Strings

Strings: Concepts, String Input/ Output functions, arrays of strings, string manipulation functions.

String: String is a set of characters that are enclosed in double quotes.

- In the C programming language, strings are created using one dimension array of character datatype. Every string in C programming language is enclosed within double quotes and it is terminated with NULL (\0) character.
- The termination character ('\0') is important in a string since it is the only way to identify where the string ends.

In C programming language, there are two methods to create strings and they are as follows...

1. Using one dimensional array of character datatype (static memory allocation)
2. Using a pointer array of character datatype (dynamic memory allocation)

- In C, strings are created as a one-dimensional array of character datatype. We can use both static and dynamic memory allocation.
- When we create a string, the size of the array must be one more than the actual number of characters to be stored.
- That extra memory block is used to store string termination character NULL (\0). The following declaration stores a string of size 5 characters.

Declaring Strings:

Syntax:

char string_name[size];

Example :

char name[10];

Initializing a String

- Strings can also be initialized at compile time and at run time.
- The process of initializing the strings at compile time is to write the string literal within double quotes.
- Compile time initialization is called as static initialization. Compile time initialization can be done in three ways. They are given below.

Type 1: char name[9] = "Hello";

H	e	l	l	o	\0	\0	\0	\0

Type 2: char stream[7] = {`H', `e', `l', `l', `o', `!', \0'};

Type 3: char str[] = "I like C.";

Initialization after Declaration

char s[10] = " Web Design";

is not the same as

char s[10];

s = "Web Design"; /*Illegal declaration. */

This generates a compiler error. The assignment operator cannot be used with a string already declared

2. Using a pointer array of character datatype (dynamic memory allocation)

char *s="Hello";

char *str;

str=(char *)malloc(5*sizeof(char)

String Constants versus Character Constants

- A string constant is a series of characters enclosed in double quotes (" "). On the other hand, a character constant is a character enclosed in single quotes (` `).
- When a character variable ch and a character array str are initialized with the same character,x, such as the following,

char ch = `x';

char str[] = "x";

One byte is reserved for the character variable ch, and two bytes are allocated for the character array str. The reason that an extra byte is needed for str is that the compiler has to append a null character to the array.

String I/O Functions:

This is also called as run time initialization. The integers, character and float data types are read by using functions such as scanf(), gets,etc. Similarly to read a string two functions are used. They are

1. scanf()
2. gets()

Reading Strings Using scanf ( )

To read strings using scanf() format specifier %s is used with the function scanf().For example

```
char name[10];
```

To input data in

to char array 'name', scanf function and %s is used as given below.

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

Note that the %s format specifier doesn't require the ampersand (&) symbol before the variable **name**. The major limitation of scanf() is that it reads until occurrence of first separator a white space characters such as space character, tab or new line and store it into given variable. For example if the input literal is "How are you?". Then the scanf statement will read only the word 'How' into the variable **name.**

To overcome this limitation, gets() and getchar() functions are used.

Reading Strings using gets( )

- The gets() functions reads the string from the keyboard including spaces until the enter key is not pressed.
- The getchar() function is used to get a single charater. The gets() and getchar() functions are defined in stdio.h library file. For example

```
char line[100];
```

```
printf("Enter a line:\n");
gets(line);
puts("Entered input is :\n");
puts(line);
```

The gets() function keeps reading characters from the standard input stream until a new line character or end-of-file is encountered. Instead of saving the new line character, the gets() function appends a null character to the array that is referenced by the argument to the gets() function.

Print strings:

To out put data of the given string two functions are used. They are

1) printf()
2) puts ()

Printing strings using printf()

The printf statement along with format specifier %s is used to print strings on the screen. The format %s can be used to display an array of characters that is terminated by the null character. For example

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

can be used to display the entire contents of the array name.

printf expects to receive a string as an additional parameter when it sees %s in the format string. This additional parameter

- Can be from a character array.
- Can be another literal string.
- Can be from a character pointer (more on this Pointer chapter).

- printf knows how much to print out because of the NULL character at the end of all strings. When printf() encounters a null\0, it stops printing.
- The string variable can be printed out with precision using printf() statement.
  For example

```
printf("%7.3s",name)
```

This specifies that only the first 3 characters have to be printed in a total field width of 7 characters & right justified in the allocated width by default.

2.7.2 Printing strings using puts( )

The puts function is a much simpler output function than printf for string printing. Prototype of puts is defined in stdio.h For example:

```
char sentence[] = "This is strings in C \n";
puts(sentence);
```

This Prints out:

This is strings in C

Consider a simple program to understand string input and output functions.

```
#include <stdio.h>
main()
{
char str[80] ; Char str2[45];
printf("Enter a string: ") ;
gets(str1) ;
printf("Enter another string: ") ;

scanf(“%s“,str2); /*Reads upto first white space character*/ Printf(“output of first string:\n”); printf("%s" , str1\n);
Printf(“output of second string:\n”);
```

```
puts("str2 ")
}
```

Result:

Input:

Enter a string: This is c language.

Enter another string: This is strings

Output:

Output of first string: This is c language

Output of second string: This

2.8 Built-in String Functions

C does not provide any operators for string. Because of this C has built in string functions in its library. C library supports a large number of string handling functions that can be used to perform many of the string manipulations. Some of the string functions are given below. To do all the operations described here it is essential to include string.h library header file in the program.

String Handling Functions in string.h

Function name	Description
strcpy(str1,str2)	copies str2 to str1 including the null (\0).
strcat(str1,str2)	Appends str2 to end of string str1

strlen(string)	Returns the length of string. Does not include the
	null(\0)
strcmp(str1,str2)	Compares str1 with str2 returns integer result.If
	str1<str2 returns negative integer.Returns zero
	when str1==str2,and
	Returns a positive integer when str1>str2.
strncpy(str1,str2,n)	It copies at most n characters of str2 to str1.If str2
	has fewer than n characters, it pads str1 with
	null('\o') characters.
strchr(string,char)	Locates the position of the first occurrence of char
	within string and returns the address of the
	characterif it finds. and null if not.For ex-(" Hello",
	'l')
strlwr()	converts all characters in a string from uppercase to
	lowercase.
strupr()	converts all characters in a string from lower case to
	uppercase.

strcpy()

C does not allow you to assign the characters to a string directly.For example as in the statement name="Java"; Instead use the strcpy() function. The strcpy() function works almost like a string-assignment operator. The syntax of the function is

strcpy(string1,string2);

Strcpy() function assigns the contents of string2 to string1.The string2 may be a character array variable or a string constant.

strcpy(Name,"Java");

strcpy(city1, city2);

This will assign the contents of the string variable city2 to the string variable city1.The size of the array city1 should be large enough to receive the contents of city2, if not compiler will lodge an error.

strcat() function:

The process of appending the characters of one string to the end of other string. Is called concatenation.The strcat() function joins two strings together. Sstrcat(string1,string2)

Here string1 & string2 are character arrays. When the function strcat is executed string2 is appended to string1.The string at string2 remains unchanged.

Example

strcpy(string1,"Web");

strcpy(string2,"Mining");

Printf("%s", strcat(string1,string2);

From the above program segment the value of string1 becomes webmining. The string at str2 remains unchanged as mining. The variation in this function is strn(str1,str2,n) .This concatenates the n integer letters to str1. For example,

strncat(stri,name,4);

This concatenates first 4 letters of string name to string stri.

strlen() function:

This function counts and returns the number of characters in a string. The length does not include a null character. The syntax is n=strlen(string); Where n is integer variable, which receives the value of length of the string.

Example:

int len;

len=strlen("Coral Draw");

Here the integer variable len will store the value 10 which is the length of given sting.

strcmp() function

In c the values of two strings cannot be directly compare in a condition like if(string1==string2). The strcmp() function is used for this purpose. This function returns a zerovalue if two strings are equal, or a non zero

number if the strings are not the same. This returns a negative if the string1 is alphabetically less than the second and a positive number if the string is greater than the second.

The syntax of strcmp() is given below:

Strcmp(string1,string2)

String1 & string2 may be string variables or string constants.

Example:

strcmp("Book","Book") will return zero because 2 strings are equal. strcmp("their","there") will return a 9 which is the numeric difference between ASCII 'i' and ASCII 'r'.

strcmp("The", "the") will return 32 which is the numeric difference between ASCII "T" & ASCII "t".

strcmpi() function

This function is same as strcmp() which compares 2 strings but not case sensitive. For example

strcmpi("THE", "the"); will return 0.

strlwr () function:

This function converts all characters in a string from uppercase to lowercase. The syntax is

strlwr(string);

For example:

strlwr("APPLICATION") converts the string to application

strupr() function:

This function converts all characters in a string from lower case to uppercase. The syntax is

strupr(string);

For example strupr(“application”) will convert the string to APPLICATION.

Programs

1. Write a program to find out the length of a given string without using the library function strlen().

```
#include <stdio.h>
void main()
{
char str[50];
char nul={‘\0’}
int len;
printf("\nENTER A STRING: ");
gets(str);
for(len=0; str[len]!=nul; len++);

printf("\n THE LENGTH OF THE STRING IS %d", len); }
```

Result:

Input:

ENTER A STRING: This is string in C

Output:

THE LENGTH OF THE STRING IS 19.

2) Write a program to find out the length of a given string using the library function strlen(). /*The "strlen()" function gives the length of a string, not including the NULL character at the end.*/

```
#include <stdio.h>
#include <string.h>
void main()
{
        char name[30] = "This is string in C";
        int len;
        len= strlen( name );
        printf( "Length of string <%s> is %d.\n", name ,len);
}
```

Output:

Length of string < This is string in C > is19.

3) Write a program to print the reverse of a given string.

```
#include <stdio.h>
#include <string.h>
void main()
{
        char ch[100];
        int i,len;
        printf("\n Enter a string: ");
        gets(ch);
        len= strlen(ch);
        printf("\n The string in the reverse order: ");
        for(i=len-1; i>=0; i--)
        printf("%c", ch[i]);
        getch();
}
```

Result:

Input

Enter a string: strings are interesting.

Output

The string in the reverse order: .gnitseretni era sgnirts

4)Write a program to check if a given string is palindrome or not.

```
#include <stdio.h>
#include <string.h>
#include <string.h>
void main()
{
        char a[100];
        int i,len, flag=0;
        printf("\n Enter a string: ");
        gets(a);
        len=strlen(a);
for(i=0;i<len; i++)
        {
                if(a[i]==a[len-i-1])
```

```
                flag=flag+1;
        }
        if(flag==len)
        printf("\n The string is palindrom");
        else
printf("\n The string is not palindrom");
    }
```

RESULT:

Input

Enter a string: madam

Output

The string is palindrome

5. Write a program to count the number of words in a given string. Two words are separated by one or more blank spaces.

```
#include <stdio.h>
#include <string.h>
void main()
{
        char a[100];
        int len, word, i =0;
        printf("\n Enter a string: ");
        gets(a);
        len=strlen(a);
for(i=0;i<len; i++)
        {
                if( a[i] !=' ' && a[i+1]==' ')
```

```
            word=word+1;
        }
    printf("\n There are %d words in the string", word);
    }
```

RESULT:

Input

Enter a string: Strings are very interesting.

Output

There are 4 words in the string

Array of Strings

A string is a 1-D array of characters, so an array of strings is a 2-D array of characters. Just like we can create a 2-D array of int, float etc; we can also create a 2-D array of character or array of strings. Here is how we can declare a 2-D array of characters.

Example:

```
char ch_arr[3][10] = {
                        {'s', 'p', 'i', 'k', 'e', '\0'},
                        {'t', 'o', 'm','\0'},
                        {'j', 'e', 'r', 'r', 'y','\0'}
                    };
```

The ch_arr is a pointer to an array of 10 characters or int(*)[10].

Therefore, if ch_arr points to address 1000 then ch_arr + 1 will point to address 1010.

From this, we can conclude that:

ch_arr + 0 points to the 0th string or 0th 1-D array.
ch_arr + 1 points to the 1st string or 1st 1-D array.
ch_arr + 2 points to the 2nd string or 2nd 1-D array.

In general, ch_arr + i points to the ith string or ith 1-D array.