Федеральное агентство по образованию

ФГБОУ ВО Уфимский государственный авиационный технический

университет

Кафедра ВМиК

ЛАБОРАТОРНАЯ РАБОТА №1

По дисциплине: «Объектно-Ориентированное Программирование»

«Объекты и Классы»

Выполнил: студент группы ПРО-222

Сафронов И.А.

Проверил: Котельников В.А.

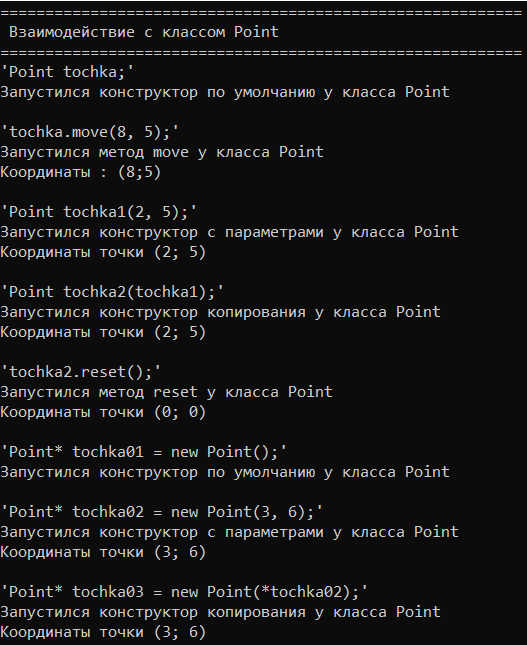
Уфа 2020

1. **Цель работы:** Определение и реализация тестовых классов, и написание программы, иллюстрирующей их использование (код, делающий нечто, и при этом обязательный отладочный вывод в консоль, для контроля того, какие методы вызываются – особенно важен отладочный вывод во всех конструкторах и деструкторах).
2. **Задачи:**

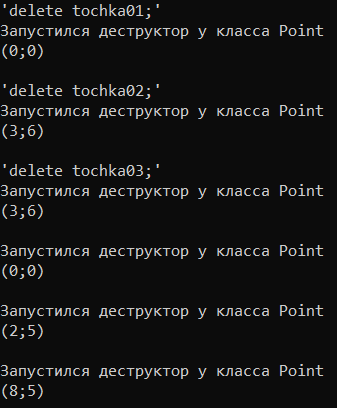
* в классе-предке есть конструктор и деструктор, а в классе-потомке нет ни того, ни того; что происходит при создании предка, при создании потомка?
* в классе-предке и в классе потомке есть и конструкторы и деструкторы: в каком порядке они вызываются при создании и удалении объекта?
* в переменную-указатель на базовый класс поместить объект-потомок: какие методы можно вызывать, какие нет?

1. **Ход** **работы**

Создание статических и динамических объектов класса Point, используя конструктор по умолчанию, конструктор с параметром, конструктор копирования. Вызов метода move и reset.

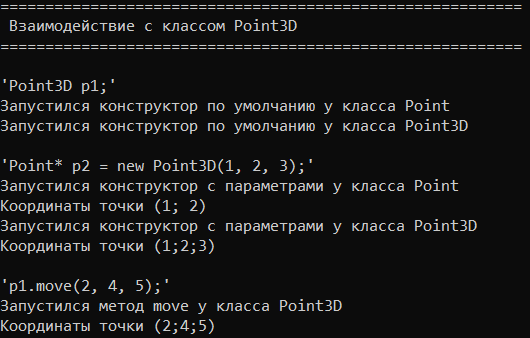


**Рис. 1 Операции с объектами, принадлежащие классу Point**

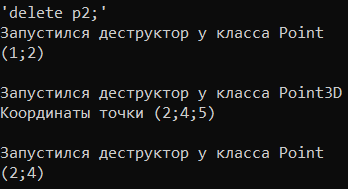


**Рис. 2 Вызов деструкторов у объектов класса Point**

Создание объектов класса-наследника Point3D использую конструктор по умолчанию и конструктора с параметром. Вызов метода move.

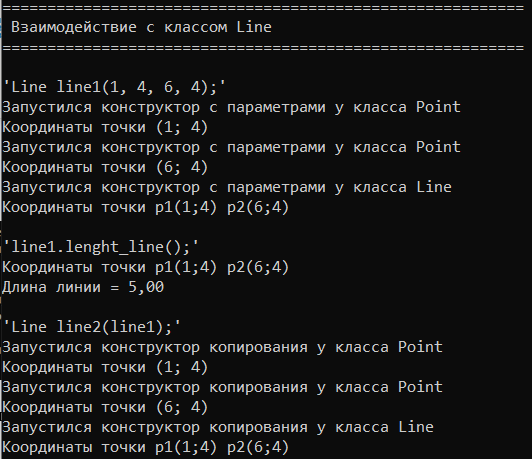


**Рис. 3 Операции с объектами класса Point3D**

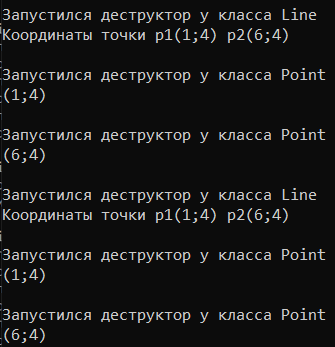


**Рис. 4 Вызов деструкторов у объектов класса Point3D**

Создание объектов класса Line. Вызов функции length\_line для нахождении длины линии.



**Рис. 5 Операции с объектами класса Line**



**Рис. 6 Вызов деструкторов у объектов класса Line**

1. **Вывод:**

В рамках лабораторной работы я разобрался:

* Что будет происходить при создании предка, при создании потомка если в классе-предке есть конструктор и деструктор, а в классе-потомке нет ни того, ни того.
* В каком порядке вызываются конструкторы и деструкторы у класса-предка и класса-потомка.
* Какие методы можно вызывать если в переменную-указатель на базовый класс поместить объект-потомок.

Также научились определять классы, атрибуты и методы, классы-наследники, конструкторов трёх различных типов, доступность атрибутов и методов; создавать и использовать статические и динамические объекты, объекты с помощью различных конструкторов, помещать объекты в переменные различных типов, объекты классов-наследников, композицию объектов, уничтожать статически и динамически созданные объекты и объекты классов-наследников

Приложение 1. Листинг программы:

#include <iostream>

using namespace std;

class Point {

public:

int x, y;

Point() {

x = 0;

y = 0;

printf("Запустился конструктор по умолчанию у класса Point\n");

}

Point(int x, int y) {

this->x = x;

this->y = y;

printf("Запустился конструктор c параметрами у класса Point\n");

printf("Координаты точки (%d; %d)\n",x , y);

}

Point(const Point& p) {

x = p.x;

y = p.y;

printf("Запустился конструктор копирования у класса Point\n");

printf("Координаты точки (%d; %d)\n", x, y);

}

void move(int mx, int my) { // Реализация метода в определении класса

x += mx;

y += my;

printf("Запустился метод move у класса Point\n");

printf("Координаты : (%d;%d)\n", x, y);

}

void reset();

~Point() {

printf("\nЗапустился деструктор у класса Point\n");

printf("(%d;%d)\n", x, y);

}

};

void Point::reset() {

x = 0;

y = 0;

printf("Запустился метод reset у класса Point\n");

printf("Координаты точки (%d; %d)\n", x, y);

}

class Point3D : public Point {

protected:

int z;

public:

Point3D(): Point() {

z = 0;

printf("Запустился конструктор по умолчанию у класса Point3D\n");

}

Point3D(int x, int y, int z) : Point(x, y) {

this->z = z;

printf("Запустился конструктор c параметрами у класса Point3D\n");

printf("Координаты точки (%d;%d;%d)\n", x, y, z);

}

Point3D(const Point3D& p3D) {

x = p3D.x;

y = p3D.y;

z = p3D.z;

printf("Запустился конструктор копирования у класса Point3D\n");

printf("Координаты точки (%d;%d;%d)\n", x, y, z);

}

void move(int mx, int my, int mz){ // Реализация метода в определении класса

x += mx;

y += my;

z += mz;

printf("Запустился метод move у класса Point3D\n");

printf("Координаты точки (%d;%d;%d)\n", x, y, z);

}

~Point3D() {

printf("\nЗапустился деструктор у класса Point3D\n");

printf("Координаты точки (%d;%d;%d)\n", x, y, z);

}

};

class Line {

public:

Point\* p1;

Point\* p2;

Line() {

p1 = new Point();

p2 = new Point();

printf("Запустился конструктор по умолчанию у класса Line\n");

printf("Координаты точки p1(%d;%d) p2(%d;%d)\n", p1->x, p1->y, p2->x, p2->y);

}

Line(int x1, int y1, int x2, int y2) {

p1 = new Point(x1, y1);

p2 = new Point(x2, y2);

printf("Запустился конструктор c параметрами у класса Line\n");

printf("Координаты точки p1(%d;%d) p2(%d;%d)\n", p1->x, p1->y, p2->x, p2->y);

}

Line(const Line& l) {

p1 = new Point(\*(l.p1));

p2 = new Point(\*(l.p2));

printf("Запустился конструктор копирования у класса Line\n");

printf("Координаты точки p1(%d;%d) p2(%d;%d)\n", p1->x, p1->y, p2->x, p2->y);

}

void lenght\_line() { // Метод для определния длины линии

int a, b;

double lenght;

a = (p1->x) - (p2->x);

b = (p1->y) - (p2->y);

lenght = sqrt(a \* a + b \* b);

printf("Координаты точки p1(%d;%d) p2(%d;%d)\n", p1->x, p1->y, p2->x, p2->y);

printf("Длина линии = %.2f\n", lenght);

}

~Line() {

printf("\nЗапустился деструктор у класса Line\n");

printf("Координаты точки p1(%d;%d) p2(%d;%d)\n", p1->x, p1->y, p2->x, p2->y);

delete p1;

delete p2;

}

};

int main() {

setlocale(0, "");

{

printf("==========================================================\n");

printf(" Взаимодействие с классом Point \n");

printf("==========================================================\n");

printf("'Point tochka;'\n");

Point tochka;

printf("\n'tochka.move(8, 5);'\n");

tochka.move(8, 5);

printf("\n'Point tochka1(2, 5);'\n");

Point tochka1(2, 5);

printf("\n'Point tochka2(tochka1);'\n");

Point tochka2(tochka1);

printf("\n'tochka2.reset();'\n");

tochka2.reset();

printf("\n'Point\* tochka01 = new Point();'\n");

Point\* tochka01 = new Point();

printf("\n'Point\* tochka02 = new Point(3, 6);'\n");

Point\* tochka02 = new Point(3, 6);

printf("\n'Point\* tochka03 = new Point(\*tochka02);'\n");

Point\* tochka03 = new Point(\*tochka02);

printf("\n'delete tochka01;'");

delete tochka01;

printf("\n'delete tochka02;'");

delete tochka02;

printf("\n'delete tochka03;'");

delete tochka03;

}

{

printf("==========================================================\n");

printf(" Взаимодействие с классом Point3D \n");

printf("==========================================================\n");

printf("\n'Point3D p1;'\n");

Point3D p1;

printf("\n'Point\* p2 = new Point3D(1, 2, 3);'\n");

Point\* p2 = new Point3D(1, 2, 3);

printf("\n'p1.move(2, 4, 5);'\n");

p1.move(2, 4, 5);

printf("\n'delete p2;'");

delete p2;

}

{

printf("==========================================================\n");

printf(" Взаимодействие с классом Line \n");

printf("==========================================================\n");

printf("\n'Line line1(1, 4, 6, 4);'\n");

Line line1(1, 4, 6, 4);

printf("\n'line1.lenght\_line();'\n");

line1.lenght\_line();

printf("\n'Line line2(line1);'\n");

Line line2(line1);

}

}