

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«Казанский национальный исследовательский технический университет  
им. А.Н. Туполева – КАИ»

Институт компьютерных технологий и защиты информации  
Отделение СПО ИКТЗИ (Колледж информационных технологий)

ЛАБОРАТОРНАЯ РАБОТА №9

по дисциплине

Основы алгоритмизации и программирования

Тема: «Разработка командной строки для управления ходом  
выполнения программы»

Работу выполнил

Студент гр.4238

Бусов В.Р.

Принял

Преподаватель Шмидт И.Р.

Казань 2024

## ВАРИАНТ 4

### Цель работы

Изучение алгоритмов вычисления функциональных выражений в обратной польской записи и особенностей их программной реализации.

### Задание на лабораторную работу

Модифицировать программу, реализованную на предыдущей лабораторной работе «Создание и использование библиотеки классов для графических примитивов». Обновленная версия программы должна включать в себя следующие изменения:

1. Удаление всех элементов управления из формы (кнопок, лейблов, полей для ввода и прочих), кроме поля рисунка PictureBox, где будет размещаться битовая карта;

2. Добавление командной строки (для ее реализации можно использовать элемент TextBox), где будут указываться команды, которые должна будет выполнять программа (прорисовка, перемещение и удаление фигур);

3. Добавить историю команд, где будут размещаться выполненные и неудачные команды.

4	Создание окружности	$y*x*w*name*C$
	Перемещение окружности	$name*d*dy*M$
	Удаление окружности	$name*D$

Рисунок 1 – Вариант выполнения задания

### Результат выполнения работы

Сделаем интерфейс окошка формы в соответствии с заданием: без каких-либо кнопок, оставляем только PictureBox, командную строку, а также поле, в котором будут отображаться выполненные и ошибочные команды. Создадим нашу фигуру с помощью координат, имени, размеров и команды C, которая означает создание фигуры. Фигура создалась, успешное выполнение отобразилось в истории команд (Рисунок 1).

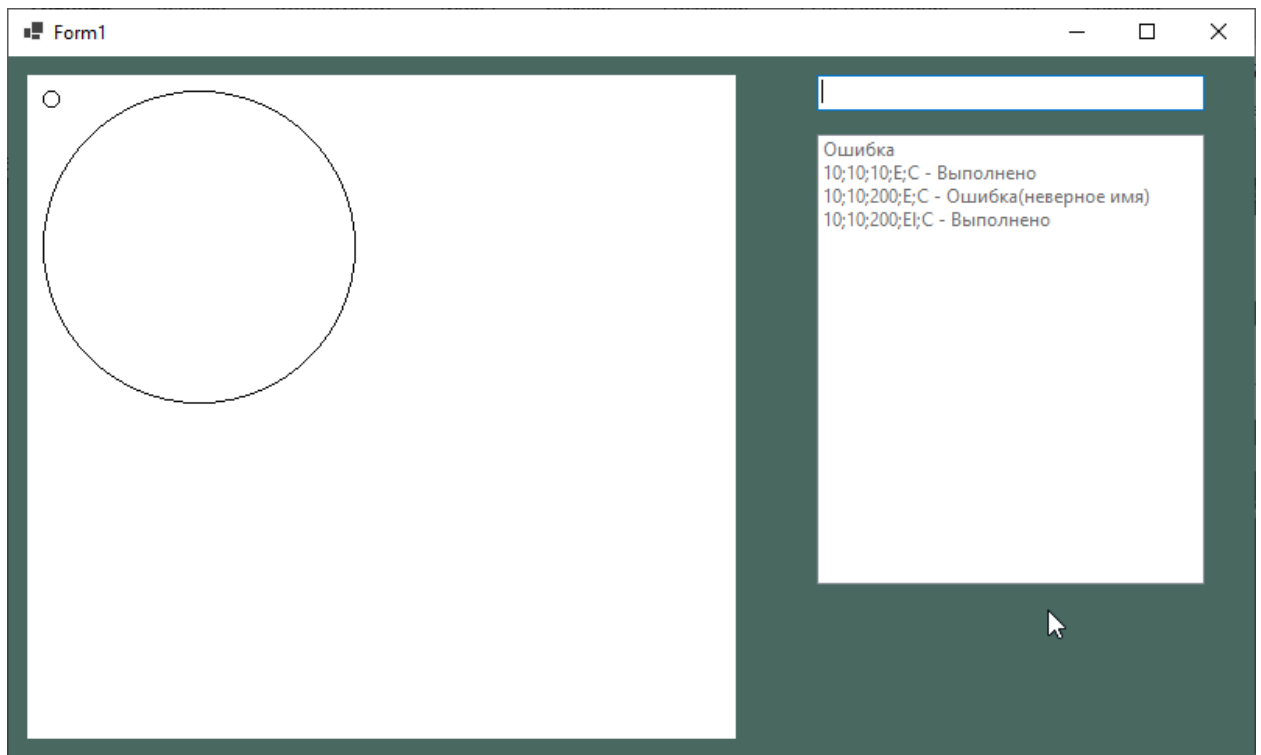


Рисунок 1 – Создание фигуры

После создания выполним перемещение фигуры. Напишем координаты, на которые хотим переместить фигуру, имя и команду M – Move (Рисунок 2).

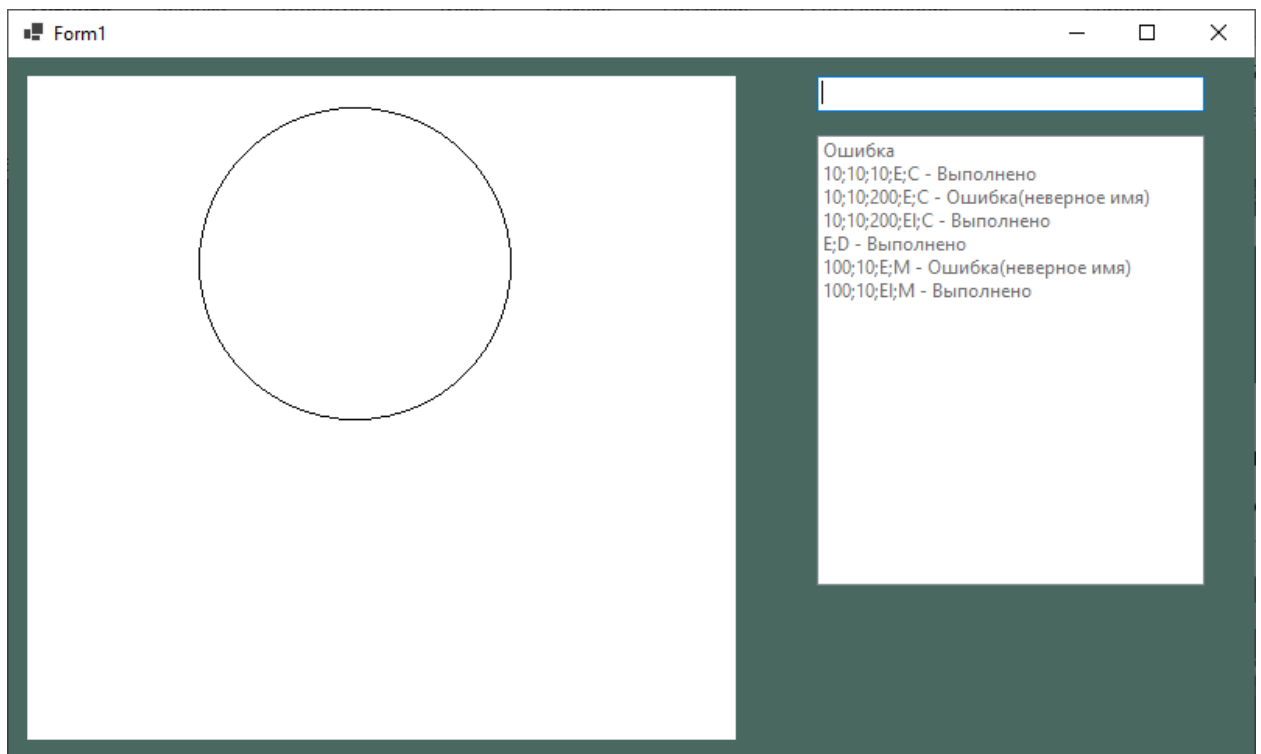


Рисунок 2 – Перемещение фигуры

После того, как мы создали и переместили фигуру, удалим её, обратившись к имени и используя команду D – Delete (Рисунок 3).

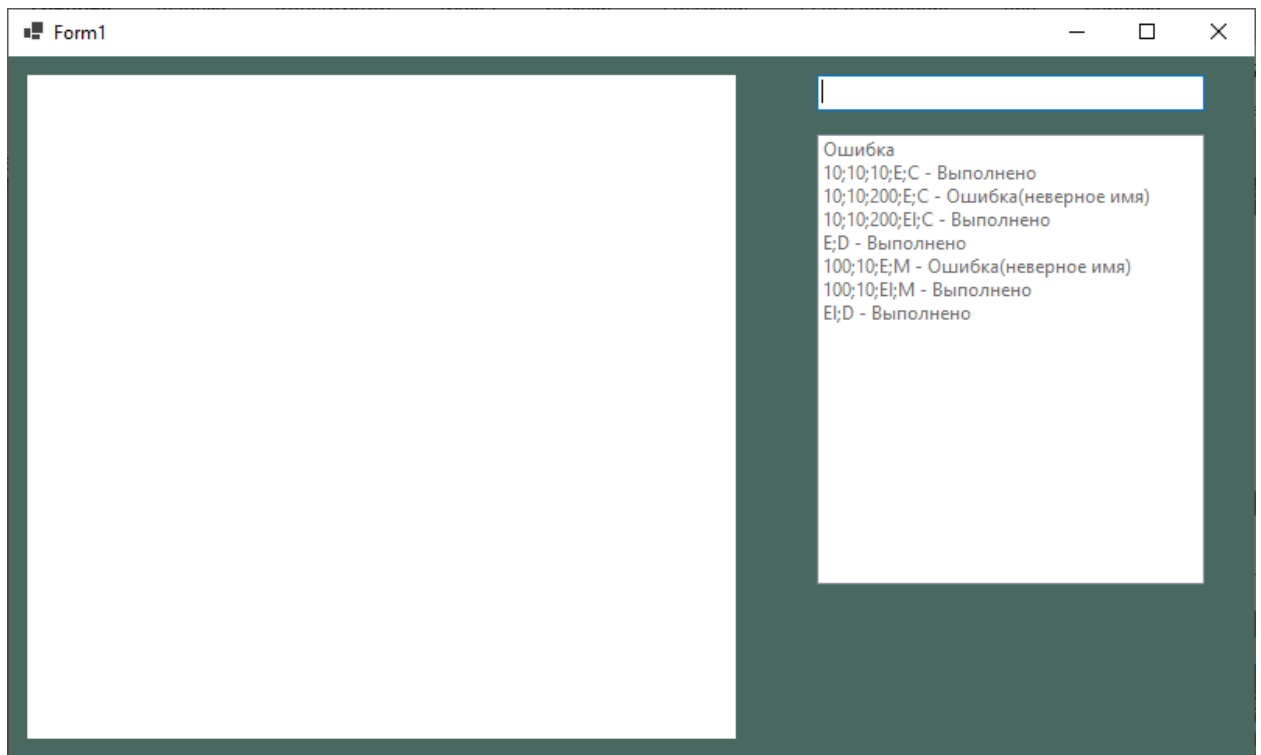


Рисунок 3 – Удаление фигуры

Проверяю различные ошибки при вводе в командную строку такие как: выход за рамки, нехватка размеров фигуры, несоответствующие символы при создании фигуры и другие ошибки. При любом неправильном вводе в поле выводится сообщение об ошибке (Рисунок 4).

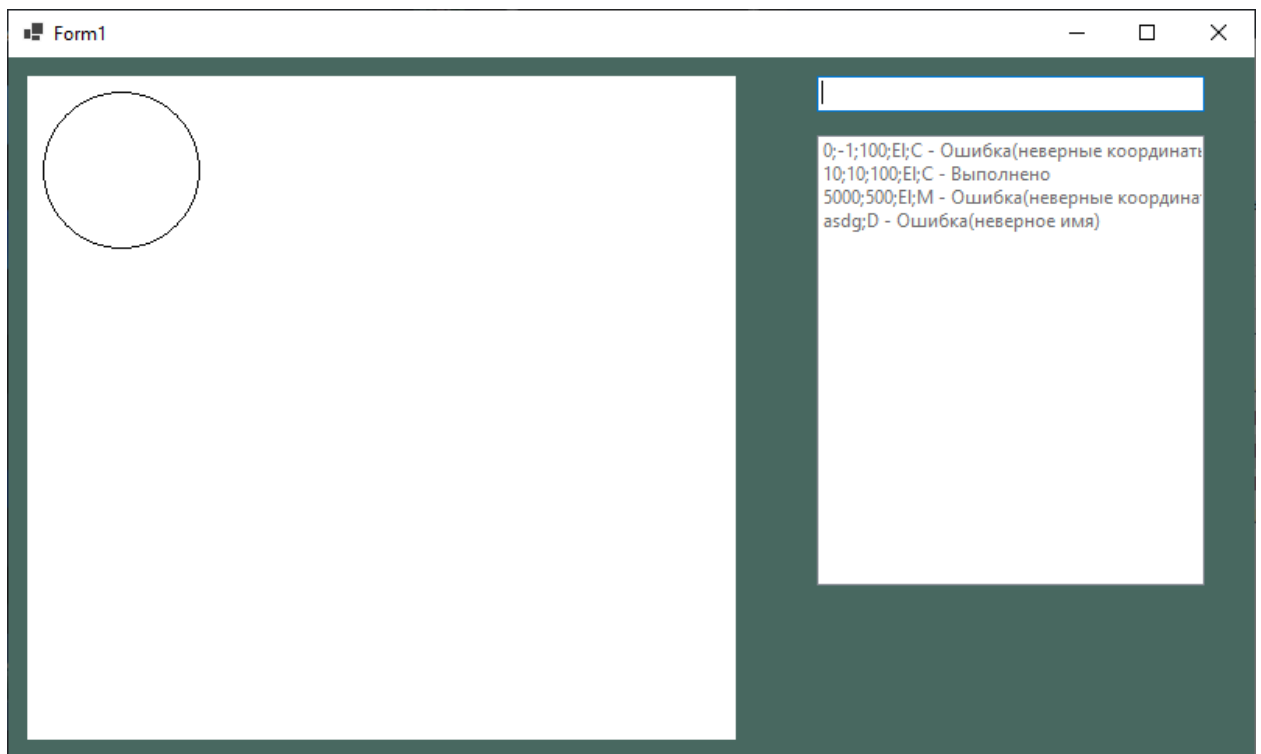


Рисунок 4 – Ошибочный ввод

## Вывод

В процессе выполнения лабораторной работы я изучил алгоритмы вычисления функциональных выражений в обратной польской записи и особенности их программной реализации.

## Листинг

### Form1.cs

```
using static System.Windows.Forms.AxHost;
using System;

namespace WinFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            Init.bitmap = new Bitmap(this.pictureBox1.ClientSize.Width,
pictureBox1.ClientSize.Height); ;
            Init.pen = new Pen(Color.Black, 1);
            Init.pb = this.pictureBox1;
            Init.pbw = Init.pb.Width;
            Init.pbh = Init.pb.Height;
        }

        private void textBox1_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Enter)
            {
                string seq = this.textBox1.Text;
                RPN.calculate_rpn(seq, this);
                this.textBox1.Text = "";
            }
        }

        public void add_item(string text)
        {
            this.listBox1.Items.Add(text);
        }
    }
}
```

### RPN.cs

```
using System;
```

```

using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;
using WinFormsApp1;

namespace WinFormsApp1
{
    static class RPN
    {
        private static Form1 form1;
        private static string seq;
        public static void calculate_rpn(string expression, Form1 f)
        {
            form = f;
            seq = expression;

            Stack<string> stack = new Stack<string>();
            string num = "";
            for (int i = 0; i < expression.Length; i++)
            {
                char c = expression[i];
                if (char.IsDigit(c)) num += c; // если цифра - добавляем к числу
                else if (c == ';' && num != "") // если разделитель - добавляем число в
стек и обнуляем
                {
                    stack.Push(num);
                    num = "";
                }
                else if (is_operator(c) && i + 1 == expression.Length)
                { // если оператор - пытаемся выполнить команду
                    apply(stack, c);
                }
                else num += c;
            }
        }

        private static bool is_operator(char c)
        {
            return c == 'C' || c == 'M' || c == 'D';
        }

        private static void apply(Stack<string> operands, char c)
        {
            if (c == 'C' && operands.Count == 4)
            {
                bool state = true;

```

```

string name = operands.Pop();

for (int i = 0; i < ShapeContainer.length; i++)
{
    if (ShapeContainer.figureList[i].name == name) state = false;
}
if (state)
{
    if (int.TryParse(operands.Pop(), out int w) &&
        int.TryParse(operands.Pop(), out int y) &&
        int.TryParse(operands.Pop(), out int x) &&
        w > 0)
    {
        Ellipse el = new Ellipse(x, y, w, name);
        if (el.move_check(0, 0))
        {
            el.draw();
            form.add_item($"{seq} - Выполнено");
            ShapeContainer.AddFigure(el);
        }
        else form.add_item($"{seq} - Ошибка(неверные координаты)");
    }
    else form.add_item($"{seq} - Ошибка(неверный формат)");
}
else form.add_item($"{seq} - Ошибка(неверное имя)");
}
else if (c == 'M' && operands.Count == 3)
{
    string name = operands.Pop();
    bool state = false;
    int index = 0;

    for (int i = 0; i < ShapeContainer.length; i++)
    {
        if (name == ShapeContainer.figureList[i].name)
        {
            index = i;
            state = true;
            break;
        }
    }

    if (state)
    {
        if (int.TryParse(operands.Pop(), out int y) &&
            int.TryParse(operands.Pop(), out int x))
        {
            Figure f = ShapeContainer.figureList[index];
            if (f.move_check(x, y))

```

```

        {
            f.move_to(x, y);
            form.add_item($"{seq} - Выполнено");
        }
        else form.add_item($"{seq} - Ошибка (неверные координаты)");
    }
    else form.add_item($"{seq} - Ошибка (неверный формат)");
}
else form.add_item($"{seq} - Ошибка (неверное имя)");
}
else if (c == 'D' && operands.Count == 1)
{
    string name = operands.Pop();
    bool state = false;

    for (int i = 0; i < ShapeContainer.length; i++)
    {
        if (name == ShapeContainer.figureList[i].name)
        {
            Figure f = ShapeContainer.figureList[i];
            f.drop_figure(f);
            form.add_item($"{seq} - Выполнено");
            state = true;
            break;
        }
    }
    if (state == false) form.add_item($"{seq} - Ошибка (неверное имя)");
}
else form.add_item("Ошибка");
}
}
}

```

## Figure.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WinFormsApp1
{
    abstract public class Figure
    {
        public string name;

        public int x, y, h, w; // объявляем переменные, характеризующие фигуру

        abstract public void draw();
    }
}

```



```

        abstract public void move_to(int x, int y);
        abstract public bool move_check(int x, int y);

        public Figure()
        {
            string t = DateTime.Now.Subtract(new DateTime(1970, 1,
1)).TotalSeconds.ToString();

        }

        public void drop_figure(Figure f, bool redraw = false)
        {
            Graphics g = Graphics.FromImage(Init.bitmap);
            if (!redraw) ShapeContainer.RemoveFigure(f);
            this.clear();
            Init.pb.Image = Init.bitmap;

            for (int i = 0; i < ShapeContainer.length; i++)
            {
                ShapeContainer.figureList[i].draw();
            }
        }

        public void clear()
        {
            Graphics g = Graphics.FromImage(Init.bitmap);
            g.Clear(Color.White);
        }
    }
}

```

## Ellipse.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WinFormsApp1
{
    public class Ellipse : Figure
    {
        public Ellipse(int x=0, int y=0, int w=40, string name="Ellipse")
        {
            this.x = x;
            this.y = y;
            this.w = w;
            this.h = w;
            this.name = name;
        }
    }
}

```

```

    }

    public override void draw()
    {
        Graphics g = Graphics.FromImage(Init.bitmap);
        g.DrawEllipse(
            Init.pen,
            new RectangleF(
                this.x,
                this.y,
                this.w,
                this.h)
            );

        Init.pb.Image = Init.bitmap;
    }

    public override bool move_check(int x, int y)
    {
        // функция проверяет, можно ли переместить фигуру на заданные координаты
        // в качестве ответа идет булево значение. true - можно переместить, false
- нельзя переместить

        bool lls = this.x + x < 0; // выход за границу левой стороной
        bool lts = this.y + y < 0; // выход за границу верхней стороной
        bool lrs = this.x + this.w + x > Init.pbw; // выход за границу правой
сторонаой
        bool lbs = this.y + this.h + y > Init.pbh; // выход за границу нижней
сторонаой

        return !(lls || lts || lrs || lbs);
    }

    public override void move_to(int x, int y)
    {
        if (this.move_check(x, y))
        {
            this.x += x;
            this.y += y;
            this.drop_figure(this, true);
            this.draw();
        }
    }
}
}

```

## ShapeContainer.cs

```

using System;
using System.Collections.Generic;

```

```
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;

namespace WinFormsApp1
{
    public static class ShapeContainer
    {
        public static List<Figure> figureList;
        public static int length;
        static ShapeContainer()
        {
            figureList = new List<Figure>();
            length = 0;
        }
        public static void AddFigure(Figure figure)
        {
            figureList.Add(figure);
            length += 1;
        }

        public static void RemoveFigure(Figure figure)
        {
            figureList.Remove(figure);
            length -= 1;
        }
    }
}
```