LECTURE NOTE 22

FUNDAMENTALS OF STRINGS

A string is a series of characters treated as a single unit. A string may include letters, digits and various special characters such as +, -, *, / and $. String literals or string constants in C are written in double quotation marks as follows:

"1000 Main Street" (a street address)

"(080)329-7082" (a telephone number)

"Kalamazoo, New York" (a city)

In C language strings are stored in array of char type along with null terminating character '\0' at the end.

When sizing the string array we need to add plus one to the actual size of the string to make space for the null terminating character, '\0'.

Syntax:

char fname[4];

The above statement declares a string called fname that can take up to 3 characters. It can be indexed just as a regular array as well.

fname[]={'t','w','o'};

character	t	w	o	\0
ASCII code	116	119	41	0

Generalized syntax is:-

char str[size];

when we declare the string in this way, we can store size-1 characters in the array because the last character would be the null character. For example,

char mesg[10]; can store maximum of 9 characters.

If we want to print a string from a variable, such as four name string above we can do this.

e.g., *printf("First name:%s",fname);*
We can insert more than one variable. Conversion specification %s is used to insert a string and then go to each %s in our string, we are printing.

A string is an array of characters. Hence it can be indexed like an array.

char ourstr[6] = "EED";

– ourstr[0] is 'E'

– ourstr[1] is 'E'

– ourstr[2] is 'D'

– ourstr[3] is '\0'

– ourstr[4] is '\0' – ourstr[5] is '\0'

'E'	'E'	'D'	\0	'\0'	'\0'
ourstr[0]	ourstr[1]	ourstr[2]	ourstr[3]	ourstr[4]	ourstr[5]

Reading strings:

If we declare a string by writing

char str[100];

then str can be read from the user by using three ways;

1. Using scanf() function
2. Using gets() function
3. Using getchar(), getch(), or getche() function repeatedly

The string can be read using scanf() by writing
scanf("%s",str);
Although the syntax of scanf() function is well known and easy to use, the main pitfall with this function is that it terminates as soon as it finds a blank space. For example, if the user enters Hello World, then str will contain only Hello. This is because the moment a blank space is encountered, the string is terminated by the scanf() function.

Example:

```
char str[10];
printf("Enter a string\n");
scanf("%s",str);
```

The next method of reading a string a string is by using gets() function. The string can be read by writing

```
gets(str);
```

gets() is a function that overcomes the drawbacks of scanf(). The gets() function takes the starting address of the string which will hold the input. The string inputted using gets() is automatically terminated with a null character.

Example:

```
char str[10];
 printf("Enter a string\n");
 gets(str);
```

The string can also be read by calling the getchar() repeatedly to read a sequence of single characters (unless a terminating character is encountered) and simultaneously storing it in a character array as follows:

```
int i=0;

char str[10],ch;

getchar(ch);

while(ch!='\0')

{

   str[i]=ch;       // store the read character in str

    i++;

   getch(ch);    // get another character

 }

str[i]='\0';       // terminate str with null character
```

Writing string

The string can be displayed on screen using three ways:

1. Using printf() function
2. Using puts() function
3. Using putchar() function repeatedly

The string can be displayed using pintf() by writing

printf("%s",str);

We can use width and precision specification along with %s. The width specifies the minimum output field width and the precision specifies the maximum number of characters to be displayed. Example:

printf("%5.3s",str);

this statement would print only the first three characters in a total field of five charaters; also these three characters are right justified in the allocated width.

The next method of writing a string is by using the puts() function. The string can be displayed by writing:

puts(str);

It terminates the line with a newline character ('\n'). It returns an EOF(-1) if an error occurs and returns a positive number on success.

Finally the string can be written by calling the putchar( ) function repeatedly to print a sequence of single characters.

```
int i=0;
char str[10];
while(str[i]!='\0')
{
    putchar(str[i]);     // print the character on the screen
     i++;
 }
```

Example: Read and display a string

```
#include<stdio.h>
#include<conio.h>
void main()
{
 char str[20];
 clrscr();
 printf("\n Enter a string:\n");
gets(str);
 scanf("The string is:\n");
puts(str);
        getch();  }
```

Output:

```
 Enter a string:
vssut burla
The string is:
vssut burla
```