Project [9]: Command Line arguments, Dynamic Memory allocation, and Strings

Due Tuesday, 12/4/2018, 11:59 pm

Project Goals

The goal of this project is to:

1. Get students familiar with command line arguments
2. Get students familiar with dynamic memory allocation
3. Get students familiar with arrays of pointers

Important Notes:

1. **Formatting:** Make sure that you follow the precise recommendations for the output content and formatting: for example, do not change the text in the first problem from "`Please enter a string of maximum 30 characters: `" to "`Enter string: `". Your assignment will be auto-graded and any changes in formatting will result in a loss in the grade.
2. **Comments:** Header comments are required on all files, for each function, and recommended throughout the rest of the program. Points will be deducted if no header/function are included.
3. **Restriction:** The use of `goto` statements anywhere within this program is prohibited. Points will be deducted if `goto` is used.

Problem 1

Write a program that asks the user to enter two strings (with a maximum length of 30) and then performs one of two possible operations provided as command line arguments to the program. The possible command line options and the associated operations are as follows:

- Option "`-i`" Operation: If given this option, the program should create a new string that is built by interspersing the two strings. The program will alternatively place one character at a time from each string into the new string. When one of the strings is exhausted, the program should copy the remaining characters from that string into the new string. This functionality should be implemented in a function called `intersperse`, which takes as parameters the two strings and returns a pointer to the newly created string. Your function should use dynamic memory allocation to generate the new string.

  For example, if string 1 is "`abcde`" and string 2 is "`1234567`", the resulting string should be "`a1b2c3d4e567`".

- Option "`-w`" Operation: If given this option, the program should create a new string that is built by concatenating the two given strings, in which there has been a '`*`' character inserted in between every character in those strings. There should be a '`*`' character between the two strings, but no such character at the end of string 2. This functionality should be implemented in a function called `widen_stars`, which takes as parameters the two strings and returns a pointer to the newly created string. Your function should use dynamic memory allocation to generate the new string.

  For example, if string 1 is "`abcde`" and string 2 is "`1234567`", the resulting string should be "`a*b*c*d*e*1*2*3*4*5*6*7`".

Your program should function as follows (items underlined are to be entered by the user):

```
> gcc combine_strings.c -o combine_strings
> ./combine_strings -i
Please enter a string of maximum 30 characters: abcde
Please enter a string of maximum 30 characters: 1234567
The combined string is: a1b2c3d4e567
```

Or

```
> gcc combine_strings.c -o combine_strings
> ./combine_strings -w
Please enter a string of maximum 30 characters: abcde
Please enter a string of maximum 30 characters: 1234567
The combined string is: a*b*c*d*e*1*2*3*4*5*6*7
```

Notes:

- You may not include the string library, but you may use the `strlength` and `strcomp` functions from the last project
- Before terminating, your program needs to de-allocate the space allocated for the newly created strings

Save your program as `combine_strings.c`

Problem 2

This program will build off the last project. You will write an interactive program for strings. The number of strings will be provided as a command line argument. The user will enter the length of the string they are about to enter and then the string itself. After the program has read in the specified number of strings it will print the strings out and then print a menu that will allow the user to either: find the length of any string they want, compare any two strings, copy any string into another, or copy any string into another.

At every iteration, your program should:

1. Print out the strings by first printing "`Your strings are:`" and then the strings in the format specified below
2. Print out a menu as follows:

```
Options:
1 - Find string length
2 - Compare strings
3 - Copy strings
4 - Concatenate strings
5 - Quit
Please enter your option:
```

Below is a description of what the program should do for each of the above options. (items underlined are to be entered by the user):

1. Find the length of any string. You will use the `strlength` function. For this option, the program should ask the user to enter the number of the string they want to find the length of, as follows:

```
Enter a string number: 1
The length of string 1 is: 5
```

2. Compare any two strings. You will use the `strcomp` function. For this option, the program should ask the user to enter the numbers of two strings and then print out which string comes first, as follows:

```
Enter the number of the first string: 4
```

```
Enter the number of the second string: 2
String 2 comes before string 4 alphabetically.
```

If the strings selected by the user are the same string, your program should output: "`The two strings are the same.`"

3. Copy any two strings. You will use the `strcopy` function. For this option, the program should ask the user to enter the numbers of two strings and then copy the source string into the destination string, as follows:
```
Enter the number of the source string: 3
Enter the number of the destination string: 6
```

4. Concatenate any two strings. Your function will use the `strconcat` function. For this option, your program should ask the user to enter the numbers of two strings and then add the source string to the end of the destination string, as follows:
```
Enter the number of the source string: 3
Enter the number of the destination string: 6
```

5. Exit the program. Your program must free any memory that was allocated before exiting.

Your program must use the four functions written in the last project but modified as necessary:

1. `strlength:` This function will take as a parameter a character pointer to a string. It will return the length of the string, not including the null terminator. This function is exactly the same as the last project.
2. `strcopy:` This function should take as parameters two character pointers to two strings (a source string and a destination string). First, the function must reallocate enough memory to hold the source string in the destination string. Then it will copy the source string into the destination string, including the null terminator. It will then return a pointer to the beginning of the destination string.
3. `strconcat:` This function should take as parameters two character pointers to two strings (a source string and a destination string). First, the function should reallocate enough memory to add the source string to the end of the destination string. Then it will add the source string to the end of the destination string. It will then return a pointer to the beginning of the destination string.
4. `strcomp:` This function will take as parameters two character pointers to two strings (string 1 and string 2). The function then compares the two strings and if the string 1 comes first alphabetically it returns 1. If the string 2 comes first alphabetically then it returns -1. If the strings are the same then the function returns 0. The function should compare the two strings one character at a time and return the appropriate value as soon as a difference is noticed between the strings. This function is exactly the same as the last project.

The program should function as follows (items underlined are to be entered by the user):
```
> gcc dynamic_strings.c -o dynamic_strings
> ./dynamic_strings 5
Enter the length of string 1: 5
Please enter string 1: first
Enter the length of string 2: 6
Please enter string 2: second
Enter the length of string 3: 5
Please enter string 3: third
Enter the length of string 4: 6
Please enter string 4: fourth
Enter the length of string 5: 5
Please enter string 5: fifth
Your strings are:
```

```
String number 1 - “first”
String number 2 - “second”
String number 3 - “third”
String number 4 - “fourth”
String number 5 - “fifth”
Options:
1 - Find string length
2 - Compare strings
3 - Copy strings
4 - Concatenate strings
5 - Quit
Please enter your option: 1
Enter a string number: 3
The length of string 3 is: 5
Your strings are:
String number 1 - “first”
String number 2 - “second”
String number 3 - “third”
String number 4 - “fourth”
String number 5 - “fifth”
Options:
1 - Find string length
2 - Compare strings
3 - Copy strings
4 - Concatenate strings
5 - Quit
Please enter your option: 2
Enter the number of the first string: 3
Enter the number of the second string: 4
String 4 comes before string 3 alphabetically.
Your strings are:
String number 1 - “first”
String number 2 - “second”
String number 3 - “third”
String number 4 - “fourth”
String number 5 - “fifth”
Options:
1 - Find string length
2 - Compare strings
3 - Copy strings
4 - Concatenate strings
5 - Quit
Please enter your option: 3
Enter the number of the source string: 2
Enter the number of the destination string: 5
Your strings are:
String number 1 - “first”
String number 2 - “second”
String number 3 - “third”
String number 4 - “fourth”
String number 5 - “second”
Options:
1 - Find string length
2 - Compare strings
3 - Copy strings
4 - Concatenate strings
5 - Quit
Please enter your option: 4
Enter the number of the source string: 4
```

```
Enter the number of the destination string: 1
Your strings are:
String number 1 - “firstfourth”
String number 2 - “second”
String number 3 - “third”
String number 4 - “fourth”
String number 5 - “second”
Options:
1 - Find string length
2 - Compare strings
3 - Copy strings
4 - Concatenate strings
5 - Quit
Please enter your option: 5
```

Notes:

- If the user enters an invalid menu option, the program should print “`Invalid Option.`” then it should print the menu again and ask for a valid option, as follows:
  ```
  Invalid Option.
  Your strings are:
  String number 1 - “first”
  String number 2 - “second”
  String number 3 - “third”
  String number 4 - “fourth”
  String number 5 - “fifth”
  Options:
  1 - Find string length
  2 - Compare strings
  3 - Copy strings
  4 - Concatenate strings
  5 - Quit
  Please enter your option:
  ```
- You may **NOT** include the string library. Inclusion of the string library will result in a grade of zero for this project. Use the functions from project 7 and make any necessary changes.
- You can assume the user will enter the size of each string correctly.
- In the last project you used static memory allocation and assumed a max string size of 20 characters. This is no longer the case. You must dynamically allocate memory for all your strings and it is your responsibility to free the memory before the program ends.
- For this program, you must take the number of strings in as a command line argument. This number will be stored as a string not as an integer. To get the integer value of a string use the `atoi(argv[1])` function from `stdlib`. This function takes as a parameter a string and gives the equivalent integer value.

Save your program as `dynamic_strings.c`

“Did you ever hear the tragedy of Darth Malloc The Wise? I thought not. It's not a story Java would tell you. It's a C legend. Malloc was a C function so powerful and so wise that it could use machine language to allocate memory itself. It had so much knowledge of machine code it could create dynamically sized arrays. Unfortunately, it taught free everything it knew and one day free deallocated all of its memory. Ironic. It could allocate memory for others, but not itself.”

Code Wars: Episode III – Revenge of the C

Grading Rubric

Grading will be done for each problem as follows:

Correctly-named file	5%
Header comment	2%
Program compiles	5%
Correctly-reading data from terminal	18%
Correct result printed	20%

Submission details

To submit your project, you will have to use the submission script. You do this by either:

1. Working on an ECC machine
2. Working on the provided VMware
3. Secure Copying your files (See Mac Support for information)

To Submit your project:

- Have a directory called "project9"
- Save your *.c files in that directory
- To submit: **(don't type the '>' symbols)**

  > cd project9
  > submit

The submission script copies all files in the current directory to our directory. You may submit as many times as you like before the deadline, we only keep the last submission.

Academic Honesty

Academic dishonesty is against university as well as the system community standards. Academic dishonesty includes, but is not limited to, the following:

Plagiarism: defined as submitting the language, ideas, thoughts or work of another as one's own; or assisting in the act of plagiarism by allowing one's work to be used in this fashion.

Cheating: defined as (1) obtaining or providing unauthorized information during an examination through verbal, visual or unauthorized use of books, notes, text and other materials; (2) obtaining or providing information concerning all or part of an examination prior to that examination; (3) taking an examination for another student, or arranging for another person to take an exam in one's place; (4) altering or changing test answers after submittal for grading, grades after grades have been awarded, or other academic records once these are official.

Cheating, plagiarism or otherwise obtaining grades under false pretenses constitute academic

dishonesty according to the code of this university. Academic dishonesty will not be tolerated and penalties can include cancelling a student's enrolment without a grade, giving an F for the course, or for the assignment. For more details, see the University of Nevada, Reno General Catalog.