Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

# Лабораторная работа №3

# « Перегрузка операций »

# Семестр 2

Выполнил работу

Студент группы РИС-22-1Б

Зырянов Ростислав

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Г. Пермь-2023

**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.

**Вариант 15**

Создать класс Pair (пара чисел). Пара должна быть представлено двумя полями: типа int для первого числа и типа double для второго. Первое число при выводе на экран должно быть отделено от второго числа двоеточием. Реализовать:

* вычитание пар чисел
* добавление константы к паре (увеличивается первое число, если константа целая, второе, если константа вещественная).

**Ответы на контрольные вопросы**

1. **Для чего используются дружественные функции и классы?** Дружественные функции и классы используются для предоставления доступа к закрытым (private) членам класса другим функциям или классам.
2. **Сформулировать правила описания и особенности дружественных функций.** Правила описания дружественных функций:

* Дружественная функция должна быть объявлена вне класса, но ее определение может быть внутри класса;
* Для объявления дружественной функции используется ключевое слово "friend";
* Дружественная функция не является методом класса, но имеет доступ к закрытым членам этого класса;
* Дружественная функция может быть перегружена, но не может быть унаследована.

Особенности дружественных классов:

* Дружественный класс объявляется внутри класса, но определяется вне класса;
* Дружественный класс имеет доступ к закрытым членам класса, в котором он объявлен;
* Дружественный класс не наследует члены и методы класса, в котором он объявлен;
* Дружественный класс может быть определен как структура или как класс.

1. **Каким образом можно перегрузить унарные операции?** Унарные операции могут быть перегружены внутри класса с помощью функций-членов. Для перегрузки унарной операции необходимо определить функцию с именем операции, которая принимает ноль аргументов и возвращает значение. Пример: **MyClass& operator++();**
2. **Сколько операндов должна иметь унарная функция-операция, определяемая внутри класса?** Унарная функция-операция, определяемая внутри класса, должна иметь один операнд, т.е. объект этого класса.
3. **Сколько операндов должна иметь унарная функция-операция, определяемая вне класса?** Унарная функция-операция, определяемая вне класса, должна иметь один операнд, т.е. объект класса.
4. **Сколько операндов должна иметь бинарная функция-операция, определяемая внутри класса?** Бинарная функция-операция, определяемая внутри класса, должна иметь два операнда, т.е. объекты этого класса.
5. **Сколько операндов должна иметь бинарная функция-операция, определяемая вне класса?** Бинарная функция-операция, определяемая вне класса, должна иметь два операнда, т.е. объекты класса.
6. **Чем отличается перегрузка префиксных и постфиксных унарных операций?** Отличие лишь в ложно передаваемом значении в функцию.
7. **Каким образом можно перегрузить операцию присваивания?** Операция присваивания можно перегрузить с помощью функции-члена с именем operator=(). Нужно каждому полю одного объекта присвоить поля из передаваемого объекта.
8. **Что должна возвращать операция присваивания?** Операция присваивания должна возвращать ссылку на объект, т.е. Student&.
9. **Каким образом можно перегрузить операции ввода-вывода?** Операции ввода-вывода можно перегрузить с помощью функций friend std::istream& operator>>(std::istream& input, Student& s) и friend std::ostream& operator<<(std::ostream& output, const Student& s), которые принимают в качестве параметров объекты istream и ostream соответственно, и объект класса Student. Операция >> считывает значения для полей объекта Student из входного потока, а операция << выводит значения полей объекта в выходной поток.
10. **В программе описан класс class Student {… Student& operator++(); …. }; и определен объект этого класса Student s; Выполняется операция ++s; Каким образом, компилятор будет воспринимать вызов функции-операции?** Компилятор будет воспринимать вызов операции инкремента как вызов метода-члена prefix operator++() класса Student для объекта s.
11. **В программе описан класс class Student {… friend Student& operator ++( Student&); …. }; и определен объект этого класса Student s; Выполняется операция ++s; Каким образом, компилятор будет воспринимать вызов функции-операции?** Компилятор будет воспринимать вызов операции инкремента как вызов дружественной функции prefix operator++(Student&) для объекта s.
12. **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?** Компилятор будет воспринимать вызов операции сравнения как вызов метода-члена bool operator<(Student&) класса Student для объекта a с аргументом b.
13. **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?** Компилятор будет воспринимать вызов операции сравнения как вызов дружественной функции bool operator>(const Person&, Person&) с аргументами a и b для объекта класса Student a.

**Диаграмма класса**

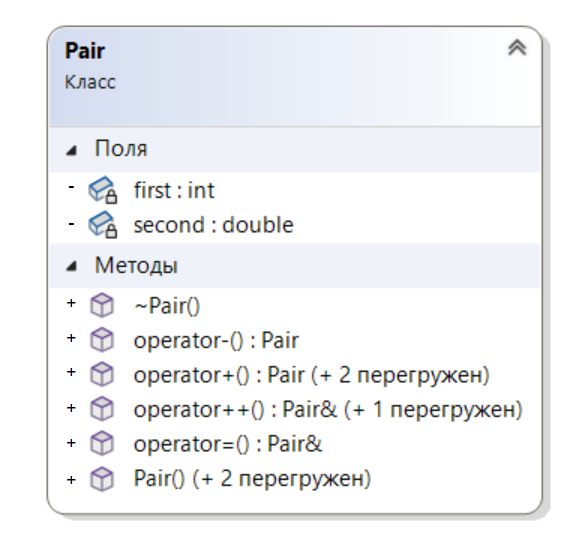


Рисунок 1 – класс Pair

**Описание класса**

#pragma once

#include <iostream>

#include <iomanip>

using namespace std;

class Pair

{

friend ostream& operator<<(ostream& out, const Pair& ob);

friend istream& operator>>(istream& in, Pair& ob);

private:

int first;

double second;

public:

Pair();

Pair(int first, double second);

Pair(const Pair& ob);

Pair operator-(Pair& pair) const;

Pair operator+(Pair& pair) const;

Pair operator+(const int data) const;

Pair operator+(const double data) const;

Pair& operator++();

Pair operator++(int);

Pair& operator=(const Pair& pair);

~Pair();

};

**Определение методов**

#include "Pair.h"

Pair::Pair()

{

first = 0;

second = 0;

}

Pair::Pair(int first, double second)

{

this->first = first;

this->second = second;

}

Pair::Pair(const Pair& ob)

{

this->first = ob.first;

this->second = ob.second;

}

Pair Pair::operator-(Pair& pair) const

{

Pair temp = \*this;

temp.first -= pair.first;

temp.second -= pair.second;

return temp;

}

Pair Pair::operator+(Pair& pair) const

{

Pair temp = \*this;

temp.first += pair.first;

temp.second += pair.second;

return temp;

}

Pair Pair::operator+(int data) const

{

Pair temp = \*this;

temp.first += data;

return temp;

}

Pair Pair::operator+(double data) const

{

Pair temp = \*this;

temp.second += data;

return temp;

}

Pair& Pair::operator++()

{

++first;

++second;

return \*this;

}

Pair Pair::operator++(int)

{

Pair temp = \*this;

++first;

++second;

return temp;

}

Pair& Pair::operator=(const Pair& pair)

{

if (this == &pair)

return \*this;

first = pair.first;

second = pair.second;

return \*this;

}

ostream& operator<<(ostream& out,const Pair& ob)

{

out << "(" << ob.first << ":" << ob.second << ")";

return out;

}

istream& operator>>(istream& in, Pair& ob)

{

cout << "first(int): "; in >> ob.first;

cout << "second(double): "; in >> ob.second;

return in;

}

Pair::~Pair()

{

}

**Результаты работы программы**

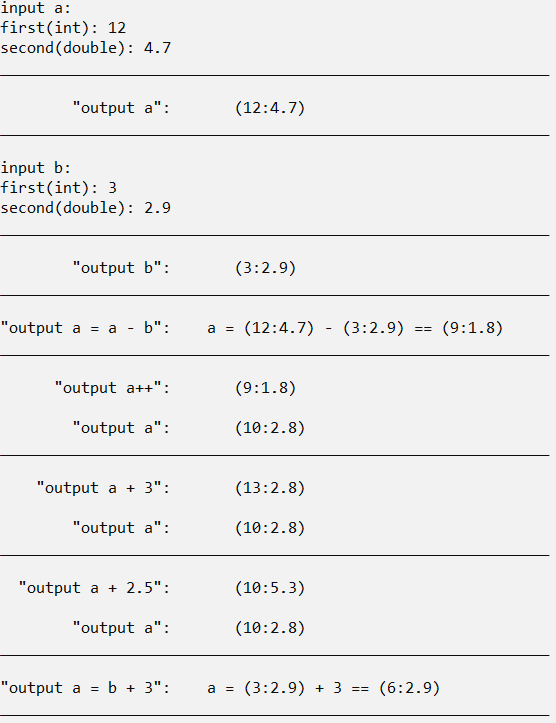


Рисунок 2 – результат работы программы