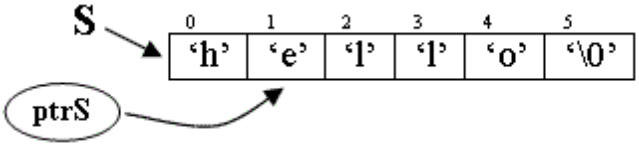
60-141
Intro to Algorithms & Programming II Winter 2014

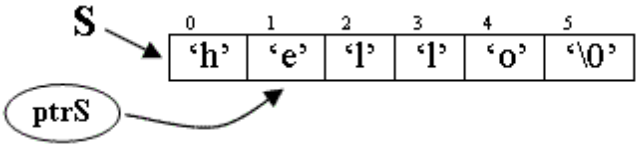
Lab #6: Character Arrays and Strings
(Due at the end of the lab period or beginning of the next)

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].



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: (Lab6.c)

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 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).'
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.
8. Advance the pointer pBuffer a few positions and display the rest of buffer3 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. 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 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.

For example:

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

would print the following:

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

EVALUATION:

You need to show your instructor the complete programs at the end of this lab, or at the beginning of your next lab. The marks you will receive for this lab are made of two parts: Lab work marks 8 and **attendance marks 2**. **Total 10 marks**.

Lab Work Mark: You will be evaluated based on your solutions for the problems based on the following scheme:

0 mark = No work done.
2 mark = Incomplete code / does not compile, with no/invalid documentation
4 marks = Complete running program with no/invalid documentation
6 marks = Incomplete code / does not compile, with proper documentation
8 marks = Complete running program with proper documentation

IMPORTANT:

ASK QUESTIONS IF YOU GET STUCK, BUT DO YOUR OWN CODE. ANY CODE SUSPECTED TO BE SIMILAR TO ANOTHER SUBMISSION WILL CAUSE BOTH SUBMISSIONS TO RECEIVE A ZERO MARK ON ALL LABS AND BE REPORTED FOR PLAGIARISM