Content 24

Strings In C

#### String in C:-

String is an array of characters. Data of the same type are stored in an array for example, Integers can be stored in an integer array, similarly, a group of characters can be stored in a character array. **Character arrays are also called strings**. A string is a one-dimensional array of characters that is terminated by a null (‘\0’).

#### Declaration of strings:

Declaring a string is very simple, same as declaring a one-dimensional array. Below is the syntax for declaring a string.

char string\_name[size];

In the above syntax, string\_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 that the strings will store. Keep in mind that there is an extra terminating character which is the null character (‘\0’) that is used to indicate the termination of string.

char name[ ] = { 'T', 'O', 'M', 'M', 'Y', '\0' } ;

Each character in the array, like “T”, occupies one byte of memory and the last character is always ‘\0’. The null character ‘\0’ looks like two characters, but it is actually only one character, with the \ indicating that what follows it is something special. Character array elements are stored in contiguous memory locations.

Equally we can make the string by assigning character values to each member.

name[0]='T';

name[1]='O';

name[2]='M';

name[3]='M';

name[4]='Y';

name[5]='\0';

The placeholder for string variables is %s.

**Note:**  There is difference between ‘\0’ and ‘0’. The ASCII value of ‘\0’ is 0, and for ‘0’, the value is 48. The terminating null ‘\0’ and ‘0’ are not same.

The null (‘\0’) is important in C programming because it is the only way the functions that work with a string can know where the string ends.  When a string not terminated by a ‘\0’, then it is not really a string but merely a collection of characters.

#### Example of string in C:-

#include<stdio.h>

int main()

{

// declare and initialize string

char str[] = "CodingInC";

printf("%s",str);

return 0;

}

//Output:- CodeWithHarry

In order to read a string that contains the spaces, we use the gets() function. The purpose of gets is to ignore the whitespaces. When a newline is reached, gets stops reading. For example:

#include <stdio.h>

int main() {

char name[50];

printf("Enter your name: ");

gets(name);

printf("My name is %s ",name);

return 0;}

#### Conclusion:-

Here we have learned about how to use strings in C programming. Each member of the array contains one of the characters in the string. By using scanf() we are not capable of receiving multi-word strings and *also we can’t use void space*. The way to get a multi-word string is by using the function gets().

**Code1:**

#include <stdio.h>

void printStr(char str[])

{

    int i = 0;

    while (str[i] != '\0')

    {

        printf("%c", str[i]);

        i++;

    }

    printf("\n");

}

int main()

{ /\* 2 method of making string

     char str[] = {'T', 'o', 'm', 'm', 'y', '\0'};

     char str[6] = "Tommy"; \*/

    char str[9];

    printf("Using Custom function gets\n");

    gets(str);

    printf("Using custom function printStr\n");

    printStr(str);

    printf("Using printf %s\n", str);

    printf("using puts: \n");

    puts(str);

    return 0;

}

**Output:**

Using Custom function gets

Tommy

Using custom function printStr

Tommy

Using printf Tommy

using puts:

Tommy