Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования

Пермский национальный исследовательский политехнический университет

Факультет электротехнический

Кафедра ИТАС

ОТЧЁТ

о лабораторной работе №3 (классы)

Выполнил:
Студент группы ИВТ-23-1Б
Пискунов Д. А.

_____

Проверил:
Доцент кафедры ИТАС
Яруллин Д.В.

_____

_____

Пермь 2024

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).

Текст программы

Файл Pair.h

```
#pragma once
#include <iostream>;

using namespace std;

class Pair {
    int first;
    double second;
public:
    Pair() {
        first = 0;
        second = 0;
    }
    Pair(int f, double s) {
        first = f;
        second = s;
    }
    Pair(const Pair& p) {
        first = p.first;
        second = p.second;
    }
    ~Pair() {}
    int get_first() {
        return first;
    }
    double get_second() {
        return second;
    }
    void set_first(int f) {
        first = f;
    }
    void set_second(int s) {
        second = s;
    }

    Pair& operator=(const Pair&);
    Pair operator-(const Pair& p);
    Pair& operator+(int);
    Pair& operator+(double);
    Pair& operator++();
    Pair operator++(int);

    friend istream& operator >> (istream& in, Pair& p);
    friend ostream& operator << (ostream& out, const Pair& p);
};
```

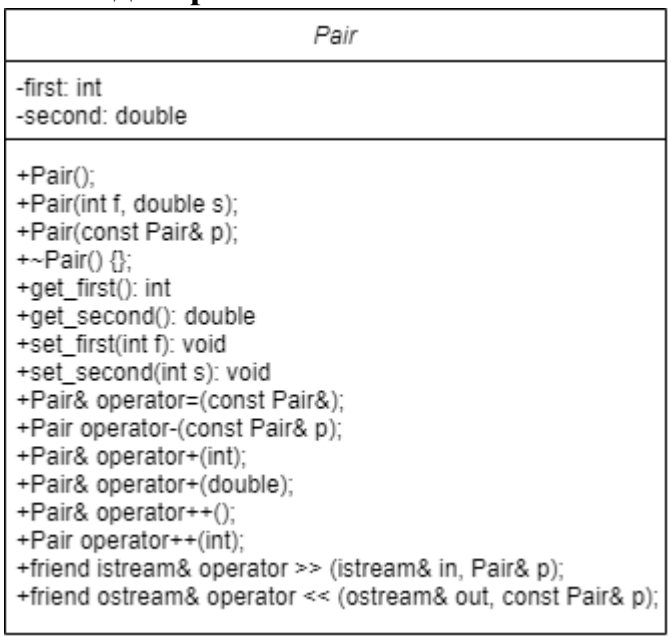
Файл pair2.cpp

```cpp
#include "Pair.h"

Pair& Pair::operator=(const Pair& p) {
    if (&p != this) {
        first = p.first;
        second = p.second;
    }
    return *this;
}
Pair Pair::operator-(const Pair& p) {
    Pair result(first - p.first, second - p.second);
    return result;
}
Pair& Pair::operator+(int f) {
    this->first += f;
    return *this;
}
Pair& Pair::operator+(double s) {
    this->second += s;
    return *this;
}
Pair& Pair::operator++() {
    first++;
    second++;
    return *this;
}
Pair Pair::operator++(int) {
    Pair tmp = *this;
    this->first++;
    this->second++;
    return tmp;
}

istream& operator >> (istream& in, Pair& p) {
    in >> p.first;
    in >> p.second;
    return in;
}
ostream& operator << (ostream& out, const Pair& p) {
    return out << p.first << "|| " << p.second;
}
```

Файл Classes3.cpp

```cpp
#include <iostream>
#include "Pair.h"

 using namespace std;

Pair& operator+(int f, Pair& p) {
    return p + f;
}
Pair& operator+(double s, Pair& p) {
    return p + s;
}

int main(){
    Pair a;
    Pair b(12,2.5);
    Pair c(b);

    b + 24;
    b + 0.24;

    a = (b - c);

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "c = " << c << endl;

    return 0;
}

```

UML- диаграмма:



Тест

```
a = 24|| 0.24
b = 36|| 2.74
c = 12|| 2.5

C:\Users\MOkASiH\Desktop\Classes3\x64\Debug\Classes3.exe (процесс 2300) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно:
```

Ответы на вопросы:

1. Для чего используются дружественные функции и классы?

Ответ: для расширения возможности работы с классом.

2. Сформулировать правила описания и особенности дружественных функций.

Ответ:

- Дружественная функция объявляется внутри класса, к элементам которого ей нужен доступ
- Объявляется с ключевым словом friend.
- В качестве параметра ей должен передаваться объект или ссылка на объект класса, поскольку указатель this ей не передается.
- Одна функция может быть дружественной сразу нескольким классам.

3. Каким образом можно перегрузить унарные операции?

Ответ:

Унарную операцию можно перегрузить:

- Как компонентную функцию класса
- Как внешнюю (глобальную) функцию

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

Ответ: 0 – если префиксная форма, 1 – int, если постфиксная.

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

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

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

Ответ: 1 операнд.

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

Ответ: 2 операнда.

8. Чем отличается перегрузка префиксных и постфиксных унарных операций?

Ответ:

- При перегрузке постфиксного оператора необходимо передать параметр int. – Чтобы компилятор правильно определил форму.
- Также эти операторы могут отличатся по типу возвращаемого значения.

9. Каким образом можно перегрузить операцию присваивания?
   Ответ: как нестатическую компонентную функцию класса

10. Что должна возвращать операция присваивания?
Ответ: Ссылку на объект класса, в который происходит копирование для реализации многочисленного присваивания: a = b = c;

11. Каким образом можно перегрузить операции ввода-вывода?

```
istream& operator >> (istream& in, Pair& p) {
        in >> p.first;
        in >> p.second;
        return in;
}
ostream& operator << (ostream& out, const Pair& p) {
        return out << p.first << "|| " << p.second;
}
```

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?

Ответ : 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?

Ответ: 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?

Ответ: программа не скомпилируется так, как приоритет оператора << выше, чем у оператора <.

если взять выражение в скобки, т. е. cout<<(a<b); То компилятор воспримет это 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?

Не скомпилируется.

Если взять выражение в скобки, т. е. cout << (a>b); то компилятор воспримет это как вызов глобальной функции: cout << operator>(a, b);