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**

**по лабораторной работе № 18.3**

Дисциплина: Информатика

Тема: «Объектно-ориентированное программирование. Перегрузка операций»

Вариант 1

Выполнил работу:

студент группы РИС-20-1Б

Азмагулов Артём Вадимович

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

Пермь

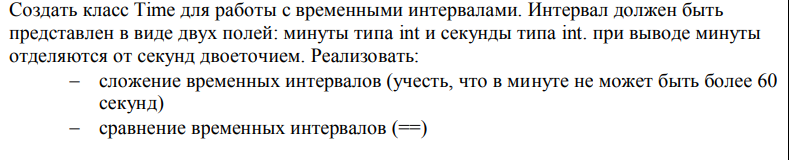
2021

**Цель работы**

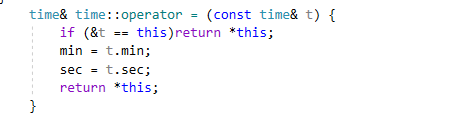
* 1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
  2. Использование перегруженных операций в классах.

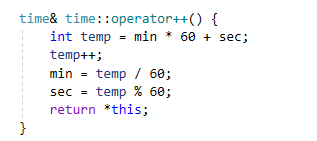
**Постановка задачи**

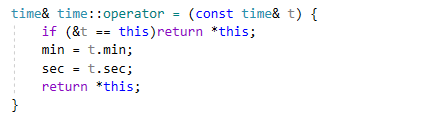
1. Определить пользовательский класс.
2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.
3. Определить в классе деструктор.
4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).
5. Перегрузить операцию присваивания.
6. Перегрузить операции ввода и вывода объектов с помощью потоков.

**Анализ задачи**

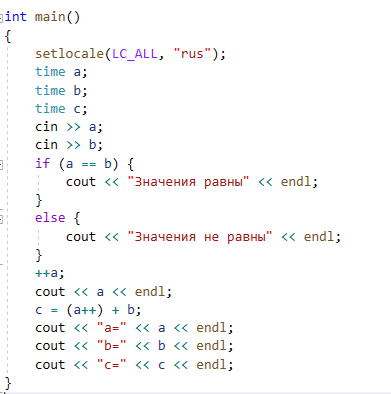
1. Необходимые действия
2. Продемонстрировать работу оператора присваивания



1. Продемонстрировать работу сложения временных промежутков 
2. Продемонстрировать работу оператора сравнения:



1. Продемонстрировать работу всех вышеперечисленных функций в главной функции:



1. Программный код

#include <iostream>

using namespace std;

class time {

public:

int min, sec;

time() { min = 0; sec = 0; }

time(int m, int s) { min = m; sec = s; }

time(const time& t) { min = t.min; sec = t.sec; }

~time() {}

int get\_min() { return min; }

int get\_sec() { return sec; }

void set\_min(int m) { min = m; }

void set\_sec(int s) { sec = s; }

//перегруженные:

time& operator=(const time&);

time& operator++();

time operator++(int);

time operator+(const time&);

bool operator==(const time&);

//глобальные ввода и вывода:

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

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

};

bool time::operator==(const time& t) {

/\*

if (t.min == this->min && t.sec == this->sec) {

return true;

}

else {

return false;

}

\*/

return (t.min == this->min && t.sec == this->sec);

}

time& time::operator = (const time& t) {

if (&t == this)return \*this;

min = t.min;

sec = t.sec;

return \*this;

}

time& time::operator++() {

int temp = min \* 60 + sec;

temp++;

min = temp / 60;

sec = temp % 60;

return \*this;

}

time time::operator++(int)

{

int temp = min \* 60 + sec;

++temp;

min = temp / 60;

sec = temp % 60;

return \*this;

}

time time::operator+(const time& t) {

int temp1 = min \* 60 + sec;

int temp2 = t.min \* 60 + t.sec;

time p;

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

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

return p;

}

istream& operator>>(istream& in, time& t) {

cout << "min?";

in >> t.min;

cout << "sec?";

in >> t.sec;

return in;

}

ostream& operator<<(ostream& out, const time& t) {

return (out << t.min << " : " << t.sec);

}

int main()

{

setlocale(LC\_ALL, "rus");

time a;

time b;

time c;

cin >> a;

cin >> b;

if (a == b) {

cout << "Значения равны" << endl;

}

else {

cout << "Значения не равны" << endl;

}

++a;

cout << a << endl;

c = (a++) + b;

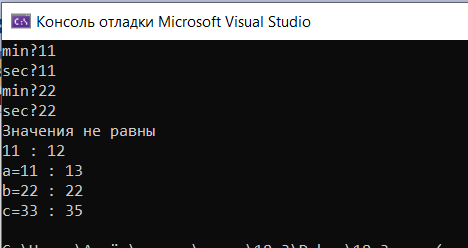
cout << "a=" << a << endl;

cout << "b=" << b << endl;

cout << "c=" << c << endl;

}

1. Демонстрация работы программы



1. Для чего используются дружественные функции и классы?

- Дружественные функции применяются для доступа к скрытым полям класса и представляют собой альтернативу методам.

1. Сформулировать правила описания и особенности дружественных функций.

- Дружественная функция объявляется *внутри класса,* к элементам которого ей нужен доступ, с ключевым словом friend. В качестве параметра ей должен передаваться объект или ссылка на объект класса, поскольку указатель this ей не передается

- Дружественная функция может быть обычной функцией или методом другого ранее определенного класса. На нее не распространяется действие спецификаторов доступа, место размещения ее объявления в классе безразлично

- Одна функция может быть дружественной сразу нескольким классам

1. Каким образом можно перегрузить унарные операции?

Унарную операцию можно перегрузить:

* + - Как компонентную функцию класса
    - Как внешнюю (глобальную) функцию

1. Сколько операндов должна иметь унарная функция-операция, определяемая внутри класса?

- ни одного, операндом является вызвавший операцию объект

1. Сколько операндов должна иметь унарная функция-операция, определяемая вне класса?

- Если функция определяется вне класса*,* она должна иметь один параметр типа класса

1. Сколько операндов должна иметь бинарная функция-операция, определяемая внутри класса?

- Бинарная функция-операция, определяемая внутри класса, должна быть представлена с помощью нестатического метода с параметрами, при этом вызвавший ее объект считается первым операндом

1. Сколько операндов должна иметь бинарная функция-операция, определяемая вне класса?

- Если функция определяется вне класса*,* она должна иметь два параметра типа класса

1. Чем отличается перегрузка префиксных и постфиксных унарных операций?

- Операции постфиксного инкремента и декремента должны иметь первый параметр типа int. Он используется только для того, чтобы отличить их от префиксной формы.

1. Каким образом можно перегрузить операцию присваивания?

- нужно поэлементно скопировать поля правого операнда, предварительно сделав проверку на самоприсваивание

1. Что должна возвращать операция присваивания?

- операция должна возвращать ссылку на объект, для которого она вызвана, и принимать в качестве параметра единственный аргумент — ссылку на присваиваемый объект

1. Каким образом можно перегрузить операции ввода-вывода?

- Операции ввода-вывода operator>> и operator<< всегда реализуются как внешние дружественные функции, т. к. левым операндом этих операций являются потоки.

1. 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 как операцию, которая находится внутри класса

1. 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 как внешнюю дружественную функцию

1. 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 как функцию внутри класса

1. 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 операцию как вызов внешней дружественной функции