### ЛАБОРАТОРНАЯ РАБОТА №6.

**ГРАФИКИ ФУНКЦИЙ**

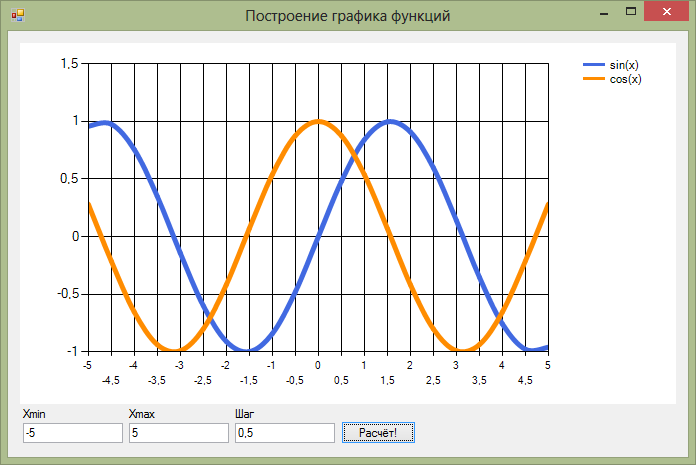
***Цель лабораторной работы:*** изучить возможности построения графиков с помощью элемента управления Сhart. Написать и отладить программу построения на экране графика заданной функции.

#### 9.1. Как строится график с помощью элемента управления Chart

Обычно результаты расчетов представляются в виде графиков и диаграмм. Библиотека .NET Framework имеет мощный элемент управ- ления Chart для отображения на экране графической информации (рис. 9.1).

Построение графика (диаграммы) производится после вычисления таблицы значений функции *y=f(x)* на интервале [Xmin, Xmax] с задан- ным шагом. Полученная таблица передается в специальный массив Points объекта Series элемента управления Сhart с помощью метода DataBindXY. Элемент управления Chart осуществляет всю работу по отображению графиков: строит и размечает оси, рисует координатную сетку, подписывает название осей и самого графика, отображает пере- данную таблицу в виде всевозможных графиков или диаграмм. В эле- менте управления Сhart можно настроить толщину, стиль и цвет линий, параметры шрифта подписей, шаги разметки координатной сетки и многое другое. В процессе работы программы изменение параметров

возможно через обращение к соответствующим свойствам элемента управления Chart. Так, например, свойство AxisX содержит значение максимального предела нижней оси графика и при его изменении во время работы программы автоматически изменяется изображение гра- фика.



*Рис 8.1. Окно программы с элементом управления*

#### Пример написания программы

Задание: составить программу, отображающую графики функций *sin(x)* и *cos(x)* на интервале [Xmin, Xmax]. Предусмотреть возможность изменения разметки координатных осей, а также шага построения таб- лицы.

Прежде всего, следует поместить на форму сам элемент управления

Chart. Он располагается в панели элементов в разделе *Данные*.

Список графиков хранится в свойстве Series, который можно из- менить, выбрав соответствующий пункт в окне свойств. Поскольку на одном поле требуется вывести два отдельных графика функций, нужно добавить ещё один элемент. Оба элемента, и существующий и добав- ленный, нужно соответствующим образом настроить: изменить тип диа-

граммы ChartType на Spline. Здесь же можно изменить подписи к гра- фикам с абстрактных *Series1* и *Series2* на *sin(x)* и *cos(x)* – за это отвечает свойство Legend. Наконец, с помощью свойства BorderWidth можно сде- лать линию графика потолще, а затем поменять цвет линии с помощью свойства Color.

Ниже приведён текст обработчика нажатия кнопки «Расчёт!», ко- торый выполняет все требуемые настройки и расчёты и отображает графики функций:

private void buttonCalc\_Click(object sender, EventArgs e)

{

// Считываем с формы требуемые значения double Xmin = double.Parse(textBoxXmin.Text); double Xmax = double.Parse(textBoxXmax.Text); double Step = double.Parse(textBoxStep.Text);

// Количество точек графика

int count = (int)Math.Ceiling((Xmax - Xmin) / Step)

+ 1;

// Массив значений X – общий для обоих графиков double[] x = new double[count];

// Два массива Y – по одному для каждого графика double[] y1 = new double[count];

double[] y2 = new double[count];

// Расчитываем точки для графиков функции for (int i = 0; i < count; i++)

{

// Вычисляем значение X x[i] = Xmin + Step \* i;

// Вычисляем значение функций в точке X y1[i] = Math.Sin(x[i]);

y2[i] = Math.Cos(x[i]);

}

// Настраиваем оси графика chart1.ChartAreas[0].AxisX.Minimum = Xmin; chart1.ChartAreas[0].AxisX.Maximum = Xmax;

// Определяем шаг сетки chart1.ChartAreas[0].AxisX.MajorGrid.Interval = Step;

// Добавляем вычисленные значения в графики chart1.Series[0].Points.DataBindXY(x, y1); chart1.Series[1].Points.DataBindXY(x, y2);

}

#### Выполнение индивидуального задания

Постройте графики функций для соответствующих вариантов из лабораторной работы №2. Таблицу данных получить путём изменения параметра X с шагом h. Самостоятельно выбрать удобные параметры настройки.

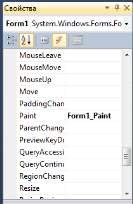
### ЛАБОРАТОРНАЯ РАБОТА №7.

**КОМПЬЮТЕРНАЯ ГРАФИКА**

***Цель лабораторной работы:*** изучить возможности Visual Studio по создание простейших графических изображений. Написать и отла- дить программу построения на экране различных графических прими- тивов.

* 1. **Событие Paint**

Для форм в C# предусмотрен способ, позволяющий приложению при необходимости перерисовывать окно формы в любой момент вре- мени. Когда вся клиентская область окна формы или часть этой области требует перерисовки, форме передается событие Paint. Все, что требу- ется от программиста, это создать обработчик данного события (рис. 10.1), наполнив его необходимой функциональностью.



*Рис. 10.1. Создание обработчика события Paint*

* 1. **Объект Graphics для рисования**

Для рисования линий и фигур, отображение текста, вывода изоб- ражений и т. д. нужно использовать объект Graphics. Этот объект предоставляет поверхность рисования и используется для создания гра- фических изображений. Ниже представлены два этапа работы с графи- кой.

* + Создание или получение объекта Graphics
  + Использование объекта Graphics для рисования

Существует несколько способов создания объектов Graphics. Од- ним 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

}

* 1. **Методы и свойства класса Graphics**

Имена большого количества методов, определенных в классе Graphics, начинается с префикса Draw\* и Fill\*. Первые из них предна- значены для рисования текста, линий и не закрашенных фигур (таких, например, как прямоугольные рамки), а вторые – для рисования закра- шенных 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 задан- ными 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 создаётся черное перо толщиной 2 пиксела. При создании пера можно выбрать его цвет, толщину и тип линии, а также другие ат- рибуты.

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, со- ответственно, 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 вариант метода DrawLine позволяет задавать координаты соединяемых точек в виде двух пар чисел. Первая пара определяет координаты первой точки по горизонтальной и вертикаль- ной оси, а вторая – координаты второй точки по этим же осям. Разница между третьим и четвертым методом заключается в использовании ко- ординат различных типов (целочисленных int и с плавающей десятич- ной точкой float).

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

private void Form1\_Paint(object sender, PaintEventArgs e)

{

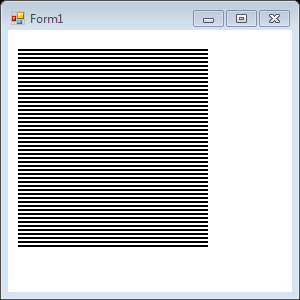
Graphics g = e.Graphics; g.Clear(Color.White);

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 горизонталь- ных линий (рис. 10.2).



*Рис. 10.2. Пример использования DrawLine*

Вызвав один раз метод DrawLines, можно нарисовать сразу не- сколько прямых линий, соединенных между собой. Иными словами, ме- тод DrawLines позволяет соединить между собой несколько точек. Ко- ординаты этих точек по горизонтальной и вертикальной оси передаются методу через массив класса Point или PointF:

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

Для демонстрации возможностей метода DrawLines создайте при- ложение. 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)

{

}

else

{

}

xPos = 10;

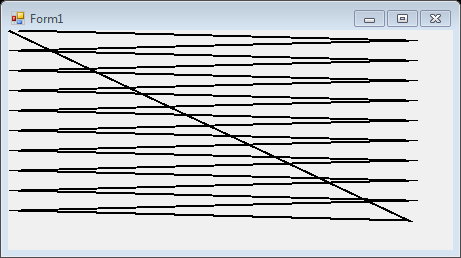
xPos = 400;

points[i] = new Point(xPos, 10 \* i);

}

}

Результат работы программы приведен на рис. 10.3.



*Рис. 10.3. Пример использования массива точек*

Для прорисовки прямоугольников можно использовать метод

DrawRectangle:

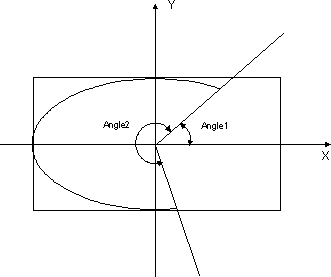
DrawRectangle(Pen, int, int, int, int);

В качестве первого параметра передается перо класса Pen. Осталь- ные параметры задают расположение и размеры прямоугольника.

Для прорисовки многоугольников можно использовать следующий метод:

DrawPolygon(Pen, Point[]);

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 прямоуголь- ной области, в которую вписан эллипс, а также двух углов, отсчитывае- мых в направлении против часовой стрелки. Первый угол Angle1 задает расположение одного конца сегмента, а второй Angle2 – расположение другого конца сегмента (рис. 10.4).



*Рис. 10.4. 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 ме- тодов с префиксом Draw\*. Прежде всего, методы с префиксом Fill\* ри- суют закрашенные фигуры, а методы с префиксом Draw\* – не закрашен- ные. 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 MSDN1 методы и свойства классов Graphics, Color, Pen и SolidBrush. Создайте собственное приложение выводящий на форму рисунок, состоящий из различных объектов (ли- ний, многоугольников, эллипсов, прямоугольников и пр.), не закрашен- ных и закрашенных полностью. Используйте разные цвета и стили ли- ний (сплошные, штриховые, штрих-пунктирные).

### ЛАБОРАТОРНАЯ РАБОТА №8.

**АНИМАЦИЯ**

***Цель лабораторной работы:*** изучить возможности Visual Studio по создание простейшей анимации. Написать и отладить программу, выводящую на экран анимационное изображение.

#### Работа с таймером

Класс для работы с таймером Timer формирует в приложении по- вторяющиеся события. События повторяются с периодичностью, ука- занной в миллисекундах в свойстве Interval. Установка свойства Enabled в значение true запускает таймер. Каждый тик таймера порож- дает событие Tick, обработчик которого обычно и создают в приложе- нии. В этом обработчике могут изменяться какие-либо величины, и вы- зываться принудительная перерисовка окна. Для создания анимации весь код, рисующий что-либо на форме, должен находиться в обработ- чике события Paint.

#### Создание анимации

Для создания простой анимации достаточно использовать таймер, при тике которого будут изменяться параметры изображения (например, координаты концов отрезка) и вызываться обработчик события Paint

1 [http://msdn.microsoft.com/ru-ru/library/system.drawing(v=vs.100).aspx](http://msdn.microsoft.com/ru-ru/library/system.drawing(v%3Dvs.100).aspx)

для рисования по новым параметрам. При таком подходе не надо забо- титься об удалении старого изображения, ведь оно создается в окне за- ново.

В качестве примера рассмотрим код анимации секундной стрелки часов:

// Глобальные переменные private int x1, y1, x2, y2, r; private double a;

private Pen pen = new Pen(Color.DarkRed, 2);

// Перерисовка формы

private void Form1\_Paint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

// Рисуем секундную стрелку g.DrawLine(pen, x1, y1, x2, y2);

}

// Действия при загрузке формы

private void Form1\_Load(object sender, EventArgs e)

{

r = 150; // Радиус стрелки

a = 0; // Угол поворота стрелки

// Определяем центр формы – начало стрелки x1 = ClientSize.Width / 2;

y1 = ClientSize.Height / 2;

// Конец стрелки

x2 = x1 + (int)(r \* Math.Cos(a)); y2 = y1 – (int)(r \* Math.Sin(a));

}

// Действия при очередном «тике» таймера

private void timer1\_Tick(object sender, EventArgs e)

{

a -= 0.1; // Уменьшаем угол на 0,1 радиану

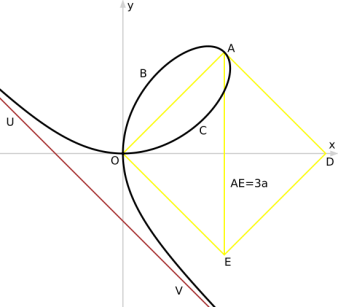
// Новые координаты конца стрелки x2 = x1 + (int)(r \* Math.Cos(a)); y2 = y1 – (int)(r \* Math.Sin(a));

// Принудительный вызов события Paint Invalidate();

}

#### Движение по траектории

Движение по траектории реализуется аналогично выше рассмот- ренному примеру. Для реализации движения по прямой нужно увеличи- вать переменные, являющиеся узловыми точками, на определённые константы: в приведённом выше примере это переменные x2 и y2. Для задания более сложной траектории можно использовать различные па- раметрические кривые.

В случае движения на плоскости обычно изменению подвергается один параметр. Рассмотрим пример реализации движения окружности по *декартову листу*. Декартов лист – это плоская кривая третьего по- рядка, удовлетворяющая уравнению в прямоугольной системе *x3+y3=3·a·x·y*. Параметр *3·a* определяется как диагональ квадрата, сто- рона которого равна наибольшей хорде петли.

*Рис. 11.1. Декартов лист*

При переходе к параметрическому виду получаем:

,

где *t=tg φ*.

Программная реализация выглядит следующим образом:

private int x1, y1, x2, y2; private double a, t, fi;

private Pen pen = new Pen(Color.DarkRed, 2);

private void Form1\_Load(object sender, EventArgs e)

{

x1 = ClientSize.Width / 2; y1 = ClientSize.Height / 2; a = 150;

fi = -0.5;

t = Math.Tan(fi);

x2 = x1 + (int)((3 \* a \* t) / (1 + t \* t \* t));

y2 = y1 – (int)((3 \* a \* t \* t) / (1 + t \* t \* t));

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics; g.DrawEllipse(pen, x2, y2, 20, 20);

}

private void timer1\_Tick(object sender, EventArgs e)

{

fi += 0.01;

t = Math.Tan(fi);

x2 = x1 + (int)((3 \* a \* t) / (1 + t \* t \* t));

y2 = y1 – (int)((3 \* a \* t \* t) / (1 + t \* t \* t)); Invalidate();

}

Описание ряда интересных кривых для создания траектории дви- жения можно найти в Википедии в статье Циклоидальная кривая2.

#### Индивидуальное задание

1. Создайте программу, показывающую пульсирующее сердце.
2. Создайте приложение, отображающее вращающийся винт са- молета.
3. Разработайте программу анимации двигающегося человечка.
4. Создайте программу, показывающую движение окружности по синусоиде.

2 <https://ru.wikipedia.org/wiki/Циклоидальная_кривая>

1. Создайте приложение, отображающее движение окружности по спирали.
2. Разработайте программу анимации падения снежинки.
3. Создайте программу, показывающую скачущий мячик.
4. Создайте приложение, отображающее движение окружности вдоль границы окна. Учтите возможность изменения размеров окна.
5. Разработайте программу анимации летающего бумеранга.
6. Создайте программу, показывающую падение нескольких звезд одновременно.
7. Создайте приложение, отображающее хаотичное движение звезды в окне.
8. Разработайте программу анимации взлета ракеты. Старт осу- ществляется по нажатию специальной «красной» кнопки.
9. Создайте программу, показывающую движение окружности вдоль многоугольника. Число вершин вводится пользователем до ани- мации.
10. Создайте приложение, отображающее броуновское движение молекулы в окне.
11. Разработайте программу анимации движения планет в Сол- нечной системе.
12. Создайте программу, показывающую движение квадратика по траектории, состоящей из 100 точек, и хранящихся в специальном мас- сиве.
13. Создайте приложение, имитирующие механические часы.
14. Разработайте программу анимации падения несколько листков с дерева. Движение не должно быть линейным.
15. Создайте программу, показывающую движение окружности по спирали с плавно изменяющейся скоростью.
16. Создайте приложение, отображающее движение автомобиля с вращающимися колесами.