**Лабораторная работа №3**

**Сортировка массивов**

Составить программу, проводящую сравнительную характеристику методов сортировки массивов.

Программа должна выполнять следующие действия:

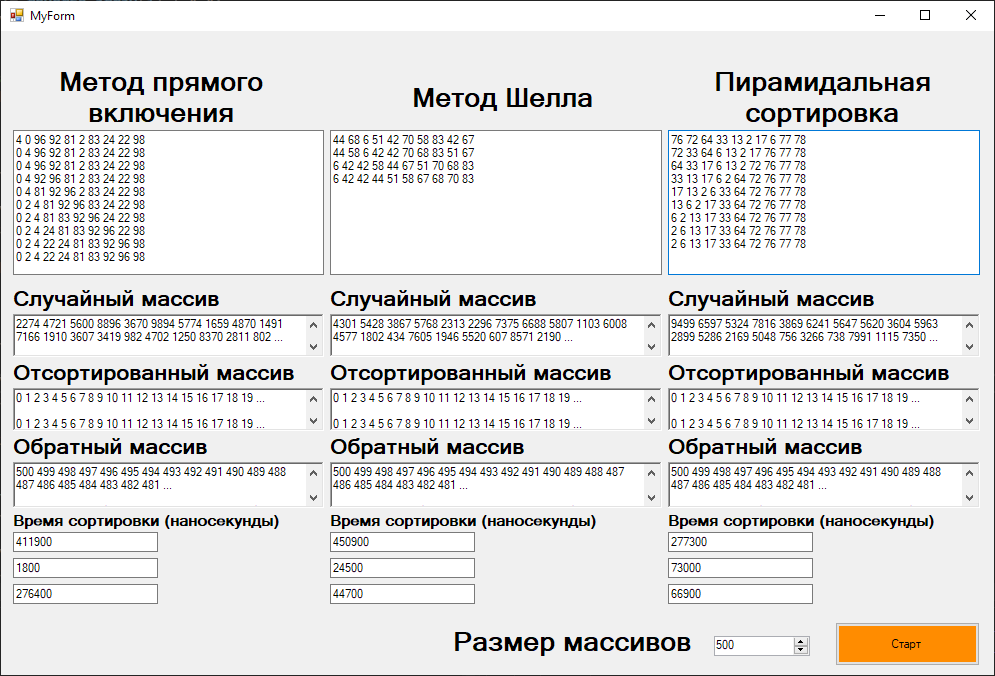
1. Производить сортировку массива соответствующими методами.

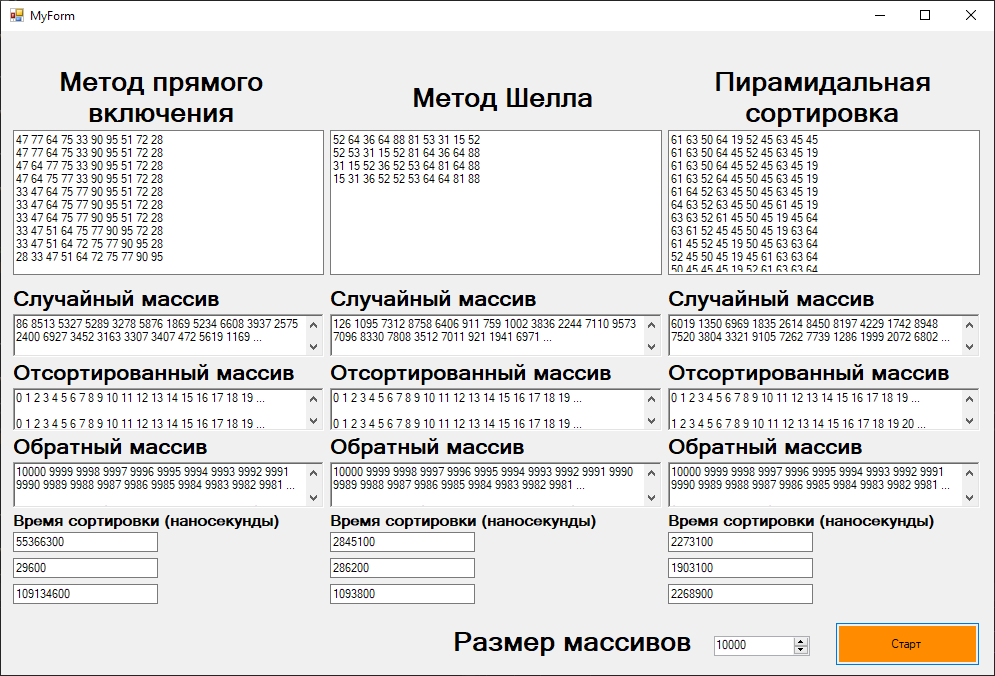
2. Иллюстрировать работу каждого метода на небольших массивах (размером до 10 элементов).

3. Производить сортировку каждым из методов случайного массива (50000), уже отсортированного массива, массива, отсортированного в обратном порядке. Засечь время. Размер массива при этом должен выбираться пользователем. После проведения сортировки, вывести данные о скорости работы методов.

2.Метод прямого включения, метод Шелла, пирамидальная сортировка.

* Создание функции метода прямого включения
* Создание функции метода Шелла
* Создание функции пирамидальной сортировки
* Для работы необходимы три массива одинакового размера, размер которых может задавать пользователь: random\_array (случайный массив), sorted\_array (отсортированный массив) и reverse\_array (массив, отсортированный в обратном порядке).
* Произвести действия для каждого из трёх уже описанных методов:
  + Заполнение массивов, вывод на экран их содержимого
  + Засечь время (возможно с использованием chrono::steady\_clock) для каждого массива и провести сортировку заданным методом
  + Вывод отсортированных массивов с временем сортировки каждого





#include "MyForm.h"

#include <chrono>

#include <ctime>

#include <iostream>

using namespace System;

using namespace System::Windows::Forms;

[STAThreadAttribute]

int main(array<String^>^ args)

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Lab3::MyForm form;

Application::Run(% form);

}

//метод прямого включения

System::Void Lab3::MyForm::insertion(int\* mass, int\* n) {

for (int i = 1; i < \*n; i++)

{

int x = mass[i], j = i;

while (j != 0 && x < mass[j - 1]) {

mass[j] = mass[j - 1];

j--;

}

mass[j] = x;

}

}

//метод Шелла

System::Void Lab3::MyForm::shell(int\* mass, int\* n) {

int step = \*n / 2;

while (step > 0) {

for (int i = 0; i < \*n - step; i++) {

int j = i;

while (j >= 0 && mass[j] > mass[j + step]) {

std::swap(mass[j], mass[j + step]);

j -= step;

}

}

step /= 2;

}

}

//пирамидальная сортировка

void push\_down(int\* mass, int root, int bottom) {

bool done = false;

int max\_child;

while (2 \* root + 1 <= bottom && !done) {

if (2 \* root + 1 == bottom)max\_child = 2 \* root + 1;

else if (mass[2 \* root + 1] > mass[2 \* root + 2])max\_child = 2 \* root + 1;

else max\_child = 2 \* root + 2;

if (mass[root] < mass[max\_child]) {

std::swap(mass[root], mass[max\_child]);

root = max\_child;

}

else done = true;

}

}

System::Void Lab3::MyForm::heap\_sort(int\* mass, int\* n) {

for (int i = \*n / 2 - 1; i >= 0; i--)

push\_down(mass, i, \*n - 1);

for (int i = \*n - 1; i >= 0; i--) {

std::swap(mass[0], mass[i]);

push\_down(mass, 0, i - 1);

}

}

System::Void Lab3::MyForm::button\_run\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (numericUpDown1->Value > 0) {

srand(time(NULL));

int m = System::Convert::ToInt32(numericUpDown1->Value);

int\* random\_array{ new int[m] };

int\* sorted\_array{ new int[m] };

int\* reverse\_array{ new int[m] };

int count = 20;

if (count > numericUpDown1->Value) count = System::Convert::ToInt32(numericUpDown1->Value);

//изолированный блок теста

{

this->textBox10->Text = ""; this->textBox11->Text = ""; this->textBox12->Text = "";

int k = 10;

//тестовый массив 1

int\* small\_array1{ new int[k] };

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

small\_array1[i] = rand() % 100;

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

this->textBox10->Text += small\_array1[i] + " ";

this->textBox10->Text += Environment::NewLine;

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

{

int x = small\_array1[i], j = i;

while (j != 0 && x < small\_array1[j - 1]) {

small\_array1[j] = small\_array1[j - 1];

j--;

}

small\_array1[j] = x;

for (int u = 0; u < k; u++)

this->textBox10->Text += small\_array1[u] + " ";

this->textBox10->Text += Environment::NewLine;

}

//тестовый массив 2

int\* small\_array2{ new int[k] };

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

small\_array2[i] = rand() % 100;

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

this->textBox11->Text += small\_array2[i] + " ";

this->textBox11->Text += Environment::NewLine;

int step = k / 2;

while (step > 0) {

for (int i = 0; i < k - step; i++) {

int j = i;

while (j >= 0 && small\_array2[j] > small\_array2[j + step]) {

std::swap(small\_array2[j], small\_array2[j + step]);

j -= step;

}

}

step /= 2;

for (int u = 0; u < k; u++)

this->textBox11->Text += small\_array2[u] + " ";

this->textBox11->Text += Environment::NewLine;

}

//тестовый массив 3

int\* small\_array3{ new int[k] };

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

small\_array3[i] = rand() % 100;

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

this->textBox12->Text += small\_array3[i] + " ";

this->textBox12->Text += Environment::NewLine;

for (int i = k / 2 - 1; i >= 0; i--) {

push\_down(small\_array3, i, k - 1);

for (int u = 0; u < k; u++)

this->textBox12->Text += small\_array3[u] + " ";

this->textBox12->Text += Environment::NewLine;

}

for (int i = k - 1; i >= 0; i--) {

std::swap(small\_array3[0], small\_array3[i]);

push\_down(small\_array3, 0, i - 1);

for (int u = 0; u < k; u++)

this->textBox12->Text += small\_array3[u] + " ";

this->textBox12->Text += Environment::NewLine;

}

}

//изолированный блок метода прямого включения

{

this->richTextBox1->Text = ""; this->richTextBox2->Text = ""; this->richTextBox3->Text = "";

this->textBox1->Text = ""; this->textBox2->Text = ""; this->textBox3->Text = "";

//случайный массив

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

random\_array[i] = rand() % 10000;

//отсортированный массив

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

sorted\_array[i] = i;

//обратный массив

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

reverse\_array[i] = m - i;

//вывод массивов

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

this->richTextBox1->Text += random\_array[i] + " ";

if (count < numericUpDown1->Value)this->richTextBox1->Text += "...";

this->richTextBox1->Text += "\n\n";

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

this->richTextBox2->Text += sorted\_array[i] + " ";

if (count < numericUpDown1->Value)this->richTextBox2->Text += "...";

this->richTextBox2->Text += "\n\n";

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

this->richTextBox3->Text += reverse\_array[i] + " ";

if (count < numericUpDown1->Value)this->richTextBox3->Text += "...";

this->richTextBox3->Text += "\n\n";

//измерение скорости сортировки прямым включением

auto begin\_random\_array = std::chrono::steady\_clock::now();

insertion(random\_array, &m);

auto end\_random\_array = std::chrono::steady\_clock::now();

auto elapsed\_ms\_random\_array = std::chrono::duration\_cast<std::chrono::nanoseconds>(end\_random\_array - begin\_random\_array);

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

this->richTextBox1->Text += random\_array[i] + " ";

if (count < numericUpDown1->Value)this->richTextBox1->Text += "...";

this->textBox1->Text += elapsed\_ms\_random\_array.count();

auto begin\_sorted\_array = std::chrono::steady\_clock::now();

insertion(sorted\_array, &m);

auto end\_sorted\_array = std::chrono::steady\_clock::now();

auto elapsed\_ms\_sorted\_array = std::chrono::duration\_cast<std::chrono::nanoseconds>(end\_sorted\_array - begin\_sorted\_array);

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

this->richTextBox2->Text += sorted\_array[i] + " ";

if (count < numericUpDown1->Value)this->richTextBox2->Text += "...";

this->textBox2->Text += elapsed\_ms\_sorted\_array.count();

auto begin\_reverse\_array = std::chrono::steady\_clock::now();

insertion(reverse\_array, &m);

auto end\_reverse\_array = std::chrono::steady\_clock::now();

auto elapsed\_ms\_reverse\_array = std::chrono::duration\_cast<std::chrono::nanoseconds>(end\_reverse\_array - begin\_reverse\_array);

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

this->richTextBox3->Text += reverse\_array[i] + " ";

if (count < numericUpDown1->Value)this->richTextBox3->Text += "...";

this->textBox3->Text += elapsed\_ms\_reverse\_array.count();

}

//изолированный блок метода Шелла

{

this->richTextBox6->Text = ""; this->richTextBox5->Text = ""; this->richTextBox4->Text = "";

this->textBox6->Text = ""; this->textBox5->Text = ""; this->textBox4->Text = "";

//случайный массив

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

random\_array[i] = rand() % 10000;

//отсортированный массив

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

sorted\_array[i] = i;

//обратный массив

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

reverse\_array[i] = m - i;

//вывод массивов

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

this->richTextBox6->Text += random\_array[i] + " ";

if (count < numericUpDown1->Value)this->richTextBox6->Text += "...";

this->richTextBox6->Text += "\n\n";

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

this->richTextBox5->Text += sorted\_array[i] + " ";

if (count < numericUpDown1->Value)this->richTextBox5->Text += "...";

this->richTextBox5->Text += "\n\n";

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

this->richTextBox4->Text += reverse\_array[i] + " ";

if (count < numericUpDown1->Value)this->richTextBox4->Text += "...";

this->richTextBox4->Text += "\n\n";

//измерение скорости сортировки прямым включением

auto begin\_random\_array = std::chrono::steady\_clock::now();

shell(random\_array, &m);

auto end\_random\_array = std::chrono::steady\_clock::now();

auto elapsed\_ms\_random\_array = std::chrono::duration\_cast<std::chrono::nanoseconds>(end\_random\_array - begin\_random\_array);

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

this->richTextBox6->Text += random\_array[i] + " ";

if (count < numericUpDown1->Value)this->richTextBox6->Text += "...";

this->textBox6->Text += elapsed\_ms\_random\_array.count();

auto begin\_sorted\_array = std::chrono::steady\_clock::now();

shell(sorted\_array, &m);

auto end\_sorted\_array = std::chrono::steady\_clock::now();

auto elapsed\_ms\_sorted\_array = std::chrono::duration\_cast<std::chrono::nanoseconds>(end\_sorted\_array - begin\_sorted\_array);

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

this->richTextBox5->Text += sorted\_array[i] + " ";

if (count < numericUpDown1->Value)this->richTextBox5->Text += "...";

this->textBox5->Text += elapsed\_ms\_sorted\_array.count();

auto begin\_reverse\_array = std::chrono::steady\_clock::now();

shell(reverse\_array, &m);

auto end\_reverse\_array = std::chrono::steady\_clock::now();

auto elapsed\_ms\_reverse\_array = std::chrono::duration\_cast<std::chrono::nanoseconds>(end\_reverse\_array - begin\_reverse\_array);

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

this->richTextBox4->Text += reverse\_array[i] + " ";

if (count < numericUpDown1->Value)this->richTextBox4->Text += "...";

this->textBox4->Text += elapsed\_ms\_reverse\_array.count();

}

//изолированный блок пирамидальной сортировки

{

this->richTextBox9->Text = ""; this->richTextBox8->Text = ""; this->richTextBox7->Text = "";

this->textBox9->Text = ""; this->textBox8->Text = ""; this->textBox7->Text = "";

//случайный массив

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

random\_array[i] = rand() % 10000;

//отсортированный массив

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

reverse\_array[i] = i;

//обратный массив

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

sorted\_array[i] = m - i;

//вывод массивов

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

this->richTextBox9->Text += random\_array[i] + " ";

if (count < numericUpDown1->Value)this->richTextBox9->Text += "...";

this->richTextBox9->Text += "\n\n";

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

this->richTextBox8->Text += sorted\_array[i] + " ";

if (count < numericUpDown1->Value)this->richTextBox8->Text += "...";

this->richTextBox8->Text += "\n\n";

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

this->richTextBox7->Text += reverse\_array[i] + " ";

if (count < numericUpDown1->Value)this->richTextBox7->Text += "...";

this->richTextBox7->Text += "\n\n";

//измерение скорости сортировки прямым включением

auto begin\_random\_array = std::chrono::steady\_clock::now();

heap\_sort(random\_array, &m);

auto end\_random\_array = std::chrono::steady\_clock::now();

auto elapsed\_ms\_random\_array = std::chrono::duration\_cast<std::chrono::nanoseconds>(end\_random\_array - begin\_random\_array);

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

this->richTextBox9->Text += random\_array[i] + " ";

if (count < numericUpDown1->Value)this->richTextBox9->Text += "...";

this->textBox9->Text += elapsed\_ms\_random\_array.count();

auto begin\_sorted\_array = std::chrono::steady\_clock::now();

heap\_sort(sorted\_array, &m);

auto end\_sorted\_array = std::chrono::steady\_clock::now();

auto elapsed\_ms\_sorted\_array = std::chrono::duration\_cast<std::chrono::nanoseconds>(end\_sorted\_array - begin\_sorted\_array);

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

this->richTextBox8->Text += sorted\_array[i] + " ";

if (count < numericUpDown1->Value)this->richTextBox8->Text += "...";

this->textBox8->Text += elapsed\_ms\_sorted\_array.count();

auto begin\_reverse\_array = std::chrono::steady\_clock::now();

heap\_sort(reverse\_array, &m);

auto end\_reverse\_array = std::chrono::steady\_clock::now();

auto elapsed\_ms\_reverse\_array = std::chrono::duration\_cast<std::chrono::nanoseconds>(end\_reverse\_array - begin\_reverse\_array);

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

this->richTextBox7->Text += reverse\_array[i] + " ";

if (count < numericUpDown1->Value)this->richTextBox7->Text += "...";

this->textBox7->Text += elapsed\_ms\_reverse\_array.count();

}

}

else MessageBox::Show(L"Неверный размер массива", L"Ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Error);

}