

- 11) Дана матрица $F(15,15)$. Вывести номер и среднее арифметическое элементов строки, начинающейся с 1. Если такой строки нет, то вывести сообщение “строки нет”.
- 12) Дана матрица $F(7,7)$. Найти наименьший элемент в каждом столбце. Вывести матрицу и найденные элементы.
- 13) Найти наибольший элемент главной диагонали матрицы $A(15,15)$ и вывести всю строку, в которой он находится.
- 14) Найти наибольшие элементы каждой строки матрицы $Z(16,16)$ и поместить их на главную диагональ. Вывести полученную матрицу.
- 15) Вычислить суммы элементов матрицы $Y(12,12)$ по столбцам и вывести их.
- 16) Найти наибольший элемент матрицы $A(10,10)$ и записать нули в ту строку и столбец, где он находится. Вывести наибольший элемент, исходную и полученную матрицу.
- 17) Дана матрица $R(9,9)$. Найти наименьший элемент в каждой строке и записать его на место первого элемента строки. Вывести исходную и полученную матрицы.
- 18) Определить количество положительных элементов каждой строки матрицы $A(10,20)$ и запомнить их в одномерном массиве N . Массив N вывести.
- 19) Вычислить количество N положительных элементов последнего столбца матрицы $X(5,5)$. Если $N < 3$, то вывести все положительные элементы матрицы, если $N \geq 3$, то вывести сумму элементов главной диагонали матрицы.
- 20) Вычислить и вывести сумму элементов матрицы $A(12,12)$, расположенных над главной диагональю матрицы.

ЛАБОРАТОРНАЯ РАБОТА 8. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ СРЕДСТВ ДЛЯ ОТОБРАЖЕНИЯ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ

Цель лабораторной работы: изучить возможности построения графиков с помощью компонента отображения графической информации **Chart**. Написать и отладить программу построения на экране графика заданной функции.

8.1. Как строится график с помощью компонента Chart

Обычно результаты расчетов представляются в виде графиков и диаграмм. Библиотека .NET Framework имеет мощный элемент управления Chart для отображения на экране графической информации (рис. 8.1).

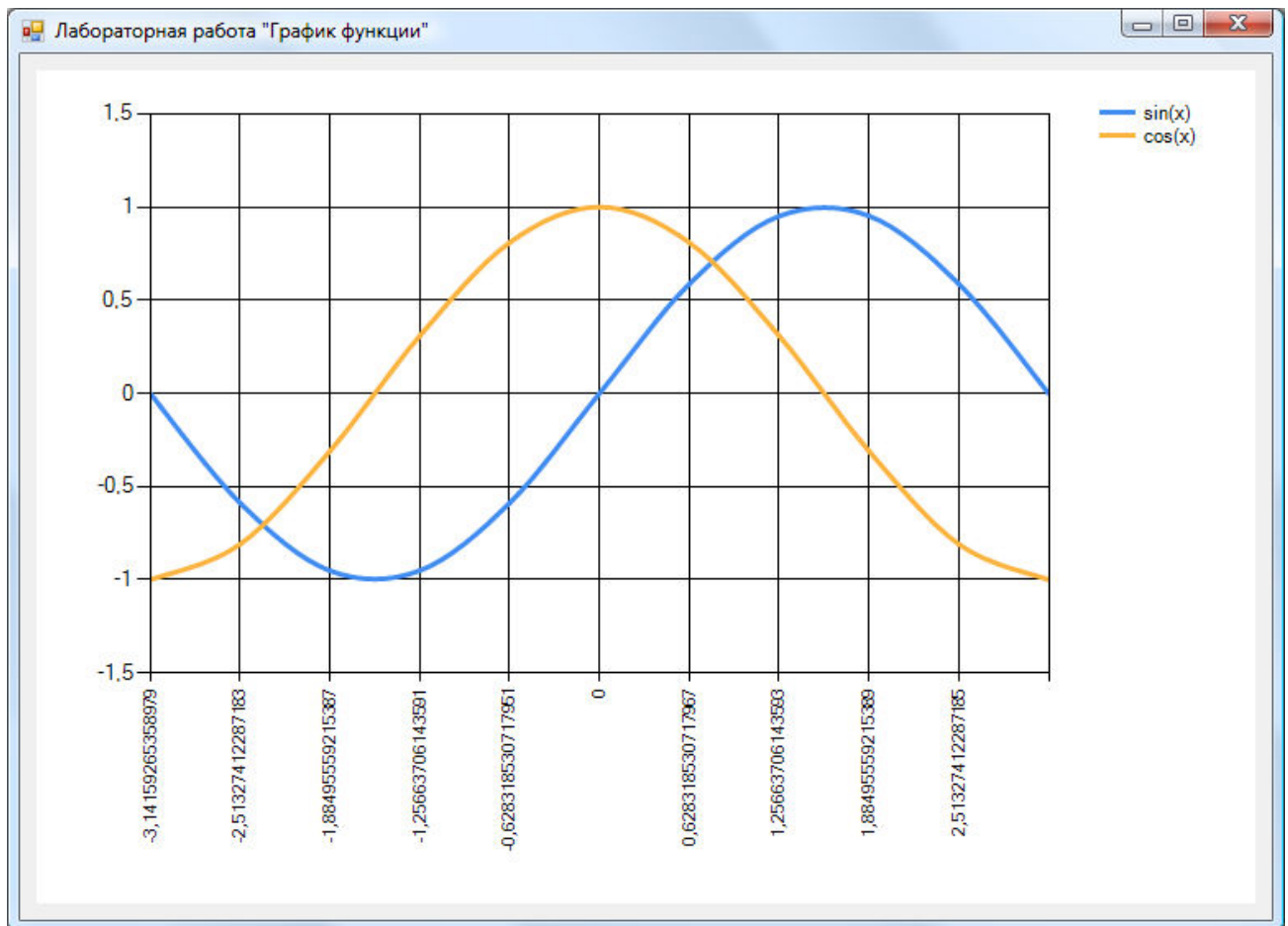


Рис 8.1. Окно программы с элементом управления.

Построение графика (диаграммы) производится после вычисления таблицы значений функции $y=f(x)$ на интервале $[Xmin, Xmax]$ с заданным шагом. Полученная таблица передается в специальный массив Points объекта Series компонента Chart с помощью метода DataBindXY. Компонент Chart осуществляет всю работу по отображению графиков: строит и размечает оси, рисует координатную сетку, подписывает название осей и самого графика, отображает переданную таблицу в виде всевозможных графиков или диаграмм. При необходимости компоненту Chart передаются данные о толщине, стиле и цвете линий, параметрах шрифта подписей, шагах разметки координатной сетки и другие настройки. В процессе работы программы изменение параметров возможно через обращение к соответствующим свойствам компонента Chart. Так, например, свойство AxisX содержит значение максимального предела нижней оси графика и при его изменении во время работы программы автоматически изменяется изображение графика.

8.2. 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 таблицы.

Прежде всего, следует определить в коде класса все необходимые переменные и константы. Конечно, можно обойтись и без этого, вставляя значения в виде чисел прямо в формулы, но это, во-первых, снизит читабельность кода программы, а во вторых, значительно усложнит изменение каких-либо параметров программы, например, интервала построения графика.

```
/// <summary>
/// Левая граница графика
/// </summary>
private double XMin = -Math.PI;

/// <summary>
/// Правая граница графика
/// </summary>
private double XMax = Math.PI;

/// <summary>
/// Шаг графика
/// </summary>
private double Step = (Math.PI * 2) / 10;

// Массив значений X – общий для обоих графиков
private double[] x;

// Два массива Y – по одному для каждого графика
private double[] y1;
private double[] y2;
```

Также в коде класса следует описать глобальную переменную типа Chart, к которой мы будем обращаться из разных методов:

```
Chart chart;
```

Поскольку данный класс не входит в пространства имен, подключаемые по умолчанию, следует выполнить дополнительные действия. Во-первых, в Обозревателе решений нужно щёлкнуть правой кнопкой по секции Ссылки и добавить ссылку на библиотеку визуализации (рис. 8.2):

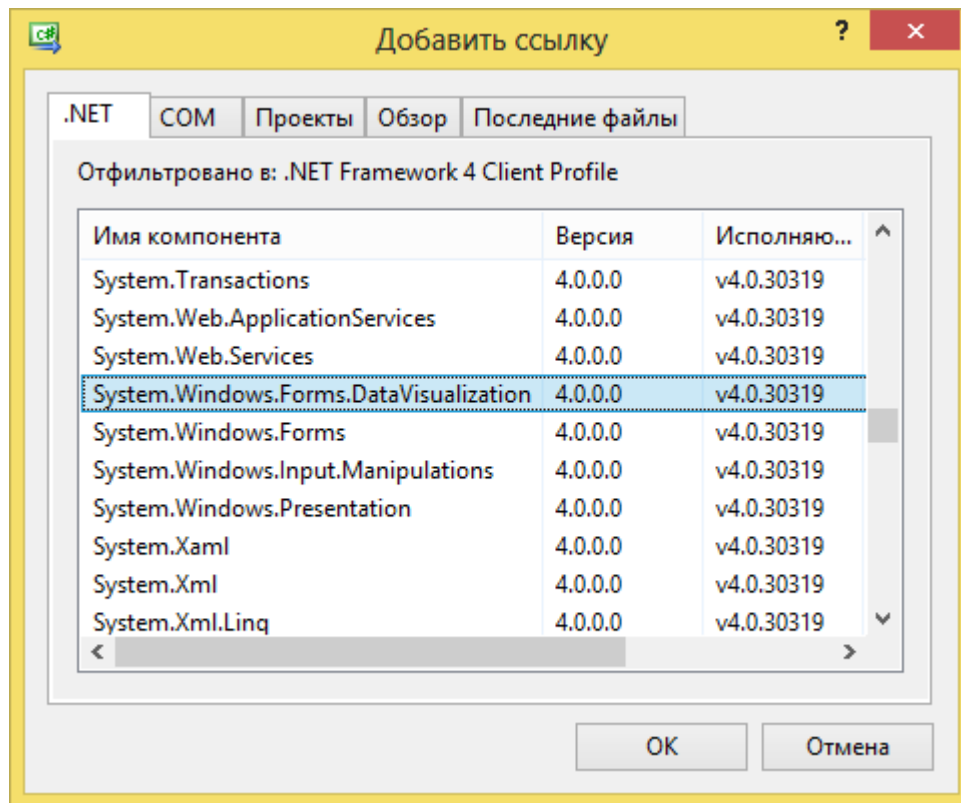


Рис. 8.2. Добавление ссылки на библиотеку визуализации.

Кроме того, следует подключить соответствующее пространство имен:

```
using System.Windows.Forms.DataVisualization.Charting;
```

Далее следует определить метод, который будет рассчитывать количество шагов и вычислять значения функций в каждой точке, внося вычисленные значения в массивы x , $y1$ и $y2$:

```
/// <summary>
/// Расчёт значений графика
/// </summary>
private void CalcFunction()
{
    // Количество точек графика
    int count = (int)Math.Ceiling((XMax - XMin) / Step)
                + 1;
    // Создаём массивы нужных размеров
    x = new double[count];
    y1 = new double[count];
    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]);
    }
}

```

После расчёта значений нужно отобразить графики на форме с помощью элемента Chart. Элемент управления Chart нельзя выбрать с помощью панели элементов – его нужно создавать прямо в коде программы. Вторым шагом следует создать область отображения графика и настроить внешний вид осей:

```

/// <summary>
/// Создаём элемент управления Chart и настраиваем его
/// </summary>
private void CreateChart()
{
    // Создаём новый элемент управления Chart
    chart = new Chart();
    // Помещаем его на форму
    chart.Parent = this;
    // Задаём размеры элемента
    chart.SetBounds(10, 10, ClientSize.Width - 20,
        ClientSize.Height - 20);

    // Создаём новую область для построения графика
    ChartArea area = new ChartArea();
    // Даем ей имя (чтобы потом добавлять графики)
    area.Name = "myGraph";
    // Задаём левую и правую границы оси X
    area.AxisX.Minimum = XMin;
    area.AxisX.Maximum = XMax;
    // Определяем шаг сетки
    area.AxisX.MajorGrid.Interval = Step;
    // Добавляем область в диаграмму
    chart.ChartAreas.Add(area);

    // Создаём объект для первого графика
    Series series1 = new Series();
    // Ссылаемся на область для построения графика
    series1.ChartArea = "myGraph";
    // Задаём тип графика – сплайны
    series1.ChartType = SeriesChartType.Spline;
    // Указываем ширину линии графика
    series1.BorderWidth = 3;
}

```

```

// Название графика для отображения в легенде
series1.LegendText = "sin(x)";
// Добавляем в список графиков диаграммы
chart.Series.Add(series1);
// Аналогичные действия для второго графика
Series series2 = new Series();
series2.ChartArea = "myGraph";
series2.ChartType = SeriesChartType.Spline;
series2.BorderWidth = 3;
series2.LegendText = "cos(x)";
chart.Series.Add(series2);

// Создаём легенду, которая будет показывать названия
Legend legend = new Legend();
chart.Legends.Add(legend);
}

```

Наконец, все эти методы следует откуда-то вызвать. Чтобы графики появлялись сразу после запуска программы, надо вызывать их в обработчике события Load формы:

```

private void Form1_Load(object sender, EventArgs e)
{
    // Создаём элемент управления
    CreateChart();

    // Расчитываем значения точек графиков функций
    CalcFunction();

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

```

8.3. 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 9. ПРОГРАММИРОВАНИЕ ГРАФИКИ

Цель лабораторной работы: изучить возможности Visual Studio по созданию простейших графических изображений. Написать и отладить программу построения на экране различных графических примитивов.