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 №2. Сортировки

Отчет по лабораторной работе №2 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 с сортировками целых чисел и проверить скорость их работы на тестовых данных, заполнить таблицу и сделать отчет.

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

Необходимо реализовать программу с сортировками целых чисел и проверить скорость их работы на тестовых данных, заполнить таблицу и сделать отчет.

Реализуйте графический интерфейс для вашей программы, где можно будет в отдельном поле внести неотсортированный массив (без указания количества элементов) или с помощью отдельного поля выбрать массив из приведенных в приложении А. Также будет меню выбора сортировки из приведенных выше. Кнопка для запуска выбранной сортировки с введенным или выбранным массивом. Поле для вывода отсортированного массива. А также поле с информацией о времени, которое заняла эта сортировка (можно в секундах/тиках).

Предусмотрена возможность создания случайного массива с возможностью выбора размера (количества элементов массива)

Предусмотрена возможность сортировки символов (используйте таблицу ASCII).

| Объект программы | Имя этого объекта в программе | Тип данных | Диапазоны представления | Простая/  структура | Вход/  выход/  Константа |
| --- | --- | --- | --- | --- | --- |
| Массив введенных данных | arr | Cli::array<UInt32> | 2500 элементов  0 - 200000000 | Сложная | Вход |
| Отсортированный массив в виде строки | Str\_arr | String^ |  | Сложная | Выход |

**Внешние спецификации данных**

* 1. **Внешние спецификации функций программы**

Программа должна сортировать введенный массив (целочисленный, символьный). Выводить отсортированный массив и время, затраченное на его сортировку, в отдельные поля. Должна быть реализована очистка полей, вывод справочной информации о программе (алгоритм работы с приложением).

* 1. **Математическая формулировка задачи**

Отсортировать введенный массив в порядке возрастания, при помощи выбранного алгоритма сортировки.

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

Ограничение ввода – 200000000.

Для хранения массива значений выбрана структура cli::array<UInt32> из 2500 элементов, при попытке использования std::vector происходит ошибка синтаксиса, член класса управляемый не может относиться к типу класса не управляемый.

* 1. **Спецификации интерфейса**

Для написания графического интерфейса была использована библиотека Windows Forms.

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

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

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

В приложении реализована работа с сортировками массивов, ввод/вывод, очистка полей, кнопка, выводящая справочную информацию о работы программы.

* 1. **Внешние данные тестирования**

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

| Идентификатор | | | T01 |
| --- | --- | --- | --- |
| Заголовок | | | Проверка выхода из приложения |
| Предусловие | | | Приложение не запущено |
| Ожидаемый результат | | | Успешный выход из приложения |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Запуск LAB\_2.exe | Открытие главного окна приложения | Pass |
| 2 | Нажатие на кнопку «Выход» | Выход из приложения | Pass |
| Результат | | | Pass |

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

| Идентификатор | | | T02 |
| --- | --- | --- | --- |
| Заголовок | | | Проверка работоспособности кнопки «Запуск» |
| Предусловие | | | Приложение запущено. |
| Ожидаемый результат | | | Вывод в поле «Вывод» отсортированного массива |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Выбрать тип вводимого массива | Вариант вводимого массива будет выведен в окно массивов | Pass |
| 2 | Ввести массив |  | Pass |
| 3 | Выбрать вариант сортировки | Вариант сортировки будет выведен в окно сортировок | Pass |
| 4 | Нажать кнопку «Запуск» | В поле вывода будет выведен отсортированный массив |  |
| Результат | | | Pass |

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

| Идентификатор | | | T03 |
| --- | --- | --- | --- |
| Заголовок | | | Проверка работоспособности кнопки «Очистить поля» |
| Предусловие | | | Приложение запущено |
| Ожидаемый результат | | | Полям будет присвоено стандартное значение |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Выбрать тип вводимого массива | Вариант вводимого массива будет выведен в окно массивов | Pass |
| 2 | Ввести массив |  | Pass |
| 3 | Выбрать вариант сортировки | Вариант сортировки будет выведен в окно сортировок | Pass |
| 4 | Нажать кнопку «Запуск» | Будет выведен результат сортировки | Pass |
| 5 | Нажать кнопку «Очистить поля» | Заполненным полям присвоены стандартные значения | Pass |
| Результат | | | Pass |

Таблица 4. Тест-кейс №4, выбор сортировки массива

| Идентификатор | | | T04 |
| --- | --- | --- | --- |
| Заголовок | | | Проверка работы дополнительной программы |
| Предусловие | | | Приложение запущено |
| Ожидаемый результат | | | Будет выведен корректный результат |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Выбрать тип вводимого массива | Вариант вводимого массива будет выведен в окно массивов | Pass |
| 2 | Ввести массив |  | Pass |
| 3 | Выбрать вариант сортировки | Будет выбрана сортировка | Pass |
| 4 | Нажать кнопку «Запуск» | Массив будет отсортирован и выведен | Pass |
| 5 | Выбрать другой вариант сортировки | Будет выбрана другая сортировка | Pass |
| 6 | Нажать кнопку «Запуск» | Массив будет отсортирован и выведен | Pass |
| Результат | | | Pass |

Таблица 5. Тест-кейс №5, контроль ввода

| Идентификатор | | | T05 |
| --- | --- | --- | --- |
| Заголовок | | | Проверка контроля ввода |
| Предусловие | | | Приложение запущено, выбран тип целочисленного массива |
| Ожидаемый результат | | | Выведена ошибка, поле ввода очищено |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Ввести некорректное большое значение |  | Pass |
| 2 | Нажать кнопку «ОК» во всплывающем окне | Введенное значение будет уменьшено до максимально допустимого | Pass |
| Результат | | | Pass |

* 1. **Пример работающей программы**

Руководство пользователя и алгоритм работы с программой встроено в приложение.

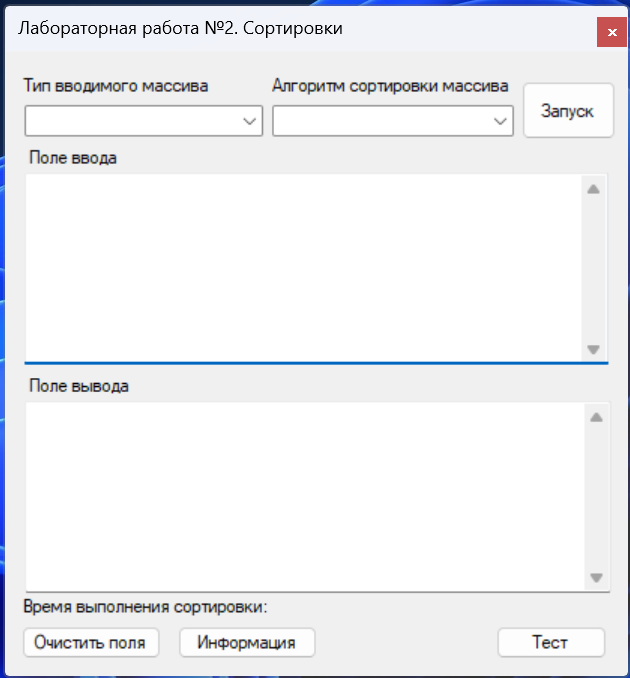


Рисунок 1. Интерфейс приложения

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

Автоматически созданное описание

Рисунок 2. Информационное окно

1. **Проектирование задачи**

Реализовать приложение с графическим интерфейсом, использую библиотеку Windows Forms, для работы с сортировками массивов. 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\_2.exe

* 1. **Уточненное словесное описание задачи**

Необходимо реализовать программу с сортировками целых чисел и проверить скорость их работы на тестовых данных.

Реализуйте графический интерфейс для вашей программы, где можно будет в отдельном поле внести неотсортированный массив (без указания количества элементов) или с помощью отдельного поля выбрать массив из приведенных в приложении А.

Виды сортировки:

1. Поразрядная
2. Вставками
3. Пузырьковая
4. Шейкерная
5. Быстрая

Также будет меню выбора сортировки из приведенных выше. Кнопка для запуска выбранной сортировки с введенным или выбранным массивом. Поле для вывода отсортированного массива. А также поле с информацией о времени, которое заняла эта сортировка (можно в секундах/тиках).

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

При выборе типа вводимого массива происходит переключение контроля входных данных. При выборе варианта сортировки происходит запоминание выбранного варианта. После нажатия кнопки «Запуск» введенный в поле ввода массив, преобразуется из ТД String^ в ТД cli::array<UInt32>^, посредством определенной функции. После преобразования полученный массив передается в функции выбранной сортировки, после чего отсортированный массив передается в функции преобразования массива из ТД cli::array<UInt32> в ТД String^, и выводится на экран.

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

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

Кнопка «Тест» тестирует определенный функции на соответствие с ожидаемым результатом.

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Объект программы | Имя этого объекта в программе | Характеристики | | | Как используется в программе |
| Тип данных | Диапазоны представления | Прост/структура/ константа |
| LAB\_2.h | Count | short int | 0 до 65535 | Простая | Количество введенных элементов |
| LAB\_2.h | start | Time\_t (long long) | −9 223 372 036 854 775 808 до +9 223 372 036 854 775 807 | Простая | Время начала выполнения |
| LAB\_2.h | End | Time\_t (long long) | −9 223 372 036 854 775 808 до +9 223 372 036 854 775 807 | Простая | Время конца выполнения |
| LAB\_2.h | Is\_word | Bool | True / false | Простая | Является ли массив символов словом или нет |
| LAB\_2.h | Chosen\_mode | Short int | –32768 до 32767 | Простая | Хранения индекса выбранного варианта ввода |
| LAB\_2.h | Pr\_mode | Short int | –32768 до 32767 | Простая | Хранения индекса предыдущего варианта ввода |

* 1. **Декомпозиция функций**

Описывается каждая функция и процедура в программе

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назначение | Имя | Параметры | | Процедура/ функция | Тестируется/  не тестируется |
| Вход:  тип | Выход: тип |
| Процедура проверки корректности ввода | correct\_input | - | void | Процедура | Нет |
| Процедура, ограничивающая ввод двойного пробела или двойной запятой, и ввод первым элементом пробела или запятую | incorrect\_input | - | void | Процедура | Нет |
| Процедура ограничения максимально возможного значения | max\_value\_input | - | void | Процедура | Нет |
| Процедура, ограничивающая максимальное количество введенных элементов | max\_quantity | - | void | Процедура | Нет |
| Процедура проверка ввода | input\_check | - | void | Процедура | Нет |
| Функция ввода массива целых чисел | int\_input\_array | - | cli::array<UInt32>^ | Функция | Нет |
| Функция ввода массива символов | char\_input\_array | - | cli::array<UInt32>^ | Функция | Нет |
| Функция шейкерной сортировки | shaker\_sort | cli::array<UInt32>^, int | cli::array<UInt32>^ | Функция | Да |
| Функция сортировки вставками | insertion\_sort | cli::array<UInt32>^, int | cli::array<UInt32>^ | Функция | Да |
| Функция пузырьковой сортировки | bubble\_sort | cli::array<UInt32>^, int | cli::array<UInt32>^ | Функция | Да |
| Процедура быстрой сортировки | quick\_sort | cli::array<UInt32>^, int, int | void | Процедура | Да |
| Функция инициализации быстрой сортировки | quick\_sort\_init | cli::array<UInt32>^ | cli::array<UInt32>^, int | Функция | Нет |
| Функция определения максимального числа в массиве | max\_value | cli::array<UInt32>^, int | unsigned int | Функция | Нет |
| Функция поразрядной сортировки | radix\_sort | cli::array<UInt32>^, int | cli::array<UInt32>^ | Функция | Да |
| Фунция перевода массива целых чисел в строку | int\_array\_to\_string | cli::array<UInt32>^, int | String^ | Функция | Нет |
| Функция перевода массива символов в строку | char\_array\_to\_string | cli::array<UInt32>^, int | String^ | Функция | Нет |
| Процедура создания случайного массива | random\_array | - | void | Процедура | Нет |
| Процедура запуска сортировки | run\_func | - | void | Процедура | Нет |
| Процедура, описывающая действия при нажатии кнопки "Запуск" | run\_btn\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Процедура проверки корректности ввода размера случайного массива | random\_input\_check | - | void | Процедура | Нет |
| Процедура, описывающая действия при изменении входных данных | input\_textBox\_TextChanged | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Процедура, описывающая действия при изменении выбранной позиции в списке "Тип вводимого массива" | comboBox2\_SelectedIndexChanged | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Процедура, описывающая действия при нажатии кнопки "Очистить поля" | clear\_btn\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Процедура, описывающая действия при нажатии кнопки "Информация" | info\_btn\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Функция, проверяющая поле ввода на наполненность/пустоту | empty\_check | - | int | Функция | Нет |
| Функция, предотвращающая двойное очищение полей, при уже пустных полях | double\_clear\_click\_check | - | bool | Функция | Нет |
| Функция, проверяющая, что алгоритм сортировки выбран | not\_choosen\_variant\_check | - | int | Функция | Нет |
| Функция, проверяющая, что тип вводимого массива сортировки выбран | not\_choosen\_mode\_check | - | int | Функция | Нет |
| Функция сравнивающая массивы на равенство | compare\_arrays | cli::array<UInt32>^, cli::array<UInt32>^ | bool | Функция | Нет |
| Функция тестирования поразрядной сортировки | test\_radix\_sort | - | int | Функция | Нет |
| Функция тестирования сортировки вставками | test\_insertion\_sort | - | int | Функция | Нет |
| Функция тестирования пузырьковой сортировки | test\_bubble\_sort | - | int | Функция | Нет |
| Функция тестирования шейкерной сортировки | test\_shaker\_sort | - | int | Функция | Нет |
| Функция тестирования быстрой сортировки | test\_quick\_sort | - | int | Функция | Нет |
| Процедура запуска тестирования | test\_cases | - | void | Процедура | Нет |
| Процедура, описывающая действия при нажатии кнопки "Тест" | button4\_Click | System::Object^ sender, System::EventArgs^ e | System::Void | Процедура | Нет |
| Процедура запуска тестирования при запуске приложения | test\_cases\_constructor | - | void | Процедура | Нет |

* 1. **Блок-схема**

Блок-схемы функций поразрядной сортировки в Приложении A.  
Блок-схемы функций сортировки вставками в Приложении Б.

Блок-схемы функций пузырьковой сортировки в Приложении В.

Блок-схемы функции шейкерной сортировки в Приложении Г.

Блок-схемы функции быстрой сортировки в Приложении Д.

* 1. **Алгоритмизация**

Листинг файла LAB\_2.h в Приложение Е.

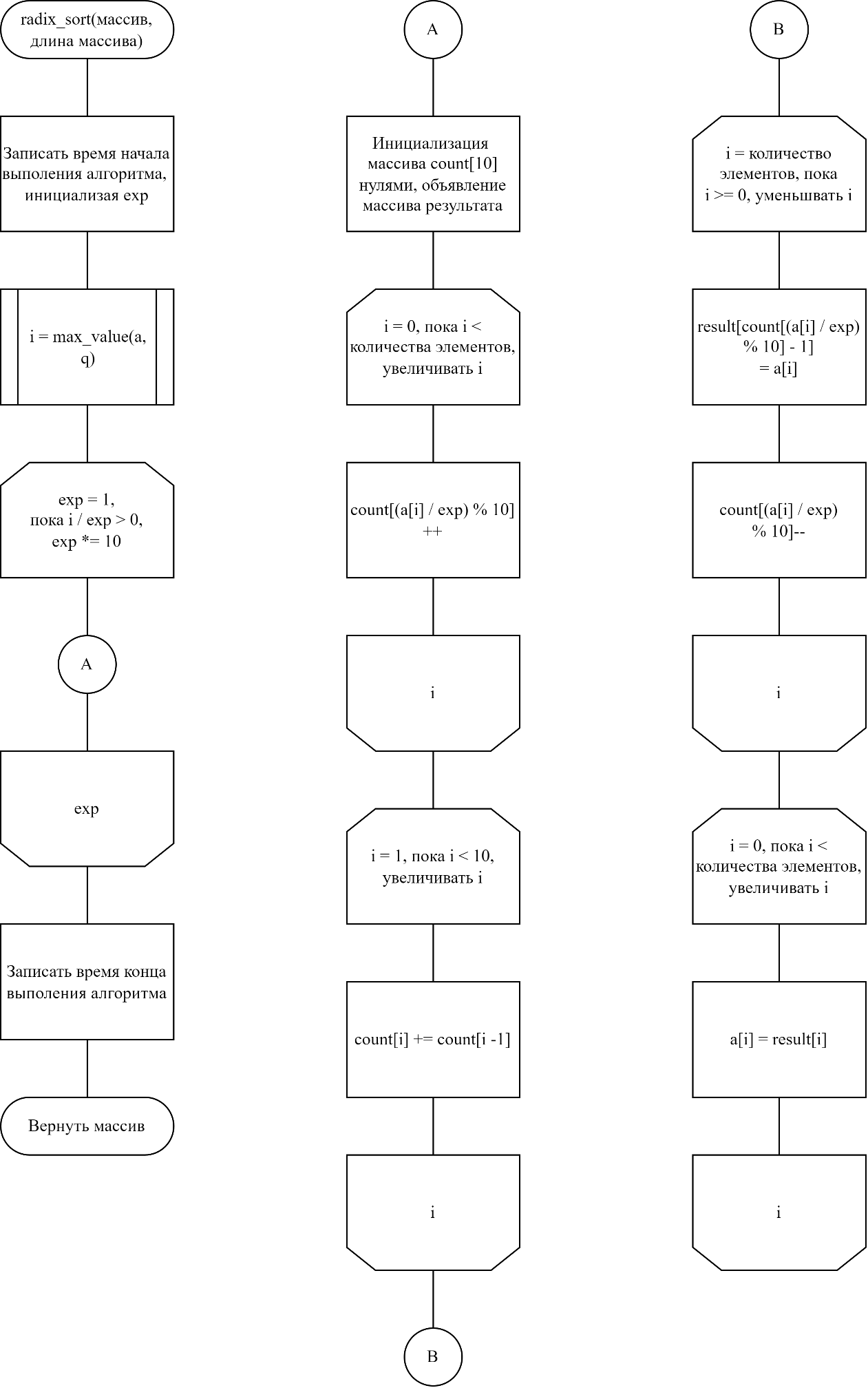
Листинг файла MyForm.cpp в Приложение Ж.

Листинг файла UnitTestLAB\_2.cpp в Приложение И.

Листинг файла Functions.h в Приложение К.

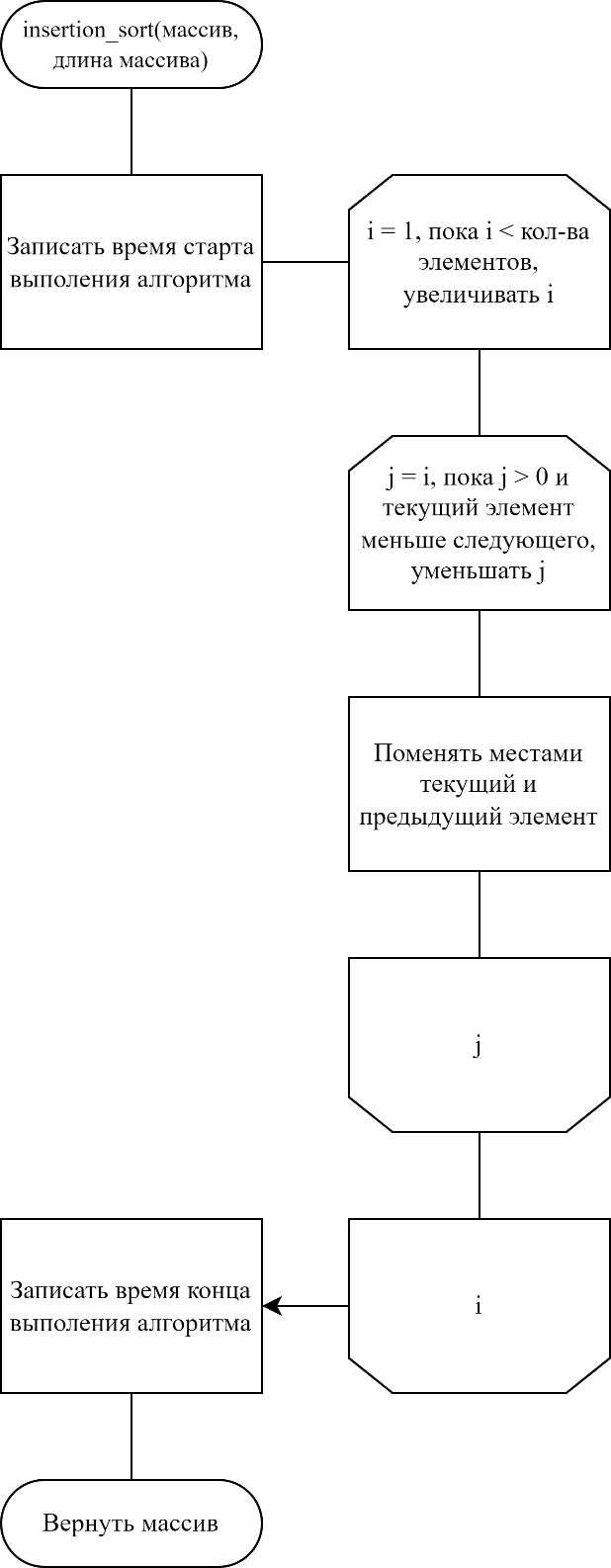
Приложение А

**Блок-схема функции radix\_sort()**



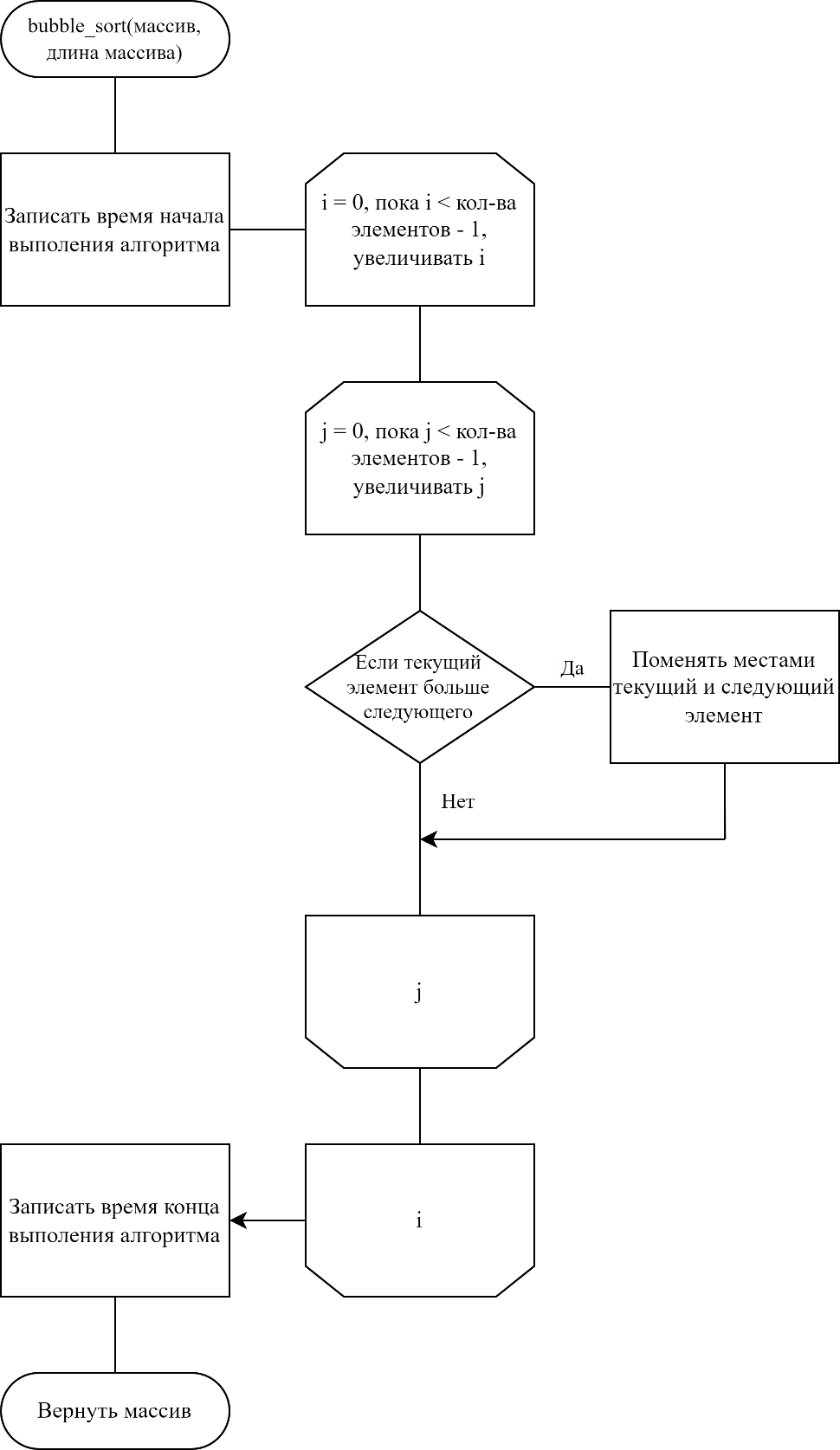
Приложение Б

**Блок-схема функции insertion\_sort()**

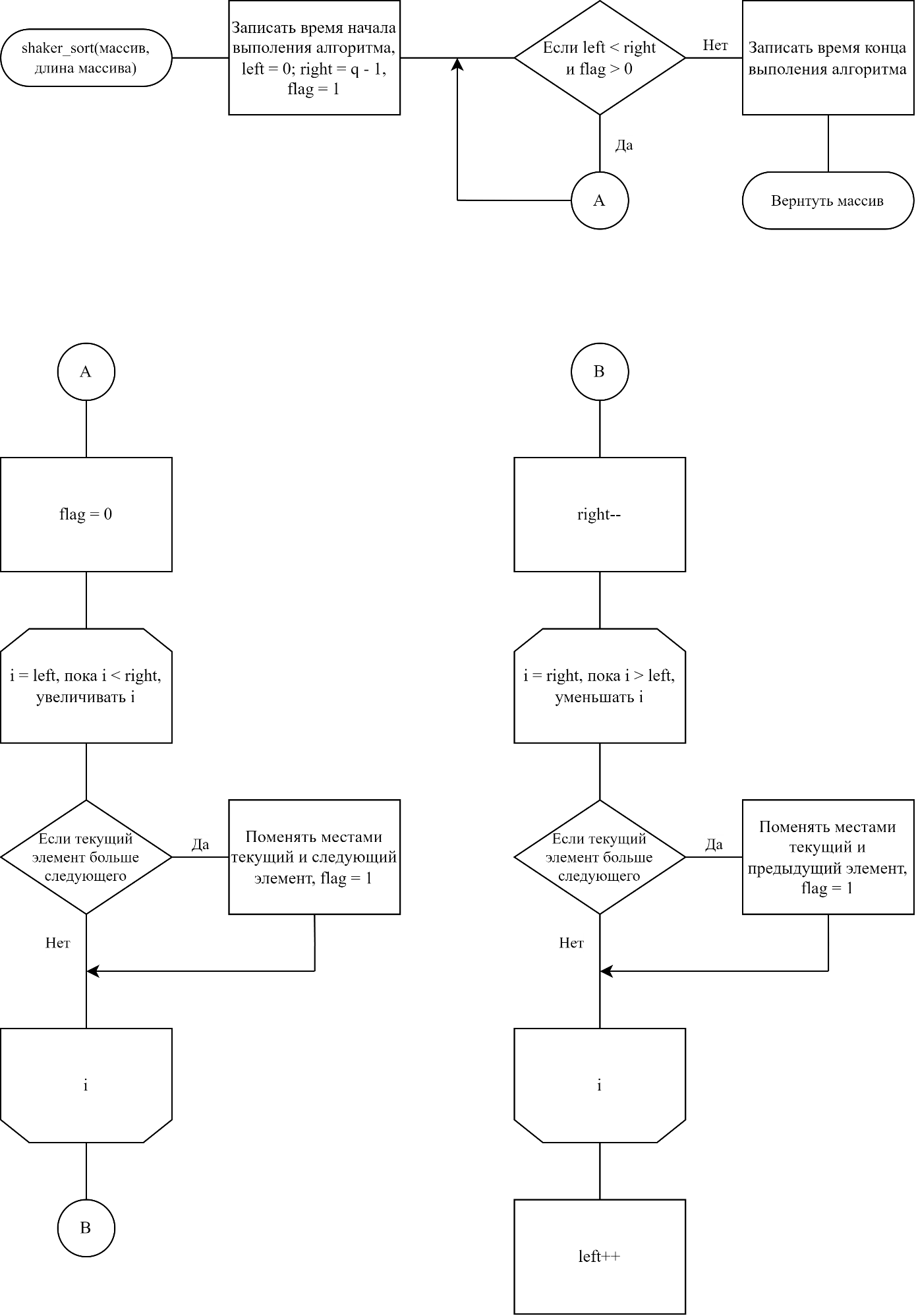


Приложение В

**Блок-схема функции buble\_sort()**

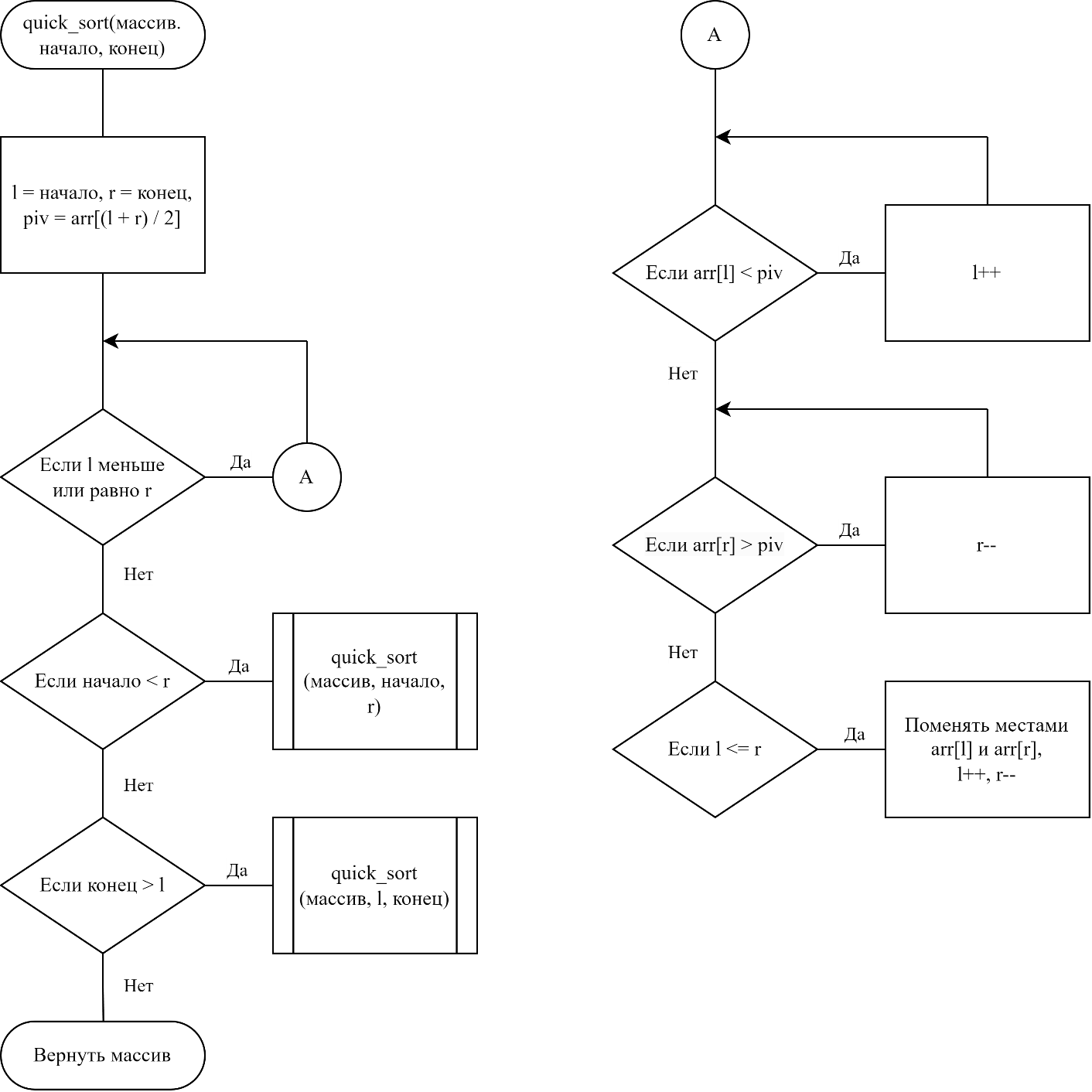


Приложение Г

**Блок-схема функции shaker\_sort()**

Приложение Д

**Блок-схема функции quick\_sort()**



Приложение Е

**Листинг программы, файл LAB\_2.h**

#pragma once

#include <ctime>

#include <stdlib.h>

namespace LAB\_2 {

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 MyForm

/// </summary>

public ref class LAB\_2 : public System::Windows::Forms::Form

{

public:

LAB\_2(void)

{

InitializeComponent();

test\_cases\_constructor();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~LAB\_2()

{

if (components)

{

delete components;

}

}

private: short int count = 0; // Целочисленная переменная, в которой хранится количество элементов в введенном массиве

private: time\_t start, end; // Переменны, в коротых хранится время начала и конца выполнения сортировки

private: bool is\_word = true; // Булевая переменная, которая фиксирует способ ввода массива символов (словом или через запятую)

private: short int chosen\_mode = -1; // Целочисленная переменная, в коротой хранится индекс выбранного варианта ввода массива

private: short int pr\_mode = -1; // Целочисленная переменная, которая хранит номер предыдущего варианта массива

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

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

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

private: System::Windows::Forms::ComboBox^ comboBox1;

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

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

private: System::Windows::Forms::ComboBox^ comboBox2;

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

private: System::Windows::Forms::Button^ button3;

private: System::Windows::Forms::Button^ button4;

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

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

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

private: System::Windows::Forms::Label^ label5;

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->comboBox1 = (gcnew System::Windows::Forms::ComboBox());

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

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

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

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

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

this->comboBox2 = (gcnew System::Windows::Forms::ComboBox());

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

this->button3 = (gcnew System::Windows::Forms::Button());

this->button4 = (gcnew System::Windows::Forms::Button());

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

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

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

this->label5 = (gcnew System::Windows::Forms::Label());

this->SuspendLayout();

//

// comboBox1

//

this->comboBox1->FormattingEnabled = true;

this->comboBox1->Items->AddRange(gcnew cli::array< System::Object^ >(5) {

L"Поразрядная", L"Вставками", L"Пузырьковая", L"Шейкерная",

L"Быстрая"

});

this->comboBox1->Location = System::Drawing::Point(267, 55);

this->comboBox1->Name = L"comboBox1";

this->comboBox1->Size = System::Drawing::Size(239, 28);

this->comboBox1->TabIndex = 3;

//

// input\_textBox

//

this->input\_textBox->Location = System::Drawing::Point(19, 125);

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

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

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

this->input\_textBox->Size = System::Drawing::Size(582, 195);

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

this->input\_textBox->TextChanged += gcnew System::EventHandler(this, &LAB\_2::input\_textBox\_TextChanged);

//

// output\_textBox

//

this->output\_textBox->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 8, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->output\_textBox->Location = System::Drawing::Point(19, 358);

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

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

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

this->output\_textBox->Size = System::Drawing::Size(584, 195);

this->output\_textBox->TabIndex = 1;

//

// button1

//

this->button1->Location = System::Drawing::Point(516, 30);

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

this->button1->Size = System::Drawing::Size(94, 60);

this->button1->TabIndex = 2;

this->button1->Text = L"Запуск";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &LAB\_2::run\_btn\_Click);

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(13, 559);

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

this->label1->Size = System::Drawing::Size(251, 20);

this->label1->TabIndex = 4;

this->label1->Text = L"Время выполнения сортировки:";

//

// time\_label

//

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

this->time\_label->Location = System::Drawing::Point(266, 560);

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

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

this->time\_label->TabIndex = 5;

//

// comboBox2

//

this->comboBox2->FormattingEnabled = true;

this->comboBox2->Items->AddRange(gcnew cli::array< System::Object^ >(6) {

L"Ввод массива целых чисел", L"Ввод массива символов",

L"Тестовый массив №1", L"Тестовый массив №2", L"Тестовый массив №3", L"Создание случайного массива"

});

this->comboBox2->Location = System::Drawing::Point(19, 55);

this->comboBox2->Name = L"comboBox2";

this->comboBox2->Size = System::Drawing::Size(237, 28);

this->comboBox2->TabIndex = 6;

this->comboBox2->SelectedIndexChanged += gcnew System::EventHandler(this, &LAB\_2::comboBox2\_SelectedIndexChanged);

//

// button2

//

this->button2->Location = System::Drawing::Point(17, 589);

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

this->button2->Size = System::Drawing::Size(139, 34);

this->button2->TabIndex = 7;

this->button2->Text = L"Очистить поля";

this->button2->UseVisualStyleBackColor = true;

this->button2->Click += gcnew System::EventHandler(this, &LAB\_2::clear\_btn\_Click);

//

// button3

//

this->button3->Location = System::Drawing::Point(173, 589);

this->button3->Name = L"button3";

this->button3->Size = System::Drawing::Size(139, 34);

this->button3->TabIndex = 8;

this->button3->Text = L"Информация";

this->button3->UseVisualStyleBackColor = true;

this->button3->Click += gcnew System::EventHandler(this, &LAB\_2::info\_btn\_Click);

//

// button4

//

this->button4->Location = System::Drawing::Point(490, 589);

this->button4->Name = L"button4";

this->button4->Size = System::Drawing::Size(111, 34);

this->button4->TabIndex = 9;

this->button4->Text = L"Тест";

this->button4->UseVisualStyleBackColor = true;

this->button4->Click += gcnew System::EventHandler(this, &LAB\_2::button4\_Click);

//

// label2

//

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(263, 25);

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

this->label2->Size = System::Drawing::Size(243, 20);

this->label2->TabIndex = 10;

this->label2->Text = L"Алгоритм сортировки массива";

//

// label3

//

this->label3->AutoSize = true;

this->label3->Location = System::Drawing::Point(13, 25);

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

this->label3->Size = System::Drawing::Size(190, 20);

this->label3->TabIndex = 11;

this->label3->Text = L"Тип вводимого массива";

//

// label4

//

this->label4->AutoSize = true;

this->label4->Location = System::Drawing::Point(19, 98);

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

this->label4->Size = System::Drawing::Size(100, 20);

this->label4->TabIndex = 12;

this->label4->Text = L"Поле ввода";

//

// label5

//

this->label5->AutoSize = true;

this->label5->Location = System::Drawing::Point(19, 332);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(111, 20);

this->label5->TabIndex = 13;

this->label5->Text = L"Поле вывода";

//

// LAB\_2

//

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

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

this->ClientSize = System::Drawing::Size(622, 641);

this->Controls->Add(this->label5);

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

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

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

this->Controls->Add(this->button4);

this->Controls->Add(this->button3);

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

this->Controls->Add(this->comboBox2);

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

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

this->Controls->Add(this->comboBox1);

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

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

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

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

this->Name = L"LAB\_2";

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

this->Text = L"Лабораторная работа №2. Сортировки";

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: void correct\_input()

{

int flag = 0, flag2 = 1;

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

int j = 0;

while (j < data->Length)

{

if (!(data[j] >= '0' && data[j] <= '9') && !(data[j] == ',' || data[j] == ' ') && (chosen\_mode == 0 || chosen\_mode > 1))

{

if (flag == 0) flag = 1;

if (flag && flag2)

{

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

flag2 = 0;

}

this->input\_textBox->Text = this->input\_textBox->Text->Remove(j, 1);

}

j++;

}

}

private: void incorrect\_input()

{

int j = 0;

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

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

{

if (data[0] == ',' || data[0] == ' ')

{

this->input\_textBox->Text = this->input\_textBox->Text->Remove(0, 1);

}

if (data->Contains(",,"))

{

this->input\_textBox->Text = this->input\_textBox->Text->Replace(",,", ",");

}

if (data->Contains(" "))

{

this->input\_textBox->Text = this->input\_textBox->Text->Replace(" ", " ");

}

}

}

private: void max\_value\_input()

{

int j = 0, i = 0;

int max\_value = 200000000;

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

String^ temp;

cli::array<int>^ a = gcnew cli::array<int>(2500);

while (j < data->Length)

{

if (j == data->Length - 1)

{

if (data[j] != ',')

{

temp += data[j];

}

if (System::Convert::ToInt32(temp) > max\_value)

{

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

this->input\_textBox->Text = this->input\_textBox->Text->Remove(j, 1);

}

}

else if (data[j + 1] != ' ' && data[j] != ',')

{

temp += data[j];

}

else

{

if (System::Convert::ToInt32(temp) > max\_value)

{

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

this->input\_textBox->Text = this->input\_textBox->Text->Remove(j, 1);

}

j++;

temp = "";

}

j++;

}

}

private: void max\_quantity()

{

int i = 0;

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

if (!data->Contains(", "))

{

is\_word = true;

if (data->Length > 2500)

{

MessageBox::Show("Достигнуто максимальное количество введенных элементов", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

this->input\_textBox->Text = this->input\_textBox->Text->Remove(data->Length - 1, 1);

}

}

else

{

is\_word = false;

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

{

if (data[j] == ',') i++;

if (i >= 2500)

{

MessageBox::Show("Достигнуто максимальное количество введенных элементов", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

this->input\_textBox->Text = this->input\_textBox->Text->Remove(j, 1);

}

}

}

}

private: void input\_check()

{

if (this->comboBox2->SelectedIndex == -1 && this->input\_textBox->Text != "")

{

MessageBox::Show("Выбирете тип вводимого массива", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

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

}

else if (this->comboBox2->SelectedIndex == 0)

{

incorrect\_input();

max\_quantity();

correct\_input();

max\_value\_input();

}

else if (this->comboBox2->SelectedIndex == 1)

{

max\_quantity();

}

}

private: cli::array<UInt32>^ int\_input\_array()

{

count = 0;

cli::array<UInt32>^ arr = gcnew cli::array<UInt32>(2500);

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

String^ temp;

int j = 0, i = 0;

while (j < data->Length)

{

if (j == data->Length - 1)

{

temp += data[j];

arr[i] = System::Convert::ToInt32(temp);

count++;

return arr;

}

else if (data[j + 1] != ' ' && data[j] != ',')

{

temp += data[j];

}

else

{

arr[i] = System::Convert::ToInt32(temp);

i++;

j++;

temp = "";

count++;

}

j++;

}

}

private: cli::array<UInt32>^ char\_input\_array()

{

count = 0;

cli::array<UInt32>^ arr = gcnew cli::array<UInt32>(2500);

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

int temp = 0;

int j = 0, i = 0;

if (data->Contains(", "))

{

//is\_word = false;

while (j < data->Length)

{

if (j == data->Length - 1)

{

temp = data[j];

arr[i] = temp;

count++;

return arr;

}

else if (data[j + 1] != ' ' && data[j] != ',')

{

temp = System::Convert::ToInt32(data[j]);

}

else

{

arr[i] = temp;

i++;

j++;

count++;

}

j++;

}

}

else

{

count = 0;

//is\_word = true;

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

{

if (!(data[i] == ' '))

{

arr[count] = data[i];

count++;

}

}

return arr;

}

}

private: cli::array<UInt32>^ shaker\_sort(cli::array<UInt32>^ a, int quantity)

{

start = clock();

int left = 0, right = quantity - 1;

int flag = 1;

while ((left < right) && flag > 0)

{

flag = 0;

for (int i = left; i < right; i++)

{

if (a[i] > a[i + 1])

{

unsigned int t = a[i];

a[i] = a[i + 1];

a[i + 1] = t;

flag = 1;

}

}

right--;

for (int i = right; i > left; i--)

{

if (a[i - 1] > a[i])

{

unsigned int t = a[i];

a[i] = a[i - 1];

a[i - 1] = t;

flag = 1;

}

}

left++;

}

end = clock();

return a;

}

private: cli::array<UInt32>^ insertion\_sort(cli::array<UInt32>^ a, int quantity)

{

start = clock();

for (int i = 1; i < quantity; i++)

{

for (int j = i; j > 0 && a[j] < a[j - 1]; j--)

{

unsigned int t = a[j];

a[j] = a[j - 1];

a[j - 1] = t;

}

}

end = clock();

return a;

}

private: cli::array<UInt32>^ bubble\_sort(cli::array<UInt32>^ a, int quantity)

{

start = clock();

for (int i = 0; i < quantity - 1; i++)

{

for (int j = 0; j < quantity - 1; j++)

{

if (a[j] > a[j + 1])

{

unsigned int t = a[j];

a[j] = a[j + 1];

a[j + 1] = t;

}

}

}

end = clock();

return a;

}

private: void quick\_sort(cli::array<UInt32>^ arr, int start, int end)

{

int l = start;

int r = end;

int piv = arr[(l + r) / 2];

while (l <= r)

{

while (arr[l] < piv) l++;

while (arr[r] > piv) r--;

if (l <= r)

{

int t = arr[l];

arr[l] = arr[r];

arr[r] = t;

l++;

r--;

}

}

if (start < r)

quick\_sort(arr, start, r);

if (end > l)

quick\_sort(arr, l, end);

}

private: cli::array<UInt32>^ quick\_sort\_init(cli::array<UInt32>^ a, int quantity)

{

start = clock();

quick\_sort(a, 0, quantity - 1);

end = clock();

return a;

}

private: unsigned int max\_value(cli::array<UInt32>^ a, int quantity)

{

unsigned int max = a[0];

for (int i = 1; i < quantity; i++)

if (a[i] > max)

max = a[i];

return max;

}

private: cli::array<UInt32>^ radix\_sort(cli::array<UInt32>^ a, int quantity)

{

start = clock();

unsigned int exp, i;

i = max\_value (a, quantity);

for (exp = 1; i / exp > 0; exp \*= 10)

{

int i, count[10] = { 0 };

cli::array<int>^ result = gcnew cli::array<int>(quantity);

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

count[(a[i] / exp) % 10]++;

for (i = 1; i < 10; i++)

count[i] += count[i - 1];

for (i = quantity - 1; i >= 0; i--)

{

result[count[(a[i] / exp) % 10] - 1] = a[i];

count[(a[i] / exp) % 10]--;

}

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

a[i] = result[i];

}

end = clock();

return a;

}

private: String^ int\_array\_to\_string(cli::array<UInt32>^ a, int quantity)

{

String^ str\_arr;

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

{

if (i < quantity - 1)

{

str\_arr += System::Convert::ToString(a[i]) + ", ";

}

else

{

str\_arr += System::Convert::ToString(a[i]);

}

}

return str\_arr;

}

private: String^ char\_array\_to\_string(cli::array<UInt32>^ a, int quantity)

{

String^ str\_arr;

if (is\_word)

{

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

{

str\_arr += Convert::ToString(Convert::ToChar(a[i]));

}

}

else

{

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

{

if (i < quantity - 1)

{

str\_arr += Convert::ToString(Convert::ToChar(a[i])) + ", ";

}

else

{

str\_arr += Convert::ToString(Convert::ToChar(a[i]));

}

}

}

return str\_arr;

}

private: void random\_array()

{

if (this->input\_textBox->Text->Contains("Введите размер случайного массива: ") && this->input\_textBox->Text->Contains(";") && chosen\_mode == 5)

{

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

int random\_array\_length = System::Convert::ToInt32((data->Replace("Введите размер случайного массива: ", ""))->Replace(";", ""));

if (random\_array\_length > 2500)

{

String^ temp = System::Convert::ToString(random\_array\_length);

MessageBox::Show("Слишком большое количество элементов. Максимальное количество - 2500", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

this->input\_textBox->Text = this->input\_textBox->Text->Remove(this->input\_textBox->Text->Length - 1, 1);

this->input\_textBox->Text = this->input\_textBox->Text->Replace(temp, "2500");

}

else

{

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

String^ temp = "";

srand(time(NULL));

for (int i = 0; i < random\_array\_length; i++)

{

if (i == random\_array\_length - 1)

temp += System::Convert::ToString(rand());

else

temp += System::Convert::ToString(rand()) + ", ";

}

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

this->comboBox2->SelectedIndex = 0;

}

}

}

private: void run\_func()

{

chosen\_mode = -1;

start = 0, end = 0, count = 0;

{

if (this->comboBox2->SelectedIndex == 0 || (this->comboBox2->SelectedIndex > 1 && this->comboBox2->SelectedIndex < 5))

{

switch (this->comboBox1->SelectedIndex)

{

case 0: this->output\_textBox->Text = int\_array\_to\_string(radix\_sort(int\_input\_array(), count), count); break;

case 1: this->output\_textBox->Text = int\_array\_to\_string(insertion\_sort(int\_input\_array(), count), count); break;

case 2: this->output\_textBox->Text = int\_array\_to\_string(bubble\_sort(int\_input\_array(), count), count); break;

case 3: this->output\_textBox->Text = int\_array\_to\_string(shaker\_sort(int\_input\_array(), count), count); break;

case 4: this->output\_textBox->Text = int\_array\_to\_string(quick\_sort\_init(int\_input\_array(), count), count); break;

}

}

else if (this->comboBox2->SelectedIndex == 1)

{

switch (this->comboBox1->SelectedIndex)

{

case 0: this->output\_textBox->Text = char\_array\_to\_string(radix\_sort(char\_input\_array(), count), count); break;

case 1: this->output\_textBox->Text = char\_array\_to\_string(insertion\_sort(char\_input\_array(), count), count); break;

case 2: this->output\_textBox->Text = char\_array\_to\_string(bubble\_sort(char\_input\_array(), count), count); break;

case 3: this->output\_textBox->Text = char\_array\_to\_string(shaker\_sort(char\_input\_array(), count), count); break;

case 4: this->output\_textBox->Text = char\_array\_to\_string(quick\_sort\_init(char\_input\_array(), count), count); break;

}

}

}

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

{

double sort\_time = (double)(end - start) / CLOCKS\_PER\_SEC;

this->time\_label->Text = System::Convert::ToString(sort\_time) + " секунд";

}

}

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

{

int res = not\_choosen\_mode\_check() + not\_choosen\_variant\_check() + empty\_check();

if (!res)

{

run\_func();

pr\_mode = -1;

}

}

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

{

if (chosen\_mode == 5)

{

random\_array();

}

input\_check();

}

private: System::Void comboBox2\_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e)

{

String^ arr\_1\_1 = "31295, 12351, 21840, 27420, 19503, 27968, 31262, 3202, 20171, 28226, 9432, 10835, 9633, 14889, 11302, 16036, 10301, 14565, 19589, 30478, 25768, 15337, 24300, 8381, 11054, 27359, 25264, 6852, 1892, 26778, 24229, 17075, 19050, 17484, 23942, 3948, 2670, 2809, 2454, 20018, 15217, 3371, 11144, 13827, 17742, 11496, 4252, 17704, 15096, 18401, 7180, 7434, 5891, 4588, 1958, 20551, 8550, 29620, 6074, 22277, 25215, 16815, 17974, 7559, 18263, 24401, 10512, 24830, 23336, 8832, 6116, 20129, 14876, 2887, 21727, 6399, 8536, 9166, 15579, 30612, 16441, 7306, 17099, 17563, 11997, 30897, 14938, 11556, 11750, 24305, 13775, 10086, 4986, 25185, 31540, 9046, 2446, 22456, 21687, 10617, 357, 2749, 16464, 23542, 83, 26622, 17060, 3567, 10695, 28357, 24874, 25873, 10889, 32584, 1932, 14698, 26943, 22816, 23632, 10107, 8246, 24060, 18615, 2191, 17503, 27799, 14309, 17739, 23960, 26599, 20803, 13957, 13357, 16024, 12834, 2138, 30623, 1720, 27297, 21370, 17060, 436, 16848, 24775, 8302, 7299, 1301, 19231, 30512, 16417, 26319, 29557, 31711, 1599, 12346, 31527, 4918, 9785, 19091, 14475, 17404, 11848, 25468, 22987, 7817, 23801, 12449, 10740, 15751, 11997, 5788, 11337, 17398, 29826, 14102, 477, 31858, 1187, 10799, 19809, 32185, 24220, 26400, 2362, 30824, 20768, 8223, 983, 4613, 31022, 5508, 30273, 28994, 11585, 9257, 26867, 24448, 10440, 4886, 23277, 9107, 3029, 25694, 24324, 31173, 793, 11588, 9084, 19741, 11280, 14754, 6104, 5601, 2425, 32396, 9457, 345, 7177, 8240, 11078, 18134, 29536, 3840, 15251, 19955, 19260, 20, 15781, 2587, 2285, 17609, 31084, 12107, 11529, 17839, 13906, 9241, 3424, 9378, 25803, 21750, 27442, 18998, 1581, 19632, 14752, 25177, 18115, 13883, 10337, 3123, 32284, 11184, 18102, 16721, 2473, 30358, 21152, 14339, 13704, 9569, 2128, 17666, 10138, 6712, 1503, 86, 219, 30459, 5951, 12663, 10483, 2693, 32370, 18099, 30728, 9006, 3065, 23503, 13015, 27284, 7056, 5105, 3331, 11932, 18712, 19134, 9644, 15283, 29141, 24767, 12868, 11354, 21977, 9993, 19358, 20961, 28096, 11805, 4424, 6770, 18590, 31757, 5915, 6226, 1590, 28128, 20752, 3684, 16749, 18488, 29014, 9711, 32510, 15154, 27809, 12704, 29806, 23950, 6340, 8659, 15171, 25360, 6530, 3598, 17521, 4645, 4001, 30670, 8213, 15562, 15937, 10887, 28005, 16554, 6764, 23782, 1767, 21126, 30958, 30553, 16098, 6362, 20071, 9169, 16403, 22974, 13726, 2759, 26272, 16394, 12312, 24957, 9554, 31677, 21738, 11654, 19198, 16800, 25035, 24968, 15428, 28198, 25197, 14145, 30076, 20375, 25102, 27336, 26862, 23405, 23370, 15125, 3393, 21247, 23091, 21444, 12343, 7004, 1187, 1046, 24983, 19578, 31953, 26425, 2666, 15248, 21027, 25033, 1129, 1216, 29763, 6294, 17847, 22521, 23669, 17576, 29900, 3356, 24872, 2176, 30897, 17245, 26570, 10913, 22899, 4077, 26849, 22863, 1234, 32726, 31320, 23447, 19257, 28966, 21526, 21391, 10483, 1871, 726, 30175, 12187, 16078, 15666, 23168, 9172, 25611, 3840, 19140, 9019, 29056, 20194, 12439, 26266, 6430, 4633, 2820, 29791, 31980, 4762, 3067, 22348, 23282, 15204, 17233, 11631, 12373, 20240, 31496, 9045, 6083, 23722, 10449, 12155, 26385, 30399, 6351, 8709, 20515, 24759, 13726, 23730, 2317, 9829, 4500, 18192, 16247, 20957, 5446, 21228, 23366, 13190, 30542, 9485, 21625, 28784, 24908, 13858, 8272, 19175, 26155, 32760, 31156, 22856, 11686, 2800, 2259, 7955, 30986, 28593, 11787, 15426, 25917, 8474, ";

String^ arr\_1\_2 = "29705, 9945, 3583, 4730, 27954, 8895, 30320, 3588, 65, 27596, 23250, 20349, 26674, 17955, 20369, 14136, 27488, 26818, 20864, 10564, 3562, 30295, 11110, 27876, 17242, 30805, 21689, 8255, 25481, 4168, 11689, 5209, 26301, 21350, 15457, 9538, 11541, 1126, 20361, 25230, 21799, 8085, 10767, 7172, 26087, 27942, 7207, 9883, 20427, 3529, 11664, 28066, 2348, 13696, 15641, 9278, 18928, 2222, 14693, 11033, 10854, 20337, 22251, 32345, 10609, 29490, 26432, 16335, 24041, 31355, 13721, 12657, 22334, 30472, 3212, 13323, 29022, 6688, 18710, 7729, 8153, 25382, 7906, 32632, 29763, 7275, 29128, 9190, 25563, 6726, 26773, 27558, 4201, 28775, 7434, 19200, 6253, 4456, 3367, 1673, 12917, 3237, 12713, 32663, 6095, 24142, 24017, 9464, 608, 1466, 19559, 23524, 16852, 22014, 32255, 29999, 11256, 13503, 19579, 707, 27747, 15172, 22481, 8526, 30556, 28111, 12302, 2616, 9208, 32046, 24731, 472, 12675, 18061, 16575, 30183, 15393, 1422, 11622, 24629, 25781, 27804, 4129, 16794, 12290, 6012, 22381, 18736, 24849, 3108, 17464, 31679, 21202, 15825, 12449, 28537, 5381, 18004, 13312, 28648, 20571, 18860, 18470, 7716, 13785, 14298, 21478, 24905, 32126, 24817, 16084, 15144, 14373, 3949, 29260, 21821, 30893, 7365, 27574, 23010, 6230, 13380, 17417, 28097, 6766, 14356, 20622, 23951, 29486, 31900, 16615, 21496, 13791, 19980, 24143, 11221, 28224, 29749, 20515, 12965, 10540, 10399, 2466, 28475, 28792, 13013, 17649, 16564, 15899, 7940, 12129, 27052, 10897, 17092, 10997, 23474, 23169, 1078, 18719, 22264, 28573, 5967, 25291, 12879, 201, 9587, 29086, 24403, 23936, 4934, 3712, 32483, 8308, 22719, 14680, 6613, 21877, 8062, 19990, 20967, 13972, 7007, 19191, 7928, 17544, 20815, 19460, 27768, 15953, 16478, 24541, 11037, 15387, 8470, 6580, 10967, 29028, 12043, 1180, 20743, 23137, 12468, 23556, 32492, 22686, 12665, 11646, 15453, 742, 8717, 18374, 31422, 11362, 21246, 6911, 1069, 26372, 32139, 28223, 27388, 1319, 20149, 3662, 3428, 24175, 10393, 22955, 375, 23543, 3580, 19920, 32087, 4011, 27688, 31159, 4006, 23282, 14071, 31133, 15421, 26769, 2861, 24582, 20769, 32204, 19210, 12289, 1970, 26317, 26912, 19629, 17958, 586, 21859, 508, 5430, 12387, 16511, 2743, 7703, 10441, 10606, 14275, 17580, 31781, 26953, 7330, 12794, 17592, 12638, 25916, 25308, 19006, 12164, 14560, 3832, 7449, 19627, 20102, 31729, 27819, 14775, 5719, 4319, 7428, 16015, 7279, 15868, 24509, 32624, 2273, 3231, 21539, 11400, 24827, 27503, 29337, 20416, 28108, 2768, 18231, 11481, 27620, 23970, 16311, 26554, 26715, 4651, 4242, 25864, 28437, 5969, 30054, 6392, 28227, 525, 20635, 14433, 29485, 26396, 28230, 22, 9264, 31992, 28237, 1650, 7170, 6662, 6605, 19252, 2622, 25361, 10168, 31117, 15644, 15863, 13507, 4216, 30847, 23714, 29311, 25930, 15926, 29754, 20933, 13450, 27515, 29849, 11352, 13570, 20416, 21504, 12742, 24854, 8364, 17160, 7055, 27221, 28512, 23771, 27785, 2894, 766, 10147, 16259, 11173, 15468, 31043, 26921, 2662, 26453, 5190, 28968, 25240, 7281, 4372, 9874, 15894, 6407, 26476, 21379, 27290, 12208, 11278, 21509, 17691, 13917, 22200, 5169, 20313, 5531, 9300, 14743, 17334, 23830, 26992, 31712, 6983, 348, 26862, 28850, 14891, 9956, 765, 20180, 13851, 7759, 20552, 27795, 27459, 30189, 32109, 32100, 21328, 17539, 31408, 21194, 21195, 4428, 21489, 9338, 28837, 32061, 17627, 9150, 11283, 4493, 23903, 24087, 25756, 22389, 19040, 13094, 2159, 7477, 5808, 25036, 31995, 3159, 30780, 28817, 30138, 30007, 32131, 30018, 11597, 29635, 12109, 32734, 13518, 20864, 31385, 11444, 19810, 18638, 30423, 8645, 13, ";

String^ arr\_1\_3 = "3436, 23639, 31599, 26571, 15934, 16130, 27725, 25052, 9294, 24904, 5209, 6239, 25362, 13145, 21331, 6946, 32237, 21144, 6759, 31417, 22263, 7500, 266, 6605, 2817, 11929, 8485, 10423, 20180, 22192, 32373, 32589, 12730, 25410, 32049, 18684, 26944, 19514, 11246, 14104, 2617, 6315, 25793, 11631, 21254, 4103, 2467, 7460, 27413, 2963, 24601, 443, 15272, 31767, 20500, 13424, 17880, 30682, 2581, 2509, 6617, 25805, 22678, 28027, 21375, 4871, 8358, 28761, 16385, 23409, 21971, 31947, 4350, 5827, 25932, 9557, 16573, 175, 23623, 16744, 3001, 23387, 19873, 19579, 9053, 26389, 15044, 15198, 32268, 579, 974, 15307, 26724, 1432, 8944, 4671, 15300, 12430, 14099, 2567, 24075, 14864, 8642, 11166, 28323, 22902, 15428, 11574, 17599, 23479, 26632, 20885, 8928, 6362, 32327, 10530, 17310, 23982, 25316, 26944, 9125, 7611, 16023, 19930, 11518, 376, 9290, 14655, 10128, 9407, 9667, 1585, 4838, 6525, 7044, 16440, 19762, 686, 9199, 13885, 11154, 23433, 7598, 6995, 26515, 16119, 8530, 10712, 1524, 17226, 15930, 31933, 6828, 15916, 32439, 4898, 3629, 1237, 30573, 12838, 18411, 31874, 26109, 6397, 25112, 3874, 8929, 15171, 19122, 22610, 27287, 30095, 30144, 7819, 16897, 28748, 1241, 27802, 31244, 9412, 2021, 24401, 5776, 14462, 15144, 31613, 7287, 8906, 26236, 8216, 2850, 20634, 16861, 26815, 1288, 14995, 811, 5222, 27552, 8548, 31020, 17482, 29314, 25687, 22481, 10023, 28105, 9765, 6586, 25771, 24507, 13327, 6907, 23851, 2476, 25386, 20769, 4502, 1055, 22523, 17520, 18343, 8751, 27715, 11887, 15966, 9261, 10012, 24999, 1712, 32139, 31557, 24231, 21630, 12792, 16315, 14923, 23492, 3533, 5769, 14654, 3603, 15784, 13792, 17598, 19521, 2390, 18282, 7327, 24671, 13815, 20956, 27283, 21483, 21340, 24779, 6824, 30279, 30900, 24256, 7442, 10090, 13416, 21902, 29995, 15062, 13123, 2275, 14776, 1737, 25302, 7966, 27246, 29624, 4891, 19552, 16053, 8185, 23384, 13286, 20907, 27584, 17990, 29590, 20332, 332, 2419, 3543, 1937, 11618, 19429, 9638, 2294, 7163, 16384, 16615, 10945, 19770, 5290, 1272, 9887, 22176, 428, 8588, 10321, 17539, 29531, 23991, 30312, 12220, 6521, 31557, 6220, 23457, 23894, 25680, 7555, 28723, 826, 22290, 27184, 9368, 31495, 1015, 11482, 28107, 9243, 20982, 24943, 1426, 2164, 15650, 1417, 23984, 21271, 892, 25071, 21340, 1458, 5132, 3423, 14067, 12672, 29744, 3211, 31722, 14189, 3891, 13153, 9478, 21371, 9249, 11151, 16591, 30564, 17882, 17479, 2626, 14465, 8550, 28479, 454, 21738, 32392, 3837, 4705, 5087, 11926, 31313, 5932, 1382, 23943, 22819, 21764, 25910, 31054, 15982, 15491, 9428, 9063, 9101, 2094, 24828, 24751, 10422, 2800, 17986, 31608, 16068, 17096, 10489, 19421, 7737, 958, 21588, 23123, 12163, 20251, 21707, 4225, 18558, 10969, 16832, 27730, 25403, 3417, 19043, 26952, 12800, 12628, 2515, 5139, 13455, 26132, 26246, 31842, 12971, 12260, 234, 30969, 16345, 19653, 30028, 29187, 11364, 29026, 29946, 9642, 1955, 21062, 31396, 6886, 6014, 19789, 7688, 13970, 26927, 18264, 19151, 9925, 24390, 4677, 24241, 6239, 24816, 28447, 29408, 27362, 3704, 145, 21668, 22148, 7950, 16801, 28994, 14466, 19400, 21659, 14627, 31158, 27622, 15946, 15922, 17260, 9818, 24942, 4486, 30322, 23688, 20075, 9281, 6254, 12874, 15167, 12216, 11124, 22327, 14262, 26836, 9176, 29989, 29153, 696, 28737, 26604, 7876, 17571, 18431, 8813, 23375, 14104, 13215, 25811, 6544, 11325, 956, 10265, 28198, 10114, 10776, 25293, 20985, 26754, 10475, 2835, 11304, 7470, 3194, 31212, 14403, 32552, 8940, 5048, 7355, 24685, 26201, 23034, 17814, 27447, 19749, 19036, 22375, 6434, 29836, 12019, 20094, ";

String^ arr\_1\_4 = "9613, 22909, 17347, 7772, 32579, 5036, 21894, 25962, 32091, 18147, 15785, 2282, 23388, 16024, 24472, 19757, 16549, 26565, 2875, 15878, 22284, 7412, 26816, 31616, 13510, 30873, 20892, 16076, 7559, 8613, 15052, 11133, 24297, 18008, 5217, 16632, 19474, 14679, 16955, 37, 7313, 25854, 4629, 6507, 12938, 2295, 21186, 14044, 3091, 30220, 28853, 13996, 25565, 15417, 15303, 27112, 17687, 32518, 14901, 11108, 5775, 25620, 11496, 25334, 24859, 20469, 10490, 18657, 21815, 25920, 27253, 32148, 21487, 31205, 17502, 18837, 32512, 30399, 20978, 6584, 5178, 21864, 19759, 2964, 6184, 12369, 11926, 7735, 5107, 6418, 31280, 23321, 27268, 10494, 24015, 2895, 2281, 32763, 1174, 19713, 30202, 9617, 17339, 7454, 1340, 64, 1164, 10574, 12549, 22295, 4016, 32297, 24545, 4482, 7600, 26022, 6453, 10596, 20662, 2518, 27938, 869, 31636, 12242, 5036, 8590, 23047, 7922, 1820, 25619, 7741, 2111, 7586, 19567, 30614, 555, 26175, 30390, 30376, 27776, 19682, 15655, 29317, 20631, 18898, 14849, 21312, 4554, 8267, 7065, 26029, 555, 28780, 32424, 7273, 12151, 8273, 17129, 25114, 31813, 24700, 3556, 24824, 22915, 20345, 28676, 25485, 13347, 193, 24560, 12984, 27410, 27360, 24, 5368, 24938, 211, 25230, 21223, 232, 31412, 16152, 25861, 16544, 4505, 20355, 11959, 1007, 12985, 17978, 27280, 26034, 28112, 4887, 11214, 17333, 8901, 27256, 6440, 7791, 12938, 17046, 17607, 8428, 14149, 15122, 17810, 25866, 1965, 8913, 30126, 18408, 2486, 4106, 31301, 25244, 6256, 16029, 30855, 31524, 300, 10407, 7151, 415, 23522, 3411, 20936, 28094, 17103, 23027, 24047, 4213, 11618, 10575, 11482, 11766, 29273, 19772, 1753, 29663, 13608, 5745, 2204, 8114, 17050, 20119, 1270, 27015, 8378, 6303, 13280, 1789, 22004, 11096, 21185, 23158, 11128, 6721, 6637, 8836, 28383, 8348, 11115, 15774, 30481, 8899, 24315, 22040, 28774, 22975, 3927, 2642, 13323, 23831, 19642, 25903, 11944, 32486, 6303, 15048, 6719, 11824, 3787, 15714, 3928, 21366, 8077, 24222, 12635, 23432, 16197, 20720, 31293, 25755, 12067, 15698, 26974, 16795, 17201, 314, 6578, 18371, 18836, 2249, 800, 6762, 30479, 6469, 27431, 26912, 31944, 8137, 18631, 21497, 11087, 22587, 8095, 27177, 18140, 24703, 5715, 30417, 28888, 3050, 19504, 4121, 19103, 2810, 12819, 7295, 13701, 15172, 27270, 17007, 21527, 13229, 29997, 30227, 12451, 24797, 10593, 6508, 11514, 17905, 17434, 25250, 8052, 13680, 3346, 31896, 16693, 5761, 22311, 5572, 20911, 2651, 24577, 20798, 4968, 1532, 21238, 19990, 1533, 13342, 29482, 14571, 360, 13159, 12539, 7097, 5758, 1248, 11932, 24563, 22715, 16040, 30719, 14034, 30018, 22250, 29177, 23532, 15136, 6924, 17569, 26120, 30929, 10025, 17657, 6705, 21253, 29122, 8656, 29961, 30884, 16304, 22502, 24160, 22543, 5213, 12626, 15109, 14761, 29236, 21537, 20700, 25494, 32598, 9280, 12593, 10946, 25231, 31900, 7500, 5413, 27784, 903, 6470, 28998, 24894, 7774, 8475, 1790, 30130, 18536, 13311, 17791, 21987, 20738, 10742, 4756, 18169, 17657, 1951, 7979, 14787, 1848, 17488, 25872, 24931, 23715, 13928, 12795, 8776, 28482, 6625, 25549, 27997, 3234, 25893, 14871, 23424, 9080, 24611, 24796, 7478, 17544, 12705, 6296, 4765, 10949, 22228, 8064, 8098, 3770, 14638, 16396, 4211, 18728, 22135, 10419, 8760, 2289, 12063, 28917, 1022, 8244, 28371, 28575, 25587, 11450, 846, 17995, 23769, 8081, 5490, 17958, 22453, 26614, 6295, 4598, 13767, 17785, 25635, 8539, 22207, 1371, 2906, 2388, 20270, 12091, 26488, 11782, 25403, 25841, 2045, 16825, 6551, 31163, 32073, 13604, 31411, 1175, 12648, 31978, 20528, 29167, 17199, 20580, 30871, 22651, 6024, 17825, 9034, ";

String^ arr\_1\_5 = "5476, 5386, 28331, 2234, 29605, 15821, 23564, 29403, 25271, 7036, 10385, 10709, 109, 12715, 29262, 6397, 31261, 27243, 25826, 16387, 2304, 12572, 14589, 17516, 3547, 12163, 19743, 14147, 4063, 24861, 15266, 25356, 8910, 28850, 4602, 13818, 30816, 5164, 3388, 7266, 8421, 23210, 32733, 24430, 16171, 8401, 17596, 23574, 13218, 8034, 25443, 9804, 702, 18450, 18826, 27767, 31389, 25750, 15837, 12067, 23530, 11261, 28672, 8439, 2561, 16962, 18895, 17127, 1620, 22050, 26090, 4267, 15117, 20779, 20337, 20847, 22773, 19990, 28428, 24780, 19604, 18774, 18735, 1015, 3738, 16369, 29084, 7992, 21373, 12431, 22569, 24478, 24164, 13008, 21187, 11506, 31556, 6371, 2227, 30697, 25672, 20309, 8992, 14493, 5751, 28454, 26325, 12606, 22929, 13858, 6981, 18693, 32569, 22894, 25292, 4970, 7513, 26210, 26680, 22423, 18371, 9622, 1416, 11292, 3348, 13376, 18420, 31460, 15070, 18365, 29997, 360, 11526, 20333, 30181, 8482, 11370, 26820, 18834, 5189, 8391, 22942, 26479, 23211, 6619, 1235, 12484, 29455, 29638, 3340, 22632, 13436, 6838, 6613, 25476, 23557, 12739, 29427, 10137, 8887, 8472, 26103, 27077, 28620, 20308, 10771, 834, 31521, 2608, 13869, 12079, 29675, 25185, 15666, 26003, 32677, 32448, 17581, 26527, 9549, 10920, 2456, 24883, 30927, 9119, 10097, 17316, 6850, 27193, 13713, 15805, 24658, 6714, 20154, 29955, 14205, 3579, 14063, 25661, 7043, 20751, 5000, 820, 11671, 9765, 27013, 7096, 17883, 9264, 22051, 28643, 4235, 11974, 13741, 5538, 2936, 19910, 11413, 11120, 30649, 10302, 18115, 19045, 5021, 9868, 8493, 22896, 12516, 32707, 20904, 9697, 31534, 13904, 253, 25898, 691, 18397, 27456, 998, 24373, 5237, 14941, 26138, 32563, 16617, 28404, 31931, 30560, 6259, 19303, 21617, 26195, 12698, 30001, 411, 16512, 23125, 16505, 3475, 6247, 3955, 23014, 18739, 14091, 826, 17078, 10392, 6310, 30543, 21258, 3109, 8603, 28118, 13956, 6859, 8108, 30169, 5908, 4488, 27501, 5870, 27151, 11310, 15864, 8212, 4994, 2918, 3811, 24031, 21445, 15348, 3695, 31701, 13139, 17213, 4098, 5399, 24792, 21766, 30936, 14404, 13708, 5758, 1781, 25242, 21440, 4735, 1118, 13918, 5046, 17169, 14984, 28798, 12982, 21552, 14865, 24950, 30814, 975, 19118, 28016, 17978, 17180, 5989, 22447, 17813, 32006, 27388, 23853, 10126, 9724, 4273, 25219, 11118, 30968, 11563, 14602, 24376, 1469, 9137, 16192, 20098, 29789, 23450, 24685, 4996, 8701, 31236, 14214, 32367, 31452, 25545, 22865, 18176, 13714, 7087, 2506, 31828, 21398, 20703, 24774, 9982, 2819, 12697, 8662, 20823, 32156, 22053, 19310, 27157, 24845, 22541, 8518, 13599, 31860, 27017, 13825, 16032, 7967, 315, 14349, 1114, 29847, 28188, 1482, 18750, 23006, 26551, 19694, 18233, 6134, 28309, 14632, 13136, 10146, 4344, 30211, 18415, 25201, 26572, 2062, 18814, 17655, 24538, 16698, 14646, 9822, 9023, 21635, 10803, 2334, 14108, 19666, 27850, 26888, 8592, 27426, 1052, 10889, 11235, 12258, 6635, 3885, 5380, 5019, 25773, 21346, 18486, 13011, 13279, 22911, 27654, 28977, 5788, 1540, 23936, 22498, 20590";

String^ arr\_2\_1 = "19, 15, 16, 20, 20, 13, 12, 12, 10, 20, 17, 20, 19, 16, 13, 19, 13, 14, 13, 19, 17, 13, 17, 20, 14, 13, 19, 16, 18, 16, 20, 20, 13, 16, 18, 13, 17, 15, 10, 20, 19, 11, 16, 10, 10, 17, 16, 18, 11, 11, 13, 19, 14, 20, 11, 14, 11, 14, 16, 12, 14, 15, 12, 19, 14, 18, 14, 16, 15, 16, 13, 11, 15, 19, 14, 19, 13, 20, 18, 19, 16, 12, 17, 13, 15, 16, 19, 13, 13, 18, 19, 10, 10, 14, 16, 16, 18, 13, 10, 11, 16, 14, 20, 10, 16, 18, 11, 13, 20, 14, 20, 17, 16, 10, 12, 19, 19, 18, 15, 13, 11, 15, 15, 15, 15, 16, 18, 16, 13, 17, 10, 13, 20, 10, 15, 16, 17, 20, 12, 19, 16, 16, 18, 11, 16, 16, 15, 12, 20, 20, 19, 11, 15, 20, 15, 10, 18, 18, 15, 10, 11, 17, 18, 18, 12, 17, 17, 13, 19, 18, 15, 15, 15, 12, 14, 20, 20, 10, 13, 15, 14, 14, 17, 13, 19, 11, 16, 15, 11, 19, 19, 12, 18, 12, 18, 19, 10, 12, 12, 20, 14, 19, 13, 18, 18, 16, 11, 10, 16, 20, 10, 10, 20, 13, 14, 18, 16, 20, 20, 11, 11, 11, 14, 18, 17, 10, 13, 16, 14, 16, 17, 15, 14, 14, 17, 10, 15, 11, 12, 15, 12, 12, 16, 13, 15, 14, 19, 14, 17, 10, 11, 20, 15, 14, 20, 11, 19, 17, 16, 14, 12, 10, 18, 18, 18, 16, 18, 10, 19, 17, 19, 13, 12, 10, 11, 10, 16, 18, 14, 11, 19, 10, 11, 18, 19, 16, 10, 10, 20, 17, 11, 16, 10, 16, 19, 19, 11, 14, 14, 12, 14, 15, 13, 17, 20, 19, 14, 19, 12, 19, 19, 12, 16, 14, 15, 17, 16, 11, 15, 11, 16, 10, 20, 18, 11, 10, 16, 20, 19, 16, 18, 15, 20, 20, 16, 11, 20, 18, 19, 16, 15, 15, 14, 17, 20, 13, 11, 14, 10, 17, 10, 14, 10, 20, 15, 12, 18, 10, 16, 13, 18, 10, 17, 16, 15, 11, 19, 18, 10, 10, 11, 19, 11, 15, 18, 10, 19, 20, 16, 14, 18, 12, 20, 17, 11, 14, 18, 20, 14, 11, 11, 13, 18, 10, 12, 17, 10, 11, 12, 12, 15, 13, 13, 20, 11, 16, 14, 15, 10, 11, 18, 17, 14, 14, 11, 20, 16, 12, 11, 15, 17, 16, 17, 12, 11, 11, 12, 14, 11, 15, 19, 10, 11, 12, 11, 18, 20, 13, 17, 16, 17, 17, 10, 19, 10, 16, 20, 15, 19, 19, 14, 19, 18, 17, 17, 16, 13, 15, 15, 14, 12, 12, 16, 18, 20, 16, 17, 15, 10, 19, 20, 19, 20, 13, 13, 11, 19, 11, 20, 14, 19, 20, 11, 19, 14, 14, 16, 12, 11, 10, 20, 14, 13, 14, 16, 15, 12, 16, 15, 14, 12, 18, 17, 19, 14, 14, 18, 10, 15, 14, 17, 10, 15, 17, 20, 18, 14, 18, 17, 16, 10, 18, 15, 10, 14, 19, 17, 19, 12, 14, 17, 18, 15, 18, 10, 17, 17, 13, 13, 13, 15, 20, 14, 19, 13, 12, 12, 13, 10, 12, 13, 17, 15, 16, 17, 15, 19, 14, 19, 18, 17, 12, 18, 20, 11, 10, 15, 19, 20, 12, 12, 15, 17, 12, 18, 13, 16, 17, 17, 11, 15, 18, 18, 12, 16, 17, 16, 16, 19, 13, 20, 14, 15, 10, 18, 18, 13, 11, 12, 15, 10, 11, 12, 15, 11, 16, 10, 20, 12, 18, 15, 13, 12, 13, 11, 14, 20, 19, 15, 15, 14, 12, 10, 18, 18, 15, 10, 15, 19, 18, 15, 15, 18, 20, 15, 13, 13, 17, 16, 15, 19, 13, 15, 14, 11, 15, 19, 12, 13, 11, 16, 14, 14, 11, 18, 13, 16, 11, 16, 11, 16, 11, 14, 11, 13, 11, 17, 17, 16, 18, 18, 17, 10, 17, 19, 17, 10, 13, 10, 18, 17, 19, 10, 16, 18, 13, 20, 10, 17, 16, 18, 11, 16, 19, 14, 16, 18, 15, 20, 18, 19, 13, 18, 16, 18, 13, 11, 11, 12, 17, 18, 14, 10, 18, 19, 13, 11, 15, 13, 16, 16, 10, 16, 20, 10, 17, 14, 11, 20, 13, 20, 18, 10, 10, 20, 20, 11, 15, 12, 11, 20, 14, 13, 16, 10, 19, 13, 12, 20, 15, 19, 15, 16, 17, 12, 20, 17, 20, 15, 11, 14, 18, 17, 11, 12, 13, 15, 18, 13, 11, 10, 17, 11, 12, 17, 16, 10, 16, 13, 18, 10, 20, 10, 14, 15, 16, 15, 18, 10, 11, 20, 13, 14, 10, 18, 20, 10, 16, 16, 15, 19, 13, 15, 14, 12, 13, 19, 10, 12, 15, 10, 19, 14, 13, 17, 19, 10, 13, 18, 11, 11, 12, 15, 11, 18, 19, 11, 19, 19, 18, 14, 12, 17, 16, 10, 18, 17, 14, 10, 16, 12, 10, 12, 19, 13, 18, 15, 17, 15, 17, 16, 17, 12, 13, 16, 13, 12, 11, 14, 12, 18, 19, 12, 12, 16, 18, 16, 11, 15, 13, 12, 10, 17, 12, 13, 19, 13, 16, 18, 18, 19, 11, 14, 15, 15, 10, 19, 15, 10, 14, 13, 20, 17, 16, 20, 10, 19, 11, 19, 14, 13, 13, 13, 12, 19, 12, 17, 18, 19, 15, 15, 12, 12, 12, 18, 14, 17, 15, 14, 19, 15, 11, 15, 16, 20, 18, 17, 12, 18, 20, 15, 12, 17, 20, 11, 10, 17, 18, 16, 16, 20, 17, 20, 10, 18, 20, 18, 16, 17, 17, 12, 11, 14, 15, 18, 10, 10, 18, 16, 12, 12, 15, 20, 15, ";

String^ arr\_2\_2 = "18, 13, 19, 13, 18, 17, 17, 14, 18, 13, 19, 14, 20, 16, 12, 20, 13, 15, 10, 15, 12, 19, 13, 11, 18, 20, 17, 20, 18, 15, 11, 12, 15, 15, 11, 15, 16, 16, 12, 17, 16, 14, 15, 15, 17, 17, 10, 20, 11, 18, 18, 10, 11, 10, 13, 10, 20, 14, 17, 11, 16, 10, 12, 12, 14, 14, 19, 16, 13, 16, 15, 13, 20, 19, 16, 19, 16, 12, 18, 19, 14, 17, 14, 10, 12, 19, 18, 17, 18, 14, 18, 18, 20, 16, 12, 20, 10, 18, 20, 20, 16, 11, 16, 10, 18, 11, 12, 17, 10, 18, 17, 11, 19, 11, 15, 20, 13, 10, 16, 15, 17, 13, 19, 17, 15, 17, 12, 12, 12, 10, 11, 10, 11, 13, 11, 13, 18, 12, 12, 18, 13, 17, 19, 12, 14, 20, 20, 11, 15, 20, 20, 16, 12, 14, 18, 11, 16, 17, 10, 19, 19, 18, 10, 13, 13, 20, 10, 17, 14, 10, 16, 17, 20, 10, 16, 10, 13, 16, 15, 11, 12, 11, 16, 15, 12, 19, 14, 11, 16, 18, 19, 19, 13, 11, 17, 13, 18, 20, 18, 18, 19, 11, 19, 10, 18, 13, 12, 18, 19, 16, 17, 16, 12, 17, 16, 15, 17, 13, 15, 11, 13, 20, 13, 10, 11, 13, 15, 17, 16, 19, 11, 15, 10, 12, 13, 18, 19, 12, 15, 18, 16, 10, 15, 11, 20, 16, 17, 16, 20, 18, 20, 15, 14, 11, 14, 18, 18, 12, 10, 12, 10, 12, 15, 17, 16, 17, 12, 19, 14, 17, 10, 15, 13, 13, 13, 18, 19, 19, 19, 19, 15, 15, 14, 16, 12, 10, 10, 20, 13, 17, 13, 20, 12, 10, 19, 12, 18, 17, 19, 11, 19, 12, 14, 13, 10, 10, 20, 12, 19, 10, 20, 16, 20, 13, 17, 10, 19, 12, 20, 11, 20, 11, 17, 16, 10, 18, 14, 13, 17, 14, 13, 20, 19, 15, 11, 10, 20, 14, 18, 17, 18, 20, 18, 13, 12, 10, 15, 15, 20, 10, 20, 15, 17, 14, 18, 14, 11, 13, 20, 11, 10, 19, 19, 13, 13, 16, 20, 10, 17, 20, 14, 12, 20, 17, 20, 13, 14, 11, 15, 17, 11, 18, 19, 17, 14, 15, 11, 14, 19, 15, 17, 18, 16, 16, 17, 16, 13, 18, 10, 13, 17, 10, 16, 19, 16, 18, 13, 12, 19, 12, 11, 14, 16, 15, 12, 11, 18, 18, 11, 19, 20, 10, 15, 17, 10, 14, 19, 11, 13, 10, 10, 17, 19, 20, 14, 10, 16, 11, 15, 16, 13, 16, 13, 18, 13, 11, 11, 17, 18, 10, 13, 12, 16, 14, 14, 11, 19, 18, 14, 16, 17, 14, 12, 12, 15, 12, 17, 11, 16, 19, 13, 13, 14, 13, 14, 16, 20, 16, 17, 19, 17, 14, 12, 18, 19, 20, 11, 11, 12, 10, 15, 12, 13, 13, 12, 17, 18, 14, 16, 16, 18, 16, 12, 10, 10, 14, 10, 11, 17, 13, 20, 19, 18, 14, 11, 10, 11, 20, 17, 18, 20, 12, 10, 10, 11, 16, 12, 16, 12, 18, 17, 20, 14, 11, 10, 20, 18, 18, 18, 16, 19, 20, 13, 13, 19, 16, 14, 17, 12, 13, 19, 11, 16, 15, 12, 16, 14, 13, 17, 19, 18, 19, 12, 19, 20, 18, 10, 17, 17, 19, 19, 13, 12, 16, 15, 13, 18, 13, 16, 14, 18, 11, 15, 18, 15, 15, 19, 18, 15, 10, 18, 10, 12, 12, 16, 12, 11, 19, 11, 18, 16, 16, 20, 15, 12, 15, 20, 10, 16, 10, 19, 10, 11, 10, 13, 18, 19, 11, 19, 15, 11, 16, 20, 14, 19, 12, 10, 18, 11, 18, 15, 10, 14, 19, 15, 19, 11, 18, 11, 20, 17, 16, 14, 15, 14, 16, 12, 10, 10, 17, 12, 10, 20, 10, 12, 18, 10, 12, 20, 14, 17, 12, 20, 11, 11, 12, 18, 10, 10, 11, 16, 20, 15, 20, 17, 17, 13, 12, 18, 19, 19, 19, 18, 13, 13, 20, 20, 15, 15, 12, 14, 13, 16, 12, 16, 12, 12, 13, 10, 16, 15, 13, 19, 18, 18, 18, 17, 19, 14, 13, 19, 10, 11, 15, 13, 16, 12, 11, 13, 14, 20, 12, 17, 14, 11, 15, 12, 19, 11, 16, 19, 12, 20, 17, 10, 18, 16, 20, 19, 10, 18, 19, 18, 13, 11, 10, 18, 12, 16, 10, 15, 12, 16, 19, 20, 12, 14, 14, 18, 15, 17, 14, 14, 10, 16, 19, 19, 18, 20, 16, 11, 15, 14, 19, 15, 18, 11, 10, 18, 17, 18, 19, 13, 20, 17, 20, 14, 18, 11, 15, 10, 11, 17, 16, 19, 19, 11, 20, 14, 10, 14, 12, 18, 17, 20, 12, 13, 10, 18, 16, 11, 13, 20, 18, 12, 14, 18, 18, 10, 17, 19, 20, 10, 12, 14, 11, 16, 17, 12, 10, 13, 18, 15, 20, 19, 20, 18, 14, 11, 15, 18, 20, 16, 10, 10, 18, 10, 20, 15, 18, 11, 14, 20, 19, 13, 19, 14, 18, 17, 19, 11, 20, 16, 17, 12, 17, 14, 20, 10, 20, 12, 10, 13, 20, 18, 13, 19, 17, 10, 15, 19, 10, 15, 14, 12, 10, 18, 18, 10, 13, 14, 14, 20, 15, 15, 19, 14, 11, 15, 13, 14, 13, 12, 11, 16, 14, 13, 19, 16, 11, 20, 16, 20, 16, 13, 17, 12, 13, 12, 11, 15, 20, 19, 20, 11, 12, 14, 11, 15, 12, 10, 18, 16, 15, 14, 20, 19, 19, 20, 19, 13, 17, 20, 13, 14, 11, 15, 16, 15, 12, 12, 11, 15, 12, 10, 16, 14, 18, 15, 10, 20, 13, 12, 11, 17, 17, 15, 11, 10, 10, 10, 10, 20, 17, 18, 14, 16, 17, 10, 14, 12, 13, 14, 10, 15, 11, 11, 12, 19, 11, ";

String^ arr\_2\_3 = "17, 15, 13, 19, 17, 14, 16, 16, 16, 20, 19, 19, 15, 16, 20, 13, 14, 11, 13, 12, 19, 20, 10, 19, 17, 19, 11, 17, 12, 12, 12, 16, 15, 10, 19, 18, 16, 15, 12, 10, 13, 15, 16, 19, 16, 18, 15, 16, 17, 15, 19, 18, 19, 13, 16, 20, 12, 14, 18, 17, 13, 12, 12, 10, 18, 10, 14, 15, 18, 17, 18, 15, 19, 19, 14, 18, 18, 16, 16, 15, 14, 19, 12, 20, 19, 16, 20, 15, 11, 11, 17, 18, 12, 13, 12, 13, 17, 12, 16, 17, 20, 10, 15, 14, 17, 14, 20, 18, 18, 17, 20, 17, 15, 15, 10, 12, 18, 10, 14, 14, 15, 14, 19, 16, 10, 13, 10, 11, 11, 20, 18, 10, 20, 19, 11, 11, 19, 16, 10, 11, 18, 13, 19, 18, 17, 20, 10, 12, 12, 11, 15, 14, 11, 12, 19, 16, 17, 11, 17, 19, 14, 12, 20, 12, 17, 16, 12, 15, 19, 19, 16, 17, 12, 11, 16, 10, 16, 15, 20, 20, 12, 13, 15, 18, 13, 13, 10, 16, 20, 19, 14, 10, 11, 16, 10, 17, 18, 18, 12, 17, 17, 20, 18, 16, 15, 18, 18, 10, 12, 20, 11, 17, 12, 20, 19, 11, 18, 13, 14, 10, 17, 18, 16, 16, 15, 12, 14, 15, 10, 20, 19, 12, 18, 13, 10, 13, 11, 19, 10, 17, 19, 20, 18, 12, 12, 19, 12, 15, 14, 16, 20, 16, 19, 11, 20, 18, 11, 19, 13, 20, 12, 18, 20, 19, 20, 10, 10, 17, 10, 14, 18, 19, 16, 15, 16, 15, 17, 17, 10, 19, 12, 20, 17, 11, 11, 12, 12, 14, 14, 20, 17, 19, 19, 10, 10, 11, 16, 18, 18, 18, 19, 11, 18, 16, 15, 11, 11, 19, 15, 12, 19, 13, 11, 18, 15, 20, 13, 17, 18, 10, 17, 14, 12, 13, 16, 17, 12, 12, 17, 15, 10, 12, 18, 11, 19, 12, 18, 12, 18, 10, 20, 13, 16, 14, 18, 10, 19, 20, 11, 20, 16, 15, 10, 17, 11, 14, 10, 15, 12, 13, 15, 13, 14, 18, 18, 16, 14, 15, 14, 15, 14, 12, 18, 12, 20, 18, 12, 16, 17, 20, 13, 17, 14, 20, 12, 20, 18, 11, 18, 18, 19, 13, 18, 15, 12, 16, 12, 17, 16, 17, 12, 16, 18, 17, 16, 19, 14, 18, 19, 12, 10, 14, 15, 17, 20, 18, 10, 10, 17, 16, 14, 13, 12, 12, 16, 11, 14, 13, 12, 16, 13, 19, 16, 13, 13, 11, 19, 14, 11, 17, 18, 12, 14, 20, 12, 18, 13, 11, 10, 11, 17, 20, 12, 18, 14, 16, 10, 17, 17, 18, 10, 18, 20, 20, 13, 14, 20, 17, 10, 12, 19, 17, 14, 17, 13, 16, 12, 19, 13, 17, 10, 13, 16, 11, 19, 11, 12, 20, 20, 15, 14, 11, 16, 19, 10, 11, 12, 20, 20, 19, 16, 17, 14, 16, 17, 15, 15, 20, 10, 20, 17, 16, 17, 20, 19, 13, 18, 10, 11, 17, 12, 12, 19, 11, 19, 12, 15, 18, 12, 10, 14, 13, 10, 12, 20, 14, 20, 15, 10, 15, 12, 13, 19, 11, 17, 11";

String^ arr\_3\_1 = "2358, 4552, 4993, 7671, 27107, 13734, 12151, 10149, 22378, 25503, 32546, 29733, 26278, 25074, 24916, 4689, 10527, 472, 4334, 1492, 14043, 21005, 7508, 11735, 2679, 23181, 18206, 32478, 25462, 17819, 5393, 5100, 8920, 6664, 13645, 23278, 21998, 27314, 32455, 24662, 25454, 8011, 9978, 32263, 3779, 14940, 32730, 29041, 17830, 16521, 21002, 7421, 27634, 9494, 10062, 30891, 6580, 24490, 6467, 30201, 32642, 3923, 30798, 23304, 25871, 20171, 24274, 22778, 30204, 7674, 1190, 24990, 20720, 1211, 26310, 79, 3306, 21219, 5960, 14873, 10454, 8576, 18834, 14622, 6495, 31070, 28363, 18232, 26885, 17104, 30351, 23520, 7791, 25260, 2023, 16719, 23810, 10392, 18040, 5346, 29618, 29316, 30714, 9923, 19992, 10616, 7660, 31942, 31685, 31632, 22406, 31889, 16855, 30533, 8176, 256, 27338, 27629, 15537, 24264, 18576, 13189, 28778, 7258, 7395, 18863, 17860, 24842, 23514, 21495, 1703, 24432, 31930, 32602, 13229, 16142, 19727, 22281, 4054, 18740, 31228, 30826, 5245, 23841, 17360, 18528, 24970, 28523, 4057, 3532, 32234, 6305, 19581, 5460, 31804, 30510, 23430, 16995, 22477, 26752, 30398, 12656, 7623, 23914, 17084, 20356, 13989, 19911, 1021, 2806, 25578, 3937, 10807, 377, 17939, 5137, 9705, 20314, 32458, 21287, 15415, 25602, 17244, 25099, 19701, 32706, 29549, 37, 16130, 16080, 2027, 6651, 1751, 21900, 28224, 1945, 31583, 25096, 9164, 2614, 27914, 30408, 9618, 19732, 13085, 17094, 2097, 28072, 9036, 31417, 19479, 20219, 10122, 26519, 16881, 15440, 26144, 15961, 4965, 7943, 12284, 9470, 17412, 19393, 10572, 2258, 15994, 10949, 24652, 10736, 16904, 16291, 18012, 6, 17461, 21527, 16214, 23640, 6848, 18991, 15134, 27912, 472, 22960, 3451, 1428, 21135, 15543, 30757, 20640, 32141, 3996, 6943, 20214, 7681, 26651, 16198, 18379, 14558, 26550, 12075, 3652, 31811, 19797, 4638, 16440, 4922, 18992, 32139, 25144, 24998, 7662, 12426, 31948, 15479, 15886, 11697, 7936, 29216, 3207, 10028, 27167, 14155, 25003, 28944, 2751, 19403, 27985, 28141, 17149, 27602, 13107, 10659, 27146, 24932, 9877, 5750, 1255, 18955, 10536, 8292, 23660, 25805, 28973, 2741, 13455, 31835, 6877, 19413, 2275, 16779, 26889, 23391, 29279, 10760, 30974, 9850, 6753, 10056, 28075, 7784, 24306, 18879, 22022, 5468, 3148, 9299, 21163, 23948, 20378, 1039, 16070, 24200, 18174, 20507, 24628, 5322, 21208, 9708, 25317, 12174, 3142, 1464, 9093, 16612, 1255, 9005, 3419, 13315, 13285, 11874, 3953, 32224, 13362, 10510, 27893, 15359, 14353, 7160, 27871, 13705, 21977, 16558, 11476, 29344, 4597, 29562, 6986, 24047, 18992, 402, 23790, 9349, 26160, 10127, 8514, 18403, 28991, 21536, 10891, 5715, 338, 768, 3306, 32124, 3907, 17445, 26825, 27444, 9314, 17211, 9336, 12437, 300, 25424, 26904, 6995, 8538, 17105, 26297, 14668, 16670, 15285, 9151, 31329, 18345, 13304, 23434, 9349, 15088, 21067, 12374, 19987, 8368, 27402, 13407, 21093, 28847, 14702, 3657, 29858, 31814, 14596, 15552, 9431, 28444, 13149, 15708, 6986, 19764, 22023, 6855, 15883, 4650, 25789, 9761, 1227, 21945, 23515, 24483, 20209, 6529, 5909, 21123, 24555, 9223, 20583, 26648, 30146, 3326, 2895, 30271, 11762, 24261, 16443, 3325, 11384, 28976, 23613, 18278, 14636, 24971, 26886, 9149, 18257, 19420, 28946, 9948, 7019, 10601, 29728, 20182, 848, 30842, 32741, 6364, 21634, 4448, 16641, 7773, 9404, 19638, 8242, 3441, 28875, 24295, 16328, 120, 9631, 3313, 27993, 30974, 16290, 7784, 23477, 483, 29769, 12477, 14620, 15544, 6632, 501, 502, 503, 504, 505, 506, 507, ";

String^ arr\_3\_2 = "508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1130, 1131, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1150, 1151, 1152, 1153, 1154, 1155, 1156, 1157, 1158, 1159, 1160, 1161, 1162, 1163, 1164, 1165, 1166, 1167, 1168, 1169, 1170, 1171, 1172, 1173, 1174, 1175, 1176, 1177, 1178, 1179, 1180, 1181, 1182, 1183, 1184, 1185, 1186, 1187, 1188, 1189, 1190, 1191, 1192, 1193, 1194, 1195, 1196, 1197, 1198, 1199, 1200, 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, ";

String^ arr\_3\_3 = "1209, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1217, 1218, 1219, 1220, 1221, 1222, 1223, 1224, 1225, 1226, 1227, 1228, 1229, 1230, 1231, 1232, 1233, 1234, 1235, 1236, 1237, 1238, 1239, 1240, 1241, 1242, 1243, 1244, 1245, 1246, 1247, 1248, 1249, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1259, 1260, 1261, 1262, 1263, 1264, 1265, 1266, 1267, 1268, 1269, 1270, 1271, 1272, 1273, 1274, 1275, 1276, 1277, 1278, 1279, 1280, 1281, 1282, 1283, 1284, 1285, 1286, 1287, 1288, 1289, 1290, 1291, 1292, 1293, 1294, 1295, 1296, 1297, 1298, 1299, 1300, 1301, 1302, 1303, 1304, 1305, 1306, 1307, 1308, 1309, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317, 1318, 1319, 1320, 1321, 1322, 1323, 1324, 1325, 1326, 1327, 1328, 1329, 1330, 1331, 1332, 1333, 1334, 1335, 1336, 1337, 1338, 1339, 1340, 1341, 1342, 1343, 1344, 1345, 1346, 1347, 1348, 1349, 1350, 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358, 1359, 1360, 1361, 1362, 1363, 1364, 1365, 1366, 1367, 1368, 1369, 1370, 1371, 1372, 1373, 1374, 1375, 1376, 1377, 1378, 1379, 1380, 1381, 1382, 1383, 1384, 1385, 1386, 1387, 1388, 1389, 1390, 1391, 1392, 1393, 1394, 1395, 1396, 1397, 1398, 1399, 1400, 1401, 1402, 1403, 1404, 1405, 1406, 1407, 1408, 1409, 1410, 1411, 1412, 1413, 1414, 1415, 1416, 1417, 1418, 1419, 1420, 1421, 1422, 1423, 1424, 1425, 1426, 1427, 1428, 1429, 1430, 1431, 1432, 1433, 1434, 1435, 1436, 1437, 1438, 1439, 1440, 1441, 1442, 1443, 1444, 1445, 1446, 1447, 1448, 1449, 1450, 1451, 1452, 1453, 1454, 1455, 1456, 1457, 1458, 1459, 1460, 1461, 1462, 1463, 1464, 1465, 1466, 1467, 1468, 1469, 1470, 1471, 1472, 1473, 1474, 1475, 1476, 1477, 1478, 1479, 1480, 1481, 1482, 1483, 1484, 1485, 1486, 1487, 1488, 1489, 1490, 1491, 1492, 1493, 1494, 1495, 1496, 1497, 1498, 1499, 1500, 1501, 1502, 1503, 1504, 1505, 1506, 1507, 1508, 1509, 1510, 1511, 1512, 1513, 1514, 1515, 1516, 1517, 1518, 1519, 1520, 1521, 1522, 1523, 1524, 1525, 1526, 1527, 1528, 1529, 1530, 1531, 1532, 1533, 1534, 1535, 1536, 1537, 1538, 1539, 1540, 1541, 1542, 1543, 1544, 1545, 1546, 1547, 1548, 1549, 1550, 1551, 1552, 1553, 1554, 1555, 1556, 1557, 1558, 1559, 1560, 1561, 1562, 1563, 1564, 1565, 1566, 1567, 1568, 1569, 1570, 1571, 1572, 1573, 1574, 1575, 1576, 1577, 1578, 1579, 1580, 1581, 1582, 1583, 1584, 1585, 1586, 1587, 1588, 1589, 1590, 1591, 1592, 1593, 1594, 1595, 1596, 1597, 1598, 1599, 1600, 1601, 1602, 1603, 1604, 1605, 1606, 1607, 1608, 1609, 1610, 1611, 1612, 1613, 1614, 1615, 1616, 1617, 1618, 1619, 1620, 1621, 1622, 1623, 1624, 1625, 1626, 1627, 1628, 1629, 1630, 1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1639, 1640, 1641, 1642, 1643, 1644, 1645, 1646, 1647, 1648, 1649, 1650, 1651, 1652, 1653, 1654, 1655, 1656, 1657, 1658, 1659, 1660, 1661, 1662, 1663, 1664, 1665, 1666, 1667, 1668, 1669, 1670, 1671, 1672, 1673, 1674, 1675, 1676, 1677, 1678, 1679, 1680, 1681, 1682, 1683, 1684, 1685, 1686, 1687, 1688, 1689, 1690, 1691, 1692, 1693, 1694, 1695, 1696, 1697, 1698, 1699, 1700, 1701, 1702, 1703, 1704, 1705, 1706, 1707, 1708, 1709, 1710, 1711, 1712, 1713, 1714, 1715, 1716, 1717, 1718, 1719, 1720, 1721, 1722, 1723, 1724, 1725, 1726, 1727, 1728, 1729, 1730, 1731, 1732, 1733, 1734, 1735, 1736, 1737, 1738, 1739, 1740, 1741, 1742, 1743, 1744, 1745, 1746, 1747, 1748, 1749, 1750, 1751, 1752, 1753, 1754, 1755, 1756, 1757, 1758, 1759, 1760, 1761, 1762, 1763, 1764, 1765, 1766, 1767, 1768, 1769, 1770, 1771, 1772, 1773, 1774, 1775, 1776, 1777, 1778, 1779, 1780, 1781, 1782, 1783, 1784, 1785, 1786, 1787, 1788, 1789, 1790, 1791, 1792, 1793, ";

String^ arr\_3\_4 = "1794, 1795, 1796, 1797, 1798, 1799, 1800, 1801, 1802, 1803, 1804, 1805, 1806, 1807, 1808, 1809, 1810, 1811, 1812, 1813, 1814, 1815, 1816, 1817, 1818, 1819, 1820, 1821, 1822, 1823, 1824, 1825, 1826, 1827, 1828, 1829, 1830, 1831, 1832, 1833, 1834, 1835, 1836, 1837, 1838, 1839, 1840, 1841, 1842, 1843, 1844, 1845, 1846, 1847, 1848, 1849, 1850, 1851, 1852, 1853, 1854, 1855, 1856, 1857, 1858, 1859, 1860, 1861, 1862, 1863, 1864, 1865, 1866, 1867, 1868, 1869, 1870, 1871, 1872, 1873, 1874, 1875, 1876, 1877, 1878, 1879, 1880, 1881, 1882, 1883, 1884, 1885, 1886, 1887, 1888, 1889, 1890, 1891, 1892, 1893, 1894, 1895, 1896, 1897, 1898, 1899, 1900, 1901, 1902, 1903, 1904, 1905, 1906, 1907, 1908, 1909, 1910, 1911, 1912, 1913, 1914, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922, 1923, 1924, 1925, 1926, 1927, 1928, 1929, 1930, 1931, 1932, 1933, 1934, 1935, 1936, 1937, 1938, 1939, 1940, 1941, 1942, 1943, 1944, 1945, 1946, 1947, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959, 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 20416, 30127, 24171, 16842, 30564, 20005, 24359, 14465, 25282, 28351, 24950, 24699, 19553, 17022, 5289, 6870, 949, 17043, 5753, 12917, 31717, 23063, 6126, 32084, 14290, 12796, 18816, 10264, 14164, 31044, 21117, 24555, 18886, 32112, 22442, 20102, 27078, 10044, 21805, 7311, 1590, 12169, 20394, 28198, 23686, 8347, 8365, 3515, 18486, 30596, 25160, 14626, 11158, 31200, 15869, 17641, 21969, 5059, 8925, 2537, 2601, 20976, 9970, 20763, 3204, 19532, 6107, 12717, 22312, 5266, 32438, 5461, 24003, 32071, 6979, 22246, 430, 19697, 6725, 12065, 7026, 12676, 2860, 25549, 15756, 2077, 22770, 27914, 5229, 19698, 19011, 1250, 1989, 18313, 19680, 21793, 257, 18217, 4165, 14182, 31506, 911, 13194, 15227, 16241, 30177, 2695, 26594, 29792, 15848, 15931, 26800, 24690, 26489, 19567, 8351, 1869, 9268, 10902, 20574, 943, 19826, 4088, 18475, 17071, 17016, 11023, 22436, 1094, 30449, 17969, 28634, 17677, 17819, 8048, 11176, 30024, 23360, 30655, 12867, 8484, 19944, 7510, 10257, 28221, 13389, 30039, 8639, 4707, 20879, 18609, 19854, 10142, 10537, 22596, 18157, 367, 3876, 31815, 18509, 26457, 2186, 21223, 12081, 22688, 4126, 16945, 1668, 15697, 4938, 20786, 2864, 10370, 15193, 7261, 24300, 4059, 22246, 26925, 23893, 7254, 30612, 7435, 18241, 30786, 6144, 7025, 14416, 16014, 24052, 20304, 6852, 28484, 19678, 25519, 3474, 3412, 31872, 28565, 28856, 18835, 10216, 6442, 3026, 24323, 10403, 29721, 744, 1109, 12846, 7992, 3467, 5804, 25005, 25963, 9248, 24996, 30175, 26054, 6011, 10387, 17234, 1777, 18096, 21775, 14547, 14801, 26405, 28336, 2089, 20324, 4844, 16406, 18179, 26093, 4432, 19758, 7214, 12626, 3502, 32434, 19723, 8898, 12423, 9967, 16193, 13210, 20178, 17355, 17071, 22410, 9111, 8710, 14245, 23028, 18361, 17856, 25939, 10900, 23572, 15420, 5249, 23910, 6599, 15075, 17025, 25028, 15571, 12172, 98, 13519, 31974, 9722, 21095, 4881, 2161, 9894, 9143, 28001, 13824, 29375, 8790, 9478, 24523, 16752, 5972, 21279, 7560, 25185, 3909, 8002, 20612, 27008, 26353, 26979, 13810, 18433, 4172, 19191, 29571, 30371, 9128, 13146, 21047, 5430, 4393, 9070, 7026, 9050, 1657, 19590, 19538, 13961, 23742, 5762, 4928, 20824, 8905, 20795, 19397, 16377, 6032, 5409, 27807, 7685, 10463, 5922, 7189, 27000, 19018, 29646, 24199, 18174, 14438, 13753, 4937, 23405, 31801, ";

String^ arr\_3\_5 = "6945, 13234, 24210, 17134, 24273, 2159, 19248, 24013, 2117, 18717, 20550, 32363, 1807, 23064, 26513, 798, 7546, 7309, 13650, 26548, 24172, 1239, 5457, 11877, 17041, 27400, 31010, 21765, 6325, 18885, 31655, 10879, 21874, 24408, 19926, 6270, 1548, 25856, 24764, 13677, 27289, 20619, 28866, 12962, 31589, 21543, 22064, 31446, 23956, 4214, 16111, 5337, 2825, 11488, 13715, 7811, 12679, 18989, 6553, 29678, 3350, 851, 27884, 22242, 16809, 28812, 6830, 10897, 30717, 24748, 6966, 6021, 16836, 15538, 6485, 11222, 2307, 18845, 31966, 19360, 3769, 15489, 32357, 1460, 15553, 21721, 10310, 8334, 20217, 28959, 26858, 29576, 6052, 20890, 6725, 24653, 25357, 32120, 8867, 143, 15982, 32284, 4773, 23549, 9743, 25785, 25734, 14959, 24242, 20000, 16547, 26212, 32303, 22742, 27451, 29615, 26460, 16746, 16243, 24173, 15663, 6405, 23453, 5690, 4391, 19011, 25273, 18853, 20711, 24641, 30524, 1505, 14117, 17400, 21265, 32445, 17346, 25545, 8414, 4821, 32570, 31064, 31368, 24586, 3391, 24990, 12537, 31793, 21153, 15162, 16673, 9736, 12913, 16136, 6714, 622, 9830, 1781, 6481, 14803, 13951, 14098";

if(pr\_mode != 5)

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

switch (this->comboBox2->SelectedIndex)

{

case 2: this->input\_textBox->Text = arr\_1\_1 + arr\_1\_2 + arr\_1\_3 + arr\_1\_4 + arr\_1\_5; break;

case 3: this->input\_textBox->Text = arr\_2\_1 + arr\_2\_2 + arr\_2\_3; break;

case 4: this->input\_textBox->Text = arr\_3\_1 + arr\_3\_2 + arr\_3\_3 + arr\_3\_4 + arr\_3\_5; break;

}

chosen\_mode = this->comboBox2->SelectedIndex;

if (chosen\_mode == 0)

{

input\_check();

this->input\_textBox->ReadOnly = false;

}

else if (chosen\_mode > 1 && chosen\_mode < 5)

{

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

}

else if (chosen\_mode == 5)

{

pr\_mode = 5;

this->input\_textBox->Text = "Введите размер случайного массива: ";

}

}

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

{

if (double\_clear\_click\_check())

{

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

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

this->comboBox1->Text = "";

this->comboBox1->SelectedIndex = -1;

this->comboBox2->Text = "";

this->comboBox2->SelectedIndex = -1;

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

this->input\_textBox->ReadOnly = false;

this->pr\_mode = -1;

}

}

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

{

MessageBox::Show("Для работы с программой\r\n1. Введите массив в поле ввода через запятую с пробелом (пример \",\_\" \_ - знак пробела)\r\n2. Выберите тип вводимого массива\r\n3. Выберите алгоритм для сортировки в выпадающем списке\r\n4. Нажмите кнопку \"Запуск\"\r\nВ специальных полях будет выведено время выполнения сортирововк и полученный отсортированный массив\r\nДля повтороной работы с программной нажмине кнопку \"Очистить поля\".\r\n\nДля ввода тестового массива выберите в выпадающем списке \"Тип вводимого массива\" вариант заготовленного массива и нажмите кнопку \"Запуск\"", "Информация о работе программы", MessageBoxButtons::OK, MessageBoxIcon::Information);

}

private: int empty\_check()

{

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

{

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

return 1;

}

else

{

return 0;

}

}

private: bool double\_clear\_click\_check()

{

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

{

MessageBox::Show("Поля уже очищены", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

return false;

}

else

{

return true;

}

}

private: int not\_choosen\_variant\_check()

{

if (this->comboBox1->SelectedIndex == -1)

{

MessageBox::Show("Не выбран вариант алгоритма сортировки", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

return 1;

}

else

{

return 0;

}

}

private: int not\_choosen\_mode\_check()

{

if (this->comboBox2->SelectedIndex == -1)

{

MessageBox::Show("Не выбран вариант вводимого массива", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

return 1;

}

else

{

return 0;

}

}

private: bool compare\_arrays(cli::array<UInt32>^ a, cli::array<UInt32>^ b)

{

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

{

if (a[i] != b[i])

{

return false;

}

}

return true;

}

private: int test\_radix\_sort()

{

cli::array<UInt32>^ test = {5, 4, 3, 2, 1};

cli::array<UInt32>^ compare = { 1, 2, 3, 4, 5 };

radix\_sort(test, 5);

if (compare\_arrays(test, compare))

{

return 0;

}

else

{

return 1;

}

}

private: int test\_insertion\_sort()

{

cli::array<UInt32>^ test = { 5, 4, 3, 2, 1 };

cli::array<UInt32>^ compare = { 1, 2, 3, 4, 5 };

insertion\_sort(test, 5);

if (compare\_arrays(test, compare))

{

return 0;

}

else

{

return 1;

}

}

private: int test\_bubble\_sort()

{

cli::array<UInt32>^ test = { 5, 4, 3, 2, 1 };

cli::array<UInt32>^ compare = { 1, 2, 3, 4, 5 };

bubble\_sort(test, 5);

if (compare\_arrays(test, compare))

{

return 0;

}

else

{

return 1;

}

}

private: int test\_shaker\_sort()

{

cli::array<UInt32>^ test = { 5, 4, 3, 2, 1 };

cli::array<UInt32>^ compare = { 1, 2, 3, 4, 5 };

shaker\_sort(test, 5);

if (compare\_arrays(test, compare))

{

return 0;

}

else

{

return 1;

}

}

private: int test\_quick\_sort()

{

cli::array<UInt32>^ test = { 5, 4, 3, 2, 1 };

cli::array<UInt32>^ compare = { 1, 2, 3, 4, 5 };

quick\_sort(test, 0, 4);

if (compare\_arrays(test, compare))

{

return 0;

}

else

{

return 1;

}

}

private: void test\_cases()

{

int result = test\_radix\_sort() + test\_insertion\_sort() + test\_bubble\_sort() + test\_shaker\_sort() + test\_quick\_sort();

if (result == 0)

{

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

}

else

{

MessageBox::Show("Тесты не пройдены, программа работает некорректно", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

this->Close();

}

}

private: System::Void button4\_Click(System::Object^ sender, System::EventArgs^ e)

{

test\_cases();

}

private: void test\_cases\_constructor()

{

int test\_flag = test\_radix\_sort() + test\_insertion\_sort() + test\_bubble\_sort() + test\_shaker\_sort() + test\_quick\_sort();

if (test\_flag != 0)

{

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

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

{

this->Close();

}

}

}

};

}

Приложение И

Листинг программы, файл MyForm.cpp

#include "LAB\_2.h"

#include <Windows.h>

using namespace System;

using namespace System::Windows::Forms;

[STAThreadAttribute]

int main()

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

LAB\_2::LAB\_2 form;

Application::Run(% form);

}

Приложение К

Листинг программы, файл UnitTestLAB\_2.cpp

#include "pch.h"

#include "CppUnitTest.h"

#include "../LAB\_2/Functions.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

int arr\_test[5] = { 1, 2, 3, 4, 5 };

namespace UnitTestLAB2

{

TEST\_CLASS(UnitTestLAB2)

{

public:

TEST\_METHOD(radix\_sort\_test)

{

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

radix\_sort(arr, 5);

Assert::IsTrue(compare\_arrays(arr, arr\_test));

}

TEST\_METHOD(insertion\_sort\_test)

{

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

insertion\_sort(arr, 5);

Assert::IsTrue(compare\_arrays(arr, arr\_test));

}

TEST\_METHOD(bubble\_sort\_test)

{

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

bubble\_sort(arr, 5);

Assert::IsTrue(compare\_arrays(arr, arr\_test));

}

TEST\_METHOD(shaker\_sort\_test)

{

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

shaker\_sort(arr, 5);

Assert::IsTrue(compare\_arrays(arr, arr\_test));

}

TEST\_METHOD(quick\_sort\_test)

{

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

quick\_sort(arr, 0, 4);

Assert::IsTrue(compare\_arrays(arr, arr\_test));

}

};

}

Приложение Л

Листинг программы, файл Functions.h

#pragma once

#include <iostream>

int\* shaker\_sort(int\* a, int quantity)

{

int left = 0, right = quantity - 1;

int flag = 1;

while ((left < right) && flag > 0)

{

flag = 0;

for (int i = left; i < right; i++)

{

if (a[i] > a[i + 1])

{

unsigned int t = a[i];

a[i] = a[i + 1];

a[i + 1] = t;

flag = 1;

}

}

right--;

for (int i = right; i > left; i--)

{

if (a[i - 1] > a[i])

{

unsigned int t = a[i];

a[i] = a[i - 1];

a[i - 1] = t;

flag = 1;

}

}

left++;

}

return a;

}

int\* insertion\_sort(int\* a, int quantity)

{

for (int i = 1; i < quantity; i++)

{

for (int j = i; j > 0 && a[j] < a[j - 1]; j--)

{

unsigned int t = a[j];

a[j] = a[j - 1];

a[j - 1] = t;

}

}

return a;

}

int\* bubble\_sort(int\* a, int quantity)

{

for (int i = 0; i < quantity - 1; i++)

{

for (int j = 0; j < quantity - 1; j++)

{

if (a[j] > a[j + 1])

{

unsigned int t = a[j];

a[j] = a[j + 1];

a[j + 1] = t;

}

}

}

return a;

}

void quick\_sort(int arr[], int start, int end)

{

int l = start;

int r = end;

int piv = arr[(l + r) / 2];

while (l <= r)

{

while (arr[l] < piv) l++;

while (arr[r] > piv) r--;

if (l <= r)

{

int t = arr[l];

arr[l] = arr[r];

arr[r] = t;

l++;

r--;

}

}

if (start < r)

quick\_sort(arr, start, r);

if (end > l)

quick\_sort(arr, l, end);

}

unsigned int max\_value(int\* a, int quantity)

{

int max = a[0];

for (int i = 1; i < quantity; i++)

if (a[i] > max)

max = a[i];

return max;

}

int\* radix\_sort(int\* a, int quantity)

{

unsigned int exp, i;

i = max\_value(a, quantity);

for (exp = 1; i / exp > 0; exp \*= 10)

{

int i, count[10] = { 0 };

int\* result = new int(quantity);

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

count[(a[i] / exp) % 10]++;

for (i = 1; i < 10; i++)

count[i] += count[i - 1];

for (i = quantity - 1; i >= 0; i--)

{

result[count[(a[i] / exp) % 10] - 1] = a[i];

count[(a[i] / exp) % 10]--;

}

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

a[i] = result[i];

}

return a;

}

bool compare\_arrays(int\* a, int\* b)

{

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

{

if (a[i] != b[i])

{

return false;

}

}

return true;

}

Приложение М

Время выполнения сортировки

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Поразрядная | Вставками | Пузырьковая | Шейкера | Быстрая |
| Массив 1 | 0,001 | 0,01 | 0,03 | 0,016 | 0 |
| Массив 2 | 0 | 0,01 | 0,03 | 0,015 | 0 |
| Массив 3 | 0 | 0,007 | 0,026 | 0,01 | 0 |