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 №6. Шаблоны

Отчет по лабораторной работе №4 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** – Необходимо реализовать программу, в которой будет реализован класс стека, который может хранить элементы любого типа данных и выполнение любых математических операций над ними
  2. **Наименование задачи** – Лабораторная работа №6. Шаблоны
  3. **Словесное описание**:

Реализуйте графический интерфейс для вашей программы, где будет отдельное поле для внесения любого типа данных и отдельное окно для просмотра сохраненных данных. Также в данном интерфейсе должна присутствовать возможность удаления элементов из стека.   
Рассмотрена возможность выполнения различных математических операций над внесенными данными (т.е сложение, вычитание, умножение и деление). Результат проделанной работы должно быть выведено в отдельном окне.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Объект программы | Имя этого объекта в программе | Тип данных | Диапазоны представления | Простая/  структура | Вход/  выход/  Константа |
| Структура данных стек | Int\_stack | int | 0 - 200000000 | Структура | Вход / выход |
| Структура данных стек | Char\_stack | char | 33 - 128 | Структура | Вход / выход |
| Структура данных стек | Double\_stack | double | 0 - 200000000 | Структура | Вход / выход |

**Внешние спецификации данных (от лица пользователя)**

* 1. **Внешние спецификации функций программы**

В программе должна быть реализована работа со структурами данных:

– Выбор структуры данных

– Добавление элемента в структуру

– Удаление элемента структуры

– Замена элемента по индексу

– Вывод структуры данных на экран

– Конвертация структуры данных в структуру символов

– Поиск наибольшего элемента в структуре

– Поиск индекса элемента по значению

– Сложение элементов стека

– Вычитание элементов стека

– Произведение элементов стека

– Частное элементов стека

* 1. **Математическая формулировка задачи**

Введены ограничения на ввод значений в структуру при включенном режиме конвертации в символы (от 33 до 128), ограничение обосновано рациональностью вывода значимых символов, а не всего интервала от 0 до 128, так как некоторые символы (от 0 до 32) являются служебными и отображаются некорректно, а значения больше 128 не предусмотрены в таблице кодировки символов ASCII.

* 1. **Спецификации интерфейса**

Модульное тестирование реализовано при помощи фреймворка CppUnitTest и встроенное тестирование, при помощи процедур и функций, проводимое как при запуске программы, так и при нажатии отдельной кнопки.

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.

* 1. **Внешние данные тестирования**

Выполнено 5 тест-кейсов и 3 Unit-теста, встроенных в проект

Таблица 1 - Тест-кейс №1. Добавление нового элемента

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | | | T01 |
| Заголовок | | | Проверка добавления нового элемента |
| Предусловие | | | Приложение запущенно, выбран тип данных |
| Ожидаемый результат | | | Вывод значения и его индекса |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Выбрать раздел «Операции» | Откроется меню раздела | Pass |
| 2 | Выбрать пункт «Добавление элемента» | Открытие окна ввода значения | Pass |
| 3 | Нажатие кнопки «Добавить» | Закрытие окно ввода значения, вывод значения и его индекса | Pass |
| Результат | | | Pass |

Таблица 2 - Тест-кейс №2. Заполнение тестовыми данными

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | | | T02 |
| Заголовок | | | Заполнение тестовыми данными |
| Предусловие | | | Приложение запущенно, тип данных выбран |
| Ожидаемый результат | | | Заполнение структуры данных тестовыми значениями |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Выбрать раздел «Редактирование» | Откроется меню раздела | Pass |
| 2 | Выбрать пункт «Заполнение структуры тестовыми данными» | Открытие окна предупреждения | Pass |
| 3 | Нажать кнопку «ОК» | Закрытие окна предупреждения, заполнение структуры тестовыми данными | Pass |
| Результат | | | Pass |

Таблица 3 - Тест-кейс №3. Работоспособность функции поиска максимально элемента

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | | | T03 |
| Заголовок | | | Проверка работоспособности функции поиска максимального элемента |
| Предусловие | | | Приложение запущено, выбран тип данных int |
| Ожидаемый результат | | | Вывод наибольшего значения |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Ввести два значения, например, 2 и 3 | В окно вывода будут выведены значения с их индексов | Pass |
| 2 | Выбрать раздел «Редактирование» | Открытие меню раздела «Операции» | Pass |
| 3 | Выбрать пункт «Поиск наибольшего элемента» | Открытие окна, где выведено наибольшее значение в структуре, в данном случае значение - 3 | Pass |
| Результат | | | Pass |

Таблица 4 - Тест-кейс №4. Работоспособность сложения элементов стека

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | | | T04 |
| Заголовок | | | Проверка работоспособности функции сложения элементов стека |
| Предусловие | | | Приложение запущенно, тип данных выбран |
| Ожидаемый результат | | | Структура очищена |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Выбрать раздел «Редактирование» | Откроется меню раздела | Pass |
| 2 | Выбрать пункт «Заполнение структуры тестовыми данными» | Открытие окна предупреждения | Pass |
| 3 | Нажать кнопку «ОК» | Закрытие окна предупреждения, заполнение структуры тестовыми данными | Pass |
| 4 | Выбрать раздел «Редактирование» | Откроется меню раздела | Pass |
| 5 | Выбрать пункт «Очистить структуру» | Структура очищена, выведено сообщение о пустоте структуры | Pass |
| Результат | | | Pass |

Таблица 5 - Тест-кейс №5. Удаление последнего элемента

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | | | T05 |
| Заголовок | | | Проверка работоспособности функции «Удаление последнего элемента» |
| Предусловие | | | Приложение запущено, тип данных выбран |
| Ожидаемый результат | | | Удаление последнего элемента |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Ввести два значения, например, 2 и 3 | В окно вывода будут выведены значения с их индексов | Pass |
| 2 | Выбрать раздел «Редактирование» | Открытие меню раздела «Операции» | Pass |
| 3 | Выбрать пункт «Удаление последнего элемента» | Последний элемент удалён | Pass |
| Результат | | | Pass |

* 1. **Пример работающей программы**

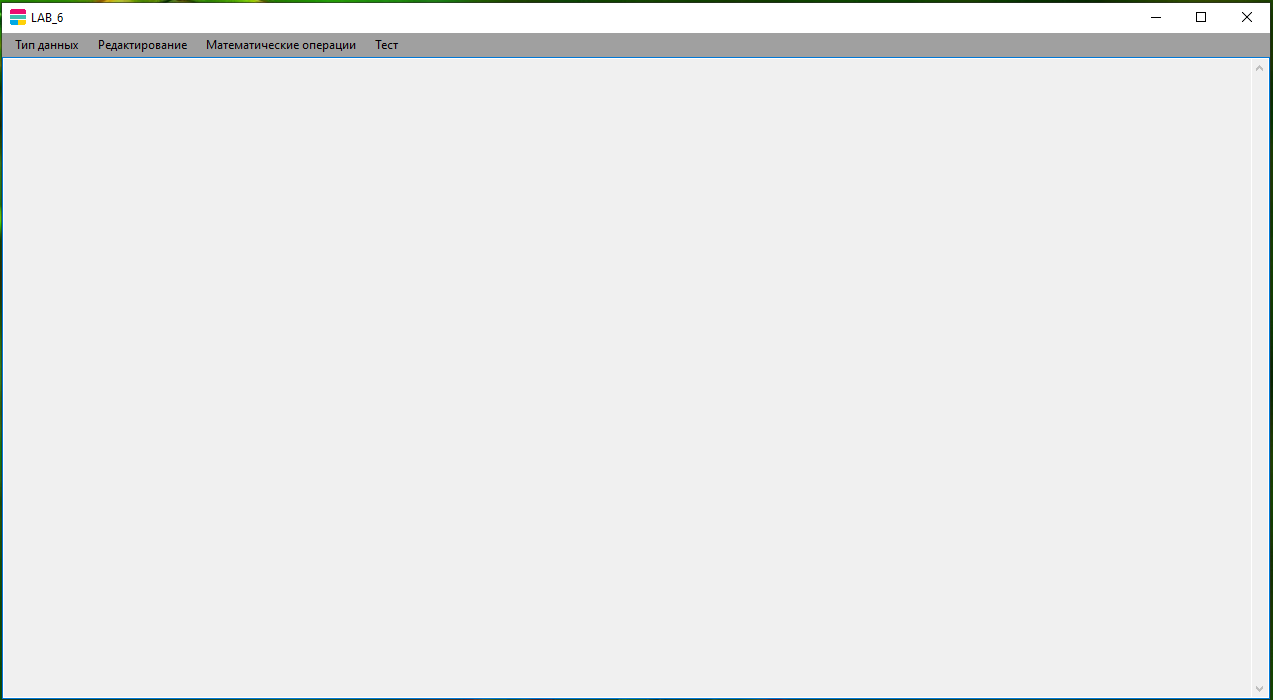


Рисунок 1 - Окно приложения

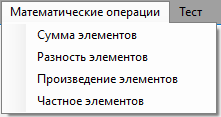


Рисунок 2 - Меню раздела "Математические операции"

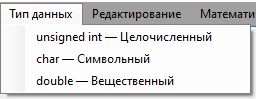


Рисунок 3 - Меню раздела "Тип данных"

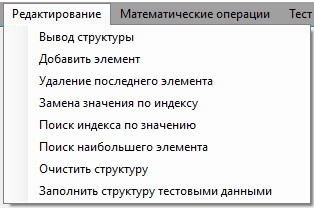


Рисунок 4 - Меню раздела "Редактирование"

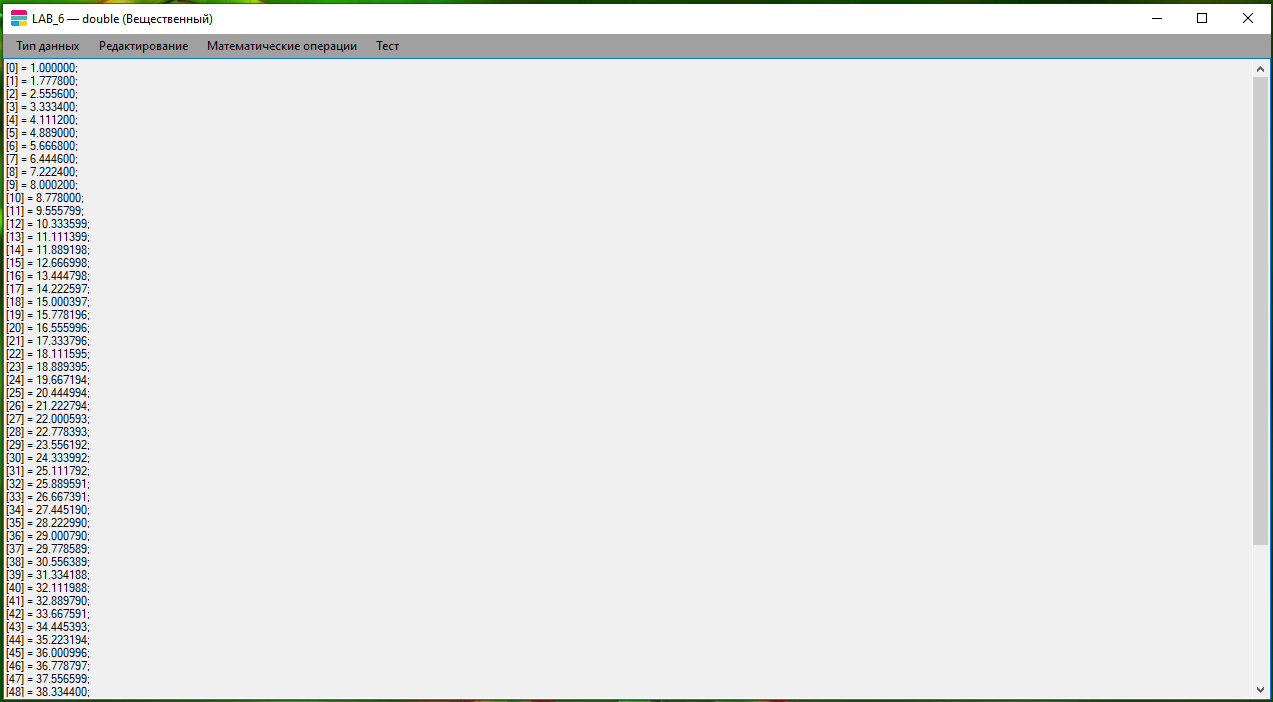


Рисунок 5 - Пример использования приложения

1. **Проектирование задачи**

Реализовать приложение с графическим интерфейсом, использую библиотеку Windows Forms, для работы с шаблонным классом структуры данных Стек. 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\_6.exe
  2. **Уточненное словесное описание задачи**

Реализуйте графический интерфейс для вашей программы, где будет отдельное поле для внесения любого типа данных и отдельное окно для просмотра сохраненных данных. Также в данном интерфейсе должна присутствовать возможность удаления элементов из стека.   
Рассмотрена возможность выполнения различных математических операций над внесенными данными (т.е сложение, вычитание, умножение и деление). Результат проделанной работы должно быть выведено в отдельном окне.

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

Для начала работы надо выбрать структуры из списка «Тип данных», при выборе типа данных, дальнейшие действия, до переключения ТД, будут выполняться только со структурой данных текущего ТД.

Для манипуляций над структурой можно выбрать операции из раздела «Редактирование».

Так же в приложении реализованы математические функции над структурами данных: сумма, разность, произведение, частное элементов структуры.

В приложении реализовано Unit-тестирование, для проверки работоспособности ключевых функций приложения необходимо выбрать пункт «Тест», программа проверит себя сама, при корректном прохождении тестов будет выведено уведомление о корректности работы программы, в противном случае приложение будет закрыто. Unit-тестирование выполняется при открытии приложения автоматически.

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

Таблица - Глобальные переменные

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Объект программы | Имя этого объекта в программе | Характеристики | | | Как используется в программе |
| Тип данных | Диапазоны представления | Прост/структура/ константа |
| LAB\_6.h | current\_typ | String^ | - | Структура | Хранит в себе название текущей структуры данных |
| LAB\_6.h | Int\_stack | Int | - | Структура | Экземпляр класса Stack, хранение данных |
| LAB\_6.h | Char\_ctack | Char | - | Структура | Экземпляр класса Stack, хранение данных |
| LAB\_6.h | Double\_stack | bouble | - | Структура | Экземпляр класса Stack, хранение данных |
| templateStack.h | sHead | Struct Stack\* | — | Структура | Указатель на структуру Stack |

* 1. **Декомпозиция функций**

Таблица - Декомпозиция функций

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назначение | Имя | Параметры | | Процедура/ функция | Тестируется/  не тестируется |
| Вход:  тип | Выход: тип |
| Функция присвоения числового значения строковому значению текущей структуры данных | Struct\_check | void | int | Функция | Не тестируется |
| Процедура, запускающая действия при нажатии "Вывод структуры" | выводСтруктурыToolStripMenuItem\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Не тестируется |
| Процедура, запускающая действия при нажатии "Добавить элемент" | добавитьЭлементToolStripMenuItem\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Не тестируется |
| Процедура, запускающая действия при нажатии "Удаление последнего элемента" | удалениеПоследнегоЭлементаToolStripMenuItem\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Не тестируется |
| Процедура, запускающая действия при нажатии "Замена значения по индексу" | заменаЗначенияПоИндексуToolStripMenuItem\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Не тестируется |
| Процедура, запускающая действия при нажатии "Поиск индекса по значению" | поискИндексаПоЗначениюToolStripMenuItem\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Не тестируется |
| Процедура, запускающая действия при нажатии "Поиск наибольшего элемента" | поискНаибольшегоЭлементаToolStripMenuItem\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Не тестируется |
| Процедура, запускающая действия при нажатии "Очистить структуру" | очиститьСтруктураToolStripMenuItem\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Не тестируется |
| Процедура, запускающая действия при нажатии "Тестовые данных" | тестовыеДанныеToolStripMenuItem\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Не тестируется |
| Процедура, запускающая действия при нажатии "Array" | arrayToolStripMenuItem\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Не тестируется |
| Процедура, запускающая действия при нажатии "Vector" | vectorToolStripMenuItem\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Не тестируется |
| Процедура, запускающая действия при нажатии "Stack" | stackToolStripMenuItem\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Не тестируется |
| Процедура, запускающая действия при нажатии "Queue" | queueToolStripMenuItem\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Не тестируется |
| Процедура конвертацию числовой структуры в символьный формат | to\_char | void | void | Процедура | Не тестируется |
| Процедура, запускающая действия при нажатии "Конвертация в символы" | конвертацияВASCIIToolStripMenuItem\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Не тестируется |
| Процедура, запускающая действия при нажатии кнопки "Добавить" | button1\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Не тестируется |
| Процедура, запускающая действия при нажатии кнопки "Отмена" | button2\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Не тестируется |
| Процедура, запускающая действия при изменении текста в textBox1 | textBox1\_TextChanged | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Не тестируется |
| Функция проверки корректности ввода | check | void | bool | Функция | Не тестируется |
| Функция получения значения из поля ввода | get\_value | Void | System::String^ | Функция | Не тестируется |
| Функция вывода структуры данных в целочисленном формате | output | void | std::string | Функция | Не тестируется |
| Функция вывода структуры данных в символьном формате | output\_char | void | std::string | Функция | Не тестируется |
| Процедура добавления нового элемента в массив | push | T val | void | Процедура | Не тестируется |
| Процедура удаления последнего элемента | pop | void | void | Процедура | Не тестируется |
| Процедура изменения значения элемента по его индексу | change\_by\_index | int index, T new\_val | void | Функция | Не тестируется |
| Функция поиска индекса элемента по его значению | Find\_index | T val | int | Функция | Не тестируется |
| Функция поиска наибольшего элемента в структуре | find\_max | void | T | Функция | Не тестируется |
| Процедура очистки структуры данных | clear | void | void | Процедура | Не тестируется |
| Процедура заполнения структуры данных тестовыми значениями | fill\_test | void | void | Процедура | Не тестируется |
| Процедура конвертации целочисленной структуры данных в символьный формат | copy\_to\_char | Имя класса структуры данных a | void | Процедура | Не тестируется |

* 1. **Блок-схема**

Блок-схема функции struct\_id() в Приложении A.1.

Блок-схема процедуры check\_index() в Приложении А.2.

Блок-схема функции check() в Приложении А.3.

Блок-схема процедуры myStack::pop() в Приложении А.4.

Блок-схема функции myStack::output\_char() в Приложении А.5.

* 1. **Алгоритмизация**

Листинг файлов в Приложении Б:

– LAB\_6.h,

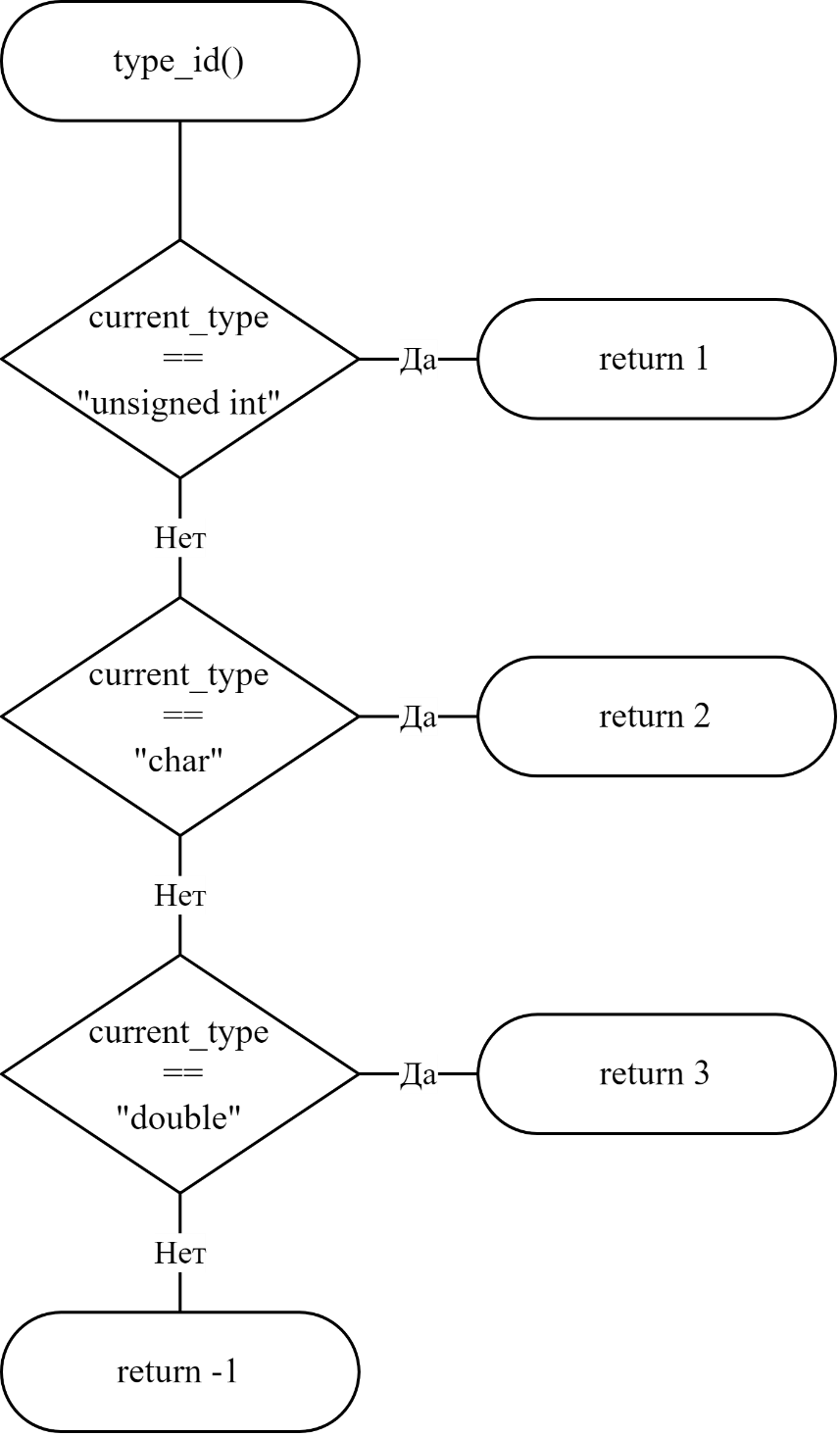
– LAB\_6.cpp,

– AddElement.h,

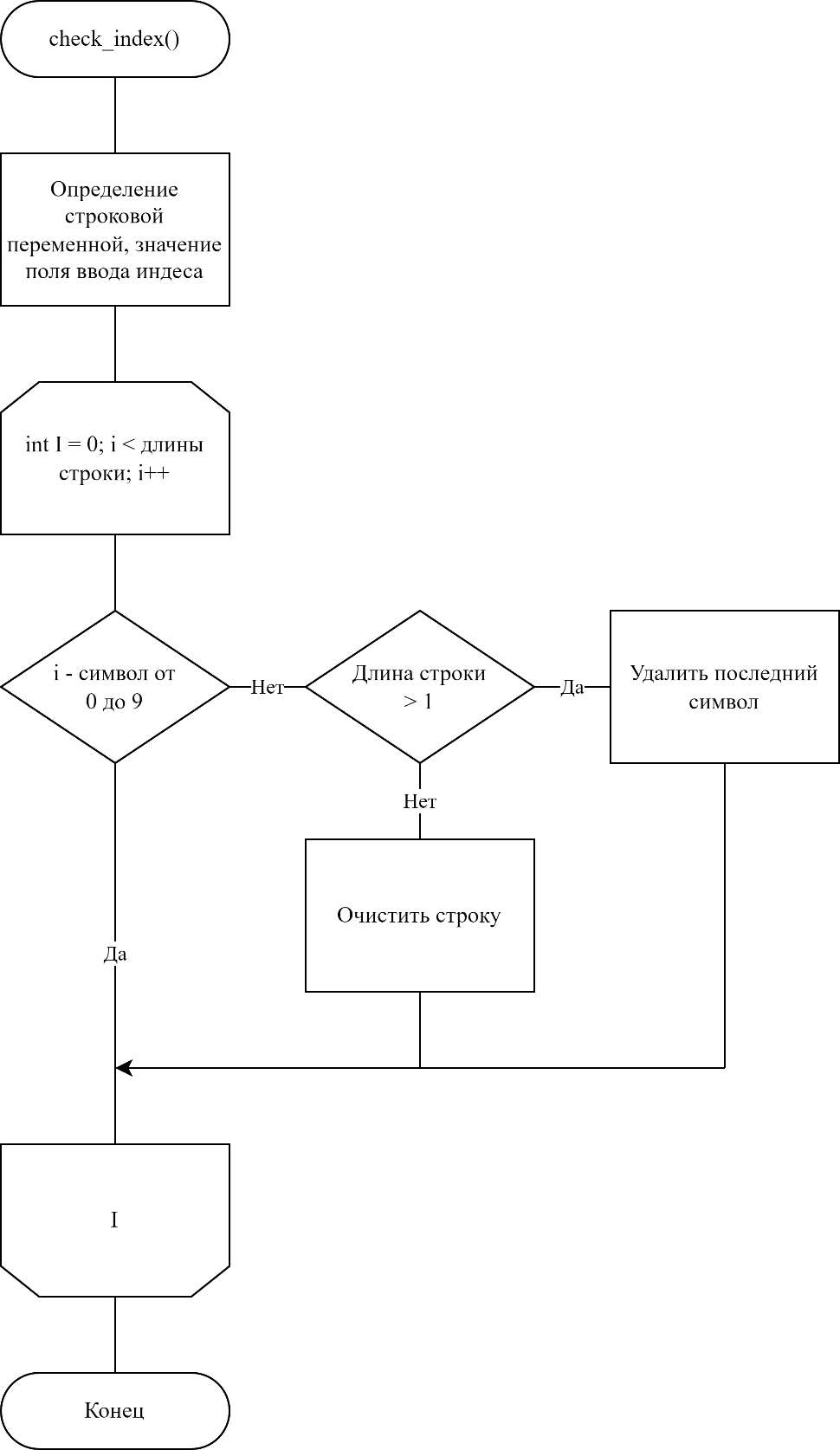
– ChangeByIndex.h,

– templateStack.h,  
– UnitTest.cpp.

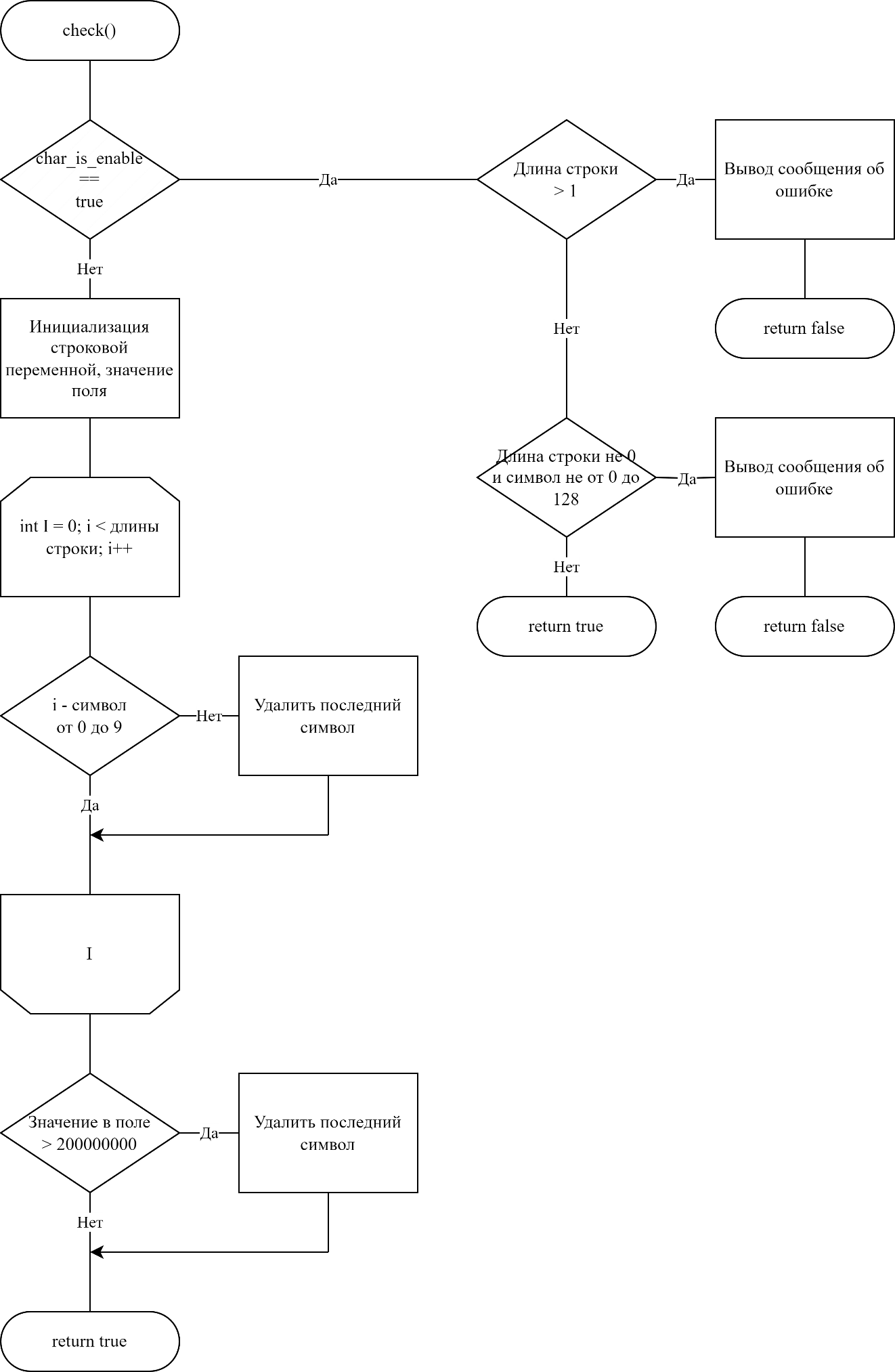
Приложение А



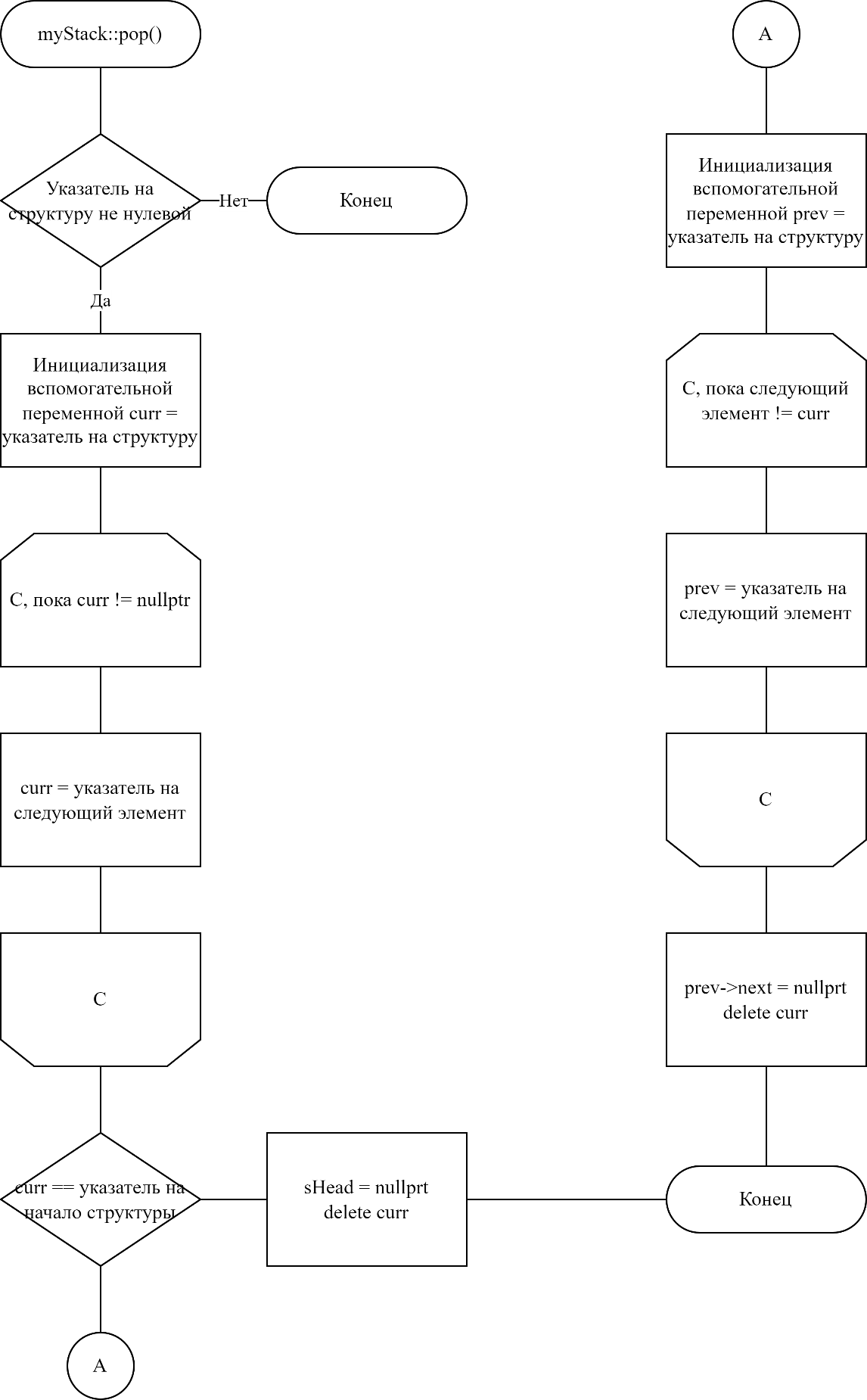
Приложение А.1 Рисунок 6 – Блок-схема функции struct\_id()



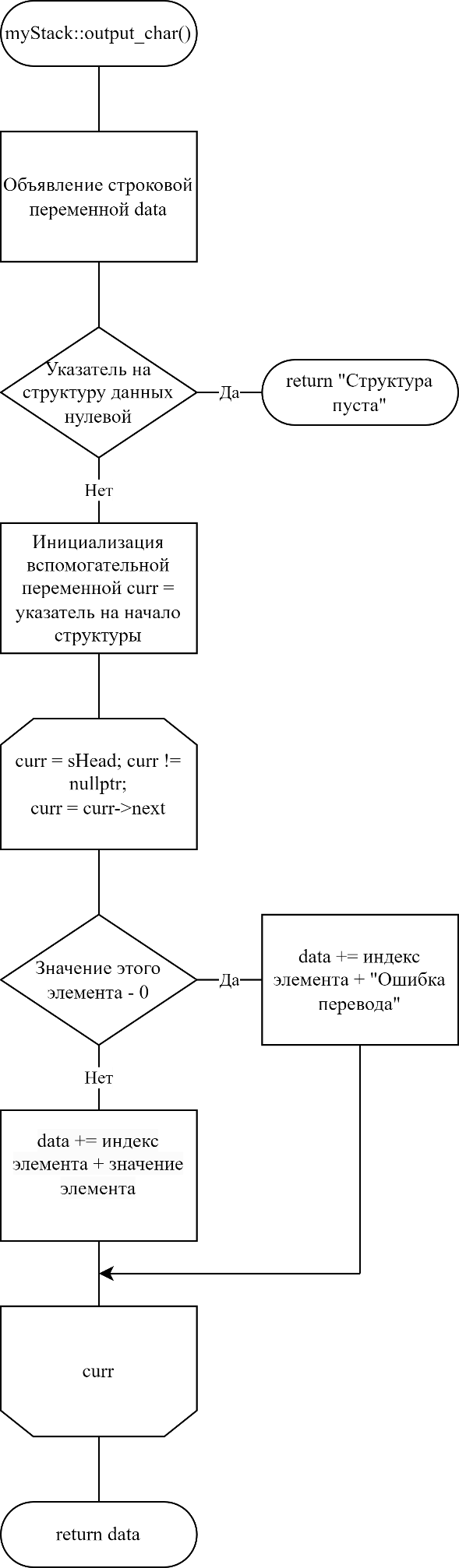
Приложение А.2 Рисунок 7 – Блок-схема процедуры check\_index()



Приложение А.3 Рисунок 8 – Блок-схема функции check()



Приложение А.4 Рисунок 9 – Блок-схема процедуры templateStack::pop()



Приложение А.5 Рисунок 10 – Блок-схема процедуры templateStack::output\_char()

Приложение Б

Листинг файла LAB\_6.h

#pragma once

#include <vector>

#include <string>

#include "templateStack.h"

#include "type.h"

#include "ChangeByIndex.h"

#include "AddElement.h"

namespace LAB6 {

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 Microsoft::VisualBasic;

templateStack<int> int\_stack(int\_type);

templateStack<char> char\_stack(char\_type);

templateStack<double> double\_stack(double\_type);

/// <summary>

/// Сводка для LAB\_6

/// </summary>

public ref class LAB\_6 : public System::Windows::Forms::Form

{

public:

LAB\_6(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

test\_cases();

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~LAB\_6()

{

int\_stack.clear();

char\_stack.clear();

double\_stack.clear();

if (components)

{

delete components;

}

}

protected:

private: String^ current\_type = ""; // Хранит в себе название текущей структуры данных

private: System::Windows::Forms::MenuStrip^ menuStrip1;

private: System::Windows::Forms::ToolStripMenuItem^ структураToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ arrayToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ vectorToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ stackToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ операцииToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ добавитьЭлементToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ удалениеПоследнегоЭлементаToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ заменаЗначенияПоИндексуToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ поискИндексаПоЗначениюToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ поискНаибольшегоЭлементаToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ очиститьСтруктураToolStripMenuItem;

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

private: System::Windows::Forms::ToolStripMenuItem^ выводСтруктурыToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ тестовыеДанныеToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ тестToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ математическиеОперацииToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ суммаЭлементовToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ разностьЭлементовToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ произведениеЭлементовToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ частноеЭлементовToolStripMenuItem;

/// <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)

{

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(LAB\_6::typeid));

this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());

this->структураToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->arrayToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->vectorToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->stackToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->операцииToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->выводСтруктурыToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->добавитьЭлементToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->удалениеПоследнегоЭлементаToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->заменаЗначенияПоИндексуToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->поискИндексаПоЗначениюToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->поискНаибольшегоЭлементаToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->очиститьСтруктураToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->тестовыеДанныеToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->математическиеОперацииToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->суммаЭлементовToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->разностьЭлементовToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->произведениеЭлементовToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->частноеЭлементовToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->тестToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

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

this->menuStrip1->SuspendLayout();

this->SuspendLayout();

//

// menuStrip1

//

this->menuStrip1->BackColor = System::Drawing::SystemColors::ControlDark;

this->menuStrip1->ImageScalingSize = System::Drawing::Size(24, 24);

this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(4) {

this->структураToolStripMenuItem,

this->операцииToolStripMenuItem, this->математическиеОперацииToolStripMenuItem, this->тестToolStripMenuItem

});

this->menuStrip1->Location = System::Drawing::Point(0, 0);

this->menuStrip1->Name = L"menuStrip1";

this->menuStrip1->Padding = System::Windows::Forms::Padding(4, 1, 0, 1);

this->menuStrip1->Size = System::Drawing::Size(1268, 24);

this->menuStrip1->TabIndex = 0;

this->menuStrip1->Text = L"menuStrip1";

//

// структураToolStripMenuItem

//

this->структураToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(3) {

this->arrayToolStripMenuItem,

this->vectorToolStripMenuItem, this->stackToolStripMenuItem

});

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

this->структураToolStripMenuItem->Size = System::Drawing::Size(83, 22);

this->структураToolStripMenuItem->Text = L"Тип данных";

//

// arrayToolStripMenuItem

//

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

this->arrayToolStripMenuItem->Size = System::Drawing::Size(250, 22);

this->arrayToolStripMenuItem->Text = L"unsigned int — Целочисленный";

this->arrayToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_6::arrayToolStripMenuItem\_Click);

//

// vectorToolStripMenuItem

//

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

this->vectorToolStripMenuItem->Size = System::Drawing::Size(250, 22);

this->vectorToolStripMenuItem->Text = L"char — Символьный";

this->vectorToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_6::vectorToolStripMenuItem\_Click);

//

// stackToolStripMenuItem

//

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

this->stackToolStripMenuItem->Size = System::Drawing::Size(250, 22);

this->stackToolStripMenuItem->Text = L"double — Вещественный";

this->stackToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_6::stackToolStripMenuItem\_Click);

//

// операцииToolStripMenuItem

//

this->операцииToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(8) {

this->выводСтруктурыToolStripMenuItem,

this->добавитьЭлементToolStripMenuItem, this->удалениеПоследнегоЭлементаToolStripMenuItem, this->заменаЗначенияПоИндексуToolStripMenuItem,

this->поискИндексаПоЗначениюToolStripMenuItem, this->поискНаибольшегоЭлементаToolStripMenuItem, this->очиститьСтруктураToolStripMenuItem,

this->тестовыеДанныеToolStripMenuItem

});

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

this->операцииToolStripMenuItem->Size = System::Drawing::Size(108, 22);

this->операцииToolStripMenuItem->Text = L"Редактирование";

//

// выводСтруктурыToolStripMenuItem

//

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

this->выводСтруктурыToolStripMenuItem->Size = System::Drawing::Size(307, 22);

this->выводСтруктурыToolStripMenuItem->Text = L"Вывод структуры";

this->выводСтруктурыToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_6::выводСтруктурыToolStripMenuItem\_Click);

//

// добавитьЭлементToolStripMenuItem

//

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

this->добавитьЭлементToolStripMenuItem->Size = System::Drawing::Size(307, 22);

this->добавитьЭлементToolStripMenuItem->Text = L"Добавить элемент";

this->добавитьЭлементToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_6::добавитьЭлементToolStripMenuItem\_Click);

//

// удалениеПоследнегоЭлементаToolStripMenuItem

//

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

this->удалениеПоследнегоЭлементаToolStripMenuItem->Size = System::Drawing::Size(307, 22);

this->удалениеПоследнегоЭлементаToolStripMenuItem->Text = L"Удаление последнего элемента";

this->удалениеПоследнегоЭлементаToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_6::удалениеПоследнегоЭлементаToolStripMenuItem\_Click);

//

// заменаЗначенияПоИндексуToolStripMenuItem

//

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

this->заменаЗначенияПоИндексуToolStripMenuItem->Size = System::Drawing::Size(307, 22);

this->заменаЗначенияПоИндексуToolStripMenuItem->Text = L"Замена значения по индексу";

this->заменаЗначенияПоИндексуToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_6::заменаЗначенияПоИндексуToolStripMenuItem\_Click);

//

// поискИндексаПоЗначениюToolStripMenuItem

//

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

this->поискИндексаПоЗначениюToolStripMenuItem->Size = System::Drawing::Size(307, 22);

this->поискИндексаПоЗначениюToolStripMenuItem->Text = L"Поиск индекса по значению";

this->поискИндексаПоЗначениюToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_6::поискИндексаПоЗначениюToolStripMenuItem\_Click);

//

// поискНаибольшегоЭлементаToolStripMenuItem

//

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

this->поискНаибольшегоЭлементаToolStripMenuItem->Size = System::Drawing::Size(307, 22);

this->поискНаибольшегоЭлементаToolStripMenuItem->Text = L"Поиск наибольшего элемента";

this->поискНаибольшегоЭлементаToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_6::поискНаибольшегоЭлементаToolStripMenuItem\_Click);

//

// очиститьСтруктураToolStripMenuItem

//

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

this->очиститьСтруктураToolStripMenuItem->Size = System::Drawing::Size(307, 22);

this->очиститьСтруктураToolStripMenuItem->Text = L"Очистить структуру";

this->очиститьСтруктураToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_6::очиститьСтруктураToolStripMenuItem\_Click);

//

// тестовыеДанныеToolStripMenuItem

//

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

this->тестовыеДанныеToolStripMenuItem->Size = System::Drawing::Size(307, 22);

this->тестовыеДанныеToolStripMenuItem->Text = L"Заполнить структуру тестовыми данными";

this->тестовыеДанныеToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_6::тестовыеДанныеToolStripMenuItem\_Click);

//

// математическиеОперацииToolStripMenuItem

//

this->математическиеОперацииToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(4) {

this->суммаЭлементовToolStripMenuItem,

this->разностьЭлементовToolStripMenuItem, this->произведениеЭлементовToolStripMenuItem, this->частноеЭлементовToolStripMenuItem

});

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

this->математическиеОперацииToolStripMenuItem->Size = System::Drawing::Size(169, 22);

this->математическиеОперацииToolStripMenuItem->Text = L"Математические операции";

//

// суммаЭлементовToolStripMenuItem

//

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

this->суммаЭлементовToolStripMenuItem->Size = System::Drawing::Size(215, 22);

this->суммаЭлементовToolStripMenuItem->Text = L"Сумма элементов";

this->суммаЭлементовToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_6::суммаЭлементовToolStripMenuItem\_Click);

//

// разностьЭлементовToolStripMenuItem

//

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

this->разностьЭлементовToolStripMenuItem->Size = System::Drawing::Size(215, 22);

this->разностьЭлементовToolStripMenuItem->Text = L"Разность элементов";

this->разностьЭлементовToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_6::разностьЭлементовToolStripMenuItem\_Click);

//

// произведениеЭлементовToolStripMenuItem

//

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

this->произведениеЭлементовToolStripMenuItem->Size = System::Drawing::Size(215, 22);

this->произведениеЭлементовToolStripMenuItem->Text = L"Произведение элементов";

this->произведениеЭлементовToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_6::произведениеЭлементовToolStripMenuItem\_Click);

//

// частноеЭлементовToolStripMenuItem

//

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

this->частноеЭлементовToolStripMenuItem->Size = System::Drawing::Size(215, 22);

this->частноеЭлементовToolStripMenuItem->Text = L"Частное элементов";

this->частноеЭлементовToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_6::частноеЭлементовToolStripMenuItem\_Click);

//

// тестToolStripMenuItem

//

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

this->тестToolStripMenuItem->Size = System::Drawing::Size(42, 22);

this->тестToolStripMenuItem->Text = L"Тест";

this->тестToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_6::тестToolStripMenuItem\_Click);

//

// textBox

//

this->textBox->Dock = System::Windows::Forms::DockStyle::Fill;

this->textBox->Location = System::Drawing::Point(0, 24);

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

this->textBox->Multiline = true;

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

this->textBox->ReadOnly = true;

this->textBox->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->textBox->Size = System::Drawing::Size(1268, 642);

this->textBox->TabIndex = 1;

//

// LAB\_6

//

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

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

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

this->ClientSize = System::Drawing::Size(1268, 666);

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

this->Controls->Add(this->menuStrip1);

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

this->MainMenuStrip = this->menuStrip1;

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

this->MinimumSize = System::Drawing::Size(400, 200);

this->Name = L"LAB\_6";

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

this->Text = L"LAB\_6";

this->WindowState = System::Windows::Forms::FormWindowState::Maximized;

this->menuStrip1->ResumeLayout(false);

this->menuStrip1->PerformLayout();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

/// <summary>

/// Функция присвоения числового значения строковому значению текущей структуры данных

/// </summary>

/// <returns>Числовое значение соответствующее строковому названию</returns>

private: type type\_id()

{

if (current\_type == "int") return int\_type;

else if (current\_type == "char") return char\_type;

else if (current\_type == "double") return double\_type;

else return none;

}

//

// Процедура, запускающая действия при нажатии "Вывод структуры"

//

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

{

if (current\_type != "")

{

switch (type\_id())

{

case 1: this->textBox->Text = gcnew String(int\_stack.output().c\_str()); break;

case 2: this->textBox->Text = gcnew String(char\_stack.output().c\_str()); break;

case 3: this->textBox->Text = gcnew String(double\_stack.output().c\_str()); break;

}

}

else MessageBox::Show("Структура данных не выбрана", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

//

// Процедура, запускающая действия при нажатии "Добавить элемент"

//

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

{

if (current\_type != "")

{

AddElement^ ae = gcnew AddElement(type\_id());

ae->ShowDialog(this);

String^ sval = ae->get\_value();

if (!ae->canceled && sval != "")

{

switch (type\_id())

{

case 1:

{

int\_stack.push(Convert::ToInt32(sval));

this->textBox->Text = gcnew String(int\_stack.output().c\_str());

break;

}

case 2:

{

char\_stack.push(Convert::ToChar(sval));

this->textBox->Text = gcnew String(char\_stack.output().c\_str());

break;

}

case 3:

{

double\_stack.push(Convert::ToDouble(sval));

this->textBox->Text = gcnew String(double\_stack.output().c\_str());

break;

}

}

}

}

else MessageBox::Show("Структура данных не выбрана", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

//

// Процедура, запускающая действия при нажатии "Удаление последнего элемента"

//

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

{

if (type\_id() != none)

{

switch (type\_id())

{

case 1:

{

int\_stack.pop();

this->textBox->Text = gcnew String(int\_stack.output().c\_str());

break;

}

case 2:

{

char\_stack.pop();

this->textBox->Text = gcnew String(char\_stack.output().c\_str());

break;

}

case 3:

{

double\_stack.pop();

this->textBox->Text = gcnew String(double\_stack.output().c\_str());

break;

}

}

}

else MessageBox::Show("Структура данных не выбрана", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

//

// Процедура, запускающая действия при нажатии "Замена значения по индексу"

//

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

{

if (type\_id() != none)

{

ChangeByIndex^ cbi = gcnew ChangeByIndex(type\_id());

cbi->ShowDialog(this);

if (cbi->get\_value() != "")

{

if (!cbi->is\_canceled())

{

switch (type\_id())

{

case 1:

{

int\_stack.change\_by\_index(Convert::ToInt32(cbi->get\_index()), Convert::ToInt32(cbi->get\_value()));

this->textBox->Text = gcnew String(int\_stack.output().c\_str());

break;

}

case 2:

{

char\_stack.change\_by\_index(Convert::ToInt32(cbi->get\_index()), Convert::ToChar(cbi->get\_value()));

this->textBox->Text = gcnew String(char\_stack.output().c\_str());

break;

}

case 3:

{

double\_stack.change\_by\_index(Convert::ToInt32(cbi->get\_index()), Convert::ToDouble(cbi->get\_value()));

this->textBox->Text = gcnew String(double\_stack.output().c\_str());

break;

}

}

}

}

}

else MessageBox::Show("Структура данных не выбрана", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

//

// Процедура, запускающая действия при нажатии "Поиск индекса по значению"

//

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

{

if (type\_id() != none)

{

int x = this->Left + (this->Width / 2) - 200;

int y = this->Top + (this->Height / 2) - 100;

String^ find\_value = Interaction::InputBox("Введите значение\r\n\r\n", "Поиск индекса по значению", "", x, y);

if (find\_value != "")

{

int index = none;

switch (type\_id())

{

case 1: index = int\_stack.find\_index(Convert::ToInt32(find\_value)); break;

case 2: index = char\_stack.find\_index(Convert::ToChar(find\_value)); break;

case 3: index = double\_stack.find\_index(Convert::ToDouble(find\_value)); break;

}

MessageBox::Show("Индекс введенного значения — " + Convert::ToString(index), "Найти индекс", MessageBoxButtons::OK, MessageBoxIcon::Information);

}

}

else MessageBox::Show("Структура данных не выбрана", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

//

// Процедура, запускающая действия при нажатии "Поиск наибольшего элемента"

//

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

{

if (type\_id() != none)

{

switch (type\_id())

{

case 1:

{

int maxi = 0;

maxi = int\_stack.find\_max();

MessageBox::Show("Наибольшее значение — " + Convert::ToString(maxi), "Наибольшее значение", MessageBoxButtons::OK, MessageBoxIcon::Information);

break;

}

case 2:

{

char maxc = char\_stack.find\_max();

MessageBox::Show("Наибольшее значение — " + Convert::ToString(maxc) + " (" + Convert::ToChar(maxc) + ")", "Наибольшее значение", MessageBoxButtons::OK, MessageBoxIcon::Information);

break;

}

case 3:

{

double maxd = double\_stack.find\_max();

MessageBox::Show("Наибольшее значение — " + Convert::ToString(maxd), "Наибольшее значение", MessageBoxButtons::OK, MessageBoxIcon::Information);

break;

}

}

}

else MessageBox::Show("Структура данных не выбрана", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

//

// Процедура, запускающая действия при нажатии "Очистить структуру"

//

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

{

if (type\_id() != none)

{

System::Windows::Forms::DialogResult result = MessageBox::Show("Вы уверены что хотите очистить структуру?", "Предупреждение", MessageBoxButtons::YesNo, MessageBoxIcon::Warning);

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

{

switch (type\_id())

{

case 1: int\_stack.clear(); this->textBox->Text = gcnew String(int\_stack.output().c\_str()); break;

case 2: char\_stack.clear(); this->textBox->Text = gcnew String(char\_stack.output().c\_str()); break;

case 3: double\_stack.clear(); this->textBox->Text = gcnew String(double\_stack.output().c\_str()); break;

}

}

}

else MessageBox::Show("Структура данных не выбрана", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

//

// Процедура, запускающая действия при нажатии "Тестовые данных"

//

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

{

if (type\_id() != none)

{

System::Windows::Forms::DialogResult result = System::Windows::Forms::MessageBox::Show("Предварительно структура будет очищена. Вы уверены, что хотите ввести тестовые данные?", "Предупреждение", MessageBoxButtons::YesNo, MessageBoxIcon::Warning);

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

{

switch (type\_id())

{

case 1: int\_stack.fill\_test(); this->textBox->Text = gcnew String(int\_stack.output().c\_str()); break;

case 2: char\_stack.fill\_test(); this->textBox->Text = gcnew String(char\_stack.output().c\_str()); break;

case 3: double\_stack.fill\_test(); this->textBox->Text = gcnew String(double\_stack.output().c\_str()); break;

}

}

}

else MessageBox::Show("Структура данных не выбрана", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

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

{

if (type\_id() != none)

{

switch (type\_id())

{

case 1:

{

MessageBox::Show("Сумма элементов целочисленного стека — " + Convert::ToString(int\_stack.sum()), "Сумма элементов", MessageBoxButtons::OK, MessageBoxIcon::Information);

break;

}

case 2:

{

String^ str = gcnew String(char\_stack.sum\_of\_char().c\_str());

MessageBox::Show("Сумма элементов символьного стека — " + str, "Сумма элементов", MessageBoxButtons::OK, MessageBoxIcon::Information);

break;

}

case 3:

{

MessageBox::Show("Сумма элементов вещественного стека — " + Convert::ToString(double\_stack.sum()), "Сумма элементов", MessageBoxButtons::OK, MessageBoxIcon::Information);

break;

}

}

}

else MessageBox::Show("Структура данных не выбрана", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

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

{

if (type\_id() != none)

{

switch (type\_id())

{

case 1:

{

MessageBox::Show("Разность элементов целочисленного стека — " + Convert::ToString(int\_stack.difference()), "Сумма элементов", MessageBoxButtons::OK, MessageBoxIcon::Information);

break;

}

case 2:

{

MessageBox::Show("Разность элементов символьного стека — \"\"", "Сумма элементов", MessageBoxButtons::OK, MessageBoxIcon::Information);

break;

}

case 3:

{

MessageBox::Show("Разность элементов вещественного стека — " + Convert::ToString(double\_stack.difference()), "Сумма элементов", MessageBoxButtons::OK, MessageBoxIcon::Information);

break;

}

}

}

else MessageBox::Show("Структура данных не выбрана", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

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

{

if (type\_id() != none)

{

switch (type\_id())

{

case 1:

{

MessageBox::Show("Произведение элементов целочисленного стека — " + Convert::ToString(int\_stack.product()), "Сумма элементов", MessageBoxButtons::OK, MessageBoxIcon::Information);

break;

}

case 2:

{

MessageBox::Show("Произведение элементов символьного стека — \"\"", "Сумма элементов", MessageBoxButtons::OK, MessageBoxIcon::Information);

break;

}

case 3:

{

MessageBox::Show("Произведение элементов вещественного стека — " + Convert::ToString(double\_stack.product()), "Сумма элементов", MessageBoxButtons::OK, MessageBoxIcon::Information);

break;

}

}

}

else MessageBox::Show("Структура данных не выбрана", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

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

{

if (type\_id() != none)

{

switch (type\_id())

{

case 1:

{

MessageBox::Show("Частное элементов целочисленного стека — " + Convert::ToString(int\_stack.quotient()), "Сумма элементов", MessageBoxButtons::OK, MessageBoxIcon::Information);

break;

}

case 2:

{

MessageBox::Show("Частное элементов символьного стека — ", "Сумма элементов", MessageBoxButtons::OK, MessageBoxIcon::Information);

break;

}

case 3:

{

MessageBox::Show("Частное элементов вещественного стека — " + Convert::ToString(double\_stack.quotient()), "Сумма элементов", MessageBoxButtons::OK, MessageBoxIcon::Information);

break;

}

}

}

else MessageBox::Show("Структура данных не выбрана", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

//

// Процедура, запускающая действия при нажатии "int"

//

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

{

this->current\_type = "int";

if (this->Text == "LAB\_6") this->Text += " — int (Целочисленный)";

else this->Text = this->Text->Remove(8) + "int (Целочисленный)";

this->textBox->Text = gcnew String(int\_stack.output().c\_str());

}

//

// Процедура, запускающая действия при нажатии "char"

//

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

{

this->current\_type = "char";

if (this->Text == "LAB\_6") this->Text += " — char (Символьный)";

else this->Text = this->Text->Remove(8) + "char (Символьный)";

this->textBox->Text = gcnew String(char\_stack.output().c\_str());

}

//

// Процедура, запускающая действия при нажатии "double"

//

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

{

this->current\_type = "double";

if (this->Text == "LAB\_6") this->Text += " — double (Вещественный)";

else this->Text = this->Text->Remove(8) + "double (Вещественный)";

this->textBox->Text = gcnew String(double\_stack.output().c\_str());

}

/// <summary>

/// Функция тестирования ввода значений в структуру данных "Array"

/// </summary>

/// <returns>Результат тестирования</returns>

private: int test1()

{

templateStack<int> test(int\_type);

test.push(1);

String^ res = gcnew String(test.output().c\_str());

if (res == "[0] = 1;\r\n")

{

test.clear();

return 0;

}

else

{

test.clear();

return 1;

}

}

/// <summary>

/// Функция тестирования ввода значений в структуру данных "Vector"

/// </summary>

/// <returns>Результат тестирования</returns>

private: int test2()

{

templateStack<char> test(char\_type);

test.push(1);

String^ res = gcnew String(test.output().c\_str());

if (res == "[0] = \u0001;\r\n")

{

test.clear();

return 0;

}

else

{

test.clear();

return 1;

}

}

/// <summary>

/// Функция тестирования ввода значений в структуру данных "Stack"

/// </summary>

/// <returns>Результат тестирования</returns>

private: int test3()

{

templateStack<double> test(double\_type);

test.push(1);

String^ res = gcnew String(test.output().c\_str());

if (res == "[0] = 1.000000;\r\n")

{

test.clear();

return 0;

}

else

{

test.clear();

return 1;

}

}

/// <summary>

/// Функция тестирования ввода значений в структуры данных

/// </summary>

/// <returns>Результат тестирования</returns>

private: int test\_cases()

{

int result = test1() + test2() + test3();

if (result != 0)

{

MessageBox::Show("Тесты не пройдены. Ключевые функции работают некорректно. Приложение будет закрыто", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

this->Close();

}

return result;

}

//

// Процедура, запускающая действия при нажатии кнопки "Тест"

//

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

{

if (test\_cases() == 0)

MessageBox::Show("Тесты пройдены. Ключевые функции работают корректно.", "Уведомление", MessageBoxButtons::OK, MessageBoxIcon::Information);

}

};

};

Листинг файла LAB\_6.cpp

#include "LAB\_6.h"

#include "Windows.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThreadAttribute]

int main()

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

LAB6::LAB\_6 form;

try

{

Application::Run(% form);

}

catch (System::ObjectDisposedException^) {}

}

Листинг файла AddElement.h

#pragma once

#include "type.h"

namespace LAB6 {

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 AddElement

/// </summary>

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

{

public:

AddElement(type t)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

this->type\_id = t;

//if () this->label1->Text = "Введите символ:";

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~AddElement()

{

if (components)

{

delete components;

}

}

private: int type\_id;

public: bool canceled;

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

protected:

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

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

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

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

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

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

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

this->SuspendLayout();

//

// textBox1

//

this->textBox1->Location = System::Drawing::Point(187, 25);

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

this->textBox1->Size = System::Drawing::Size(252, 26);

this->textBox1->TabIndex = 0;

this->textBox1->TextChanged += gcnew System::EventHandler(this, &AddElement::textBox1\_TextChanged);

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(12, 25);

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

this->label1->Size = System::Drawing::Size(155, 20);

this->label1->TabIndex = 1;

this->label1->Text = L"Введите значение:";

//

// button1

//

this->button1->Location = System::Drawing::Point(332, 76);

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

this->button1->Size = System::Drawing::Size(107, 35);

this->button1->TabIndex = 1;

this->button1->Text = L"Добавить";

this->button1->UseVisualStyleBackColor = true;

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

//

// button2

//

this->button2->Location = System::Drawing::Point(220, 76);

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

this->button2->Size = System::Drawing::Size(95, 35);

this->button2->TabIndex = 3;

this->button2->Text = L"Отмена";

this->button2->UseVisualStyleBackColor = true;

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

//

// AddElement

//

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

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

this->ClientSize = System::Drawing::Size(451, 129);

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

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

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

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

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

this->Name = L"AddElement";

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

this->Text = L"Добавить элемент";

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

//

// Процедура, запускающая действия при нажатии кнопки "Добавить"

//

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

{

if (this->textBox1->Text != "")

{

if (check())

{

this->canceled = false;

this->Close();

}

}

else MessageBox::Show("Значение для нового элемента не введено", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

//

// Процедура, запускающая действия при нажатии кнопки "Отмена"

//

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

{

this->canceled = true;

this->Close();

}

//

// Процедура, запускающая действия при изменении текста в textBox1

//

private: System::Void textBox1\_TextChanged(System::Object^ sender, System::EventArgs^ e)

{

check();

}

// Функция проверки корректности ввода

// Вход void

// Выход bool

private: bool check()

{

if(type\_id == char\_type)

{

if (this->textBox1->Text != "" && (this->textBox1->Text->Length > 1 || !(this->textBox1->Text[0] > 0 && this->textBox1->Text[0] < 255)))

{

MessageBox::Show("Введено несколько символов или символ некорректен", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

return false;

}

return true;

}

else if (type\_id == int\_type)

{

String^ str = this->textBox1->Text;

int res = 0;

if (str != "")

{

bool check = int::TryParse(str, res);

if (!check)

{

MessageBox::Show("Некорректный ввод", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

this->textBox1->Text = this->textBox1->Text->Remove(str->Length - 1, 1);

return false;

}

}

return true;

}

else if (type\_id == double\_type)

{

String^ str = this->textBox1->Text;

double res = 0.0;

if (str != "")

{

bool check = Double::TryParse(str, res);

if (!check)

{

MessageBox::Show("Некорректный ввод", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

this->textBox1->Text = this->textBox1->Text->Remove(str->Length - 1, 1);

return false;

}

}

return true;

}

}

//

// Функция получения значения из поля ввода

// Вход void

// Выход System::String^

//

public: System::String^ get\_value()

{

if(type\_id == int\_type)

return this->textBox1->Text;

else if (type\_id == char\_type)

return Convert::ToString(Convert::ToInt32(Convert::ToChar(this->textBox1->Text)));

else if(type\_id == double\_type)

return this->textBox1->Text;

}

};

}

Листинг файла ChangeByInedx.h

#pragma once

#include "type.h"

namespace LAB6 {

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 ChangeByIndex

/// </summary>

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

{

public:

ChangeByIndex(type t)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

this->type\_id = t;

if (t == char\_type) this->label2->Text = "Введите новый символ:";

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~ChangeByIndex()

{

if (components)

{

delete components;

}

}

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

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

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

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

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

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

protected:

private: bool canceled;

private: int type\_id;

/// <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->index\_textBox = (gcnew System::Windows::Forms::TextBox());

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

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

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

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

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

this->SuspendLayout();

//

// index\_textBox

//

this->index\_textBox->Location = System::Drawing::Point(231, 33);

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

this->index\_textBox->Size = System::Drawing::Size(225, 26);

this->index\_textBox->TabIndex = 0;

this->index\_textBox->TextChanged += gcnew System::EventHandler(this, &ChangeByIndex::index\_textBox\_TextChanged);

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(12, 33);

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

this->label1->Size = System::Drawing::Size(217, 20);

this->label1->TabIndex = 1;

this->label1->Text = L"Введите индекс элемента:";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(12, 84);

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

this->label2->Size = System::Drawing::Size(204, 20);

this->label2->TabIndex = 2;

this->label2->Text = L"Введите новое значение:";

//

// textBox1

//

this->textBox1->Location = System::Drawing::Point(231, 84);

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

this->textBox1->Size = System::Drawing::Size(225, 26);

this->textBox1->TabIndex = 3;

this->textBox1->TextChanged += gcnew System::EventHandler(this, &ChangeByIndex::textBox1\_TextChanged);

//

// ok\_button

//

this->ok\_button->Location = System::Drawing::Point(381, 140);

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

this->ok\_button->Size = System::Drawing::Size(75, 39);

this->ok\_button->TabIndex = 4;

this->ok\_button->Text = L"OK";

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

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

//

// cancel\_button

//

this->cancel\_button->Location = System::Drawing::Point(275, 140);

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

this->cancel\_button->Size = System::Drawing::Size(85, 39);

this->cancel\_button->TabIndex = 5;

this->cancel\_button->Text = L"Отмена";

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

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

//

// ChangeByIndex

//

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

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

this->ClientSize = System::Drawing::Size(473, 201);

this->Controls->Add(this->cancel\_button);

this->Controls->Add(this->ok\_button);

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

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

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

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

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

this->Name = L"ChangeByIndex";

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

this->Text = L"Замена значения по индексу";

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

//

// Процедура, запускающая действия при нажатии кнопки "Отмена"

//

private: System::Void cancel\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

this->Close();

canceled = true;

}

//

// Процедура, запускающая действия при нажатии кнопки "OK"

//

private: System::Void ok\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (this->textBox1->Text != "" || this->index\_textBox->Text != "")

{

check\_index();

if (check())

{

this->Close();

canceled = false;

}

}

else MessageBox::Show("Значение для нового элемента не введено", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

//

// Функция получения значения из поля ввода индекса

// Вход void

// Выход System::String^

//

public: String^ get\_index()

{

return this->index\_textBox->Text;

}

//

// Функция получения значения глобальной переменной canceled

// Вход void

// Выход bool

//

public: bool is\_canceled()

{

return canceled;

}

//

// Процедура проверки корректонсти ввода индекса

//

private: void check\_index()

{

String^ str = index\_textBox->Text;

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

{

if (!(str[i] >= '0' && str[i] <= '9'))

{

if (str->Length > 1)

this->index\_textBox->Text = this->index\_textBox->Text->Remove(this->index\_textBox->TextLength - 1);

else

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

}

}

}

//

// Процедура, запускающая действия при изменении текста в index\_textBox

//

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

{

check\_index();

}

//

// Процедура, запускающая действия при изменении текста в textBox1

//

private: System::Void textBox1\_TextChanged(System::Object^ sender, System::EventArgs^ e)

{

check();

}

//

// Функция проверки корректности ввода в поле нового значения

// Вход void

// Выход bool

//

private: bool check()

{

if (type\_id == char\_type)

{

if (this->textBox1->Text != "" && (this->textBox1->Text->Length > 1 || !(this->textBox1->Text[0] > 0 && this->textBox1->Text[0] < 255)))

{

MessageBox::Show("Введено несколько символов или символ некорректен", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

return false;

}

return true;

}

else if (type\_id == int\_type)

{

String^ str = this->textBox1->Text;

int res = 0;

if (str != "")

{

bool check = int::TryParse(str, res);

if (!check)

{

MessageBox::Show("Некорректный ввод", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

this->textBox1->Text = this->textBox1->Text->Remove(str->Length - 1, 1);

return false;

}

}

return true;

}

else if (type\_id == double\_type)

{

String^ str = this->textBox1->Text;

double res = 0.0;

if (str != "")

{

bool check = Double::TryParse(str, res);

if (!check)

{

MessageBox::Show("Некорректный ввод", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

this->textBox1->Text = this->textBox1->Text->Remove(str->Length - 1, 1);

return false;

}

}

return true;

}

}

//

// Функция получения нового значения

// Вход void

// Выход System::String^

//

public: System::String^ get\_value()

{

if (type\_id == int\_type)

return this->textBox1->Text;

else if (type\_id == char\_type)

return Convert::ToString(Convert::ToChar(this->textBox1->Text));

else if (type\_id == double\_type)

return this->textBox1->Text;

}

};

}

Листинг файла templateStack.h

#pragma once

#include "type.h"

#include <cmath>

template <class T>

class templateStack

{

private:

typedef struct Stack {

T value;

int index;

Stack\* next;

}Stack;

private: type t;

private: Stack\* sHead; // Указатель на структура данных Stack

//

// Конструктор класса myStack

//

public: templateStack(type t)

{

this->t = t;

sHead = nullptr;

}

//

// Деструктор класса myStack

//

public: ~templateStack()

{

clear();

}

public: std::string output()

{

if (t == char\_type)

return output\_char();

else

return output\_not\_char();

}

/// <summary>

/// Функция вывода структуры данных в целочисленном формате

/// </summary>

/// <returns>Строка std::string</returns>

private: std::string output\_not\_char()

{

std::string stack = "";

if (sHead == nullptr) return "Стек пуст";

else

{

Stack\* curr = sHead;

while (curr != nullptr)

{

stack += "[" + std::to\_string(curr->index) + "] = " + std::to\_string(curr->value) + ";\r\n";

curr = curr->next;

}

return stack;

}

}

/// <summary>

/// Функция вывода структуры данных в символьном формате

/// </summary>

/// <returns>Строка std::string</returns>

private: std::string output\_char()

{

std::string stack = "";

if (sHead == nullptr) return "Стек пуст";

else

{

Stack\* curr = sHead;

while (curr != nullptr)

{

if (curr->value == '\0')

stack += "[" + std::to\_string(curr->index) + "] = " + "Ошибка перевода" + ";\r\n";

else

stack += "[" + std::to\_string(curr->index) + "] = " + (char(curr->value)) + ";\r\n";

curr = curr->next;

}

return stack;

}

}

/// <summary>

/// Процедура добавления нового элемента в вектор

/// </summary>

/// <param name="val">Значение нового элемента</param>

public: void push(T val)

{

if (sHead == nullptr) stack\_start(val);

else add\_element(val);

}

/// <summary>

/// Процедура удаления последнего элемента

/// </summary>

public: void pop()

{

if (sHead != nullptr)

{

Stack\* curr = sHead;

while (curr->next != nullptr) curr = curr->next;

if (curr == sHead)

{

sHead = nullptr;

delete curr;

}

else

{

Stack\* prev = sHead;

while (prev->next != curr) prev = prev->next;

prev->next = nullptr;

delete curr;

}

}

}

/// <summary>

/// Процедура изменения значения элемента по его индексу

/// </summary>

/// <param name="index">Индекс элемента для изменения</param>

/// <param name="new\_val">Новое значение элемента</param>

public: void change\_by\_index(int index, T new\_val)

{

Stack\* curr = sHead;

while (curr != nullptr)

{

if (curr->index == index)

break;

else if (curr != nullptr)

curr = curr->next;

}

if (curr != nullptr)

curr->value = new\_val;

else

MessageBox::Show("Введенного индекса не существует", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

/// <summary>

/// Функция поиска индекса элемента по его значению

/// </summary>

/// <param name="val">Значение элемента для поиска</param>

/// <returns name="">Индекс элемента в переданным значением</returns>

public: int find\_index(T val)

{

Stack\* curr = sHead;

while (curr->next != nullptr)

{

if (curr->value == val) break;

curr = curr->next;

}

if (curr != nullptr && curr->value == val)

return curr->index;

else

return -1;

}

/// <summary>

/// Функция поиска наибольшего элемента в структуре

/// </summary>

/// <returns>Значение наибольшего элемента структуры</returns>

public: T find\_max()

{

Stack\* curr = sHead;

if (curr != nullptr)

{

T max = curr->value;

while (curr != nullptr)

{

if (max < curr->value)

max = curr->value;

curr = curr->next;

}

return max;

}

return -1;

}

/// <summary>

/// Процедура очистки структуры данных

/// </summary>

public: void clear()

{

if (sHead != nullptr)

{

while (sHead != nullptr)

{

Stack\* curr = sHead;

while (curr->next != nullptr) curr = curr->next;

if (curr == sHead)

{

sHead = nullptr;

delete curr;

}

else

{

Stack\* prev = sHead;

while (prev->next != curr) prev = prev->next;

prev->next = nullptr;

delete curr;

}

}

}

}

/// <summary>

/// Процедура заполнения структуры данных тестовыми значениями

/// </summary>

public: void fill\_test()

{

clear();

if (t == int\_type)

{

for (int i = 1; i < 302; i += 5)

{

push(i);

}

}

else if (t == char\_type)

{

for (int i = 1; i < 302; i += 5)

{

if (i > 33 && i < 128)

push(i);

else

push('\0');

}

}

else if (t == double\_type)

{

for (float i = 1; i < 48; i += 0.7778)

{

push(i);

}

}

}

/// <summary>

/// Процедура добавления первого элемента

/// </summary>

private: void stack\_start(T val)

{

Stack\* temp = new Stack{ val, 0, nullptr };

sHead = temp;

}

/// <summary>

/// Процедура добавления элементов с индексом > 0

/// </summary>

/// <param name="val"></param>

private: void add\_element(T val)

{

Stack\* curr = sHead;

while (curr->next != nullptr)

curr = curr->next;

Stack\* new\_el = new Stack{ val, (curr->index + 1), nullptr };

curr->next = new\_el;

}

public: T sum()

{

T result = 0;

for (Stack\* curr = sHead; curr != nullptr; curr = curr->next)

{

if (curr != nullptr)

result += curr->value;

}

return result;

}

public: std::string sum\_of\_char()

{

std::string result = "";

for (Stack\* curr = sHead; curr != nullptr; curr = curr->next)

{

if (curr != nullptr)

if(curr->value > 33 && curr->value < 128)

result += (char)curr->value;

}

return result;

}

public: T difference()

{

T result = 0;

for (Stack\* curr = sHead; curr != nullptr; curr = curr->next)

{

if (curr != nullptr)

result -= curr->value;

}

return result;

}

public: T product()

{

int max = (std::pow(2, (sizeof(int) \* 4)));

T result = 1;

for (Stack\* curr = sHead; curr != nullptr; curr = curr->next)

{

if (curr != nullptr && result < max)

result \*= curr->value;

else break;

}

return result;

}

public: T quotient()

{

T result = 1;

for (Stack\* curr = sHead; curr != nullptr; curr = curr->next)

{

if (curr != nullptr && curr->value != 0)

result /= curr->value;

}

return result;

}

};

Листинг файла UnitTest.cpp

#include "pch.h"

#include "..\LAB\_6\type.h"

#include "templateStackUnitTest.h"

#include "CppUnitTest.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest

{

TEST\_CLASS(UnitTest)

{

public:

TEST\_METHOD(push)

{

tempStack stack = tempStack(int\_type);

stack.push(1);

stack.push(1000);

Assert::IsTrue(stack.output() == "[0] = 1;\r\n[1] = 1000;\r\n");

}

TEST\_METHOD(max)

{

tempStack stack = tempStack(int\_type);

stack.push(1);

stack.push(1000);

Assert::IsTrue(stack.find\_max() == 1000);

}

TEST\_METHOD(clear)

{

tempStack stack = tempStack(int\_type);

stack.push(1);

stack.push(1000);

stack.clear();

Assert::IsTrue(stack.output() == "Стек пуст");

}

};

}