# Практическое занятие № 2. Программирование линейных алгоритмов. Объявление переменных. Явное и неявное преобразование типов данных

**1 Цель занятия**

Получить практические навыки программирования линейных алгоритмов в виде приложений Windows Forms, определения типов переменных, объявления переменных и использования методов преобразования типов переменных

**2 Перечень оборудования и программного обеспечения**

Персональный компьютер

Microsoft Office (Word)

Microsoft Visual Studio

**3 Краткие теоретические сведения**

**3.1 Линейный алгоритм** – это набор команд, выполняемых последовательно во времени, друг за другом.

Для описания линейных алгоритмов используются три оператора алгоритмического языка:

- оператор вызова процедуры **ввода**;

- оператор **присваивания**;

- оператор вызова процедуры **вывода**.

начало

Ввод исходных данных

Обработка (вычисления)

Вывод результата

конец

**3.2 Правила синтаксиса языка С#**

При написании программы придерживаются следующих правил:

{ } операторные скобки объединяют несколько операторов в один блок;

; конец оператора;

, разделитель при перечислении констант, переменных;

( ) содержат параметры функций или операторов.

С# представляет несколько механизмов *комментирования* кода:

• построчное //

• многострочное /\* \*/

• позволяющее автоматически генерировать документацию в XML-формате (начинается с символов ///, за которыми следуют специальные тэги).

**3.3 Система типов языка С#. Синтаксис объявления переменных**

Типы разделяются на встроенные типы и типы, определенные программистом. Встроенные типы изначально принадлежат языку программирования и составляют его базис. На их основе программист может строить собственные, им самим определенные типы данных.

Типы данных также разделяются на статические и динамические типы. Для данных статического типа память отводится в момент объявления, требуемый размер данных (памяти) известен при их объявлении. Для данных динамического типа размер данных в момент объявления обычно неизвестен и память им выделяется динамически по запросу в процессе выполнения программы.

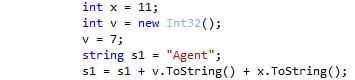
|  |  |  |  |
| --- | --- | --- | --- |
| **Логический тип** | | | |
| **Имя типа** | **Системный тип** | **Значения** | **Размер** |
| bool | System.Boolean | true, false | 8 бит |
| **Арифметические целочисленные типы** | | | |
| **Имя типа** | **Системный тип** | **Диапазон** | **Размер** |
| sbyte | System.SByte | -128 — 127 | Знаковое, 8 бит |
| byte | System.Byte | 0 — 255 | Беззнаковое, 8 бит |
| short | System.Short | -32768 —32767 | Знаковое, 16 бит |
| ushort | System.UShort | 0 — 65535 | Беззнаковое, 16 бит |
| int | System.Int32 | (-2\*10^9 — 2\*10^9) | Знаковое, 32 бит |
| uint | System.UInt32 | (0 — 4\*10^9) | Беззнаковое, 32 бит |
| long | System.Int64 | (-9\*10^18 — 9\*10^18) | Знаковое, 64 бит |
| ulong | System.UInt64 | (0— 18\*10^18) | Беззнаковое, 64 бит |
| **Арифметический тип с плавающей точкой** | | | |
| **Имя типа** | **Системный тип** | **Диапазон** | **Точность** |
| float | System.Single | +1.5\*10^-45 -/+3.4\*10^38 | 7 цифр |
| double | System.Double | +5.0\*10^-324 -/+1.7\*10^308 | 15-16 цифр |
| **Арифметический тип с фиксированной точкой** | | | |
| **Имя типа** | **Системный тип** | **Диапазон** | **Точность** |
| decimal | System.Decimal | +1.0\*10^-28 - +7.9\*10^28 | 28-29 значащих цифр |
| **Символьные типы** | | | |
| **Имя типа** | **Системный тип** | **Диапазон** | **Точность** |
| char | System.Char | U+0000 - U+ffff | 16 бит Unicode символ |
| string | System.String | Строка из символов Unicode |

Общий синтаксис объявления переменных в C# похож на синтаксис других языков, хотя и имеет ряд отличий. Общая структура объявления:

[<модификаторы>] <тип> <объявители>;

При объявлении переменных чаще всего задаются модификаторы public, private, static и другие. Модификаторы могут не указываться в объявлении, но задание типа необходимо всегда. Кроме этого, любые действия могут выполняться только с инициализированными переменными.

**Пример 1:**

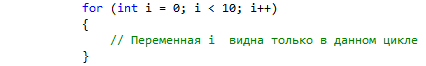


Чтобы присваивание было допустимым, типы переменных и выражения должны быть согласованными.

**Время жизни и область видимости переменных.** В языке C# меньше возможностей для объявления переменных, чем в любом другом языке. В языке нет настоящих глобальных переменных. Первая важнейшая роль переменных, - они задают свойства структур, интерфейсов, классов. В языке C# такие переменные называются полями (fields). Время существования и область видимости полей определяются объектом, которому они принадлежат.

Переменные, объявленные на уровне процедуры, называются локальными. В некоторых языках, например в Паскале, локальные переменные должны быть объявлены в вершине процедурного блока. В C#, также как и в языке C++, принята другая стратегия. Переменную можно объявлять в любой точке процедурного блока. Область ее видимости распространяется от точки объявления до конца процедурного блока.

**Пример 2:**



**Автоматическое преобразование типов.** Когда данные одного типа присваиваются переменной другого типа, неявное преобразование типов происходит автоматически при следующих условиях:

- оба типа совместимы

- диапазон представления чисел целевого типа шире, чем исходного.

Если оба эти условия удовлетворяются, то происходит расширяющее преобразование. Например, тип int достаточно крупный, чтобы вмещать в себя все действительные значения типа byte, а кроме того, оба типа, int и byte, являются совместимыми целочисленными типами, и поэтому для них вполне возможно неявное преобразование.

Числовые типы, как целочисленные, так и с плавающей точкой, вполне совместимы друг с другом для выполнения расширяющих преобразований.

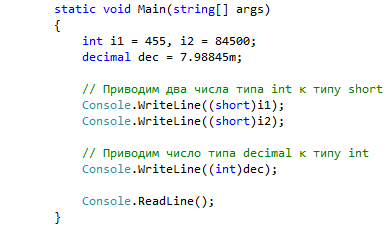
**Приведение несовместимых типов.** Неявные преобразования типов не способны удовлетворить все потребности в программировании, поскольку допускают лишь расширяющие преобразования совместимых типов. Во всех остальных случаях приходится обращаться к приведению типов. Приведение — это команда компилятору преобразовать результат вычисления выражения в указанный тип. А для этого требуется явное преобразование типов. Ниже приведена общая форма приведения типов:

(<целевой\_тип>) <выражение>

Здесь <целевой\_тип> обозначает тот тип, в который желательно преобразовать указанное выражение.

Если приведение типов приводит к сужающему преобразованию, то часть информации может быть потеряна. Например, в результате приведения типа long к типу int часть информации потеряется, если значение типа long окажется больше диапазона представления чисел для типа int, поскольку старшие разряды этого числового значения отбрасываются. Когда же значение с плавающей точкой приводится к целочисленному, то в результате усечения теряется дробная часть этого числового значения. Так, если присвоить значение 1,23 целочисленной переменной, то в результате в ней останется лишь целая часть исходного числа (1), а дробная его часть (0,23) будет потеряна.

**Пример 3:**

****

**Объект Convert.** Для того чтобы преобразовать переменные к нужному типу данных используется объект Convert с определенным методом:

<переменная целого типа> = Convert.ToInt32(<строковая переменная>); - преобразует строковое представление числа в эквивалентное ему 32-битовое целое число со знаком.

<переменная действ. типа> = Convert.ToDouble(<строковая переменная>);

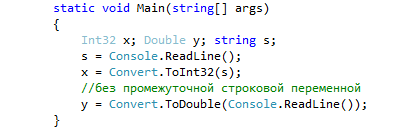
<переменная логич. типа> = Convert.ToBoolean(<строковая переменная>);

<переменная целого типа> = Convert.ToByte(<строковая переменная>);

<переменная символ. типа> = Convert.ToChar(<строковая переменная>);

<переменная целого типа> = Convert.ToInt64(<строковая переменная>);

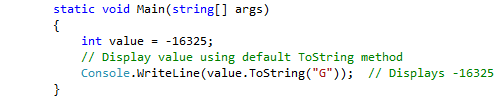
**Пример 4:**



Типы вводимых значений должны совпадать с типами указанных переменных, иначе возникает ошибка. Поэтому нужно внимательно следить за правильностью вводимых данных.

Метод ToString() преобразует указанное значение в его эквивалентное строковое представление.

**Пример 5.** Преобразование значения заданного 32-разрядного знакового целого числа в эквивалентное строковое представление:



**3.4 Математические операции, константы и методы языка С#**

|  |  |
| --- | --- |
| **Категории операций** | **Операции** |
| Арифметические | + - \* / % |
| Логические | ! && || |
| Строковые (конкатенаторы) | + |
| Инкремент, декремент | ++ -- |
| Сравнения | == != < > <= >= |
| Присваивание | = += -= \*= /= %= &= |= ^= |
| Приведение типа | тип() |
| Создание объекта | new() |

**Класс Math** предоставляет константы и статические методы для тригонометрических, логарифмических и иных общих математических функций.

**Константы:**

|  |  |
| --- | --- |
| E | Представляет основание натурального логарифма, определяемое константой *e* |
| PI | Представляет отношение длины окружности к ее диаметру, определяемое константой *π*. |

**Тригонометрические методы**:

|  |  |
| --- | --- |
| [Cos](http://www.cplusplus.com/reference/clibrary/cmath/cos/)() | Вычислить косинус |
| [Sin](http://www.cplusplus.com/reference/clibrary/cmath/sin/)() | Вычислить синус |
| [Tan](http://www.cplusplus.com/reference/clibrary/cmath/tan/)() | Вычислить тангенс |

**Экспоненциальные и логарифмические методы:**

|  |  |
| --- | --- |
| [Exp](http://www.cplusplus.com/reference/clibrary/cmath/exp/)() | Вычислить экспоненциальную функцию |
| [Log](http://www.cplusplus.com/reference/clibrary/cmath/log/)() | Вычислить натуральный логарифм |
| [Log10](http://www.cplusplus.com/reference/clibrary/cmath/log10/)() | Вычислить десятичный логарифм |

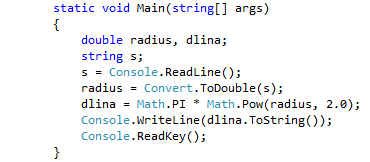
**Степенные методы:**

|  |  |
| --- | --- |
| [Pow](http://www.cplusplus.com/reference/clibrary/cmath/pow/)() | Возвести в степень |
| [Sqrt](http://www.cplusplus.com/reference/clibrary/cmath/sqrt/)() | Вычислить корень |

**Методы округления, абсолютного, максимального и минимального значений:**

|  |  |
| --- | --- |
| C[eil](http://www.cplusplus.com/reference/clibrary/cmath/ceil/)ing() | Округлить до большего целого значения |
| [Floor](http://www.cplusplus.com/reference/clibrary/cmath/floor/)() | Округлить до меньшего целого значения |
| Round() | Округлить значение двойной точности с плавающей запятой до заданного количества дробных разрядов |
| [Abs](http://www.cplusplus.com/reference/clibrary/cmath/fabs/)() | Вычислить абсолютное значение (модуль) |
| [Max](http://www.cplusplus.com/reference/clibrary/cmath/fmod/)() | Определить большее из двух чисел |
| [Min](http://www.cplusplus.com/reference/clibrary/cmath/fmod/)() | Определить меньшее из двух чисел |

**Пример 6:**



**3.5 Обработка пользовательских ошибок в языке С#**

Механизм обработки ошибок С# позволяет закодировать обработку для каждого типа ошибочных условий, а также отделить код, потенциально порождающий ошибки, от кода, обрабатывающего их.

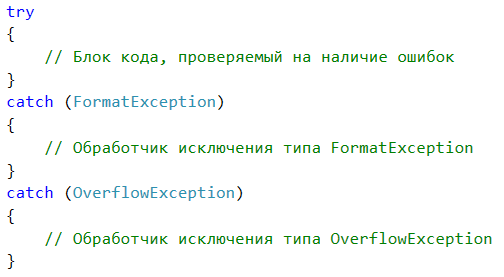
**Исключения (exceptions).** Под "исключениями" или исключительными ситуациями подразумеваются пользовательские ошибки, программные ошибки и ошибки времени выполнения.

Пользовательские ошибки обычно возникают из-за тех, кто запускает приложение, а не тех, кто его создает. Например, ввод конечным пользователем в текстовом поле неправильно оформленной строки может привести к генерации ошибки подобного рода, если в коде не была предусмотрена возможность обработки некорректного ввода.

Программными ошибками обычно называют аномалии, которые могут возникать во время выполнения программы и которые трудно, а порой и вообще невозможно, предусмотреть во время программирования приложения. К числу таких возможных исключений относятся попытки подключения к базе данных, которой больше не существует, попытки открытия поврежденного файла или попытки установки связи с машиной, которая в текущий момент находится в автономном режиме. В каждом из этих случаев программист (и конечный пользователь) мало что может сделать с подобными "исключительными" обстоятельствами.

В библиотеках базовых классов .NET определено множество различных исключений, таких как ArgumentOutOfRangeException, FormatException, IndexOutOfRangeException, FileNotFoundException и т.д.

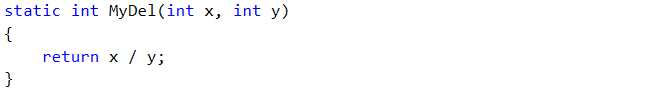
**Try и catch.** Основу обработки исключительных ситуаций в С# составляет пара ключевых слов try и catch. Эти ключевые слова действуют совместно и не могут быть использованы порознь. Ниже приведена общая форма определения блоков try/catch для обработки исключительных ситуаций:

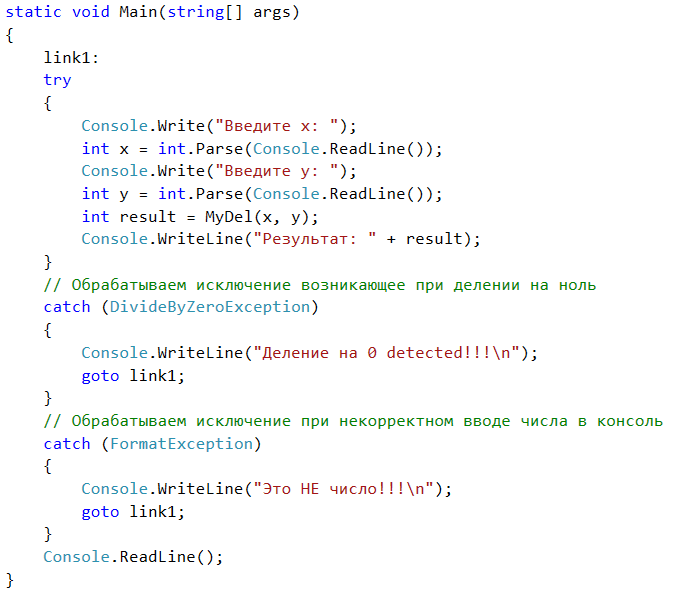


...

где FormatException и OverflowException — это типы возникающих исключительных ситуаций. Когда исключение генерируется оператором try, оно перехватывается составляющим ему пару оператором catch, который затем обрабатывает это исключение. В зависимости от типа исключения выполняется и соответствующий оператор catch. Так, если типы генерируемого исключения и того, что указывается в операторе catch, совпадают, то выполняется именно этот оператор, а все остальные пропускаются.

**Пример 7.** Обработка исключения, возникающего при делении числа на 0 (DivideByZeroException), а так же пользовательской ошибки при вводе не числа (FormatException):





**Последствия неперехвата исключений.** Перехват одного из стандартных исключений, как в приведенном выше примере, дает еще одно преимущество: он исключает аварийное завершение программы. Как только исключение будет сгенерировано, оно должно быть перехвачено каким-то фрагментом кода в определенном месте программы. Вообще говоря, если исключение не перехватывается в программе, то оно будет перехвачено исполняющей системой. Но дело в том, что исполняющая система выдаст сообщение об ошибке и прервет выполнение программы.

Такие сообщения об ошибках полезны для отладки программы, но, по меньшей мере, нежелательны при ее использовании на практике! Именно поэтому так важно организовать обработку исключительных ситуаций в самой программе.

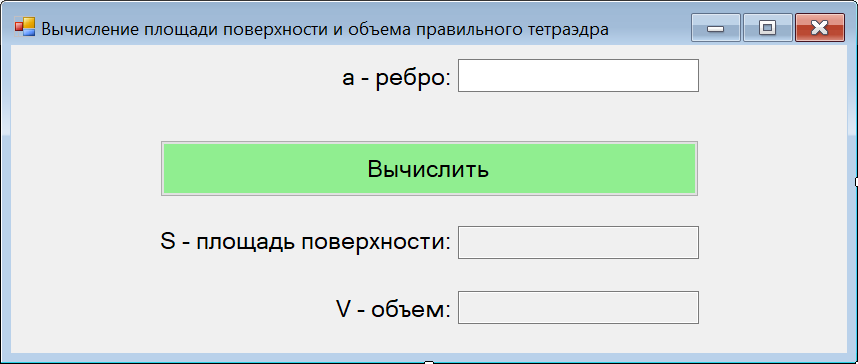
**Часто встречающиеся исключения**

|  |  |  |
| --- | --- | --- |
| **№** | **Исключение** | **Описание** |
| 1 | FormatException | Исключение, выбрасываемое, если формат аргумента не соответствует спецификациям параметра вызываемого метода |
| 2 | DivideByZeroException | Исключение, выбрасываемое при попытке деления целого или дробного числа на нуль |
| 3 | NotFiniteNumberException | Исключение, которое выбрасывается, когда значение с плавающей запятой является плюс бесконечностью, минус бесконечностью или не является числовым (NaN) |
| 4 | OverflowException | Исключение, которое выбрасывается, когда при выполнении арифметических операций, операций приведения типов и преобразования происходит переполнение. |
| 5 | FileNotFoundException | Это исключение создается, когда попытка доступа к файлу, не существующему на диске, заканчивается неудачей. |
| 6 | ApplicationException | Это исключение выбрасывается при происхождении устранимой ошибки приложения |

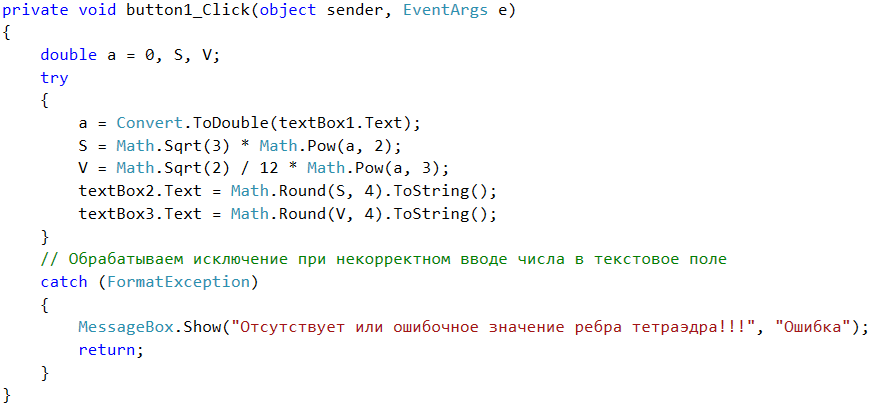
**Задание.** Дано ребро *a* правильного тетраэдра. Найти его площадь поверхности *S* и объем *V*.

**Решение.**

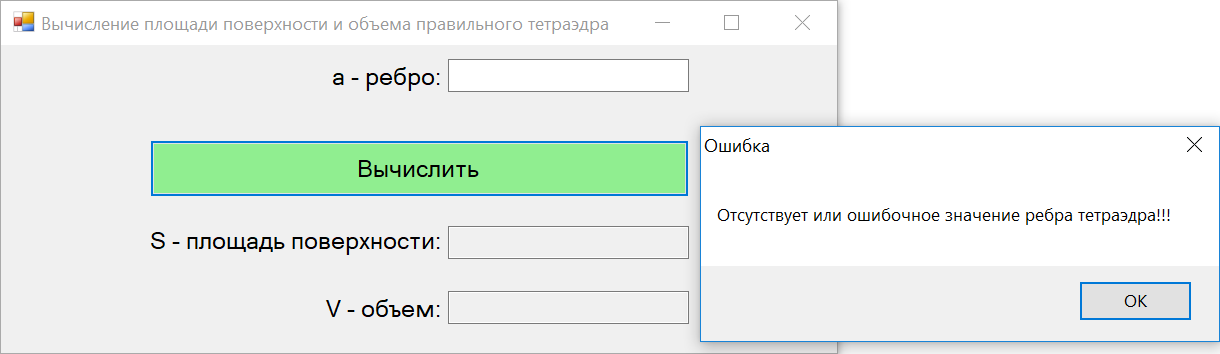
а) Интерфейс приложения Windows Forms:

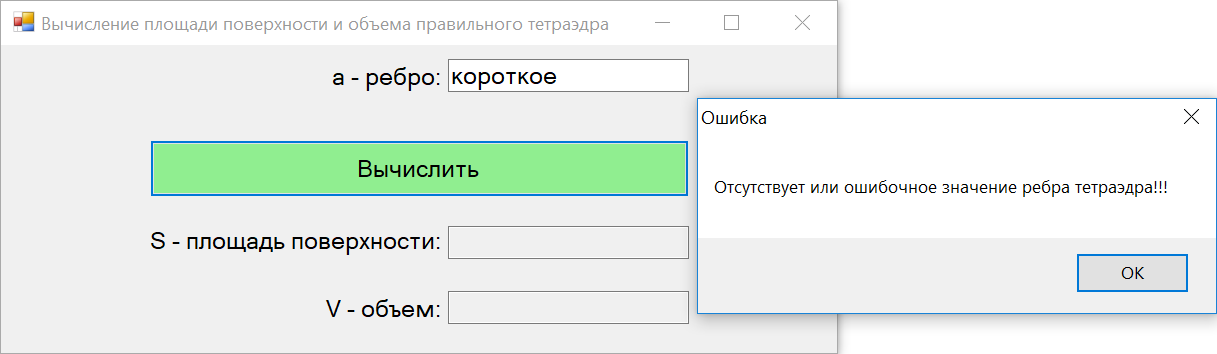


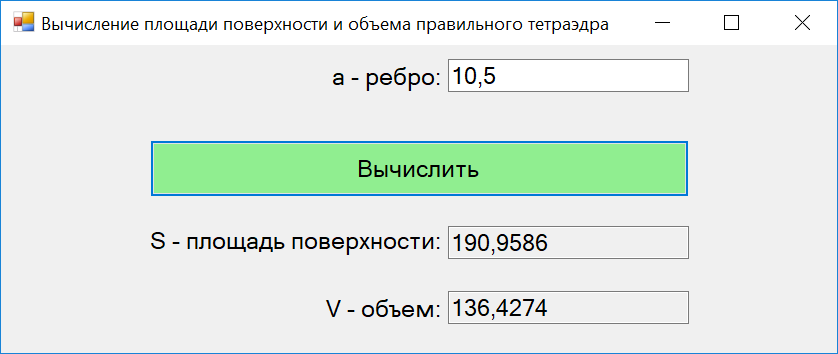
Метод button1\_Click(), содержащий код приложения:



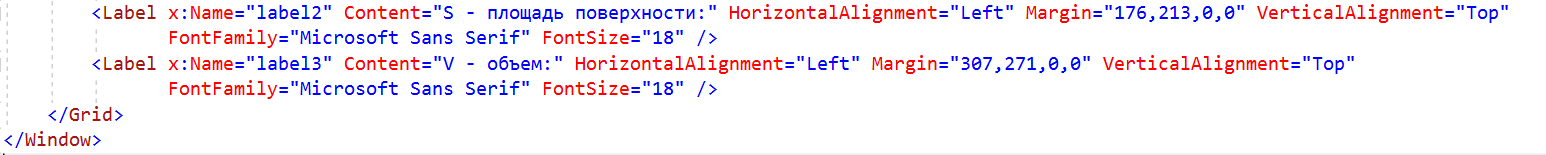
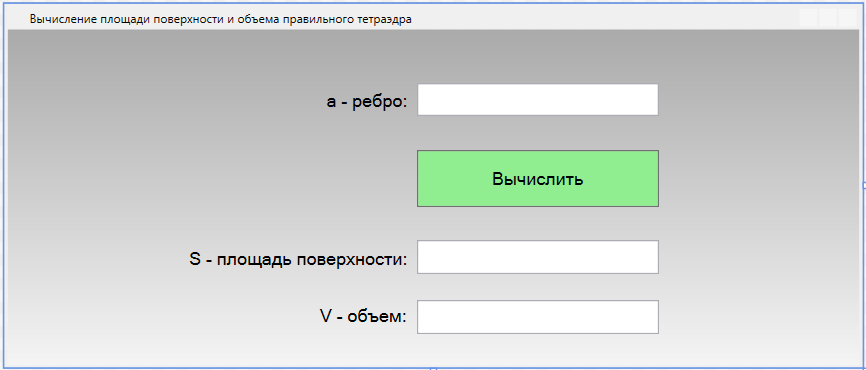
Скрины отлаженной программы:



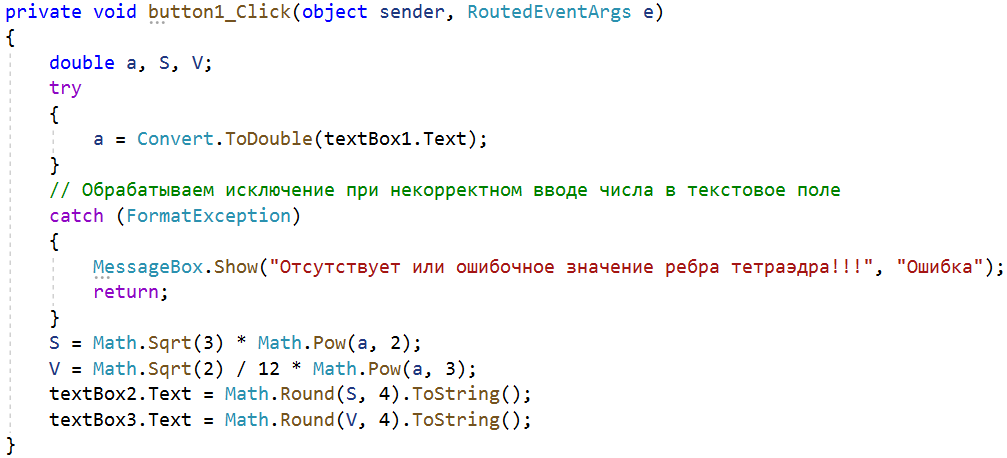




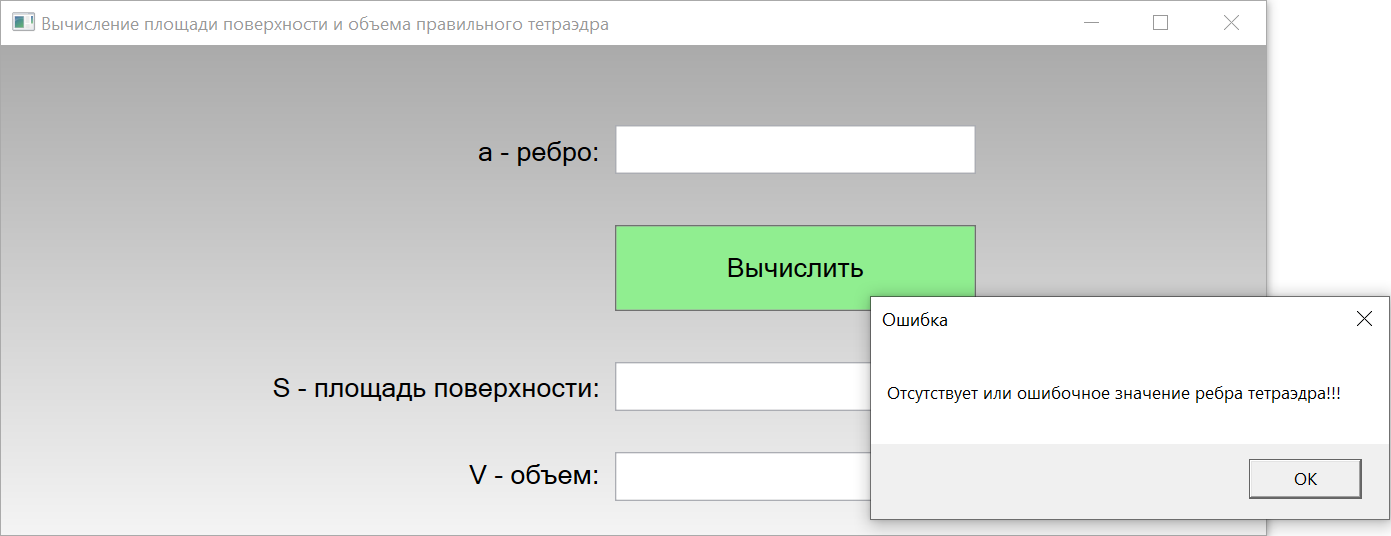
б) Интерфейс приложения WPF:

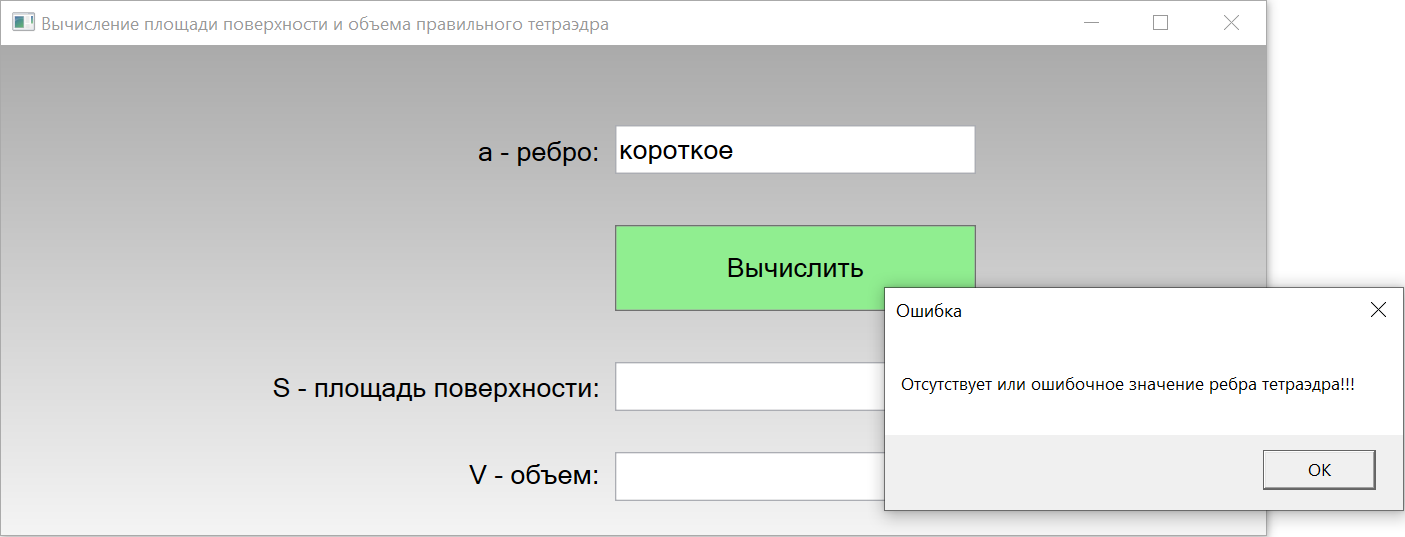
 

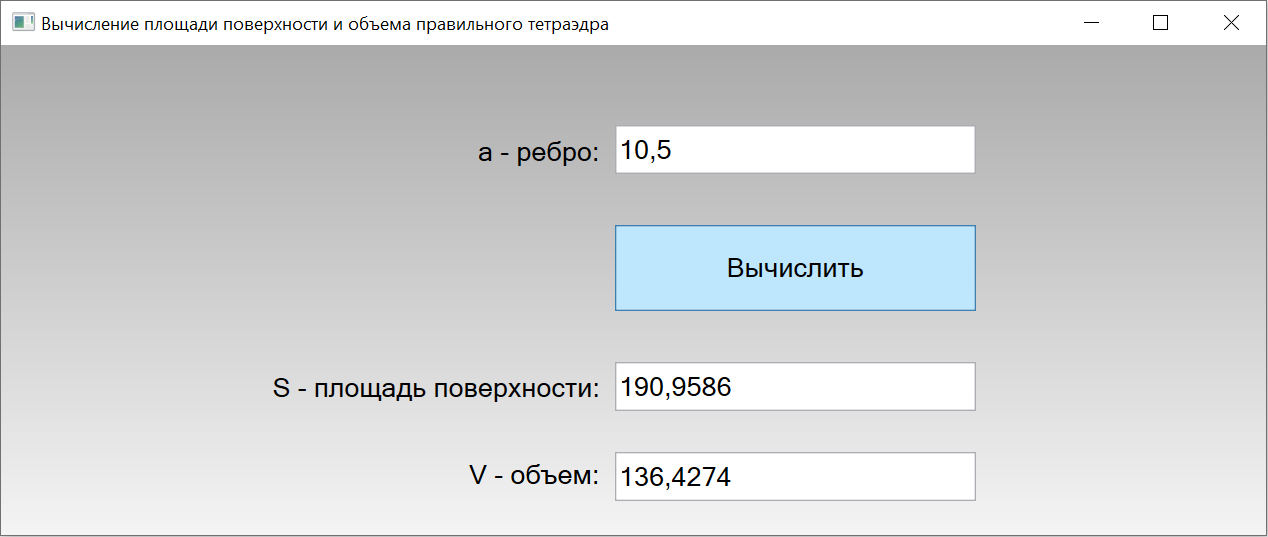
Метод button1\_Click(), содержащий код приложения:



Скрины отлаженной программы:



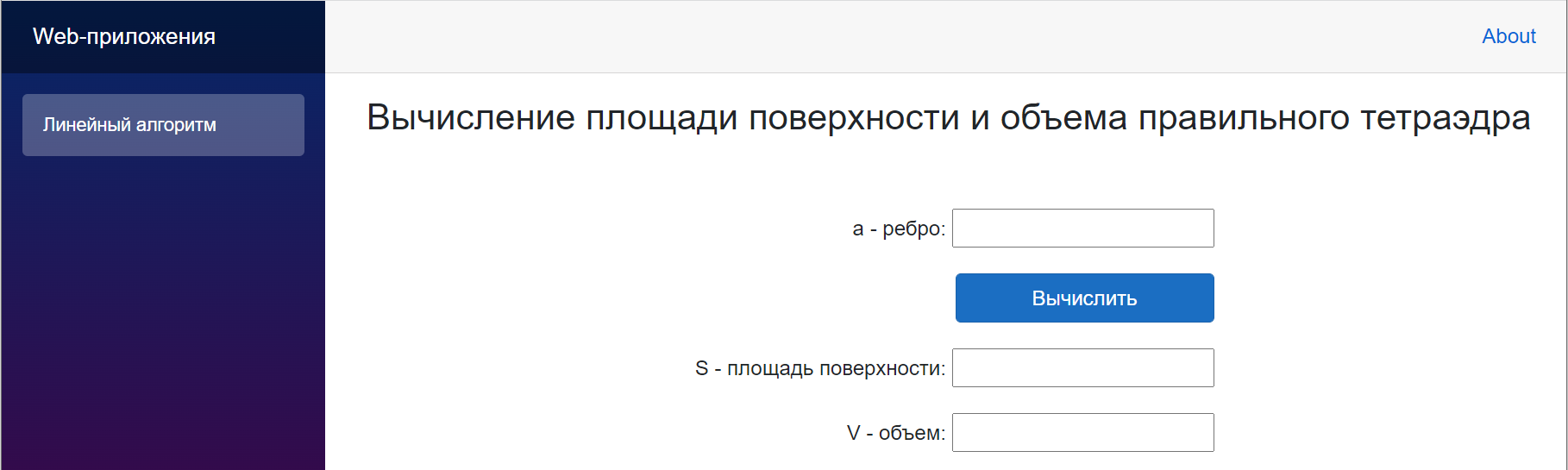




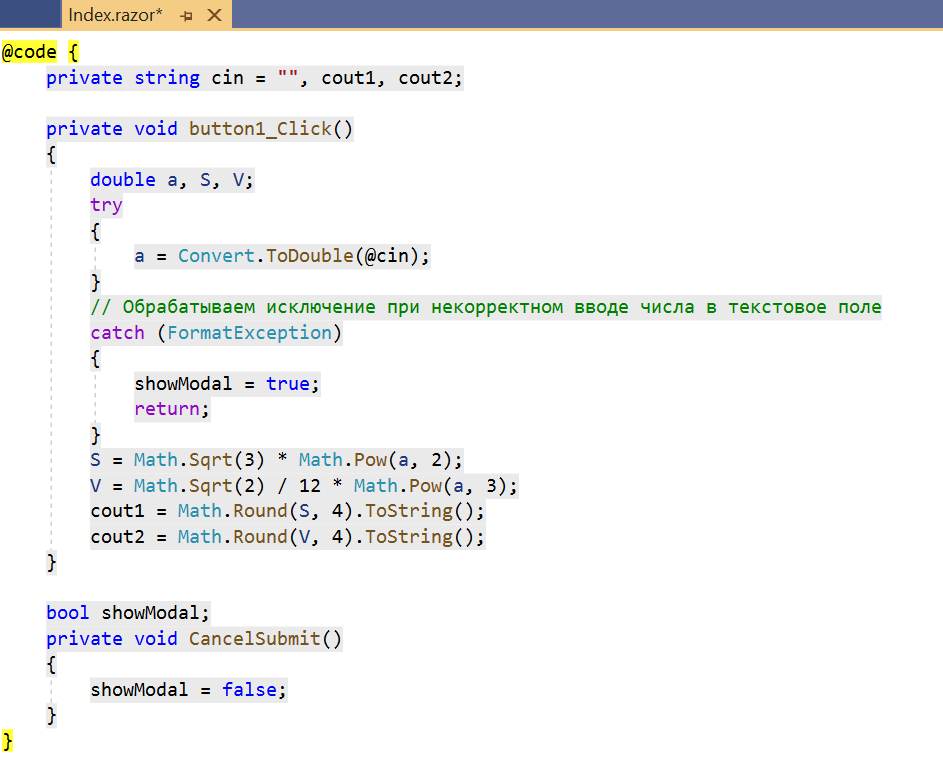
в) Интерфейс приложения WebAssembly Blazor:

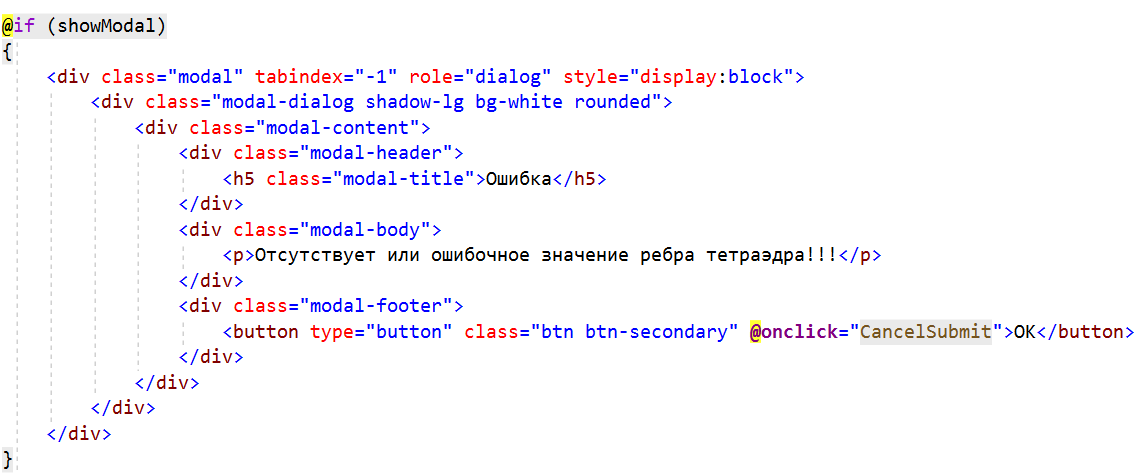




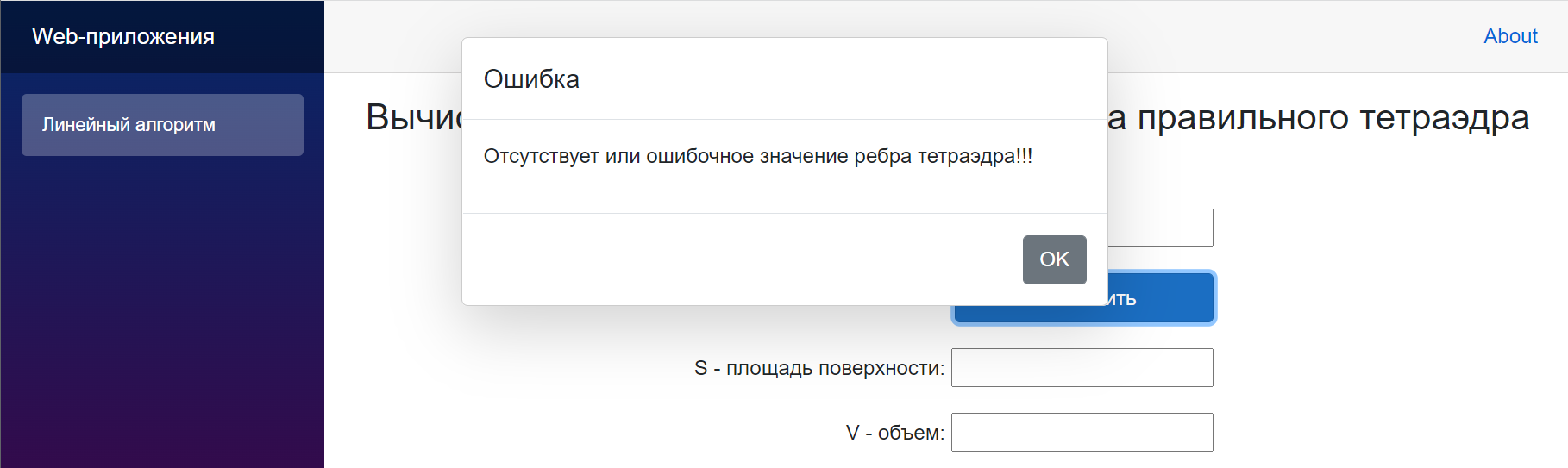


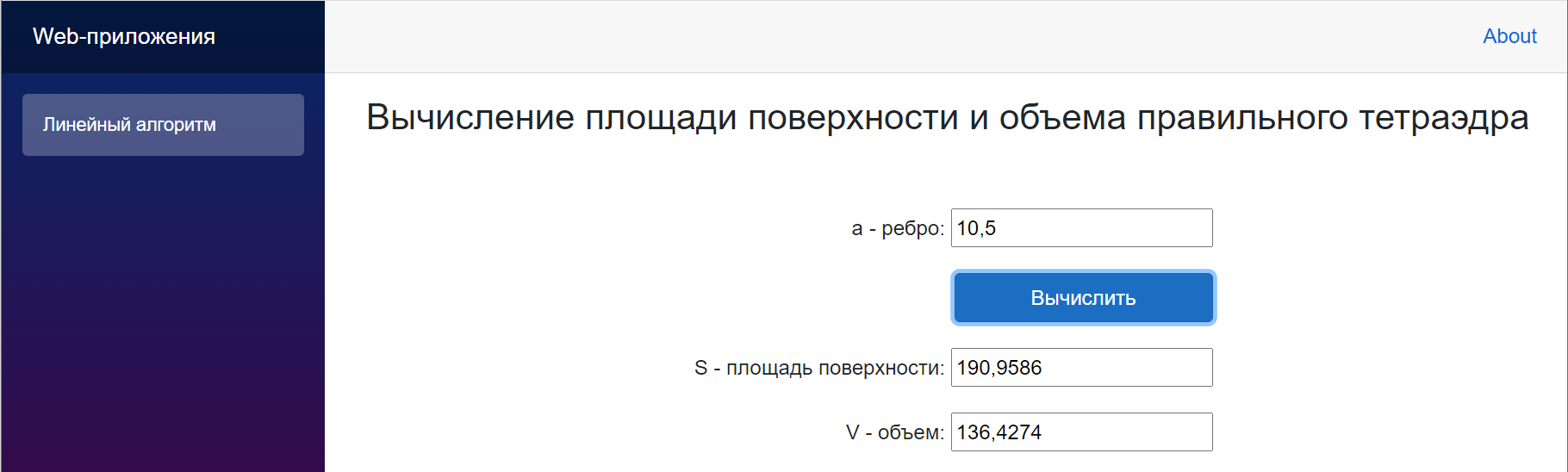
Коды приложения:





Скрины отлаженной программы:





**4 Порядок выполнения работы**

4.1 Изучить теоретические сведения и задание к работе.

4.2 В соответствии с вариантом задания создать форму приложения Windows Forms. Для этого в меню **Файл** приложения **MS Visual Studio** выберите команду **Создать** и щелкните **Проект**. В области шаблонов проектов **Visual C#** выберите группу **Windows**, щелкните элемент **Приложение Windows Forms**, введите имя проекта и нажмите кнопку **ОК**.

Откроется конструктор Windows Forms с формой Windows. Это пользовательский интерфейс для создаваемого приложения. ***Форма*** является ядром Windows-программ, написанных на С#. Форма инкапсулирует основные функции, необходимые для: создания окна, его отображения на экране и получения сообщений. Форма может представлять собой окно любого типа, включая основное окно приложения, дочернее или даже диалоговое окно.

Первоначально окно создается пустым. Затем в него добавляются ***меню*** и ***элементы управления***, например экранные кнопки, списки, переключатели и флажки. Таким образом, форму можно представить в виде контейнера для других Windows-объектов. Несмотря на различия между ними, способы их создания и обработки примерно одинаковы.

В меню **Вид** щелкните **Панель элементов**, чтобы открыть список элементов управления. Разверните список **Стандартные элементы управления** и перетащите на форму необходимые элементы управления: **Label**, **TextBox**, **Button**.

4.3 В соответствии с вариантом задания разработать и отладить код приложения Windows Forms, используя исключение, выбрасываемое, если форматы входных значений не соответствуют спецификациям параметров обрабатывающих их методов.

Чтобы открыть редактор кода дважды щелкните кнопку **Button**. Visual C# вставит метод с именем button1\_Click(), который будет выполняться при нажатии кнопки. Измените метод необходимым образом.

Элемент управления **TextBox** позволяет осуществлять ввод и вывод текстовой информации. Значения помещаются и извлекаются из textBox.Text. Для преобразования переменных в другой тип воспользуйтесь объектом Convert.

Для выдачи предупреждающих сообщений используйте метод MessageBox.Show("Сообщение").

Нажмите клавишу **F5** чтобы скомпилировать и запустить приложение.

**5 Содержание отчета**

5.1 Название работы

5.2 Цель работы

5.3 Программный код и скрины работы разработанного приложения Windows Forms в соответствии с вариантом задания.

**6 Варианты заданий**

6.1 Найти расстояние между двумя точками с заданными координатами *x*1 и *x*2 на числовой оси: |*x*2 − *x*1|.

6.2 Даны два неотрицательных числа *a* и *b*. Найти их среднее геометрическое, т. е. квадратный корень из их произведения: (a·*b*)1/2.

6.3 Даны координаты двух противоположных вершин прямоугольника: (*x1*, *y1*), (*x2*, *y2*). Стороны прямоугольника параллельны осям координат. Найти периметр и площадь данного прямоугольника.

6.4 Даны два ненулевых числа. Найти сумму, разность, произведение и частное их квадратов.

6.5 Даны два ненулевых числа. Найти сумму, разность, произведение и частное их модулей.

6.6 Даны два круга с общим центром и радиусами *R*1 и *R*2 (*R*1 > *R*2). Найти площади этих кругов *S*1 и *S*2, а также площадь *S*3 кольца, внешний радиус которого равен *R*1, а внутренний радиус равен *R*2:

*S*1 = *π*·(*R*1)2, *S*2 = *π*·(*R*2)2, *S*3 = *S*1 − *S*2.

6.7 Дана длина *L* окружности. Найти ее радиус *R* и площадь *S* круга, ограниченного этой окружностью, учитывая, что *L* = 2·π·*R*, *S* = π·*R*2.

6.8 Дана площадь *S* круга. Найти его диаметр *D* и длину *L* окружности, ограничивающей этот круг, учитывая, что *L* = π·*D*, *S* = π·*D*2/4.

6.9 Решить линейное уравнение *A*·*x* + *B* = 0, заданное своими коэффициентами *A* и *B* (коэффициент *A* не равен 0).

6.10 Дано значение угла α в радианах (0 ≤ α < 2·π). Определить значение этого же угла в градусах, учитывая, что 180° = π радианов.

6.11 Даны три точки *A*, *B*, *C* на числовой оси. Точка *C* расположена между точками *A* и *B*. Найти произведение длин отрезков *AC* и *BC*.

6.12 Найти расстояние между двумя точками с заданными координатами (*x*1, *y*1) и (*x*2, *y*2) на плоскости. Расстояние вычисляется по формуле: .

6.13 Даны катеты прямоугольного треугольника *a* и *b*. Найти его гипотенузу *c* и периметр *P*: *c* = (*a*2 + *b*2)1/2, *P* = *a* + *b* + *c*.

6.14 Дана длина ребра куба *a*. Найти объем куба *V* = *a*3 и площадь его поверхности *S* = 6·*a*2.

6.15 Даны длины ребер *a*, *b*, *c* прямоугольного параллелепипеда. Найти его объем *V* = *a*·*b*·*c* и площадь поверхности *S* = 2·(*a*·*b* + *b*·*c* + *a*·*c*).

6.16 Найти длину окружности *L* = 2·π·*R* и площадь круга *S* = π·*R*2 заданного радиуса *R*.

6.17 Даны два числа *a* и *b*. Найти их среднее арифметическое: (*a* + *b*)/2.

**7 Используемая литература**

7.1 Г. Гагарина, Е. В. Кокорева, Б. Д. Виснадул Технология разработки программного обеспечения. Форум, Инфра-М, 2009

7.2 Эндрю Троелсен Язык программирования С# 2010 и платформа .NET 4.0, М., ВИЛЬЯМС, 2011

7.3 А. Марченко C#. Введение в программирование, М, Вильямс, 2009

7.4 http://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx.