Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Институт информационных технологий и управления в технических системах

ОТЧЕТ

по лабораторной работе №4

ПЕРЕГРУЗКА ОПЕРАТОРОВ.

Выполнил: ст. гр. ИТб-21

Куркчи А.Э.

Проверила:

Тимофеев И.С.

Севастополь

2015

# 1. Цель

Приобретение практических навыков при написании объектно-ориентированных программ с использованием перегруженных операторов. Освоение особенностей отладки объектно-ориентированных программ.

2. Постановка задач

Для заданного по варианту класса выполнить следующие действия:

* Описать все типы конструкторов и декструктор (по необходимости);
* Переопределить оператор вывода в поток <<;
* Переопределить оператор ввода из потока >>;
* Переопределить заданные по варианту операторы;

Создать два объекта заданного по варианту класса и на их примере продемонстрировать корректную работу всех перегруженных операторов.

Вариант №1

Создать класс целых чисел MyInt. Перегрузить операторы:

1. ! как унарный метод класса, проверяющий четность числа;
2. -- (префиксный) как унарную дружественную функцию, уменьшающую значение числа на 3;
3. + как бинарный метод класса, прибавляющий к значению объекта целое число;
4. == как бинарную дружественную функцию, проверяющую равенство двух объектов.

3. Текст программы

#include <iostream>

#include <math.h>

**using** **namespace** std**;**

**typedef** long long LL**;**

class mint **{**

protected**:**

LL value**;**

public**:**

mint**()** **{**

value **=** 0**;**

**}**

mint**(**long long v**)** **{**

value **=** v**;**

**}**

mint**(**const mint**&** mi**)** **{**

value **=** mi**.**value**;**

**}**

bool **operator** **!** **()** **{**

**return** **(**value **%** 2**)** **==** 0**;**

**}**

mint **operator** **+** **(**const mint**&** mi**)** const **{**

**return** **(**value **+** mi**.**value**);**

**}**

mint **operator** **+** **(**const int**&** i**)** const **{**

**return** **(**value **+** i**);**

**}**

friend mint**&** **operator** **--** **(**mint**&** mi**);**

friend bool **operator** **==** **(**const mint**&** mi1**,**const mint**&** mi2**);**

friend istream**&** **operator** **>>** **(**istream**&** stream**,**mint**&** mi**);**

friend ostream**&** **operator** **<<** **(**ostream**&** stream**,**const mint**&** mi**);**

**};**

mint**&** **operator** **--** **(**mint**&** mi**)** **{**

mi**.**value **-=** 3**;**

**return** mi**;**

**}**

bool **operator** **==** **(**const mint**&** mi1**,**const mint**&** mi2**)** **{**

**return** mi1**.**value **==** mi2**.**value**;**

**}**

istream**&** **operator** **>>** **(**istream**&** stream**,**mint**&** mi**)** **{**

stream **>>** mi**.**value**;**

**return** stream**;**

**}**

ostream**&** **operator** **<<** **(**ostream**&** stream**,**const mint**&** mi**)** **{**

stream **<<** mi**.**value**;**

**return** stream**;**

**}**

int main**(**int argc**,**char**\*\*** argv**)** **{**

cout **<<** "SevGU semestr.03 OOP lab.04" **<<** endl**;**

cout **<<** "Demonstration start" **<<** endl**;**

mint m1**;**

mint m2**;**

// cout << "Scanning m1" << endl;

cout **<<** "> cin >> m1;" **<<** endl**;**

cin **>>** m1**;**

// cout << "Scanning m2" << endl;

cout **<<** "> cin >> m2;" **<<** endl**;**

cin **>>** m2**;**

// cout << "Using prefix decrement operator to make -3" << endl;

cout **<<** "> --m1;" **<<** endl**;**

**--**m1**;**

// cout << "Printing m1" << endl;

cout **<<** "> cout << m1 << endl;" **<<** endl**;**

cout **<<** m1 **<<** endl**;**

// cout << "Checking is m2 even" << endl;

cout **<<** "> cout << ((!m2) ? \"true\" : \"false\") << endl;" **<<** endl**;**

cout **<<** **((!**m2**)** **?** "true" **:** "false"**)** **<<** endl**;**

// cout << "Checking is m1 equals m2" << endl;

cout **<<** "> cout << ((!m1 == m2) ? \"true\" : \"false\") << endl;" **<<** endl**;**

cout **<<** **((**m1 **==** m2**)** **?** "true" **:** "false"**)** **<<** endl**;**

// cout << "Adding 5 to m1" << endl;

cout **<<** "> m1 = m1 + 5;" **<<** endl**;**

m1 **=** m1 **+** 5**;**

// cout << "Adding m1 to m2" << endl;

cout **<<** "> m2 = m1 + m2;" **<<** endl**;**

m2 **=** m1 **+** m2**;**

// cout << "Printing m1" << endl;

cout **<<** "> cout << m1 << endl;" **<<** endl**;**

cout **<<** m1 **<<** endl**;**

// cout << "Printing m2" << endl;

cout **<<** "> cout << m2 << endl;" **<<** endl**;**

cout **<<** m2 **<<** endl**;**

cout **<<** "Demonstration end" **<<** endl**;**

**return** 0**;**

**}**

4. Тестовые примеры

На рисунке 1 представлен вывод программы

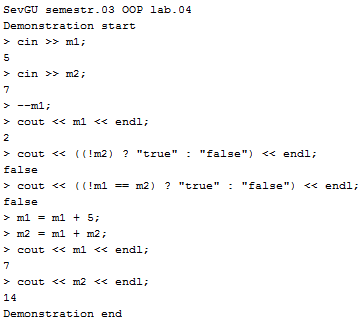


Рисунок 1 – Вывод программы

После ввода 2х значений начинается демонстрация корректности программы. В строках, начинающихся с «>», печатается выполняемая в программном коде строка, в остальных строках – вывод программы.

Вывод

В ходе лабораторной работы были приобретены практические навыки написания объектно-ориентированных программ с использованием перегруженных операторов. Освоены особенности отладки объектно-ориентированных программ.