STRINGS

Definition:

- ✓ A String is a sequence of characters enclosed by Double quotation. The end of the string is marked with NULL character (\0).
- ✓ It is also termed as character string or a string of characters, is a sequence of elements of char data type.
- ✓ A String literal is a constant, it values cannot be changed.

Ex: Consider string "pinku" is stored as

p	i	n	k	u	**\0**
0	1	2	3	4	5

- ✓ If string is used in program, it is stored in consecutive bytes in memory and compiler places null character at the end

Declaring String Variables:

- ✓ The String is declared as array of characters.

Syntax: **char <string_name> [<array-length>];**

Where,

- ✓ string_name is variable name/name of string
- ✓ Array_name is size of array of string

Ex: char name[20];

Here 20 memory locations are allocated ranging from 0 to 19

- The size of character is 1 byte, each character occupies 1 byte and can hold maximum of 20 characters including NULL in above example.

Note:- *String Delimiter (**NULL**)*

- ➢ NULL character indicated end of string
- ➢ Some cases we should explicitly insert NULL character to terminate, but in literal string it is inserted automatically.
- ➢ An extra position must be provided in each string declaration for NULL character.

String Size and Length:

- ✓ String size is the number of bytes allocated during the declaration.
- ✓ String length is the number of characters provided in storing till null, but not including null.

Ex:- char name[10];

String size

name[10]	s	a	i	v	i	s	a	l	\0	
	0	1	2	3	4	5	6	7	8	9

Here *string size is 10* and *string length is 8.*

Initializing String:

- ✓ There are various ways to initialize strings.

They are,

- ➢ Initializing locations characters by character
- ➢ Partial array initialization
- ➢ Initialization without specifying the size
- ➢ Array initialization with a string

Initializing locations characters by character:

- ✓ Here the characters are stored in the specified order, and the remaining locations are initialized to NULL.

Ex:- 1. char name[7] = {'s','u','h','a','i','l'};

name →	s	u	h	a	i	l	\0
	0	1	2	3	4	5	6

2. char name[7]={'a,'u','g','u','s','t','\0'};

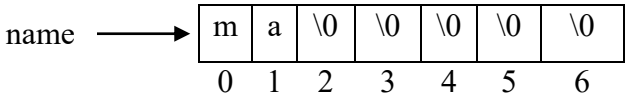
name →	a	u	g	u	s	t	\0

We can specify the null character also explicitly.

Partial array initialization

- ✓ The number of character to be initialized will be less than size, during which the characters are stored from left to right and remaining locations are set to NULL.

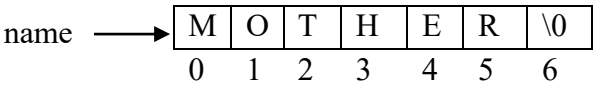
  Ex:- char name[7]={'m','a'};



Initialization without specifying the size

- ✓ Here the size of string is not specified in declaration; the compiler will set array size based on the total number of character initialized and null should be initialized expectedly.

  Ex:- char name[ ]={'M','O','T','H',E','R','\0'};



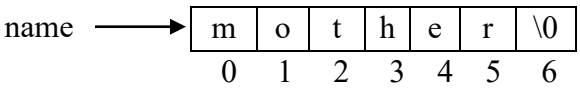
Size will be set as 7 characters

Array initialization with a string

- ✓ Here the string is assigned where in which additional one byte is reserved for NULL character.

  Ex: char name[ ]="mother";

- ✓ Here string length is 6 , but string size is 6+1=7 bytes, where 1 byte is used for NULL character.



Note:-

We can assign strings explicitly character by character as

char name[7];

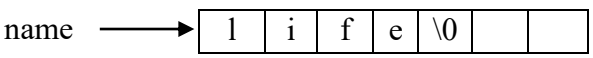
char name[0]='l';

char name[1]='i';

char name[2]='f';

char name[3]='e';

char name[4]='\0';



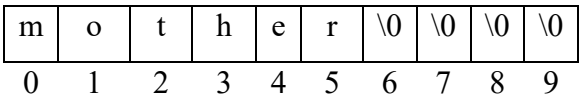
A String as an array of character

- ✓ We know that a string is an array of character, we can perform operations on individual positions.

Ex1: char a;

char name[10]= "mother";

a = name[2];

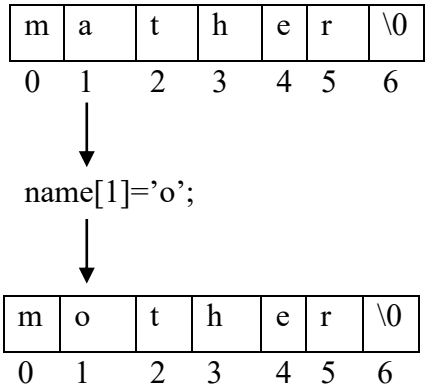


The value of a is 't'.

Ex2: char name[7]="mather";

We can change the value of name to "mother " as

name[1]='o';



String versus an Array of char

- ➢ A string is treated a single unit even though it is array of characters, where as in array of char each character is considered.
- ➢ Manipulation functions are available to perform operations on string like - copy, compare etc.. In array of char operations has to be performed on individual elements.

String and assignment operator

Generally we assign value to a string as

char name[10] = "mother";

- ✓ But when we assign one string value to another string it will be treated as illegal of which we need to make use of built-in string manipulation functions.

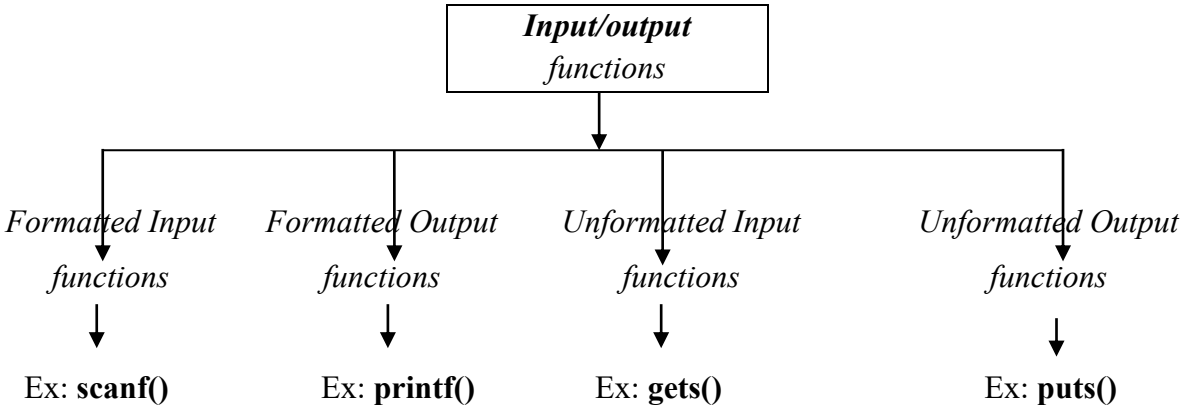
Ex: char name1[10]= "mother"; **//valid** name2=name1; **//invalid**

char name2[10] ***Copying a sting using = operator is invalid.***

name2="father"; **//invalid**

String Input / Output Functions

The different string input/output functions are,



Formatted Input function: scanf()

- ✓ Reading is possible using scanf function. The conversion specification for reading a string is %s.
- ✓ Since string is an array of characters and it is an address, the symbol "&" may not be needed while using scanf statement. While entering data, be sure not to type the quotation marks.

Ex:

```
char name[10];
printf("enter name");
scanf("%s",name);
```

Output:

```
Enter name
vishal
```

V	i	s	h	a	l	\0			
0	1	2	3	4	5	6	7	8	9

- ✓ During entering string values all whitespaces are removed and only non white space characters are copied.

Ex:

```
scanf ("%s",name);
mom dad
blank space
```

M	o	m	\0	?	?	?	?	?	?
0	1	2	3	4	5	6	7	8	9

The disadvantages of using scanf are

- ✓ The scanf functions stops reading at occurrence of first white space character, even if there are many other characters.

Ex:
```
char name[7];
scanf("%s"name);
```

Output:

mom dad
↓
whitespace

  - ➢ It stores only "mom" in name and string "dad" is neglected.
- ✓ We cannot enter a value which is longer than the number of characters in variable's declaration minus 1. There will not be having space for null character.
- ✓ To include spaces and other special symbols we can make use of edit set conversion code. i.e., (%[ ])

Syntax: **scanf(" %[…..]",str);**

- ✓ *Edit characters:* represents valid character that should be included.

Ex:
```
scanf("%[^\n]",name);
```

- ✓ The above scanf statements accept the all characters except "\n".

Formatted Output function: printf()

- ✓ Using printf function, it is possible to print a string, all characters, but not including, the null character.
- ✓ They are two ways to print strings using printf function

Ex:
```
printf("God Bless You All");
```

  - ➢ This prints all the characters including spaces which is in double quotes.

Output:

  - ➢ God Bless You All

Ex:
```
char name[10]="mother";
printf("%s",name);
```

  - ➢ The above printf statement will display output as

mother

Write a C program to read and display string using formatted input and output statement.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  char name[20];
  printf("enter name");
  scanf("%s",name);
  printf("the name is %s",name);
  getch();
}
```

Output:

Enter name

Niharika

The name is Niharika