Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Уфимский государственный авиационный технический университет»

Факультет информатики и робототехники

Кафедра Вычислительной математики и кибернетики

**Лабораторная работа №4 Часть 1**

По дисциплине «Объектно-ориентированное программирование»

«Круги на форме»

Выполнил: студент группы ПРО-222

Вальшин Руслан Ильдусович

УФА 2020

1. **Цель работы:**

Создать простейшее приложение с GUI, содержащее:

* определение простейшего класса CCircle с координатами и постоянным радиусом;
* хранилище из Л.Р.3 для хранения объектов класса CCircle;
* форму с объектом для рисования (например, PaintBox).

1. **Задачи:**

Реализовать следующее поведение:

* при нажатии мышкой на форме создается новый объект CCircle с координатами нажатия и помещается в хранилище;
* при событии Paint должны отрисовываться на форме все объекты из хранилища
* при создании объекта он становится единственным выделенным
* при нажатии ЛКМ (левой клавиши мыши) на какой-либо объект CCircle на форме, он становится «выделенным» и отрисовывается отлично от других объектов
* при нажатии на кнопку Del, все выделенные объекты должны удаляться
* при выделении объекта с помощью ЛКМ и удерживаемой клавиши Ctrl, выделенными становятся несколько объектов

1. **Ход работы:**

Определение класса CCirlce. В классе есть конструктор по умолчанию, конструктор с тремя параметрами и конструктор копирования. У объекта этого класса есть параметры x, y и radius.

        public class CCircle

        {

            public int x;

            public int y;

            public int radius;

            public CCircle()

            {

                x = 0;

                y = 0;

                radius = 0;

            }

            public CCircle(int x, int y, int radius)

            {

                this.x = x;

                this.y = y;

                this.radius = radius;

            }

            public CCircle(CCircle circle)

            {

                this.x = circle.x;

                this.y = circle.y;

                this.radius = circle.radius;

            }

        }

Портирование класса Storage из ЛР№3 c C++ на C#

public class Storage<T>

        {

            private class Node

            {

                public T obj;

                public Node previous;

                public Node next;

            }

            private int size;

            private Node first;

            private Node last;

            private Node current;

            public Storage()

            {

                size = 0;

            }

            public void add(T obj) // Добавляет объект в хранилище в конец списка

            {

                Node temp = new Node();

                temp.obj = obj;

                size++;

                if (first == null)

                {

                    first = temp;

                    last = temp;

                    current = temp;

                }

                else

                {

                    last.next = temp;

                    temp.previous = last;

                    last = temp;

                }

            }

            public void del() // Удаляет текущий элемент

            {

                if (current != null)

                {

                    // Переназначение "указателей" соседних элементов

                    if (current.previous != null)

                        current.previous.next = current.next;

                    if (current.next != null)

                        current.next.previous = current.previous;

                    // Перевод current на следующий или предыдущий элемент

                    Node oldCurrent = current;

                    if (current.next != null)

                        current = current.next;

                    else if (current.previous != null)

                        current = current.previous;

                    else

                        current = null;

                    // Смена "указателей" first и last, если current был им равен

                    if (oldCurrent == first)

                        first = current;

                    if (oldCurrent == last)

                        last = current;

                    // Коррекция размера списка

                    size--;

                }

            }

            public void previous() // Переносит current на предыдущий элемент в списке, если предыдущий элемент существует

            {

                if (current != null)

                    if (current.previous != null)

                        current = current.previous;

            }

            public void next() // Переносит current на следующий элемент в списке, если следующий элемент существует

            {

                if (current != null)

                    if (current.next != null)

                        current = current.next;

            }

            public bool check(T obj) // Проверяет наличие объекта хранилище, не изменяя current

            {

                Node buffer = first;

                for (int i = 0; i < size; i++, buffer = buffer.next)

                    if (buffer.obj.Equals(obj))

                        return true;

                return false;

            }

            public bool checkAndSetCurrent(T obj) // Проверяет наличие объекта хранилище и устанавливает current на этот объект

            {

                Node buffer = first;

                for (int i = 0; i < size; i++, buffer = buffer.next)

                    if (buffer.obj.Equals(obj))

                    {

                        current = buffer;

                        return true;

                    }

                return false;

            }

            public int getSize()

            {

                return size;

            }

            public T getFirst() // Возвращает первый объект в списке

            {

                return first.obj;

            }

            public T getLast() // Возвращает последний объект в списке

            {

                return last.obj;

            }

            public T getCurrent() // Возвращает текущий объект

            {

                return current.obj;

            }

            public void setFirst() // Устанавливает current на начало списка

            {

                current = first;

            }

            public void setLast() // Устанавливает current на конец списка

            {

                current = last;

            }

            public bool eol() // End Of List

            {

                if (current.next == null)

                    return true;

                else

                    return false;

            }

        };

Объявление глобальных переменных

bool ctrlIsPressed; // Флаг для проверки нажатия кнопки Ctrl

        Storage<CCircle> storage; // Хранилище всех объектов

        Storage<CCircle> selectedStorage; // Хранилище выбранных объектов

        int radius = 15;

        bool inTheCircle; // Флаг для проверки нажатия на круг

        Graphics g;

        Bitmap image;

Обработка нажатий кнопок Ctrl и Del

        private void Form1\_KeyDown(object sender, KeyEventArgs e)

        {

            ctrlIsPressed = e.Control; // Если нажали кнопку Ctrl устанавливаем флаг

            if (e.KeyCode == Keys.Delete) // Если нажали кнопку Del

                deleteSelected(); // Удаляем выбранные круги

        }

        private void Form1\_KeyUp(object sender, KeyEventArgs e)

        {

            ctrlIsPressed = e.Control; // Если кнопку Ctrl отпустили, изменяем флаг на false

        }

Обработка нажатия ЛКМ

        private void pictureBox\_MouseUp(object sender, MouseEventArgs e)

        {

            if (storage == null) // Если глобальные объекты не инициализированы, инициализируем их

            {

                storage = new Storage<CCircle>();

                selectedStorage = new Storage<CCircle>();

                image = new Bitmap(1920, 1080);

                g = Graphics.FromImage(image);

                g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;

            }

            if (e.Button == MouseButtons.Left && ctrlIsPressed) // Если одновременно нажали на ЛКМ и Ctrl

            {

                CCircle circle = inTheAreaOfCircle(e.X, e.Y); // Проверяем, попали ли мы в окрестность круга

                if (circle != null) // Если попали

                {

                    if (inTheCircle) // Проверяем, попали ли мы ровно на круг

                        if (selectedStorage.checkAndSetCurrent(circle) == true) // Если в хранилище выделенных объектов есть выбранный

                            deselectOne(); // Убираем его из списка выбранных

                        else

                        {

                            // Иначе добавляем его в хранилище выбранных и выводим на экран

                            selectedStorage.add(circle);

                            printSelectedCircle(circle.x, circle.y, circle.radius);

                        }

                    // Если попали в пустую область рядом с кругом, ничего не делаем

                }

            }

            else if(e.Button == MouseButtons.Left) // Если просто нажали на ЛКМ

            {

                deselectAll(); // Убираем все элементы из списка выбранных

                CCircle circle = inTheAreaOfCircle(e.X, e.Y); // Проверяем, попали ли мы в окрестность круга

                if (circle == null) // Если не попали

                {

                    circle = new CCircle(e.X, e.Y, radius); // Создадим объект

                    storage.add(circle); // Добавим его в хранилище

                    selectedStorage.add(circle); // Добавим его в хранилище выбранных

                    printSelectedCircle(circle.x, circle.y, circle.radius); // И выведем на экран уже выбранным

                }

                else if (inTheCircle) // Если попали ровно по кругу

                {

                    // Добавляем его в хранилище выбранных и выводим на экран

                    selectedStorage.add(circle);

                    printSelectedCircle(circle.x, circle.y, circle.radius);

                }

                // Если попали в область рядом с кругом, ничего не делаем

            }

        }

Функции визуального выбора и снятия выбора круга

        private void printSelectedCircle(int x, int y, int radius) // Рисуем круг выбранным

        {

            g.FillEllipse(Brushes.White, x - radius - 1, y - radius - 1, 2 \* (radius + 1), 2 \* (radius + 1));

            g.FillEllipse(Brushes.LightSkyBlue, x - radius - 1, y - radius - 1, 2 \* (radius + 1), 2 \* (radius + 1));

        }

        private void deselectPrintedCircle(int x, int y, int radius) // Риусем обычный круг

        {

            g.FillEllipse(Brushes.White, x - radius - 2, y - radius - 2, 2 \* (radius + 2), 2 \* (radius + 2));

            g.FillEllipse(Brushes.SteelBlue, x - radius, y - radius, 2 \* radius, 2 \* radius);

        }

Функция проверки нажатия на круг или область рядом с кругом

private CCircle inTheAreaOfCircle(int X, int Y) // Проверяем нажатие на точку в пределах 2R круга

        {

            storage.setFirst();

            for (int i = 0; i < storage.getSize(); i++, storage.next())

            {

                int x = storage.getCurrent().x;

                int y = storage.getCurrent().y;

                int radius = storage.getCurrent().radius;

                int temp = (x - X) \* (x - X) + (y - Y) \* (y - Y);

                if (temp <= 4 \* radius \* radius) // Если нажали на точку в пределах 2R круга

                {

                    inTheCircle = (temp <= radius \* radius); // Если нажали на точку в пределах круга, устанавливаем флаг

                    return storage.getCurrent();  // Выводим найденный объект круга

                }

            }

            return null; // Если точка не принадлежит ни одному кругу, выводим нулевой указатель

        }

Функция отмены выбора одного элемента

        private void deselectOne() // Убираем один элемент из списка выбранных

        {

            CCircle circle = selectedStorage.getCurrent();

            deselectPrintedCircle(circle.x, circle.y, circle.radius);

            selectedStorage.del();

        }

Функция отмены выбора всех элементов

        private void deselectAll() // Убираем все элементы из списка выбранных

        {

            selectedStorage.setFirst();

            for (int i = 0; i < selectedStorage.getSize(); i++, selectedStorage.next())

            {

                CCircle circle = selectedStorage.getCurrent();

                deselectPrintedCircle(circle.x, circle.y, circle.radius);

            }

            selectedStorage = new Storage<CCircle>();

        }

Функция удаления выбранных элементов из хранилища и экрана

        private void deleteSelected() // Удаляем выбранные элементы

        {

            selectedStorage.setFirst();

            for (int i = 0; i < selectedStorage.getSize(); i++, selectedStorage.next())

            {

                CCircle circle = selectedStorage.getCurrent();

                storage.setFirst();

                for (int j = 0; j < storage.getSize(); j++, storage.next())

                    if (circle.Equals(storage.getCurrent()))

                    {

                        storage.del();

                        break;

                    }

                g.FillEllipse(Brushes.White, circle.x - radius - 2, circle.y - radius - 2, 2 \* (circle.radius + 2), 2 \* (circle.radius + 2));

            }

            selectedStorage = new Storage<CCircle>();

        }

Функция перерисовки картинки

        private void pictureBox\_Paint(object sender, PaintEventArgs e) // Рисуем картинку

        {

            pictureBox.Image = image;

        }

Приложение А. Ссылка на GitHub репозиторий и скриншот истории коммитов.

<https://github.com/Ruslan376M/OOP_Lab4>

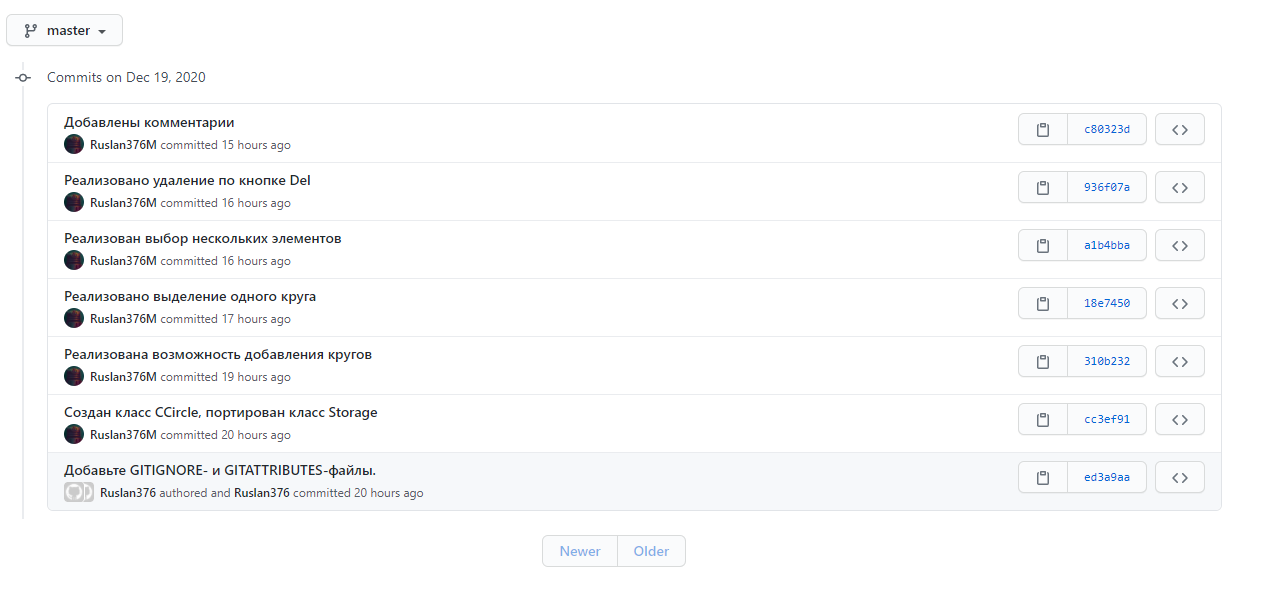


Рисунок 1 - Скриншот истории коммитов

Приложение B. Примеры работы программы.

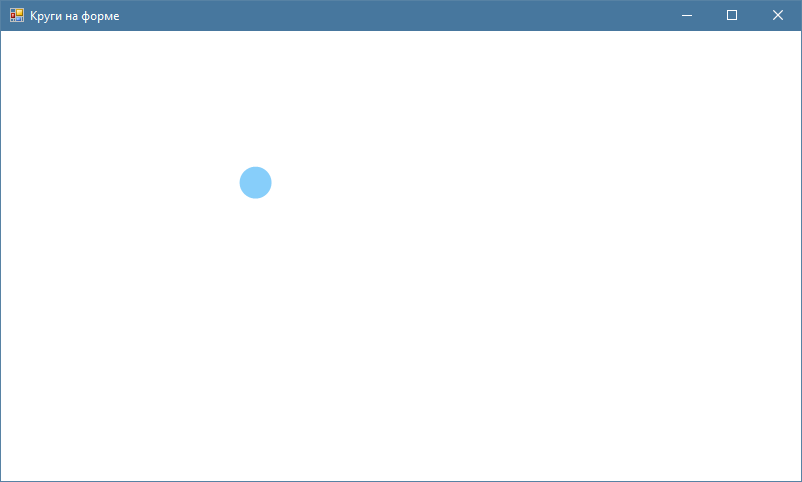


Рисунок 2 - Пример работы программы

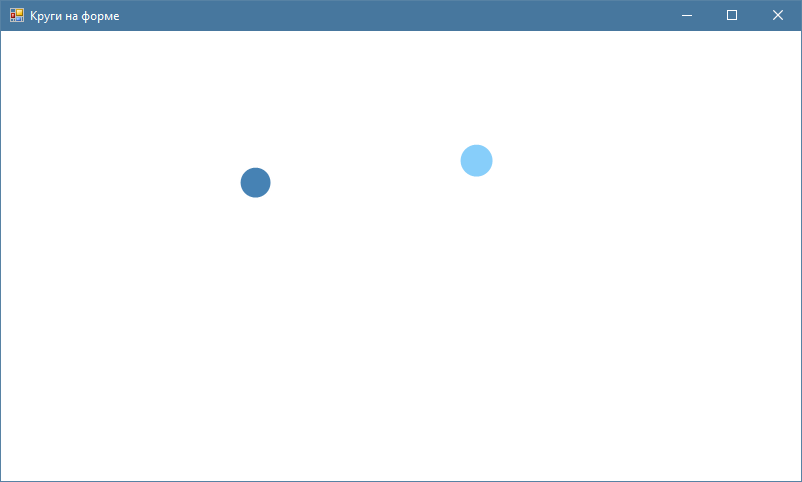


Рисунок 3 - Пример работы программы

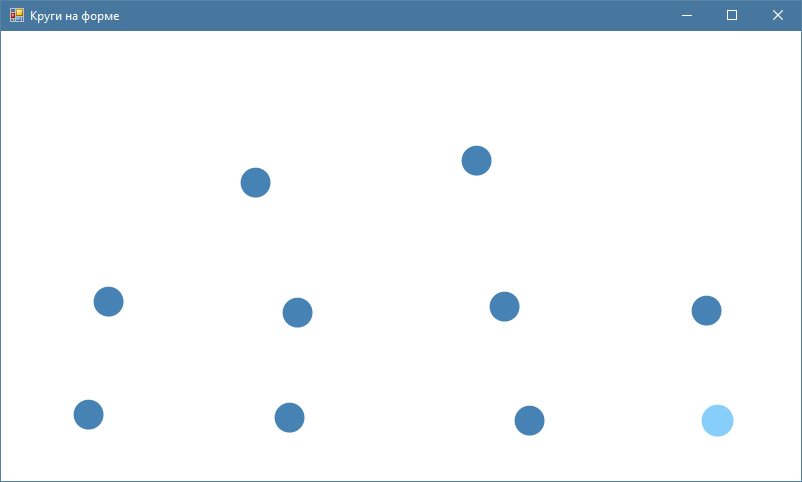


Рисунок 4 - Пример работы программы

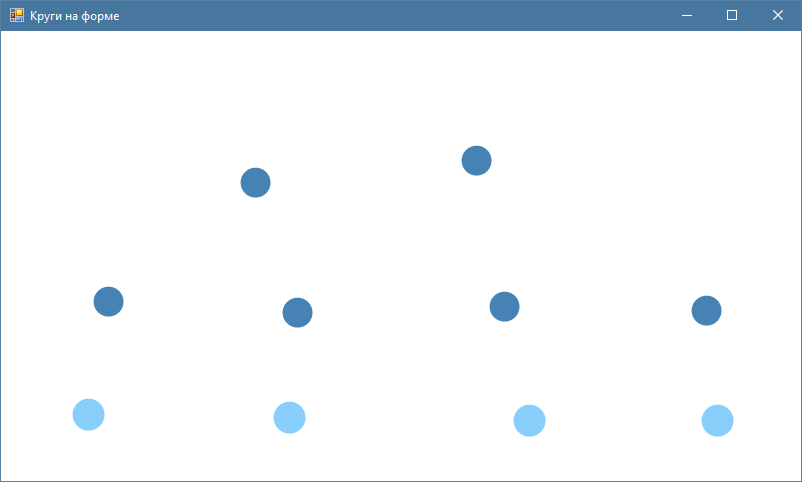


Рисунок 5 - Пример работы программы

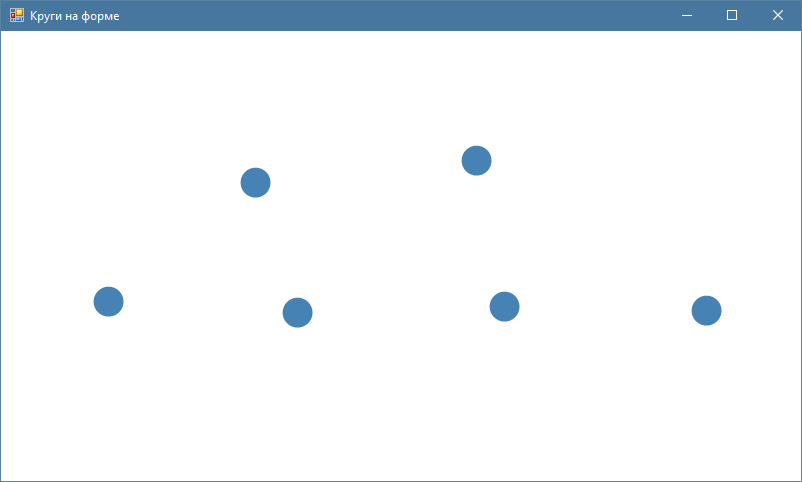


Рисунок 6 - Пример работы программы

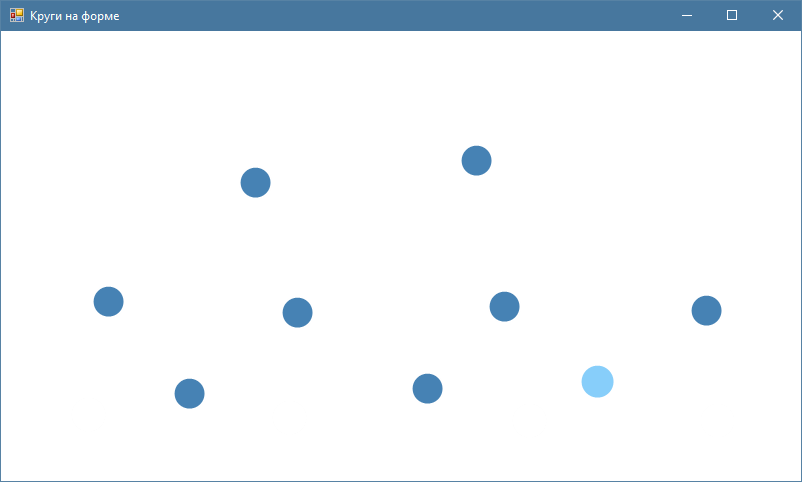


Рисунок 7 - Пример работы программы

Приложение C. Исходный код программы.

using System.Drawing;

using System.Windows.Forms;

namespace Лабораторная\_работа\_\_4

{

    public partial class Form1 : Form

    {

        public Form1()

        {

            InitializeComponent();

            this.DoubleBuffered = true;

            SetStyle(ControlStyles.AllPaintingInWmPaint | ControlStyles.UserPaint, true);

            SetStyle(ControlStyles.OptimizedDoubleBuffer, true);

        }

        public class CCircle

        {

            public int x;

            public int y;

            public int radius;

            public CCircle()

            {

                x = 0;

                y = 0;

                radius = 0;

            }

            public CCircle(int x, int y, int radius)

            {

                this.x = x;

                this.y = y;

                this.radius = radius;

            }

            public CCircle(CCircle circle)

            {

                this.x = circle.x;

                this.y = circle.y;

                this.radius = circle.radius;

            }

        }

        public class Storage<T>

        {

            private class Node

            {

                public T obj;

                public Node previous;

                public Node next;

            }

            private int size;

            private Node first;

            private Node last;

            private Node current;

            public Storage()

            {

                size = 0;

            }

            public void add(T obj) // Добавляет объект в хранилище в конец списка

            {

                Node temp = new Node();

                temp.obj = obj;

                size++;

                if (first == null)

                {

                    first = temp;

                    last = temp;

                    current = temp;

                }

                else

                {

                    last.next = temp;

                    temp.previous = last;

                    last = temp;

                }

            }

            public void del() // Удаляет текущий элемент

            {

                if (current != null)

                {

                    // Переназначение "указателей" соседних элементов

                    if (current.previous != null)

                        current.previous.next = current.next;

                    if (current.next != null)

                        current.next.previous = current.previous;

                    // Перевод current на следующий или предыдущий элемент

                    Node oldCurrent = current;

                    if (current.next != null)

                        current = current.next;

                    else if (current.previous != null)

                        current = current.previous;

                    else

                        current = null;

                    // Смена "указателей" first и last, если current был им равен

                    if (oldCurrent == first)

                        first = current;

                    if (oldCurrent == last)

                        last = current;

                    // Коррекция размера списка

                    size--;

                }

            }

            public void previous() // Переносит current на предыдущий элемент в списке, если предыдущий элемент существует

            {

                if (current != null)

                    if (current.previous != null)

                        current = current.previous;

            }

            public void next() // Переносит current на следующий элемент в списке, если следующий элемент существует

            {

                if (current != null)

                    if (current.next != null)

                        current = current.next;

            }

            public bool check(T obj) // Проверяет наличие объекта хранилище, не изменяя current

            {

                Node buffer = first;

                for (int i = 0; i < size; i++, buffer = buffer.next)

                    if (buffer.obj.Equals(obj))

                        return true;

                return false;

            }

            public bool checkAndSetCurrent(T obj) // Проверяет наличие объекта хранилище и устанавливает current на этот объект

            {

                Node buffer = first;

                for (int i = 0; i < size; i++, buffer = buffer.next)

                    if (buffer.obj.Equals(obj))

                    {

                        current = buffer;

                        return true;

                    }

                return false;

            }

            public int getSize()

            {

                return size;

            }

            public T getFirst() // Возвращает первый объект в списке

            {

                return first.obj;

            }

            public T getLast() // Возвращает последний объект в списке

            {

                return last.obj;

            }

            public T getCurrent() // Возвращает текущий объект

            {

                return current.obj;

            }

            public void setFirst() // Устанавливает current на начало списка

            {

                current = first;

            }

            public void setLast() // Устанавливает current на конец списка

            {

                current = last;

            }

            public bool eol() // End Of List

            {

                if (current.next == null)

                    return true;

                else

                    return false;

            }

        };

        bool ctrlIsPressed; // Флаг для проверки нажатия кнопки Ctrl

        Storage<CCircle> storage; // Хранилище всех объектов

        Storage<CCircle> selectedStorage; // Хранилище выбранных объектов

        int radius = 15;

        bool inTheCircle; // Флаг для проверки нажатия на круг

        Graphics g;

        Bitmap image;

        Pen circlePen = new Pen(Brushes.Red, 5);

        private void Form1\_KeyDown(object sender, KeyEventArgs e)

        {

            ctrlIsPressed = e.Control; // Если нажали кнопку Ctrl устанавливаем флаг

            if (e.KeyCode == Keys.Delete) // Если нажали кнопку Del

                deleteSelected(); // Удаляем выбранные круги

        }

        private void Form1\_KeyUp(object sender, KeyEventArgs e)

        {

            ctrlIsPressed = e.Control; // Если кнопку Ctrl отпустили, изменяем флаг на false

        }

        private void pictureBox\_MouseUp(object sender, MouseEventArgs e)

        {

            if (storage == null) // Если глобальные объекты не инициализированы, инициализируем их

            {

                storage = new Storage<CCircle>();

                selectedStorage = new Storage<CCircle>();

                image = new Bitmap(1920, 1080);

                g = Graphics.FromImage(image);

                g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;

            }

            if (e.Button == MouseButtons.Left && ctrlIsPressed) // Если одновременно нажали на ЛКМ и Ctrl

            {

                CCircle circle = inTheAreaOfCircle(e.X, e.Y); // Проверяем, попали ли мы в окрестность круга

                if (circle != null) // Если попали

                {

                    if (inTheCircle) // Проверяем, попали ли мы ровно на круг

                        if (selectedStorage.checkAndSetCurrent(circle) == true) // Если в хранилище выделенных объектов есть выбранный

                            deselectOne(); // Убираем его из списка выбранных

                        else

                        {

                            // Иначе добавляем его в хранилище выбранных и выводим на экран

                            selectedStorage.add(circle);

                            printSelectedCircle(circle.x, circle.y, circle.radius);

                        }

                    // Если попали в пустую область рядом с кругом, ничего не делаем

                }

            }

            else if(e.Button == MouseButtons.Left) // Если просто нажали на ЛКМ

            {

                deselectAll(); // Убираем все элементы из списка выбранных

                CCircle circle = inTheAreaOfCircle(e.X, e.Y); // Проверяем, попали ли мы в окрестность круга

                if (circle == null) // Если не попали

                {

                    circle = new CCircle(e.X, e.Y, radius); // Создадим объект

                    storage.add(circle); // Добавим его в хранилище

                    selectedStorage.add(circle); // Добавим его в хранилище выбранных

                    printSelectedCircle(circle.x, circle.y, circle.radius); // И выведем на экран уже выбранным

                }

                else if (inTheCircle) // Если попали ровно по кругу

                {

                    // Добавляем его в хранилище выбранных и выводим на экран

                    selectedStorage.add(circle);

                    printSelectedCircle(circle.x, circle.y, circle.radius);

                }

                // Если попали в область рядом с кругом, ничего не делаем

            }

        }

        private void printSelectedCircle(int x, int y, int radius) // Рисуем круг выбранным

        {

            g.FillEllipse(Brushes.White, x - radius - 1, y - radius - 1, 2 \* (radius + 1), 2 \* (radius + 1));

            g.FillEllipse(Brushes.LightSkyBlue, x - radius - 1, y - radius - 1, 2 \* (radius + 1), 2 \* (radius + 1));

        }

        private void deselectPrintedCircle(int x, int y, int radius) // Риусем обычный круг

        {

            g.FillEllipse(Brushes.White, x - radius - 2, y - radius - 2, 2 \* (radius + 2), 2 \* (radius + 2));

            g.FillEllipse(Brushes.SteelBlue, x - radius, y - radius, 2 \* radius, 2 \* radius);

        }

        private CCircle inTheAreaOfCircle(int X, int Y) // Проверяем нажатие на точку в пределах 2R круга

        {

            storage.setFirst();

            for (int i = 0; i < storage.getSize(); i++, storage.next())

            {

                int x = storage.getCurrent().x;

                int y = storage.getCurrent().y;

                int radius = storage.getCurrent().radius;

                int temp = (x - X) \* (x - X) + (y - Y) \* (y - Y);

                if (temp <= 4 \* radius \* radius) // Если нажали на точку в пределах 2R круга

                {

                    inTheCircle = (temp <= radius \* radius); // Если нажали на точку в пределах круга, устанавливаем флаг

                    return storage.getCurrent();  // Выводим найденный объект круга

                }

            }

            return null; // Если точка не принадлежит ни одному кругу, выводим нулевой указатель

        }

        private void deselectOne() // Убираем один элемент из списка выбранных

        {

            CCircle circle = selectedStorage.getCurrent();

            deselectPrintedCircle(circle.x, circle.y, circle.radius);

            selectedStorage.del();

        }

        private void deselectAll() // Убираем все элементы из списка выбранных

        {

            selectedStorage.setFirst();

            for (int i = 0; i < selectedStorage.getSize(); i++, selectedStorage.next())

            {

                CCircle circle = selectedStorage.getCurrent();

                deselectPrintedCircle(circle.x, circle.y, circle.radius);

            }

            selectedStorage = new Storage<CCircle>();

        }

        private void deleteSelected() // Удаляем выбранные элементы

        {

            selectedStorage.setFirst();

            for (int i = 0; i < selectedStorage.getSize(); i++, selectedStorage.next())

            {

                CCircle circle = selectedStorage.getCurrent();

                storage.setFirst();

                for (int j = 0; j < storage.getSize(); j++, storage.next())

                    if (circle.Equals(storage.getCurrent()))

                    {

                        storage.del();

                        break;

                    }

                g.FillEllipse(Brushes.White, circle.x - radius - 2, circle.y - radius - 2, 2 \* (circle.radius + 2), 2 \* (circle.radius + 2));

            }

            selectedStorage = new Storage<CCircle>();

        }

        private void pictureBox\_Paint(object sender, PaintEventArgs e) // Рисуем картинку

        {

            pictureBox.Image = image;

        }

    }

}