1. What is the difference between a (binary) operator and a function?

// <http://stackoverflow.com/questions/4706572/difference-between-operator-and-function-in-c>

Operator don’t carry any meaning. During semantic analysis, the meaning of an operator is determined. They are like symbols (e.g. +,-,+=)

A function is a constructor, destructor, conversion function (that looks like operator type()) or operator function (function template specialization and instantiation can yield these in turn).

1. Is it possible using operator overloading to change the effect of + on integers? Why or why not?

Yes it is possible to change the effect of + on integer by overloading it. We might want to overload it to allow us to add a certain value based on a type we have defined (e.g. adding the salary of 2 Employee object).

1. Why can’t we overload << or >> as member operators?

We can’t overload the << or >> because of the rules. If the first (LH) operand is NOT an object of the class, you can only use a stand-alone function. The (LH) operand for the << or >> is NOT an object of the class.

4. Below is the definition for a class called Percent. Objects of type Percent represent percentages such as 10% or 99%. Give the definitions of the overloaded operators >> and << so that they can be used for input and output with objects of the class Percent. Assume that input always consists of an integer followed by the character ’%’, such as 25%. All percentages are whole numbers and are stored in an int member variable named value. You do not yet need to define the other overloaded operators and do not yet need to define the constructor. Firstly, you only have to define the overloaded operators >> and <<.

#include <iostream>

using namespace std;

class Percent

{

public:

friend bool operator ==(const Percent& first,

const Percent& second);

friend bool operator <(const Percent& first,

const Percent& second);

Percent( );

friend istream& operator >>(istream& inputStream,

Percent& aPercent);

friend ostream& operator <<(ostream& outputStream,

const Percent& aPercent);

//There will be other members and friends.

private:

int value;

};

1. Add to the class overloaded operators to add, subtract and multiply 2 percentages. If you have to re-write the code so that wherever possible the overloaded operators are made member functions of the class.
2. #pragma once
3. #include <iostream>
4. using namespace std;
5. class Percent
6. {
7. public:
8. friend bool operator ==(const Percent& first,
9. const Percent& second);
10. friend bool operator <(const Percent& first,
11. const Percent& second);
12. Percent();
13. friend istream& operator>>(istream& inputStream,
14. Percent& aPercent);
15. friend ostream& operator<<(ostream& outputStream,
16. const Percent& aPercent);
17. //There will be other members and friends.
18. private:
19. int value;
20. };

//http://www.cprogramming.com/tutorial/operator\_overloading.html

#include "Percent.h"

#include <iostream>

using namespace std;

istream& operator>>(istream& inputStream,

Percent& aPercent)

{

inputStream >> aPercent.value;

return inputStream;

}

1. Implement all of the overloaded operators in the class (and any stand-alone ones still declared as friend functions of the class)

*N.B. Think carefully about the multiplication: 50% \* 50% should mean 50% OF 50%, and therefore provide the answer 25% not 2500%*

1. Write a program which fully tests your class.

5. Write a Money class, with data members euro and cent. Add overloaded operators (as member functions where this is possible) to do the following:

* Subtract 2 Money amounts.
* Multiply a Money amount by an integer to facilitate code like this:

myMoney = yourMoney \* 2;

* Divide a Money amount by a decimal
* Output a Money amount using the insertion operator
* Compare 2 Money amounts (with <, > and ==)