МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра информационные технологии и автоматизированные системы

**Дисциплина Информатика**

**Перегрузка операторов**

Выполнил студент ИВТ-22-2б:

Мифтахов Марат Ринатович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Пермь, 2023

**Постановка задачи**

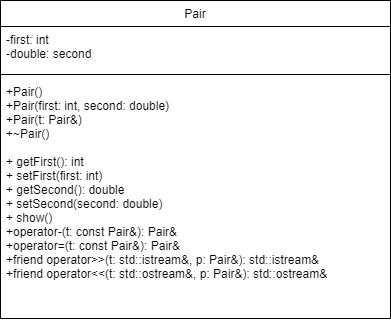
1. Определить пользовательский класс.
2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.
3. Определить в классе деструктор.
4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).
5. Перегрузить операцию присваивания.
6. Перегрузить операции ввода и вывода объектов с помощью потоков.
7. Перегрузить операции, указанные в варианте.
8. Написать программу. Продемонстрировать создание объектов и работу всех перегруженных операций.

**Задание**

1. Класс Pair
2. Поля int и double
3. Вывод через чисел через двоеточие
4. Реализовать вычитание пар чисел
5. Добавление константы к паре

**Диаграмма класса**

Ниже представлена UML-диаграмма класса Pair.



**Программный код**

Файл pair.cpp:

#include <iostream>

#include "Pair.h"

Pair::Pair() {

first = 0;

second = 0.0;

}

Pair::Pair(int a, double b) {

first = a;

second = b;

}

Pair::Pair(Pair&t) {

first = t.first;

second = t.second;

}

int Pair::getFirst() {

return first;

}

void Pair::setFirst(int tf) {

first = tf;

}

double Pair::getSecond() {

return second;

}

void Pair::setSecond(double tf) {

second = tf;

}

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

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

first = ptr.first;

second = ptr.second;

return \*this;

}

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

std::cout << "First? " << std::endl; in >> p.first;

std::cout << "Second?" << std::endl; in >> p.second;

return in;

}

std::ostream& operator<<(std::ostream& out, Pair& p) {

return out << p.first << ":" << p.second;

}

void Pair::show() {

std::cout << first << ":" << second << std::endl;

}

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

first -= a.first;

second -= a.second;

return \*this;

}

Файл pair.h:

#include <iostream>

class Pair {

private:

int first;

double second;

public:

Pair();

Pair(int, double);

Pair(Pair&);

~Pair() {};

int getFirst();

void setFirst(int);

double getSecond();

void setSecond(double);

void show();

template <typename T>

Pair& operator+(const T);

Pair& operator-(const Pair&);

Pair& operator=(const Pair&);

friend std::istream& operator>>(std::istream&, Pair&);

friend std::ostream& operator<<(std::ostream&, Pair&);

};

template <typename T>

Pair& Pair::operator+(const T p) {

if (p != (int)p) {

second += p;

}

else {

first += p;

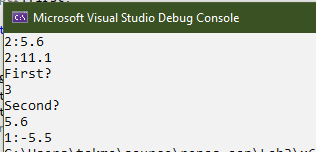
}

return \*this;

}

**Вывод программы**

Ниже представлен вывод программы на консоль.



**Контрольные вопросы**

1. Для чего используются дружественные функции и классы?

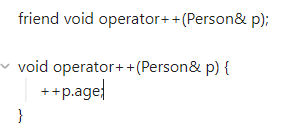
Ответ: для расширения возможностей класса/ интерфейса



1. Сформулировать правила описания и особенности дружественных функций.

Ответ: описание с ключевым словом friend. В качестве параметра передается объект класса. На дружественную функцию не распространяется действие спецификаторов доступа. Функция может быть дружественной сразу нескольким классам.

1. Каким образом можно перегрузить унарные операции?

Ответ: как функцию класса и как дружественную функцию

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

Ответ: один либо ноль

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

Ответ: два либо один (так как передается указатель на объект класса)

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

Ответ: один

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

Ответ: два

1. Чем отличается перегрузка префиксных и постфиксных унарных операций?

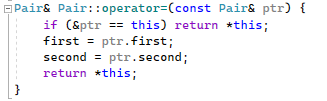
Ответ: при перегрузке постфиксных операций передать в функцию параметр int для объяснения разницы между операциями ++. Также операторы отличаются по типу возвращаемого значения

1. Каким образом можно перегрузить операцию присваивания?

Ответ: как компонентную функцию класса

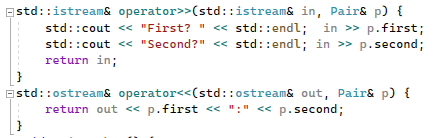
1. Что должна возвращать операция присваивания?

Ответ: ссылку на данный объект



1. Каким образом можно перегрузить операции ввода-вывода?

Ответ: при помощи дружественных функций

\_

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: 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?

Ответ: 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?

Ответ: приоритет операции « выше. Не скомпилируется. В другом случае вызов 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?

Ответ: приоритет операции « выше. Не скомпилируется. В другом случае вызов operator>(a, b)