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

Максимов А. А.

Проверил доц. Кафедры ИТАС

Полякова Ольга Андреевна

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 15:

Создать класс Pair (пара чисел). Пара должна быть представлено двумя полями: типа int для первого числа и типа double для второго. Первое число при выводе на экран должно быть отделено от второго числа двоеточием. Реализовать: вычитание пар чисел добавление константы к паре (увеличивается первое число, если константа целая, второе, если константа вещественная).

**Контрольные вопросы**

*1. Для чего используются дружественные функции и классы?*

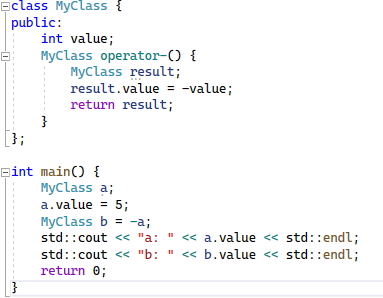
Дружественные функции и классы используются для предоставления доступа к закрытым (private) членам класса другим функциям и классам, что позволяет избежать использования глобальных переменных и упрощает реализацию сложных алгоритмов.

*2. Сформулировать правила описания и особенности дружественных функций.*

Правила и особенности описания дружественных функций включают объявление функции с ключевым словом `friend` внутри класса, с определением самой функции вне класса. Дружественная функция не является методом класса и не может обращаться к ключевому слову `this`. С помощью дружественных функций можно улучшить инкапсуляцию, увеличить удобство использования класса и повысить его гибкость.

*3. Каким образом можно перегрузить унарные операции?*

Унарные операции можно перегрузить с помощью перегрузки операторов в классе.



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

Унарная функция-операция не должна иметь операндов.

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

Унарная функция-операция должна иметь один операнд.

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

Бинарная функция-операция должна иметь два операнда.

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

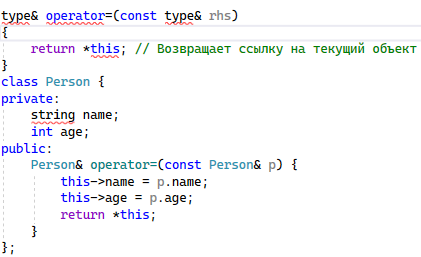
Бинарная функция-операция должна иметь два операнда.

*8. Чем отличается перегрузка префиксных и постфиксных унарных операций?*

Перегрузка префиксных унарных операций выполняется непосредственно перед операцией, а постфиксных - непосредственно после операции.

*9. Каким образом можно перегрузить операцию присваивания?*

Операция присваивания может быть перегружена в классе через перегрузку оператора "=" для определения собственной логики копирования одного объекта в другой.

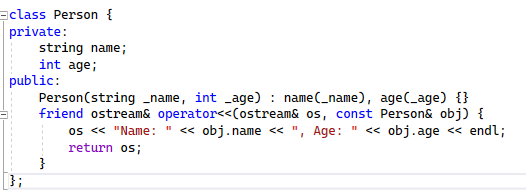


*10. Что должна возвращать операция присваивания?*

Операция присваивания должна возвращать значение, присвоенное переменной.

*11. Каким образом можно перегрузить операции ввода-вывода?*

Операции ввода-вывода можно перегрузить путем определения функций операторов << и >> для пользовательских типов данных.



*12. 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 вызов функции-операции как инкремент объекта s. Это будет эквивалентно вызову функции-операции вида s.operator++(), где s - объект класса Student.

*13. 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 вызов функции-операции как унарный оператор инкремента (++). Так как в определении класса указан friend для функции-операции, она может получить доступ к закрытым членам класса и изменить их значение после инкремента. В результате вызова ++s, значение s будет увеличено на 1.

*14. 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 класса Student по определенным критериям, заданным в функции-операции `<`. Результатом операции будет значение типа bool, которое будет выведено с помощью оператора `<<` в поток вывода данных.

*15. В программе описан класс*

*class Student*

*{*

*…*

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

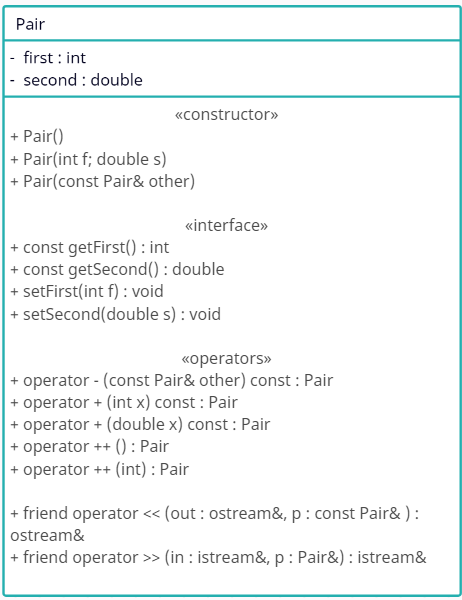
*….*

*};*

*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 бинарную операцию с двумя аргументами: объектом "a" класса Student и объектом "b" класса Student. Поскольку эта функция-операция определена как friend в классе Student, то она имеет доступ к приватным членам класса и может сравнивать объекты по необходимым свойствам. Результатом выполнения операции будет логическое значение (true или false), которое потом будет передано в объект cout для печати на экране.

UML таблица



Код программы

Pair.h:

#pragma once

#include <iostream>

class Pair {

private:

int first;

double second;

public:

Pair();

Pair(int f, double s);

Pair(const Pair& other);

int getFirst() const;

double getSecond() const;

void setFirst(int f);

void setSecond(double s);

Pair operator- (const Pair& other) const;

Pair operator+ (int x) const;

Pair operator+ (double x) const;

Pair operator++ ();

Pair operator++ (int);

friend std::ostream& operator<< (std::ostream& out, const Pair& p);

friend std::istream& operator>> (std::istream& is, Pair& p);

};

Pair.cpp:

#include "Pair.h"

Pair::Pair() {

first = 0;

second = 0.0;

}

Pair::Pair(int f, double s) {

first = f;

second = s;

}

Pair::Pair(const Pair& other) {

first = other.first;

second = other.second;

}

int Pair::getFirst() const {

return first;

}

double Pair::getSecond() const {

return second;

}

void Pair::setFirst(int f) {

first = f;

}

void Pair::setSecond(double s) {

second = s;

}

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

int f = first - other.first;

double s = second - other.second;

return Pair(f, s);

}

Pair Pair::operator+ (int x) const {

int f = first + x;

double s = second;

return Pair(f, s);

}

Pair Pair::operator+ (double x) const {

int f = first;

double s = second + x;

return Pair(f, s);

}

Pair Pair::operator++ () {

++first;

return \*this;

}

Pair Pair::operator++ (int) {

Pair temp(\*this);

++first;

return temp;

}

std::ostream& operator<< (std::ostream& out, const Pair& p) {

out << p.first << ":" << p.second;

return out;

}

std::istream& operator>>(std::istream& is, Pair& p) {

double f, s;

is >> f >> s;

p.setFirst(f);

p.setSecond(s);

return is;

};

main.cpp:

#include <iostream>

#include "Pair.h"

using namespace std;

int main() {

Pair p1, p2;

cout << "Введите первую пару: ";

cin >> p1;

cout << "Введите вторую пару: ";

cin >> p2;

Pair p3 = p1 - p2;

cout << "p1 - p2 = " << p3 << endl;

Pair p4 = p1 + 2;

cout << "p1 + 2 = " << p4 << endl;

Pair p5 = p2 + 1.5;

cout << "p2 + 1.5 = " << p5 << endl;

cout << "++p1 = " << ++p1 << endl;

cout << "p1++ = " << p1++ << endl;

cout << "p1 = " << p1 << endl;

return 0;

}