**Project 1**

Customer Report

**CIS-17a 48983**

**Name: Brian Ramirez**

**Date: 10/25/2015**

**Introduction:**

Title: Customer Report

This is a program that uses an input file to load arrays with customer information. The customer information consists of the customer’s name, customer ID, and the customer’s balance. Inside the input file every odd line has a customer’s name and after each of those lines there is the corresponding customer’s ID and balance. This program can help people get a sense of what it looks like when businesses make a customer report.

**Summary:**

Project size: 252 lines

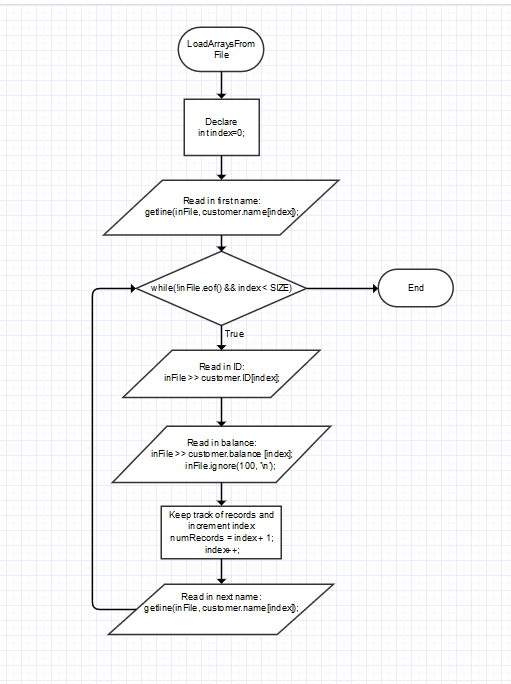
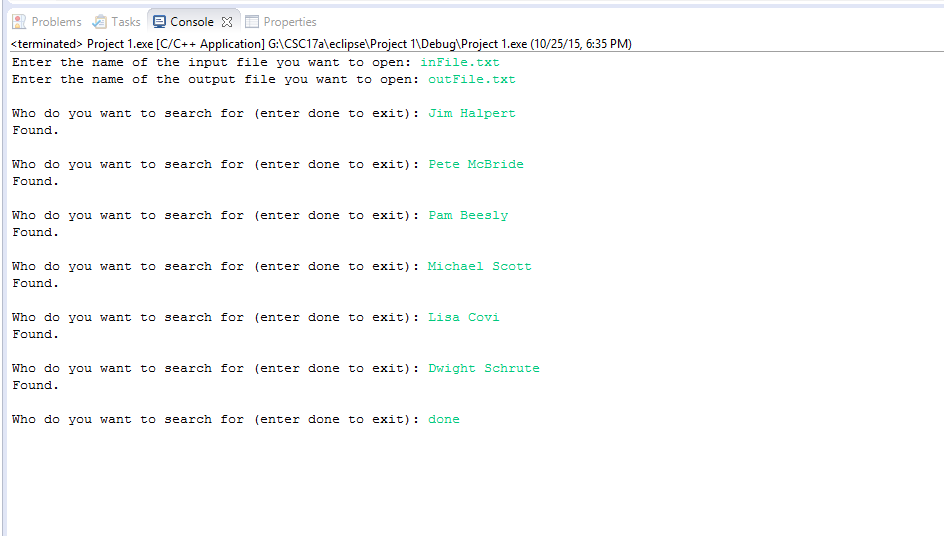
Number of variables: about 20

This program includes many concepts from the class textbook, which we have reviewed in class. The program took me a little less than a week to complete, because I kept running into errors. I am not as satisfied with this program as I thought I would be. This is not a game so it does not contain a lot of logic. However, it was still a good project that allowed me to practice some of the concepts in the textbook.

**Description:**

Input/Output:

In this program the main input that will be entered by the user is the name of the person that the user is searching for.

****

Flowchart:

Pseudocode:

This following pseudocode is for the LoadArraysFromFile function.

Index=0

get customer name from input file

while it is not the end of the file and array

{

Read in customer ID

Read in customer balance

Ignore(100, ‘\n’)

Number of records = index + 1

Increment index

Get the next customer name from input file

}

Major Variables:

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Variable Name | Description | Location |
| ifstream | inFile | Input file | main() |
| ofstream | outFile | Output file | main() |
| const int | SIZE | Size of array | main() |
| Customer | customer | Holds customer info | main() |
| Customer | cstmr | Holds matching customer info | main() |
| string | inFileName | Holds input file name | main() |
| string | outFileName | Holds output file name | main() |
| string | inputName | Name that the user enters | main() |
| int | numRecords | Holds number of records | main() |
| int | match | Holds position of the matched name | main() |
| int | inputMatch | Holds number of matching names | main() |
| int | count | Counter for loop | main() |
| string | stop | Sentinel value to stop program | main() |
| int | index | Holds element number for array | LoadArraysFromFile()  And SearchForMatch() |
| int | postion | Holds position of matching name | SearchForMatch() |
| bool | found | Boolean value to tell if name is found | SearchForMatch() |
| float | total | Accumulates total balance | PrintReportToFile() |
| float | AVG | Holds average balance | PrintReportToFile() |

C++ Concepts:

|  |  |  |
| --- | --- | --- |
| Chapter | Keywords | Example |
| 2 | System libraries | #include <fstream> |
| cout statements | cout<<”Enter name of input file”; |
| strings | string inFileName; |
| 3 | cin statements | cin >> inFileName; |
| Mathematical expressions | AVG = total / inputMatch; |
| cin.ignore() | cin.ignore(100, ‘\n’); |

|  |  |  |
| --- | --- | --- |
| 4 | If/else | if (match == -1) |
| Relational operators | if (match == -1) |
| 5 | For loop | for (count = 0; count < inputMatch; count++) |
| Increment operator | Index++ |
| While loop | while(index < size && !found) |
| 6 | Function prototypes | int SearchForMatch (string[], int, string); |
| Calling a function | PrintReportToFile(outFile, cstmr, count); |
| 7 | arrays | string source[] |
| Arrays as function parameter | int SearchForMatch (string[], int, string); |
| 9 | Pointers | string \*name; |
| Dynamic memory | cstmr.ID = new int [SIZE]; |
| 10 | string type | string source[] |
| 11 | Structured data | struct Customer  {  string \*name;  int \*ID;  float \*balance;  }; |
| 12 | File operations | inFile.close(); |

Reference:

1. Gaddis textbook
2. <http://www.cplusplus.com/>

Program:

// Including libraries

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

// Declaring structure

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* This structure contains all

\* the variables that are tied

\* to the customer. It contains a

\* string for the customer's name,

\* an int for the customer's ID, and

\* a float for the customer's balance.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct Customer

{

string \*name;

int \*ID;

float \*balance;

};

// Function prototypes

void LoadArraysFromFile (ifstream&, Customer, int&, const int);

int SearchForMatch (string[], int, string);

void PrintReportToFile (ofstream&, Customer, int);

int main()

{

// Declare variables

const int SIZE = 70; // Size of the array

Customer customer; // Holds customer information

Customer cstmr; // Holds matching customer information

ifstream inFile; // Input - The input file that holds names, IDs, and balances

string inFileName; // Input - Holds the name of the input file

string outFileName; // Input - Holds the name of the output file

ofstream outFile; // Output - The output file that will display the report

string inputName; // Input - Name that user is searching for

int numRecords; // Holds the number of records from inFile.txt

int match; // Holds the position of the matched name

int inputMatch; // Holds number of matching names

int count = 0; // Counter for loop

string stop = "done"; // Sentinel value to stop program

// Dynamically allocating arrays

customer.name = new string[SIZE];

customer.ID = new int[SIZE];

customer.balance = new float[SIZE];

cstmr.name = new string[SIZE];

cstmr.ID = new int [SIZE];

cstmr.balance = new float[SIZE];

// Prompt user for file names

cout << "Enter the name of the input file you want to open: ";

cin >> inFileName;

cout << "Enter the name of the output file you want to open: ";

cin >> outFileName;

cin.ignore(100, '\n');

cout << endl;

// Load the arrays from inFile.txt

//Opening input file

inFile.open(inFileName.c\_str());

//Calling on function LoadArraysFromFile

LoadArraysFromFile(inFile, customer, numRecords, SIZE);

//Closing input file

inFile.close();

// Prompt user to search for a name

cout << "Who do you want to search for (enter done to exit): ";

getline(cin, inputName);

// A while loop that does not end until user enters "done"

while (!(inputName == stop))

{

// Calling function SearchForMatch

match = SearchForMatch (customer.name, SIZE, inputName);

// If match = -1, then name is not found

if (match == -1)

{

cout << inputName << " was not found." << endl;

cout << endl;

}

// If match is not -1 then name is found

// and the matching customer info is

// transfered to another set of arrays

else

{

cout << "Found." << endl;

cout << endl;

inputMatch += 1;

// Copying customer info from first set of arrays

// to the second set of arrays

cstmr.name[count] = customer.name[match];

cstmr.ID[count] = customer.ID[match];

cstmr.balance[count] = customer.balance[match];

//Incrementing counter for the loop

count++;

}

// Prompting for next name

cout << "Who do you want to search for (enter done to exit): ";

getline(cin, inputName);

}

// Print Report

// Opening output file

outFile.open(outFileName.c\_str());

// Calling on function PrintReportToFile

PrintReportToFile(outFile, cstmr, count);

// Closing output file

outFile.close();

// Delete dynamically allocated memory

delete [] customer.name;

delete [] customer.ID;

delete [] customer.balance;

delete [] cstmr.name;

delete [] cstmr.ID;

delete [] cstmr.balance;

return 0;

}

// Defining function LoadArraysFromFile

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* This function reads in the customer name,

\* customer ID, and the customer balance. It

\* uses a while loop to read in the customer

\* information into every index of the array,

\* until the end of the file or the end of the

\* array. This function also keeps track of the

\* number of records that was inside the input

\* file.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LoadArraysFromFile(ifstream& inFile, Customer customer, int& numRecords, const int SIZE)

{

//Initializing index to 0

int index = 0;

//Reading in first name

getline(inFile, customer.name[index]);

//While loop that does not stop until it reaches

//the end of the file and if it does not reach

//the end of the array

while(!inFile.eof() && index < SIZE)

{

//Reading in customer ID and balance

inFile >> customer.ID[index];

inFile >> customer.balance [index];

inFile.ignore(100, '\n');

//Keeping track of number of customer records

numRecords = index + 1;

//Incrementing index

index++;

//Reading in next name

getline(inFile, customer.name[index]);

}

}

// Defining function SearchForMatch

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* This function searches the array for the name,

\* entered by the user. It uses a while loop to

\* step through the array of names. Then the function

\* returns either -1 or the index, where the name is

\* found.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int SearchForMatch (string source[], int size, string match)

{

//Initializing index to 0

int index = 0;

//Initializing position to -1

int position = -1;

//Initializing bool variable to false

bool found = false;

//While loop that does not stop until

//the end of the array or until found = true

while(index < size && !found)

{

//If name matches the name at source[index]

//then found = true and position is set to

//the index

if(source[index] == match)

{

found = true;

position = index;

}

//Incrementing index

index++;

}

//Returns position of name back to main

return position;

}

// Defining function PrintReportToFile

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* This function outputs the customer report to the

\* output file. The function first outputs a chart

\* with a format. It uses a for loop to output each

\* customer record, that the user searched for. The

\* function also keeps track of the total balance,

\* then calculates and outputs the average.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void PrintReportToFile (ofstream &outFile, Customer cstmr, int inputMatch)

{

// Counter for loop

int count;

// Set accumulator to 0

float total = 0;

// Setting average to 0

float AVG = 0;

// Output the chart with the format

outFile << left;

outFile << setw(9) << "ID #" << setw(25) << "NAME" << "BALANCE DUE"

<< endl;

outFile << setw(9) << "----" << setw(25) << "--------------------"

<< "-----------" << endl;

// Output the ID's, names, and the balances

for (count = 0; count < inputMatch; count++)

{

//Outputting customer info to the output file

outFile << setprecision(2) << fixed;

outFile << setw(9) << cstmr.ID[count]

<< setw(25)<< cstmr.name[count]

<< setw(5) << "$"<< cstmr.balance[count] << endl;

// Accumulate total

total += cstmr.balance[count];

}

//Sets the output to the right

outFile << right;

// Compute average

AVG = total / inputMatch;

// Output average

outFile << setw(35) << "Average Balance Due: $"

<< setw(9) << AVG << endl;

}