WEEK - 8	
1	Strings: Read & Write
2	"String.h" Predefined Functions
3	Sample C Programs

Strings Handling

A collection of characters is called as string. In the absence of string data type to store strings in C we have to simulate string using an array of characters. Any group of characters (except double quote sign) defined between double quotation marks is a constant string.

Character strings are often used to build meaningful and readable programs. The common operations performed on character strings are:

- Reading and writing strings
- Combining strings together
- Copying one string to another
- Comparing strings for equality
- Extracting a portion of a string

Declaring a String

An array of characters is also a collection of characters. So an array of characters is used to represent a string in C. For example, to store a name, which may be up to 50 characters, you would declare a string as follows:

```
/* name can contain up to 50 characters */
char name[50];
/* to store 10 names where each name can contain 20 characters */
char names[10][20];
```

How String is stored?

A string is stored as an array in the memory. However, when you declare a string of 20 characters, it doesn't always contain 20 characters. It may contain only 15 characters. So to identify the end of the actual characters in a string C is storing a null character (with ASCII code 0) at the end of the string.

Null character is written as '**\0**'. It is the character for ASCII code 0. As no other character contains ASCII code 0, when we encounter a null character in a string we can stop taking characters after that character.

name

J	O	H	N	S	O	N	P	\0	...

Note: Though you declare a string as of 50 characters, actually you can store only 49 characters. This is because one byte is used to store null character.

Initializing String Variables:

Character arrays may be initialized when they are declared. C permits a character array to be initialized in either of the following two forms:

```
char text[9] = "WELCOME"; / char city[9] = "NEW YORK";
char text[9] = {'W', 'E', 'L', 'C', 'O', 'M', 'E'};
char text[9] = {'W', 'E', 'L', 'C', 'O', 'M', 'E', '\0'};
```

Input/output of Strings

You can read a string from keyboard. C has provided a conversion character **%s** for inputting and outputting string. You can read a string using **scanf()** & print using **printf()**.

The following code snippet is to read name of the user and display the same.

```
char name[30];
printf("What's your name: ");
scanf("%s",name);
printf("Welcome %s",name);
```

There are few important points that you have to understand about reading strings using scanf()

- You must not precede string variable with & (ampersand). The reason will be evident to you once you understand pointers. Giving & before the string variable is a logical error. Compiler may not complain about it but program will not work.
- You can read the data from keyboard only up to first whitespace character (space or tab). That means if I want to enter my name, I can enter only Johson and not complete name "Johnson P D". This is because of the fact that scanf() function assumes the end of the input for a field once it encounters a whitespace character.

Note: While using %s to read a string with scanf(), do not precede variable with & symbol.

Another important point about strings is, all standard functions such as scanf() and printf() take care of null character at the end of the string.

For instance, when you enter a string using scanf(), the function will automatically put a null at the end of the string. And in the same way while you are printing, printf() will take characters until null character is encountered.

String I/O

In order to read and write string of characters the functions gets() and puts() are used gets() function reads the string and puts() function takes the string as argument and writes on the screen.

(1) **gets()**

This function reads a sequence of characters entered through the keyword and is useful for interactive programming. Since a string has no predetermined length gets() needs a way to know to stop. It reads character until it reaches a new line character. To terminate input at the keyword, press the 'Enter' key.

```
Syntax: gets(variable);
```

(2) **puts()**

The function puts() writes a string argument onto the screen. The puts() can only output a string of characters. It takes less space and runs faster than printf().

```
Syntax: puts(variable);
```

Example: Program to print the accepted string.

```
/*Program to print accepted string*/
#include<stdio.h>
#include<conio.h>
main()
{
      char ch[30];
      clrscr();
      printf("Enter String: ");
      gets(ch);
      printf("\nEntered String: ");
      puts(ch);
}
```

Output:

```
Enter String: C is a good language.
Entered String: C is a good language.
```

The gets() and puts() functions offer simple alternatives with regard to the use of scanf() and printf() which are of reading and displaying strings respectively. These executes faster than printf() and scanf() due to the non-use of format specifies and occupies less space.

fflush(stdin):

The fflush(stdin) is a standard library function, which is used to clear the buffer, which it receives as a parameter.

```
Syntax: fflush(<std.buf.ref.>);
```

Program to read integer and character

```
/*Program to read integer and character*/
#include<stdio.h>
main()
```

```
{
      char ch;
      int a;
      clrscr();
      printf("Enter Integer value: ");
      scanf("%d",&a);
      fflush(stdin);
      printf("Enter character value: ");
      ch=getchar();
      printf("\nEntered integer = %d \tEntered character = %c",a,ch);
}
```

Output:

```
Enter Integer value: 22
Enter character value: k
Entered integer = 22    Entered character = k
```

String "string.h" Library

C provides a set of functions that performs operations on strings. These functions take a string and take actions such as reversing content of string, converting the string to upper case and return length of the string. The following are functions from string library. All functions are declared in **string.h** header file.

strlen() function

This function counts and returns the number of characters in a string. The length does not include a null character.

```
Syntax: len = strlen(string);
      where len is integer variable which receives the value of length
      of the string.
```

Example: Program to find length of the string using strlen() function.

```
#include<stdio.h>
#include<string.h>
main()
{
      char name[100];
      int length;
      printf("Enter a string: ");
      gets(name);
      length = strlen(name);
      printf("\nNumber of characters in the string is %d",length);
      getch();
}
```

```
Output:
Enter a string: peter
Number of characters in the string is 5
```

strlwr() function

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

```
Syntax: strlwr(string);

For Example: strlwr("EXFORSYS");
converts to exforsys.
```

strupr() function

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

```
Syntax: strupr(string);

For Example: strupr("exforsys");
converts to EXFORSYS.
```

Example: Program to convert lowercase to uppercase and uppercase to lowercase.

```
#include<stdio.h>
#include<string.h>
main()
{
      char name[100];
      clrscr();
      printf("Enter a string: ");
      gets(name);
      printf("\nUppercase String: ");
      puts(strupr(name));
      strlwr(name);
      printf("\nLowercase String: ");
      puts(name);
      getch();
}
Output:
Enter a string: peter
Uppercase String: PETER
Lowercase String: peter
```

strrev() function

This function reverses the characters in a string.

```
Syntax: strrev(string);

For Example: strrev("exforsys");
           reverses the characters to sysrofxe.
```

strcpy() function

C does not allow to assign the characters to a string directly as in the statement name="exforsys"; instead use the strcpy() function found in most compilers.

```
Syntax: strcpy(string1,string2);

For Example: strcpy(name,"exforsys");
          String exforsys is assigned to the string called name.
```

strcmp() function

In C we cannot directly compare the value of 2 strings in a condition like if(string1==string2)

Most libraries however contain the strcmp() function, which returns a zero if 2 strings are equal, or a non zero number (>0 or <0) if the strings are not same.

```
Syntax: strcmp(string1,string2);

string1 and string2 may be variables or constants. Some computers
return a negative value if string1 is alphabetically less than the
second and a positive if the string1 is greater than the second.

For Example:
strcmp("their","there");
     will return -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' and ASCII 'T'.

strcmp("hello","hello");
will return 0 as two strings are equal.
```

strcmpi() function

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

```
Syntax: strcmpi(string1,string2);

For Example:
strcmpi("the","THE");
will return 0 as it ignores case of the string.
```

Example: Program to check whether given string is palindrome or not.

```
#include<stdio.h>
#include<string.h>
main()
{
     char name1[20],name2[20];
     clrscr();
     printf("Enter a string: ");
     gets(name1);
     strcpy(name2,name1);
```

```
      strrev(name2);
      printf("Reversed String is ",name2);
      if(strcmp(name1,name2) == 0)
            printf("String is Palindrome");
      else
            printf("String is Not Palindrome");
      getch();
}
Output:
Enter a string: peter
Reversed String: retep
String is not Palindrome
```

strcat() function

When you combine two strings, you add the characters of one string to the end of other string. This process is called concatenation. The strcat() function joins 2 strings together. It takes the following form:

```
Syntax: strcat(string1,string2);
```

When the function strcat is executed string2 is appended to string1, the string at string 2 remains unchanged.

For Example:

```
      strcat(st1,"hello ");
      strcat(st2,"world");
      printf("%s",strcat(st1,st2));
From the above program segment of value of st1 becomes "hello world".
The string at st2 remains unchanged as "world".
```

Example: Program to sort an array of 10 strings

```
#include<stdio.h>
#include<string.h>
main()
{
      char names[10][20];
      int i,j;
      char temp[20];
      clrscr();
      printf("Enter array of strings:\n");
      for(i=0;i<10;i++)
            gets(names[i]);
      for(i=0;i<9;i++)
            for(j=i+1;j<10;j++)
            if(strcmp(names[i],names[j])>0)
            {
```

```
                strcpy(temp,names[i]);
                strcpy(names[i],names[j]);
                strcpy(names[j],temp);
          }
     printf("Sorted array of strings: \n");
     for(i=0;i<10;i++)
          puts(names[i]);
}
```

Output:

```
Enter array of strings:
santosh
anand
praneeth
manohar
naidu
hari
rajesh
aditya
julie
pavan
Sorted array of strings:
aditya
anand
hari
julie
manohar
naidu
pavan
praneeth
rajesh
santosh
```

strstr() function

This function returns the address (pointer) in string1 where string2 is starting in string1.

```
Syntax: strstr(string1,string2);
```

strchr() function

This function returns the address (pointer) of first occurrence of character **ch** in string.

```
Syntax: strchr(string,ch);
```

Sample C Programs for String Operations without using String Functions

Example: Program to read a line of text

```
#include<stdio.h>
main()
{
      char line[81],character;
      int c=0;
      clrscr();
      printf("Enter text. Press Enter at end\n");
      do {
            character = getchar();
            line[c] = character;
            c++;
      }
      while(character!='\n');
      c=c-1;
      line[c] = '\0';
      printf("\n%s\n",line);
      getch();
}
```

Output:

```
Enter text. Press Enter at end
This program reads a string and prints.

This program reads a string and prints.
```

Example: Program to Accept a string and display string in uppercase.

```
#include<stdio.h>
main()
{
       char st[20];
       int i;
       clrscr();
       /* accept a string */
       printf("Enter a string: ");
       gets(st);
       /* display it in upper case */
       for ( i = 0 ; st[i] != '\0';  i++)
              if ( st[i] >= 'a'  &&  st[i] <= 'z' )
                     putch( st[i] - 32);
              else
                     putch( st[i]);
       getch();
}
```

Output:

```
Enter a string: c programming
C PROGRAMMING
```

Example: Program to write string using %s format.

```
#include<stdio.h>
main()
{
      char country[15]="United Kingdom";
      clrscr();
      printf("\n\n");
      printf("-------------------\n");
      printf("|%15s|\n",country);
      printf("|%5s|\n",country);
      printf("|%15.6s|\n",country);
      printf("|%.6s|\n",country);
      printf("|%15.0s|\n",country);
      printf("|%.3s|\n",country);
      printf("|%s|\n",country);
      printf("-------------------\n");
      getch();
}
```

Output:

```
------------------
| United Kingdom|
|United Kingdom|
|         United|
|United|
|               |
|Uni|
|United Kingdom|
------------------
```

Example: Program to find length of the string.

```
#include<stdio.h>
main()
{
      char str[20];
      int i = 0;
      clrscr();
      printf("\nEnter any string: ");
      gets(str);
      while (str[i] != '\0')
```

```
            i++;
      printf("\nLength of string: %d", i);
      getch();
}
```

Output:

```
Enter any string: ANIL NEERUKONDA
Length of string: 15
```

Example: Program to accept a string and display it in reverse.

```
#include<stdio.h>
main()
{
      char st[20];
      int i;
      clrscr();
      printf("Enter a string: "); /* accept a string */
      gets(st);
      /* get length of the string */
      for ( i = 0 ; st[i] != '\0';  i++);
      /* display it in reverse order */
      for ( i -- ; i >= 0 ; i --)
            putch(st[i]);
      getch();
}
```

Output:

```
Enter a string: HELLO WORLD
DLROW OLLEH
```

Example: Program to Concatenate of 2 Strings

```
#include<stdio.h>
#include<conio.h>
main()
{
      char string1[30], string2[20];
      int i, length=0, temp;
      printf("Enter the Value of String1: \n");
      gets(string1);
      printf("\nEnter the Value of String2: \n");
      gets(string2);
      for(i=0; string1[i]!='\0'; i++)
            length++;
      temp = length;
      for(i=0; string2[i]!='\0'; i++)
      {
            string1[temp] = string2[i];
```

```
            temp++;
      }
      string1[temp] = '\0';
      printf("\nThe concatenated string is:\n");
      puts(string1);
      getch();
}
```

Output:

```
Enter the Value of String1:
C Language

Enter the Value of String2:
is good

The concatenated string is:
C Language is good
```

Example: Program to Copy one string to another string.

```
#include <stdio.h>
#include <conio.h>
main()
{
      char string1[20], string2[20];
      int i;
      clrscr();
      printf("Enter the value of STRING1: \n");
      gets(string1);
      for(i=0; string1[i]!='\0'; i++)
            string2[i]=string1[i];
      string2[i]='\0';
      printf("\nThe value of STRING2 is:\n");
      puts(string2);
      getch();
}
```

Output:

```
Enter the value of STRING1:
c programs are cool

The value of STRING2 is:
c programs are cool
```

Example: Program to Compare two given strings.

```
#include<stdio.h>
main()
{
```

```
      char string1[15],string2[15];
      int i,temp = 0;
      clrscr();
      printf("Enter the string1 value:\n");
      gets(string1);
      printf("\nEnter the String2 value:\n");
      gets(string2);
      for(i=0;string1[i] != '\0';i++) {
            if(s1[i] == s2[i])
                  temp = 1;
            else
                  temp = 0;
      }
      if(temp==1)
            printf("Both strings are same.");
      else
            printf("Both strings not same.");
      getch();
}
```

Output1:

```
Enter the string1 value:
c programs

Enter the String2 value:
cprograms

Both strings not same.
```

Output2:

```
Enter the string1 value:
c programs

Enter the String2 value:
c programs

Both strings are same.
```

Example: Program to print alphabet set in decimal and character form.

```
#include<stdio.h>
main()
{
      char c;
      clrscr();
      printf("\n\n");
```

```
      for(c=65;c<=122;c=c+1)
      {
            if(c>90 && c<97)
                  continue;
            printf("%5d - %c",c,c);
      }
      printf("\n");
      getch();
}
```

Output:

```
65 - A   66 - B   67 - C   68 - D   69 - E   70 - F   71 - G   72 - H
73 -I   74 - J   75 - K   76 - L   77 - M   78 - N   79 - O   80 - P
81 - Q   82 - R   83 - S   84 - T   85 - U   86 - V   87 - W   88 - X
89 - Y   90 - Z   97- a   98 - b   99 - c  100 - d  101 - e  102 - f
103 - g  104 - h  105 - i  106 - j  107 - k  108 - l  109 - m  110 - n
111 - o  112 - p  113 - q  114 - r  115 - s  116 - t  117 - u  118 - v
119 - w  120 - x  121 - y  122 - z
```