ITSC 2181 – Introduction to Computer Systems
Spring 2024
Module 05 – Unit 2: Lab

Objectives

- Practice how to write small C programs, including identifying and correcting `C` compiler error messages and warnings.
- Practice the use of functions from the **string library** (`string.h`)
- Practice the use of **string processing** in C.
- Practice the use of **file I/O** in C.
- Practice the use of **command-line arguments**.

General Instructions

- Please do not write your email or user ID anywhere in the program. To identify your code, you may use your UNC Charlotte 800#
- In this lab, you will write a few short C programs. Each program needs to be in its own source code file, i.e., a file with the *`.c`* extension.
- **You need to test your code thoroughly before submitting it**.
- **To earn any credit, a program has to compile.** *You may comment-out lines that have errors to obtain partial credit for work done.*
- Programs need to compile cleanly to receive full credit, i.e., they do not produce any errors or warnings.
- It is essential that you **do your own work.**
  - **Do not use any external resources** (Internet, AI, friends, etc.). All cases of cheating will be taken very seriously.
  - **If you need help, please ask the instructor, TA/IA or CCI Tutoring center.**

Setup / Prep. Work

Before you start the lab, make sure that you have done the following prep work:

1. Read section 2.8. I/O in C (Standard and File) of your textbook. Make sure you understand the code to do the following:
   a. Create formatted output using printf.
   b. Read text files.
   c. Use **`fgets`** and **`fscanf`** to process input.

2. Read subsection 2.9.2. Command Line Arguments of your textbook. Make sure to you understand the code examples posted in *Canvas*:
   a. C Programming - 17 - Advanced C Features.pdf (lecture slides)
   b. cmd_line_args.c (code samples and demonstrations)

Program: Restaurant Supplies (100 points)

Write a program named **`lab5.c`** that does the following:

1. Opens a text file for processing. The file contains a list of supplies purchased by a restaurant.

   a. The **file name** is provided in the command-line arguments.

      For example:

      ```
      ./a.out customer_data.txt
      ```

   b. You cannot hard-code the file name in the submitted version of your code; however, you may use an initialized string during testing.

   c. See the *Canvas* assignment for a sample file.

2. If the file does not exist or cannot be opened for any reason, the program needs to print the following message:

   ```
   Unable to open the input file.
   ```

3. The file contains a list of purchases, one on each line. The data includes the name of the item, the quantity purchased and the cost per unit. For example:

```
Cream 12 1.39
Sorbet 17 5.40
Cookies 40 3.89
```

4. The program reads the entire contents of the file, one item at a time, until the end of the file is reached.

5. As the program reads a new line of data, it prints the item's information.
    a. The output also needs to include a subtotal (see sample below).
    b. The output needs to be formatted just as shown in the sample run below.

6. Once all the data is read, the program displays the total number of items purchased and the total cost. The output needs to be formatted just as shown in the sample run below.

7. When the program is done processing the file, it closes it appropriately.

Implementation Details:

1. The program must use command-line arguments for the file name.

2. The program must include the following two functions. Note that code which does not use these functions will not earn any credit.

    ```
    float parse_data(char string[]);

    void pretty_print(char item[], int total, float cost);
    ```

    The `parse_data()` function takes one line of input and extracts the information about each item. This function returns the subtotal for the item, based on the cost per unit and total units purchased.

    The `pretty_print()` function takes information about one item and displays it in a format that matches the sample output (see below).

Sample Run:

```
Item: Cream      Quantity: 12   Item Cost: $ 1.39   Subtotal: $ 16.68
Item: Sorbet     Quantity: 17   Item Cost: $ 5.40   Subtotal: $ 91.80
Item: Cookies    Quantity: 40   Item Cost: $ 3.89   Subtotal: $155.60
```

```
Item: Vinegar    Quantity:  5   Item Cost: $ 3.29   Subtotal: $ 16.45
Item: Beef       Quantity:  4   Item Cost: $10.79   Subtotal: $ 43.16
Item: Avocado    Quantity: 40   Item Cost: $ 0.79   Subtotal: $ 31.60
Item: Celery     Quantity: 30   Item Cost: $ 1.98   Subtotal: $ 59.40
Item: Broccoli   Quantity: 34   Item Cost: $ 1.89   Subtotal: $ 64.26
Item: Spinach    Quantity: 20   Item Cost: $ 1.49   Subtotal: $ 29.80
Item: Garlic     Quantity: 27   Item Cost: $ 0.99   Subtotal: $ 26.73
Item: Mushrooms  Quantity:  5   Item Cost: $ 3.78   Subtotal: $ 18.90
Item: Potatoes   Quantity: 11   Item Cost: $ 5.99   Subtotal: $ 65.89
Item: Carrots    Quantity: 14   Item Cost: $ 1.99   Subtotal: $ 27.86

------------------------------------------------------
Total Items: 13       Total Cost: $648.13
```

Submission Instructions

1. Create a folder (directory) on your computer.
2. Name the folder `ITSC_2181_M05_U2_Lab_`***`student-id`***
   Replace *student-id* with your UNCC student ID (800#), e.g., *8001231234*
3. Download and copy your program (source code) files into this folder.
4. You should consider keeping the files for every lab in a separate folder (directory) from your other course materials.
5. Submit the source code (`.c` files) of your programs via *Canvas*.

Grading Rubric

- This lab is worth a total of **100 points**.
- **Do your own work. Do not use any external resources** (Internet, friends, etc.). All cases of cheating will be taken very seriously. **If you need help, please ask the instructor, TA/IA or CCI Tutoring center.**
- Programs need to compile to earn any credit. *You may comment-out lines that have errors to obtain partial credit for work done.*
- Your work will be graded on three (3) major components: Logic and flow of program, output, and formatting/organization. Refer to the following table for details.

Logic and Flow of Program - 60%	
Fully Correct and code compiles without errors.	Full Credit
Minor Errors and code compiles without errors OR some required functionality is missing.	75% Credit
Major Errors and/or code compiles with warnings OR significant portions of the required functionality are missing.	50% Credit
Completely incorrect/missing/does not compile.	No Credit
Output - 30%	
Fully Correct and matches formatting and layout of sample output provided.	Full Credit
Minor Errors.	75% Credit
Major Errors.	50% Credit
Completely incorrect output.	No credit
Formatting/Organization of Code - 10%	
Code is clear, easy to read and formatter according to the guidelines. Whitespace has been used appropriately, including indentation and blank lines. Comments are used when needed.	Full Credit
Needs minor improvement.	75% Credit
Needs major improvement.	50% Credit
No formatting/organization at all.	No Credit

Additional Deductions

- Code that produces warnings: -10% per program
- Incorrectly named files: -2 points per file
- *Students who cheat on any course assignment, lab, test or other activity will have their course grade reduced by one letter grade, regardless of the activity's point value, if it is their first offense at UNC Charlotte.*