Commonly Functions - strlen()

- The strlen() function takes a string as an argument and returns its length. The returned value is of type size_t (the unsigned integer type).

```
#include <stdio.h>
#include <string.h>
int main()
{
    char a[20]="Program";
    char b[20]={'P','r','o','g','r','a','m','\0'};

    //using the %zu format specifier to print size_t
    printf("Length of a = %zu \n",strlen(a));
    printf("Length of b = %zu \n",strlen(b));

    return 0;
}
```

OUTPUT:

```
Length of a = 7
Length of b = 7
```

String in C: Commonly Functions - strcpy() (1)

- 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.

String in C: Commonly Functions - strcpy() (2)

```
#include <stdio.h>
#include <string.h>

int main() {
   char str1[20] = "C programming";
   char str2[20];

   // copying str1 to str2
   strcpy(str2, str1);
   // print str2:
   puts(str2); // C programming

   return 0;
}
```

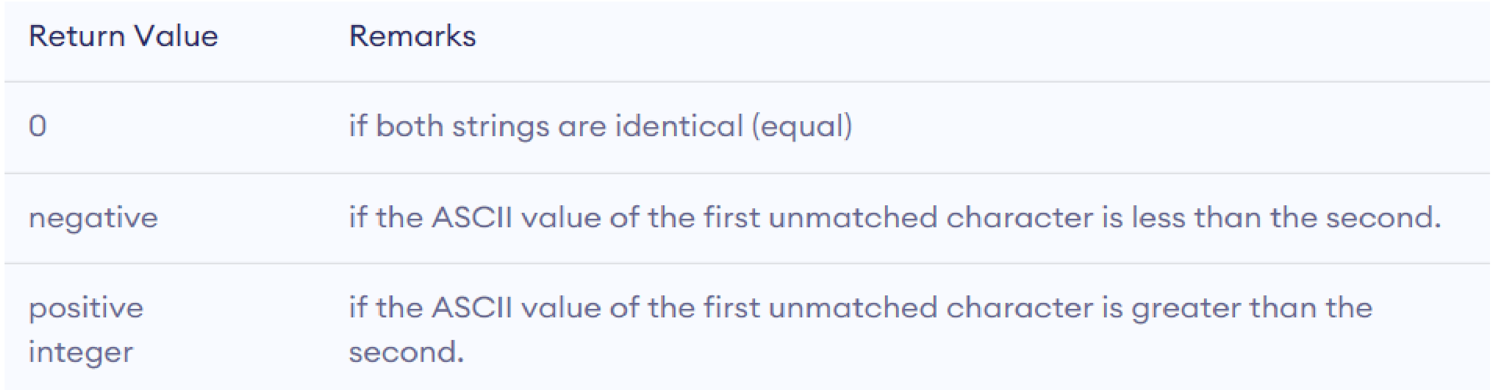
String in C: Commonly Functions – strcmp() (1)

- Prototype:

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

✓ The strcmp() compares two strings character by character.

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



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

String in C: Commonly Functions – strcmp() (2)

```
#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
```

String in C: Commonly Functions - strcat() (1)

- The strcat() function contcatenates (joins) two strings.
- The strcat() function concatenates the destination string and the source string, and the result is stored in the destination string.
- The function definition of strcat() is:

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

String in C: Commonly Functions - strcat() (2)

- The strcat() function contcatenates (joins) two strings.
- The function definition of strcat() is:

```c
#include <stdio.h>
#include <string.h>

int main() {
   char str1[100] = "This is ", str2[] = "programiz.com";

   // concatenates str1 and str2
   // the resultant string is stored in str1.
   strcat(str1, str2);

   puts(str1);
   puts(str2);

   return 0;
}
```

OUTPUT:

```
This is programiz.com

programiz.com
```

String in C: Commonly Functions - strcpy() (1)

- *strcpy()* is a standard library function in C/C++ and is used to copy one string to another.
- In C it is present in string.h header file and in C++ it is present in cstring header file.
- **Syntax**:

```
char* strcpy(char* dest, const char* src);
```

String in C: Commonly Functions - strcpy() (2)

- **Parameters**: This method accepts following paramters:
- **dest**: Pointer to the destination array where the content is to be copied.
- **src**: string which will be copied.
- **Return Value**: After copying the source string to the destination string, the *strcpy()* function returns a pointer to the destination string.
- Below program explains different usages of this library function:
- **Syntax**:

```
char* strcpy(char* dest, const char* src);
```

String in C: Commonly Functions - strcpy() (3)

```
// C program to illustrate
// strcpy() function ic C/C++
#include<stdio.h>
#include<string.h>
int main () {
    char str1[]="Hello Geeks!";
    char str2[] = "GeeksforGeeks";
    char str3[40];
    char str4[40];
    char str5[] = "GfG";

    strcpy(str2, str1);
    strcpy(str3, "Copy successful");
    strcpy(str4, str5);
    printf ("str1: %s\nstr2: %s\nstr3: %s\nstr4:
                %s\n", str1, str2, str3, str4);
    return 0;
}
```

OUTPUT

```
str1: Hello Geeks!

str2: Hello Geeks!

str3: Copy successful

str4: GfG
```

String in C: More Commonly-used String Functions

- **strcat** - concatenate two strings
- **strchr** - string scanning operation
- **strcmp** - compare two strings
- **strcpy** - copy a string
- **strlen** - get string length
- **strncat** - concatenate one string with part of another
- **strncmp** - compare parts of two strings
- **strncpy** - copy part of a string
- **strrchr** - string scanning operation

Section 4

BACKSLASH ESCAPES

Backslash Escapes

String literals may not directly in the source code contain embedded newlines or other control characters, or some other characters of special meaning in string.

To include such characters in a string, the backslash escapes may be used.

\\	- Literal backslash
\\"	- Double quote
\\'	- Single quote
\n	- Newline (line feed)
\r	- Carriage return
\b	- Backspace
\t	- Horizontal tab
\f	- Form feed
\a	- Alert (bell)
\v	- Vertical tab
\0	- Null character

Backslash Escapes: Example

```
#include<stdio.h>
int main(){
    int number = 2021;
    printf("You\nare\nlearning\n\'c\' language\n\"Do you know C language?\"");
    return 0;
}
```

OUTPUT:

```
You
are
learning
'c' language
"Do you know C language"
```

Thank you

Q&A