Introduction to Scientific and Engineering Computation

(BIL 102E)

LECTURE 10

Character Strings

Strings

A string is a character array, with a null character (‘\0’) used to mark the end of the string.

For instance,

char str[7] = {‘H’, ’e’, ’l’, ’l’, ’o’, ’!’, ’\0’};

defines a character string.

It is possible to use double quotes to enclose a string constant. For example,

char str[7] = “Hello!”;

The size of the array is one more than the length of the string since the null character is added at the end!!!

Note that the compiler automatically adds the null character at the end of the string constant. We can also use:

char str[] = “Hello!”;

Reading and Writing Strings

gets() and **puts()** functions can be used to read character strings from the standard input stream and write character strings to standard output stream, respectively.

```
#include <stdio.h>
#define MAX_CHARS 80
main(){
  char str[]="Hello World!";
  char str_in[MAX_CHARS+1];
  puts(str);
  printf("Enter a string (at most %d characters please)\n",
            MAX_CHARS);
  gets(str_in);
  puts("You entered : ");
  puts(str_in);
}
```

Note that we reserve 1 more than maximum number of characters anticipated.

```
Hello World!
Enter a string (at most 80 characters please)
This is entered from the keyboard!
You entered :
This is entered from the keyboard!
```


It is also possible to use %s specifier with printf and scanf functions to read and write strings:

```
#include <stdio.h>
#define MAX_CHARS 80
main(){
  char str[]="Hello World!";
  char str_in[MAX_CHARS+1];
  printf("%s \n",str);
  printf("Enter a string (less than %d characters please)\n",
            MAX_CHARS);
  scanf("%s",str_in);
  printf("You entered : %s \n",str_in);
}
```

```
Hello World!
Enter a string (at most 80 characters please)
This is entered from the keyboard!
You entered : This
```

Note that scanf() does not read anything after a space character. So in order to read strings that can contain space characters you should use gets() function!!!

Strings as arguments to functions

Since strings are arrays of characters they can be passed to functions as arguments by the help of pointers. For example,

```
#include <stdio.h>
#define MAX_CHARS 80
void toUpperCase(char *str){
  char *ps;
  for(ps=str; *ps != '\0'; ps++)
    if(*ps >= 'a' && *ps <= 'z')
      *ps = *ps + 'A' - 'a';
}
main(){
  char str_in[MAX_CHARS+1];
  printf("Enter a string (at most %d characters please)\n",
             MAX_CHARS);
  gets(str_in);
  toUpperCase(str_in);
  printf("You entered : %s \n",str_in);
}
```

The end of the string is reached when the null character is reached.

If this is a lower case letter subtract 32 (97-65) from the value of the memory location!

Call toUpperCase() function so that the string entered is converted to upper case letters!

```
Enter a string (at most 80 characters please)
This is entered from the keyboard!
You entered : THIS IS ENTERED FROM THE KEYBOARD!
```


string.h

There are several readily defined functions to work with strings in the string library. These include,

strlen(str)

Finds the length of the null terminated string str (returns an integer).

strcpy(deststr, str)

Copies the contents of a null terminated string str to deststr. It is programmers responsibility to make sure that deststr can contain all the characters in str.

strcmp(str1, str2)

Compares the null terminated strings str1and str2. If both are equal 0 is returned. If otherwise a value less than or greater than 0 is returned depending on the lexicographical order of s1 and s2.

strcat(deststr, str)

Appends a copy of str to deststr. This is used for concatenation of character strings.

Examples:

```
#include <stdio.h>
#include <string.h>
#define MAX_CHARS 80
main(){
  char str_in[MAX_CHARS+1], str[MAX_CHARS+1];
  char *pch=str;
  printf("Enter a string (at most %d characters please)\n",
             MAX_CHARS);
  gets(str_in);
  printf("Length of the string : %d\n",strlen(str_in));
  strcpy(str, str_in);
  printf("str becomes after strcpy: %s\n",str);
  printf("The same memory can be reached by pch: %s\n",pch);
}
```

We could have used strcpy(pch, str_in) here.

```
Enter a string (at most 80 characters please)
This is entered from the keyboard!
Length of the string : 35
str becomes after strcpy: This is entered from the keyboard!
The same memory can be reached by pch: This is entered from the keyboard!
```



```
#include <stdio.h>
#include <string.h>
#define MAX_CHARS 80
main(){
  char str_in[MAX_CHARS+1], str[]="Turan";
  char *pch=str;
  printf("Enter your name :");
  gets(str_in);
  if(strcmp(str_in, str) == 0)
    printf("Your name is the same as mine 8=)\n");
  else if(strcmp(str_in, str)<0)
    printf("Your name is before my name in the telephone directory.\n");
  else
    printf("Your name is after my name in the telephone directory.\n");
}
```

Note that “Turan” is different than “turan” or “TURAN”.

```
Enter your name :Turan
Your name is the same as mine 8=)
```

```
Enter your name :Ahmet
Your name is before my name in the telephone directory.
```

```
Enter your name :Turgut
Your name is after my name in the telephone directory.
```

```c
#include <stdio.h>
#include <string.h>
#define MAX_CHARS 80
main(){
  char name[MAX_CHARS+1], surname[MAX_CHARS+1];
  char full_name[2*MAX_CHARS+2];
  printf("Enter your name : ");
  gets(name);
  printf("Enter your surname : ");
  gets(surname);
  strcpy(full_name, name);
  strcat(full_name," ");
  strcat(full_name, surname);
  printf("Hello %s!\n",full_name);
}
```

Note that the source string can be a constant string (BUT THE DESTINATION STRING CANNOT BE!)

```
Enter your name : Turan
Enter your surname : Soylemez
Hello Turan Soylemez!
```


More functions

It is also possible to specify the maximum operation size by using the following functions:

strncpy(deststr, str, n)

Similar to strcpy() except that at most n characters are copied. Note that strncpy() does not necessarily put a null character at the end of the copied part of the string. This is programmer's responsibility.

strncmp(str1, str2, n)

Similar to strcmp() except that at most n characters are compared.

strncat(deststr, str, n)

Similar to strcat() except that at most n characters (not counting the null character) from str are appended to deststr.


```c
#include <stdio.h>
#include <string.h>
#define MAX_CHARS 10
#define BUFLEN 255
main(){
  char buffer[BUFLEN+1], name[MAX_CHARS+1], surname[MAX_CHARS+1];
  char full_name[2*MAX_CHARS+2];
  printf("Enter your name : ");
  gets(buffer);
  strncpy(name, buffer, MAX_CHARS);
  if(strlen(buffer)>=MAX_CHARS)
          name[MAX_CHARS]='\0';
  printf("Enter your surname : ");
  gets(buffer);
  strncpy(surname, buffer, MAX_CHARS);
  if(strlen(buffer)>=MAX_CHARS)
          surname[MAX_CHARS]='\0';
  strcpy(full_name, name);
  strcat(full_name," ");
  strncat(full_name, buffer, 2*MAX_CHARS-strlen(name) );
  printf("Name : %s\n",name);
  printf("Surname : %s\n",surname);
  printf("Full Name : %s\n",full_name);
}
```

This program makes sure that the program works even if the user enters a long name or surname.

If MAX_CHARS characters are copied then put a null termination at the end.

```
Enter your name : Mustafa
Enter your surname : Turkyilmazoglu
Name : Mustafa
Surname : Turkyilmaz
Full Name : Mustafa Turkyilmazogl
```