*C++ requirements*

* The store number must be of type unsigned int. The sales value must be of of type long long int.
* Your program must properly check for end of file. See the section *Reading in files* below and also see your Gaddis text book for details on reading in file and checking for end of file.
* Your program must properly open and close all files.

Failure to follow the C++ requirements could reduce the points received from passing the tests.

*General overview*

In this program you will be reading sales information from a file and writing out a bar chart for each of the stores. The bar charts will be created by writing out a sequence of X characters.

You will have to create input files for your program. You can use a text editor such as *Notepad* or *Notepad++* to create this. There may also be an editor in your IDE that you can use. You can use the *TextEdit* program on *macOS* but you will need to change the format to *Make Plain Text*. You will not be uploading these text files to *zyBooks/zyLabs*. The submit tests will be using their own files. Make sure that some of your files contain a newline at the end of the last line of text in the file. You can do this by hitting the enter key after the last number of the last line in the file. This will match the type of file usually created by programs. This is the type of file that is used for the zyBooks tests. See the description below on reading in files.

*Reading in files*

When reading data from a file you need to make sure you are checking for end of file properly.

See Demo of file input and output of data in this unit for an explanation of how to properly check for end of file.

The general method shown in the *Gaddis* text book is:

ifstream inputFile;

inputFile.open("input.txt");

int num;

if (inputFile)

{

// the file opened successfully

while (inputFile >> num)

{

// process the num value

cout << num << endl;

}

inputFile.close();

}

else

{

cout << "The file could not be opened" << endl;

}

If you want to read in two values with every read you can simply replace inputFile >> num with something like inputFile >> num1 >> num2 as shown here:

while (inputFile >> num1 >> num2)

{

// process the num1 and num2 values

cout << num1 << " " << num2 << endl;

}

Text files are more complicated that they seem. Different operating systems handle text files differently. On Windows lines of text end with \r followed by \n. On Unix (or Linux) the lines end with just \n. On old Macs lines ended with \r but now use the Unix convention. The use of the >> operator takes care of these line ending issues for you.

But it is still more complicated than that. Most text files have every line ending with either \r \n (for Windows) or \n (for Unix/Linux and MacOS) but it is also possible to create a text file where the last line does NOT have the line ending characters. The use of the following code will work for all line endings even when the last line of text input does not end with any line endings.

if (inputFile >> num1 >> num2)

{

// executes only if the read worked

}

or

while (inputFile >> num1 >> num2)

{

// executes while the read works

}

There are other ways to test for end of file but you have to make sure your code will work for all of the cases discussed above. It is *STRONGLY* recommended that you use the process outlined above.

*General overview (continued)*

Your program will read in a file name from cin. It will then open the input file.

Your program must also open an output file called *saleschart.txt*. You will write the bar char headings and data to this file.

Your program needs to have the following general flow:

read in the file name

open the input file for this file name

display a message if the input file does not open and quit your program

open the output file ("saleschart.txt")

display a message if the output file does not open, close the input file, and quit your program

while (readFile into store number and sales for store

display an error if the store number or sales are invalid

if the data is good output the bar char to the output file

end while

close the input and output files

The processing loop will read the input data and process it until it gets and end of file indication from the file read operation

Assuming you have read in valid data AND this is the first sales data being processed your program should output some headings to the output file before processing the data. The headings are as follows:

SALES BAR CHART

(Each X equals 1,000 dollars)

*Note:* Your program must not output the headings to the output file if all of the data in the input file is invalid, or if there is not any valid data in the input file.

You need to come up with a way of keeping track if this is the first valid read or not. .

Assuming you have valid data the processing will consist displaying the output

Once the loop has completed you need to close the input file.

If the input file could not be opened your program should output an error message to cout. Assume the file we are reading in from is called *sales.txt*, and the file does not exist. The error message written to cout is:

File "sales.txt" could not be opened

The store number is of type unsigned int. Your program must verify that the store number is in the range 1 to 99 (inclusive). If the store number is invalid display the following message:

If the store number is less than 1 or greater than 99 you need to output the following message to cout:

The store number xx is not valid

Where *xx* is the store number.

If the sales data is read in as a long long int. If the sales value is less than 0 you need to output the following message to cout:

The sales value for store xx is negative

Where *xx* is the store number.

Don't forget to close both files, if they were opened.

*Write the bar chart information to the file*.

You will be outputting a string of X (upper case X) characters where each X represents $1,000 in sales for that store. For each 1,000 in sales you output one X. You do not round up the sales, so sales of $6,000 and sales of $6,999 would both output 6 X characters.

You will output the sales bar chart to the output file.

Assuming a store number of 9 and sales of $6,999. the *display* function will write the following to the output file:

Store 9: XXXXXX

*Note* that the store width is 2 characters, so the output is:

Store yy: XXXXXXX

The *yy* has a width of 2 even if the store number is 1 through 9.

*The format of the input file*

The data in the input file is in the order store number followed by the store sales. There will be zero or more of these input pairs in the file.

Here is the contents of a sample input text file:

1 10000

2 25000

3 37000

4 29000

5 8000

*Sample runs*

*Here is an example run. Assume the following input being read in from cin:*

sales.txt

Assume that the content of the file *sales.txt* are as follows:

1 10000

2 25000

3 37000

4 29000

5 8000

The output (to file *saleschart.txt*) for this input would be:

SALES BAR CHART

(Each X equals 1,000 dollars)

Store 1: XXXXXXXXXX

Store 2: XXXXXXXXXXXXXXXXXXXXXXXXX

Store 3: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Store 4: XXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Store 5: XXXXXXXX

You are reading from an input file and you are writing to an output file. Make sure you close both files after you are finished using them. You must do this in your program, you cannot just let the operating system close the files for you.

In this lab. and some future labs, you will be creating an output file. There could be output to cout as well.

For tests where there is output written to an output file the contents of the output file will determine if you passed that test or not. For cases where you have written out to cout the tests will check the output sent to cout.

Failure to follow the requirements for lab lessons can result in deductions to your points, even if you pass the validation tests. Logic errors, where you are not actually implementing the correct behavior, can result in reductions even if the test cases happen to return valid answers. This will be true for this and all future lab lessons.

*Expected output*

There are eight tests. Each test will have a new set of input data. You must match, exactly, the expected output.Tests 2, 5, 6, and 7 check the output sent to cout. Tests 1, 3, 4, and 8 check the output sent to file *saleschart.txt*.

You will get yellow highlighted text when you run the tests if your output is not what is expected. This can be because you are not getting the correct result. It could also be because your formatting does not match what is required. The checking that zyBooks does is very exacting and you must match it exactly. More information about what the yellow highlighting means can be found in course "How to use zyBooks" - especially section "1.4 zyLab basics".

Finally, do not include a system("pause"); statement in your program. This will cause your verification steps to fail.

*Note:* that the system("pause"); command runs the pause command on the computer where the program is running. The pause command is a Windows command. Your program will be run on a server in the cloud. The cloud server may be running a different operating system (such as Linux).