

IT 105: Introduction to Programming

Dr. Manish Khare
Dr. Bakul Gohel

Lecture 20




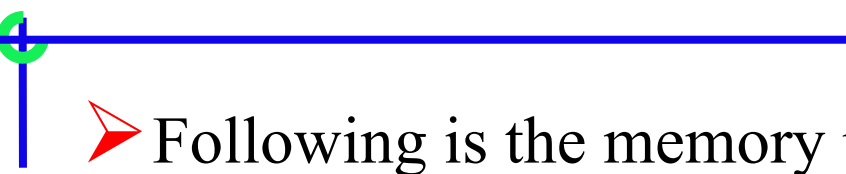


STRING

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**.
- `char name[10] = {'s', 't', 'r', 'i', 'n', 'g', '\0'};`
- So we can say that a string is just a one-dimensional array of characters with a null character ('\0') as it's the last element.

- 
- A string literal is just a sequence of characters enclosed in double quotes (""). It is also known as a string constant.
 - If you follow the rule of array initialization then you can write the previous statement as follows –
 - `char greeting[] = "Hello";`

- 
- Following is the memory presentation of the above defined string in C.

Index	0	1	2	3	4	5
Variable	H	e	l	l	o	\0
Address	0x23451	0x23452	0x23453	0x23454	0x23455	0x23456

- Actually, you do not place the *null* character at the end of a string constant. The C compiler automatically places the '\0' at the end of the string when it initializes the array.



➤ Reading Strings

If we declare a string by writing

```
Char str [100];
```

Then Str can be read by using three ways:

Using scanf function

Using gets() function

Using getchar(), getch() or getche() function repeatedly



➤ Writing Strings

Strings can be displayed on screen using three ways:

Using `printf()` function

Using `puts()` function

Using `putchar()` function repeatedly



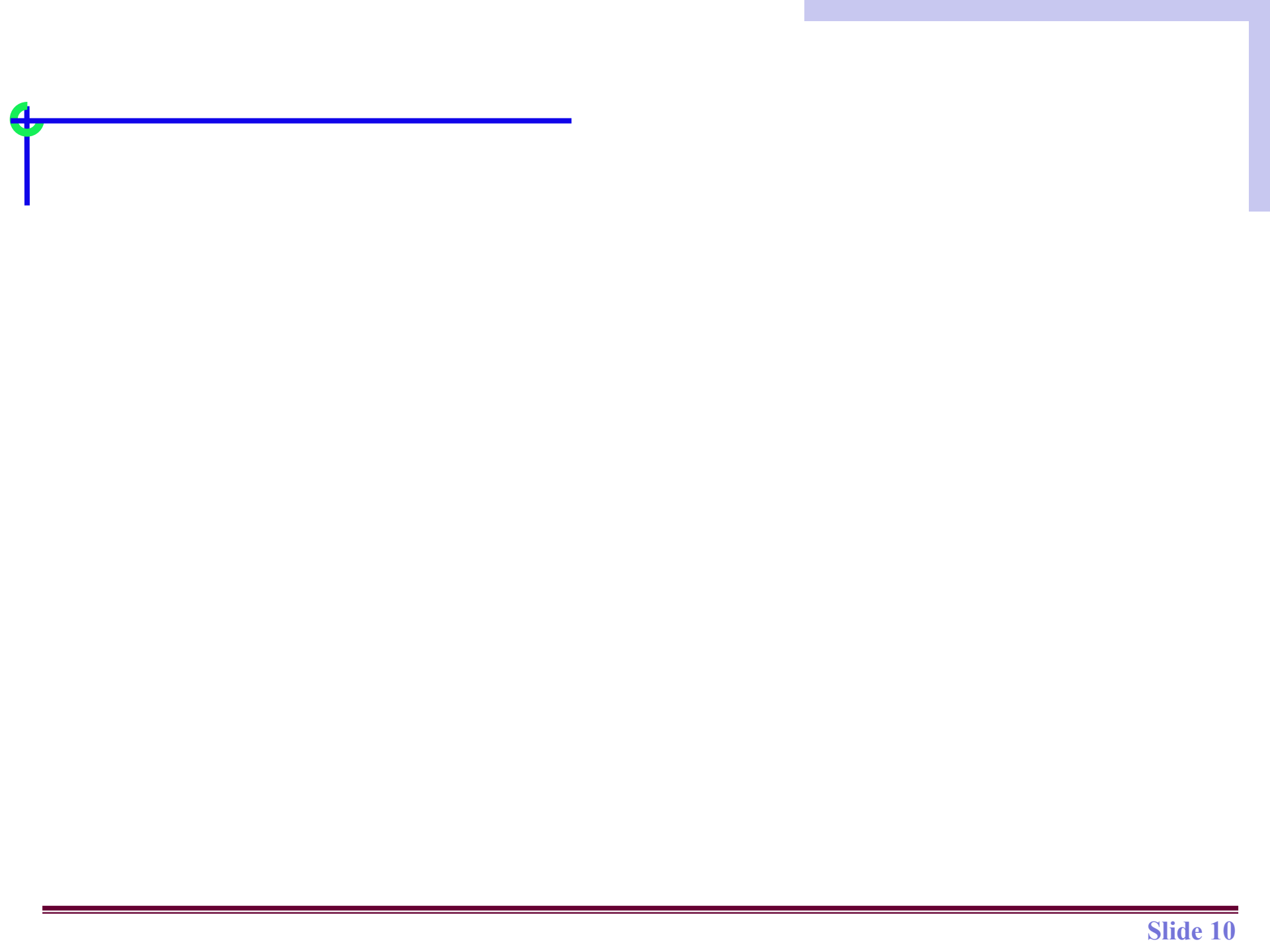
➤ In C, we can store a string either in fixed length format or in variable length format

➤ String

- Fixed Length
- Variable Length
 - Length Controlled
 - Delimited

Operations on Strings

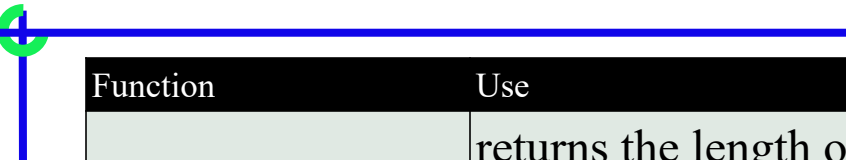
- Finding the length of a string
- Converting characters of a string into Upper case
- Converting characters of a string into Lower case
- Concatenating two strings to form a new string
- Appending a string to another string
- Comparing two strings
- Reversing a String
- Extracting a substring from left of the string
- Extracting a substring from Right of the string
- Extracting a substring from the middle of the string
- Inserting a string in another string
- Indexing
- Deleting a String from main string
- Replacing a pattern with another pattern in a string



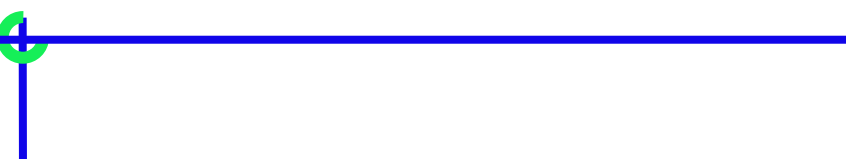
Predefined string functions

➤ We can perform different kinds of string functions like joining of 2 strings, comparing one string with another or finding the length of the string.

Function	Use
strlen	calculates the length of string
strcat	Appends one string at the end of another
strncat	Appends first n characters of a string at the end of another
strcpy	Copies a string into another
strncpy	Copies first n characters of one string into another
strcmp	Compares two strings
strncmp	Compares first n characters of two strings
strchr	Finds first occurrence of a given character in a string
strrchr	Finds last occurrence of a given character in a string
strstr	Finds first occurrence of a given string in another string



Function	Use
strcspn	returns the length of the initial part of the string s1 not containing any of the characters of the string s2.
strspn	returns the length of the initial part of the string s1 only containing the characters of the string s2.
strpbrk	returns the first occurrence of any of the characters of the string s2 in the string s1.
strtok	finds s2 in s1 and returns a pointer to it and returns NULL if not found.



➤ These functions are defined in "**string.h**" header file, so we need to include this header file also in our code by writing

- **#include <string.h>**

strlen

➤ **strlen(s1)** calculates the length of string s1.

```
#include <stdio.h>
#include <string.h>
int main()
{
    char name[ ]= "Hello";
    int len1, len2;
    len1 = strlen(name);
    len2 = strlen("Hello World");
    printf("length of %s = %d\n", name, len1);
    printf("length of %s = %d\n", "Hello World", len2);
    return 0;
}
```

Output

length of Hello = 5

length of Hello World =
11

strlen doesn't count '\0' while calculating the length of a string.

strcat

➤ **strcat(s1, s2)** concatenates(joins) the second string s2 to the first string s1.

```
#include <stdio.h>
#include <string.h>
int main()
{
    char s2[ ]= "World";
    char s1[20]= "Hello";
    strcat(s1, s2);
    printf("Source string = %s\n", s2);
    printf("Target string = %s\n", s1);
    return 0;
}
```

Output

Source string = World

Target string =
HelloWorld

strncat

➤ **strncat(s1, s2, n)** concatenates(joins) the first 'n' characters of the second string s2 to the first string s1.

```
#include <stdio.h>
#include <string.h>
int main()
{
    char s2[ ]= "World";
    char s1[20]= "Hello";
    strncat(s1, s2, 2);
    printf("Source string = %s\n", s2);
    printf("Target string = %s\n", s1);
    return 0;}
```

Output

Source string = World

Target string =
HelloWo

strcpy

➤ **strcpy(s1, s2)** copies the second string s2 to the first string s1.

```
#include <stdio.h>
#include <string.h>
int main()
{
    char s2[ ]= "Hello";
    char s1[10];
    strcpy(s1, s2);
    printf("Source string = %s\n", s2);
    printf("Target string = %s\n", s1);
    return 0;
}
```

Output

Source string = Hello

Target string = Hello

strncpy

➤ **strncpy(s1, s2, n)** copies the first 'n' characters of the second string s2 to the first string s1.

```
#include <stdio.h>
#include <string.h>
int main()
{
    char s2[ ]= "Hello";
    char s1[10];
    strncpy(s1, s2, 2);
    s1[2] = '\0'; /* null character manually added */
    printf("Source string = %s\n", s2);
    printf("Target string = %s\n", s1);
    return 0;
}
```

Output

Source string = Hello

Target string = He

strcmp

➤ **strcmp(s1, s2)** compares two strings and finds out whether they are same or different. It compares the two strings character by character till there is a mismatch. If the two strings are **identical**, it returns a **0**. If not, then it returns the difference between the ASCII values of the first non-matching pair of characters.

```
#include <stdio.h>
#include <string.h>
int main()
{
    char s1[ ]= "Hello";
    char s2[ ]= "World";
    int i, j;
    i = strcmp(s1, "Hello");
    j = strcmp(s1, s2);
    printf("%d \n %d\n", i, j);
    return 0;
}
```

Output

0

-15

The difference between the ASCII values of H(72) and W(87) is - 15.

strncmp

➤ **strncmp(s1, s2, n)** compares the first 'n' characters of s1 and s2.

```
#include <stdio.h>
#include <string.h>
int main()
{
    char s1[ ]= "Hello";
    char s2[ ]= "Heral";
    int i, j;
    i = strncmp(s1, s2, 2);
    printf("%d\n", i);
    return 0;}
```

Output

0

strchr

➤ **strchr(s1, c)** returns a pointer to the first occurrence of the character **c** in the string **s1** and returns **NULL** if the character **c** is not found in the string **s1**.

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

int main()
{
    char s1[ ]= "Hello Blogsdope";
    char c = 'B';
    char *p;
    p = strchr(s1, c);
    printf("%s\n", p);
    return 0;
}
```

Output

Blogsdope

strrchr

➤ **strrchr(s1, c)** returns a pointer to the last occurrence of the character **c** in the string **s1** and returns **NULL** if the character **c** is not found in the string **s1**.

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

int main()
{
    char s1[ ]= "Hello Blogsdope";
    char c = 'o';
    char *p;
    p = strrchr(s1, c);
    printf("%s\n", p);
    return 0;
}
```

Output
ope

strstr

➤ **strstr(s1, s2)** finds the first occurrence of the string s2 in the string s1.

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

int main()
{
    char s1[ ]= "HelloBlogsdope";
    char s2[ ]= "Blog";
    char *p;
    p = strstr(s1, s2);
    printf("%s\n", p);
    return 0;
}
```

Output

Blogsdope

strspn

➤ **strspn(s1, s2)** returns the length of the initial part of the string s1 only containing the characters of the string s2.

```
#include <stdio.h>
#include <string.h>
int main()
{
    char s1[ ]= "123abc";
    char s2[ ] = "123456790";
    int i = strspn(s1, s2);
    printf("%d\n", i);
    return 0;
}
```

Output

3

strcspn

➤ **strcspn(s1, s2)** returns the length of the initial part of the string s1 **not** containing any of the characters of the string s2.

```
#include <stdio.h>
#include <string.h>
int main()
{
    char s1[ ]= "HelloBlogsdope";
    char s2[ ] = "lo";
    int i = strcspn(s1, s2);
    printf("The first character is
    matched at %d\n", i+1);
    return 0;
}
```

Output

The first character is
matched at 3

strpbrk

➤ **strpbrk(s1, s2)** returns the first occurrence of any of the characters of the string s2 in the string s1.

```
#include <stdio.h>
#include <string.h>
int main()
{
    char s1[ ]= "Hey 123 Blogsdope";
    char s2[ ]= "123456790";
    char *p;
    p = strpbrk(s1, s2);
    printf("%s\n", p);
    return 0;
}
```

Output

123 Blogsdope

strtok

➤ **strtok(s1, s2)** finds s2 in s1 and returns a pointer to it and returns NULL if not found.

```
int main()
{
    char s1[ ]= "Hey, 123, Blogsdope";
    char *p;
    p = strtok(s1, ",");
    while(p != NULL)
    {
        printf("%s\n", p);
        p = strtok(NULL, ",");
    }
    return 0;
}
```

Output

Hey

123

Blogsdope

Exercise

Write a program in C to print the following pattern

Pattern


A

Ap

App

Appl

Apple



```
#include <stdio.h>
#include <string.h>
int main(void)
{
    //variable
    char str[] = "Apple";

    //length of the string
    int len = strlen(str);


    int r, c;
```

```
//output
for (r = 0; r < len; r++) {
    for (c = 0; c <= r; c++) {
        printf("%c", str[c]);
    }
    printf("\n");
}

printf("End of code\n");
return 0;
}
```



Write a program in C to print the reverse of string “Hello World”



```
#include <stdio.h>
#include <string.h>
int main(void)
{
    //variable
    char str[100], tmp;
    int i, len, mid;

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

    //find number of characters
    len = strlen(str);
    mid = len/2;
```

```
//reverse
for (i = 0; i < mid; i++) {
    tmp = str[len - 1 - i];
    str[len - 1 - i] = str[i];
    str[i] = tmp;
}

//output
printf("Reversed string: %s\n", str);

printf("End of code\n");
return 0;
}
```

Pointers and Strings

