Part C - Encapsulation   
  
**Member Operators / Helper Functions**   
  
Workshop 6

In this workshop, you are to design and code a class that overloads operators.

**Learning Outcomes**

Upon successful completion of this workshop, you will have demonstrated the abilities to

* overload an operator as a member function
* overload an operator as a helper function
* overload an operator as a friend function
* reflect on what you have learned in this workshop

**FLYING POINTS Calculator – Basic task**

FlyingPoints Class

Define a class named **FlyingPoints** that stores separately the information related to any flying points you receive (incoming points) and for any flying points you spend (outgoing points). Any flying points received are added to the flying points’s incoming variable (in), and any flying points spent are added to the flying points’s outgoing variable (out). The **updateBalance()** function is called as needed, to subtract points spent from points received.

Include in your class design the following functions:

* **FlyingPoints()** - a no-argument constructor that sets the state of the current object to a safe empty state (i.e. sets your in and out variables to zero).
* **FlyingPoints(int, int)** - a two-argument constructor that receives an initial value for points you have received (in) and for points you have spent (out).
* **void updateBalance()** - a *private* member function that subtracts what you have spent from what you have received, i.e.:
  + sets in to **in - out** and sets out to 0, if in > out, or
  + sets out to **out - in** and sets in to 0, otherwise.
* **void display()const** - a query that displays the in and out data to the standard output.
* **FlyingPoints& operator+=(const FlyingPoints& f)** - a member operator that adds **FlyingPoints f** to the current object (adds f’s in to the current object’s in, and adds f’s out to the current object’s out), balances the current object, and returns a reference to the current object.
* **FlyingPoints operator+(const FlyingPoints&, const FlyingPoints&)** - a non-friend helper operator function that adds two **FlyingPoints** objects, balances the result, and returns a copy of the result.
* **bool operator==(const FlyingPoints&, const FlyingPoints&)** - a friend operator function that compares two **FlyingPoints** objects for equality (checks that the in value of both is the same and the out value of both is the same, and returns true or false depending on the result).

Make sure to call the **updateBalance()** member function from your other functions whenever appropriate.

Place your class definition in a header file named **FlyingPoints.h** and your function definitions in an implementation file named **FlyingPoints.cpp**.

Client Module

The main module that uses your **FlyingPoints** class is listed on the left, while the results of execution are listed on the right:

|  |  |
| --- | --- |
| **// Workshop 6 - Flying Points Calculator**  **// w6.cpp**  **#include <iostream>**  **#include "FlyingPoints.h"**  **using namespace std;**  **void read(const char\* msg, FlyingPoints& f)**  **{**  **int inPoints, outPoints;**  **cout << "Enter " << msg << endl;**  **cout << "Enter number of incoming flying points: ";**  **cin >> inPoints;**  **cout << "Enter number of spent flying points: ";**  **cin >> outPoints;**  **f = FlyingPoints(inPoints, outPoints);**  **}**  **int main()**  **{**  **FlyingPoints left, right, result, ref;**  **cout << "Flying Points Calculator\n";**  **cout << "========================\n";**  **read("Left Operand...", left);**  **read("Right Operand...", right);**  **cout << "Result: " << endl;**  **left.display();**  **cout << " + ";**  **right.display();**  **cout << " = ";**  **result = left + right;**  **result.display();**  **cout << endl;**  **read("2nd Right Operand...", right);**  **cout << "Result: " << endl;**  **result.display();**  **cout << " += ";**  **right.display();**  **cout << " => ";**  **result += right;**  **result.display();**  **cout << endl;**  **read("Reference...", ref);**  **if(result == ref)**  **{**  **cout << "Result == Reference" << endl;**  **}**  **else**  **{**  **cout << "Result != Reference" << endl;**  **}**    **return 0;**  **}** | **Flying Points Calculator**  **========================**  **Enter Left Operand...**  **Enter number of incoming flying points: 1000**  **Enter number of spent flying points: 600**  **Enter Right Operand...**  **Enter number of incoming flying points: 100**  **Enter number of spent flying points: 300**  **Result:**  **1000 points available to fly.**  **600 points spent.**  **+ 100 points available to fly.**  **300 points spent.**  **= 200 points available to fly.**  **0 points spent.**  **Enter 2nd Right Operand...**  **Enter number of incoming flying points: 500**  **Enter number of spent flying points: 600**  **Result:**  **200 points available to fly.**  **0 points spent.**  **+= 500 points available to fly.**  **600 points spent.**  **=> 100 points available to fly.**  **0 points spent.**  **Enter Reference...**  **Enter number of incoming flying points: 1100**  **Enter number of spent flying points: 1000**  **Result == Reference** |

Typescript

Create a typescript of your complete solution using the following commands:

|  |
| --- |
| **+ At the prompt, type: script w6.txt**  **+ At the prompt, type: whoami**  **+ At the prompt, type: cat <all source files>**  **+ At the prompt, type: g++ -o w6 <modules> -Wall**  **+ At the prompt, type: w6**  **+ At the input prompt: enter test data**  **+ At the prompt, type: exit** |

These commands will produce a file named **w6.txt**.  Download this file to your local computer.

**Bonus activity - Upgrade**

Design and code the **FlyingPoints** class described above.  Add to your class design the following member functions:

* **FlyingPoints& operator-=(const FlyingPoints& f)** - a member operator that subtracts **FlyingPoints f** from the current object (subtract f’s in from this in, subtract f’s out from this out points), balances the result, and returns a reference to the current object.
* **FlyingPoints& operator\*=(const FlyingPoints& f)** - a member operator that multiplies the current object by **FlyingPoints f** (i.e. multiples the current object’s in by f’s in and multiplies the current object’s out by f’s out), balances the result, and returns a reference to the current object.
* **FlyingPoints& operator/=(const FlyingPoints& f)** - a member operator that divides the current object by **FlyingPoints f** as follows: if f’s in is not 0, divide the current object’s in by f’s in, otherwise set the current object’s in to 0. If f’s out is not 0, divide the current object’s out by f’s out, otherwise set the current object’s out to 0. The result is then balanced and the function returns a reference to the current object.

Main Program

The main module that uses your **FlyingPoints** class is listed on the left, while the results of execution are listed on the right:

|  |  |
| --- | --- |
| **// Workshop 6 - Flying Points Calculator**  **// w6bonus.cpp**  **#include <iostream>**  **#include "FlyingPoints.h"**  **using namespace std;**  **void read(const char\* msg, FlyingPoints& f)**  **{**  **int inPoints, outPoints;**  **cout << "Enter " << msg << endl;**  **cout << "Enter number of incoming flying points: ";**  **cin >> inPoints;**  **cout << "Enter number of spent flying points: ";**  **cin >> outPoints;**  **f = FlyingPoints(inPoints, outPoints);**  **cin.ignore();**  **}**  **int main()**  **{**  **char op[3];**  **bool quit = false;**  **FlyingPoints left, right, result, ref;**  **cout << "Flying Points Calculator\n";**  **cout << "========================\n";**  **read("Left Operand...", left);**  **do**  **{**  **cout << "+= -= \*= /= (## to quit): ";**  **cin.get(op, 3);**  **char c = cin.get();**  **if(c != '\n' || op[1] != '=' && op[1] != '#')**  **{**  **cerr << "Try Again!" << endl;**  **cin.ignore(2000, '\n');**  **}**  **else if(strcmp(op, "##") == 0)**  **{**  **read("Reference...", ref);**  **quit = true;**  **}**  **else**  **{**  **read("Right Operand...", right);**  **cout << "Result: " << endl;**  **left.display();**  **switch(op[0])**  **{**  **case '+':**  **cout << " += ";**  **left += right;**  **break;**  **case '-':**  **cout << " -= ";**  **left -= right;**  **break;**  **case '\*':**  **cout << " \*= ";**  **left \*= right;**  **break;**  **case '/':**  **cout << " /= ";**  **left /= right;**  **break;**  **}**  **right.display();**  **cout << " => ";**  **left.display();**  **cout << endl;**  **}**  **} while (!quit);**  **if(result == ref)**  **{**  **cout << "Result == Reference" << endl;**  **}**  **else**  **{**  **cout << "Result != Reference" << endl;**  **}**    **return 0;**  **}** | **Flying Points Calculator**  **========================**  **Enter Left Operand...**  **Enter number of incoming flying points: 1000**  **Enter number of spent flying points: 600**  **+= -= \*= /= (## to quit): +=**  **Enter Right Operand...**  **Enter number of incoming flying points: 100**  **Enter number of spent flying points: 300**  **Result:**  **1000 points available to fly.**  **600 points spent.**  **+= 100 points available to fly.**  **300 points spent.**  **=> 200 points available to fly.**  **0 points spent.**  **+= -= \*= /= (## to quit): \*=**  **Enter Right Operand...**  **Enter number of incoming flying points: 300**  **Enter number of spent flying points: 100**  **Result:**  **200 points available to fly.**  **0 points spent.**  **\*= 300 points available to fly.**  **100 points spent.**  **=> 60000 points available to fly.**  **0 points spent.**  **+= -= \*= /= (## to quit): -=**  **Enter Right Operand...**  **Enter number of incoming flying points: 300**  **Enter number of spent flying points: 1200**  **Result:**  **60000 points available to fly.**  **0 points spent.**  **-= 300 points available to fly.**  **1200 points spent.**  **=> 60900 points available to fly.**  **0 points spent.**  **+= -= \*= /= (## to quit): /=**  **Enter Right Operand...**  **Enter number of incoming flying points: 200**  **Enter number of spent flying points: 100**  **Result:**  **60900 points available to fly.**  **0 points spent.**  **/= 200 points available to fly.**  **100 points spent.**  **=> 304 points available to fly.**  **0 points spent.**  **+= -= \*= /= (## to quit): ##**  **Enter Reference...**  **Enter number of incoming flying points: 3000**  **Enter number of spent flying points: 2696**  **Result == Reference** |

Typescript

Create a typescript of your complete solution using the following commands:

|  |
| --- |
| **+ At the prompt, type: script w6bonus.txt**  **+ At the prompt, type: whoami**  **+ At the prompt, type: cat <all source files>**  **+ At the prompt, type: g++ -o w6bonus <modules> -Wall**  **+ At the prompt, type: w6bonus**  **+ At the input prompt, enter test data**  **+ At the prompt type: exit** |

These commands will produce a file named **w6bonus.txt**.  Download this file to your local computer.

**SUBMISSION**

Submit your solution following the instructions given to you by your instructor. For those of you using BlackBoard for submission, the instructions are given below. If you have NOT been told to submit using BlackBoard, follow the instructor-specific instructions instead.

BlackBoard

Upload your typescript file to BlackBoard:

* Login to BlackBoard
* Select your course code
* Select Workshop 6 under Assignments
* Upload **w6.txt**, and optionally **w6bonus.txt**
* Write a short note to your instructor
  + Under “Add comments”, add a sentence or two regarding what you thing you learned in this workshop in the notes textbox
  + press "Save Changes"
* When ready to submit, press "Submit". Note you can save a draft until you are ready to submit.