# Лабораторная работа № 3 Построение графиков

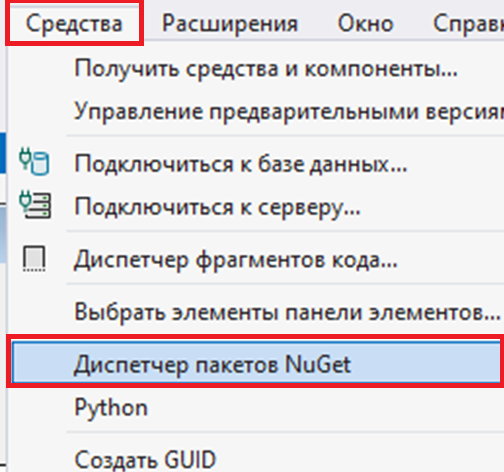
**Цель лабораторной работы**

Получение навыков по рисованию графиков на C#.

**Постановка задачи**

**Построение графиков с помощью ScottPlot.**

Рассмотрим пример добавления к проекту библиотеки для построения графиков ScottPlot. Необходимо открыть диспетчер пакетов **NuGet** (рисунок 1).



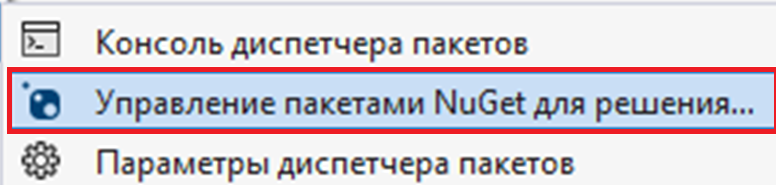


Рисунок 1 – Установка пакетов с помощью NuGet

Необходимо перейти на вкладку «Обзор» найти библиотеки ScottPlot и ScottPlot.WinForms, и нажать «Установить» (рисунок 2).

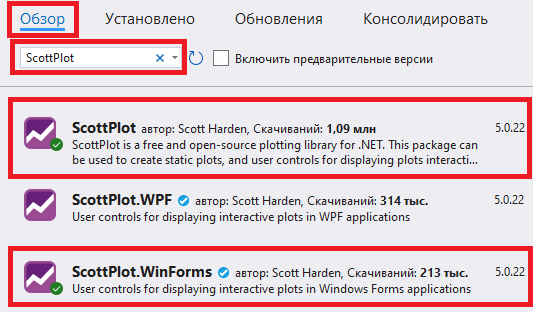


Рисунок 2 – Установка пакета ScottPlot

Элемент для построения графиков (FormsPlot) появится в панели элементов (рисунок 3). После установки пакета **возможно придется перезапустить приложение** (если появляется ошибка при добавлении элемента FormsPlot).

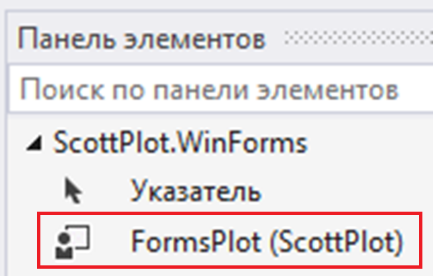


Рисунок 3 – Отображение ScottPlot в панели элементов

Добавим подписи осей, название графика и отобразим что получилось (рисунок 4).

formsPlot1.Plot.XLabel("X, м");

formsPlot1.Plot.YLabel("Y, м");

formsPlot1.Plot.Title("График");

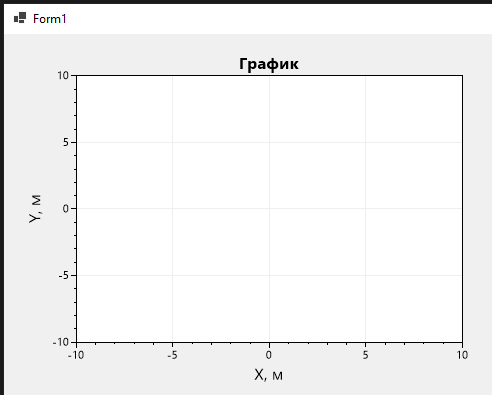


Рисунок 4 – Пустой график ScottPlot на форме

Построим график функции *f(x) = x*2 на участке от 0 до 100 (рисунок 5).

var xs = new List<double>();

var ys = new List<double>();

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

{

xs.Add(i);

ys.Add(i \* i);

}

var scatter = formsPlot1.Plot.Add.Scatter(xs, ys);

// Убираем маркеры (точки) и оставляем линию

scatter.MarkerStyle = ScottPlot.MarkerStyle.None;

scatter.Label = "x^2"; // Подписии линии графика

formsPlot1.Plot.ShowLegend();

plot.Axes.AutoScale(); // Автоматически приближать к области графика

formsPlot.Refresh(); // Вызов метода перерисовки

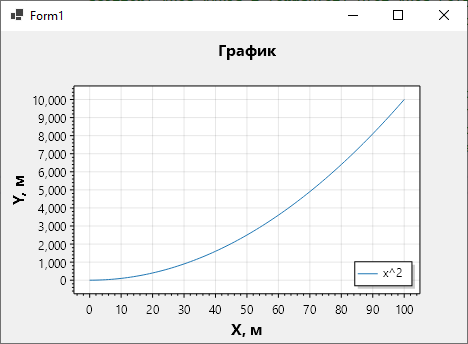


Рисунок 5 – Отображение графика функции на форме

Основное преимущество ScottPlot это **интерактивность**, график можно масштабировать и перемещаться по графику, также он сам определяет интервалы подписей у осей.

**Чтобы переместиться по графику** нужно зажать ЛКМ и двигать мышью, чтобы приблизиться (рисунок 6) или отдалиться – вращать колесико мыши, также чтобы приблизить некоторую область можно зажать колесико и выделить прямоугольник. Чтобы вернуться к исходному виду нужно нажать на колесико.

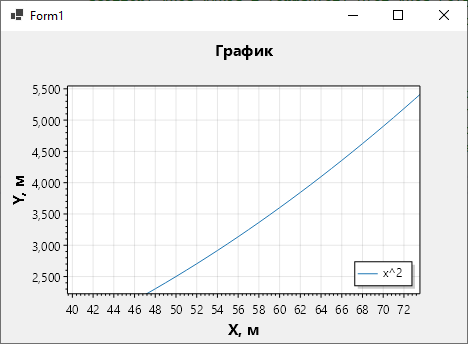


Рисунок 6 – Приближение участка графика

ScottPlot поддерживает рисование по второстепенным осям, это полезно, когда 2 набора данных сильно отличаются по значениям, но их требуется представить на одном графике (Например, давление и температура).

var xs = new List<double>();

var ys = new List<double>();

var ys2 = new List<double>();

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

{

xs.Add(i);

ys.Add(i \* i);

ys2.Add(Math.Sin(i));

}

Отобразим *x*2 на основной оси, а sin(*x*) на второстепенной оси *Oy.* Также сделаем график синуса пунктирным (рисунок 7).

var scatter1 = formsPlot1.Plot.Add.Scatter(xs, ys);

scatter1.Axes.XAxis = formsPlot1.Plot.Axes.Bottom;

scatter1.Axes.YAxis = formsPlot1.Plot.Axes.Left;

scatter1.MarkerStyle = ScottPlot.MarkerStyle.None;

scatter1.Label = "x^2";

var scatter2 = formsPlot1.Plot.Add.Scatter(xs, ys2);

scatter2.Axes.XAxis = formsPlot1.Plot.Axes.Bottom;

scatter2.Axes.YAxis = formsPlot1.Plot.Axes.Right;

scatter2.MarkerStyle = ScottPlot.MarkerStyle.None;

scatter2.LinePattern = ScottPlot.LinePattern.Dashed;

scatter2.Label = "sin(x)";

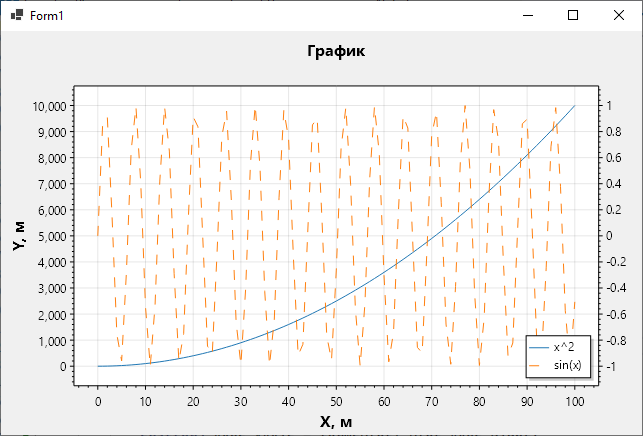


Рисунок 7 – Отображение графиков  
на главной и второстепенной осях

Рассмотрим пример построения гистограмм (рисунок 8):

var plot = formsPlot1.Plot;

double[] xs1 = { 1, 2, 3, 4 };

double[] ys1 = { 5, 10, 7, 13 };

var bars1 = plot.Add.Bars(xs1, ys1);

bars1.Label = "Alpha";

double[] xs2 = { 6, 7, 8, 9 };

double[] ys2 = { 7, 12, 9, 15 };

var bars2 = plot.Add.Bars(xs2, ys2);

bars2.Label = "Beta";

plot.ShowLegend(Alignment.UpperLeft);

// Убираем отсут от нижней границы

plot.Axes.Margins(bottom: 0);

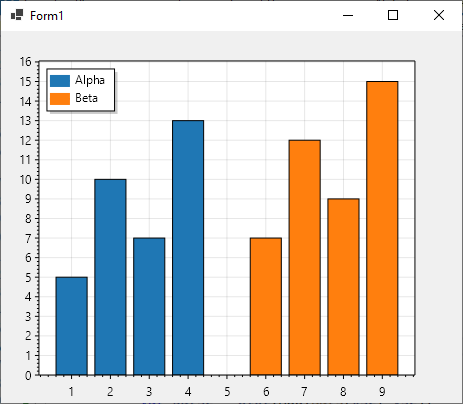


Рисунок 8 – Отображение гистограммы

Рассмотрим пример построения круговой диаграммы (рисунок 9):

var plot = formsPlot1.Plot;

List<PieSlice> slices = new()

{

new PieSlice(){ Value = 5, FillColor = Colors.Red, Label = "Red" },

new PieSlice(){ Value = 2, FillColor = Colors.Orange, Label ="Orange"},

new PieSlice(){ Value = 8, FillColor = Colors.Gold, Label ="Yellow"},

new PieSlice(){ Value = 4, FillColor = Colors.Green, Label ="Green"},

new PieSlice(){ Value = 8, FillColor = Colors.Blue, Label ="Blue" }};

var pie = plot.Add.Pie(slices);

pie.ExplodeFraction = 0.1;

plot.ShowLegend();

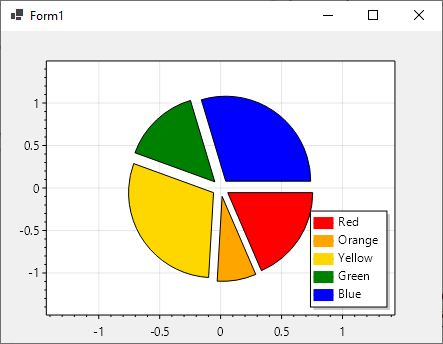


Рисунок 9 – Отображение гистограммы

**Анимация.**

Здесь необходим компонент Timer. Требуется добавить 2 кнопки – для запуска и остановки движения.

public partial class Form1 : Form

{

private double x = 0, speed = 0.1;

private List<double> xs = new();

private List<double> ys = new();

public Form1()

{

InitializeComponent();

}

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

{

timer.Start();

}

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

{

timer.Stop();

}

private double getF(double x) => Math.Sin(x);

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

{

x += speed;

xs.Add(x);

ys.Add(getF(x));

var plot = formsPlot.Plot;

plot.Clear();

var scatter = plot.Add.Scatter(xs, ys);

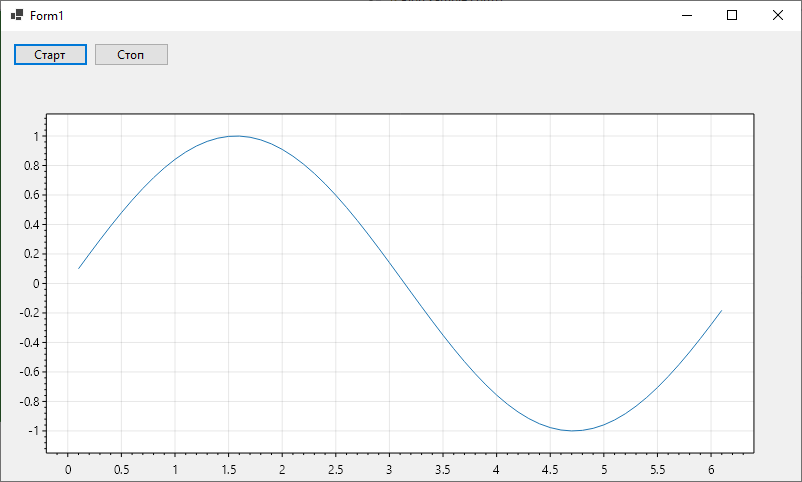
scatter.MarkerStyle = MarkerStyle.None;

plot.Axes.AutoScale();

formsPlot.Refresh();

}

}

****

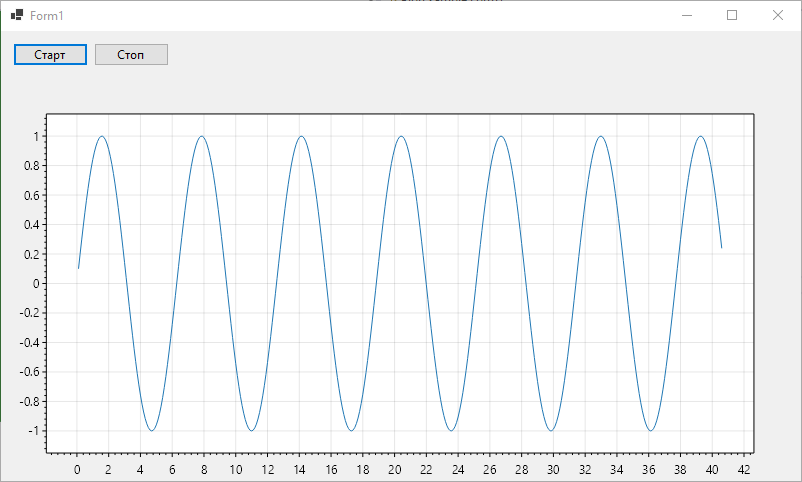
****

Рисунок 10 – Отображение анимации синусоиды

Примеры построения графиков и работы в ScottPlot приводятся в документации на сайте: <https://scottplot.net/cookbook/5.0/>

**Задание на лабораторную работу**

1. Разработать Windows-приложение для построения графиков.
2. Заголовок окна приложения должен содержать ФИО исполнителя, группу и номер варианта.
3. В программе должна быть предусмотрена обработка исключений, возникающих из-за ошибочного ввода пользователя.
4. Написать XML комментарии, поясняющие работу программы.

**Варианты заданий на лабораторную работу**

**Задание.**

Необходимо построить 2D графики в среде MS Visual Studio на языке C# (компонент ScottPlot).

1. Построить график функции *y=f(x)*. Должны быть подписаны оси координат с единицами измерения, отображены линии сеток. Предусмотреть возможность задания цвета и толщины графика.
2. Реализовать возможность установки (и удаления) маркера на график, предусмотреть ввод имени маркера. Маркер должен появляться в легенде.
3. Реализовать возможность отображения значения функции рядом с мышью при наведении мыши на график.

Переменные являются осями координат для графиков, параметры вводятся с формы.

|  |  |
| --- | --- |
| **№ Варианта** | **График** |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |

**Пример разработки приложения**

**Задание.** Рассмотрим построение графика

.

**Шаг 1: Установка необходимых библиотек**

Перед началом работы необходимо установить два NuGet пакета:

ScottPlot - основная библиотека для построения графиков

ScottPlot.WinForms - компоненты для интеграции с Windows Forms

**Шаг 2: Подготовка главной формы (рисунок 11)**

На форме должны быть расположены следующие элементы:

* Элемент FormsPlot (из ScottPlot.WinForms) для отображения графика
* NumericUpDown элементы для настройки параметров:
  + fromNumeric - начальное значение x
  + toNumeric - конечное значение x
  + pointsNumeric - количество точек
  + widthNumeric - толщина линии графика
* Кнопка setColorButton для выбора цвета линии

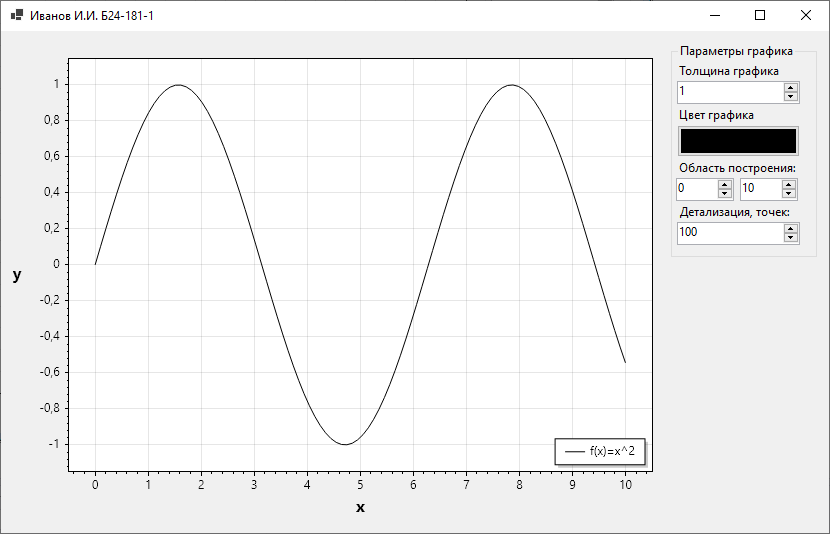


Рисунок 11 – Главное окно программы для построения графика

**Шаг 3: Основной функционал формы (MainForm)**

1. **Инициализация**:
   * В конструкторе создаются экземпляры ScatterPlotter и MarkerPlotter
   * Настраивается отображение легенды на графике
2. **Загрузка формы**:
   * Устанавливаются подписи осей X и Y
   * Вызывается метод BuildPlot() для первоначального построения графика
3. **Функция для графика**:
   * Метод f(double x) определяет функцию для построения (в примере - sin(x))
4. **Обработчики событий**:
   * Изменение параметров (fromNumeric, toNumeric, pointsNumeric) вызывает перестроение графика
   * Изменение толщины линии (widthNumeric) обновляет отображение
   * Кнопка выбора цвета (setColorButton) открывает диалог выбора цвета
5. **Конфигурация графиков**:
   * ScatterConfigurate - настраивает внешний вид линии графика
   * MarkerConfigurate - настраивает внешний вид маркера
6. **Построение графика**:
   * BuildPlot() создает данные для графика с помощью DataBuilder
   * formsPlot\_MouseDown обрабатывает клики по графику для отображения маркера в ближайшей точке

public partial class MainForm : Form

{

    private ScottPlot.Color lineColor = Colors.Black;

    private readonly ScatterPlotter scatterPlotter;

    private readonly MarkerPlotter markerPlotter;

    public MainForm()

    {

        InitializeComponent();

        formsPlot.Plot.ShowLegend();

        scatterPlotter = new ScatterPlotter(formsPlot);

        scatterPlotter.Configuration += ScatterConfigurate;

        markerPlotter = new MarkerPlotter(formsPlot);

        markerPlotter.Configuration += MarkerConfigurate;

    }

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

    {

        formsPlot.Plot.Axes.Bottom.Label.Text = "x";

        formsPlot.Plot.Axes.Left.Label.Text = "y";

        formsPlot.Plot.Axes.Left.Label.Rotation = 0;

        BuildPlot();

    }

    private double f(double x) => Math.Sin(x);

    private void widthNumeric\_ValueChanged(object sender, EventArgs e)

    {

        scatterPlotter.Draw();

    }

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

    {

        var cd = new ColorDialog();

        if (cd.ShowDialog() == DialogResult.OK)

        {

            lineColor = ScottPlot.Color.FromColor(cd.Color);

            setColorButton.BackColor = cd.Color;

        }

        scatterPlotter.Draw();

    }

    private void fromNumeric\_ValueChanged(object sender, Even-tArgs e)

    {

        if (fromNumeric.Value > toNumeric.Value)

            toNumeric.Value = fromNumeric.Value;

        BuildPlot();

    }

    private void toNumeric\_ValueChanged(object sender, Even-tArgs e)

    {

        if (toNumeric.Value < fromNumeric.Value)

            fromNumeric.Value = toNumeric.Value;

        BuildPlot();

    }

    private void pointsNumeric\_ValueChanged(object sender, EventArgs e)

    {

        BuildPlot();

    }

    private void ScatterConfigurate(Scatter scatter)

    {

        scatter.LineWidth = (int)widthNumeric.Value;

        scatter.LineColor = lineColor;

        scatter.MarkerStyle = MarkerStyle.None;

        scatter.LegendText = "f(x)=x^2";

    }

    private void MarkerConfigurate(Marker marker)

    {

        marker.MarkerColor = Colors.Red;

        marker.LegendText = $"x: {marker.X:0.###} y: {mark-er.Y:0.###}";

    }

    private void BuildPlot()

    {

        var from = (double)fromNumeric.Value;

        var to = (double)toNumeric.Value;

        var pointsCount = (int)pointsNumeric.Value;

        var builder = new DataBuilder(f, new DataRange(from, to), pointsCount);

        scatterPlotter.UpdateData(builder.Build());

    }

    private void formsPlot\_MouseDown(object sender, MouseEven-tArgs e)

    {

        if (scatterPlotter.Scatter == null)

            return;

        var mousePixel = new Pixel(e.Location.X, e.Location.Y);

        var mouseLocation = formsPlot.Plot.GetCoordinates(mousePixel);

        var nearest = scatterPlot-ter.Scatter.Data.GetNearest(mouseLocation, formsPlot.Plot.LastRender);

        if (nearest.IsReal)

        {

            markerPlotter.SetPosition(nearest.X, nearest.Y);

        }

    }

}

**Шаг 4: Вспомогательные классы**

DataBuilder

Генерирует данные для построения графика:

* DataRange - запись для хранения диапазона значений
* DataXY - запись для хранения пар значений X и Y
* Метод Build() создает равномерно распределенные точки в заданном диапазоне и вычисляет значения функции в этих точках

public record DataRange(double Min, double Max);

public record DataXY(ICollection<double> X, ICollec-tion<double> Y);

public class DataBuilder

{

    public DataBuilder(

        Func<double, double> function,

        DataRange range,

        int pointsCount)

    {

        Function = function;

        Range = range;

        PointsCount = pointsCount;

    }

    public Func<double, double> Function { get; set; }

    public DataRange Range { get; set; }

    public int PointsCount { get; set; }

    public DataXY Build()

    {

        var xs = new double[PointsCount];

        var ys = new double[PointsCount];

        var dx = (Range.Max - Range.Min) / (PointsCount - 1);

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

        {

            xs[i] = Range.Min + i \* dx;

            ys[i] = Function(xs[i]);

        }

        return new DataXY(xs, ys);

    }

}

Создадим класс для построения графика ScatterPlotter

Он отвечает за отображение линии графика:

* UpdateData() - обновляет данные графика
* Draw() - перерисовывает график с текущими параметрами
* Позволяет настраивать внешний вид через событие Configuration

public class ScatterPlotter

{

    private readonly FormsPlot formsPlot;

    private DataXY? data;

    public Scatter? Scatter { get; private set; }

    public Action<Scatter>? Configuration { get; set; }

    public ScatterPlotter(FormsPlot formsPlot)

    {

        this.formsPlot = formsPlot;

    }

    public void Draw()

    {

        if (data == null)

            throw new ArgumentException("Не заданы данные для графика");

        if (Scatter != null)

            formsPlot.Plot.Remove(Scatter);

        Scatter = formsPlot.Plot.Add.Scatter(data.X.ToArray(), data.Y.ToArray());

        Configuration?.Invoke(Scatter);

        formsPlot.Refresh();

    }

    public void UpdateData(DataXY data)

    {

        this.data = data;

        Draw();

        formsPlot.Plot.Axes.AutoScale();

    }

}

Создадим класс для добавления маркера на график MarkerPlotter

Он отвечает за отображение маркера на графике:

* SetPosition() - устанавливает позицию маркера
* Draw() - перерисовывает маркер
* Позволяет настраивать внешний вид через событие Configuration

public class MarkerPlotter

{

    private readonly FormsPlot formsPlot;

    private double x;

    private double y;

    public Marker? Marker { get; private set; }

    public Action<Marker>? Configuration { get; set; }

    public MarkerPlotter(FormsPlot formsPlot)

    {

        this.formsPlot = formsPlot;

    }

    public void Draw()

    {

        if (Marker != null)

            formsPlot.Plot.Remove(Marker);

        Marker = formsPlot.Plot.Add.Marker(x, y);

        Configuration?.Invoke(Marker);

        formsPlot.Refresh();

    }

    public void SetPosition(double x, double y)

    {

        this.x = x;

        this.y = y;

        Draw();

    }

}

**Контрольные вопросы**

1. Что такое ScottPlot и для чего он используется?
2. Как установить библиотеку ScottPlot в проект C#?
3. Как создать простой график с использованием ScottPlot?
4. Какой метод используется для отображения графика в WinForms?
5. Какие типы графиков поддерживает ScottPlot?
6. Как построить линейный график и чем он отличается от точечного графика?
7. Как изменить заголовок графика и подписи осей?
8. Как изменить цвет и стиль линий на графике?
9. Как обновить данные на уже существующем графике без его перерисовки?
10. Как добавить возможность масштабирования?
11. Как сохранить график в файл (например, в формате PNG или SVG)?
12. Какие методы доступны для экспорта графиков в другие форматы?
13. Как добавить несколько графиков на одну ось?
14. Как настроить отображение легенды для нескольких графиков?
15. Как реализовать анимацию графика, чтобы данные обновлялись в реальном времени?
16. Как изменить шрифт и размер текста на графике?
17. Как добавить аннотации или текстовые метки к определенным точкам на графике?
18. Как загрузить данные из файла (например, CSV) и построить график на их основе?