Carleton University
Department of Systems and Computer Engineering
SYSC 2006 - Foundations of Imperative Programming - Fall 2013

Lab 5 - Character Strings

Objective

The objective of this lab is to write some C functions that operate on C strings.

Attendance/Demo

To receive credit for this lab, you must make a reasonable effort to complete the exercises and demonstrate the code you complete.

When you have finished all the exercises, call a TA, who will review the code you wrote. For those who don't finish early, a TA will ask you to demonstrate whatever code you've completed, starting about 30 minutes before the end of the lab period. Finish any exercises that you don't complete by the end of the lab on your own time. Also, you must submit your lab work to cuLearn by the end of the lab period.

General Requirements

For those students who already know C or C++: when writing the functions, do not use structs or pointers. They aren't necessary for this lab.

None of the functions you write should perform console input; i.e., contain `scanf` statements. None of your functions should produce console output; i.e., contain `printf` statements.

None of the functions you write are permitted to call the functions that are declared in `string.h` from the C standard library. Your functions are permitted to call any of the string functions we've defined (the ones with names starting with `CU_str`).

You have been provided with file main.c. This file contains the This file contains the four string processing functions that were presented in lectures, and incomplete implementations of four functions you have to design and code. It also contains a *test harness* consisting of functions that will test your code and a `main` function that calls these test functions. **Do not modify `main()` or any of the test functions.**

Instructions

1. Create a folder named Lab 5.
2. Launch Pelles C, and create a new project named CU_strings inside the Lab 5 folder. The project type must be Win32 Console program (EXE). You should now have a folder named CU_strings inside a folder named Lab 5. Check this.
3. Download file main.c from cuLearn. Move this file into your CU_strings folder. **You must**

also add main.c to your project: from the menu bar, select Project > Add files to project... In the dialogue box, select main.c, then click Open. An icon labelled main.c will appear in the Pelles C project window.

4. Open main.c.
5. Build the project. It should build without any compilation or linking errors.
6. Execute the project. The test harness will run. Read the console output carefully. The harness will report several errors while it tests the functions you'll complete for Exercises 1 through 4, which is what we'd expect, because you haven't started working on these functions.

Exercise 1

Write a function that returns the position of the first occurrence of a specified character in a string. The function prototype is:

```
int CU_strchr(const char str[], char ch);
```

Parameter `str` is a character array containing the string that the function searches. Parameter `ch` is the character the function searches for. The function returns the position (array index) of `ch` in `str`. If the character isn't found, the function returns -1.

The terminating null character is considered to be part of the string. As such, in this code fragment:

```
char s[] = "hello";
int posn = CU_strchr(s, '\0') ;
```

variable `posn` is assigned 5, because `str[5]` contains `'\0'` (the character we're searching for).

Exercise 2

Write a function that returns the position of the last occurrence of a specified character in a string. The function prototype is:

```
int CU_strrchr(const char str[], char ch);
```

Parameter `str` is a character array containing the string that the function searches. Parameter `ch` is the character the function searches for. The function returns the position (array index) of the last occurrence of `ch` in `str`. If the character isn't found, the function returns -1.

The terminating null character is considered to be part of the string. As such, in this code fragment:

```
char s[] = "hello";
int posn = CU_strrchr(s, '\0') ;
```

variable `posn` is assigned 5, because `str[5]` contains `'\0'`.

Note: your function can't simply search the character array from right to left. For example, this array contains the characters `'h'`, `e'`, `'l'`, `'l'`, `'o'`, `'\0'`, `'a'`, `'\0'`, `'l'` followed by one uninitialized character:

```
char s[10] = "hello";
s[6] = 'a';
s[7] = '\0';
s[8] = 'l';
// s[9] is uninitialized
```

The function should search for the last occurrence of the specified character in `"hello"`. The characters after the `'\0'` that terminates `"hello"` are ignored. Therefore,

`CU_strrchr(s, '\0')` should return 5, not 7 (the position of the null terminator for `"hello"` is 5).
`CU_strrchr(s, 'l')` should return 3, not 8 (the position of the last `'l'` in `"hello"` is 3)
`CU_strrchr(s, 'a')` should return -1, not 6 (`'a'` is not in `"hello"`)

Exercise 3

Write a function that compares at most a specified number of characters in two strings. The function prototype is:

```
int CU_strncmp(const char s[], const char t[], int count)
```

Parameter `count` is the maximum number of characters to compare. Characters that follow a terminating `'\0'` are not compared. The function returns a negative value if `s` < `t`, 0 if `s` equals `t`, and a positive value if `s` > `t`.

This function is somewhat similar to `CU_strcmp`, which was presented in one of the lectures. Note that your function must handle the case where the terminating null in one or both strings is reached before the function has compared `count` characters.

Exercise 4

Write a function that finds the location of the first occurrence in string `str1` of any of the characters that are part of string `str2`. The search does not include the terminating null character of either string. The function prototype is:

```
int CU_strpbrk(const char str1], const char str2[]);
```

Parameter `str1` contains the string to be scanned. Parameter `str2` contains the string consisting of the characters to be searched for.

The function returns the position (array index) of the first occurrence in `str1` of any of the characters

that are part of `str2`. If none of the characters in string `str2` is found in `str1` before the terminating null in `str1` is reached, this function returns -1.

Wrap-up

1. Remember to have a TA review and grade your solutions to the exercises before you leave the lab.
2. The next thing you'll do is package the project in a ZIP file (compressed folder). From the menu bar, select Project > ZIP Files... A Save As dialog box will appear. Click Save. Pelles C will create a compressed (zipped) folder named CU_strings.zip, which will contain copies of the the source code and several other files associated with the project. (The original files will not be removed). The compressed folder will be stored in your project folder (i.e., folder CU_strings).
3. Log in to cuLearn, click the Submit Lab 5 link and submit CU_strings.zip. After you click the Add submission button, drag the file to the File submissions box. After the icon for the file appears in the box, click the Save changes button. At this point, the submission status for your file is "Draft (not submitted)". You can resubmit the file by clicking the Edit my submission button. After you've finished uploading your file, remember to click the Submit assignment button. This will change the submission status to "Submitted for grading".

Challenge Exercise

Write a function that finds the first occurrence of the entire string `substr` in the string pointed to by `str`. The function prototype is:

```
int CU_strstr(const char str[], const char substr[]);
```

Parameter `str` contains the string to examine. Parameter `substr` contains the string to search for; that is, the sequence of characters to match. This function returns the position (array index) of the first occurrence of `substr` in `str`. If the entire sequence of characters in `substr` is not present in `str`, the function returns -1.

Examples:

```
CU_strstr("This is a simple string", "simple");
```

returns 10.

```
CU_strstr("This is a simple string", "sample");
```

returns -1.

Using the test harness provided to you as a starting point, write a function that tests your `strstr` function.

Posted: October 6, 2013.