Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

ПНИПУ

**Лабораторная работа  
“Классы”. №3**

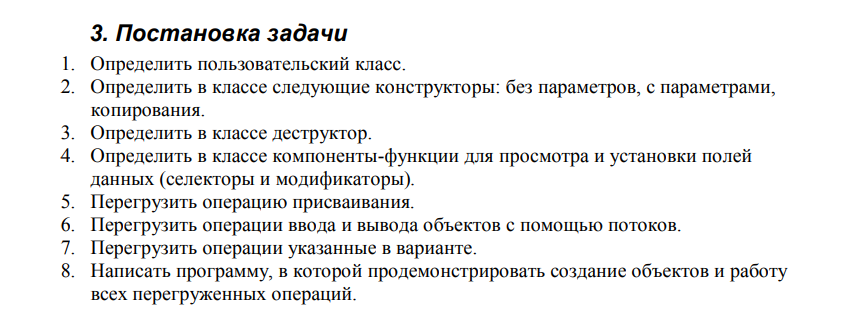
Выполнил:   
студент группы ИВТ 23 2Б   
Чудинов Данил Николаевич

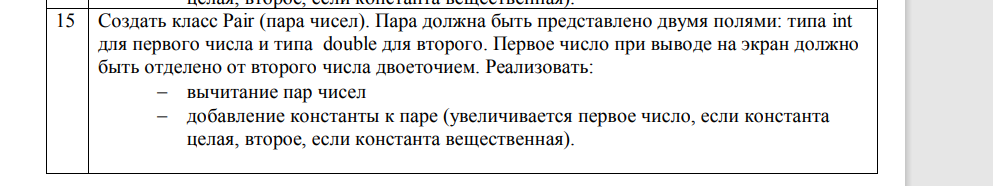
Проверила:   
доцент кафедры ИТАС   
О.А. Полякова

Пермь, 2024 г.

**«Классы и объекты. Перегрузка»**

**Анализ задачи:**

***Вариант 15:***

****

**Код на языке C++:**

#include <iostream>

using namespace std;

class Pair {

private:

int x;

double y;

public:

Pair() {

this->x = 0;

this->y = 0;

cout << "Вызван конструктор без параметров: " << this << endl;

cout << "X равно: " << this->x << '\t' << "Y равно: " << this->y << endl << endl;

}

Pair(int i, double j) {

this->x = i;

this->y = j;

cout << "Вызван конструктор с параметрами: " << this << endl;

cout << "X равно: " << this->x << '\t' << "Y равно: " << this->y << endl << endl;

}

Pair(const Pair& d) {

this->x = d.x;

this->y = d.y;

cout << "Вызван конструктор копирования: " << this << endl;

cout << "X равно: " << this->x << '\t' << "Y равно: " << this->y << endl << endl;

}

double diff() {

return this->x - this->y;

}

void dobavlenieConst(int j) {

cout << "К " << this->x << " добавляем const " << j << ": " << endl;

this->x = x + j;

}

void dobavlenieConst(double i) {

cout << "К " << this->x << " добавляем const " << i << ": " << endl;

this->y = y + i;

}

void GetInfo() {

cout << this->x << " : " << this->y << endl;

}

void operator=(const Pair& a) {

this->x = a.x;

this->y = a.y;

}

void setX(int b) {

this->x = b;

}

void setY(double a) {

this->y = a;

}

int returnX() {

return this->x;

}

double returnY() {

return this->y;

}

void operator -(const Pair& a) {

this->x = x - a.x;

this->y = y - a.y;

}

void operator +(int j) {

cout << "К " << this->x << " добавляем const " << j << ": " << endl;

this->x = x + j;

}

void operator +(double k) {

cout << "К " << this->y << " добавляем const " << k << ": " << endl;

this->y = y + k;

}

~Pair() {

cout << "Вызван деструктор: " << this << endl;

}

};

ostream& operator << (ostream& os, Pair& b) {

cout << "Вызвана перегрузка оператора « " << endl;

os << b.returnX() << " : " << b.returnY();

return os;

}

istream& operator >> (istream& is, Pair& a) {

cout << "Вызвана перегрузка оператора » " << endl;

int first;

double second;

is >> first >> second;

a.setX(first);

a.setY(second);

return is;

}

int main() {

system("chcp 1251>NULL");

Pair h1;

Pair h2(5, 6.2);

cout << "Разность чисел: " << h2.diff() << endl << endl;

cout << "Добавление константы целого типа: " << endl;

h2.dobavlenieConst(5);

h2.GetInfo();

cout << endl;

cout << "Добавление константы дробного типа: " << endl;

h2.dobavlenieConst(7.2);

h2.GetInfo();

cout << endl;

cout << "Перегрузка оператора присваивания: " << endl;

Pair h4;

h4 = h2;

cout << "Присваиваем в него значения из первого конструктора во второй: " << endl;

h4.GetInfo();

cout << endl << endl;

cin >> h4;

cout << endl << endl;

cout << h4;

cout << endl << endl;

cout << "Перегрузка оператора вычитания: " << endl;

h4 - h2;

cout << h4;

cout << endl << endl;

Pair h3(h2);

cout << "Перегрузка оператора сложения: " << endl;

h3 + 5;

h3 + 7.5;

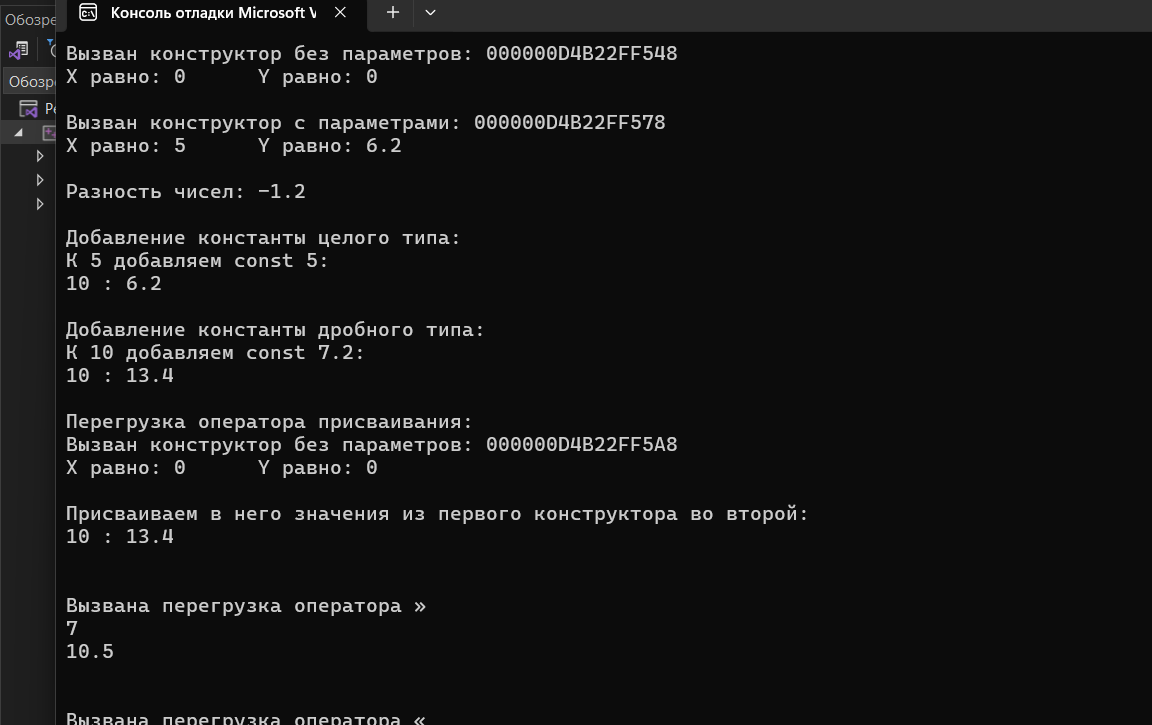
cout << h3;

cout << endl << endl;

return 0;

}

**Работа программы:**



**Ответ на вопросы:**

1. Дружественные функции и классы используются для предоставления специальных разрешений к членам класса другого класса или функции. Это означает, что эти функции и классы имеют доступ к закрытым и защищенным членам другого класса.

2. Правила описания дружественных функций:

- Дружественные функции объявляются внутри класса в области public, protected или private.

- Функции-друзья не являются методами класса, хотя к их закрытым и защищенным членам можно обращаться как членам самого класса.

- Дружественные функции не наследуются.

- Функции-друзья не являются методами класса, поэтому они не наследуются дочерними классами при наследовании.

3. Унарные операции могут быть перегружены с помощью функций-членов класса или с помощью функций-друзей. Для перегрузки унарных операций важно формально указать количество операндов в соответствии с требованиями языка программирования.

4. Унарная операция, определяемая внутри класса, должна иметь один операнд(класс над которым выполняется операция).

5. Унарная операция, определяемая вне класса, также должна иметь один операнд.

6. Бинарная функция-операция, определяемая внутри класса, должна принимать два операнда, обычно первым операндом является объект, к которому применяется операция, а вторым - операнд, с которым операция выполняется.

7. Для бинарной функции-операции, определяемой вне класса, также требуется два операнда.

8. Перегрузка префиксных и постфиксных унарных операций отличается тем, что:

- Префиксная версия (++obj) увеличивает значение объекта и возвращает измененное значение.

- Постфиксная версия (obj++) также увеличивает значение объекта, но возвращает его старое значение (незамененное).

9. Операцию присваивания можно перегрузить внутри класса с помощью перегрузки `operator=`. Возвращаемым значением является ссылка на объект, что позволяет использовать цепочки присваивания.

10. Операции ввода-вывода можно перегрузить с использованием перегрузки операторов `«` и `»` для объектов класса. Для этого обычно используется дружественные функции.

11. Когда в программе имеется класс `Student` и определен объект `s` этого класса, а затем выполняется операция `++s`, компилятор воспринимает вызов функции-операции `operator++()` как префиксный инкремент унарной операции (перегрузка оператора унарного плюса).

12. В случае, если есть класс `Student`, объявлена дружественная функция `operator++(Student&)`, а затем выполняется операция `++s`, компилятор будет воспринимать вызов функции-операции `operator++()` как префиксный инкремент унарной операции (перегрузка оператора унарного плюса).

14. Если в программе есть класс `Student` с перегруженным оператором `<`, а затем определены объекты `a` и `b` этого класса, и выполняется операция `cout « a < b;`, компилятор будет воспринимать вызов функции-операции как попытку вывести результат сравнения объектов `a` и `b` с использованием перегруженного оператора `<`.

15. В случае, если в программе есть класс `Student` с дружественной функцией `operator >` и определены объекты `a` и `b` этого класса, и выполняется операция `cout « a > b;`, компилятор будет воспринимать вызов функции-операции как попытку вывести результат сравнения объектов `a` и `b` с использованием дружественной функции-оператора `>` для класса `Student`.

**Вывод:** Задача была выполнена. Всё получилось.