Федеральное государственное автономное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет», ПНИПУ

ЛАБОРАТОРНАЯ РАБОТА

ПЕРЕГРУЗКА ОПЕРАЦИЙ

Выполнил: студент группы РИС-23-3б

Артем Владимирович Швецов

Проверила: доцент кафедры ИТАС

Ольга Андреевна Полякова

Пермь 2024

**Постановка задачи**

1. Определить пользовательский класс.
2. Определить в классе конструкторы: без параметров, с параметрами, копирования.
3. Определить в классе деструктор.
4. Определить в классе селекторы и модификаторы.
5. Перегрузить операцию присваивания.
6. Перегрузить операцию ввода и вывода объектов с помощью потоков.
7. Перегрузить операции, указанные в варианте.

Для варианта 15:

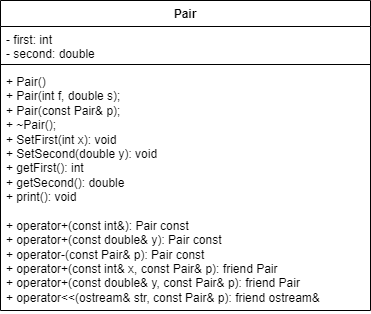
Создать класс Pair (пара чисел). Пара должна быть представлена двумя полями: первое – int, второе – double. При выводе на экран первое число должно быть отделено от второго двоеточием. Реализовать:

* Вычитание пар чисел
* Добавление константы к паре (если константа типа int, то прибавление к первому полю, если double – ко второму)

**Анализ задачи**

1. Создается новый файл с расширением .h, где определяется новый класс Pay.
2. Создается парный .cpp файл, в котором определяются методы нового класса.
3. При перегрузке операций ввода-вывода с помощью потоков функции определяются как дружественные. Остальные перегрузки компонентные.

**UML диаграмма**



**Код**

Pair.h

#pragma once

#include <iostream>

using namespace std;

class Pair

{

private:

int first;

double second;

public:

Pair();

Pair(int f, double s);

Pair(const Pair& p);

~Pair();

void SetFirst(int x);

void SetSecond(double y);

int getFirst();

double getSecond();

void print();

//перегруженные операции

Pair operator+(const int& x) const;

Pair operator+(const double& y) const;

Pair operator-(const Pair& p) const;

Pair operator=(const Pair& p);

friend Pair operator+(const int& x, const Pair& p);

friend Pair operator+(const double& y, const Pair& p);

friend ostream& operator<<(ostream& str, const Pair& p);

};

Pair.cpp

#pragma once

#include "Pair.h"

//перегрузка операции присваивания

Pair::Pair() {

first = 0;

second = 0;

}

Pair::Pair(int f = 0, double s = 0) {

this->first = f;

this->second = s;

}

Pair::Pair(const Pair& p) {

this->first = p.first;

this->second = p.second;

}

Pair::~Pair() {

}

int Pair::getFirst() { return this->first; }

void Pair::SetFirst(int x) { this->first = x; }

double Pair::getSecond() { return this->second; }

void Pair::SetSecond(double y) { this->second = y; }

void Pair::print() {

cout << this->first << " : " << this->second << endl;

}

//перегруженные операции

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

return Pair(this->first + x, this->second);

}

Pair Pair::operator+(const double& y) const {

return Pair(this->first, this->second + y);

}

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

return Pair(this->first - p.first, this->second - p.second);

}

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

this->first = p.first;

this->second = p.second;

return \*this;

}

//Дружественные перегруженные операции

Pair operator+(const int& x, const Pair& p) {

return p + x;

}

Pair operator+(const double& y, const Pair& p) {

return p + y;

}

ostream& operator<<(ostream& str, const Pair& p) {

str << " " << p.first << " : " << p.second << endl;

return str;

}

Main.cpp

#include <iostream>

#include "Pair.h"

using namespace std;

int main()

{

system("chcp 1251 > null" );

system("cls");

Pair a(5, 8);

Pair b(9, 4);

cout << "a: " << a << "b: " << b;

cout << "Разность двух чисел" << a - b << endl;

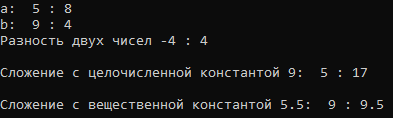
cout << "Сложение с целочисленной константой " << "9: " << a + 9.0 << endl;

cout << "Сложение с вещественной константой " << "5.5: " << b + 5.5 << endl;

return 0;

}

**Решение**



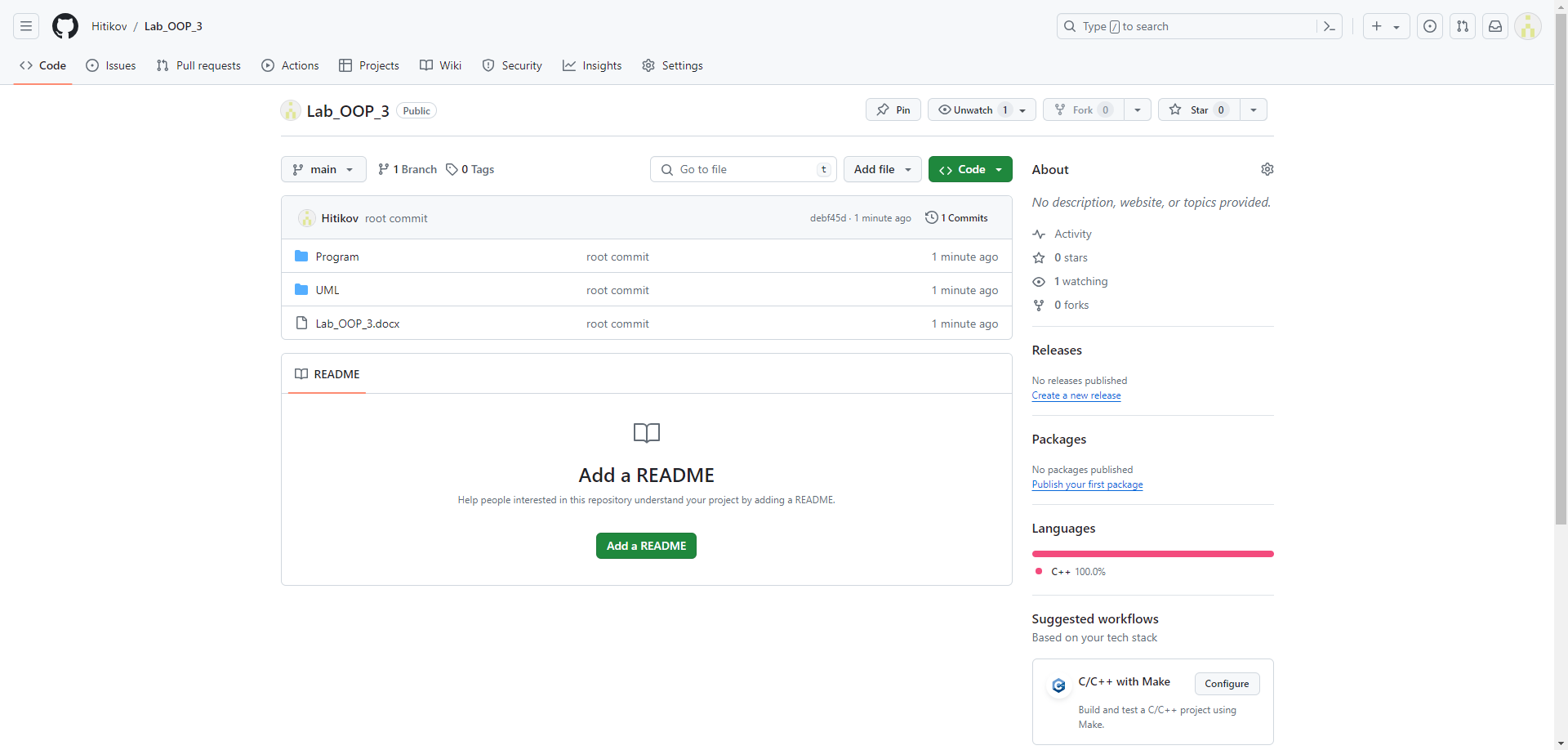
**Выводы**

В ходе выполнения работы были изучены механизмы перегрузки операций.

**Ответы на контрольные вопросы:**

1. Используются для доступа к защищенным полям, если определяется не как метод.
2. Описываются вне дружественных классов, могут быть частью другого класса или отдельной функцией, в дружественном классе необходимо указать с модификатором friend.
3. Используя методы или дружественные функции, используя ключевое слово operator.
4. 0
5. 1
6. 1
7. 2
8. Необходимостью использования параметра при определении.
9. Через использование метода возвращающего значение копирующего класса.
10. Значение текущего объекта.
11. Через использование дружественных функций, возвращающих поток.
12. Как метод.
13. Как дружественную функцию.
14. Как метод объекта a.
15. Как дружественную функцию.

**Github**



<https://github.com/Hitikov/Lab_OOP_3>