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: 09.03.01 - «Информатика и вычислительная техника»

**О Т Ч Е Т**

**по лабораторной работе “Классы” №3**

**по дисциплине**

**«Основы алгоритмизации и программирования» семестр 2**

Выполнил студент гр. ИВТ-21-1б

Ипатов Дмитрий Сергеевич

Проверил:

Ст. Преподаватель кафедры ИТАС

Яруллин Д.В.

(оценка) (подпись)

г. Пермь-2022

**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, в которой продемонстрировать создание объектов и работу всех перегруженных операций.

**Анализ задачи:**

1. Операция присваивания определена в любом классе по умолчанию как поэлементное копирование. Эта операция вызывается каждый раз, когда одному существующему объекту присваивается значение другого. Если класс содержит поля, память под которые выделяется динамически, необходимо определить собственную операцию присваивания. Чтобы сохранить семантику присваивания, операция-функция должна возвращать ссылку на объект, для которого она вызвана, и принимать в качестве параметра единственный аргумент — ссылку на присваиваемый объект.
2. Операции ввода-вывода operator>> и operator<< всегда реализуются как внешние дружественные функции, т. к. левым операндом этих операций являются потоки.
3. Бинарную операцию можно перегрузить:
   * 1. Как компонентную функцию класса
     2. Как внешнюю (глобальную) функцию

Бинарная функция-операция, определяемая *внутри класса,* должна быть представлена с помощью нестатического метода с параметрами, при этом вызвавший ее объект считается первым операндом

**Код программы:**

#include <iostream>

using namespace std;

class Pair

{

private:

int first;

double second;

public:

Pair() { first = 0; second = 0; }

Pair(double m, int n) { first = n; second = m; }

Pair(const Pair& t) { first = t.first; second = t.second; }

~Pair() {}

int get\_first() { return first; }

double get\_second() { return second; }

void set\_first(int n) { first = n; }

void set\_second(int m) { second = m; }

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

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

Pair& operator--();

Pair operator --(int);

bool operator>(Pair& t);

bool operator<(Pair& t);

Pair& operator=(const Pair& t);

};

istream& operator>>(istream& in, Pair& p)

{

cout << "first:"; in >> p.first;

cout << "second:"; in >> p.second;

return in;

}

ostream& operator<<(ostream& out, const Pair& p)

{

return (out << p.first << ":" << p.second << endl);

}

Pair Pair::operator --(int)

{

first--;

return \*this;

}

Pair& Pair::operator--()

{

second--;

return \*this;

}

Pair& Pair::operator=(const Pair& t)

{

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

first = t.first;

second = t.second;

return \*this;

}

bool Pair::operator<(Pair& t)

{

if (first < t.first && second < t.second) return true;

else return false;

}

bool Pair::operator>(Pair& t)

{

if (first > t.first && second > t.second) return true;

else return false;

}

int main()

{

setlocale(LC\_ALL, "rus");

bool d;

Pair a, b, c;

cin >> a >> b;

c = a--;

--b;

cout << c << b;

d=a>b;

cout << d << '\n';

d = c < a;

cout << d;

return 0;

}

**Скриншоты результатов:**

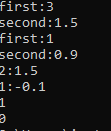
****

Рис. 1 Результаты программы

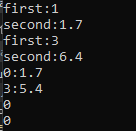
****

Рис. 2 Другие результаты программы

**Контрольные вопросы:**

1. Для чего используются дружественные функции и классы?

Они используются для того, чтобы дать доступ объекту или классу к полям класса, компонентом которого они не являются.

1. Сформулировать правила описания и особенности дружественных функций.

Дружественная функция объявляется внутри класса, к элементам которого ей нужен доступ, с ключевым словом friend.

1. Каким образом можно перегрузить унарные операции?

Как компонентную (внутри класса) или как глобальную функцию с ссылкой на класс.

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

Один, при этом операндом является вызвавший её объект.

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

Один, при этом она должна иметь один параметр типа класса.

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

Два, при этом вызвавший ее объект считается первым операндом.

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

Если функция определяется *вне класса,* она должна иметь два параметра типа класса.

1. Чем отличается перегрузка префиксных и постфиксных унарных операций?

Разница заключается в значении, которое вернется. Префиксные возвращают объект после его инкрементирования или декрементирования, когда постфиксные же возвращают объект до того как с ним будут выполнены данные действия.

1. Каким образом можно перегрузить операцию присваивания?

Пишется тип класса с ссылкой. Также должно быть написано ключевое слово “operator” и вслед за ним оператор присваивания. В скобках, в качестве параметра указывается ссылка на константное значение (объект класса).

1. Что должна возвращать операция присваивания?

Должна возвращать ссылку на объект, для которого она вызвана.

1. Каким образом можно перегрузить операции ввода-вывода?

В классе нужно определить потоки ввода и вывода как дружественные функции, написать ключевое слово “operator”, после написать “<<” или “>>” в зависимости от того вывод это или ввод, в качестве параметров указываем ссылку на переменную ввода, вывода, и ссылку на объект класса (в случае вывода делаем его константой, чтобы данные никак нельзя было изменить во время вывода).

1. 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?

S увеличится на 1 и далее уже с изменённым объектом будут выполнятся какие-либо действия.

1. 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?

Вернется значение до инкремента и с ним будут выполнятся действия, и только потом оно увеличится на 1.

1. 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?

Он сравнит значения объекта а и объекта b, так как это прописал пользователь. И если значение(я) a меньше значения(й) b, функция вернет true, или false, если это не так.

1. 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?

Он сравнит значения объекта а и объекта b, так как это прописал пользователь. И если значение(я) a больше значения(й) b, функция вернет true, или false, если это не так.