Credit Card Project, Fall 2025

Tong Yi

Contents

Warning:

1. This is copyrighted materials; you are not allowed to upload to the Internet.
2. Our project is different from similar products in Internet.
   (a) Ask help only from teaching staff of this course.
   (b) Use solutions from artificial intelligence like ChatGPT or online tutoring websites like, but not limited to, chegg.com, violates academic integrity and is not allowed.

1 Goals

In this project, we will learn how to read and analyze a Comma-Separated-Values (CSV) file, process a string, and formatted outputs.

We work with Tab-Separated-Values (TSV) files in **Lab 3**. The difference between CSV and TSV files is the separator. In CSV, the separator is comma symbol, while in TSV, the separator is a tab. If a column data contains spaces, we should use CSV file.

2 Task A: Add the data in a Text File with Only One column

Create a file named `data.txt` with the following contents. On Mac/Linux, you can use VS Code, TextEdit, Vim, or Emacs. On Windows, you can use VS Code, Notepad, Notepad++, or Vim. The file `data.txt` should contain only one column of double-precision numbers. Example:

```
109.19
81.87
30.6
```

In Task A, do the following,

1. Create a file named **add.cpp**.
2. In the program, prompt the user to enter the name of the file to read.
3. Open the specified file.
4. Read all the numbers from the file.
5. Calculate the sum of these numbers.
6. Print the result.

Here is a sample output for the above `data.txt`.

```
Enter a file name: data.txt (with return key)
sum = 221.66
```

Submit `add.cpp` to gradescope. Note that the grading script generated random double numbers to test. As a result, your output will be different in each running.

3 Task B: Read a CSV file and Calculate its Total

A credit card report has four columns: Date, Description, Category, and Amount.

In the following example, date is August 6, 2025, description is XYZ Clinic, where the service was performed, category is Healthcare, and amount is 10 dollars.

Date	Description	Category	Amount
August 6, 2025	XYZ Clinic	Healthcare	$10.00

1. Create a sample CSV file as follows. Call it `spending.csv`.

```
Date,Description,Category,Amount
1/3/25,Con Edison,Utilities,53.33
1/6/25,Target,Shopping,58.54
1/7/25,Musical,Entertainment,286.37
1/9/25,Walmart,Shopping,326.52
1/14/25,Con Edison,Utilities,66.31
1/17/25,Costco,Shopping,387.94
1/20/25,United Airline,Travel,87.41
```

```
1/21/25,Costco,Groceries,94.95
1/21/25,Fish and you,Dining,335.55
1/23/25,Delta,Travel,485.41
2/8/25,Dim Sum,Dining,162.74
2/13/25,Con Edition,Utilities,396.31
2/17/25,American Airline,Travel,76.21
2/23/25,Emblem Premium,Healthcare,104.02
2/26/25,Chicken Soup,Dining,320.29
3/11/25,National Grid,Utilities,125.71
3/16/25,Target,Shopping,307.82
3/17/25,Lion King,Entertainment,392.2
3/27/25,Costco,Shopping,462.65
4/7/25,United Airline,Travel,321.48
4/9/25,Con Edison,Utilities,223.47
4/11/25,CVS Pharmacy,Healthcare,85.82
4/30/25,NYP Hospital,Healthcare,119.7
5/3/25,Dim Sum,Dining,451.93
5/3/25,Con Edison,Utilities,319.39
5/11/25,Chicken Soup,Dining,116.43
5/12/25,Walmart,Groceries,492.52
5/16/25,Fish and you,Dining,316.55
5/20/25,National Grid,Utilities,52.84
5/20/25,CVS Pharmacy,Healthcare,474.15
5/25/25,Northwell,Healthcare,313.67
6/1/25,NYP Radiology,Healthcare,329.79
6/8/25,Target,Shopping,76.76
6/18/25,Chicken Soup,Dining,421.75
6/25/25,United Airline,Travel,101.32
6/26/25,CVS Target,Healthcare,370.16
6/27/25,Dim Sum,Dining,229.79
7/3/25,NYP Hospital,Healthcare,474.65
7/7/25,Broadway Show,Entertainment,115.05
7/9/25,Target,Shopping,270.84
7/15/25,American Airline,Travel,197.29
7/18/25,Dim Sum,Dining,154.92
7/19/25,Con Edison,Utilities,215.9
7/20/25,Emblem Premium,Healthcare,262.37
7/21/25,Mama Mia,Entertainment,279.81
7/22/25,Costco,Shopping,149.81
7/23/25,Con Edison,Utilities,83.27
7/24/25,National Grid,Utilities,253.08
7/25/25,Walmart,Groceries,348.82
7/28/25,Dim Sum,Dining,255.16
```

Listing 1: spending.csv file

(a) The first line is column header.

(b) Each record, represented by one row in the table, records date, description, category and amount of a credit card service.

(c) Give the above data file a meaningful name, say `spending.csv`. Warning: if you test your source code in **onlinegdb**, after you upload source code to the server, you need to upload a data file as well. However, you may need to rename the data file as `spending.txt`, since onlinegdb does not recognize a file whose suffix is `csv`.

2. Name the source code as `read_csv.cpp`. You need to implements codes for the following steps.

   (a) Enter a file name for the data.

   (b) Calculate the sum of all data in column Amount.

   (c) Print out the sum.

3. Here is a sample input/output for the above data, where "Enter a csv file:" is a prompt, and spending.csv (with return key) is input from a user, and output is `sum = 12408.77`.

```
Enter a csv file: spending.csv
sum = 12408.77
```

4. To correctly read a file's contents, we must first understand its structure. This includes identifying introductory information, such as column headers, and knowing the meaning of each column. This process is similar to reading console input, but the data source is a file rather than user input from console.

3.1 Print Number in Fixed Decimal Numbers

Money is shown in two decimal numbers. We do the following.

1. Include `iomanip` library by `#include <iomanip>`

2. Use the following statement.

```
//Assume variable total is properly declared and initialized.
std::cout << std::fixed << std::setprecision(2) << total << std::endl;
```

   (a) `std::fixed` displays floating-point numbers in fixed-point notation rather than scientific notation. For example, 1234.5 prints as 1234.50 instead of 1.2345e+03.

   (b) With `std::fixed`, `std::setprecision(2)` sets the number of digits after the decimal point.

4 Task C: Read a CSV file and Sum Up Entries in a Time Range

Write a C++ program named `search_by_date.cpp` that reads credit card transaction data from a CSV file, processes date formats, and calculates the total spending within a specified date range.

1. Name the source code as `search_by_date.cpp`.

2. Enter a CSV file name. It records credit card transactions.

   The first row represents column headers.

   `Date,Description,Category,Amount`

   Each following row should contain:

   (a) Date (in m/d/yy format, e.g., 8/3/25)

   (b) Description of the transaction

   (c) Category (e.g., Food, Utilities, Entertainment)

   (d) Amount (a number)

3. Input Date Range

   Ask the user to enter a start date and an end date. Dates entered by the user may also be in m/d/yy format.

4. Standardize Date Format

   (a) To make date comparisons easier, define the following function in your program:
   string convert(string date);
   Function behavior:
   Input: A date in m/d/yy format (e.g., `"8/3/25"`).
   Output: The same date in mm/dd/yyyy format (e.g., `"08/03/2025"`).

   (b) Warning: without calling convert function, we cannot use `<` to compare two dates in m/d/yy format. For example, `"8/17/25"` is before `"8/5/25"` in dictionary order. Here is why.

   i. The first two characters of both strings are `'8'` and `'/'`.

   ii. The third character of `"8/17/25"` is `'1'`, while the third character of `"8/5/25"` is `'5'`.

   iii. Since `'1'` is before `'5'` in a dictionary, `"8/17/25"` is before `"8/5/25"`. That is, `"8/17/25" < "8/5/` which is not what we want.
   By contrast, after converting `"8/17/25"` to `"08/17/2025"`, and `"8/5/25"` to `"08/05/2025"` respectively, we have `"08/05/2025" < "08/17/2025"`.

   (c) By contrast, when mm/dd/yyyy format is adopted, `"08/05/2025"` is before `"08/17/2025"`.
   Here is a **link** to illustrate the above explanation.

```cpp
//Code link: https://onlinegdb.com/DeRu0AZtK
#include <iostream>
#include <string>
using namespace std;

int main() {
    string date = "8/17/25"; //m/d/yy format
    string date2 = "8/5/25";
    bool bComp = date < date2;
    cout << boolalpha << bComp << endl; //print "true" without
       quotes
```

```
    //when boolalpha is put before << bComp
    //displays true when bComp is true,
    //or false when bComp is false.

    //Without boolalpha, display 1 when bComp is true,
    //or 0 when bComp is false.

    string date3 = "08/17/2025"; //mm/dd/yyyy format
    string date4 = "08/05/2025";
    bComp = date3 < date4;
    cout << boolalpha << bComp << endl; //print "false" without
       quotes
    return 0;
}
```

(d) In short, to provide flexibility in date input, we allow m/d/yy format. However, to compare dates correctly, we need to define `convert` function.

5. Calculate Total Amount.

   (a) Read each transaction from the file.

   (b) Convert the transaction date to mm/dd/yyyy using convert function. Include the transaction in the total if its date is between the start date and end date (inclusive).

   (c) Display the total amount.

   Example

   Suppose the following CSV file is named sample.csv.

```
Date,Description,Category,Amount
8/2/25,Coffee,Food,4.50
8/3/25,Electric Bill,Utilities,120.75
8/17/25,Movie,Entertainment,15.00
```

   Program interaction: prompts are shown in red, and highlighted text indicates user input.

```
Enter a csv file name: sample.csv
Enter start date (m/d/yy): 8/1/25
Enter end date (m/d/yy): 8/5/25

8/2/25,Coffee,Food,4.50
8/3/25,Electric Bill,Utilities,120.75

Total amount: 125.25
```

5 Task D: Calculate Monthly Total

You are given credit card transactions covering at most 12 months. Write code to calculate the total spending for each month.

Store these totals in an array of size 12, where:
index 0 represents January
index 1 represents February
...
index 11 represents December

5.1 Use an Array to Store Monthly Total

Suppose we have transactions in a CSV file.

```
Date,Description,Category,Amount
1/15/25,Costco,Grocery,120.50
1/25/25,Con Edison,Utilities,89.75
2/2/25,Gas,Travel,45.00
3/5/25,Movie,Entertainment,15.00
```

Array after processing:

```
Index    Month    Total Spending
0        January       210.25
1        February       45.00
2        March          15.00
...  ...
11       December        0.00
```

5.2 Key Steps

1. Name your file `monthly_total.cpp`.
2. Declare an array of type double with size 12. This array saves monthly total.
3. Initialize all elements of the array to 0.0.
4. While processing the file:
   (a) Read each record.
   (b) Extract the month from the date.
   (c) Add the transaction amount to the corresponding element of the array.
   Hint: Think about the relationship between the month number (1–12) and the index of the corresponding array element (0–11).

Here is a sample input/output for Listing 1.

```
Enter a csv file name for credit card transactions: spending.csv
MON       TOTAL
Jan     2182.33
Feb     1059.57
Mar     1288.38
```

```
Apr      750.47
May     2537.48
Jun     1529.57
Jul     3060.97
Aug        0.00
Sep        0.00
Oct        0.00
Nov        0.00
Dec        0.00
```

6 Task E: Calculate Monthly Category Total

6.1 Goal

In this task, do the following:

1. Name source code as `monthly_category_total.cpp`.
2. Enter the name of a CSV file to read.
3. List all available categories in dictionary (alphabetical) order. Label them starting from 0 (e.g., the first category is 0, the second is 1, and so on).
4. Choose one category by its label.
5. Calculate and print the monthly totals for that category — i.e., the total amount spent in that category for each month.

6.2 Sample Output

Contents of spending2.csv are as follows.

```
day,store,category,cost
1/3/25,Con Edison,Utilities,53.33
1/6/25,Target,Shopping,58.54
1/9/25,Walmart,Shopping,326.52
1/14/25,Con Edison,Utilities,66.31
1/17/25,Costco,Shopping,387.94
1/20/25,United Airline,Travel,87.41
1/23/25,Delta,Travel,485.41
2/13/25,Con Edition,Utilities,396.31
2/17/25,American Airline,Travel,76.21
3/11/25,National Grid,Utilities,125.71
3/16/25,Target,Shopping,307.82
3/27/25,Costco,Shopping,462.65
```

Explanation of the data:

There are three categories in this example: **Utilities**, **Shopping**, and **Travel**. After sorting them in alphabetical order, the categories become: **Shopping**, **Travel**, and **Utilities**.

Since different files may contain different categories, you should:

1. Read the file.
2. Put all unique categories into an array. We assume that a file has at most 20 different categories. Hence, the capacity of the array of categories is 20.
3. Sort the array in ascending (alphabetical) order.
4. The program should display a list of categories, indexed sequentially starting from 0. For instance, Shopping would be labeled 0, Travel 1, and Utilities 2.

   To pass gradescope, the format should be a label followed immediately by a dot symbol (.), then a category name. For example, "0.Shopping" (without quotes) in one line. The actual label differs from file to file, since different files have different categories.
5. Next, the user will be prompted to enter the numerical label of the category they wish to analyze. Upon providing a valid label, the program will calculate and display the total monthly expenses for that chosen category. For example, if a user selects label 0 (Shopping):

   (a) In January, there are three transactions under Shopping. Their amounts are 58.54, 326.52, and 387.94. The total for Shopping in January is:

   $$58.54 + 326.52 + 387.94 = 773.00$$

   (b) In March, there are two transactions under Shopping. Their amounts are 307.82 and 462.65. The total for Shopping in March is:

   $$307.82 + 462.65 = 770.47$$

   (c) No other transactions under Shopping in the above file.

   (d) Here is a sample run to illustrate the above analysis. The bold red text represents program prompts, the highlighted text represents user inputs, and the remaining text shows the outputs.

```
Enter a csv file name: spending2.csv
select one of the following categories
0.Shopping
1.Travel
2.Utilities
choose a number in [0, 2]: 0
Enter a choice from 0 to 2: 0
Month   Shopping TOTAL
Jan              773.00
Feb                0.00
Mar              770.47
Apr                0.00
May                0.00
```

```
Jun                0.00
Jul                0.00
Aug                0.00
Sep                0.00
Oct                0.00
Nov                0.00
Dec                0.00
```

(e) Here is another sample run on the same data file, with the input category label being 1 (Travel).

```
Enter a csv file name: spending2.csv
select one of the following categories
0.Shopping
1.Travel
2.Utilities
choose a number in [0, 2]: 1
Month      Travel  TOTAL
Jan             572.82
Feb              76.21
Mar               0.00
Apr               0.00
May               0.00
Jun               0.00
Jul               0.00
Aug               0.00
Sep               0.00
Oct               0.00
Nov               0.00
Dec               0.00
```