BASIC PROGRAMMING LANGUAGE

LESSON 7

Strings

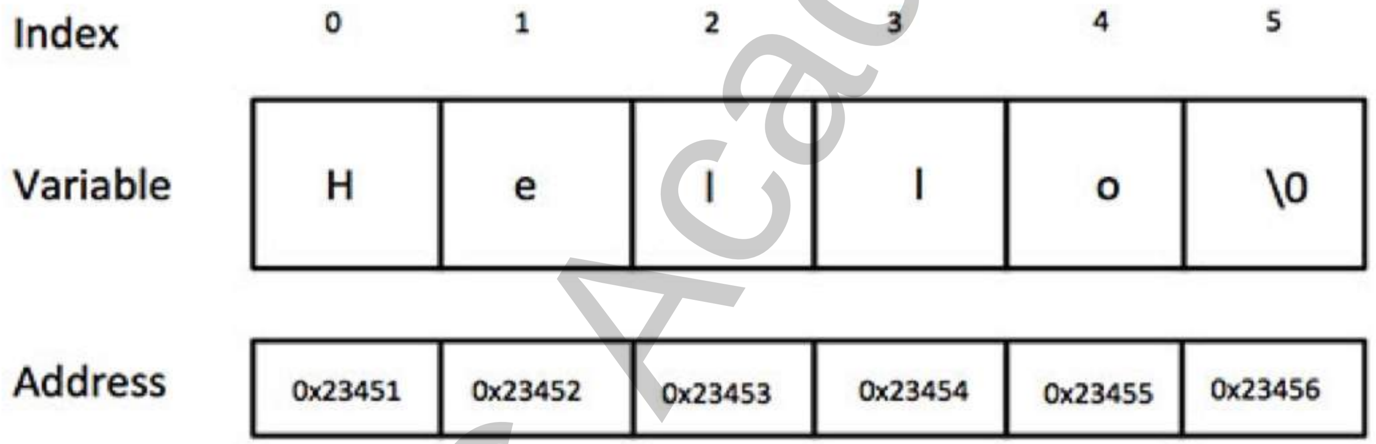
CONTENT

String Concept

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

String Concept

- Following is the memory presentation of the “Hello” string in C:

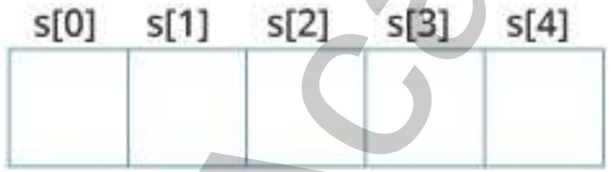


Index	0	1	2	3	4	5
Variable	H	e	l	l	o	\0
Address	0x23451	0x23452	0x23453	0x23454	0x23455	0x23456

Declaring String Variables

- You can declare a string like an array of characters.
- Example of declaring a string variable with 5 characters:

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



- Above string can store maximum 4 characters and the last character must be null character (\0).
- 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.

String Initialization

- You can initialize strings in many ways:

```
char s[] = "Hello";
char s[50] = "Hello";
char s[] = {'H', 'e', 'l', 'l', 'o', '\0'};
char s[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

- If we are trying to initialize 5 characters (the last character is '\0') to a char array, you must declare a string with size = 6:

```
char s[6] = "Hello"; // This is good
char s[5] = "Hello"; // This is bad and you should never do this
```

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 s[100];
s = "C Programming";  // Error! Array type is not assignable
```

- Note: Use the `strcpy()` function to copy the string instead.

String Input / Output

- You can input / output a string from console with two kinds of statements:
- Formatted I/O:
  - `scanf()`
  - `printf()`
- String I/O functions:
  - `gets();`
  - `puts();`

String Input / Output

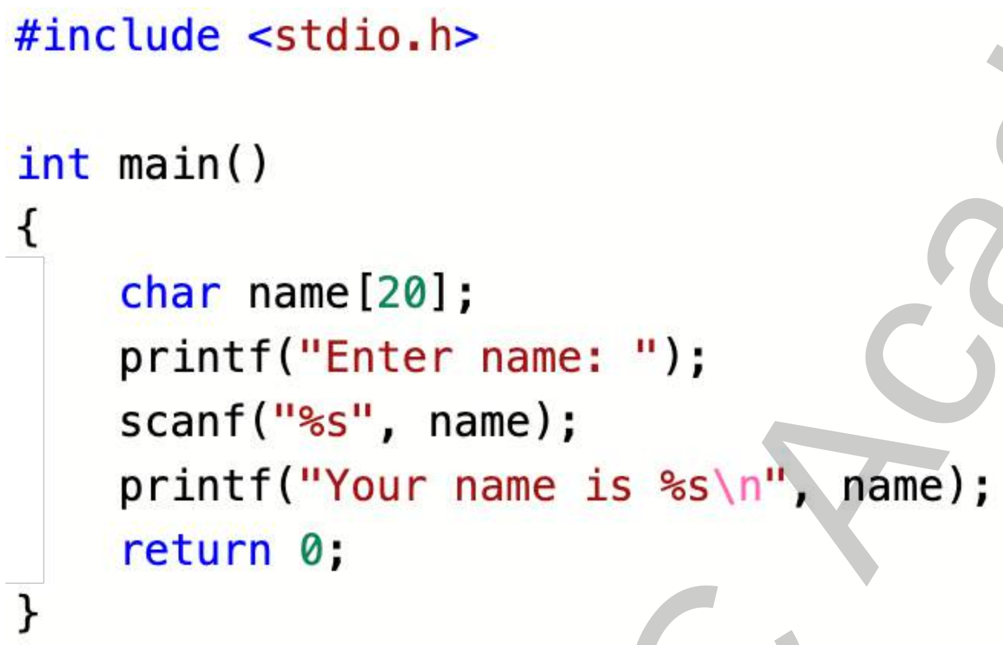
- The `scanf()` and `printf()` functions are used to accept and display mixed data types with a single statement.
- The syntax to accept a string:

```
scanf("%s", str);
```

- The syntax to display a string:

```
printf("%s", str);
```

String Input / Output Example 1



```
#include <stdio.h>

int main()
{
    char name[20];
    printf("Enter name: ");
    scanf("%s", name);
    printf("Your name is %s\n", name);
    return 0;
}
```

String Input / Output

- String I/O operations are also carried out using functions from the standard I/O library called `<stdio.h>`.
- The `gets()` function is the simplest method of accepting a string through standard input.
- Input characters are accepted till the Enter key is pressed.
- The `gets()` function replaces the terminating ‘\n’ new line character with the ‘\0’ character.
- Syntax:

```
gets(str);
```

String Input / Output

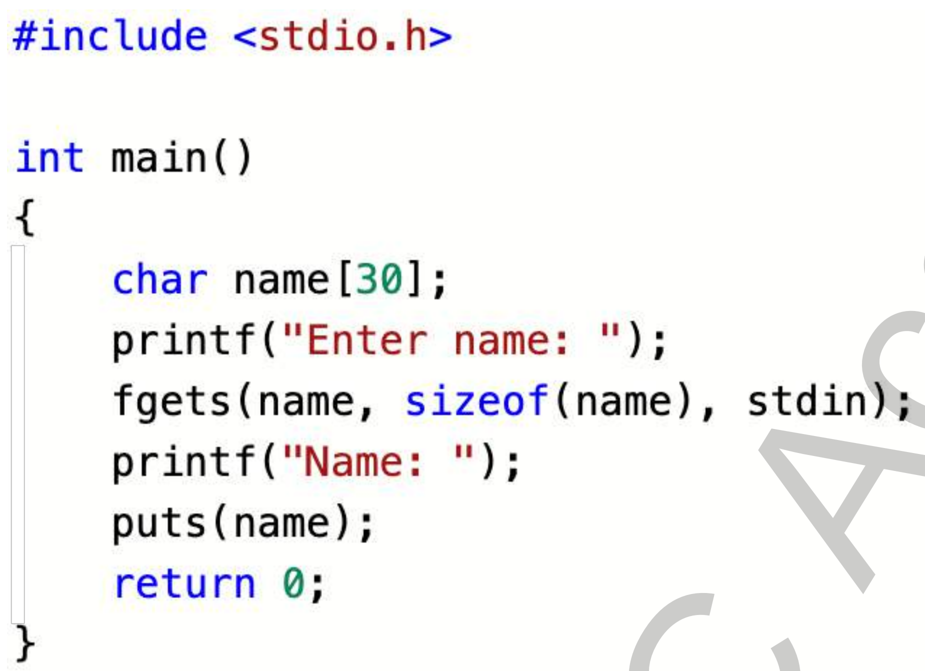
- The puts() function is used to display a string on the standard output device.
- The puts function in C is used to write a line or string to the output stream (stdout) that is up to, but does not include, the null character.
- The puts function also appends a newline character to the output and returns an integer.
- Syntax :

```
puts(str);
```

How to Read a Line of Text?

- `gets()` and `fgets()` are functions in C language to take input of string with spaces in between characters. The problem of `gets()` is that it suffers from buffer overflow that is it takes more input than it is supposed to take.
- This problem is solved using `fgets()`.
- You can use the `fgets()` function to read a line of string and use the `puts()` function to display the string.
- Example of read a text line:

How to Read a Line of Text?



```
#include <stdio.h>

int main()
{
    char name[30];
    printf("Enter name: ");
    fgets(name, sizeof(name), stdin);
    printf("Name: ");
    puts(name);
    return 0;
}
```

String Functions

- You need to often manipulate 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 "string.h".
- Some commonly string functions:

String Functions

Function	Description
strlen()	computes string's length
strcpy()	copies a string to another
strcat()	concatenates(joins) two strings
strcmp()	compares two strings
strlwr()	converts string to lowercase
strupr()	converts string to uppercase

strcat() Functions

- Joins two string values into one.
- Syntax:

```
char *strcat(str1, str2);
```

- Concatenates the `str2` at the end of `str1`.
- The function returns `str1`.

strcmp() Functions

- Compares two strings and returns an integer value based on the results of the comparison.
- Syntax:

```
int strcmp(str1, str2);
```

- The function returns a value:
    - Less than zero if str1 < str2
    - Zero if str1 is same as str2
    - Greater than zero if str1 > str2

strcpy() Functions

- Copies the value in one string onto another.
- Syntax:

```
char *strcpy(str1, str2);
```

- The value of `str2` is copied onto `str1`.
- The function returns `str1`.

strlen() Functions

- Determines the length of a string.
- Syntax:

```
int strlen(str);
```

- The function returns an integer value for the length of `str`.

strchr() Functions

- Determines the occurrence of a character in a string.
- Syntax:

```
char *strchr(str, ch);
```

- The function returns a value:
  - Pointer to the first occurrence of the character (pointed by `ch`) in the string, `str`
  - `NULL` if it is not present

String Functions Example

```c
#include <stdio.h>
#include <string.h>
int main() {
    char str1[10] = "Hello";
    char str2[10] = "World";
    char str3[10];
    int len, re;
    strcpy(str3, str1);
    printf("strcpy(str3, str1): %s\n", str3);
    strcat(str1, str2);
    printf("strcat(str1, str2): %s\n", str1);
    len = strlen(str1);
    printf("strlen(str1): %d\n", len);
    re = strcmp(str1, str2);
    printf("strcmp(str1, str2): %d\n", re);
    return 0;
}
```

Summary

- 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.
- String I/O operations are carried out using functions from the standard I/O library called <stdio.h>.
- Functions for handling strings are found in the standard header file <string.h>:
  - `strcat(), strcmp(), strchr(), strcpy(), strlen()`

Thank you!