Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

по лабораторной работе №3

Тема: «Перегрузка операций.»

Выполнил

Студент группы РИС-22-2б

Арапов М.В.

Проверил доц. Кафедры ИТАС

Полякова О. А.

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 2:

Создать класс Time для работы с временными интервалами. Интервал должен быть представлен в виде двух полей: минуты типа int и секунды типа int. при выводе минуты отделяются от секунд двоеточием. Реализовать: вычитание временных интервалов (учесть, что в минуте не может быть более 60 секунд) сравнение временных интервалов (!=)

**Контрольные вопросы**

1. Для чего используются дружественные функции и классы?

Дружественные функции и классы используются для расширения доступа к закрытым членам класса, т.е. для того, чтобы некоторые функции или классы имели доступ к приватным или защищенным членам класса.

1. Сформулировать правила описания и особенности дружественных функций.

Основные правила описания дружественных функций:

- функция должна быть объявлена вне класса, но определена внутри него с помощью ключевого слова friend;

- дружественная функция имеет доступ ко всем членам класса (в том числе и приватным);

- дружественная функция не является членом класса и не может вызывать его методы;

- дружественные функции могут быть перегружены.

1. Каким образом можно перегрузить унарные операции?

Унарные операции можно перегрузить с помощью ключевого слова operator и символа операции (например, операция инкремента ++).

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

Унарная функция-операция, определяемая внутри класса, должна иметь один операнд - объект класса, для которого вызывается операция.

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

Унарная функция-операция, определяемая вне класса, должна иметь один операнд - объект класса, для которого вызывается операция (операнд передается как параметр функции).

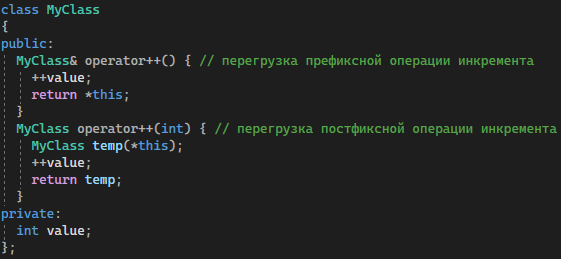
1. Сколько операндов должна иметь бинарная функция-операция, определяемая внутри класса?

Бинарная функция-операция, определяемая внутри класса, должна иметь два операнда - объекты класса, между которыми выполняется операция.

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

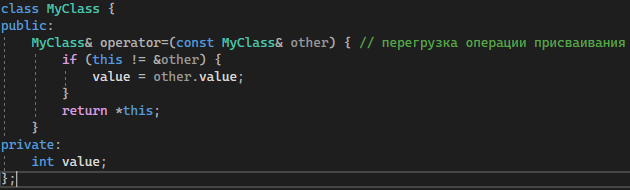
Бинарная функция-операция, определяемая вне класса, должна иметь два операнда - объекты класса, между которыми выполняется операция (операнды передаются как параметры функции).

1. Чем отличается перегрузка префиксных и постфиксных унарных операций?

Перегрузка префиксных и постфиксных унарных операций отличается тем, что префиксная операция выполняется перед изменением значения объекта, а постфиксная - после изменения значения объекта. Например: 

1. Каким образом можно перегрузить операцию присваивания?

Операцию присваивания можно перегрузить с помощью ключевого слова operator= и параметра-ссылки на объект того же типа (для передачи значения). Например:

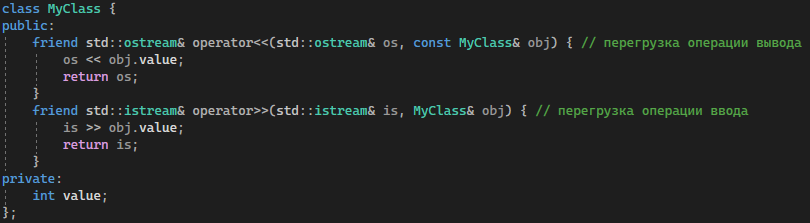


1. Что должна возвращать операция присваивания?

Операция присваивания должна возвращать ссылку на объект того же типа (MyClass&), чтобы можно было использовать операцию цепочкой (например, a = b = c).

1. Каким образом можно перегрузить операции ввода-вывода?

Операции ввода-вывода можно перегрузить с помощью ключевых слов operator<< и operator>> и объекта класса std::ostream (для вывода) или std::istream (для ввода). Например:



1. В программе описан класс

class Student

Student&operator++0;

и определен объект этого класса

Students;

Выполняется операция

++s;

Каким образом, компилятор будет воспринимать вызов функции-операции?  
Компилятор будет воспринимать вызов функции-операции как инкремент объекта Students.

1. В программе описан класс

class Student

френд Студент& оператор + +( Студент&);

и определен объект этого класса

Students;

Выполняется операция

++8;

Каким образом, компилятор будет воспринимать вызов функции-операции?

Компилятор не сможет воспринимать вызов функции-операции, так как в коде определен объект Students, а не Студент.

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 вызов функции-операции как сравнение объектов a и b с помощью оператора "<".

1. В программе описан класс

class Student

friend bool operator >(const Person&, Person&)

и определены объекты этого класса

Student a,b;

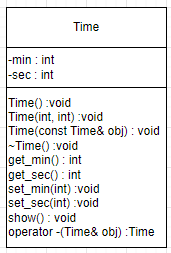
Выполняется операция

cout<<a>b;

Каким образом, компилятор будет воспринимать вызов функции-операции?

Компилятор будет воспринимать вызов функции-операции как сравнение объектов a и b с помощью оператора ">".

UML таблица



Код программы

Time.h:

#pragma once

#include <iostream>

using namespace std;

class Time

{

int min, sec;

public:

Time();

Time(int, int);

Time(const Time& obj);

~Time();

int get\_min();

int get\_sec();

void set\_min(int);

void set\_sec(int);

void show();

Time operator -(Time& obj);

};

Time.cpp:

#include "Time.h"

Time Time::operator -(Time& obj)

{

Time temp;

temp.min = this->min - obj.min;

temp.sec = this->sec - obj.sec;

if (temp.sec < 0)

{

temp.sec = 60 - abs(temp.sec);

--temp.min;

}

return temp;

}

Time::Time()

{

this->min = 0;

this->sec = 0;

}

Time::Time(int min, int sec)

{

this->min = min;

this->sec = sec;

}

Time::Time(const Time& obj)

{

this->min = 0;

this->sec = 0;

}

Time::~Time()

{

cout << "Destuctor" << endl << endl;

}

int Time::get\_min()

{

return this->min;

}

void Time::set\_min(int min)

{

this->min = min;

}

int Time::get\_sec()

{

return this->sec;

}

void Time::set\_sec(int sec)

{

this->sec = sec;

}

void Time::show()

{

cout << this->min << " : " << this->sec << endl;

}

Laba3:

#include "Time.h"

void main()

{

setlocale(LC\_ALL, "ru");

Time a(2, 22);

Time b(1, 30);

Time c=a-b;

c.show();

}