Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное   
образовательное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

ИНСТИТУТ ЭЛЕКТРОННЫХ И ИНФОРМАЦИОННЫХ СИСТЕМ

КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

Лабораторная Работа №1. Рекурсия

Отчет по лабораторной работе №1 по учебной дисциплине «Алгоритмические языки и программирование»

по специальности 09.03.01 Информатика и вычислительная техника

|  |  |
| --- | --- |
|  | Руководитель  / И. Ю. Кулаков  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |
|  | Студент группы 3092  / Н. Д. Хорев  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

1. **Постановка задачи**

Разработать программу с интерфейсом для работы с рекурсиями.

* 1. **Наименование задачи** – Лабораторная работа №1. Рекурсия.
  2. **Словесное описание:**

Разработать программу с графическим интерфейсом для работы с рекурсиями:

1. По формулам 9 и 16:

9: F(n) = 1, при n = 1;

F(n) = n + 2 \* F(n - 1) , если n чётно;

F(n) = 1 + 3 \* F(n - 2), если n > 1 и при этом n нечётно.

16: F(n) = 1, при n = 1;

F(n) = 2, при n = 2;

F(n) = [(8\*n + F(n - 3))/9], если n > 2 и при этом n четно

F(n) = [(4\*n + F(n - 1) + F(n - 2))/7], если n > 2 и при этом n нечетно

1. Дополнительная программа (9 вариант): Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом 0. Выведите первое, третье, пятое и т.д. из введенных чисел. Завершающий ноль выводить не надо.  
   В этой задаче нельзя использовать глобальные переменные и передавать какие-либо параметры в рекурсивную функцию. Функция получает данные, считывая их с клавиатуры. Функция не возвращает значение, а сразу же выводит результат на экран. Основная программа должна состоять только из вызова этой функции.

Программа должна содержать поля ввода, вывода значений, очистки полей, выбора рекурсивной формулы (1 программа), старта программы, информацию о программе.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Объект программы | Имя этого объекта в программе | Тип данных | Диапазоны представления | Простая/  структура | Вход/  выход/  Константа |
| Значения, введённые пользователем в поле ввода 1-й программы | N | unsigned int | 1 – 70 (рекурсия 1)  1 – 21187 (рекурсия 2) | Простая | Вход |
| Значения, введённые пользователем в поле ввода 2-й программы | num | int | 0-999999 | Простая | Вход |

**Внешние спецификации данных**

**1.3. Внешние спецификации функций программы**

Программа должна обрабатывать введённые значения и исключать ненужные (нечисловые значения, числа через пробел), ограничивать ввод количества символов во избежания ошибки, обеспечивать работу кнопок запуска, очистки полей, вывода окна с информацией о программе

Описываются функциональные требования к разрабатываемой программе и ее функциональные возможности.

**1.4. Математическая формулировка задачи**

Для вводимых значений используется тип данных unsigned int, так как отрицательные значения приводят к ошибке работы программы. Математической моделью являются формулы ниже, которые должны быть представлены в виде рекурсивных функций(1-2 в первой программе, 3 в дополнительной):  
1) F(n) = 1, при n = 1;

F(n) = n + 2 \* F(n - 1) , если n чётно;

F(n) = 1 + 3 \* F(n - 2), если n > 1 и при этом n нечётно.

2) F(n) = 1, при n = 1;

F(n) = 2, при n = 2;

F(n) = [(8\*n + F(n - 3))/9], если n > 2 и при этом n четно

F(n) = [(4\*n + F(n - 1) + F(n - 2))/7], если n > 2 и при этом n нечетно  
3) Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом 0. Выведите первое, третье, пятое и т.д. из введенных чисел.

Первая программа:

* Ограничение для 1-й рекурсивной функции от 1 до 70 выбран для того, чтобы обеспечить достаточную скорость работы программы, поскольку значения выше 70 значительно замедляют работу программы вплоть до полной её остановки и закрытия.
* Ограничение для 2-й рекурсивной функции от 1 до 20000 обусловлена тем, что, превышая 21187 программа принудительно закрывается, а значение ниже выбрано для того, чтобы на других устройствах программа работала, а не закрывалась.

Вторая программа:

* Ограничение для длины вводимых символов в 6 обусловлена тем, чтобы выводимые числа в поля вывода не заходили за рамки и для того, чтобы не выдавало ошибку «необработанное исключение».
* Ограничение для количества строк в 6 сделано по той же причине, что и для ввода символов в поле ввода.

**1.5. Спецификации интерфейса**

Для создания приложения была использована библиотека Windows Forms, которая позволила быстро создать графический интерфейс приложения и так же быстро написать поведения для расставленных элементов.

Реализован контролируемый ввод значений, который не даёт вводить некорректные значения для данной программы, возможность выбора рекурсивных функций, возможность очистки полей ввода и вывода.

Наличие кнопки «Информация» позволяет ознакомиться с особенностями программы и узнать ограничения на вводимые значения.

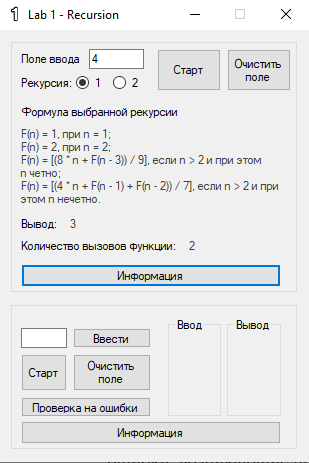


Рисунок 1 Интерфейс приложения

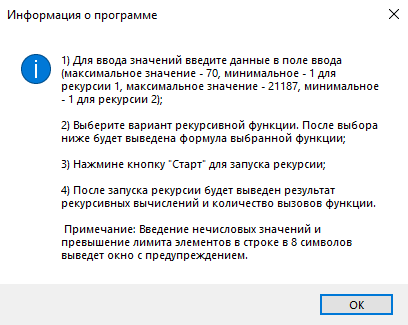


Рисунок 2 Информация о программе 1

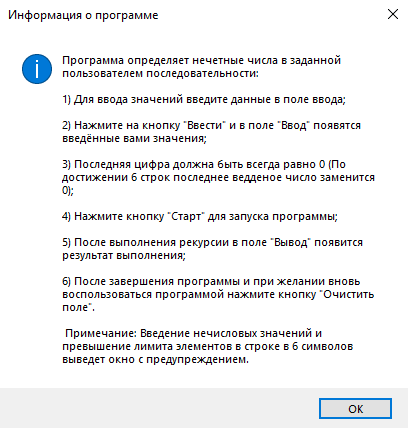


Рисунок 3 Информация о программе 2

* 1. **Внешние данные тестирования(Тест-кейсы)**

Таблица 1. Выход из программы

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | | | T01 |
| Заголовок | | | Проверка выхода из программы |
| Предусловие | | | Программа не запущена |
| Ожидаемый результат | | | Успешный выход из программы |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Запуск 1lab.exe | Открытие главного окна программы | Pass |
| 2 | Нажатие на кнопку «Закрыть» | Выход из программы | Pass |
| Результат | | | Pass |

Таблица 2. Работоспособность кнопки «Старт»

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | | | T02 |
| Заголовок | | | Проверка работы кнопки «Старт» |
| Предусловие | | | Программа  запущена |
| Ожидаемый результат | | | Успешный вывод текста с формулой |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Выбрать одну из кнопок с рекурсией | Вывод текста с формулой рекурсии 1 | Pass |
| 2 | Ввести значение 4 в поле ввода | Ввод корректных данных | Pass |
| 3 | Нажатие на кнопку «Старт» | Вывод результата 3 и количества вызовов рекурсии 2 | Pass |
| Результат | | | Pass |

Таблица 3. Работоспособность кнопки «Очистить поле»

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | | | T03 |
| Заголовок | | | Проверка работы кнопки «Очистить поле» |
| Предусловие | | | Программа  запущена |
| Ожидаемый результат | | | Успешная очистка полей |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Выбрать одну из кнопок с рекурсией | Вывод текста с формулой рекурсии | Pass |
| 2 | Ввести значение в поле ввода | Ввод корректных данных | Pass |
| 3 | Нажатие на кнопку «Старт» | Вывод результата и количества вызовов рекурсии | Pass |
| 4 | Нажатие на кнопку «Очистить поле» | Очистка поля ввода, вывода результата, количества рекурсивных функций, формулы рекурсии | Pass |
| Результат | | | Pass |

Таблица 4. Работоспособность условий

| Идентификатор | | | T04 |
| --- | --- | --- | --- |
| Заголовок | | | Проверка контроля ввода |
| Предусловие | | | Программа запущена |
| Ожидаемый результат | | | Выведена ошибка, возвращение к стандартным значениям |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Ввести некорректное значение | Введено некорректное значение | Pass |
| 2 | Выбрать вариант рекурсии | Выведена формула рекурсии | Pass |
| 3 | Нажатие на кнопку «Старт» | Вывод окна с ошибкой | Pass |
| 4 | Нажатие на кнопку «ОК» в всплывающем окне | Закрытие окна с ошибкой и очистка поля ввода с сброшенным выбором рекурсии | Pass |
| Результат | | | Pass |

Таблица 5. Работоспособность второй программы

| Идентификатор | | | T05 |
| --- | --- | --- | --- |
| Заголовок | | | Работоспособность второй программы |
| Предусловие | | | Программа запущена |
| Ожидаемый результат | | | Корректное выполнение программы (см. п 1.2. - 2)) |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Ввести числа в поле ввода и нажать «Ввести» | Введены значения, ограниченные условием, список заканчивается 0 | Pass |
| 2 | Нажать «Старт» | В поле вывода выведены числа из списка через 1 | Pass |
| Результат | | | Pass |

**1.7. Пример работающей программы** (руководство пользователя)

Показывает скриншоты программы, если реализация может быть непонятной, то пишем руководство.

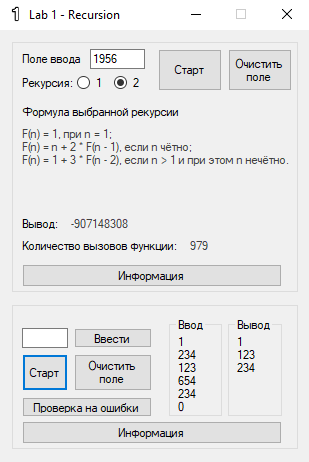


Рисунок 4 Работающая программа

1. **Проектирование задачи** (заполняется от лица программиста)
   1. **Наименование программы**

1lab.exe

* 1. **Уточненное словесное описание задачи**

Разработать программу с графическим интерфейсом для работы с рекурсиями с заданными формулами.

1. Дополнительная программа (9 вариант): Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом 0. Выведите первое, третье, пятое и т.д. из введенных чисел. Завершающий ноль выводить не надо.  
   В этой задаче нельзя использовать глобальные переменные и передавать какие-либо параметры в рекурсивную функцию. Функция получает данные, считывая их с клавиатуры. Функция не возвращает значение, а сразу же выводит результат на экран. Основная программа должна состоять только из вызова этой функции.

Программа должна содержать поля ввода, вывода значений, очистки полей, выбора рекурсивной формулы (1 программа), старта программы, информацию о программе.

* 1. **Уточненное словесное описание задачи**

Написать программу с графическим интерфейсом для работы с рекурсиями по заданным формулам. Программа должна содержать поля ввода, вывода значений, очистка всех полей и флажков, старта программы. Должна обеспечивать корректную работу при помощи различных условий, не допускать некорректные значения: буквы и символы; только цифровые значения. Программа должна ограничивать длину ввода и количество строк для того, чтобы текст не вылезал за пределы.

* 1. **Выбор метода решения поставленной задачи**

Для работы первой программы требуется переносить значения из поля ввода в рекурсивную функцию, в поле ввода любое значение является строкой(тип данных String^), кнопка «Старт» запускает процесс конвертации строки в числовое значение, после чего полученное значение присваивается переменной N, которая в свою очередь является глобальным параметром для 1 и 2 рекурсивных функций и после выполнения программы результат выводится в поле вывода результата работы рекурсии.

Чтобы получить какой-то результат нужно выбрать одну из рекурсивных функций. В ходе выполнения программы и работы рекурсии подсчитывается количество обращений к рекурсивной функции, которые по завершении работы программы проходят тот же путь конвертации и выводятся в поле вывода количества вызовов рекурсивной функции.

Для работы второй программы нужно так же переносить значения из поля ввода в рекурсивную функцию, но прежде, чем она будет работать требуется сделать массив из введённых значений, для этого используется разделение введённых значений на отдельные части при помощи Split. После ввода до заданного количества строк, последним будет в любом случае равно 0 вне зависимости от введённого значения в последнюю строку.

После создания массива из чисел они передаются в рекурсивную функцию, которая в свою очередь имеет два параметра: массив numbers из полученных значений в виде типа данных String^ и index типа int для индексации каждого значения в массиве. Рекурсия проходится по каждому элементу массива, сравнивая значения их индексов, путём их нечётности, если же индекс элемента нечётный, то этот элемент вносится в поле вывода, так же рекурсия проверяет наличие нуля в массиве, если она доходит до него, то заканчивает свою работу. После успешной работы второй программы она выводит в поле вывода полученные в ходе работы рекурсии элементы из введённого изначально массива чисел.

* 1. **Уточненные глобальные данные программы и пользовательские типы**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Объект программы | Имя этого объекта в программе | Характеристики | | | Как используется в программе |
| Тип данных | Диапазоны представления | Простой/структура/ константа |
| MyForm.h | N | unsigned short int | 0-65535 | Простой | Передача введённого числа в параметр рекурсивной функции |
| MyForm.h | count | unsigned int | 0-4294967295 | Простой | Значение вызовов рекурсивной функции |
| MyForm.h | choice\_check | bool | 0-1 | Простой | Обеспечивает выбор рекурсивной функции |

* 1. **Декомпозиция функций**

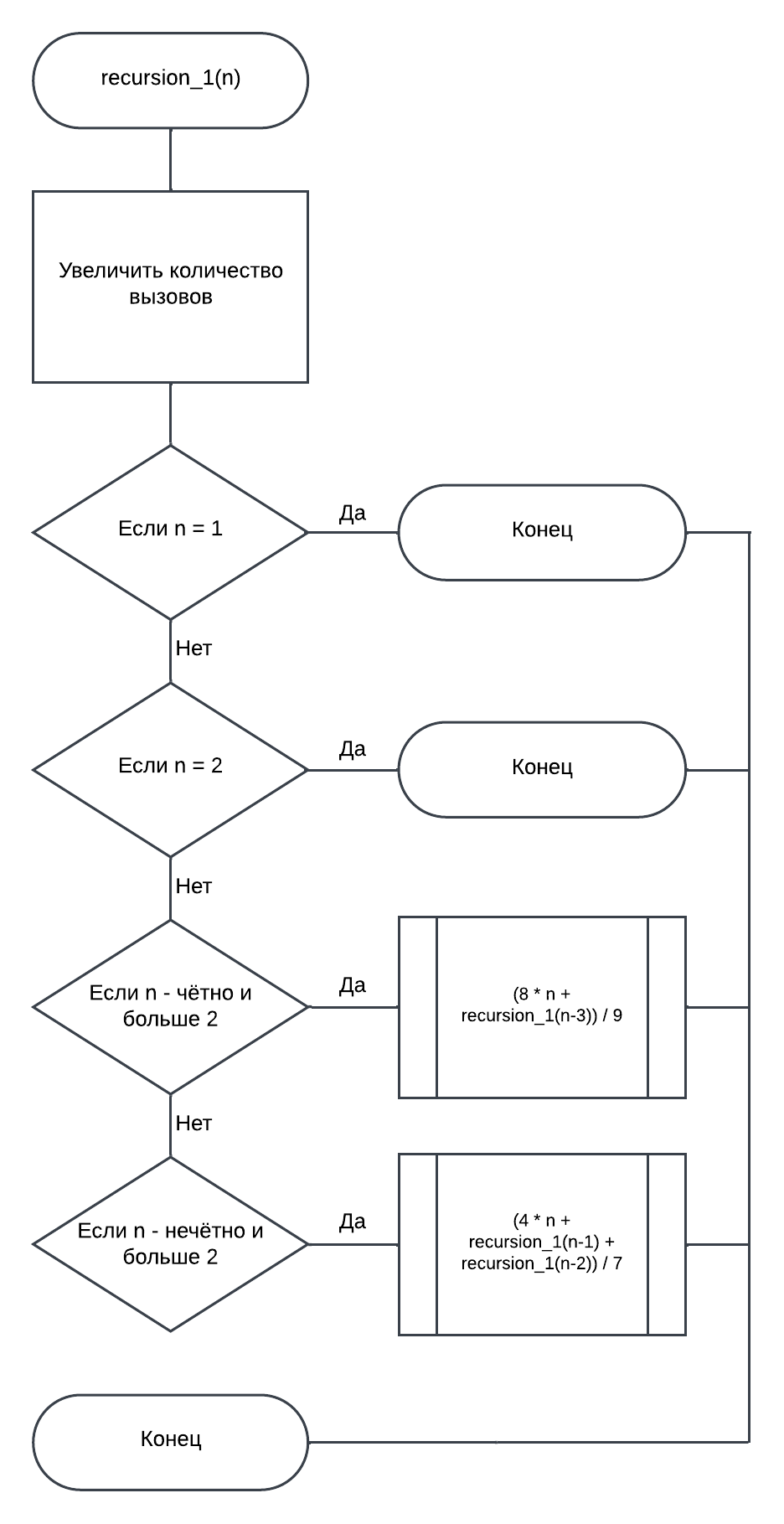
Описывается каждая функция и процедура в программе

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назначение | Имя | Параметры | | Процедура/ функция | Тестируется/  не тестируется |
| Вход:  тип | Выход: тип |
| Работа 1-й рекурсивной функции от параметра n | recursion\_1 | unsigned int | int | Функция | Да |
| Работа 2-й рекурсивной функции от параметра n | recursion\_2 | Unsigned int | int | Функция | Да |
| Работа 3-й рекурсивной функции от параметров numbers и index | recursion\_3 | array<String^>^, index | void | Функция | Нет |
| Проверка данных для 1-й рекурсивной функции | ValidacityCheck1 | Отсутствует | bool | Процедура | Нет |
| Проверка данных для 2-й рекурсивной функции | ValidacityCheck2 | Отсутствует | bool | Процедура | Нет |
| Вывод формулы 1-й рекурсивной функции и её выбор | recursionbtn1\_CheckedChanged | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Вывод формулы 2-й рекурсивной функции и её выбор | recursionbtn2\_CheckedChanged | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| При активном поле ввода позволяет нажать Enter | input\_field1\_KeyPress | System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ en | System::Void | Процедура | Нет |
| При активном поле ввода позволяет нажать Enter | input\_field2\_KeyPress | System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ en | System::Void | Процедура | Нет |
| Кнопка старта выполнения рекурсивной функции и ряда условий | button\_start1\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Кнопка очистки полей ввода, вывода, снятие флажков | clr\_inp\_field1\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Информационное окно программы 1 | info1\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Кнопка ввода значений в массив в программе 2 | input\_button\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Кнопка очистки полей ввода и выовда в программе 2 | clr\_inp\_field2\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Кнопка старта работы программы 2 с рядом условий, разделением чисел и передачей их в рекурсию 3 | button\_start2\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Информационное окно программы 2 | info2\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |

* 1. **Блок-схема**

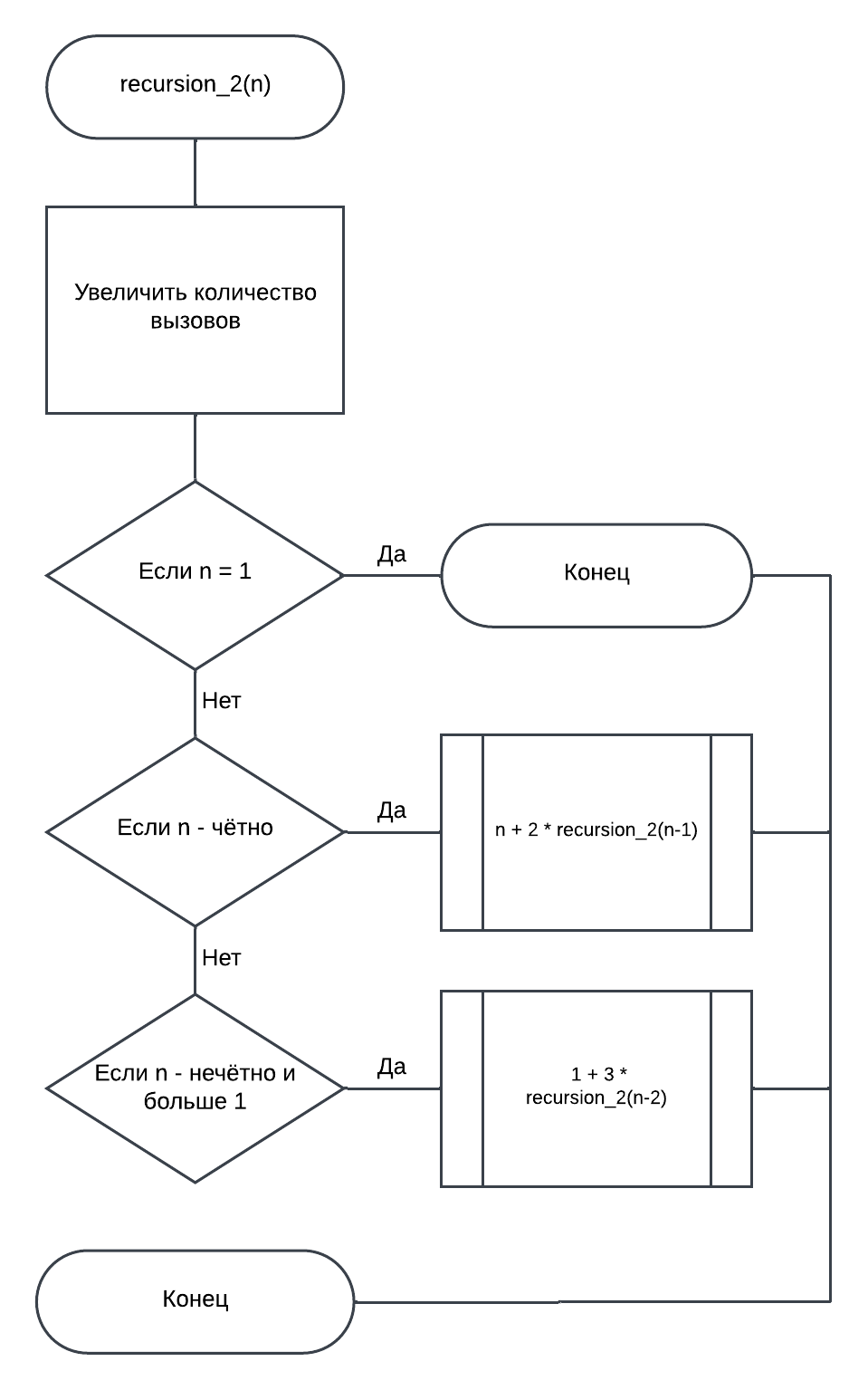
**Приложение А**

**Блок-схема функции recursion\_1**

****

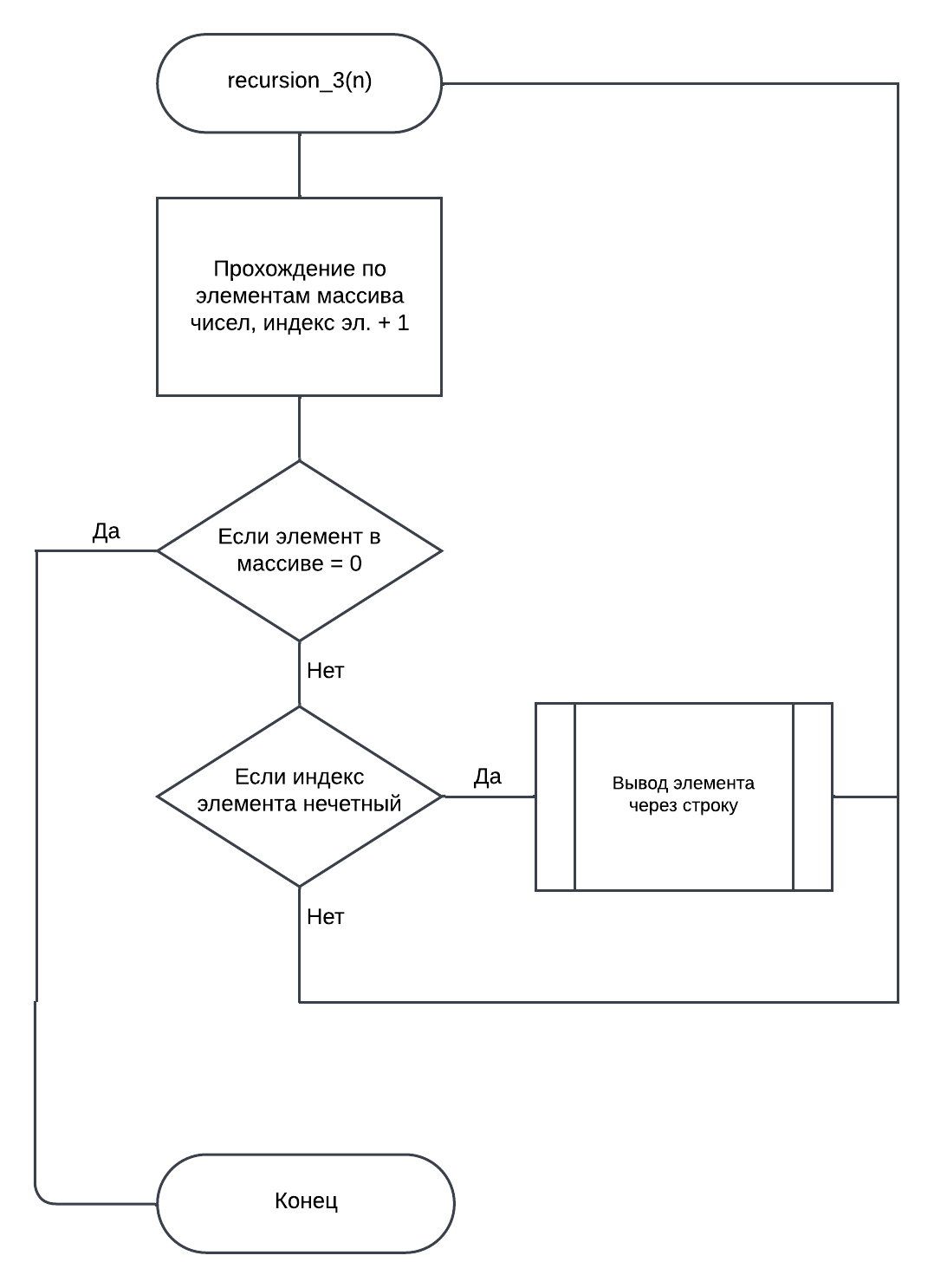
**Приложение Б**

**Блок-схема функции recursion\_2**

****

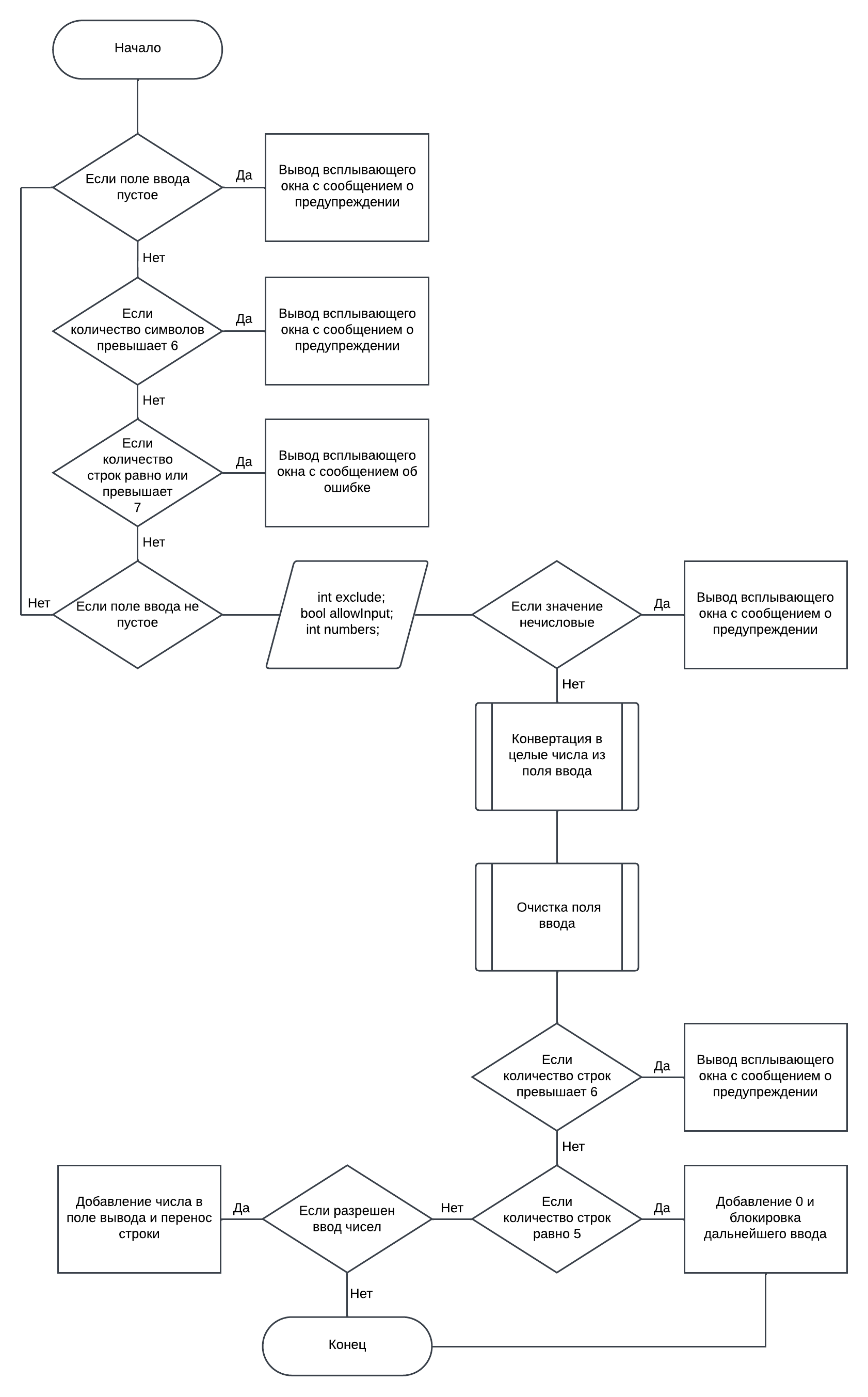
**Приложение В**

**Блок-схема функции recursion\_3**

****

**Приложение Г**

**Блок-схема процедуры input\_button\_Click**

****

**Приложение Д**

**Блок-схема процедуры button\_start1\_Click**

* 1. **Алгоритмизация**

Приводятся алгоритмы процедур/функций и программы

**Требования к исходному тексту программы**:

Программа должна быть разработана в стиле структурного программирования. Структурные уровни должны отделяться отступами. Текст программы должен содержать комментарии:

1. шапка (верхняя часть файла исходного текста):
   * название программы;
   * назначение программы (или данного файла в составе программы);
   * автор (фамилия, имя, группа, курс);
   * дата создания(если делали несколько дней, то интервал времени);
2. комментировани глобальных переменных и констант:
   * назначение переменных;
3. комментирование функций:
   * назначение функции;
   * входные параметры;
   * выходные параметры;
4. комментирование сложных для понимания участков программы:
   * назначение участка;
   * описание механизма работы.

**Приложение**

**Листинг 1** **—** **файл MyForm.cpp**

#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThreadAttribute]

int main(array<String^>^ args) {

Application::SetCompatibleTextRenderingDefault(false);

Application::EnableVisualStyles();

My1lab::MyForm form;

Application::Run(% form);

}

**Приложение**

**Листинг 2** **— файл MyForm.h**

#pragma once

#include <Windows.h>

namespace My1lab {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

}

protected:

~MyForm()

{

if (components)

{

delete components;

}

}

// Пользовательские поля

private: unsigned short int N; // Глобальная переменна, используемая для передачи значения введёного в поле ввода в параметр n рекурсивной функции 1 или 2

private: unsigned int count = 0; // Глобальная переменна, используемая для фиксации количества рекурсивных вызовов

private: bool choice\_check; // Булевая переменная, отображающая выбранную рекурсию

private: System::Windows::Forms::Button^ button\_start1;

private: System::Windows::Forms::RadioButton^ recursionbtn1;

private: System::Windows::Forms::RadioButton^ recursionbtn2;

private: System::Windows::Forms::Label^ input\_field\_label;

private: System::Windows::Forms::Label^ recursion\_label;

private: System::Windows::Forms::TextBox^ input\_field1;

private: System::Windows::Forms::Label^ formula\_of\_recursion\_label;

private: System::Windows::Forms::Label^ output;

private: System::Windows::Forms::Label^ count\_of\_recursion\_label;

private: System::Windows::Forms::GroupBox^ first\_ex;

private: System::Windows::Forms::GroupBox^ second\_ex;

private: System::Windows::Forms::Button^ clr\_inp\_field1;

private: System::Windows::Forms::Button^ input\_button;

private: System::Windows::Forms::GroupBox^ groupBox2;

private: System::Windows::Forms::GroupBox^ groupBox1;

private: System::Windows::Forms::Button^ button\_start2;

private: System::Windows::Forms::Button^ clr\_inp\_field2;

private: System::Windows::Forms::TextBox^ input\_field2;

private: System::Windows::Forms::Button^ err\_check;

private: System::Windows::Forms::Button^ info1;

private: System::Windows::Forms::Button^ info2;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::Label^ Rresult;

private: System::Windows::Forms::Label^ Ncount;

private: System::Windows::Forms::Label^ out\_label;

private: System::Windows::Forms::Label^ in\_label;

private: System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(MyForm::typeid));

this->button\_start1 = (gcnew System::Windows::Forms::Button());

this->recursionbtn1 = (gcnew System::Windows::Forms::RadioButton());

this->recursionbtn2 = (gcnew System::Windows::Forms::RadioButton());

this->input\_field\_label = (gcnew System::Windows::Forms::Label());

this->recursion\_label = (gcnew System::Windows::Forms::Label());

this->input\_field1 = (gcnew System::Windows::Forms::TextBox());

this->formula\_of\_recursion\_label = (gcnew System::Windows::Forms::Label());

this->output = (gcnew System::Windows::Forms::Label());

this->count\_of\_recursion\_label = (gcnew System::Windows::Forms::Label());

this->first\_ex = (gcnew System::Windows::Forms::GroupBox());

this->Ncount = (gcnew System::Windows::Forms::Label());

this->Rresult = (gcnew System::Windows::Forms::Label());

this->label1 = (gcnew System::Windows::Forms::Label());

this->info1 = (gcnew System::Windows::Forms::Button());

this->clr\_inp\_field1 = (gcnew System::Windows::Forms::Button());

this->second\_ex = (gcnew System::Windows::Forms::GroupBox());

this->groupBox2 = (gcnew System::Windows::Forms::GroupBox());

this->out\_label = (gcnew System::Windows::Forms::Label());

this->clr\_inp\_field2 = (gcnew System::Windows::Forms::Button());

this->info2 = (gcnew System::Windows::Forms::Button());

this->err\_check = (gcnew System::Windows::Forms::Button());

this->button\_start2 = (gcnew System::Windows::Forms::Button());

this->input\_field2 = (gcnew System::Windows::Forms::TextBox());

this->input\_button = (gcnew System::Windows::Forms::Button());

this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());

this->in\_label = (gcnew System::Windows::Forms::Label());

this->first\_ex->SuspendLayout();

this->second\_ex->SuspendLayout();

this->groupBox2->SuspendLayout();

this->groupBox1->SuspendLayout();

this->SuspendLayout();

//

// button\_start1

//

this->button\_start1->Location = System::Drawing::Point(219, 20);

this->button\_start1->Margin = System::Windows::Forms::Padding(4, 6, 4, 6);

this->button\_start1->Name = L"button\_start1";

this->button\_start1->Size = System::Drawing::Size(96, 65);

this->button\_start1->TabIndex = 0;

this->button\_start1->Text = L"Старт";

this->button\_start1->UseVisualStyleBackColor = true;

this->button\_start1->Click += gcnew System::EventHandler(this, &MyForm::button\_start1\_Click);

//

// recursionbtn1

//

this->recursionbtn1->AutoSize = true;

this->recursionbtn1->Location = System::Drawing::Point(98, 58);

this->recursionbtn1->Margin = System::Windows::Forms::Padding(4, 6, 4, 6);

this->recursionbtn1->Name = L"recursionbtn1";

this->recursionbtn1->Size = System::Drawing::Size(43, 24);

this->recursionbtn1->TabIndex = 1;

this->recursionbtn1->TabStop = true;

this->recursionbtn1->Text = L"1";

this->recursionbtn1->UseVisualStyleBackColor = true;

this->recursionbtn1->CheckedChanged += gcnew System::EventHandler(this, &MyForm::recursionbtn1\_CheckedChanged);

//

// recursionbtn2

//

this->recursionbtn2->AutoSize = true;

this->recursionbtn2->Location = System::Drawing::Point(153, 58);

this->recursionbtn2->Margin = System::Windows::Forms::Padding(4, 6, 4, 6);

this->recursionbtn2->Name = L"recursionbtn2";

this->recursionbtn2->Size = System::Drawing::Size(43, 24);

this->recursionbtn2->TabIndex = 2;

this->recursionbtn2->TabStop = true;

this->recursionbtn2->Text = L"2";

this->recursionbtn2->UseVisualStyleBackColor = true;

this->recursionbtn2->CheckedChanged += gcnew System::EventHandler(this, &MyForm::recursionbtn2\_CheckedChanged);

//

// input\_field\_label

//

this->input\_field\_label->AutoSize = true;

this->input\_field\_label->Location = System::Drawing::Point(10, 25);

this->input\_field\_label->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->input\_field\_label->Name = L"input\_field\_label";

this->input\_field\_label->Size = System::Drawing::Size(100, 20);

this->input\_field\_label->TabIndex = 3;

this->input\_field\_label->Text = L"Поле ввода";

//

// recursion\_label

//

this->recursion\_label->AutoSize = true;

this->recursion\_label->Location = System::Drawing::Point(10, 62);

this->recursion\_label->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->recursion\_label->Name = L"recursion\_label";

this->recursion\_label->Size = System::Drawing::Size(82, 20);

this->recursion\_label->TabIndex = 4;

this->recursion\_label->Text = L"Рекурсия:";

//

// input\_field1

//

this->input\_field1->Location = System::Drawing::Point(117, 20);

this->input\_field1->Margin = System::Windows::Forms::Padding(4, 6, 4, 6);

this->input\_field1->Name = L"input\_field1";

this->input\_field1->Size = System::Drawing::Size(80, 26);

this->input\_field1->TabIndex = 5;

this->input\_field1->KeyPress += gcnew System::Windows::Forms::KeyPressEventHandler(this, &MyForm::input\_field1\_KeyPress);

//

// formula\_of\_recursion\_label

//

this->formula\_of\_recursion\_label->AutoSize = true;

this->formula\_of\_recursion\_label->ForeColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(64)),

static\_cast<System::Int32>(static\_cast<System::Byte>(64)), static\_cast<System::Int32>(static\_cast<System::Byte>(64)));

this->formula\_of\_recursion\_label->Location = System::Drawing::Point(10, 140);

this->formula\_of\_recursion\_label->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->formula\_of\_recursion\_label->Name = L"formula\_of\_recursion\_label";

this->formula\_of\_recursion\_label->Size = System::Drawing::Size(0, 20);

this->formula\_of\_recursion\_label->TabIndex = 6;

//

// output

//

this->output->AutoSize = true;

this->output->Location = System::Drawing::Point(10, 278);

this->output->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->output->Name = L"output";

this->output->Size = System::Drawing::Size(64, 20);

this->output->TabIndex = 7;

this->output->Text = L"Вывод:";

//

// count\_of\_recursion\_label

//

this->count\_of\_recursion\_label->AutoSize = true;

this->count\_of\_recursion\_label->Location = System::Drawing::Point(10, 312);

this->count\_of\_recursion\_label->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->count\_of\_recursion\_label->Name = L"count\_of\_recursion\_label";

this->count\_of\_recursion\_label->Size = System::Drawing::Size(242, 20);

this->count\_of\_recursion\_label->TabIndex = 8;

this->count\_of\_recursion\_label->Text = L"Количество вызовов функции:";

//

// first\_ex

//

this->first\_ex->Controls->Add(this->Ncount);

this->first\_ex->Controls->Add(this->Rresult);

this->first\_ex->Controls->Add(this->label1);

this->first\_ex->Controls->Add(this->info1);

this->first\_ex->Controls->Add(this->clr\_inp\_field1);

this->first\_ex->Controls->Add(this->button\_start1);

this->first\_ex->Controls->Add(this->input\_field1);

this->first\_ex->Controls->Add(this->input\_field\_label);

this->first\_ex->Controls->Add(this->recursionbtn1);

this->first\_ex->Controls->Add(this->recursion\_label);

this->first\_ex->Controls->Add(this->count\_of\_recursion\_label);

this->first\_ex->Controls->Add(this->recursionbtn2);

this->first\_ex->Controls->Add(this->output);

this->first\_ex->Controls->Add(this->formula\_of\_recursion\_label);

this->first\_ex->Location = System::Drawing::Point(18, 9);

this->first\_ex->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->first\_ex->Name = L"first\_ex";

this->first\_ex->Padding = System::Windows::Forms::Padding(4, 5, 4, 5);

this->first\_ex->Size = System::Drawing::Size(429, 395);

this->first\_ex->TabIndex = 12;

this->first\_ex->TabStop = false;

//

// Ncount

//

this->Ncount->AutoSize = true;

this->Ncount->ForeColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(64)), static\_cast<System::Int32>(static\_cast<System::Byte>(64)),

static\_cast<System::Int32>(static\_cast<System::Byte>(64)));

this->Ncount->Location = System::Drawing::Point(262, 312);

this->Ncount->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->Ncount->Name = L"Ncount";

this->Ncount->Size = System::Drawing::Size(0, 20);

this->Ncount->TabIndex = 13;

//

// Rresult

//

this->Rresult->AutoSize = true;

this->Rresult->ForeColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(64)), static\_cast<System::Int32>(static\_cast<System::Byte>(64)),

static\_cast<System::Int32>(static\_cast<System::Byte>(64)));

this->Rresult->Location = System::Drawing::Point(84, 278);

this->Rresult->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->Rresult->Name = L"Rresult";

this->Rresult->Size = System::Drawing::Size(0, 20);

this->Rresult->TabIndex = 12;

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(10, 106);

this->label1->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(238, 20);

this->label1->TabIndex = 11;

this->label1->Text = L"Формула выбранной рекурсии";

//

// info1

//

this->info1->Location = System::Drawing::Point(15, 351);

this->info1->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->info1->Name = L"info1";

this->info1->Size = System::Drawing::Size(390, 35);

this->info1->TabIndex = 10;

this->info1->Text = L"Информация";

this->info1->UseVisualStyleBackColor = true;

this->info1->Click += gcnew System::EventHandler(this, &MyForm::info1\_Click);

//

// clr\_inp\_field1

//

this->clr\_inp\_field1->Location = System::Drawing::Point(324, 20);

this->clr\_inp\_field1->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->clr\_inp\_field1->Name = L"clr\_inp\_field1";

this->clr\_inp\_field1->Size = System::Drawing::Size(96, 65);

this->clr\_inp\_field1->TabIndex = 9;

this->clr\_inp\_field1->Text = L"Очистить поле";

this->clr\_inp\_field1->UseVisualStyleBackColor = true;

this->clr\_inp\_field1->Click += gcnew System::EventHandler(this, &MyForm::clr\_inp\_field1\_Click);

//

// second\_ex

//

this->second\_ex->Controls->Add(this->groupBox2);

this->second\_ex->Controls->Add(this->clr\_inp\_field2);

this->second\_ex->Controls->Add(this->info2);

this->second\_ex->Controls->Add(this->err\_check);

this->second\_ex->Controls->Add(this->button\_start2);

this->second\_ex->Controls->Add(this->input\_field2);

this->second\_ex->Controls->Add(this->input\_button);

this->second\_ex->Controls->Add(this->groupBox1);

this->second\_ex->Location = System::Drawing::Point(18, 414);

this->second\_ex->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->second\_ex->Name = L"second\_ex";

this->second\_ex->Padding = System::Windows::Forms::Padding(4, 5, 4, 5);

this->second\_ex->Size = System::Drawing::Size(429, 232);

this->second\_ex->TabIndex = 13;

this->second\_ex->TabStop = false;

//

// groupBox2

//

this->groupBox2->Controls->Add(this->out\_label);

this->groupBox2->Location = System::Drawing::Point(324, 29);

this->groupBox2->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->groupBox2->Name = L"groupBox2";

this->groupBox2->Padding = System::Windows::Forms::Padding(4, 5, 4, 5);

this->groupBox2->Size = System::Drawing::Size(81, 152);

this->groupBox2->TabIndex = 1;

this->groupBox2->TabStop = false;

this->groupBox2->Text = L"Вывод";

//

// out\_label

//

this->out\_label->AutoSize = true;

this->out\_label->Location = System::Drawing::Point(9, 26);

this->out\_label->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->out\_label->Name = L"out\_label";

this->out\_label->Size = System::Drawing::Size(0, 20);

this->out\_label->TabIndex = 1;

//

// clr\_inp\_field2

//

this->clr\_inp\_field2->Location = System::Drawing::Point(93, 85);

this->clr\_inp\_field2->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->clr\_inp\_field2->Name = L"clr\_inp\_field2";

this->clr\_inp\_field2->Size = System::Drawing::Size(117, 57);

this->clr\_inp\_field2->TabIndex = 4;

this->clr\_inp\_field2->Text = L"Очистить поле";

this->clr\_inp\_field2->UseVisualStyleBackColor = true;

this->clr\_inp\_field2->Click += gcnew System::EventHandler(this, &MyForm::clr\_inp\_field2\_Click);

//

// info2

//

this->info2->Location = System::Drawing::Point(15, 188);

this->info2->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->info2->Name = L"info2";

this->info2->Size = System::Drawing::Size(390, 35);

this->info2->TabIndex = 7;

this->info2->Text = L"Информация";

this->info2->UseVisualStyleBackColor = true;

this->info2->Click += gcnew System::EventHandler(this, &MyForm::info2\_Click);

//

// err\_check

//

this->err\_check->Location = System::Drawing::Point(15, 151);

this->err\_check->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->err\_check->Name = L"err\_check";

this->err\_check->Size = System::Drawing::Size(195, 31);

this->err\_check->TabIndex = 6;

this->err\_check->Text = L"Проверка на ошибки";

this->err\_check->UseVisualStyleBackColor = true;

this->err\_check->Click += gcnew System::EventHandler(this, &MyForm::err\_check\_Click);

//

// button\_start2

//

this->button\_start2->Location = System::Drawing::Point(15, 85);

this->button\_start2->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->button\_start2->Name = L"button\_start2";

this->button\_start2->Size = System::Drawing::Size(69, 57);

this->button\_start2->TabIndex = 5;

this->button\_start2->Text = L"Старт";

this->button\_start2->UseVisualStyleBackColor = true;

this->button\_start2->Click += gcnew System::EventHandler(this, &MyForm::button\_start2\_Click);

//

// input\_field2

//

this->input\_field2->Location = System::Drawing::Point(15, 45);

this->input\_field2->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->input\_field2->Name = L"input\_field2";

this->input\_field2->Size = System::Drawing::Size(67, 26);

this->input\_field2->TabIndex = 3;

this->input\_field2->KeyPress += gcnew System::Windows::Forms::KeyPressEventHandler(this, &MyForm::input\_field2\_KeyPress);

//

// input\_button

//

this->input\_button->Location = System::Drawing::Point(93, 45);

this->input\_button->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->input\_button->Name = L"input\_button";

this->input\_button->Size = System::Drawing::Size(117, 31);

this->input\_button->TabIndex = 2;

this->input\_button->Text = L"Ввести";

this->input\_button->UseVisualStyleBackColor = true;

this->input\_button->Click += gcnew System::EventHandler(this, &MyForm::input\_button\_Click);

//

// groupBox1

//

this->groupBox1->Controls->Add(this->in\_label);

this->groupBox1->Location = System::Drawing::Point(236, 29);

this->groupBox1->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->groupBox1->Name = L"groupBox1";

this->groupBox1->Padding = System::Windows::Forms::Padding(4, 5, 4, 5);

this->groupBox1->Size = System::Drawing::Size(80, 152);

this->groupBox1->TabIndex = 0;

this->groupBox1->TabStop = false;

this->groupBox1->Text = L"Ввод";

//

// in\_label

//

this->in\_label->AutoSize = true;

this->in\_label->Location = System::Drawing::Point(9, 26);

this->in\_label->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->in\_label->Name = L"in\_label";

this->in\_label->Size = System::Drawing::Size(0, 20);

this->in\_label->TabIndex = 0;

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(9, 20);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->AutoSize = true;

this->AutoSizeMode = System::Windows::Forms::AutoSizeMode::GrowAndShrink;

this->ClientSize = System::Drawing::Size(465, 665);

this->Controls->Add(this->second\_ex);

this->Controls->Add(this->first\_ex);

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedSingle;

this->Icon = (cli::safe\_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));

this->Margin = System::Windows::Forms::Padding(4, 6, 4, 6);

this->MaximizeBox = false;

this->Name = L"MyForm";

this->Text = L"Lab 1 - Recursion";

this->first\_ex->ResumeLayout(false);

this->first\_ex->PerformLayout();

this->second\_ex->ResumeLayout(false);

this->second\_ex->PerformLayout();

this->groupBox2->ResumeLayout(false);

this->groupBox2->PerformLayout();

this->groupBox1->ResumeLayout(false);

this->groupBox1->PerformLayout();

this->ResumeLayout(false);

} #pragma endregion

/////////////////////

/\*Первая программа.\*/

/////////////////////

int recursion\_1(unsigned int n)

{

++count; // Фиксация количества рекурсивных вызовов.

if (n == 1)

{

return 1;

}

else if (n == 2)

{

return 2;

}

else if (n > 2 && n % 2 == 0) // > 2 и четное.

{

return ((8 \* n + recursion\_1(n - 3)) / 9);

}

else if (n > 2 && n % 2 != 0) // > 2 и не чётное.

{

return ((4 \* n + recursion\_1(n - 1) + recursion\_1(n - 2)) / 7);

}

}

int recursion\_2(unsigned int n)

{

++count; // Фиксация количества рекурсивных вызовов.

if (n == 1)

{

return 1;

}

else if (n % 2 == 0) // четное

{

return (n + 2 \* recursion\_2(n - 1));

}

else if (n > 1 && n % 2 != 0) // > 1 и не чётное.

{

return (1 + 3 \* recursion\_2(n - 2));

}

}

int high\_valid\_value1{ 70 }, low\_valid\_value1{ 1 };

int high\_valid\_value2{ 20000 }, low\_valid\_value2{ 1 };

bool ValidacityCheck1() // Проверка 1 на корректность введеных значений.

{

String^ error\_overflow = ""; // Изначальное значение для текстовой переменной, чтобы её можно было менять удобнее.

try

{

if (this->recursionbtn1->Checked == true && this->input\_field1->Text != "") // Проверяется, не является ли текст введенного значения пустым и выбрана ли рекурсионная формула 1.

{

int dataI = System::Convert::ToInt32(this->input\_field1->Text); // Текст из поля ввода конвертируется в целое число(dataI).

if (dataI > high\_valid\_value1) // проверка макс. значения 1

{

error\_overflow = "Введённое значение превышает допустимое " + high\_valid\_value1 + "!\nПосмотрите информацию, чтобы узнать подробнее.";

throw error\_overflow; // Выдаёт ошибку с заданным текстом.

}

else if (dataI < low\_valid\_value1) // проверка мин. значения 1.

{

error\_overflow = "Число введённое вами не должно быть меньше " + low\_valid\_value1 + "!";

throw error\_overflow; // Выдаёт ошибку с заданным текстом.

}

}

}

catch (String^ error\_overflow) // Берёт из try полученное значение текста ошибки.

{

this->input\_field1->ForeColor = Color::Red; // Подсветка текста в поле ввода красным.

System::Windows::Forms::DialogResult result = MessageBox::Show(error\_overflow, "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error); // Выводит диалоговое окно result с полученной из условия ошибкой с кнопкой OK и иконкой Error.

if (result == System::Windows::Forms::DialogResult::OK) // Условие при нажатии кнопки OK очищаются поля и убираются галочки.

{

this->input\_field1->Text = ""; // Очищает поле ввода.

this->input\_field1->ForeColor = Color::Black; // Меняет текст обратно на черный.

this->recursionbtn1->Checked = false; // Убирает галочку с кнопки рекурскии 1.

this->recursionbtn2->Checked = false; // Убирает галочку с кнопки рекурскии 2.

}

}

return 0;

}

bool ValidacityCheck2() // Проверка 2 на корректность введеных значений.

{

String^ error\_overflow = ""; // Изначальное значение для текстовой переменной, чтобы её можно было менять удобнее.

try

{

if (this->recursionbtn2->Checked == true && this->input\_field1->Text != "") // Проверяется, не является ли текст введенного значения пустым и выбрана ли рекурсионная формула 2.

{

int dataI = System::Convert::ToInt32(this->input\_field1->Text); // Целочисленному значению присваивается преобразованный текст из поля ввода.

if (dataI > high\_valid\_value2) // проверка макс. значения 2.

{

error\_overflow = "Введённое значение превышает допустимое " + high\_valid\_value2 + "!\nПосмотрите информацию, чтобы узнать подробнее.";

throw error\_overflow; // Выдаёт ошибку с заданным текстом.

}

else if (dataI < low\_valid\_value2) // проверка мин. значения 2.

{

error\_overflow = "Число введённое вами не должно быть меньше " + low\_valid\_value2 + "!";

throw error\_overflow; // Выдаёт ошибку с заданным текстом.

}

}

}

catch (String^ error\_overflow) // Берёт из try полученное значение текста ошибки.

{

this->input\_field1->ForeColor = Color::Red;

System::Windows::Forms::DialogResult result = MessageBox::Show(error\_overflow, "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error); // Выводит диалоговое окно result с полученной из условия ошибкой с кнопкой OK и иконкой Error.

if (result == System::Windows::Forms::DialogResult::OK) // Условие при нажатии кнопки OK очищаются поля и убираются галочки.

{

this->input\_field1->Text = ""; // Очищает поле ввода.

this->input\_field1->ForeColor = Color::Black; // Меняет текст обратно на черный.

this->recursionbtn1->Checked = false; // Убирает галочку с кнопки рекурскии 1.

this->recursionbtn2->Checked = false; // Убирает галочку с кнопки рекурскии 2.

}

}

return 0;

}

private: System::Void recursionbtn1\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) // Проверяет нажата ли кнопка первой рекурсивной формулы.

{

if (this->recursionbtn1->Checked == true) // Условие для вывода текста первой рекурсивной функции.

{

this->formula\_of\_recursion\_label->Text =

"F(n) = 1, при n = 1;\r\n"

"F(n) = 2, при n = 2;\r\n"

"F(n) = [(8 \* n + F(n - 3)) / 9], если n > 2 и при этом \nn четно;\r\n"

"F(n) = [(4 \* n + F(n - 1) + F(n - 2)) / 7], если n > 2 и при \nэтом n нечетно.";

choice\_check = true;

}

else

{

this->formula\_of\_recursion\_label->Text = ""; // Очистка поля вывода текста в случае выбора другой рекурсивной функции.

}

}

private: System::Void recursionbtn2\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) // Проверяет нажата ли кнопка второй рекурсивной формулы.

{

if (this->recursionbtn2->Checked == true) // Условие для вывода текста второй рекурсивной функции.

{

this->formula\_of\_recursion\_label->Text =

"F(n) = 1, при n = 1;\r\n"

"F(n) = n + 2 \* F(n - 1), если n чётно;\r\n"

"F(n) = 1 + 3 \* F(n - 2), если n > 1 и при этом n нечётно.";

choice\_check = false;

}

else

{

this->formula\_of\_recursion\_label->Text = ""; // Очистка поля вывода текста в случае выбора другой рекурсивной функции.

}

}

private: System::Void input\_field1\_KeyPress(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ en) // Нажатие Enter для первой программы.

{

if (en->KeyChar == (char)Keys::Enter)

{

button\_start1\_Click(sender, en); // Вызываем обработчик события нажатия кнопки button\_start1.

en->Handled = true; // Предотвращаем дальнейшую обработку нажатия клавиши Enter.

}

}

private: System::Void button\_start1\_Click(System::Object^ sender, System::EventArgs^ e)

{

count = 0; // Начальное значение для вызовов рекурсивной функции.

if (this->input\_field1->Text == "") // Условие для проверки на наличие значений в поле ввода.

{

MessageBox::Show("Поле ввода пусто!", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

}

if (this->recursionbtn1->Checked == false && this->recursionbtn2->Checked == false) // Условие для проверки на наличие выбора рекурсивной функции.

{

MessageBox::Show("Не выбрана ни одна из рекурсивных формул!", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

}

if (this->input\_field1->Text != "")

{

if(input\_field1->Text->Length > 8) // Проверка на длину введённого значения.

{

MessageBox::Show("Длина введенного числа превышает максимальное допустимое значение.", "Предупреждение", MessageBoxButtons::OK, MessageBoxIcon::Warning);

return;

}

int exclude;

if (!Int32::TryParse(input\_field1->Text, exclude)) // Условие исключает нечисловые значения.

{

// Если введенное значение не является числом, выводим сообщение об ошибке.

MessageBox::Show("Поле ввода должно содержать только числовые значения и не содержать пробелов между цифрами/числами!", "Предупреждение", MessageBoxButtons::OK, MessageBoxIcon::Warning);

return;

}

N = System::Convert::ToInt32(this->input\_field1->Text); // Преобразованный текст из поля ввода присваевается глобальной переменной N.

if (ValidacityCheck1() == true) // Проверяет допустимые значения через функцию ValidacityCheck1 и снимает флажки, если есть недопустимые значения.

{

this->recursionbtn1->Checked = false;

this->recursionbtn2->Checked = false;

}

if (ValidacityCheck2() == true) // Проверяет допустимые значения через функцию ValidacityCheck2 и снимает флажки, если есть недопустимые значения.

{

this->recursionbtn1->Checked = false;

this->recursionbtn2->Checked = false;

}

if (this->recursionbtn1->Checked || this->recursionbtn2->Checked) // Проверяет нажата ли какая-либо из рекурсий.

{

if (choice\_check == true) // Проверка для первой рекурсии.

{

this->Rresult->Text = System::Convert::ToString(recursion\_1(N));

this->Ncount->Text = System::Convert::ToString(count);

}

else if (choice\_check == false) // Проверка для второй рекурсии.

{

this->Rresult->Text = System::Convert::ToString(recursion\_2(N));

this->Ncount->Text = System::Convert::ToString(count);

}

}

}

}

private: System::Void clr\_inp\_field1\_Click(System::Object^ sender, System::EventArgs^ e) // Очистка всех полей и галочек.

{

this->recursionbtn1->Checked = false; // Убирает галочку с кнопки рекурскии 1.

this->recursionbtn2->Checked = false; // Убирает галочку с кнопки рекурскии 2.

this->input\_field1->Text = ""; // Очищает поле ввода.

this->formula\_of\_recursion\_label->Text = ""; // Очищает поле с информацией о выбранной рекурсивной функции.

this->Rresult->Text = ""; // Очищает поле результата.

this->Ncount->Text = ""; // Очищает поле подсчёта вызовов.

}

private: System::Void info1\_Click(System::Object^ sender, System::EventArgs^ e) // Информационное окно 1.

{

String^ info\_p1 =

"1) Для ввода значений введите данные в поле ввода (максимальное значение - " + high\_valid\_value1 + ", минимальное - " + low\_valid\_value1 + " для рекурсии 1, максимальное значение - " + high\_valid\_value2 + ", минимальное - " + low\_valid\_value2 + " для рекурсии 2);\n\n"

"2) Выберите вариант рекурсивной функции. После выбора ниже будет выведена формула выбранной функции;\n\n"

"3) Нажмине кнопку \"Старт\" для запуска рекурсии;\n\n4) После запуска рекурсии будет выведен результат рекурсивных вычислений и количество вызовов функции.\n\n"

"Примечание: Введение нечисловых значений и превышение лимита элементов в строке в 8 символов выведет окно с предупреждением.";

MessageBox::Show(info\_p1, "Информация о программе", MessageBoxButtons::OK, MessageBoxIcon::Information);

}

/////////////////////

/\*Вторая программа.\*/

/////////////////////

void recursion\_3(array<String^>^ numbers, int index) // Объявление рекурсивной функции с параметрами: 1) массив строк numbers для массива введенных значений; 2) Целочисленное index для индексации значений в массиве.

{

if (numbers[index] == "0") // Проверяет равен ли элемент массива с текущим индексом 0.

{

return; // Если элемент равен "0", функция прекращает выполнение и возвраща ет управление вызывающему коду.

}

if ((index + 1) % 2 != 0) // Проверка, является ли индекс плюс один нечетным числом для выбора через один.

{

out\_label->Text += numbers[index] + "\r\n"; // Если индекс плюс один нечетный, то добавляем текущий элемент массива и перенос новой строки в списке вывода.

}

recursion\_3(numbers, index + 1); // Рекурсивный вызов функции с увеличенным на единицу индексом для проверки последующих элементов.

}

private: System::Void input\_field2\_KeyPress(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ en) // Нажатие Enter для первой программы.

{

if (en->KeyChar == (char)Keys::Enter) //Проверяем, нажата ли клавиша Enter.

{

input\_button\_Click(sender, en); // Вызываем обработчик события нажатия кнопки input\_button.

en->Handled = true; // Предотвращаем дальнейшую обработку нажатия клавиши Enter.

}

}

private: System::Void input\_button\_Click(System::Object^ sender, System::EventArgs^ e) // Кнопка ввода значений в массив.

{

if (this->input\_field2->Text == "") // Проверка пустого поля ввода.

{

MessageBox::Show("Поле ввода не может быть пустым!", "Предупреждение", MessageBoxButtons::OK, MessageBoxIcon::Warning);

}

if (in\_label->Text->Split('\n')->Length - 1 >= 7) // Проверка количества строк в списке вводе массива.

{

MessageBox::Show("Превышено максимальное количество строк.", "Предупреждение", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

}

if (input\_field2->Text->Length > 6) // Проверка количества введённых символов в поле ввода.

{

MessageBox::Show("Длина введенного числа превышает максимальное допустимое значение.", "Предупреждение", MessageBoxButtons::OK, MessageBoxIcon::Warning);

return;

}

if (input\_field2->Text != "")

{

int exclude;

if (!Int32::TryParse(input\_field2->Text, exclude)) // Проверка на наличие нечисловых значений.

{

MessageBox::Show("Поле ввода должно содержать только числовые значения и не содержать пробелов между цифрами/числами!", "Предупреждение", MessageBoxButtons::OK, MessageBoxIcon::Warning);

return;

}

bool allowInput = true; // Булевое значение для проверки ввода новых значений после записи в массив предыдущих.

int num = System::Convert::ToInt32(input\_field2->Text); // Конвертирует num введённые значения в числовые.

this->input\_field2->Text = ""; // Очищает поле ввода для последующего ввода значений.

if (in\_label->Text->Split('\n')->Length - 1 == 6 && num != 0) // Проверка на количество строк в списке ввода.

{

MessageBox::Show("Количество строк достигло максимального значения.", "Предупреждение", MessageBoxButtons::OK, MessageBoxIcon::Warning);

return;

}

if (in\_label->Text->Split('\n')->Length - 1 == 5 && num != 0) // Условие для добавления в конец 0 по достижении 5 строк.

{

in\_label->Text += "0\n";

allowInput = false;

}

else

{

if (allowInput) // Проверка, разрешен ли ввод новых значений.

{

in\_label->Text += num + "\r\n"; // Добавление введённого числа в ввод и перенос строки в списке ввода.

}

}

}

}

private: System::Void clr\_inp\_field2\_Click(System::Object^ sender, System::EventArgs^ e) // Очистка всех полей.

{

this->input\_field2->Text = ""; // Очищает поле ввода.

this->in\_label->Text = ""; // Очищает список ввода массива чисел.

this->out\_label->Text = ""; // Очищает вывод массива чисел.

if (!button\_start2->Enabled) // Взводит кнопку старта.

{

button\_start2->Enabled = true;

}

}

private: System::Void button\_start2\_Click(System::Object^ sender, System::EventArgs^ e)

{

array<String^>^ numbers = in\_label->Text->Split(gcnew array<wchar\_t> {'\n', '\r'}, StringSplitOptions::RemoveEmptyEntries); // Разделение значений в массиве для того, чтобы возможно было обеспечить работу рекурсивной функции.

if (numbers->Length == 0) // Условие для проверки пуст ли список ввода.

{

MessageBox::Show("Список чисел пуст.", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

}

if (numbers[numbers->Length - 1] != "0") // Проверяем, что последний элемент является 0.

{

MessageBox::Show("Последнее число не равно \"0\".", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

}

recursion\_3(numbers, 0); // Начинает рекурсию после всех проверок и записи массива чисел.

button\_start2->Enabled = false; // Отключает кнопку старта для предотвращения дальнейшего нажатия.

}

private: System::Void info2\_Click(System::Object^ sender, System::EventArgs^ e) // Информационное окно 2.

{

String^ info\_p2 =

"Программа определяет нечетные числа в заданной пользователем последовательности:\n\n"

"1) Для ввода значений введите данные в поле ввода;\n\n2) Нажмите на кнопку \"Ввести\" и в поле \"Ввод\" появятся введённые вами значения;\n\n"

"3) Последняя цифра должна быть всегда равно 0 (По достижении 6 строк последнее ведденое число заменится 0);\n\n"

"4) Нажмите кнопку \"Старт\" для запуска программы; \n\n5) После выполнения рекурсии в поле \"Вывод\" появится результат выполнения;\n\n"

"6) После завершения программы и при желании вновь воспользоваться программой нажмите кнопку \"Очистить поле\". \n\n Примечание: Введение нечисловых значений и превышение лимита элементов в строке в 6 символов выведет окно с предупреждением.";

MessageBox::Show(info\_p2, "Информация о программе", MessageBoxButtons::OK, MessageBoxIcon::Information);

}

bool runTests() {

// Путь к исполняемому файлу с тестами

String^ testExecutablePath = "../1lab/x64/Debug/UnitTest.exe";

// Создание процесса для запуска тестов

STARTUPINFOA si;

PROCESS\_INFORMATION pi;

ZeroMemory(&si, sizeof(si));

ZeroMemory(&pi, sizeof(pi));

si.cb = sizeof(si);

// Передача параметров командной строки для запуска тестов

String^ commandLine = "\"" + testExecutablePath + "\"";

// Ожидание завершения процесса

WaitForSingleObject(pi.hProcess, INFINITE);

// Закрытие дескрипторов процесса и потока

CloseHandle(pi.hProcess);

CloseHandle(pi.hThread);

DWORD exitCode;

if (!GetExitCodeProcess(pi.hProcess, &exitCode)) {

MessageBox::Show("Ошибка при получении статуса завершения тестов.");

return false;

}

// Если тесты завершились успешно, возвращаем true, иначе false

return 0;

}

private: System::Void err\_check\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (runTests()==0) {

MessageBox::Show("Тесты успешно пройдены!", "Успех");

}

else {

MessageBox::Show("Тесты не пройдены!", "Ошибка");

}

}

};

}

**Приложение**

**Листинг 3** **— файл functions.h**

#pragma once

namespace func

{

int recursion\_1(unsigned int n)

{

if (n == 1)

{

return 1;

}

else if (n == 2)

{

return 2;

}

else if (n > 2 && n % 2 == 0)

{

return ((8 \* n + recursion\_1(n - 3)) / 9);

}

else if (n > 2 && n % 2 != 0)

{

return ((4 \* n + recursion\_1(n - 1) + recursion\_1(n - 2)) / 7);

}

}

int recursion\_2(unsigned int n)

{

if (n == 1)

{

return 1;

}

else if (n % 2 == 0)

{

return (n + 2 \* recursion\_2(n - 1));

}

else if (n > 1 && n % 2 != 0)

{

return (1 + 3 \* recursion\_2(n - 2));

}

}}

**Приложение**

**Листинг 4 — файл test.cpp**

#include "pch.h"

#include "../1lab/functions.h"

using namespace func;

///////////////////

//Первая рекурсия//

///////////////////

TEST(recursion1, Test1) {

EXPECT\_EQ(2, recursion\_1(3));

EXPECT\_TRUE(true);

}

TEST(recursion1, Test2) {

EXPECT\_EQ(21, recursion\_1(27));

EXPECT\_TRUE(true);

}

TEST(recursion1, Test3) {

EXPECT\_EQ(51, recursion\_1(63));

EXPECT\_TRUE(true);

}

///////////////////

//Вторая рекурсия//

///////////////////

TEST(recursion2, Test1) {

EXPECT\_EQ(1054004169, recursion\_2(825));

EXPECT\_TRUE(true);

}

TEST(recursion2, Test2) {

EXPECT\_EQ(-907148308, recursion\_2(1956));

EXPECT\_TRUE(true);

}

TEST(recursion2, Test3) {

EXPECT\_EQ(-1704166639, recursion\_2(113));

EXPECT\_TRUE(true);

}