МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

УТВЕРЖДАЮ
Зав.кафедрой,
доцент, к. ф.-м. н.
_____ С. В. Миронов

ОТЧЕТ О ПРАКТИКЕ

студента 2 курса 251 группы факультета КНиИТ
Рыданова Никиты Сергеевича

вид практики: учебная
кафедра: математической кибернетики и компьютерных наук
курс: 2
семестр: 1
продолжительность: 20 нед., с 01.09.20 г. по 12.01.21 г.

Руководитель практики от университета,

доцент, к. ф.-м. н. _____ А. С. Иванова

Руководитель практики от организации (учреждения, предприятия),

доцент, к. ф.-м. н. _____ А. С. Иванова

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

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью практики является освоение механизма построения оконного интерфейса приложений в среде Visual Studio. В результате прохождения практики должны быть отработаны навыки:

- создания нового проекта;
- добавления и настройки элементов управления;
- отладка корректного ввода данных для решения поставленной задачи;
- разработки алгоритма решения поставленной задачи с использованием оконного интерфейса;
- тестирования приложения;
- документирования разработанного кода.

1 Вычисление факториала

Задание: Разработать приложение для вычисления факториала по приведенному примеру.

Создано окно приложения, содержащее два элемента `TextBox`, два элемента `Label` и один элемент `Button`. Для отображения сообщений об ошибках в окно добавлен элемент `ErrorProvider`. Вид окна представлен на рисунке 1.

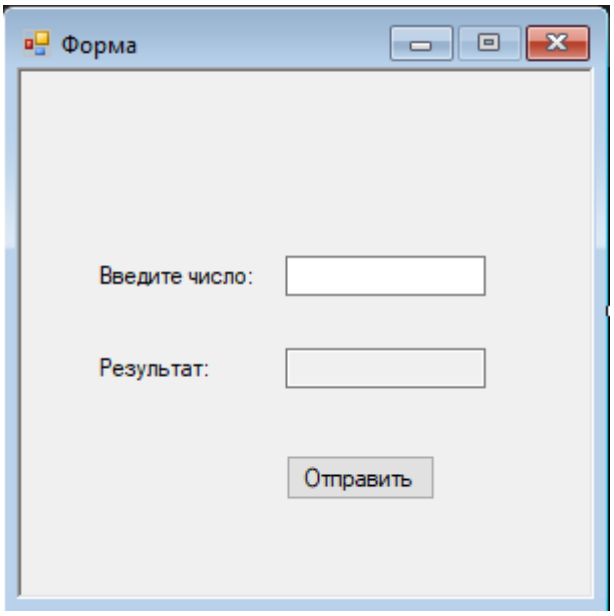

Рисунок 1 – Внешний вид формы программы для вычисления факториала

У элементов изменены значения некоторых атрибутов. Значения измененных атрибутов представлены в таблице 1.

Таблица 1 – Значения атрибутов элементов в приложении «Факториал»

Наименование атрибута	Значение
Для формы	
Text	Форма
FormBorderStyle	FixedSingle
MaximizeBox	False
Для первой надписи	
(Name)	lblInput
Text	Введите целое число
Для второй надписи	
(Name)	lblOutput
Text	Результат
Для первого текстового поля	
(Name)	txtInput
Для второго текстового поля	
(Name)	txtOutput
Для кнопки	
(Name)	btnCalculate
Text	Вычислить
Для обработчика ошибок	
(Name)	errorProvider1

Для работы программы была написана функция вычисления факториала:

```
#pragma once
typedef long long ll;

ll fact(ll N) {
        if (N < 0)
                return -1;
        else if (N == 0 || N == 1)
                return 1;
        else return N * fact(N - 1);
}
```

Здесь переменная N — число, для которого нужно вычислить факториал.

На нажатие кнопки «Вычислить» установлено выполнение следующего кода:

```
    private: System::Void btnCalculate_Click(System::Object^ sender,
    ↪  System::EventArgs^ e) {
                ClearAll();
```

```
3               ll InputNumber;
4               bool result = Int64::TryParse(txtInput->Text, InputNumber);
5               if (!result) {
6                       errorProvider1->SetError(txtInput, "Введено не целое
                        ↪  число");
7                       return;
8               }
9               if (InputNumber > 20) {
10                      errorProvider1->SetError(txtInput, "Число слишком
                        ↪  большое");
11                      return;
12              }
13              ll OutputNumber = fact(InputNumber);
14              if (OutputNumber == -1) {
15                      errorProvider1->SetError(txtInput, "Введено
                        ↪  отрицательное число");
16                      return;
17              }
18              txtOutput->Text = System::Convert::ToString(OutputNumber);
19      }
```

После запуска приложения на экране появляется окно (см. рисунок 2)

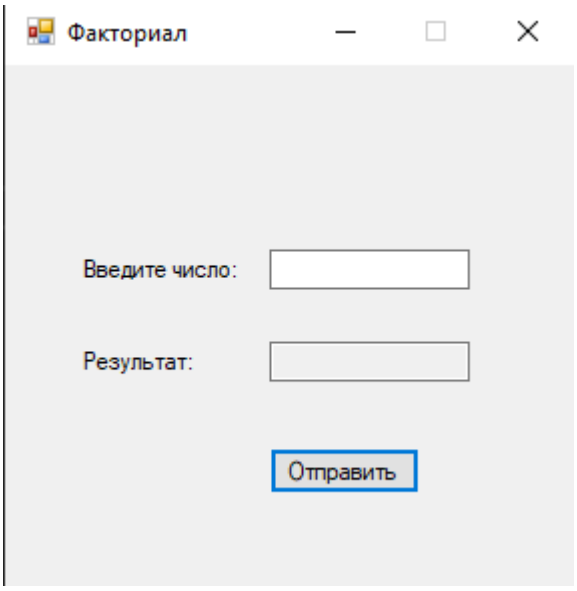

Рисунок 2 – Скриншот запуска программы

При вводе целого числа после нажатия кнопки в поле вывода приводится результат вычисления факториала для заданного числа (см. рисунок 3).

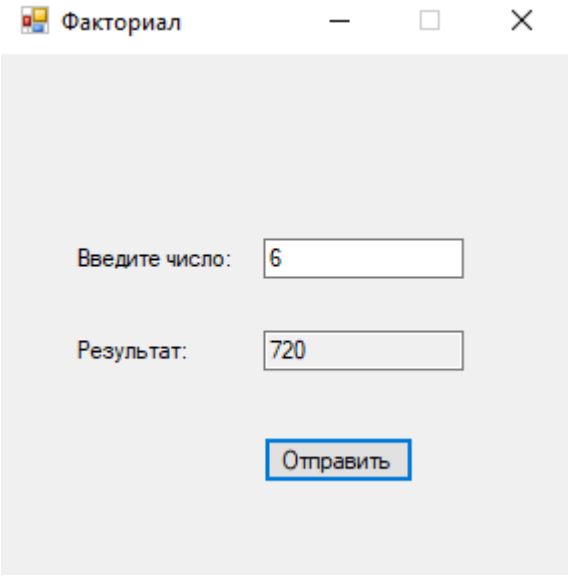

Рисунок 3 – Результат работы

Ввод некорректных значений обрабатывается элементом `ErrorProvider` и сопровождается сообщением об ошибке (см. рисунок 4)

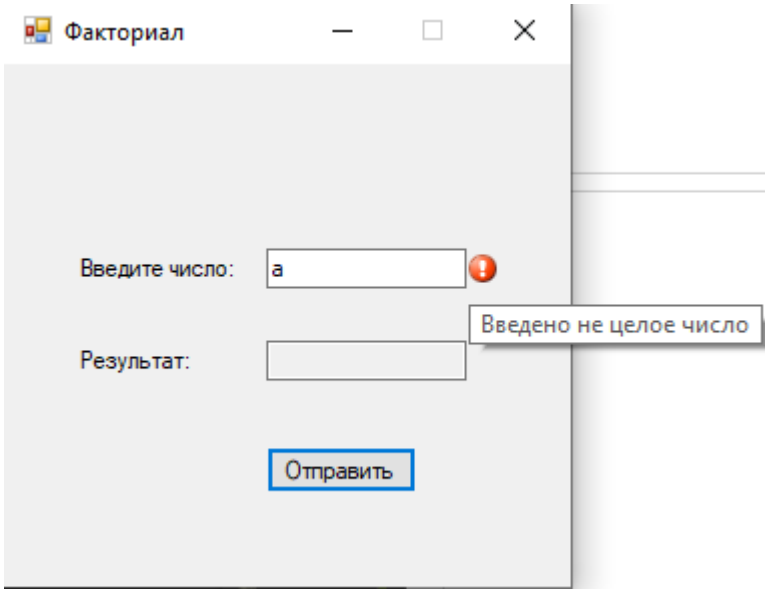

Рисунок 4 – Сообщение об ошибке

Полный код программы приведен в приложении А.

2 Простые вычисления

Задание: Вычислить значение выражения $\dfrac{\cos x + \sin y}{\ln(x+y)}$

Создано окно, содержащее три элемента `TextBox`, три элемента `Label` и один элемент `Button`. Вид окна представлен на рисунке 5.

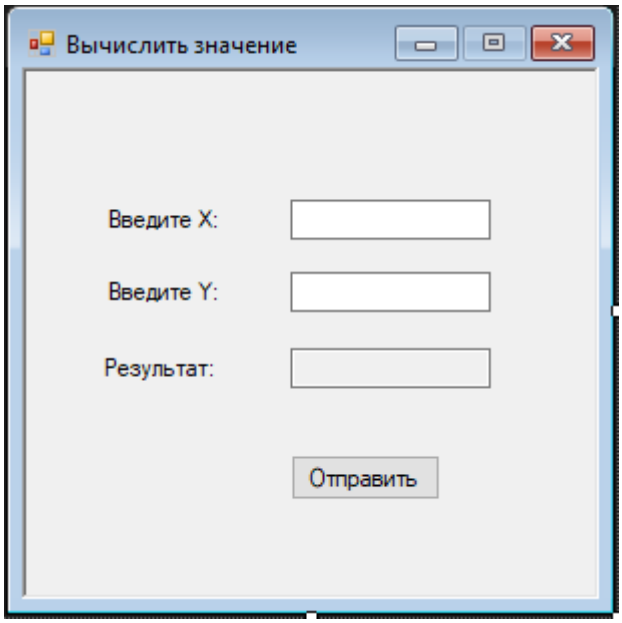

Рисунок 5 – Внешний вид формы в конструкторе

У элементов изменены значения некоторых атрибутов. Значения измененных атрибутов представлены в таблице 2.

Таблица 2 – Значения атрибутов элементов в приложении «Простые вычисления»

Наименование атрибута	Значение
Для формы	
Text	Вычислить значение
FormBorderStyle	FixedSingle
MaximizeBox	False
Для первой надписи	
(Name)	xLabel
Text	Введите X
Для второй надписи	
(Name)	yLabel
Text	Введите Y
Для третьей надписи	
(Name)	lblOutput
Text	Результат:
Для первого текстового поля	
(Name)	xInput
Для второго текстового поля	
(Name)	yInput
Для третьего текстового поля	
(Name)	txtOutput
Для кнопки	
(Name)	btnCalculate
Text	Вычислить
Для обработчика ошибок	
(Name)	errorProvider1

Для работы программы была написана функция вычисления заданного выражения:

```
#pragma once
#include <math.h>
typedef long long ll;

double f(ll x, ll y) {
        return (cos(x) + sin(y)) / (log(x + y));
}
```

и функция, проверяющая выбранное поле на корректность [1, 2]:

```
    bool VarValidation(System::Windows::Forms::TextBox^ Input, ll& x) { //
    ↪ Проверка числа для поля x из текста объекта Input
```

```
                ll InputNumber;
                bool IsVariableValid = Int64::TryParse(Input->Text,
                ↪ InputNumber); // Пробуем записать в InputNumber число из
                ↪ потока

                if (!IsVariableValid) { // Если не вышло
                        errorProvider1->SetError(Input, "Переменная не целое
                        ↪ число");
                        return 0;
                }
                x = InputNumber; // Обновляем значение переменной
                return 1;
        }
```

Логику работы программы реализует фрагмент кода, привязанный к кнопке

```
        private: System::Void btnCalculate_Click(System::Object^ sender,
        ↪ System::EventArgs^ e) {
                ClearAll();

                ll x = 0;
                ll y = 0;

                bool XisOkay = VarValidation(xInput, x); // Проверяем
                ↪ корректность x
                bool YisOkay = VarValidation(yInput, y); // Проверяем
                ↪ корректность y

                if (!XisOkay || !YisOkay) return; // Если какая-то из них
                ↪ некорректна - завершим работу. Все необходимые выводы
                ↪ исключений уже были произведены

                ll summary = x + y; // Считаем сумму для логарифма

                if (summary == 1) { // Если аргумент равен единице - логарифм
                ↪ равен нулю
                        errorProvider1->SetError(txtOutput, "Деление на
                        ↪ ноль");
                        return;
                }

```

```
19                  if (summary <= 0) { // Если аргумент меньше нуля - неверно
20                          errorProvider1->SetError(txtOutput, "Недопустимое
                            ↪  значение для логарифма");
21                          return;
22                  }
23
24                  double OutputNumber = f(x, y); // Получаем значение функции
                    ↪  для заданных чисел x, y
25
26                  txtOutput->Text = System::Convert::ToString(OutputNumber); //
                    ↪  Отображаем ответ
27          }
```

Функция `ClearAll()` реализует очищение полей от ошибок. После запуска приложения на экране появляется окно (см. рисунок 6)

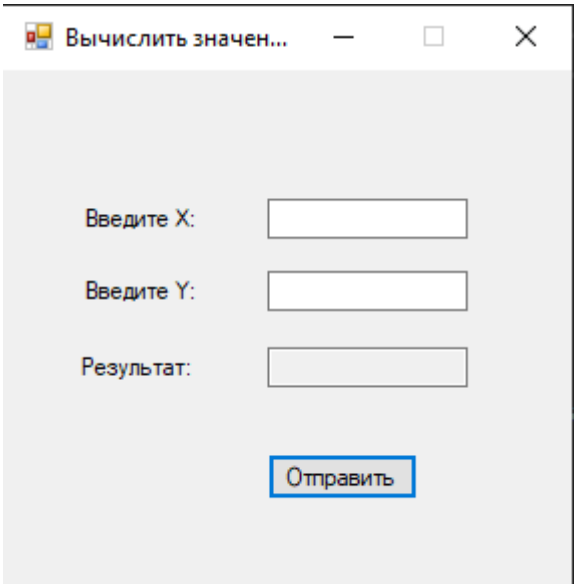

Рисунок 6 – Скриншот запуска программы

При вводе целого числа после нажатия кнопки в поле вывода приводится результат вычисления факториала для заданного числа (см. рисунок 7).

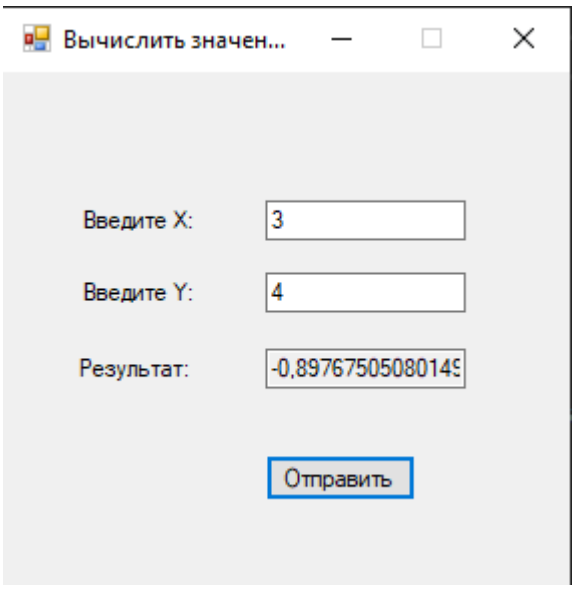

Рисунок 7 – Результат работы

Ввод некорректных значений обрабатывается элементом `ErrorProvider` и сопровождается сообщением об ошибке (см. рисунок 8)

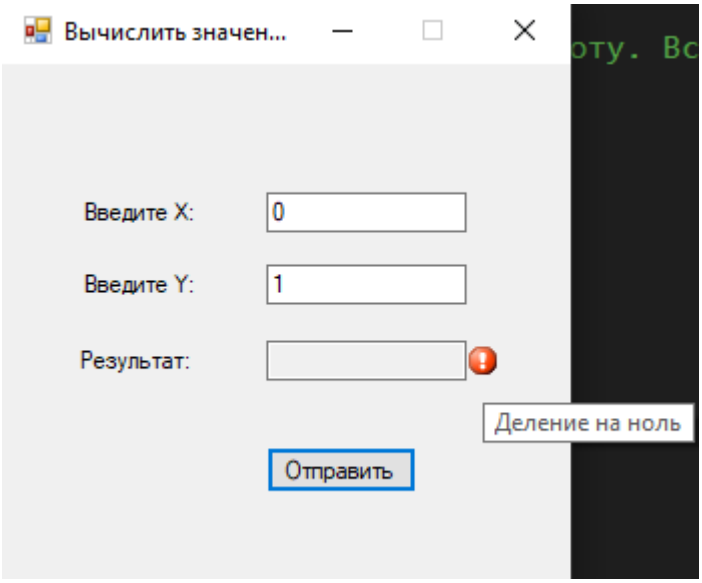

Рисунок 8 – Сообщение об ошибке

Полный код программы приведен в приложении А.

3 Рекурсивные вычисления

Задание: Создать рекурсивную функцию, которая для заданного целого n вычисляет сумму ряда $\sum_{i=1}^{n} 2^i$

Для этого были использовано 3 элемента `Label`, 2 элемента `TextBox`, 1 элемент `PictureBox` и один элемент `Button`.

Создано окно, содержащее три элемента `TextBox`, три элемента `Label` и один элемент `Button`. После запуска приложения появляется окно (см. рисунок 9) .

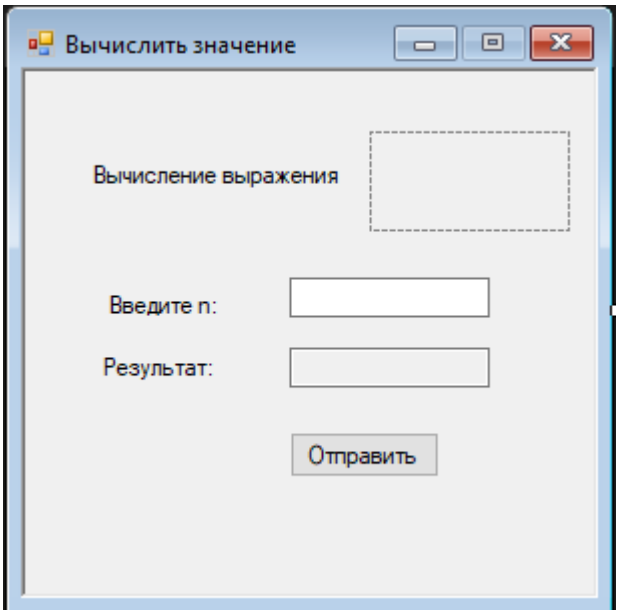

Рисунок 9 – Внешний вид формы в конструкторе

У элементов изменены значения некоторых атрибутов. Значения измененных атрибутов представлены в таблице 3.

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

Наименование атрибута	Значение
Для формы	
Text	Вычислить значение
FormBorderStyle	FixedSingle
MaximizeBox	False
Для первой надписи	
(Name)	formulaText
Text	Вычисление выражения
Для второй надписи	
(Name)	xLabel
Text	Введите N
Для третьей надписи	
(Name)	lblOutput
Text	Результат:
Для первого текстового поля	
(Name)	xInput
Для второго текстового поля	
(Name)	txtOutput
Для кнопки	
(Name)	btnCalculate
Text	Вычислить
Для обработчика ошибок	
(Name)	errorProvider1
Для изображения выражения	
(Name)	pictureBox1

Программа содержит функции (`ClearAll`, `VarValidation`), аналогичные функциям в программе «Простые вычисления» за исключением логики работы кнопки:

```
1     private: System::Void btnCalculate_Click(System::Object^ sender,
      ↪  System::EventArgs^ e) {
2                 ClearAll();
3
4                 ll n = 0;
5
6                 bool NisOkay = VarValidation(nInput, n); // Проверяем
                  ↪  корректность n
7
```

```
8                   if (n <= 0) {
9                           errorProvider1->SetError(nInput, "Недопустимая
                            ↪  степень");
10                          return;
11                  }
12
13
14                  if (!NisOkay) return; // Если число некорректно - завершим
                    ↪  работу. Все необходимые выводы исключений уже были
                    ↪  произведены
15
16                  ll OutputNumber = f(n); // Получаем значение ряда для
                    ↪  заданного n
17
18                  txtOutput->Text = System::Convert::ToString(OutputNumber); //
                    ↪  Отображаем ответ
19          }
```

и функции подсчета суммы ряда

```
1   #pragma once
2
3   long long myPow(int n) {
4           if (n == 0)
5                   return 1;
6           else
7                   return 2 * myPow(n - 1);
8   }
9
10  long long f(int n) {
11          if (n == 0)
12                  return 0;
13          else
14                  return myPow(n) + f(n - 1);
15  }
```

Неверно введеннные данные обрабатываются элементом `ErrorProvider`. Полный код программы представлен в приложении А.

4 Обработка табличных данных. Часть 1.

Задание: Найти сумму нечетных элементов, меньших заданного числа. Вывести максимальный четный элемент.

Создано окно приложения, содержащее два элемента `TextBox`, два элемента `Label`, 4 элемента `Button`, и 1 элемент `DataGridView`. Для отображения сообщений об ошибках в окно добавлен элемент `ErrorProvider`. Вид окна представлен на рисунке 10.

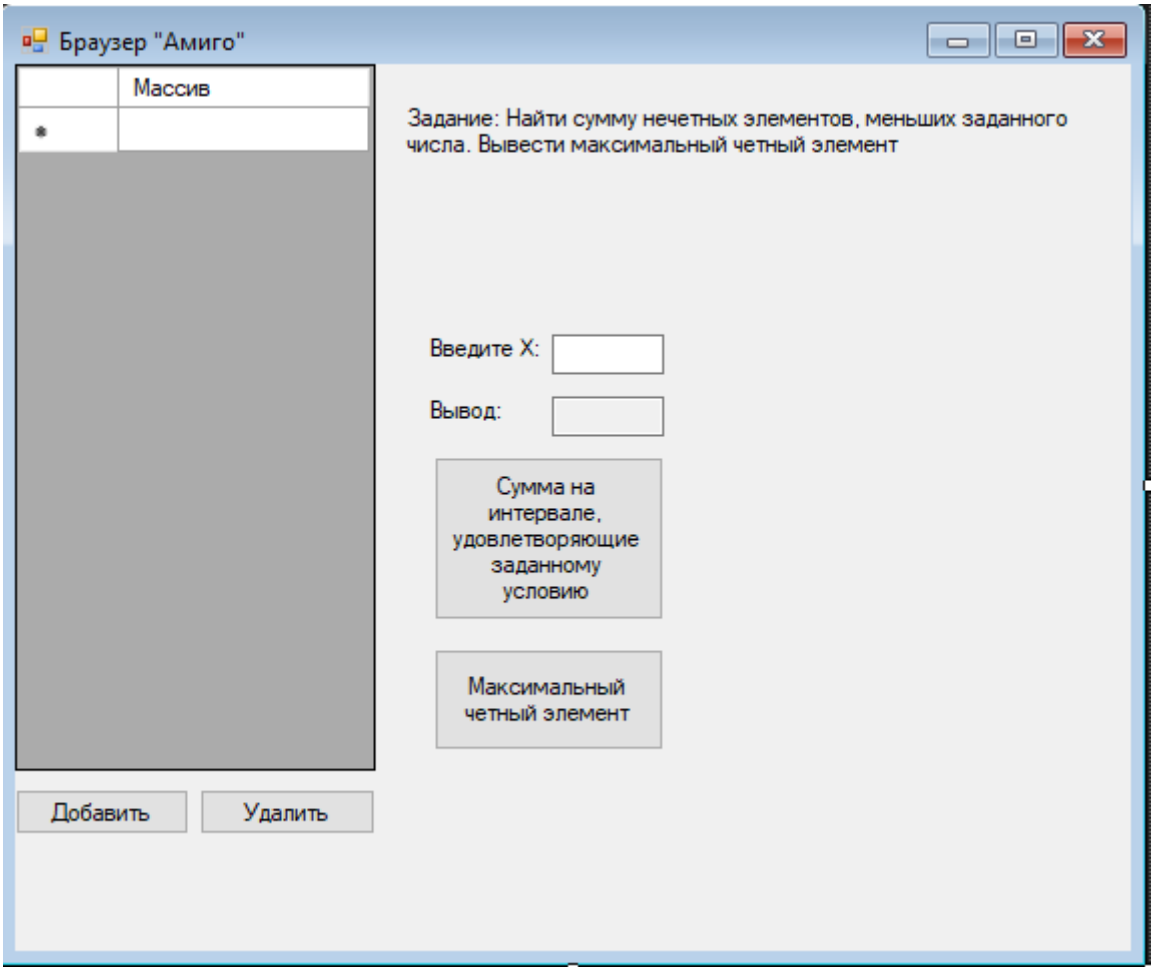

Рисунок 10 – Внешний вид формы программы

У элементов изменены значения некоторых атрибутов. Значения измененных атрибутов представлены в таблице 4.

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

Наименование атрибута	Значение
Для формы	
Text	Браузер Амиго
FormBorderStyle	FixedSingle
MaximizeBox	False
Для первой надписи	
(Name)	xLabel
Text	Введите X
Для второй надписи	
(Name)	lblOutput
Text	Вывод
Для первого текстового поля	
(Name)	txtX
Для второго текстового поля	
(Name)	txtOutput
Для кнопки суммирования	
(Name)	btnSummary
Text	Сумма на интервале, удовлетворяющие заданному условию
Для кнопки для нахождения максимального четного элемента	
(Name)	btnFindMaxEven
Text	Максимальный четный элемент
Для кнопки для нахождения максимального четного элемента	
(Name)	btnFindMaxEven
Text	Максимальный четный элемент
Для кнопки добавления ряда	
(Name)	btnAdd
Text	Добавить
Для кнопки удаления ряда	
(Name)	btnRemove
Text	Удалить
Для таблицы	
(Name)	grid
(Name)	grid
Для обработчика ошибок	
(Name)	errorProvider1

Для изменения количества рядов в таблице были реализованы кнопки добавления и удаления ряда. Код соответствующих функций приведен ниже:

```
    private: System::Void btnAdd_Click(System::Object^ sender,
    ↪ System::EventArgs^ e) { // Добавление
                        this->grid->Rows->Add(1);
        }
    private: System::Void btnRemove_Click(System::Object^ sender,
    ↪ System::EventArgs^ e) { // Удаление
                if (!this->grid->CurrentRow->IsNewRow) {
                        int i = this->grid->CurrentRow->Index;
                        this->grid->Rows->Remove(this->grid->Rows[i]);
                }
        }
```

Решение каждой из двух задач реализовано в соответствующих кнопках. Код представлен ниже:

```
private: System::Void btnSummary_Click(System::Object^ sender, System::EventArgs^ e) { //
↪ Вычисление необходимой суммы
        ClearOutput();
        bool noBadCells = true;
        bool check2 = true;
        if ((int)WrongCells->size() > 0) {
                noBadCells = false;
        }
        int xValue;
        bool isCorrectValue = Int32::TryParse(txtX->Text, xValue);
        if (!isCorrectValue) {
                errorProvider1->SetError(txtX, "Неверное значение для X");
        }
        if (!isCorrectValue) {
                return;
        }
        else {
                errorProvider1->SetError(txtX, String::Empty);
        }
        if (!noBadCells) {
                return;
        }
        int summary = 0;
        for (int i = 0; i < this->grid->RowCount; ++i) {
                int val = System::Convert::ToInt32(this->grid->Rows[i]->Cells[0]->Value);
                if (val % 2 == 1 && val < xValue) {
                        summary += val;
                }
        }
        ClearOutput();
        txtOutput->Text = System::Convert::ToString(summary);
        }
```

```
32 private: System::Void btnFindMaxEven_Click(System::Object^ sender, System::EventArgs^ e) {
   ↪ // Вывод необходимого максимального элемента на экран
33         ClearOutput();
34         if (WrongCells->size() > 0) {
35                 return;
36         }
37         int max_val = -1e9;
38         int max_ind = -1;
39         for (int i = 0; i < this->grid->RowCount; ++i) {
40                 int val = System::Convert::ToInt32(this->grid->Rows[i]->Cells[0]->Value);
41                 if (val > max_val && val % 2 == 0) {
42                         max_val = val;
43                         max_ind = i;
44                 }
45         }
46         ClearOutput();
47         txtOutput->Text = System::Convert::ToString(max_val);
48 }
```

Контроль за корректностью введенных данных осуществляется через поддерживание невалидных ячеек в set из библиотеки STL. Работа с ней осуществляется через обработку события `CellLeave` [2–5]:

```
1  private: System::Void grid_CellLeave(System::Object^ sender,
   ↪ System::Windows::Forms::DataGridViewCellEventArgs^ e) {
2          int val = 0;
3
4          System::String^ Value = System::Convert::ToString(
5                  this->grid->Rows[grid->CurrentRow->Index]->Cells[0]->EditedFormattedValue);
6
7          bool isCorrectValue = Int32::TryParse(Value, val); // Пробуем записать в InputNumber
           ↪ число из потока
8          if (!isCorrectValue && (Value != "" && grid->CurrentRow->Index != grid->RowCount))
           ↪ {
9                  errorProvider1->SetError(grid, "Неверный тип для ячейки");
10                 WrongCells->insert(grid->CurrentRow->Index);
11         }
12         else {
13                 WrongCells->erase(grid->CurrentRow->Index);
14                 if ((int)WrongCells->size() == 0) {
15                         errorProvider1->SetError(grid, String::Empty);
16                 }
17         }
18 }
```

После запуска приложения на экране появляется окно (см. рисунок 11)

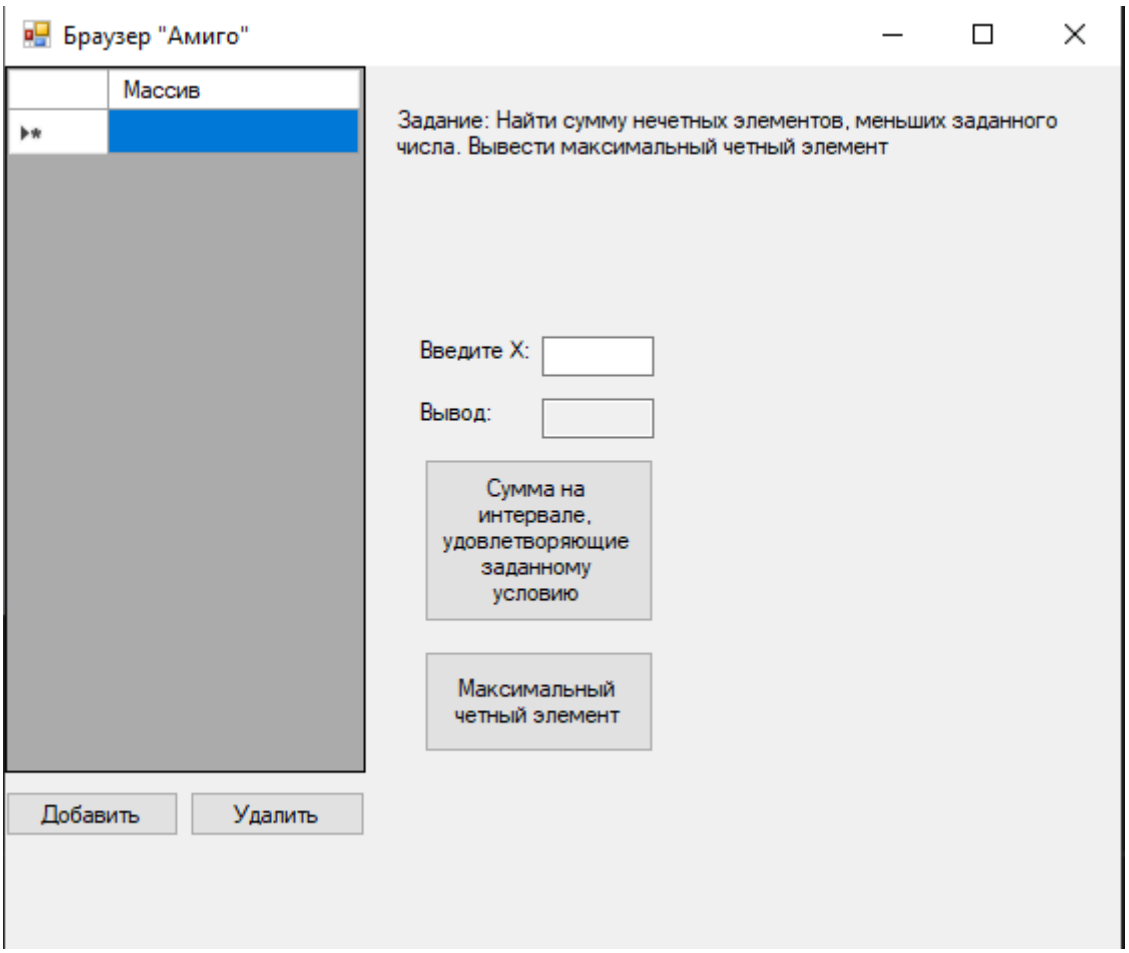

Рисунок 11 – Скриншот запуска программы

При вводе целого числа после нажатия кнопки в поле вывода приводится результат вычисления суммы в заданном интервале (см. рисунок 12).

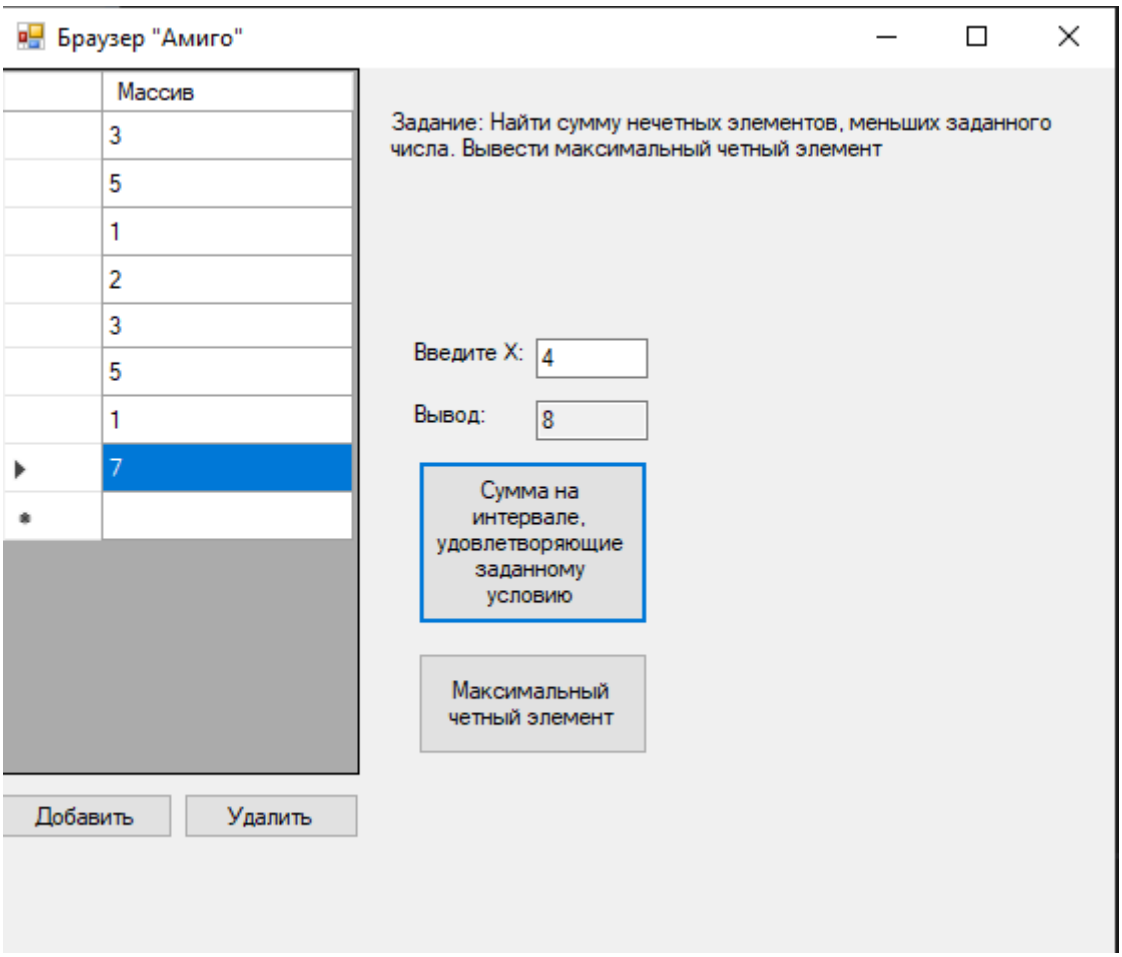

Рисунок 12 – Результат работы

При вводе целого числа после нажатия кнопки в поле вывода приводится результат вычисления максимального четного элемента (см. рисунок 13).

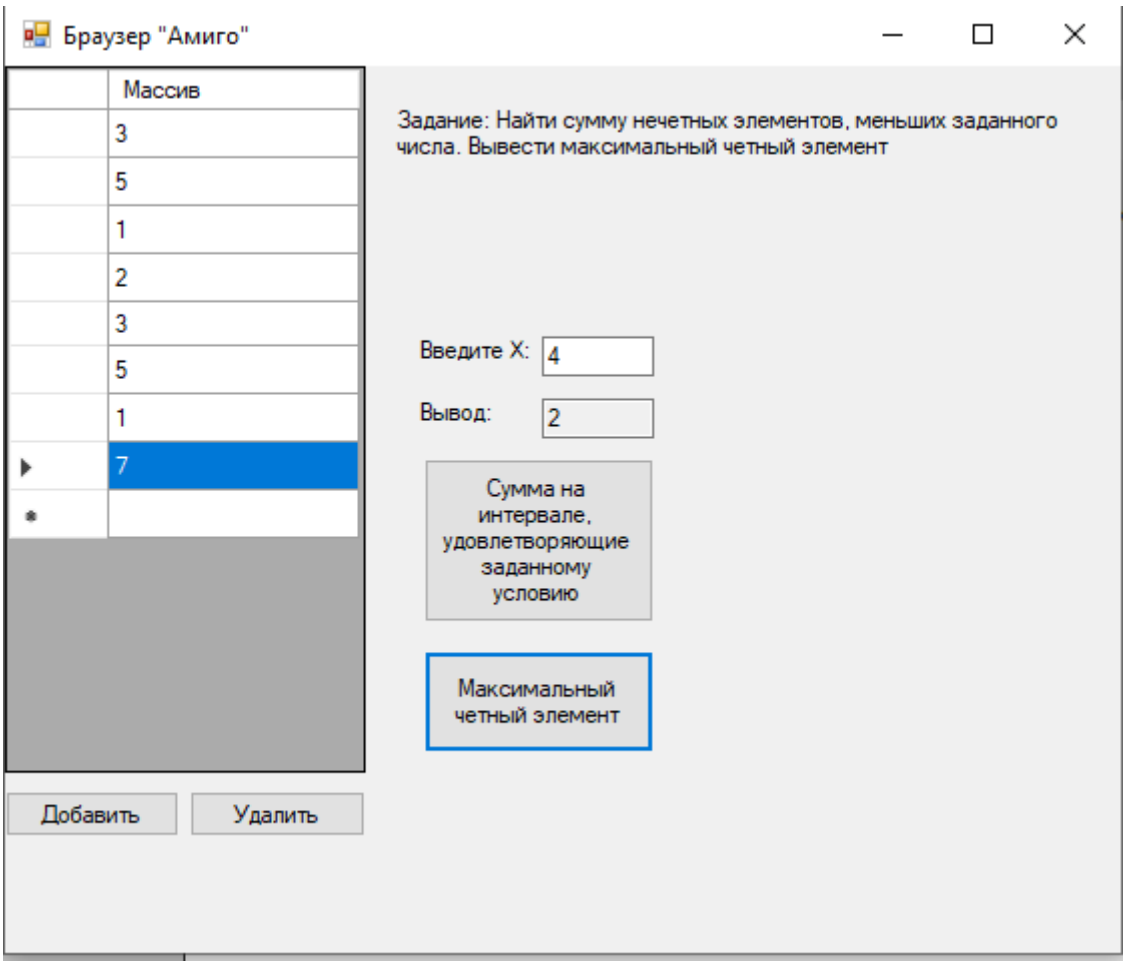

Рисунок 13 – Результат работы

Ввод некорректных значений обрабатывается элементом `ErrorProvider` и сопровождается сообщением об ошибке (см. рисунок 14)

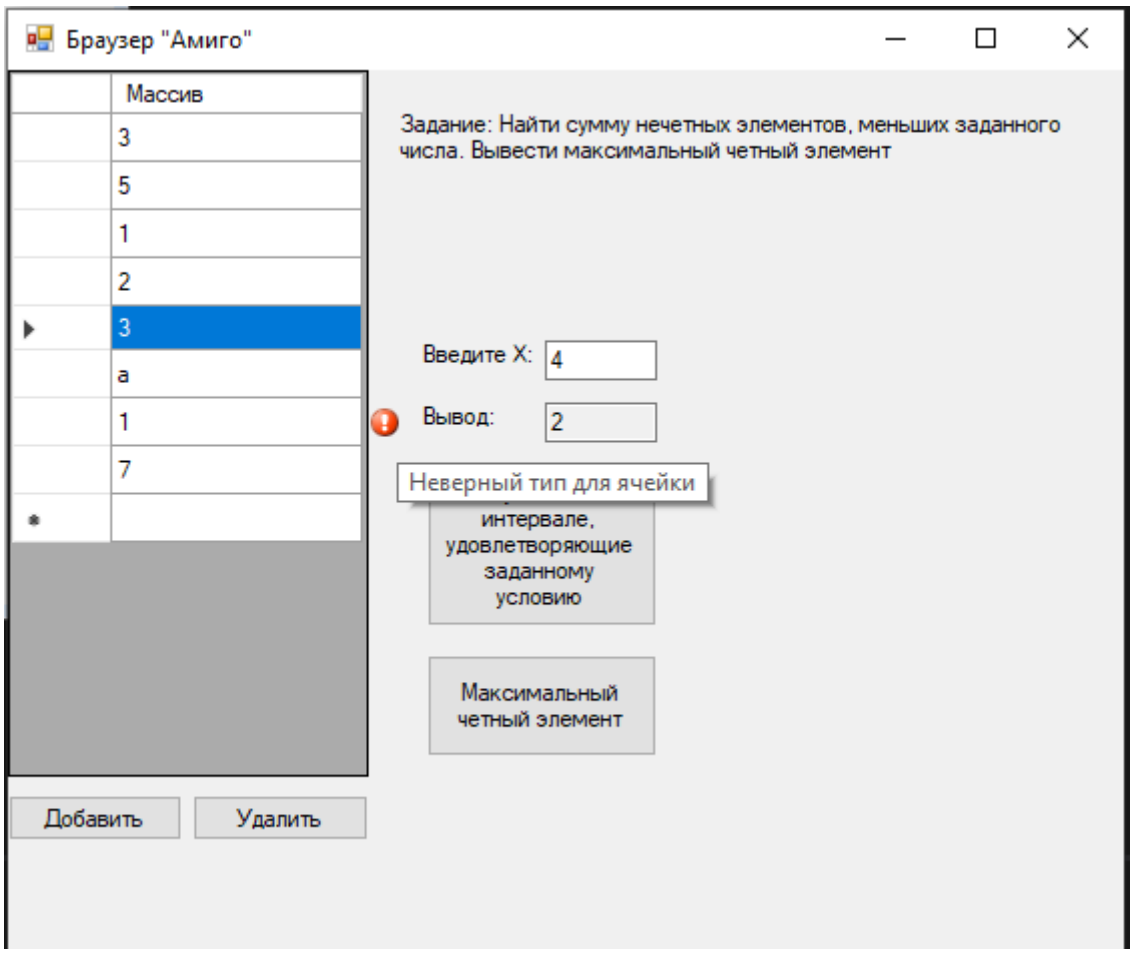

Рисунок 14 – Сообщение об ошибке

Полный код программы приведен в приложении А.

5 Обработка табличных данных. Часть 2.

Задание: Ваш выбор: Все нечетные столбцы заменить столбцом X. (Нумерация столбцов массива начинается с нуля.)

Создано окно приложения, содержащее 5 элементов `Button`, 3 элемента `DataGridView` и 5 элементов `Label`. Для отображения сообщений об ошибках в окно добавлен элемент `ErrorProvider`. Вид окна представлен на рисунке 15.

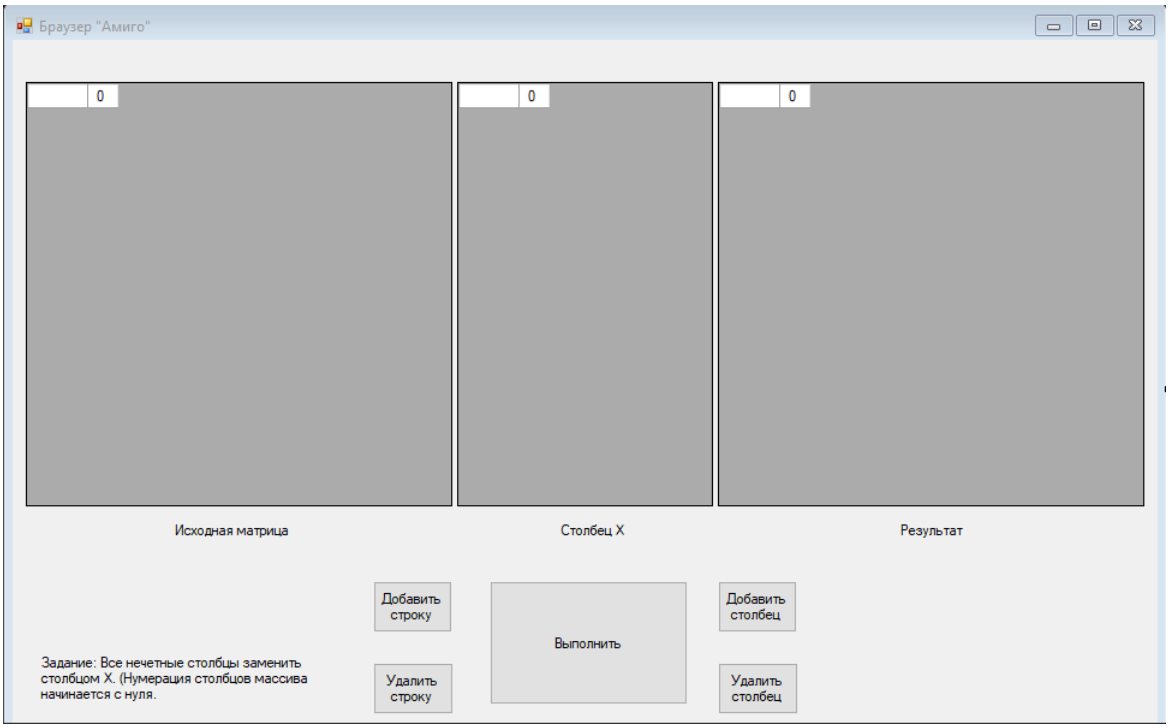

Рисунок 15 – Внешний вид формы программы

У элементов изменены значения некоторых атрибутов. Значения измененных атрибутов представлены в таблице 5.

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

Наименование атрибута	Значение
Для формы	
Text	Браузер Амиго
FormBorderStyle	FixedSingle
MaximizeBox	False
Для первой надписи	
(Name)	taskLabel
Text	Задание: Все нечетные столбцы заменить столбцом X.
Для второй надписи	
(Name)	initLabel
Text	Исходная матрица
Для третьей надписи	
(Name)	xLabel
Text	Столбец X
Для четвертой надписи	
(Name)	resultLabel
Text	Результат
Для кнопки "Выполнить"	
(Name)	btnCalc
Для кнопки добавления ряда	
(Name)	btnAddRow
Text	Добавить
Для кнопки удаления ряда	
(Name)	btnRemoveRow
Text	Удалить
Для кнопки добавления столбца	
(Name)	btnAddColumn
Text	Добавить
Для кнопки удаления столбца	
(Name)	btnRemoveColumn
Text	Удалить
Для обработчика ошибок	
(Name)	errorProvider1

После запуска приложения на экране появляется окно (см. рисунок 16)

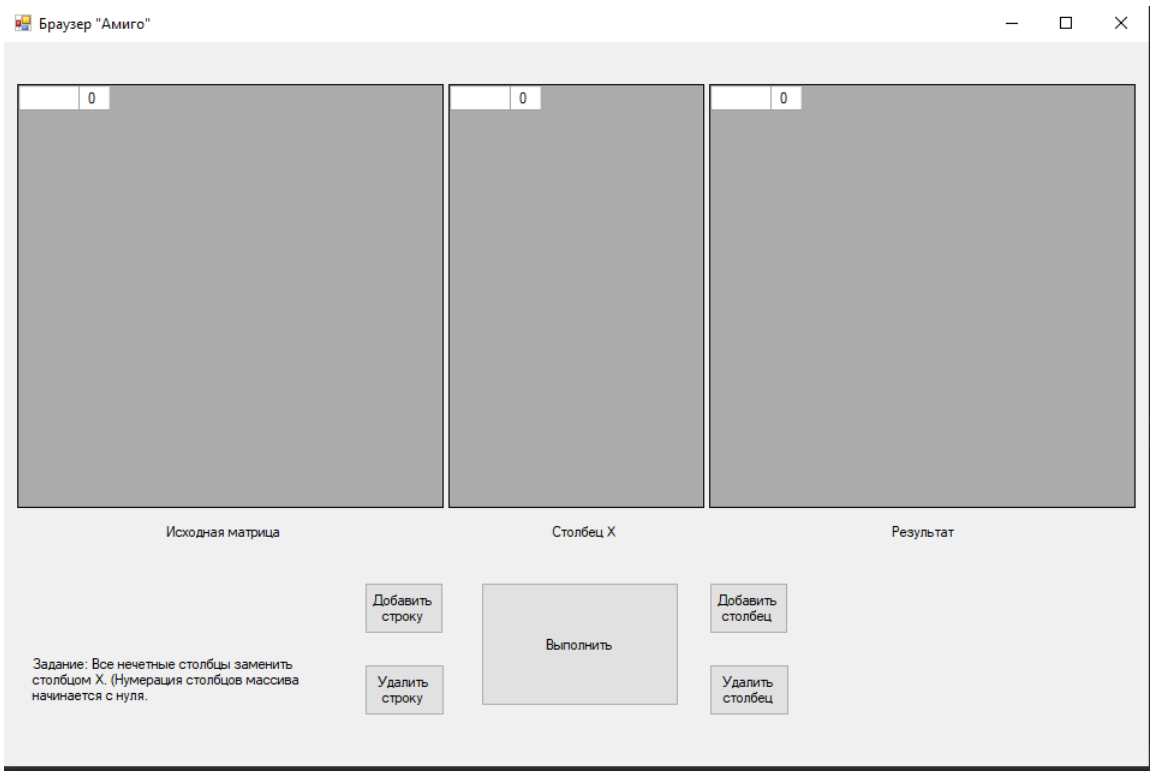

Рисунок 16 – Скриншот запуска программы

При вводе целого числа после нажатия кнопки в поле вывода приводится результат вычисления суммы в заданном интервале (см. рисунок 17).

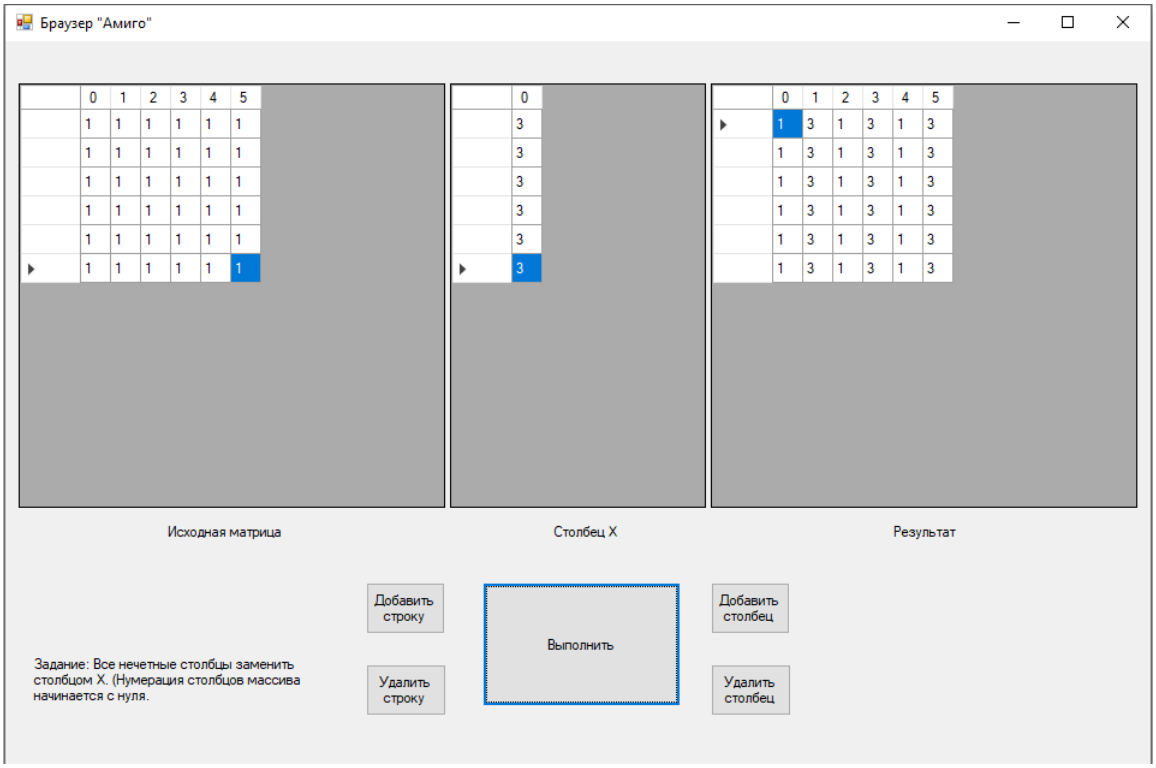

Рисунок 17 – Результат работы

Ввод некорректных значений обрабатывается элементом `ErrorProvider`

и сопровождается сообщением об ошибке (см. рисунок 18)

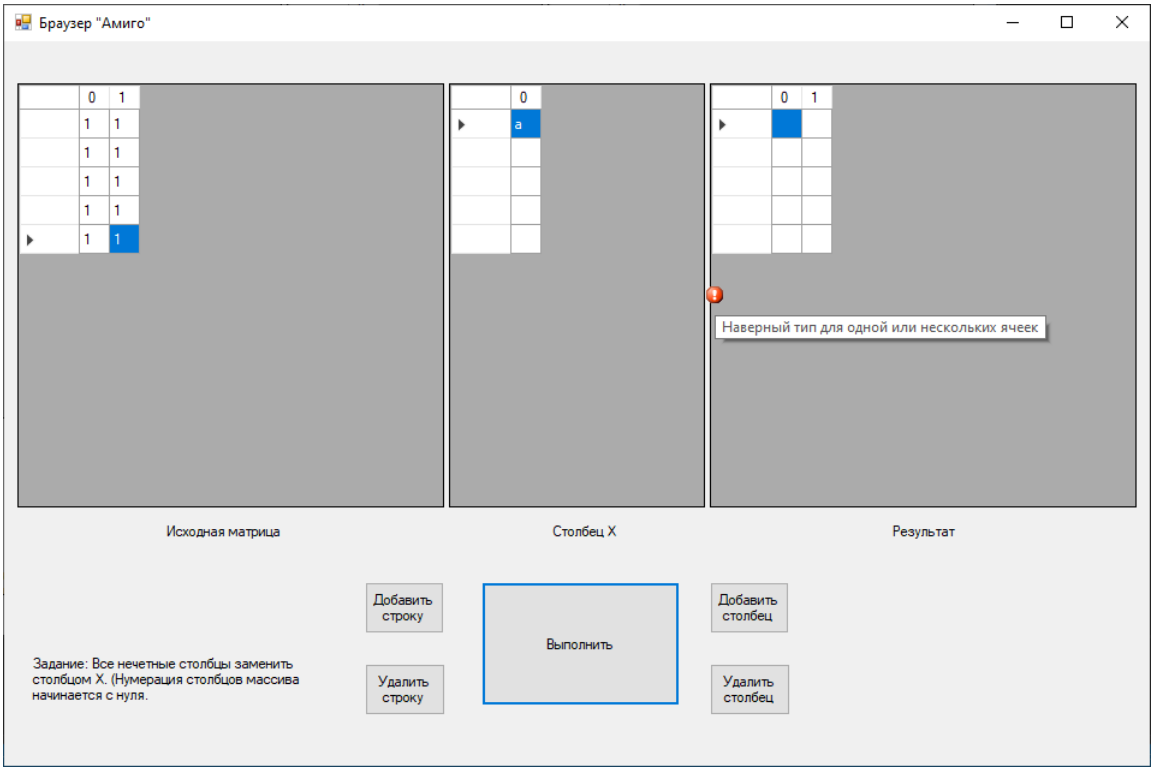

Рисунок 18 – Сообщение об ошибке

Полный код программы приведен в приложении А.

6 Матричный калькулятор

Задание: 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
4. Сложение/вычитание двух матриц
5. Сложение/вычитание двух векторов
6. Скалярное и векторное произведение двух векторов
7. Транспонированная матрица
8. Определитель, ранг матрицы

Выводить сообщения об ошибках (ввод не числа, несоответствие размерностей)

Всего приложение содержит 11 элементов `Button`, 13 элементов `RadioButton`, 5 элементов `Label`, 4 элемента `TextBox` и 3 элемента `DataGridView`.

Интерфейс приложения можно условно разделить на четыре части, две из которых отвечают за определение соответствующих аргументов операции, третья за непосредственно саму операцию, а четвертая графически отображает аргументы и результат Вид окна представлен на рисунке 19.

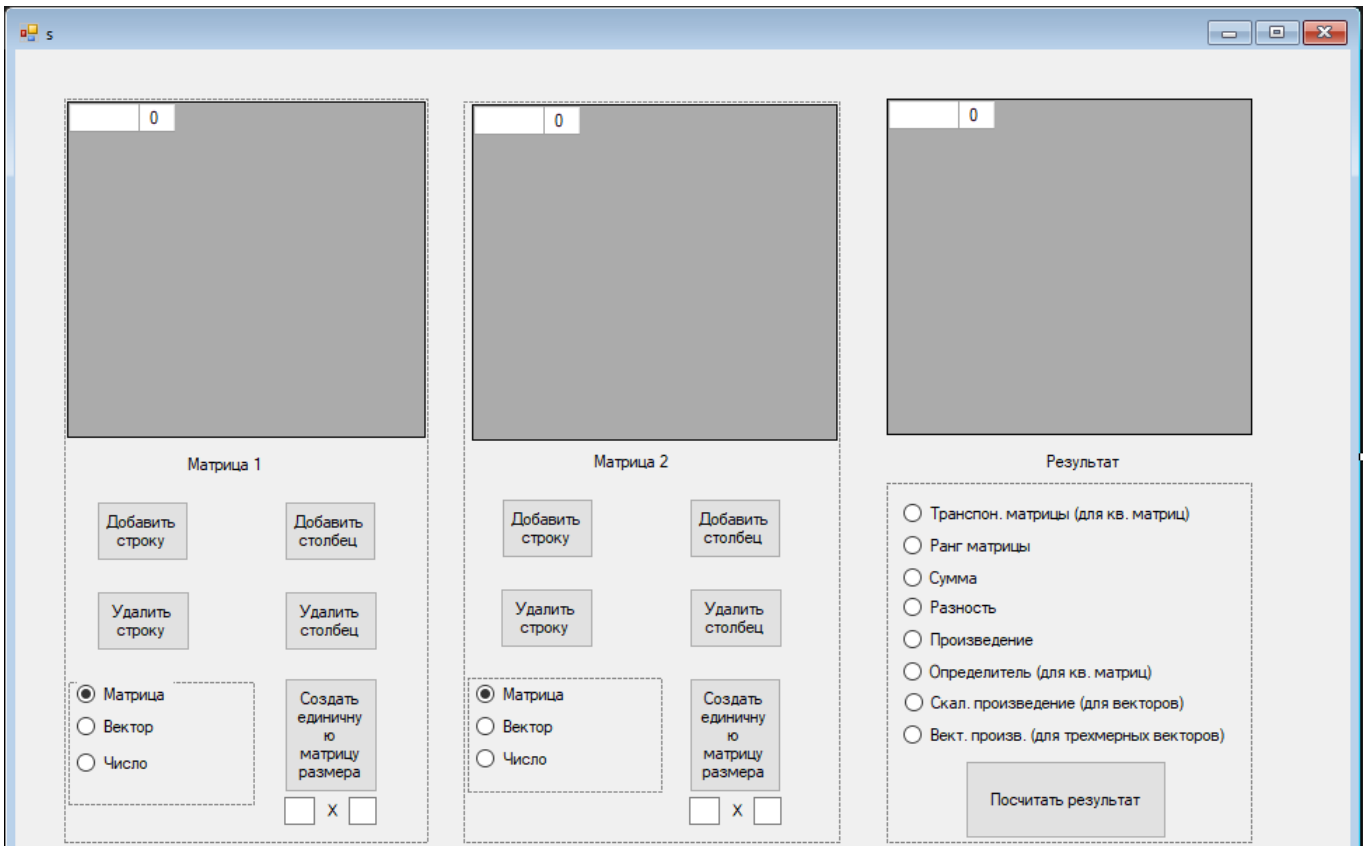

Рисунок 19 – Внешний вид формы программы

У элементов изменены значения некоторых атрибутов. Значения измененных атрибутов представлены в таблице 6.

Наименование атрибута	Значение
Для формы	
Text	Браузер Амиго
FormBorderStyle	FixedSingle
MaximizeBox	False
Для первой группы	
(Name)	MatrixGroup1
Для второй группы	
(Name)	MatrixGroup2
Для третьей группы	
(Name)	Result_Group
Для надписи первой матрицы	
(Name)	initLabel
Text	Матрица 1
Для надписи второй матрицы	

(Name)	xLabel
Text	Матрица 2
Для третьей надписи	
(Name)	resultLabel
Text	Результат
Для радиокнопки "Матрица"для группы 1	
(Name)	Grid1_MatrixRBtn
Text	Матрица
Для радиокнопки "Вектор"для группы 1	
(Name)	Grid1_VectorRBtn
Text	Вектор
Для радиокнопки "Число"для группы 1	
(Name)	Grid1_NumRBtn
Text	Число
Для радиокнопки "Матрица"для группы 2	
(Name)	Grid2_MatrixRBtn
Text	Матрица
Для радиокнопки "Вектор"для группы 2	
(Name)	Grid2_VectorRBtn
Text	Вектор
Для радиокнопки "Число"для группы 2	
(Name)	Grid2_NumRBtn
Text	Число
Для радиокнопки транспонирования для группы 3	
(Name)	Transposition_RBtn
Text	Транспон. матрицы (для кв. матриц)
Для радиокнопки ранга матрицы для группы 3	
(Name)	Rank_RBtn
Text	Ранг матрицы
Для радиокнопки суммы для группы 3	
(Name)	Sum_RBtn
Text	Сумма
Для радиокнопки разности для группы 3	

(Name)	Difference_RBtn
Text	Разность
Для радиокнопки произведения для группы 3	
(Name)	Mult_RBtn
Text	Произведение
Для радиокнопки определителя для группы 3	
(Name)	Det_RBtn
Text	Определитель (для кв. матриц)
Для радиокнопки скалярного произведения для группы 3	
(Name)	ScalarMultiply_RBtn
Text	Скалярное произведение (для векторов)
Для радиокнопки векторного произведения для группы 3	
(Name)	VectorMultiply_RBtn
Text	Векторное произведение (для трехмерных векторов)
Для кнопки создания единичной матрицы для группы 1	
(Name)	CreateMatrix1_Btn
Text	Создать единичную матрицу размера
Для кнопки создания единичной матрицы для группы 2	
(Name)	CreateMatrix2_Btn
Text	Создать единичную матрицу размера
Для знака размерности для группы 1	
(Name)	Scale_label1
Text	X
Для знака размерности для группы 2	
(Name)	Scale_label2
Text	X
Для текстового поля N размерности для группы 1	
(Name)	N1_TextBox
Для текстового поля M размерности для группы 1	
(Name)	M1_TextBox
Для текстового поля N размерности для группы 2	
(Name)	N2_TextBox
Для текстового поля M размерности для группы 2	

(Name)	M2_TextBox
Для кнопки "Добавить строку"для группы 1	
(Name)	Grid1_btnAddRow
Text	Добавить строку
Для кнопки "Добавить столбец"для группы 1	
(Name)	Grid1_btnAddColumn
Text	Добавить столбец
Для кнопки "Удалить строку"для группы 1	
(Name)	Grid1_btnRmvRow
Text	Удалить строку
Для кнопки "Удалить столбец"для группы 1	
(Name)	Grid1_btnRmvColumn
Text	Удалить столбец
Для кнопки "Добавить строку"для группы 2	
(Name)	Grid2_btnAddRow
Text	Добавить строку
Для кнопки "Добавить столбец"для группы 2	
(Name)	Grid2_btnAddColumn
Text	Добавить столбец
Для кнопки "Удалить строку"для группы 2	
(Name)	Grid2_btnRmvRow
Text	Удалить строку
Для кнопки "Удалить столбец"для группы 2	
(Name)	Grid2_btnRmvColumn
Text	Удалить столбец
Для кнопки добавления ряда	
(Name)	btnAddRow
Text	Добавить
Для кнопки удаления ряда	
(Name)	btnRemoveRow
Text	Удалить
Для кнопки добавления столбца	
(Name)	btnAddColumn

Text	Добавить
Для кнопки удаления столбца	
(Name)	btnRemoveColumn
Text	Удалить
Для обработчика ошибок	
(Name)	errorProvider1

Таблица 6 – Значения атрибутов элементов в приложении «Матричный калькулятор

Программа работает за счет дополнительно написанной библиотеки для работы с матрицами с помощью `std::vector` библиотеки STL. В ней реализованы все операции. Код этой работы содержится в приложении А [6–9].

Кроме того, написаны две вспомогательный функции, представляющие из себя связки между `std::vector` и `DataGridView`. Код приведен ниже:

```
#pragma once
#include <vector>
#include "MyForm.h"

using std::vector;
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;

//template<typename T>
vector<vector<int>> GetVectorFromGrid(DataGridView^ grid) { // Функция получения вектора из таблицы
        vector<vector<int>> vec(grid->RowCount, vector<int>(grid->ColumnCount));
        for (int i = 0; i < vec.size(); ++i) {
                for (int j = 0; j < vec[i].size(); ++j) {
                        int Value = System::Convert::ToInt32(grid->Rows[i]->Cells[j]->Value);
                        vec[i][j] = Value;
                }
        }
        return vec;
```

```
}

//template<typename T>
void FillGridWithVector(vector<vector<int>>& vec, DataGridView^ grid) { // Функция заполнения таблицы вектором
        int ColumnsAmount = grid->Columns->Count;
        int RowsAmount = grid->Rows->Count;
        if (RowsAmount != vec.size() || ColumnsAmount != vec[0].size())
         return;
        for (int i = 0; i < RowsAmount; ++i) {
                for (int j = 0; j < ColumnsAmount; ++j) {
                        grid->Rows[i]->Cells[j]->Value = vec[i][j];
                }
        }
};
```

Работа программы продемонстрирована на рисунках 20,21,22

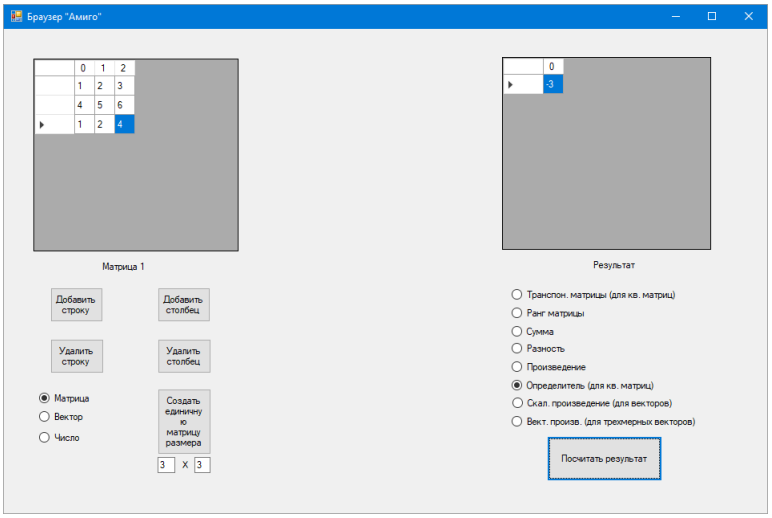

Рисунок 20 – Пример нахождения определителя матрицы

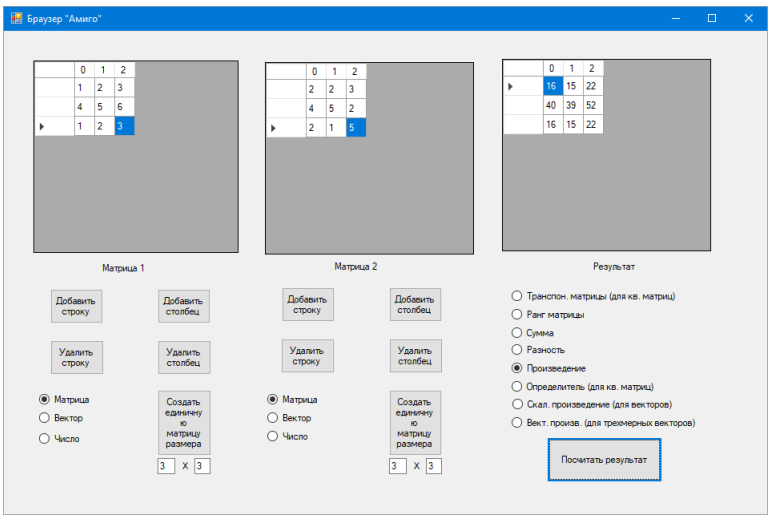

Рисунок 21 – Пример нахождения векторного произведения

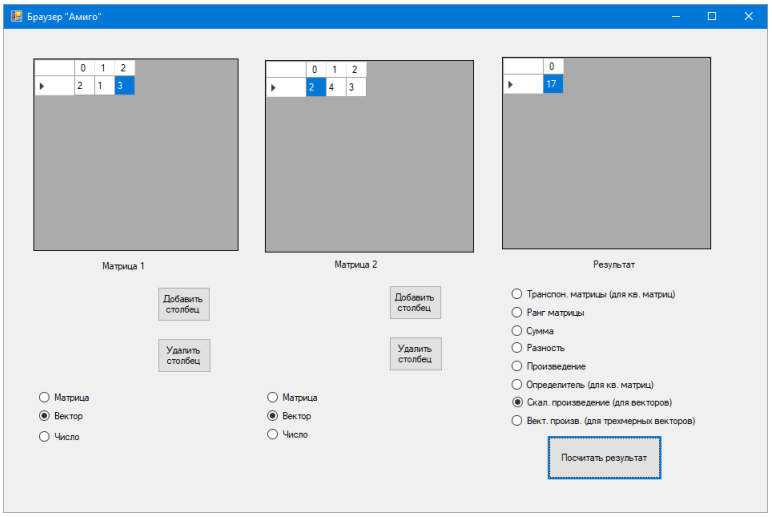

Рисунок 22 – Пример нахождения скалярного произведения векторов

Можно заметить, что интерфейс приложения динамически изменяется в зависимости от выбранных параметров в соответствующих радиокнопках [10].

Контроль за корректностью введенных данных осуществляется внутри соответствующих функций с помощью элемента `ErrorProvider`.

В случае, если вводятся некорректные данные или операция не поддерживается над операндами этого типа — будет выведено сообщение об ошибке (см. рисунок 23)

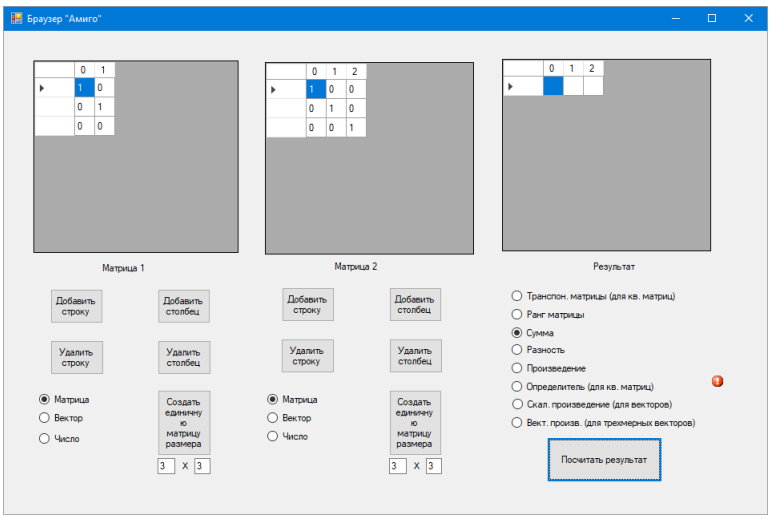

Рисунок 23 – Пример обработки ошибки

Полный код программы приведен в приложении А.

7 Использование коллекций

Задание: Создать словарь, состоящий из строк. В качестве ключа выступает фамилия, в качестве значения — должность. Вывести на экран фамилии людей, занимающих данную должность. Вывести должность, занимаемую данным человеком. Вид окна представлен на рисунке 24.

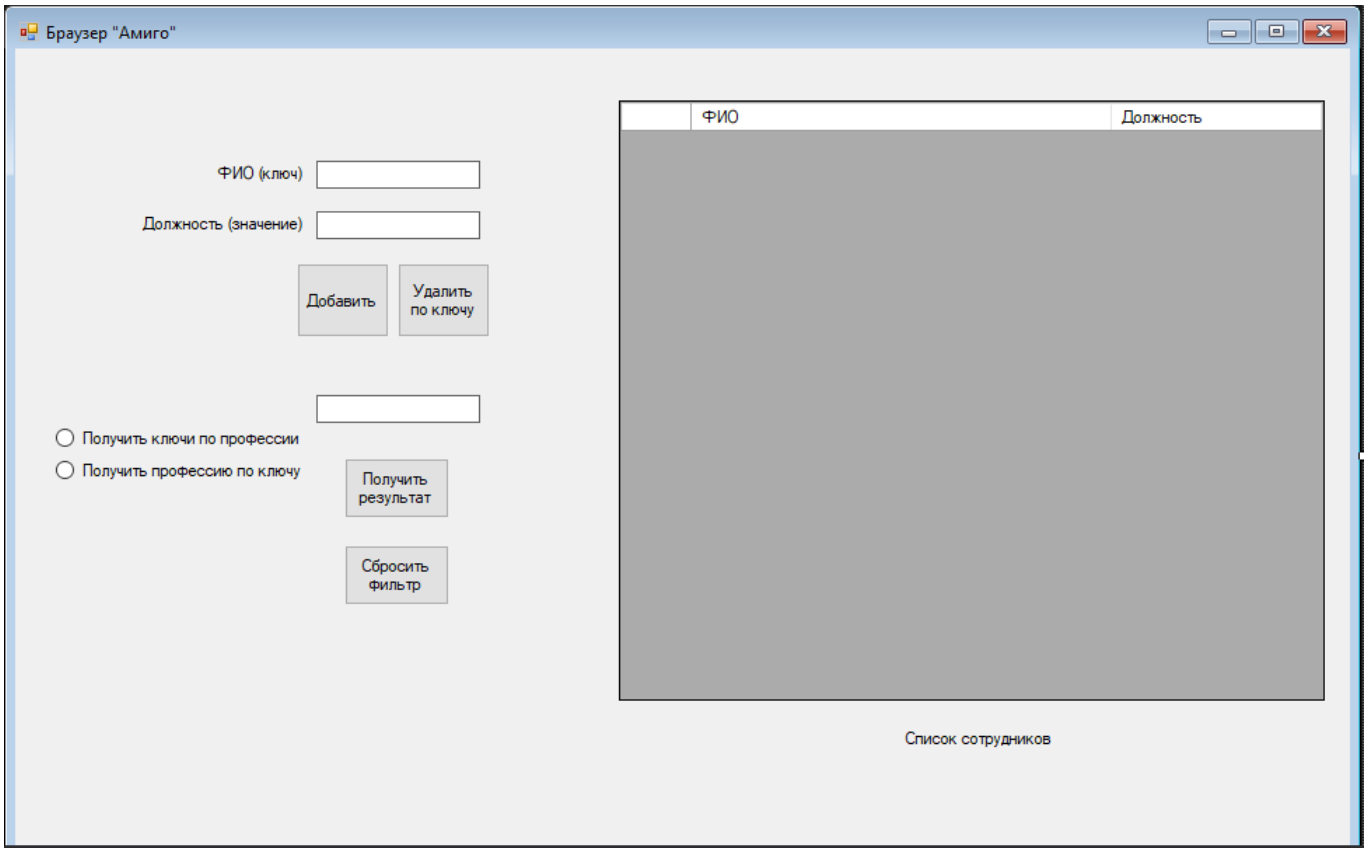

Рисунок 24 – Внешний вид формы программы

У элементов изменены значения некоторых атрибутов. Значения измененных атрибутов представлены в таблице 7.

Наименование атрибута	Значение
Для формы	
`Text`	`Браузер Амиго`
`FormBorderStyle`	`FixedSingle`
`MaximizeBox`	`False`
Для надписи ФИО	
`(Name)`	`NameLabel`
`Text`	`ФИО (ключ)`
Для текстового поля должности	

(Name)	PositionTextBox
Для текстового поля ФИО	
(Name)	NameTextBox
Для надписи должности	
(Name)	PositionLabel
Text	Должность(значение)
Для надписи к таблице	
(Name)	resultLabel
Text	Список сотрудников
Для кнопки добавления значения по ключу	
(Name)	AddBtn
Text	Добавить
Для кнопки удаления значения по ключу	
(Name)	RemoveBtn
Text	Добавить
Для текстового поля результата	
(Name)	ResultTextBox
Для радиокнопки ключа по профессии	
(Name)	GetNamesBtn
Text	Получить ключи по профессии
Для радиокнопки профессии по ключу	
(Name)	GetPositionBtn
Text	Получить профессию по ключу
Для кнопки получения результата	
(Name)	ResultBtn
Text	Получить результат
Для кнопки сброса	
(Name)	ResetBtn
Text	Сбросить фильтр
Для обработчика ошибок	
(Name)	errorProvider1

Таблица 7 – Значения атрибутов элементов в приложении «Матричный калькулятор

Программа написана с использованием контейнеров из .NET framework и работы с ними соответственно назначению кнопки. Ниже приведен пример работы функции добавления элемента по ключу:

```
private: System::Void AddBtn_Click(System::Object^ sender, System::EventArgs^
↪  e) { // Добавление пары ключ - значение
        String^ name = NameTextBox->Text->ToString();
        String^ position = PosTextBox->Text->ToString();
        if (name == String::Empty || position == String::Empty) { // Если
        ↪  что-то не ввели - сообщим об этом
                errorProvider1->SetError(AddBtn, "Недопустимые значения");
                return;
        }
        if (!d.ContainsKey(name)) { // Если ключа нет, добавим
                d.Add(name, position);
                System::Collections::Generic::List<String^>^ lst = gcnew
                ↪  System::Collections::Generic::List<String^>(); // Создадим
                ↪  новый объект
                if (!p.ContainsKey(position))
                        p.Add(position, lst);
                p[position]->Add(name); // Добавим в лист по этому ключу новое
                ↪  значение
                gridResult->Rows->Add(1);
                int _row = gridResult->RowCount;
                auto ResultPair =
                ↪  System::Collections::Generic::KeyValuePair<String^,
                ↪  String^>(name, position);
                FillRowWithDict(gridResult->Rows[_row - 1], ResultPair);
                }
        }
};
```

После запуска приложения открывается следующее окно (см. рисунок 25)

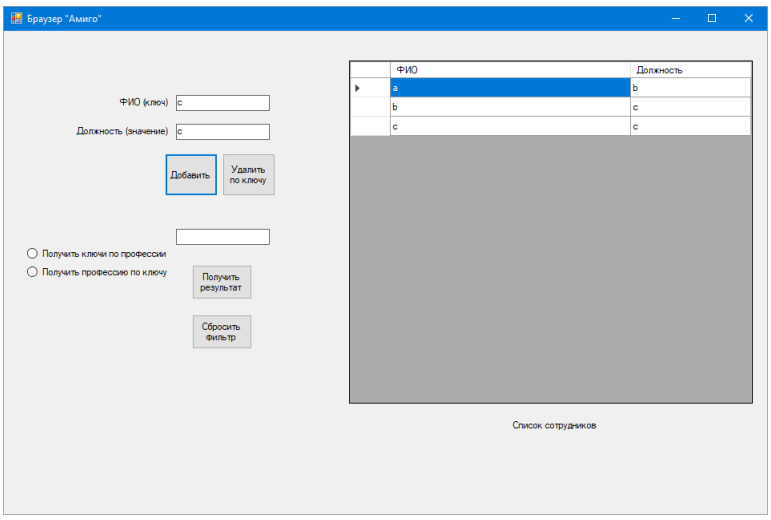

Рисунок 25 – Состояние приложения после добавление нескольких элементов по ключу

Ниже приведен пример работы программы: После запуска приложения открывается следующее окно (см. рисунки 26,27)

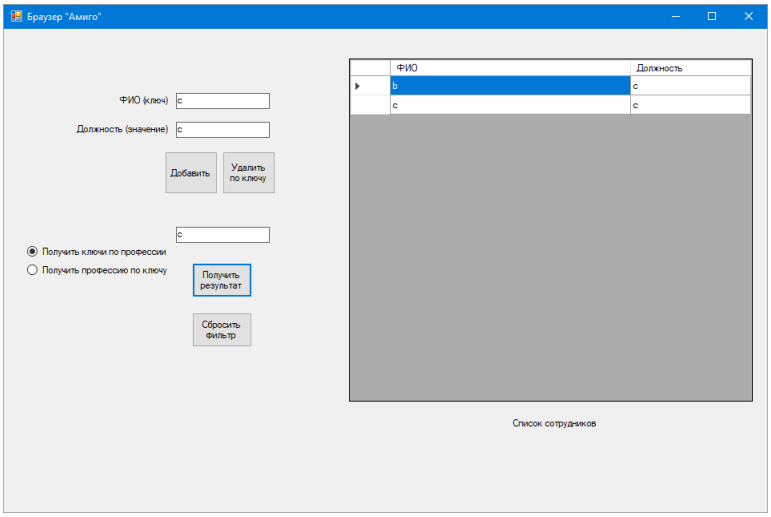

Рисунок 26 – Результат поиска по значению

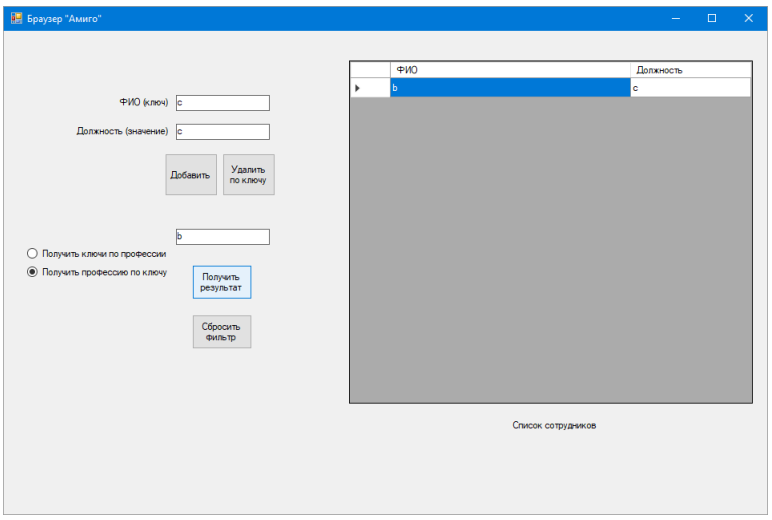

Рисунок 27 – Результат поиска по ключу

В случае, если соответствующие поля не заполнены, выполнение программы игнорируется.

Полный код программы приведен в приложении А.

8 Файловые диалоги и работа с файлами

Задание: Создать таблицу Work. В другой файл вывести данные о рабочих, занимающих данную должность. (Вариант 14)

Вид окна представлен на рисунке 28.

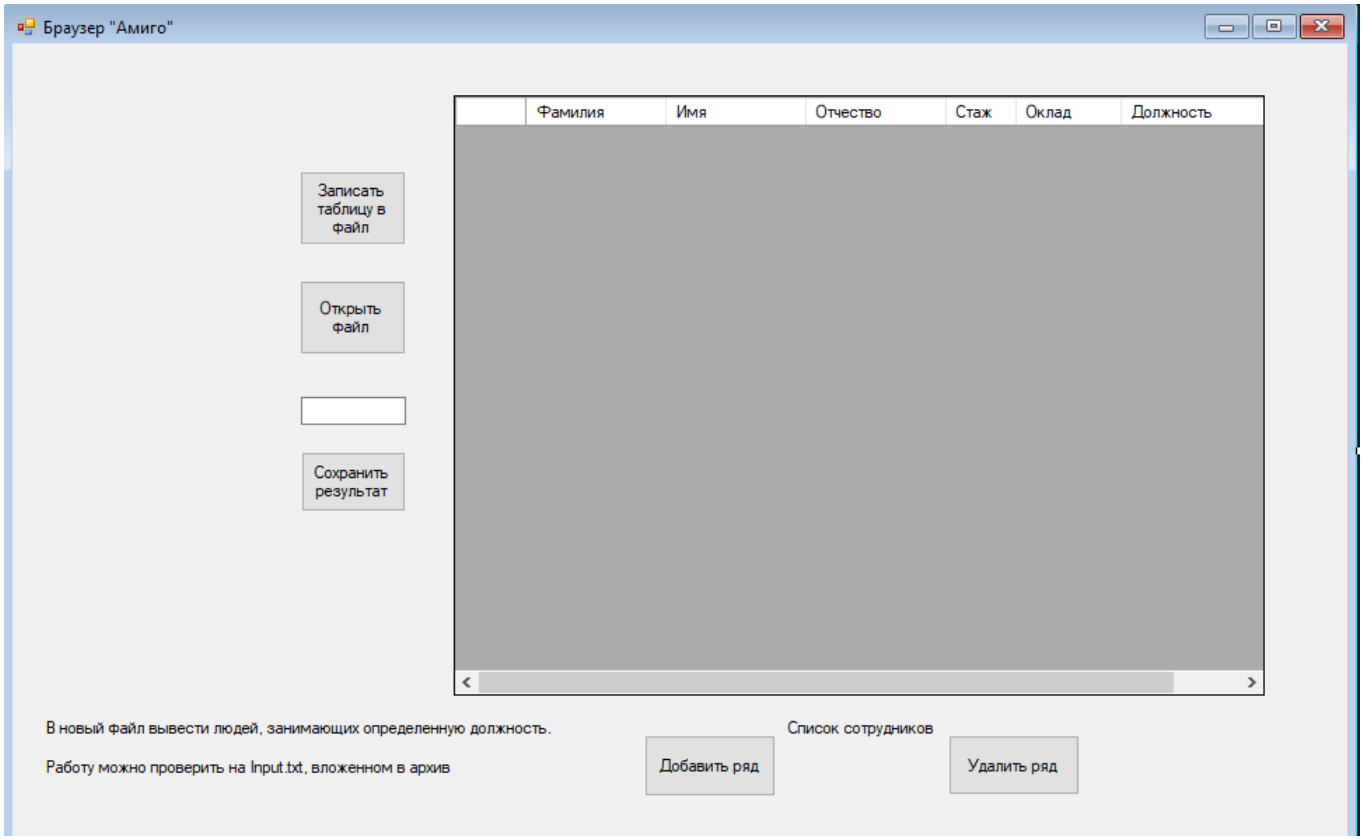

Рисунок 28 – Внешний вид формы программы

У элементов изменены значения некоторых атрибутов. Значения измененных атрибутов представлены в таблице 8.

Наименование атрибута	Значение
Для формы	
`Text`	`Браузер Амиго`
`FormBorderStyle`	`FixedSingle`
`MaximizeBox`	`False`
Для таблицы	
`(Name)`	`gridResult`
Для кнопки записи в файл	
`(Name)`	`SaveFileBtn`
Для кнопки открытия файла	

(Name)	OpenFileBtn
Для кнопки сохранения результата	
(Name)	ResultBtn
Для текстового поля должности	
(Name)	ResultTextBox
Для обработчика ошибок	
(Name)	errorProvider1

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

Кроме того, это приложение содержит элементы openFileDialogue и saveFileDialogue, реализующие открытие и сохранение файлов.

Работа с ними производится в кнопках. Ниже приведен пример работы функции открытия файла:

```
private: System::Void OpenFileBtn_Click(System::Object^ sender, System::EventArgs^ e) {
        System::IO::Stream^ myStream;
        if (this->openFileDialog->ShowDialog() == System::Windows::Forms::DialogResult::OK)
        ↪  {
                CreateEmptyMatrix(gridResult, 1);
                if ((myStream = openFileDialog->OpenFile()) != nullptr) {
                        System::IO::StreamReader^ sw = gcnew
                        ↪  System::IO::StreamReader(myStream, System::Text::Encoding::
                        GetEncoding(65001)); // UTF-8
                        System::String^ s = "";
                        int innerIdx = 0;
                        while ((s = sw->ReadLine()) != nullptr && s != "") {
                                gridResult->Rows->Add(1);
                                int idx = 0;
                                int current = 0;
                                while (idx != s->Length) {
                                        System::String^ currentWord = "";
                                        while (idx < s->Length && s[idx] != ' ') {
                                                currentWord += s[idx++];
                                        }
                                        if (idx < s->Length && s[idx] == ' ') ++idx;
                                        gridResult->Rows[innerIdx]->Cells[current++]->Value
                                        ↪  = currentWord;
                                        currentWord = "";
                                }
                                ++innerIdx;
                        }
                sw->Close();
                }
```

```
27              }
28  }
```

После запуска программы на экране появляется окно следующего вида (см. рисунок 29)

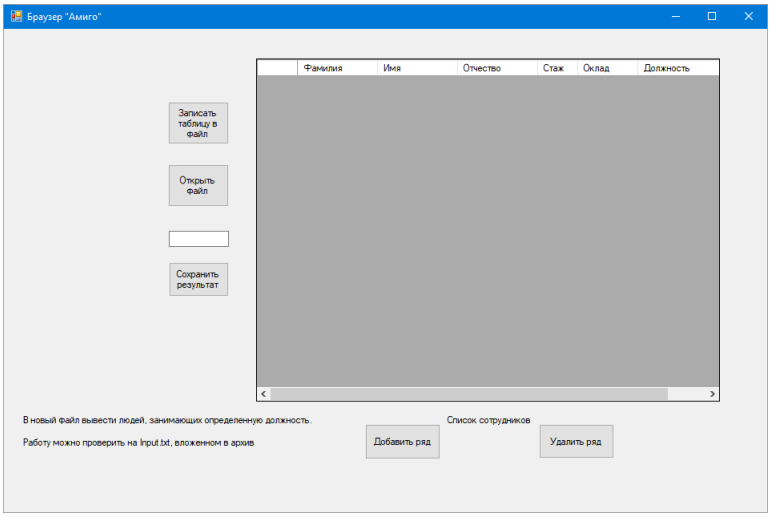

Рисунок 29 – Внешний вид окна приложения

После открытия файла состояние программы изменится на следующее (см. рисунок 30)

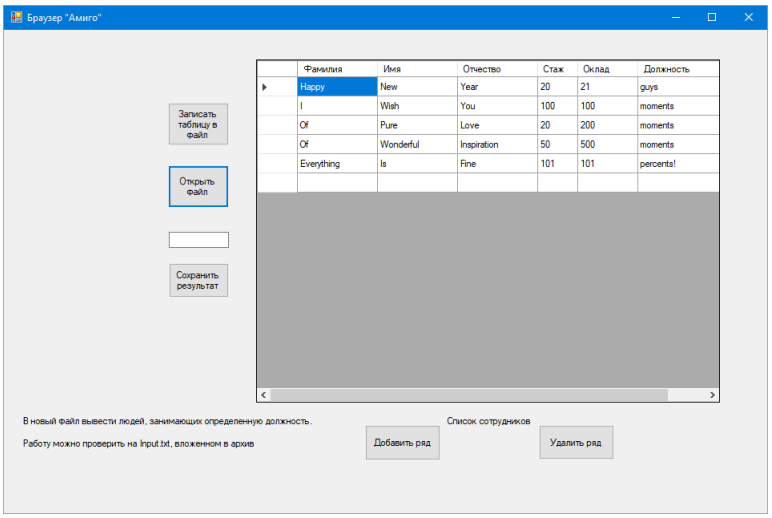

Рисунок 30 – Состояние программы после открытия файла

Результатом работы программы является новый файл (см. рисунок 31)

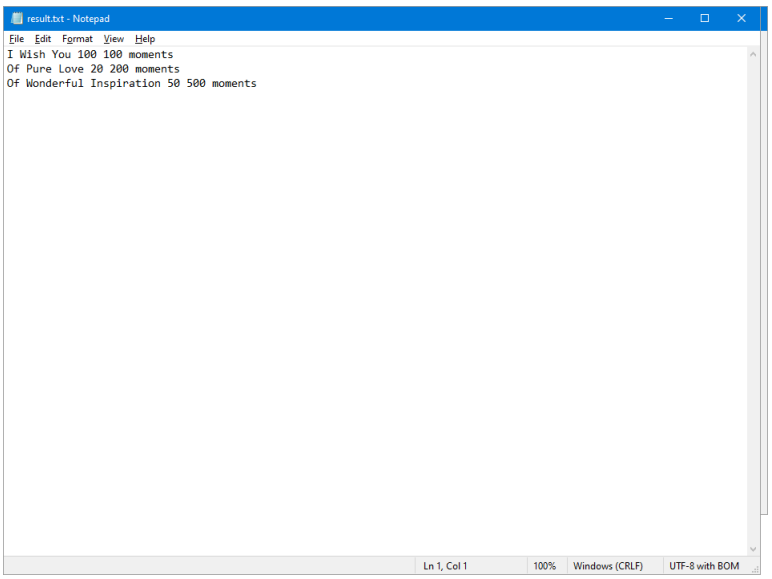

Рисунок 31 – Результат работы программы

Возникновение исключительных ситуаций ограничено интерфейсом, постановкой задачи и программными средствами `.NET framework`.

Полный код программы приведен в приложении А.

9 Приложение «Тест»

Задание: Создать приложение для проведения тестирования. Оно должно содержать:

1. Набор вопросов по какой-то теме (и вопросы и ответы должны быть реальные) -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.

Вид окна представлен на изображениях 32,33.

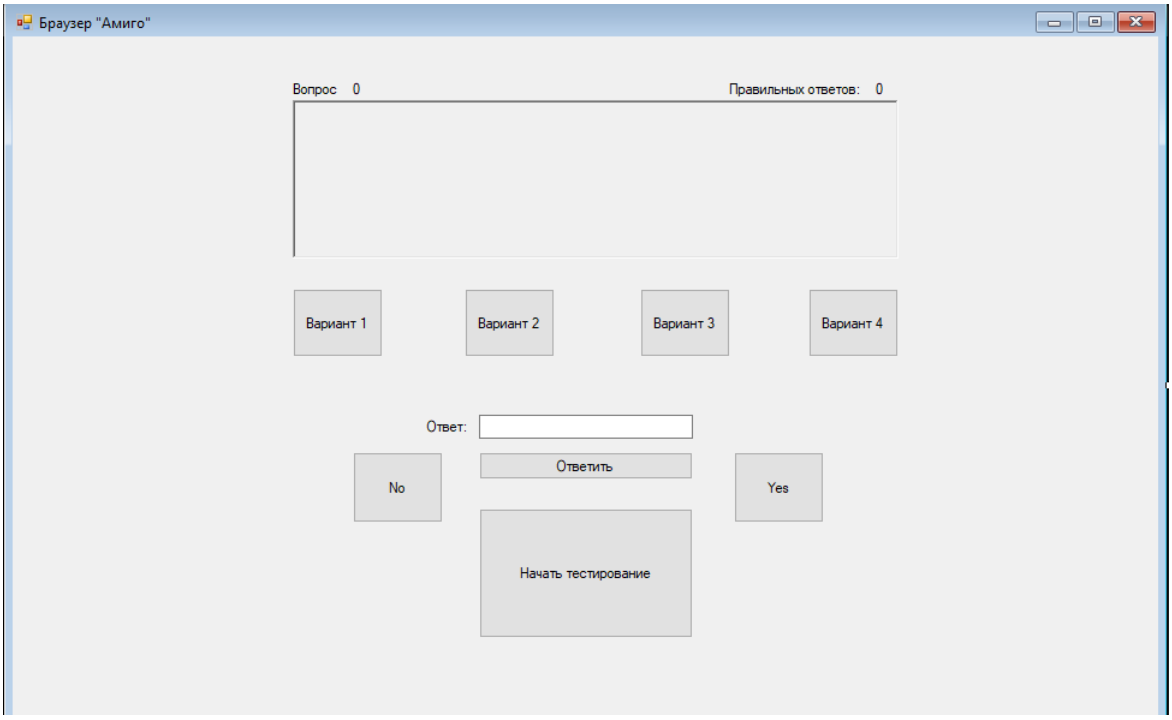

Рисунок 32 – Внешний вид формы 1

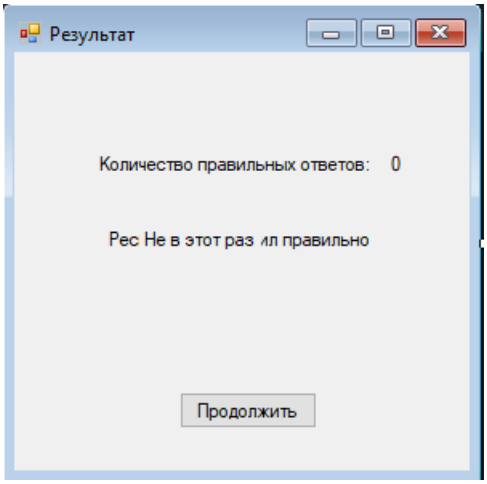

Рисунок 33 – Внешний вид формы 2

У элементов изменены значения некоторых атрибутов. Значения измененных атрибутов представлены в таблице 9.

Наименование атрибута	Значение
Для формы	
Text	Браузер Амиго
FormBorderStyle	FixedSingle
MaximizeBox	False
Для кнопки варианта 1	
(Name)	Option1Btn
Text	Вариант 1
Для кнопки варианта 2	
(Name)	Option2Btn
Text	Вариант 2
Для кнопки варианта 3	
(Name)	Option3Btn
Text	Вариант 3
Для кнопки варианта 4	
(Name)	Option4Btn
Text	Вариант 4
Для кнопки "Да"	
(Name)	YesBtn
Text	Да
Для кнопки "Нет"	

(Name)	NoBtn
Text	Нет

Таблица 9 – Значения атрибутов элементов в приложении «Приложение «Тест» »

После запуска приложения появляется окно (см. рисунок 34)

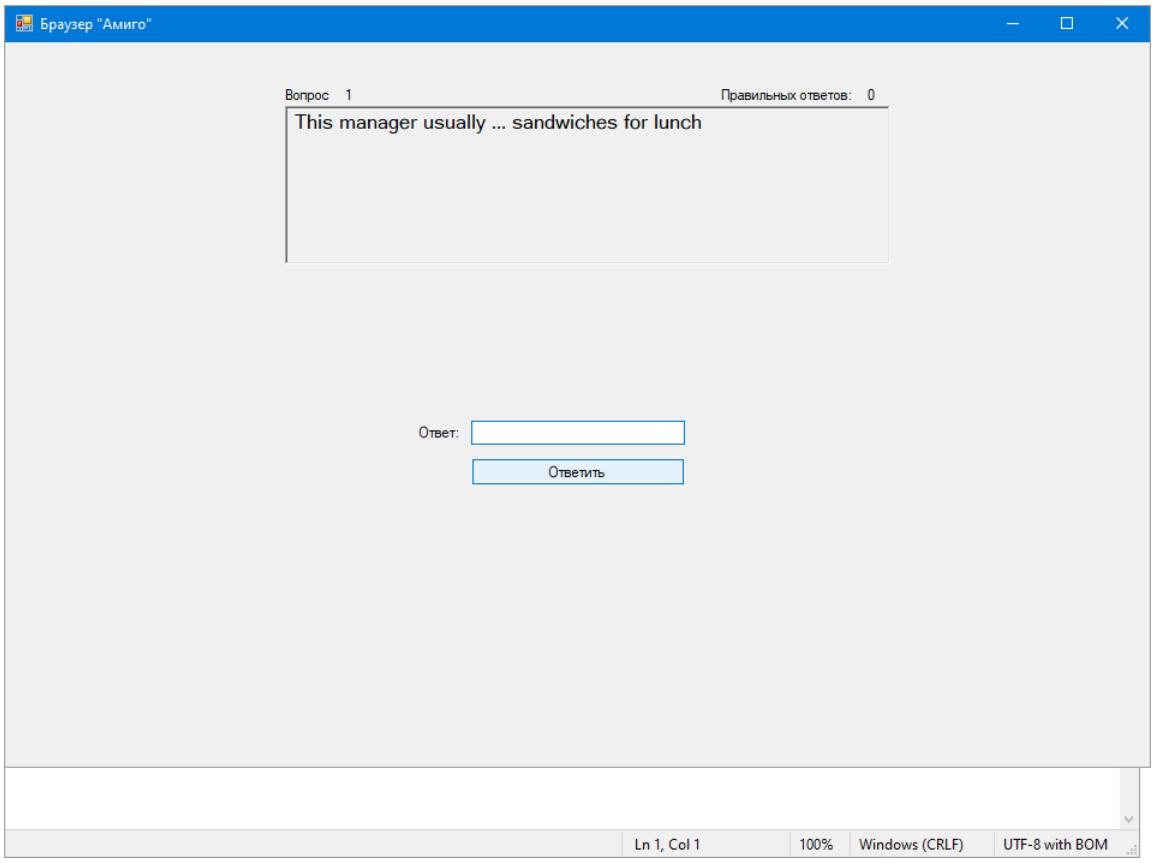

Рисунок 34 – Внешний вид окна после запуска приложения

Приложение содержит различные виды вопросов. Например, окно для вопросов с двумя вариантами ответа выглядит следующим образом (см. рисунок 35)

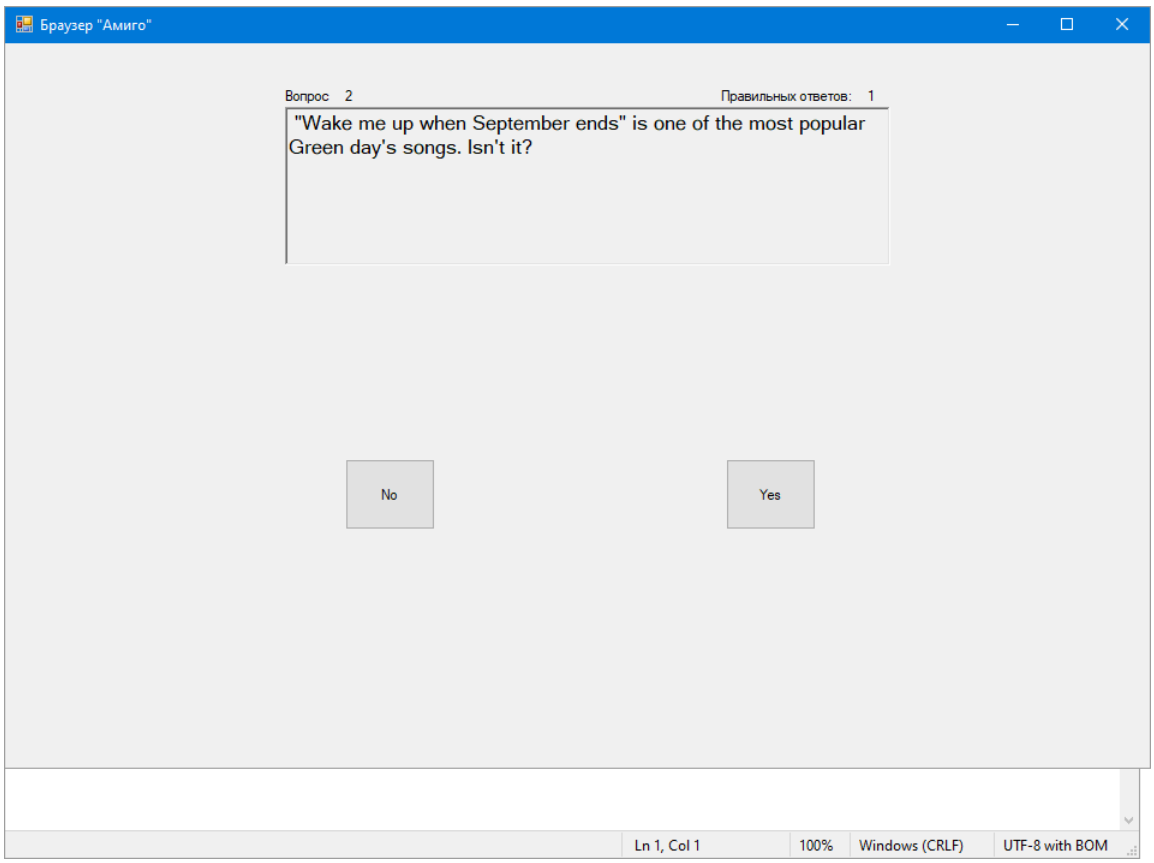

Рисунок 35 – Внешний вид для вопросов с двумя вариантами ответа

В коде приложения была создана структура наследовования, которая упрощает работу с различными вариантами вопросов:

```
ref struct Question {
        public: int questionId;
        public: Question() {};
        public: Question(int questionId, String^ text) :
        ↪  questionId(questionId), text(text) {};
        public: String^ text;
        public: virtual bool CheckAnswer() = 0;
        public: virtual Question^ Clone() = 0;
        };

ref struct ShortAnswerQuestion : Question {
        public: String^ expectedAnswer;
        public: String^ userAnswer;
        public: ShortAnswerQuestion() {};
        public: ShortAnswerQuestion(int questionId, String^ expectedAnswer) {
                this->questionId = questionId;
                this->expectedAnswer = expectedAnswer;
        }
```

```
18          public: virtual bool CheckAnswer() override {
19                  return expectedAnswer->Equals(userAnswer);
20          }
21          public: virtual Question^ Clone() override {
22                  ShortAnswerQuestion^ obj = (gcnew ShortAnswerQuestion());
23                  obj->expectedAnswer = this->expectedAnswer;
24                  obj->questionId = this->questionId;
25                  obj->text = this->text;
26                  obj->userAnswer = this->userAnswer;
27          Question^ toBeReturned = obj;
28          return toBeReturned;
29          }
30  };
31
32  ref struct SeveralAnswerQuestion : Question {
33          public: int count;
34          public: int userAnswerId;
35          public: int expectedAnswerId;
36          public: virtual bool CheckAnswer() override {
37                  return userAnswerId == expectedAnswerId;
38          }
39          public: SeveralAnswerQuestion() {};
40          public: SeveralAnswerQuestion(int questionId, int expectedAnswer, int
            ↪  count) {
41                  this->questionId = questionId;
42                  this->expectedAnswerId = expectedAnswerId;
43                  this->count = count;
44          }
45          public: virtual Question^ Clone() override {
46                  SeveralAnswerQuestion^ obj = (gcnew SeveralAnswerQuestion());
47                  obj->expectedAnswerId = this->expectedAnswerId;
48                  obj->questionId = this->questionId;
49                  obj->text = this->text;
50                  obj->count = this->count;
51                  Question^ toBeReturned = obj;
52                  return toBeReturned;
53          }
54  };
```

В ходе выполнения программа сообщает пользователю о том, ввел ли он правильный ответ или нет (см. рисунки 36,37)

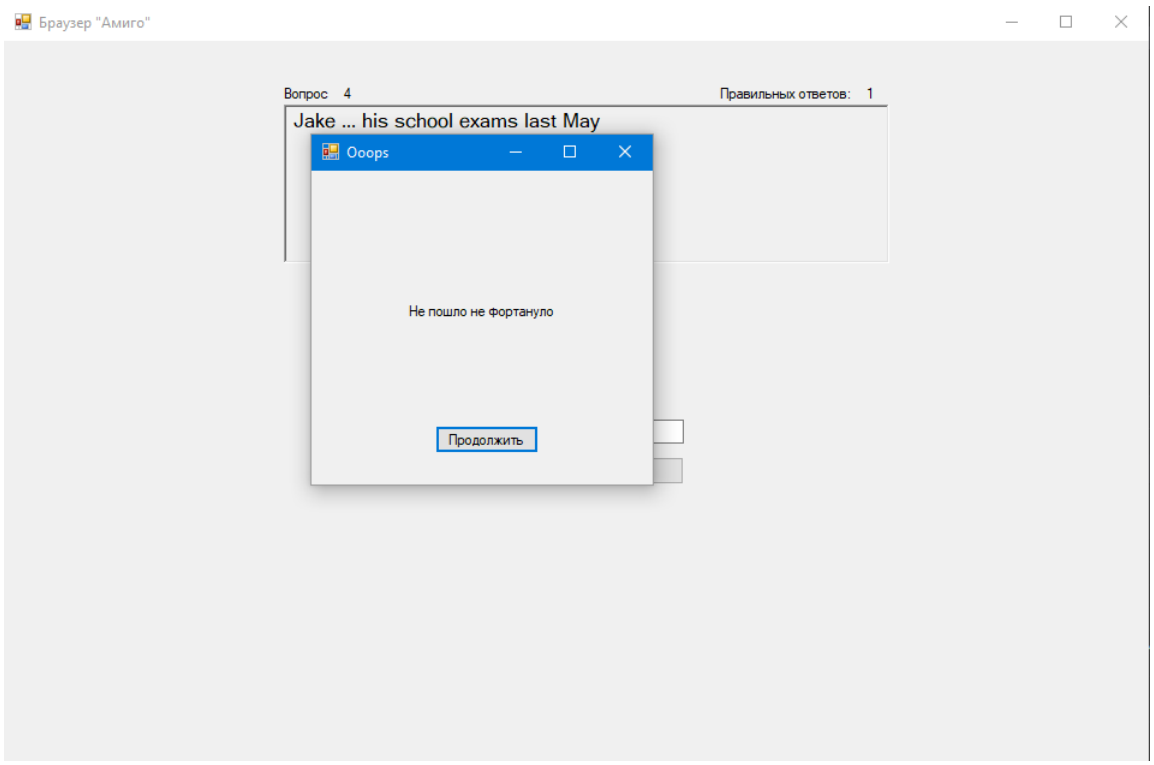

Рисунок 36 – Окно в случае неправильного ответа на вопрос

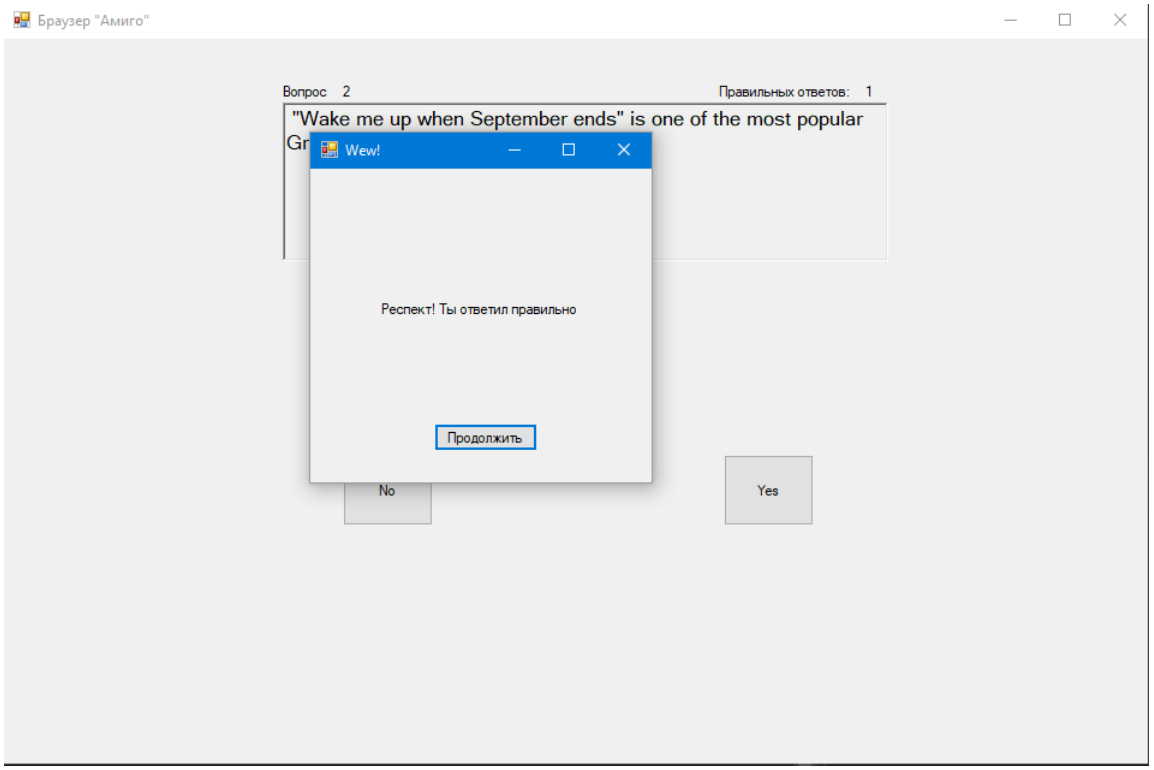

Рисунок 37 – Окно в случае правильного ответа на вопрос

Программа не содержит исключительных ситуаций, поэтому в их обработке нет необходимости.

Полный код программы приведен в приложении А.

ЗАКЛЮЧЕНИЕ

В ходе прохождения практики были получены основы разработки приложений Windows Forms. Были изучены особенности и основные инструменты `.NET framework`.

ПРИЛОЖЕНИЕ А

Репозиторий Github, содержащий полный код программ

https://github.com/WrongWayboyyyy/StudyPracticePublic

`\task1` — «Вычисление факториала»

`\task2` — «Простые вычисления»

`\task3` — «Рекурсивные вычисления»

`\task4` — «Обработка табличных данных. Часть 1»

`\task5` — «Обработка табличных данных. Часть 2»

`\task6` — «Матричный калькулятор»

`\task7` — «Использование коллекций»

`\task8` — «Файловые диалоги и работа с файлами»

`\task9` — «Тест»

`\latex` — Файлы исходного проекта LaTeX

`practice.pdf` — PDF-файл отчета

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Создание приложения Windows Forms с помощью .NET Framework (C++) [Электронный ресурс]. — URL: http://msdn.microsoft.com/ru-ru/library/vstudio/ms235634(v=vs.100).aspx (Дата обращения 10.01.2021). Загл. с экр. Яз. рус.

2 Конструктор Windows Forms [Электронный ресурс]. — URL: https://msdn.microsoft.com/ru-ru/library/e06hs424.aspx (Дата обращения 10.01.2021). Загл. с экр. Яз. рус.

3 *Meyers, S.* Effective STL / S. Meyers. — Boston: Williams, 2011.

4 *Нойес, Б.* Data Binding with Windows Forms 2.0: Programming Smart Client Data Applications with .NET (Microsoft .Net Development Series) / Б. Нойес. — Москва: ИЛ, 2006.

5 *Конова, Е.* Алгоритмы и программы. Язык C++. / Е. Конова. — СПб: Издательство «Лань», 2017.

6 *Эстербю, О.* Прямые методы для разреженных матриц / О. Эстербю. — Москва: Мир, 1987.

7 *Пахомов, Б.* C/C++ и MS Visual C++ / Б. Пахомов. — СПб: БХВ-Петербург, 2012.

8 *Седжвик, Р.* Алгоритмы на C++ / Р. Седжвик. — Москва: Диалектика, 2011.

9 *Horton's, I.* Beginning Visual C++ 2010 / I. Horton's. — Москва: Wrox, 2010.

10 *Панюкова, Т.* Языки и методы программирования. Создание простых GUI-приложений с помощью Visual C++ / Т. Панюкова. — Москва: URSS, 2013.