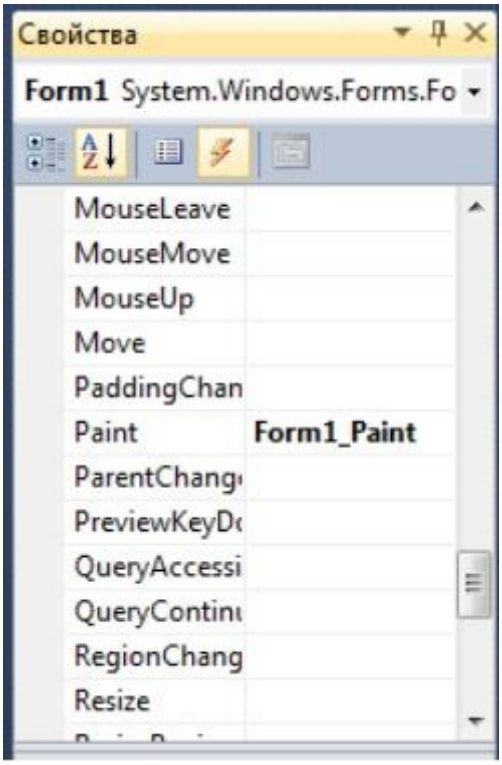
Необходимо изучить возможности Visual Studio по созданию простейших графических изображений. Написать и отладить программу построения на экране различных графических примитивов.

Для форм в C# предусмотрен способ, позволяющий приложению при необходимости перерисовывать окно формы в любой момент времени. Когда вся клиентская область окна формы или часть этой области требует перерисовки, форме передается событие Paint. Все, что требуется от программиста, – это создать обработчик данного события, наполнив его необходимой функциональностью.



Создание обработчика события Paint

Для рисования линий и фигур, отображения текста, вывода изображений и т. д. нужно использовать объект Graphics. Этот объект предоставляет поверхность рисования и используется для создания графических изображений. Ниже представлены два этапа работы с графикой.

- Создание или получение объекта Graphics.
- Использование объекта Graphics для рисования.

Существует несколько способов создания объектов Graphics. Одним из самых используемых является получение ссылки на объект Graphics через объект PaintEventArgs при обработке события Paint формы или элемента управления:

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
Graphics g = e.Graphics;
// Далее вставляется код рисования
}
```

Имена большого количества методов, определенных в классе Graphics, начинаются с префикса Draw* и Fill*. Первые из них предназначены для рисования текста, линий и незакрашенных фигур (таких, например, как прямоугольные рамки), а вторые – для рисования закрашенных геометрических фигур. Ниже рассматривается применение наиболее часто используемых методов, более полную информацию можно найти в документации по Visual Studio.

Метод DrawLine рисует линию, соединяющую две точки с заданными координатами. У метода есть несколько перегруженных версий:

```
public void DrawLine(Pen, Point, Point);
public void DrawLine(Pen, PointF, PointF);
public void DrawLine(Pen, int, int, int, int);
public void DrawLine(Pen, float, float, float, float);
```

Первый параметр задает инструмент для рисования линии – перо.

Перья создаются как объекты класса Pen, например:

```
Pen p = new Pen(Brushes.Black, 2);
```

Здесь создается черное перо толщиной 2 пиксела. При создании пера можно выбрать его цвет, толщину и тип линии, а также другие атрибуты.

Остальные параметры перегруженных методов DrawLine задают координаты соединяемых точек. Эти координаты могут быть заданы как объекты класса Point и PointF, а также в виде целых чисел и чисел с плавающей десятичной точкой.

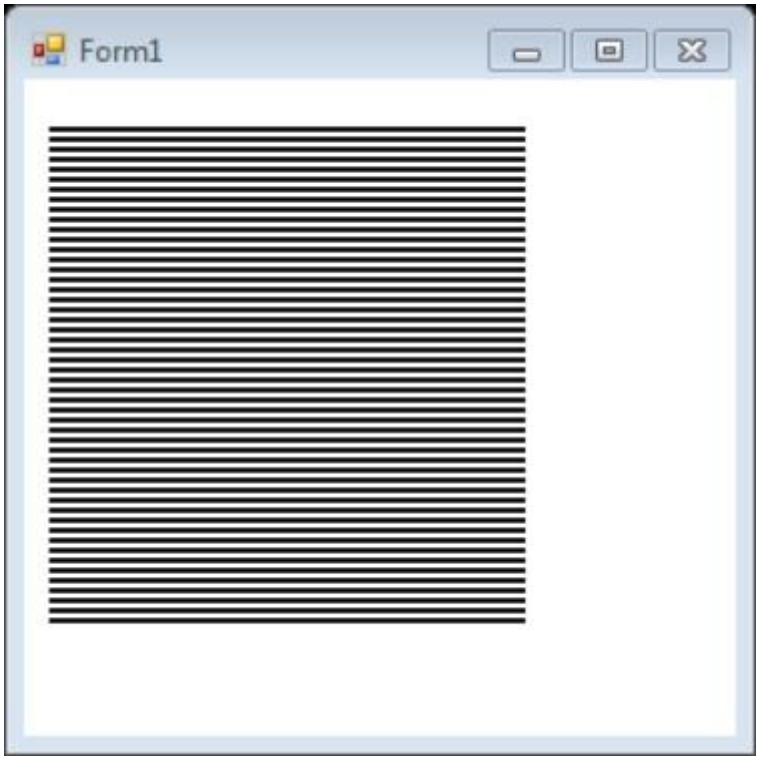
В классах Point и PointF определены свойства X и Y, задающие, соответственно, координаты точки по горизонтальной и вертикальной оси. При этом в классе Point эти свойства имеют целочисленные значения, а в классе PointF – значения с плавающей десятичной точкой.

Третий и четвертый варианты метода DrawLine позволяют задавать координаты соединяемых точек в виде двух пар чисел. Первая пара определяет координаты первой точки по горизонтальной и вертикальной оси, а вторая – координаты второй точки по этим же осям. Разница между третьим и четвертым методом заключается в использовании координат различных типов (целочисленных int и с плавающей десятичной точкой float).

Чтобы испытать метод DrawLine в работе, создайте приложение DrawLineApp (аналогично тому, как Вы создавали предыдущее приложение). В этом приложении создайте следующий обработчик события Paint:

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
Graphics g = e.Graphics;
g.Clear(Color.White);
for (int i = 0; i < 50; i++)
    g.DrawLine(new Pen(Brushes.Black, 2),10, 4 * i + 20, 200, 4 * i + 20);
}
```

Здесь мы вызываем метод DrawLine в цикле, рисуя 50 горизонтальных линий.



Пример использования DrawLine

Вызвав один раз метод DrawLines, можно нарисовать сразу несколько прямых линий, соединенных между собой. Иными словами, метод DrawLines позволяет соединить между собой несколько точек. Координаты этих точек по горизонтальной и вертикальной осям передаются методу через массив класса Point или PointF

```
public void DrawLines(Pen, Point[]);
public void DrawLines(Pen, PointF[]);
```

Для демонстрации возможностей метода DrawLines создайте приложение. Код будет выглядеть следующим образом:

```
Point[] points = new Point[50];
Pen pen = new Pen(Color.Black, 2);
private void Form1_Paint(object sender, PaintEventArgs e)
{
Graphics g = e.Graphics;
g.DrawLines(pen, points);
}
private void Form1_Load(object sender, EventArgs e)
{
  for (int i = 0; i < 20; i++)
  {
    int xPos;
    if (i % 2 == 0)
    {
      xPos = 10;
    }
    else
    {
      xPos = 400;
    }
    points[i] = new Point(xPos, 10 * i);
  }
}
```

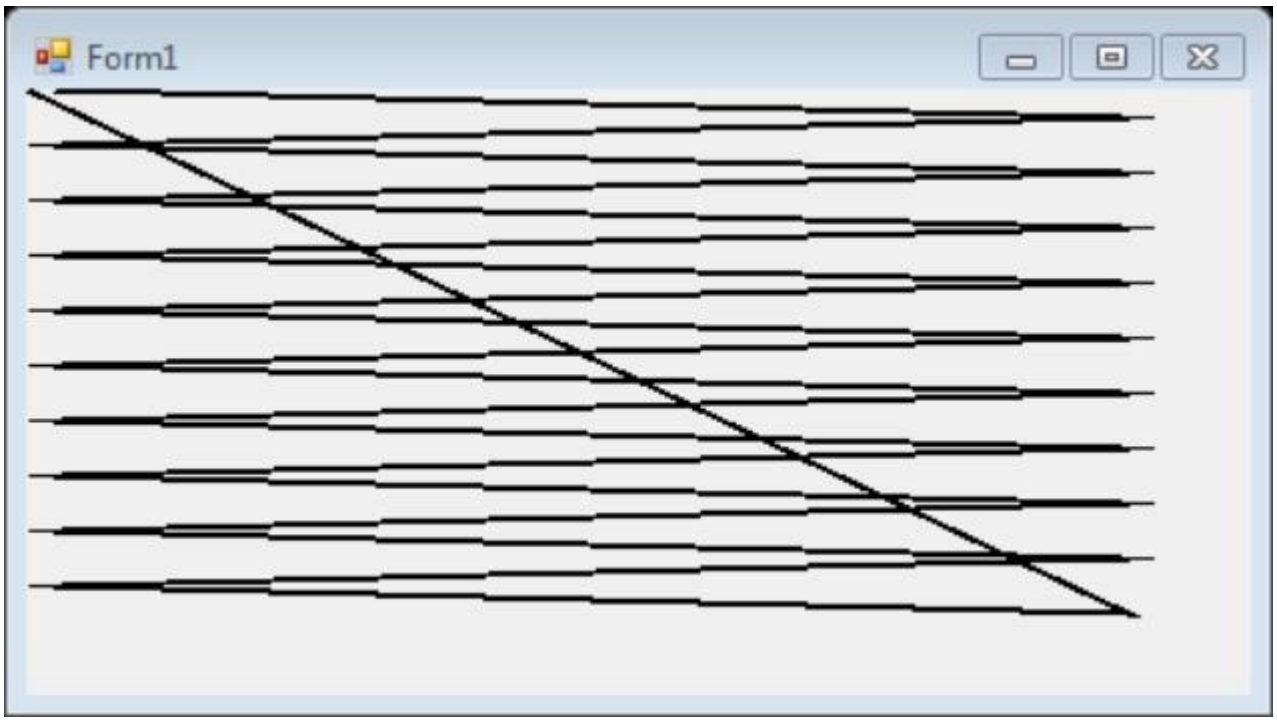
Для прорисовки прямоугольников можно использовать метод DrawRectangle:

```
DrawRectangle(Pen, int, int, int, int);
```

В качестве первого параметра передается перо класса Pen. Остальные параметры задают расположение и размеры прямоугольника.

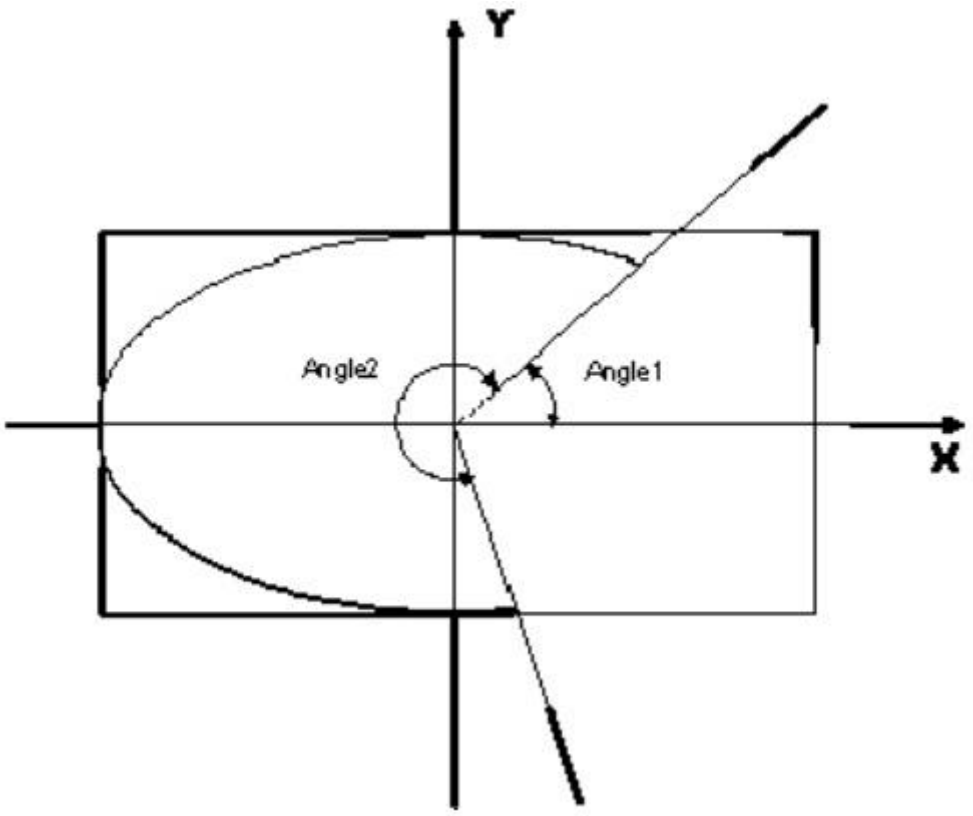
Для прорисовки многоугольников можно использовать следующий метод:

```
DrawPolygon(Pen, Point[]);
```



Пример использования массива точек

Метод DrawEllipse рисует эллипс, вписанный в прямоугольную область, расположение и размеры которой передаются ему в качестве параметров. При помощи метода DrawArc программа может нарисовать сегмент эллипса. Сегмент задается при помощи координат прямоугольной области, в которую вписан эллипс, а также двух углов, отсчитываемых в направлении против часовой стрелки. Первый угол Angle1 задает расположение одного конца сегмента, а второй Angle2 – расположение другого конца сегмента.



Углы и прямоугольник, задающие сегмент эллипса

В классе Graphics определен ряд методов, предназначенных для рисования закрашенных фигур. Имена некоторых из этих методов, имеющих префикс Fill*:

- FillRectangle (рисование закрашенного прямоугольника),
- FillRectangles (рисование множества закрашенных многоугольников),
- FillPolygon (рисование закрашенного многоугольника),
- FillEllipse (рисование закрашенного эллипса),
- FillPie (рисование закрашенного сегмента эллипса),
- FillClosedCurve (рисование закрашенного сплайна),
- FillRegion (рисование закрашенной области типа Region).

Есть два отличия методов с префиксом Fill* от одноименных методов с префиксом Draw*. Прежде всего, методы с префиксом Fill* рисуют закрашенные фигуры, а методы с префиксом Draw* – незакрашенные. Кроме этого, в качестве первого параметра методам с префиксом Fill*

передается не перо класса Pen, а кисть класса SolidBrush. Ниже приведем пример, выводящий закрашенный прямоугольник:

```
SolidBrush B = new SolidBrush(Color.DeepPink);
g.FillRectangle(B, 0, 0, 100, 100);
```

Индивидуальное задание

Изучите с помощью справки MSDN(https://learn.microsoft.com/ru-ru/dotnet/api/system.drawing?view=net-7.0&redirectedfrom=MSDN) методы и свойства классов Graphics, Color, Pen и SolidBrush. Создайте собственное приложение – выводящий на форму рисунок, состоящий из различных объектов (линий, многоугольников, эллипсов, прямоугольников и пр.), не закрашенных и закрашенных полностью. Используйте разные цвета и стили линий (сплошные, штриховые, штрих-пунктирные).