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

Анистратов А.Д.

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.

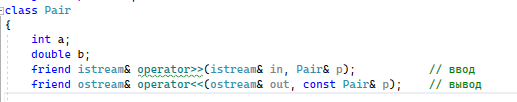
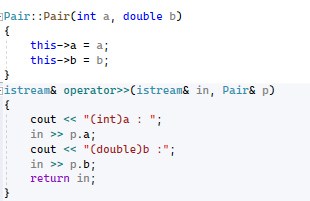
ВАРИАНТ 15:

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**

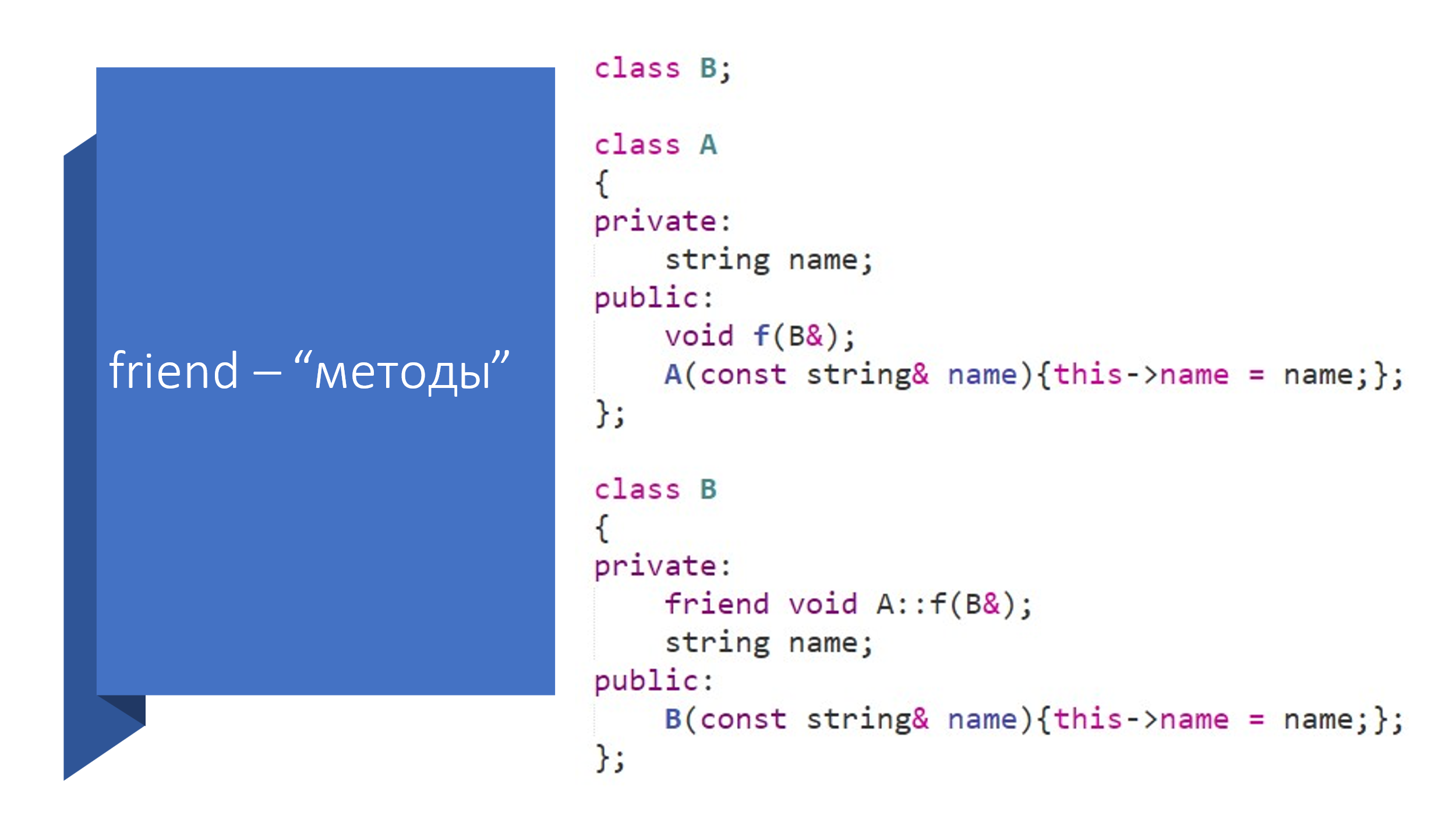
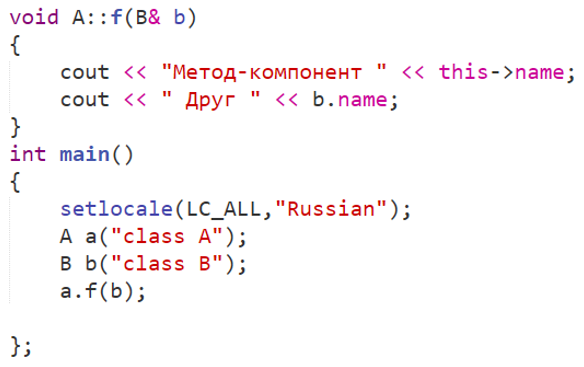
1. Для чего используются дружественные функции и классы?

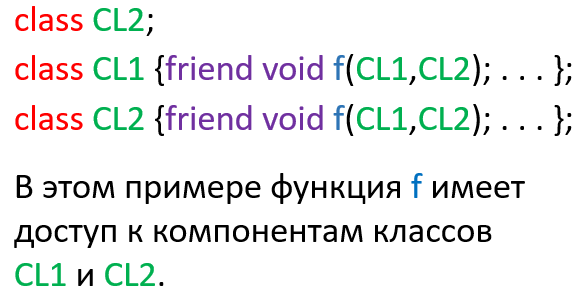
Если необходимо иметь доступ извне к приватным свойствам класса, то есть расширить его интерфейс.

1. Сформулировать правила описания и особенности дружественных функций.

Правила описания и особенности дружественных функций:

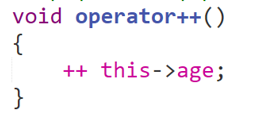
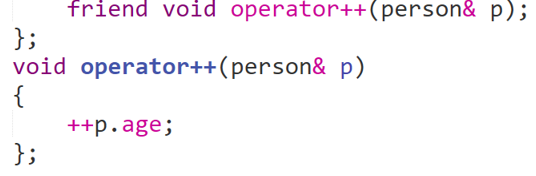
* Дружественная функция объявляется *внутри класса,* к элементам которого ей нужен доступ, с ключевым словом friend. В качестве параметра ей должен передаваться объект или ссылка на объект класса, поскольку указатель this ей не передается.
* Дружественная функция может быть обычной функцией или методом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

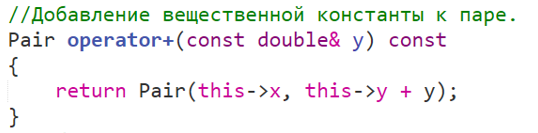
0 – если префиксная форма, 1 – int( по сути не передается, просто заглушка), если постфиксная

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

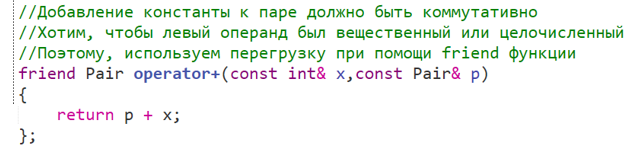
1 операнд – указатель на объект класса, т.к. this не передается. .

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

1 операнд.



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

2 операнда.

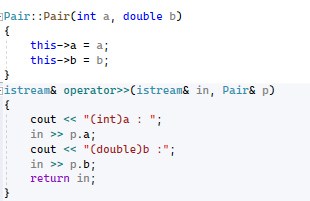
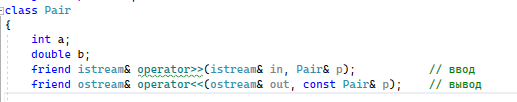
1. Чем отличается перегрузка префиксных и постфиксных унарных операций?

* Префиксные и постфиксные операции по сути являются версией одного оператора в разных формах. Если при перегрузке префиксного оператора не нужно передавать никаких параметров, то при перегрузке постфиксного оператора необходимо передать незначащий параметр *int.* – Чтобы объяснить компилятору разницу.
* Также эти операторы могут отличатся по типу возвращаемого значения. Допустим, если префиксный оператор (инкремента или декремента) модифицирует какое-либо информационное поле, а затем возвращает ссылку на объект этого класса, то постфиксный оператор должен сохранить состояние объекта класса во временную переменную, затем модифицировать поле класса, затем вернуть копию предыдущего состояния. – Это накладывает некоторые ограничения на использование постфиксных операторов, т.к. они не позволяют взаимодействовать напрямую с объектом класса.

1. Каким образом можно перегрузить операцию присваивания?

Как нестатическую компонентную функцию класса

1. Что должна возвращать операция присваивания?
   * Ссылку на объект класса, в который происходит копирование (левый операнд). Это нужно для реализации многочисленного присваивания.
   * Например: a = b = c; Где a, b и c – Объекты одного класса.
2. Каким образом можно перегрузить операции ввода-вывода?



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?

Как вызов метода класса: s.operator++();

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?

Как вызов глобальной функции operator++(s);

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?

* Приоритет оператора << выше, чем у оператора<. Программа просто не скомпилируется.
* Если выполнялась бы операция cout<<(a<b); То компилятор это воспринял бы как вызов метода класса a.operator<(b);

15. В программе описан класс class Student

{

…

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

….

};

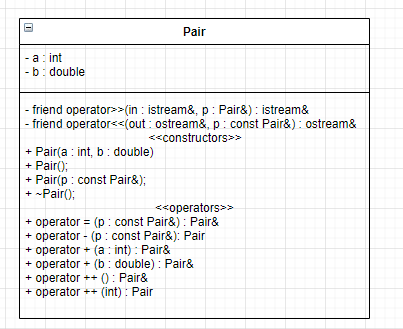
и определены объекты этого класса Student a,b;

Выполняется операция cout<<a>b;

Каким образом, компилятор будет воспринимать вызов функции-операции?

* Приоритет у оператора << выше, программа не будет работать.
* Если выполняется операция cout << (a>b); то компилятор это будет воспринимать как вызов глобальной функции: cout << operator>(a, b);

UML таблица



Код программы

LAB\_OOP\_3.cpp

#pragma once

#include <iostream>

using namespace std;

class Pair

{

int a;

double b;

friend istream& operator>>(istream& in, Pair& p); // ввод

friend ostream& operator<<(ostream& out, const Pair& p); // вывод

public:

Pair(int a, double b); // конструктор с параметрами

Pair(); // конструктор без параметров

Pair(const Pair& p); // конструктор копирования

~Pair(); // деструктор

Pair& operator=(const Pair& p); // присваивание

Pair operator-(const Pair& p); // вычитание двух объектов

Pair& operator+(int a); // прибавление int числа

Pair& operator+(double b); // прибавление double числа

Pair& operator++(); // префиксный инкремент

Pair operator++(int); // постфиксный инкремент

};

Pair.cpp:

#include "Pair.h"

Pair::Pair(int a, double b)

{

this->a = a;

this->b = b;

}

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

{

cout << "(int)a : ";

in >> p.a;

cout << "(double)b :";

in >> p.b;

return in;

}

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

{

return (out << p.a << " : " << p.b);

}

Pair::Pair()

{

a = 0;

b = 0;

}

Pair::~Pair()

{

}

Pair::Pair(const Pair& p)

{

a = p.a;

b = p.b;

}

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

{

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

a = p.a;

b = p.b;

return \*this;

}

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

{

Pair res(a - p.a, b - p.b);

return res;

}

Pair& Pair::operator+(int a)

{

this->a += a;

return \*this;

}

Pair& Pair::operator+(double b)

{

this->b += b;

return \*this;

}

Pair& Pair::operator++()

{

++a; ++b;

return \*this;

}

Pair Pair::operator ++(int)

{

Pair temp = \*this;

this->a++; this->b++;

return temp;

}

Скриншоты работы программы

