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: «Информатика»

Тема: “Объектно-ориентированное программирование. Перегрузка операций”

Вариант 12

Выполнила:

Студент группы ИВТ-20-2Б Ананина Арина Юрьевна

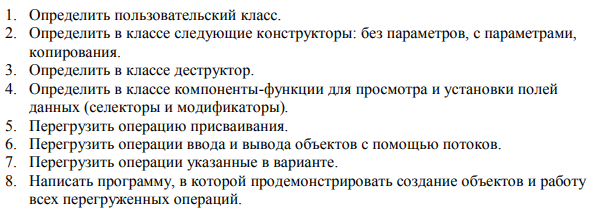
Проверила:

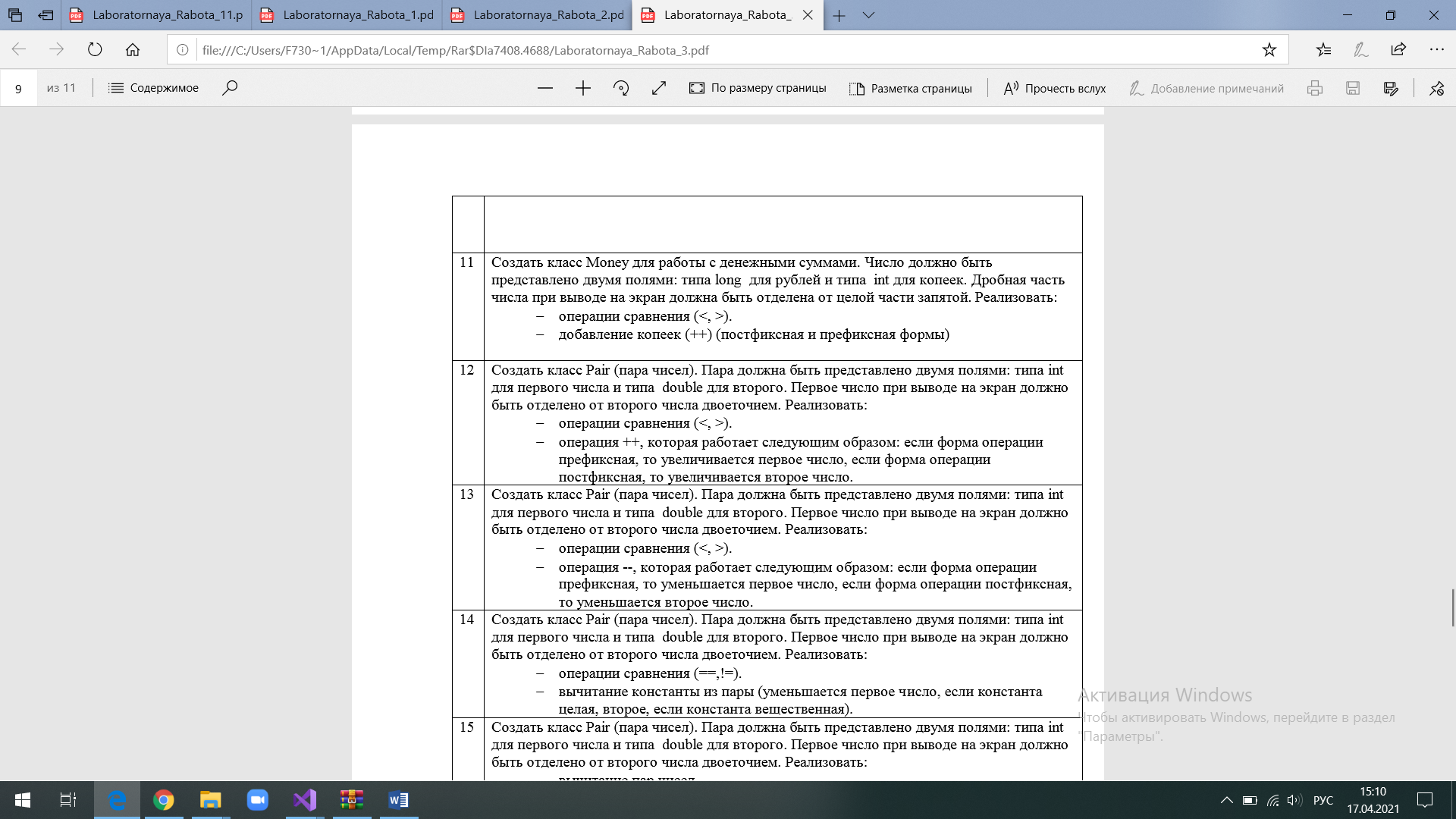
Доцент кафедры ИТАС

Полякова О.А.

**Пермь, 2021**

**Постановка задачи**





**Анализ задачи**

**1.** Для решения задачи необходимо:

**1.1.** Организовать класс Time для хранения минут (mins), секунд (secs).

**1.2.** Организовать необходимые методы для ввода данных в поля mins и secs: гетторы, сетторы, конструкторы, деструктор.

**1.3.** Организовать перегрузку оператора >> дружественной классу Time.

**1.4.** Организовать перегрузку оператора << дружественной классу Time.

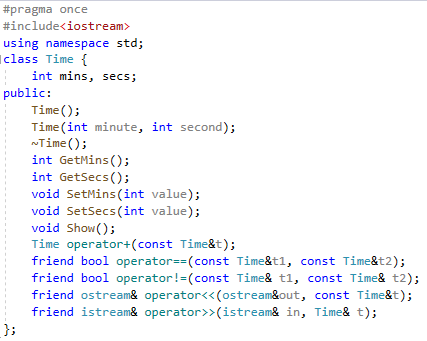
**1.5.** Организовать перегрузку оператора == дружественной классу Time.

**1.6.** Организовать перегрузку оператора != дружественной классу Time.

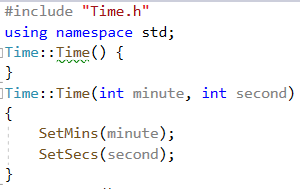
**1.7.** Организовать перегрузку оператора + дружественной классу Time.

**2.** В ходе работы были использованы следующие типы данных:

**2.1.** Организовать класс Time для хранения минут (mins), секунд (secs). в заголовочном файле Number.h.



**2.2.** Организовать соответствующий файл Number.cpp, в котором будут описаны методы, конструкторы и дружественные функции класса Number.



3

**2.3.** Переменные типа int: min, sec, где min – введённое количество минут, sec – введённое количество секунд.

int min, sec;

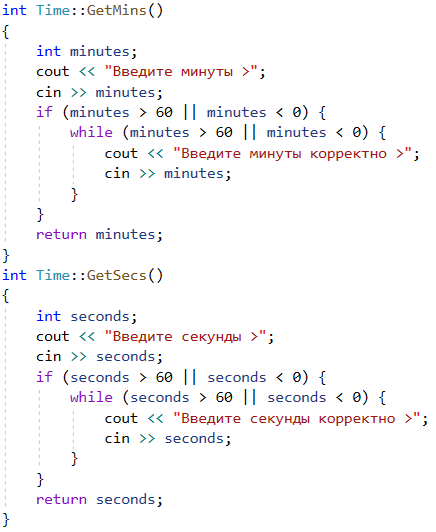
**2.4.** Объекты obj, obj2, tmp класса Time.

Time obj, tmp;

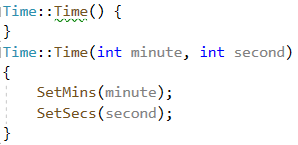
Time obj2(min, sec);

**3.** Для решения задачи данные были представлены в следующем виде:

**3.1.** Данные вводятся через объекты класса Time через гетторы.

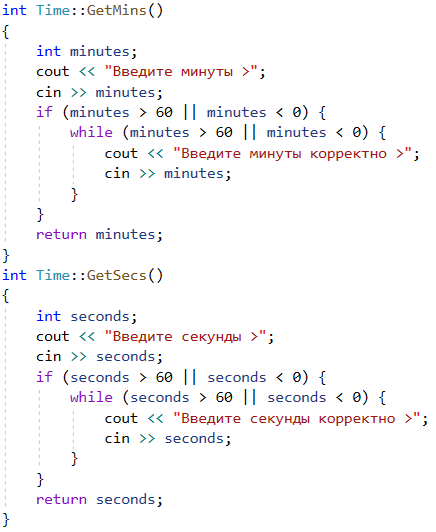


**3.2.** Данные также вводятся через объекты класса Time через конструктор с параметрами.

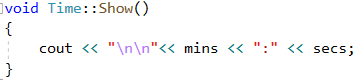


**4.** Для операций ввода и вывода использовались следующие операторы и функции:

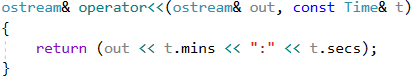
**4.1.** Для ввода минут и секунд используются гетторы (методы, описанные в классе для ввода), в которых используется функция cin.



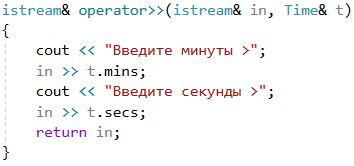
**4.2.** Для вывода данных на консоль используется метод Show().



**4.3.** Для вывода данных на консоль также используется перегрузка оператора <<.

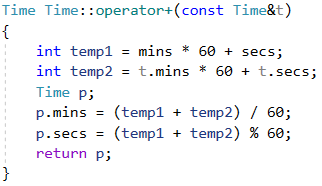


**4.4.** Для ввода минут и секунд также может быть использована перегрузка оператора >>



**5.** Поставленные задачи будут решены следующими действиями:

**5.1.** В перегрузке оператора + выполняется сначала перевод в секунды для объектов класса Time, а затем вычисляются минуты и секунды целочисленным делением на 60 и взятием остатка от деления на 60 соответственно.



**5.2.** В перегрузке оператора ==/!= выполняется сравнение минут, а затем секунд для объектов класса Time и возвращается значение true или false.

**Код**

Pair.h

#pragma once

#include <iostream>

#include <string>

using namespace std;

class Pair

{

int First;

double Second;

public:

Pair();

Pair(int First, double Second);

Pair(const Pair& p);

~Pair();

void Show();

void Set\_First(int N);

void Set\_Second(double D);

bool operator< (const Pair& r);

bool operator> (const Pair& r);

Pair& operator++ ();

Pair operator++ ( int );

};

Pair.cpp

#include <iostream>

#include "Pair.h"

#include <string>

using namespace std;

Pair::Pair()

{

First = 1;

Second = 1;

}

Pair::Pair(int First, double Second)

{

this->First = First;

this->Second = Second;

}

Pair::Pair(const Pair& p)

{

this->First = p.First;

this->Second = p.Second;

}

Pair::~Pair() {}

void Pair::Show()

{

cout << First << " : " << Second << endl;

}

void Pair::Set\_First(int N)

{

First = N;

}

void Pair::Set\_Second(double D)

{

Second = D;

}

bool Pair::operator< (const Pair& r) {

return (First < r.First) && (Second < r.Second);

}

bool Pair::operator> (const Pair& r) {

return (First > r.First) && (Second > r.Second);

}

Pair& Pair::operator++ ()

{

First++;

return \*this;

}

Pair Pair::operator++ (int)

{

Pair old(First, Second);

Second++;

return old;

}

Main.cpp

#include <iostream>

#include "Pair.h"

#include <string>

using namespace std;

int main()

{

Pair A;

A.Show();

Pair B(23, 20.5);

B.Show();

Pair C(B);

C.Show();

B++;

B.Show();

Pair incB = ++C;

incB.Set\_Second(100.5);

incB.Show();

if (incB < B) {

cout << "incB < B" << endl;

}

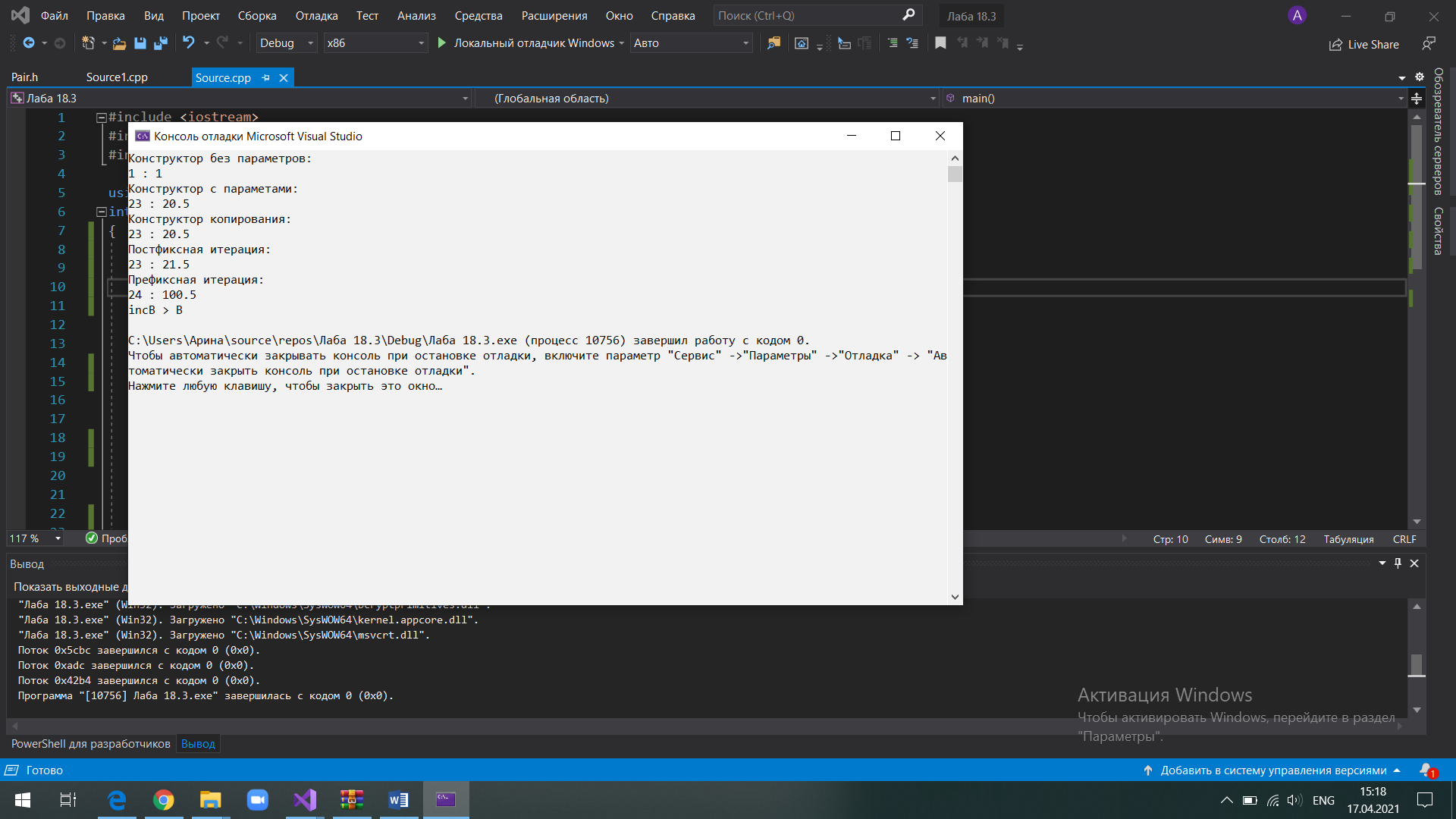
if (incB > B) {

cout << "incB > B" << endl;

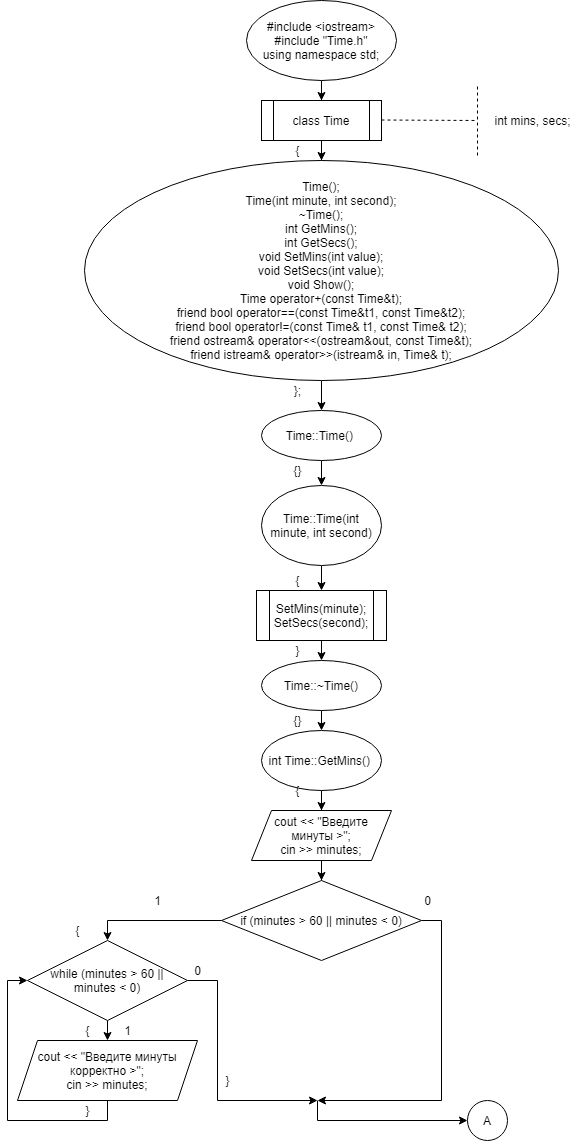
}

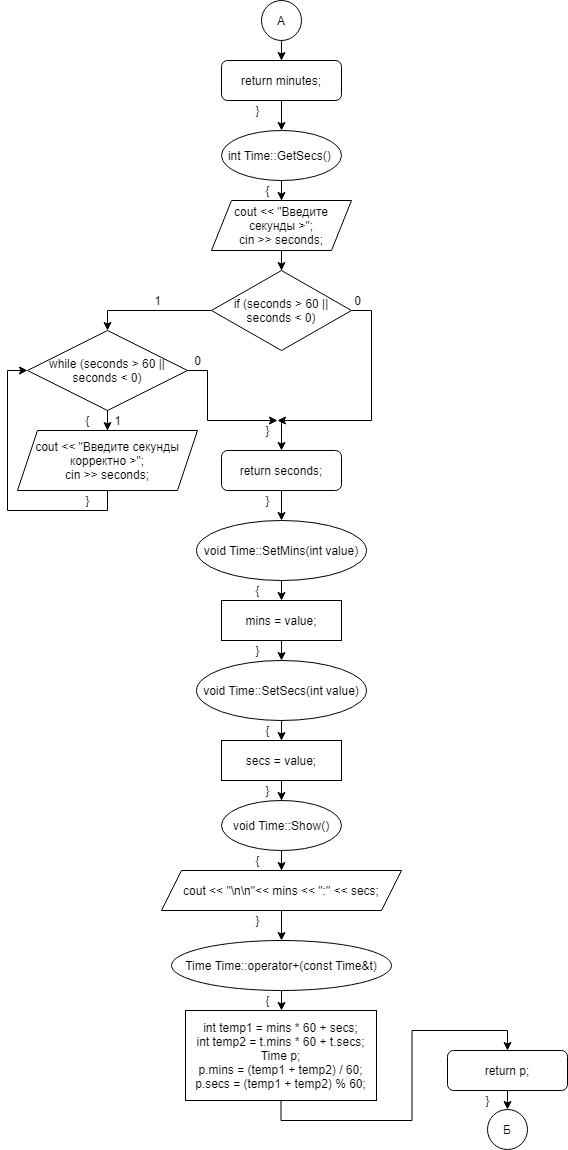
}

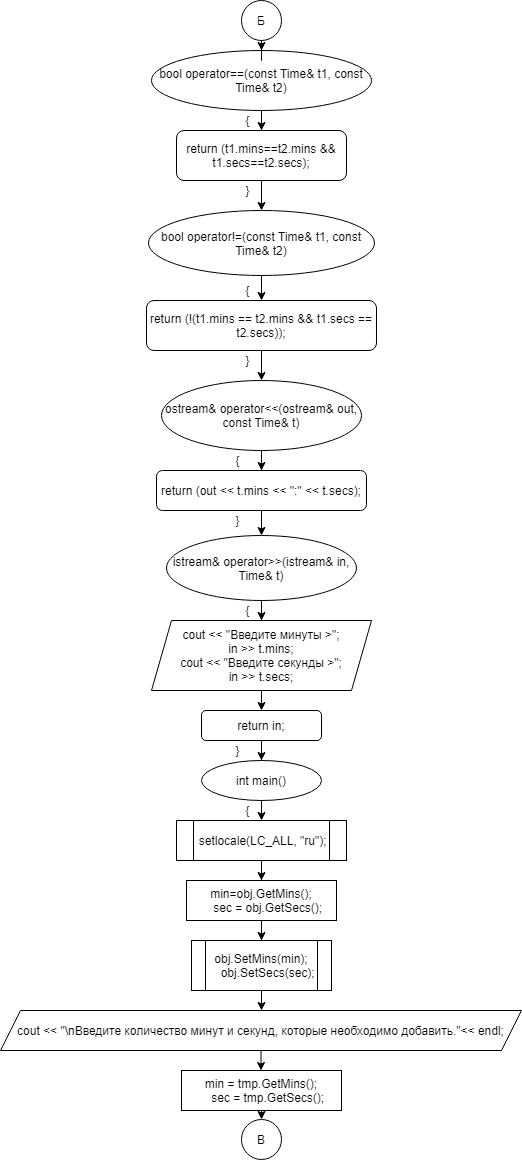
**Выполнение**

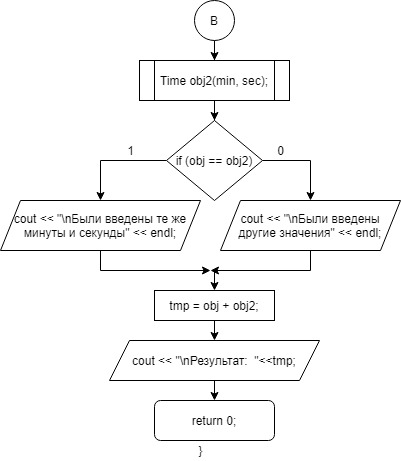


**Блок-схема**



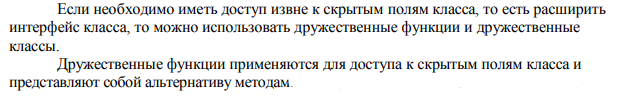




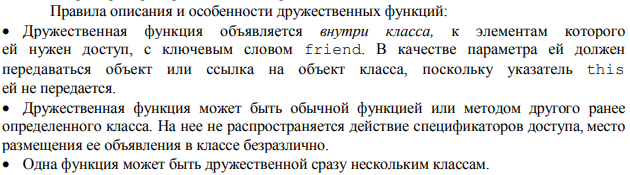


**Контрольные вопросы**



1. 

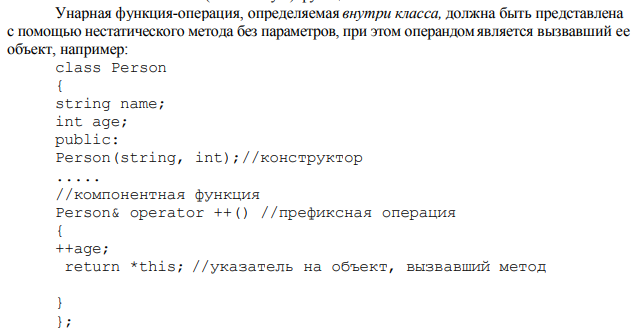


2. 

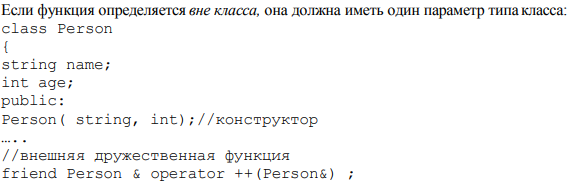


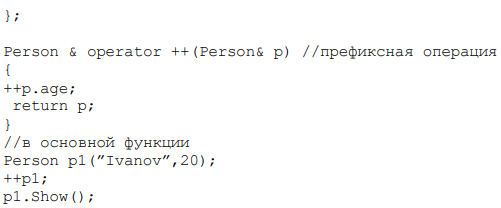
3. 



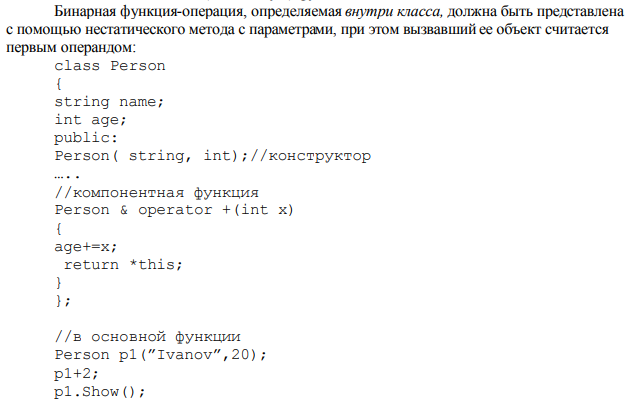
4.



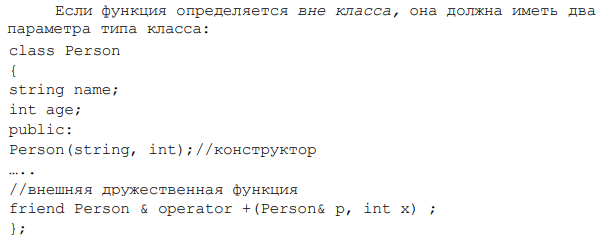
5.





6.

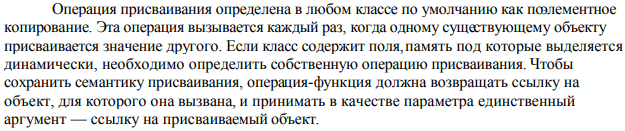


7. 



8. 

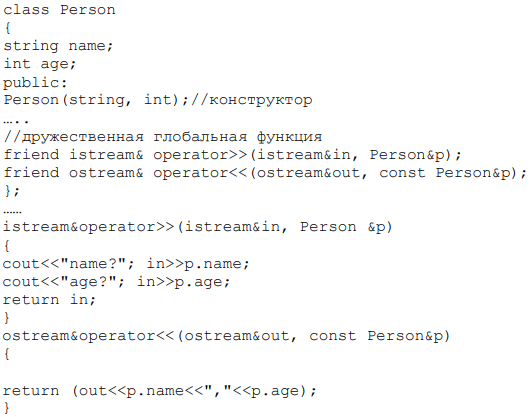


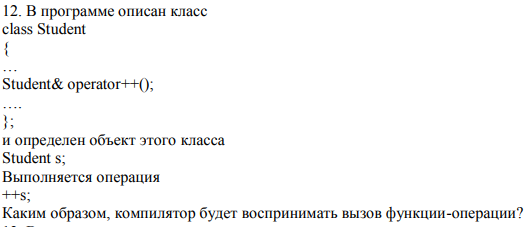
9. 



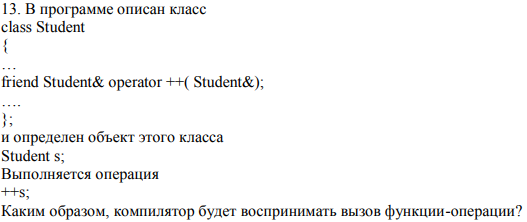
10. Ссылку на объект, для которого операция была вызвана.



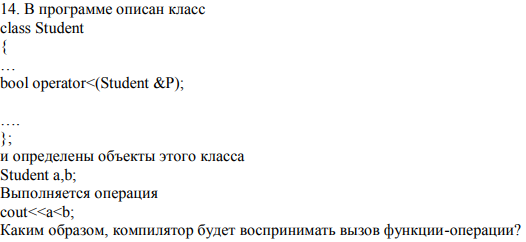




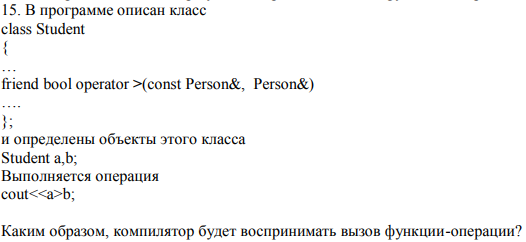
12. Будет осуществлён вызов функции перегрузки оператора префиксного инкремента.



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).



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