**Практическое занятие № 10-11. Методы. Объявление метода. Вызов метода. Параметры**

**1 Цель занятия**

Получить практические навыки разработки и использования методов в модулях.

**2 Перечень оборудования и программного обеспечения**

Персональный компьютер

Microsoft Office (Word)

Microsoft Visual Studio

**3 Краткие теоретические сведения**

**3.1 Объявление методов**

В С# определение метода состоит из любых модификаторов (таких как спецификация доступности), типа возвращаемого значения, за которым следует имя метода, затем списка аргументов в круглых скобках и далее — тела метода в фигурных скобках:

[<модификаторы>] <тип возврата> <имя метода>([<параметры>])

{

// Тело метода

}

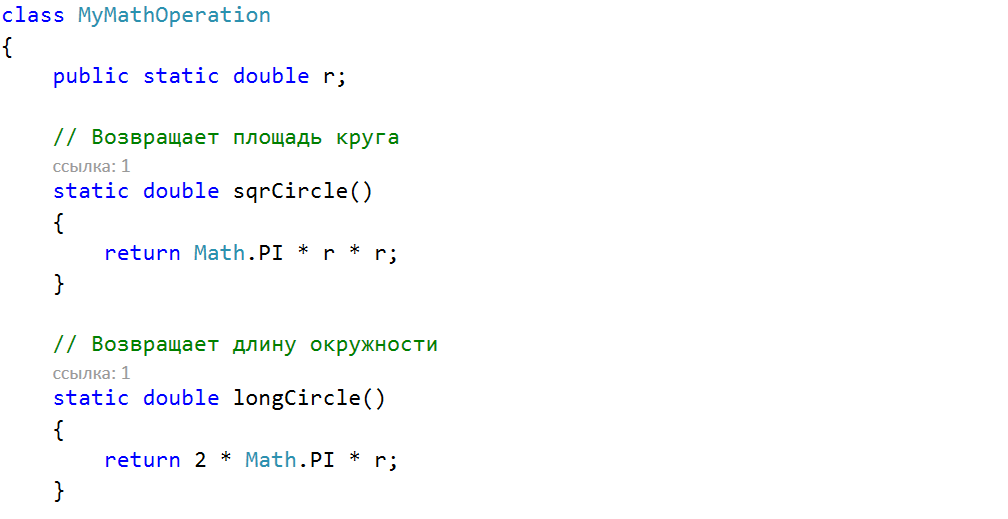
Каждый параметр состоит из имени типа параметра и имени, по которому к нему можно обратиться в теле метода. Вдобавок, если метод возвращает значение, то для указания точки выхода должен использоваться оператор возврата вместе с возвращаемым значением.

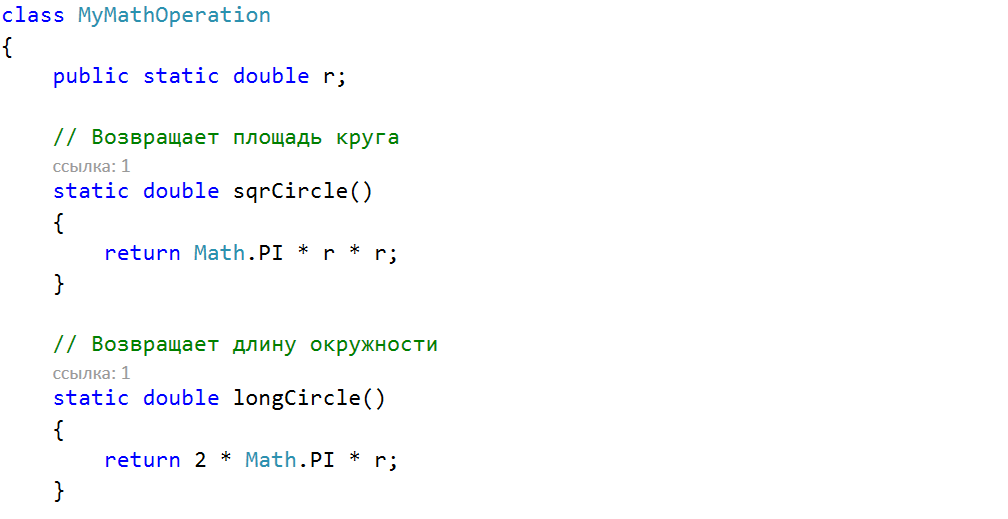
Если метод не возвращает ничего, то в качестве типа возврата указывается void, поскольку вообще опустить тип возврата невозможно. Если же он не принимает аргументов, то все равно после имени метода должны присутствовать пустые круглые скобки. При этом включать в тело метода оператор возврата не обязательно — метод возвращает управление автоматически по достижении закрывающей фигурной скобки.

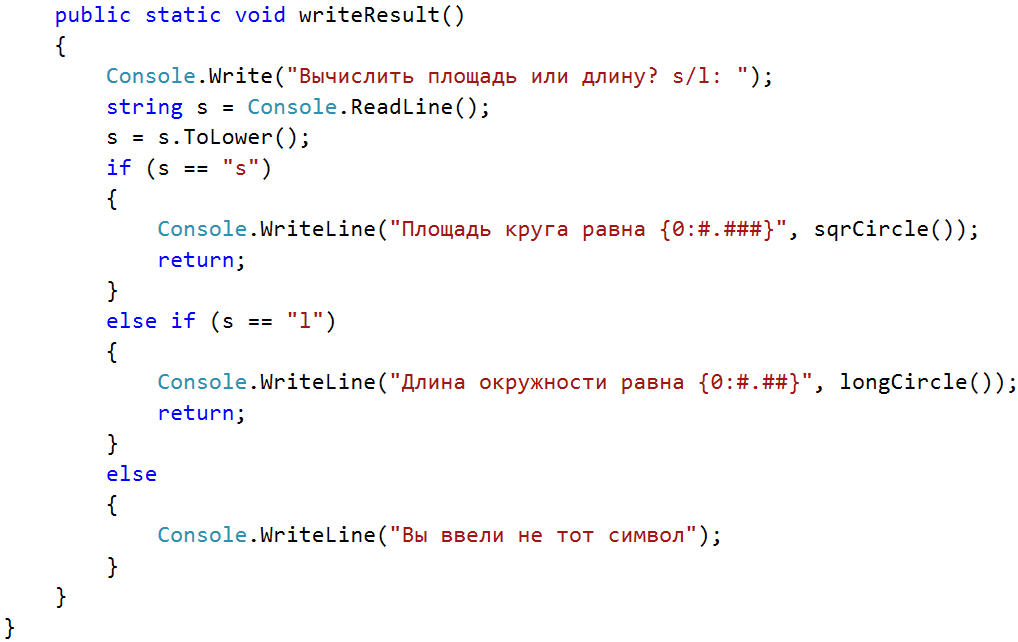
**3.2 Возврат из метода и возврат значения**

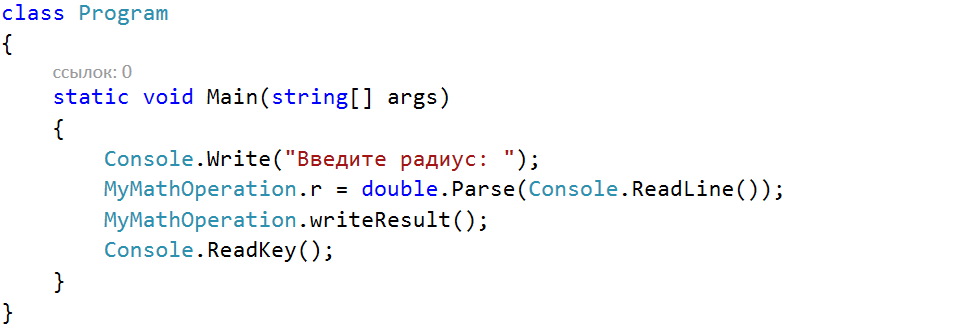
В целом, возврат из метода может произойти при двух условиях. Во-первых, когда встречается фигурная скобка, закрывающая тело метода. И во-вторых, когда выполняется оператор return. Имеются две формы оператора return: одна — для методов типа void (возврат из метода), т.е. тех методов, которые не возвращают значения, а другая — для методов, возвращающих конкретные значения (возврат значения).

**Пример 1:**









**3.3 Использование параметров**

При вызове метода ему можно передать одно или несколько значений. Значение, передаваемое методу, называется аргументом. А переменная, получающая аргумент, называется формальным параметром, или просто параметром. Параметры объявляются в скобках после имени метода. Синтаксис объявления параметров такой же, как и у переменных. А областью действия параметров является тело метода. За исключением особых случаев передачи аргументов методу, параметры действуют так же, как и любые другие переменные.

В общем случае параметры могут передаваться методу либо по значению, либо по ссылке.

*При передаче по значению* вызываемый метод получает копию этой переменной, а это значит, что все изменения в ней по завершении метода будут утеряны.

*При передаче по ссылке* (*по адресу*) вызываемый метод получает саму переменную, поэтому любые изменения, которым она подвергнется внутри метода, останутся в силе после его завершения. Параметр ссылки можно объявить с помощью ключевого слова ref перед именем типа. Передача по ссылке более эффективна для сложных типов данных из-за большого объема информации, которую приходится копировать при передаче их по значению.

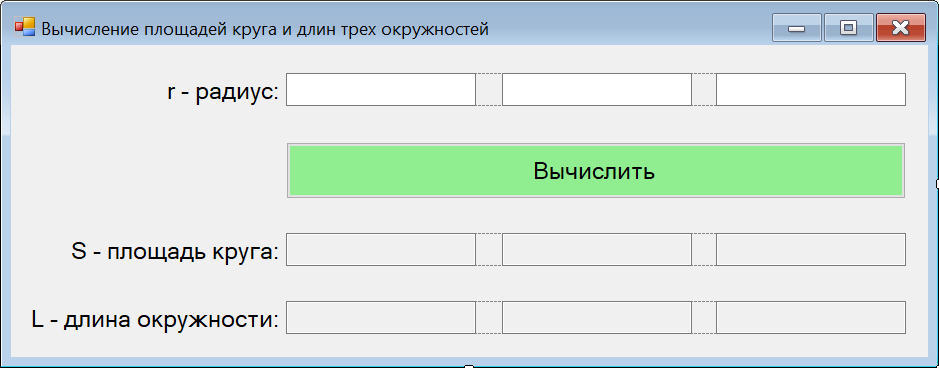
*Выходные параметры* передают данные из метода, а не в него. Для выходных параметров используется ключевое слово out в объявлении метода и вызове. Выходному параметру должно быть присвоено значение внутри метода.

Как и параметр ссылки, выходной параметр является ссылкой на место хранения данных от вызывающего объекта. Однако переменной с параметром out не нужно присваивать значение перед вызовом. Выходные параметры полезны, когда вы хотите иметь возможность возвращать значения из метода посредством использования параметра без присвоения начального значение параметра.

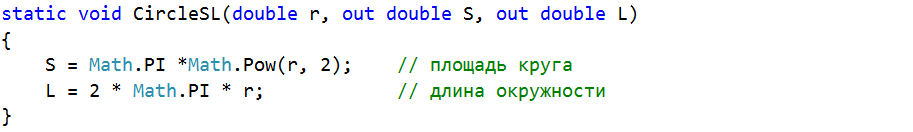
**Задание 1.** Разработать метод CircleSL(*r*, *S*, *L*), вычисляющий по радиусу окружности *r* ее площадь круга *S* = π·*r*2 и длину *L* = = 2·π·*r* (*r* — входной, *S* и *L* — выходные параметры; все параметры являются вещественными). С помощью этого метода найти площади кругов и длины трех окружностей с данными радиусами.

**Решение.**

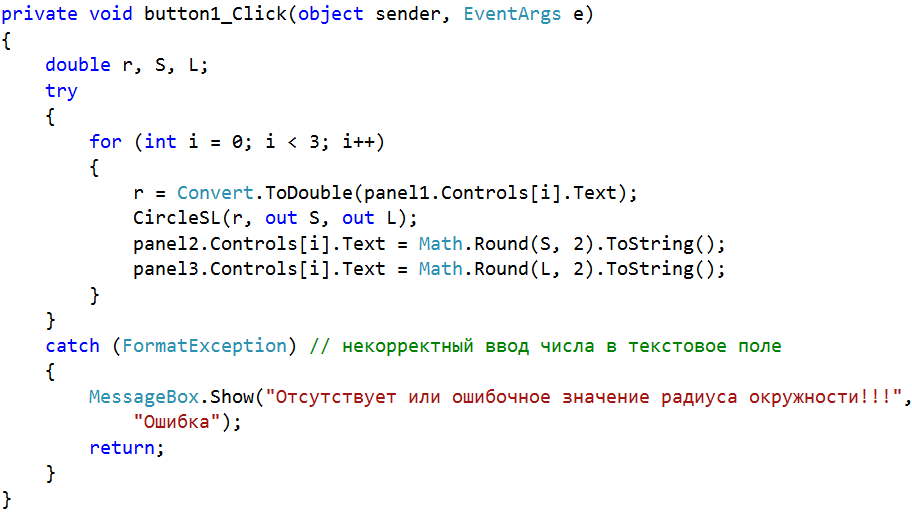
а) Интерфейс приложения Windows Forms:



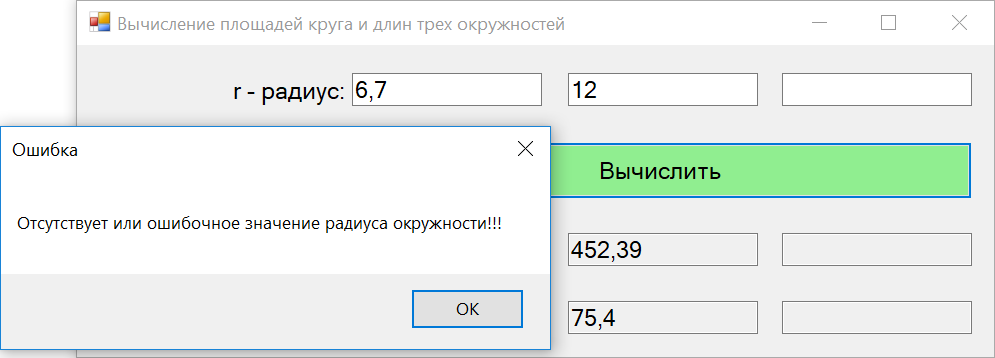
Код метода CircleSL(*r*, *S*, *L*):

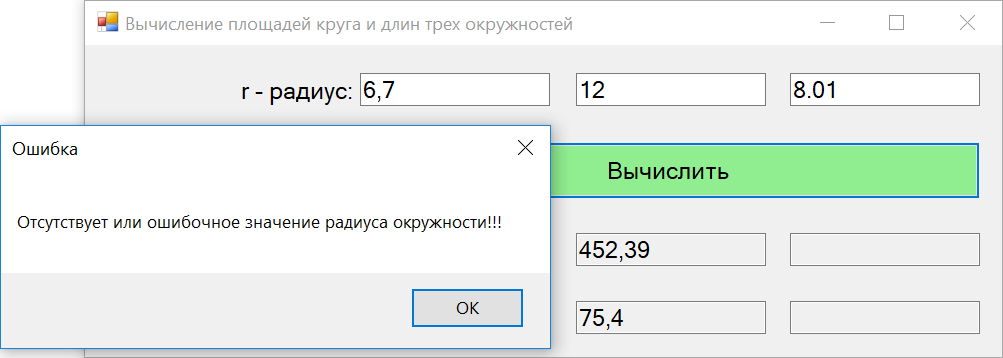


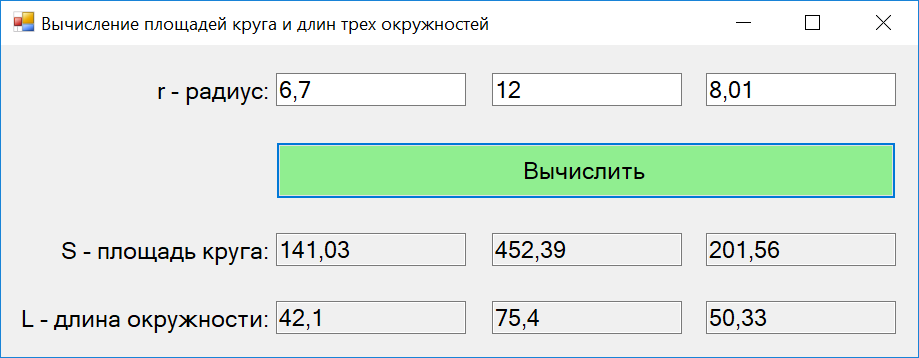
Код, выполняемый при щелчке на кнопку «Вычислить»:



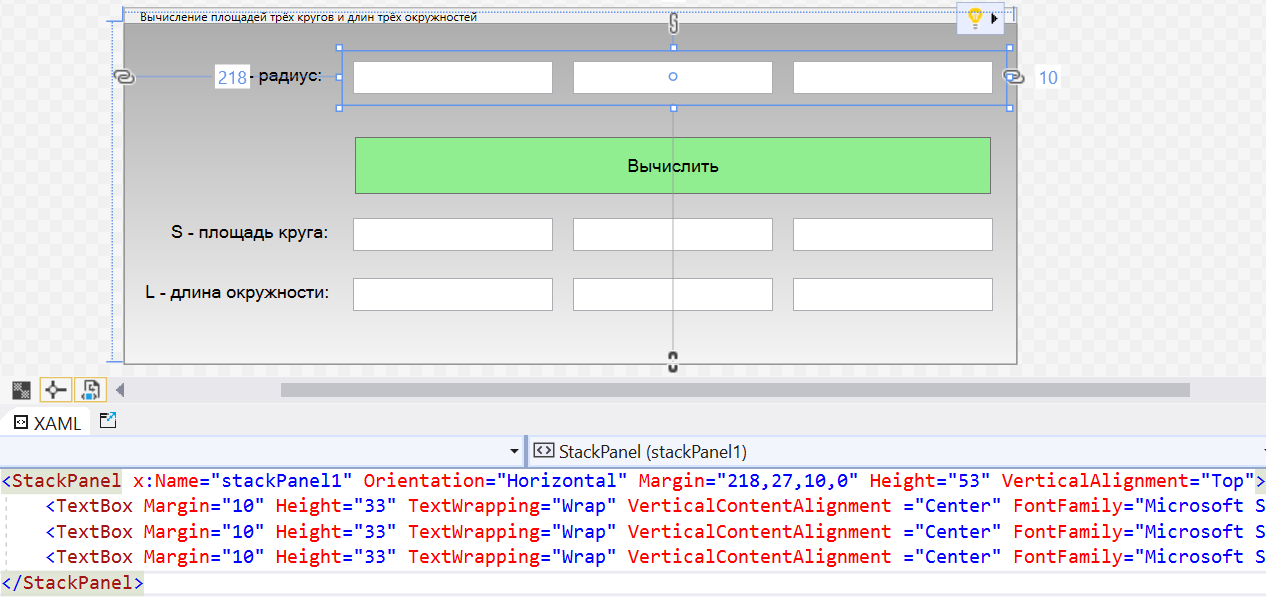
Скрины отлаженной программы:

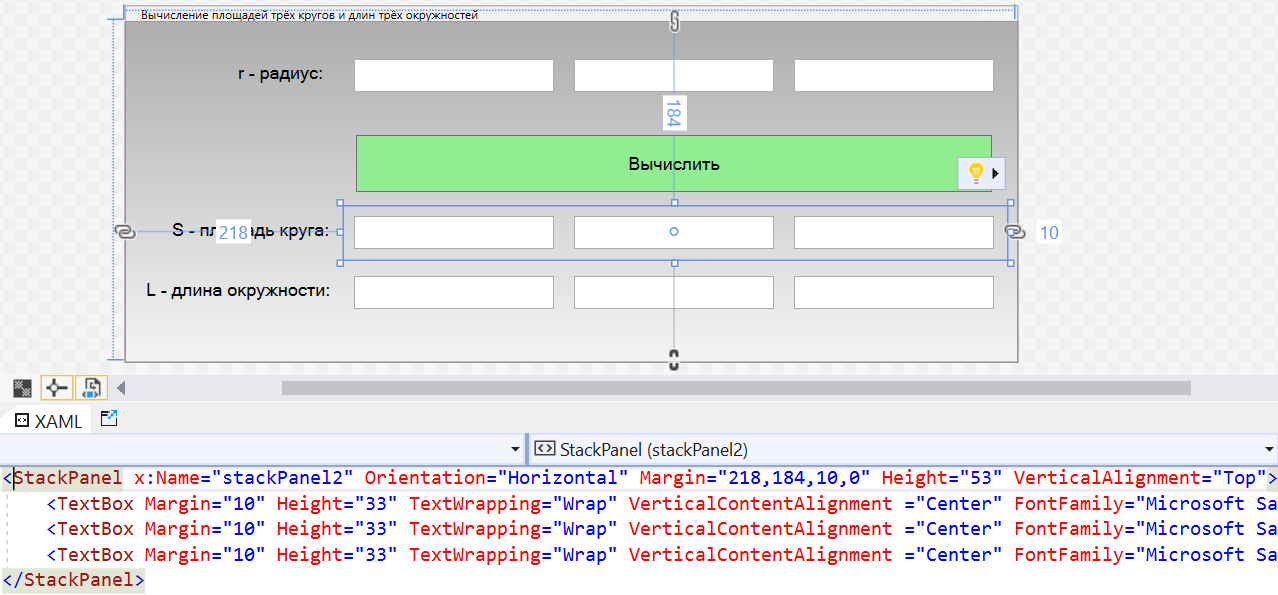


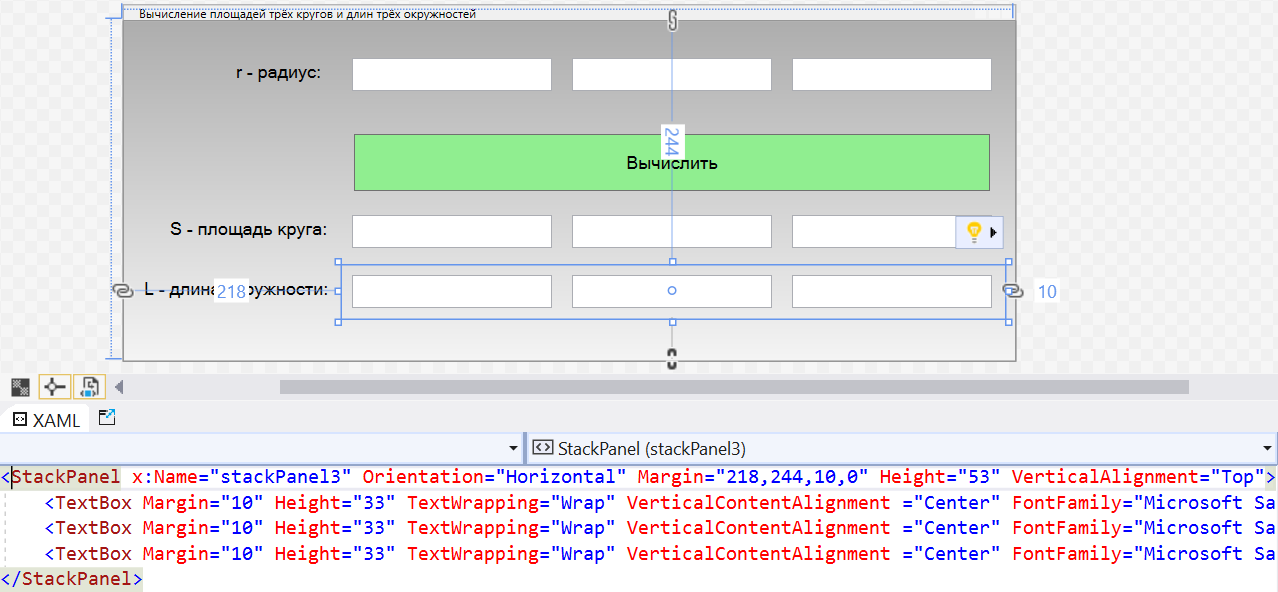




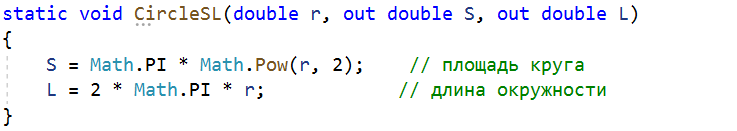
б) Интерфейс приложения WPF:



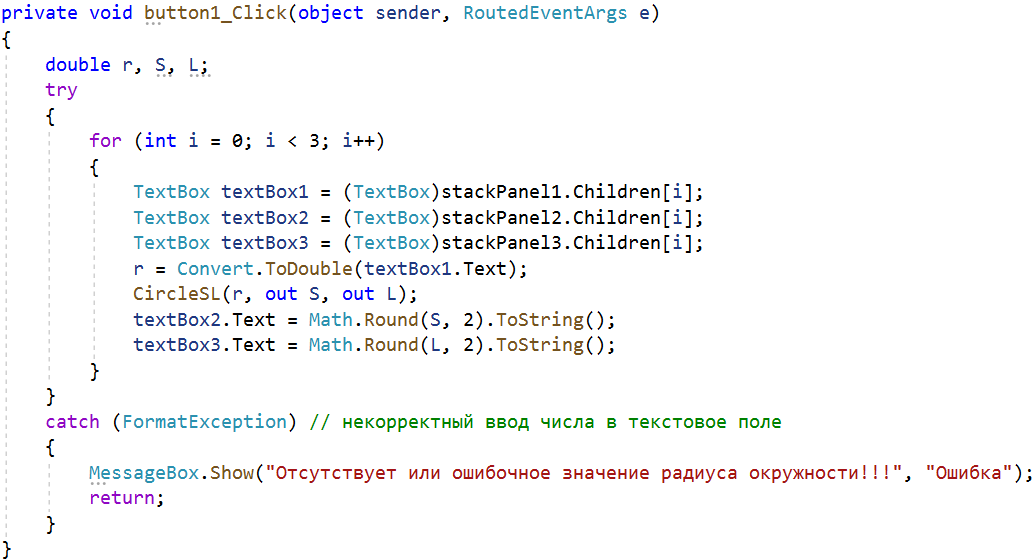




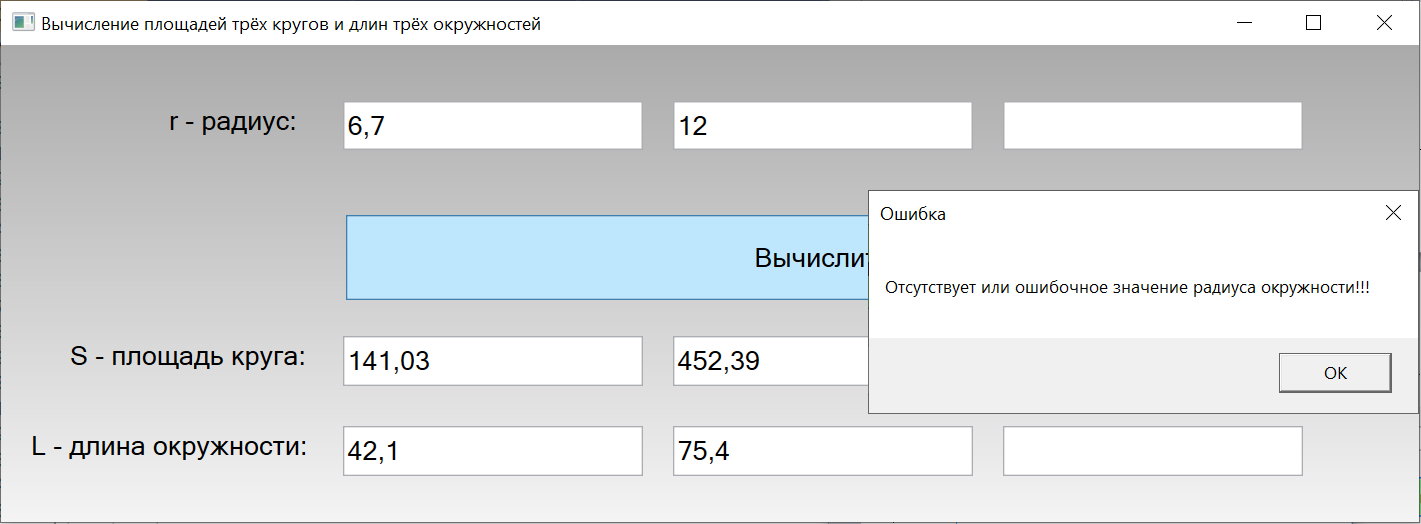
Код метода CircleSL(*r*, *S*, *L*):

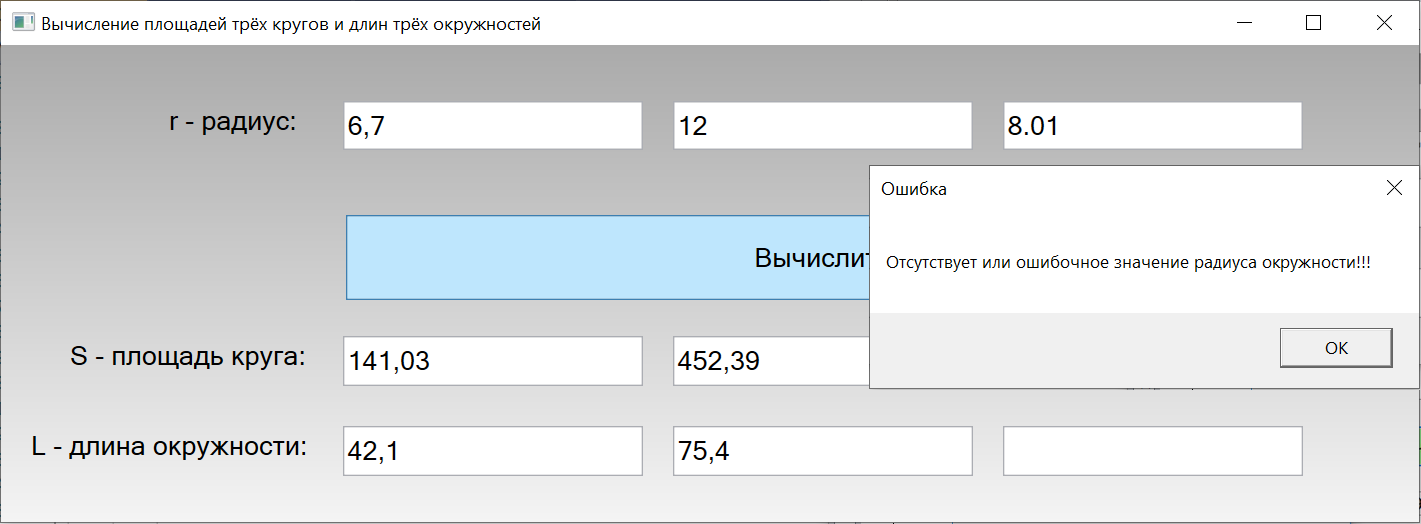


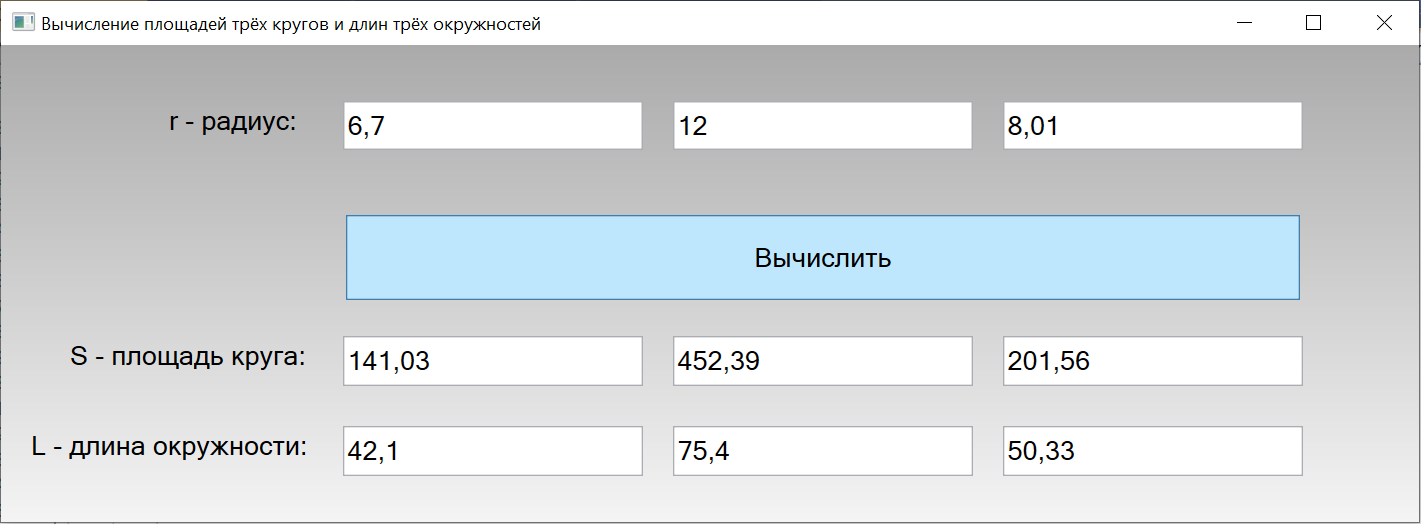
Код, выполняемый при щелчке на кнопку «Вычислить»:



Скрины отлаженной программы:

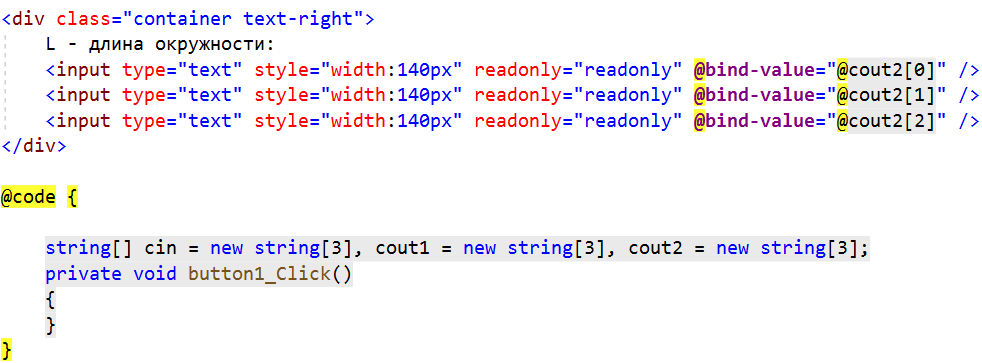


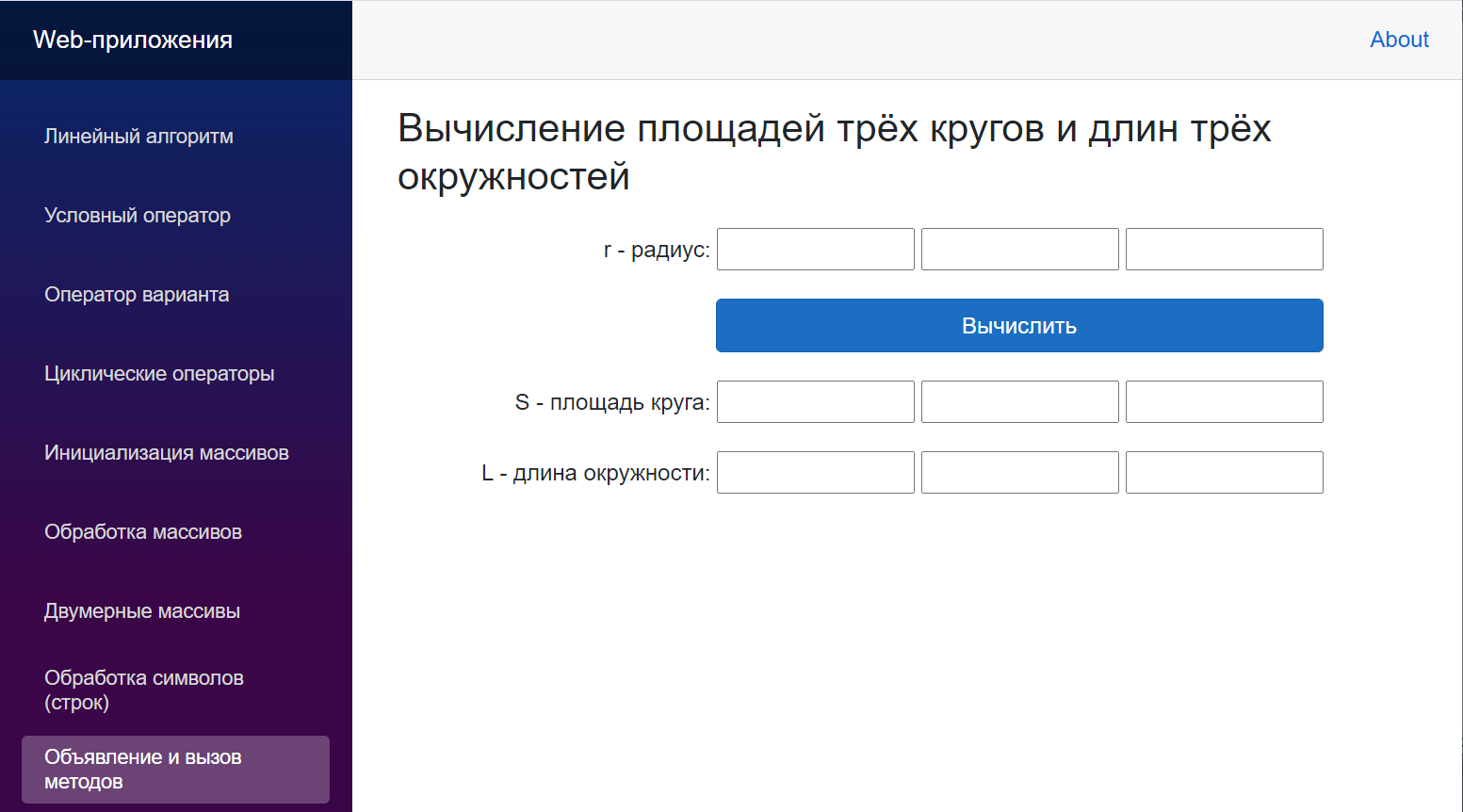




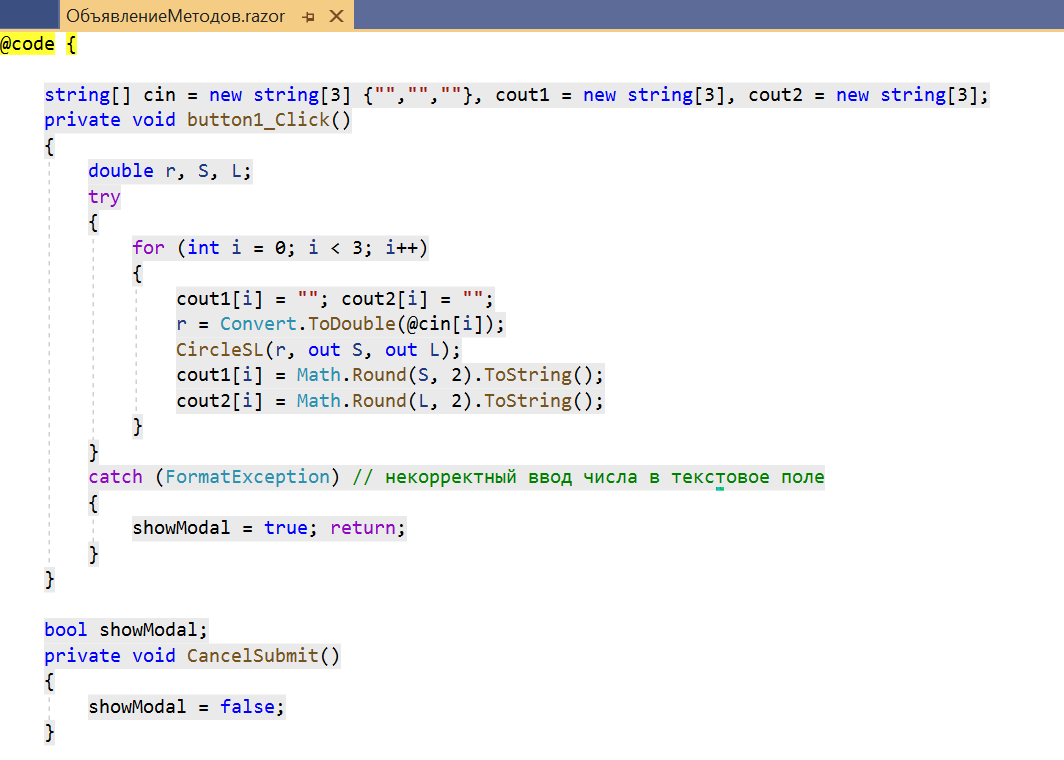
в) Интерфейс приложения WebAssembly Blazor:

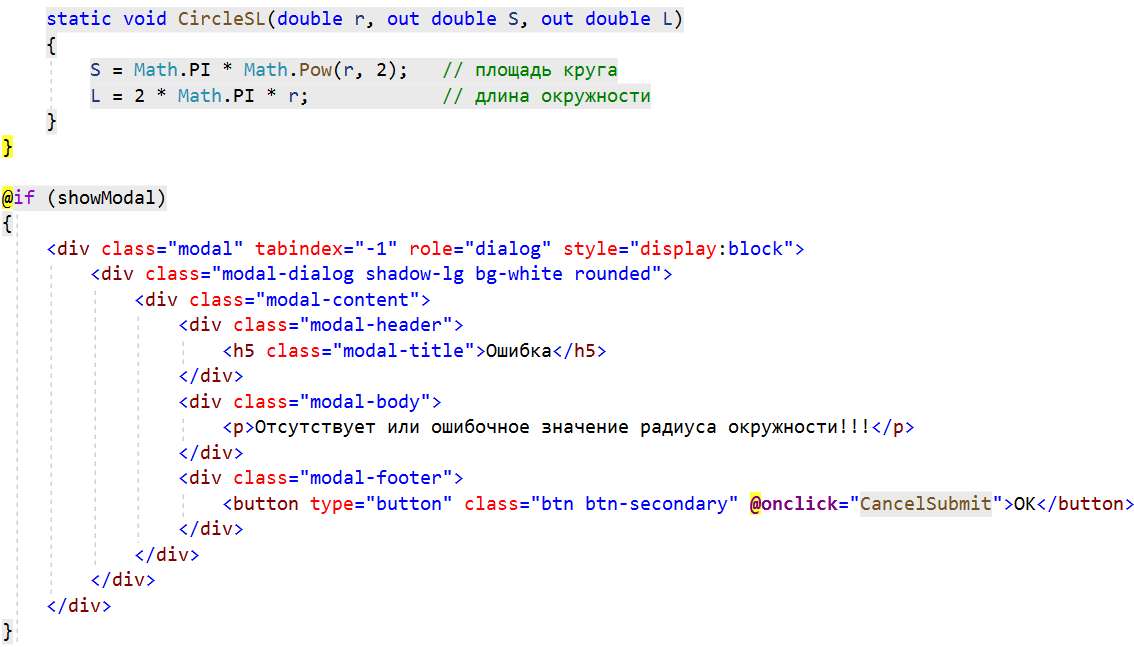




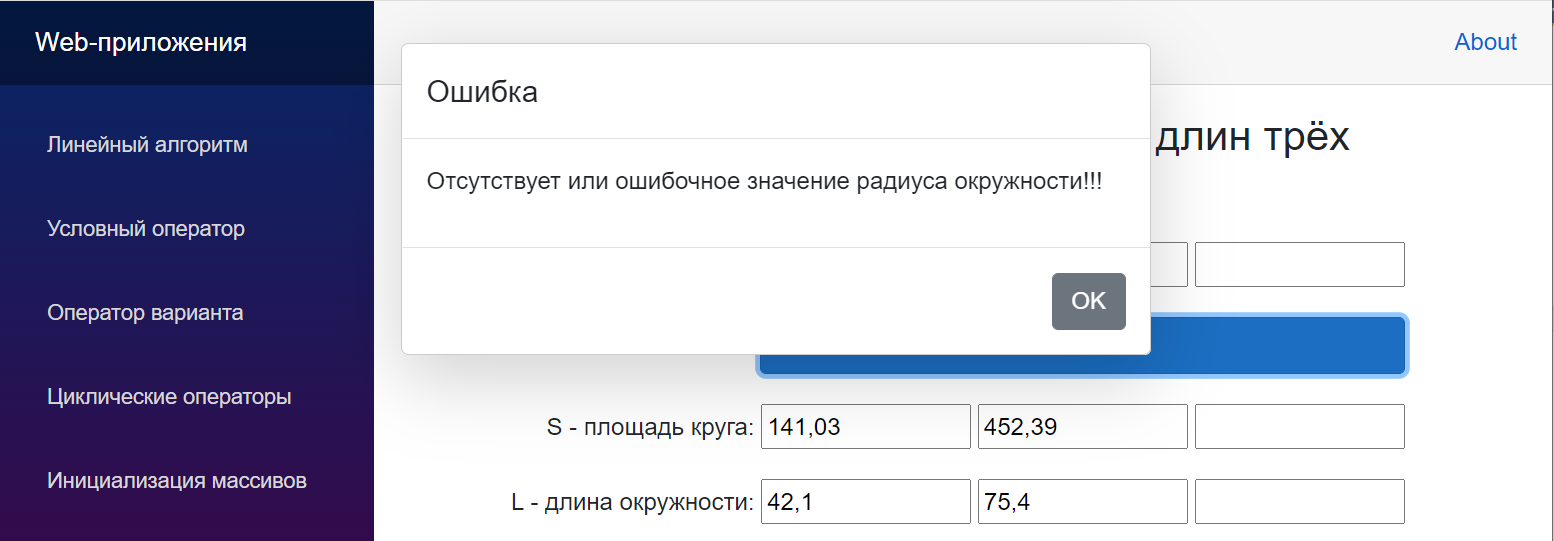


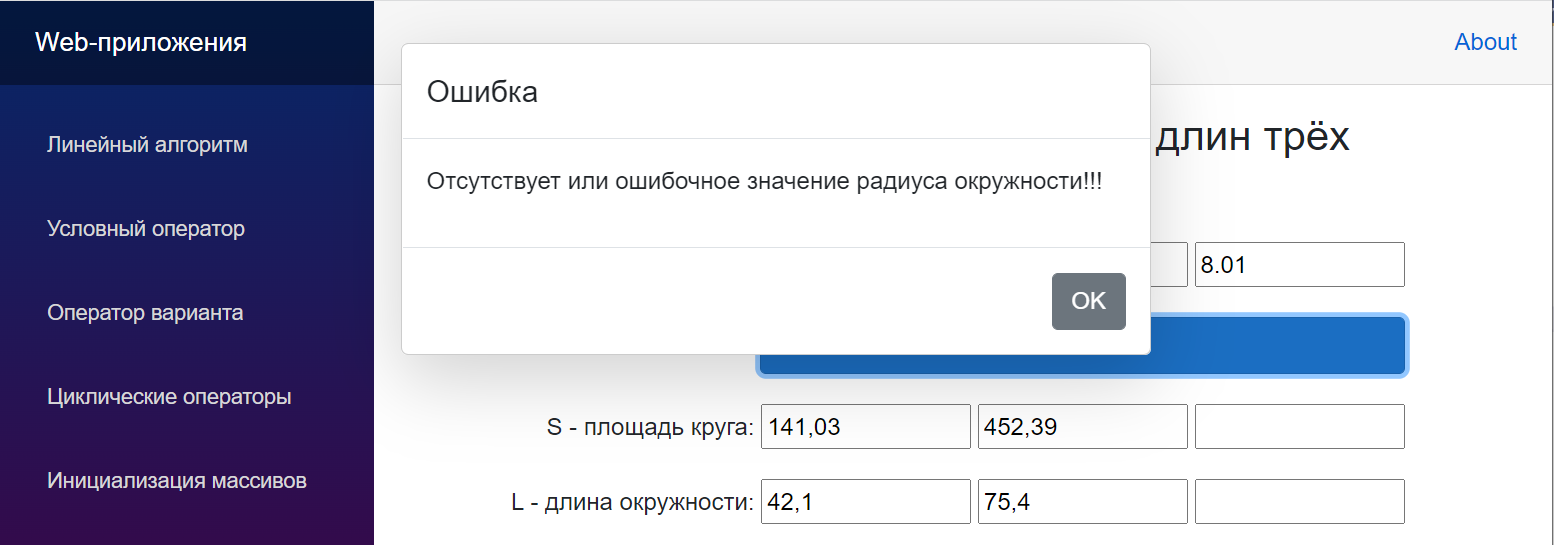
Коды приложения:





Скрины отлаженной программы:



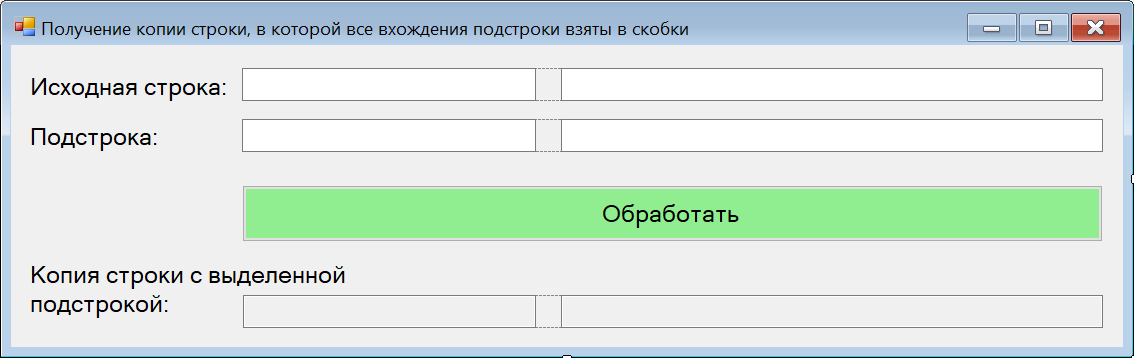




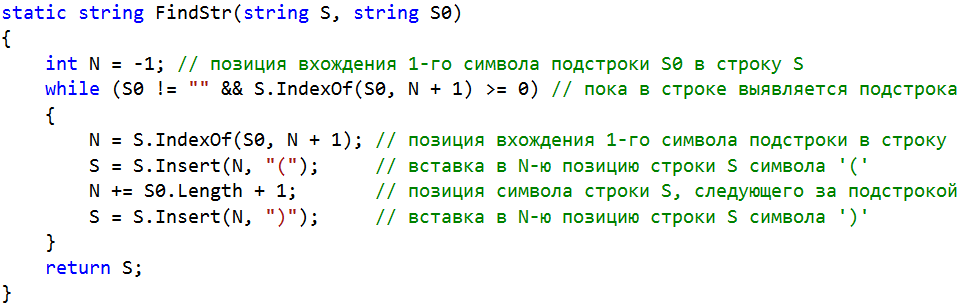
**Задание 2.** Описать функцию FindStr(*S*, *S*0) строкового типа, возвращающую копию строки *S*, в которой каждая подстрока, совпадающая со строкой *S*0, заключается в круглые скобки. Например, для строк *S* = «ertabcsdftyuabczevbh» и *S*0 = «abc» функция вернет строку «ert(abc)sdftyu(abc)zevbh». Вывести значения этой функции для двух данных пар строк *S* и *S*0.

**Решение.**

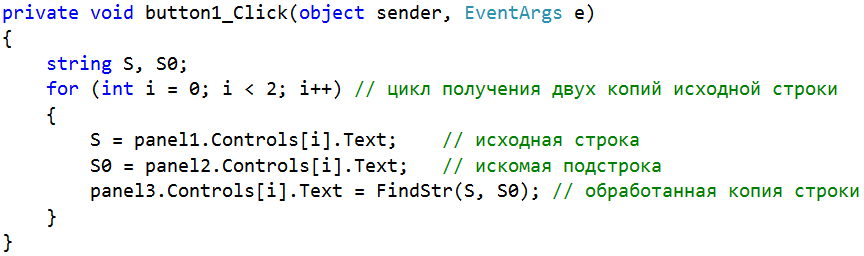
а) Интерфейс приложения Windows Forms:

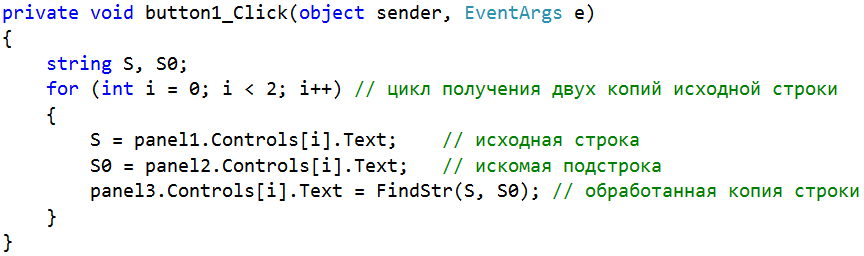


Код метода FindStr(*S*, *S*0):

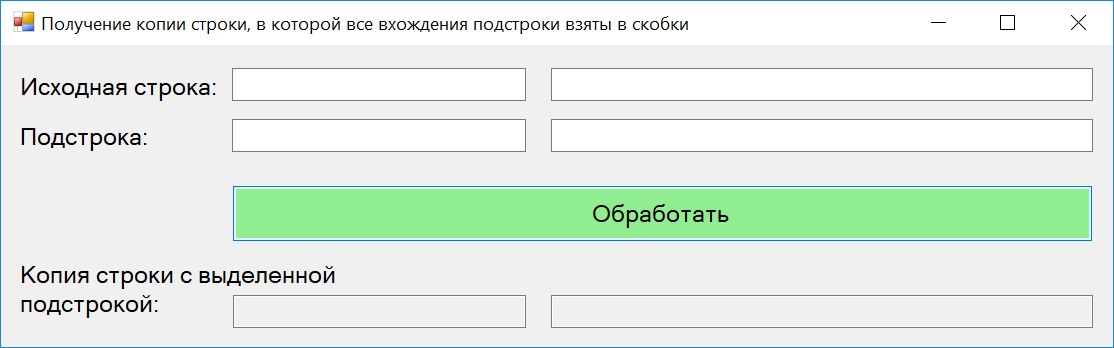


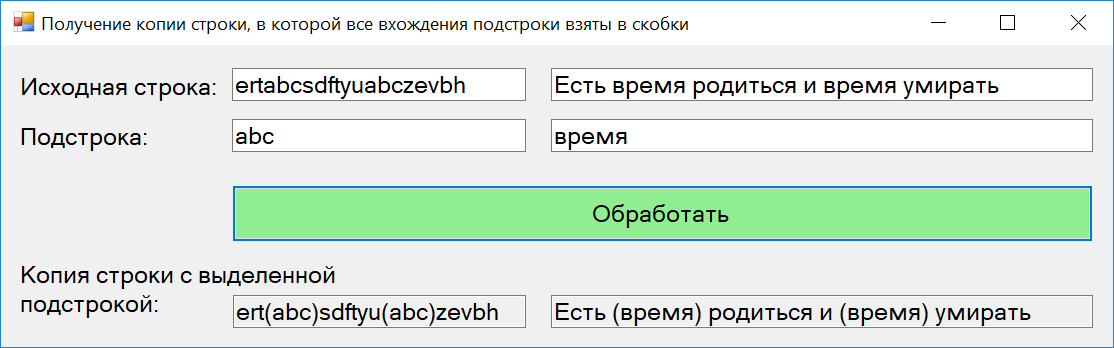
Код, выполняемый при щелчке на кнопку «Обработать»:

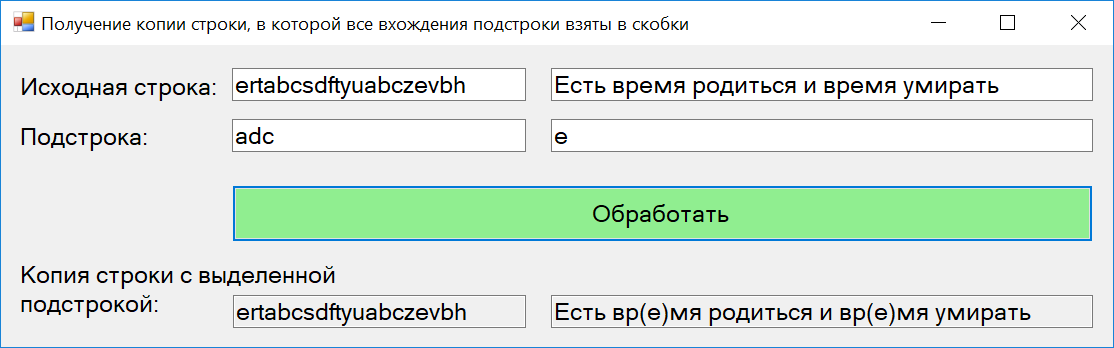




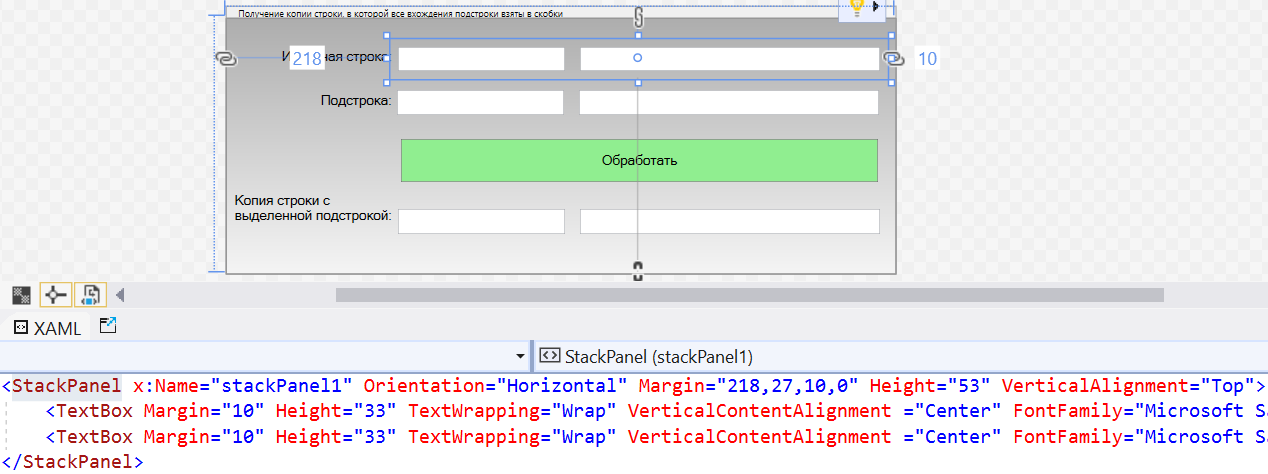
Скрины отлаженной программы:

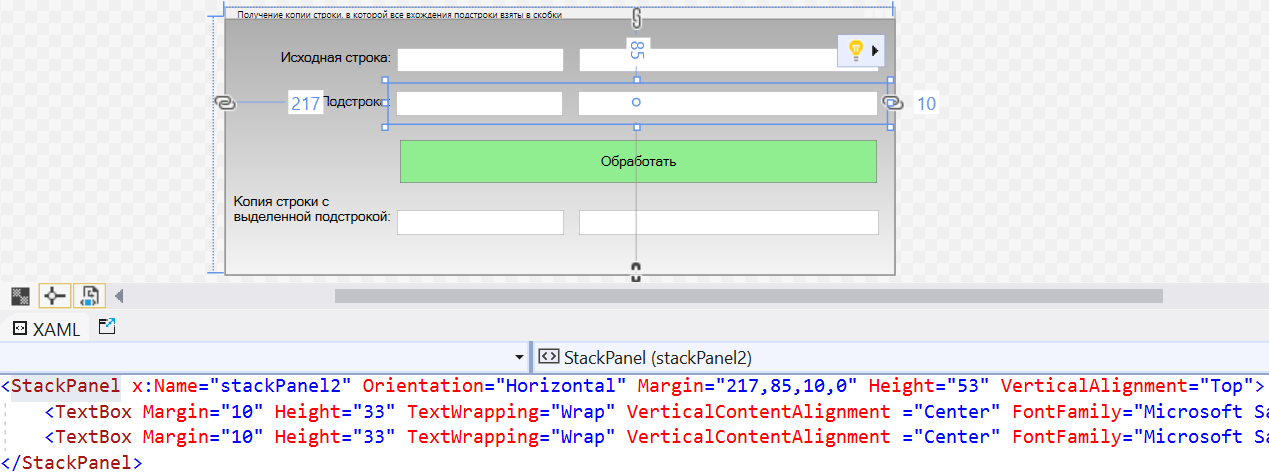


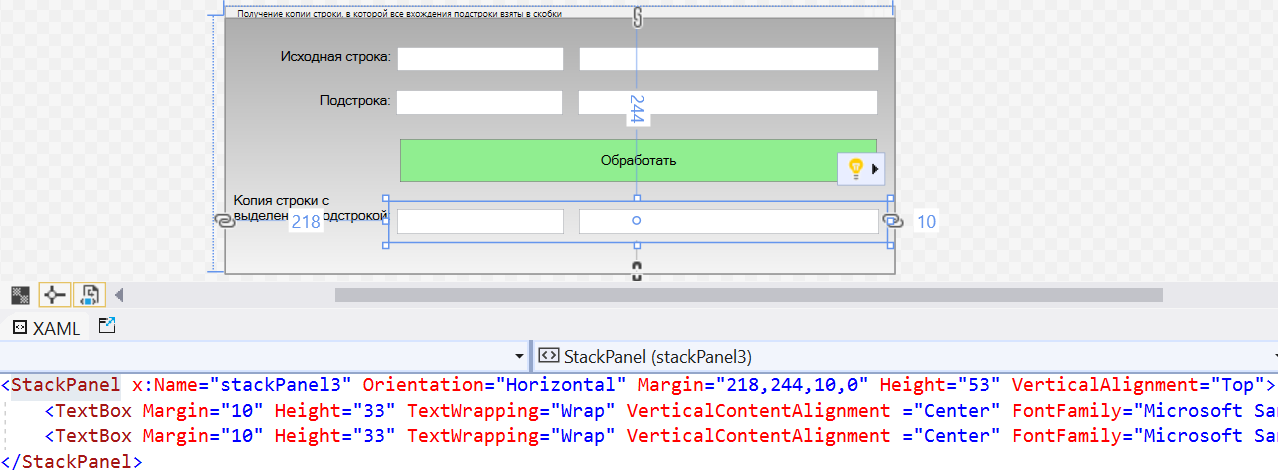




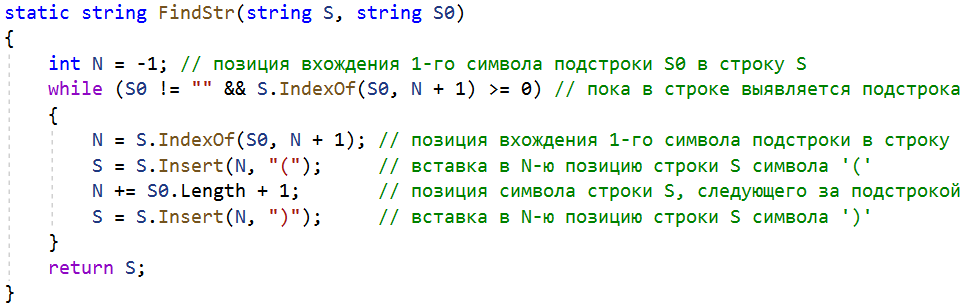
б) Интерфейс приложения WPF:



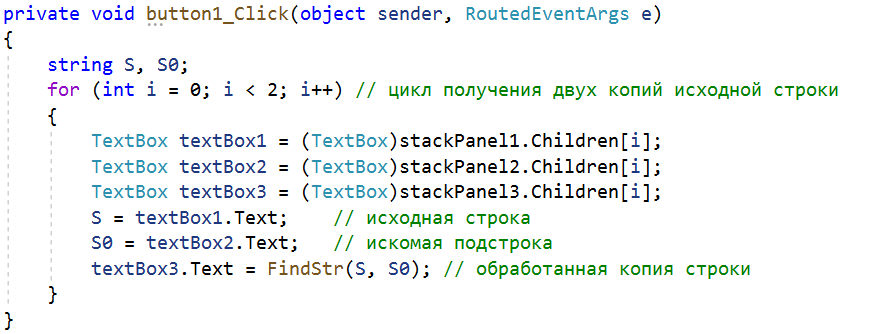




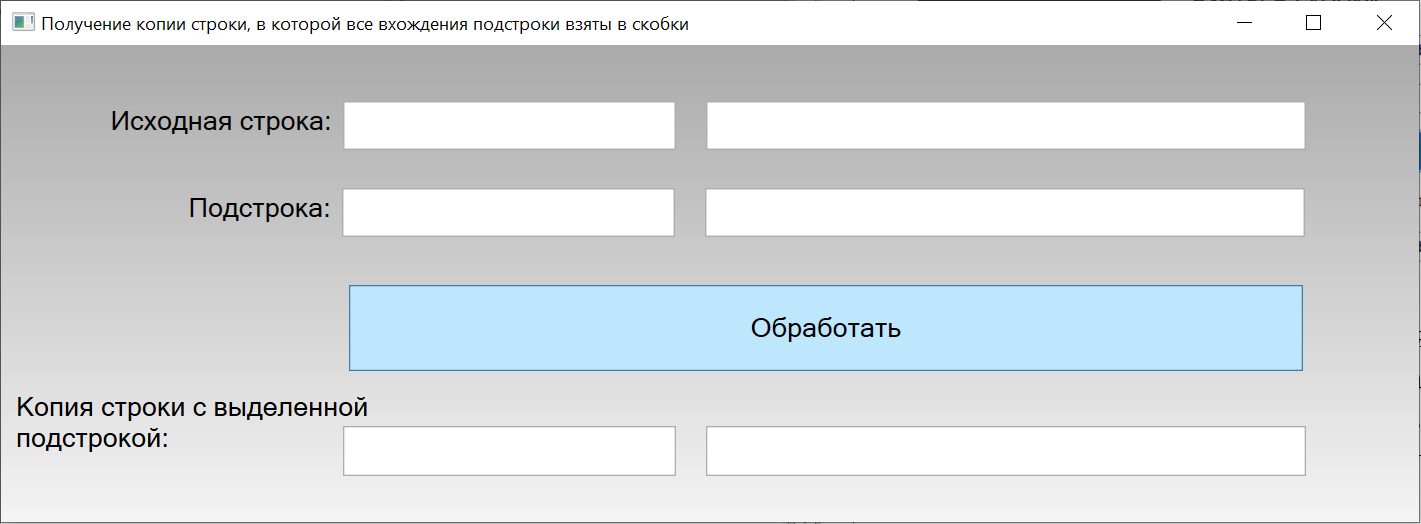
Код метода FindStr(*S*, *S*0):

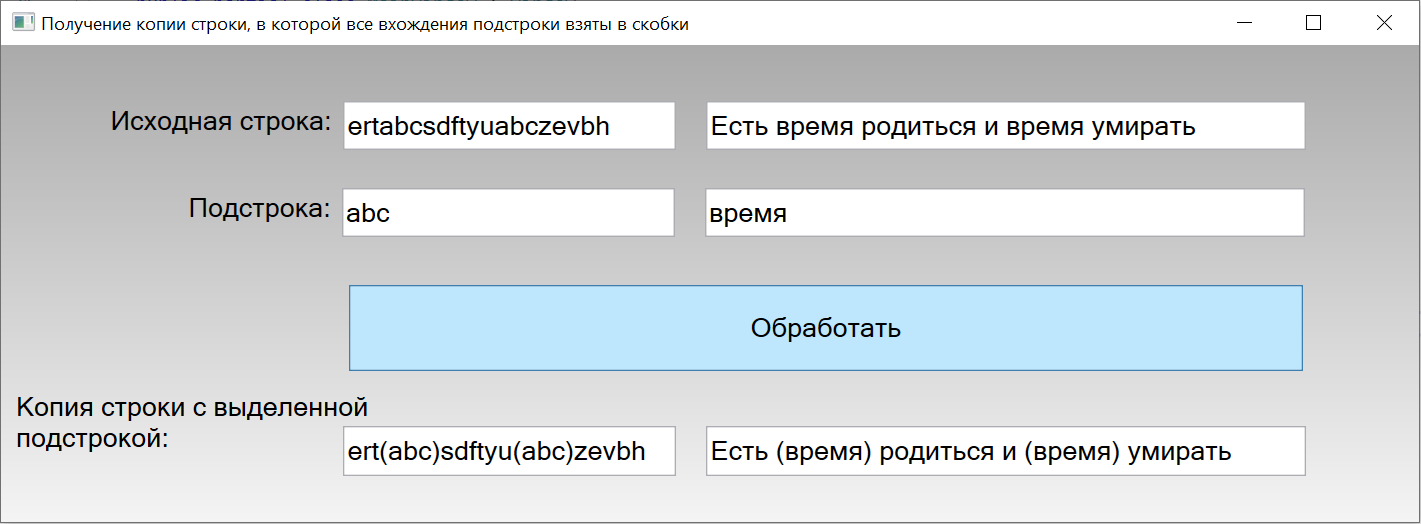


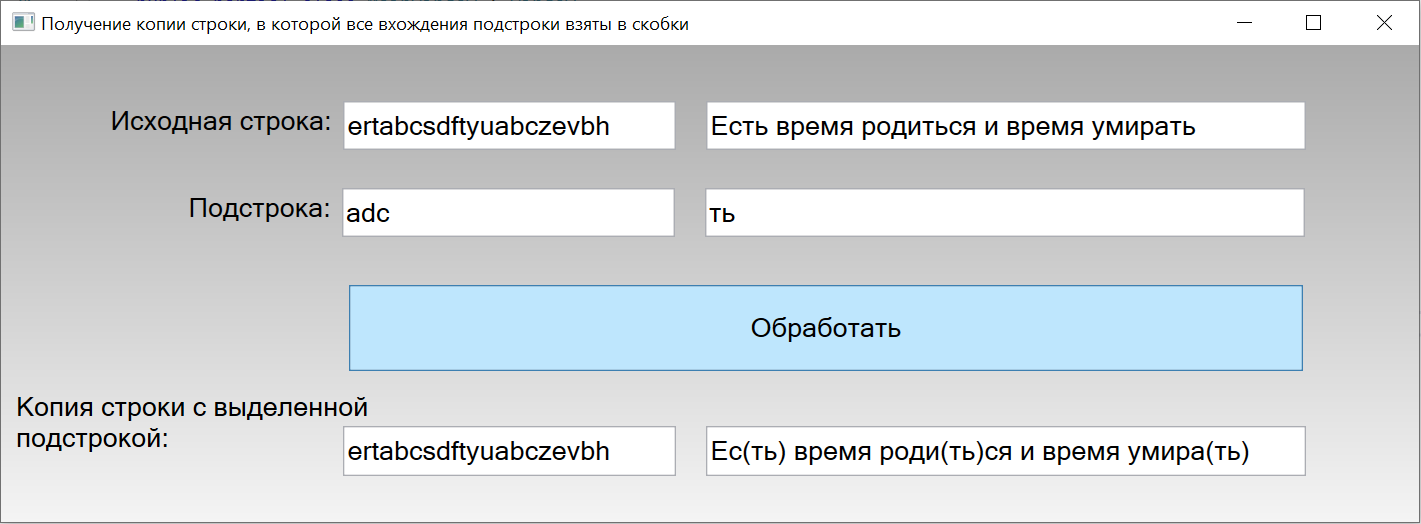
Код, выполняемый при щелчке на кнопку «Обработать»:



Скрины отлаженной программы:

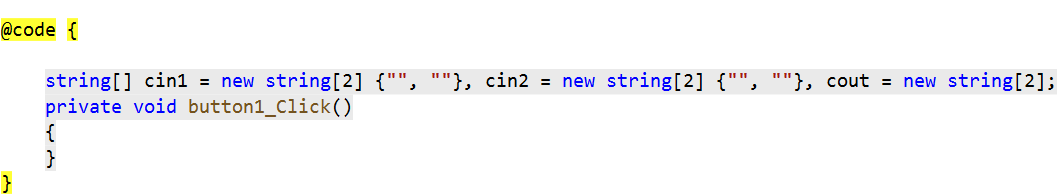


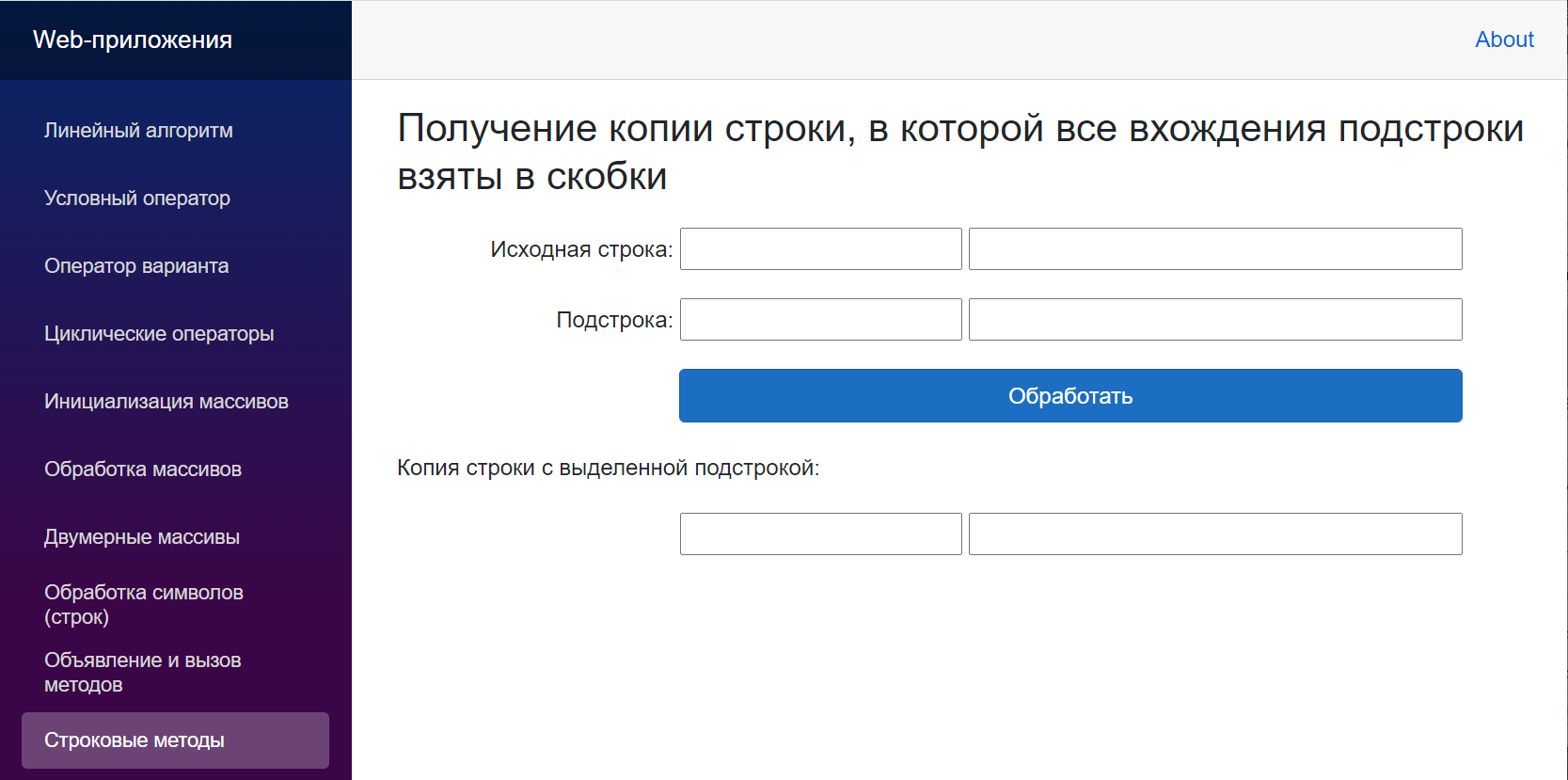




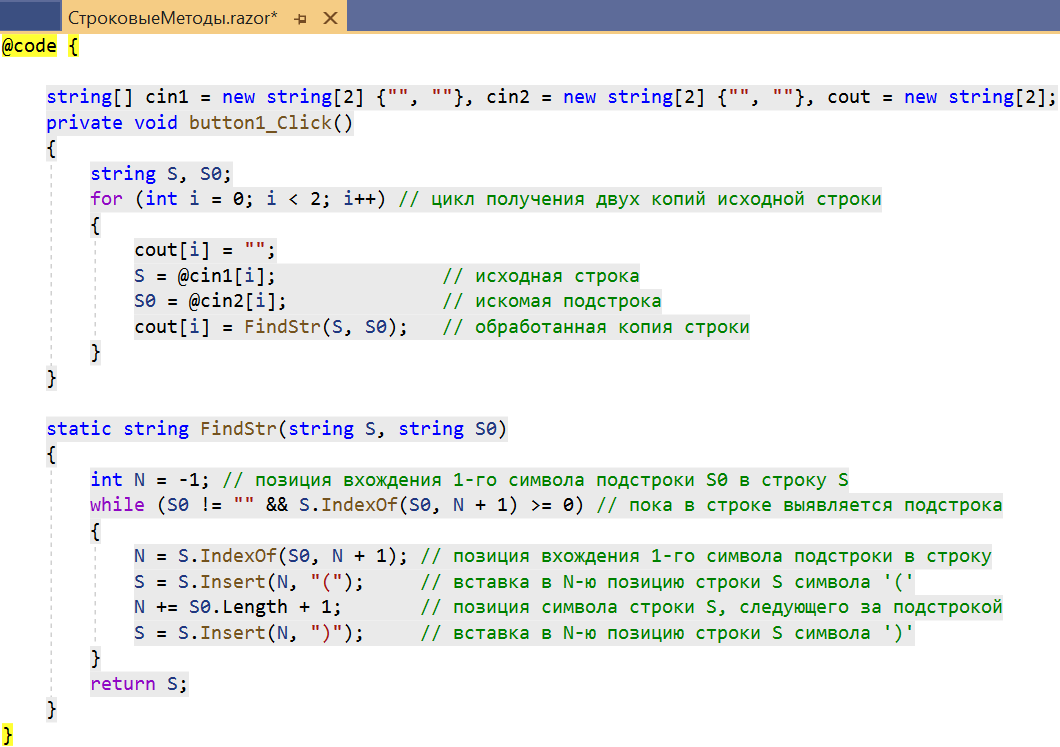
в) Интерфейс приложения WebAssembly Blazor:



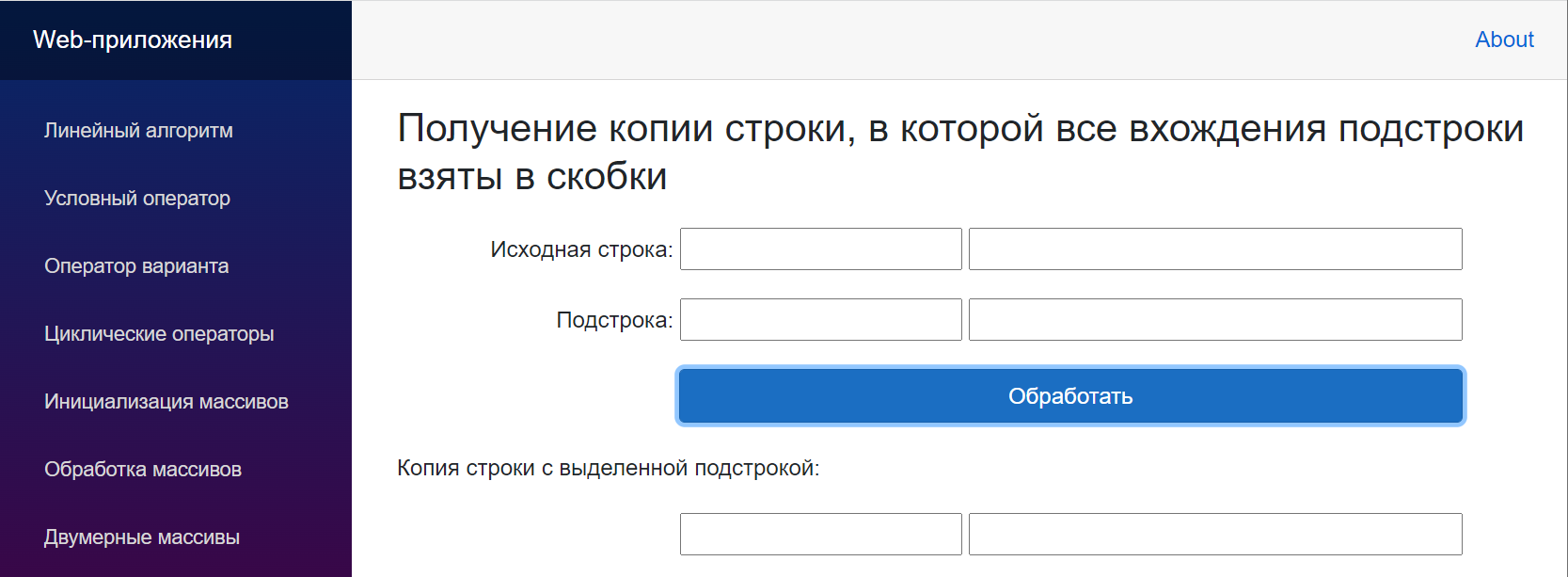


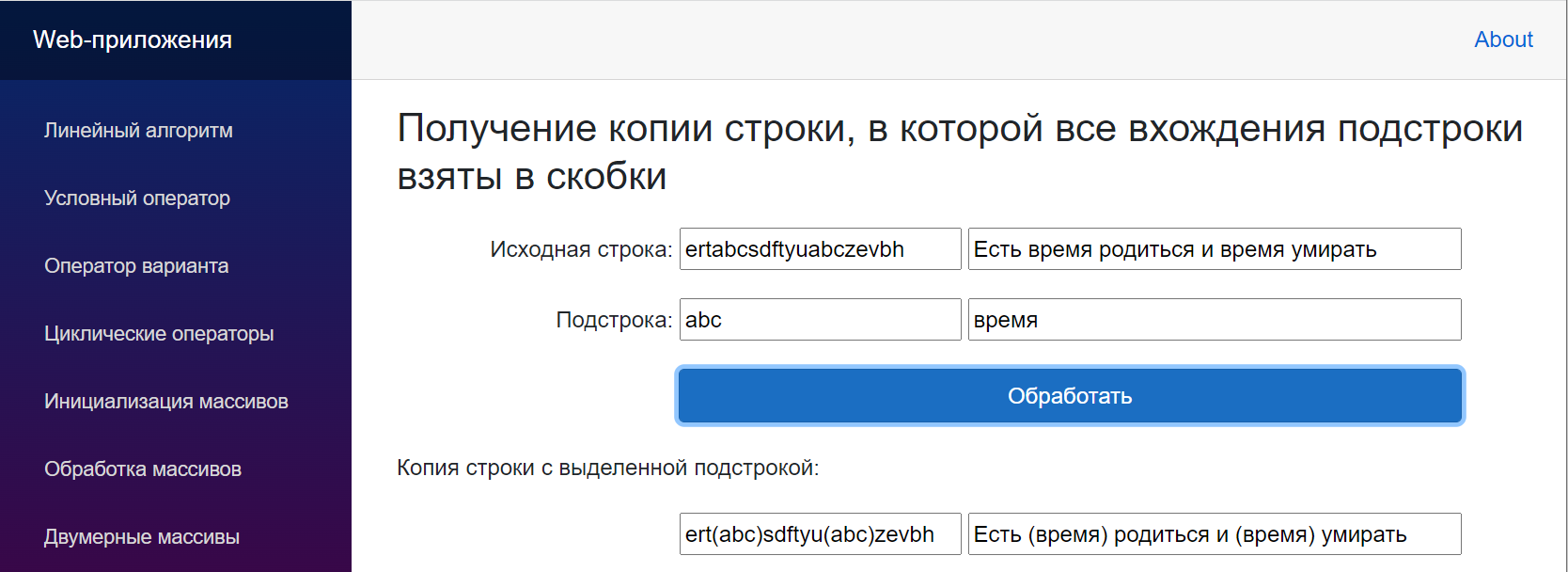


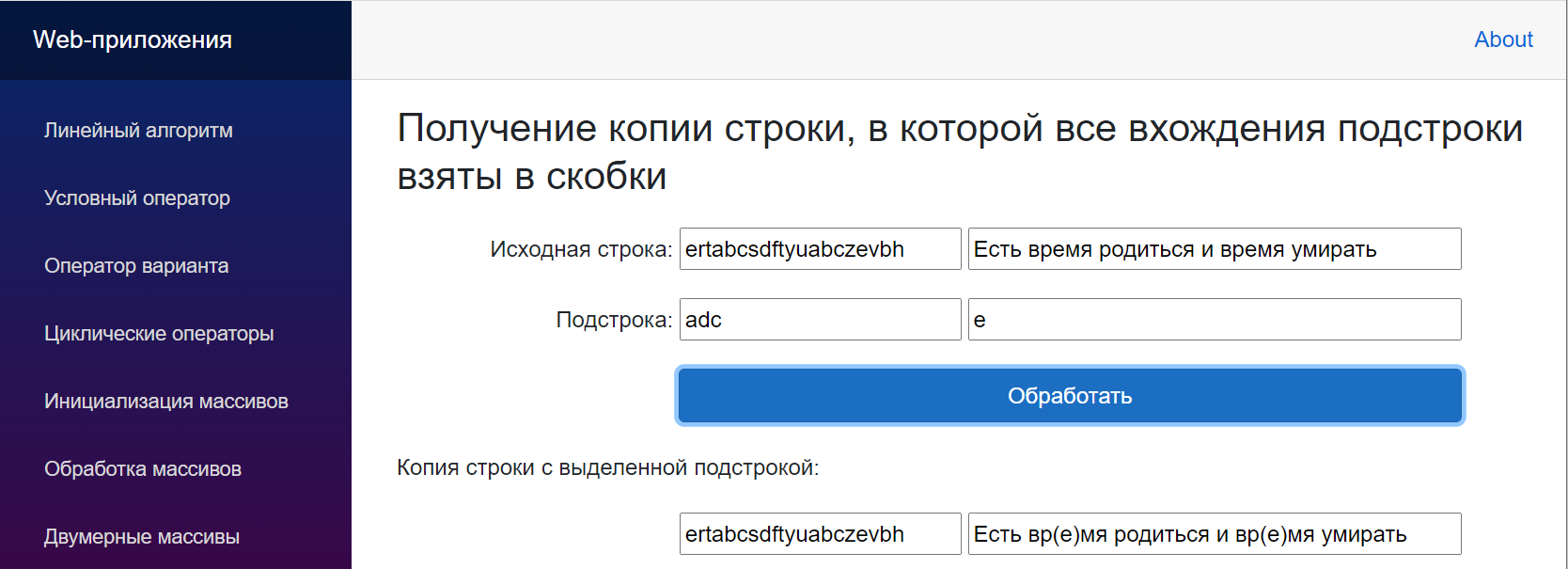
Коды приложения:



Скрины отлаженной программы:







**4 Порядок выполнения работы**

4.1 Изучить теоретические сведения и задание к работе.

4.2 В соответствии с вариантом задания создать формы приложений Windows Forms (WPF, WebAssembly Blazor).

4.3 В соответствии с вариантом задания разработать и отладить коды приложений Windows Forms (WPF, WebAssembly Blazor), используя создание и применение методов и исключения, выбрасываемые, если форматы или значения входных данных не соответствуют спецификациям параметров обрабатывающих их методов.

**5 Содержание отчета**

5.1 Название работы.

5.2 Цель работы.

5.3 Программные коды и скрины работы разработанных приложений Windows Forms в соответствии с вариантом задания.

**6 Варианты заданий**

6.1.1 Разработать метод PowerA3(*A*, *B*), вычисляющий третью степень числа *A* и возвращающий ее в переменной *B* (оба параметра являются вещественными). С помощью этого метода найти третьи степени пяти данных чисел.

6.1.2 Описать функцию IsIdent(*S*) целого типа, проверяющую, является ли строка *S* допустимым идентификатором, то есть непустой строкой, которая содержит только латинские буквы, цифры и символ подчеркивания «\_» и не начинается с цифры. Если *S* является допустимым идентификатором, то функция возвращает 0. Если *S* является пустой строкой, то возвращается –1, если *S* начинается с цифры, то возвращается –2. Если S содержит недопустимые символы, то возвращается номер первого недопустимого символа. Проверить с помощью функции IsIdent(*S*) две данные строки.

6.2.1 Разработать метод RemoveX(*A*, *N*, *X*), удаляющий из целочисленного массива *A* размера *N* элементы, равные целому числу *X*. Массив *A* и число *N* являются входными и выходными параметрами. С помощью этой процедуры удалить числа *XA*, *XB*, *XC* из массивов *A*, *B*, *C* размера *NA*, *NB*, *NC* соответственно и вывести размер и содержимое полученных массивов.

6.2.2 Описать функцию FillStr(*S*, *N*) строкового типа, возвращающую строку длины *N*, заполненную повторяющимися копиями строки-шаблона *S* (последняя копия строки-шаблона может входить в результирующую строку частично). Используя эту функцию, сформировать по данному числу *N* и двум строкам-шаблонам две результирующие строки длины *N*.

6.3.1 Разработать метод Mean(*X*, *Y*, *AMean*, *GMean*), вычисляющий среднее арифметическое *AMean* = (*X* + *Y*) / 2 и среднее геометрическое *GMea*n = двух положительных чисел *X* и *Y* (параметры вещественного типа). С помощью этого метода найти среднее арифметическое и среднее геометрическое для пар (*A*, *B*), (*A*, *C*), (*A*, *D*), если даны числа *A*, *B*, *C*, *D*.

6.3.2 Описать процедуру UpCaseRus(*S*), преобразующую все строчные русские буквы строки *S* в прописные (остальные символы строки *S* не из-меняются). Строка *S* является входным и выходным параметром. Используя процедуру UpCaseRus(*S*), преобразовать две данные строки.

6.4.1 Разработать метод TrianglePS(*a*, *P*, *S*), вычисляющий по стороне *a* равностороннего треугольника его периметр *P* = 3 × *a* и площадь *S* = /4 × *a*2 (*a* — входной, *P* и *S* — выходные параметры; все параметры являются вещественными). С помощью этого метода найти периметры и площади трех равносторонних треугольников с данными сторонами.

6.4.2 Описать процедуру LowCaseRus(*S*), преобразующую все прописные русские буквы строки *S* в строчные (остальные символы строки *S* не изменяются). Строка *S* является входным и выходным параметром. Используя процедуру LowCaseRus(*S*), преобразовать две данные строки.

6.5.1 Разработать метод RectPS(*x*1, *y*1, *x*2, *y*2, *P*, *S*), вычисляющий периметр *P* и площадь *S* прямоугольника со сторонами, параллельными осям координат, по координатам (*x*1, *y*1), (*x*2, *y*2) его противоположных вершин (*x*1, *y*1, *x*2, *y*2 — входные, *P* и *S* — выходные параметры вещественного типа). С помощью этого метода найти периметры и площади трех прямоугольников с данными противоположными вершинами.

6.5.2 Описать процедуру TrimLeftC(*S*, *C*), удаляющую в строке *S* начальные символы, совпадающие с символом *C*. Строка *S* является входным и выходным параметром. Дан символ *C* и две строки. Используя процедуру TrimLeftC(*S*, *C*), преобразовать данные строки.

6.6.1 Разработать метод DigitCountSum(*K*, *C*, *S*), находящий количество *C* цифр целого положительного числа *K*, а также их сумму *S* (*K* — входной, *C* и *S* — выходные параметры целого типа). С помощью этого метода найти количество и сумму цифр для каждого из пяти данных целых чисел.

6.6.2 Описать процедуру TrimRightC(*S*, *C*), удаляющую в строке *S* конечные символы, совпадающие с символом *C*. Строка *S* является входным и выходным параметром. Дан символ *C* и две строки. Используя процедуру TrimRightC(*S*, *C*), преобразовать данные строки.

6.7.1 Разработать метод InvDigits(*K*), меняющий порядок следования цифр целого положительного числа *K* на обратный (*K* — параметр целого типа, являющийся одновременно входным и выходным). С помощью этого метода поменять порядок следования цифр на обратный для каждого из пяти данных целых чисел.

6.7.2 Описать функцию InvertStr(*S*, *K*, *N*) строкового типа, возвращающую инвертированную подстроку строки *S*, содержащую в обратном порядке *N* символов строки *S*, начиная с ее *K*-го символа. Если *K* превосходит длину строки *S*, то возвращается пустая строка; если длина строки меньше *K*+*N*, то инвертируются все символы строки, начиная с ее *K*-го символа. Вывести значения функции InvertStr(*S*, *K*, *N*) для данной строки *S* и каждой из трех пар положительных целых чисел: (*K*1, *N*1), (*K*2, *N*2), (*K*3, *N*3).

6.8.1 Разработать метод AddRightDigit(*D*, *K*), добавляющий к целому положительному числу *K* справа цифру *D* (*D* — входной параметр целого типа, лежащий в диапазоне 0–9, *K* — параметр целого типа, являющийся одновременно входным и выходным). С помощью этого метода последовательно добавить к данному числу *K* справа данные цифры *D*1 и *D*2, выводя результат каждого добавления.

6.8.2 Описать функцию PosSub(*S*0, *S*, *K*, *N*) целого типа, возвращающую номер позиции, начиная с которой в строке *S* содержится первое вхождение строки *S*0, причем анализируются только *N* символов строки *S*, начиная с ее *K*-го символа (таким образом, PosSub(*S*0, *S*, *K*, *N*) обеспечивает поиск в подстроке). Если *K* превосходит длину строки *S*, то возвращается 0, если длина строки меньше *K*+*N*, то анализируются все символы строки, начиная с ее *K*-го символа. Если в требуемой подстроке строки *S* вхождения *S*0 отсутствуют, то функция возвращает 0. Вывести значения функции PosSub(*S*0, *S*, *K*, *N*) для данных строк *S*0, *S* и каждой из трех пар положительных целых чисел: (*K*1, *N*1), (*K*2, *N*2), (*K*3, *N*3).

6.9.1 Разработать метод AddLeftDigit(*D*, *K*), добавляющий к целому положительному числу *K* слева цифру *D* (*D* — входной параметр целого типа, лежащий в диапазоне 1–9, *K* — параметр целого типа, являющийся одновременно входным и выходным). С помощью этого метода последовательно добавить к данному числу K слева данные цифры *D*1 и *D*2, выводя результат каждого добавления.

6.9.2 Описать функцию PosLast(*S*0, *S*) целого типа, возвращающую номер позиции, начиная с которой в строке *S* содержится последнее вхождение подстроки *S*0. Считать, что перекрывающихся вхождений подстрок *S*0 строка *S* не содержит. Если в строке *S* отсутствуют подстроки *S*0, то функция возвращает 0. Вывести значения этой функции для двух данных пар строк *S*0 и *S*.

6.10.1 Разработать метод Swap(*X*, *Y*), меняющий содержимое переменных *X* и *Y* (*X* и *Y* — вещественные параметры, являющиеся одновременно входными и выходными). С его помощью для данных переменных *A*, *B*, *C*, *D* последовательно поменять содержимое следующих пар: *A* и *B*, *C* и *D*, *B* и *C* и вывести новые значения *A*, *B*, *C*, *D*.

6.10.2 Описать функцию PosK(*S*0, *S*, *K*) целого типа, возвращающую номер позиции, начиная с которой в строке *S* содержится *K*-е вхождение под-строки *S*0 (*K*>0). Если количество вхождений *S*0 в строке *S* меньше *K*, то функция возвращает 0. Считать, что перекрывающихся вхождений подстрок *S*0 строка *S* не содержит. Вывести значения этой функции для двух данных троек: *S*0, *S* и *K*.

6.11.1 Разработать метод Minmax(*X*, *Y*), записывающий в переменную *X* минимальное из значений *X* и *Y*, а в переменную *Y* — максимальное из этих значений (*X* и *Y* — вещественные параметры, являющиеся одновременно входными и выходными). Используя четыре вызова этого метода, найти минимальное и максимальное из данных чисел *A*, *B*, *C*, *D*.

6.11.2 Описать функцию WordK(*S*, *K*) строкового типа, возвращающую *K*-е слово строки *S* (словом считается набор символов, не содержащий пробелов и ограниченный пробелами или началом/концом строки). Если количество слов в строке меньше *K*, то функция возвращает пустую строку. Используя эту функцию, выделить из данной строки *S* слова с данными номерами *K*1, *K*2, *K*3.

6.12.1 Разработать метод SortInc3(*A*, *B*, *C*), меняющий содержимое переменных *A*, *B*, *C* таким образом, чтобы их значения оказались упорядоченными по возрастанию (*A*, *B*, *C* — вещественные параметры, являющиеся одновременно входными и выходными). С помощью этого метода упорядочить по возрастанию два данных набора из трех чисел: (*A*1, *B*1, *C*1) и (*A*2, *B*2, *C*2).

6.12.2 Описать процедуру SplitStr(*S*, *W*, *N*), которая формирует по данной строке *S* массив *W* слов, входящих в *S* (массив *W* и его размер *N* являются выходными параметрами). Словом считается набор символов, не содержащий пробелов и ограниченный пробелами или началом/концом строки; предполагается, что строка *S* содержит не более 10 слов. Используя функцию SplitStr(*S*, *W*, *N*), найти количество слов *N*, содержащихся в каждой из двух заданных строк *S*, и сами эти слова.

6.13.1 Разработать метод SortDec3(*A*, *B*, *C*), меняющий содержимое переменных *A*, *B*, *C* таким образом, чтобы их значения оказались упорядоченными по убыванию (*A*, *B*, *C* — вещественные параметры, являющиеся одновременно входными и выходными). С помощью этого метода упорядочить по убыванию два данных набора из трех чисел: (*A*1, *B*1, *C*1) и (*A*2, *B*2, *C*2).

6.13.2 Описать функцию CompressStr(*S*) строкового типа, выполняющую сжатие строки *S* по следующему правилу: каждая подстрока строки *S*, состоящая из более чем четырех одинаковых символов *C*, заменяется текстом вида «*С*{*K*}», где *K* — количество символов *C* (предполагается, что строка *S* не содержит фигурных скобок «{» и «}»). Например, для строки *S* = «bbbccccce» функция вернет строку «bbbc{5}e». С помощью функции CompressStr(*S*) сжать пять данных строк.

6.14.1 Разработать метод ShiftRight3(*A*, *B*, *C*), выполняющий правый циклический сдвиг: значение *A* переходит в *B*, значение *B* — в *C*, значение *C* — в *A* (*A*, *B*, *C* — вещественные параметры, являющиеся одновременно входными и выходными). С помощью этого метода выполнить правый циклический сдвиг для двух данных наборов из трех чисел: (*A*1, *B*1, *C*1) и (*A*2, *B*2, *C*2).

6.14.2 Описать функцию DecompressStr(*S*) строкового типа, восстанавливающую строку, сжатую процедурой CompressStr(*S*) (см. пред. вариант). Параметр *S* содержит сжатую строку; восстановленная строка является возвращаемым значением функции. С помощью функции DecompressStr(*S*) восстановить две данные сжатые строки.

6.15.1 Разработать метод ShiftLeft3(*A*, *B*, *C*), выполняющий левый циклический сдвиг: значение *A* переходит в *C*, значение *C* — в *B*, значение *B* — в *A* (*A*, *B*, *C* — вещественные параметры, являющиеся одновременно входными и выходными). С помощью этого метода выполнить левый циклический сдвиг для двух данных наборов из трех чисел: (*A*1, *B*1, *C*1) и (*A*2, *B*2, *C*2).

6.15.2 Описать функцию DecToBin(*N*) строкового типа, возвращающую строковое представление целого неотрицательного числа *N* в двоичной системе счисления. Результирующая строка состоит из символов «0»–«1» и не содержит ведущих нулей (за исключением представления числа 0). Используя эту функцию, получить двоичные представления двух данных чисел.

6.16.1 Разработать метод Invert(*A*, *N*), меняющий порядок следования элементов вещественного массива *A* размера *N* на противоположный (инвертированиемассива). Массив *A* является входным и выходным параметром. С помощью этой процедуры инвертировать массивы *A*, *B*, *C* размера *NA*, *NB*, *NC* соответственно.

6.16.2 Описать функцию DecToHex(*N*) строкового типа, возвращающую строковое представление целого неотрицательного числа *N* в 16-ричной системе счисления. Результирующая строка состоит из символов «0»–«9», «A»–«F» и не содержит ведущих нулей (за исключением представления числа 0). Используя эту функцию, получить 16-ричные представления пяти данных чисел.

6.17.1 Разработать метод PowerA234(*A*, *B*, *C*, *D*), вычисляющий вторую, третью и четвертую степень числа *A* и возвращающий эти степени соответственно в переменных *B*, *C* и *D* (все параметры являются вещественными). С помощью этого метода найти вторую, третью и четвертую степень трех данных чисел.

6.17.2 Описать функцию BinToDec(*S*) целого типа, определяющую целое неотрицательное число по его строковому представлению *S* в двоичной системе счисления. Параметр *S* имеет строковый тип, состоит из символов «0»–«1» и не содержит ведущих нулей (за исключением значения «0»). Используя эту функцию, вывести два числа, для которых даны их двоичные представления.

**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.