Практическая работа №16_3

Тема: Разработка приложений с графическими компонентами

Цель работы: изучение приемов создания приложений Windows Forms с использованием графических методов

Задачи:

– изучение приемов создания приложений Windows Forms с графическими объектами;
– создание проектов с использованием графических методов.

Материально-техническое обеспечение:

Место проведения: Компьютерный класс.

Время на выполнение работы: 2 часа.

Оборудование: ПК

Средства обучения: операционная система, текстовый процессор MS Word, программные средства определенного вида

Исходные данные:

1. Конспект занятия.
2. Задание для практической работы.

Перечень справочной литературы:

1) Программирование на языке высокого уровня. Программирование на языке С++: учеб. пособие / Т. И. Немцова, С. Ю. Голова, А. И. Терентьев ; под ред. Л. Г. Гагариной. – М. : ИД «ФОРУМ» : ИНФРА-М, 2018. – 512 с. – (Среднее профессиональное образование).

Краткие теоретические сведения:

Пространство имен **System.Drawing** обеспечивает доступ к функциональным возможностям графического интерфейса GDI+.

Основные пространства имен:

Пространство имен	Описание
System.Drawing	Базовое пространство имен GDI+, определяющее множество типов для основных операций визуализации (шрифты, перья, основные кисти и т.д.), а также основной тип **Graphics**
System.Drawing.Drawing2D	Предлагает типы, используемые для более сложной двумерной/векторной графики (градиентные кисти, стили концов линий для перьев, геометрические трансформации и т.д.)
System.Drawing.Imaging	Предлагает типы, обеспечивающие обработку графических изображений (изменение палитры, извлечение метаданных изображения, работа с метафайлами и т.д.)
System.Drawing.Printing	Предлагает типы, обеспечивающие отображение графики на печатной странице, непосредственное взаимодействие с принтером и определение полного формата задания печати
System.Drawing.Text	Дает возможность управлять коллекциями шрифтов

Класс **Graphics** предоставляет методы рисования на устройстве отображения. Такие классы, как **Rectangle** и **Point**, инкапсулируют элементы GDI+.

Класс **Pen** используется для рисования линий и кривых, а классы, производные от абстрактного класса **Brush**, используются для заливки фигур.

Класс - Pen

Определяет объект, используемый для рисования прямых линий и кривых. Этот класс не наследуется.

Pen(Color)	Инициализирует новый экземпляр класса Pen с указанным цветом.
Pen(Color, Single)	Инициализирует новый экземпляр класса Pen с указанными свойствами Color и Width. (Width - устанавливает ширину пера Pen, в единицах объекта Graphics, используемого для рисования)

Класс - Brush

Определяет объекты, которые используются для заливки внутри графических фигур, таких как прямоугольники, эллипсы, круги, многоугольники и дорожки.

Это абстрактный базовый класс, который не может быть реализован. Для создания объекта "кисть" используются классы, производные от Brush, такие как *SolidBrush*, *TextureBrush* и *LinearGradientBrush*.

Сплайн (основной или фундаментальный сплайн, аналог лекала в черчении) — это последовательность отдельных кривых, объединенных в одну большую кривую. Сплайн задается массивом точек и параметром упругости. Сплайн обязательно проходит через заданные массивом *pm* точки на плоскости, а параметр упругости (*elasticity* — вещественное число типа *float*) определяет плавность их соединения.

Если **упругость = 0.0f** (бесконечная физическая упругость), то соединение точек будет выполнено прямыми отрезками.

Если **упругость = 1.0f** (отсутствие физической упругости), то кривая имеет наименьший суммарный изгиб.

Если **упругость > 1.0f,** то кривая напоминает сдавленный берегами ручей, стремящийся увеличить изгиб своих излучин и течь по более длинному пути. Лучше всего исследовать влияние упругости, меняя ее и рисуя сплайн заново.

Для вызова метода достаточно написать

g.DrawCurve(pen, pm, elasticity, FillMode.Alternate); **// незамкнутый сплайн или**

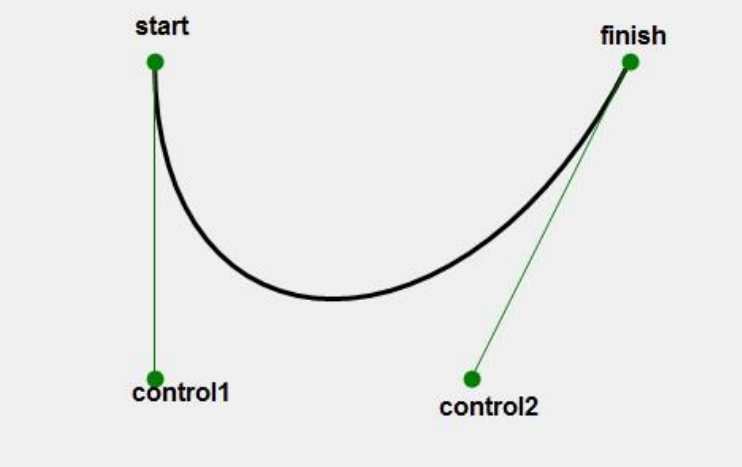
g.DrawClosedCurve(pen, pm, elasticity, FillMode.Alternate); **// замкнутый сплайн**

Отличие второго метода от первого в том, что последняя точка массива добавляется от первой (её можно не добавлять к массиву).

Кривой Безье называется кривая, задаваемая полиномом 3 порядка: $y=ax^3+bx^2+cx+d$. То есть для ее определения нужно задать четыре параметра (a,b,c,d).

Кривые Безье используются очень часто в 2d- и 3d-графике. Они позволяют задавать любой тип соединений («сшивания») кривых, в этом они являются более универсальными, чем рассмотренные выше сплайны (для которых кроме заданных точек дополнительно вводится только один параметр — упругость).

Для полного определения кривой Безье, проходящей через начальную (**start**) и конечную (**finish**) точки необходимо задать еще две управляющие точки (**control1** и **control2**), задающих касательные к этим точкам и радиусы кривизны.



Отдаляя управляющую точку **control1** от точки **start**, мы увеличиваем радиус кривизны, смещая ее относительно точки **start**, мы изменяем направление касательной к кривой в точке **start**. Все это справедливо и для пары **finish-control2**.

Эти четыре точки полностью определяют коэффициенты (**a,b,c,d**) кривой Безье. Полезно самостоятельно исследовать, как меняется форма кривой Безье при перемещении управляющих точек.

Ход работы:

Требования к содержанию отчета:

- Номер и название практической работы.
- Цель работы.
- По каждой заданию (задаче/примеру) экранные формы (при наличии) и листинг программного кода, показывающие порядок выполнения практической работы, и результаты, полученные в ходе её выполнения.
- Ответы на контрольные вопросы в тетради.

Порядок выполнения работы:

Все проекты практической работы размещать в своей сетевой в новой папке **Пр16_3_ФИО**

В начале каждого файла проекта установить комментарии: пр.р.№_____ (указать номер), свою Фамилию. Формулировку задания

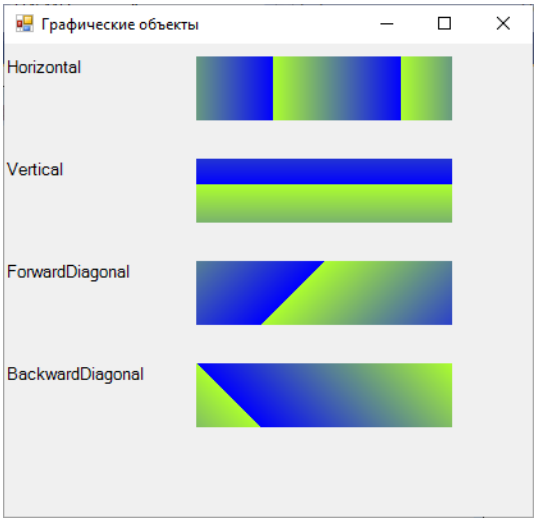
Задание 1. Работа с типами Brush. Работа с **LinearGradientBrush**

*Типы, производные от **System.Drawing.Brush**, используются для заполнения имеющегося региона заданным цветом, узором или изображением.*

*Сам класс **Brush** является абстрактным типом, поэтому он не позволяет создать соответствующий экземпляр непосредственно.*

*Однако **Brush** может играть роль базового класса для родственных ему типов кисти (например, **SolidBrush**, **HatchBrush**, **LinearGradientBrush** и др.).*

*Имея кисть, вы получаете возможность вызвать любой из методов **FillXXX()** типа **Graphics**.*



Постановка задачи: вывести на форму различные типы градиентных заливок в прямоугольной форме для каждого стиля с подписями названия стиля

✓ Создайте новый проект **pr16_3.1_Фамилия** типа Windows Forms. Измените заголовок формы

✓ В шаблон обработчика события **Paint** введите следующий код:

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void Form1_Paint(object sender, PaintEventArgs e)
    {
        Graphics g = e.Graphics;
        Rectangle r = new Rectangle(10, 10, 100, 100);
        //Градиентная кисть.
        LinearGradientBrush theBrush = null;
        int yOffset = 10;
        //Получение членов перечня LinearGradientMode.
        Array obj = Enum.GetValues(typeof(LinearGradientMode));
        //Отображение прямоугольников для членов LinearGradientMode.
        for (int x = 0; x < obj.Length; x++)
        {
            //Конфигурация кисти.
            LinearGradientMode temp = (LinearGradientMode)obj.GetValue(x);
            theBrush = new LinearGradientBrush (r, Color.GreenYellow, Color.Blue, temp);
            //Вывод имени из перечня LinearGradientMode.
            g.DrawString(temp.ToString(), new Font("Times Sew Raman", 10),
                new SolidBrush(Color.Black), 0, yOffset);
            //Закраска прямоугольника подходящей кистью.
            g.FillRectangle(theBrush, 150, yOffset, 200, 50);
            yOffset += 80;
        }
    }
}
```

✓ Протестируем программу. При необходимости откорректируйте код.

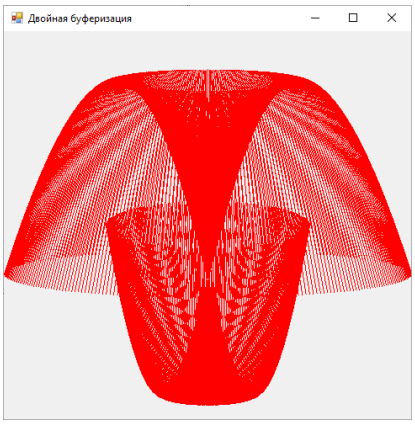
Задание 2. Двойная буферизация.

*Мерцание является распространенной проблемой при программировании графики. Графические операции, требующие нескольких сложных операций рисования, могут привести к тому, что визуализируемые изображения будут мерцать или иметь неприемлемый внешний вид. Чтобы устранить эти неполадки, **.NET Framework** предоставляет **доступ к двойной буферизации**.*

При двойной буферизации все операции рисования сначала выполняются в памяти, а лишь затем на экране компьютера. После завершения всех операций рисования содержимое буфера

копируется из памяти непосредственно на связанную с ним область экрана.

✓ Создайте новый проект **pr16_3.2_Фамилия** типа Windows Forms. Измените заголовок формы



✓ Реализуйте подключение двойной буферизации и в шаблон обработчика события **Paint** введите следующий код:

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
        this.Size = new Size(500, 500);
        this.DoubleBuffered = true;     //Включение двойной буферизации
    }

    private void Form1_Paint(object sender, PaintEventArgs e)
    {
        Graphics g = e.Graphics;
        //Двойная буферизация
        int num = 300;
        int cx = ClientSize.Width;
        int cy = ClientSize.Height;
        Pen pen = new Pen(Color.Red);
        PointF[] aptf = new PointF[4];
        for (int i = 0; i < num; i++)
        {
            double dAngle = 2 * i * Math.PI / num;
            aptf[0].X = cx / 2 + cx / 2 * (float)Math.Cos(dAngle);
            aptf[0].Y = 5 * cy / 8 + cy / 16 * (float)Math.Sin(dAngle);
            aptf[1] = new PointF(cx / 2, -cy);
            aptf[2] = new PointF(cx / 2, 2 * cy);
            dAngle += Math.PI;
            aptf[3].X = cx / 2 + cx / 4 * (float)Math.Cos(dAngle);
            aptf[3].Y = cy / 2 + cy / 16 * (float)Math.Sin(dAngle);
            g.DrawBeziers(pen, aptf);
        }
    }
}
```

✓ Протестируем программу. При необходимости откорректируйте код. Установите комментарии

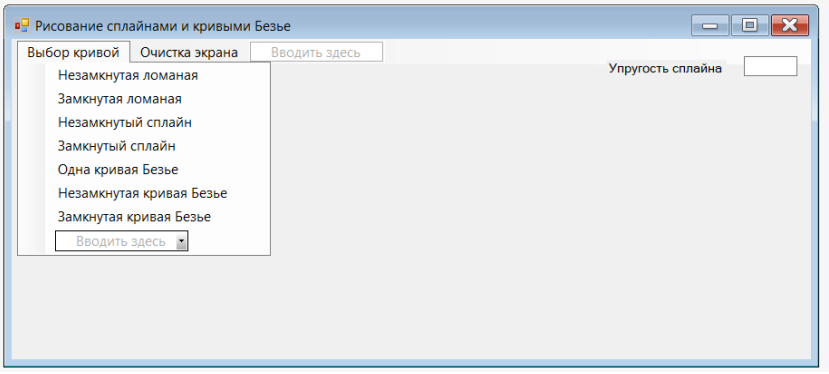
Задание 3. Вывод графических примитивов на форму. Рисование сплайнами и кривыми Безье
Постановка задачи. Требуется рисовать объекты 2d-графики, используя плавные кривые.

✓ Создайте новый проект **pr16_3.3_Фамилия** типа Windows Forms.

✓ На форме размером **1200x600** поместите надпись (**label1**) «**Упругость сплайна**» и окно для ввода **textBox1**.

✓ Создайте меню с вкладками: «**Выбор кривой**» и «**Очистка экрана**». Задайте семь типов кривой (подменю «Выбор кривой»):

1) Незамкнутая ломаная,
2) Замкнутая ломаная,
3) Незамкнутый сплайн,
4) Замкнутый сплайн,
5) Одна кривая Безье,
6) Незамкнутая кривая Безье,
7) Замкнутая кривая Безье.



✓ Реализуйте программный код обработчика событий при выборе соответствующего пункта меню:

Итак, начнем разбираться со сплайнами и кривой Безье. Нам понадобятся объекты классов **Graphics, SolidBrush, Pen, Point** и **Random**. Каждой позиции меню нужно задать соответствующие обработчики событий, например, для незамкнутой ломаной будет задан метод ***незамкнутаяЛоманаяToolStripMenuItem_Click()***, а для очистки экрана — метод ***очисткаЭкранаToolStripMenuItem_Click()***.

1) Добавьте два метода класса Form1:
   - возвращающий массив 10 точек «Звезды»

```
private Point[] star5(int x, int y, int r)  // звезда
{
    Point point1 = new Point(x, y - 6 * r);
    Point point2 = new Point(x + 2 * r, y - 3 * r);
    Point point3 = new Point(x + 6 * r, y - 2 * r);
    Point point4 = new Point(x + 3 * r, y + r);
    Point point5 = new Point(x + 4 * r, y + 5 * r);
    Point point6 = new Point(x, y + 3 * r);
    Point point7 = new Point(x - 4 * r, y + 5 * r);
    Point point8 = new Point(x - 3 * r, y + r);
    Point point9 = new Point(x - 6 * r, y - 2 * r);
    Point point10 = new Point(x - 2 * r, y - 3 * r);
    Point[] pm =
    {
     point1,
     point2,
     point3,
     point4,
     point5,
     point6,
     point7,
     point8,
     point9,
     point10
    };
    return pm;
}
```

- описание изменения цвета

```
public Color RandomColor()
{
    int r, g, b;
    byte[] bytes1 = new byte[3];      // массив 3 цветов
    Random rnd1 = new Random();
    rnd1.NextBytes(bytes1);           // генерация в массив
    r = Convert.ToInt16(bytes1[0]);
    g = Convert.ToInt16(bytes1[1]);
    b = Convert.ToInt16(bytes1[2]);
    return Color.FromArgb(r, g, b);   // возврат цвета
}
```

Для рисования будем использовать холст и его свойства **g: SmoothingMode** и **FillMode**, добавленные в библиотеке **System.Drawing.Drawing2D**; а также методы класса *Graphics*: **FromHwnd(), Clear(), DrawLine(), DrawLines(), DrawCurve(), DrawCloseCurve(), DrawBezier(), DrawBeziers(), Fillellipse().**

Линии, также как прямоугольники и эллипсы, являются графическими примитивами.

*Для того, чтобы нарисовать отрезок прямой, надо указать координаты начала (**Point start**) и конца (**Point end**) отрезка, выбрать перо (**pen**) для рисования и применить метод объекта **g.DrawLine(pen, start, end)**.*

*Для рисования незамкнутой ломаной, состоящей из отрезков, нужно задать массив последовательных точек **Point[] pm** и применить метод **g.DrawLines(pen, pm)**. Если в качестве последней точки массива указать координаты первой точки, то контур станет замкнутым.*

*Для рисования кривых следует использовать методы **DrawCurve()** или **DrawCloseCurve()** для рисования **сплайном**, или же **DrawBezier()** и **DrawBeziers()** для рисования **кривых Безье.***

Основные пояснения в виде комментариев в тексте программы (файл Form1.cs):

2) Подключите дополнительную необходимую библиотеку, инициализируйте холст и установите его свойства:

```
public partial class Form1 : Form
{
    Graphics g;
    public Form1()
    {
        InitializeComponent();
        g = Graphics.FromHwnd(this.Handle);  // холст
        // сглаживание: enum SmoothingMode
        g.SmoothingMode = SmoothingMode.AntiAlias;
    }
```

3) Опишите код обработчика события подменю **Замкнутая ломаная**

```
private void замкнутаяЛомаяToolStripMenuItem_Click(object sender, EventArgs e)
{
    Pen greenPen = new Pen(Color.Green, 2f);
    // Задание 10 точек, определяющих замкнутую область "звезда"
    Point[] pm = star5(250, 220, 25);
    // Рисование замкнутой ломаной из отрезков
    g.DrawClosedCurve(greenPen, pm, 0f, FillMode.Winding);
    // Упругость: 0f - беск.физ.упругость - прямыми;
}
```

Значения ***FillMode***

Alternate	*0*	*Задает режим заполнения с чередованием.*
Winding	*1*	*Задает режим заполнения с поворотом.*

4) Используя **DrawLines** самостоятельно опишите код обработчика события подменю **Незамкнутая ломаная.** Координаты подберите самостоятельно
5) Протестируем программу. При необходимости откорректируйте код.
6) Опишите код обработчика события подменю **Незамкнутый сплайн**

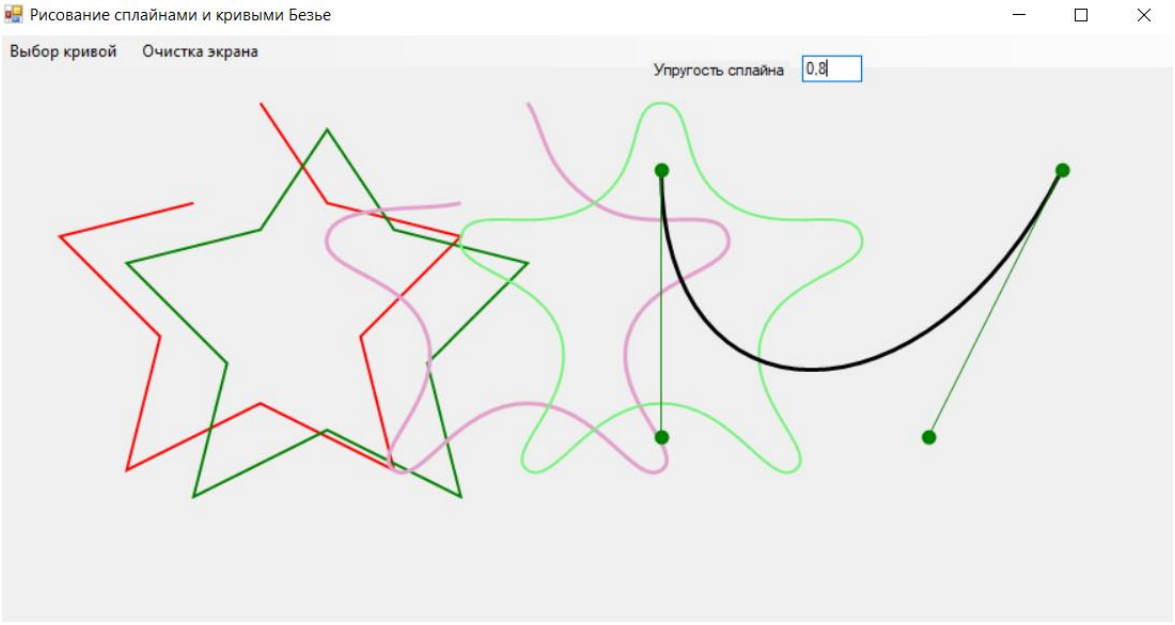
```
private void незамкнутыйСплайнToolStripMenuItem_Click(object sender, EventArgs e)
{
    Pen pen = new Pen(RandomColor(), 3f);
    // Задание точек, определяющих кривую
    Point[] pm = star5(400, 200, 25);
    float elasticity = Convert.ToSingle(textBox1.Text);
    // Рисование незамкнутой кривой через 10 точек сплайном
    g.DrawCurve(pen, pm, elasticity);

}
```

7) Опишите самостоятельно код обработчика события подменю **Замкнутый сплайн.** Координаты подберите самостоятельно
8) Опишите код обработчика события подменю **Одна кривая Безье**

```
private void однаКриваяБезьеToolStripMenuItem_Click(object sender, EventArgs e)
{
    // кривая Безье: 4 точки (x,y) - старт,
    // управление1, управление2, end
    Point start = new Point(500, 100);
    Point control1 = new Point(500, 300);
    Point control2 = new Point(700, 300);
    Point finish = new Point(800, 100);
    g.DrawBezier(new Pen(Color.Black, 3), start, control1, control2, finish);  // это кривая Безье
    //далее - для наглядности показаны четыре параметра (a,b,c,d)
    SolidBrush br = new SolidBrush(Color.Green);
    Pen pen = new Pen(Color.Green, 1);
    g.FillEllipse(br, start.X - 5, start.Y - 5, 11, 11);
    g.FillEllipse(br, control1.X - 5, control1.Y - 5, 11, 11);
    g.FillEllipse(br, control2.X - 5, control2.Y - 5, 11, 11);
    g.FillEllipse(br, finish.X - 5, finish.Y - 5, 11, 11);
    g.DrawLine(pen, start, control1);
    g.DrawLine(pen, finish, control2);
}
```

Образцы построения:

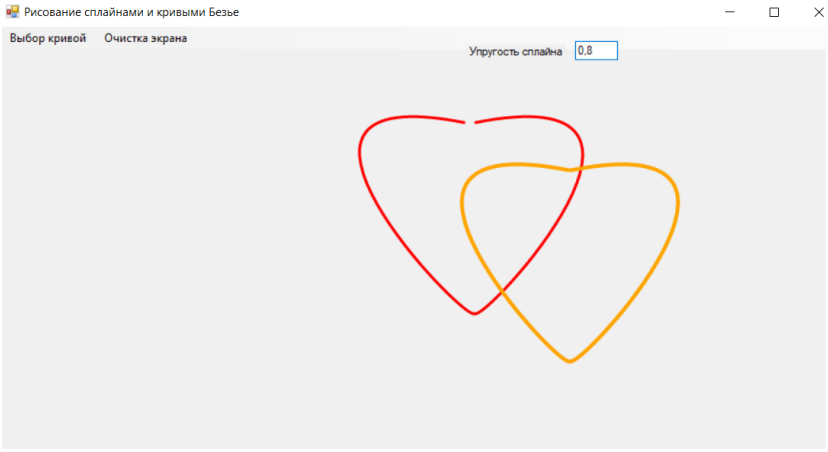


9) Опишите код обработчика события подменю **Незамкнутая кривая Безье**

```
private void незамкнутаяКриваяБезьеToolStripMenuItem_Click(object sender, EventArgs e)
{
    // Зададим точки для 2-х кривых Безье - всего 7.
    Point start = new Point(500, 100);
    Point control1 = new Point(750, 50);
    Point control2 = new Point(510, 310);
    Point end1 = new Point(500, 300);
    Point control3 = new Point(490, 310);
    Point control4 = new Point(240, 50);
    Point end2 = new Point(490, 100);
    Point[] bezierPoints =
    {
    start, control1, control2, end1, control3, control4, end2
    };
    // Рисуем
    g.DrawBeziers(new Pen(Color.Red, 3), bezierPoints);
}
```

10) Опишите код обработчика события подменю **Замкнутая кривая Безье.** Координаты подберите самостоятельно

✓ Протестируем программу. При необходимости откорректируйте код.



11) Опишите код обработчика события меню **Очистка экрана.**

Задание 4. (проект **pr16_3.4_Фамилия** типа Windows Forms) Выполните построение графиков с помощью компонента отображения графической информации **Chart** и построение графиков с использованием **Graphics** функций по заданному варианту.

При реализации программы предусмотреть ввод данных пользователя средствами диалогового окна.

Вариант 1. Исходные данные: $f_1(x) = 3x^5 - ctgx^3;\ f_2(x) = ln(sin4x+1)^2;$ $f_3(x) = \sqrt[3]{2x^2+x^4+1}.$

Вариант.2 Исходные данные: $f_1(x) = x^4 + 2x^3 - x;\ f_2(x) = e^{-x} + \sqrt[4]{x};$ $f_3(x) = \ln(x^3 + x^2).$

Вариант 3. Исходные данные: $f_1(x) = \frac{(3x-1)^2}{x^5};\ f_2(x) = ln^2|\sqrt{x+5}|;$ $f_3(x) = \cos(\sqrt{1+x^2}).$

Вариант 4. Исходные данные: $f_1(x) = x^2 + \sin(7x);\ f_2(x) = |x^3 + 10^x|;$ $f_3(x) = \sqrt[7]{2x^4+x^2+1}.$

Вариант 5. Исходные данные: $f_1(x) = |x|^{2x+1};$ $f_2(x) = \sin x^2; f_3(x) = ln^2|x| + \sqrt{x}.$

Вариант 6. Исходные данные: $f_1(x) = sin^2x^3;$ $f_2(x) = \sqrt[5]{6x-x^2+1}; f_3(x) = 2\sin(x-e^{-x}).$

Вариант 7. Исходные данные: $f_1(x) = 2xe^{-x};$ $f_2(x) = (x-1)^3 + cos(x^3); f_3(x) = 2\sqrt{x^3}\sin(x^3).$

Вариант 8. Исходные данные: $f_1(x) = \ln(x^2+5);\ f_2(x) = \sin(e^x+2);$ $f_3(x) = \text{tg}(5x+1).$

Вариант 9. Исходные данные: $f_1(x) = 2\sqrt{|x^3|}\sin(x^3);$ $f_2(x) = (x+1)^2 cosx^3; f_3(x) = \sqrt{x^4+2}+\sin x^2.$

Вариант 10. Исходные данные: $f_1(x) = \cos x + x^3;\ f_2(x) = \sqrt{x^3}sinx;$ $f_3(x) = 8 + \cos(3x).$

Вариант 23. Исходные данные: $f_1(x) = \cos x + x^3;\ f_2(x) = \sqrt{x^3}sinx;$ $f_3(x) = e^{-3x} + \sqrt[3]{3x^2+1}.$

Вариан 24. Исходные данные: $f_1(x) = x^2 + \sin(7x);\ f_2(x) = |x^3 + 10^x|;$ $f_3(x) = ln^2(x) + \sqrt{x}.$

Вариант 25. Исходные данные: $f_1(x) = |x|^{2x+1}$ $f_2(x) = \sqrt[5]{6x-x^2+1}; f_3(x) = 2\sin(x-e^{-x}).$

Вариант 11. Исходные данные: $f_1(x) = x\sin(x+4);\ f_2(x) = \ln(4x^2+1);$ $f_3(x) = \ln\sqrt[5]{5+x^2}.$

Вариант 12. Исходные данные: $f_1(x) = x^4 + 2x^3 - x;\ f_2(x) = 1.3\sqrt{4+x^2};$ $f_3(x) = |x+1|^x.$

Вариант 13. Исходные данные: $f_1(x) = |x|^5 ctg|x|;\ f_2(x) = \ln(x^2+1);$ $f_3(x) = e^{-2x} - \sqrt[3]{|x+1|}.$

Вариант 14. Исходные данные: $f_1(x) = |x|\sin(3x);\ f_2(x) = x^3\cos(x+2);$ $f_3(x) = \sin x^2 + x^{0.25}.$

Вариант 15. Исходные данные: $f_1(x) = x^5 ctg(2x^3);$ $f_2(x) = \frac{5}{tg(2x+3)+1}; f_3(x) = \text{tg}(x^2+1)e^{-x}/$

Вариант 16. Исходные данные: $f_1(x) = xsin(x-1);$ $f_2(x) = (x-1)^3 + cosx^3; f_3(x) = \sqrt{|x|^3}sinx^3.$

Вариант 17. Исходные данные: $f_1(x) = e^{-3x} + cosx;\ f_2(x) = sin^3x^4$ $f_3(x) = e^{-3x} + \sqrt[3]{3x^2+1}.$

Вариант 18. Исходные данные: $f_1(x) = \sqrt{sin^2x + cos^4x};$ $f_2(x) = ln^2(x) + \sqrt{x}; f_3(x) = tg^2(x) + \sqrt{x}.$

Вариант 19. Исходные данные: Исходные данные: $f_1(x) = sin^2x^3;$ $f_2(x) = \sqrt[5]{6x-x^2+1}; f_3(x) = 2\sin(x-e^{-x}).$

Вариант 20. Исходные данные: $f_1(x) = \sqrt[3]{|x|+2} - 1;\ f_2(x) = sin(x^3) + x^{0.5};$ $f_3(x) = ln^2(x) + \sqrt{x}.$

Вариант 21. Исходные данные: $f_1(x) = 2\sqrt{|x^3|}\sin(x^3);$ $f_2(x) = ln^2(x) + \sqrt{x}; f_3(x) = tg^2(x) + \sqrt{x}.$

Вариант 22. Исходные данные: $f_1(x) = \ln(x^2+5);\ f_2(x) = \sin(e^x+2);$ $f_3(x) = \sqrt[3]{2x^2+x^4+1}.$

Контрольные вопросы:

1) Для чего предназначен LinearGradientBrush?
2) Как выполняются операции рисования при двойной буферизации? Приведите пример кода использования двойного буфера.
3) Что такое сплайн?
4) Что необходимо для полного определения кривой Безье?
5) Приведите примеры методов построения замкнутого и незамкнутого слайна и кривой Безье.

Класс - Graphics

Инкапсулирует поверхность рисования GDI+. Этот класс не наследуется.

Методов в этом классе огромное количество, поэтому рассмотрим некоторые из них:

Имя	Описание
AddMetafileComment	Добавляет комментарий к текущему объекту Metafile.
BeginContainer()	Сохраняет графический контейнер, содержащий текущее состояние данного объекта Graphics, а затем открывает и использует новый графический контейнер.
BeginContainer(Rectangle, Rectangle, GraphicsUnit)	Сохраняет графический контейнер, содержащий текущее состояние данного объекта Graphics, а также открывает и использует новый графический контейнер с указанным преобразованием масштаба.
BeginContainer(RectangleF, RectangleF, GraphicsUnit)	Сохраняет графический контейнер, содержащий текущее состояние данного объекта Graphics, а также открывает и использует новый графический контейнер с указанным преобразованием масштаба.
Clear	Очищает всю поверхность рисования и выполняет заливку поверхности указанным цветом фона.
CopyFromScreen(Point, Point, Size)	Выполняет передачу данных о цвете, соответствующих прямоугольной области пикселей, блоками битов с экрана на поверхность рисования объекта Graphics.
CopyFromScreen(Point, Point, Size, CopyPixelOperation)	Выполняет передачу данных о цвете, соответствующих прямоугольной области пикселей, блоками битов с экрана на поверхность рисования объекта Graphics.
CopyFromScreen(Int32, Int32, Int32, Int32, Size)	Выполняет передачу данных о цвете, соответствующих прямоугольной области пикселей, блоками битов с экрана на поверхность рисования объекта Graphics.
Dispose	Освобождает все ресурсы, используемые данным объектом Graphics.
DrawArc(Pen, Rectangle, Single, Single)	Рисует дугу, которая является частью эллипса, заданного структурой Rectangle.
DrawArc(Pen, Int32, Int32, Int32, Int32, Int32, Int32)	Рисует дугу, которая является частью эллипса, заданного парой координат, шириной и высотой.
DrawArc(Pen, Single, Single, Single, Single, Single, Single)	Рисует дугу, которая является частью эллипса, заданного парой координат, шириной и высотой.
DrawBezier(Pen, Point, Point, Point, Point)	Рисует кривую Безье, определяемую четырьмя структурами Point.
DrawLine(Pen, Point, Point)	Проводит линию, соединяющую две структуры Point.
DrawLine(Pen, Int32, Int32, Int32, Int32)	Проводит линию, соединяющую две точки, задаваемые парами координат.
DrawLine(Pen, Single, Single, Single, Single)	Проводит линию, соединяющую две точки, задаваемые парами координат.
DrawLines(Pen, Point[])	Рисует набор сегментов линий, которые соединяют массив структур Point.
DrawPath	Рисует объект GraphicsPath.
DrawPie(Pen, Rectangle, Single, Single)	Рисует сектор, который определяется эллипсом, заданным структурой Rectangle и двумя радиальными линиями.
DrawPie(Pen, Int32, Int32, Int32, Int32, Int32, Int32)	Рисует сектор, определяемый эллипсом, который задан парой координат, шириной, высотой и двумя радиальными линиями.

DrawPie(Pen, Single, Single, Single, Single, Single, Single)	Рисует сектор, определяемый эллипсом, который задан парой координат, шириной, высотой и двумя радиальными линиями.
DrawPolygon(Pen, Point[])	Рисует многоугольник, определяемый массивом структур Point.
DrawRectangle(Pen, Rectangle)	Рисует прямоугольник, определяемый структурой Rectangle.
DrawRectangle(Pen, Int32, Int32, Int32, Int32)	Рисует прямоугольник, который определен парой координат, шириной и высотой.
DrawRectangle(Pen, Single, Single, Single, Single)	Рисует прямоугольник, который определен парой координат, шириной и высотой.
DrawRectangles(Pen, Rectangle[])	Рисует набор прямоугольников, определяемых структурами Rectangle.
DrawString(String, Font, Brush, PointF)	Создает указываемую текстовую строку в заданном месте с помощью определяемых объектов Brush и Font.
DrawString(String, Font, Brush, PointF, StringFormat)	Рисует заданную текстовую строку в заданном месте с помощью определяемых объектов Brush и Font, используя атрибуты форматирования заданного формата StringFormat.
DrawString(String, Font, Brush, Single, Single)	Создает указываемую текстовую строку в заданном месте с помощью определяемых объектов Brush и Font.
Equals(Object)	Определяет, равен ли заданный объект текущему объекту. (Унаследовано от Object.)
ExcludeClip(Rectangle)	Обновляет вырезанную область данного объекта Graphics, чтобы исключить из нее часть, определяемую структурой Rectangle.
ExcludeClip(Region)	Обновляет вырезанную область данного объекта Graphics, чтобы исключить из нее часть, определяемую структурой Region.
FillClosedCurve(Brush, Point[])	Заполняет внутреннюю часть замкнутой фундаментальной кривой, определяемой массивом структур Point.
FillClosedCurve(Brush, Point[], FillMode)	Заполняет внутреннюю часть замкнутой фундаментальной сплайновой кривой, определяемой массивом структур Point, используя указанный режим заливки.
FillClosedCurve(Brush, Point[], FillMode, Single)	Заполняет внутреннюю часть замкнутой фундаментальной кривой, определяемой массивом структур Point, используя указанные режим заливки и натяжение.
FillEllipse(Brush, Rectangle)	Заполняет внутреннюю часть эллипса, определяемого ограничивающим прямоугольником, который задан структурой Rectangle.
FillEllipse(Brush, Int32, Int32, Int32, Int32)	Заполняет внутреннюю часть эллипса, определяемого ограничивающим прямоугольником, заданным с помощью пары координат, ширины и высоты.
FillEllipse(Brush, Single, Single, Single, Single)	Заполняет внутреннюю часть эллипса, определяемого ограничивающим прямоугольником, заданным с помощью пары координат, ширины и высоты.
FillPath	Заполняет внутреннюю часть объекта GraphicsPath.
FillPie(Brush, Rectangle, Single, Single)	Заполняет внутреннюю часть сектора, определяемого эллипсом, который задан структурой RectangleF, и двумя радиальными линиями.
FillPie(Brush, Int32, Int32, Int32, Int32, Int32, Int32)	Заполняет внутреннюю часть сектора, определяемого эллипсом, который задан парой координат, шириной, высотой и двумя радиальными линиями.

FillPie(Brush, Single, Single, Single, Single, Single, Single)	Заполняет внутреннюю часть сектора, определяемого эллипсом, который задан парой координат, шириной, высотой и двумя радиальными линиями.
FillPolygon(Brush, Point[])	Заполняет внутреннюю часть многоугольника, определяемого массивом точек, заданных структурами Point.
FillPolygon(Brush, Point[], FillMode)	Заполняет внутреннюю часть многоугольника, определенного массивом точек, заданных структурами Point, используя указанный режим заливки.
FillRectangle(Brush, Rectangle)	Заполняет внутреннюю часть прямоугольника, определяемого структурой Rectangle.
FillRectangle(Brush, Int32, Int32, Int32, Int32)	Заполняет внутреннюю часть прямоугольника, который определен парой координат, шириной и высотой.
FillRectangle(Brush, Single, Single, Single, Single)	Заполняет внутреннюю часть прямоугольника, который определен парой координат, шириной и высотой.
FillRectangles(Brush, Rectangle[])	Заполняет внутреннюю часть набора прямоугольников, определяемых структурами Rectangle.
FillRegion	Заполняет внутреннюю часть объекта Region.
Flush()	Вызывает принудительное выполнение всех отложенных графических операций и немедленно возвращается, не дожидаясь их окончания.
Flush(FlushIntention)	Вызывает принудительное выполнение всех отложенных графических операций. При этом в соответствии с настройкой метод дожидается или не дожидается окончания операций для возврата.