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 3091  / Р. А. Михайлов  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

1. **Постановка задачи**

Разработка программы с графическим интерфейсом для работы с рекурсиями.

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

Разработать программу с графическим интерфейсом для работы с рекурсиями.

Требования к дополнительной программе: дана последовательность натуральных чисел (одно число в строке), завершающаяся числом 0. Определите наибольшее значение числа в этой последовательности. В этой задаче нельзя использовать глобальные переменные и передавать какие-либо параметры в рекурсивную функцию. Функция получает данные, считывая их с клавиатуры. Функция возвращает единственное значение: максимум считанной последовательности. Гарантируется, что последовательность содержит хотя бы одно число (кроме нуля).

| Объект программы | Имя этого объекта в программе | Тип данных | Диапазоны представления | Простая/  структура | Вход/  выход/  Константа |
| --- | --- | --- | --- | --- | --- |
| Целое число из поля ввода | n | unsigned short int | 0 - 64 | Простая | Вход |
| Целое число (элемент последовательности) из поля ввода | n\_dop | unsigned int | 0-200000000 | Простая | Вход |
| Последовательность значений | arr | unsigned int | 250 элементов | Структура | Вход/выход |

**Внешние спецификации данных**

* 1. **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 (алгоритм работы с приложением).

* 1. **Математическая формулировка задачи**

Для всех вводимых значений в программе выбран ТД 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 – 64, при значении большем 65 происходит переполнение стека, вывод некорректных.

Формула второй рекурсивной функции:  
F(n) = 1, n < 3;  
F(n) = F(n - 1) + F(n - 2), если n > 2 и при этом n нечётное;  
F(n) = ∑F(i) 1<=i<=n-1, если n > 2 и при этом n чётное;

Ограничение ввода для второй рекурсии – 35, при расчете от 36 и больше получается слишком большое значение, происходит остановка программы.

Для хранения последовательности значений выбрана структура System::array<UInt32>, при попытке использования std::vector происходит ошибка синтаксиса, член класса управляемый не может относиться к типу класса не управляемый.

* 1. **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 предусмотрена встроенное тестирование при запуске программы, и отдельная кнопка для тестирования работоспособности ключевых расчетных функций.  
В приложении реализована работа с рекурсивными функция, ввод/вывод, очистка полей, кнопки, выводящие справочную информацию о работы программы.

* 1. **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 LAB\_1.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 элемент последовательности, 5 и 0 |  | Pass |
| 2 | Нажать кнопку «Ввести» | Введенный элемент последовательности будет добавлен в массив и перенесен в поле ввода | Pass |
| 3 | Нажать кнопку «Запуск» | В поле вывода будет выведен результат вычислений, 5 | 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. **Пример работающей программы**

Руководство пользователя и алгоритм работы с программой встроено в приложение.

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рисунок 1. Интерфейс приложения

Изображение выглядит как текст, электроника, снимок экрана, Операционная система

Автоматически созданное описание

Рисунок 2. Информационное окно, инструкция №1

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 3. Информационное окно, инструкция №2

1. **Проектирование задачи**

Реализовать приложение с графическим интерфейсом, использую библиотеку Windows Forms, для работы с рекурсиями. Модульное тестирование, используя фреймворк CppUnitTest и встроенное тестирование, при помощи процедур и функций, проводимое как при запуске программы, так и при нажатии отдельной кнопки.

* 1. **Наименование программы**

LAB\_1.exe

* 1. **Уточненное словесное описание задачи**

Написать код программы и графический интерфейс, в которых 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). Также должна быть кнопка для запуска рекурсии с введенным значением в поле изначальных данных. И поле, которое отображает количество вызовов функций в рекурсии.

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

Для первой и второй рекурсий:  
Работа алгоритма производится при помощи ввода данных в поле «Поле ввода», с последующим нажатием кнопки «Запуск», алгоритм начинает работать. При нажатии кнопки «Запуск» процедура переводит 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 q записывается количество рекурсивных вызовов. По окончании работы алгоритма функция возвращает значение ТД int, которое переводится в ТД String^ и выводится в поле «Вывод», в качестве результата вычислений. Аналогичные действия проводятся со значением переменной int q, которые выводятся в поле «Количество вызовов рекурсивной функции».

Для дополнительной программы:  
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 массив элементов ТД int arr[255], в качестве индекса элемента выступает глобальная переменная unsigned short int i, которая увеличивается на 1 после каждого нажатия кнопки «Ввести» последовательность должна оканчиваться «0»). После ввода последовательности пользователь нажимает кнопку «Запуск», тем самым запуская алгоритм, и меняя значение глобальной переменной unsigned short int i на 0.

В рекурсивную функцию передается по отдельности элементы последовательности из массива int arr[255], постепенно увеличивая значение глобальной переменной unsigned short int i на 1. После передачи последнего элемента, равного 0, происходит рекурсивный подъём. В итоге функция возвращает значение int, которое переводится в ТД String^, после чего выводится в поле «Вывод». Аналогичные действия проводятся со значением переменной int q, которые выводятся в поле «Количество вызовов рекурсивной функции».

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 определенный функции на соответствие с ожидаемым результатом.

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Объект программы | Имя этого объекта в программе | Характеристики | | | Как используется в программе |
| Тип данных | Диапазоны представления | Прост/структура/ константа |
| LAB\_1.h | n | Unsigned short int | 0 ­– 65535 | Простая | Ввод значений |
| LAB\_1.h | n\_dop | Unsigned int | 0 - 4294967295 | Простая | Ввод значений |
| LAB\_1.h | q | Unsigned int | 0 - 4294967295 | Простая | Счетчик рекурсивных вызовов |
| LAB\_1.h | i | Unsigned short int | 0 ­– 65535 | Простая | Индексация элементов в массиве |
| LAB\_1.h | Choice\_check | Bool | True/false | Простая | Фиксация выбора рекурсии |
| LAB\_1.h | Double\_click\_check | Bool | True/false | Простая | Ограничение двойного нажатия кнопки |
| LAB\_1.h | arr | Array<UInt32>^ | 0 - 4294967295 | Структура | Массив элементов последовательности |

* 1. **Декомпозиция функций**

Описывается каждая функция и процедура в программе

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назначение | Имя | Параметры | | Процедура/ функция | Тестируется/  не тестируется |
| Вход:  тип | Выход: тип |
| Вычисление рекурсивной функции от n №1 | recursion\_1 | unsigned int | int | Функция | Да |
| Вычисление рекурсивной функции от n №2 | recursion\_2 | Unsigned int | int | Функция | Да |
| Вывод формулы рекурсивной функции №1 | radioButton1\_activity | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Вывод формулы рекурсивной функции №1 | radioButton2\_activity | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Проверка входных данных рекурсии №1 | check\_validaty\_for\_1 |  | void | Процедура | Нет |
| Проверка входных данных рекурсии №2 | check\_validaty\_for\_2 |  | void | Процедура | Нет |
| Проверка входных данных на наличие символов, кроме 0–9, в области 1 и 2 рекурсии | null\_to\_nine |  | void | Процедура | Нет |
| Ограничение входных данных на максимальное значение, в области 1 и 2 рекурсии | max\_value\_input |  | void | Процедура | Нет |
| Процедура проверки входных данных, в области 1 и 2 рекурсии | check\_validaty\_1\_2 |  | void | Процедура | Нет |
| Процедура, описывающая действия при нажатии кнопки "Запуск", в области 1 и 2 рекурсии | run\_btn\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Процедура, описывающая действия при нажатии кнопки "Очистить поля", в области 1 и 2 рекурсии | button1\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Процедура, описывающая действия при нажатии кнопки "Информация", в области 1 и 2 рекурсии | info\_btn\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Процедура описывающая действия при изменении данных в поле ввода, в области 1 и 2 рекурсии | input\_textBox\_TextChanged | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Определение максимального элемента из двух, для дополнительной программы | max | Unsigned int, Unsigned int | Unsigned int | Функция | Да |
| Основная функция дополнительной программы, определяет максимальный элемент последовательности | dop |  | Unsigned int | Функция | Да |
| Процедура, описывающая действия при нажатии кнопки "Ввести" в области дополнительной программы | button2\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Процедура, описывающая действия при нажатии кнопки "Очистить поля" в области дополнительной программы | dop\_clear\_btn\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Процедура, описывающая действия при нажатии кнопки "Запуск" в области дополнительной программы | dop\_run\_btn\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Процедура, ограничивающая введение первым элементом последовательности числа "0", в области дополнительной программы | dop\_zero\_first |  | Void | Процедура | Нет |
| Проверка входных данных на наличие символов, кроме 0-9, в области дополнительной программы | dop\_valid\_input |  | Void | Процедура | Нет |
| Процедура, ограничивающая ввод числом 200.000.000, в области дополнительной программы | dop\_max\_value\_input |  | Void | Процедура | Нет |
| Процедура, ограничивающая повторный ввод значений после предыдущих вычислений, в области дополнительной программы | double\_input\_protect |  | Void | Процедура | Нет |
| Функция, удаляющая первые незначащие нули числа, в области дополнительной программы | delete\_first\_null | String^ | String^ | Функция | Нет |
| Процедура, описывающая действия при изменении данных в поле ввода, в области дополнительной программы | textBox2\_TextChanged | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Процедура описывающая действия при нажатии кнопки "Информация", в области дополнительной программы | dop\_info\_btn\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Тест первой рекурсивной функции, при аргументе = 1, ожидаемый результат 1 | test\_1 |  | Int | Функция | Нет |
| Тест первой рекурсивной функции, при аргументе = 2 ожидаемый результат 5 | test\_2 |  | Int | Функция | Нет |
| Тест второй рекурсивной функции, при аргументе = 2 ожидаемый результат 1 | test\_3 |  | Int | Функция | Нет |
| Тест второй рекурсивной функции, при аргументе = 3 ожидаемый результат 2 | test\_4 |  | Int | Функция | Нет |
| Тест функции по определению максимального элемента из двух, при аргументах = 1 и 0 ожидаемый результат 1 | test\_5 |  | Int | Функция | Нет |
| Тест дополнительной рекурсивной функции по определению максимального элемента последовательности, ожидаемый результат 4 | test\_6 |  | Int | Функция | Нет |
| Процедура запускающая тестирование четырёх функций | test\_cases |  | Void | Процедура | Нет |
| Процедура, передаваемая в конструктор формы, запускающая тестирование четырёх функций | test\_cases\_constructor |  | void | Процедура | Нет |
| Процедура, описывающая действия при нажатии кнопки "Протестировать программу на наличие ошибок" | test\_btn\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |

* 1. **Блок-схема**

Блок-схемы функций recursion\_1() в Приложении A.  
Блок-схемы функций recursion\_2() в Приложении Б.

Блок-схемы функций dop() в Приложении В.

Блок-схемы процедуры run\_btn\_Click() в Приложении Г.

Блок-схемы процедуры dop\_run\_btn\_Click () в Приложении Д.

* 1. **Алгоритмизация**

Листинг файла LAB\_1.h в Приложение Е.

Листинг файла MyForm.cpp в Приложение Ж.

Листинг файла UnitTest.cpp в Приложение И.

Листинг файла Functions.h в Приложение К.

Приложение А

**Блок-схема функции recursion\_1()**

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Приложение Б

**Блок-схема функции recursion\_2()**

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Приложение В

**Блок-схема функции dop()**

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Приложение Г

**Блок-схема процедуры run\_btn\_Click()**

Изображение выглядит как текст, снимок экрана, Шрифт, графический дизайн

Автоматически созданное описание

Приложение Д

**Блок-схема процедуры dop\_run\_btn\_Click()**

Изображение выглядит как текст, снимок экрана, Шрифт, графический дизайн

Автоматически созданное описание

Приложение Е

**Листинг программы, файл LAB\_1.h**

#pragma once

namespace LAB\_1 {

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 LAB\_1

/// </summary>

public ref class LAB\_1 : public System::Windows::Forms::Form

{

public:

LAB\_1(void)

{

InitializeComponent();

test\_cases\_constructor();

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~LAB\_1()

{

if (components)

{

delete components;

}

}

// Пользовательские поля

private: unsigned short int n; // Глобальная переменна, используемая для передачи в рекурсивную функцию значения из поля ввода, в области 1 и 2 рекурсии

private: unsigned int n\_dop; // Глобальная переменна, используемая для передачи в рекурсивную функцию значения из поля ввода, в области дополнительной программы

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

private: unsigned short int i = 0; // Глобальная переменная, используемая для инициализации массива int и передачи из него значений в рекурсивную функцию, в области дополнительной программы

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

private: bool double\_click\_bool = false; // Булевая переменная, ограничивающая повторный запуск рекурсии, в области дополнительной программы

private: array<UInt32>^ arr = gcnew array<UInt32>(250); // Массив int, используемый для фиксирования введенной последовательности, в области дополнительной программы

//

private: System::Windows::Forms::Button^ info\_btn;

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

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

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

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::RadioButton^ radio\_button\_2;

private: System::Windows::Forms::RadioButton^ radio\_button\_1;

private: System::Windows::Forms::Button^ run\_btn;

private: System::Windows::Forms::TextBox^ textBox3;

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

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

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

private: System::Windows::Forms::Label^ call\_quantity\_label;

private: System::Windows::Forms::Label^ call\_quantity\_output;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::TextBox^ dop\_input;

private: System::Windows::Forms::TextBox^ dop\_output;

private: System::Windows::Forms::Button^ dop\_clear\_btn;

private: System::Windows::Forms::Button^ dop\_info\_btn;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::TextBox^ textBox2;

private: System::Windows::Forms::Button^ button2;

private: System::Windows::Forms::Button^ dop\_run\_btn;

private: System::Windows::Forms::Button^ test\_btn;

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

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)

{

this->info\_btn = (gcnew System::Windows::Forms::Button());

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

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

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

this->button1 = (gcnew System::Windows::Forms::Button());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->radio\_button\_2 = (gcnew System::Windows::Forms::RadioButton());

this->radio\_button\_1 = (gcnew System::Windows::Forms::RadioButton());

this->run\_btn = (gcnew System::Windows::Forms::Button());

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

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

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

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

this->call\_quantity\_output = (gcnew System::Windows::Forms::Label());

this->label2 = (gcnew System::Windows::Forms::Label());

this->dop\_input = (gcnew System::Windows::Forms::TextBox());

this->dop\_output = (gcnew System::Windows::Forms::TextBox());

this->dop\_clear\_btn = (gcnew System::Windows::Forms::Button());

this->dop\_info\_btn = (gcnew System::Windows::Forms::Button());

this->label3 = (gcnew System::Windows::Forms::Label());

this->label4 = (gcnew System::Windows::Forms::Label());

this->textBox2 = (gcnew System::Windows::Forms::TextBox());

this->button2 = (gcnew System::Windows::Forms::Button());

this->dop\_run\_btn = (gcnew System::Windows::Forms::Button());

this->test\_btn = (gcnew System::Windows::Forms::Button());

this->SuspendLayout();

//

// info\_btn

//

this->info\_btn->Location = System::Drawing::Point(12, 62);

this->info\_btn->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);

this->info\_btn->Name = L"info\_btn";

this->info\_btn->Size = System::Drawing::Size(225, 41);

this->info\_btn->TabIndex = 0;

this->info\_btn->Text = L"Информация";

this->info\_btn->UseVisualStyleBackColor = true;

this->info\_btn->Click += gcnew System::EventHandler(this, &LAB\_1::info\_btn\_Click);

//

// input\_label

//

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

this->input\_label->Location = System::Drawing::Point(14, 25);

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

this->input\_label->Size = System::Drawing::Size(104, 20);

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

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

//

// input\_textBox

//

this->input\_textBox->Location = System::Drawing::Point(123, 22);

this->input\_textBox->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);

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

this->input\_textBox->Size = System::Drawing::Size(110, 26);

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

this->input\_textBox->TextChanged += gcnew System::EventHandler(this, &LAB\_1::input\_textBox\_TextChanged);

//

// choise\_label

//

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

this->choise\_label->Location = System::Drawing::Point(248, 24);

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

this->choise\_label->Size = System::Drawing::Size(134, 20);

this->choise\_label->TabIndex = 3;

this->choise\_label->Text = L"Выбор рекурсии:";

//

// button1

//

this->button1->Location = System::Drawing::Point(251, 62);

this->button1->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(235, 41);

this->button1->TabIndex = 10;

this->button1->Text = L"Очистить поля";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &LAB\_1::button1\_Click);

//

// textBox1

//

this->textBox1->BorderStyle = System::Windows::Forms::BorderStyle::None;

this->textBox1->Cursor = System::Windows::Forms::Cursors::Default;

this->textBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->textBox1->Location = System::Drawing::Point(18, 118);

this->textBox1->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);

this->textBox1->Multiline = true;

this->textBox1->Name = L"textBox1";

this->textBox1->ReadOnly = true;

this->textBox1->Size = System::Drawing::Size(594, 86);

this->textBox1->TabIndex = 11;

this->textBox1->Text = L"Здесь будет формула выбранной рекурсивной функции";

//

// radio\_button\_2

//

this->radio\_button\_2->AutoSize = true;

this->radio\_button\_2->Location = System::Drawing::Point(443, 25);

this->radio\_button\_2->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);

this->radio\_button\_2->Name = L"radio\_button\_2";

this->radio\_button\_2->Size = System::Drawing::Size(43, 24);

this->radio\_button\_2->TabIndex = 9;

this->radio\_button\_2->TabStop = true;

this->radio\_button\_2->Text = L"2";

this->radio\_button\_2->UseVisualStyleBackColor = true;

this->radio\_button\_2->CheckedChanged += gcnew System::EventHandler(this, &LAB\_1::radioButton2\_activity);

//

// radio\_button\_1

//

this->radio\_button\_1->AutoSize = true;

this->radio\_button\_1->Location = System::Drawing::Point(389, 25);

this->radio\_button\_1->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);

this->radio\_button\_1->Name = L"radio\_button\_1";

this->radio\_button\_1->Size = System::Drawing::Size(43, 24);

this->radio\_button\_1->TabIndex = 8;

this->radio\_button\_1->TabStop = true;

this->radio\_button\_1->Text = L"1";

this->radio\_button\_1->UseVisualStyleBackColor = true;

this->radio\_button\_1->CheckedChanged += gcnew System::EventHandler(this, &LAB\_1::radioButton1\_activity);

//

// run\_btn

//

this->run\_btn->Location = System::Drawing::Point(503, 22);

this->run\_btn->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);

this->run\_btn->Name = L"run\_btn";

this->run\_btn->Size = System::Drawing::Size(109, 81);

this->run\_btn->TabIndex = 12;

this->run\_btn->Text = L"Запуск";

this->run\_btn->UseVisualStyleBackColor = true;

this->run\_btn->Click += gcnew System::EventHandler(this, &LAB\_1::run\_btn\_Click);

//

// label1

//

this->label1->AutoSize = true;

this->label1->BackColor = System::Drawing::SystemColors::Control;

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 20, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label1->Location = System::Drawing::Point(-7, 229);

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

this->label1->Size = System::Drawing::Size(636, 46);

this->label1->TabIndex = 17;

this->label1->Text = L"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

//

// output\_label

//

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

this->output\_label->Location = System::Drawing::Point(76, 206);

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

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

this->output\_label->TabIndex = 13;

//

// label1\_output

//

this->label1\_output->AutoSize = true;

this->label1\_output->Location = System::Drawing::Point(11, 206);

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

this->label1\_output->Size = System::Drawing::Size(64, 20);

this->label1\_output->TabIndex = 14;

this->label1\_output->Text = L"Вывод:";

//

// call\_quantity\_label

//

this->call\_quantity\_label->AutoSize = true;

this->call\_quantity\_label->Location = System::Drawing::Point(11, 238);

this->call\_quantity\_label->Name = L"call\_quantity\_label";

this->call\_quantity\_label->Size = System::Drawing::Size(341, 20);

this->call\_quantity\_label->TabIndex = 15;

this->call\_quantity\_label->Text = L"Количество вызовов рекурсивной функции:";

//

// call\_quantity\_output

//

this->call\_quantity\_output->AutoSize = true;

this->call\_quantity\_output->Location = System::Drawing::Point(355, 238);

this->call\_quantity\_output->Name = L"call\_quantity\_output";

this->call\_quantity\_output->Size = System::Drawing::Size(0, 20);

this->call\_quantity\_output->TabIndex = 16;

//

// label2

//

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label2->Location = System::Drawing::Point(13, 294);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(284, 25);

this->label2->TabIndex = 18;

this->label2->Text = L"Дополнительная программа";

//

// dop\_input

//

this->dop\_input->Location = System::Drawing::Point(338, 323);

this->dop\_input->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);

this->dop\_input->Multiline = true;

this->dop\_input->Name = L"dop\_input";

this->dop\_input->ReadOnly = true;

this->dop\_input->Size = System::Drawing::Size(105, 135);

this->dop\_input->TabIndex = 19;

//

// dop\_output

//

this->dop\_output->Location = System::Drawing::Point(483, 323);

this->dop\_output->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);

this->dop\_output->Multiline = true;

this->dop\_output->Name = L"dop\_output";

this->dop\_output->ReadOnly = true;

this->dop\_output->Size = System::Drawing::Size(105, 135);

this->dop\_output->TabIndex = 20;

//

// dop\_clear\_btn

//

this->dop\_clear\_btn->Location = System::Drawing::Point(24, 378);

this->dop\_clear\_btn->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);

this->dop\_clear\_btn->Name = L"dop\_clear\_btn";

this->dop\_clear\_btn->Size = System::Drawing::Size(159, 38);

this->dop\_clear\_btn->TabIndex = 21;

this->dop\_clear\_btn->Text = L"Очистить поля";

this->dop\_clear\_btn->UseVisualStyleBackColor = true;

this->dop\_clear\_btn->Click += gcnew System::EventHandler(this, &LAB\_1::dop\_clear\_btn\_Click);

//

// dop\_info\_btn

//

this->dop\_info\_btn->Location = System::Drawing::Point(24, 422);

this->dop\_info\_btn->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);

this->dop\_info\_btn->Name = L"dop\_info\_btn";

this->dop\_info\_btn->Size = System::Drawing::Size(159, 36);

this->dop\_info\_btn->TabIndex = 22;

this->dop\_info\_btn->Text = L"Информация";

this->dop\_info\_btn->UseVisualStyleBackColor = true;

this->dop\_info\_btn->Click += gcnew System::EventHandler(this, &LAB\_1::dop\_info\_btn\_Click);

//

// label3

//

this->label3->AutoSize = true;

this->label3->Location = System::Drawing::Point(341, 299);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(49, 20);

this->label3->TabIndex = 23;

this->label3->Text = L"Ввод";

//

// label4

//

this->label4->AutoSize = true;

this->label4->Location = System::Drawing::Point(479, 301);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(60, 20);

this->label4->TabIndex = 24;

this->label4->Text = L"Вывод";

//

// textBox2

//

this->textBox2->Location = System::Drawing::Point(23, 338);

this->textBox2->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);

this->textBox2->Name = L"textBox2";

this->textBox2->Size = System::Drawing::Size(160, 26);

this->textBox2->TabIndex = 25;

this->textBox2->TextChanged += gcnew System::EventHandler(this, &LAB\_1::textBox2\_TextChanged);

//

// button2

//

this->button2->Location = System::Drawing::Point(189, 338);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(114, 36);

this->button2->TabIndex = 26;

this->button2->Text = L"Ввести";

this->button2->UseVisualStyleBackColor = true;

this->button2->Click += gcnew System::EventHandler(this, &LAB\_1::button2\_Click);

//

// dop\_run\_btn

//

this->dop\_run\_btn->Location = System::Drawing::Point(189, 378);

this->dop\_run\_btn->Name = L"dop\_run\_btn";

this->dop\_run\_btn->Size = System::Drawing::Size(114, 80);

this->dop\_run\_btn->TabIndex = 27;

this->dop\_run\_btn->Text = L"Запуск";

this->dop\_run\_btn->UseVisualStyleBackColor = true;

this->dop\_run\_btn->Click += gcnew System::EventHandler(this, &LAB\_1::dop\_run\_btn\_Click);

//

// test\_btn

//

this->test\_btn->BackColor = System::Drawing::SystemColors::ButtonFace;

this->test\_btn->FlatAppearance->BorderColor = System::Drawing::SystemColors::ActiveCaption;

this->test\_btn->FlatStyle = System::Windows::Forms::FlatStyle::System;

this->test\_btn->Location = System::Drawing::Point(10, 488);

this->test\_btn->Name = L"test\_btn";

this->test\_btn->Size = System::Drawing::Size(602, 48);

this->test\_btn->TabIndex = 28;

this->test\_btn->Text = L"Протестировать программу на наличие ошибок";

this->test\_btn->UseVisualStyleBackColor = false;

this->test\_btn->Click += gcnew System::EventHandler(this, &LAB\_1::test\_btn\_Click);

//

// LAB\_1

//

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

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

this->ClientSize = System::Drawing::Size(624, 548);

this->Controls->Add(this->test\_btn);

this->Controls->Add(this->dop\_run\_btn);

this->Controls->Add(this->button2);

this->Controls->Add(this->textBox2);

this->Controls->Add(this->label4);

this->Controls->Add(this->label3);

this->Controls->Add(this->dop\_info\_btn);

this->Controls->Add(this->dop\_clear\_btn);

this->Controls->Add(this->dop\_output);

this->Controls->Add(this->dop\_input);

this->Controls->Add(this->label2);

this->Controls->Add(this->call\_quantity\_output);

this->Controls->Add(this->call\_quantity\_label);

this->Controls->Add(this->label1\_output);

this->Controls->Add(this->output\_label);

this->Controls->Add(this->run\_btn);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->button1);

this->Controls->Add(this->radio\_button\_2);

this->Controls->Add(this->radio\_button\_1);

this->Controls->Add(this->choise\_label);

this->Controls->Add(this->input\_textBox);

this->Controls->Add(this->input\_label);

this->Controls->Add(this->info\_btn);

this->Controls->Add(this->label1);

this->Cursor = System::Windows::Forms::Cursors::Default;

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

this->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);

this->Name = L"LAB\_1";

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

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

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

int recursion\_1(unsigned int n)

{

++q;

if (n == 1)

{

return 1;

}

else if (n % 2 == 0)

{

return (n + 3 \* recursion\_1(n - 1));

}

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

{

return (2 + 2 \* recursion\_1(n - 2));

}

}

int recursion\_2(unsigned int n)

{

++q;

if (n < 3)

{

return 1;

}

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

{

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

}

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

{

unsigned int sum = 0;

for (unsigned int i = 1; i <= n - 1; i++)

sum += recursion\_2(i);

return sum;

}

}

System::Void radioButton1\_activity(System::Object^ sender, System::EventArgs^ e)

{

check\_validaty\_for\_1();

if (this->radio\_button\_1->Checked == true)

{

this->textBox1->Text = "F(n) = 1, при n = 1;" + "\r\n" + "F(n) = n + 3 \* F(n - 1), если n чётно;" + "\r\n" + "F(n) = 2 + 2 \* F(n - 2), если n > 1 и при этом n нечётно." + "\r\n";

choice\_check = true;

}

else

{

this->textBox1->Text = "Здесь будет формула выбранной рекурсивной функции.";

}

}

System::Void radioButton2\_activity(System::Object ^ sender, System::EventArgs ^ e)

{

check\_validaty\_for\_2();

if (this->radio\_button\_2->Checked == true)

{

this->textBox1->Text = "F(n) = 1, при n < 3; " + "\r\n" + "F(n) = F(n - 1) + F(n - 2), если n > 2 и при этом n нечётно; " + "\r\n" + "F(n) = summ F(i) 1 <= i <= n - 1, если n > 2 и при этом n чётно." + "\r\n";

choice\_check = false;

}

else

{

this->textBox1->Text = "Здесь будет формула выбранной рекурсивной функции.";

}

}

void check\_validaty\_for\_1()

{

String^ error\_overflow = "При выборе первого варианта рекурсивной функции ввходные данные не могут быть равными 0.";

try

{

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

{

int data\_i = System::Convert::ToInt32(this->input\_textBox->Text);

if (data\_i == 0)

{

throw error\_overflow;

}

}

}

catch (String^ error\_overflow)

{

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

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

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

{

this->input\_textBox->Text = "";

this->input\_textBox->ForeColor = Color::Black;

}

}

}

void check\_validaty\_for\_2()

{

String^ error\_overflow = "Введено слишком большое число для второго варианта рекурсии. Попробуйте еще раз. Максимальное число для второй рекурсивной функции - 35.";

try

{

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

{

int data\_i = System::Convert::ToInt32(this->input\_textBox->Text);

if (data\_i >= 36)

{

throw error\_overflow;

}

}

}

catch (String^ error\_overflow)

{

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

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

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

{

this->input\_textBox->Text = "";

this->input\_textBox->ForeColor = Color::Black;

}

}

}

void null\_to\_nine()

{

String^ error = "Введено неверное значение. Попробуйте ещё раз.";

String^ data = this->input\_textBox->Text;

try

{

for (int i = 0; i < this->input\_textBox->Text->Length; i++)

{

if (!((input\_textBox->Text[i] >= '0') && (input\_textBox->Text[i] <= '9')) || (input\_textBox->Text[i] == ',') || (input\_textBox->Text[i] == '.'))

{

throw error;

}

}

}

catch (String^ error)

{

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

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

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

{

this->input\_textBox->Text = "";

this->input\_textBox->ForeColor = Color::Black;

}

}

}

void max\_value\_input()

{

String^ error\_overflow = "Введено слишком большое число. Попробуйте еще раз. Максимальное число - 64.";

try

{

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

{

int data\_i = System::Convert::ToInt32(this->input\_textBox->Text);

if (data\_i >= 65)

{

throw error\_overflow;

}

}

}

catch (String^ error\_overflow)

{

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

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

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

{

this->input\_textBox->Text = "";

this->input\_textBox->ForeColor = Color::Black;

}

}

}

void check\_validaty\_1\_2()

{

null\_to\_nine();

max\_value\_input();

if (this->radio\_button\_1->Checked)

{

check\_validaty\_for\_1();

}

if (this->radio\_button\_2->Checked)

{

check\_validaty\_for\_2();

}

}

System::Void run\_btn\_Click(System::Object^ sender, System::EventArgs^ e)

{

q = 0;

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

{

this->n = System::Convert::ToInt32(this->input\_textBox->Text);

if (this->radio\_button\_1->Checked || this->radio\_button\_2->Checked)

{

if (choice\_check == true)

{

this->output\_label->Text = System::Convert::ToString(recursion\_1(n));

this->call\_quantity\_output->Text = System::Convert::ToString(q);

}

else

{

this->output\_label->Text = System::Convert::ToString(recursion\_2(n));

this->call\_quantity\_output->Text = System::Convert::ToString(q);

}

}

}

}

System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e)

{

this->input\_textBox->Text = "";

this->textBox1->Text = "Здесь будет формула выбранной рекурсивной функции.";

this->output\_label->Text = "";

this->call\_quantity\_output->Text = "";

this->radio\_button\_1->Checked = false;

this->radio\_button\_2->Checked = false;

}

System::Void info\_btn\_Click(System::Object^ sender, System::EventArgs^ e)

{

MessageBox::Show("Работа с программой\r\n\r\nДля работы с рекурсивными функциями:\r\n\r\n- Введите данные. Для ввода значений введите данные в поле ввода (максимальное значение - 64 для рекурсии #1, 35 для рекурсии #2);\r\n\r\n- Выберите вариант рекурсивной функции. После выбора рекурсии ниже будет выведена формула выбранной функции;\r\n\r\n- Нажмине кнопку \"Запуск\" для запуска рекурсии.\r\n\r\n- После запуска рекурсии будет выведен результат рекурсивных вычислений и количество вызовов рекурсивной функции.\r\n\r\nДля повторного использования программы нажмите кнопку \"Очистить поля\" и повторие вышеперечисленные действия.", "Информация о работе программы", MessageBoxButtons::OK, MessageBoxIcon::Information);

}

System::Void input\_textBox\_TextChanged(System::Object^ sender, System::EventArgs^ e)

{

check\_validaty\_1\_2();

};

unsigned int max(unsigned int a, unsigned int b)

{

if (a > b)

return a;

else

return b;

}

unsigned int dop()

{

q++;

unsigned int num = 0;

num = arr[i];

i++;

if (num == 0)

{

return 0;

}  
 else

{

unsigned int tmp\_max = dop();

return max(tmp\_max, num);

}

}

System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (i > 255)

{

System::Windows::Forms::DialogResult result = MessageBox::Show("Введено максимальное количество элементов", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

double\_click\_bool = true;

this->textBox2->ForeColor = Color::Red;

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

{

this->textBox2->Text = "";

this->textBox2->ForeColor = Color::Black;

}

this->dop\_output->Text = System::Convert::ToString(i);

}

else

{

if (!(this->textBox2->Text == ""))

{

arr[i] = System::Convert::ToInt32(delete\_first\_null(this->textBox2->Text));

if (this->dop\_input->Text == "")

this->dop\_input->Text = System::Convert::ToString(arr[i]);

else

this->dop\_input->Text += "\r\n" + System::Convert::ToString(arr[i]);

this->textBox2->Text = "";

i++;

}

else

{

MessageBox::Show("Введите элемент последовательности.", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

}

}

System::Void dop\_clear\_btn\_Click(System::Object^ sender, System::EventArgs^ e)

{

this->dop\_output->Text = "";

this->dop\_input->Text = "";

this->textBox2->Text = "";

this->double\_click\_bool = false;

this->q = 0;

this->arr->Clear(arr, 0, arr->Length);

this->call\_quantity\_output->Text = "";

}

System::Void dop\_run\_btn\_Click(System::Object^ sender, System::EventArgs^ e)

{

q = 0;

if (this->double\_click\_bool == true)

{

MessageBox::Show("Ответ уже вычислен и выведен в поле \"Вывод\".", "Информация", MessageBoxButtons::OK, MessageBoxIcon::Information);

}

else

{

if (!(this->dop\_input->Text == "") && (this->dop\_input->Text->EndsWith("0")))

{

i = 0;

this->dop\_output->Text = System::Convert::ToString(dop());

i = 0;

this->double\_click\_bool = true;

this->call\_quantity\_output->Text = System::Convert::ToString(q);

q = 0;

}

else if (!(this->dop\_input->Text == "") && !(this->dop\_input->Text->EndsWith("0")))

{

MessageBox::Show("Введите ноль для обозначения конца последовательности.", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

else if (this->dop\_input->Text == "")

{

MessageBox::Show("Введите последовательность.", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

}

}

void dop\_zero\_first()

{

String^ error\_zero\_first = "Первое элемент в последовательности не может быть нулём.";

try

{

if (this->dop\_input->Text == "" && this->textBox2->Text == "0")

{

throw error\_zero\_first;

}

}

catch (String^ erroer\_zero\_first)

{

this->textBox2->ForeColor = Color::Red;

System::Windows::Forms::DialogResult result = MessageBox::Show(error\_zero\_first, "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

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

{

this->textBox2->Text = "";

this->textBox2->ForeColor = Color::Black;

}

}

}

void dop\_valid\_input()

{

String^ error\_not\_a\_number = "Введено неверное значение. Попробуйте ещё раз.";

String^ data = this->textBox2->Text;

try

{

for (int i = 0; i < this->textBox2->Text->Length; i++)

{

if (!((textBox2->Text[i] >= '0') && (textBox2->Text[i] <= '9')) || (textBox2->Text[i] == ',') || (textBox2->Text[i] == '.'))

{

throw error\_not\_a\_number;

}

}

}

catch (String^ error\_not\_a\_number)

{

this->textBox2->ForeColor = Color::Red;

System::Windows::Forms::DialogResult result = MessageBox::Show(error\_not\_a\_number, "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

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

{

this->textBox2->Text = "";

this->textBox2->ForeColor = Color::Black;

}

}

}

void dop\_max\_value\_input()

{

unsigned int n\_ = 200000000;

String^ error\_max = "Введено слишком большое число. Максимальное значение - 200.000.000.";

if (this->textBox2->Text != "")

{

try

{

if (System::Convert::ToInt64(this->textBox2->Text) > n\_)

{

throw error\_max;

}

}

catch (String^ error\_max)

{

this->textBox2->ForeColor = Color::Red;

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

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

{

this->textBox2->Text = "";

this->textBox2->ForeColor = Color::Black;

}

}

}

}

void double\_input\_protect()

{

String^ double\_input\_protect = "Ответ уже вычислен. Нажмине кнопку \"Очистить поля\", чтобы ввести новую последовательность";

try

{

if (!(this->dop\_output->Text == "") && !(this->textBox2->Text == ""))

throw double\_input\_protect;

}

catch (String^ double\_input\_protect)

{

this->textBox2->ForeColor = Color::Red;

System::Windows::Forms::DialogResult result = MessageBox::Show(double\_input\_protect, "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

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

{

this->textBox2->Text = "";

this->textBox2->ForeColor = Color::Black;

}

}

}

String^ delete\_first\_null(String^ data)

{

int count = 0;

bool check = true;

if (data->StartsWith("0"))

{

for (int i = 0; i < data->Length; i++)

{

if (data[i] == '0' && check)

{

count++;

}

else

{

check = false;

}

}

data->Remove(0, count);

}

return data;

}

System::Void textBox2\_TextChanged(System::Object^ sender, System::EventArgs^ e)

{

double\_input\_protect();

dop\_valid\_input();

dop\_zero\_first();

dop\_max\_value\_input();

}

System::Void dop\_info\_btn\_Click(System::Object^ sender, System::EventArgs^ e)

{

MessageBox::Show("Данная программа определяет максимальный элемент последовательности и выводит его в поле \"Вывод\", последовательность должна оканчивается нулём и состоять минимум из двух элементов, включая последний.\r\n\r\nДля работы с программой:\r\n\r\n- Введите последовательность в поле ввода по одному элементу;\r\n\r\n- Для добавления элемента в программу нажмите кнопку \"Ввести\", после чего элемент будет добавлен в поле \"Ввод\";\r\n\r\n- Для запуска программы нажмите кнопку \"Запуск\".\r\n\r\nДля повторного использования программы нажмите кнопку \"Очистить поля\" и повторие вышеперечисленные действия.", "Информация о работе программы", MessageBoxButtons::OK, MessageBoxIcon::Information);

}

int test\_1()

{

if (recursion\_1(1) == 1)

return 0;

else

return 1;

}

int test\_2()

{

if (recursion\_1(2) == 5)

return 0;

else

return 1;

}

int test\_3()

{

if (recursion\_2(2) == 1)

return 0;

else

return 1;

}

int test\_4()

{

if (recursion\_2(3) == 2)

return 0;

else

return 1;

}

int test\_5()

{

if (max(1, 0) == 1)

return 0;

else

return 1;

}

int test\_6()

{

for (i = 0; i < 5; i++)

{

if (i == 4)

arr[i] = 0;

else

arr[i] = i + 1;

}

i = 0;

if (dop() == 4)

return 0;

else

return 1;

this->arr->Clear(arr, 0, arr->Length);

}

void test\_cases()

{

int test\_flag = (test\_1() + test\_2() + test\_3() + test\_4() + test\_5() + test\_6());

if (test\_flag == 0)

{

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

}

else

{

System::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();

}

}

i = 0;

}

void test\_cases\_constructor()

{

int test\_flag = (test\_1() + test\_2() + test\_3() + test\_4() + test\_5() + test\_6());

if (test\_flag != 0)

{

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

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

{

this->Close();

}

}

i = 0;

}

System::Void test\_btn\_Click(System::Object^ sender, System::EventArgs^ e)

{

test\_cases();

}

}

Приложение И

Листинг программы, файл MyForm.cpp

#include "LAB\_1.h"

#include <Windows.h>

using namespace System;

using namespace System::Windows::Forms;

[STAThreadAttribute]

int main()

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

LAB\_1::LAB\_1 form;

Application::Run(% form);

}

Приложение К

Листинг программы, файл UnitTest.cpp

#include "pch.h"

#include "CppUnitTest.h"

#include "../LAB\_1/Functions.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest

{

TEST\_CLASS(UnitTest)

{

public:

// Тест первой рекурсивной функции, при аргументе = 1, ожидаемый результат = 1

TEST\_METHOD(rec1\_test1)

{

Assert::IsTrue(recursion\_1(1) == 1);

}

// Тест первой рекурсивной функции, при аргументе = 2, ожидаемый результат = 5

TEST\_METHOD(rec1\_test2)

{

Assert::IsTrue(recursion\_1(2) == 5);

}

// Тест второй рекурсивной функции, при аргументе = 2, ожидаемый результат = 1

TEST\_METHOD(rec2\_test1)

{

Assert::IsTrue(recursion\_2(2) == 1);

}

// Тест второй рекурсивной функции, при аргументе = 3, ожидаемый результат = 2

TEST\_METHOD(rec2\_test2)

{

Assert::IsTrue(recursion\_2(3) == 2);

}

// Тест функции по определению максимального элемента из двух, при аргументах = 1 и 0, ожидаемый результат = 1

TEST\_METHOD(max\_test)

{

Assert::IsTrue(max(1, 0) == 1);

}

// Тест функции по определению максимального элемента последовательности, при аргументах = {1, 2, 3, 4, 0}, ожидаемый результат = 4

TEST\_METHOD(dop\_test)

{

Assert::IsTrue(dop() == 4);

}

};

}

Приложение Л

Листинг программы, файл Functions.h

// Формула первой рекурсивной функции

int recursion\_1(unsigned int n)

{

if (n == 1)

{

return 1;

}

else if (n % 2 == 0)

{

return (n + 3 \* recursion\_1(n - 1));

}

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

{

return (2 + 2 \* recursion\_1(n - 2));

}

}

// Формула второй рекурсивной функции

int recursion\_2(unsigned int n)

{

if (n < 3)

{

return 1;

}

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

{

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

}

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

{

int sum = 0;

for (int i = 1; i <= n - 1; i++)

sum += recursion\_2(i);

return sum;

}

}

// Определение максимального элемента из двух, для дополнительной программы

int max(int a, int b)

{

if (a > b)

return a;

else

return b;

}

int arr[5] = {1, 2, 3, 4, 0}, i = 0;

unsigned int dop()

{

unsigned int num = 0;

num = arr[i];

i++;

if (num == 0)

{

return 0;

}

unsigned int tmp\_max = dop();

return max(tmp\_max, num);

}