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 19

Выполнил:

Студент группы РИС-20-2Б

Пономарёв Артём Викторович

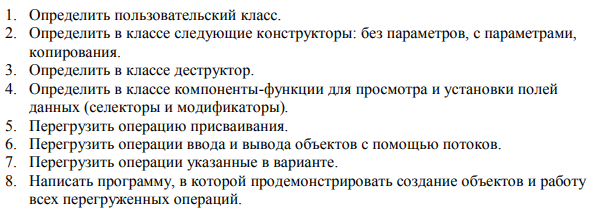
Проверила:

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**



2

**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.

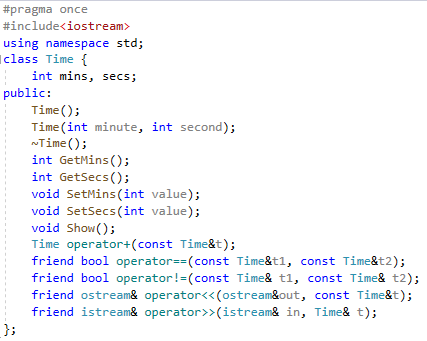
**1.5.** Организовать перегрузку оператора == дружественной классу Time.

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

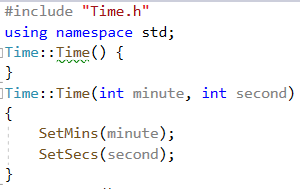
**1.7.** Организовать перегрузку оператора + дружественной классу Time.

**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;

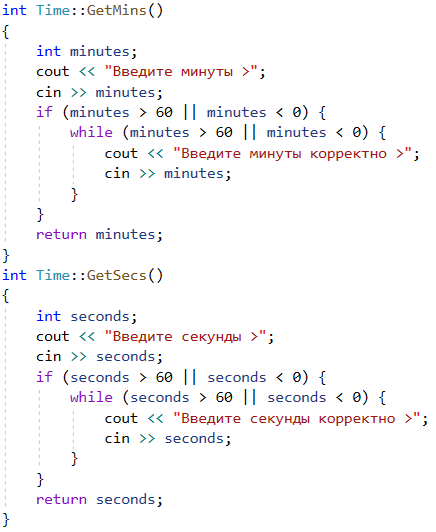
**2.4.** Объекты obj, obj2, tmp класса Time.

Time obj, tmp;

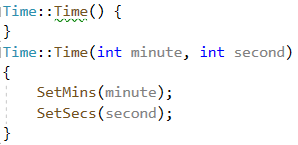
Time obj2(min, sec);

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

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



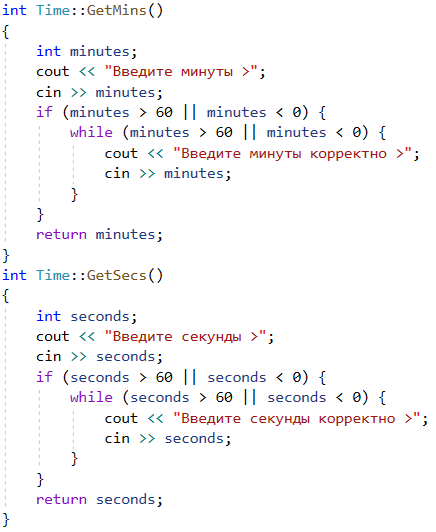
**3.2.** Данные также вводятся через объекты класса Time через конструктор с параметрами.



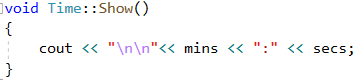
4

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

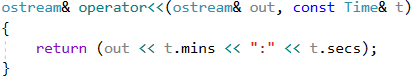
**4.1.** Для ввода минут и секунд используются гетторы (методы, описанные в классе для ввода), в которых используется функция cin.



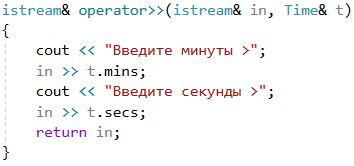
**4.2.** Для вывода данных на консоль используется метод Show().



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



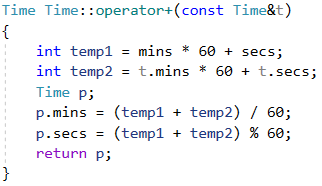
**4.4.** Для ввода минут и секунд также может быть использована перегрузка оператора >>



5

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

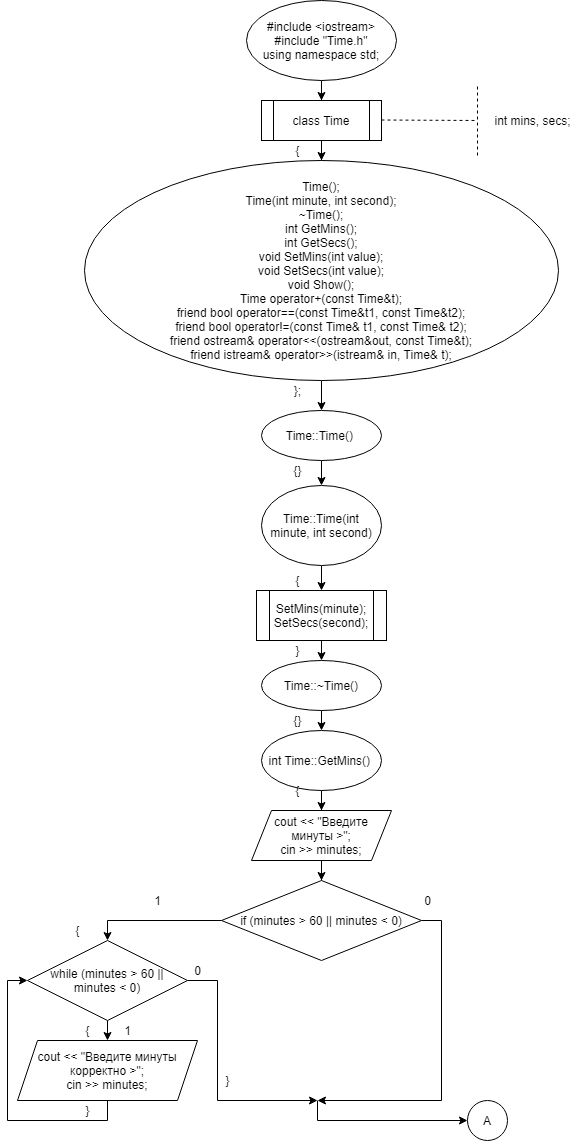
**5.1.** В перегрузке оператора + выполняется сначала перевод в секунды для объектов класса Time, а затем вычисляются минуты и секунды целочисленным делением на 60 и взятием остатка от деления на 60 соответственно.



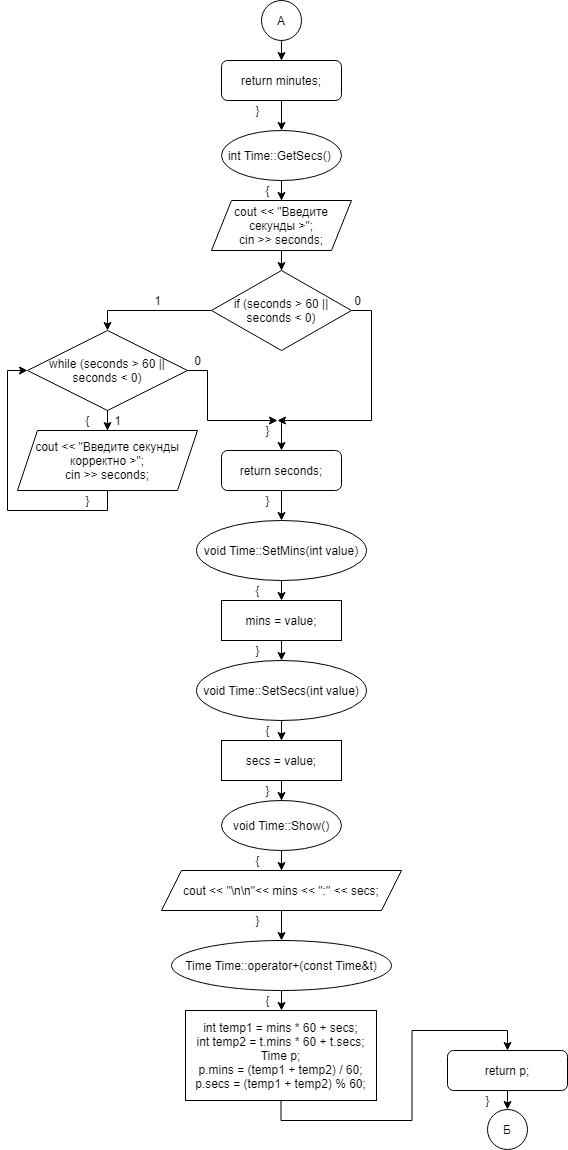
**5.2.** В перегрузке оператора ==/!= выполняется сравнение минут, а затем секунд для объектов класса Time и возвращается значение true или false.

6

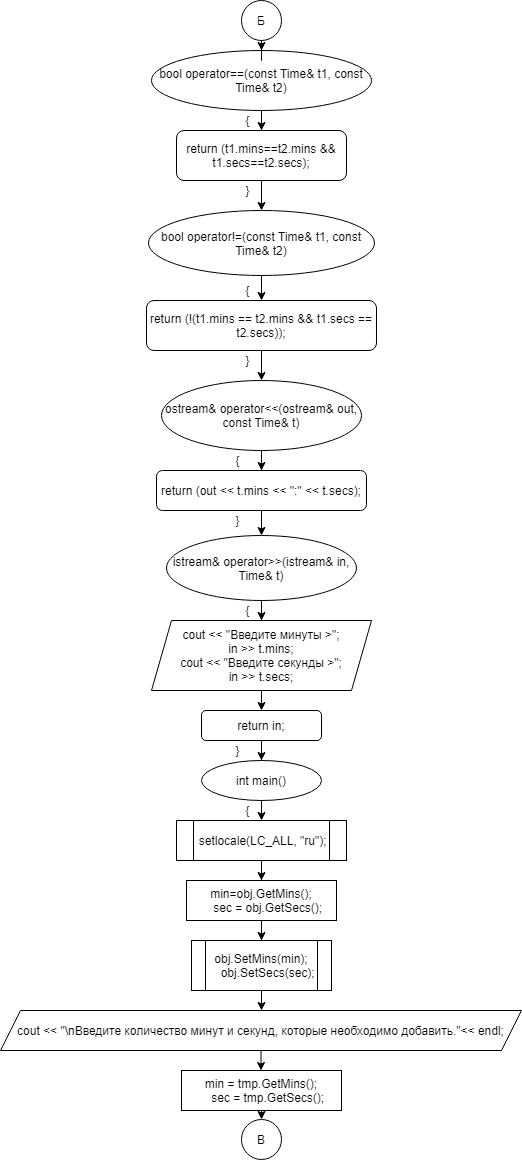
**Блок-схема**



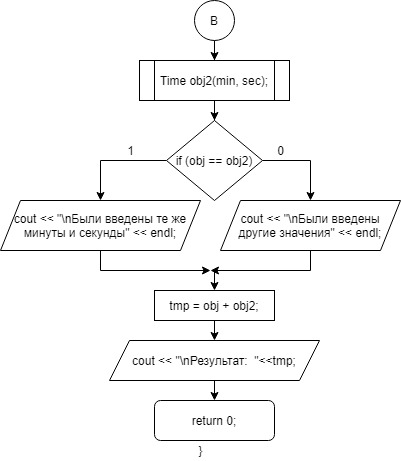
7



8



9



10

**Код**

#include <iostream>

#include "Time.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "ru");

Time obj, tmp;

int min, sec;

min=obj.GetMins();

sec = obj.GetSecs();

obj.SetMins(min);

obj.SetSecs(sec);

cout << "\nВведите количество минут и секунд, которые необходимо добавить."<< endl;

min = tmp.GetMins();

sec = tmp.GetSecs();

Time obj2(min, sec);

//проверка перегрузки оператора равенства

if (obj == obj2)

cout << "\nБыли введены те же минуты и секунды" << endl;

else

cout << "\nБыли введены другие значения" << endl;

tmp = obj + obj2;

//tmp.Show();

cout << "\nРезультат: "<<tmp;

}

11

#pragma once

#include<iostream>

using namespace std;

class Time {

int mins, secs;

public:

Time();

Time(int minute, int second);

~Time();

int GetMins();

int GetSecs();

void SetMins(int value);

void SetSecs(int value);

void Show();

Time operator+(const Time&t);

friend bool operator==(const Time&t1, const Time&t2);

friend bool operator!=(const Time& t1, const Time& t2);

friend ostream& operator<<(ostream&out, const Time&t);

friend istream& operator>>(istream& in, Time& t);

};

12

#include "Time.h"

using namespace std;

Time::Time() {

}

Time::Time(int minute, int second)

{

SetMins(minute);

SetSecs(second);

}

Time::~Time() {

}

int Time::GetMins()

{

int minutes;

cout << "Введите минуты >";

cin >> minutes;

if (minutes > 60 || minutes < 0) {

while (minutes > 60 || minutes < 0) {

cout << "Введите минуты корректно >";

cin >> minutes;

}

}

return minutes;

}

int Time::GetSecs()

{

int seconds;

cout << "Введите секунды >";

cin >> seconds;

if (seconds > 60 || seconds < 0) {

while (seconds > 60 || seconds < 0) {

cout << "Введите секунды корректно >";

cin >> seconds;

}

}

return seconds;

}

void Time::SetMins(int value)

{

mins = value;

}

void Time::SetSecs(int value)

{

secs = value;

}

void Time::Show()

{

cout << "\n\n"<< mins << ":" << secs;

}

Time Time::operator+(const Time&t)

{

int temp1 = mins \* 60 + secs;

int temp2 = t.mins \* 60 + t.secs;

Time p;

p.mins = (temp1 + temp2) / 60;

p.secs = (temp1 + temp2) % 60;

return p;

}

bool operator==(const Time& t1, const Time& t2)

{

return (t1.mins==t2.mins && t1.secs==t2.secs);

}

bool operator!=(const Time& t1, const Time& t2)

{

13

return (!(t1.mins == t2.mins && t1.secs == t2.secs));

}

ostream& operator<<(ostream& out, const Time& t)

{

return (out << t.mins << ":" << t.secs);

}

istream& operator>>(istream& in, Time& t)

{

cout << "Введите минуты >";

in >> t.mins;

cout << "Введите секунды >";

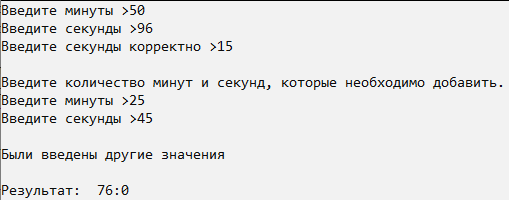
in >> t.secs;

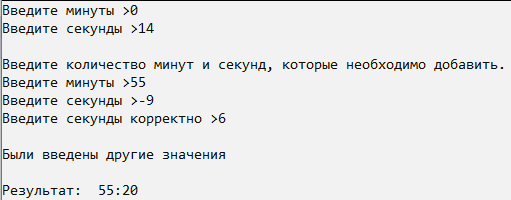
return in;

}

14

**Скриншоты**

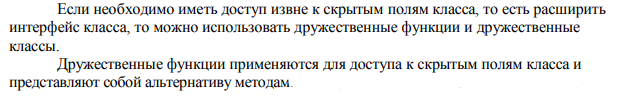




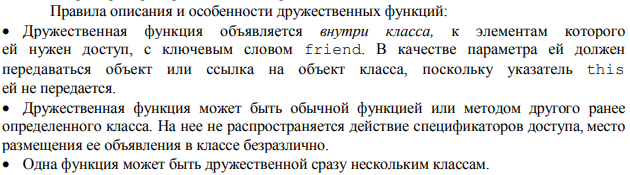
15

**Контрольные вопросы**



1. 

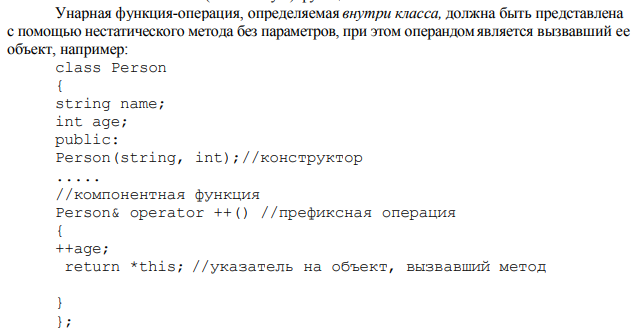


2. 



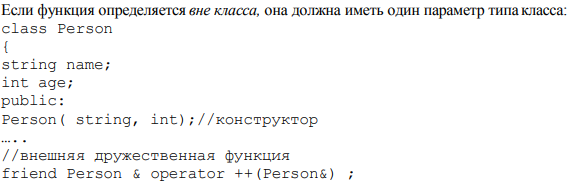
3. 

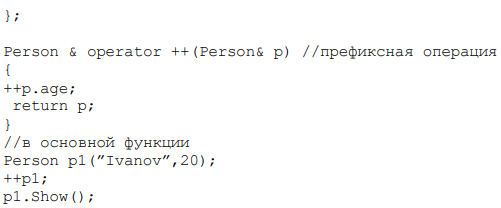


4.

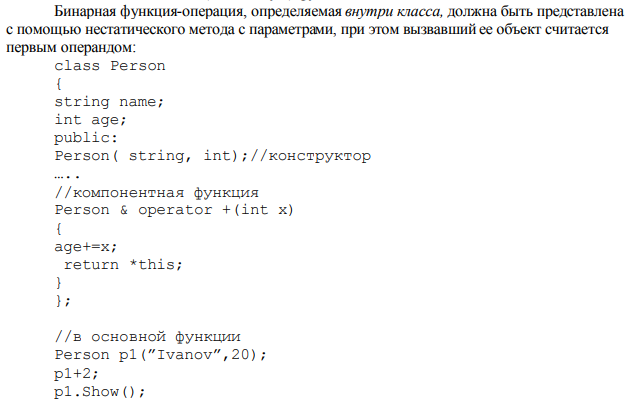


16

5.

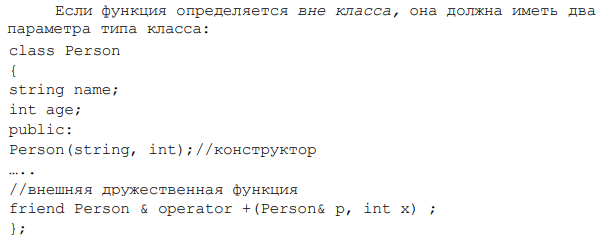




6.



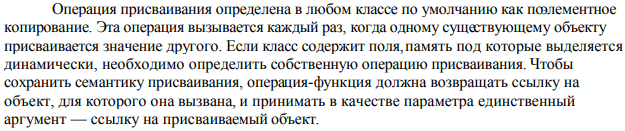
17

7. 



8. 

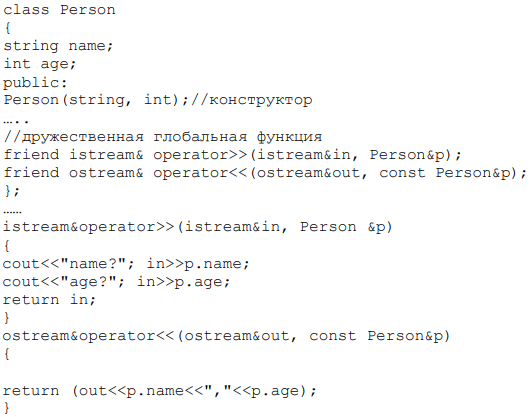


9. 

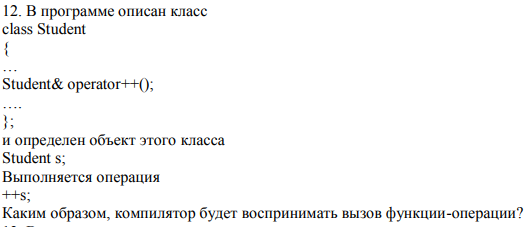


10. Ссылку на объект, для которого операция была вызвана.

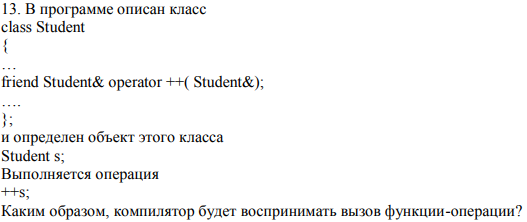




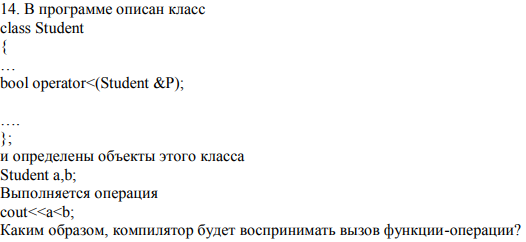
18



12. Будет осуществлён вызов функции перегрузки оператора префиксного инкремента.

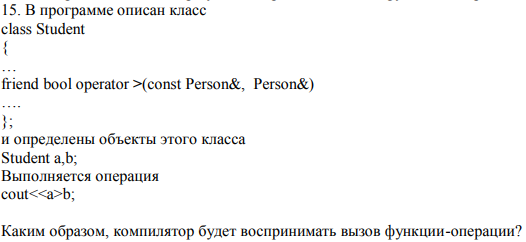


13. Будет осуществлён вызов функции перегрузки оператора постфиксного инкремента.



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).

20