Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (ПНИПУ)

Электротехнический факультет

Дисциплина: основы алгоритмизации и программирования, 2 семестр

**ОТЧЁТ**

Тема: «Перегрузка операций»

Выполнил

Студент РИС-22-2б

Зубов Р.А.

Проверил

Доцент кафедры ИТАС

Полякова О. А.

Пермь 2023

# **Содержание**

Постановка задачи**4**

Коды программ**5**

Результаты программ**6**

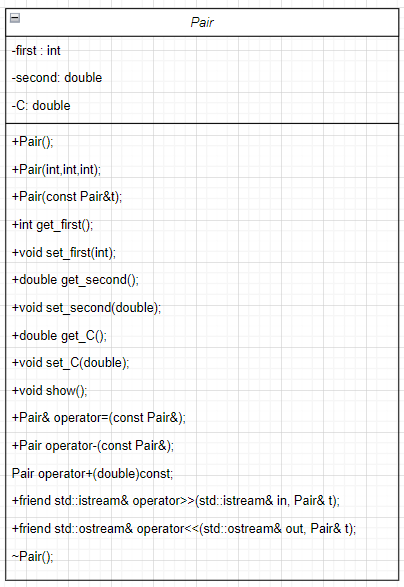
**Постановка задачи**

Определить пользовательский класс. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования. Определить в классе деструктор. Определить в классе компоненты – функции для просмотра и установки полей данных(селекторы и модификаторы). Перегрузить операцию присваивания. Перегрузить операции ввода и вывода объектов с помощью потоков. Перегрузить операции указанные в варианте. Написать программу, в которой продемонстрировать создание объектов и работу всех перегруженных операций.

**Задача**

Создать класс Pair(пара чисел). Пара должна быть представлена двумя полями : типа int для первого числа и типа double для второго. Первое число при выводе на экран должно быть отделено от второго двоеточием. Реализовать: вычитание пар чисел; добавление константы к паре(увеличивается первое число, если константа целая, второе, если константа вещественная)

**UML диаграммы**

****

**Описание класса**

[Класс Pair содержит два поля типа int и одно поле типа double](http://mycpp.ru/cpp/book/c03_14.html).

Конструкторы класса Pair:

* Конструктор по умолчанию создает объект с полями first = 0, second = 0 и C = 0.
* Конструктор с параметрами создает объект с заданными значениями полей first, second и C.
* Конструктор копирования создает новый объект с теми же значениями полей, что и у переданного объекта.

Методы класса Pair:

* Метод get\_first() возвращает значение поля first.
* Метод get\_second() возвращает значение поля second.
* Метод set\_first(int f) устанавливает значение поля first равным f.
* Метод set\_second(double s) устанавливает значение поля second равным s.

Перегруженные операторы класса Pair:

* Оператор присваивания (=).
* Оператор вычитания (-).
* Оператор сложения (+).

Дружественные функции класса Pair:

* Функция operator>> перегружает оператор >> для класса Pair и позволяет вводить значения полей объекта класса Pair из потока ввода.
* Функция operator<< перегружает оператор << для класса Pair и позволяет выводить значения полей объекта класса Pair в поток вывода.

**Определение компонентных функций**

Компонентные функции класса Pair перегружают операторы =, - и + для объектов класса Pair.

Оператор присваивания (=) перегружен следующим образом:

* Если переданный объект является текущим объектом, то возвращается ссылка на текущий объект.
* Иначе поля текущего объекта устанавливаются равными полям переданного объекта.

Оператор вычитания (-) перегружен следующим образом:

* Создается новый объект класса Pair.
* Поля нового объекта устанавливаются равными разности соответствующих полей текущего и переданного объектов.
* Новый объект возвращается.

Оператор сложения (+) перегружен следующим образом:

* Создается новый объект класса Pair.
* Если переданный параметр C является целым числом, то первое поле нового объекта устанавливается равным сумме первого поля текущего объекта и C, а второе поле нового объекта устанавливается равным второму полю текущего объекта.
* Иначе первое поле нового объекта устанавливается равным первому полю текущего объекта, а второе поле нового объекта устанавливается равным сумме второго поля текущего объекта и C.
* Новый объект возвращается.

**Определение глобальных функций**

Глобальные функции std::istream& operator>>(std::istream&in, Pair&t) и std::ostream& operator<<(std::ostream & out, Pair &t) перегружают операторы >> и << для объектов класса Pair соответственно.

Оператор ввода (>>) перегружен следующим образом:

* Вводится первое поле объекта.
* Вводится второе поле объекта.
* Возвращается ссылка на поток ввода.

Оператор вывода (<<) перегружен следующим образом:

* Выводится первое поле объекта.
* Выводится разделитель " : ".
* Выводится второе поле объекта.
* Выводится символ новой строки.
* Возвращается ссылка на поток вывода.

**Определение функции main()**

 main() создает два объекта класса Pair, запрашивает у пользователя их значения с помощью глобальной функции std::istream& operator>>(std::istream&in, Pair&t), выводит их значения с помощью глобальной функции std::ostream& operator<<(std::ostream & out, Pair &t), вычитает один объект из другого и выводит результат с помощью глобальной функции std::ostream& operator<<(std::ostream & out, Pair &t), а также добавляет константу к одному из объектов и выводит результат с помощью глобальной функции std::ostream& operator<<(std::ostream & out, Pair &t).

**Контрольные вопросы**

1. Для чего используются дружественные функции и классы?

Чтобы предоставить доступ к private – полям класса, методам другого

класса, который является дружественным.

Чтобы предоставить доступ к private-полям глобальным не

компонентным функциям.

2. Сформулируйте правила описания и особенности дружественных

функций.

- Дружественная функция объявляется внутри класса с ключевым

словом friend.

- Поскольку дружественная функция не является компонентной

(ей не передается указатель this), то необходимо, чтобы она принимала в

качестве параметра объект класса по ссылке, по значению или по адресу.

- Дружественная функция может быть дружественной сразу к

нескольким классам.

- На неё не распространяются спецификаторы доступа, поэтому

то место, где она описана в классе, неважно.

3. Каким образом можно перегрузить унарные операции.

-Как компонентные нестатические функции класса.

тип operator “знак оператора”(void);

-Как обычная глобальная не компонентная функция, которая

также может быть дружественная классу.

тип operator “знак оператора”(class A), где class A – передача

объекта класса.

- Унарные операторы перегружаются чаще всего как методы класса.

4. Сколько операндов должна иметь унарная функция-операция,

определяемая внутри класса.

- Унарная операция по определению работает с одним операндом. Раз

она перегружается как компонентная нестатическая функция, то она не

должна принимать параметров. (неявно принимает указатель this).

5. Сколько операндов должна иметь унарная функция-операция,

определяемая вне класса.

- Т.к. оператор перегружается как глобальная функция, то параметр this

ему не передается, следовательно необходимо явно передавать объект класса.

6. Сколько операндов должная иметь бинарная функция-операция,

определяемая внутри класса?

- Если оператор перегружается как компонентная функция, то левым

операндом по умолчанию является объект класса – this. Правым операндом

является тот объект, что передается в качестве параметра в перегружаемый

оператор.

Например: A operator+(const int& value); - тут для класса A

перегружается оператор сложения как метод класса. –>

a.operator+(5) ≡ a + 5. где a – объект класса.

Ответ: один.

7. Сколько операндов должная иметь бинарная функция-операция,

определяемая вне класса?

- Если оператор перегружается как не компонентная функция, чаще

всего дружественная классу. То указатель this не передается, поэтому,

необходимо явно передавать объект класса в качестве параметра. Также

необходимо передавать объект другого класса, с которым должен

взаимодействовать исходный класс посредством оператора.

Например:

Из лабораторной работы №3: -

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

Существуют бинарные операторы, перегрузка которых вне класса

невозможна: это “->”, “[]”, “()”, “=”

Ответ: два

8. Чем отличается перегрузка префиксных и постфиксных унарных операций.

-Префиксные и постфиксные операции по сути являются версией

одного оператора в разных формах. Если при перегрузке префиксного

оператора не нужно передавать никаких параметров, то при перегрузке

постфиксного оператора необходимо передать незначащий параметр int. –

Чтобы объяснить компилятору разницу.

Также эти операторы могут отличатся по типу возвращаемого

значения. Допустим, если префиксный оператор (инкремента или

декремента) модифицирует какое-либо информационное поле, а затем

возвращает ссылку на объект этого класса, то постфиксный оператор должен

сохранить состояние объекта класса во временную переменную, затем

модифицировать поле класса, затем вернуть копию предыдущего состояния.

– Это накладывает некоторые ограничения на использование постфиксных

операторов, т.к. они не позволяют взаимодействовать напрямую с объектом

класса.

9. Каким образом можно перегрузить операцию присваивания.

- Оператор присваивания можно перегрузить только как нестатическую

компонентную функцию класса.

10. Что должна возвращать операция присваивания?

- Ссылку на объект класса, в который происходит копирование (левый

операнд). Это нужно для реализации многочисленного присваивания.

Например: a = b = c; Где a, b и c – Объекты одного класса.

11. Каким образом можно перегрузить операции ввода-вывода?

- Для того, чтобы обеспечить взаимодействие пользовательского класса

и потокового класса (левым операндом является объект потокового класса,

правым операндом является объект пользовательского класса), необходимо

перегрузить оператор<< или оператор>> как дружественную функцию. С

двумя параметрами – первый: объект класса std::ostream или std::istream,

второй: объект пользовательского класса.

Например: friend std::ostream& operator<<(std::ostream& stream, const

Pair& p)

Тут operator << объявлен в классе как дружественная функция.

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?

-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?

-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?

Приоритет оператора << выше, чем у оператора<. Программа просто не

скомпилируется.

Если выполнялась бы операция cout<<(a<b); То компилятор это воспринял

бы 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?

Приоритет у оператора << выше, программа не будет работать.

Если выполняется операция cout << (a>b); то компилятор это будет

воспринимать как вызов глобальной функции: cout << operator>(a, b);