**Chapter-1 (String)**

String is a collection of character or group of character, it is achieve in C language by using array character. The string in C language is one-dimensional array of character which is terminated by a null character '\0'. In other words string is a collection of character which is enclose between double cotes ( " " ).

**Declaration of string**

Strings are declared in C in similar manner as arrays. Only difference is that, strings are of char type.

Example: char s[5];

**Initializing Array string**

String are initialize into various way in c language;

Example:

char str[]="abcd";

OR char str[5]="abcd";

ORchar str[5]={'a','b','c','d','\0'};

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

ORchar str[5]={'a','b','c','d','\0'};

In c language string can be initialize using pointer.

char \*c="abcd";

**Reading String from user**

Example:

char str[5];

scanf("%s",&str);

Example:

#include<stdio.h>

#include<conio.h>

void main()

{

char str[10];

printf("Enter name: ");

scanf("%s",name);

printf("Your name is: %s.",name);

getch();

}

Example of reading string

Enter name: Hitesh kumar

Your name is: Hitesh

Note: String variable str can only take only one word. It is because when white space is encountered, the scanf() function terminates. to overcome this problem you can use gets() function.

Syntax

char str[5];

gets(str);

gets(): gets() are used to get input as a string from keyword, using gets() we can input more than one word at a time.

puts(): puts() are used to print output on screen, generally puts() function are used with gets() function.

**Example of String program**

#include<stdio.h>

#include<conio.h>

void main()

{

char str[10];

printf("Enter any string: ");

gets(str);

printf("String are: ");

puts(str);

getch();

}

Explanation: Here gets() function are used for input string and puts() function are used to show string on console or monitor.

Output

Enter String: hello word

String are: hello word

**C Library String functions**

All the library function of String is available in String.h header file.

|  |  |  |
| --- | --- | --- |
| S.no | Function | Purpose |
| 1 | strcpy(s1, s2) | Copies string s2 into string s1. |
| 2 | strcat(s1, s2) | Concatenates string s2 onto the end of string s1. |
| 3 | strlen(s1) | Returns the length of string s1. |
| 4 | strcmp(s1, s2) | Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2. |

|  |  |  |
| --- | --- | --- |
| 5 | strchr(s1, ch) | Returns a pointer to the first occurrence of character ch in string s1. |

|  |  |  |
| --- | --- | --- |
| 6 | strstr(s1, s2) | Returns a pointer to the first occurrence of string s2 in string s1. |

**Arithmetic Operations On Character :**

C Programming Allows you to Manipulate on String

Whenever the Character is variable is used in the expression then it is automatically Converted into Integer Value called ASCII value

All Characters can be Manipulated with that Integer Value.(Addition,Subtraction)

Examples :

ASCII value of : ‘a’ is 97

ASCII value of : ‘z’ is 121

Possible Ways of Manipulation :

Way 1: Displays ASCII value[ Note that %d in Printf ]

char x = 'a';

printf("%d",x); // Display Result = 97

Way 2 : Displays Character value[ Note that %c in Printf ]

char x = 'a';

printf("%c",x); // Display Result = a

Way 3 : Displays Next ASCII value[ Note that %d in Printf ]

char x = 'a' + 1 ;

printf("%d",x);

// Display Result = 98 ( ascii of 'b' )

Way 4 Displays Next Character value[Note that %c in Printf ]

char x = 'a' + 1;

printf("%c",x); // Display Result = 'b'

Way 5 : Displays Difference between 2 ASCII in Integer[Note %d in Printf ]

char x = 'z' - 'a';

printf("%d",x);

/\* Display Result = 25

(difference between ASCII of z and a ) \*/

Way 6 : Displays Difference between 2 ASCII in Char [Note that %c in Printf ]

char x = 'z' - 'a';

printf("%c",x);

/\* Display Result = ↓

( difference between ASCII of z and a ) \*/

**Tricks**:

1. In declaration of string size must be required to mention otherwise it gives an error.

Syntax

char str[]; // Invalid

char str[10]; // Valid

1. In declaration of the string size must be unsigned integer value (not -ve or zero value) which is greater than zero only.

Example

char str[]; // Invalid

char str[0]; // Invalid

char str[-1]; // Invalid

char str[10]; // Valid

Syntax

char variable\_name[SIZE];

char str[5];

1. In Initialization of the string if the specific number of character is not initialized it then rest of all character will be initialized with NULL.

Example

char str[5]={'5','+','A'};

str[0]; ---> 5

str[1]; ---> +

str[2]; ---> A

str[3]; ---> NULL

str[4]; ---> NULL

1. In initialization of the string we cannot initialized more than size of string elements.

Example

char str[2]={'5','+','A','B'}; // Invalid

1. In initialization of the string the size is optional in this case how many variable elements are initialized it, that array element will created.

Example

char str[]={'5','+','A','B'}; // Valid

sizeof(str) --> 4byte

1. When we are working with character array explicitly NULL character does not occupies any physical memory at the end of the character array.

Example

char str[]={'h','e','l','l','o'};

sizeof(str) --> 5byte

String data at the end of the string NULL character occupies physical memory.

Example

char str[]="hello";

sizeof(str) --> 6 byte

1. Strings are always enclosed within double quotes. Whereas, character is enclosed within single quotes in C.
2. Loop in C-string:

char s[100];

for (int i = 0; s[i]; ++i) { ... }

Quite useful (also avoids the strlen usage, that you could forget is O(n) and put on for condition.)

1. %s is not Required

Like scanf statement %s is not necessary while accepting string:-scanf("%s",name);

and here is gets( ) syntax which is simpler than scanf( ) –gets(name);

1. Spaces are allowed in gets( )

gets(name); - Whenever the above line encounters then interrupt will wait for user to enter some text on the screen. When user starts typing the characters then all characters will be copied to string and when user enters newline character then process of accepting string will be stopped.