Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Уфимский государственный авиационный технический университет»

Факультет информатики и робототехники

Кафедра вычислительной математики и кибернетики

Отчет по лабораторной работе №4.1

по дисциплине «Объектно-ориентированное программирование»

на тему: «Круги на форме»

Вариант 1

Выполнил:

Студент группы ПРО-228Б И.И.Гареев

Проверил: В.А. Котельников

Уфа – 2022

Содержание

[1 Цель работы 3](#_Toc91276456)

[2 Программная часть 4](#_Toc91276457)

[2.1 Статическое создание объектов и их удаление 4](#_Toc91276458)

[2.2 Динамическое создание объектов и их удаление 4](#_Toc91276459)

[2.3 Изменение координат объектов с заданными координатами 5](#_Toc91276460)

[2.4 Наследование 5](#_Toc91276461)

[2.5 Помещение объектов в переменные различных типов 6](#_Toc91276462)

[2.6 Композиция 6](#_Toc91276463)

[3 Заключение 7](#_Toc91276464)

[4 Приложение А 8](#_Toc91276465)

# Цель работы

Создать простейшее приложение с GUI, содержащее:

1. Определение простейшего класса CCircle с координатами и постоянным радиусом;
2. Хранилище из Л.Р.3 для хранения объектов класса CCircle;
3. Форму с объектом для рисования.

# Программная часть

## Создание базового класса

Создаем базовый класс Figure с виртуальными методами DrawEllipse, DoCheckedTrue, DoCheckedFalse и переменными типа bool isClicked, isChecked.

## Создание класса CCircle

Создаем класс CCircle потомок класса Figure. В нем определены конструктор с параметрами, переменная типа bool isClicked, в которой отлавливаем координаты мыши для выделения круга. Также в этом классе есть метод DrawEllipse, в котором рисуем круг, методы DoCheckedTrue и DoCheckedFalse и переменная типа bool isChecked, которая возвращает Checked.

## Создание хранилища

Создаю хранилище как в Л.Р.3. Оно содержит в себе конструктор без параметров, конструктор с параметрами, метод SetObject и AddObject для добавления объектов, метод DeleteObject для удаления объектов и DeleteCheckObject для удаления выделенных объектов. Также в хранилище есть переменная типа bool isCheckedStorage, в которой я проверяю нажатие на круг, и если true, то вызываю у объекта метод DoCheckedTrue, метод NotChecked, который снимает выделение с объектов и метод Draw, который вызывает у объектов метод DrawEllipse.

## Событие panel1\_MouseClick

При нажатии на panel1 проверяется условие, если нажато на пустое место, то круг добавляется в хранилище, иначе, если нажат ctrl круг становится выделенным, в противном случае выделенным становится только один круг.

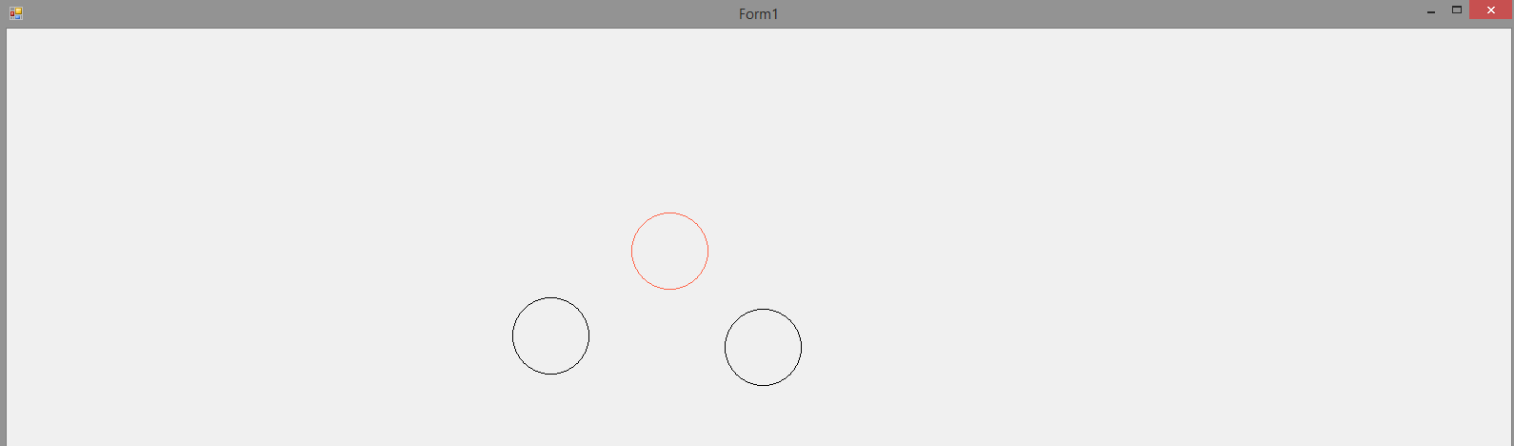


Рисунок 2.1

На Рисунок 2.1 видно, что при нажатии на круг, только он становится выделенным.

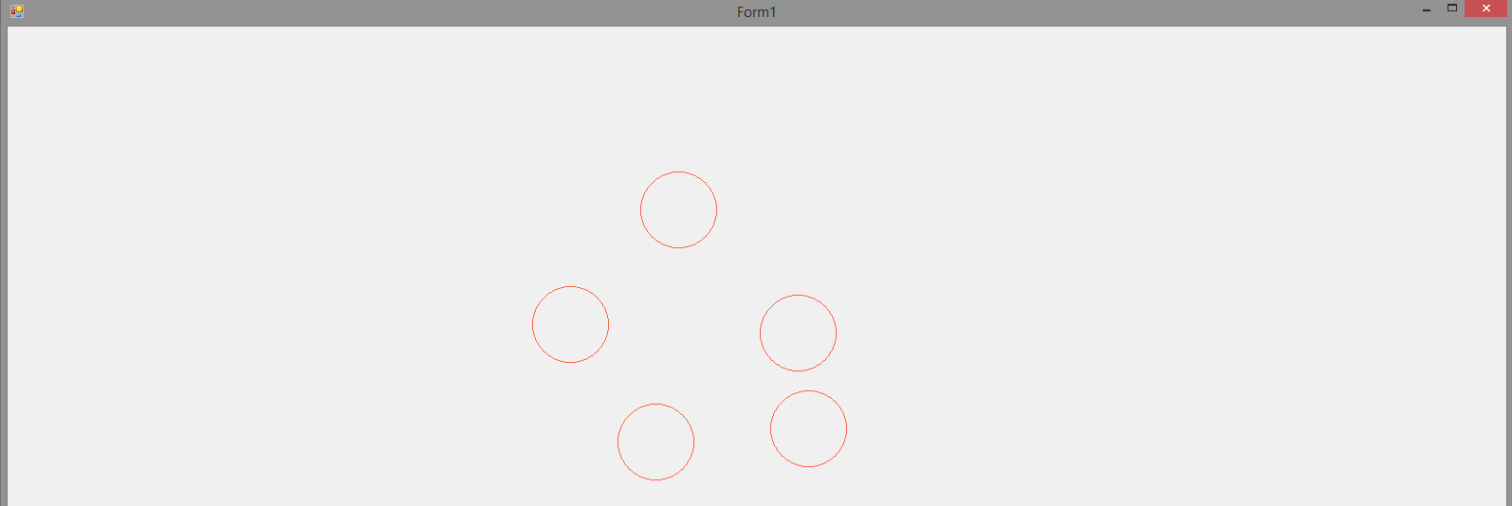


Рисунок 2.2

На Рисунок 2.2 видно, что при зажатии клавиши ctrl, можно выделять сразу несколько кругов.

## Событие panel1\_Paint

С помощью Graphics рисую круг.

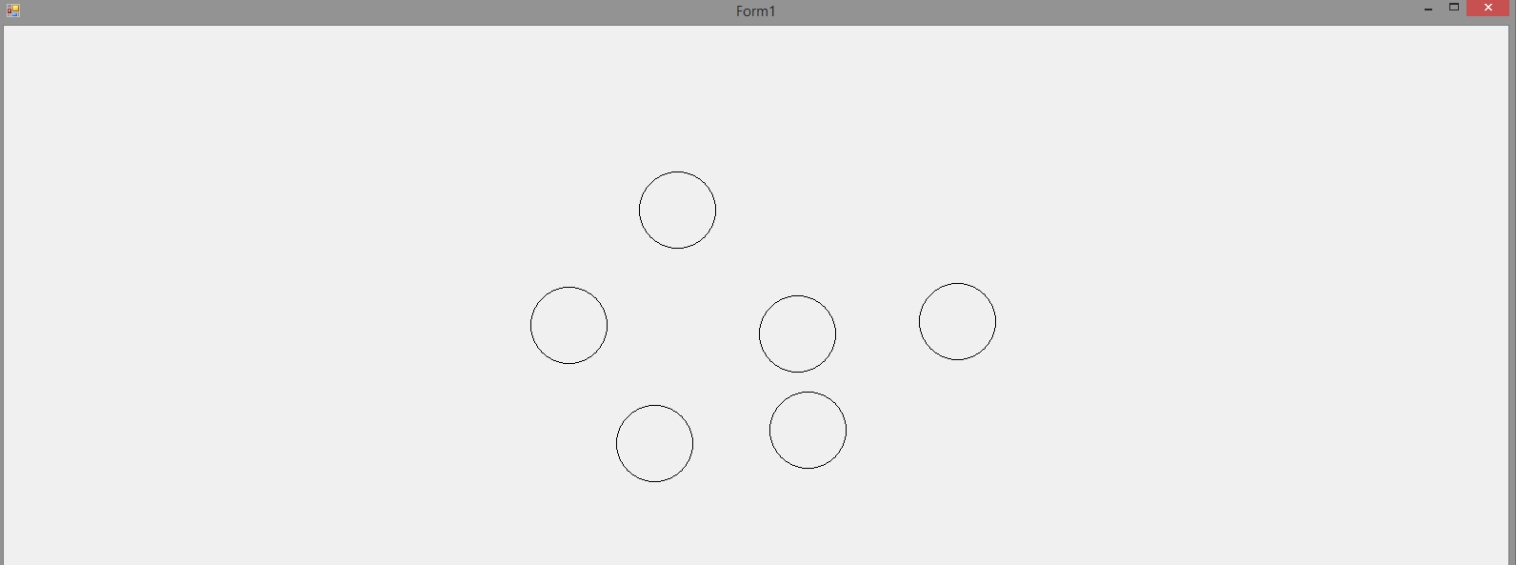


Рисунок 2.3

На Рисунок 2.3 показано рисование круга.

## Cобытие Form1\_KetDown

При нажатии на кнопку Delete выделенные круги удаляются.

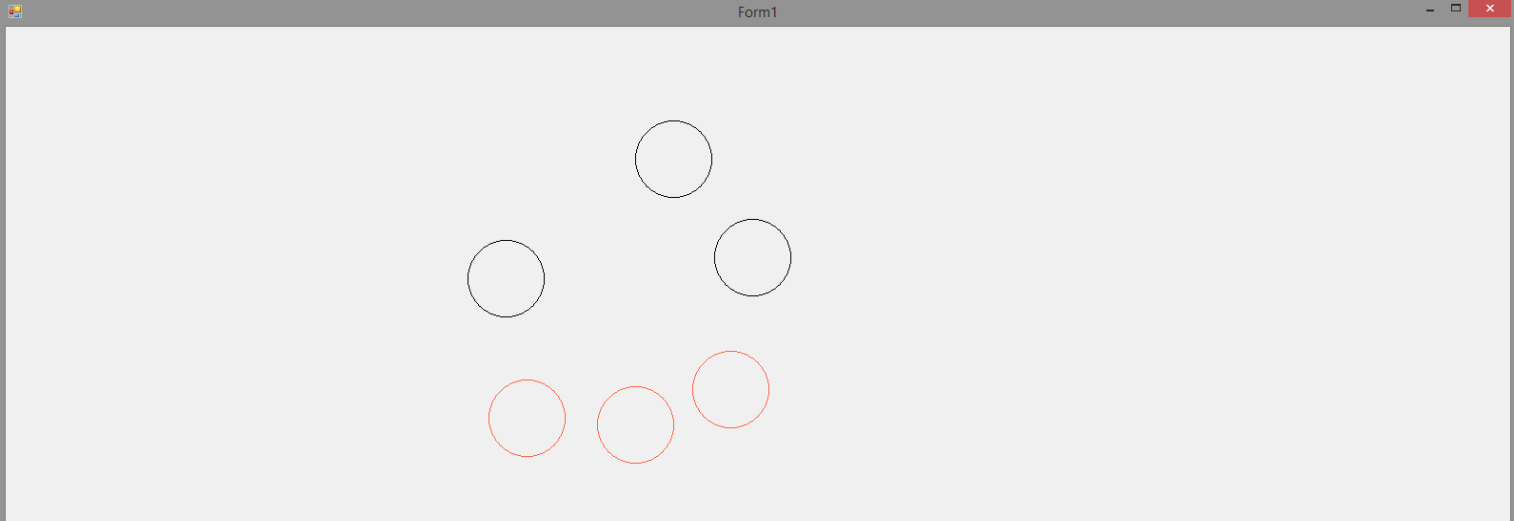


Рисунок 2.4

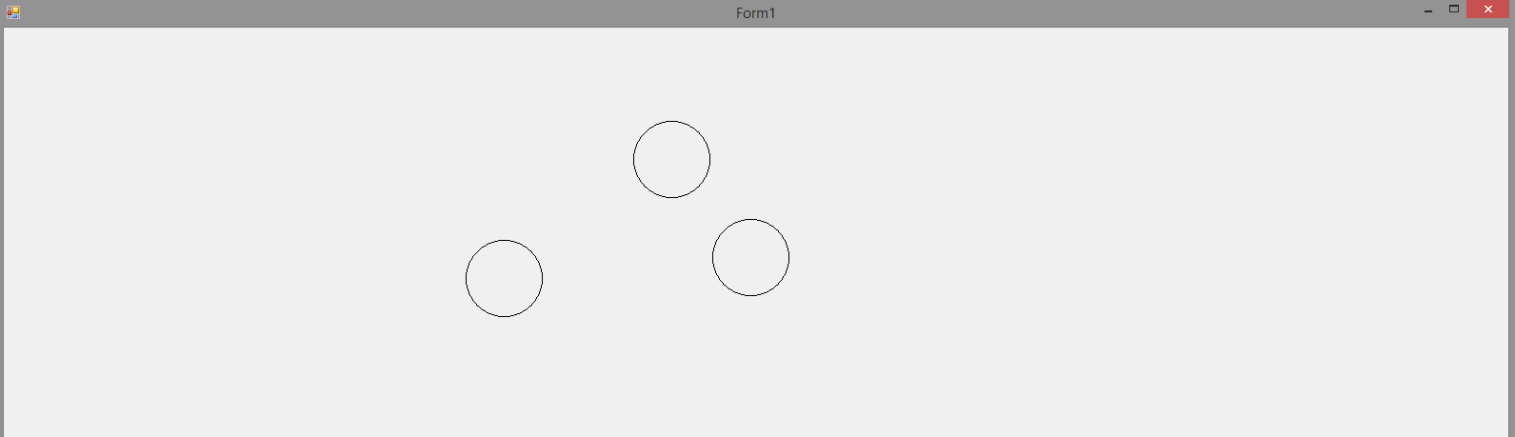


Рисунок 2.5

На Рисунок 2.4 показаны выделенные круги, а на Рисунок 2.5 видно, что при нажатии на кнопку Delete выделенные круги удаляются.

# Заключение

В ходе данной лабораторной работы я научился определять и реализовывать тестовые классы, иллюстрировать их использование, создавать и уничтожать обьекты классов.

<https://github.com/IlyasGareev/LLR2>

# Приложение А

#include <iostream>

#include<conio.h>

class Point {

protected:

int x, y;

public:

Point() { //конструктор

printf("Point()\n"); //отладочный вывод

x = 0;

y = 0;

}

Point(int x, int y) { //конструктор с параметрами

printf("Point(int x, int y)\n");

this->x = x;

this->y = y;

}

Point(const Point& p) { //конструктор копирования

printf("Point(const Point &p)\n");

x = p.x;

y = p.y;

}

virtual ~Point() { //деструктор

printf("%d, %d\n", x, y); //вывод текущего состояния объекта

printf("~Point()\n");

}

void move(int dx, int dy) {

x = x + dx;

y = y + dy;

}

void reset();

};

void Point::reset() {

x = 0;

y = 0;

}

class ColoredPoint : public Point { //класс потомок

protected:

int color;

public:

ColoredPoint() : Point() {

printf("ColoredPoint()\n");

color = 0;

}

ColoredPoint(int x, int y, int color) : Point(x, y) {

printf("ColoredPoint(int x, int y)\n");

this->color = color;

}

ColoredPoint(const ColoredPoint& p) {

printf("ColoredPoint(const Point &p)\n");

x = p.x;

y = p.y;

color = p.color;

}

~ColoredPoint() {

printf("%d, %d color=%d\n", x, y, color); //вывод текущего состояния объекта

printf("~ColoredPoint()\n");

}

void ChangeColor(int NewColor) {

color = NewColor;

}

};

class Section {

protected:

Point\* p1;

Point\* p2;

public:

Section() { //конструктор

printf("Section()\n"); //отладочный вывод

p1 = new Point;

p2 = new Point;

}

Section(int x1, int y1, int x2, int y2) { //конструктор с параметрами

printf("Section(int x1, int y1, int x2, int y2)\n");

p1 = new Point(x1, y1);

p2 = new Point(x2, y2);

}

Section(const Section& s) { //конструктор копирования

printf("Section(const Section &s)\n");

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

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

}

~Section() { //деструктор

delete p1;

delete p2;

printf("~Section()\n");

}

};

int main()

{

setlocale(LC\_ALL, "rus");

{

printf("Статическое создание объектов и их удаление\n");

Point p;

Point p2(1, 5);

Point p3(p2);

}

printf("--------------------------------------------------------------\n");

{

printf("Динамическое создание объектов и их удаление\n");

Point\* p = new Point;

Point\* p2 = new Point(7, 7);

Point\* p3 = new Point(\*p2);

delete p;

delete p2;

delete p3;

}

printf("--------------------------------------------------------------\n");

{

printf("Изменение координат объектов с заданными координатами\n");

Point\* p = new Point(7, 17);

Point\* p2 = new Point(4, 4);

Point\* p3 = new Point(\*p2);

p->move(3, 3);

p2->move(1, 1);

p3->reset();

p3->move(2, 2);

delete p;

delete p2;

delete p3;

}

printf("--------------------------------------------------------------\n");

{

printf("Объекты класса наследника\n");

ColoredPoint\* p = new ColoredPoint(3, 4, 87);

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

ColoredPoint\* p3 = new ColoredPoint(\*p2);

delete p;

delete p2;

delete p3;

}

printf("--------------------------------------------------------------\n");

{

printf("Помещение объектов в переменные различных типов\n");

Point\* p1 = new ColoredPoint(3, 4, 87);

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

p1->reset();

p2->ChangeColor(4);

delete p1;

delete p2;

}

printf("--------------------------------------------------------------\n");

{

printf("Композиция\n");

Section\* s1 = new Section(1, 2, 3, 4);

Section\* s2 = new Section(\*s1);

delete s1;

delete s2;

}

\_getch();

}