Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №18.3**

Дисциплина: основы алгоритмизации и программирования

Тема: “Классы и объекты. Перегрузка операций.”

Вариант 15

Выполнил:

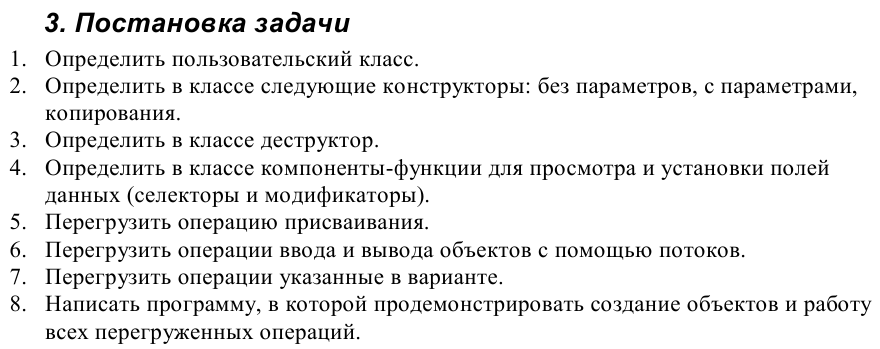
студент группы ИВТ-20-2Б Чувашев Максим

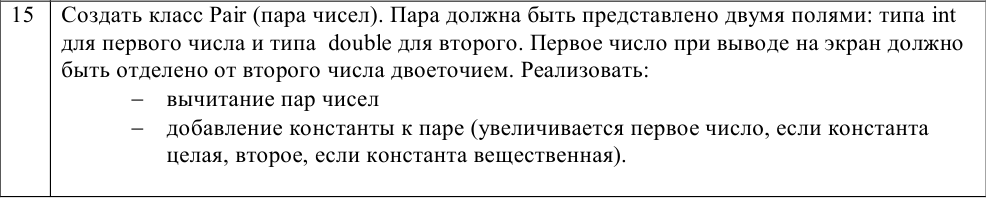
Проверила: доцент кафедры ИТАС

Полякова О.А.

**Пермь, 2021**

**Постановка задачи**





**Анализ задачи**

1. Используемые типы данных
   1. Double
   2. Int
   3. Pair
2. Действия над используемыми данными
   1. Int, double – арифметические операции
3. Для решения задачи данные были представлены в следующем виде:
   1. Данные вводятся через конструктор класса pair, а так же геттеры и сеттеры.

class Pair

{

private:

int first;

double second;

public:

// конструктор копирования без параметров

Pair()

{

first = 0; // При создании объекта поле заполняется 0

second = 0; // При создании объекта поле заполняется 0

};

// конструктор копирования с параметрами

Pair(int f, double s)

{

first = f; // При создании объекта поле заполняется переданным в качестве параметра числом

second = s; // При создании объекта поле заполняется переданынм в качестве параметра числом

}

// конструктор копирования

Pair(const Pair& other)

{

this->first = other.first;

this->second = other.second;

}

// деструктор

~Pair()

{

};

int get\_first()

{

return first;

}

int get\_second()

{

return second;

}

void set\_first(int m)

{

first = m;

}

void set\_second(int s)

{

second = s;

}

//перегруженные операции

Pair & operator=(const Pair&);

Pair operator -(const Pair&); // перегрузка оператора "-", вычитание полей объектов

Pair operator +(double constant); // перегрузка оператора "+" прибавление константы дабл к полям объекта класса

//глобальные функции ввода-вывода

friend istream& operator>>(istream& in, Pair& other); // объявление дружественного оператора ввода

friend ostream& operator<<(ostream& out, const Pair& other); // объявление дружественного оператора вывода

};

1. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Для ввода пары чисел используются геттеры, а так же оператор ввода cin
   2. Для вывода данных на консоль используется оператор вывода cout
   3. Для вывода данных на консоль также используется перегрузка оператора <<.

//перегрузка глобальной функции-операции вывода

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

{

return (out << other.first << " : " << other.second);

}

* 1. Для ввода минут и секунд также может быть использована перегрузка оператора >>

//перегрузка глобальной функции-операции ввода

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

{

cout << "first?";

in >> other.first;

cout << "second?";

in >> other.second;

return in;

}

1. Поставленные задачи будут решены следующими действиями:
   1. В перегрузке оператора + выполняется сложение обеих полей объекта класса и возвращается указатель на получившуюся пару.

Pair Pair::operator+(double constant)

{

if (constant == int(constant))

{

this->first = this->first + constant;

}

else

{

this->second = this->second + constant;

}

return \*this;

}

* 1. Подобным образом перегружен оператор вычитание

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

{

Pair tmp;

tmp.first = this->first - other.first;

tmp.second = this->second - other.second;

return tmp;

}

* 1. Подобным образом реализована перегрузка оператора равенства

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

{

if(&other==this) return \*this; //проверка на самоприсваивание

this->first = other.first;

this->second = other.second;

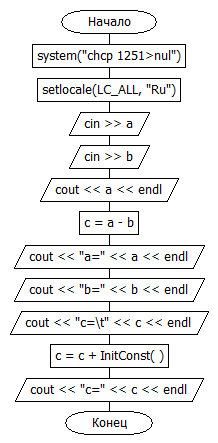
return \*this;

}

**Блок-схема**

**Sourse.cpp**





**Pair.cpp**



**Pair.h**



**Код**

#include "Pair.h"

#include <iostream>

using namespace std;

double InitConst()

{

double constant;

cout << "Введите константу которую хотите добавить:\t";

cin >> constant;

return constant;

}

void main()

{

system("chcp 1251>nul");

setlocale(LC\_ALL, "Ru");

Pair a;//конструктор без параметров

Pair b; //конструктор без параметров

Pair c; //конструктор без параметров

cin>>a;//ввод переменной

cin>>b;//ввод переменной

cout<<a<<endl;//вывод переменной

c = a - b;// вычитание двух объектов класа

cout << "a=" << a << endl; //вывод переменной

cout << "b=" << b << endl; //вывод переменной

cout << "c=\t" << c << endl; //вывод переменной

c = c + InitConst();

cout<<"c="<<c<<endl; //вывод переменной

}

#include "Pair.h"

#include <iostream>

using namespace std;

//перегрузка операции присваивания

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

{

if(&other==this) return \*this; //проверка на самоприсваивание

this->first = other.first;

this->second = other.second;

return \*this;

}

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

{

Pair tmp;

tmp.first = this->first - other.first;

tmp.second = this->second - other.second;

return tmp;

}

Pair Pair::operator+(double constant)

{

if (constant == int(constant))

{

this->first = this->first + constant;

}

else

{

this->second = this->second + constant;

}

return \*this;

}

//перегрузка глобальной функции-операции ввода

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

{

cout << "first?";

in >> other.first;

cout << "second?";

in >> other.second;

return in;

}

//перегрузка глобальной функции-операции вывода

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

{

return (out << other.first << " : " << other.second);

}

#pragma once

#include <iostream>

using namespace std;

class Pair

{

private:

int first;

double second;

public:

// конструктор копирования без параметров

Pair()

{

first = 0; // При создании объекта поле заполняется 0

second = 0; // При создании объекта поле заполняется 0

};

// конструктор копирования с параметрами

Pair(int f, double s)

{

first = f; // При создании объекта поле заполняется переданным в качестве параметра числом

second = s; // При создании объекта поле заполняется переданынм в качестве параметра числом

}

// конструктор копирования

Pair(const Pair& other)

{

this->first = other.first;

this->second = other.second;

}

// деструктор

~Pair()

{

};

int get\_first()

{

return first;

}

int get\_second()

{

return second;

}

void set\_first(int m)

{

first = m;

}

void set\_second(int s)

{

second = s;

}

//перегруженные операции

Pair & operator=(const Pair&);

Pair operator -(const Pair&); // перегрузка оператора "-", вычитание полей объектов

Pair operator +(double constant); // перегрузка оператора "+" прибавление константы дабл к полям объекта класса

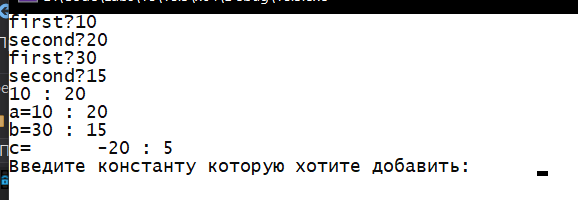
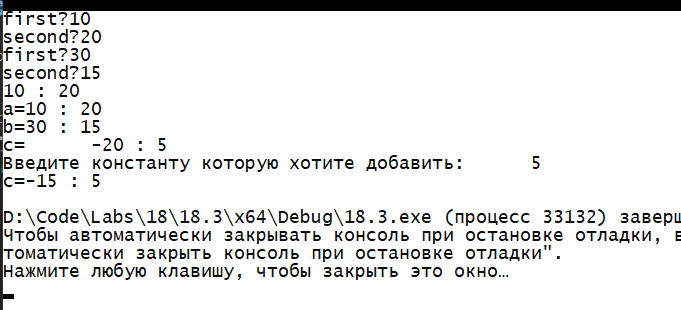
//глобальные функции ввода-вывода

friend istream& operator>>(istream& in, Pair& other); // объявление дружественного оператора ввода

friend ostream& operator<<(ostream& out, const Pair& other); // объявление дружественного оператора вывода

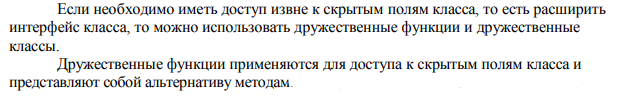
};

**Скриншоты**

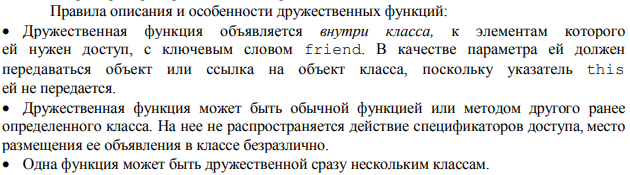
1. 
2. 

**Контрольные вопросы**



1. 

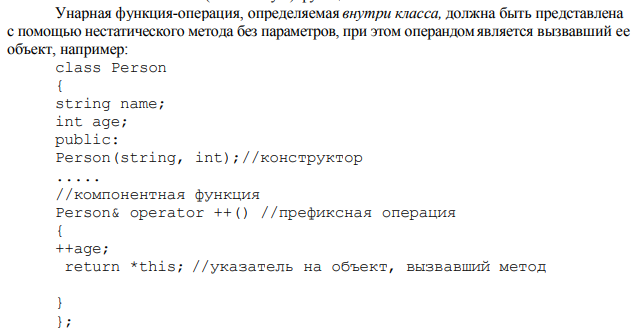


2. 



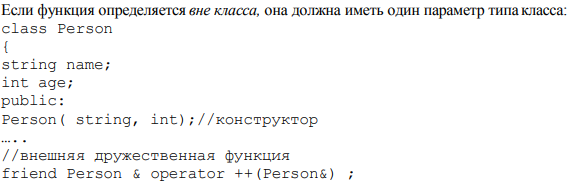
3. 

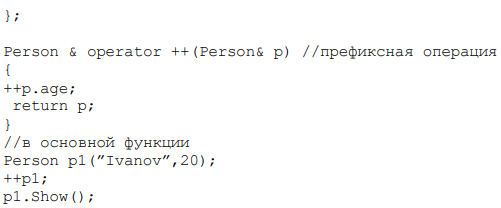


4.

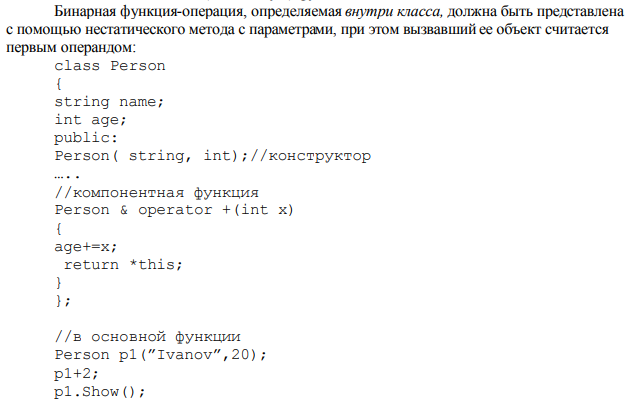


16

5.

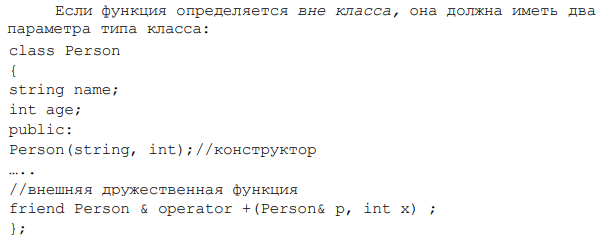




6.



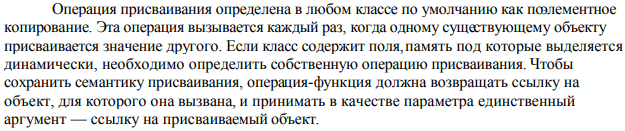
17

7. 



8. 

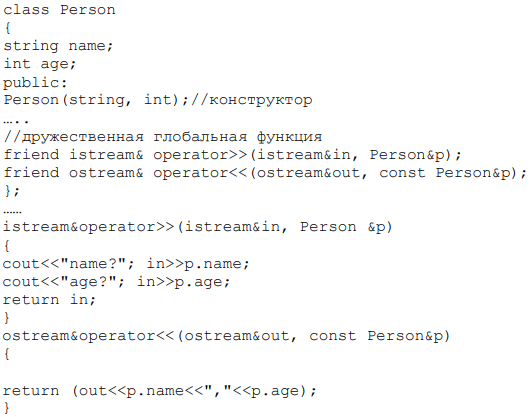


9. 

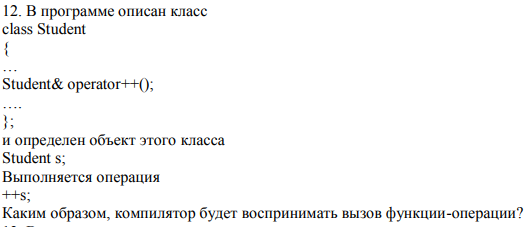


10. Ссылку на объект, для которого операция была вызвана.

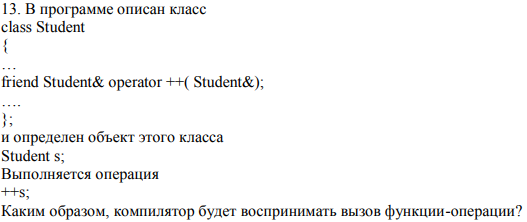




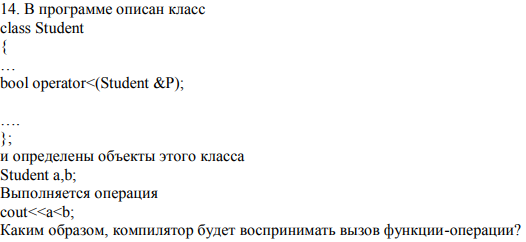
18



12. Будет осуществлён вызов функции перегрузки оператора префиксного инкремента.

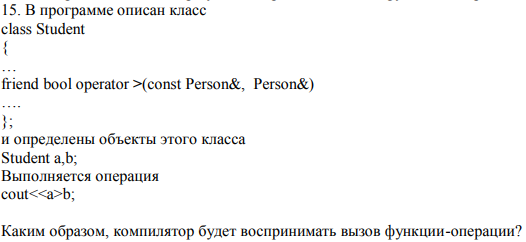


13. Будет осуществлён вызов функции перегрузки оператора постфиксного инкремента.



14. Будет осуществлён вызов функции перегрузки оператора сравнения (<), затем через cout будет выведен результат сравнения (1 или 0).

19



15. Будет осуществлён вызов функции перегрузки оператора сравнения (>), затем через cout будет выведен результат сравнения (1 или 0).