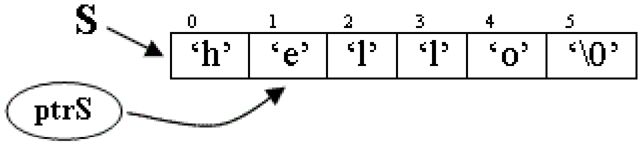
60-141 – Introduction to Programming II   Winter, 2017

Lab #6: Character Arrays and Strings

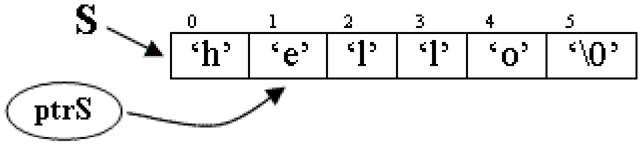
(Due at the beginning of the next lab period)

Objective: Learn to use and manipulate strings and arrays of characters.

In C, a string is simply defined as an array of characters terminated by the null character, or string delimiter, '\0'. If we want to store a string "hello", which has only 5 characters, we need an array of minimum size 6. For example the array S below that has been declared as char S[6]. The array index (or subscript) values are indicated above each box containing a single character in the figure below.



We can also create a pointer to S: char *ptrS = S. Then, if we apply pointer arithmetic and increment the pointer ptrS++ we can show that the pointer ptrS now points to the second item in the array with index 1 (refer to the figure below). Now we can refer to the letter 'e' by either using the pointer as *ptrS, or using the array index as S[1].



Lab Work to do:
Write a complete C language program called Lab6.c. The requirements and specifications for the program are listed below in Parts A-C. Read the entire set of lab instructions carefully before proceeding in order to appreciate and understand how to approach the design of your program. Apply both Top-Down and Bottom-Up aspects of design.

Your program must treat each Part separately, but it is not required to have a menu, nor to loop back to receive multiple inputs – just do what is asked of you!

Part A. Character array and string.

1. Declare a char array called buffer1 and initialize it to "this is the first buffer." using this method: {'t', 'h', 'i', 's', ' ', 'i', ... '\0'};
2. Declare a char array called buffer2 and initialize it to "this is the second buffer." using this method: "this is the second buffer";
3. Declare a char array called buffer3 of size 80 and leave it un-initialized.

4. Use the scanf function to initialize buffer3 from the keyboard. (Note: use %s as the formatting character for string). If you encounter problems, document them and work out solutions for them.
5. Now, use printf to print all the three buffers. (Again, use %s as the formatting character for string data).
6. Declare a pointer variable named pBuffer and initialize it to point to buffer3. (Use char * pBuffer).
7. Display the contents of buffer3 using pBuffer; do not use subscripts.
8. Advance the pointer pBuffer a few positions and display the rest of buffer3 (i.e. only from that position to the end of the string) using pBuffer.

Part B. String Manipulation: Reverse.

1. Write a function called Reverse that accepts as a parameter a char array (or char pointer) and returns a void type. The function reverses the contents of the array (or string) being passed. Example: the string "Hello" will be reversed to "olleH". You have to be careful about the '\0' symbol to keep it at the end of the array and watch out for odd and even length of strings.
2. Test your function and display the string (array) before and after it is reversed.
3. IMPORTANT: The function does NOT print the string in reverse; it actually reverses it in the memory.

Part C. String Tokenization

1. Write a function called ParseSentence that takes as input parameter a null (i.e. '\0') terminated string S, where S would contain an English sentence.
2. Assume that the delimiters are space, comma, semicolon and period.
3. The function extracts each word from the sentence (without any punctuation or spacing) and then prints one word per line.
4. The function returns a void type.

For example:

```
char str[] = "hello world, how are you today.";
ParseSentence(str);
```

would print the following:

```
hello
world
how
are
you
today
```

Summary of the lab requirements: You must write a single C program called Lab6.c that contains declarations for various char type data structures, and also contains definitions for functions Reverse() and ParseSentence(), following the requirements and specifications outlined in Parts A, B and C above. Don't forget to test your program with various inputs. Remember to fully document your programs.

EVALUATION OF WORK AND ATTENDANCE: Total 5 marks.
You need to show to your lab instructor the work you have completed for this lab assignment, generally in the form of a working program. The marks you will receive for the lab are made of two parts, the programming part and lab attendance. Do not email your work to the Instructor or to any teaching assistant.

Lab Work Mark: You will be evaluated based on your solution for the problem assigned, using the following scheme: You must attend the lab and show your work in order to earn any of the following marks.

0 mark = No appreciable and relevant work done.
1 mark = Incomplete code / does not compile, with no, little or invalid documentation.
2 marks = Complete running program with no, little or invalid documentation.
3 marks = Incomplete code / does not compile, with suitable documentation.
4 marks = Complete running program with suitable documentation (minor errors may be accepted).

Attendance Mark: You will receive 1 mark for attendance during the lab period.

IMPORTANT:
ASK QUESTIONS IF YOU GET STUCK, BUT DO YOUR OWN CODING. SUBMITTING WORK COPIED FROM OTHER STUDENTS WILL RESULT IN A MARK OF ZERO (0) FOR THE LAB WORK MARK.