Debug F

Download Debug F

Lab F

Background

Classes are an excellent way to group data and functions that act on that data. For example, an ATM machine can contain many dollar bills that a user can withdraw, but what if I need a program to act on many ATM machines? Keeping track of which ATM to withdraw from and which variables to subtract the withdraw amount from might get complicated. Classes allows us to essentially create a new data type that can contain zero or more member variables, such as the amount of money in the machine. Furthermore, you may add functions to act upon that money. This is desirable for error checking. For example, if the user requests $1,000, but only $100 is in the machine, you don't want to charge their bank account $1,000 and only give them $100. The member functions can make this check prior to relinquishing the funds.

Problem

You will be developing a program that implements a highly simplified "stack" model for mathematics. For example, if you want to add 5 and 7 together, you would set the left operand to 5 and the right operand to 7, add, and then get the result. For those who are interested, the Java virtual machine uses this "stack" model for calculations, and old Intel machines used a "floating point unit" (FPU) which used a similar stack model. Granted they are more complicated, but nonetheless they are similar!

Your program will provide a list of mathematical operations that the user can choose from. The user will then provide the two operands. Your program will perform the operation and output the result back to the user. This will repeat until the user chooses to quit.

NOTE: You will receive NO credit for this lab if you do not declare a class or do not use the class for all calculations.

Non-member Function

Implement a menu **function** that asks the user what operation they want to use. This function will be a **void** function and use a **by-reference parameter** for the user's desired operation. Use the following menu:

```
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exponent
6. Quit
Enter operation:
```

If the user enters a number that is not part of the menu, print the menu again. Continue to print the menu until the user enters either 1, 2, 3, 4, 5, or 6. Your menu function must not accept an operation that is outside the range of [1,6].

Notice that this function is a standalone function and not part of any class.

Class Member Functions

You are implementing a class that uses a "stack"-like calculator. This means you will need to store two operands and a result. Remember, that both operands and the result may be decimals.

Prototype your class above int main and define your member functions below int main. You may name your class whatever you like, just as long as it has some relationship to what the class is actually doing (e.g., don't name your class Fluffy).

The member functions you will need to implement are:

```
SetLeft   - Takes a value and sets the left operand to the value; returns nothing
SetRight  - Analagous to SetLeft, except operates on the right operand
Multiply  - Stores the product of left * right into result; returns nothing
Divide    - Stores the quotient of left / right into result; returns nothing
Add       - Stores the sum of left + right into result; returns nothing
Subtract  - Stores the difference of left - right into result; returns nothing
Exponent  - Raises the left operand to the power of the right and stores the result into res
GetResult - An accessor that returns the value of the result
```

Inputs

Your int main() function will use the variable sent to your menu function by-reference as a parameter in order to determine the desired operation.

Inside of int main(), check the operation. If the user enters 6 (Quit), quit the program. Otherwise, ask the user:

```
Enter operands:
```

The user will supply **two** operands. For example, if the user wants to add, and enters 5 and 3, this means 5 + 3. Order is important here, especially for subtraction and division!

Process

You now have all of the information needed to calculate a result for the user: the desired operation and the operands. Call the appropriate class member functions to set both operands and to execute the operation that the user requested. Notice that setting both operands is the exact same for every operation the user can request. Do not repeat this code. If you find yourself typing SetLeft or SetRight multiple times in the same program, you need to stop and re-evaluate.

Outputs

After your member function call to the appropriate operation, output a new line.

Then output the result as:

```
Result = XX.YYY
```

Replace XX.YYY with the actual value. All results will be in fixed notation and precise to three (3) decimal points. Use I/O manipulators to enforce this notation.

EXAMPLE: If the user selects menu 1 (add) and enters operands 5.5 and 7.2, then the output will be

```
Result = 12.700
```

Notice that all operands and results may be decimals!

Lastly, after giving the result, repeat the entire process starting with outputting the menu. Your program will continue until the user selects 6 (quit).

Again, notice that the result is the same for all operations. Do not repeat any code here either!

Additional Requirements

1. You must prototype your class above int main and define your member functions below int main.
2. You must prototype your menu function above int main and define your menu function below int main.
3. You must use the class prototype given to you. You may change the names, but the return types and parameter lists must be the same.
4. You must use access protection for all of your class' members. All member variables will be private and all member functions will be public
5. You must use the class to
    - Set all operands
    - Calculate the result
    - Retrieve the result
6. This entire lab is to give you practice using classes and member variables/functions. If you do not use them, you will receive a 0 for this lab!
7. You must use accessor notation (const) where appropriate for your class.
8. The menu function that you write must be a void function and have a by-reference parameter for the operation that the user wants to use.
9. Do NOT repeat code unnecessarily.
10. You must use I/O manipulators to set the appropriate decimal notation and precision
11. You must use the appropriate namespace for all standard functions and objects, such as cout and cin (i.e., we don't want to see std::cout).
12. Do NOT use any global variables (global constants are OK)

C++ Topics Covered

Classes (mutators and accessors)
Class member function access protection
Pass-by-value functions/member functions
Accessor member function syntax

Textbook Chapters Covered

Chapter 10.1 (Structures)
Chapter 10.2 (Classes)

Relevant Reading

Classes - simple
Classes - accessors and functions
Void functions
By-value parameters
By-reference parameters

Built with concrete5 CMS.

Log in