CL1002 – Programming Fundamentals Lab

Lab # 13

Strings

Instructor: Muhammad Saood Sarwar

Email: saood.sarwar@nu.edu.pk

Department of Computer Science,

National University of Computer and Emerging Sciences FAST Peshawar

Strings in C

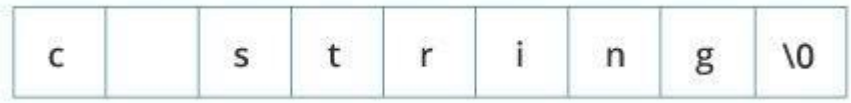
- Strings are used for storing text/characters.
- For example, "Hello World" is a string of characters.
- Unlike many other programming languages, C does not have a String type to easily create string variables. Instead, you must use the char type and create an array of characters to make a string in C:

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

For example:

```
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.

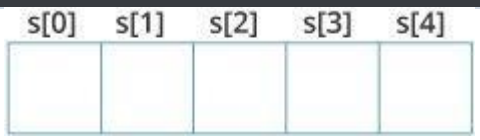


Memory Diagram

How to declare a string?

Here's how you can declare strings:

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



String Declaration in C

Here, we have declared a string of 5 characters.

How to initialize strings?

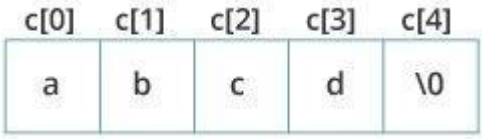
You can initialize strings in a number of ways.

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

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

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

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



String Initialization in C

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,

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

Modify Strings

To change the value of a specific character in a string, refer to the index number, and use **single quotes**:

Example

```c
char greetings[] = "Hello World!";
greetings[0] = 'J';
printf("%s", greetings);
// Outputs Jello World! instead of Hello World!
```

Another Way of Creating Strings

In the examples above, we used a "string literal" to create a string variable. This is the easiest way to create a string in C.

You should also note that you can create a string with a set of characters. This example will produce the same result as the example in the beginning of this page:

Example

```
char greetings[] = {'H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '!', '\0'};
printf("%s", greetings);
```

Why do we include the \0 character at the end? This is known as the "null terminating character", and must be included when creating strings using this method. It tells C that this is the end of the string.

Read String from the user.

You can use the scanf() function to read a string.

The scanf() function reads the sequence of characters until it encounters whitespace (space, newline, tab, etc.).

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;
}
```

Output

```
Enter name: Dennis Ritchie
Your name is Dennis.
```

Even though Dennis Ritchie was entered in the above program, only "Dennis" was stored in the name string.

It's because there was a space after Dennis.

Also notice that we have used the code name instead of &name with scanf().

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

This is because name is a char array, and we know that array names decay to pointers in C.

Thus, the name in scanf() already points to the address of the first element in the string, which is why we don't need to use &.

How to read a line of text?

You can use the fgets() function to read a line of string. And, you can use puts() to display the string.

Example 2: 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;
}
```

Output

```
Enter name: Dennis Ritchie
Name: Dennis Ritchie
```

Here, 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);.

Note: The gets() function can also be 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.

Passing Strings to Functions

Strings can be passed to a function in a similar way to arrays. Learn more about passing arrays to a function.

```c
#include <stdio.h>
void displayString(char str[]);

int main()
{
   char str[50];
   printf("Enter string: ");
   fgets(str, sizeof(str), stdin);
   displayString(str);    // Passing string to a function.
   return 0;
}
void displayString(char str[])
{
   printf("String Output: ");
   puts(str);
}
```

References:

https://www.programiz.com/c-programming/c-strings