Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

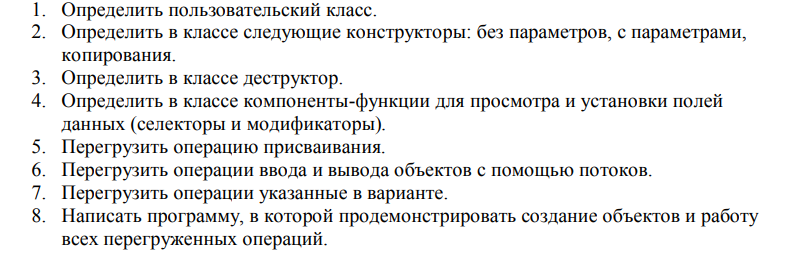
**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: Лабораторная работа по классам №3

Выполнил работу  
Студент группы РИС-22-1Б  
Ишемцева Мария Андреевна  
Проверил  
Доцент кафедры ИТАС  
Полякова Ольга Андреевна

Постановка задачи:

 Анализ задачи:

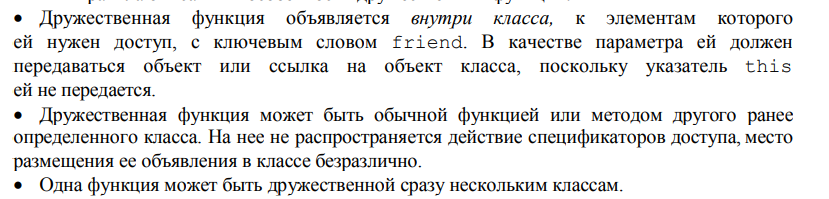
1. Класс Pair содержит поля int first и double second в private секции. Доступ будем получать при помощи конструкторов и сеттеров.

2. Три конструктора – с параметрами, без них и конструктор копирования, а также установление полей с помощью сеттеров и получение значений при помощи геттеров. Реализована ф-ия show() для вывода полей нужного обьекта(в данном случае, для сеттеров, т.к. каждый конструктор имеет свой вывод). Перегрузка оператора сложения с помощью ключевого слова «operator» и возвращаемым значением является объект класса. Перегрузка оператора дикремента происходит аналогично перегрузке оператора сложения.

3. В ф-ии main() происходит создание объектов, вызов конструкторов, вывод на экран булевых значений, операции префиксного(приоритетного) дикремента и постфиксного.

4. Ответы на вопросы:

1. если сумма у одного объекта совпадет с суммой другого, то будет получено true, иначе – false

2. 

3. Как метод, как внешнюю ф-ию.

4. 0.

5. 1.

6. 1. Ее операнд – вызвавший ее же объект.

7. 2 операнда типа класса.

8. префиксная приоритетнее, в нее должно(но не должно) передаваться значение.

9. Class&operator(const Class&other).

10. Объект.

11. При помощи дружественной глобальной ф-ии.

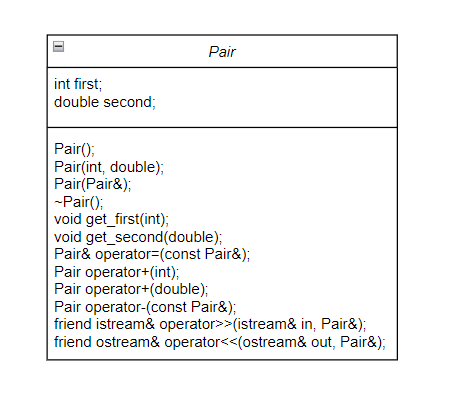
12. Префиксный инкремент.

13. Перегруженная дружественная ф-ия.

14. Если а действительно меньше b вернется 1.

15. Если а действительно больше b вернется 1.

UML диаграмма:



Код:

#include <iostream>

using namespace std;

class Pair {

int first;

double second;

public:

Pair();

Pair(int, double);

Pair(Pair&);

~Pair();

void get\_first(int);

void get\_second(double);

//перегрузка операции присваивания

Pair& operator=(const Pair&);

Pair operator+(int);

Pair operator+(double);

Pair operator-(const Pair&);

friend istream& operator>>(istream& in, Pair&);

friend ostream& operator<<(ostream& out, Pair&);

};

#include <iostream>

#include "Pair.h"

using namespace std;

Pair::Pair() {

second = 0.0;

first = 0;

cout << "Конструктор без параметров для " << this << endl;

}

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

first = f;

second = s;

cout << "Коснтруктор с параметрами для - " << this << endl;

}

Pair::Pair(Pair& T) {

first = T.first;

second = T.second;

cout << "Конструктор копирования для - " << this << endl;

}

Pair::~Pair() {

cout << "Дуструктор для - " << this << endl;

}

void Pair::get\_first(int a) {

first = a;

}

void Pair::get\_second(double b) {

second = b;

}

//перегрузка операции присваивания

Pair& Pair::operator=(const Pair& t)

{

//проверка на самоприсваивание

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

first = t.first;

second = t.second;

return \*this;

}

Pair Pair::operator+(int a)

{

first += a;

return \*this;

}

Pair Pair::operator+(double a)

{

second += a;

return \*this;

}

Pair Pair::operator-(const Pair& P)

{

int temp1 = first - P.first;

double temp2 = second - P.second;

Pair p;

p.first = temp1;

p.second = temp2;

return p;

}

istream& operator>>(istream& in, Pair& p) {

cout << "first? "; in >> p.first;

cout << "second? "; in >> p.second;

return in;

}

ostream& operator<<(ostream& out, Pair& p)

{

return (out << p.first << " : " << p.second << endl);

}

#include <iostream>

#include "Pair.h"

using namespace std;

int main()

{

setlocale(0, "RUS");

Pair a;

Pair b;

cout << "Ввод пар чисел" << endl;

cin >> a;

cin >> b;

cout << "Вывод пар чисел" << endl;

cout << a;

cout << b;

Pair c;

cout << "Вычитание одной пары из другой" << endl;

c = a - b;

cout << c;

cout << "Первая пара" << endl;

cout << a;

a = a + 10;

cout << "Первая пара после прибавления 10" << endl;

cout << a;

cout << "Первая пара после прибавления 1.5" << endl;

a = a + 1.5;

cout << a;

return 0;

}

Вывод программы:

