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

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 – «Программная инженерия»

**Лабораторная работа №3.**

**«Перегрузка операций»**

Выполнил студент гр. РИС-24-2б

Молочко Артём Анатольевич

Проверил:

Доц. Каф. ИТАС

Ольга Андреевна Полякова

(оценка) (подпись)

(дата)

г. Пермь, 2025

**Постановка задачи**

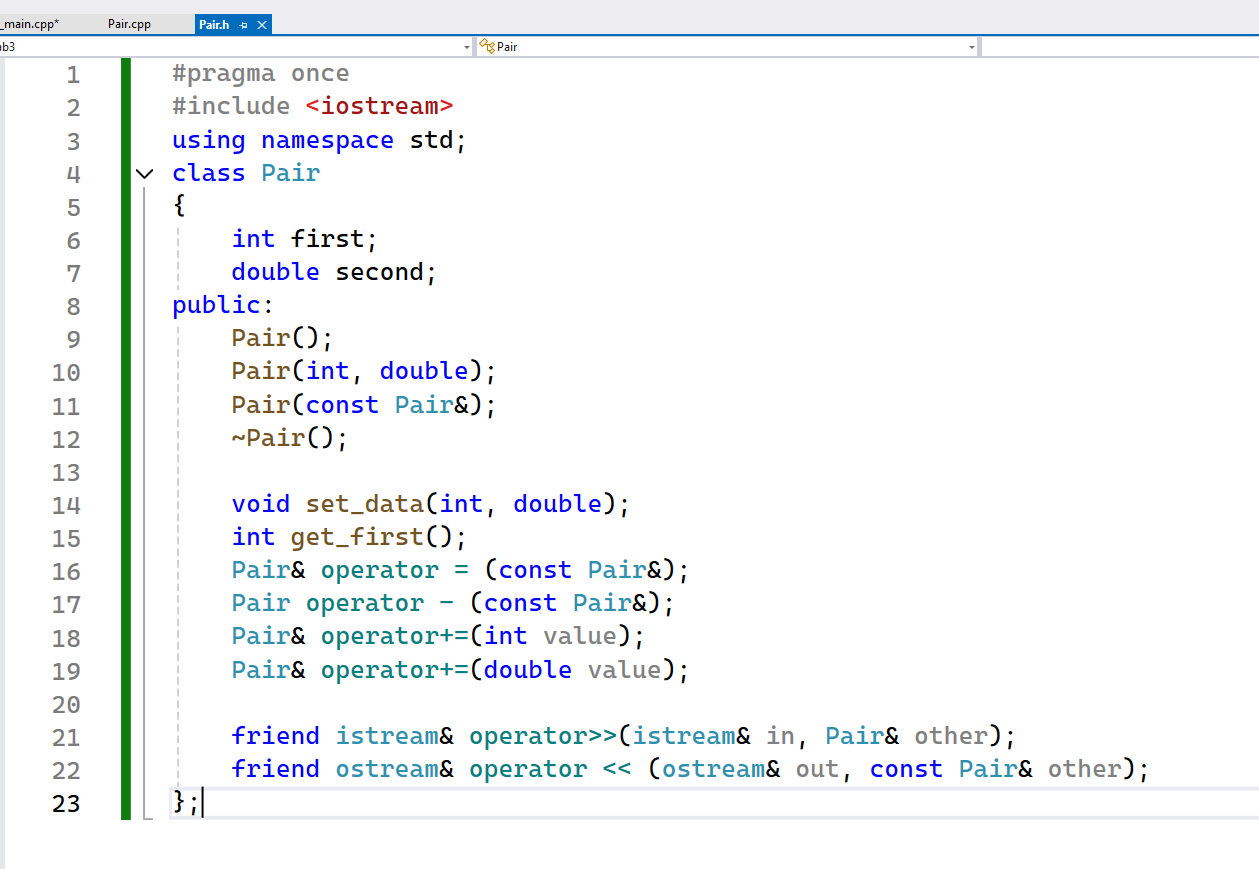
Разработать класс Pair (пара чисел), в котором:

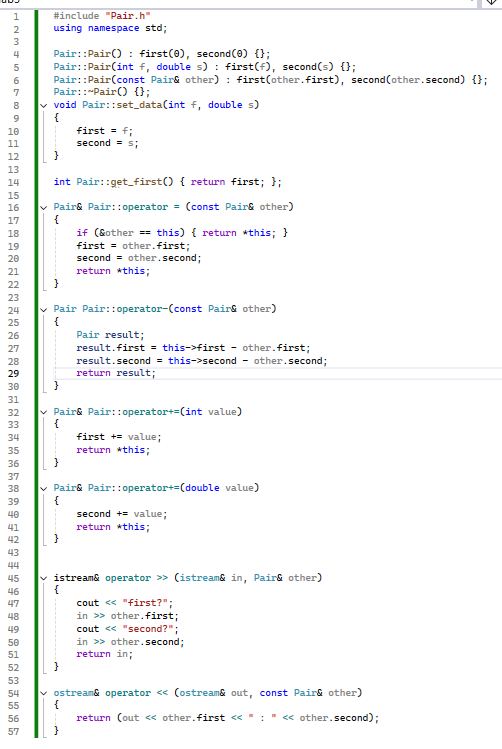
1. **Поля данных:**
   * first — целое число (int).
   * second — вещественное число (double).
2. **Конструкторы:**
   * Конструктор по умолчанию (без параметров).
   * Конструктор с параметрами (инициализация first и second).
   * Конструктор копирования.
3. **Деструктор.**
4. **Методы доступа:**
   * Селекторы (получение значений полей).
   * Модификаторы (установка значений полей).
5. **Перегруженные операции:**
   * Присваивания (=).
6. Ввода (>>) и вывода (<<) объектов через потоки. Формат вывода: first : second
   * Арифметические операции:
     + Вычитание пар (-).
     + Добавление константы к паре (+).
       - Если константа целая (int), увеличивается first.
       - Если константа вещественная (double), увеличивается second.
7. **Демонстрационная программа:**
   * Создание объектов класса Pair разными конструкторами.
   * Тестирование всех перегруженных операций.
   * Взаимодействие с пользователем (ввод/вывод данных).

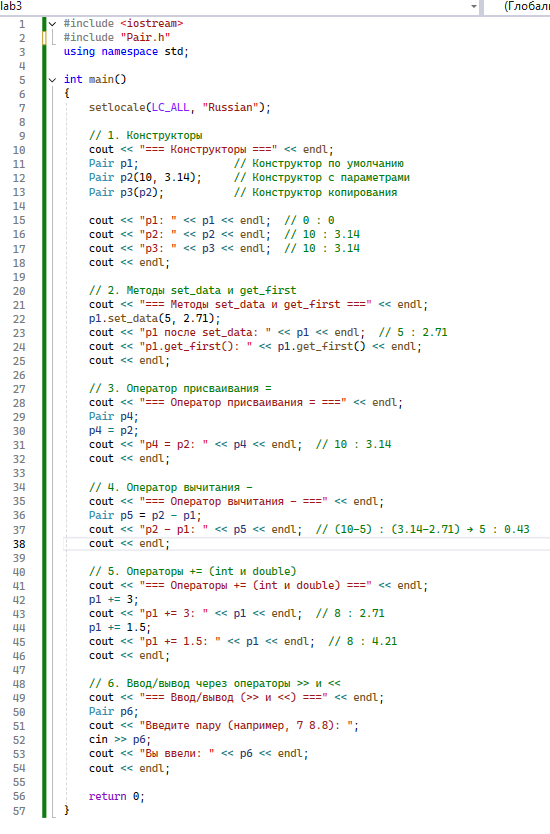
**Анализ задачи**

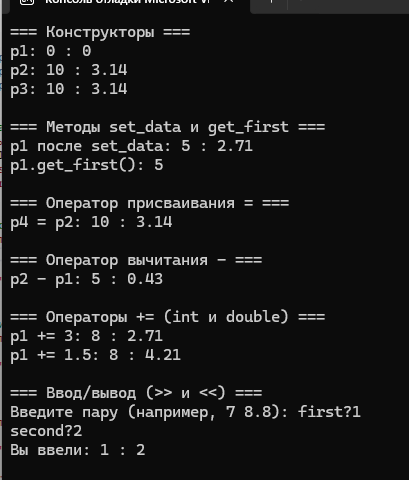
* **Конструктор по умолчанию** (Pair()) инициализирует поля first = 0, second = 0.0. Корректно задает начальные значения.
* **Параметризованный конструктор** (Pair(int, double)) позволяет явно установить значения полей.
* **Конструктор копирования** (Pair(const Pair&)) выполняет глубокое копирование.
* **Деструктор** (~Pair()) пустой, так как класс не использует динамическую память.
* **set\_data(int, double)** — модификатор для установки значений полей.
* **get\_first()** — селектор для поля first.
* **Добавление константы (+=)**
  + operator+=(int) увеличивает first.
  + operator+=(double) увеличивает second.
  + Реализация соответствует заданию.
* **Ввод (>>)**
  + Запрашивает значения полей поочередно.
* **Вывод (<<)**
  + Формат вывода first : second соответствует условию.

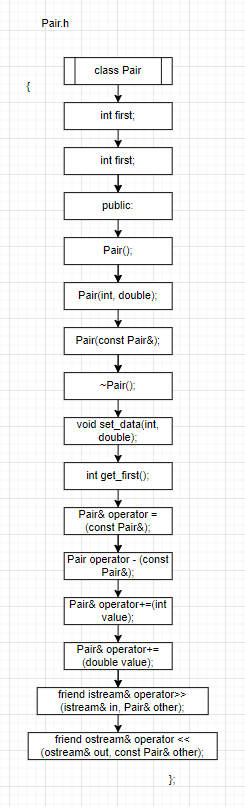
**Код программы**

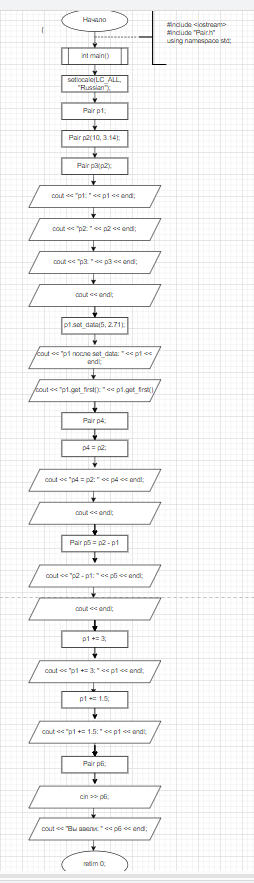


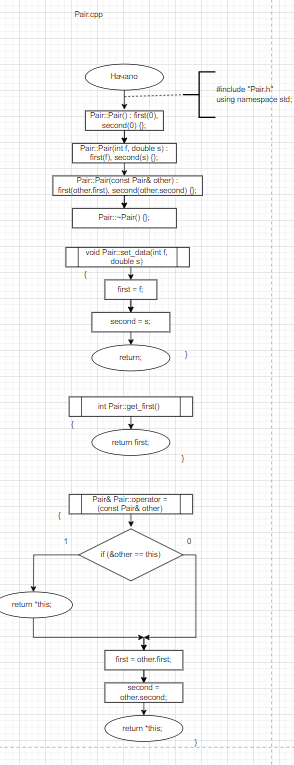
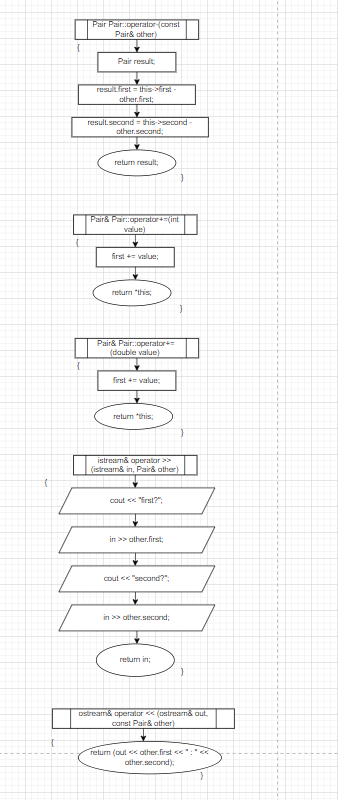






**Блок-схема**





**Контрольные вопросы**

**1. Для чего используются дружественные функции и классы?**

**Ответ:** Дружественные функции и классы предоставляют доступ к приватным и защищенным членам класса, не являясь его частью.

**2. Правила описания и особенности дружественных функций.**

**Ответ:**

* Объявляются внутри класса с ключевым словом friend.
* **Не являются методами класса** (не имеют this).
* Определяются **вне класса** без указания области видимости (ClassName::).
* Имеют доступ ко всем полям класса, включая приватные.

#### ****3. Как перегрузить унарные операции?****

**Ответ:**  
Унарные операции перегружаются:

1. **Внутри класса** (как методы):
   * Префиксная форма: operator++().
   * Постфиксная форма: operator++(int)
2. **Вне класса** (как дружественные функции):
   * Префиксная: operator++(Person&).
   * Постфиксная: operator++(Person&, int).

**Пример (префиксный ++ внутри класса):**

Person& operator++() {

++age;

return \*this;

}

};

#### ****4. Сколько операндов у унарной функции-операции внутри класса?****

**Ответ:**

* **Префиксная:** 0 (operator++()).
* **Постфиксная:** 1 (operator++(int)).

#### ****5. Сколько операндов у унарной функции-операции вне класса?****

**Ответ:**  
1 (объект, с которым работает операция).

Person& operator++(Person& p) { ... } // Один параметр

#### ****6. Сколько операндов у бинарной функции-операции внутри класса?****

**Ответ:**  
1

class Person {

public:

Person operator+(int x) { ... } // this + x

};

#### ****7. Сколько операндов у бинарной функции-операции вне класса?****

**Ответ:** 2

Person operator+(const Person& p, int x) { ... } // p + x

#### ****8. Отличие перегрузки префиксных и постфиксных унарных операций****

**Ответ:**

* **Префиксная** (++x):
  + Возвращает **ссылку** на измененный объект.

Person& operator++() {

++age;

return \*this;

}

**Постфиксная** (x++):

* Возвращает **копию** объекта до изменения.

Person operator++(int)

{

Person temp = \*this;

++age;

return temp;

}

#### ****9. Как перегрузить операцию присваивания?****

**Ответ:**

* Определяется как метод класса.
* Должна проверять **самоприсваивание**.
* Возвращает \*this для цепочки присваиваний.

class Person {

public:

Person& operator=(const Person& p) {

if (&p == this) return \*this; // Проверка на самоприсваивание

name = p.name;

age = p.age;

return \*this;

}

};

#### ****10. Что должна возвращать операция присваивания?****

**Ответ:**  
**Ссылку** на текущий объект (ClassName&).

#### ****11. Как перегрузить операции ввода-вывода?****

**Ответ:**  
Только как **дружественные функции**, так как левый операнд — поток (istream/ostream).

class Person {

friend istream& operator>>(istream& in, Person& p);

friend ostream& operator<<(ostream& out, const Person& p);

};

istream& operator>>(istream& in, Person& p) {

in >> p.name >> p.age;

return in;

}

ostream& operator<<(ostream& out, const Person& p) {

out << p.name << ", " << p.age;

return out;

}

#### ****12. Вызов****++s****для****Student& operator++();

**Ответ:** Компилятор вызовет s.operator++().

#### ****13. Вызов****++s****для****friend Student& operator++(Student&);

**Ответ:** Компилятор вызовет operator++(s).

#### ****14. Вызов****cout << a < b;****для****bool operator<(Student &P);

**Ответ:** Ошибка. Правильно: cout << (a < b); (из-за приоритета операторов).

#### ****15. Вызов****cout << a > b;****для****friend bool operator>(const Student&, Student&);

**Ответ:** Ошибка. Правильно: cout << (a > b);.