Лабораторная работа № 1.

Задание 1. Линейные алгоритмы

Цель задания: научиться составлять каркас простейшей программы в среде Visual Studio. Написать и отладить программу линейного алгоритма.

1.1. Структура приложения

Перед началом программирования необходимо создать проект. *Проект* содержит все исходные материалы для приложения, такие как файлы исходного кода, ресурсов, значки, ссылки на внешние файлы, на которые опирается программа, и данные конфигурации, такие как параметры компилятора.

Кроме понятия проект часто используется более глобальное понятие – *решение (solution)*. Решение содержит один или несколько проектов, один из которых может быть указан как стартовый проект. Выполнение решения начинается со стартового проекта.

Таким образом, при создании простейшей C# программы в Visual Studio создается папка решения, в которой для каждого проекта создается подпапка проекта, а уже в ней – другие подпапки с результатами компиляции приложения.

Проект – это основная единица, с которой работает программист. При создании проекта можно выбрать его тип, а Visual Studio создаст каркас проекта в соответствии с выбранным типом.

В предыдущей лабораторной работе мы попробовали создавать оконные приложения, или иначе *Приложения Windows Forms*. Примером другого типа проекта является привести проект *консольного* приложения.

По своим «внешним» проявлениям консольные напоминают приложения DOS, запущенные в Windows. Тем не менее, это настоящие Win32-приложения, которые под DOS работать не будут. Для консольных приложений доступен Win32 API, а кроме того, они могут использовать консоль – окно, предоставляемое системой, которое работает в текстовом режиме и в которое можно вводить данные с клавиатуры. Особенность консольных приложений в том, что они работают не в графическом, а в текстовом режиме.

Проект в Visual Studio состоит из файла проекта (файл с расширением `.csproj`), одного или нескольких файлов исходного текста (с расширением `.cs`), файлов с описанием окон формы (с расширением`.designer.cs`), файлов ресурсов (с расширением `.resx`), а также ряда служебных файлах.

В *файле проекта* находится информация о модулях, составляющих данный проект, входящих в него ресурсах, а также параметров построения программы. Файл проекта автоматически создается и изменяется средой Visual Studio и не предназначен для ручного редактирования.

Файл исходного текста – программный модуль, предназначен для размещения текстов программ. В этом файле программист размещает текст программы, написанный на языке C#. Модуль имеет следующую структуру:

```
// Раздел подключенных пространств имен using System;
// Пространство имен нашего проекта
```

```
namespace MyFirstApp
{
    // Класс окна
    public partial class Form1 : Form
    {
        //  Методы  окна  public
        Form1() {
    InitializeComponent(); } } }
```

В разделе подключения пространств имен (каждая строка которого располагается в начале файла и начинается ключевым словом `using`) описываются используемые пространства имен. Каждое пространство имен включает в себя классы, выполняющие определенную работу, например, классы для работы с сетью располагаются в пространстве `System.Net`, а для работы с файлами – в `System.IO`. Большая часть пространств, которые используются в обычных проектах, уже подключена при создании нового проекта, но при необходимости можно дописать дополнительные пространства имен.

Для того чтобы не происходило конфликтов имен классов и переменных, классы нашего проекта также помещаются в отдельное пространство имен. Определяется оно ключевым словом `namespace`, после которого следует имя пространства (обычно оно совпадает с именем проекта). Внутри пространства имен помещаются наши классы – в новом проекте это класс окна, который содержит все методы для управления поведением окна. Обратите внимание, что в определении класса присутствует ключевое слово `partial`, это говорит о том, что в исходном тексте представлена только часть класса, с которой мы работаем непосредственно, а служебные методы для обслуживания окна скрыты в другом модуле (при желании их тоже можно посмотреть, но редактировать вручную не рекомендуется).

Наконец, внутри класса располагаются переменные, методы и другие элементы программы. Фактически, основная часть программы размещается внутри класса при создании обработчиков событий.

При компиляции программы Visual Studio создает исполняемые .exe-файлы в каталоге `bin`.

1.2. Работа с проектом

Проект в Visual Studio состоит из многих файлов, и создание сложной программы требует хранения каждого проекта в отдельной папке. При создании нового проекта Visual Studio по умолчанию сохраняет его в отдельной папке. Рекомендуется создать для себя свою папку со своей фамилией внутри папки своей группы, чтобы все проекты хранились в одном месте. После этого можно запускать Visual Studio и создавать новый проект (как это сделать, показано в предыдущей лабораторной работе).

Сразу после создания проекта рекомендуется сохранить его в подготовленной папке: *Файл → Сохранить все*. При внесении значительных изменений в проект следует еще раз сохранить проект той же командой, а перед запуском программы на выполнение среда обычно сама сохраняет проект на случай какого-либо сбоя. Для открытия существующего проекта используется команда *Файл → Открыть проект*, либо можно найти в папке файл проекта с расширением `.sln` и сделать на нем двойной щелчок.

1.3. Описание данных

Типы данных имеют особенное значение в C#, поскольку это *строго типизированный язык*. Это означает, что все операции подвергаются строгому контролю со стороны компилятора на соответствие типов, причем недопустимые операции не компилируются. Такая строгая проверка типов позволяет предотвратить ошибки и повысить надежность программ. Для обеспечения контроля типов все переменные, выражения и значения должны принадлежать к определенному типу. Такого понятия, как *бестиповая* переменная, допустимая в ряде скриптовых языков, в C# вообще не существует. Более того, тип значения определяет те операции, которые разрешается выполнять над ним. Операция, разрешенная для одного типа данных, может оказаться недопустимой для другого.

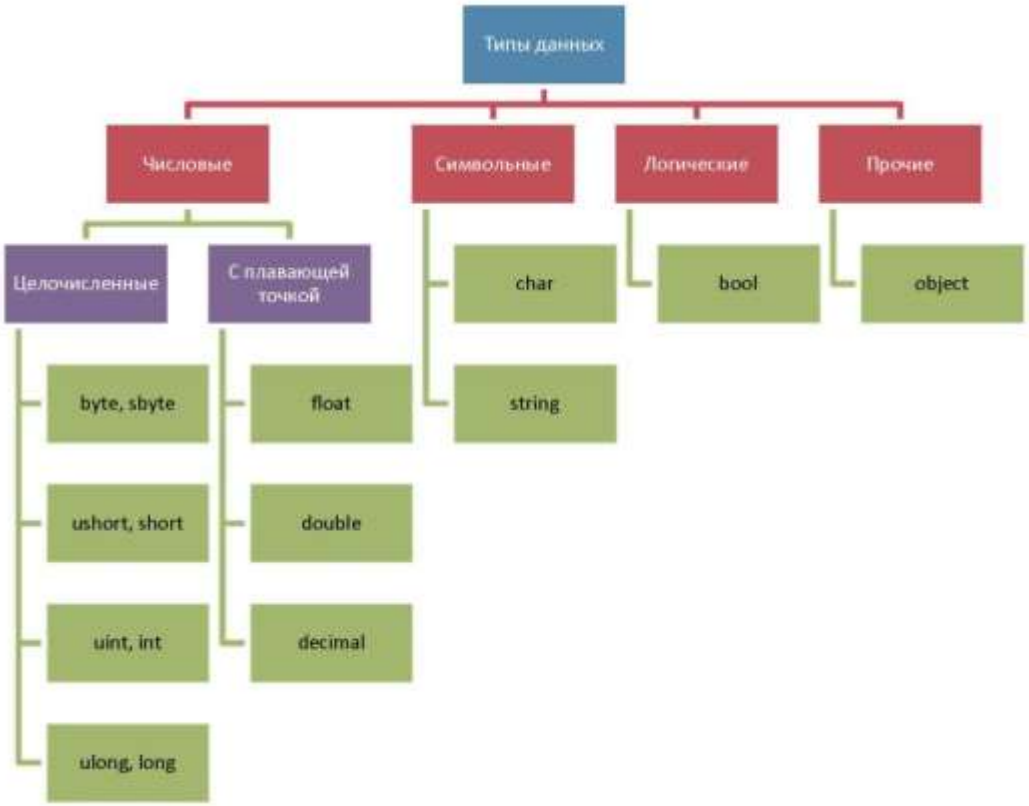

Рис. 1.1. Структура типов данных

Целочисленные типы

В C# определены девять целочисленных типов: `char`, `byte`, `sbyte`, `short`, `ushort`, `int`, `uint`, `long` и `ulong`. Тип `char` может хранить числа, но чаще используется для представления символов. Остальные восемь целочисленных типов предназначены для числовых расчетов.

Некоторые целочисленные типы могут хранить как положительные, так и отрицательные значения (`sbyte`, `short`, `int` и `long`), другие же – только положительные (`char`, `byte`, `ushort`, `uint` и `ulong`).

Типы с плавающей точкой

Такие типы позволяют представлять числа с дробной частью. В C# имеются три разновидности типов данных с плавающей точкой: `float`, `double` и `decimal`. Первые два типа представляют числовые значения с одинарной и двойной точностью, вычисления над ними

выполняются аппаратно и поэтому быстро. Тип `decimal` служит для представления чисел с плавающей точкой высокой точности без округления, характерного для типов `float` и `double`. Вычисления с использованием этого типа выполняются программно и поэтому более медленны.

Числа, входящие в выражения, C# по умолчанию считает целочисленными. Поэтому следующее выражение будет иметь значение 0, ведь если 1 нацело разделить на 2, то получится как раз 0:

```
double x = 1 / 2;
```

Чтобы этого не происходило, в подобных случаях нужно явно указывать тип чисел с помощью символов-модификаторов: `f` для `float` и `d` для `double`. Приведенный выше пример правильно будет выглядеть так:

```
double x = 1d / 2d;
```

Иногда в программе возникает необходимость записать числа в экспоненциальной форме. Для этого после мантиссы числа записывается символ «e» и сразу после него – порядок. Например, число $2{,}5 \cdot 10^{-2}$ будет записано в программе следующим образом:

```
2.5e-2
```

Символьные типы

В C# символы представлены не 8-разрядным кодом, как во многих других языках программирования, а 16-разрядным кодом, который называется юникодом (`Unicode`). В юникоде набор символов представлен настолько широко, что он охватывает символы практически из всех естественных языков на свете.

Основным типом при работе со строками является тип `string`, задающий строки переменной длины. Тип `string` представляет последовательность из нуля или более символов в кодировке Юникод. По сути, текст хранится в виде последовательной доступной только для чтения коллекции объектов `char`.

Логический тип данных

Тип `bool` представляет два логических значения: «истина» и «ложь». Эти логические значения обозначаются в C# зарезервированными словами `true` и `false` соответственно. Следовательно, переменная или выражение типа `bool` будет принимать одно из этих логических значений.

Рассмотрим самые популярные данные – *переменные* и *константы*. Переменная – это ячейка памяти, которой присвоено некоторое имя, и это имя используется для доступа к данным, расположенным в данной ячейке. Для каждой переменной задается *тип данных* – диапазон всех возможных значений для данной переменной. Объявляются переменные непосредственно в тексте программы. Лучше всего сразу присвоить им начальное значение с помощью знака присвоения «=» (*переменная = значение*):

```
int a; // Только объявление
int b = 7; // Объявление и инициализация
```

Для того чтобы присвоить значение символьной переменной, достаточно заключить это значение (т. е. символ) в одинарные кавычки:

```
char ch;    // Только объявление
char symbol = 'Z'; // Объявление и инициализация
```

Частным случаем переменных являются *константы*. Константы – это переменные, значения которых не меняются в процессе выполнения программы. Константы описываются как обычная переменная, только с ключевым словом `const` впереди:

```
const int c = 5;
```

1.4. Ввод/вывод данных в программу

Рассмотрим один из способов ввода данных через элементы, размещенные на форме. Для ввода данных чаще всего используют элемент управления `TextBox`, через обращение к его свойству `Text`. Свойство `Text` хранит в себе строку введенных символов. Поэтому данные можно считать таким образом:

```
private void button1_Click(object sender,EventArgs e)
  {
   string s = textBox1.Text;
  }
```

Однако со строкой символов трудно производить арифметические операции, поэтому лучше всего при вводе числовых данных перевести строку в целое или вещественное число. Для этого у типов `int` и `double` существуют методы `Parse` для преобразования строк в числа. С этими числами можно производить различные арифметические действия. Таким образом, предыдущий пример можно переделать следующим образом:

```
private void button1_Click(object sender, EventArgs e)
    {
     string s = textBox1.Text;
     int a = int.Parse(s);
     int b = a * a;
     }
```

В языках программирования в дробных числах чаще всего используется точка, например: «15.7». Однако в **C**# методы преобразования строк в числа (вроде `double.Parse()` или `Convent.ToFloatQ)` учитывают региональные настройки Windows, в которых в качестве десятичной точки используется символ *запятой* (например, «15,7»). Поэтому в полях `TextBox` в формах следует вводить дробные числа с *запятой,* а не с точкой. В противном случае преобразование не выполнится, а программа остановится с ошибкой.

Перед выводом числовые данные следует преобразовать назад в строку. Для этого у каждой переменной существует метод `ToStringQ`, который возвращает в результате строку с символьным представлением значения. Вывод данных можно осуществлять в элементы `TextBox` или `Label`, используя свойство `Text`. Например:

```
private void buttonl_Click(object sender, EventArgs e)
     {
     string s = textBoxl.Text;
     int a = int.Parse(s);
     int b = a * a;
     labell.Text = b.ToString();
```

```
}
```

1.5. Арифметические действия и стандартные функции

При вычислении выражения, стоящего в правой части оператора присвоения, могут использоваться арифметические операции:

- умножение (x);
- сложение (+);
- вычитание (-);
- деление (/);
- остаток от деления (%).

Для задания приоритетов операций могут использоваться круглые скобки `( )`. Также могут использоваться стандартные математические функции, представленные методами класса `Math`:

- `Math.Sin(a)` – синус;
- `Math.Sinh(a)` – гиперболический синус;
- `Math.Cos(a)` – косинус (аргумент задается в радианах);
- `Math.Atan(a)` – арктангенс (аргумент задается в радианах);
- `Math.Log(a)` – натуральный логарифм;
- `Math.Exp(a)` – экспонента;
- `Math.Pow(x, y)` – возводит переменную x в степень y;
- `Math.Sqrt(a)` – квадратный корень;
- `Math.Abs(a)` – модуль числа;
- `Math.Truncate(a)` – целая часть числа;
- `Math.Round(a)` – округление числа.

В тригонометрических функциях все аргументы задаются в радианах.

1.6. Пример написания программы

Задание: составить программу вычисления для заданных значений x, y, z арифметического выражения:

$$u = \mathrm{tg}^2(x+y) - \mathrm{e}^{y-z}\sqrt{\cos x^2 + \sin z^2}$$

Панель диалога программы организовать в виде, представленном на рис. 1.2.

Для вывода результатов работы программы в программе используется текстовое окно, которое представлено обычным элементом управления. После установки свойства `Multiline` в `True` появляется возможность растягивать элемент управления не только по горизонтали, но и по вертикали. А после установки свойства `ScrollBars` в значение `Both` в окне появится вертикальная, а при необходимости и горизонтальная полосы прокрутки. Информация, которая отображается построчно в окне, находится в массиве строк `Lines`, каждая строка которого имеет тип `string`.

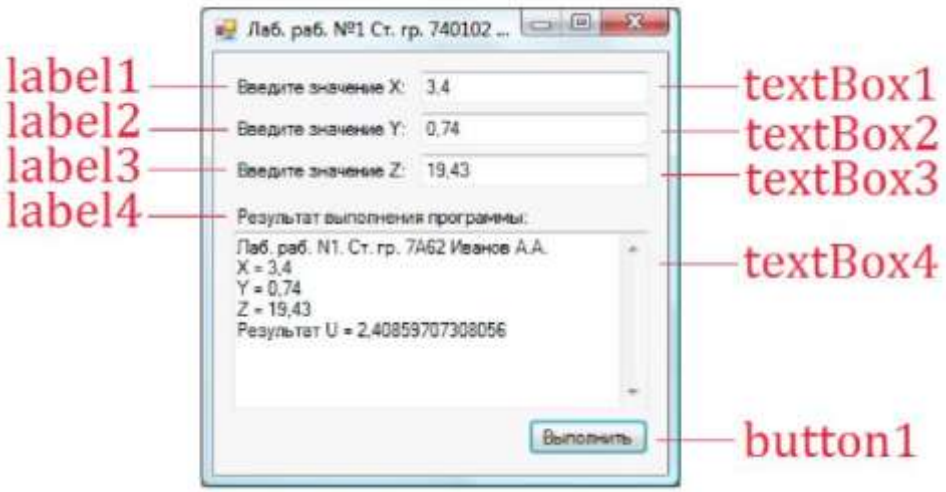

Рис. 1.2. Внешний вид программы

Однако нельзя напрямую обратиться к этому свойству для добавления новых строк, поскольку размер массивов в C# определяется в момент их инициализации. Для добавления нового элемента используется свойство `Text`, к текущему содержимому которого можно добавить новую строку:

```
textBox4.Text += Environment.NewLine + "Привет";
```

В этом примере к текущему содержимому окна добавляется символ перевода курсора на новую строку (который может отличаться в разных операционных системах и потому представлен свойством класса `Environment`) и сама новая строка. Если добавляется числовое значение, то его предварительно нужно привести в символьный вид методом `ToString()`.

Работа с программой происходит следующим образом. Нажмите (щелкните мышью) кнопку «*Выполнить*». В окне `textBox4` появляется результат. Измените исходные значения `x`, `y`, `z` в окнах `textBox1`– `textBox3` и снова нажмите кнопку «*Выполнить*» – появятся новые результаты.

Полный текст программы имеет следующий вид:

```
using System;
using System.Windows.Forms;
namespace MyFirstApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
         InitializeComponent();
        }
        private void Form1_Load(object sender,EventArgs e)
            {
            // Начальное значение X
            textBox1.Text = "3,4";
```

```
            // Начальное значение Y
            textBox2.Text = "0,74";
            // Начальное значение Z
            textBox3.Text = "19,43";
            }
        private void button1_Click(object sender,EventArgs e)
           {
            // Считывание значения X
            double x = double.Parse(textBox1.Text);
            // Вывод значения X в окно
            textBox4.Text += Environment.NewLine +
                "X = " + x.ToString();
            // Считывание значения Y
            double y = double.Parse(textBox2.Text);
            // Вывод значения Y в окно
            textBox4.Text += Environment.NewLine +
                "Y = " + y.ToString();
            // Считывание значения Z
            double z = double.Parse(textBox3.Text);
            // Вывод значения Z в окно
            textBox4.Text += Environment.NewLine +
                "Z = " + z.ToString();
            // Вычисляем арифметическое выражение
            double a = Math.Tan(x + y) *
                Math.Tan(x + y);
            double b = Math.Exp(y - z);
            double c = Math.Sqrt(Math.Cos(x*x) +
                Math.Sin(z*z));
            double u = a -b * c;
            // Выводим результат в окно
            textBox4.Text += Environment.NewLine +
          "Результат U = " + u.ToString();
          }
        }
    }
```

Если просто скопировать этот текст и заменить им то, что было в редакторе кода Visual Studio, то программа не заработает. Правильнее будет создать обработчики событий `Load` у формы и `Click` у кнопки и уже в них вставить соответствующий код. Это замечание относится и ко всем последующим лабораторным работам.

1.7. Выполнение индивидуального задания

Ниже приведено 20 вариантов задач. По указанию преподавателя выберите свое индивидуальное задание. Уточните условие задания, количество, наименование, типы исходных данных. В соответствии с этим установите необходимое количество окон `TextBox`, тексты заголовков на форме, размеры шрифтов, а также типы переменных и функции преобразования при вводе и выводе результатов. Для проверки правильности
программы после задания приведен контрольный пример: тестовые значения переменных, используемых в выражении, и результат, который при этом получается.

Варианты задания

1. $t=\dfrac{2\cos\left(x-\dfrac{\pi}{6}\right)}{0.5+\sin^2 y}\left(1+\dfrac{z^2}{3-z^2/5}\right).$

При $x=14.26,\ y=-1.22,\ z=3.5\times10^{-2}$ $t=0.564849.$

2. $u=\dfrac{\sqrt[3]{8+|x-y|^2+1}}{x^2+y^2+2}-e^{|x-y|}\left(\operatorname{tg}^2 z+1\right)^x.$

При $x=-4.5,\ y=0.75\times10^{-4},\ z=0.845\times10^2$ $u=-55.6848.$

3. $v=\dfrac{1+\sin^2(x+y)}{\left|x-\dfrac{2y}{1+x^2y^2}\right|}x^{|y|}+\cos^2\left(\operatorname{arctg}\dfrac{1}{z}\right).$

При $x=3.74\times10^{-2},\ y=-0.825,\ z=0.16\times10^2,\ v=1.0553.$

4. $w=\left|\cos x-\cos y\right|^{\left(1+2\sin^2 y\right)}\left(1+z+\dfrac{z^2}{2}+\dfrac{z^3}{3}+\dfrac{z^4}{4}\right).$

При $x=0.4\times10^4,\ y=-0.875,\ z=-0.475\times10^{-3}$ $w=1.9873.$

5. $\alpha=\ln\left(y^{-\sqrt{|x|}}\right)\left(x-\dfrac{y}{2}\right)+\sin^2\operatorname{arctg}(z).$

При $x=-15.246,\ y=4.642\times10^{-2},\ z=20.001\times10^2$ $\alpha=-182.036.$

6. $\beta=\sqrt{10\left(\sqrt[3]{x}+x^{y+2}\right)}\left(\arcsin^2 z-|x-y|\right).$

При $x=16.55\times10^{-3},\ y=-2.75,\ z=0.15$ $\beta=-38.902.$

7. $\gamma=5\operatorname{arctg}(x)-\dfrac{1}{4}\arccos(x)\dfrac{x+3|x-y|+x^2}{|x-y|z+x^2}.$

При $x=0.1722,\ y=6.33,\ z=3.25\times10^{-4}$ $\gamma=-172.025.$

8. $\varphi=\dfrac{e^{|x-y|}|x-y|^{x+y}}{\operatorname{arctg}(x)+\operatorname{arctg}(z)}+\sqrt[3]{x^6+\ln^2 y}.$

При $x=-2.235\times10^{-2},\ y=2.23,\ z=15.221$ $\varphi=39.374.$

9. $\psi = \left| x^{\frac{y}{x}} - \sqrt[3]{\frac{y}{x}} \right| + (y-x)\dfrac{\cos y - \dfrac{z}{(y-x)}}{1+(y-x)^2}.$

При $x = 1.825\times10^{2}$, $y = 18.225$, $z = -3.298\times10^{-2}$ $\psi = 1.2131$.

10. $a = 2^{-x}\sqrt{x + \sqrt[4]{|y|}}\sqrt[3]{e^{x-1/\sin z}}.$

При $x = 3.981\times10^{-2}$, $y = -1.625\times10^{3}$, $z = 0.512$ $a = 1.26185$.

11. $b = y^{\sqrt[3]{|y|}} + \cos^3(y)\dfrac{|x-y|\left(1+\dfrac{\sin^2 z}{\sqrt{x+y}}\right)}{e^{|x-y|}+\dfrac{x}{2}}.$

При $x = 6.251$, $y = 0.827$, $z = 25.001$ $b = 0.7121$.

12. $c = 2^{(y^x)} + \left(3^x\right)^y - \dfrac{y\left(\operatorname{arctg} z - \dfrac{\pi}{6}\right)}{|x| + \dfrac{1}{y^2+1}}.$

При $x = 3.251$, $y = 0.325$, $z = 0.466\times10^{-4}$ $c = 4.025$.

13. $f = \dfrac{\sqrt[4]{y+\sqrt[3]{x-1}}}{|x-y|\left(\sin^2 z + \operatorname{tg} z\right)}.$

При $x = 17.421$, $y = 10.365\times10^{-3}$, $z = 0.828\times10^{5}$ $f = 0.33056$.

14. $g = \dfrac{y^{x+1}}{\sqrt[3]{|y-2|+3}} + \dfrac{x+\dfrac{y}{2}}{2|x+y|}(x+1)^{-1/\sin z}.$

При $x = 12.3\times10^{-1}$, $y = 15.4$, $z = 0.252\times10^{3}$ $g = 82.8257$.

15. $h = \dfrac{x^{y+1}+e^{y-1}}{1+x|y-\operatorname{tg} z|}\left(1+|y-x|\right) + \dfrac{|y-x|^2}{2} - \dfrac{|y-x|^3}{3}.$

При $x = 2.444$, $y = 0.869\times10^{-2}$, $z = -0.13\times10^{3}$ $h = -0.49871$.

16. $y = \sqrt{cx} - 2.7\dfrac{|c|+|x|}{c^2x^2}\cdot e^{cx} + \cos\dfrac{(a+b)^2}{cx-b}$

$a = 3.7$; $b = 0.07$; $c = 1.5$; $x = 5.75$

17. $y = 4.5\dfrac{(a+b)^2}{(a-b)^2} - \sqrt{(a+b)(a-b)} + 10^{-1}\dfrac{\ln(a-b)}{\ln(a+b)} \cdot e^{x^2}$

$a = 7.5;\ b = 1.2;\ x = 0.5$

18. $y = 2.4\left|\dfrac{x^2+b}{a}\right| + (a-b)\sin^2(a-b) + 10^{-2}(x-b)$

$a = 5.1;\ b = 0.7;\ x = -0.05$

19. $y = \dfrac{ax - \sqrt{b}}{5.7(x^2+b^2)} - \dfrac{|x+b| - a^2}{x^2}\operatorname{tg}^2 b$

$a = 0.1;\ b = 2.4;\ x = -0.3$

20. $y = \sqrt{\dfrac{c - dx^2}{x}} + \dfrac{\ln(x^2+c)}{0.7x + ad} - \dfrac{10^{-2}}{c - dx^3}$

$a = 4.5;\ c = 7.4;\ d = -2.1;\ x = 0.15$

Задание 2. Разветвляющиеся алгоритмы

Цель задания: научиться пользоваться элементами управления для организации переключений (`RadioButton`). Написать и отладить программу разветвляющегося алгоритма.

2.1. Логические переменные и операции над ними

Переменные логического типа описываются посредством служебного слова `bool`. Они могут принимать только два значения – `False` (ложь) и `True` (истина). Результат `False` (ложь) и `True` (истина) возникает при использовании операций сравнения > (больше), < (меньше), ! (не равно), >= (больше или равно), <= (меньше или равно), == (равно). Описываются логические переменные следующим образом:

```
bool b;
```

В языке C# имеются логические операции, применяемые к переменным логического типа. Это операции *логического отрицания* (!), *логическое И* (&&) и *логическое ИЛИ* (||). Операция логического отрицания является *унарной операцией*. Результат операции ! есть `False`, если операнд истинен, и `True`, если операнд имеет значение «ложь». Так,

`!True` → `False` (неправда есть ложь),

`!False` → `True` (не ложь есть правда).

Результат операции *логическое И* (&&) есть истина, только если оба ее операнда истинны, и ложь во всех других случаях. Результат операции *логическое ИЛИ* (||) есть истина, если какой-либо из ее операндов истинен, и ложен только тогда, когда оба операнда ложны.

2.2. Условные операторы

Операторы ветвления позволяют изменить порядок выполнения операторов в программе. К операторам ветвления относятся условный оператор `if` и оператор выбора `switch`.

Условный оператор `if` используется для разветвления процесса обработки данных на два

направления. Он может иметь одну из форм: сокращенную или полную.

Форма сокращенного оператора `if`:

`if (B) S;` где `B` – логическое или арифметическое выражение, истинность которого проверяется; `S1, S2` – операторы.

При выполнении полной формы оператора `if` сначала вычисляется выражение `B`, затем анализируется его результат: если `B` истинно, то выполняется оператор `S1`, а оператор `S2` пропускается; если `B` ложно, то выполняется оператор `S2`, а `S1` – пропускается. Таким образом, с помощью полной формы оператора `if` можно выбрать одно из двух альтернативных действий процесса обработки данных.

Для примера вычислим значение функции:

$$y(x) = \begin{cases} \sin(x), \text{если } x \le a \\ \cos(x), \text{если } a < x < b \\ \operatorname{tg}(x), \text{если } x \ge b \end{cases}$$

Указанное выражение может быть запрограммировано в виде

```
if (x <= a)
    y = Math.Sin(x);
if ((x > a) && (x < b))
    y = Math.Cos(x); if (x >=
b)
    y = Math.Sin(x) / Math.Cos(x);
```

или с помощью оператора `else`:

```
if (x <= a)
    y = Math.Sin(x); else
    if (x < b)
        y = Math.Cos(x); else
        y = Math.Sin(x) / Math.Cos(x);
```

В С-подобных языках программирования, к которым относится и C#, операция сравнения представляется двумя знаками равенства, например:

```
if (a == b)
```

Оператор выбора `switch` предназначен для разветвления процесса вычислений по нескольким направлениям. Формат оператора:

```
switch (<выражение>)
{
case <константное_выражение_1>:
        [<оператор 1>];
        <оператор перехода>;
case <константное_выражение_2>:
        [<оператор 2>];
        <оператор перехода>;
```

```
        ...
case <константное_выражение_n>:
        [<оператор n>];
        <оператор перехода>;
[default:
        <оператор>;]
}
```

Выражение, записанное в квадратных скобках, является необязательным элементом в операторе switch. Если оно отсутствует, то может отсутствовать и оператор перехода.

Выражение, стоящее за ключевым словом switch, должно иметь арифметический, символьный или строковый тип. Все константные выражения должны иметь разные значения, но их тип должен совпадать с типом выражения, стоящим после switch или приводиться к нему. Ключевое слово case и расположенное после него константное выражение называют также *меткой* case.

Выполнение оператора начинается с вычисления выражения, расположенного за ключевым словом switch. Полученный результат сравнивается с меткой case. Если результат выражения соответствует метке case, то выполняется оператор, стоящий после этой метки, за которым обязательно должен следовать оператор перехода: break, goto, return и т. д. При использовании оператора break происходит выход из switch и управление передается оператору, следующему за switch. Если же используется оператор goto, то управление передается оператору, помеченному меткой, стоящей после goto. Наконец, оператор return завершает выполнение текущего метода. Если ни одно выражение case не совпадает со значением оператора switch, управление передается операторам, следующим за необязательной подписью default. Если подписи default нет, то управление передается за пределы оператора switch.

Пример использования оператора switch:

```
int dayOfWeek = 5;
switch (dayOfWeek)
{
case 1:
        MessageBox.Show("Понедельник");
        break;
case 2:
        MessageBox.Show("Вторник");
        break;
case 3:
        MessageBox.Show("Среда");
        break;
case 4:
```

```
            MessageBox.Show("Четверг");
            break;
    case 5:
            MessageBox.Show("Пятница");
            break;
    case 6:
            MessageBox.Show("Суббота");
            break;
    case 7:
            MessageBox.Show("Воскресенье");
            break;
    default:
            MessageBox.Show("Неверное значение!");
            break;
    }
```

Поскольку на момент выполнения оператора `switch` в этом примере переменная `dayOfWeek` равнялась 5, то будут выполнены операторы из блока `case` 5.

В ряде случаев оператор switch можно заменить несколькими операторами if, однако не всегда такая замена будет легче для чтения. Например, приведенный выше пример можно переписать так:

```
int dayOfWeek = 5; if (dayOfWeek ==1)
    MessageBox.Show("Понедельник");
    else
    if (dayOfWeek == 2) MessageBox.Show("Вторник");
    else
        if (dayOfWeek == 3)
            MessageBox.Show("Среда");
            else
            if (dayOfWeek == 4)
                MessageBox.Show("Четверг");
                else
                if (dayOfWeek == 5)
                    MessageBox.Show("Пятница");
                    else
                    if (dayOfWeek == 6)
                        MessageBox.Show("Суббота");
                        else
                        if (dayOfWeek == 7)
                            MessageBox.Show("Воскресенье");
                            else
                            MessageBox.Show("Неверное значение!");
```

2.3. Кнопки-переключатели

При создании программ в Visual Studio для организации разветвлений часто используются элементы управления в виде кнопок-переключателей (`RadioButton`). Состояние такой кнопки (включено–выключено) визуально отражается на форме, а в программе можно узнать его с помощью свойства `Checked`: если кнопка включена, это свойство будет содержать `True`, в противном случае `False`. Если пользователь выбирает один из вариантов переключателя в группе, все остальные автоматически отключаются.

Группируются радиокнопки с помощью какого-либо контейнера – часто это бывает элемент `GroupBox`. Радиокнопки, размещенные в разных контейнерах, образуют независимые группы.

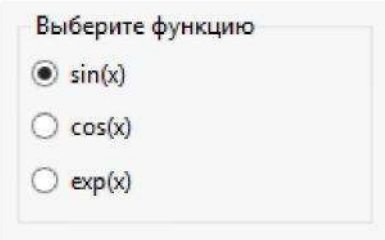

Рис. 2.1. Группа радиокнопок

```
if (radioButton1.Checked)
    MessageBox.Show("Выбрана функция: синус");
else if (radioButton2.Checked)
    MessageBox.Show("Выбрана функция: косинус");
else if (radioButton1.Checked)
    MessageBox.Show("Выбрана функция: экспонента");
```

2.4. Пример написания программы

Задание: ввести три числа – `x`, `y`, `z`. Вычислить

$$U=\begin{cases} y\cdot\sin(x)+z, & \text{при } z-x=0 \\ y\cdot e^{\sin(x)}-z, & \text{при } z-x<0 \\ y\cdot\sin(\sin(x))+z, & \text{при } z-x>0 \end{cases}$$

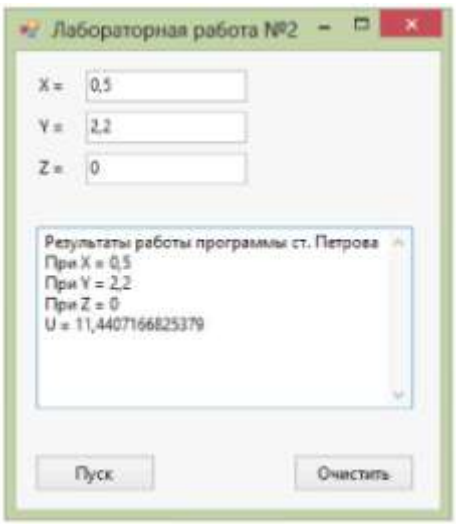

Рис. 2.2. Окно лабораторной работы

2.4.1. Создание формы

Создайте форму, в соответствии с рис. 2.2.

Разместите на форме элементы Label, TextBox и Button. Поле для вывода результатов также является элементом TextBox с установленным в True свойством Multiline и свойством ScrollBars установленным в Both.

2.4.2. Создание обработчиков событий

Обработчики событий создаются аналогично тому, как и в предыдущих лабораторных работах. Текст обработчика события нажатия на кнопку «Пуск» приведен ниже.

```
private void button1_Click(object sender, EventArgs e)
{
    // Получение исходных данных из TextBox
    double x = Convert.ToDouble(textBox2.Text);
    double y = Convert.ToDouble(textBox1.Text);
    double z = Convent.ToDouble(textBox3.Text);
    // Ввод исходных данных в окно результатов
    textBox4.Text = "Результаты работы программы " +
    "ст. Петрова И.И. " +
    Environment.NewLine;
    textBox4.Text += "При X = " + textBox2.Text +
                     Environment.NewLine;
    textBox4.Text += "При Y = " + textBoxl.Text +
                     Environment.NewLine;
    textBox4.Text += "При Z = " + textBox3.Text +
                     Environment.NewLine;
    // Вычисление выражения
    double u;
    if ((z - x) == 0)
       u = y * Math.Sin(x) * Math.Sin(x) + z;
       else
            if ((z - x) < 0)
               u = y * Math.Exp(Math.Sin(x)) - z;
            else
            u = y * Math.Sin(Math.Sin(x)) + z;
    // Вывод результата
    textBox4.Text += "U = " + u.ToString() +
    Environment.NewLine;
}
```

Запустите программу и убедитесь в том, что все ветви алгоритма выполняются правильно.

Варианты задания

По указанию преподавателя выберите индивидуальное задание из нижеприведенного списка. В качестве *f(x)* использовать по выбору: *sh(x)*, x^2, e^x. Отредактируйте вид формы и текст

программы, в соответствии с полученным заданием.

Усложненный вариант задания для продвинутых студентов: с помощью радиокнопок (`RadioButton`) дать пользователю возможность во время работы программы выбрать одну из трех приведенных выше функций.

1. $a=\begin{cases}(f(x)+y)^2-\sqrt{f(x)y}, & xy\rangle 0\\ (f(x)+y)^2+\sqrt{|f(x)y|}, & xy\langle 0\\ (f(x)+y)^2+1, & xy=0.\end{cases}$

2. $b=\begin{cases}\ln(f(x))+(f(x)^2+y)^3, & x/y\rangle 0\\ \ln|f(x)/y|+(f(x)+y)^3, & x/y\langle 0\\ (f(x)^2+y)^3, & x=0\\ 0, & y=0.\end{cases}$

3. $c=\begin{cases}f(x)^2+y^2+\sin(y), & x-y=0\\ (f(x)-y)^2+\cos(y), & x-y\rangle 0\\ (y-f(x))^2+\operatorname{tg}(y), & x-y\langle 0\end{cases}$

4. $d=\begin{cases}(f(x)-y)^3+\operatorname{arctg}(f(x)), & x\rangle y\\ (y-f(x))^3+\operatorname{arctg}(f(x)), & y\rangle x\\ (y+f(x))^3+0.5, & y=x.\end{cases}$

5. $e=\begin{cases}i\sqrt{f(x)}, & i-\textit{нечетное},\ x\rangle 0\\ i/2\sqrt{|f(x)|}, & i-\textit{четное},\ x\langle 0\\ \sqrt{|if(x)|}, & \textit{иначе}.\end{cases}$

6. $g=\begin{cases}e^{f(x)-|b|}, & 0.5\langle xb\langle 10\\ \sqrt{|f(x)+b|}, & 0.1\langle xb\langle 0.5\\ 2f(x)^2, & \textit{иначе}.\end{cases}$

7. $s=\begin{cases}e^{f(x)}, & 1\langle xb\langle 10\\ \sqrt{|f(x)+4*b|}, & 12\langle xb\langle 40\\ bf(x)^2, & \textit{иначе}.\end{cases}$

8. $j=\begin{cases}\sin(5f(x)+3m|f(x)|), & -1\langle m\langle x\\ \cos(3f(x)+5m|f(x)|), & x\rangle m\\ (f(x)+m)^2, & x=m.\end{cases}$

9. $l=\begin{cases}2f(x)^3+3p^2, & x\rangle |p|\\ |f(x)-p|, & 3\langle x\langle |p|\\ (f(x)-p)^2, & x=|p|.\end{cases}$

10. $k=\begin{cases}\ln(|f(x)|+|q|), & |xq|\rangle 10\\ e^{f(x)+q}, & |xq|\langle 10\\ f(x)+q, & |xq|=10\end{cases}$

11. $m=\dfrac{\max(f(x),y,z)}{\min(f(x),y)}+5.$

12. $n=\dfrac{\min(f(x)+y,y-z)}{\max(f(x),y)}.$

13. $p=\dfrac{|\min(f(x),y)-\max(y,z)|}{2}.$

14. $q=\dfrac{\max(f(x)+y+z,xyz)}{\min(f(x)+y+z,xyz)}.$

15. $c=\begin{cases}f(\sin(x))^2+\sin(f(y)), & x-y=0\\ (f(\cos(x)))+\cos(f(y)), & x-y\rangle 0\\ (y-f(\operatorname{tg}(x)))^2+\operatorname{tg}(y), & x-y\langle 0.\end{cases}$

16. $a=\begin{cases}\dfrac{(\mathrm{ax}^2+2)}{(\mathrm{x}^2+1)}f(x), & 1<|\mathrm{x}|<3,\\ a^2+f(x), & |\mathrm{x}|\ \geq 3\\ \mathrm{ax}\dfrac{\mathrm{f(x)}}{(\mathrm{x}+2)}, & |\mathrm{x}|\ \leq 1.\end{cases}$

17. $c=\begin{cases}f(x)^3-y^3\cdot\cos(x), & x+y=0\\ (f(x)\cdot y)^2-\cos(y), & x+y\rangle 0\\ (y\cdot f(x))^2+\pi, & x+y\langle 0.\end{cases}$

18. $k=\begin{cases}\ln(|f(x^2)|+|k|), & |x\cdot k|\rangle 10\\ \pi^{f(x)+q}, & |x\cdot k|\langle 10\\ f(x)-k, & |x\cdot k|=10\end{cases}$

19. $c=\begin{cases}\sin(f(x))+\cos(f(y)), & x-y=0\\ \operatorname{tg}(f(x+y), & x-y\rangle 0\\ \sin^2(f(x))+\cos^2(f(y)), & x-y\langle 0.\end{cases}$

20. $r=\max(\min(f(x),y),z)$

Задание 3. Графики функций

Цель задания: изучить возможности построения графиков с помощью элемента управления Chart. Написать и отладить программу построения на экране графика заданной функции.

3.1. Как строится график с помощью элемента управления 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. 3.1).

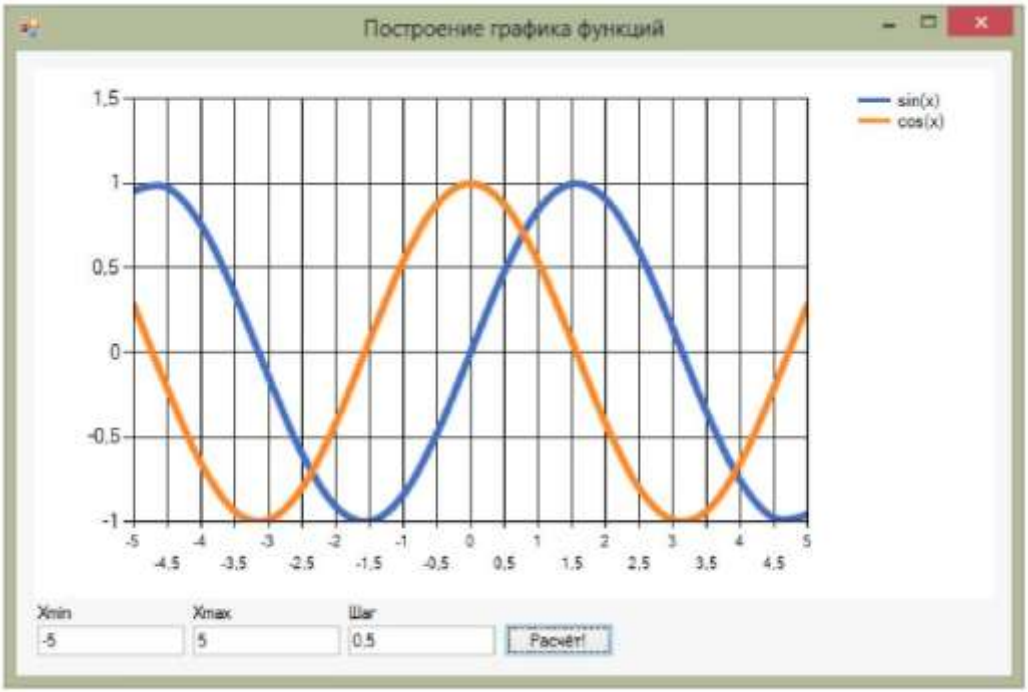

Рис. 3.1. Окно программы с элементом управления

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` элемента управления Chart 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 переданную таблицу в виде всевозможных графиков или диаграмм. В элементе управления Chart 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 графика.

3.2. 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 поместить на форму сам элемент управления Chart. Он располагается в панели элементов в разделе *Данные*.

Список графиков хранится в свойстве Series, который можно изменить, выбрав соответствующий пункт в окне свойств. Поскольку на одном поле требуется вывести два отдельных графика функций, нужно добавить еще один элемент. Оба элемента, и существующий и добавленный, нужно соответствующим образом настроить: изменить тип диаграммы ChartType на Spline. Здесь же можно изменить подписи к графикам с абстрактных *Series1* и *Series2* на sin(*x*) и cos(*x*) – за это отвечает свойство Legend. Наконец, с помощью свойства BorderWidth можно сделать линию графика потолще, а затем поменять цвет линии с помощью свойства Color.

Ниже приведен текст обработчика нажатия кнопки «Расчет!», который выполняет все требуемые настройки и расчеты 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);
```

}

3.3. Выполнение индивидуального задания

Постройте график функции для своего варианта . Таблицу данных получить путем изменения параметра X с шагом dx. Добавьте второй график для произвольной функции.

1) $y = 10^{-2}bc / x + \cos\sqrt{a^3 x}$,
$x_0 = -1.5; x_k = 3.5; dx = 0.5;$
$a = -1.25; b = -1.5; c = 0.75;$

2) $y = 1.2(a-b)^3 e^{x^2} + x$,
$x_0 = -0.75; x_k = -1.5; dx = -0.05;$
$a = 1.5; b = 1.2;$

3) $y = 10^{-1} ax^3 tg(a - bx)$,
$x_0 = -0.5; x_k = 2.5; dx = 0.05;$
$a = 10.2; b = 1.25;$

4) $y = ax^3 + \cos^2(x^3 - b)$,
$x_0 = 5.3; x_k = 10.3; dx = 0.25;$
$a = 1.35; b = -6.25;$

5) $y = x^4 + \cos(2 + x^3 - d)$,
$x_0 = 4.6; x_k = 5.8; dx = 0.2;$
$d = 1.3;$

6) $y = x^2 + tg(5x + b / x)$,
$x_0 = -1.5; x_k = -2.5; dx = -0.5;$
$b = -0.8;$

7) $y = 9(x + 15\sqrt{x^3 + b^3})$,
$x_0 = -2.4; x_k = 1; dx = 0.2;$
$b = 2.5;$

8) $y = 9x^4 + \sin(57.2 + x)$,
$x_0 = -0.75; x_k = -2.05; dx = -0.2;$

9) $y = 0.0025bx^3 + \sqrt{x + e^{0.82}}$,
$x_0 = -1; x_k = 4; dx = 0.5;$
$b = 2.3;$

10) $y = x \cdot \sin(\sqrt{x + b - 0.0084})$,
$x_0 = -2.05; x_k = -3.05; dx = -0.2;$
$b = 3.4;$

11) $y = x + \sqrt{\left|x^3 + a - be^x\right|}$,
$x_0 = -4; x_k = -6.2; dx = -0.2;$
$a = 0.1;$

12) $y = 9(x^3 + b^3)tgx$,
$x_0 = 1; x_k = 2.2; dx = 0.2;$
$b = 3.2;$

13) $y = |x - b|^{1/2} / \left|b^3 - x^3\right|^{3/2} + \ln|x - b|$,
$x_0 = -0.73; x_k = -1.73; dx = -0.1;$
$b = -2;$

14) $y = (x^{5/2} - b)\ln(x^2 + 12.7)$,
$x_0 = 0.25; x_k = 5.2; dx = 0.3;$
$b = 0.8;$

15) $y = 10^{-3}|x|^{5/2} + \ln|x + b|$,
$x_0 = 1.75; x_k = -2.5; dx = -0.25;$
$b = 35.4;$

16) $y = 15.28|x|^{-3/2} + \cos(\ln|x| + b)$,
$x_0 = 1.23; x_k = -2.4; dx = -0.3;$
$b = 12.6;$

17) $y = 0.00084(\ln|x|^{5/4} + b)/(x^2 + 3.82)$,
$x_0 = -2.35; x_k = -2; dx = 0.05;$
$b = 74.2;$

18) $y = 0.8 \cdot 10^{-5}(x^3 + b^3)^{7/6}$,
$x_0 = -0.05; x_k = 0.15; dx = 0.01;$
$b = 6.74;$

19) $y = (\ln(\sin(x^3 + 0.0025)))^{3/2} + 0.8 \cdot 10^{-3}$,
$x_0 = 0.12; x_k = 0.64; dx = 0.2;$

20) $y = a + x^{2/3}\cos(x + e^x)$,
$x_0 = 5.62; x_k = 15.62; dx = 0.5;$
$a = 0.41$