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. структуры данных

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 3092  / И. С. Цветков  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |
|  |  |

## Постановка задачи

Разработка программы с графическим интерфейсом для работы со структурами данных.

## Наименование задачи

Лабораторная работа №4. Структуры данных.

## Словесное описание

Разработка программы с графическим интерфейсом для работы со структурами данных, соответствующая следующим требованиям:

­­­– Поддержка четырёх видов структур данных, таких как: массив, вектор, стек и очередь.

– Наличие меню для выбора структуры данных, поле ввода для добавления, замены и поворота элемента.

– Функция нахождения медианы структуры данных.

## Внешние спецификации функций программы

Программа должна предоставлять инструменты для создания и изменения 4 видов структур данный

## Математическая формулировка задачи

Пользователь взаимодействует с четырьмя видами структур хранения данных:

– Массив

– Вектор

– Стек

– Очередь

Он может добавлять новые элементы для структуры, изменять их, поворачивать, а также находить медиану структуры. Функция изменения элементов и нахождения медианы доступны не для всех видов структур, так некоторые из них не позволяют это сделать в виду особенностей собственного алгоритма работы.

## Спецификация интерфейса

Для написания графического интерфейса была использована библиотека Windows Forms.

Плюсы интерфейса заключаются в том, что реализован удобный выбор варианта структуры и способа взаимодействия с ней.

Отображение элементов выбранной структуры данных происходит в отдельном окне отображения.

Каждый элемент структуры данных подписан соответствующим ему индексом в ней.

## 1.6 Внешние данные тестирования

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

| Идентификатор | | | T01 |
| --- | --- | --- | --- |
| Заголовок | | | Проверка выхода из приложения |
| Предусловие | | | Приложение не запущено |
| Ожидаемый результат | | | Успешный выход из приложения |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Запуск UniLab4.exe | Открытие главного окна приложения | Pass |
| 2 | Нажатие на кнопку «Выход» | Выход из приложения | Pass |
| Результат | | | Pass |

## Таблица 2. Тест-кейс №2, внесение элемента в структуру

| Идентификатор | | | T02 |
| --- | --- | --- | --- |
| Заголовок | | | Проверка внесения элемента в структуру |
| Предусловие | | | Приложение запущено |
| Ожидаемый результат | | | Успешный вывод элемента в поле отображения элементов структуры |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Выбор варианта структуры | Отображение выбранного массива | Pass |
| 2 | Ввод значения элемента и нажатие на кнопку «Ввод» | Вывод элемента в поле отображение элементов структуры | Pass |
| Результат | | | Pass |

## Таблица 3. Тест-кейс №3, замена элемента

| Идентификатор | | | T03 |
| --- | --- | --- | --- |
| Заголовок | | | Проверка замены элемента |
| Предусловие | | | Приложение запущено и создана подходящая структура |
| Ожидаемый результат | | | Успешная замена элемента структуры |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Ввод нового значения элемента и нажатие на кнопку «Заменить» | Отображение нового значения элемента структуры в поле отображения | Pass |
| Результат | | | Pass |

## Таблица 4. Тест-кейс №4, поворот элемента

| Идентификатор | | | T04 |
| --- | --- | --- | --- |
| Заголовок | | | Проверка поворота элемента структуры |
| Предусловие | | | Приложение запущено и создана подходящая структура |
| Ожидаемый результат | | | Успешный поворот по элементу структуры |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Ввод индекса элемента структуры и нажатие на кнопку «Заменить» | Отображение поворота структуры по заданному элементу в поле отображения | Pass |
| Результат | | | Pass |

## Таблица 5. Тест-кейс №5, нахождение медианы

| Идентификатор | | | T05 |
| --- | --- | --- | --- |
| Заголовок | | | Проверка нахождения медианы структуры |
| Предусловие | | | Приложение запущено и создана подходящая структура |
| Ожидаемый результат | | | Успешная открытие информационного окна |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Нажатие на кнопку «Информация» | Отображение окна информации о работе с программой | Pass |
| Результат | | | Pass |

## 1.7 Пример работающей программы

Интерфейс программы сделан максимально удобным и не нагруженным лишними элементами.

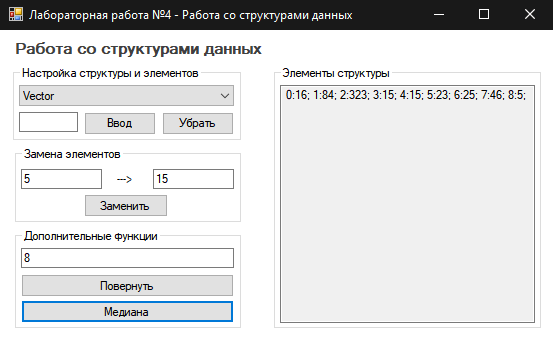


Рисунок 1 – Интерфейс программы

# 2. Проектирование задачи

Реализовать приложение с графическим интерфейсом, используя библиотеку WinForms, для работы со структурами данных. Создание удобного интерфейса для создания и изменения структур, а также возможность нахождения медианы некоторых структур.

# 2.1 Наименование программы

Программа имеет название UniLab4.exe.

# 2.2 Уточненное словесное описание задачи

Написать код программы и графический интерфейс, в котором можно создавать и редактировать четыре вида структур данных. Для трёх вариантов массивов, представленных в методических материалах данной работы, нужно реализовать работу пяти видов сортировок. Нужно создать меню выбора структуры данных, сделать окна вывода элемента структуры. По мимо этого отображать введённые в структуру элементы вместе с индексами, которые им соответствуют. Реализовать возможность нахождения медианы определённых структур данных.

# 2.3 Выбор метода решения поставленной задачи

Программа сортировок работает следующим образом: Пользователь выбирает одну из структур данных, представленных в меню программы. Исходя из его выбора, создаётся структура, в которую можно вводить элементы. После чего, исходя из особенностей той или иной структуры можно совершать те или иные манипуляции по изменению данных структур.

При нажатии кнопки «Заменить» происходит замена элемента, который вы хотите заменить. Для этого нужно указать соответствующий индекс элемента, который вы хотите заменить и его новое значение.

Кнопка «Поворот» совершает поворот структуры данных по элементу, который вы можете выбрать, указав его индекс.

Кнопка «Медиана» находит медиану тех структур, которые позволяют это сделать.

# Декомпозиция функций

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назначение | Имя | Параметры | | Процедура  Или функция | Включена в  проверку |
| Вход: тип | Выход: тип |
| Функция обновления данных | UpdateOutput | int | void | Функция | Нет |
| Кнопка меню выбора структуры | structureComboBox\_SelectedIndexChanged | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Кнопка добавления элемента | insertButton\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Кнопка удаления элемента | removeButton\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Кнопка замены элемента | replaceButton\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
|  | rotateButton\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Кнопка нахождения медианы структуры | medianButton\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Да |

# Алгоритмизация:

– Листинг файла UniLab4.h в Приложение А;

– Листинг файла MainForm.cpp в Приложение Б.

– Листинг файла IDataStructure.h в Приложение В

# Приложение А

# Листинг программы, файл UniLab4.h

#pragma once

#include "IDataStructure.h"

namespace UniLab4 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

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

{

public:

MainForm(void)

{

InitializeComponent();

dataStructure = new MyArray();

}

protected:

~MainForm()

{

if (components)

{

delete components;

}

delete dataStructure;

}

private: IDataStructure\* dataStructure;

private: System::Windows::Forms::ComboBox^ structureComboBox;

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

#pragma region Windows Form Designer generated code

void InitializeComponent(void)

{

this->structureComboBox = (gcnew System::Windows::Forms::ComboBox());

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

this->groupBox1->SuspendLayout();

this->groupBox2->SuspendLayout();

this->groupBox3->SuspendLayout();

this->groupBox4->SuspendLayout();

this->SuspendLayout();

//

// structureComboBox

//

this->structureComboBox->BackColor = System::Drawing::SystemColors::Window;

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

this->structureComboBox->DropDownStyle = System::Windows::Forms::ComboBoxStyle::DropDownList;

this->structureComboBox->FormattingEnabled = true;

this->structureComboBox->Items->AddRange(gcnew cli::array< System::Object^ >(4) { L"Array", L"Vector", L"Stack", L"Queue" });

this->structureComboBox->Location = System::Drawing::Point(6, 19);

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

this->structureComboBox->Size = System::Drawing::Size(215, 21);

this->structureComboBox->TabIndex = 0;

this->structureComboBox->SelectedIndexChanged += gcnew System::EventHandler(this, &MainForm::structureComboBox\_SelectedIndexChanged);

//

// inputTextBox

//

this->inputTextBox->Location = System::Drawing::Point(6, 46);

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

this->inputTextBox->Size = System::Drawing::Size(59, 20);

this->inputTextBox->TabIndex = 1;

//

// insertButton

//

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

this->insertButton->Location = System::Drawing::Point(71, 46);

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

this->insertButton->Size = System::Drawing::Size(72, 23);

this->insertButton->TabIndex = 2;

this->insertButton->Text = L"Ввод";

this->insertButton->TextAlign = System::Drawing::ContentAlignment::TopCenter;

this->insertButton->UseVisualStyleBackColor = true;

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

//

// removeButton

//

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

this->removeButton->Location = System::Drawing::Point(149, 46);

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

this->removeButton->Size = System::Drawing::Size(72, 23);

this->removeButton->TabIndex = 3;

this->removeButton->Text = L"Убрать";

this->removeButton->TextAlign = System::Drawing::ContentAlignment::TopCenter;

this->removeButton->UseVisualStyleBackColor = true;

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

//

// indexTextBox

//

this->indexTextBox->Location = System::Drawing::Point(6, 22);

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

this->indexTextBox->Size = System::Drawing::Size(81, 20);

this->indexTextBox->TabIndex = 4;

//

// replaceTextBox

//

this->replaceTextBox->Location = System::Drawing::Point(138, 22);

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

this->replaceTextBox->Size = System::Drawing::Size(81, 20);

this->replaceTextBox->TabIndex = 5;

//

// replaceButton

//

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

this->replaceButton->Location = System::Drawing::Point(69, 47);

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

this->replaceButton->Size = System::Drawing::Size(84, 23);

this->replaceButton->TabIndex = 6;

this->replaceButton->Text = L"Заменить";

this->replaceButton->UseVisualStyleBackColor = true;

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

//

// rotateTextBox

//

this->rotateTextBox->Location = System::Drawing::Point(6, 19);

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

this->rotateTextBox->Size = System::Drawing::Size(213, 20);

this->rotateTextBox->TabIndex = 7;

//

// rotateButton

//

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

this->rotateButton->Location = System::Drawing::Point(6, 45);

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

this->rotateButton->Size = System::Drawing::Size(213, 23);

this->rotateButton->TabIndex = 8;

this->rotateButton->Text = L"Повернуть";

this->rotateButton->UseVisualStyleBackColor = true;

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

//

// medianButton

//

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

this->medianButton->Location = System::Drawing::Point(6, 71);

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

this->medianButton->Size = System::Drawing::Size(213, 23);

this->medianButton->TabIndex = 9;

this->medianButton->Text = L"Медиана";

this->medianButton->UseVisualStyleBackColor = true;

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

//

// outputTextBox

//

this->outputTextBox->Location = System::Drawing::Point(6, 19);

this->outputTextBox->Multiline = true;

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

this->outputTextBox->ReadOnly = true;

this->outputTextBox->Size = System::Drawing::Size(255, 238);

this->outputTextBox->TabIndex = 10;

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

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

this->label1->ForeColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(64)), static\_cast<System::Int32>(static\_cast<System::Byte>(64)),

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

this->label1->Location = System::Drawing::Point(12, 9);

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

this->label1->Size = System::Drawing::Size(254, 18);

this->label1->TabIndex = 11;

this->label1->Text = L"Работа со структурами данных";

//

// label3

//

this->label3->AutoSize = true;

this->label3->Location = System::Drawing::Point(99, 25);

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

this->label3->Size = System::Drawing::Size(22, 13);

this->label3->TabIndex = 13;

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

//

// groupBox1

//

this->groupBox1->Controls->Add(this->structureComboBox);

this->groupBox1->Controls->Add(this->removeButton);

this->groupBox1->Controls->Add(this->inputTextBox);

this->groupBox1->Controls->Add(this->insertButton);

this->groupBox1->Location = System::Drawing::Point(13, 36);

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

this->groupBox1->Size = System::Drawing::Size(228, 75);

this->groupBox1->TabIndex = 14;

this->groupBox1->TabStop = false;

this->groupBox1->Text = L"Настройка структуры и элементов";

//

// groupBox2

//

this->groupBox2->Controls->Add(this->indexTextBox);

this->groupBox2->Controls->Add(this->label3);

this->groupBox2->Controls->Add(this->replaceTextBox);

this->groupBox2->Controls->Add(this->replaceButton);

this->groupBox2->Location = System::Drawing::Point(15, 117);

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

this->groupBox2->Size = System::Drawing::Size(226, 76);

this->groupBox2->TabIndex = 15;

this->groupBox2->TabStop = false;

this->groupBox2->Text = L"Замена элементов";

//

// groupBox3

//

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

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

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

this->groupBox3->Location = System::Drawing::Point(15, 199);

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

this->groupBox3->Size = System::Drawing::Size(226, 100);

this->groupBox3->TabIndex = 16;

this->groupBox3->TabStop = false;

this->groupBox3->Text = L"Дополнительные функции";

//

// groupBox4

//

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

this->groupBox4->Location = System::Drawing::Point(274, 36);

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

this->groupBox4->Size = System::Drawing::Size(267, 263);

this->groupBox4->TabIndex = 17;

this->groupBox4->TabStop = false;

this->groupBox4->Text = L"Элементы структуры";

//

// MainForm

//

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

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

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

this->ClientSize = System::Drawing::Size(553, 315);

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

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

this->Controls->Add(this->groupBox2);

this->Controls->Add(this->groupBox1);

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

this->Name = L"MainForm";

this->Text = L"Лабораторная работа №4 - Работа со структурами данных";

this->groupBox1->ResumeLayout(false);

this->groupBox1->PerformLayout();

this->groupBox2->ResumeLayout(false);

this->groupBox2->PerformLayout();

this->groupBox3->ResumeLayout(false);

this->groupBox3->PerformLayout();

this->groupBox4->ResumeLayout(false);

this->groupBox4->PerformLayout();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

// -- [ Функции ] --

// Функция обновления данных

private: void UpdateOutput() {

std::vector<int> values = dataStructure->GetValues();

inputTextBox->Clear();

outputTextBox->Clear();

for (int i = 0; i < values.size(); ++i) {

outputTextBox->AppendText(" " + i + ":" + values[i] + ";" + "\n");

}

}

// -- [ Кнопки ] --

// Кнопка меню выбора структуры

private: System::Void structureComboBox\_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {

String^ selectedStructure = structureComboBox->SelectedItem->ToString();

delete dataStructure;

if (selectedStructure == "Array") {

dataStructure = new MyArray();

}

else if (selectedStructure == "Vector") {

dataStructure = new MyVector();

}

else if (selectedStructure == "Stack") {

dataStructure = new MyStack();

}

else if (selectedStructure == "Queue") {

dataStructure = new MyQueue();

}

UpdateOutput();

}

// Кнопка добавления элемента

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

int value = Int32::Parse(inputTextBox->Text);

dataStructure->Insert(value);

UpdateOutput();

}

// Кнопка удаления элемента

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

dataStructure->RemoveLast();

UpdateOutput();

}

// Кнопка замены элемента

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

int index = Int32::Parse(indexTextBox->Text);

int newValue = Int32::Parse(replaceTextBox->Text);

dataStructure->Replace(index, newValue);

UpdateOutput();

}

// Кнопка поворота элемента

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

int count = Int32::Parse(rotateTextBox->Text);

dataStructure->Rotate(count);

UpdateOutput();

}

// Кнопка нахождения медианы структуры

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

try {

double median = dataStructure->FindMedian();

MessageBox::Show("Медиана: " + median.ToString());

}

catch (std::exception& e) {

MessageBox::Show(gcnew String(e.what()));

}

}

};

}

# Приложение Б

# Листинг программы, файл MainForm.cpp

#include "MainForm.h"

using namespace::System;

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

[STAThreadAttribute]

int main(array<System::String^>^ args)

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

UniLab4::MainForm form;

Application::Run(% form);

return 0;

}

# Приложение В

# Листинг программы, файл IDataStructure.h

#pragma once

#include <vector>

#include <stack>

#include <queue>

#include <algorithm>

#include <numeric>

#include <stdexcept>

namespace UniLab4 {

// -- [ Функции взаимодействия со структурами ] --

class IDataStructure {

public:

virtual void Insert(int value) = 0;

virtual void RemoveLast() = 0;

virtual void Replace(int index, int newValue) = 0;

virtual void Rotate(int count) = 0;

virtual double FindMedian() = 0;

virtual std::vector<int> GetValues() = 0;

virtual ~IDataStructure() {}

};

// -- [ Массив ] --

class MyArray : public IDataStructure {

std::vector<int> data;

public:

// Добавление элемента

void Insert(int value) override {

data.push\_back(value);

}

// Удаление элемента

void RemoveLast() override {

if (!data.empty()) {

data.pop\_back();

}

}

// Замена элемента

void Replace(int index, int newValue) override {

if (index >= 0 && index < data.size()) {

data[index] = newValue;

}

}

// Поворот элемента

void Rotate(int count) override {

if (data.empty()) return;

count %= data.size();

std::rotate(data.rbegin(), data.rbegin() + count, data.rend());

}

// Медиана массива

double FindMedian() override {

if (data.empty()) throw std::runtime\_error("Структура данных пуста.");

std::vector<int> sortedData = data;

std::sort(sortedData.begin(), sortedData.end());

size\_t n = sortedData.size();

if (n % 2 == 0) {

return (sortedData[n / 2 - 1] + sortedData[n / 2]) / 2.0;

}

else {

return sortedData[n / 2];

}

}

std::vector<int> GetValues() override {

return data;

}

};

// -- [ Вектор ] --

class MyVector : public IDataStructure {

std::vector<int> data;

public:

// Добавление элемента

void Insert(int value) override {

data.push\_back(value);

}

// Удаление элемента

void RemoveLast() override {

if (!data.empty()) {

data.pop\_back();

}

}

// Замена элемента

void Replace(int index, int newValue) override {

if (index >= 0 && index < data.size()) {

data[index] = newValue;

}

}

// Поворот элемента

void Rotate(int count) override {

if (data.empty()) return;

count %= data.size();

std::rotate(data.rbegin(), data.rbegin() + count, data.rend());

}

// Медиана вектора

double FindMedian() override {

if (data.empty()) throw std::runtime\_error("Структура данных пуста.");

std::vector<int> sortedData = data;

std::sort(sortedData.begin(), sortedData.end());

size\_t n = sortedData.size();

if (n % 2 == 0) {

return (sortedData[n / 2 - 1] + sortedData[n / 2]) / 2.0;

}

else {

return sortedData[n / 2];

}

}

std::vector<int> GetValues() override {

return data;

}

};

// -- [ Стэк ] --

class MyStack : public IDataStructure {

std::stack<int> data;

public:

// Добавление элемента

void Insert(int value) override {

data.push(value);

}

// Удаление элемента

void RemoveLast() override {

if (!data.empty()) {

data.pop();

}

}

// Замена элемента

void Replace(int index, int newValue) override {

throw std::runtime\_error("Замена элемента не поддерживается для cтруктуры типа Stack.");

}

// Поворот элемента

void Rotate(int count) override {

throw std::runtime\_error("Поворот не поддерживается для cтруктуры типа Stack.");

}

// Медиана стэка

double FindMedian() override {

throw std::runtime\_error("Нахождение медианы не поддерживается для cтруктуры типа Stack.");

}

std::vector<int> GetValues() override {

std::stack<int> temp = data;

std::vector<int> result;

while (!temp.empty()) {

result.push\_back(temp.top());

temp.pop();

}

std::reverse(result.begin(), result.end());

return result;

}

};

// -- [ Очередь ] --

class MyQueue : public IDataStructure {

std::queue<int> data;

public:

// Добавление элемента

void Insert(int value) override {

data.push(value);

}

// Удаление элемента

void RemoveLast() override {

throw std::runtime\_error("Удаление последнего элемента не поддерживается для cтруктуры типа Queue.");

}

// Замена элемента

void Replace(int index, int newValue) override {

throw std::runtime\_error("Замена элемента не поддерживается для cтруктуры типа Queue.");

}

// Поворот элемента

void Rotate(int count) override {

throw std::runtime\_error("Поворот не поддерживается для cтруктуры типа Queue.");

}

// Медиана очереди

double FindMedian() override {

throw std::runtime\_error("Нахождение медианы не поддерживается для cтруктуры типа Queue.");

}

std::vector<int> GetValues() override {

std::queue<int> temp = data;

std::vector<int> result;

while (!temp.empty()) {

result.push\_back(temp.front());

temp.pop();

}

return result;

}

};

}