Reference:

Good description of Operator Overloading.

C++ How to program. DEITEL DEITEL. Chapter 8

Read the following, credit to Stack Overflow forum members:

I am trying to display p object of Point class that I have created through member function . I have passed Point p as argument in void displayPoint(Point p) member function of my program. But I am getting the following compilation error in my program!

point.cpp[Error] no match for 'operator<<' (operand types are 'std::ostream {aka std::basic\_ostream}' and 'Point')

void Point::displayPoint(Point p){

cout << p; // ERROR OCCURING HERE!!!

}

You have not overloaded << operator to output the object of class Point directly.

So you can't do that. You can either:

1. Add an overloaded operator<< or,
2. Call corresponding get functions to get the data members of Point.

For example, using get functions:

void Point::displayPoint(Point p){

cout << p.getX() << " " << p.gety() << endl;

}

You can take a look at Operator Overloading C++ about overloading operator<<.

std::cout << myPoint is just syntactic sugar for operator<<(std::cout, myPoint).

So you have to overload operator<< for your class according to your needs:

std::ostream& operator<<(std::ostream& os, const Point& p)

{

os << p.getX() << "/" << p.getY();

return os;

}

You have not defined

std::ostream& operator<<( std::ostream&, const Point&);

This function is needed whenever you want to put a Point into std::ostream using << as in:

void Point::displayPoint(Point p){

cout << p; // operator<< must be overloaded to make this work

}

Possible implementation of this method for your purpose is:

std::ostream& operator<<( std::ostream& s, const Point& p)

{

s << p.getX() << ", " << p.getY();

return s;

}

You can look [here](http://www.learncpp.com/cpp-tutorial/93-overloading-the-io-operators/) for some examples of overload for a very similar class as yours Point.

Also, if you get the following error; [operator << must take exactly one argument](https://stackoverflow.com/questions/10744787/operator-must-take-exactly-one-argument)

See these are very good:

<https://stackoverflow.com/questions/10744787/operator-must-take-exactly-one-argument>

and

<https://stackoverflow.com/questions/35259277/getting-an-error-of-the-form-operator-must-take-exactly-one-argument-when-ov>

I think that the issue here is that you're (probably unintentionally) mixing and matching two different approaches.

When you write

friend std::ostream& operator <<( std::ostream& os , const BigInt &param );

inside of the class definition, you are saying that there will be a *free function* named operator << that takes in an ostream and a BigInt. When you write

std::ostream& BigInt::operator <<( std::ostream& os , const BigInt & param )

{

...

};

you are defining a *member function* of BigInt named operator << taking two arguments. Those two arguments, combined with the implicit this pointer, adds up to three arguments - more than you intended. Note that although this function is called operator<<, it's *not* the operator<< you declared as a friend in the class definition.

To fix this, you have a few options. First, when you define operator<<, you can omit the BigInt:: prefix, which will fix things. Alternatively, combine your implementation with the friend definition inside the class definition:

friend std::ostream& operator <<( std::ostream& os , const BigInt & param )

{

...

}