Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

**Лабораторная работа «Класс №3»**

**по дисциплине**

**«Основы алгоритмизации и программирования»**

**(Семестр** 2)

Выполнил студент гр. ИВТ-21-1б

Гребнев Алексей Дмитриевич

Проверил:

Яруллин Денис Владимирович

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(оценка) (подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

г.Пермь -2022

**Постановка задачи**

1. Определить пользовательский класс.
2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.
3. Определить в классе деструктор.
4. Определить в классе компоненты-функции для просмотра и установки полей данных(селекторы и модификаторы).
5. Перегрузить операцию присваивания.
6. Перегрузить операции ввода и вывода объектов с помощью потоков.
7. Перегрузить операции указанные в варианте.
8. Написать программу, в которой продемонстрировать создание объектов и работу всех перегруженных операций.

**Исходные данные**

#include <iostream>

#include <sstream>

#include <cmath>

#include <string>

using namespace std;

class Money

{

private:

long rub;

int cop;

public:

Money()

{

rub = 0;

cop = 0;

};

Money(long k, int p)

{

rub = k;

cop = p;

};

Money(const Money& g)

{

rub = g.rub;

cop = g.cop;

};

friend bool operator==(const Money& m1, const Money& m2);

friend bool operator!=(const Money& m1, const Money& m2);

~Money()

{

cout << "Вызван деструктор: " << endl;

};

long get\_rub()

{

return rub;

};

int get\_cop()

{

return cop;

};

void set\_rub(long k)

{

rub = k;

};

void set\_cop(int p)

{

cop = p;

};

void show\_money()

{

cout << "Для объекта " << this << " => " << "Рубли: " << rub << "; Копейка: " << cop << endl;

}

Money& operator--()

{

this->cop -= 1;

return \*this;

}

Money& operator--(int p)

{

this->cop -= 1;

return \*this;

}

};

bool operator==(const Money& m1, const Money& m2)

{

if ((m1.rub == m2.rub) && (m1.cop == m2.cop))

{

return true;

}

else

{

return false;

}

}

bool operator!=(const Money& m1, const Money& m2)

{

if ((m1.rub != m2.rub) && (m1.cop != m2.cop))

{

return true;

}

else

{

return false;

}

}

int main()

{

setlocale(LC\_ALL, "RUS");

Money m\_first;

Money m\_second;

m\_first.set\_rub(123);

m\_first.set\_cop(33);

m\_second.set\_rub(43);

m\_second.set\_cop(21);

m\_first.get\_rub(); m\_first.get\_cop();

m\_second.get\_rub(); m\_second.get\_cop();

m\_first--; m\_first.show\_money();

--m\_second; m\_second.show\_money();

cout << endl << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl << endl;

Money m\_third;

Money m\_fourth;

m\_third.set\_rub(62);

m\_third.set\_cop(65);

m\_fourth.set\_rub(87);

m\_fourth.set\_cop(99);

m\_third.get\_rub(); m\_third.get\_cop();

m\_fourth.get\_rub(); m\_fourth.get\_cop();

m\_third.show\_money();

m\_fourth.show\_money();

//Сравнение по operator==

bool answer\_compare1;

answer\_compare1 = m\_third == m\_fourth;

if (answer\_compare1 == true)

{

cout << "Одна сумма равна другой?: 'Да!' " << endl;

}

else

cout << "Одна сумма равна другой?: 'Нет!' " << endl;

bool answer\_compare2;

answer\_compare2 = m\_third != m\_fourth;

if (answer\_compare2 == true)

{

cout << "Одна сумма неравна другой?: 'Да!' " << endl;

}

else

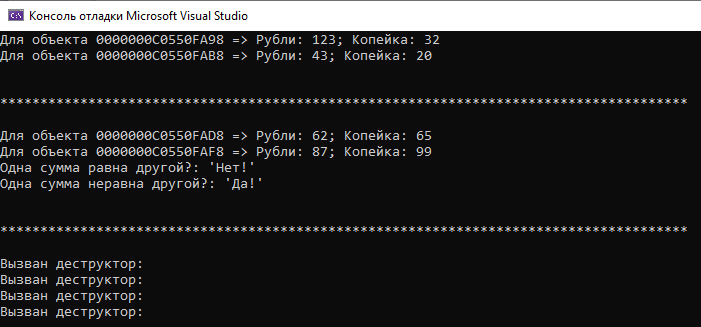
cout << "Одна сумма неравна другой?: 'Нет!' " << endl;

cout << endl << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl << endl;

return 0;

}

**Результат**



**Контрольные вопросы:**

1. **Для чего используются дружественные функции классы?**

Они используются для того, чтобы дать доступ объекту или классу к полям класса, компонентом которого они не являются.

1. **Сформулировать правила описания и особенности дружественных функций.**

Дружественная функция объявляется внутри класса, к элементам которого ей нужен доступ, с ключевым словом friend.

1. **Каким образом можно перегрузить унарные операции?**

Как компонентную (внутри класса) или как глобальную функцию с ссылкой на класс.

1. **Сколько операндов должна иметь унарная функция-операция, определяемая внутри класса?**

Один, при этом операндом является вызвавший её объект.

1. **Сколько операндов должна иметь унарная функция-операция, определяемая вне класса?**

Один, при этом она должна иметь один параметр типа класса.

1. **Сколько операндов должна иметь бинарная функция-операция, определяемая внутри класса?**

Два, при этом вызвавший ее объект считается первым операндом.

1. **Сколько операндов должна иметь бинарная функция-операция, определяемая вне класса?**

Если функция определяется *вне класса,* она должна иметь два параметра типа класса.

1. **Чем отличается перегрузка префиксных и постфиксных унарных операций?**

Разница заключается в значении, которое вернется. Префиксные возвращают объект после его инкрементирования или декрементирования, когда постфиксные же возвращают объект до того как с ним будут выполнены данные действия.

1. **Каким образом можно перегрузить операцию присваивания?**

Пишется тип класса с ссылкой. Также должно быть написано ключевое слово “operator” и вслед за ним оператор присваивания. В скобках, в качестве параметра указывается ссылка на константное значение(объект класса).

1. **Что должна возвращать операция присваивания?**

Должна возвращать ссылку на объект, для которого она вызвана.

1. **Каким образом можно перегрузить операции ввода-вывода?**

В классе нужно определить потоки ввода и вывода как дружественные функции, написать ключевое слово “operator”, после написать “<<” или “>>” в зависимости от того вывод это или ввод, в качестве параметров указываем ссылку на переменную ввода, вывода, и ссылку на объект класса (в случае вывода делаем его константой, чтобы данные никак нельзя было изменить во время вывода).

1. В программе описан класс   
   class Student

{

…

Student & operator++();

….

};

и определен объект этого класса

Students;

Выполняется операция

++s;

**Каким образом, компилятор будет воспринимать вызов функции-операции?**

S увеличится на 1 и далее уже с изменённым объектом будут выполнятся какие-либо действия.

1. В программе описан класс

class Student

{

…

Friend Student& operator++(Student&);

….

};

И определен объект этого класса Students;

Выполняется операция

++s;

**Каким образом, компилятор будет воспринимать вызов функции-операции?**

Вернется значение до инкремента и с ним будут выполнятся действия, и только потом оно увеличится на 1.

1. В программе описан класс

Class Student

{

…

Bool operator<(Student&P);

….

};

И определены объекты этого класса Student a,b;

Выполняется операция cout<<a<b;

**Каким образом, компилятор будет воспринимать вызов функции-операции?**

Он сравнит значения объекта а и объекта b, так как это прописал пользователь. И если значения a меньше значений b, функция вернет true, или false, если это не так.

1. В программе описан класс class Student

{

…

Friend bool operator**>**(const Person&, Person&)

….

};

И определены объекты этого класса Student a,b;

Выполняется операция cout<<a>b;

**Каким образом, компилятор будет воспринимать вызов функции-операции?**

Он сравнит значения объекта а и объекта b, так как это прописал пользователь. И если значение(я) aбольше значения(й) b, функция вернет true, или false, если это не так.