Handling of Character Strings

As it was mentioned earlier character is stored internally as an integer. For convenience the **C** compiler as well as the programmer treat certain values as characters and manipulate them accordingly. Like an integer array, a group of characters can be stored in a character array. These character arrays are often called *strings*. ***A string is an array of characters.*** There is no string built-in data type in **C**. But we can declare string as an array of characters. To recognize a character array, it should end with a **null** character ('\0'). For example, the string **SCIENCE** would be stored as

'S'	'C'	'I'	'E'	'N'	'C'	'E'	'\0'

The length of a string is the number of characters it contains excluding null character. Hence, the number of locations needed to store a string is one more than length of string.

In this example, the length of the string is 7. Since, the first null character in a string is assumed to be the end of a string, more exactly the length of a string can be defined as the number of non-null characters between the first character and first null character in a string.

Declaring and initializing string variables

The general form of declaration of string variable is

char string-name[size];

where, string-name is the name of a string and size is the maximum number of characters the string-name can contain.

Example:

```
char name[30];
char designation[25];
char date_of_join[8];
```

String variables can be initialized at the time of declaration.

Example:

```
char name[30] = "Millennium";
char designation[25] = "Computers";
```

A string can also be initialized at the time of declaration in the following ways.

```
char name[30] = "Millennium";
```

or

```
char name[30] ={'M','i','l','l','n','n','i','u','m','\0'};
```

or

```
char name[] = "Millennium";
```

or

```
char name[] = {'M','i','l','l','e','n','n','i','u','m','\0'};
```

Note that, when initialize a character array by listing its elements, we must explicitly specify the null terminator. When a character string is assigned to a character array, **C** adds the null character at the end of string automatically.

A two-dimensional array can be initialized to a list of strings. For example,

```
char day[7][15] = {   "Monday",
                      "Tuesday",
                      "Wednesday",
                      "Thursday",
                      "Friday",
                      "Saturday",
                      "Sunday"
                  };
```

Reading and writing strings

The scanf(), printf() function is used with %s with format specification to read and print a string.

Example:

```
char str[30];
scanf("%s",str);
printf("%s",str);
```

In the case of reading strings, the ampersand (&) is not required before the string variable name. As mentioned earlier, one of the limitations of the scanf() function is that it is not capable of holding multiword strings, even though it can read them.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  char line[80];
  clrscr();
  printf("Enter a line\n");
  scanf("%s",line);
  printf("The entered line is:%s",line);
  getch();
}
```

While executing this program, if you have typed as
This is a line

Then the output is
The entered line is: *This*

To overcome this type of situation, we can use **gets()** function in place of scanf() function. Similarly **puts()** function can be used in place of printf() function.

The gets() function will continue to read and hold a string of input until it encounters a new line character ('\n'). It will replace the new line character with a null character ('\0'), thus creating a string. The above program can be written using gets() and puts() function as….

```
#include<stdio.h>
#include<conio.h>
void main()
{
   char line[80];
   clrscr();
   printf("Enter a line\n");
   gets(line);
   puts("The entered string is:");
   puts(line);
   getch();
}
```

5.1 String handling Functions: string.h

Every **C** compiler provides a large set of string handling library functions, which are contained in the header file ***string.h***

The following table shows some of the functions available in string.h header file.

Function	Meaning
strcat()	String concatenate. Append one string to another. First character of string2 overwrites null character of string1.
strlen()	Returns the length of the string not counting the null character.
strlwr()	Converts a string to lower case.
strupr()	Converts a string to upper case.
strcpy()	Copies a string into another.
strcmp()	Compares two strings
strrev()	Reverses a string.

Note: The functions deal with strings (a string of characters) ending with null character '\0'. To use these functions, including the following line in the program.

#include<string.h>

5.2 strcat() function

The **strcat()** function concatenates the source string at the end of the target string. For example "**Computing**" and "**Techniques**" on concatenation would result in string "**ComputingTechniques**". The general form is **strcat(string1, string2)**; strong2 appends to string1 and the first character to string2 overwrites null character of first string1. This function returns the first argument i.e., string1. The string2 remains unchanged.

Example:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
   char s1[30],s2[15];
   clrscr();
   printf("Enter String1:");
   gets(s1);
   printf("Enter String2:");
   gets(s2);
   printf("The entire string is:%s",strcat(s1,s2));
   getch();
}
```

Output

```
Enter String1:Millennium
Enter String2:Software Solutions
The entire string is:MillenniumSoftware Solutions
```

5.3 strcmp() function:

strcmp(string1, string2);

strcmp() function compares two strings to find out whether they are same or different. The two strings are compared character by character until there is a mismatch or end of one of the strings is reached, whichever occurs first. If the two strings are same, strcmp() returns a value 0. If they are not same, it returns the numeric difference between the **ASCII** values of the first non-matching characters. That is, it returns less than 0 if string1 is less than string2, and greater than 0 if string1 is greater than string2.

Example:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
   char s1[25],s2[25];
   int c;
   clrscr();
   printf("Enter string1:");
   gets(s1);
   printf("Enter string2:");
   gets(s2);
   c=strcmp(s1,s2);
   if(c>0)
   printf("String1 > String2");
   else if(c<0)
```

```
   printf("String2 > String1");
   else
   printf("Both are equal");
   getch();
}
```

Output

```
Enter String1:abc
Enter String2:ABC
String1 > String2
```

5.4 strcpy() function

strcpy(String1, String2);

The **strcpy()** function is used to copy the character string from String2 to String1. This function returns the result string String1 the String2 remains unchanged. String2 may be a character array or a string constant.

Example:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
   char s1[15],s2[15];
   clrscr();
   printf("Enter String1:")
   gets(s1);
   printf("The String2 is:%s",strcpy(s2,s1));
   getch();
}
```

Output

```
Enter String1:Millennium
The String2 is:Millennium
```

5.5 strlen() function

This function counts the number of characters present in a string. The counting ends at the first null character.

strlen (String1);

The **strlen()** function returns the length of the argument String1 excluding the null character. The argument may be a string constant.

Example:

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

```
   char str[30];
   clrscr();
   printf("Enter a string:");
   gets(str);
   printf("The length of a string is:%d",strlen(str));
   getch();
}
```

Output

```
Enter a string: Millennium Software Solutions
The length of a string is:29
```

5.6 strupr(), strlwr(), strrev() functions

The **strupr()** function is converted the string into upper case and **strlwr()** function is converted the string into lower case and **strrev()** function prints the entire string in reverse order.

Example:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
   char str[15];
   clrscr();
   printf("Enter a string:");
   gets(str);
   printf("The upper case string is:%s",strupr(str));
   printf("\nThe lower case string is:%s",strlwr(str));
   printf("\nThe reverse string is:%s",strrev(str));
   getch();
}
```

Output

```
Enter a string:millennium
The upper case string is:MILLENNIUM
The lower case string is:millennium
The reverse string is:muinnellim
```