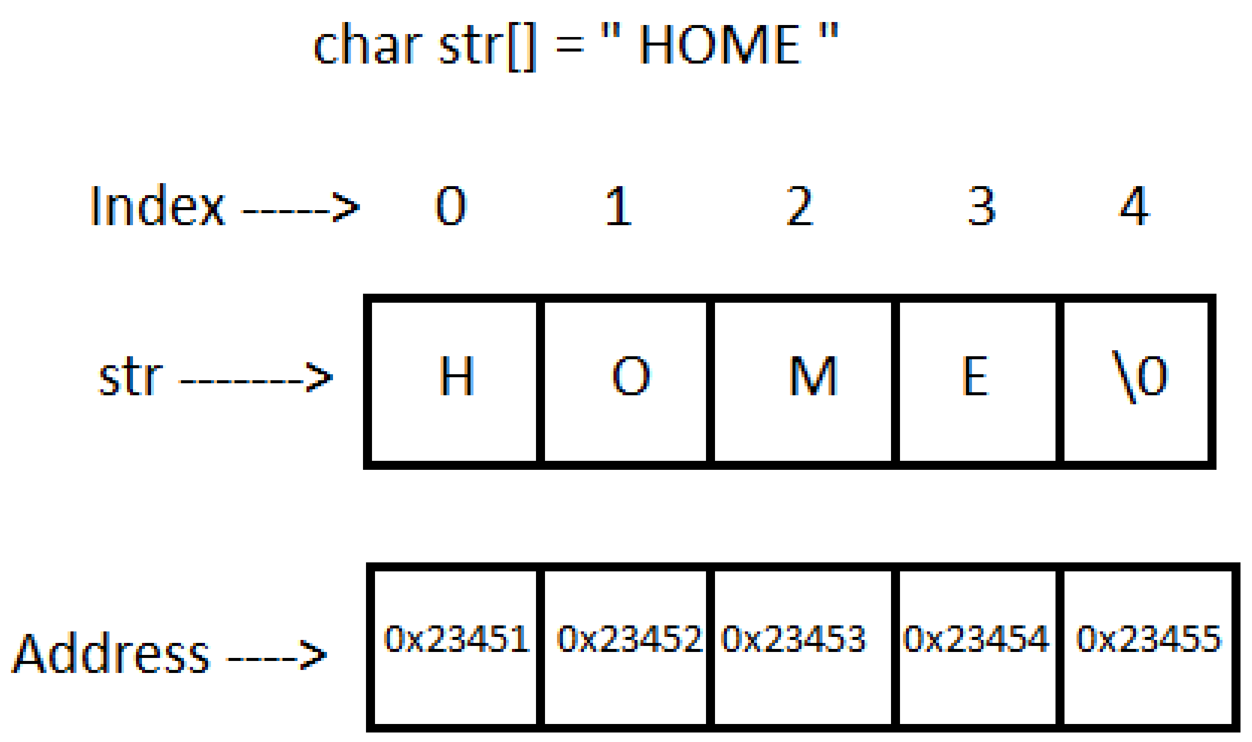
C PROGRAMMING

Strings & Character Arrays

Strings

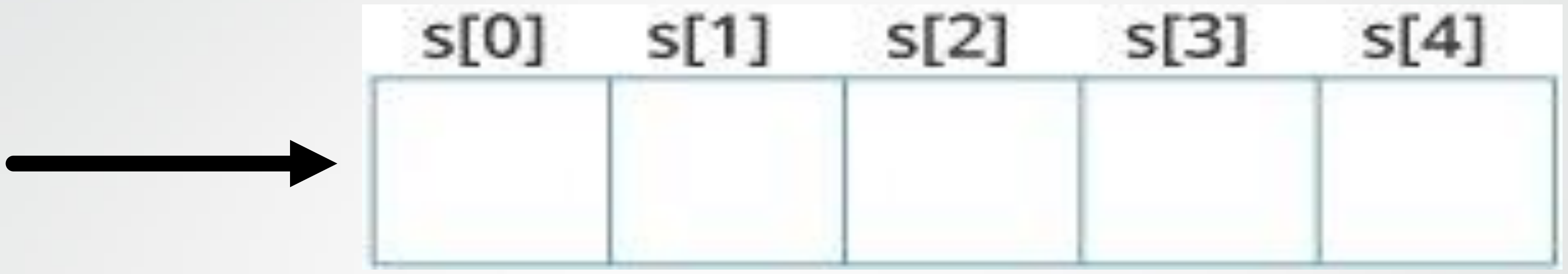
- String is a sequence of characters that are treated as a single data item and terminated by a null character that is '\0'.
- C language does not support strings as a data type.
- A string is actually a one-dimensional array of characters in C language.

Example


char str[] = " HOME "
Index -----> 0 1 2 3 4
str -------> H O M E \0
Address ----> 0x23451 0x23452 0x23453 0x23454 0x23455

How to declare a character array?

char s[5];



Valid

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

char c[50] = "abcd";

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

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

Invalid

```
char ch[3] = "hello";

char str[4];

str = "hello";
```

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 –**

```
#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.
```

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.

String Functions

gets()

- gets() is a pre-defined function in C which is used to read a string or a text line. And store the input in a well-defined string variable.
- The function terminates its reading session as soon as it encounters a newline character.
- Syntax:

```
gets( variable name );
```

String Functions

fgets()

- The function reads a text line or a string from the specified file or console. And then stores it to the respective string variable.
- fgets also terminates reading whenever it encounters a newline character.
- But furthermore, unlike gets(), the function also stops when EOF is reached or even if the string length exceeds the specified limit, n-1.
- The gets() function doesn’t have the provision for the case if the input is larger than the buffer.

As a result, memory clogging may occur.

That is why we use the fgets() function.

String Functions

- **Syntax --**

fgets(char *str, int n, FILE *stream)

str - It is the variable in which the string is going to be stored.

n - It is the maximum length of the string that should be read.

stream - It is the filehandle, from where the string is to be read.

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.

- **Example2 –**

```
#include <stdio.h>
void main()
{
    char name[30];
    printf("Enter name: ");
    fgets(name, sizeof(name),
stdin);
    printf("Name: ");
    puts(name);
}
```

We used fgets() function to read a string from the user.

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);.

Passing string to a Function

```
#include <stdio.h>
void displayString(char str[]);
void main()
{
    char str[50];
    printf("Enter string: ");
    fgets(str, sizeof(str), stdin);
    displayString(str);     // Passing string to a
function.
}
void displayString(char str[])
{
    printf("String Output: ");
    puts(str);
}
```

Commonly Used String Functions

- strlen() - calculates the length of a string
- strcpy() - copies one string to another
- strcmp() - compares two strings
- strcat() - concatenates two strings
- strrev() - It is used to show the reverse of a string

Commonly Used String Functions

- strlen() will return the length of the string passed to it.

E.g. int i = strlen(“Hello);

- strcpy() copies the second string argument to the first string argument.

E.g. strcpy(s1, “HelloWorld");
        strcpy(s2, s1);

Commonly Used String Functions

- strcmp() will return the ASCII difference between first unmatching character of two strings.

```
E.g. int i = strcmp(“hi", “hello");
     printf("%d", i);
```

- strcat() will concatenates two strings.

```
E.g. strcat("hello", "world");
```

Commonly Used String Functions

- strrev() used to reverse the given string expression.

E.g. strrev(s1);