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 №4. ARRAY, VECTOR, STACK, QUEUE

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 для работы структурами данных: Array, Vector, Stack, Queue
  2. **Наименование задачи** – Лабораторная работа №4. Array, Vector, Stack, Queue
  3. **Словесное описание**:

Напишите код программы, в которой реализован способ выбора структуры данных для работы из перечисленных: Array, Vector, Stack, Queue.

Напишите функции для вставки элементов в структуру данных, удаления элемента, замену элемента по индексу, вывод значений структуры на экран. Обратите внимание, что в программе можно выбирать любую структуру и функционал должен работать с каждой из них (то есть не для одной структуры делаем программу, а для всех).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Объект программы | Имя этого объекта в программе | Тип данных | Диапазоны представления | Простая/  структура | Вход/  выход/  Константа |
| Структура данных массив | arr | int | 0 - 200000000 | Структура | Вход / выход |
| Структура данных массив | c\_arr | char | 33 - 128 | Структура | Вход / выход |
| Структура данных вектор | vector | int | 0 - 200000000 | Структура | Вход / выход |
| Структура данных вектор | c\_vector | char | 33 - 128 | Структура | Вход / выход |
| Структура данных стек | stack | int | 0 - 200000000 | Структура | Вход / выход |
| Структура данных стек | c\_stack | char | 33 - 128 | Структура | Вход / выход |
| Структура данных очередь | queue | int | 0 - 200000000 | Структура | Вход / выход |
| Структура данных очередь | c\_queue | char | 33 - 128 | 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

* 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**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | | | T01 |
| Заголовок | | | Проверка добавления нового элемента |
| Предусловие | | | Приложение запущенно, выбрана структура 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 |

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | | | T02 |
| Заголовок | | | Конвертация структуры в символы |
| Предусловие | | | Приложение запущенно, структура выбрана |
| Ожидаемый результат | | | Вывод структуры в символьном формате |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Выбрать раздел «Операции» | 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 | Нажать на кнопку «Конвертация в символы» | Открытие окна предупреждение о том, что в структуре есть значения, которые невозможно перевести в символы | Pass |
| 5 | Нажать кнопку «ОК» | Закрытие окна предупреждения, вывод структуры в символьном формате, значения, которые невозможно перевести в символы заменены строкой «Ошибка перевода» | Pass |
| Результат | | | Pass |

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | | | T03 |
| Заголовок | | | Проверка работоспособности функции поиска максимального элемента |
| Предусловие | | | Приложение запущено, структура дынных выбрана |
| 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 «Операции» | 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 |

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | | | T04 |
| Заголовок | | | Проверка работоспособности функции очистки структуры данных |
| Предусловие | | | Приложение запущенно, структура выбрана |
| Ожидаемый результат | | | Структура очищена |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Выбрать раздел «Операции» | 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 «Операции» | Откроется меню раздела | Pass |
| 5 | Выбрать пункт «Очистить структуру» | Структура очищена, выведено сообщение о пустоте структуры | Pass |
| Результат | | | Pass |

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | | | T05 |
| Заголовок | | | Проверка работоспособности функции «Удаление последнего элемента» |
| Предусловие | | | Приложение запущено, структура выбрана |
| 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 «Операции» | Открытие меню раздела «Операции» | Pass |
| 3 | Выбрать пункт «Удаление последнего элемента» | Последний элемент удалён | Pass |
| Результат | | | Pass |

* 1. **Пример работающей программы**

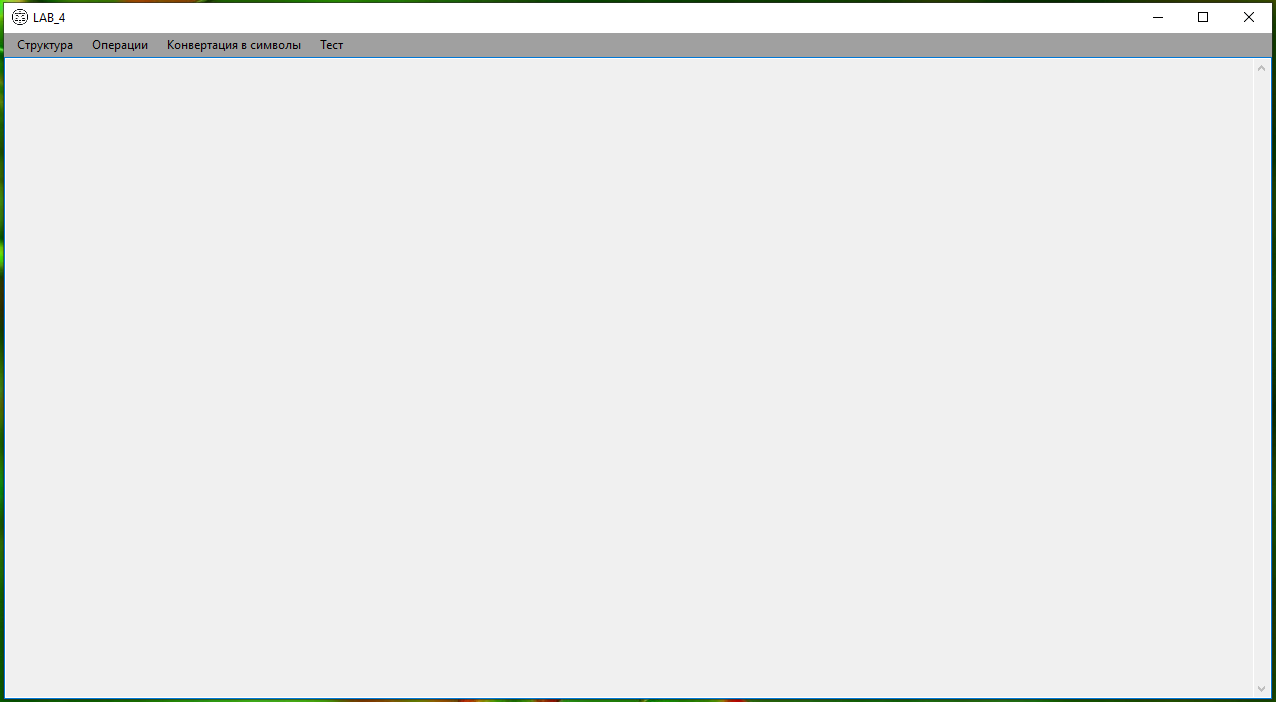


Рисунок 1 - Окно приложения

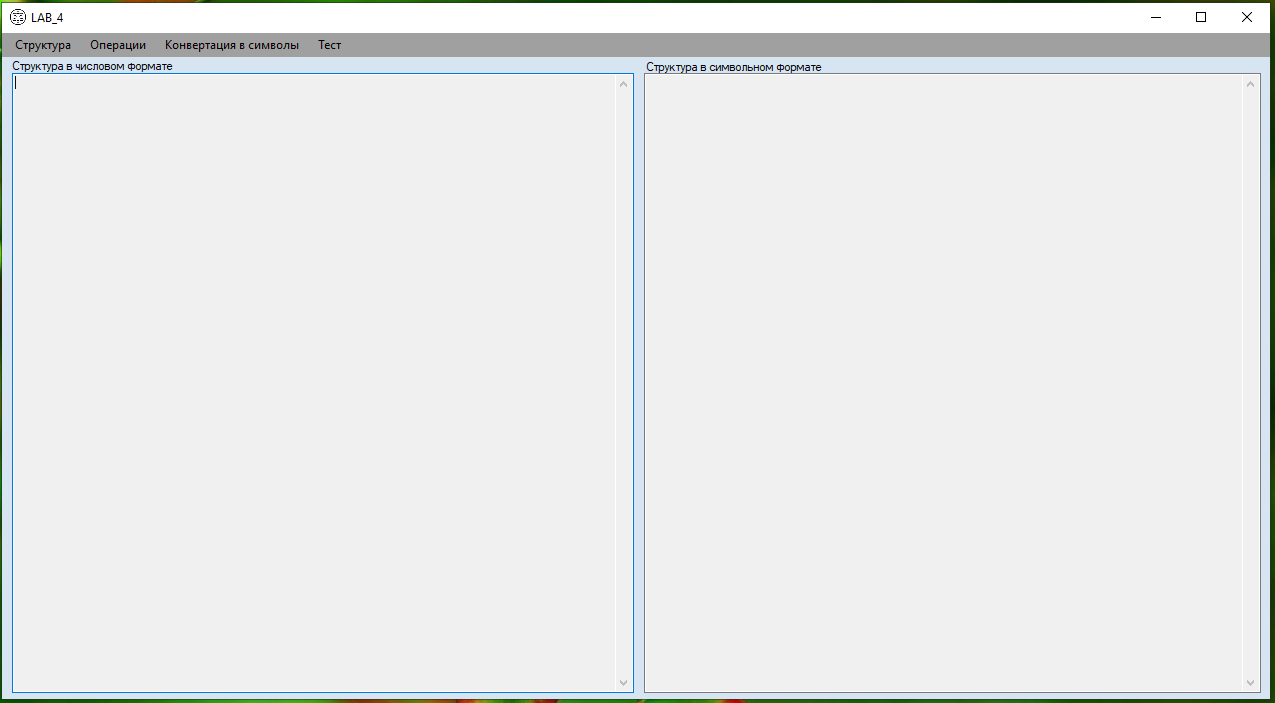


Рисунок 2 - Окно приложения в режиме конвертации в символы

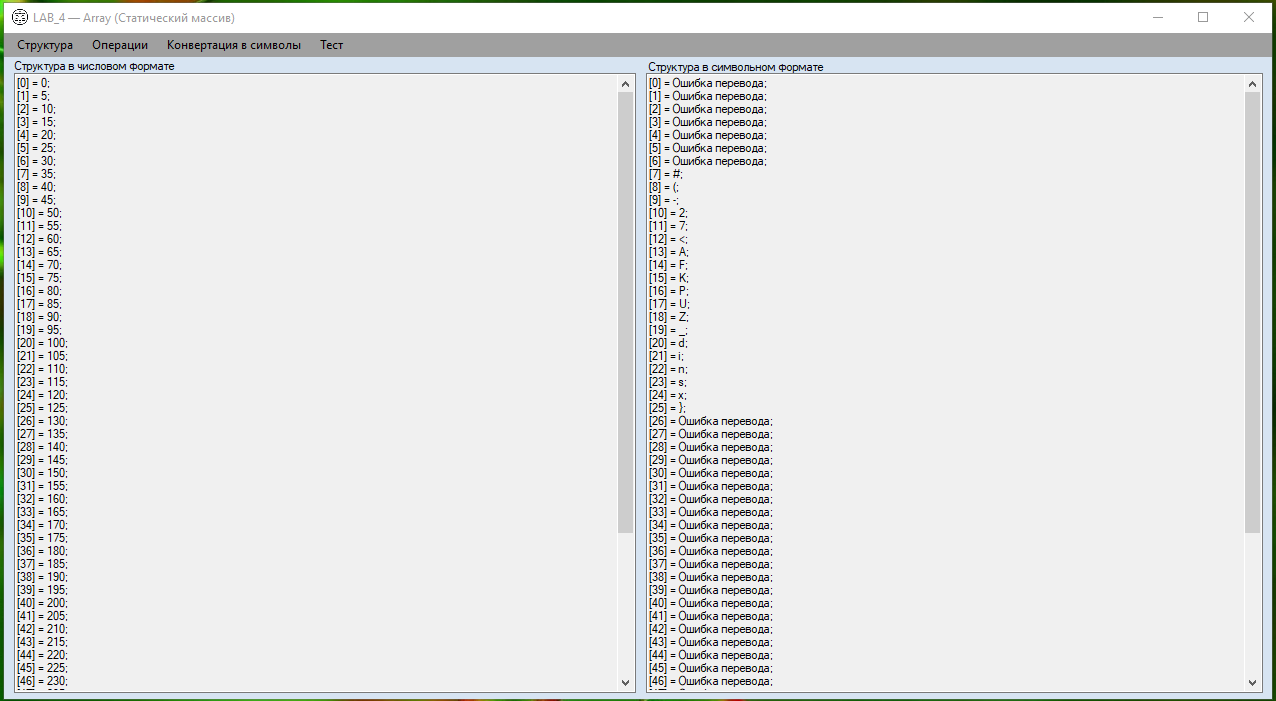


Рисунок 3 - Пример использования приложения

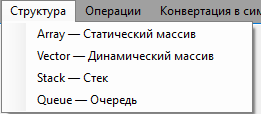


Рисунок 4 - Меню раздела "Структура"

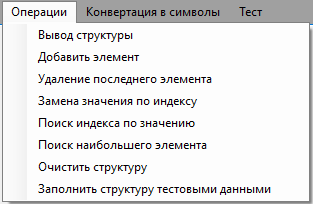


Рисунок 5 - Меню раздела "Операции"

1. **Проектирование задачи**

Реализовать приложение с графическим интерфейсом, использую библиотеку Windows Forms, для работы со структурами данных (Array, Vector, Stack, Queue). 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\_4.exe
  2. **Уточненное словесное описание задачи**

Напишите код программы, в которой реализован способ выбора структуры данных для работы из перечисленных: Array, Vector, Stack, Queue.

Напишите функции для вставки элементов в структуру данных, удаления элемента, замену элемента по индексу, вывод значений структуры на экран. Обратите внимание, что в программе можно выбирать любую структуру и функционал должен работать с каждой из них (то есть не для одной структуры делаем программу, а для всех).

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

Для начала работы надо выбрать структуры из списка «Структура», при выборе структуры данных, дальнейшие действия, до переключения структуры, будут выполняться только с текущей структурой данных.

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

Так же в приложении реализована работа со структурами символов, чтобы перейти в этот режим необходимо выбрать пункт «Конвертация в символы», в этом режиме все манипуляции выполняются на выбранной структурой, но уже с типом данных char. Для отключения данного режима необходимо повторное нажатие на пункт «Конвертация в символы».

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**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Объект программы | Имя этого объекта в программе | Характеристики | | | Как используется в программе |
| Тип данных | Диапазоны представления | Прост/структура/ константа |
| LAB\_4.h | Current\_struct | System::String^ |  | Структура | Хранит в себе название текущей структуры данных |
| LAB\_4.h | Char\_is\_enable | bool | True / False | Простая | Хранит информацию в текущем режиме приложения |
| myArray.h | n | Static const unsigned int | 2500 | Константа | Размер массива |
| myArray | arr | T (шаблонный) | 2500 элементов | Структура | Массив элементов |
| myArray.h | Curr\_index | Unsigned int | 0 - 2499 | Простая | Хранит следующий заполняемый индекс |
| myVector.h | Vector | Struct vector\* | — | Структура | Указатель на структуру Vector |
| myStack.h | sHead | Struct Stack\* | — | Структура | Указатель на структуру Stack |
| myQueue | qHead | Struct Queue\* | — | Структура | Указатель на структуру Queue |

* 1. **Декомпозиция функций**

Описывается каждая функция и процедура в программе

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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.

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

Блок-схема функции check() в Приложении В.

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

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

* 1. **Алгоритмизация**

Листинг файла LAB\_4.h в Приложение Е.

Листинг файла LAB\_4.cpp в Приложение Ж.

Листинг файла AddElement.h в Приложение И.

Листинг файла ChangeByIndex.h в Приложение К.

Листинг файла myArray.h в Приложение Л.

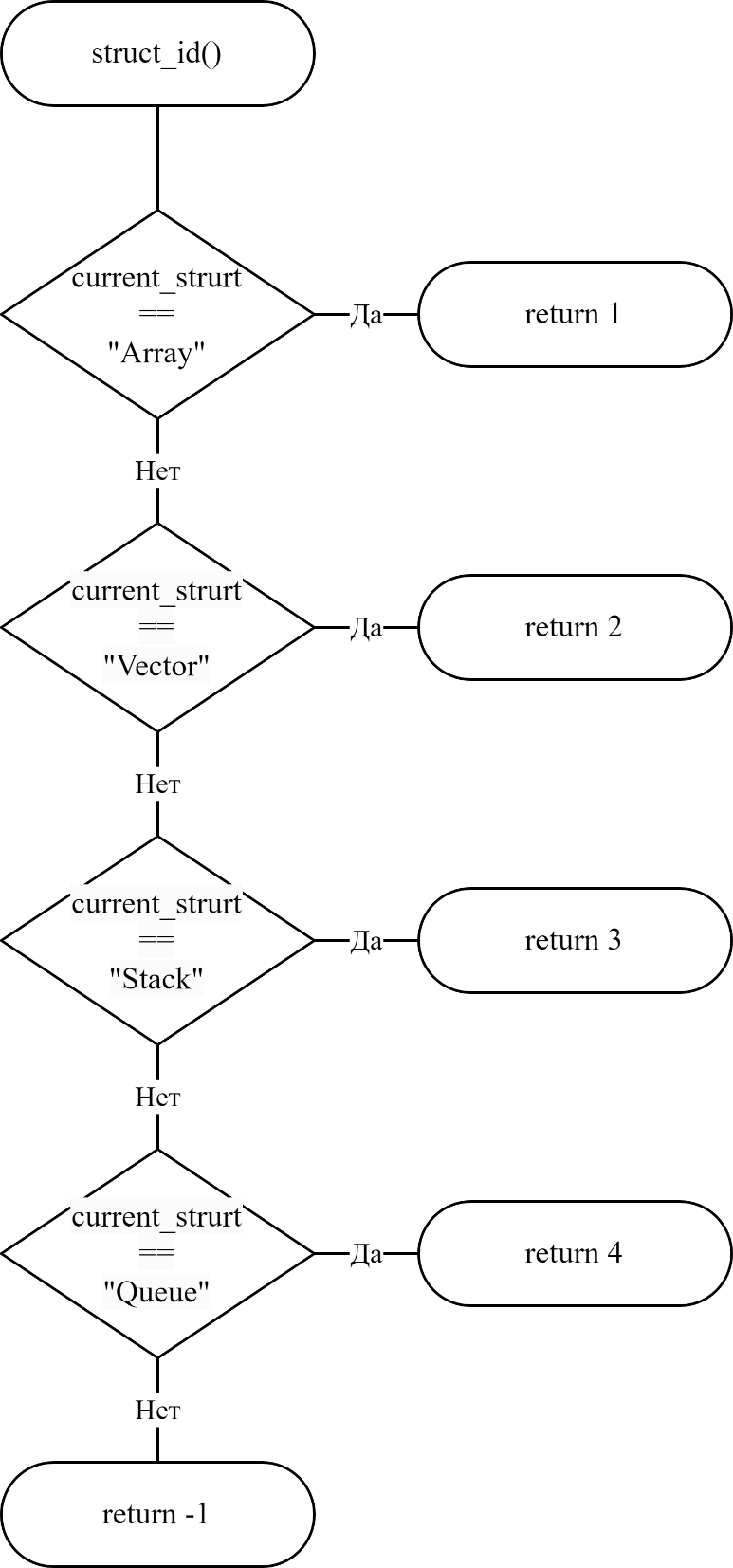
Листинг файла myVector.h в Приложение М.

Листинг файла myStack.h в Приложение Н.

Листинг файла myQueue.h в Приложение О.  
Листинг файла UnitTest.h в Приложение П.  
Листинг файла UnitTest.cpp в Приложение Р.  
Листинг файла pch.h в Приложение С.

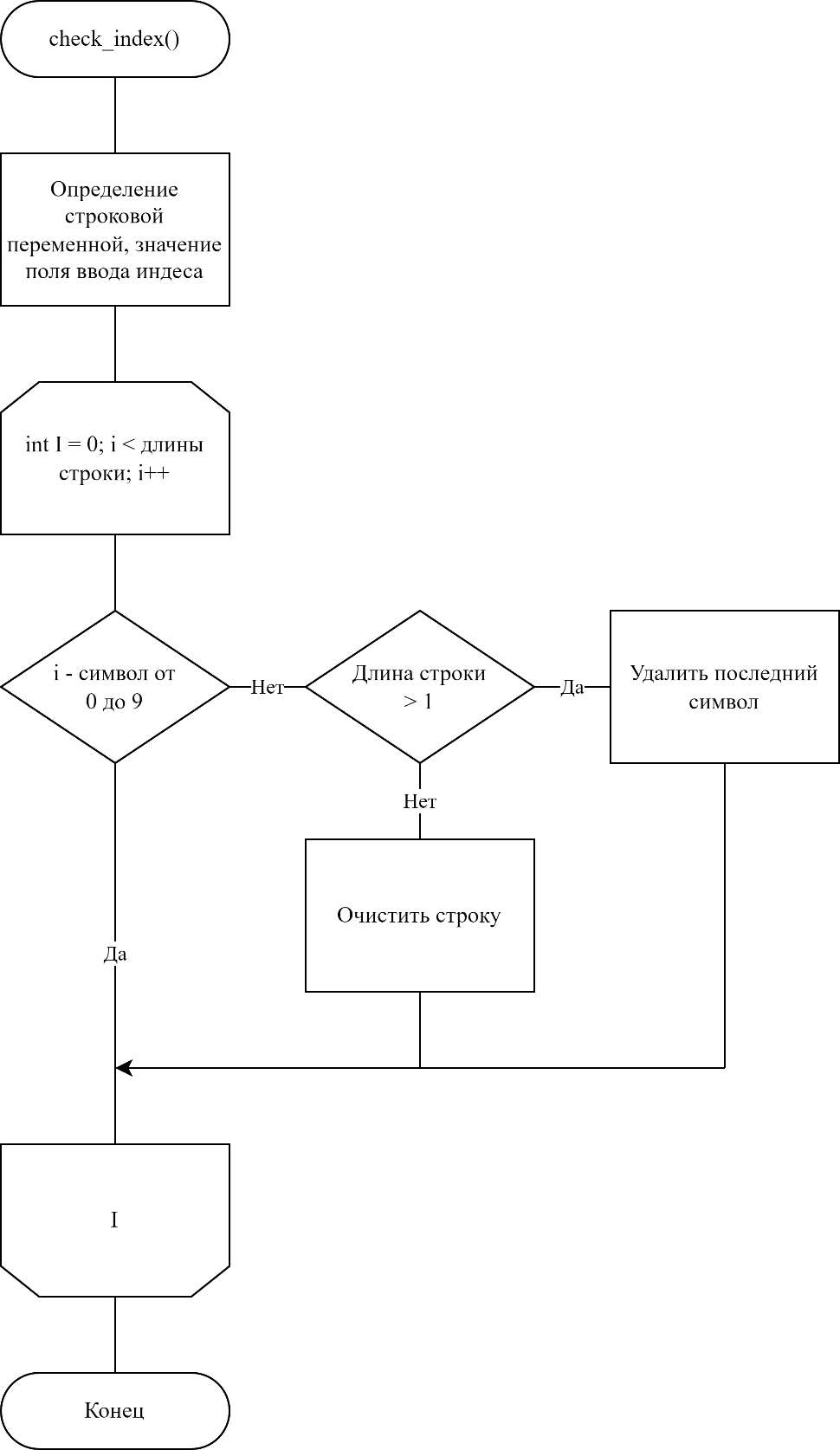
Приложение А

Блок-схема функции struct\_id()



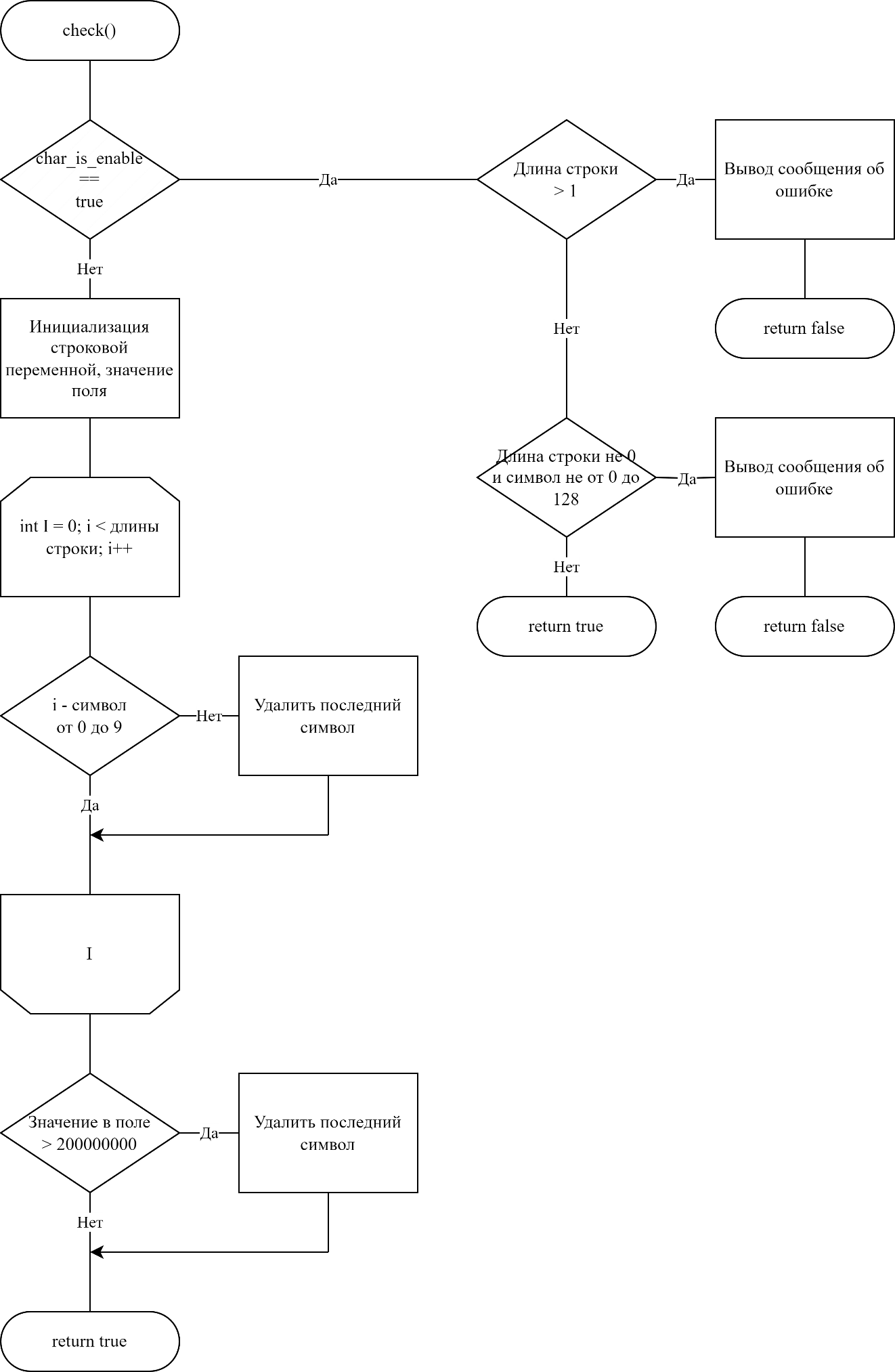
Приложение Б

Блок-схема процедуры check\_index()



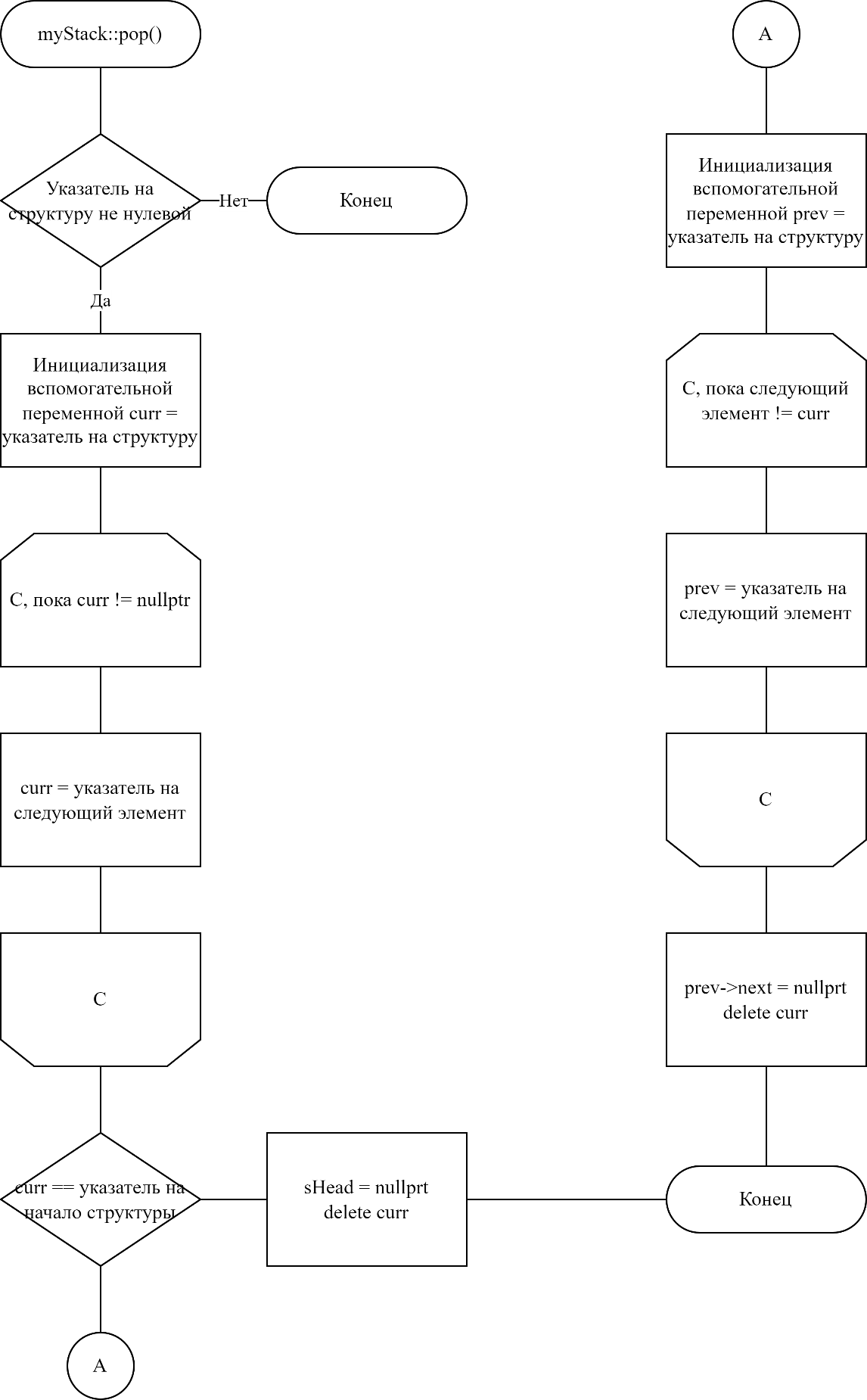
Приложение В

Блок-схема функции check()



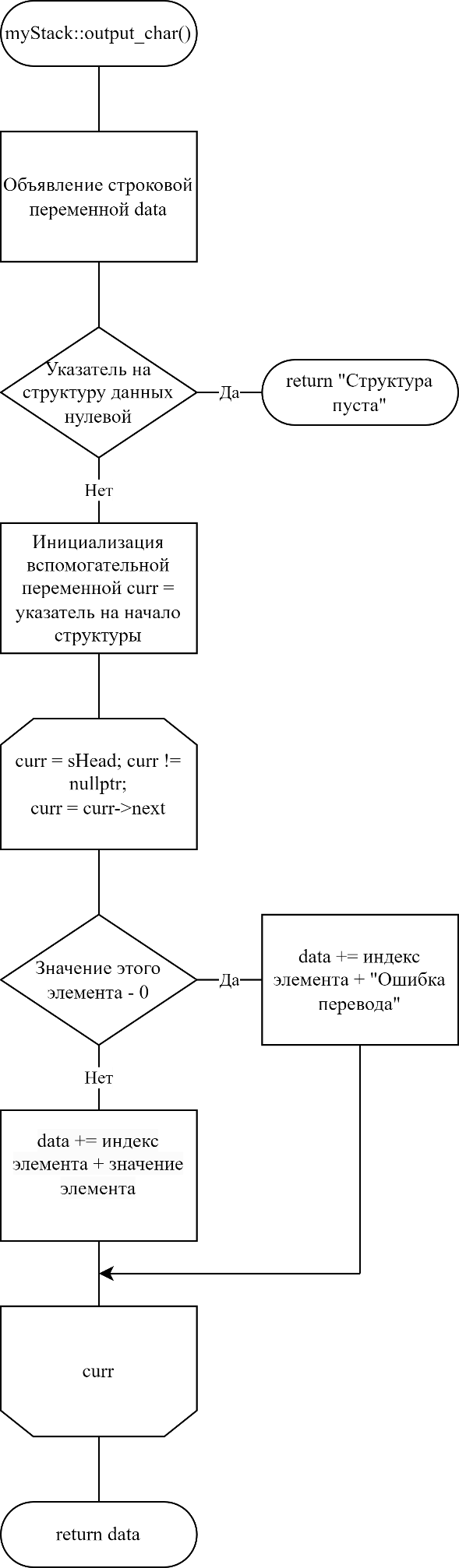
Приложение Г

Блок-схема процедуры myStack::pop()



Приложение Д

Блок-схема функции myStack::output\_char ()



Приложение Е

Листинг файла LAB\_4.h

#pragma once

#include <vector>

#include <string>

#include "myArray.h"

#include "myVector.h"

#include "myQueue.h"

#include "myStack.h"

#include "ChangeByIndex.h"

#include "AddElement.h"

namespace LAB4 {

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;

myArray<int> arr;

myVector<int> vector;

myStack<int> stack;

myQueue<int> queue;

myArray<char> c\_arr;

myVector<char> c\_vector;

myStack<char> c\_stack;

myQueue<char> c\_queue;

/// <summary>

/// Сводка для LAB\_4

/// </summary>

public ref class LAB\_4 : public System::Windows::Forms::Form

{

public:

LAB\_4(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~LAB\_4()

{

vector.clear();

stack.clear();

queue.clear();

if (components)

{

delete components;

}

}

protected:

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

private: String^ current\_struct = "";

private: bool char\_is\_enable = false;

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^ queueToolStripMenuItem;

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^ конвертацияВASCIIToolStripMenuItem;

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

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

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

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

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

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

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

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\_4::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->queueToolStripMenuItem = (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->конвертацияВASCIIToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

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

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

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

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

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^ >(3) {

this->структураToolStripMenuItem,

this->операцииToolStripMenuItem, this->конвертацияВASCIIToolStripMenuItem

});

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^ >(4) {

this->arrayToolStripMenuItem,

this->vectorToolStripMenuItem, this->stackToolStripMenuItem, this->queueToolStripMenuItem

});

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

this->структураToolStripMenuItem->Size = System::Drawing::Size(75, 22);

this->структураToolStripMenuItem->Text = L"Структура";

//

// arrayToolStripMenuItem

//

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

this->arrayToolStripMenuItem->Size = System::Drawing::Size(251, 22);

this->arrayToolStripMenuItem->Text = L"Array — Статический массив";

this->arrayToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_4::arrayToolStripMenuItem\_Click);

//

// vectorToolStripMenuItem

//

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

this->vectorToolStripMenuItem->Size = System::Drawing::Size(251, 22);

this->vectorToolStripMenuItem->Text = L"Vector — Динамический массив";

this->vectorToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_4::vectorToolStripMenuItem\_Click);

//

// stackToolStripMenuItem

//

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

this->stackToolStripMenuItem->Size = System::Drawing::Size(251, 22);

this->stackToolStripMenuItem->Text = L"Stack — Стек";

this->stackToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_4::stackToolStripMenuItem\_Click);

//

// queueToolStripMenuItem

//

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

this->queueToolStripMenuItem->Size = System::Drawing::Size(251, 22);

this->queueToolStripMenuItem->Text = L"Queue — Очередь";

this->queueToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_4::queueToolStripMenuItem\_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(75, 22);

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

//

// 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\_4::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\_4::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\_4::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\_4::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\_4::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\_4::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\_4::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\_4::тестовыеДанныеToolStripMenuItem\_Click);

//

// конвертацияВASCIIToolStripMenuItem

//

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

this->конвертацияВASCIIToolStripMenuItem->Size = System::Drawing::Size(153, 22);

this->конвертацияВASCIIToolStripMenuItem->Text = L"Конвертация в символы";

this->конвертацияВASCIIToolStripMenuItem->Click += gcnew System::EventHandler(this, &LAB\_4::конвертацияВASCIIToolStripMenuItem\_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;

//

// char\_textBox

//

this->char\_textBox->Location = System::Drawing::Point(653, 21);

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

this->char\_textBox->Multiline = true;

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

this->char\_textBox->ReadOnly = true;

this->char\_textBox->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->char\_textBox->Size = System::Drawing::Size(614, 646);

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

this->char\_textBox->Visible = false;

//

// int\_label

//

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

this->int\_label->Location = System::Drawing::Point(8, 26);

this->int\_label->Margin = System::Windows::Forms::Padding(2, 0, 2, 0);

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

this->int\_label->Size = System::Drawing::Size(167, 13);

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

this->int\_label->Text = L"Структура в числовом формате";

this->int\_label->Visible = false;

//

// char\_label

//

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

this->char\_label->Location = System::Drawing::Point(661, 26);

this->char\_label->Margin = System::Windows::Forms::Padding(2, 0, 2, 0);

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

this->char\_label->Size = System::Drawing::Size(182, 13);

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

this->char\_label->Text = L"Структура в символьном формате";

this->char\_label->Visible = false;

//

// LAB\_4

//

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->char\_label);

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

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

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(1284, 668);

this->Name = L"LAB\_4";

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

this->Text = L"LAB\_4";

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

this->menuStrip1->ResumeLayout(false);

this->menuStrip1->PerformLayout();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: int struct\_check()

{

if (current\_struct == "Array") return 1;

else if (current\_struct == "Vector") return 2;

else if (current\_struct == "Stack") return 3;

else if (current\_struct == "Queue") return 4;

else return -1;

}

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

{

if (current\_struct != "")

{

switch (struct\_check())

{

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

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

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

case 4: this->textBox->Text = gcnew String(queue.output().c\_str()); break;

}

to\_char();

}

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

}

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

{

if (current\_struct != "")

{

AddElement^ ae = gcnew AddElement(char\_is\_enable);

ae->ShowDialog(this);

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

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

{

int val = Convert::ToInt32(sval);

switch (struct\_check())

{

case 1:

{

arr.push(val);

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

break;

}

case 2:

{

vector.push(val);

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

break;

}

case 3:

{

stack.push(val);

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

break;

}

case 4:

{

queue.push(val);

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

break;

}

}

}

to\_char();

}

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

}

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

{

if (struct\_check() != -1)

{

switch (struct\_check())

{

case 1:

{

arr.pop();

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

break;

}

case 2:

{

vector.pop();

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

break;

}

case 3:

{

stack.pop();

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

break;

}

case 4:

{

queue.pop();

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

break;

}

}

to\_char();

}

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

}

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

{

if (struct\_check() != -1)

{

ChangeByIndex^ cbi = gcnew ChangeByIndex(char\_is\_enable);

cbi->ShowDialog(this);

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

{

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

{

int val = Convert::ToInt32(cbi->get\_val());

switch (struct\_check())

{

case 1:

{

arr.change\_by\_index(Convert::ToInt32(cbi->get\_index()), val);

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

break;

}

case 2:

{

vector.change\_by\_index(Convert::ToInt32(cbi->get\_index()), val);

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

break;

}

case 3:

{

stack.change\_by\_index(Convert::ToInt32(cbi->get\_index()), val);

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

break;

}

case 4:

{

queue.change\_by\_index(Convert::ToInt32(cbi->get\_index()), val);

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

break;

}

}

}

to\_char();

}

}

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

}

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

{

if (struct\_check() != -1)

{

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

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

if (!char\_is\_enable)

{

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

if (find\_value != "")

{

int value = Convert::ToInt32(find\_value);

int index = -1;

switch (struct\_check())

{

case 1: index = arr.find\_index(value); break;

case 2: index = vector.find\_index(value); break;

case 3: index = stack.find\_index(value); break;

case 4: index = queue.find\_index(value); break;

}

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

}

}

else

{

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

if (find\_value != "")

{

int value = Convert::ToChar(find\_value);

int index = -1;

switch (struct\_check())

{

case 1: index = c\_arr.find\_index(value); break;

case 2: index = c\_vector.find\_index(value); break;

case 3: index = c\_stack.find\_index(value); break;

case 4: index = c\_queue.find\_index(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);

}

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

{

if (struct\_check() != -1)

{

if (!char\_is\_enable)

{

int index = -1;

switch (struct\_check())

{

case 1: index = arr.find\_max(); break;

case 2: index = vector.find\_max(); break;

case 3: index = stack.find\_max(); break;

case 4: index = queue.find\_max(); break;

}

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

}

else

{

int index = -1;

switch (struct\_check())

{

case 1: index = c\_arr.find\_max(); break;

case 2: index = c\_vector.find\_max(); break;

case 3: index = c\_stack.find\_max(); break;

case 4: index = c\_queue.find\_max(); break;

}

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

}

}

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

}

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

{

if (struct\_check() != -1)

{

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 (struct\_check())

{

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

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

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

case 4: queue.clear(); this->textBox->Text = gcnew String(queue.output().c\_str()); break;

}

to\_char();

}

}

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

}

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

{

if (struct\_check() != -1)

{

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 (struct\_check())

{

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

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

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

case 4: queue.fill\_test(); this->textBox->Text = gcnew String(queue.output().c\_str()); break;

}

to\_char();

}

}

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

}

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

{

this->current\_struct = "Array";

if (this->Text == "LAB\_4") this->Text += " — Array (Статический массив)";

else this->Text = this->Text->Remove(8) + "Array (Статический массив)";

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

if (char\_is\_enable) to\_char();

}

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

{

this->current\_struct = "Vector";

if (this->Text == "LAB\_4") this->Text += " — Vector (Динамический массив)";

else this->Text = this->Text->Remove(8) + "Vector (Динамический массив)";

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

if (char\_is\_enable) to\_char();

}

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

{

this->current\_struct = "Stack";

if (this->Text == "LAB\_4") this->Text += " — Stack (Стек)";

else this->Text = this->Text->Remove(8) + "Stack (Стек)";

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

if (char\_is\_enable) to\_char();

}

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

{

this->current\_struct = "Queue";

if (this->Text == "LAB\_4") this->Text += " — Queue (Очередь)";

else this->Text = this->Text->Remove(8) + "Queue (Очередь)";

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

if (char\_is\_enable) to\_char();

}

private: void to\_char()

{

if (char\_is\_enable)

{

this->textBox->Dock = System::Windows::Forms::DockStyle::None;

this->textBox->Size = System::Drawing::Size(this->Width / 2 - 20, this->Height - 85);

this->textBox->Location = System::Drawing::Point(10, 40);

this->char\_textBox->Size = System::Drawing::Size(this->Width / 2 - 25, this->Height - 85);

this->char\_textBox->Location = System::Drawing::Point(this->Width / 2, 40);

this->char\_textBox->Visible = true;

char\_is\_enable = true;

this->int\_label->Visible = true;

this->char\_label->Location = System::Drawing::Point(this->Width / 2, 27);

this->char\_label->Visible = true;

switch (struct\_check())

{

case 1:

{

c\_arr.clear();

c\_arr.copy\_to\_char(arr);

this->char\_textBox->Text = gcnew String(c\_arr.output\_char().c\_str());

}; break;

case 2:

{

c\_vector.clear();

c\_vector.copy\_to\_char(vector.get\_arr(), vector.get\_size());

this->char\_textBox->Text = gcnew String(c\_vector.output\_char().c\_str());

}; break;

case 3:

{

c\_stack.clear();

c\_stack.copy\_to\_char(stack.get\_values());

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

}; break;

case 4:

{

c\_queue.clear();

c\_queue.copy\_to\_char(queue.get\_values());

this->char\_textBox->Text = gcnew String(c\_queue.output\_char().c\_str());

}; break;

}

}

else

{

switch (struct\_check())

{

case 1: c\_arr.clear(); break;

case 2: c\_vector.clear(); break;

case 3: c\_stack.clear(); break;

case 4: c\_queue.clear(); break;

}

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

this->textBox->Size = System::Drawing::Size(this->Width, 991);

this->char\_textBox->Size = System::Drawing::Size(this->Width / 2 - 10, 991);

this->char\_textBox->Location = System::Drawing::Point(this->Width / 2 - 10, 33);

this->char\_textBox->Visible = false;

char\_is\_enable = false;

this->int\_label->Visible = false;

this->char\_label->Visible = false;

}

}

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

{

if (char\_is\_enable) char\_is\_enable = false;

else char\_is\_enable = true;

to\_char();

}

};

}

Приложение Ж

Листинг файла LAB\_4.cpp

#include "LAB\_4.h"

#include "Windows.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThreadAttribute]

int main()

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

LAB4::LAB\_4 form;

Application::Run(% form);

}

Приложение И

Листинг файла AddElement.h

#pragma once

namespace LAB4 {

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(bool char\_is\_enable)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

this->char\_is\_enable = char\_is\_enable;

if (char\_is\_enable) this->label1->Text = "Введите символ:";

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~AddElement()

{

if (components)

{

delete components;

}

}

private: bool char\_is\_enable;

public: bool cancled;

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

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

{

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

{

if (check())

{

this->cancled = false;

this->Close();

}

}

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

}

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

{

this->cancled = true;

this->Close();

}

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

{

check();

}

private: bool check()

{

if (char\_is\_enable)

{

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;

}

/\*else if (Convert::ToInt32(this->textBox1->Text) < 0 || Convert::ToInt32(this->textBox1->Text) > 128)

{

System::Windows::Forms::DialogResult result = MessageBox::Show("Введенное значение невозможно перевести в символ", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

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

{

this->textBox1->Text = this->textBox1->Text->Remove(this->textBox1->TextLength - 1);

}

}\*/

return true;

}

else

{

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

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

{

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

{

this->textBox1->Text = this->textBox1->Text->Remove(this->textBox1->TextLength - 1);

}

}

return true;

}

}

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

{

if (char\_is\_enable)

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

else

return this->textBox1->Text;

}

};

}

Приложение К

Листинг файла ChangeByInedx.h

#pragma once

namespace LAB4 {

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(bool char\_is\_enable)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

this->char\_is\_enable = char\_is\_enable;

if (char\_is\_enable) 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^ cancle\_button;

protected:

private: bool cancled;

private: bool char\_is\_enable;

/// <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->cancle\_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);

//

// cancle\_button

//

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

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

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

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

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

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

this->cancle\_button->Click += gcnew System::EventHandler(this, &ChangeByIndex::cancle\_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->cancle\_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

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

{

this->Close();

cancled = true;

}

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();

cancled = false;

}

}

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

}

public: String^ get\_index()

{

return this->index\_textBox->Text;

}

public: bool is\_cancled()

{

return cancled;

}

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 = "";

}

}

}

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

{

check\_index();

}

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

{

check();

}

private: bool check()

{

if (char\_is\_enable)

{

if (this->textBox1->Text->Length > 1)

{

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

return false;

}

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

{

MessageBox::Show("Введенный символ некорректен", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

return false;

}

/\*else if (Convert::ToInt32(this->textBox1->Text) < 0 || Convert::ToInt32(this->textBox1->Text) > 128)

{

System::Windows::Forms::DialogResult result = MessageBox::Show("Введенное значение невозможно перевести в символ", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

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

{

this->textBox1->Text = this->textBox1->Text->Remove(this->textBox1->TextLength - 1);

}

}\*/

return true;

}

else

{

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

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

{

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

{

this->textBox1->Text = this->textBox1->Text->Remove(this->textBox1->TextLength - 1);

}

}

return true;

}

}

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

{

if (char\_is\_enable)

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

else

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

}

};

}

Приложение Л

Листинг файла myArray.h

#pragma once

template <class T>

class myArray

{

private: static const int n = 2500;

private: T arr[n];

private: int curr\_index = 0;

public: myArray()

{

for (int i = 0; i < n; i++)

{

arr[i] = 0;

}

}

public: std::string output()

{

if (curr\_index > 0)

{

std::string data = "";

for (int i = 0; i < curr\_index; i++)

{

data += "[" + std::to\_string(i) + "] = " + std::to\_string(arr[i]) + ";\r\n";

}

return data;

}

else return "Массив пуст";

}

public: std::string output\_char()

{

if (curr\_index > 0)

{

std::string data = "";

for (int i = 0; i < curr\_index; i++)

{

if(arr[i] == 0)

data += "[" + std::to\_string(i) + "] = " + "Ошибка перевода" + ";\r\n";

else

data += "[" + std::to\_string(i) + "] = " + (char(arr[i])) + ";\r\n";

}

return data;

}

else return "Массив пуст";

}

public: void push(T val)

{

arr[curr\_index] = val;

curr\_index++;

}

public: void pop()

{

arr[curr\_index] = 0;

curr\_index--;

}

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

{

if (index < curr\_index)

arr[index] = new\_val;

else

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

}

public: int find\_index(T val)

{

int i = 0;

while (arr[i] != val && i < n)

{

i++;

}

if (i >= n - 1) return -1;

else return i;

}

public: T find\_max()

{

if (curr\_index != 0)

{

T max = arr[0];

for (int i = 0; i < curr\_index; i++)

{

if (arr[i] > max) max = arr[i];

}

return max;

}

return -1;

}

public: void clear()

{

for (int i = 0; i < n; i++)

{

arr[i] = 0;

}

curr\_index = 0;

}

public: void fill\_test()

{

clear();

{

for (int i = 0; i < 301; i += 5)

{

push(i);

}

}

}

public: void copy\_to\_char(myArray<int> a)

{

int n = 0;

for (int i = 0; i < a.get\_index(); i++)

{

if (get\_el(a, i) > 32 && get\_el(a, i) < 128)

{

push(get\_el(a, i));

}

else

{

push(0);

if (n == 0)

{

n = 1;

MessageBox::Show("В структуре присутствуют значения, которые невозможно перевести в символы", "Предупреждение", MessageBoxButtons::OK, MessageBoxIcon::Warning);

}

}

}

}

private: int get\_el(myArray<int> a, int index)

{

return a.get\_value(index);

}

public: int get\_index()

{

return curr\_index;

}

public: int get\_value(int index)

{

return arr[index];

}

};

Приложение М

Листинг файла myVector.h

template <class T>

class myVector

{

private: typedef struct vector

{

T\* arr\_ptr;

int size;

int curr\_index;

}vector;

private: vector\* Vector;

public: myVector()

{

Vector = new vector{ nullptr, 0, 0 };

}

public: ~myVector()

{

clear();

}

public: std::string output()

{

if (Vector->size > 0)

{

std::string data = "";

for (int i = 0; i < Vector->curr\_index; i++)

{

data += "[" + std::to\_string(i) + "] = " + std::to\_string(Vector->arr\_ptr[i]) + ";\r\n";

}

return data;

}

else return "Вектор пуст";

}

public: std::string output\_char()

{

if (Vector->curr\_index > 0)

{

std::string data = "";

for (int i = 0; i < Vector->curr\_index; i++)

{

char ch = char(0);

if (Vector->arr\_ptr[i] == ch)

data += "[" + std::to\_string(i) + "] = " + "Ошибка перевода" + ";\r\n";

else

data += "[" + std::to\_string(i) + "] = " + (char(Vector->arr\_ptr[i])) + ";\r\n";

}

return data;

}

else return "Вектор пуст";

}

public: void init()

{

T\* arr = new T[1];

Vector->arr\_ptr = arr;

Vector->size = 1;

}

private: void new\_mem()

{

T\* new\_arr = new T[++Vector->size];

for (int i = 0; i < Vector->size - 1; i++)

{

new\_arr[i] = Vector->arr\_ptr[i];

}

delete[] Vector->arr\_ptr;

Vector->arr\_ptr = new\_arr;

}

public: void push(T val)

{

if (Vector->size == 0)

init();

if (Vector->curr\_index != 0 && Vector->curr\_index == Vector->size)

new\_mem();

Vector->arr\_ptr[Vector->curr\_index] = val;

Vector->curr\_index++;

}

public: void pop()

{

if (Vector->size > 0)

{

T\* new\_arr = new T[--Vector->size];

for (int i = 0; i < Vector->size; i++)

{

new\_arr[i] = Vector->arr\_ptr[i];

}

delete[] Vector->arr\_ptr;

Vector->arr\_ptr = new\_arr;

Vector->curr\_index--;

}

}

public: void pop\_front()

{

if (Vector->size > 0)

{

T\* new\_arr = new T[--Vector->size];

for (int i = 1; i < Vector->size; i++)

{

new\_arr[i] = Vector->arr\_ptr[i];

}

delete[] Vector->arr\_ptr;

Vector->arr\_ptr = new\_arr;

Vector->curr\_index--;

}

}

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

{

if(index < Vector->curr\_index)

Vector->arr\_ptr[index] = new\_val;

else

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

}

public: int find\_index(T val)

{

int i = 0;

while (Vector->arr\_ptr[i] != val && i < Vector->size)

{

i++;

}

if (i >= Vector->size) return -1;

else return i - 1;

}

public: int find\_max()

{

if (Vector->curr\_index != 0)

{

T max = Vector->arr\_ptr[0];

for (int i = 0; i < Vector->curr\_index; i++)

{

if (Vector->arr\_ptr[i] > max) max = Vector->arr\_ptr[i];

}

return max;

}

return -1;

}

public: void clear()

{

delete[] Vector->arr\_ptr;

Vector->arr\_ptr = nullptr;

Vector->curr\_index = 0;

Vector->size = 0;

}

public: void fill\_test()

{

clear();

for (int i = 0; i < 301; i += 5)

{

push(i);

}

}

public: void copy\_to\_char(int arr[], int n)

{

int k = 0;

if (n > 0)

{

for (int i = 0; i < n; i++)

{

if (arr[i] > 32 && arr[i] < 128)

{

push(arr[i]);

}

else

{

push(0);

if (k == 0)

{

k = 1;

MessageBox::Show("В структуре присутствуют значения, которые невозможно перевести в символы", "Предупреждение", MessageBoxButtons::OK, MessageBoxIcon::Warning);

}

}

}

}

}

public: int\* get\_arr()

{

return Vector->arr\_ptr;

}

public: int get\_size()

{

return Vector->size;

}

public: int get\_value(int index)

{

return Vector->arr\_ptr[index];

}

};

Приложение Н

Листинг файла myStack.h

#pragma once

template <class T>

class myStack

{

private:

typedef struct Stack {

T value;

int index;

Stack\* next;

}Stack;

private: Stack\* sHead;

public: myStack()

{

sHead = nullptr;

}

public: ~myStack()

{

clear();

}

public: std::string output()

{

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;

}

}

public: std::string output\_char()

{

std::string stack = "";

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

else

{

Stack\* curr = sHead;

while (curr != nullptr)

{

if(curr->value == char(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;

}

}

public: void push(T val)

{

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

else add\_element(val);

}

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;

}

}

}

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);

}

public: int find\_index(T val)

{

Stack\* curr = sHead;

while (curr->value != val) curr = curr->next;

if (curr != nullptr)

return curr->index;

else

return -1;

}

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;

}

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;

}

}

}

}

public: void fill\_test()

{

clear();

for (int i = 0; i < 301; i += 5)

{

push(i);

}

}

private: void stack\_start(T val)

{

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

sHead = temp;

}

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: void copy\_to\_char(std::vector<int> values)

{

int n = 0, i = 0;

if (values.size() != 0)

{

while (i != values.size())

{

if (values[i] > 32 && values[i] < 128)

{

push(Convert::ToChar(values[i]));

}

else

{

push(0);

if (n == 0)

{

n = 1;

MessageBox::Show("В структуре присутствуют значения, которые невозможно перевести в символы", "Предупреждение", MessageBoxButtons::OK, MessageBoxIcon::Warning);

}

}

i++;

}

}

}

public: std::vector<int> get\_values()

{

std::vector<int> values;

Stack\* curr = sHead;

while (curr != nullptr)

{

if (curr != nullptr)

{

values.push\_back(curr->value);

}

curr = curr->next;

}

return values;

}

};

Приложение О

Листинг файла myQueue.h

#pragma once

template <class T>

class myQueue

{

private:

typedef struct Queue {

T value;

int index;

Queue\* next;

}Queue;

private:

typedef struct iQueue {

int value;

int index;

Queue\* next;

}iQueue;

private: Queue\* qHead;

public: myQueue()

{

qHead = nullptr;

}

public: ~myQueue()

{

clear();

}

public: std::string output()

{

std::string queue = "";

if (qHead == nullptr) return "Очередь пуста";

else

{

Queue\* curr = qHead;

while (curr != nullptr)

{

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

curr = curr->next;

}

return queue;

}

}

public: std::string output\_char()

{

std::string queue = "";

if (qHead == nullptr) return "Очередь пуст";

else

{

Queue\* curr = qHead;

while (curr != nullptr)

{

if (curr->value == char(0))

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

else

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

curr = curr->next;

}

return queue;

}

}

public: void push(T val)

{

if (qHead == nullptr) queue\_start(val);

else add\_element(val);

}

public: void pop()

{

if (qHead != nullptr)

{

Queue\* curr = qHead;

if (curr == qHead)

{

qHead = curr->next;

delete curr;

}

else

{

qHead = nullptr;

delete curr;

}

refresh\_index();

}

}

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

{

Queue\* curr = qHead;

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);

}

public: int find\_index(int val)

{

Queue\* curr = qHead;

while (curr->value != val) curr = curr->next;

if (curr != nullptr)

return curr->index;

else

return -1;

}

public: int find\_max()

{

Queue\* curr = qHead;

if (curr != nullptr)

{

int max = curr->value;

while (curr != nullptr)

{

if (max < curr->value)

max = curr->value;

curr = curr->next;

}

return max;

}

return -1;

}

public: void clear()

{

if (qHead != nullptr)

{

while (qHead != nullptr)

{

Queue\* curr = qHead;

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

if (curr == qHead)

{

qHead = nullptr;

delete curr;

}

else

{

Queue\* prev = qHead;

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

delete curr;

prev->next = nullptr;

}

}

}

}

public: void fill\_test()

{

clear();

for (int i = 0; i < 301; i += 5)

{

push(i);

}

}

private: void queue\_start(T val)

{

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

qHead = temp;

}

private: void add\_element(T val)

{

Queue\* curr = qHead;

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

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

curr->next = new\_el;

}

private: void refresh\_index()

{

int index = 0;

Queue\* curr = qHead;

while (curr != nullptr)

{

curr->index = index++;

curr = curr->next;

}

}

public: void copy\_to\_char(std::vector<int> values)

{

int n = 0, i = 0;

if (values.size() != 0)

{

while (i != values.size())

{

if (values[i] > 32 && values[i] < 128)

{

push(Convert::ToChar(values[i]));

}

else

{

push(0);

if (n == 0)

{

n = 1;

MessageBox::Show("В структуре присутствуют значения, которые невозможно перевести в символы", "Предупреждение", MessageBoxButtons::OK, MessageBoxIcon::Warning);

}

}

i++;

}

}

}

public: std::vector<int> get\_values()

{

std::vector<int> values;

Queue\* curr = qHead;

while (curr != nullptr)

{

if (curr != nullptr)

{

values.push\_back(curr->value);

}

curr = curr->next;

}

return values;

}

};

Приложение П

Листинг файла UnitTest.cpp

#include "pch.h"

#include "CppUnitTest.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest

{

TEST\_CLASS(UnitTest)

{

public:

TEST\_METHOD(pushArray)

{

myArray<int> a;

a.push(1);

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

}

TEST\_METHOD(pushVector)

{

myVector<int> v;

v.push(2);

Assert::IsTrue(v.output() == "[0] = 2;\r\n");

}

TEST\_METHOD(pushStack)

{

myStack<int> s;

s.push(3);

Assert::IsTrue(s.output() == "[0] = 3;\r\n");

}

TEST\_METHOD(pushQueue)

{

myQueue<int> q;

q.push(4);

Assert::IsTrue(q.output() == "[0] = 4;\r\n");

}

};

}

Приложение Р

Листинг файла pch.h

// pch.h: это предварительно скомпилированный заголовочный файл.

// Перечисленные ниже файлы компилируются только один раз, что ускоряет последующие сборки.

// Это также влияет на работу IntelliSense, включая многие функции просмотра и завершения кода.

// Однако изменение любого из приведенных здесь файлов между операциями сборки приведет к повторной компиляции всех(!) этих файлов.

// Не добавляйте сюда файлы, которые планируете часто изменять, так как в этом случае выигрыша в производительности не будет.

#ifndef PCH\_H

#define PCH\_H

#include <string>

#include <vector>

#include "..\\LAB\_4\myArray.h"

#include "..\\LAB\_4\myVector.h"

#include "..\\LAB\_4\myStack.h"

#include "..\\LAB\_4\myQueue.h"

// Добавьте сюда заголовочные файлы для предварительной компиляции

#endif //PCH\_H