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  / И. С. Цветков  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

# 1 Постановка задачи

Разработка программы с графическим интерфейсом для работы с рекурсиями.

# 1.1 Наименование задачи

Лабораторная работа №1. Рекурсии.

# 1.2 Словесное описание

Разработайте программу с графическим интерфейсом для работы с рекурсиями.

Написать дополнительную программу, которая будет считывать количество единиц в строке, которые найдены до момента встречи в строке двух нулей, идущих подряд.

| Объект программы | Имя объекта в программе | Тип данных | Диапазоны представления | Простая/  структура | Вход/  выход/  Константа |
| --- | --- | --- | --- | --- | --- |
| Целое число | n | unsigned short int | 0 - 64 | Простая | Вход |
| Целое число | count | UInt32 | от 0 до 4 294 967 295 | Простая | Вход |
| Булевое значение | choiceСheck | Bool | True/False | Простая | Вход |

Внешние спецификации данных

# 1.3 Внешние спецификации функций программы

Программа должна рассчитывать значение функции по входным

данным. Выводить полученное значение и количество вызовов функции в отдельные поля. Должна быть реализована очистка полей, вывод справочной информации о программе.

# 1.4 Математическая формулировка задачи

Для всех вводимых значений в программе выбран тип данных unsigned int, так как пользователь ограничил ввод данных неотрицательными значениями.

Формула первой рекурсивной функции:  
F(n) = 1, при n = 1;  
F(n) = n + 3 \* F(n - 1), если n чётное;  
F(n) = 2 + 2 \* F(n - 2), если n > 1 и при этом n нечётное.

Ограничение ввода для первой рекурсии – 30, при больших значениях программа работает медленно или происходит её остановка.

Формула второй рекурсивной функции:  
F(n) = 1, при n = 1;

F(n) = 2, при n = 2;

F(n) = (7\*n + F(n - 3))/9, если n > 2 и при этом n четно;

F(n) = (5\*n + F(n - 1) + F(n - 2))/7, если n > 2 и при этом n нечетно.

Ограничение ввода для второй рекурсии – 81, при расчете от 82 и больше программа работает медленно или происходит её остановка.

Для хранения последовательности значений выбрана переменная типа string.

# 1.5 Спецификации интерфейса

Для написания графического интерфейса была использована библиотека Windows Forms.

Плюсы интерфейса заключаются в том, что реализован контролируемый ввод, при попытке ввода некорректных данных срабатывает функция контроля ввода.

Реализована обработка повторного нажатия ключевых кнопок, что предотвращает некорректный вывод данных.

В приложении предусмотрено встроенное тестирование при запуске программы, и отдельная кнопка для тестирования работоспособности ключевых расчетных функций.

В приложении реализована работа с рекурсивными функциями, ввод/вывод, очистка полей, кнопки, выводящие справочную информацию о работе программы.

# 1.6 Внешние данные тестирования

# Таблица 1. Тест-кейс №1, выход из приложения

| Идентификатор | | | T01 |
| --- | --- | --- | --- |
| Заголовок | | | Проверка выхода из приложения |
| Предусловие | | | Приложение не запущено |
| Ожидаемый результат | | | Успешный выход из приложения |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Запуск UniLab1.exe | Открытие главного окна приложения | Pass |
| 2 | Нажатие на кнопку «Выход» | Выход из приложения | Pass |
| Результат | | | Pass |

# Таблица 2. Тест-кейс №2, кнопка "Запуск"

| Идентификатор | | | T02 |
| --- | --- | --- | --- |
| Заголовок | | | Проверка работоспособности кнопки «Запуск» |
| Предусловие | | | Приложение запущено. |
| Ожидаемый результат | | | Вывод в поле «Вывод» результата рек. Вычислений, 10. |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Выбрать вариант рекурсии №1 | Вывод формулы рекурсии №1 | Pass |
| 2 | Ввести данные в поле ввода, 5 | Данные корректны | Pass |
| 3 | Нажатие кнопки «Запуск» | Проведен расчет, выведено значение 10 | Pass |
| Результат | | | Pass |

# Таблица 3. Тест-кейс №3, кнопка "Очистить поля"

| Идентификатор | | | T03 |
| --- | --- | --- | --- |
| Заголовок | | | Проверка работоспособности кнопки «Очистить поля» |
| Предусловие | | | Приложение запущено |
| Ожидаемый результат | | | Полям будет присвоено стандартное значение |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Ввести значение в поле ввода |  | Pass |
| 2 | Выбрать формулу рекурсии | Будет выведена формула выбранной формулы | Pass |
| 3 | Нажать кнопку «Запуск» | Будет выведен результат вычислений | Pass |
| 4 | Нажать кнопку «Очистить поля» | Заполненным полям присвоены стандартные значения | Pass |
| Результат | | | Pass |

# Таблица 4. Тест-кейс №4, дополнительная программа

| Идентификатор | | | T04 |
| --- | --- | --- | --- |
| Заголовок | | | Проверка работы дополнительной программы |
| Предусловие | | | Приложение запущено |
| Ожидаемый результат | | | Будет выведен корректный результат |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Ввести поочередно последовательность чисел:  1, 1, 0 , 1, 0, 0, 1 |  | Pass |
| 2 | Нажать кнопку «Ввести» (нажимать при вводе каждого элемента отдельно) | Введенный элемент последовательности будет добавлен в строку и перенесен в поле ввода | Pass |
| 3 | Нажать кнопку «Запуск» | В поле вывода будет выведен результат вычислений, равный 3 | Pass |
| Результат | | | Pass |

# Таблица 5. Тест-кейс №5, контроль ввода

| Идентификатор | | | T05 |
| --- | --- | --- | --- |
| Заголовок | | | Проверка контроля ввода |
| Предусловие | | | Приложение запущена |
| Ожидаемый результат | | | Выведена ошибка, поле ввода очищено |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Ввести некорректное большое значение |  | Pass |
| 2 | Выбрать вариант рекурсии | Выведена формула рекурсии, выведена ошибка | Pass |
| 3 | Нажать кнопку «ОК» во всплывающем окне | Поле ввода очищено | Pass |
| Результат | | | Pass |

# 1.7 Пример работающей программы

Руководство пользователя и алгоритм работы с программой встроено в приложение.

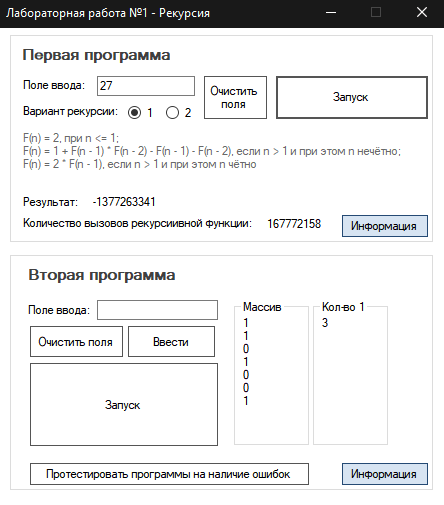


Рисунок 1 – Интерфейс программы

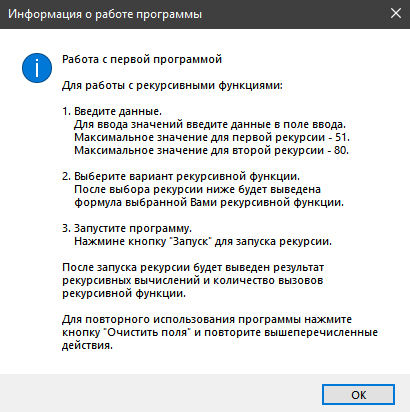


Рисунок 2 – Информационное окно для Первой части программы

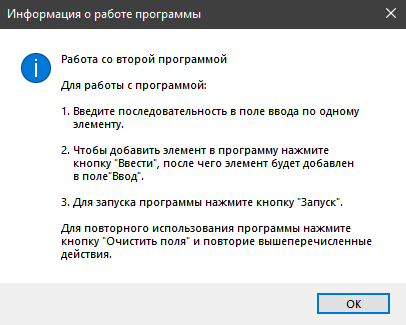


Рисунок 3 – Информационное окно для Второй части программы

# 2. Проектирование задачи

Реализовать приложение с графическим интерфейсом, используя библиотеку Windows Forms, для работы с рекурсиями. Встроенное тестирование, при помощи процедур и функций, проводимое при нажатии отдельной кнопки.

# 2.1 Наименование программы

Программа имеет название UniLab1.exe.

# 2.2 Уточненное словесное описание задачи

Написать код программы и графический интерфейс, в котором можно анализировать работу рекурсии. Для трёх вариантов рекурсии необходимо написать программы с их реализациями, также необходимо в интерфейсе иметь поле для ввода изначальных данных рекурсии, а также меню выбора рекурсии с выводом формулы (для удобства пользователя). Кроме того, должна быть кнопка для запуска рекурсии с введенным значением в поле изначальных данных. А также поле, которое отображает количество вызовов функций в рекурсии.

# 2.3 Выбор метода решения поставленной задачи

Для первой и второй рекурсивных функций происходит следующий процесс:

Работа алгоритма производится при помощи ввода данных в поле «Поле ввода», с последующим нажатием кнопки «Запуск», алгоритм начинает работать. При нажатии кнопки «Запуск» процедура конвертирует входные данные из ТД String^ в ТД unsigned short int, это значение передается в глобальную переменную n.

После чего, в зависимости от выбранной рекурсии, n передается в функцию выбранной рекурсии в качестве параметра. Во время выполнения алгоритма рекурсии в глобальную переменную int count записывается количество рекурсивных вызовов. По окончании работы алгоритма функция возвращает значение ТД int, которое структурой переводится в ТД String^ и выводится в поле «Результат», в качестве результата вычислений. Аналогичные действия проводятся со значением переменной int count, которые выводятся в поле «Количество вызовов рекурсивной функции».

Для третей рекурсивной функции происходит следующий процесс:  
Работа алгоритма производится при помощи ввода данных, в виде числовой последовательности в поле «Поле ввода», посредством ввода последовательности по одному элементу и нажатием кнопки «Ввести» (каждый элемент последовательности заносится в строку). Далее нажимается кнопка запуск, после которой создаётся поток строки и выполняется функция, которая поэлементно считает количество единиц в ряду до тех пор, пока не будет найдено два идущих подряд нуля или не закончится строка. После конечный результат выводится на экран в отведённое для этого поле.

При нажатии кнопки «Очистить поля» значения всех поля, измененных пользователем в данной области, будут изменены на значения по умолчанию.

Кнопки «Информация» выводят на экран всплывающее окно, в котором описан алгоритм работы с программой в данной области.

Кнопка «Протестировать программу на наличие ошибок» тестирует определенные функции на соответствие с ожидаемым результатом.

# Уточненные глобальные данные программы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Объект программы | Имя | Характеристики | | | Назначение |
| Тип данных | Диапазон значений | Тип функции |
| MainForm.h | n | Unsigned short int | 0 ­– 65535 | Простая | Ввод значений |
| MainForm.h | count | Unsigned int | 0 - 4294967295 | Простая | Счетчик рекурсивных вызовов |
| MainForm.h | choiceСheck | Bool | True/False | Простая | Переключатель для проверки выбора кнопки |

# Декомпозиция функций

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назначение | Имя | Параметры | | Процедура  или функция | Включена в проверку |
| Вход:  тип | Выход: тип |
| Вычисление рекурсивной функции от n №1 | recursion1 | Unsigned int | int | Функция | Да |
| Вычисление рекурсивной функции от n №2 | recursion2 | Unsigned int | int | Функция | Да |
| Счётчик единиц в строке до двух 0 подряд | unitsCounter | String | int | Функция | Да |
| Функция подготовки массива для вывода | ConvertToIntegerArray | String | String | Функция | Нет |
| Функция ограничений для первой рекурсивной формулы | validityCheckFor1 | int | int | Функция | Да |
| Функция ограничений для второй рекурсивной формулы | validityCheckFor2 | int | int | Функция | Да |
| Проверка изменеия текста в поле ввода первой программы | textField1TextChanged | Void | Void | Функция | Нет |
| Кнопка выбора первой рекурсивной формулы | radioBtn1\_CheckedChanged | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Кнопка выбора первой рекурсивной формулы | radioBtn2\_CheckedChanged | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Кнопка запуска первой программы | startBtn1\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Кнопка очистки полей первой программы | resetBtn1\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Кнопка информации о первой программе | infoBtn1\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Кнопка ввода элемента массива во второй программе | inputBtn\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Кнопка запуска второй программы | startBtn2\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Кнопка очистки полей второй программы | resetBtn2\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Кнопка информации о второй программе | infoBtn2\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Кнопка тестирования программы | errorTestBtn\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Тест работы первой рекурсивной функции | testRecursion1 | int | int | Функция | Нет |
| Тест работы второй рекурсивной функции | testRecursion2 | int | int | Функция | Нет |
| Тест перегрузки первой функции | testOverflow1 | int | int | Функция | Нет |
| Тест перегрузки второй функции | testOverflow2 | int | int | Функция | Нет |
| Тест счётчика единиц | testUnitsCounter | int | int | Функция | Нет |

# Блок-схемы:

– Блок-схема функции recursion1() в Приложении A;

– Блок-схема функции recursion2() в Приложении Б;

– Блок-схема функции unitsCounter() в Приложении В;

– Блок-схема процедуры startBtn1\_Click() в Приложении Г;

– Блок-схема процедуры startBtn2\_Click() в Приложении Д.

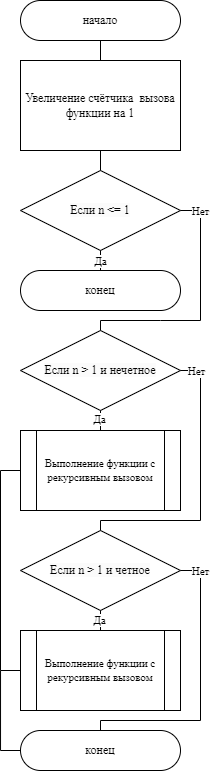
# Алгоритмизация:

– Листинг файла UniLab1.h в Приложение Е;

– Листинг файла MainForm.cpp в Приложение Ж.

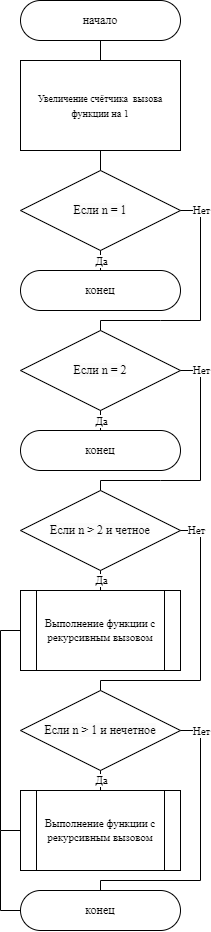
Приложение А

# Блок-схема функции recursion1()



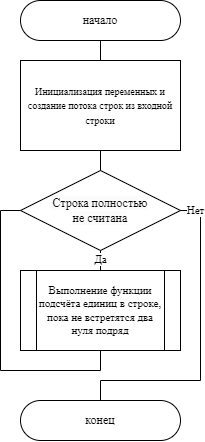
# Приложение Б

# Блок-схема функции recursion2()



# Приложение В

# Блок-схема функции unitsCounter()



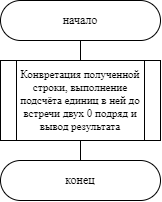
# Приложение Г

# Блок-схема процедуры startBtn1\_Click



# Приложение Д

# Блок-схема процедуры startBtn2\_Click



# Приложение Е

# Листинг программы, файл UniLab1.h

#pragma once

#include <iostream>

#include <string>

#include <sstream>

#include <vector>

#include <msclr/marshal\_cppstd.h>

namespace UniLab1 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace System::Collections::Generic;

using namespace std;

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

{

public:MainForm(void) {

InitializeComponent();

}

protected:~MainForm() {

if (components) {

delete components;

}

}

private: unsigned short int n; // Переменная для подстановки в рекурсивную функцию

private: unsigned int count = 0; // Переменная для подсчёта вывова рекрсивной функции

private: bool choiceСheck; // Переменная проверки выбора

private: System::Windows::Forms::Label^ fieldLabel1;

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

private: System::Windows::Forms::Label^ progLabel1;

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

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

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

private: System::Windows::Forms::GroupBox^ progBox1;

private: System::Windows::Forms::GroupBox^ progBox2;

private: System::Windows::Forms::Button^ resetBtn1;

private: System::Windows::Forms::Button^ infoBtn1;

private: System::Windows::Forms::Button^ startBtn1;

private: System::Windows::Forms::Button^ startBtn2;

private: System::Windows::Forms::Button^ resetBtn2;

private: System::Windows::Forms::Label^ progLabel2;

private: System::Windows::Forms::Button^ infoBtn2;

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

private: System::Windows::Forms::GroupBox^ groupBox4;

private: System::Windows::Forms::GroupBox^ groupBox3;

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

private: System::Windows::Forms::RadioButton^ radioBtn2;

private: System::Windows::Forms::RadioButton^ radioBtn1;

private: System::Windows::Forms::TextBox^ textField1;

private: System::Windows::Forms::TextBox^ textField2;

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

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

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

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

private: System::Windows::Forms::Label^ fieldLabel2;

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

void InitializeComponent(void)

{

this->fieldLabel1 = (gcnew System::Windows::Forms::Label());

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

this->progLabel1 = (gcnew System::Windows::Forms::Label());

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

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

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

this->progBox1 = (gcnew System::Windows::Forms::GroupBox());

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

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

this->textField1 = (gcnew System::Windows::Forms::TextBox());

this->radioBtn2 = (gcnew System::Windows::Forms::RadioButton());

this->radioBtn1 = (gcnew System::Windows::Forms::RadioButton());

this->infoBtn1 = (gcnew System::Windows::Forms::Button());

this->startBtn1 = (gcnew System::Windows::Forms::Button());

this->resetBtn1 = (gcnew System::Windows::Forms::Button());

this->progBox2 = (gcnew System::Windows::Forms::GroupBox());

this->fieldLabel2 = (gcnew System::Windows::Forms::Label());

this->textField2 = (gcnew System::Windows::Forms::TextBox());

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

this->groupBox4 = (gcnew System::Windows::Forms::GroupBox());

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

this->groupBox3 = (gcnew System::Windows::Forms::GroupBox());

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

this->infoBtn2 = (gcnew System::Windows::Forms::Button());

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

this->startBtn2 = (gcnew System::Windows::Forms::Button());

this->resetBtn2 = (gcnew System::Windows::Forms::Button());

this->progLabel2 = (gcnew System::Windows::Forms::Label());

this->progBox1->SuspendLayout();

this->progBox2->SuspendLayout();

this->groupBox4->SuspendLayout();

this->groupBox3->SuspendLayout();

this->SuspendLayout();

//

// fieldLabel1

//

this->fieldLabel1->AutoSize = true;

this->fieldLabel1->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->fieldLabel1->Location = System::Drawing::Point(10, 49);

this->fieldLabel1->Name = L"fieldLabel1";

this->fieldLabel1->Size = System::Drawing::Size(69, 13);

this->fieldLabel1->TabIndex = 0;

this->fieldLabel1->Text = L"Поле ввода:";

//

// radioLabel

//

this->radioLabel->AutoSize = true;

this->radioLabel->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->radioLabel->Location = System::Drawing::Point(10, 75);

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

this->radioLabel->Size = System::Drawing::Size(102, 13);

this->radioLabel->TabIndex = 2;

this->radioLabel->Text = L"Вариант рекурсии:";

//

// progLabel1

//

this->progLabel1->AutoSize = true;

this->progLabel1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->progLabel1->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->progLabel1->Location = System::Drawing::Point(9, 16);

this->progLabel1->Name = L"progLabel1";

this->progLabel1->Size = System::Drawing::Size(156, 18);

this->progLabel1->TabIndex = 8;

this->progLabel1->Text = L"Первая программа";

//

// formulaLabel

//

this->formulaLabel->AutoSize = true;

this->formulaLabel->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 8.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->formulaLabel->ForeColor = System::Drawing::SystemColors::ControlDarkDark;

this->formulaLabel->Location = System::Drawing::Point(10, 102);

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

this->formulaLabel->Size = System::Drawing::Size(329, 13);

this->formulaLabel->TabIndex = 10;

this->formulaLabel->Text = L"Здесь будет отображаться выбранный Вами вариант рекурсии";

//

// resultLabel

//

this->resultLabel->AutoSize = true;

this->resultLabel->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->resultLabel->Location = System::Drawing::Point(10, 166);

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

this->resultLabel->Size = System::Drawing::Size(62, 13);

this->resultLabel->TabIndex = 13;

this->resultLabel->Text = L"Результат:";

//

// countLabel

//

this->countLabel->AutoSize = true;

this->countLabel->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->countLabel->Location = System::Drawing::Point(10, 187);

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

this->countLabel->Size = System::Drawing::Size(236, 13);

this->countLabel->TabIndex = 14;

this->countLabel->Text = L"Количество вызовов рекурсиивной функции:";

//

// progBox1

//

this->progBox1->Controls->Add(this->countNum);

this->progBox1->Controls->Add(this->resultNum);

this->progBox1->Controls->Add(this->textField1);

this->progBox1->Controls->Add(this->radioBtn2);

this->progBox1->Controls->Add(this->radioBtn1);

this->progBox1->Controls->Add(this->infoBtn1);

this->progBox1->Controls->Add(this->startBtn1);

this->progBox1->Controls->Add(this->resetBtn1);

this->progBox1->Controls->Add(this->countLabel);

this->progBox1->Controls->Add(this->resultLabel);

this->progBox1->Controls->Add(this->formulaLabel);

this->progBox1->Controls->Add(this->progLabel1);

this->progBox1->Controls->Add(this->fieldLabel1);

this->progBox1->Controls->Add(this->radioLabel);

this->progBox1->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->progBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 8.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->progBox1->Location = System::Drawing::Point(12, 1);

this->progBox1->Name = L"progBox1";

this->progBox1->Size = System::Drawing::Size(423, 214);

this->progBox1->TabIndex = 15;

this->progBox1->TabStop = false;

//

// countNum

//

this->countNum->AutoSize = true;

this->countNum->Location = System::Drawing::Point(254, 188);

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

this->countNum->Size = System::Drawing::Size(0, 13);

this->countNum->TabIndex = 21;

//

// resultNum

//

this->resultNum->AutoSize = true;

this->resultNum->Location = System::Drawing::Point(80, 166);

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

this->resultNum->Size = System::Drawing::Size(0, 13);

this->resultNum->TabIndex = 20;

//

// textField1

//

this->textField1->Location = System::Drawing::Point(87, 47);

this->textField1->Name = L"textField1";

this->textField1->Size = System::Drawing::Size(98, 20);

this->textField1->TabIndex = 2;

this->textField1->TextChanged += gcnew System::EventHandler(this, &MainForm::textField1TextChanged);

//

// radioBtn2

//

this->radioBtn2->AutoSize = true;

this->radioBtn2->BackColor = System::Drawing::SystemColors::ControlLightLight;

this->radioBtn2->Cursor = System::Windows::Forms::Cursors::Hand;

this->radioBtn2->Location = System::Drawing::Point(156, 75);

this->radioBtn2->Name = L"radioBtn2";

this->radioBtn2->Size = System::Drawing::Size(31, 17);

this->radioBtn2->TabIndex = 19;

this->radioBtn2->TabStop = true;

this->radioBtn2->Text = L"2";

this->radioBtn2->UseVisualStyleBackColor = false;

this->radioBtn2->CheckedChanged += gcnew System::EventHandler(this, &MainForm::radioBtn2\_CheckedChanged);

//

// radioBtn1

//

this->radioBtn1->AutoSize = true;

this->radioBtn1->BackColor = System::Drawing::SystemColors::ControlLightLight;

this->radioBtn1->Cursor = System::Windows::Forms::Cursors::Hand;

this->radioBtn1->Location = System::Drawing::Point(118, 75);

this->radioBtn1->Name = L"radioBtn1";

this->radioBtn1->Size = System::Drawing::Size(31, 17);

this->radioBtn1->TabIndex = 18;

this->radioBtn1->TabStop = true;

this->radioBtn1->Text = L"1";

this->radioBtn1->UseVisualStyleBackColor = false;

this->radioBtn1->CheckedChanged += gcnew System::EventHandler(this, &MainForm::radioBtn1\_CheckedChanged);

//

// infoBtn1

//

this->infoBtn1->BackColor = System::Drawing::SystemColors::GradientInactiveCaption;

this->infoBtn1->Cursor = System::Windows::Forms::Cursors::Hand;

this->infoBtn1->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->infoBtn1->ForeColor = System::Drawing::SystemColors::ControlText;

this->infoBtn1->Location = System::Drawing::Point(332, 186);

this->infoBtn1->Name = L"infoBtn1";

this->infoBtn1->Size = System::Drawing::Size(86, 22);

this->infoBtn1->TabIndex = 17;

this->infoBtn1->Text = L"Информация";

this->infoBtn1->UseVisualStyleBackColor = false;

this->infoBtn1->Click += gcnew System::EventHandler(this, &MainForm::infoBtn1\_Click);

//

// startBtn1

//

this->startBtn1->BackColor = System::Drawing::SystemColors::ButtonHighlight;

this->startBtn1->Cursor = System::Windows::Forms::Cursors::Hand;

this->startBtn1->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->startBtn1->Font = (gcnew System::Drawing::Font(L"Lucida Fax", 8.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->startBtn1->Location = System::Drawing::Point(266, 47);

this->startBtn1->Name = L"startBtn1";

this->startBtn1->Size = System::Drawing::Size(152, 43);

this->startBtn1->TabIndex = 16;

this->startBtn1->Text = L"Запуск";

this->startBtn1->UseVisualStyleBackColor = false;

this->startBtn1->Click += gcnew System::EventHandler(this, &MainForm::startBtn1\_Click);

//

// resetBtn1

//

this->resetBtn1->BackColor = System::Drawing::SystemColors::ButtonHighlight;

this->resetBtn1->Cursor = System::Windows::Forms::Cursors::Hand;

this->resetBtn1->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->resetBtn1->Location = System::Drawing::Point(194, 47);

this->resetBtn1->Name = L"resetBtn1";

this->resetBtn1->Size = System::Drawing::Size(63, 43);

this->resetBtn1->TabIndex = 15;

this->resetBtn1->Text = L"Очистить поля";

this->resetBtn1->UseVisualStyleBackColor = false;

this->resetBtn1->Click += gcnew System::EventHandler(this, &MainForm::resetBtn1\_Click);

//

// progBox2

//

this->progBox2->Controls->Add(this->fieldLabel2);

this->progBox2->Controls->Add(this->textField2);

this->progBox2->Controls->Add(this->errorTestBtn);

this->progBox2->Controls->Add(this->groupBox4);

this->progBox2->Controls->Add(this->groupBox3);

this->progBox2->Controls->Add(this->infoBtn2);

this->progBox2->Controls->Add(this->inputBtn);

this->progBox2->Controls->Add(this->startBtn2);

this->progBox2->Controls->Add(this->resetBtn2);

this->progBox2->Controls->Add(this->progLabel2);

this->progBox2->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->progBox2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 8.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->progBox2->Location = System::Drawing::Point(12, 221);

this->progBox2->Name = L"progBox2";

this->progBox2->Size = System::Drawing::Size(423, 242);

this->progBox2->TabIndex = 16;

this->progBox2->TabStop = false;

//

// fieldLabel2

//

this->fieldLabel2->AutoSize = true;

this->fieldLabel2->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->fieldLabel2->Location = System::Drawing::Point(15, 54);

this->fieldLabel2->Name = L"fieldLabel2";

this->fieldLabel2->Size = System::Drawing::Size(69, 13);

this->fieldLabel2->TabIndex = 37;

this->fieldLabel2->Text = L"Поле ввода:";

//

// textField2

//

this->textField2->Location = System::Drawing::Point(87, 51);

this->textField2->Name = L"textField2";

this->textField2->Size = System::Drawing::Size(121, 20);

this->textField2->TabIndex = 21;

//

// errorTestBtn

//

this->errorTestBtn->BackColor = System::Drawing::SystemColors::ButtonHighlight;

this->errorTestBtn->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->errorTestBtn->Location = System::Drawing::Point(20, 214);

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

this->errorTestBtn->Size = System::Drawing::Size(279, 22);

this->errorTestBtn->TabIndex = 36;

this->errorTestBtn->Text = L"Протестировать программы на наличие ошибок";

this->errorTestBtn->UseVisualStyleBackColor = false;

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

//

// groupBox4

//

this->groupBox4->BackColor = System::Drawing::SystemColors::ControlLightLight;

this->groupBox4->Controls->Add(this->outputLabel);

this->groupBox4->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->groupBox4->Location = System::Drawing::Point(303, 51);

this->groupBox4->Name = L"groupBox4";

this->groupBox4->Size = System::Drawing::Size(75, 146);

this->groupBox4->TabIndex = 35;

this->groupBox4->TabStop = false;

this->groupBox4->Text = L"Кол-во 1";

//

// outputLabel

//

this->outputLabel->AutoSize = true;

this->outputLabel->Location = System::Drawing::Point(6, 16);

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

this->outputLabel->Size = System::Drawing::Size(0, 13);

this->outputLabel->TabIndex = 0;

//

// groupBox3

//

this->groupBox3->BackColor = System::Drawing::SystemColors::ControlLightLight;

this->groupBox3->Controls->Add(this->inputLabel);

this->groupBox3->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->groupBox3->Location = System::Drawing::Point(224, 51);

this->groupBox3->Name = L"groupBox3";

this->groupBox3->Size = System::Drawing::Size(75, 146);

this->groupBox3->TabIndex = 34;

this->groupBox3->TabStop = false;

this->groupBox3->Text = L"Массив";

//

// inputLabel

//

this->inputLabel->AutoSize = true;

this->inputLabel->Location = System::Drawing::Point(6, 16);

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

this->inputLabel->Size = System::Drawing::Size(0, 13);

this->inputLabel->TabIndex = 0;

//

// infoBtn2

//

this->infoBtn2->BackColor = System::Drawing::SystemColors::GradientInactiveCaption;

this->infoBtn2->Cursor = System::Windows::Forms::Cursors::Hand;

this->infoBtn2->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->infoBtn2->ForeColor = System::Drawing::SystemColors::ControlText;

this->infoBtn2->Location = System::Drawing::Point(332, 214);

this->infoBtn2->Name = L"infoBtn2";

this->infoBtn2->Size = System::Drawing::Size(86, 22);

this->infoBtn2->TabIndex = 18;

this->infoBtn2->Text = L"Информация";

this->infoBtn2->UseVisualStyleBackColor = false;

this->infoBtn2->Click += gcnew System::EventHandler(this, &MainForm::infoBtn2\_Click);

//

// inputBtn

//

this->inputBtn->BackColor = System::Drawing::SystemColors::ButtonHighlight;

this->inputBtn->Cursor = System::Windows::Forms::Cursors::Hand;

this->inputBtn->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->inputBtn->Location = System::Drawing::Point(118, 77);

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

this->inputBtn->Size = System::Drawing::Size(87, 31);

this->inputBtn->TabIndex = 30;

this->inputBtn->Text = L"Ввести";

this->inputBtn->UseVisualStyleBackColor = false;

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

//

// startBtn2

//

this->startBtn2->BackColor = System::Drawing::SystemColors::ButtonHighlight;

this->startBtn2->Cursor = System::Windows::Forms::Cursors::Hand;

this->startBtn2->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->startBtn2->Location = System::Drawing::Point(20, 114);

this->startBtn2->Name = L"startBtn2";

this->startBtn2->Size = System::Drawing::Size(188, 83);

this->startBtn2->TabIndex = 28;

this->startBtn2->Text = L"Запуск";

this->startBtn2->UseVisualStyleBackColor = false;

this->startBtn2->Click += gcnew System::EventHandler(this, &MainForm::startBtn2\_Click);

//

// resetBtn2

//

this->resetBtn2->BackColor = System::Drawing::SystemColors::ButtonHighlight;

this->resetBtn2->Cursor = System::Windows::Forms::Cursors::Hand;

this->resetBtn2->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->resetBtn2->Location = System::Drawing::Point(20, 77);

this->resetBtn2->Name = L"resetBtn2";

this->resetBtn2->Size = System::Drawing::Size(93, 31);

this->resetBtn2->TabIndex = 27;

this->resetBtn2->Text = L"Очистить поля";

this->resetBtn2->UseVisualStyleBackColor = false;

this->resetBtn2->Click += gcnew System::EventHandler(this, &MainForm::resetBtn2\_Click);

//

// progLabel2

//

this->progLabel2->AutoSize = true;

this->progLabel2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->progLabel2->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->progLabel2->Location = System::Drawing::Point(15, 16);

this->progLabel2->Name = L"progLabel2";

this->progLabel2->Size = System::Drawing::Size(155, 18);

this->progLabel2->TabIndex = 20;

this->progLabel2->Text = L"Вторая программа";

//

// MainForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

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

this->BackColor = System::Drawing::SystemColors::ControlLightLight;

this->ClientSize = System::Drawing::Size(447, 477);

this->Controls->Add(this->progBox2);

this->Controls->Add(this->progBox1);

this->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 8.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->ForeColor = System::Drawing::SystemColors::ControlText;

this->MaximizeBox = false;

this->Name = L"MainForm";

this->RightToLeft = System::Windows::Forms::RightToLeft::No;

this->ShowIcon = false;

this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;

this->Text = L"Лабораторная работа №1 - Рекурсия";

this->progBox1->ResumeLayout(false);

this->progBox1->PerformLayout();

this->progBox2->ResumeLayout(false);

this->progBox2->PerformLayout();

this->groupBox4->ResumeLayout(false);

this->groupBox4->PerformLayout();

this->groupBox3->ResumeLayout(false);

this->groupBox3->PerformLayout();

this->ResumeLayout(false);

}

#pragma endregion

// -- [ Функции ] --

// Рекурсивная функция для первой формулы

int recursion1(unsigned int n) {

++count;

if (n <= 1) {

return 2;

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

return ((1 + ((recursion1(n - 1)) \* (recursion1(n - 2)))) - (recursion1(n - 1)) - (recursion1(n - 2)));

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

return (2 \* (recursion1(n - 1)));

}

return 0;

}

// Рекурсивная функция для второй формулы

int recursion2(unsigned int n) {

++count;

if (n == 1) {

return 1;

} else if (n == 2) {

return 2;

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

return ((7 \* n + (recursion2(n - 3))) / 9);

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

return ((5 \* n + (recursion2(n - 1)) + (recursion2(n - 2))) / 7);

}

return 0;

}

// Функция для второй программы - счётчик единиц

int unitsCounter(const string& input) {

int unitCnt = 0;

int prevNumber = -1;

stringstream ss(input);

string token;

while (ss >> token) {

int number = stoi(token);

if (number == 1) {

unitCnt++;

}

if (number == 0 && prevNumber == 0) {

break;

}

prevNumber = number;

}

return unitCnt;

}

// -- [ Функции проверки изменения полей ] --

// Проверка для первой функции

int validityCheckFor1() {

String^ errorOverFlow1 = "В целях безопасности для рекурсии первого вида программа не принимает значения выше 30.";

String^ errorOverFlow2 = "При выборе первого варианта рекурсивной функции вводимое значение, которое больше единицы, должно быть нечётным.";

try {

if (this->radioBtn1->Checked == true && this->textField1->Text != "") {

int dataN = System::Convert::ToInt32(this->textField1->Text);

if (dataN > 30) {

throw errorOverFlow1;

} else if (dataN < 1 || dataN % 2 == 0) {

throw errorOverFlow2;

}

}

} catch (String^ errorOverFlow) {

this->textField1->ForeColor = Color::Red;

System::Windows::Forms::DialogResult result = MessageBox::Show(errorOverFlow, "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

if (result == System::Windows::Forms::DialogResult::OK) {

this->textField1->Text = "";

this->textField1->ForeColor = Color::Black;

this->radioBtn1->Checked = false;

this->radioBtn2->Checked = false;

}

}

return 0;

}

// Проверка для второй функции

int validityCheckFor2() {

String^ errorOverFlow1 = "В целях безопасности для рекурсии второго вида программа не принимает значения выше 80.";

String^ errorOverFlow2 = "При выборе второй рекурсивной функции вводимое значение должно быть не меньше единицы.";

try {

if (this->radioBtn2->Checked == true && this->textField1->Text != "") {

int dataN = System::Convert::ToInt32(this->textField1->Text);

if (dataN > 80) {

throw errorOverFlow1;

} else if (dataN < 1) {

throw errorOverFlow2;

}

}

} catch (String^ errorOverFlow) {

this->textField1->ForeColor = Color::Red;

System::Windows::Forms::DialogResult result = MessageBox::Show(errorOverFlow, "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

if (result == System::Windows::Forms::DialogResult::OK) {

this->textField1->Text = "";

this->textField1->ForeColor = Color::Black;

this->radioBtn1->Checked = false;

this->radioBtn2->Checked = false;

}

}

return 0;

}

// Проверка изменения текста в поле ввода первой программы

System::Void textField1TextChanged(System::Object^ sender, System::EventArgs^ e) {

this->radioBtn1->Checked = false;

this->radioBtn2->Checked = false;

this->resultNum->Text = "";

this->countNum->Text = "";

validityCheckFor1();

validityCheckFor2();

};

// -- [ Кнопки ] --

// Кнопка выбора первой рекурсивной формулы

private: System::Void radioBtn1\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

validityCheckFor1();

if (this->radioBtn1->Checked == true) {

this->formulaLabel->Text = "F(n) = 2, при n <= 1;\r\nF(n) = 1 + F(n - 1) \* F(n - 2) - F(n - 1) - F(n - 2), если n > 1 и при этом n нечётно;\r\nF(n) = 2 \* F(n - 1), если n > 1 и при этом n чётно";

choiceСheck = true;

} else {

this->formulaLabel->Text = "Здесь будет отображаться выбранный Вами вариант рекурсии";

}

}

// Кнопка выбора второй рекурсивной функции

private: System::Void radioBtn2\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

validityCheckFor2();

if (this->radioBtn2->Checked == true) {

this->formulaLabel->Text = "F(n) = 1, при n = 1;\r\nF(n) = 2, при n = 2;\r\nF(n) = [(7 \* n + F(n - 3)) / 9], если n > 2 и при этом n чётно;\r\nF(n) = [(5 \* n + F(n - 1) + F(n - 2)) / 7], если n > 2 и при этом n нечётно";

choiceСheck = false;

} else {

this->formulaLabel->Text = "Здесь будет отображаться выбранный Вами вариант рекурсии";

}

}

// Кнопка запуска первой программы

private: System::Void startBtn1\_Click(System::Object^ sender, System::EventArgs^ e) {

if (this->textField1->Text == "") {

MessageBox::Show("Введите значение в \"Поле ввода\" для начала подсчета.", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

}

if (this->radioBtn1->Checked == false && this->radioBtn2->Checked == false) {

MessageBox::Show("Выберите один из двух вариантов рекурсии для начала подсчета.", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

}

int n = System::Convert::ToInt32(this->textField1->Text);

if (this->radioBtn1->Checked) {

this->resultNum->Text = System::Convert::ToString(recursion1(n));

} else if (this->radioBtn2->Checked) {

this->resultNum->Text = System::Convert::ToString(recursion2(n));

}

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

}

// Кнопка очистки полей первой программы

private: System::Void resetBtn1\_Click(System::Object^ sender, System::EventArgs^ e) {

this->textField1->Text = "";

this->formulaLabel->Text = "Здесь будет отображаться выбранный Вами вариант рекурсии";

this->resultNum->Text = "";

this->countNum->Text = "";

this->radioBtn1->Checked = false;

this->radioBtn2->Checked = false;

}

// Кнопка информации о первой программе

private: System::Void infoBtn1\_Click(System::Object^ sender, System::EventArgs^ e) {

MessageBox::Show("Работа с первой программой\n\nДля работы с рекурсивными функциями:\r\n\r\n1) Введите данные.\n Для ввода значений введите данные в поле ввода.\n Максимальное значение для первой рекурсии - 30.\n Максимальное значение для второй рекурсии - 80.\r\n\r\n2) Выберите вариант рекурсивной функции.\n После выбора рекурсии ниже будет выведена\n формула выбранной Вами рекурсивной функции.\r\n\r\n3) Запустите программу.\n Нажмите кнопку \"Запуск\" для запуска рекурсии.\r\n\r\nПосле запуска рекурсии будет выведен результат рекурсивных вычислений и количество вызовов рекурсивной функции.\r\n\r\nДля повторного использования программы нажмите кнопку \"Очистить поля\" и повторите вышеперечисленные действия.", "Информация о работе программы", MessageBoxButtons::OK, MessageBoxIcon::Information);

}

// Кнопка ввода элемента массива во второй программе

private: System::Void inputBtn\_Click(System::Object^ sender, System::EventArgs^ e) {

if (this->textField2->Text == "") {

MessageBox::Show("Введите значение элемента массива в \"Поле ввода\" прежде чем нажать на кнопку \"Ввести\".", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

}

if (inputLabel->Text->Split('\n')->Length - 1 >= 9) {

MessageBox::Show("Максимальное количество элементов в массиве достигнуто.", "Предупреждение", MessageBoxButtons::OK, MessageBoxIcon::Warning);

return;

}

int num = Convert::ToInt32(this->textField2->Text);

this->textField2->Text = "";

inputLabel->Text += num + "\r\n";

}

// Кнопка запуска второй программы

private: System::Void startBtn2\_Click(System::Object^ sender, System::EventArgs^ e) {

string output = msclr::interop::marshal\_as<string>(inputLabel->Text);

int count = unitsCounter(output);

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

}

// Кнопка очистки полей второй программы

private: System::Void resetBtn2\_Click(System::Object^ sender, System::EventArgs^ e) {

this->inputLabel->Text = "";

this->outputLabel->Text = "";

this->textField2->Text = "";

}

// Кнопка информации о второй программе

private: System::Void infoBtn2\_Click(System::Object^ sender, System::EventArgs^ e) {

MessageBox::Show("Работа со второй программой\r\n\r\nДля работы с программой:\r\n\r\n1) Введите последовательность в поле ввода по одному\n элементу.\r\n\r\n2) Чтобы добавить элемент в программу нажмите\n кнопку \"Ввести\", после чего элемент будет добавлен\n в поле \"Ввод\".\r\n\r\n3) Для запуска программы нажмите кнопку \"Запуск\".\r\n\r\nДля повторного использования программы нажмите кнопку \"Очистить поля\" и повторите вышеперечисленные действия.", "Информация о работе программы", MessageBoxButtons::OK, MessageBoxIcon::Information);

}

// Кнопка тестирования программы

private: System::Void errorTestBtn\_Click(System::Object^ sender, System::EventArgs^ e) {

int allTests = testRecursion1() + testRecursion2() + testOverflow1() + testOverflow2() + testUnitsCounter();

if (allTests == 0) {

MessageBox::Show("Все тесты пройдены. Программа работает корректно.", "Пройдено", MessageBoxButtons::OK, MessageBoxIcon::Information);

}

else {

Windows::Forms::DialogResult result = MessageBox::Show("Тесты не пройдены. Программа работает не корректно. Окно программы будет закрыто.", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

if (result == System::Windows::Forms::DialogResult::OK)

{

this->Close();

}

}

}

// -- [ ТЕСТЫ ] --

// Тест работы первой рекурсивной функции

int testRecursion1() {

if (recursion1(5) == 10) {

return 0;

} else {

return 1;

}

}

// Тест работы второй рекурсивной функции

int testRecursion2() {

if (recursion2(5) == 4) {

return 0;

} else {

return 1;

}

}

// Тест перегрузки первой функции

int testOverflow1() {

recursion1(31);

if (validityCheckFor1() == 0) {

return 0;

} else {

return 1;

}

}

// Тест перегрузки второй функции

int testOverflow2() {

recursion2(81);

if (validityCheckFor2() == 0) {

return 0;

} else {

return 1;

}

}

// Тест для второй программы

int testUnitsCounter() {

string input1 = "1 1 0 0 1 1";

string input2 = "1 0 1 0 0 1";

string input3 = "1 1 1 0 1 1";

string input4 = "0 0 1 1 1 1";

string input5 = "1 1 1 1 1 1 1 1 1";

if (unitsCounter(input1) == 2 && unitsCounter(input2) == 2 && unitsCounter(input3) == 5 && unitsCounter(input4) == 0 && unitsCounter(input5) == 9)

{

return 0;

} else {

return 1;

}

}

};

}

# Приложение И

# Листинг программы, файл MainForm.cpp

#include "MainForm.h"

using namespace::System;

using namespace::System::Windows::Forms;

[STAThreadAttribute]

int main()

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

UniLab1::MainForm form;

Application::Run(% form);

}