Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**о работе по информатике**

Семестр: 2

На тему: Лабораторная работа №3: «Перегрузка операций».

**Вариант 13**

Выполнил студент ИВТ-22-2б:

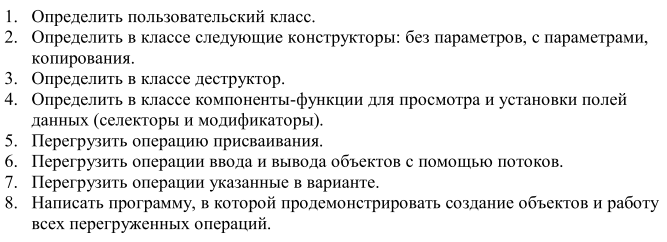
Коняев Александр Сергеевич

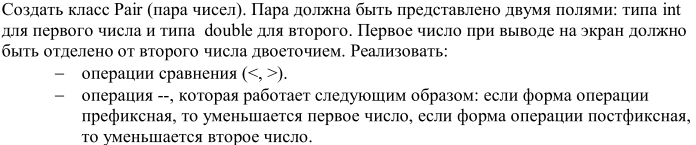
Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

Пермь 2023

**Постановка задачи**





**Анализ задачи**

* Конструктор без параметров
* Конструктор с параметрами
* Конструктор копирования
* Метод get\_first – получение первого числа
* Метод set\_first – запись первого числа
* Метод get\_second – получение второго числа
* Метод set\_second – запись второго числа
* Перегруженные операторы: =, <, >, --, >>, <<

**UML – диаграмма**

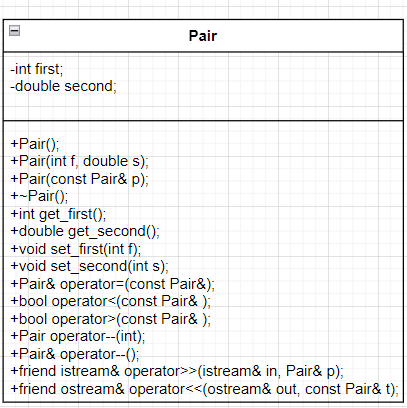


Рисунок 1 – UML-диаграмма класса «Pair»

**Код программы.**

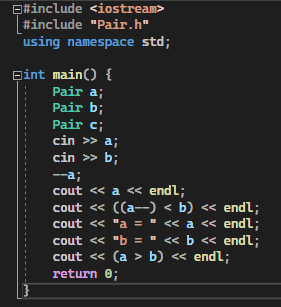


Рисунок 2 – Функция main

**Вывод программы.**

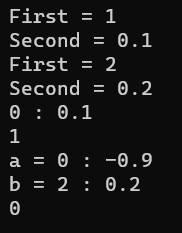
****

Рисунок 3 – Вывод программы

**Ответы на вопросы**

1. Дружественные функции и классы в C++ используются для предоставления доступа к закрытым (private) членам класса другим функциям и классам.

Когда функция или класс объявляются дружественными, они получают доступ к закрытым членам класса, таким как переменные и методы, которые не могут быть использованы извне. Это может быть полезно в тех случаях, когда нужно предоставить доступ к приватным данным из другого класса или функции без необходимости делать эти данные открытыми (public).

Дружественные функции и классы объявляются с помощью ключевого слова friend внутри класса. Например:

class MyClass {

private:

int myPrivateVariable;

void myPrivateMethod();

friend void myFriendFunction(MyClass& obj);

friend class MyFriendClass;};

1. Основные правила описания дружественных функций в C++:

1. Дружественная функция объявляется внутри класса с помощью ключевого слова friend.

2. Дружественная функция не является методом класса и не имеет доступа к членам класса без явного указания объекта.

3. Дружественная функция может быть определена вне класса и может иметь доступ к закрытым членам класса.

4. Дружественная функция может быть перегружена, но только одна версия может быть объявлена дружественной.

5. Дружественная функция не наследуется подклассами.

6. Дружественная функция не может быть виртуальной.

7. Дружественная функция не может быть конструктором или деструктором.

8. Дружественная функция не может иметь доступ к приватным и защищенным членам других классов.

Особенности дружественных функций:

1. Дружественные функции могут быть полезны для упрощения кода и улучшения производительности, но их использование следует ограничивать, чтобы не нарушать инкапсуляцию.

2. Дружественные функции могут использоваться для реализации операций, которые не являются методами класса, но требуют доступа к закрытым членам класса.

3. Дружественные функции могут быть использованы для улучшения безопасности данных, поскольку они позволяют ограничить доступ к закрытым данным только для определенных функций и классов.

4. Дружественные функции могут быть использованы для реализации операторов, таких как операторы ввода/вывода, которые должны иметь доступ к приватным членам класса.

1. В C++ унарные операции могут быть перегружены с помощью функций-членов класса или с помощью глобальных функций.

Перегрузка унарных операций с помощью функций-членов класса:

1. Определите функцию-член класса с именем оператора, который вы хотите перегрузить, и ключевым словом operator перед ним.

2. Функция должна принимать ноль аргументов.

3. Функция должна возвращать значение, которое будет результатом операции.

4. В теле функции используйте ключевое слово this для доступа к членам класса.

Пример перегрузки унарного оператора + с помощью функции-члена класса:

class MyClass {

public:

MyClass operator+() {

// реализация операции

}

};

1. Унарная функция-операция, определяемая внутри класса, должна иметь один операнд.

В C++ унарные операции могут быть перегружены с помощью функций-членов класса или с помощью глобальных функций. В обоих случаях унарная операция принимает один операнд.

Например, если вы хотите перегрузить унарный оператор + для класса MyClass, вы можете определить его как функцию-член класса или как глобальную функцию, но в любом случае он должен принимать один операнд:

class MyClass {

public:

MyClass operator+() {

// реализация операции

}

};

MyClass operator+(const MyClass& obj) {

// реализация операции

}

1. Унарная функция-операция, определяемая вне класса, должна также иметь один операнд.

В C++ унарные операции могут быть перегружены с помощью глобальных функций. В этом случае унарная операция принимает один операнд - объект класса, к которому применяется операция.

Например, если вы хотите перегрузить унарный оператор + для класса MyClass, определите его как глобальную функцию, которая принимает объект класса MyClass в качестве аргумента:

class MyClass {

// определение класса

};

MyClass operator+(const MyClass& obj) {

// реализация операции

}

1. Бинарная функция-операция, определяемая внутри класса, должна иметь один операнд, который передается в качестве второго аргумента.

В C++ бинарные операции могут быть перегружены с помощью функций-членов класса или с помощью глобальных функций. В случае функций-членов класса первый операнд автоматически принимается за объект, для которого вызывается оператор, а второй операнд передается в качестве аргумента.

Например, если вы хотите перегрузить бинарный оператор + для класса MyClass, вы можете определить его как функцию-член класса, которая принимает один операнд:

class MyClass {

public:

MyClass operator+(const MyClass& obj) {

// реализация операции

}

};

1. Бинарная функция-операция, определяемая вне класса, должна иметь два операнда - левый и правый операнды.

В C++ бинарные операции могут быть перегружены с помощью глобальных функций. В этом случае бинарная операция принимает два операнда - левый и правый, которые передаются в качестве аргументов.

Например, если вы хотите перегрузить бинарный оператор + для класса MyClass, определите его как глобальную функцию, которая принимает два операнда - объекты класса MyClass:

class MyClass {

// определение класса

};

MyClass operator+(const MyClass& obj1, const MyClass& obj2) {

// реализация операции

}

1. Перегрузка префиксных и постфиксных унарных операций отличается тем, как они применяются к объекту класса и как они возвращают результат.

Префиксная унарная операция применяется к объекту класса до того, как он используется в выражении. Это означает, что операция выполняется до того, как объект возвращается из выражения. Возвращаемое значение - это измененный объект.

Постфиксная унарная операция применяется к объекту класса после того, как он используется в выражении. Это означает, что операция выполняется после того, как объект возвращается из выражения. Возвращаемое значение - это копия исходного объекта, до его изменения.

Например, если вы перегружаете унарный оператор инкремента (`++`) для класса `MyClass`, то префиксная версия изменяет объект и возвращает его измененное значение, а постфиксная версия возвращает копию исходного объекта до его изменения.

Пример перегрузки префиксной и постфиксной унарных операций для класса `MyClass`:

```

class MyClass {

public:

MyClass operator++() { // префиксная версия

// изменение объекта

return \*this;

}

MyClass operator++(int) { // постфиксная версия

MyClass temp(\*this); // создание копии исходного объекта

// изменение объекта

return temp; // возвращение копии

1. Операция присваивания в языке программирования может быть перегружена с помощью перегрузки оператора присваивания. Для этого необходимо создать функцию с именем operator=, которая будет принимать один параметр - ссылку на объект того же типа, что и текущий. Внутри этой функции можно определить новое поведение для операции присваивания.

Например, если у нас есть класс MyClass, то перегрузка операции присваивания может выглядеть следующим образом:

class MyClass {

public:

MyClass& operator=(const MyClass& other) {

// определение нового поведения для операции присваивания

return \*this;

}

};

1. Операция присваивания должна возвращать ссылку на текущий объект, т.е. объект, который был присвоен. Это позволяет использовать операцию присваивания в цепочке, например:

MyClass a, b, c;

a = b = c;

1. Операции ввода-вывода в языке программирования могут быть перегружены с помощью перегрузки операторов << и >>. Для этого необходимо создать функции с именами operator<< и operator>>, которые будут принимать два параметра: ссылку на поток ввода или вывода и ссылку на объект, который нужно записать или считать.

Например, если у нас есть класс MyClass, то перегрузка операций ввода-вывода может выглядеть следующим образом:

#include <iostream>

class MyClass {

public:

int value;

friend std::ostream& operator<<(std::ostream& os, const MyClass& obj) {

return os << obj.value;

}

friend std::istream& operator>>(std::istream& is, MyClass& obj) {

return is >> obj.value;

}

};

int main() {

MyClass obj;

std::cout << "Введите значение: ";

std::cin >> obj;

std::cout << "Вы ввели: " << obj << std::endl;

return 0;

}

1. Компилятор воспримет операцию ++s как вызов функции-оператора "operator++()" для объекта s класса student.

В данном случае, функция-оператор "operator++()" перегружена для класса student и возвращает ссылку на объект student. При вызове оператора "++s", будет выполнено увеличение значения полей объекта s, которое определено внутри функции-оператора "operator++()".

Таким образом, операция "++s" увеличит значение полей объекта s и вернет ссылку на этот же объект student.

1. Компилятор воспримет операцию ++s как вызов функции-оператора "operator++(student&)", которая определена как friend в классе student.

В данном случае, функция-оператор "operator++(student&)" принимает ссылку на объект student и возвращает ссылку на этот же объект student. При вызове оператора "++s", будет выполнено увеличение значения полей объекта s, которое определено внутри функции-оператора "operator++(student&)".

Таким образом, операция "++s" увеличит значение полей объекта s и вернет ссылку на этот же объект student. Ключевое слово friend используется для разрешения доступа к приватным членам класса из функции-оператора, которая не является методом класса.

1. Компилятор воспримет операцию a<b, как операцию сравнения объектов класса.
2. Компилятор воспримет операцию a>b, как операцию сравнения объектов класса.