

## **ЛАБОРАТОРНАЯ РАБОТА № 2.**

### **ЛИНЕЙНЫЕ АЛГОРИТМЫ**

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

#### **2.1. 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`.

## 2.2. Работа с проектом

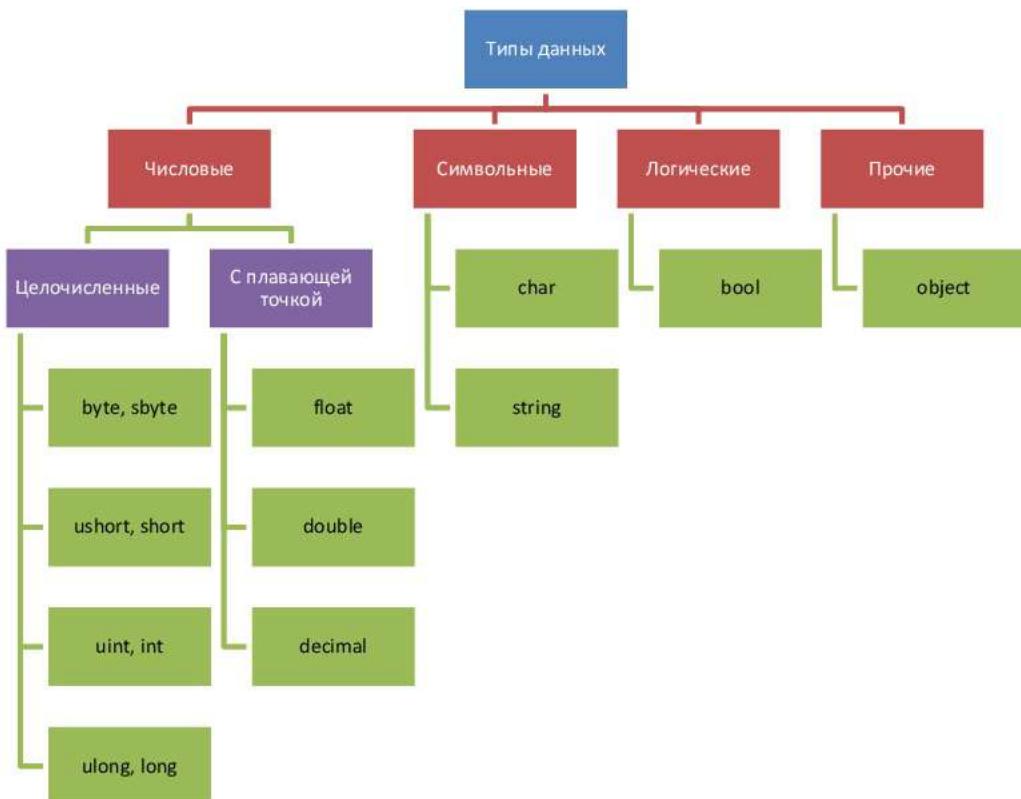
Проект в Visual Studio состоит из многих файлов, и создание сложной программы требует хранения каждого проекта в отдельной папке. При создании нового проекта Visual Studio по умолчанию сохраняет его в отдельной папке. Рекомендуется создать для себя свою папку со своей фамилией внутри папки своей группы, 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.

## 2.3. 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 зна-

чения определяет те операции, которые разрешается выполнять над ним. Операция, разрешенная для одного типа данных, может оказаться недопустимой для другого.



*Рис. 2.1. 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, характер-

ного 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;
```

## 2.4. 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 `Convert.ToDouble()`) 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 `ToString()`, который возвращает в результате строку с символьным представлением значения. Вывод данных можно осуществлять в элементы `TextBox` или `Label`, используя свойство `Text`. Например:

```
private void button1_Click(object sender, EventArgs e)
{
    string s = textBox1.Text;
    int a = int.Parse(s);
    int b = a * a;
    label1.Text = b.ToString();
}
```

## 2.5. Арифметические действия и стандартные функции

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

- умножение ( $\times$ );
- 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.

## 2.6. Пример написания программы

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

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

Панель диалога программы организовать в виде, представленном на рис. 2.2.

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.

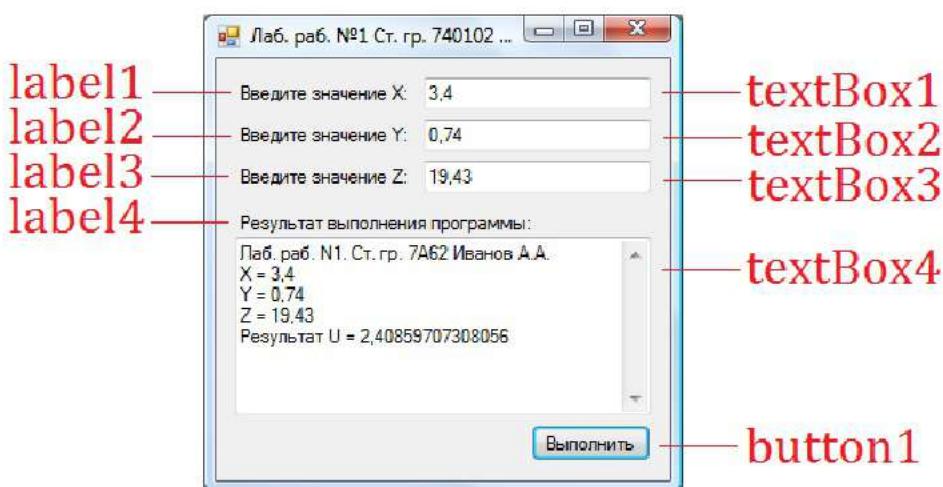


Рис. 2.2. Внешний вид программы

Информация, которая отображается построчно в окне, находится в массиве строк `Lines`, каждая строка которого имеет тип `string`. 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.

## 2.7. 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.

### Индивидуальные задания

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

При  $x = 14.26$ ,  $y = -1.22$ ,  $z = 3.5 \times 10^{-2}$   $t = 0.564849$ .

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

При  $x = -4.5$ ,  $y = 0.75 \times 10^{-4}$ ,  $z = 0.845 \times 10^2$   $u = -55.6848$ .

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

При  $x = 3.74 \times 10^{-2}$ ,  $y = -0.825$ ,  $z = 0.16 \times 10^2$ ,  $v = 1.0553$ .

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

При  $x = 0.4 \times 10^4$ ,  $y = -0.875$ ,  $z = -0.475 \times 10^{-3}$   $w = 1.9873$ .

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

При  $x = -15.246$ ,  $y = 4.642 \times 10^{-2}$ ,  $z = 20.001 \times 10^2$   $\alpha = -182.036$ .

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

При  $x = 16.55 \times 10^{-3}$ ,  $y = -2.75$ ,  $z = 0.15$   $\beta = -38.902$ .

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

При  $x = 0.1722$ ,  $y = 6.33$ ,  $z = 3.25 \times 10^{-4}$   $\gamma = -172.025$ .

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

При  $x = -2.235 \times 10^{-2}$ ,  $y = 2.23$ ,  $z = 15.221$   $\varphi = 39.374$ .

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

При  $x=1.825 \times 10^2$ ,  $y=18.225$ ,  $z=-3.298 \times 10^{-2}$   $\psi=1.2131$ .

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

При  $x=3.981 \times 10^{-2}$ ,  $y=-1.625 \times 10^3$ ,  $z=0.512$   $a=1.26185$ .

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

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

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

При  $x=3.251$ ,  $y=0.325$ ,  $z=0.466 \times 10^{-4}$   $c=4.025$ .

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

При  $x=17.421$ ,  $y=10.365 \times 10^{-3}$ ,  $z=0.828 \times 10^5$   $f=0.33056$ .

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

При  $x=12.3 \times 10^{-1}$ ,  $y=15.4$ ,  $z=0.252 \times 10^3$   $g=82.8257$ .

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

При  $x=2.444$ ,  $y=0.869 \times 10^{-2}$ ,  $z=-0.13 \times 10^3$   $h=-0.49871$ .

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

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

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

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

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

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

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

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

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

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