

МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

УТВЕРЖДАЮ

Зав.кафедрой,

г. Саратов, к. ф.-м. н., доцент

\_\_\_\_\_ С. В. Миронов

**ОТЧЕТ О ПРАКТИКЕ**

студента 2 курса 251 группы факультета КНиИТ

Смирнова Егора Ильича

вид практики: научно-исследовательская работа (учебная практика,  
рассредоточенная)

кафедра: математической кибернетики и компьютерных наук

курс: 2

семестр: 3

продолжительность: 18 нед., с 02.09.2024 г. по 12.01.2025 г.

Руководитель практики от университета,

к. ф.-м. н., доцент

\_\_\_\_\_

Сафрончик М. И.

Руководитель практики от организации (учреждения, предприятия),

к. ф.-м. н., доцент

\_\_\_\_\_

Сафрончик М. И.

Тема практики: «Разработка приложений Windows.Forms на языке C++ в среде Microsoft Visual Studio»

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	6
1 Вычисление факториала .....	7
1.1 Условие задания .....	7
1.2 Вид формы в конструкторе .....	7
1.3 Таблица с описанием переименованных элементов формы .....	8
1.4 Примеры работы .....	8
1.5 Примеры кода .....	9
2 Простые вычисления (Задание 2 вариант 7) .....	11
2.1 Условие задания .....	11
2.2 Вид формы в конструкторе .....	11
2.3 Таблица с описанием переименованных элементов формы .....	11
2.4 Примеры работы .....	12
2.5 Примеры кода .....	14
3 Рекурсивные вычисления (Задание 3 вариант 7) .....	15
3.1 Условие задания .....	15
3.2 Вид формы в конструкторе .....	15
3.3 Таблица с описанием переименованных элементов формы .....	16
3.4 Примеры работы .....	17
3.5 Примеры кода .....	18
4 Обработка табличных данных. Часть 1 (Задание 4 вариант 7) .....	19
4.1 Условие задания .....	19
4.2 Вид формы в конструкторе .....	19
4.3 Таблица с описанием переименованных элементов формы .....	20
4.4 Примеры работы .....	22
4.5 Примеры кода .....	24
5 Обработка табличных данных. Часть 2 (Задание 5 вариант 13) .....	25
5.1 Условие задания .....	25
5.2 Вид формы в конструкторе .....	26
5.3 Таблица с описанием переименованных элементов формы .....	26
5.4 Примеры работы .....	27
5.5 Примеры кода .....	28
6 Матричный калькулятор (Задание 6) .....	30
6.1 Условие задания .....	30

6.2	Вид формы в конструкторе .....	30
6.3	Таблица с описанием переименованных элементов формы .....	30
6.4	Примеры работы .....	33
6.5	Примеры кода .....	35
7	Использование коллекций (Задание 7 вариант 8) .....	37
7.1	Условие задания .....	37
7.2	Вид формы в конструкторе .....	37
7.3	Таблица с описанием переименованных элементов формы .....	38
7.4	Примеры работы .....	40
7.5	Примеры кода .....	41
8	Работа с файлами (Задание 8 вариант 1) .....	43
8.1	Условие задания .....	43
8.2	Вид формы в конструкторе .....	43
8.3	Таблица с описанием переименованных элементов формы .....	44
8.4	Примеры работы .....	46
8.5	Примеры кода .....	47
9	Приложение Тест (Задание 9) .....	48
9.1	Условие задания .....	48
9.2	Вид формы в конструкторе .....	48
9.3	Таблица с описанием переименованных элементов формы .....	48
9.4	Примеры работы .....	49
9.5	Примеры кода .....	51
ЗАКЛЮЧЕНИЕ .....		52
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....		53
Приложение А	Фрагменты кода программы «Вычисление факториала» ...	54
Приложение Б	Фрагменты кода программы «Простые вычисления» .....	55
Приложение В	Фрагменты кода программы «Рекурсивные вычисления» ...	56
Приложение Г	Фрагменты кода программы «Обработка табличных данных. Часть 1.» .....	57
Приложение Д	Фрагменты кода программы «Обработка табличных данных. Часть 2.» .....	59
Приложение Е	Фрагменты кода программы «Матричный калькулятор» ....	62
Приложение Ж	Фрагменты кода программы «Использование коллекций» .	65
Приложение З	Фрагменты кода программы «Работа с файлами» .....	68

Приложение И	Фрагменты кода программы «Тест» .....	71
Приложение К	Архив с отчетом о выполненной работе .....	73

## **ВВЕДЕНИЕ**

Целью практики является освоение механизма построения оконного интерфейса приложений в среде Microsoft Visual Studio на языке C++/CLI с использованием .NET Framework и Windows Forms [1, 2].

В результате прохождения практики должны быть отработаны следующие навыки:

- Создание нового проекта
- Добавление и настройка элементов управления
- Проверка пользовательского ввода данных для решения поставленной задачи, обработка ошибок ввода
- Разработка алгоритма решения поставленной задачи с использованием оконного интерфейса
- Тестирование приложения
- Документирование разработанного кода

# 1 Вычисление факториала

## 1.1 Условие задания

Разработать приложение для вычисления факториала по приведенному примеру [3].

Приложение должно содержать следующие компоненты:

1. Заголовок формы должен отражать суть задания.
2. Все элементы формы должны быть внятно подписаны (кнопки подписаны, у текстового поля должно быть написано, для чего оно нужно и т. д.)
3. В коде должны быть комментарии и отступы (код должен быть легко читаем).
4. В коде программы все элементы формы должны быть переименованы (btnName — для кнопок, lblName — для ссылок, txtName — для текстового поля и т. д.) Наименования должны быть понятными.
5. Приложение должно корректно работать (выводить ответ или ошибку с соответствующим сообщением) для следующих данных: ввод буквы, ввод отрицательного числа, ввод нуля, ввод положительного числа ( $< 10$ ), ввод большого положительного числа. После вывода ошибок при вводе корректных данных поля ошибок должны очищаться.

## 1.2 Вид формы в конструкторе

Форма имеет вид как показано на рис.1:

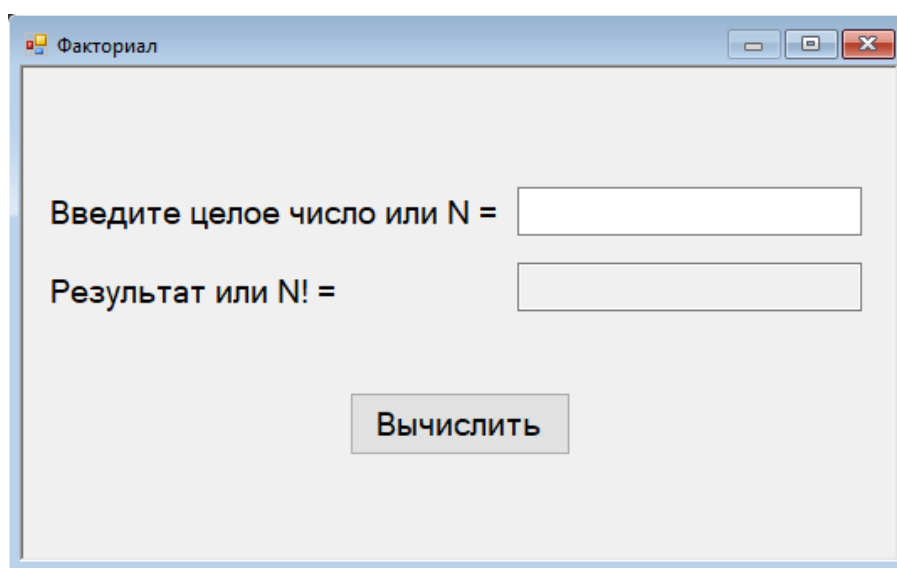


Рисунок 1 – Вид формы в конструкторе

### 1.3 Таблица с описанием переименованных элементов формы

Все элементы формы были переименованы и их атрибуты изменены. Проведенные изменения представлены в таблице 1

Таблица 1 – Значения атрибутов элементов в приложении «Вычисление факториала»

Описание элементов формы	Список измененных атрибутов	Новое значение атрибута
Форма	Text	Факториал
Первая надпись (label)	Name	lblInput
Первая надпись (label)	Text	Введите целое число или N=
Вторая надпись (label)	Name	lblOutput
Вторая надпись (label)	Text	Результат или N!=
Первое текстовое поле (textBox)	Name	txtInput
Второе текстовое поле (textBox)	Name	txtOutput
Второе текстовое поле (textBox)	ReadOnly	True
Кнопка (button)	Name	btnCalculate
Кнопка (button)	Text	Вычислить
Обработчик ошибок (errorProvider)	Name	errPr

### 1.4 Примеры работы

При запуске приложения на экране появляется окно как показано на рис.2.

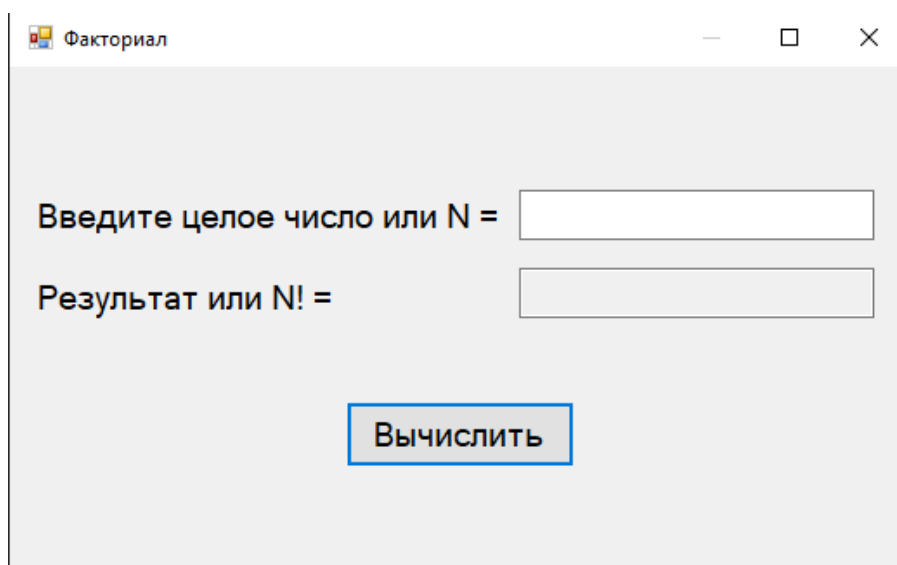


Рисунок 2 – Запуск приложения



При запуске с корректными данными как показано на рис.3

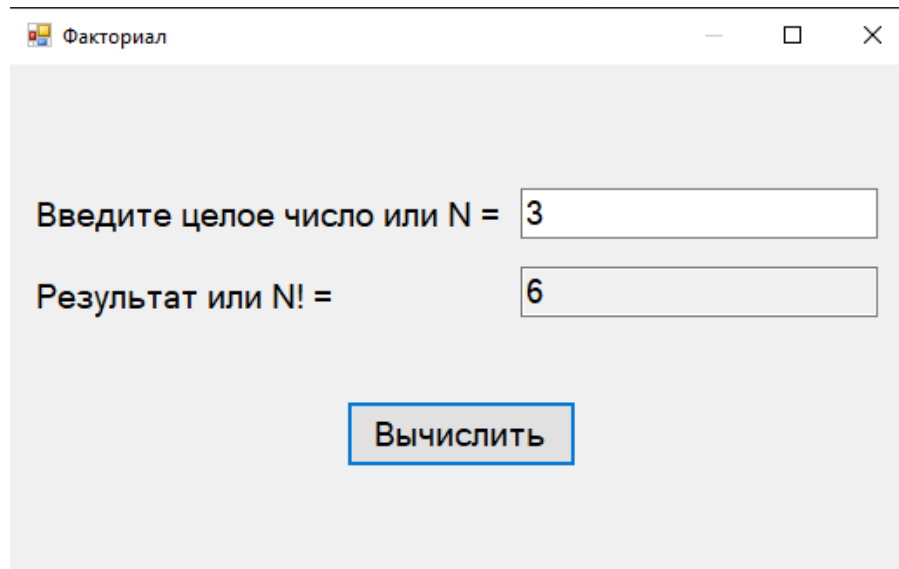


Рисунок 3 – Запуск с корректными данными

При запуске с некорректными данными как показано на рис.4

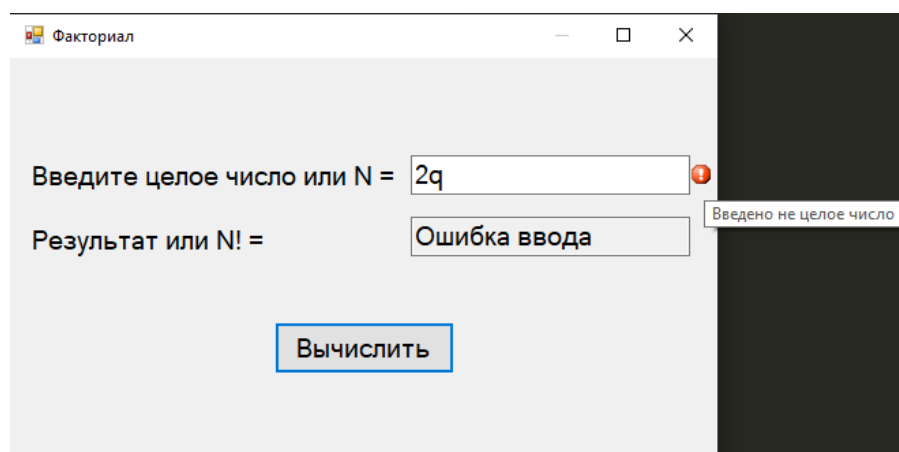


Рисунок 4 – Запуск с некорректными данными

## 1.5 Примеры кода

Была написана функция вычисления факториала:

```
1 // нахождение факториала
2 long long fact(long long N) {
3     if (N < 0) { // отрицательное число
4         return -1;
5     }
6     else if (N == 0 || N == 1) {
7         return 1;
```

```
8     }  
9     else {  
10        return N * fact(N - 1); // n! = n * (n - 1)!  
11    }  
12 }
```

Другие фрагменты кода расположены в приложении А. Полный код программы приведен в приложении К

## 2 Простые вычисления (Задание 2 вариант 7)

### 2.1 Условие задания

Вычислить:  $\frac{y^2 \sin(x^2)}{x+y^2}$

Приложение должно содержать следующие компоненты:

1. Заголовок формы должен отражать суть задания.
2. Все элементы формы должны быть внятно подписаны (кнопки подписаны, у тестового поля должно быть написано, для чего оно нужно и т. д.)
3. В коде должны быть комментарии и отступы (код должен быть легко читаем).
4. Должна быть проверка ошибок — ввод не числа, ввод числа, находящегося за пределами ОДЗ, ввод числа, принадлежащего ОДЗ.
5. Если надо ввести 2 значения, то в случае ввод букв в оба поля, ошибка должна быть у обоих полей; в случае ввода одной буквы — только у того поля, где буква.

### 2.2 Вид формы в конструкторе

Форма имеет вид как показано на рис.5:

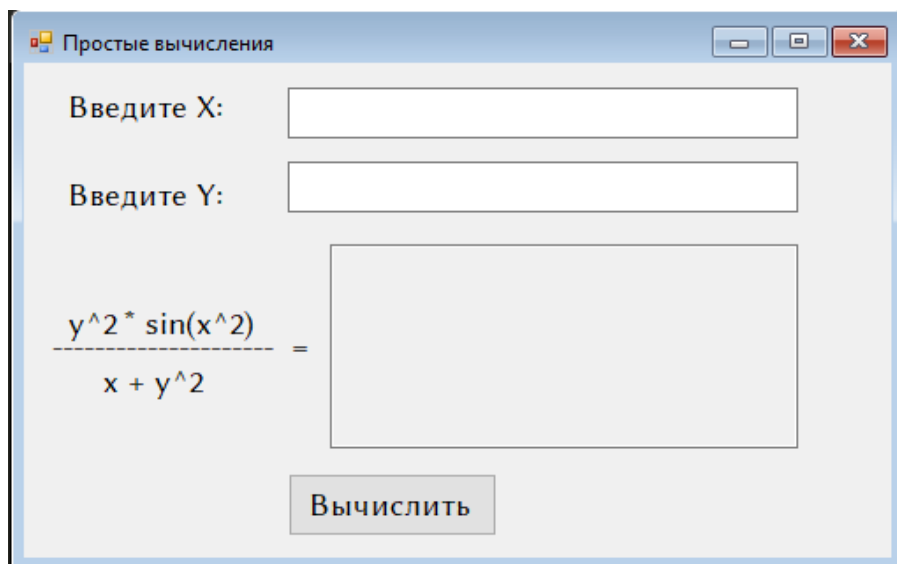


Рисунок 5 – Вид формы в конструкторе

### 2.3 Таблица с описанием переименованных элементов формы

Все элементы формы были переименованы и их атрибуты изменены. Проведенные изменения представлены в таблице 2



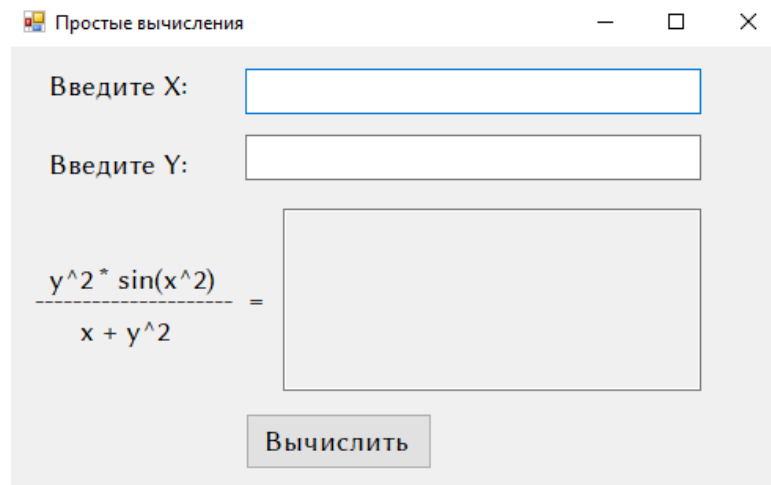


Рисунок 6 – Запуск приложения

При запуске с корректными данными как показано на рис.7

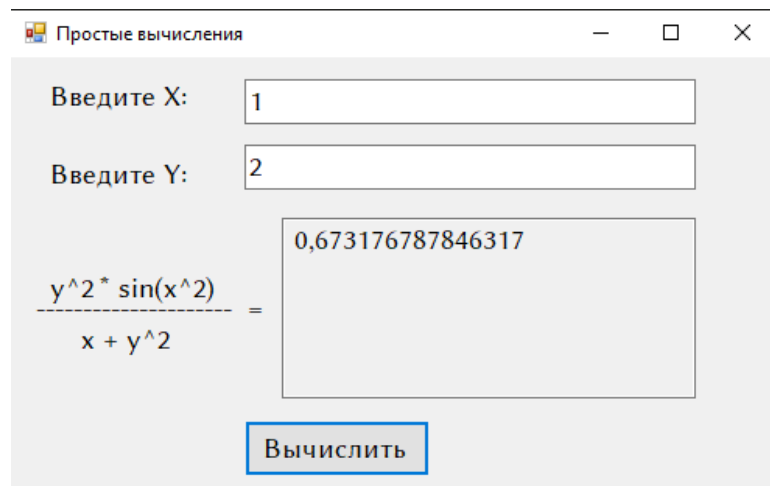


Рисунок 7 – Запуск с корректными данными

При запуске с некорректными данными как показано на рис.8

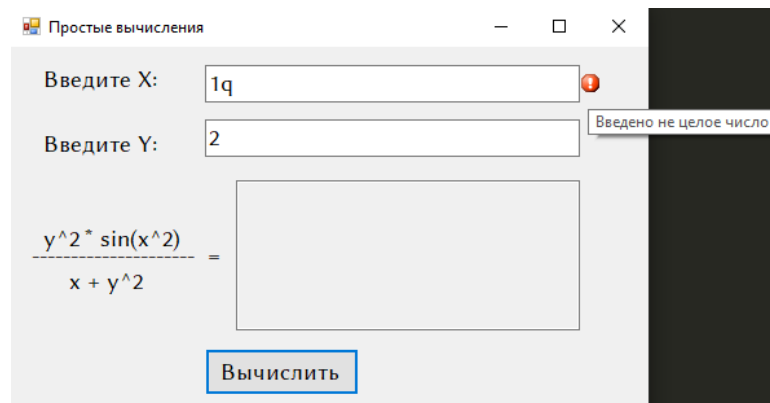


Рисунок 8 – Запуск с некорректными данными

## 2.5 Примеры кода

Функция подсчета формулы с введенными значениями переменных:

```
1 double solve(long long x, long long y) {  
2     return (y * y * sin(x * x) * 1.0) / (x + y*y) ;  
3 }
```

Функция очистки полей:

```
1 private: void ClearAll() { // очистка полей  
2     this->txtOut->Text = "";  
3     errPrX->SetError(txtInX, String::Empty);  
4     errPrY->SetError(txtInY, String::Empty);  
5 }
```

Другие фрагменты кода расположены в приложении Б. Полный код программы приведен в приложении К

### 3 Рекурсивные вычисления (Задание 3 вариант 7)

#### 3.1 Условие задания

Создать рекурсивную функцию, которая для заданного целого  $N$  и вещественного  $X$ , определяет  $X^N$  по следующей формуле: [4]

$$\begin{cases} 1, & \text{при } N = 0 \\ X \times X^{N-1}, & \text{при } N > 0 \\ \frac{1}{X^{|N|}}, & \text{при } N < 0 \end{cases}$$

Приложение должно содержать следующие компоненты:

1. Заголовок формы должен отражать суть задания.
2. Все элементы формы должны быть внятно подписаны (кнопки подписаны, у тестового поля должно быть написано, для чего оно нужно и т. д.)
3. В коде должны быть комментарии и отступы (код должен быть легко читаем).
4. В коде программы все элементы формы должны быть переименованы (btnName — для кнопок, lblName — для ссылок, txtName — для текстового поля и т. д.) Наименования должны быть понятными.
5. Приложение должно корректно работать (выводить ответ или ошибку с соответствующим сообщением) для следующих данных: ввод буквы, ввод отрицательного числа, ввод нуля, ввод положительного числа ( $< 10$ ), ввод большого положительного числа. После вывода ошибок при вводе корректных данных поля ошибок должны очищаться.

#### 3.2 Вид формы в конструкторе

Форма имеет вид как показано на рис.9:

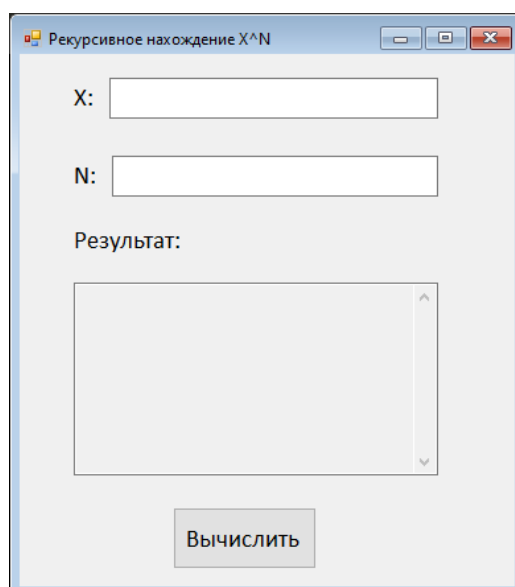


Рисунок 9 – Вид формы в конструкторе

### 3.3 Таблица с описанием переименованных элементов формы

Все элементы формы были переименованы и их атрибуты изменены. Проведенные изменения представлены в таблице 3

Таблица 3 – Значения атрибутов элементов в приложении «Рекурсивные вычисления»

Описание элементов формы	Список измененных атрибутов	Новое значение атрибута
Форма	Text	Рекурсивные вычисления $X^N$
Первая надпись (label)	Name	lblX
Первая надпись (label)	Text	X:
Вторая надпись (label)	Name	lblN
Вторая надпись (label)	Text	N:
Третья надпись (label)	Name	lblResult
Третья надпись (label)	Text	Результат:
Первое текстовое поле (textBox)	Name	txtInX
Второе текстовое поле (textBox)	Name	txtInY
Третье текстовое поле (textBox)	Name	txtOut
Третье текстовое поле (textBox)	ReadOnly	True
Кнопка (button)	Name	btnStart
Кнопка (button)	Text	Вычислить



Обработчик ошибок 1 (errorProvider)	Name	errPrX
Обработчик ошибок 2 (errorProvider)	Name	errPrN

### 3.4 Примеры работы

При запуске приложения на экране появляется окно как показано на рис.10.

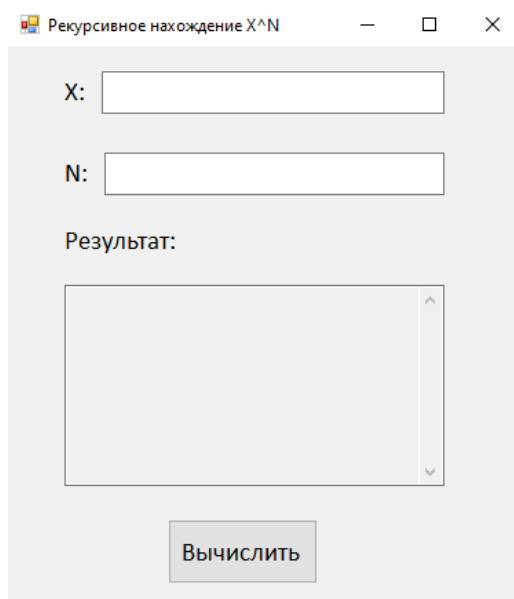


Рисунок 10 – Запуск приложения

При запуске с корректными данными как показано на рис.11

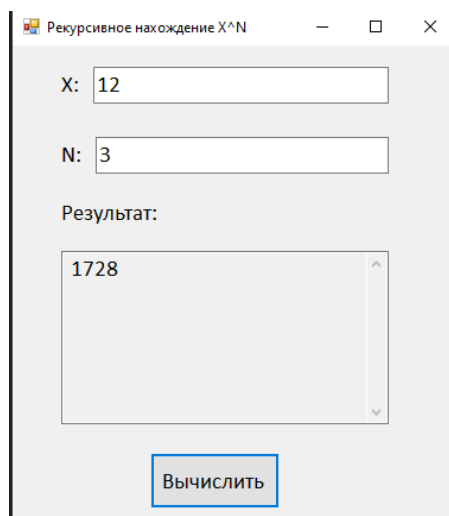


Рисунок 11 – Запуск с корректными данными

При запуске с некорректными данными как показано на рис.12

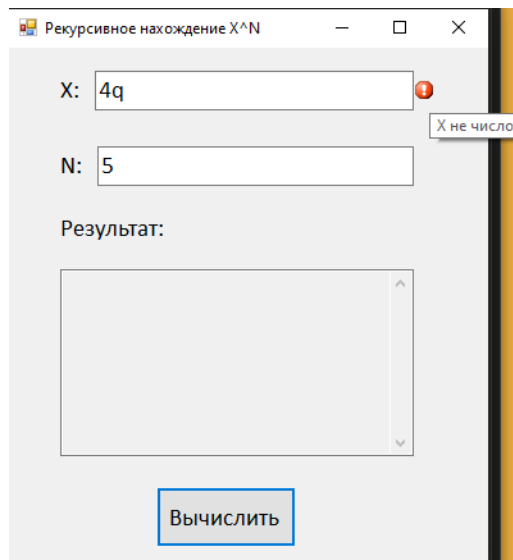


Рисунок 12 – Запуск с некорректными данными

### 3.5 Примеры кода

Функция рекурсивно вычисляющая  $x^n$ :

```

1 // Функция рекурсивно вычисляющая  $x^n$ 
2 double recursion(double x, long long n) {
3     // если  $n == 0$ 
4     if (n == 0) {
5         // то возвращаем 1
6         return 1;
7     }
8     // если  $n > 0$ 
9     else if (n > 0) {
10        // то возвращаем  $x * x^{(n-1)}$ 
11        return x * recursion(x, n - 1);
12    }
13    // если  $n < 0$ 
14    else {
15        // то возвращаем  $1 / (x^{|n|})$ 
16        return 1 / recursion(x, abs(n));
17    }
18 }
```

Другие фрагменты кода расположены в приложении В. Полный код программы приведен в приложении К

## 4 Обработка табличных данных. Часть 1 (Задание 4 вариант 7)

### 4.1 Условие задания

Найти среднее арифметическое нечетных элементов, не попадающих в заданный интервал. Вывести номера максимальных нечетных элементов.

Создать приложение для выполнения задания. Использовать элемент формы DataGridView [5].

ДИАПАЗОН  $[a, b]$  означает, что  $mas[i][j] \geq a \ \&\& \ mas[i][j] \leq b$ .

Приложение должно выполнять следующие действия:

1. Возможность удалять и добавлять строки таблицы. Проект не должен аварийно завершаться при удалении несуществующей таблицы.
2. Проверять ввод не числовых данных как в таблицу, так и в остальные текстовые поля (если есть в задании).
3. Если есть диапазон значений  $[a, b]$ , проверять, что  $a < b$ .
4. Заголовок формы должен отражать суть задания.
5. Все элементы формы должны быть внятно подписаны (кнопки подписаны, у тестового поля должно быть написано, для чего оно нужно и т. д.)
6. В коде должны быть комментарии и отступы (код должен быть легко читаем).
7. В коде программы все элементы формы должны быть переименованы (btnName — для кнопок, lblName — для ссылок, txtName — для текстового поля и т. д.) Наименования должны быть понятными.
8. Приложение должно корректно работать (выводить ответ или ошибку с соответствующим сообщением). После вывода ошибок при вводе корректных данных поля ошибок должны очищаться.

### 4.2 Вид формы в конструкторе

Форма имеет вид как показано на рис.13:

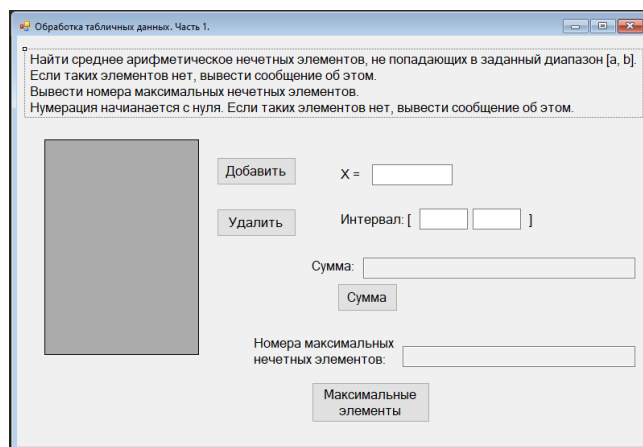


Рисунок 13 – Вид формы в конструкторе

### 4.3 Таблица с описанием переименованных элементов формы

Все элементы формы были переименованы и их атрибуты изменены. Проведенные изменения представлены в таблице 4

Таблица 4 – Значения атрибутов элементов в приложении «Обработка табличных данных. Часть 1»

Описание элементов формы	Список измененных атрибутов	Новое значение атрибута
Форма	Text	Обработка табличных данных 1
Первая надпись (label)	Name	lblTask

Первая надпись (label)	Text	<p>Найти среднее арифметическое нечетных элементов, не попадающих в заданный диапазон [a, b].</p> <p>Если таких элементов нет, вывести сообщение об этом. Вывести номера максимальных нечетных элементов. Нумерация начинаается с нуля. Если таких элементов нет, вывести сообщение об этом.</p>
Вторая надпись (label)	Name	lblX
Вторая надпись (label)	Text	X =
Третья надпись (label)	Name	lblInterval1
Третья надпись (label)	Text	Интервал: [
Четвёртая надпись (label)	Name	lblInterval2
Четвёртая надпись (label)	Text	]
Пятая надпись (label)	Name	lblSum
Пятая надпись (label)	Text	Сумма:
Шестая надпись (label)	Name	lblMaxEl
Шестая надпись (label)	Text	Номера максимальных нечетных элементов
Первое текстовое поле (textBox)	Name	XTxtBox
Второе текстовое поле (textBox)	Name	ATxtBox

Третье текстовое поле (textBox)	Name	BTxtBox
Четвёртое текстовое поле (textBox)	Name	SumTxtBox
Четвёртое текстовое поле (textBox)	ReadOnly	True
Пятое текстовое поле (textBox)	Name	MaxTxtBox
Пятое текстовое поле (textBox)	ReadOnly	True
Первая кнопка (button)	Name	AddBtn
Первая кнопка (button)	Text	Добавить
Вторая кнопка (button)	Name	DelBtn
Вторая кнопка (button)	Text	Удалить
Третья кнопка (button)	Name	SumBtn
Третья кнопка (button)	Text	Сумма:
Четвёртая кнопка (button)	Name	MaxBtn
Четвёртая кнопка (button)	Text	Максимальные элементы
Таблица (dataGridView)	Name	ArDtGr
Обработчик ошибок 1 (errorProvider)	Name	errPrX
Обработчик ошибок 2 (errorProvider)	Name	errPrA
Обработчик ошибок 3 (errorProvider)	Name	errPrB

#### 4.4 Примеры работы

При запуске приложения на экране появляется окно как показано на рис.14.

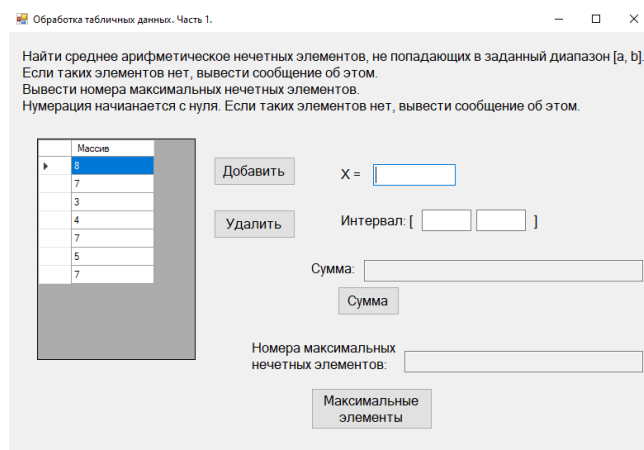


Рисунок 14 – Запуск приложения

При запуске с корректными данными, при нажатии на кнопку добавить происходит как показано на рис.15:

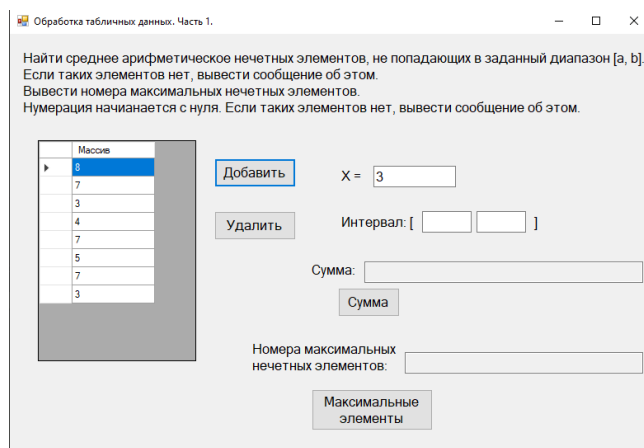


Рисунок 15 – Запуск с корректными данными

При запуске с некорректными данными, при нажатии на кнопку добавить происходит как показано на рис.16:

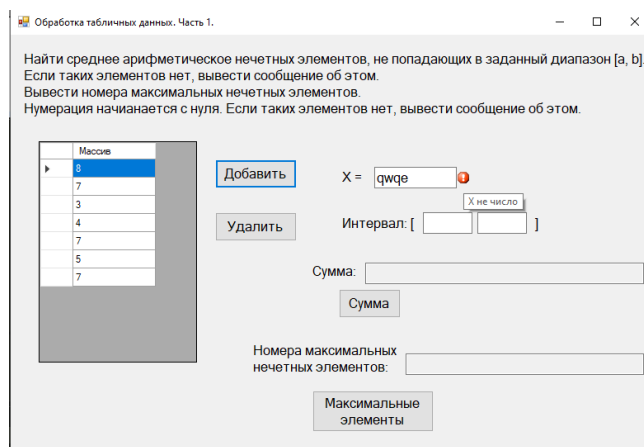


Рисунок 16 – Запуск с некорректными данными

## 4.5 Примеры кода

Функция вставки строки:

```
1  // вставка строки
2  private: System::Void AddBtn_Click(System::Object^ sender, System::EventArgs^ e) {
3      ClearAll();
4      long long InputX;
5      // переводим строку из TextBox в число
6      bool parseX = Int64::TryParse(this->XTxtBox->Text, InputX);
7      // ввели не число
8      if (!parseX) {
9          errPrX->SetError(XTxtBox, "X не число");
10     }
11     else {
12         ArDtGr->Rows->Add();
13         ArDtGr->Rows[ArDtGr->RowCount - 1]->Cells[0]->Value = InputX;
14     }
15 }
```

Другие фрагменты кода расположены в приложении Г. Полный код программы приведен в приложении К



## 5 Обработка табличных данных. Часть 2 (Задание 5 вариант 13)

### 5.1 Условие задания

Поменять местами нулевую строку и строку, сумма элементов которой максимальна (сумму элементов нулевой строки не учитывать).

Разработать приложение в соответствии со своим вариантом.

Проверить работу приложения на приведенных тестовых примерах.

ЗАДАНИЕ ПРЕДПОЛАГАЕТ НАЛИЧИЕ ДВУХ ТАБЛИЦ. В первую вводятся данные (должна быть проверка, что введены цифры). Вторую лучше сделать доступной только для чтения. В нее записывается результат.

ДИАПАЗОН  $[a, b]$  означает, что  $mas[i][j] \geq a \ \&\& \ mas[i][j] \leq b$ .

Приложение должно содержать следующие компоненты:

1. Заголовок формы должен отражать суть задания.
2. Все элементы формы должны быть внятно подписаны (кнопки подписаны, у текстового поля должно быть написано, для чего оно нужно и т. д.)
3. В коде должны быть комментарии и отступы (код должен быть легко читаем).
4. Таблица может быть задана двумя способами:
  - либо ввести количество строк и столбцов (тогда необходима проверка, что введено не число) и создать нужную таблицу. При изменении количества строк и столбцов, старая таблица должна быть удалена.
  - либо добавлять и удалять строки и столбцы с помощью отдельных кнопок. Проследить, чтобы приложение не завершалось аварийно (не удалять нулевую строку).
5. Должна быть проверка ошибок — ввод не числа, ввод числа, приводящего к переполнению стека или выхода результата за границы диапазона типа. В таблице также должны быть введены целые числа. В случае ошибочного ввода поля результатов должны автоматически очищаться.
6. Если данные введены корректно, но отсутствуют необходимые данные (например, надо найти сумму нечетных чисел, а в таблице только четные), то в поле результата должно быть выведено об этом сообщение.
7. В случае задач с вводом диапазона  $[a, b]$  необходима обязательная проверка, что  $a < b$ .

## 5.2 Вид формы в конструкторе

Форма имеет вид как показано на рис.17:

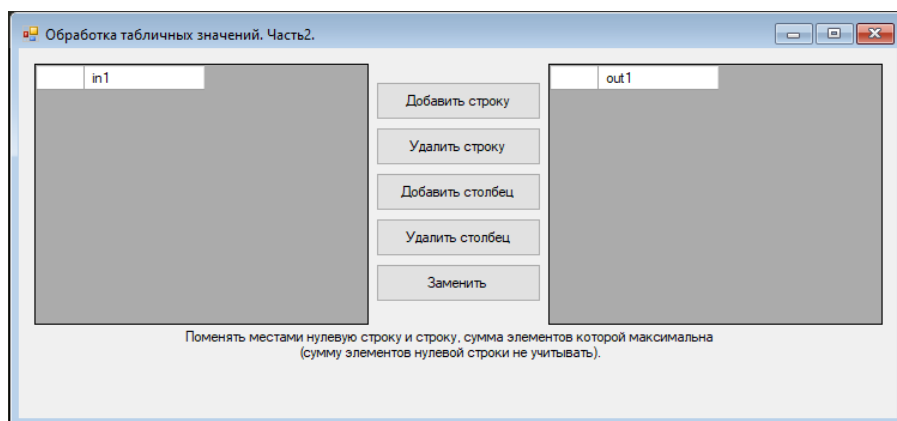


Рисунок 17 – Вид формы в конструкторе

## 5.3 Таблица с описанием переименованных элементов формы

Все элементы формы были переименованы и их атрибуты изменены. Проведенные изменения представлены в таблице 5

Таблица 5 – Значения атрибутов элементов в приложении «Обработка табличных данных. Часть 2»

Описание элементов формы	Список измененных атрибутов	Новое значение атрибута
Форма	Text	Обработка табличных данных 2
Первая надпись (label)	Name	taskDescription
Первая надпись (label)	Text	Поменять местами нулевую строку и строку, сумма которой максимальна (сумму элементов нулевой строки не учитывать)
Первая кнопка (button)	Name	btnAddRow
Первая кнопка (button)	Text	Добавить строку
Вторая кнопка (button)	Name	btnRemoveRow

Вторая кнопка (button)	Text	Удалить строку
Третья кнопка (button)	Name	btnAddColumn
Третья кнопка (button)	Text	Добавить к столбец
Четвёртая кнопка (button)	Name	btnRemoveColumn
Четвёртая кнопка (button)	Text	Удалить столбец
Пятая кнопка (button)	Name	btnStart
Пятая кнопка (button)	Text	Заменить
Первая таблица (dataGridView)	Name	dataGridInput
Вторая таблица (dataGridView)	Name	dataGridOutput
Обработчик ошибок 1 (errorProvider)	Name	erZeroRow
Обработчик ошибок 2 (errorProvider)	Name	erZeroColumn
Обработчик ошибок 3 (errorProvider)	Name	erChanges

## 5.4 Примеры работы

При запуске приложения на экране появляется окно как показано на рис.18.

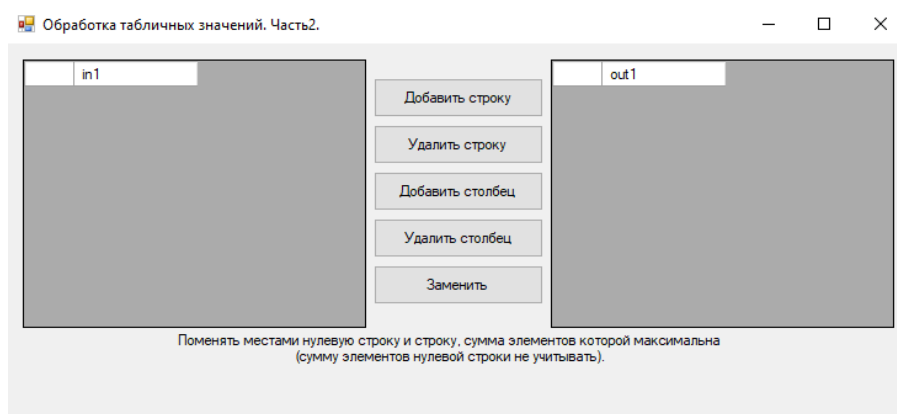


Рисунок 18 – Запуск приложения

При запуске с корректными данными, при нажатии на кнопку заменить происходит как показано на рис.19:

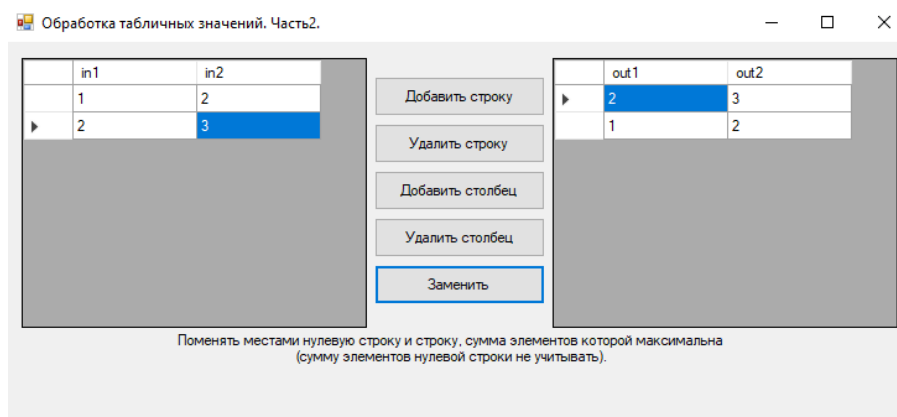


Рисунок 19 – Запуск с корректными данными

При запуске с некорректными данными, при нажатии на кнопку заменить происходит как показано на рис.20:

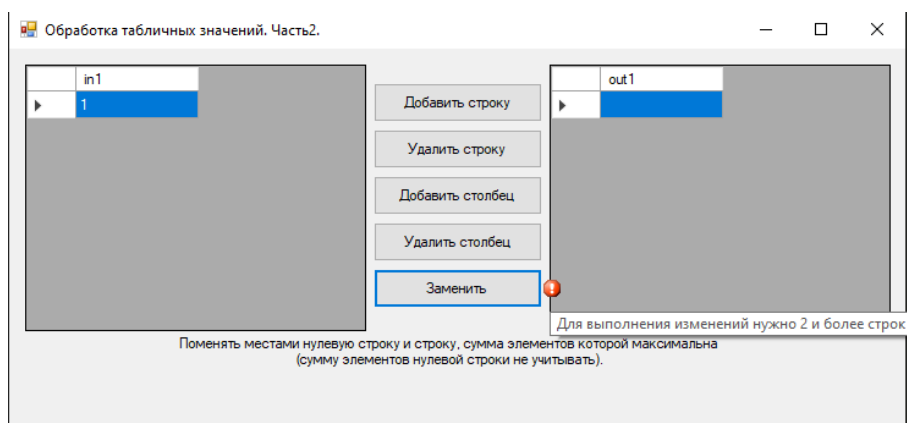


Рисунок 20 – Запуск с некорректными данными

## 5.5 Примеры кода

Функция добавления строки в таблицу:

```

1 // Добавление строки в таблицу
2 private: System::Void btnAddRow_Click(System::Object^ sender, System::EventArgs^ e) {
3     ClearAll();
4     dataGridInput->Rows->Add(1);
5     dataGridOutput->Rows->Add(1);
6     // Увеличиваем счетчик, если добавили строку
7     countRow++;
8 }

```

Функция удаления строки из таблицы:

```

1  // Удаление строки
2  private: System::Void btnRemoveRow_Click(System::Object^ sender, System::EventArgs^
    ↪ e) {
3      ClearAll();
4      if (dataGridInput->CurrentRow == nullptr) {
5          this->erZeroRow->SetError(btnRemoveRow, "Нельзя удалить не существующую
    ↪ строку");
6          return;
7      }
8      if (!dataGridInput->CurrentRow->IsNewRow) {
9          int input = dataGridInput->CurrentRow->Index;
10         int output = dataGridOutput->CurrentRow->Index;
11         dataGridInput->Rows->Remove(dataGridInput->Rows[input]);
12         dataGridOutput->Rows->Remove(dataGridOutput->Rows[output]);
13     }
14 }

```

Другие фрагменты кода расположены в приложении Д. Полный код программы приведен в приложении К

## 6 Матричный калькулятор (Задание 6)

### 6.1 Условие задания

Создать приложение, реализующее основные операции с векторами и матрицами:

1. Ввод матрицы, вектора
2. Создание матриц (единичная, матрица как набор векторов)
3. Умножение на число, вектор, матрицу [6]
4. Сложение/вычитание двух матриц
5. Сложение/вычитание двух векторов
6. Скалярное и векторное произведение двух векторов
7. Транспонированная матрица
8. Определитель, ранг матрицы

### 6.2 Вид формы в конструкторе

Форма имеет вид как показано на рис.21:

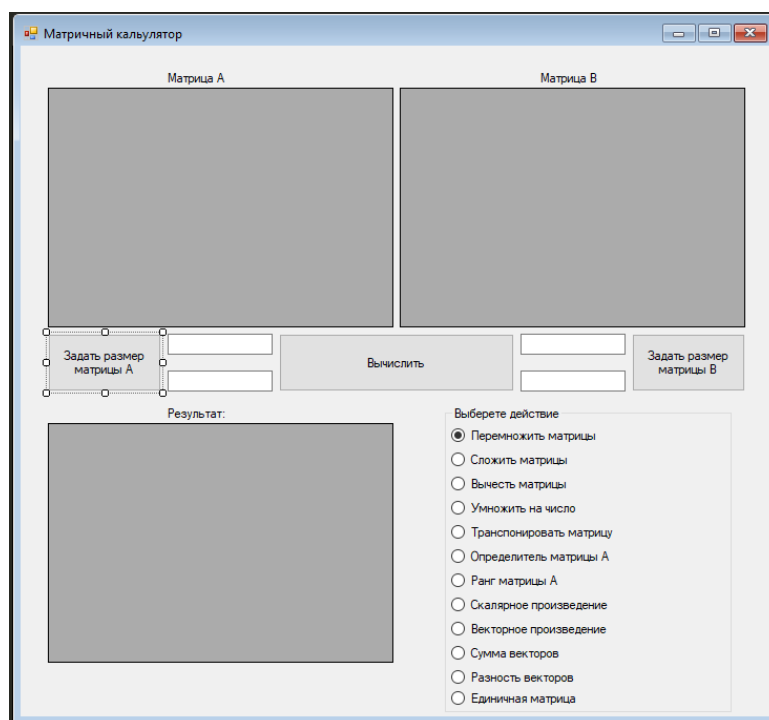


Рисунок 21 – Вид формы в конструкторе

### 6.3 Таблица с описанием переименованных элементов формы

Для написания данного приложения дополнительно использовались элементы формы:

- GroupBox [7]
- RadioButton [8]

Все элементы формы были переименованы и их атрибуты изменены. Проведенные изменения представлены в таблице 6

Таблица 6 – Значения атрибутов элементов в приложении «Матричный калькулятор»

<b>Описание элементов формы</b>	<b>Список измененных атрибутов</b>	<b>Новое значение атрибута</b>
Форма	Text	Матричный калькулятор
Первая надпись (label)	Name	labelA
Первая надпись (label)	Text	Матрица A
Вторая надпись (label)	Name	labelB
Вторая надпись (label)	Text	Матрица B
Третья надпись (label)	Name	labelRes
Третья надпись (label)	Text	Результат:
Первое текстовое поле (textBox)	Name	tbRowAInput
Второе текстовое поле (textBox)	Name	tbColumnAInput
Третье текстовое поле (textBox)	Name	tbRowBInput
Четвёртое текстовое поле (textBox)	Name	tbColumnBInput
Первая кнопка (button)	Name	btnSizeA
Первая кнопка (button)	Text	Задать размер матрицы A
Вторая кнопка (button)	Name	btnSizeB
Вторая кнопка (button)	Text	Задать размер матрицы B
Третья кнопка (button)	Name	btnResult
Третья кнопка (button)	Text	Вычислить
Первая таблица (dataGridView)	Name	matrixA

Вторая таблица (dataGridView)	Name	matrixB
Третья таблица (dataGridView)	Name	matrixResult
Кнопка выбора 1 (radioButton)	Name	rBtnMultMatr
Кнопка выбора 1 (radioButton)	Text	Перемножить матрицы
Кнопка выбора 2 (radioButton)	Name	rBtnSum
Кнопка выбора 2 (radioButton)	Text	Сложить матрицы
Кнопка выбора 3 (radioButton)	Name	rBtnSubstract
Кнопка выбора 3 (radioButton)	Text	Вычесть матрицы
Кнопка выбора 4 (radioButton)	Name	rBtnMultNum
Кнопка выбора 4 (radioButton)	Text	Умножить на число
Кнопка выбора 5 (radioButton)	Name	rBtnTransposition
Кнопка выбора 5 (radioButton)	Text	Транспонировать матрицу
Кнопка выбора 6 (radioButton)	Name	rBtnDetermA
Кнопка выбора 6 (radioButton)	Text	Определитель матрицы A
Кнопка выбора 7 (radioButton)	Name	rBtnRankA
Кнопка выбора 7 (radioButton)	Text	Ранг матрицы A
Кнопка выбора 8 (radioButton)	Name	rBtnScalar



Кнопка выбора 8 (radioButton)	Text	Скалярное произведение
Кнопка выбора 9 (radioButton)	Name	rBtnVector
Кнопка выбора 9 (radioButton)	Text	Векторное произведение
Кнопка выбора 10 (radioButton)	Name	rBtnSumVec
Кнопка выбора 10 (radioButton)	Text	Сумма векторов
Кнопка выбора 11 (radioButton)	Name	rBtnSubVec
Кнопка выбора 11 (radioButton)	Text	Разность векторов
Кнопка выбора 12 (radioButton)	Name	rBtnUnitMatrix
Кнопка выбора 12 (radioButton)	Text	Единичная матрица
groupBox	Name	grChoose
groupBox	Text	Выбор действия
Обработчик ошибок 1 (errorProvider)	Name	erPrSizeA
Обработчик ошибок 2 (errorProvider)	Name	erPrSizeB
Обработчик ошибок 3 (errorProvider)	Name	erPrResult

## 6.4 Примеры работы

При запуске приложения на экране появляется окно как показано на рис.22.

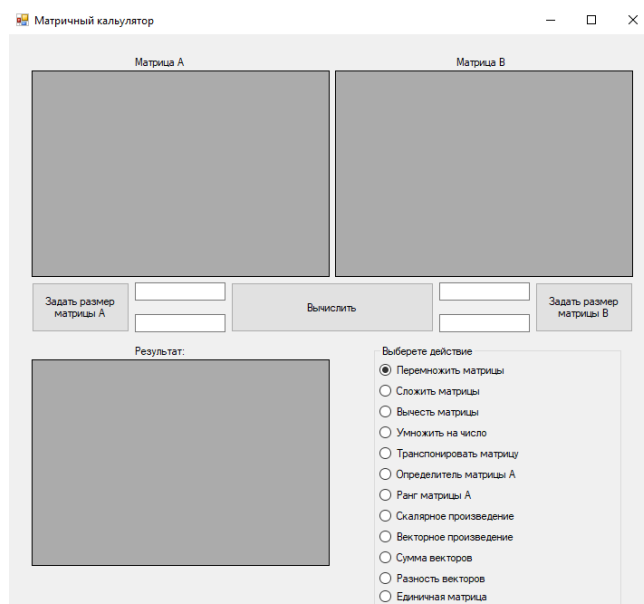


Рисунок 22 – Запуск приложения

При запуске с корректными данными, при нажатии на кнопку «Вычислить» с выбранным действием «Сложить матрицы» происходит как показано на рис.23:

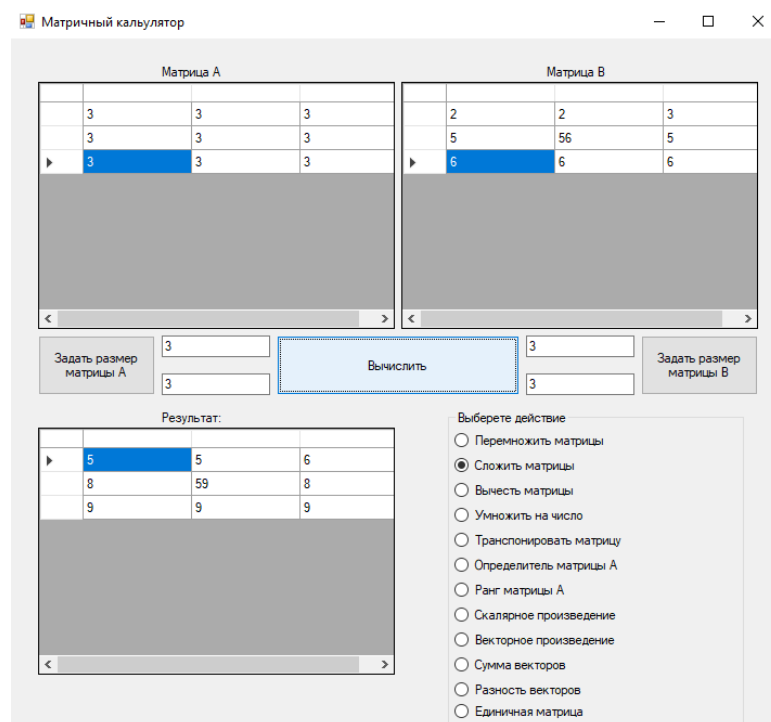


Рисунок 23 – Запуск с корректными данными

При запуске с некорректными данными, при нажатии на кнопку «Вычислить» с выбранным действием «Сложить матрицы» происходит как показано на рис.24:

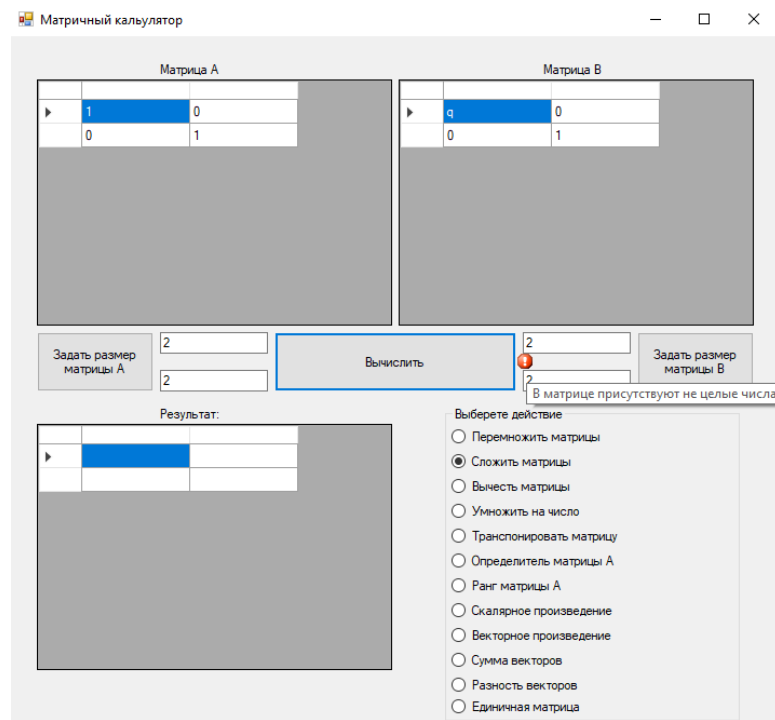


Рисунок 24 – Запуск с некорректными данными

## 6.5 Примеры кода

Функция умножения матрицы  $A$  на число  $num$

```

1 // умножение матрицы A на число num
2 void MultNum(int row, int column, int num) {
3     int a;
4     bool check;
5     for (int i = 0; i < row; i++) {
6         for (int j = 0; j < column; j++) {
7             // Проверка на не целое число
8             check =
                ↳ Int32::TryParse(System::Convert::ToString(matrixA->Rows[i]->Cells[j]->Value),
                ↳ a);
9             if (!check) {
10                 throw gcnew FormatException("В матрице присутствуют не целые числа");
11             }
12             // Умножение на число
13             matrixResult->Rows[i]->Cells[j]->Value = a * num;
14         }
15     }
16 }

```

Другие фрагменты кода расположены в приложении Е. Полный код про-

граммы приведен в приложении К

## **7 Использование коллекций (Задание 7 вариант 8)**

### **7.1 Условие задания**

Создать очередь, состоящую из целых чисел. Предусмотреть возможность создания очереди из набора чисел, добавления одного элемента (функция push), удаления одного элемента (функция pop), вывода результата на экран, удаления всех элементов с помощью кнопок. Найти сумму четных элементов, попадающих в заданный интервал  $[a, b]$ . Получить новую очередь, вставив после максимальных элементов новый элемент.

Приложение должно содержать следующие пункты:

1. Заголовок формы должен отражать суть задания.
2. Все элементы формы должны быть внятно подписаны (кнопки подписаны, у тестового поля должно быть написано, для чего оно нужно и т. д.)
3. В коде должны быть комментарии и отступы (код должен быть легко читаем).
4. В коде программы все элементы формы должны быть переименованы (btnName — для кнопок, lblName — для ссылок, txtName — для текстового поля и т. д.) Наименования должны быть понятными.
5. Должна быть возможность для ввода и вывода первоначальных данных.
6. Должна быть возможность для вставки и удаления одного элемента.
7. Должны использоваться коллекции.
8. Ответы на задания должны быть в разных полях.
9. Если нет данных для выполнения задания, выводить соответствующие данные.
10. Для графов использовать список смежности.

### **7.2 Вид формы в конструкторе**

Форма имеет вид как показано на рис.25:

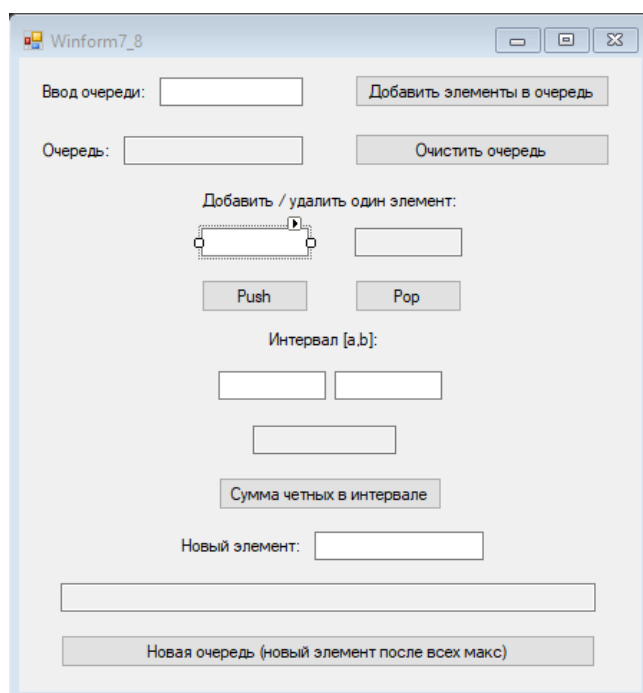


Рисунок 25 – Вид формы в конструкторе

### 7.3 Таблица с описанием переименованных элементов формы

Все элементы формы были переименованы и их атрибуты изменены. Проведенные изменения представлены в таблице 7

Таблица 7 – Значения атрибутов элементов в приложении «Использование коллекций»

Описание элементов формы	Список измененных атрибутов	Новое значение атрибута
Форма	Text	Использование коллекций
Первая надпись (label)	Name	lbl1
Первая надпись (label)	Text	Ввод очереди:
Вторая надпись (label)	Name	lbl2
Вторая надпись (label)	Text	Очередь:
Третья надпись (label)	Name	lbl3
Третья надпись (label)	Text	Добавить/удалить один элемент:
Четвёртая надпись (label)	Name	lbl4
Четвёртая надпись (label)	Text	Интервал [a, b]
Пятая надпись (label)	Name	lbl5

Пятая надпись (label)	Text	Новый элемент:
Первое текстовое поле (textBox)	Name	tBInputQue
Второе текстовое поле (textBox)	Name	tBOutputQue
Второе текстовое поле (textBox)	ReadOnly	True
Третье текстовое поле (textBox)	Name	tBInputPush
Четвёртое текстовое поле (textBox)	Name	tBOutputPop
Четвёртое текстовое поле (textBox)	ReadOnly	True
Пятое текстовое поле (textBox)	Name	tBInputA
Шестое текстовое поле (textBox)	Name	tBInputB
Седьмое текстовое поле (textBox)	Name	tBOutputSum
Седьмое текстовое поле (textBox)	ReadOnly	True
Восьмое текстовое поле (textBox)	Name	tBInputNAM
Девятое текстовое поле (textBox)	Name	tBOutputNAM
Девятое текстовое поле (textBox)	ReadOnly	True
Первая кнопка (button)	Name	btnInputQue
Первая кнопка (button)	Text	Добавить элементы в очередь
Вторая кнопка (button)	Name	btnClearQue
Вторая кнопка (button)	Text	Очистить очередь
Третья кнопка (button)	Name	btnPush

Третья кнопка (button)	Text	Push
Четвёртая кнопка (button)	Name	btnPop
Четвёртая кнопка (button)	Text	Pop
Пятая кнопка (button)	Name	btnSum
Пятая кнопка (button)	Text	Сумма четных в интервале
Шестая кнопка (button)	Name	btnNewAfterMax
Шестая кнопка (button)	Text	Новая очередь (новый элемент после всех макс)
Обработчик ошибок (errorProvider)	Name	eP1

## 7.4 Примеры работы

При запуске приложения на экране появляется окно как показано на рис.26.

Рисунок 26 – Запуск приложения

При запуске с корректными данными, при нажатии на кнопку «Добавить элементы в очередь» происходит как показано на рис.27:



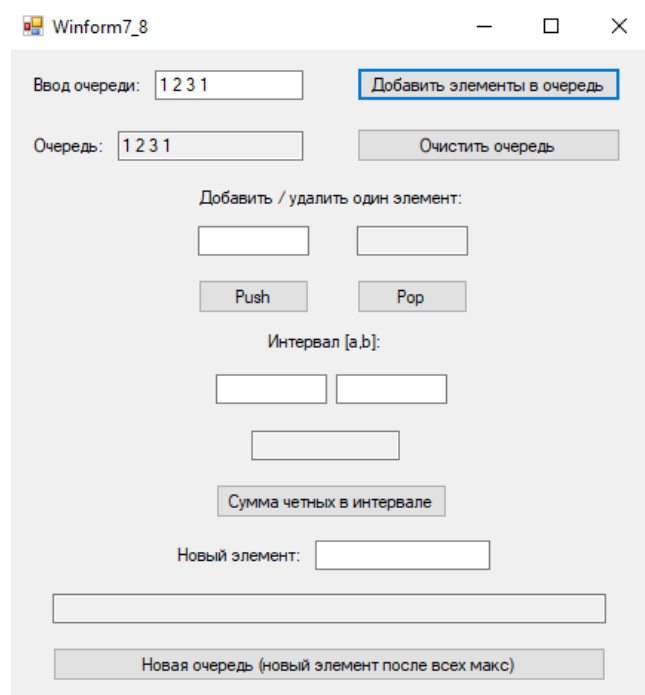


Рисунок 27 – Запуск с корректными данными

При запуске с некорректными данными, при нажатии на кнопку «Добавить элементы в очередь» происходит как показано на рис.28:

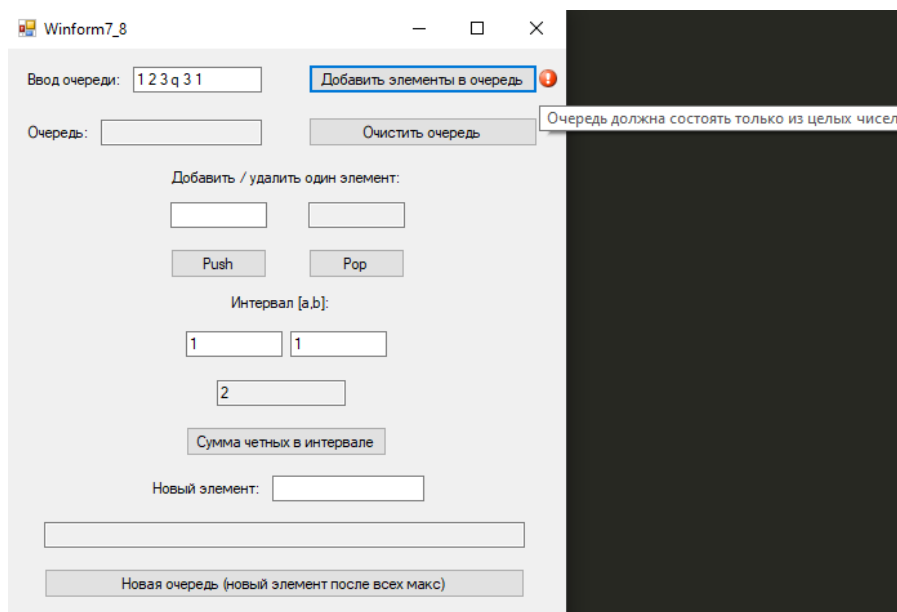


Рисунок 28 – Запуск с некорректными данными

## 7.5 Примеры кода

Функция добавления одного элемента в очередь:

```

1 private: System::Void btnPush_Click(System::Object^ sender, System::EventArgs^ e) {
2     ClearEP();
3     // Вспомогательная очередь
4     System::Collections::Generic::Queue<int> buffer;
5     // Считываем число
6     int number;
7     bool res = Int32::TryParse(tBInputPush->Text, number);
8     String^ str = "";
9     // Проверка на число
10    if (!res) {
11        this->eP1->SetError(btnPush, "Не целое число");
12        buffer.Clear();
13        return;
14    }
15    q.Enqueue(number);
16    QueueOutput();
17 }

```

Другие фрагменты кода расположены в приложении Ж. Полный код программы приведен в приложении К

## **8 Работа с файлами (Задание 8 вариант 1)**

### **8.1 Условие задания**

Создать таблицу Student. В другой файл вывести студентов, у которых сдана сессия. (Вариант 1)

Приложение должно содержать следующие пункты:

1. Заголовок формы должен отражать суть задания.
2. Все элементы формы должны быть внятно подписаны (кнопки подписаны, у тестового поля должно быть написано, для чего оно нужно и т.д.)
3. Структура представляет собой таблицу DataGridView, в ячейках которой записаны данные разных типов.
4. Предусмотреть кнопки:
  - считывание данных из файла и запись данных в таблицу (предполагается, что в файле данные корректные);
  - возможность добавлять и удалять строки в таблице, соответственно, вводить данные вручную.
  - запись в файл всех данных. Проверять корректность ввода данных: даты должны быть реальными, номер телефона состоять из цифр и, может быть, знак тире, оценки студентов — от 2 до 5. Все остальное можно не проверять. Если есть срок пребывания, дата прибытия и дата отбытия, то срок пребывания должен быть равен разнице между датой отбытия и датой прибытия.
  - запись в файл данных по определенному критерию. Критерий можно вводить вручную через TextBox или выбирать с помощью Radiobox и т. д. Запись в файл должна быть такой, чтобы этот файл можно было открыть в приложении.
  - запись данных по определенному критерию в новую таблицу. При выборе другого критерия старая таблица должна удаляться.
5. При неправильном вводе каких-либо данных таблица выбранных данных должна очищаться.
6. В коде должны быть комментарии и отступы (код должен быть легко читаем).

### **8.2 Вид формы в конструкторе**

Форма имеет вид как показано на рис.29:



Рисунок 29 – Вид формы в конструкторе

### 8.3 Таблица с описанием переименованных элементов формы

Для написания данного приложения дополнительно использовались элементы формы:

- OpenFileDialog [9]
- SaveFileDialog [10]

Все элементы формы были переименованы и их атрибуты изменены. Проведенные изменения представлены в таблице 8

Таблица 8 – Значения атрибутов элементов в приложении «Работа с файлами»

Описание элементов формы	Список измененных атрибутов	Новое значение атрибута
Форма	Text	Work with file
Первая надпись (label)	Name	lblTask

Первая надпись (label)	Text	Создать таблицу Student, содержащую следующие поля: Фамилия, имя, отчество, дата рождения, оценки за сессию (5 оценок). Выполнить следующие действия: считать из файла и вывести на экран список всех студентов; в другой файл вывести студентов, у которых сдана сессия.
Первая кнопка (button)	Name	btnReadFile
Первая кнопка (button)	Text	Открыть файл
Вторая кнопка (button)	Name	btnAddRow
Вторая кнопка (button)	Text	Добавить строку
Третья кнопка (button)	Name	btnRemoveRow
Третья кнопка (button)	Text	Удалить строку
Четвёртая кнопка (button)	Name	btnSaveInFile
Четвёртая кнопка (button)	Text	Сохранить как
Первая таблица (dataGridView)	Name	dGrInput
Вторая таблица (dataGridView)	Name	dGrOutput
Обработчик ошибок (errorProvider)	Name	errPr
openFileDialog	Name	openFile
saveFileDialog	Name	saveFile

## 8.4 Примеры работы

При запуске приложения на экране появляется окно как показано на рис.30.

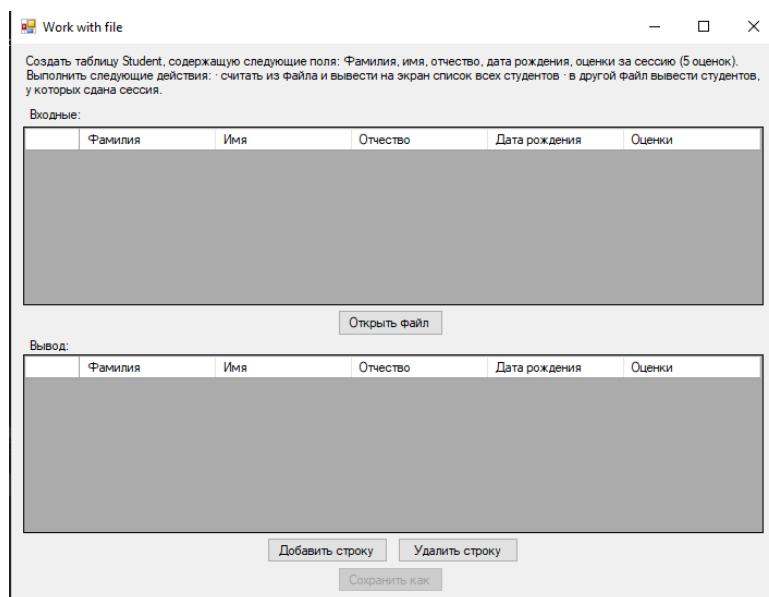


Рисунок 30 – Запуск приложения

При запуске с корректными данными, при нажатии на кнопку «Открыть файл» и выборе txt файла происходит как показано на рис.31:

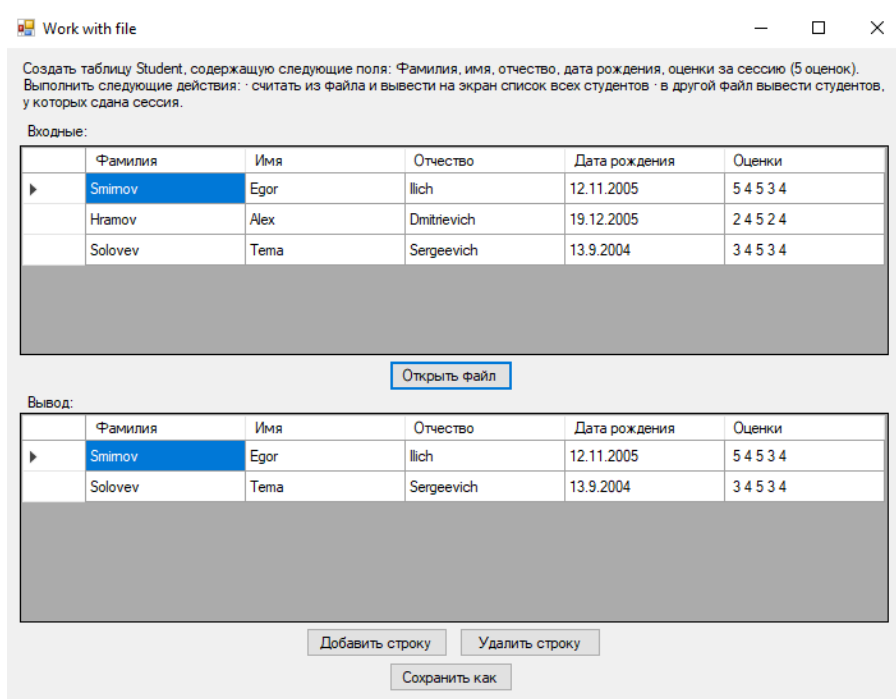


Рисунок 31 – Запуск с корректными данными

При запуске с некорректными данными, при нажатии на кнопку «Открыть файл» и выборе txt файла происходит как показано на рис.32:

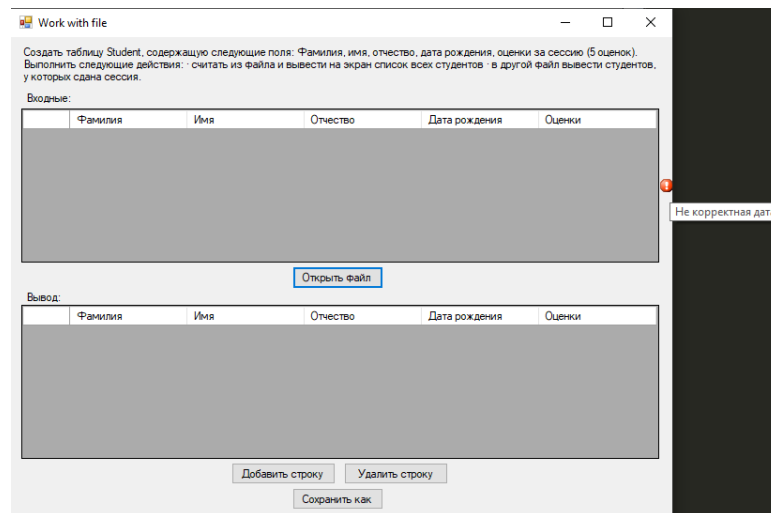


Рисунок 32 – Запуск с некорректными данными

## 8.5 Примеры кода

Функция сохранения в файл:

```

1 // Сохранение файла
2 private: System::Void btnSaveInFile_Click(System::Object^ sender, System::EventArgs^
   ↪ e) {
3     System::IO::Stream^ myStream;
4     if (this->saveFile->ShowDialog() == System::Windows::Forms::DialogResult::OK)
5         if ((myStream = saveFile->OpenFile()) != nullptr) {
6             System::IO::StreamWriter^ sw =
7                 gcnew System::IO::StreamWriter(myStream,
8                     System::Text::Encoding::GetEncoding(1251)
9                 );
10            try {
11                for (int i = 0; i < this->dGrOutput->RowCount - 1; ++i) {
12                    sw->WriteLine(dGrOutputToString(i));
13                }
14                sw->Write(dGrOutputToString(this->dGrOutput->RowCount - 1));
15            }
16            catch (...) {
17            }
18            sw->Close();
19        }
20    }

```

Другие фрагменты кода расположены в приложении 3. Полный код программы приведен в приложении К

## 9 Приложение Тест (Задание 9)

### 9.1 Условие задания

Создать приложение для проведения тестирования.

Должно содержать:

1. Набор вопросов по какой-то теме (и вопросы и ответы должны быть реальными) — не менее 10
2. Вопросы должны выбираться случайным образом.
3. Вопросы должны быть нескольких типов — "Да/нет Выбор одного ответа, Выбор нескольких ответов, Короткий ответ.
4. Необходимо создать сообщения для правильного и неправильного ответа (Молодец, Не правильно и т.д.)
5. Необходимо подсчитать количество правильных ответов и вывести результат.

### 9.2 Вид формы в конструкторе

Форма имеет вид как показано на рис.33:

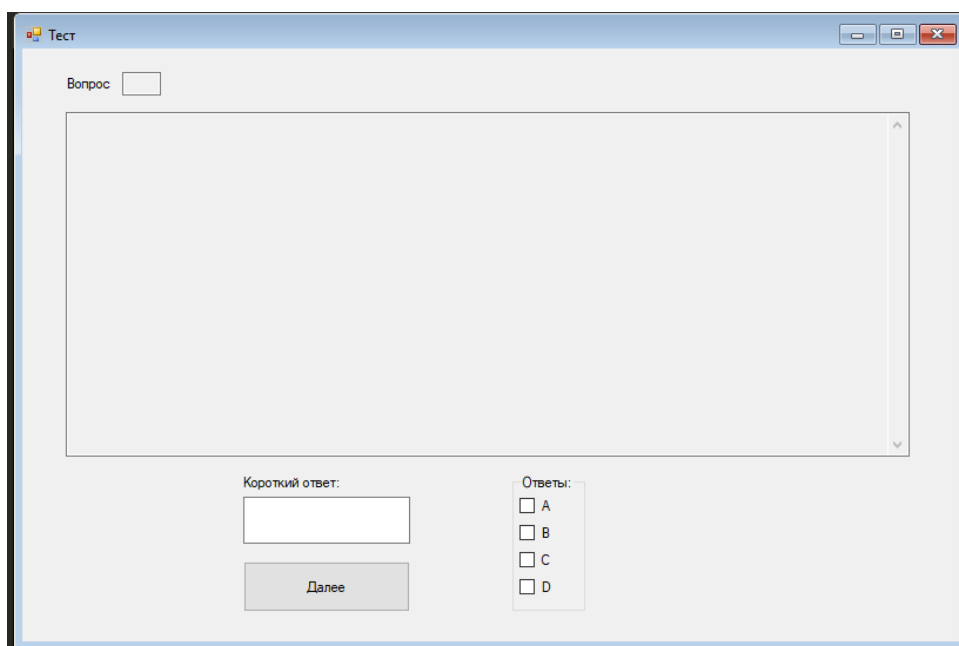


Рисунок 33 – Вид формы в конструкторе

### 9.3 Таблица с описанием переименованных элементов формы

Все элементы формы были переименованы и их атрибуты изменены. Проведенные изменения представлены в таблице 9



Таблица 9 – Значения атрибутов элементов в приложении «Тест»

<b>Описание элементов формы</b>	<b>Список измененных атрибутов</b>	<b>Новое значение атрибута</b>
Форма	Text	Тест
Первая надпись (label)	Name	lblQuest
Первая надпись (label)	Text	Вопрос:
Вторая надпись (label)	Name	lblShortAnswer
Вторая надпись (label)	Text	Короткий ответ:
Кнопка (button)	Name	actionBtn
Кнопка (button)	Text	Далее
Первое текстовое поле (textBox)	Name	countBox
Первое текстовое поле (textBox)	ReadOnly	True
Второе текстовое поле (textBox)	Name	questBox
Второе текстовое поле (textBox)	ReadOnly	True
Третье текстовое поле (textBox)	Name	shortAnswerBox
groupBox	Name	answerGroup
checkBox	Name	answerA
checkBox	Name	answerB
checkBox	Name	answerC
checkBox	Name	answerD
Обработчик ошибок (errorProvider)	Name	errPr

## 9.4 Примеры работы

При запуске приложения на экране появляется окно с одним из случайно выбранных вопросов как показано на рис.34.

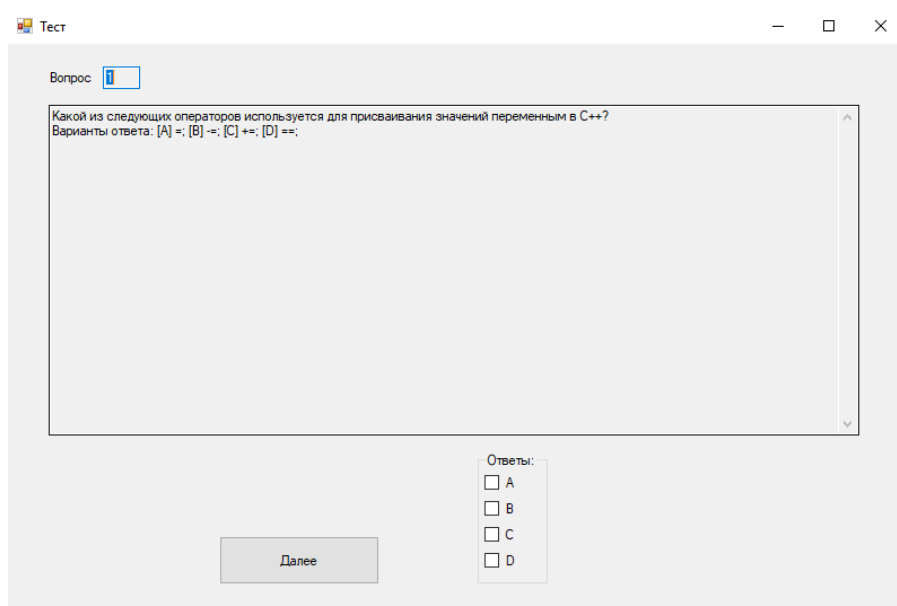


Рисунок 34 – Запуск приложения

При запуске с корректными данными (правильным ответом в данном примере), при нажатии на кнопку «Далее» происходит как показано на рис.35:

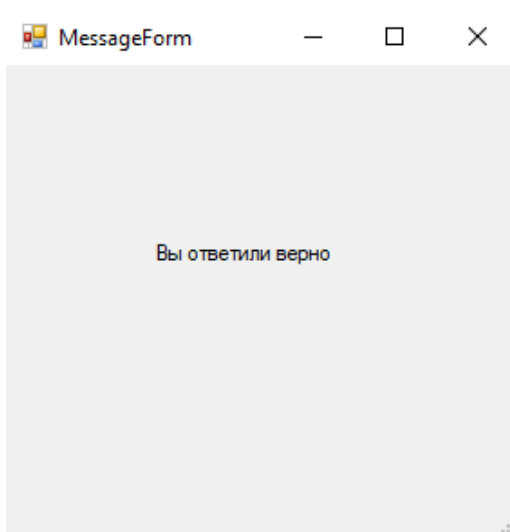


Рисунок 35 – Запуск с корректными данными

При запуске с некорректными данными, при нажатии на кнопку «Далее» происходит как показано на рис.36:

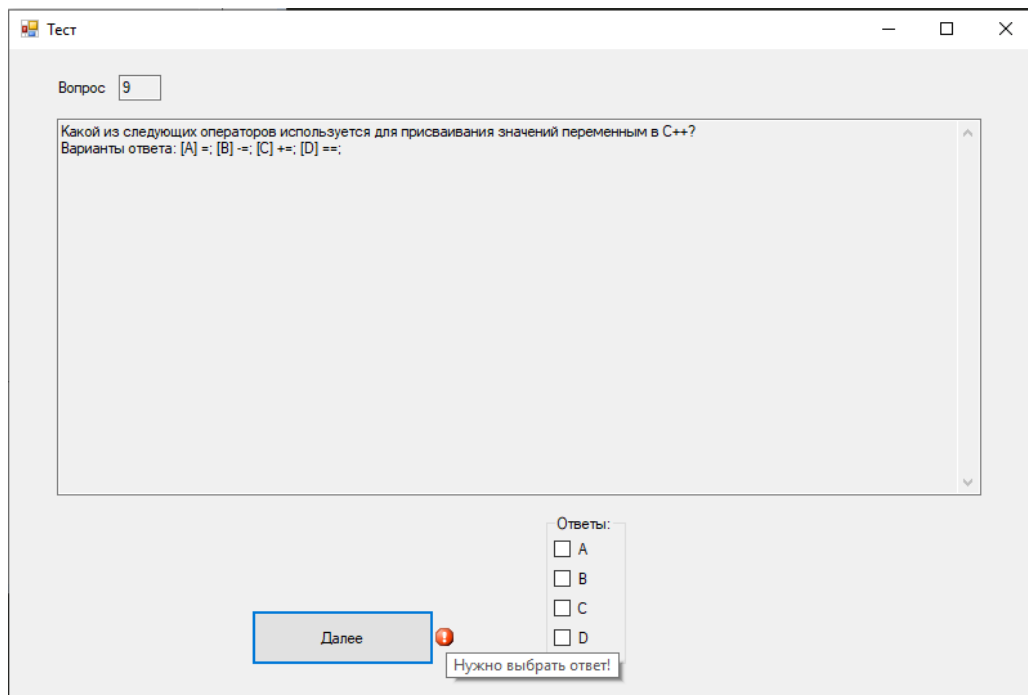


Рисунок 36 – Запуск с некорректными данными

## 9.5 Примеры кода

Функция отображения в зависимости от вопроса нужных элементов на форме:

```

1 // Функция выбора формы
2 void ChooseForm() {
3     int i = RandIndex->at(iter);
4     Quest temp = Questions->at(i);
5     // В зависимости от вопроса вызываем нужную функцию отображения элементов
6     ↪ на форме
7     if (temp.getType() == YesNo) {
8         YesNoForm();
9     }
10    else if (temp.getType() == OneAnswer || temp.getType() == SomeAnswers) {
11        OneAnswerForm();
12    }
13    else {
14        ShortAnswerForm();
15    }
16 }
```

Другие фрагменты кода расположены в приложении И. Полный код программы приведен в приложении К

## **ЗАКЛЮЧЕНИЕ**

В ходе практики было реализовано несколько приложений в среде Microsoft Visual Studio с целью закрепления навыков построения оконного интерфейса и программирования с использованием C++/CLI.

На практике разработаны приложения, содержащие такие элементы интерфейса, как TextBox, Label, Button, DataGridView, OpenFileDialog, SaveFileDialog, ErrorProvider.

Было освоено динамическое создание элементов формы в процессе выполнения программы, изучены виды панелей и способы размещения элементов в них.

Также были применены навыки программирования на основе обратных вызовов при обработке событий формы.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Sutter, Herb. C++/CLI Overview [Text] / Herb Sutter // presentation to ISO C++ standards committee. "— 2003.
- 2 Лаврентьев, ДО. Использование технологии Windows forms для разработки информационных систем (на примере электронного журнала) [Текст] / ДО Лаврентьев, ВЮ Белаш // Вестник Калужского университета. — 2020. — № 4. — С. 82–85.
- 3 Рекурсивное вычисление факториала числа [Электронный ресурс]. — [Б. м. : б. и.]. — URL: <https://foxford.ru/wiki/informatika/rekursivnoe-vychislenie-faktoriala-chisla> (дата обращения: 07.01.2025).
- 4 Shepherdson, John C. Computability of recursive functions [Text] / John C Shepherdson, Howard E Sturgis // Journal of the ACM (JACM). "— 1963. "— Vol. 10, no. 2. "— P. 217–255.
- 5 DataGridView Класс [Электронный ресурс]. — [Б. м. : б. и.]. — URL: <https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.datagridview> (дата обращения: 07.01.2025).
- 6 Hamraz, Bahram. A matrix-calculation-based algorithm for numerical change propagation analysis [Text] / Bahram Hamraz, Nicholas HM Caldwell, P John Clarkson // IEEE Transactions on Engineering Management. "— 2012. "— Vol. 60, no. 1. "— P. 186–198.
- 7 GroupBox Класс [Электронный ресурс]. — [Б. м. : б. и.]. — URL: <https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.groupbox> (дата обращения: 07.01.2025).
- 8 RadioButton Класс [Электронный ресурс]. — [Б. м. : б. и.]. — URL: <https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.radioButton> (дата обращения: 07.01.2025).
- 9 OpenFileDialog Класс [Электронный ресурс]. — [Б. м. : б. и.]. — URL: <https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.openfiledialog> (дата обращения: 07.01.2025).
- 10 SaveFileDialog Класс [Электронный ресурс]. — [Б. м. : б. и.]. — URL: <https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.savefiledialog> (дата обращения: 07.01.2025).

## ПРИЛОЖЕНИЕ А

### Фрагменты кода программы «Вычисление факториала»

Функция очистки текстовых полей в форме:

```
1 private: void ClearAll() { // очистка полей
2     this->txtOutput->Text = "Ошибка ввода";
3     errPr->SetError(txtInput, String::Empty);
4 }
```

Нажатие кнопки «Вычислить» установлено выполнение следующего кода:

```
1 private: System::Void btnCalculate_Click(System::Object^ sender, System::EventArgs^ e)
2     ↪ {
3     ClearAll();
4     long long InputNumber;
5     // переводим сторку из TextBox в число
6     bool result = Int64::TryParse(this->txtInput->Text, InputNumber);
7     // ввели не число
8     if (!result) {
9         errPr->SetError(txtInput, "Введено не целое число");
10    }
11    else {
12        if (InputNumber >= 20) {
13            errPr->SetError(txtInput, "Слишком большое число");
14        }
15        else{
16            // результат
17            long long OutputNumber = fact(InputNumber);
18            // отрицательное число
19            if (OutputNumber == -1) {
20                errPr->SetError(txtInput, "Введено отрицательное число");
21            }
22            // все нормально
23            else {
24                // записываем в поле вывода
25                this->txtOutput->Text = System::Convert::ToString(OutputNumber);
26            }
27        }
28    }
```

## ПРИЛОЖЕНИЕ Б

### Фрагменты кода программы «Простые вычисления»

Нажатие кнопки «Вычислить» установлено выполнение следующего кода:

```
1 private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
2     ClearAll();
3     long long InputX;
4     long long InputY;
5     // переводим строку из TextBox в число
6     bool parseX = Int64::TryParse(this->txtInX->Text, InputX);
7     bool parseY = Int64::TryParse(this->txtInY->Text, InputY);
8     // ввели не число
9     if (!parseX) {
10         errPrX->SetError(txtInX, "Введено не целое число");
11         if (!parseY) {
12             errPrY->SetError(txtInY, "Введено не целое число");
13         }
14     }
15     else if (!parseY) {
16         errPrY->SetError(txtInY, "Введено не целое число");
17     }
18     // проверка на деление на ноль
19     else {
20         if ((InputX == 0 && InputY == 0) ||
21             (InputX == -4 && abs(InputY) == 2)) {
22             errPrX->SetError(txtInX, "Деление на ноль");
23             errPrY->SetError(txtInY, "Деление на ноль");
24         }
25         else{
26             // все нормально
27             // результат
28             double resOutput = solve(InputX, InputY);
29             // записываем в поле вывода
30             this->txtOut->Text = System::Convert::ToString(resOutput);
31         }
32     }
33 }
```

## ПРИЛОЖЕНИЕ В

### Фрагменты кода программы «Рекурсивные вычисления»

Функция очистки текстовых полей в форме:

```
1 private: void ClearAll() { // очистка полей
2     this->txtOut->Text = "";
3     errPrX->SetError(txtInX, String::Empty);
4     errPrN->SetError(txtInN, String::Empty);
5 }
```

Нажатие кнопки «Вычислить» установлено выполнение следующего кода:

```
1 private: System::Void btnStart_Click(System::Object^ sender, System::EventArgs^ e) {
2     ClearAll();
3     double InputX;
4     long long InputN;
5     // переводим строку из TextBox в число
6     bool parseX = double::TryParse(this->txtInX->Text, InputX);
7     bool parseN = Int64::TryParse(this->txtInN->Text, InputN);
8     // ввели не число
9     if (!parseX) {
10         errPrX->SetError(txtInX, "X не число");
11         if (!parseN) {
12             errPrN->SetError(txtInN, "N не целое число");
13         }
14     }
15     else if (!parseN) {
16         errPrN->SetError(txtInN, "N не целое число");
17     }
18     else {
19         // все нормально
20         double result;
21         // вычисляем результат
22         result = recursion(InputX, InputN);
23         // записываем в поле вывода
24         this->txtOut->Text = System::Convert::ToString(result);
25     }
26 }
```



## ПРИЛОЖЕНИЕ Г

### Фрагменты кода программы «Обработка табличных данных. Часть 1.»

Нажатие кнопки «Сумма» установлено выполнение следующего кода:

```
1  // среднее арифметическое
2  private: System::Void SumBtn_Click(System::Object^ sender, System::EventArgs^ e) {
3      ClearAll();
4      long long InputA;
5      long long InputB;
6      // переводим сторку из TextBox в число
7      bool parseA = Int64::TryParse(this->ATxtBox->Text, InputA);
8      bool parseB = Int64::TryParse(this->BTxtBox->Text, InputB);
9      // проверки
10     if (!parseA) {
11         errPrA->SetError(ATxtBox, "A не число");
12         if (!parseB) {
13             errPrB->SetError(BTxtBox, "B не число");
14         }
15     }
16     else if (!parseB) {
17         errPrB->SetError(BTxtBox, "B не число");
18     }
19     else if (InputA < 0) {
20         errPrA->SetError(ATxtBox, "A не может быть отрицательным");
21     }
22     else if (InputB > ArDtGr->RowCount - 1) {
23         errPrB->SetError(BTxtBox, "B больше чем кол-во элементов");
24     }
25     else {
26         // подсчет среднего арифметического
27         long long countres = 0;
28         long long count = 0;
29         double res = 0;
30         double sum = 0;
31         for (long long i = 0; i <= ArDtGr->RowCount - 1; i++) {
32             if ((long long)ArDtGr->Rows[i]->Cells[0]->Value % 2 != 0) {
33                 res += (long long)ArDtGr->Rows[i]->Cells[0]->Value;
34                 countres++;
35             }
36         }
37         for (long long i = InputA; i <= InputB; i++) {
```

```

38     if ((long long)ArDtGr->Rows[i]->Cells[0]->Value % 2 != 0) {
39         sum += (long long)ArDtGr->Rows[i]->Cells[0]->Value;
40         count++;
41     }
42 }
43
44 res -= sum;
45 countres -= count;
46
47 // обработка случая, где таких элементов нет
48 if (res != 0 && countres != 0) {
49     res = (res) / countres;
50     this->SumTxtBox->Text = res.ToString();
51 }
52 else {
53     this->SumTxtBox->Text = "Таких элементов нет";
54 }
55 }
56 }

```

**Нажатие кнопки «Удалить» установлено выполнение следующего кода:**

```

1 // удаление строки
2 private: System::Void DelBtn_Click(System::Object^ sender, System::EventArgs^ e) {
3     // очищаем поля
4     ClearAll();
5     // Если строка не последняя, то удаляем строку
6     if (ArDtGr->RowCount != 0) {
7         ArDtGr->Rows->RemoveAt(ArDtGr->RowCount - 1);
8     }
9 }

```

## ПРИЛОЖЕНИЕ Д

### Фрагменты кода программы «Обработка табличных данных. Часть 2.»

Нажатие кнопки «Удалить столбец» установлено выполнение следующего кода:

```
1 // Удаление столбца
2 private: System::Void btnRemoveColumn_Click(System::Object^ sender,
   ↪ System::EventArgs^ e) {
3     ClearAll();
4     if (countColumn == 1) {
5         erZeroColumn->SetError(btnRemoveColumn, "Нельзя удалить этот столбец");
6         return;
7     }
8     dataGridInput->Columns->Remove("in" + System::Convert::ToString(countColumn));
9     dataGridOutput->Columns->Remove("out" +
   ↪ System::Convert::ToString(countColumn));
10    // Уменьшаем счетчик, если удалили столбец
11    --countColumn;
12 }
```

Нажатие кнопки «Заменить» установлено выполнение следующего кода:

```
1 private: System::Void btnStart_Click(System::Object^ sender, System::EventArgs^ e) {
2     ClearAll();
3     if (countRow < 2) {
4         erChanges->SetError(btnStart, "Для выполнения изменений нужно 2 и более
   ↪ строки");
5         return;
6     }
7
8     // Условно берем строку по индексу 1 как опорный максимум
9     int indexMaxRow = 1;
10    int sumMax = 0;
11    for (int i = 0; i < countColumn; ++i) {
12        int cellValue = 0;
13        bool cell =
   ↪ Int32::TryParse(System::Convert::ToString(dataGridInput->Rows[0]->Cells[i]->Value),
   ↪ cellValue);
14        if (!cell) {
15            erChanges->SetError(btnStart, "Не число");
16            return;
17        }
18    }
19 }
```

```

17     }
18     sumMax += cellValue;
19     dataGridOutput->Rows[0]->Cells[i]->Value = System::Convert::ToString(cellValue);
20 }
21
22 // Находим максимальную строку
23 for (int i = 1; i < countRow; ++i) {
24     int sum = 0;
25     for (int j = 0; j < countColumn; ++j) {
26         int cellValue = 0;
27         bool cell =
28             ↪ Int32::TryParse(System::Convert::ToString(dataGridInput->Rows[i]->Cells[j]->Value),
29             ↪ cellValue);
30         if (!cell) {
31             erChanges->SetError(btnStart, "Не число");
32             return;
33         }
34         sum += cellValue;
35         dataGridOutput->Rows[i]->Cells[j]->Value = System::Convert::ToString(cellValue);
36     }
37     if (sum > sumMax) {
38         indexMaxRow = i;
39         sumMax = sum;
40     }
41 }
42
43 // Меняем 0 строку с максимальной
44 for (int i = 0; i < countColumn; ++i) {
45     int cellValue = 0;
46     int cellMaxValue = 0;
47     bool cell =
48         ↪ Int32::TryParse(System::Convert::ToString(dataGridInput->Rows[0]->Cells[i]->Value),
49         ↪ cellValue);
50     bool cellMax =
51         ↪ Int32::TryParse(System::Convert::ToString(dataGridInput->Rows[indexMaxRow]->Cells[i]->
52         ↪ cellMaxValue);
53     if (!cell || !cellMax) {
54         erChanges->SetError(btnStart, "Не число");
55         return;
56     }
57     dataGridOutput->Rows[indexMaxRow]->Cells[i]->Value =
58         ↪ System::Convert::ToString(cellValue);

```

```
52     dataGridOutput->Rows[0]->Cells[i]->Value =  
        ↪ System::Convert::ToString(cellMaxValue);  
53     }  
54 }
```

## ПРИЛОЖЕНИЕ Е

### Фрагменты кода программы «Матричный калькулятор»

Функция нахождения ранга:

```
1 // Нахождение ранга
2 int Rank(int** matrix, int row, int column) {
3     if (row > column) {
4         column = row;
5     }
6     else {
7         row = column;
8     }
9
10    // Заполняем доп матрицу, если она не квадратная изначально
11    // то добавляем недостающее кол-во строк/столбцов с элементами = 0
12    double** tempMatrix = new double* [row];
13    for (int i = 0; i < row; ++i) {
14        tempMatrix[i] = new double[row];
15        for (int j = 0; j < column; ++j) {
16            if (i >= matrixA->RowCount) {
17                tempMatrix[i][j] = 0;
18            }
19            else if (j >= matrixA->ColumnCount) {
20                tempMatrix[i][j] = 0;
21            }
22            else {
23                tempMatrix[i][j] = matrix[i][j];
24            }
25        }
26    }
27
28    int rank = row;
29    for (int row_i = 0; row_i < rank; ++row_i) {
30        // Если диагональный элемент != 0
31        if (tempMatrix[row_i][row_i]) {
32            for (int column_i = 0; column_i < column; ++column_i) {
33                if (column_i != row_i) {
34                    double mult = tempMatrix[column_i][row_i] /
35                        tempMatrix[row_i][row_i];
36                    for (int i = 0; i < rank; ++i) {
37                        tempMatrix[column_i][i] -= mult * tempMatrix[row_i][i];
```

```

38     }
39     }
40     }
41 }
42 //Если диагональный элемент == 0, то
43 else {
44     bool reduce = true;
45     // находим не нулевой элемент в данном столбце
46     for (int i = row_i + 1; i < column; ++i) {
47         // Меняем строку с ненулевым элементом с данной строкой
48         if (tempMatrix[i][row_i]) {
49             double* temp = tempMatrix[row_i];
50             tempMatrix[row_i] = tempMatrix[i];
51             tempMatrix[i] = temp;
52             reduce = false;
53             break;
54         }
55     }
56     // Если не нашли не один ненулевой элемент в данном столбце
57     // то все элементы этого столбца == 0
58     if (reduce) {
59         // уменьшаем ранг
60         rank--;
61         for (int i = 0; i < column; ++i) {
62             tempMatrix[i][row_i] = tempMatrix[i][rank];
63         }
64     }
65     // Прогоняем еще раз эту строку
66     row_i--;
67 }
68 }
69 return rank;
70 }

```

### Функция умножения матрицы A на число num:

```

1 // умножение матрицы A на число num
2 void MultNum(int row, int column, int num) {
3     int a;
4     bool check;
5     for (int i = 0; i < row; i++) {

```

```

6      for (int j = 0; j < column; j++) {
7          // Проверка на не целое число
8          check =
            ↪ Int32::TryParse(System::Convert::ToString(matrixA->Rows[i]->Cells[j]->Value),
            ↪ a);
9          if (!check) {
10             throw gcnew FormatException("В матрице присутствуют не целые числа");
11         }
12         // Умножение на число
13         matrixResult->Rows[i]->Cells[j]->Value = a * num;
14     }
15 }
16 }

```



## ПРИЛОЖЕНИЕ Ж

### Фрагменты кода программы «Использование коллекций»

Нажатие кнопки «Новая очередь (новый элемент после всех макс)» установлено выполнение следующего кода:

```
1 private: System::Void btnNewAfterMax_Click(System::Object^ sender,
    ↪ System::EventArgs^ e) {
2     ClearEP();
3     this->tBOutputNAM->Clear();
4     // Вспомогательная очередь
5     System::Collections::Generic::Queue<int> buffer;
6     System::Collections::Generic::Queue<int> buffer2;
7     if (!q.Count) {
8         this->eP1->SetError(btnNewAfterMax, "Пустая очередь");
9         return;
10    }
11    int number, max = q.Peek();
12    bool res = Int32::TryParse(tBInputNAM->Text, number);
13    if (!res) {
14        this->eP1->SetError(tBInputNAM, "Не целое число");
15        return;
16    }
17    // Пока очередь не пуста
18    while (q.Count) {
19        // Записываем во вспомогательную очередь первый элемент
20        buffer2.Enqueue(q.Peek());
21        // Находим максимум
22        if (q.Peek() > max) {
23            max = q.Peek();
24        }
25        // Удаляем первый элемент очереди
26        q.Dequeue();
27    }
28    // Пока очередь не пуста
29    while (buffer2.Count) {
30        // Записываем в основную очередь первый элемент
31        q.Enqueue(buffer2.Peek());
32        // Удаляем первый элемент очереди
33        buffer2.Dequeue();
34    }
```

```

35 // Пока очередь не пуста
36 while (q.Count) {
37     // Записываем во вспомогательную очередь первый элемент
38     buffer.Enqueue(q.Peek());
39     // Записываем после максимальных элементов новый элемент
40     if (q.Peek() == max) {
41         buffer.Enqueue(number);
42     }
43     buffer2.Enqueue(q.Peek());
44     // Удаляем первый элемент очереди
45     q.Dequeue();
46 }
47 String^ result = "";
48 // Пока очередь не пуста
49 while (buffer2.Count) {
50     // Записываем в основную очередь первый элемент
51     q.Enqueue(buffer2.Peek());
52     // Удаляем первый элемент очереди
53     buffer2.Dequeue();
54 }
55 while (buffer.Count) {
56     // Записываем результат
57     result += System::Convert::ToString(buffer.Peek() + " ");
58     // Удаляем первый элемент очереди
59     buffer.Dequeue();
60 }
61 this->tBOutputNAM->Text = result;
62 }

```

Нажатие кнопки «Сумма четных в интервале» установлено выполнение следующего кода:

```

1 private: System::Void btnSum_Click(System::Object^ sender, System::EventArgs^ e) {
2     ClearEP();
3     // Вспомогательная очередь
4     System::Collections::Generic::Queue<int> buffer;
5     int a, b;
6     bool resA = Int32::TryParse(tBInputA->Text, a);
7     bool resB = Int32::TryParse(tBInputB->Text, b);
8     int sum = 0, i = 0;
9     if (!resA) {

```

```

10     this->eP1->SetError(tBInputA, "Не целое число");
11     return;
12 }
13 if (!resB) {
14     this->eP1->SetError(tBInputB, "Не целое число");
15     return;
16 }
17 if (a < 0 || a > b || a > q.Count) {
18     this->eP1->SetError(tBInputA, "Не верный интервал");
19     return;
20 }
21 if (b > q.Count) {
22     this->eP1->SetError(tBInputB, "Не верный интервал");
23     return;
24 }
25
26 // Пока очередь не пуста
27 while (q.Count) {
28     // Записываем во вспомогательную очередь первый элемент
29     buffer.Enqueue(q.Peek());
30     // Суммируем четные элементы
31     if (q.Peek() % 2 == 0 && (i >= a && i <= b)) {
32         sum += q.Peek();
33     }
34     // Удаляем первый элемент очереди
35     q.Dequeue();
36     i++;
37 }
38 // Пока очередь не пуста
39 while (buffer.Count) {
40     // Записываем в основную очередь первый элемент
41     q.Enqueue(buffer.Peek());
42     // Удаляем первый элемент очереди
43     buffer.Dequeue();
44 }
45 // Выводим результат
46 this->tBOutputSum->Text = Convert.ToString(sum);
47 }

```

## ПРИЛОЖЕНИЕ 3

### Фрагменты кода программы «Работа с файлами»

Нажатие кнопки «Открыть файл» установлено выполнение следующего кода:

```
1 // Чтение данных
2 private: System::Void btnReadFile_Click(System::Object^ sender, System::EventArgs^
   ↪ e) {
3     this->errPr->SetError(this->dGrInput, System::String::Empty);
4     System::IO::Stream^ myStream;
5     btnSaveInFile->Enabled = true;
6     if (this->openFile->ShowDialog() == System::Windows::Forms::DialogResult::OK) {
7         if ((myStream = openFile->OpenFile()) != nullptr) {
8             System::IO::StreamReader^ sw =
9                 gcnew System::IO::StreamReader(myStream,
10                    System::Text::Encoding::GetEncoding(1251)
11                );
12
13             // Очистка таблиц
14             ClearTables();
15
16             System::String^ strBuf = "";
17             array<System::String^>^ arrBuf;
18             array<System::String^>^ dateBuf;
19             int i = 0;
20             // Вывод в таблицы
21             while (sw->Peek() + 1) {
22                 DateTime dateTemp;
23                 this->dGrInput->Rows->Add(1);
24                 // Чтение строки из файла
25                 strBuf = sw->ReadLine();
26                 try {
27                     // Вывод ФИО в таблицу dGrInput
28                     arrBuf = strBuf->Split( ' ');
29                     this->dGrInput->Rows[i]->Cells[0]->Value = arrBuf[0];
30                     this->dGrInput->Rows[i]->Cells[1]->Value = arrBuf[1];
31                     this->dGrInput->Rows[i]->Cells[2]->Value = arrBuf[2];
32
33                     // Проверка на корректность даты
34                     dateBuf = arrBuf[3]->Split( '. ');
```

```

35     strBuf = dateBuf[0] + "." + dateBuf[1] + "." + dateBuf[2];
36     bool res = DateTime::TryParse(strBuf, dateTemp);
37     if (!res) {
38         this->errPr->SetError(this->dGrInput, "Не корректная дата");
39         // Очистка таблиц
40         ClearTables();
41         return;
42     }
43     // Вывод даты в таблицу dGrInput
44     this->dGrInput->Rows[i]->Cells[3]->Value = dateTemp.Day + "." +
        ↪ dateTemp.Month + "." + dateTemp.Year;
45     int grade;
46     bool flag = true;
47     // Вывод оценок в таблицу dGrInput
48     for (int j = 4; j < 9; ++j) {
49         bool res = Int32::TryParse(arrBuf[j], grade);
50
51         if (!res || grade > 5 || grade < 2) {
52             this->errPr->SetError(this->dGrInput, "Не корректные оценки");
53             // Очистка таблиц
54             ClearTables();
55             return;
56         }
57         else if (grade <= 5 && grade >= 2) {
58             this->dGrInput->Rows[i]->Cells[4]->Value += " " +
        ↪ System::Convert::ToString(grade);
59         }
60         if (grade == 2) {
61             // Если есть 2 в оценках, то flag = false
62             flag = false;
63         }
64     }
65     // Если flag == false => есть хотя бы одна 2 => его в таблицу результата
        ↪ выводить не нужно
66     if (!flag) {
67         ++i;
68         continue;
69     }
70     // Вывод в таблицу dGrOutput сдавших сессию
71     this->dGrOutput->Rows->Add(1);
72     for (int j = 0; j < 5; ++j)

```

```

73         this->dGrOutput->Rows[
74             this->dGrOutput->RowCount - 1
75         ]->Cells[j]->Value = this->dGrInput->Rows[i]->Cells[j]->Value;
76     }
77     catch (...) {
78     }
79     ++i;
80 }
81
82 }
83 }
84
85 }

```

Нажатие кнопки «Удалить строку» установлено выполнение следующего кода:

```

1 // Удаление строки
2 private: System::Void btnRemoveRow_Click(System::Object^ sender,
3     ↪ System::EventArgs^ e) {
4     this->errPr->SetError(this->dGrOutput, System::String::Empty);
5     if (!this->dGrOutput->CurrentRow) {
6         this->errPr->SetError(this->dGrOutput, "Нельзя удалить несуществующую
7         ↪ строку");
8         return;
9     }
10    if (!this->dGrOutput->CurrentRow->IsNewRow) {
11        int i = this->dGrOutput->CurrentRow->Index;
12        this->dGrOutput->Rows->Remove(this->dGrOutput->Rows[i]);
13    }
14 }

```

## ПРИЛОЖЕНИЕ И

### Фрагменты кода программы «Тест»

Нажатие кнопки «Далее» установлено выполнение следующего кода:

```
1  System::Void actionBtn_Click(System::Object^ sender, System::EventArgs^ e) {
2      errPr->Clear();
3      std::string answer = "";
4      // Считываем ответ на вопрос
5      if (iter != Questions->size()) {
6          answer = ReadAnswer();
7      }
8      // если ответ правильного формата
9      if (answer != "") {
10         int i = RandIndex->at(iter);
11         bool flag = false;
12         // проверка ответа
13         if (Questions->at(i).getRightAnswer() == answer) {
14             // выставление баллов за ответ
15             Questions->at(i).setResult(1);
16             flag = true;
17         }
18         else {
19             // выставление баллов за ответ
20             Questions->at(i).setResult(0);
21         }
22         // Вывод результата ответа на данный вопрос (правильно ответили или нет)
23         MessageForm^ msgForm = gcnew MessageForm();
24         msgForm->setFlag(flag);
25         msgForm->ShowDialog();
26         iter++;
27         // Переход к следующему вопросу
28         if (iter != Questions->size()) {
29             countBox->Text = Convert::ToString(iter + 1);
30             ChooseForm();
31             int i = RandIndex->at(iter);
32             questBox->Text = gcnew String(Questions->at(i).getText().c_str());
33         }
34         // Если вопросы кончились выводим результат
35         else {
36             questBox->Text = gcnew String("Вы набрали " + CalcResult() + " баллов");
37         }
38     }
```

```
38     }
39 }
```

**Функция подсчета правильных ответов:**

```
1 // Подсчет правильных ответов
2 int CalcResult() {
3     int sum = 0;
4     // Проходимся по вектору вопросов
5     for (int i = 0; i < Questions->size(); ++i) {
6         // Прибавляем число из результата вопроса (1 или 0)
7         sum += Questions->at(i).getResult();
8     }
9     // Выводим кол-во правильных ответов
10    return sum;
11 }
```



## ПРИЛОЖЕНИЕ К

### Архив с отчетом о выполненной работе

В приложенном ZIP-архиве, для ознакомления, содержатся следующие файлы:

**Папка** Otchet — L<sup>A</sup>T<sub>E</sub>X-вариант отчета о практике;

**Папка** task-1 — задание №1;

**Папка** task-2 — задание №2 вариант 7;

**Папка** task-3 — задание №3 вариант 7;

**Папка** task-4 — задание №4 вариант 7;

**Папка** task-5 — задание №5 вариант 13;

**Папка** task-6 — задание №6;

**Папка** task-7 — задание №7 вариант 8;

**Папка** task-8 — задание №8 вариант 1;

**Папка** task-9 — задание №9;

Otchet.pdf — отчет о практике;