Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

по лабораторной работе № 4

«Перегрузка операторов»

Выполнил

ст. гр. ИТб22

Серегин А.В.

Проверил:

Тимофеев И.С.

Севастополь

2015

1. Цель работы

Приобретение практических навыков при написании объектно-ориентированных программ с использованием перегруженных операторов. Освоение особенностей отладки объектно-ориентированных программ.

2. Варианты заданий

Для заданного по варианту класса выполнить следующие действия:

- описать все типы конструкторов и деструктор (по необходимости);

- переопределить оператор вывода в поток <<;

- переопределить оператор ввода из потока >>;

- переопределить заданные по варианту операторы;

- предусмотреть обработку ошибок.

Создать два объекта заданного по варианту класса и на их примере продемонстрировать корректную работу всех перегруженных операторов.

Вариант 5

Создать класс координат *Coord* (содержит поля *х, у* и *delta*). Перегрузить операторы:

1) ++ как унарный метод класса, увеличивающий значение координат на шаг (*delta*);

2) ! как унарную дружественную функцию, проверяющую нулевое значение координат;

3) - как бинарный метод класса, складывающий два объекта (только координаты);

4) >= как бинарную дружественную функцию, сравнивающую среднее квадратичное значение координат двух объектов.

3. Текст программы

Coord.h

#pragma once

#ifndef \_COORD\_H\_

#define \_COORD\_H\_

#include <iostream>

class Coord

{

private:

double x\_;

double y\_;

double delta\_;

public:

Coord();

Coord(const double& \_x, const double& \_y);

Coord(const double& \_x, const double& \_y, const double& \_delta);

void setX(const double& \_x);

void setY(const double& \_y);

void setDelta(const double& \_delta);

double getX() const;

double getY() const;

double getDelta() const;

void operator++();

void operator--();

bool operator!() const;

Coord operator+(const Coord& \_right) const;

Coord operator-(const Coord& \_right) const;

friend bool operator>=(const Coord& \_left, const Coord& \_right);

friend std::ostream& operator<<(std::ostream& os, const Coord& \_coord);

friend std::istream& operator>>(std::istream& os, Coord& \_coord);

~Coord(){};

};

#endif

Coord.cpp

#include "Coord.h"

Coord::Coord()

{

x\_ = 0.0;

y\_ = 0.0;

delta\_ = 1.0;

}

Coord::Coord(const double & \_x, const double & \_y)

{

x\_ = \_x;

y\_ = \_y;

delta\_ = 1.0;

}

Coord::Coord(const double & \_x, const double & \_y, const double & \_delta)

{

x\_ = \_x;

y\_ = \_y;

delta\_ = \_delta;

}

void Coord::setX(const double & \_x)

{

x\_ = \_x;

}

void Coord::setY(const double & \_y)

{

y\_ = \_y;

}

void Coord::setDelta(const double & \_delta)

{

delta\_ = \_delta;

}

double Coord::getX() const

{

return x\_;

}

double Coord::getY() const

{

return y\_;

}

double Coord::getDelta() const

{

return delta\_;

}

void Coord::operator++()

{

x\_ += delta\_;

y\_ += delta\_;

}

void Coord::operator--()

{

x\_ -= delta\_;

y\_ -= delta\_;

}

bool Coord::operator!() const

{

return x\_ && y\_;

}

Coord Coord::operator+(const Coord & \_right) const

{

Coord temp(x\_ + \_right.x\_, y\_ + \_right.y\_);

return temp;

}

Coord Coord::operator-(const Coord & \_right) const

{

Coord temp(x\_ - \_right.x\_, y\_ - \_right.y\_);

return temp;

}

bool operator>=(const Coord& \_left, const Coord& \_right)

{

return (std::pow(\_left.x\_, 2.0) + std::pow(\_left.y\_, 2.0)) / 2.0 >=

(std::pow(\_right.x\_, 2.0) + std::pow(\_right.y\_, 2.0)) / 2.0;

}

std::ostream& operator<<(std::ostream& os, const Coord& \_coord)

{

os << "X = " << \_coord.x\_ << " Y = " << \_coord.y\_;

return os;

}

std::istream& operator>>(std::istream& is, Coord& \_coord)

{

is >> \_coord.x\_ >> \_coord.y\_;

return is;

}

main.cpp

#include <iostream>

#include <iomanip>

#include "Coord.h"

int main()

{

Coord a,b;

std::cout.setf(std::ios::fixed);

std::cout.setf(std::ios::boolalpha);

std::cout.precision(3);

std::cout << "First point coord (0, 0): " << (!a) << std::endl;

std::cout << "Input first point coord:" << std::endl;

std::cin >> a;

std::cout << "Input second point coord:" << std::endl;

std::cin >> b;

std::cout << "First point coord >= then the second: " << (a >= b) << std::endl;

a++;

b--;

std::cout << "First point coord are inc." << std::endl;

std::cout << "Second point coord are dec." << std::endl;

std::cout << "First point: " << a << std::endl;

std::cout << "Second point: " << b << std::endl;

std::cout << "First point coord >= then the second: " << (a >= b) << std::endl;

std::cout << "First point coord + second: " << (a + b) << std::endl;

system("pause");

return 0;

}

4. Тестовые примеры.

На рисунке ниже приведен результат выполнения программы.

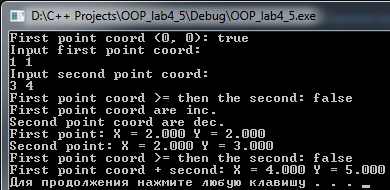


Рисунок 1 – Результаты выполнения программы.

Выводы

В ходе лабораторной работы мы было изучена возможность перегрузки различных операторов языка C++. Изучили какие операторы можно перегружать, а какие нельзя. Научились перегружать унарные и бинарные операторы.