# Практическое занятие № 5. Программирование циклических алгоритмов

**1 Цель занятия**

Получить практические навыки использования циклических алгоритмов в приложениях Windows Forms.

**2 Перечень оборудования и программного обеспечения**

Персональный компьютер

Microsoft Office (Word)

Microsoft Visual Studio

**3 Краткие теоретические сведения**

**3.1 Циклы**

Циклы являются управляющими конструкциями, позволяя в зависимости от определенных условий выполнять некоторое действие множество раз. В C# имеются следующие виды циклов:

• for

• foreach

• while

• do**...**while

**3.2 Цикл for**

Цикл for имеет следующее формальное определение:

for (<инициализация счетчика>; <условие>; <изменение счетчика>)

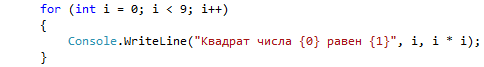
{

//действия

}

Рассмотрим стандартный цикл for.

**Пример 1:**



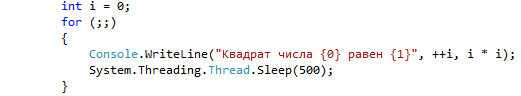
Первая часть объявления цикла  создает и инициализирует счетчик i. Счетчик необязательно должен представлять тип int. Это может быть и другой числовой тип, например, float. И перед выполнением цикла его значение будет равно 0. В данном случае это то же самое, что и объявление переменной.

Вторая часть - условие, при котором будет выполняться цикл. В данном случае цикл будет выполняться, пока i не достигнет 9.

И третья часть - приращение счетчика на единицу. Опять же нам необязательно увеличивать на единицу. Можно уменьшать: i--*.*

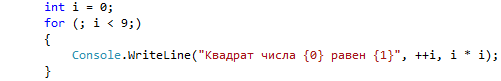
В итоге блок цикла сработает 9 раз, пока значение i не станет равным 9. И каждый раз это значение будет увеличиваться на 1. Нам необязательно указывать все условия при объявлении цикла.

**Пример 2:**



Формально определение цикла осталось тем же, только теперь блоки в определении у нас пустые: . У нас нет инициализированной переменной-счетчика, нет условия, поэтому цикл будет работать вечно - бесконечный цикл. Мы можем опустить при определении цикла ряд блоков.

**Пример 3:**



Этот пример, по сути, эквивалентен первому примеру: у нас также есть счетчик, только создан он вне цикла. У нас есть условие выполнения цикла. И есть приращение счетчика уже в самом блоке for.

**3.3 Цикл foreach**

Цикл foreach предназначен для перебора элементов в контейнерах. Формальное объявление цикла foreach:

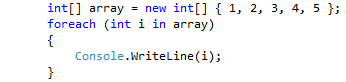
foreach (<тип\_данных> <название\_переменной> in <контейнер>)

{

//действия

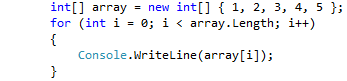
}

**Пример 4:**



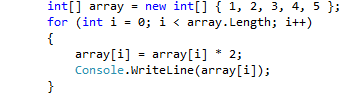
Здесь в качестве контейнера выступает массив данных типа int. Поэтому мы объявляем переменную с типом int. Подобные действия мы можем сделать и с помощью цикл for.

**Пример 5:**



В то же время цикл for более гибкий по сравнению с foreach. Если foreach последовательно извлекает элементы контейнера и только для чтения, то в цикле for мы можем перескакивать на несколько элементов вперед в зависимости от приращения счетчика, а также можем изменять элементы.

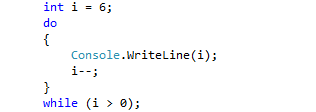
**Пример 6:**



**3.4 Цикл do**

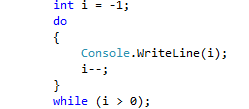
В цикле do сначала выполняется код цикла, а потом происходит проверка условия в инструкции while. И пока это условие истинно, цикл повторяется.

**Пример 7:**



Здесь код цикла сработает 6 раз, пока i не станет равным нулю. Но важно отметить, что цикл do гарантирует хотя бы однократное выполнение действий, даже если условие в инструкции while не будет истинно.

**Пример 8:**

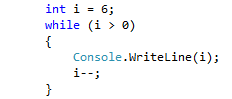


Хотя у нас переменная i меньше 0, цикл все равно один раз выполнится.

**3.5 Цикл while**

В отличие от цикла do цикл while сразу проверяет истинность некоторого условия, и если условие истинно, то код цикла выполняется.

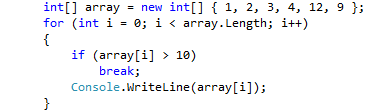
**Пример 9:**



**3.6 Операторы continue и break**

Иногда возникает ситуация, когда требуется выйти из цикла, не дожидаясь его завершения. В этом случае мы можем воспользоваться оператором break.

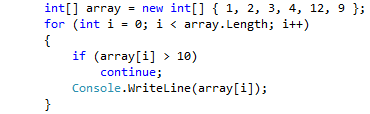
**Пример 10:**



Поскольку в цикле идет проверка, больше ли элемент массива 10, то мы никогда не увидим на консоли последние два элемента, так как, увидев, что элемент массива больше 10, сработает оператор break, и цикл завершится.

Теперь поставим себе другую задачу. А что если мы хотим, чтобы при проверке цикл не завершался, а просто переходил к следующему элементу. Для этого мы можем воспользоваться оператором continue.

**Пример 11:**

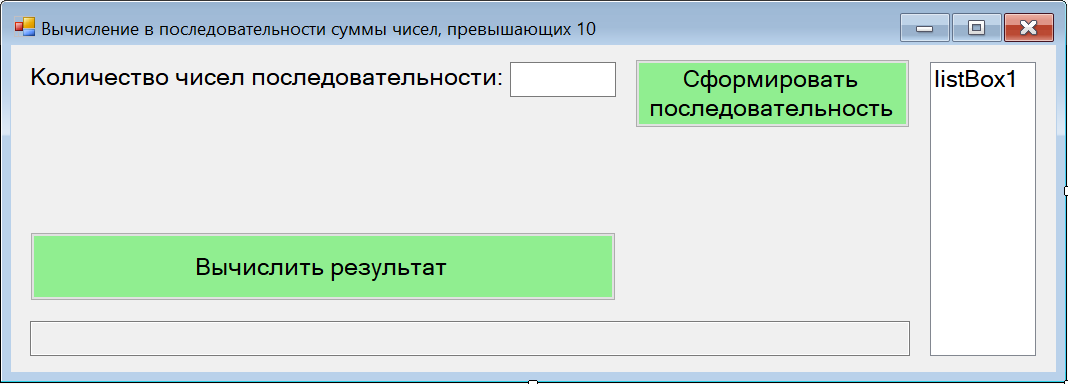


В этом случае цикл, когда дойдет до числа 12, которое не удовлетворяет условию проверки, просто пропустит это число и перейдет к следующему элементу массива.

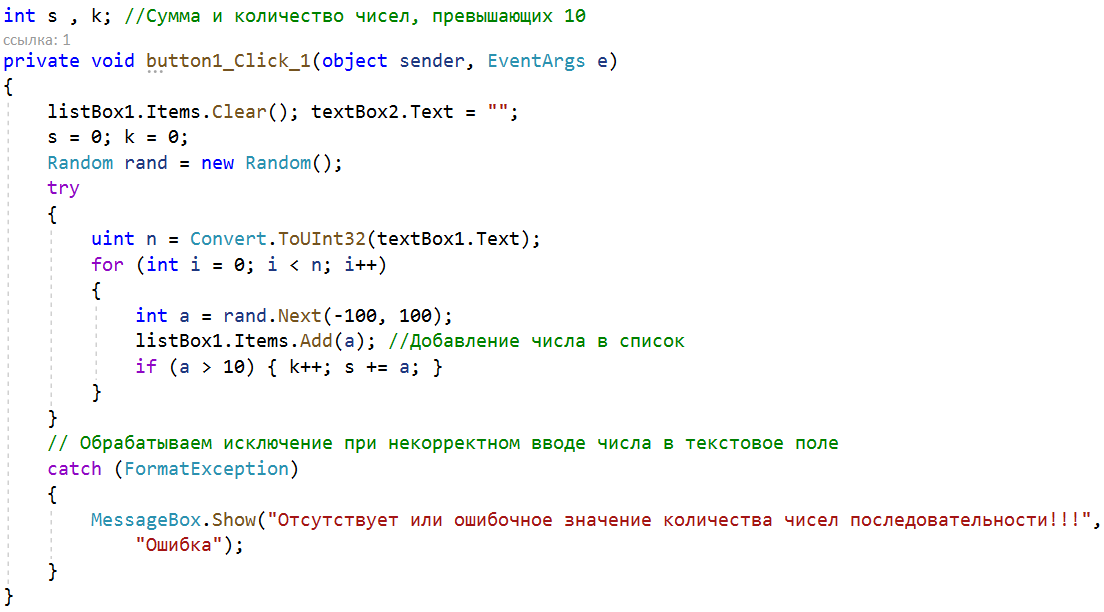
**Задание.** Осуществить ввод последовательности *N* целых чисел и подсчитать сумму чисел, больших 10. Если окажется, что таковых чисел в последовательности не было, вывести соответствующее сообщение.

**Решение.**

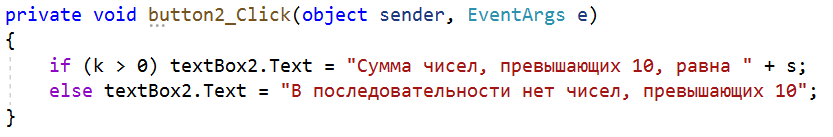
а) Интерфейс приложения Windows Forms:



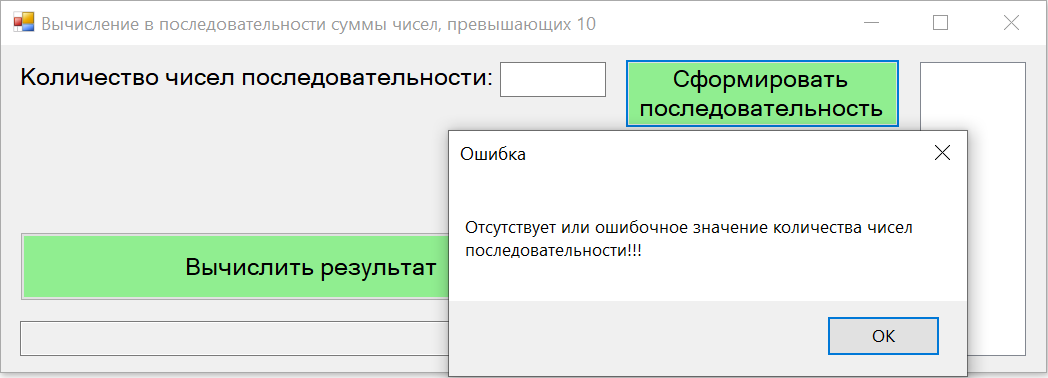
Код, выполняемый при щелчке на кнопку «Сформировать последовательность»:

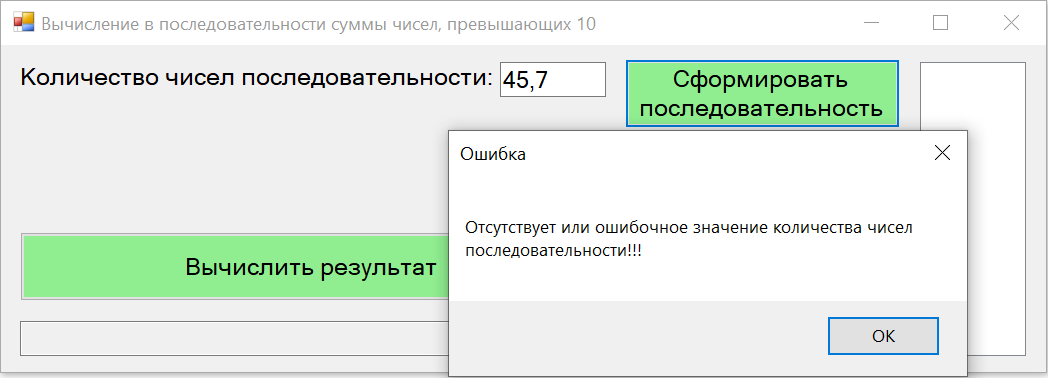


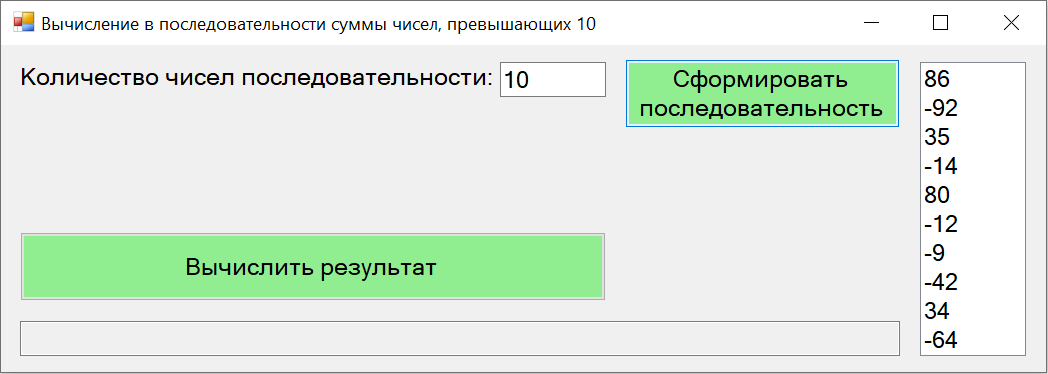
Код, выполняемый при щелчке на кнопку «Вычислить результат»:

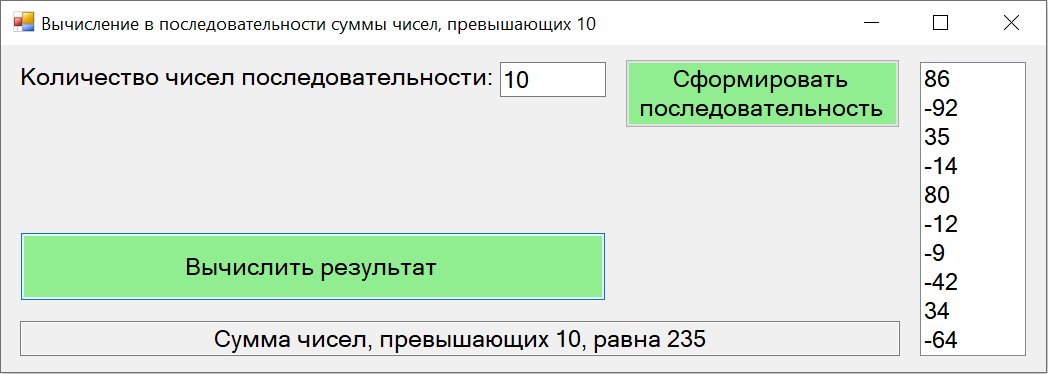


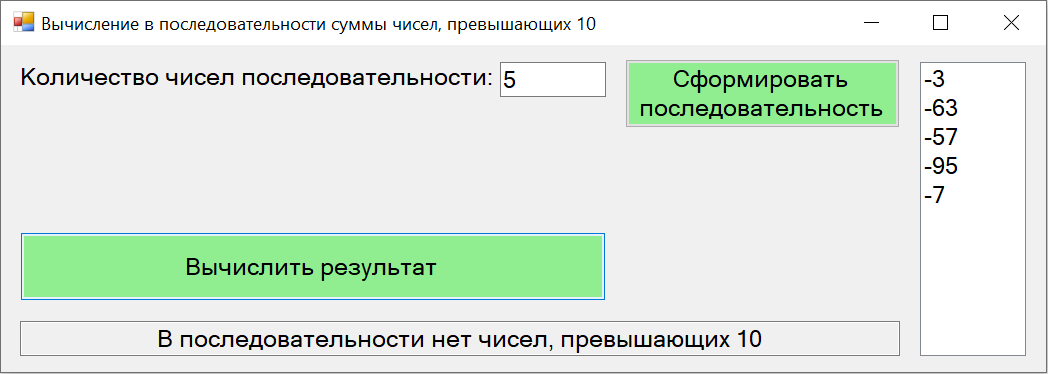
Скрины отлаженной программы:



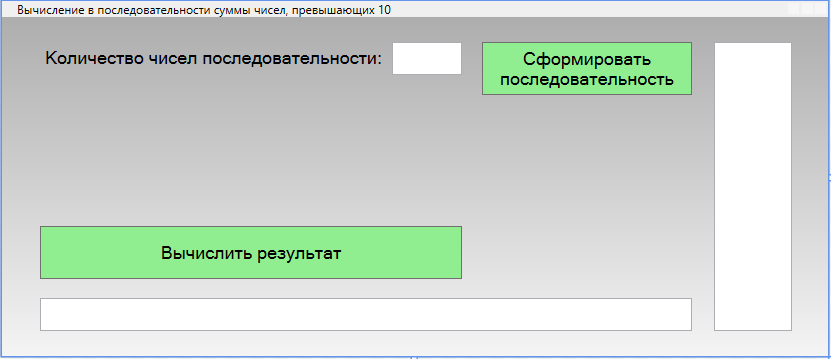




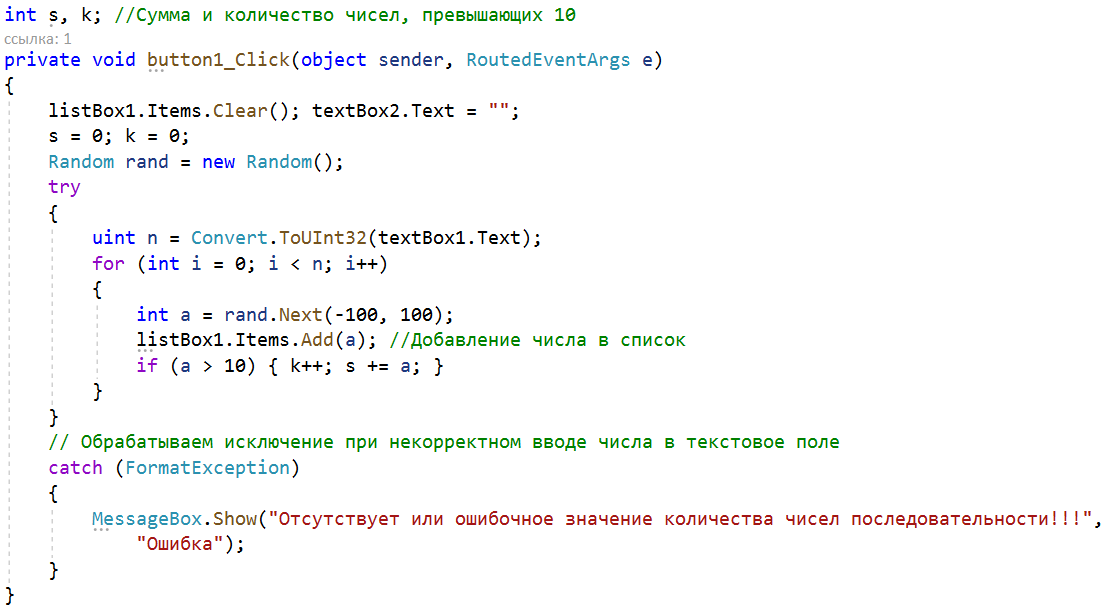




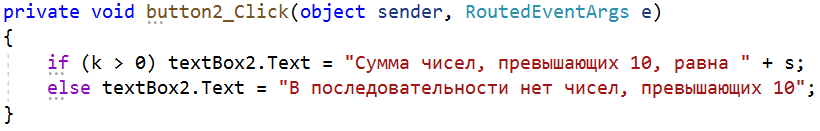
б) Интерфейс приложения WPF:



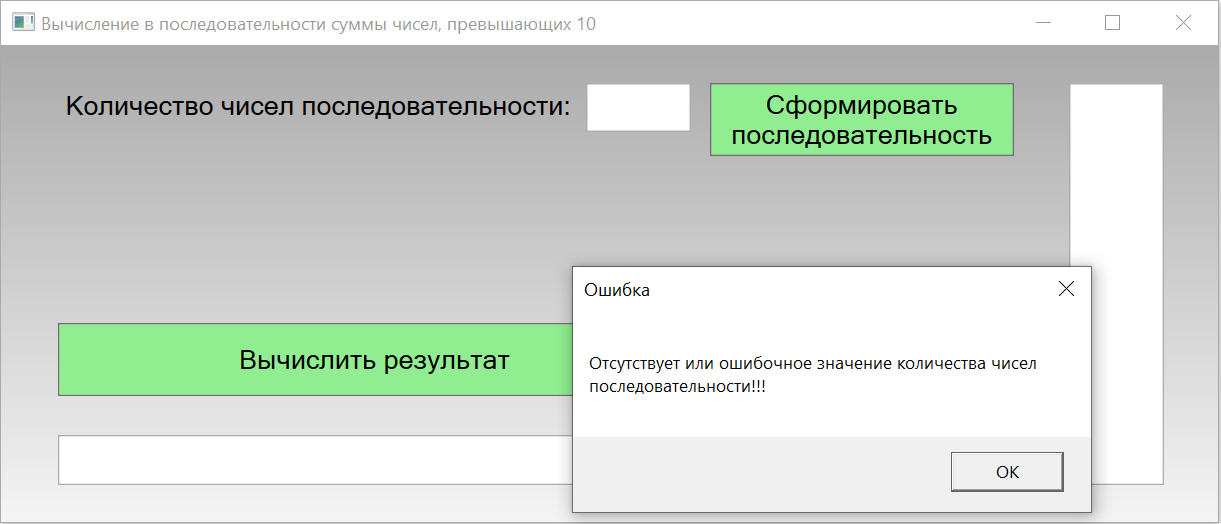
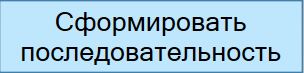
Код, выполняемый при щелчке на кнопку «Сформировать последовательность»:

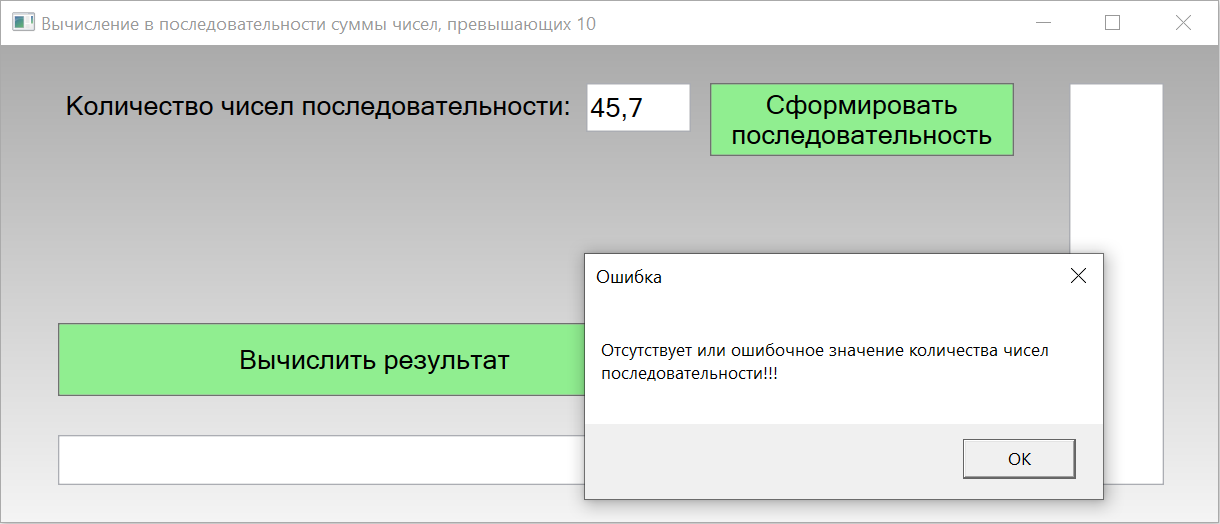
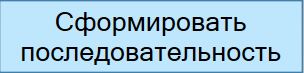


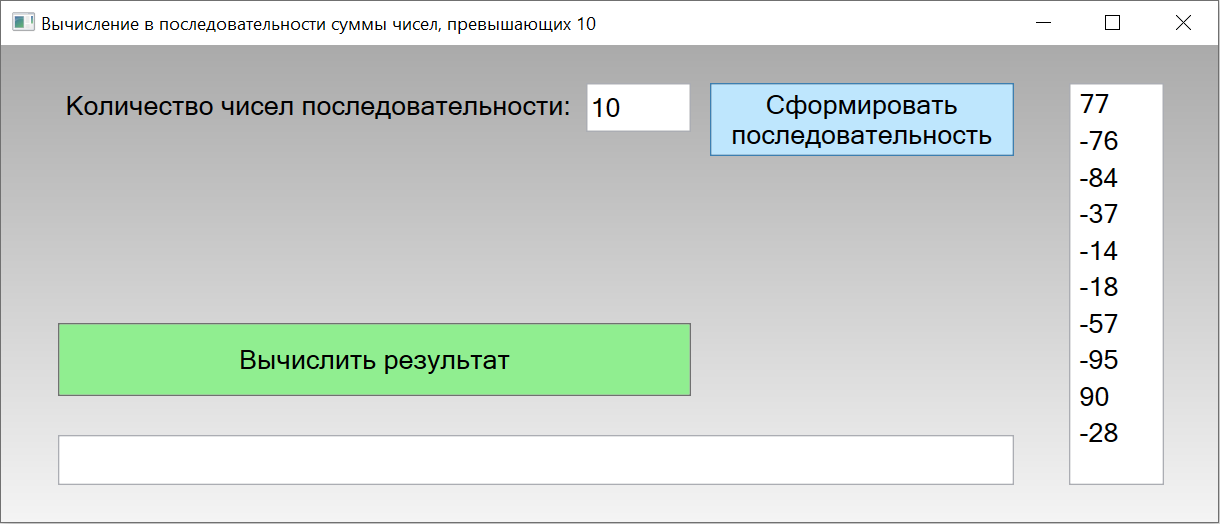
Код, выполняемый при щелчке на кнопку «Вычислить результат»:

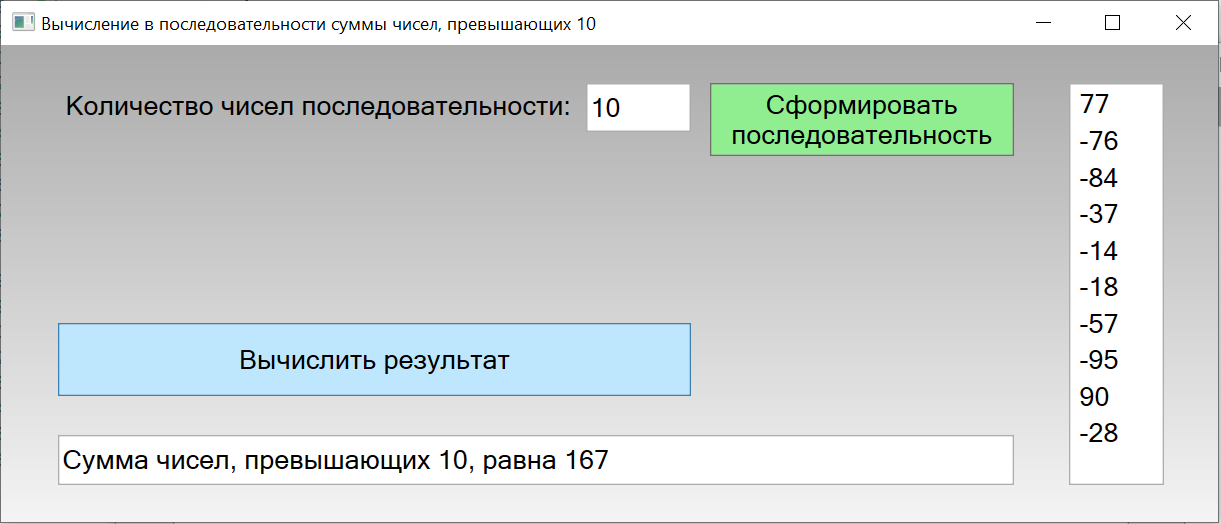


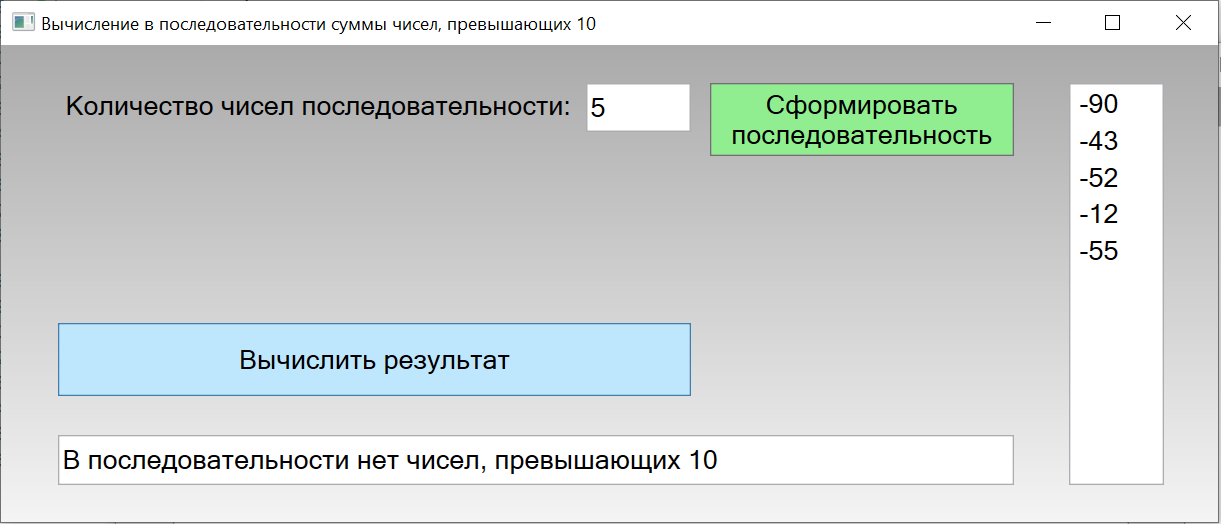
Скрины отлаженной программы:



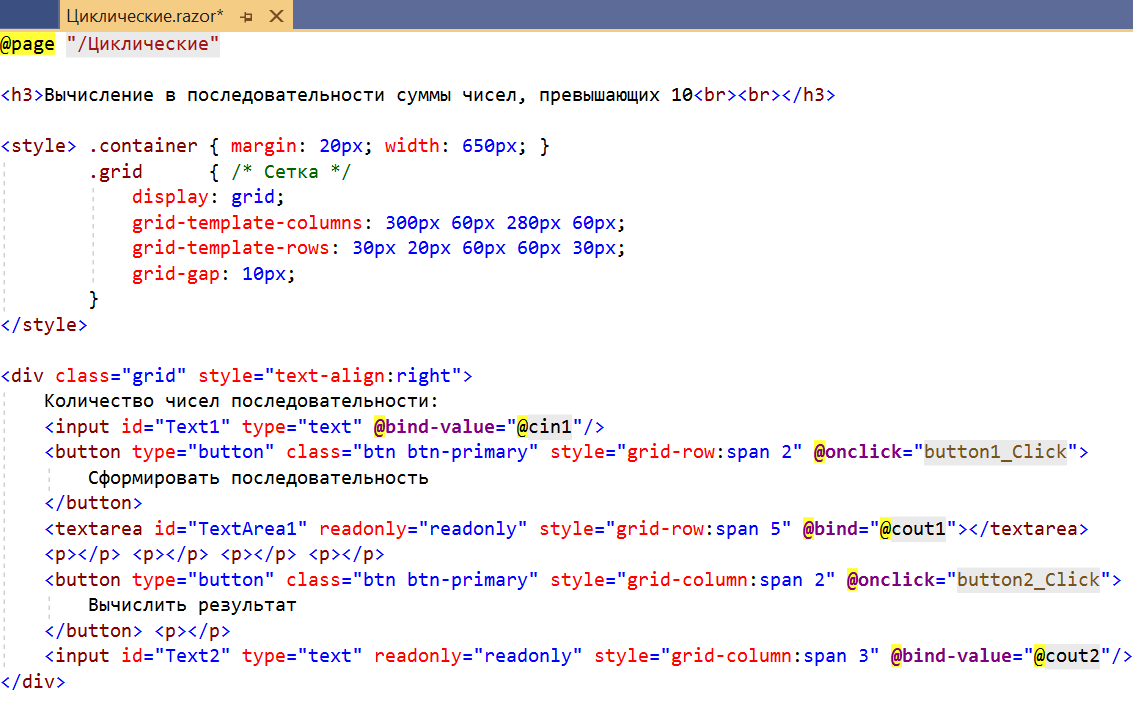


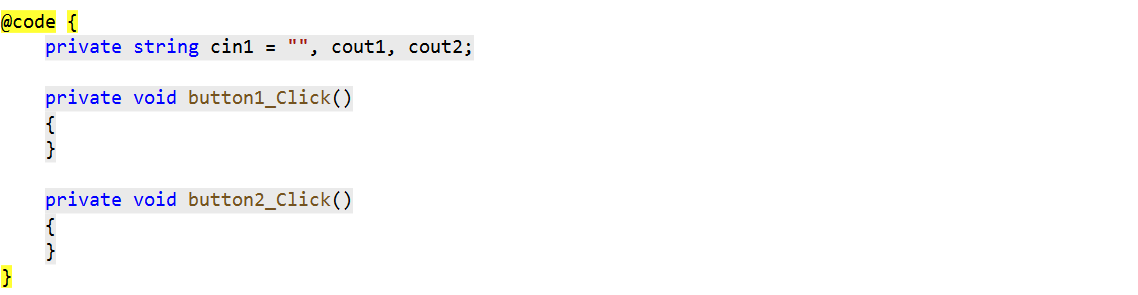
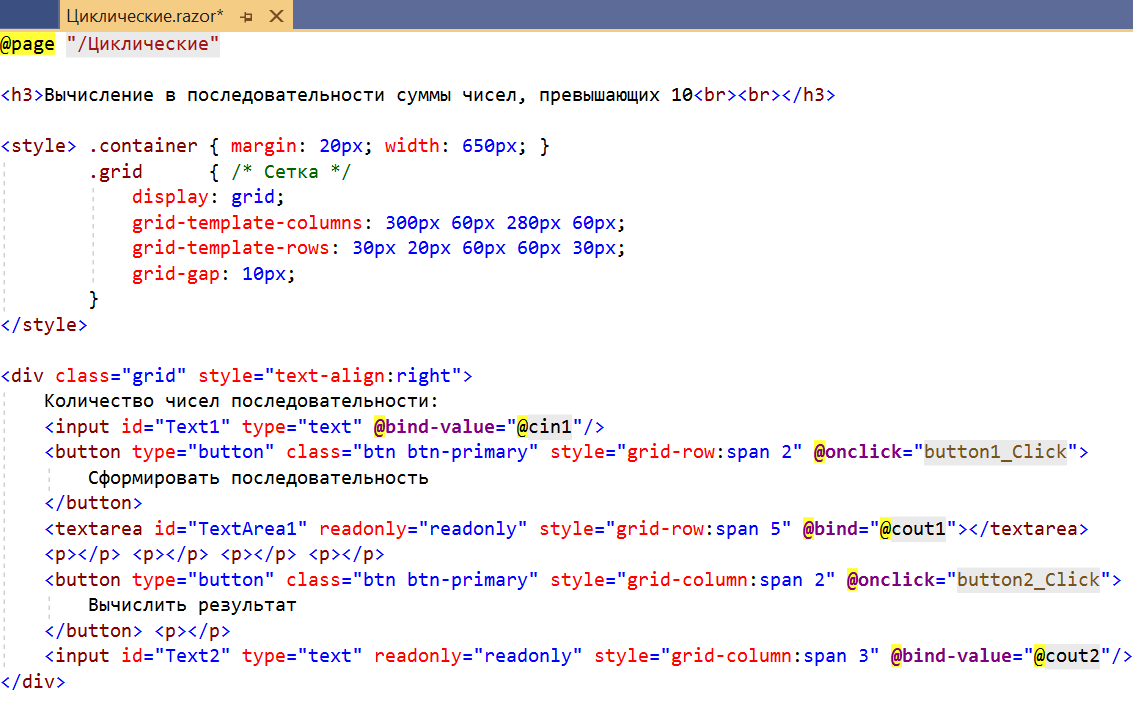


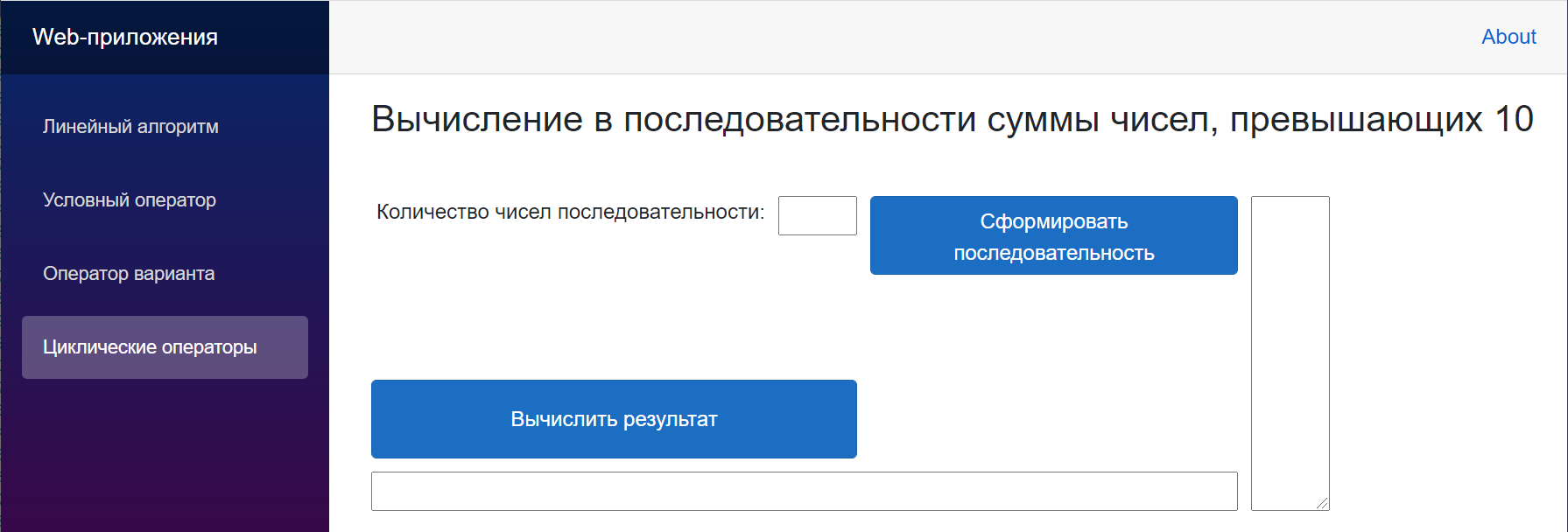




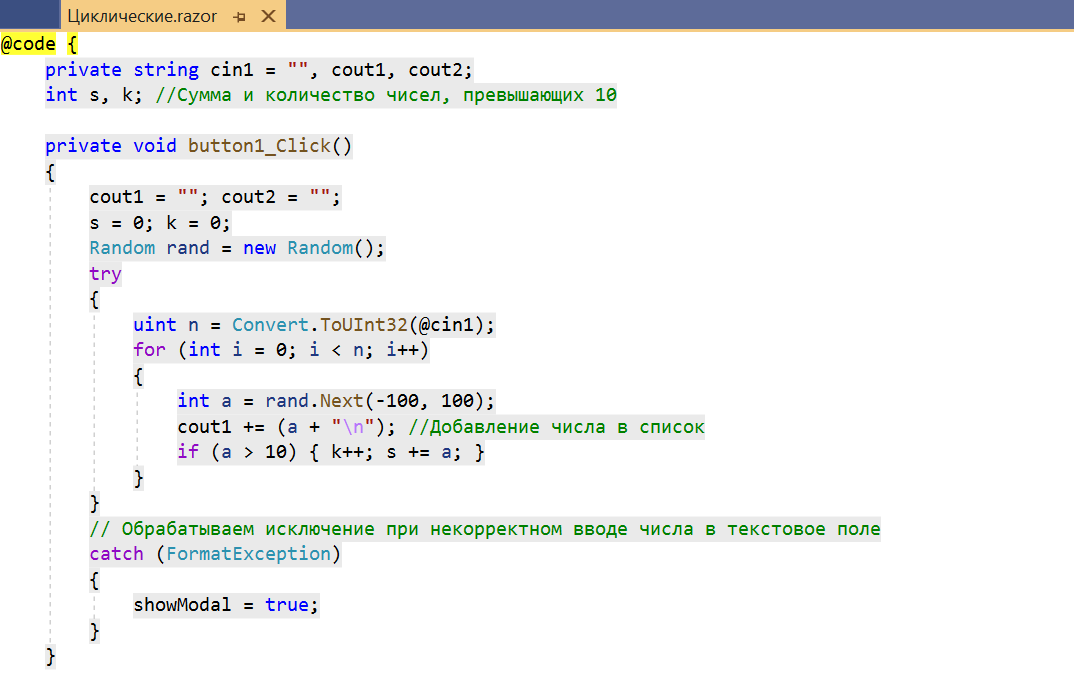
в) Интерфейс приложения WebAssembly Blazor:

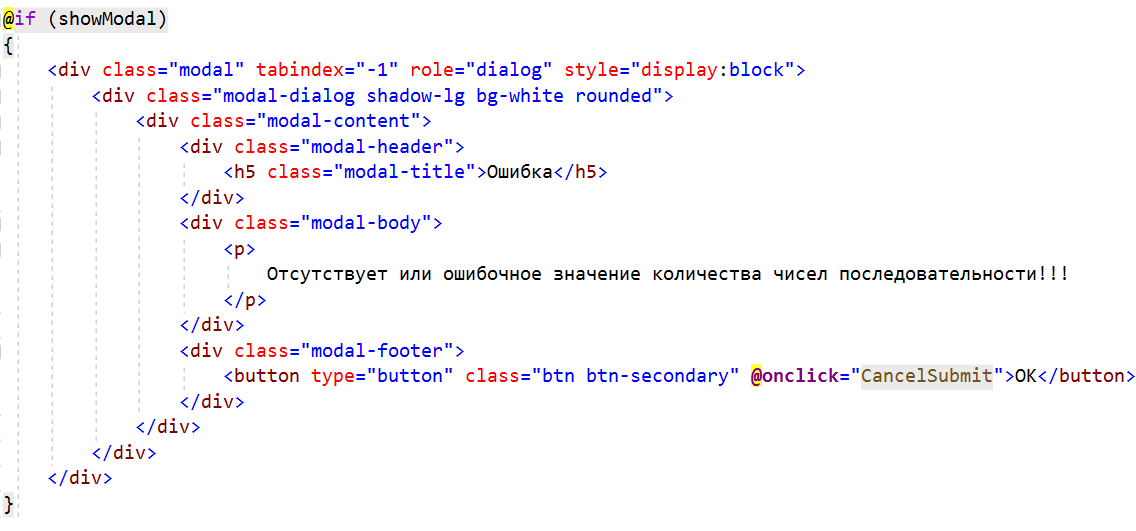
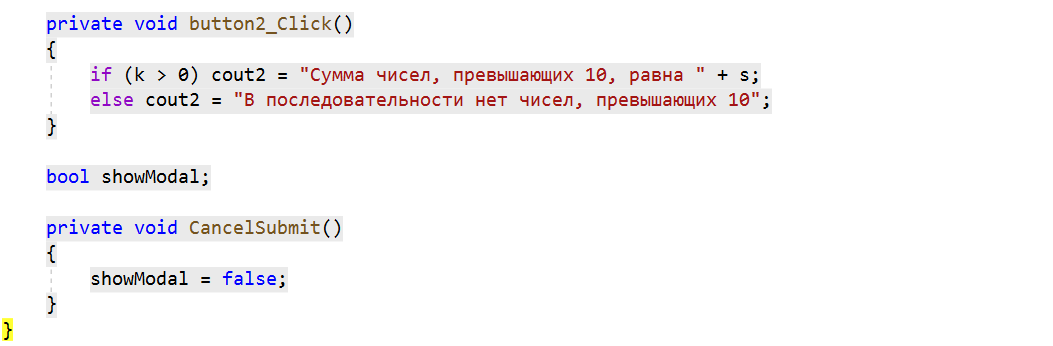




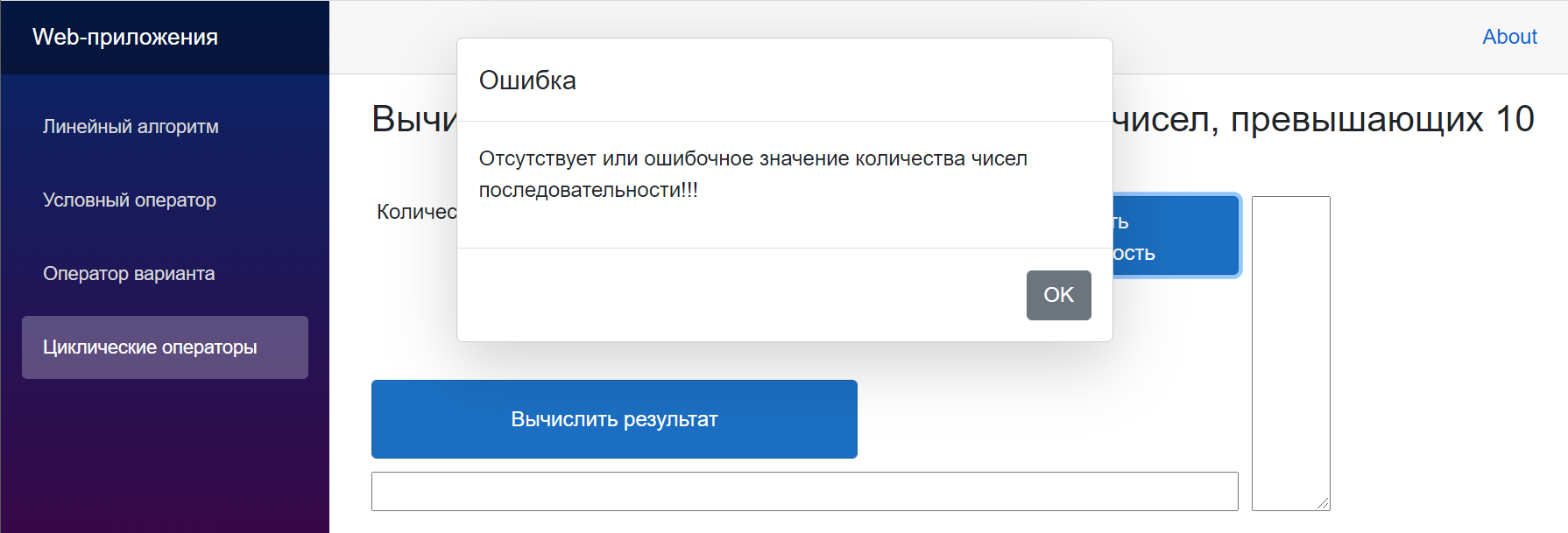


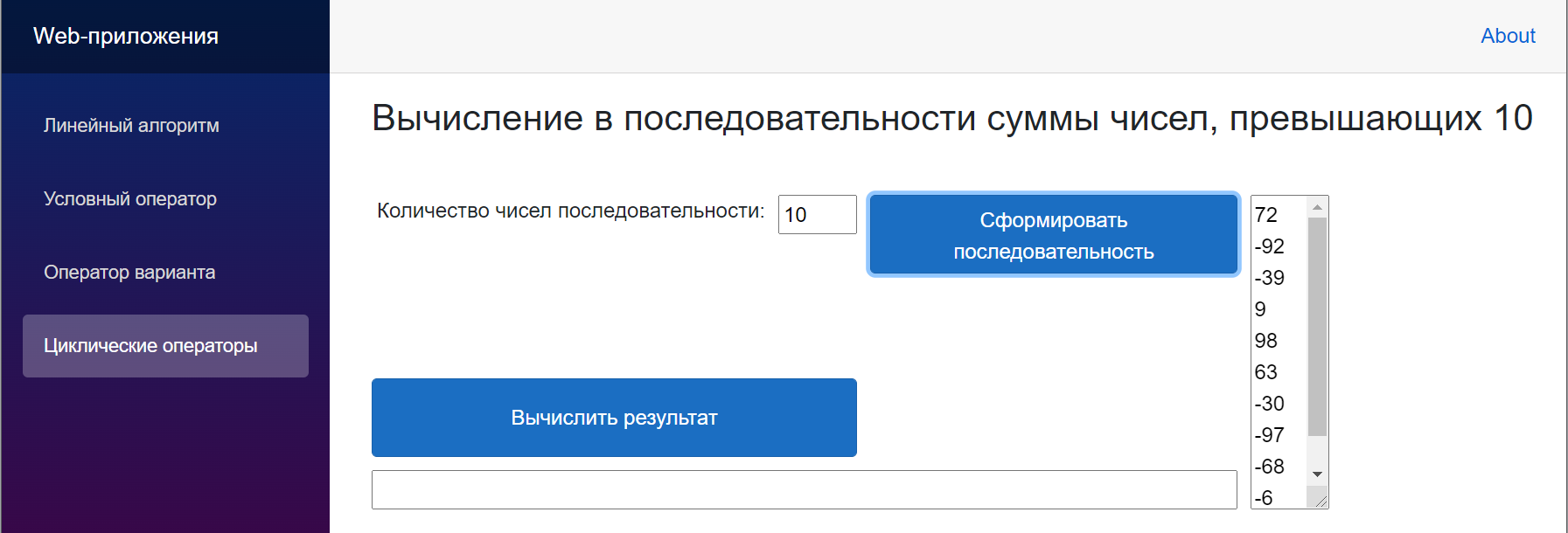
Коды приложения:

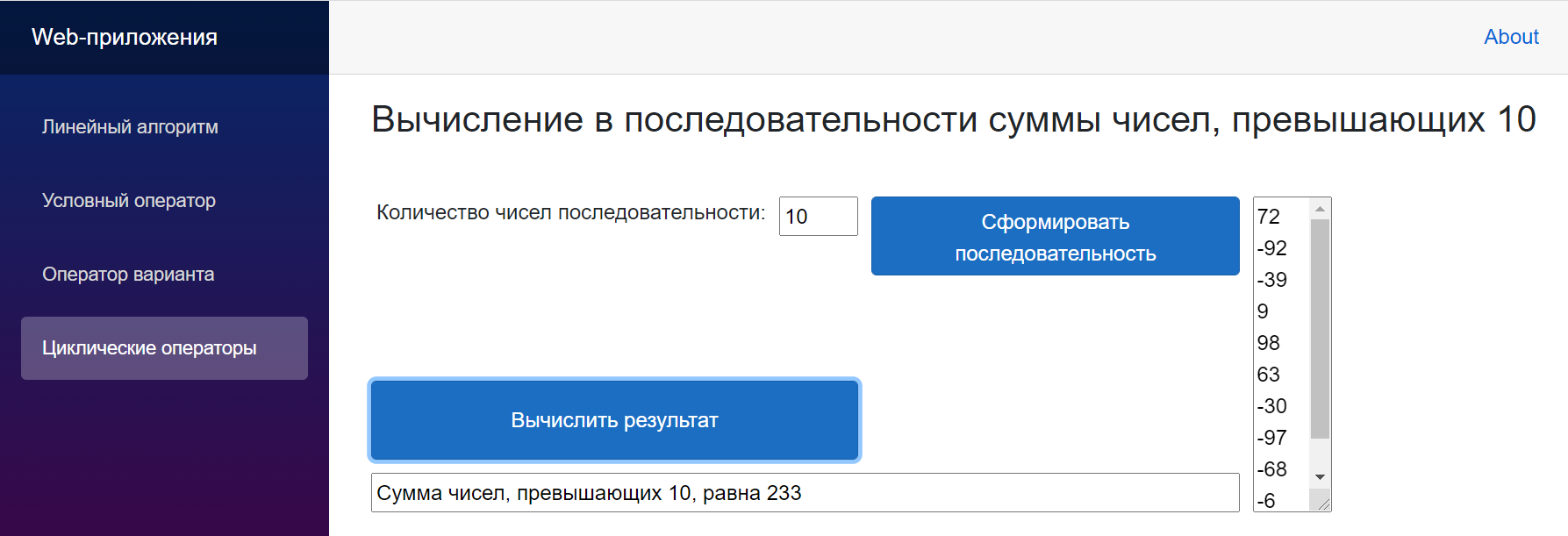


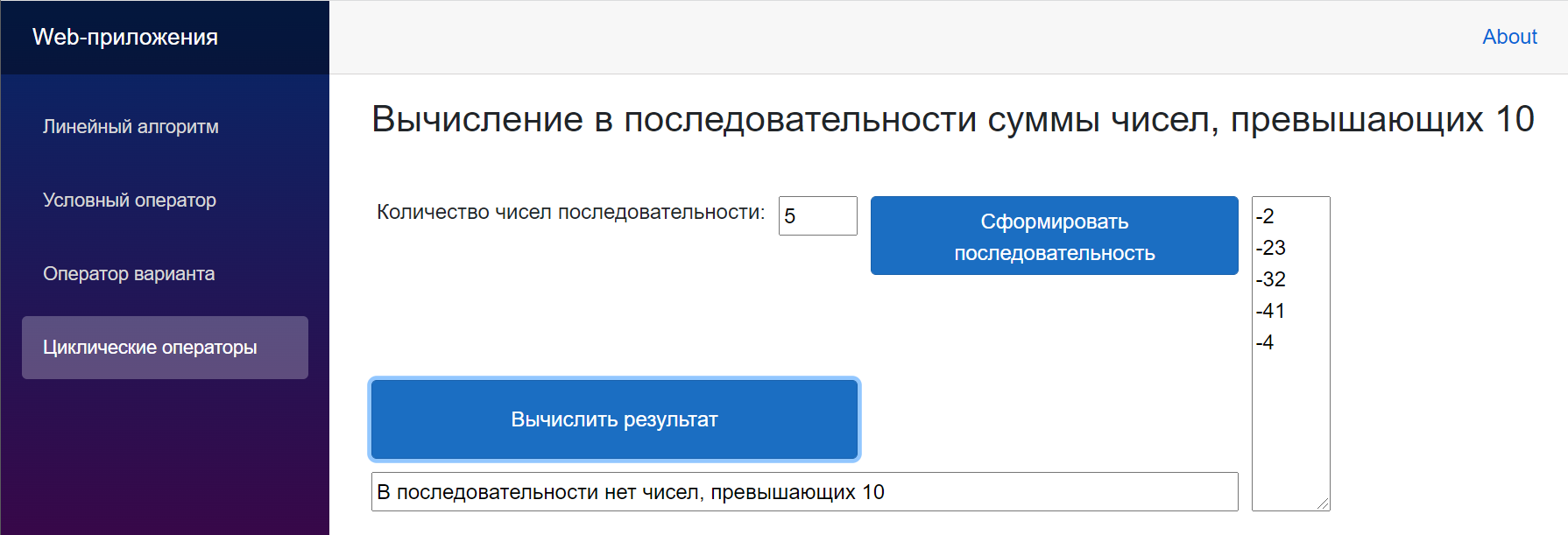


Скрины отлаженной программы:









**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 (WPF, WebAssembly Blazor).

4.3 В соответствии с вариантом задания разработать и отладить коды приложения Windows Forms (WPF, WebAssembly Blazor), используя оператор цикла и исключение, выбрасываемое, если форматы входных значений 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 разработанного приложения Windows Forms в соответствии с вариантом задания.

**6 Варианты задания**

6.1 Осуществить ввод последовательности *N* целых чисел. Ввести два целых числа *N*1 и *N*2. и найти сумму чисел в диапазоне номеров *N*1 ... *N*2. Если *N*1 равно *N*2, вывести на экран соответствующее сообщение.

6.2 Осуществить ввод последовательности *N* целых чисел, определить максимальное четное число, его порядковый номер и подсчитать сумму его цифр. Если окажется, что чётных чисел в последовательности не было, вывести соответствующее сообщение.

6.3 Осуществить ввод последовательности *N* целых чисел и сравнить, что больше, сумма положительных или произведение отрицательных.

6.4 Осуществить ввод последовательности *N* целых чисел и определить предпоследнее отрицательное число. Если окажется, что в последовательности было менее двух отрицательных чисел, вывести соответствующее сообщение.

6.5 Осуществить ввод последовательности *N* целых чисел, определить третье положительное число и подсчитать количество цифр в нем.

6.6 Осуществить ввод последовательности *N* ненулевых целых чисел и проверить, чередуются ли в ней положительные и отрицательные числа. Если чередуются, то вывести «Чередуются», если нет, то вывести порядковый номер первого элемента, нарушающего закономерность и сообщение о нарушении.

6.7 Осуществить ввод последовательности *N* целых чисел и определить номер последнего четного числа в последовательности. Если четных чисел нет — вывести сообщение «Нет четных чисел».

6.8 Осуществить ввод последовательности *N* целых чисел. Вывести на экран среднее арифметическое значение для всех нечетных элементов последовательности. Предусмотреть выдачу сообщения «Нет нечетных чисел», если нечетных чисел нет.

6.9 Осуществить ввод последовательности *N* целых чисел. Вывести на экран произведение тех элементов последовательности, номера которых кратны трем.

6.10 Осуществить ввод последовательности *N* целых чисел. Если произведение отрицательных чисел последовательности больше, чем произведение нечетных чисел, то присвоить переменной *Т* значение 1, иначе — присвоить *T* значение 0. Вывести на экран значение *Т* с пояснениями.

6.11 Осуществить ввод последовательности *N* целых чисел, осуществляя суммирование чисел с нечетными номерами. Результат вывести на экран.

6.12 Осуществить ввод последовательности *N* целых чисел. Получите корень из суммы квадратов всех элементов последовательности (модуль вектора).

6.13 Осуществить ввод последовательности *N* целых чисел. Найти сумму тех из них, которые делятся на 5 или на 7. Если окажется, что таких чисел в последовательности не было, вывести соответствующее сообщение.

6.14 Осуществить ввод последовательности *N* целых чисел. Найти произведение двузначных нечетных чисел, кратных 13. Если окажется, что таких чисел в последовательности не было, вывести соответствующее сообщение.

6.15 Осуществить ввод последовательности *N* целых чисел. Найти сумму чисел от 100 до 200, кратных 17. Если окажется, что таких чисел в последовательности не было, вывести соответствующее сообщение.

6.16 Осуществить ввод последовательности *N* целых чисел. Найти количество чисел, не делящихся ни на одно из чисел 2, 3 и 5.

6.17 Осуществить ввод последовательности *N* целых чисел. Найти количество простых чисел в последовательности.

**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.