МИНИСТРЕСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

По объектно-ориентированному программированию

«Разработка графического редактора на языке С++ с использованием механизмов ООП»

Вариант 5

Выполнила:

студентка гр. ПМ-21-1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Затонская В.В.

Проверил:

доц., к.п.н. кафедры АСУ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кургасов В. В.

ЛИПЕЦК 2022

**Цель работы:**

Реализовать на языке С++ редактор графической схемы(инфологическая схема).В ходе выполнения работы обязательно применение объектно-ориентированных возможностей языка C++:наследование и динамического полиморфизма. Каждый тип элемента схемы должен быть представлен в программе в виде отдельного класса, который наследован от базового класса “графический элемент”(имеющего чисто виртуальную функцию прорисовки). Также необходим один класс “поле рисования ”,который содержит все графические элементы и отвечает за вызов функций прорисовки. Хранение графических элементов осуществляется с использованием контейнеров стандартной библиотеки C++.

**Задание кафедры:**

Закрепить навыки использования механизмов ООП на примере реализации графического редактора. Закрепить навыки использования механизмов ООП на примере реализации графического редактора.

**Код программы:**

Файл Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Zatonskaya\_IDZ

{

public partial class Form1 : Form

{

string figure\_type = "Ellipse";

int count\_of\_elps = 0;

int count\_of\_prllgrm = 0;

int count\_of\_rctngle = 0;

int count\_of\_dmnd = 0;

int count\_of\_arrw = 0;

int count\_of\_txt = 0;

int count\_of\_str = 0;

int prev\_x = 0;

int prev\_y = 0;

bool mouse\_down = false;

bool focus = false;

bool moving = false;

int catch\_point\_index = -1;

int catch\_figure\_index = -1;

string type\_of\_focused\_figure = "Ellipse";

string[] str = new string[40];

Point[][] elps;

Point[][] prllgrm;

Point[][] rctngle;

Point[][] dmnd;

Point[][] arrw;

Point[][] txt;

public Form1()

{

InitializeComponent();

elps = new Point[100][];

for (int i = 0; i < 100; i++)

elps[i] = new Point[2];

prllgrm = new Point[100][];

for (int i = 0; i < 100; i++)

prllgrm[i] = new Point[2];

rctngle = new Point[100][];

for (int i = 0; i < 100; i++)

rctngle[i] = new Point[2];

dmnd = new Point[100][];

for (int i = 0; i < 100; i++)

dmnd[i] = new Point[2];

arrw = new Point[100][];

for (int i = 0; i < 100; i++)

arrw[i] = new Point[2];

txt = new Point[100][];

for (int i = 0; i < 100; i++)

txt[i] = new Point[1];

}

private void panel3\_MouseUp(object sender, MouseEventArgs e) // Создание объекта

{

mouse\_down = false;

if (!moving)

{

switch (figure\_type)

{

case "Ellipse":

elps[count\_of\_elps][1].X = e.X;

elps[count\_of\_elps][1].Y = e.Y;

count\_of\_elps++;

break;

case "Parallelogram":

prllgrm[count\_of\_prllgrm][1].X = e.X;

prllgrm[count\_of\_prllgrm][1].Y = e.Y;

count\_of\_prllgrm++;

break;

case "Rectangle":

rctngle[count\_of\_rctngle][1].X = e.X;

rctngle[count\_of\_rctngle][1].Y = e.Y;

count\_of\_rctngle++;

break;

case "Diamond":

dmnd[count\_of\_dmnd][1].X = e.X;

dmnd[count\_of\_dmnd][1].Y = e.Y;

count\_of\_dmnd++;

break;

case "Arrow":

arrw[count\_of\_arrw][1].X = e.X;

arrw[count\_of\_arrw][1].Y = e.Y;

count\_of\_arrw++;

break;

case "Text":

txt[count\_of\_txt][0].X = e.X;

txt[count\_of\_txt][0].Y = e.Y;

count\_of\_str++;

count\_of\_txt++;

break;

}

}

panel3.Invalidate();

}

private void panel3\_MouseDown(object sender, MouseEventArgs e) // Берем координаты первой точки фигуры

{

mouse\_down = true;

switch (figure\_type)

{

case "Ellipse":

elps[count\_of\_elps][0].X = e.X;

elps[count\_of\_elps][0].Y = e.Y;

break;

case "Parallelogram":

prllgrm[count\_of\_prllgrm][0].X = e.X;

prllgrm[count\_of\_prllgrm][0].Y = e.Y;

break;

case "Rectangle":

rctngle[count\_of\_rctngle][0].X = e.X;

rctngle[count\_of\_rctngle][0].Y = e.Y;

break;

case "Diamond":

dmnd[count\_of\_dmnd][0].X = e.X;

dmnd[count\_of\_dmnd][0].Y = e.Y;

break;

case "Arrow":

arrw[count\_of\_arrw][0].X = e.X;

arrw[count\_of\_arrw][0].Y = e.Y;

break;

}

}

private void panel3\_MouseMove(object sender, MouseEventArgs e)

{

if (mouse\_down && focus) // Изменение положения захваченной фигуры.

{

switch(type\_of\_focused\_figure)

{

case "Ellipse":

prev\_x = elps[catch\_figure\_index][catch\_point\_index].X;

prev\_y = elps[catch\_figure\_index][catch\_point\_index].Y;

elps[catch\_figure\_index][catch\_point\_index].X = e.X;

elps[catch\_figure\_index][catch\_point\_index].Y = e.Y;

elps[catch\_figure\_index][0].X += e.X - prev\_x;

elps[catch\_figure\_index][0].Y += e.Y - prev\_y;

break;

case "Parallelogram":

prev\_x = prllgrm[catch\_figure\_index][catch\_point\_index].X;

prev\_y = prllgrm[catch\_figure\_index][catch\_point\_index].Y;

prllgrm[catch\_figure\_index][catch\_point\_index].X = e.X;

prllgrm[catch\_figure\_index][catch\_point\_index].Y = e.Y;

prllgrm[catch\_figure\_index][0].X += e.X - prev\_x;

prllgrm[catch\_figure\_index][0].Y += e.Y - prev\_y;

break;

case "Rectangle":

prev\_x = rctngle[catch\_figure\_index][catch\_point\_index].X;

prev\_y = rctngle[catch\_figure\_index][catch\_point\_index].Y;

rctngle[catch\_figure\_index][catch\_point\_index].X = e.X;

rctngle[catch\_figure\_index][catch\_point\_index].Y = e.Y;

rctngle[catch\_figure\_index][0].X += e.X - prev\_x;

rctngle[catch\_figure\_index][0].Y += e.Y - prev\_y;

break;

case "Diamond":

prev\_x = dmnd[catch\_figure\_index][catch\_point\_index].X;

prev\_y = dmnd[catch\_figure\_index][catch\_point\_index].Y;

dmnd[catch\_figure\_index][catch\_point\_index].X = e.X;

dmnd[catch\_figure\_index][catch\_point\_index].Y = e.Y;

dmnd[catch\_figure\_index][0].X += e.X - prev\_x;

dmnd[catch\_figure\_index][0].Y += e.Y - prev\_y;

break;

case "Text":

txt[catch\_figure\_index][0].X = e.X;

txt[catch\_figure\_index][0].Y = e.Y;

break;

}

}

else if (mouse\_down) // Берем координаты второй точки фигуры.

{

switch (figure\_type)

{

case "Ellipse":

elps[count\_of\_elps][1].X = e.X;

elps[count\_of\_elps][1].Y = e.Y;

break;

case "Parallelogram":

prllgrm[count\_of\_prllgrm][1].X = e.X;

prllgrm[count\_of\_prllgrm][1].Y = e.Y;

break;

case "Rectangle":

rctngle[count\_of\_rctngle][1].X = e.X;

rctngle[count\_of\_rctngle][1].Y = e.Y;

break;

case "Diamond":

dmnd[count\_of\_dmnd][1].X = e.X;

dmnd[count\_of\_dmnd][1].Y = e.Y;

break;

case "Arrow":

arrw[count\_of\_arrw][1].X = e.X;

arrw[count\_of\_arrw][1].Y = e.Y;

break;

}

}

else if(figure\_type != "Arrow") // Поиск точки захвата фигуры

{

moving = false;

focus = false;

catch\_point\_index = -1;

catch\_figure\_index = -1;

for (int i = 0; i < count\_of\_elps; i++)

{

if (Math.Abs(elps[i][1].X - e.X) < 5 || Math.Abs(elps[i][1].Y - e.Y) < 5)

{

moving = true;

focus = true;

catch\_point\_index = 1;

catch\_figure\_index = i;

type\_of\_focused\_figure = "Ellipse";

}

}

for (int i = 0; i < count\_of\_prllgrm; i++)

{

if (Math.Abs(prllgrm[i][1].X - e.X) < 5 || Math.Abs(prllgrm[i][1].Y - e.Y) < 5)

{

moving = true;

focus = true;

catch\_point\_index = 1;

catch\_figure\_index = i;

type\_of\_focused\_figure = "Parallelogram";

}

}

for (int i = 0; i < count\_of\_rctngle; i++)

{

if (Math.Abs(rctngle[i][1].X - e.X) < 5 || Math.Abs(rctngle[i][1].Y - e.Y) < 5)

{

moving = true;

focus = true;

catch\_point\_index = 1;

catch\_figure\_index = i;

type\_of\_focused\_figure = "Rectangle";

}

}

for (int i = 0; i < count\_of\_dmnd; i++)

{

if (Math.Abs(dmnd[i][1].X - e.X) < 5 || Math.Abs(dmnd[i][1].Y - e.Y) < 5)

{

moving = true;

focus = true;

catch\_point\_index = 1;

catch\_figure\_index = i;

type\_of\_focused\_figure = "Diamond";

}

}

for (int i = 0; i < count\_of\_txt; i++)

{

if (Math.Abs(txt[i][0].X - e.X) < 5 || Math.Abs(txt[i][0].Y - e.Y) < 5)

{

moving = true;

focus = true;

catch\_point\_index = 0;

catch\_figure\_index = i;

type\_of\_focused\_figure = "Text";

}

}

}

panel3.Invalidate();

}

private void panel3\_Paint(object sender, PaintEventArgs e) // отрисовка всех элементов

{

Graphics g = e.Graphics;

if (focus)

{

switch (type\_of\_focused\_figure) // рисование точек для захвата фигуры

{

case "Ellipse":

g.DrawEllipse(new Pen(Color.Red), elps[catch\_figure\_index][catch\_point\_index].X, elps[catch\_figure\_index][catch\_point\_index].Y, 10, 10);

break;

case "Parallelogram":

g.DrawEllipse(new Pen(Color.Red), prllgrm[catch\_figure\_index][catch\_point\_index].X, prllgrm[catch\_figure\_index][catch\_point\_index].Y, 10, 10);

break;

case "Rectangle":

g.DrawEllipse(new Pen(Color.Red), rctngle[catch\_figure\_index][catch\_point\_index].X, rctngle[catch\_figure\_index][catch\_point\_index].Y, 10, 10);

break;

case "Diamond":

g.DrawEllipse(new Pen(Color.Red), dmnd[catch\_figure\_index][catch\_point\_index].X, dmnd[catch\_figure\_index][catch\_point\_index].Y, 10, 10);

break;

case "Text":

g.DrawEllipse(new Pen(Color.Blue), txt[catch\_figure\_index][0].X, txt[catch\_figure\_index][0].Y, 10, 10);

break;

}

}

for (int i = 0; i < count\_of\_elps; i++)

{

g.DrawEllipse(new Pen(Color.Black), elps[i][0].X, elps[i][0].Y, elps[i][1].X - elps[i][0].X, elps[i][1].Y - elps[i][0].Y);

}

for (int i = 0; i < count\_of\_rctngle; i++)

{

g.DrawRectangle(new Pen(Color.Black), rctngle[i][0].X, rctngle[i][0].Y, rctngle[i][1].X - rctngle[i][0].X, rctngle[i][1].Y - rctngle[i][0].Y);

}

for (int i = 0; i < count\_of\_prllgrm; i++)

{

Point parallelogram\_point1 = new Point(prllgrm[i][0].X, prllgrm[i][0].Y);

Point parallelogram\_point2 = new Point(prllgrm[i][1].X, prllgrm[i][0].Y);

Point parallelogram\_point3 = new Point(prllgrm[i][1].X - Convert.ToInt32(Math.Sqrt(prllgrm[i][1].X)), prllgrm[i][1].Y);

Point parallelogram\_point4 = new Point(prllgrm[i][0].X - Convert.ToInt32(Math.Sqrt(prllgrm[i][1].X)), prllgrm[i][1].Y);

Point[] parallelogram\_points =

{

parallelogram\_point1,

parallelogram\_point2,

parallelogram\_point3,

parallelogram\_point4

};

g.DrawPolygon(new Pen(Color.Black), parallelogram\_points);

}

for (int i = 0; i < count\_of\_dmnd; i++)

{

Point diamond\_point1 = new Point(dmnd[i][0].X, dmnd[i][0].Y - (dmnd[i][1].Y - dmnd[i][0].Y));

Point diamond\_point2 = new Point(dmnd[i][0].X + (dmnd[i][1].X - dmnd[i][0].X), dmnd[i][0].Y);

Point diamond\_point3 = new Point(dmnd[i][0].X, dmnd[i][0].Y + (dmnd[i][1].Y - dmnd[i][0].Y));

Point diamond\_point4 = new Point(dmnd[i][0].X - (dmnd[i][1].X - dmnd[i][0].X), dmnd[i][0].Y);

Point[] diamond\_points =

{

diamond\_point1,

diamond\_point2,

diamond\_point3,

diamond\_point4

};

g.DrawPolygon(new Pen(Color.Black), diamond\_points);

}

Pen arrow\_pen = new Pen(Brushes.Black, 4f);

arrow\_pen.EndCap = System.Drawing.Drawing2D.LineCap.ArrowAnchor;

for (int i = 0; i < count\_of\_arrw; i++)

{

g.DrawLine(arrow\_pen, arrw[i][0].X, arrw[i][0].Y, arrw[i][1].X, arrw[i][1].Y);

}

Font myfont = new Font("Arial", 8.0f);

for (int i = 0; i < count\_of\_txt; i++)

{

g.DrawString(str[i], myfont, Brushes.Black, txt[i][0].X, txt[i][0].Y);

}

if (mouse\_down == true)

{

g.DrawEllipse(new Pen(Color.Black), elps[count\_of\_elps][0].X, elps[count\_of\_elps][0].Y, elps[count\_of\_elps][1].X - elps[count\_of\_elps][0].X, elps[count\_of\_elps][1].Y - elps[count\_of\_elps][0].Y);

g.DrawRectangle(new Pen(Color.Black), rctngle[count\_of\_rctngle][0].X, rctngle[count\_of\_rctngle][0].Y, rctngle[count\_of\_rctngle][1].X - rctngle[count\_of\_rctngle][0].X, rctngle[count\_of\_rctngle][1].Y - rctngle[count\_of\_rctngle][0].Y);

Point parallelogram\_point1 = new Point(prllgrm[count\_of\_prllgrm][0].X, prllgrm[count\_of\_prllgrm][0].Y);

Point parallelogram\_point2 = new Point(prllgrm[count\_of\_prllgrm][1].X, prllgrm[count\_of\_prllgrm][0].Y);

Point parallelogram\_point3 = new Point(prllgrm[count\_of\_prllgrm][1].X - Convert.ToInt32(Math.Sqrt(prllgrm[count\_of\_prllgrm][1].X)), prllgrm[count\_of\_prllgrm][1].Y);

Point parallelogram\_point4 = new Point(prllgrm[count\_of\_prllgrm][0].X - Convert.ToInt32(Math.Sqrt(prllgrm[count\_of\_prllgrm][1].X)), prllgrm[count\_of\_prllgrm][1].Y);

Point[] parallelogram\_points =

{

parallelogram\_point1,

parallelogram\_point2,

parallelogram\_point3,

parallelogram\_point4

};

g.DrawPolygon(new Pen(Color.Black), parallelogram\_points);

Point diamond\_point1 = new Point(dmnd[count\_of\_dmnd][0].X, dmnd[count\_of\_dmnd][0].Y - (dmnd[count\_of\_dmnd][1].Y - dmnd[count\_of\_dmnd][0].Y));

Point diamond\_point2 = new Point(dmnd[count\_of\_dmnd][0].X + (dmnd[count\_of\_dmnd][1].X - dmnd[count\_of\_dmnd][0].X), dmnd[count\_of\_dmnd][0].Y);

Point diamond\_point3 = new Point(dmnd[count\_of\_dmnd][0].X, dmnd[count\_of\_dmnd][0].Y + (dmnd[count\_of\_dmnd][1].Y - dmnd[count\_of\_dmnd][0].Y));

Point diamond\_point4 = new Point(dmnd[count\_of\_dmnd][0].X - (dmnd[count\_of\_dmnd][1].X - dmnd[count\_of\_dmnd][0].X), dmnd[count\_of\_dmnd][0].Y);

Point[] diamond\_points =

{

diamond\_point1,

diamond\_point2,

diamond\_point3,

diamond\_point4

};

g.DrawPolygon(new Pen(Color.Black), diamond\_points);

g.DrawLine(arrow\_pen, arrw[count\_of\_arrw][0].X, arrw[count\_of\_arrw][0].Y, arrw[count\_of\_arrw][1].X, arrw[count\_of\_arrw][1].Y);

g.DrawString(str[count\_of\_str], myfont, Brushes.Black, txt[count\_of\_txt][0].X, txt[count\_of\_txt][0].Y);

}

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

figure\_type = "Ellipse";

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

figure\_type = "Parallelogram";

}

private void radioButton3\_CheckedChanged(object sender, EventArgs e)

{

figure\_type = "Rectangle";

}

private void radioButton4\_CheckedChanged(object sender, EventArgs e)

{

figure\_type = "Diamond";

}

private void radioButton6\_CheckedChanged(object sender, EventArgs e)

{

figure\_type = "Arrow";

}

private void radioButton5\_CheckedChanged(object sender, EventArgs e)

{

figure\_type = "Text";

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

str[count\_of\_str] = textBox1.Text;

}

private void panel1\_Paint(object sender, PaintEventArgs e)

{

}

}

Form1.Designer.rs

namespace Zatonskaya\_IDZ

{

partial class Form1

{

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

private void InitializeComponent()

{

this.panel1 = new System.Windows.Forms.Panel();

this.textBox1 = new System.Windows.Forms.TextBox();

this.radioButton5 = new System.Windows.Forms.RadioButton();

this.radioButton6 = new System.Windows.Forms.RadioButton();

this.radioButton4 = new System.Windows.Forms.RadioButton();

this.radioButton3 = new System.Windows.Forms.RadioButton();

this.radioButton2 = new System.Windows.Forms.RadioButton();

this.radioButton1 = new System.Windows.Forms.RadioButton();

this.panel2 = new System.Windows.Forms.Panel();

this.panel3 = new System.Windows.Forms.Panel();

this.panel1.SuspendLayout();

this.SuspendLayout();

//

// panel1

//

this.panel1.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(163)))), ((int)(((byte)(206)))), ((int)(((byte)(159)))));

this.panel1.Controls.Add(this.textBox1);

this.panel1.Controls.Add(this.radioButton5);

this.panel1.Controls.Add(this.radioButton6);

this.panel1.Controls.Add(this.radioButton4);

this.panel1.Controls.Add(this.radioButton3);

this.panel1.Controls.Add(this.radioButton2);

this.panel1.Controls.Add(this.radioButton1);

this.panel1.Dock = System.Windows.Forms.DockStyle.Top;

this.panel1.Location = new System.Drawing.Point(0, 0);

this.panel1.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.panel1.Name = "panel1";

this.panel1.Size = new System.Drawing.Size(1886, 165);

this.panel1.TabIndex = 0;

this.panel1.Paint += new System.Windows.Forms.PaintEventHandler(this.panel1\_Paint);

//

// textBox1

//

this.textBox1.Location = new System.Drawing.Point(395, 57);

this.textBox1.Margin = new System.Windows.Forms.Padding(3, 4, 3, 4);

this.textBox1.Name = "textBox1";

this.textBox1.Size = new System.Drawing.Size(112, 26);

this.textBox1.TabIndex = 7;

this.textBox1.TextChanged += new System.EventHandler(this.textBox1\_TextChanged);

//

// radioButton5

//

this.radioButton5.AutoSize = true;

this.radioButton5.Font = new System.Drawing.Font("Comic Sans MS", 10F, System.Drawing.FontStyle.Bold);

this.radioButton5.Location = new System.Drawing.Point(395, 14);

this.radioButton5.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.radioButton5.Name = "radioButton5";

this.radioButton5.Size = new System.Drawing.Size(92, 33);

this.radioButton5.TabIndex = 6;

this.radioButton5.TabStop = true;

this.radioButton5.Text = "Текст";

this.radioButton5.UseVisualStyleBackColor = true;

this.radioButton5.CheckedChanged += new System.EventHandler(this.radioButton5\_CheckedChanged);

//

// radioButton6

//

this.radioButton6.AutoSize = true;

this.radioButton6.Font = new System.Drawing.Font("Comic Sans MS", 10F, System.Drawing.FontStyle.Bold);

this.radioButton6.Location = new System.Drawing.Point(13, 100);

this.radioButton6.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.radioButton6.Name = "radioButton6";

this.radioButton6.Size = new System.Drawing.Size(115, 33);

this.radioButton6.TabIndex = 5;

this.radioButton6.TabStop = true;

this.radioButton6.Text = "Стрелка";

this.radioButton6.UseVisualStyleBackColor = true;

this.radioButton6.CheckedChanged += new System.EventHandler(this.radioButton6\_CheckedChanged);

//

// radioButton4

//

this.radioButton4.AutoSize = true;

this.radioButton4.Font = new System.Drawing.Font("Comic Sans MS", 10F, System.Drawing.FontStyle.Bold);

this.radioButton4.Location = new System.Drawing.Point(220, 57);

this.radioButton4.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.radioButton4.Name = "radioButton4";

this.radioButton4.Size = new System.Drawing.Size(120, 33);

this.radioButton4.TabIndex = 3;

this.radioButton4.TabStop = true;

this.radioButton4.Text = "Условие";

this.radioButton4.UseVisualStyleBackColor = true;

this.radioButton4.CheckedChanged += new System.EventHandler(this.radioButton4\_CheckedChanged);

//

// radioButton3

//

this.radioButton3.AutoSize = true;

this.radioButton3.Font = new System.Drawing.Font("Comic Sans MS", 10F, System.Drawing.FontStyle.Bold);

this.radioButton3.Location = new System.Drawing.Point(220, 14);

this.radioButton3.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.radioButton3.Name = "radioButton3";

this.radioButton3.Size = new System.Drawing.Size(135, 33);

this.radioButton3.TabIndex = 2;

this.radioButton3.TabStop = true;

this.radioButton3.Text = "Операция";

this.radioButton3.UseVisualStyleBackColor = true;

this.radioButton3.CheckedChanged += new System.EventHandler(this.radioButton3\_CheckedChanged);

//

// radioButton2

//

this.radioButton2.AutoSize = true;

this.radioButton2.Font = new System.Drawing.Font("Comic Sans MS", 10F, System.Drawing.FontStyle.Bold);

this.radioButton2.Location = new System.Drawing.Point(13, 57);

this.radioButton2.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.radioButton2.Name = "radioButton2";

this.radioButton2.Size = new System.Drawing.Size(158, 33);

this.radioButton2.TabIndex = 1;

this.radioButton2.TabStop = true;

this.radioButton2.Text = "Ввод/Вывод";

this.radioButton2.UseVisualStyleBackColor = true;

this.radioButton2.CheckedChanged += new System.EventHandler(this.radioButton2\_CheckedChanged);

//

// radioButton1

//

this.radioButton1.AutoSize = true;

this.radioButton1.Font = new System.Drawing.Font("Comic Sans MS", 10F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.radioButton1.Location = new System.Drawing.Point(13, 14);

this.radioButton1.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.radioButton1.Name = "radioButton1";

this.radioButton1.Size = new System.Drawing.Size(180, 33);

this.radioButton1.TabIndex = 0;

this.radioButton1.TabStop = true;

this.radioButton1.Text = "Начало/Конец";

this.radioButton1.UseVisualStyleBackColor = true;

this.radioButton1.CheckedChanged += new System.EventHandler(this.radioButton1\_CheckedChanged);

//

// panel2

//

this.panel2.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(64)))), ((int)(((byte)(64)))), ((int)(((byte)(64)))));

this.panel2.Dock = System.Windows.Forms.DockStyle.Bottom;

this.panel2.Location = new System.Drawing.Point(0, 1035);

this.panel2.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.panel2.Name = "panel2";

this.panel2.Size = new System.Drawing.Size(1886, 15);

this.panel2.TabIndex = 1;

//

// panel3

//

this.panel3.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(211)))), ((int)(((byte)(237)))), ((int)(((byte)(209)))));

this.panel3.Dock = System.Windows.Forms.DockStyle.Fill;

this.panel3.Location = new System.Drawing.Point(0, 165);

this.panel3.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.panel3.Name = "panel3";

this.panel3.Size = new System.Drawing.Size(1886, 870);

this.panel3.TabIndex = 2;

this.panel3.Paint += new System.Windows.Forms.PaintEventHandler(this.panel3\_Paint);

this.panel3.MouseDown += new System.Windows.Forms.MouseEventHandler(this.panel3\_MouseDown);

this.panel3.MouseMove += new System.Windows.Forms.MouseEventHandler(this.panel3\_MouseMove);

this.panel3.MouseUp += new System.Windows.Forms.MouseEventHandler(this.panel3\_MouseUp);

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(9F, 20F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(211)))), ((int)(((byte)(237)))), ((int)(((byte)(209)))));

this.ClientSize = new System.Drawing.Size(1886, 1050);

this.Controls.Add(this.panel3);

this.Controls.Add(this.panel2);

this.Controls.Add(this.panel1);

this.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.Name = "Form1";

this.panel1.ResumeLayout(false);

this.panel1.PerformLayout();

this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.Panel panel1;

private System.Windows.Forms.Panel panel2;

private System.Windows.Forms.Panel panel3;

private System.Windows.Forms.RadioButton radioButton6;

private System.Windows.Forms.RadioButton radioButton4;

private System.Windows.Forms.RadioButton radioButton3;

private System.Windows.Forms.RadioButton radioButton2;

private System.Windows.Forms.RadioButton radioButton1;

private System.Windows.Forms.RadioButton radioButton5;

private System.Windows.Forms.TextBox textBox1;

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Zatonskaya\_IDZ

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

}**Ответы на контрольные вопросы:**

**1.** **Что такое делегат?**

Делегат — класс, который позволяет хранить в себе ссылку на метод с определённой сигнатурой произвольного класса. Экземпляры делегатов содержат ссылки на конкретные методы конкретных классов. Делегаты используются, в частности, для определения прототипа функции обратного вызова, например, в событийной модели

**2.Что такое событие? Как его можно создать?**

Событие в объектно-ориентированном программировании — это сообщение, которое возникает в различных точках исполняемого кода при выполнении определённых условий.

Объявить через event с название делегата.

**3. Как передать в событие один или несколько параметров?**

Delegate void делегат(int i)

Ref class класс{

Public:

event делегат^ событие;

Void функция()

{

событие(5); // передача в событие

}

}

Ref class класс\_функции {

Public:

Void перадача\_в\_эту\_функцию по подписке(int m)

{

Return m;

}

}

Main(){

класс\_функции ^ экземпляр = gcnew класс\_функции ();

класс-> событие += gcnew делегат(экземпляр, &класс\_функции:: перадача\_в\_эту\_функцию по подписке);}

**4.** **Что такое обработчик события?**

Событие в объектно-ориентированном программировании – это сообщение, которое возникает в различных точках использования кода при выполнении определенных условий.

**5. Как отслеживать наступление события?**

Подписать функцию на событие и использовать объект события в этой функции для проверки.

**Вывод:**

Я научилась создавать и использовать собственные события для возможности отслеживания другими классами наступления определенных условий в текущем классе.