Strings

A String is a sequence of characters terminated with a null character ‘\0’. Null character denotes string termination.

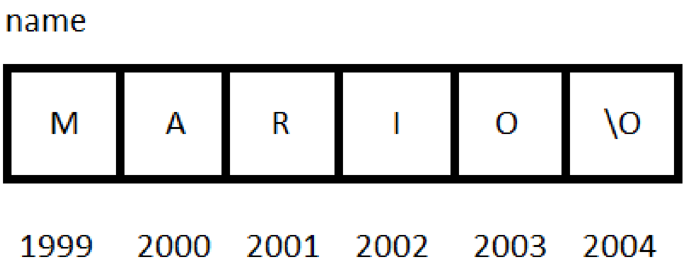
```
char name[ ] = {'M', 'A', 'R', 'I', '0', '\0'};

char nameTwo[ ] = {'L', 'U', 'I', 'G', 'I', '\0'};
```

Initialization

```
char name[ ] = {'M', 'A', 'R', 'I', '0', '\0'};

char name[ ] = “MARIO”;

char nameTwo[ ] = {'L', 'U', 'I', 'G', 'I', '\0'};

char nameTwo[ ] = “LUIGI”;
```

How it’s allocated in memory

```
char name[ ] = {'M', 'A', 'R', 'I', '0', '\0'};

char name[ ] = “MARIO”;
```



Format specifier

“%s”

char name[ ] = “MARIO”;

printf(“%s”, name);

Taking String input with space in C

- Using gets
  Syntax : char *gets(char *str)
  gets() has been removed from c11. So it might give you a warning when implemented.

- To overcome the above limitation, we can use fgets as :
  Syntax : char *fgets(char *str, int size, FILE *stream)
  Example : fgets(str, 20, stdin); as here, 20 is MAX_LIMIT according to declaration. It stops when n-1 chars input or new line is entered.

- Using %[^\n]%*c inside scanf
  Example : scanf(“%[^\n]%*c”, str);

- Using %[^\n]s inside scanf.
  Example : scanf(“%[^\n]s”, str);

String using Pointers

char *str = “hello world”;

Store string in memory & the assigned address is stored in the char pointer ‘str’

char *str = “hello world”; //can be reinitialized again

char str[] = “hello world”; // cannot be reinitialized again

Standard Library Functions → <string.h>

strlen(str) – It counts number of character excluding '\0'

strcpy(newString, oldString) – It copies value of old string to new string

strcat(firstString, secondString) – It concatenates first string with second string

strcmp(firstStr, secondStr) – It compares two string with the help of ASCII values and returns a value.

Positive → first > second(ASCII)

Negative → first < second(ASCII)