**Lab Sections**

1. Objectives
2. Introduction
3. Definitions & Important Terms
4. Declaration Syntax
5. Experiments

|  |
| --- |
|  |

Member Operator Overloading

**Member Operator Overloading**

1. **Objectives**

**After you complete this experiment you will be able to overload an operator as a member function of a class.**

1. **Introduction**

When you overload an operator you provide your personal implementation for the operator. It comes in handy when you create a new object and you need to refine the implementation of an existing operator to match your needs. Overloading operators is really effective when the original meaning of the operator can be ported to the new object. For example, the “+” operator should mean some type of addition will take place. For strings and chars addition is concatenation; for doubles and ints it is numeric summation and for a new type of array class it may be the summation of all the corresponding elements in two adjacent arrays. Overloading C++ operators is an optional topic. However, we think it is important that you understand how to implement operator overloading, so we will cover it.

1. **Definitions & Important Terms**

We will define several terms you need to know to understand classes. They are as follows:

1. The **arity** of an operator is the number of parameters (operands) it requires. The arity of an operator cannot change.
2. A **non-member** function of a class does not have access to the state (private area) of a class.
3. **Chaining** occurs when a C++ statement contains several instances of the same overloaded operator.
4. The **current object** is the object that performed the function invocation.
5. **“this”** is a pointer to the current object.
6. **All member functions of a class have access to the “this” pointer.**
7. **Non-member and friend functions of a class do not have access to the “this” pointer.**
8. **“\*this”** is the current object.
9. When an object is passed **“implicitly”** it is passed through the “**this” pointer**.
10. When an object is passed **“explicitly”** it is passed through its corresponding formal parameter.
11. **Declaration Syntax**

**Notice that the non-member function does not have a prototype in the class declaration and the syntax for the implementation of the non-member function body in the following code:**

class Class\_name

{

public:

constructors

destructor

member functions

accessors

mutators

public data

private:

helper functions

data

};

--------------

return\_type Class\_name::function\_name(formal parameter list)

{

body

}

More information on classes can be found in your course textbook and on the web.

1. **Experiments**

**Step 1: In this experiment you will investigate the implementation for overloading the operator “+” as a member function of a class with chaining. Enter, save, compile and execute the following program in MSVS. Call the new project “MemberOpOverloadingExp” and the program “MemberOpOverloading.cpp”. Answer the questions below:**

#include <iostream>

#include <string>

using namespace std;

const int SIZE = 10;

class Bank\_Acct

{

public:

Bank\_Acct( ); //default constructor

Bank\_Acct(double new\_balance, string Cname); //explicit value

//constructor

void Print( ); //accessor function

Bank\_Acct & operator+(double amount); //mutator function

private:

double balance;

string name;

};

Bank\_Acct::Bank\_Acct()

{

balance = 0;

name = "NoName";

}

Bank\_Acct::Bank\_Acct(double amount, string Cname)

{

balance = amount;

name = Cname;

}

void Bank\_Acct::Print()

{

cout<<endl<<"Object "<<name;

cout<<endl<<"The new balance is "<<balance<<endl;

}

Bank\_Acct & Bank\_Acct::operator+(double amount)

{

balance += amount;

return \*this;

}

int main()

{

Bank\_Acct my\_Acct;

cout.setf(ios::showpoint);

cout.setf(ios::fixed);

cout.precision(2);

cout<<"Original balance of my\_Acct"<<endl;

my\_Acct.Print( );

//the following statement contains chaining

my\_Acct + 18.75 + 14.35 + 10054.96;

cout<<"The balance of my\_Acct after addition to balance 3 times"<<endl;

my\_Acct.Print();

return 0;

}

1. Referring to the program in Step 1, if the arity of the “+” operator is two, why is there only one formal parameter in operator+’s function header?
2. What is the return type of the operator+ function?
3. Referring to the operator+ function, what is the name of the Bank\_Acct object it returned?
4. Why is there a “Bank\_Acct::” prefixed to the header of the operator+ function?
5. Can you explain how chaining is performed for the operator+ function in the program in Step 1.
6. Please explain why the operator “<<” cannot be overloaded as a member function? (hint: modify the code that was used in the laboratory “Classes: Part 2” to overload the “<<” operator as a friend member function.)