МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет: Информационных технологий

Кафедра: Программной инженерии

Выполнила: студентка 1 курса 5 группы

специальности ПОИТ Дзивнель М.А.

Проверил: преподаватель

Белодед Николай Иванович

**РЕФЕРАТ**

По дисциплине “Основы алгоритмизации и программирования”

На тему “Сортировки”

Минск

2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ ……………………………………………………………………….2

Сортировка пузырьком……….……………………………………………2

Шейкерная сортировка……………………………………………………..3

Сортировка выбором……………………………………………………….3

Сортировка вставками……………………………………………………...4

Быстрая сортировка………………………………………………………...5

Тестирование сортировок………………………………………………….6

Визуализация тестирования………………………………………………11

Код…………………………………………………………………..12

Результат……………………………………………………………17

ВЫВОД …………………………………………………………………………..17

**ВВЕДЕНИЕ**

При работе со структурами данных очень часто возникает потребность с их сортировкой. Например, сортировка нужна для облегчения поиска данных, для более удобного чтения их, для работы с ними. Существует множество сортировок, и основные из них будут приведены ниже.

**СОРТИРОВКА ПУЗЫРЬКОМ**

Сортировка пузырьком - это самый простой алгоритм сортировки, который работает путем многократной замены соседних элементов, если они находятся в неправильном порядке.

**Реализация на C/C++**

int\* bubble\_sort(int\* arr,int n\_arr) {

int temp = 0; //Временная переменная

for (int i = 0; i < n\_arr - 1; i++) {

for (int j = i + 1; j < n\_arr; j++) {

if (arr[i] > arr[j]) {

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

return arr;

}

**ШЕЙКЕРНАЯ СОРТИРОВКА (СОРТИРОВКА ПЕРЕМЕШИВАНИЯ)**

Разновидность пузырька. На первом проходе как обычно — задвигаем максимум в конец. Потом резко разворачиваемся и толкаем минимум в начало. Отсортированные крайние области массива увеличиваются в размерах после каждой итерации.

**Реализация на C/C++**

int\* shaker\_sort(int\* arr,int n\_arr) {

int left, right, i;

left = 0;

right = n\_arr - 1;

while (left <= right) {

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

if (arr[i - 1] > arr[i]) {

swap(arr[i - 1], arr[i]);

}

}

left++;

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

if (arr[i - 1] > arr[i]) {

swap(arr[i - 1], arr[i]);

}

}

right--;

}

return arr;

}

**СОРТИРОВКА ВЫБОРОМ**

В чём идея сортировок выбором?

1. В неотсортированном подмассиве ищется локальный максимум (минимум).
2. Найденный максимум (минимум) меняется местами с последним (первым) элементом в подмассиве.
3. Если в массиве остались неотсортированные подмассивы — смотри пункт 1.

**Реализация на C/C++**

int\* selection\_sort(int\* arr,int n\_arr) {

int temp; //Временная переменная

int max;

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

max = i;

for (int j = i - 1; j > 0; j--)

max = (arr[j] > arr[max]) ? j : max;

if (i != max) {

temp = arr[i];

arr[i] = arr[max];

arr[max] = temp;

}

}

return arr;

}

**СОРТИРОВКА ВСТАВКАМИ**

Сортировка вставками — алгоритм сортировки, в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов.

**Реализация на C/C++**

int\* insertion\_sort(int\* arr,int n\_arr) {

int temp,i,j;

for (i = 1; i < n\_arr; i++) {

temp = arr[i];

for (j = i - 1; j >= 0 && arr[j] > temp; j--)

arr[j + 1] = arr[j];

arr[j + 1] = temp;

}

return arr;

}

**БЫСТРАЯ СОРТИРОВКА**

Быстрая сортировка - в целом это один из самых быстрых алгоритмов сортировки массивов. Является примером принципа «разделяй и властвуй». Идея алгоритма заключается в том, что выбирается опорный элемент, относительно которого будет происходит сортировка. Равные и бОльшие элементы помещаются справа, меньшие – слева. Затем к полученным подмассивам рекурсивно применяются два первых пункта.

**Реализация на C/C++**

int\* quick\_sort(int b, int e, int\* arr) //b-первый элемент массива,а е-последний

{

int l = b, r = e;

int piv = arr[(l + r) / 2]; // Опорным элементом для примера возьмём средний

while (l <= r)

{

while (arr[l] < piv)

l++;

while (arr[r] > piv)

r--;

if (l <= r)

swap(arr[l++], arr[r--]);

}

if (b < r)

quick\_sort(b, r, arr);

if (e > l)

quick\_sort(l, e, arr);

return arr;

}

//вызов осуществляется следующим образом quick\_sort(0, n-1, arr);

**КОД ДЛЯ ТЕСТИРОВАНИЯ**

Для того, чтобы определить самую эффективную сортировку, необходимо протестировать каждую из сортировок в лучшей и случайной ситуациях. Было создано два массива на 200 000 элементов. Первый заполнен случайными числами, а второй упорядоченными (и первый элемент со вторым поменяны местами) для создания лучшей ситуации. Каждый массив отсортирован всеми представленными сортировками

#include <iostream>

#include "graph.h"

#define ARR\_N 200000

#define SZ 5

using namespace std;

void bubble\_sort(int\* arr, int n\_arr);

void shaker\_sort(int\* arr, int n\_arr);

void selection\_sort(int\* arr, int n\_arr);

void insertion\_sort(int\* arr, int n\_arr);

void quick\_sort(int b, int e, int\* arr);

int main() {

setlocale(LC\_ALL, "Russian");

system("color F0");

system("cls");

cout << "Please, wait.." << endl;

srand(unsigned(time(0))\*200\*100);

//Создание массивов

static int arrA[ARR\_N];

static int arrB[ARR\_N];

int\* arrTime = new int[SZ];

//Заполнение массива A случайными числами ( случайная ситуация )

for (int i = 0; i < ARR\_N; i++) {

arrA[i] = rand();

}

//Заполнение массива B упорядоченными числами ( лучшая ситуация )

for (int i = 0; i < ARR\_N; i++) {

arrB[i] = i;

}

// создание лучшей ситуации в массиве arrB

int temp = arrB[1];

arrB[1] = arrB[2];

arrB[2] = temp;

//Время

unsigned int start\_time, end\_time;

//----------------------------------------случайная ситуация-----------------------------------

//Пузырьковская сортировка

start\_time = clock();

bubble\_sort(arrA, ARR\_N);

end\_time = clock();

unsigned int bubble\_sort\_time = end\_time - start\_time;

arrTime[0] = bubble\_sort\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

// Шейкерная сортировка

start\_time = clock();

shaker\_sort(arrA, ARR\_N);

end\_time = clock();

unsigned int shaker\_sort\_time = end\_time - start\_time;

arrTime[1] = shaker\_sort\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

// Сортировка выбором

start\_time = clock();

selection\_sort(arrA, ARR\_N);

end\_time = clock();

unsigned int selection\_sort\_time = end\_time - start\_time;

arrTime[2] = selection\_sort\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

// Сортировка вставками

start\_time = clock();

insertion\_sort(arrA, ARR\_N);

end\_time = clock();

unsigned int insertion\_sort\_time = end\_time - start\_time;

arrTime[3] = insertion\_sort\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

// Быстрая сортировка

start\_time = clock();

quick\_sort(0, ARR\_N-1, arrA);

end\_time = clock();

unsigned int quick\_sort\_time = end\_time - start\_time;

arrTime[4] = quick\_sort\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

//Вывод времени

cout << "--------------случайная ситуация-----------------\n";

cout << "\nВремя сортировки пузырьковой:" << bubble\_sort\_time << "мс.\n";

cout << "\nВремя сортировки Шейкером:" << shaker\_sort\_time << "мс.\n";

cout << "\nВремя сортировки выбором:" << selection\_sort\_time << "мс.\n";

cout << "\nВремя сортировки вставкой:" << insertion\_sort\_time << "мс.\n";

cout << "\nВремя сортировки быстрой:" << quick\_sort\_time << "мс.\n\n\n";

system("pause");

system("cls");

DrawGraphics(arrTime);

system("pause");

cout << "Please, wait.." << endl;

//----------------------------------------лучшая ситуация-----------------------------------

//Пузырьковская сортировка

start\_time = clock();

bubble\_sort(arrB, ARR\_N);

end\_time = clock();

bubble\_sort\_time = end\_time - start\_time;

arrTime[0] = bubble\_sort\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

// Шейкерная сортировка

start\_time = clock();

shaker\_sort(arrB, ARR\_N);

end\_time = clock();

shaker\_sort\_time = end\_time - start\_time;

arrTime[1] = shaker\_sort\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

// Сортировка выбором

start\_time = clock();

selection\_sort(arrB, ARR\_N);

end\_time = clock();

selection\_sort\_time = end\_time - start\_time;

arrTime[2] = selection\_sort\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

// Сортировка вставками

start\_time = clock();

insertion\_sort(arrB, ARR\_N);

end\_time = clock();

insertion\_sort\_time = end\_time - start\_time;

arrTime[3] = insertion\_sort\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

// Быстрая сортировка

start\_time = clock();

quick\_sort(0, ARR\_N - 1, arrB);

end\_time = clock();

quick\_sort\_time = end\_time - start\_time;

arrTime[4] = quick\_sort\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

//Вывод времени

cout << "--------------лучшая ситуация-----------------\n";

cout << "\nВремя сортировки пузырьковой:" << bubble\_sort\_time << "мс.\n";

cout << "\nВремя сортировки Шейкером:" << shaker\_sort\_time << "мс.\n";

cout << "\nВремя сортировки выбором:" << selection\_sort\_time << "мс.\n";

cout << "\nВремя сортировки вставкой:" << insertion\_sort\_time << "мс.\n";

cout << "\nВремя сортировки быстрой:" << quick\_sort\_time << "мс.\n\n";

system("pause");

system("cls");

DrawGraphics( arrTime);

delete[]arrTime;

return 0;

}

void bubble\_sort(int\* arr, int n\_arr) {

int temp = 0; //Временная переменная

for (int i = 0; i < n\_arr - 1; i++) {

for (int j = i + 1; j < n\_arr; j++) {

if (arr[i] > arr[j]) {

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

}

void shaker\_sort(int\* arr, int n\_arr) {

int left, right, i;

left = 0;

right = n\_arr - 1;

while (left <= right) {

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

if (arr[i - 1] > arr[i]) {

swap(arr[i - 1], arr[i]);

}

}

left++;

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

if (arr[i - 1] > arr[i]) {

swap(arr[i - 1], arr[i]);

}

}

right--;

}

}

void selection\_sort(int\* arr, int n\_arr) {

int temp; //Временная переменная

int max;

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

max = i;

for (int j = i - 1; j > 0; j--)

max = (arr[j] > arr[max]) ? j : max;

if (i != max) {

temp = arr[i];

arr[i] = arr[max];

arr[max] = temp;

}

}

}

void insertion\_sort(int\* arr, int n\_arr) {

int temp, i, j;

for (i = 1; i < n\_arr; i++) {

temp = arr[i];

for (j = i - 1; j >= 0 && arr[j] > temp; j--)

arr[j + 1] = arr[j];

arr[j + 1] = temp;

}

}

void quick\_sort(int b, int e, int\* arr) //b-первый элемент массива,а е-последний

{

int l = b, r = e;

int piv = arr[(l + r) / 2]; // Опорным элементом для примера возьмём средний

while (l <= r)

{

while (arr[l] < piv)

l++;

while (arr[r] > piv)

r--;

if (l <= r)

swap(arr[l++], arr[r--]);

}

if (b < r)

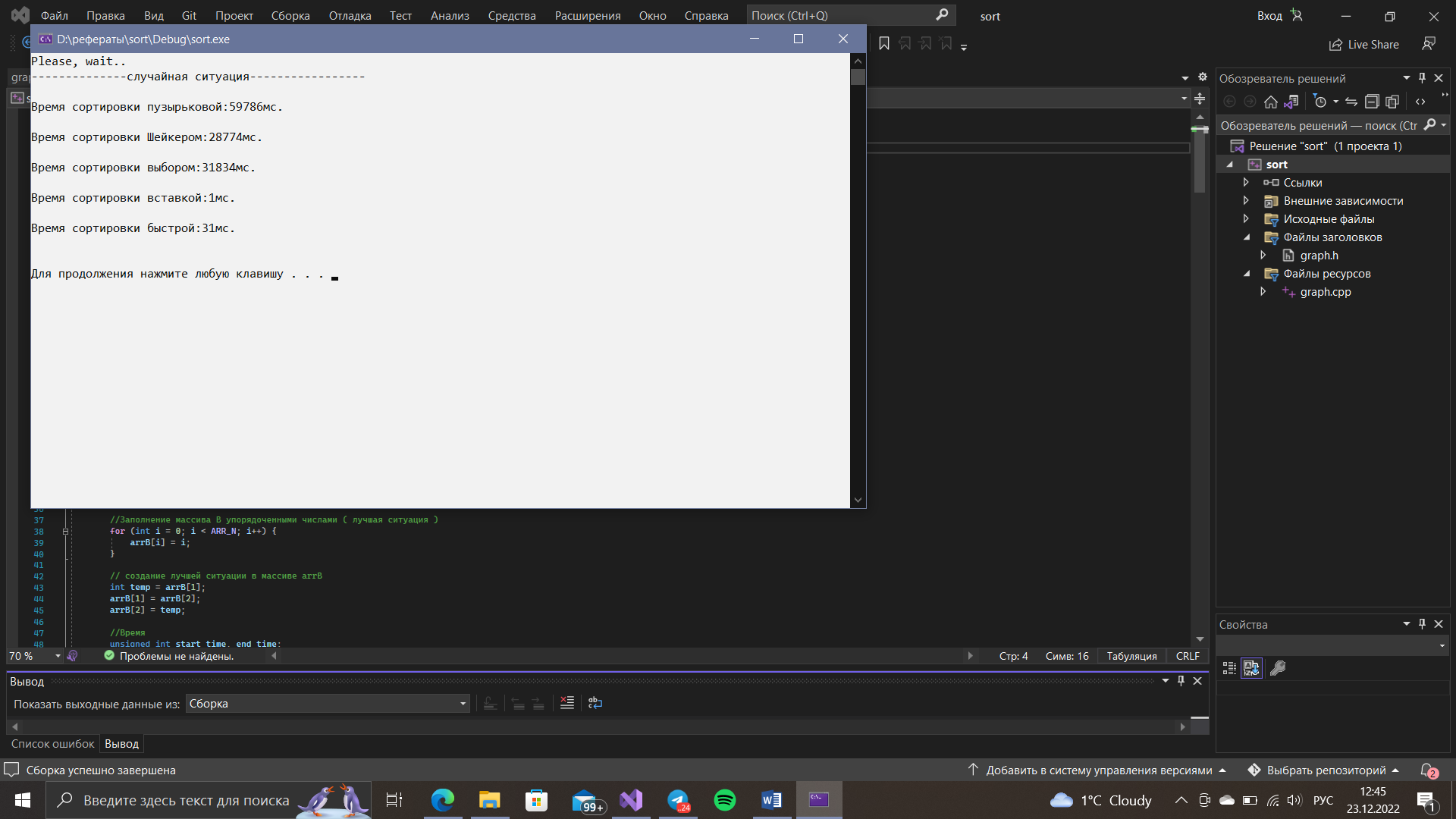
quick\_sort(b, r, arr);

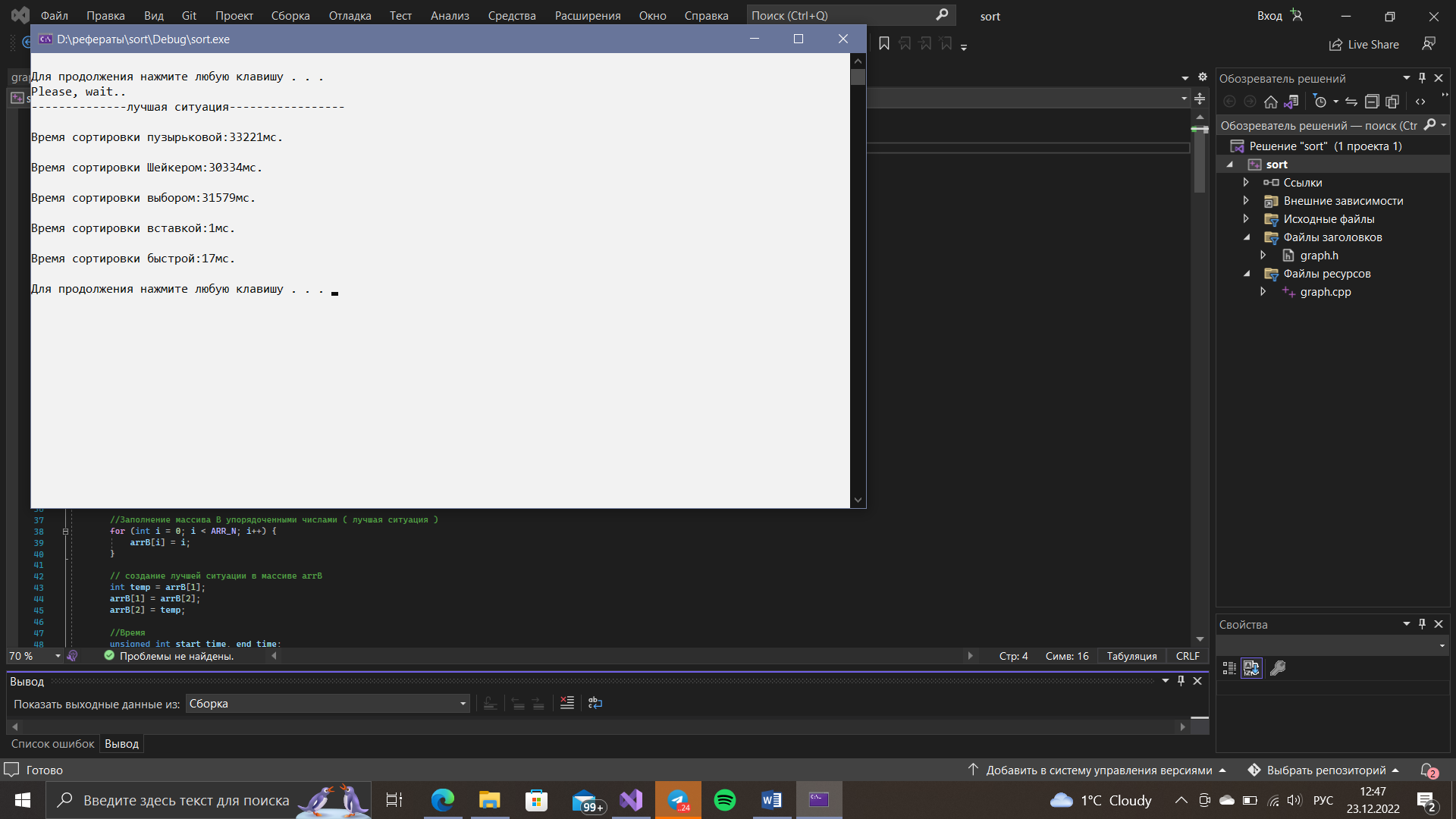
if (e > l)

quick\_sort(l, e, arr);

}

Результат выполнения кода:





**ВИЗУАЛИЗАЦИЯ**

Для графического отображения разницы действия сортировок я решила воспользоваться возможностями языка С++, а именно базовыми возможностями платформы WinAPI. Я написала код, реализующий передачу значений времени действия сортировок в модуль graph, отвечающий за непосредственное построение и вывод графиков(в моем случае диаграмм). Ниже следует код:

#include <Windows.h>

#include <iostream>

#include <string>

#include <wchar.h>

#define HEIGHT 400

#define WIDTH 1000

#define ZERO\_POINT 30

const int DIST = 80, WID\_BLOCK = 100;

const int H\_LINES = 70;

using namespace std;

void DrawAxes(int max\_h)

{

int k = 0;

int hx0, hy0, hx, hy, wx0, wy0, wx, wy;

///оси/////////

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

hx0 = ZERO\_POINT;

hy0 = ZERO\_POINT;

hx = hx0;

hy = hy0 + HEIGHT;

HPEN pen;

pen = CreatePen(PS\_SOLID, 3, RGB(0, 0, 0));

SelectObject(hdc, pen);

MoveToEx(hdc, hx0, hy0, NULL);

LineTo(hdc, hx, hy);

wx0 = ZERO\_POINT;

wy0 = ZERO\_POINT + HEIGHT;

wx = wx0 + WIDTH;

wy = wy0;

MoveToEx(hdc, wx0, wy0, NULL);

LineTo(hdc, wx, wy);

DeleteObject(pen);

//////серые линии///////////

int x, y;

for (int j = hy0 + HEIGHT; j > ZERO\_POINT; j -= H\_LINES)

{

x = hx0 + WIDTH;

y = j;

HPEN pen\_sm;

pen\_sm = CreatePen(PS\_SOLID, 1, RGB(169, 169, 169));

SelectObject(hdc, pen\_sm);

SelectObject(hdc, GetStockObject(GRAY\_BRUSH));

MoveToEx(hdc, hx0, j, NULL);

LineTo(hdc, x, y);

DeleteObject(pen\_sm);

}

/////////подпись осей///////////

int section = max\_h / 5;

HPEN pen\_text;

pen\_text = CreatePen(PS\_SOLID, 1, RGB(0, 0, 0));

SelectObject(hdc, pen\_text);

//////////числа на оси////////////

string s1 = to\_string(section);

wstring stemp = std::wstring(s1.begin(), s1.end());

LPCWSTR sw1 = stemp.c\_str();

TextOut(hdc, 0, ZERO\_POINT + HEIGHT - H\_LINES, sw1, s1.size());

string s2 = to\_string(section\*2);

wstring stemp2 = std::wstring(s2.begin(), s2.end());

LPCWSTR sw2 = stemp2.c\_str();

TextOut(hdc, 0, ZERO\_POINT + HEIGHT - 2 \* H\_LINES, sw2, s2.size());

string s3 = to\_string(section \* 3);

wstring stemp3 = std::wstring(s3.begin(), s3.end());

LPCWSTR sw3 = stemp3.c\_str();

TextOut(hdc, 0, ZERO\_POINT + HEIGHT - 3 \* H\_LINES, sw3, s3.size());

string s4 = to\_string(section \* 4);

wstring stemp4 = std::wstring(s4.begin(), s4.end());

LPCWSTR sw4 = stemp4.c\_str();

TextOut(hdc, 0, ZERO\_POINT + HEIGHT - 4 \* H\_LINES, sw4, s4.size());

string s5 = to\_string(section \* 5);

wstring stemp5 = std::wstring(s5.begin(), s5.end());

LPCWSTR sw5 = stemp5.c\_str();

TextOut(hdc, 0, ZERO\_POINT + HEIGHT - 5 \* H\_LINES, sw5, s5.size());

////////названия сортировок/////////

LPCWSTR bub = L"Bubble";

TextOut(hdc, ZERO\_POINT + DIST + 5, ZERO\_POINT + HEIGHT + 10, bub, 6);

LPCWSTR shak = L"Shacker";

TextOut(hdc, ZERO\_POINT + 2 \* DIST + WID\_BLOCK + 5, ZERO\_POINT + HEIGHT + 10, shak, 7);

LPCWSTR sel = L"Selection";

TextOut(hdc, ZERO\_POINT + 3 \* DIST + 2 \* WID\_BLOCK + 5, ZERO\_POINT + HEIGHT + 10, sel, 9);

LPCWSTR ins = L"Insertion";

TextOut(hdc, ZERO\_POINT + 4 \* DIST + 3 \* WID\_BLOCK + 5, ZERO\_POINT + HEIGHT + 10, ins, 9);

LPCWSTR quick = L"Quick sort";

TextOut(hdc, ZERO\_POINT + 5 \* DIST + 4 \* WID\_BLOCK + 5, ZERO\_POINT + HEIGHT + 10, quick, 10);

/////имя оси/////

LPCWSTR ms = L"(мс)";

TextOut(hdc, 15, 5, ms, 4);

DeleteObject(pen\_text);

}

void DrawBubble(int h\_block, int time)//блок сортировки пузырьком

{

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

int x1 = ZERO\_POINT + DIST, y1 = ZERO\_POINT + HEIGHT - h\_block;

int x2 = x1 + WID\_BLOCK, y2 = ZERO\_POINT + HEIGHT;

HBRUSH brush;

brush = CreateSolidBrush(RGB(70, 130, 180));

SelectObject(hdc, brush);

Rectangle(hdc, x1, y1, x2, y2);

DeleteObject(brush);

////подпись времени/////

HPEN pen\_text;

pen\_text = CreatePen(PS\_SOLID, 1, RGB(0, 0, 0));

SelectObject(hdc, pen\_text);

string tm = to\_string(time);

wstring stemp = std::wstring(tm.begin(), tm.end());

LPCWSTR timew = stemp.c\_str();

TextOut(hdc, x1+35, y1-20, timew, tm.size());

}

void DrawShake(int h\_block, int time)//блок сортировки шейкером

{

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

int x1 = ZERO\_POINT + 2 \* DIST + WID\_BLOCK, y1 = ZERO\_POINT + HEIGHT - h\_block;

int x2 = x1 + WID\_BLOCK, y2 = ZERO\_POINT + HEIGHT;

HBRUSH brush;

brush = CreateSolidBrush(RGB(70, 130, 180));

SelectObject(hdc, brush);

Rectangle(hdc, x1, y1, x2, y2);

DeleteObject(brush);

////подпись времени/////

HPEN pen\_text;

pen\_text = CreatePen(PS\_SOLID, 1, RGB(0, 0, 0));

SelectObject(hdc, pen\_text);

string tm = to\_string(time);

wstring stemp = std::wstring(tm.begin(), tm.end());

LPCWSTR timew = stemp.c\_str();

TextOut(hdc, x1 + 35, y1 - 20, timew, tm.size());

}

void DrawSelection(int h\_block, int time)//блок сортировки выбором

{

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

int x1 = ZERO\_POINT + 3 \* DIST + 2 \* WID\_BLOCK, y1 = ZERO\_POINT + HEIGHT - h\_block;

int x2 = x1 + WID\_BLOCK, y2 = ZERO\_POINT + HEIGHT;

HBRUSH brush;

brush = CreateSolidBrush(RGB(70, 130, 180));

SelectObject(hdc, brush);

Rectangle(hdc, x1, y1, x2, y2);

DeleteObject(brush);

////подпись времени/////

HPEN pen\_text;

pen\_text = CreatePen(PS\_SOLID, 1, RGB(0, 0, 0));

SelectObject(hdc, pen\_text);

string tm = to\_string(time);

wstring stemp = std::wstring(tm.begin(), tm.end());

LPCWSTR timew = stemp.c\_str();

TextOut(hdc, x1 + 35, y1 - 20, timew, tm.size());

}

void DrawInsertion(int h\_block, int time)//блок сортировки вставкой

{

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

int x1 = ZERO\_POINT + 4 \* DIST + 3 \* WID\_BLOCK, y1 = ZERO\_POINT + HEIGHT - h\_block;

int x2 = x1 + WID\_BLOCK, y2 = ZERO\_POINT + HEIGHT;

HBRUSH brush;

brush = CreateSolidBrush(RGB(70, 130, 180));

SelectObject(hdc, brush);

Rectangle(hdc, x1, y1, x2, y2);

DeleteObject(brush);

////подпись времени/////

HPEN pen\_text;

pen\_text = CreatePen(PS\_SOLID, 1, RGB(0, 0, 0));

SelectObject(hdc, pen\_text);

string tm = to\_string(time);

wstring stemp = std::wstring(tm.begin(), tm.end());

LPCWSTR timew = stemp.c\_str();

TextOut(hdc, x1 + 35, y1 - 20, timew, tm.size());

}

void DrawQuick(int h\_block, int time)//блок быстрой сортировки

{

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

int x1 = ZERO\_POINT + 5 \* DIST + 4 \* WID\_BLOCK, y1 = ZERO\_POINT + HEIGHT - h\_block;

int x2 = x1 + WID\_BLOCK, y2 = ZERO\_POINT + HEIGHT;

HBRUSH brush;

brush = CreateSolidBrush(RGB(70, 130, 180));

SelectObject(hdc, brush);

Rectangle(hdc, x1, y1, x2, y2);

DeleteObject(brush);

////подпись времени/////

HPEN pen\_text;

pen\_text = CreatePen(PS\_SOLID, 1, RGB(0, 0, 0));

SelectObject(hdc, pen\_text);

string tm = to\_string(time);

wstring stemp = std::wstring(tm.begin(), tm.end());

LPCWSTR timew = stemp.c\_str();

TextOut(hdc, x1 + 35, y1 - 20, timew, tm.size());

}

void DrawGraphics(int \*arrTime)

{

system("color F0");

int maxTime = arrTime[0];

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

{

if (arrTime[i] > maxTime)

{

maxTime = arrTime[i];

}

}

int countNum = -1;

int num = maxTime;

while (num > 0)

{

num /= 10;

countNum++;

}

int max = floor(maxTime / pow(10, countNum) + 1) \* pow(10, countNum);

int section = max / 5;

double scale = static\_cast<double>(H\_LINES)/section;

int time\_bubble = arrTime[0] \* scale;

int time\_shake = arrTime[1] \* scale;

int time\_selection = arrTime[2] \*scale;

int time\_incertion = arrTime[3] \*scale;

int time\_quick = arrTime[4] \* scale;

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

RECT window = {};

DrawAxes(max);

DrawBubble(time\_bubble, arrTime[0]);

DrawShake(time\_shake, arrTime[1]);

DrawSelection(time\_selection, arrTime[2]);

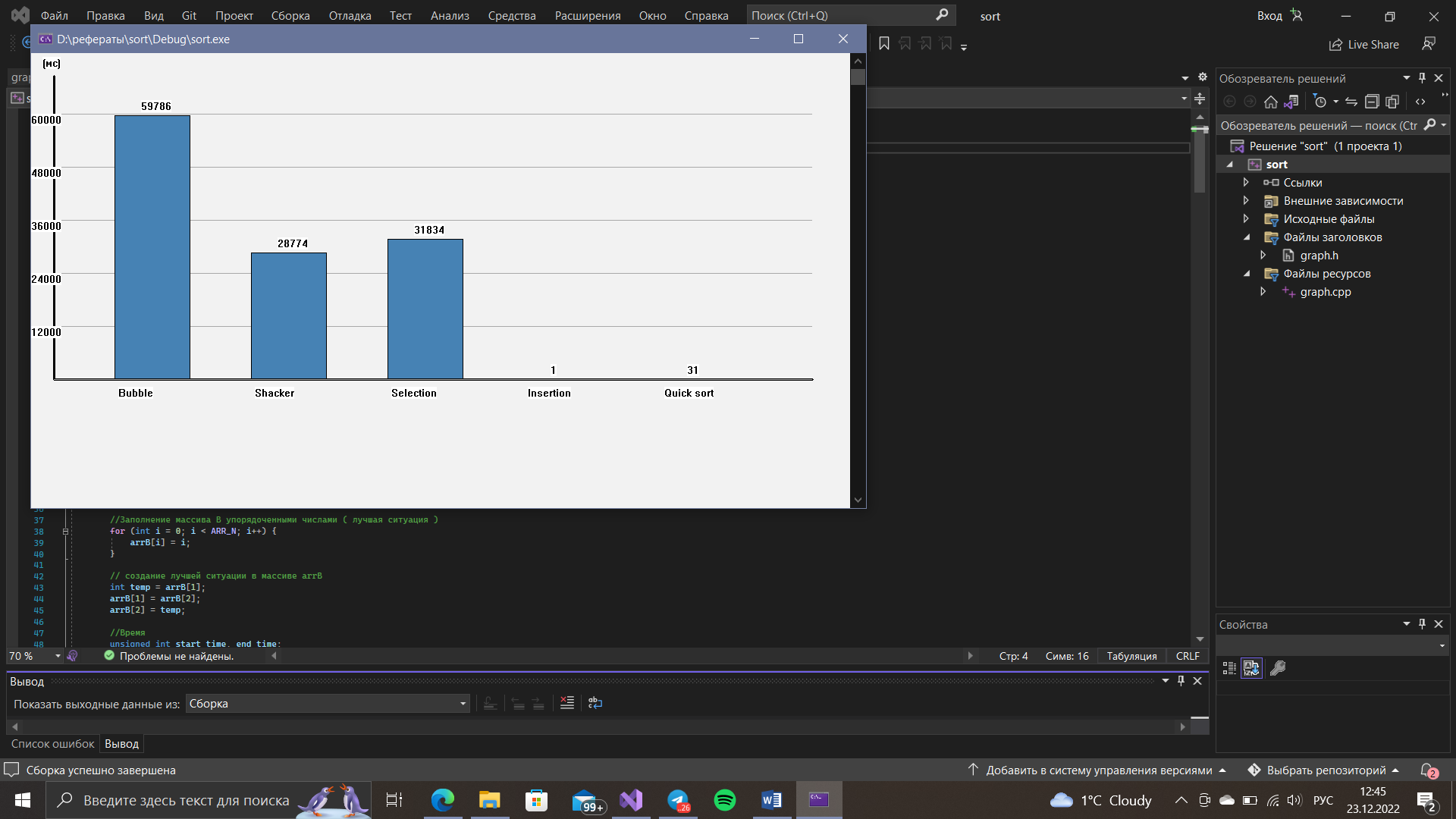
DrawInsertion(time\_incertion, arrTime[3]);

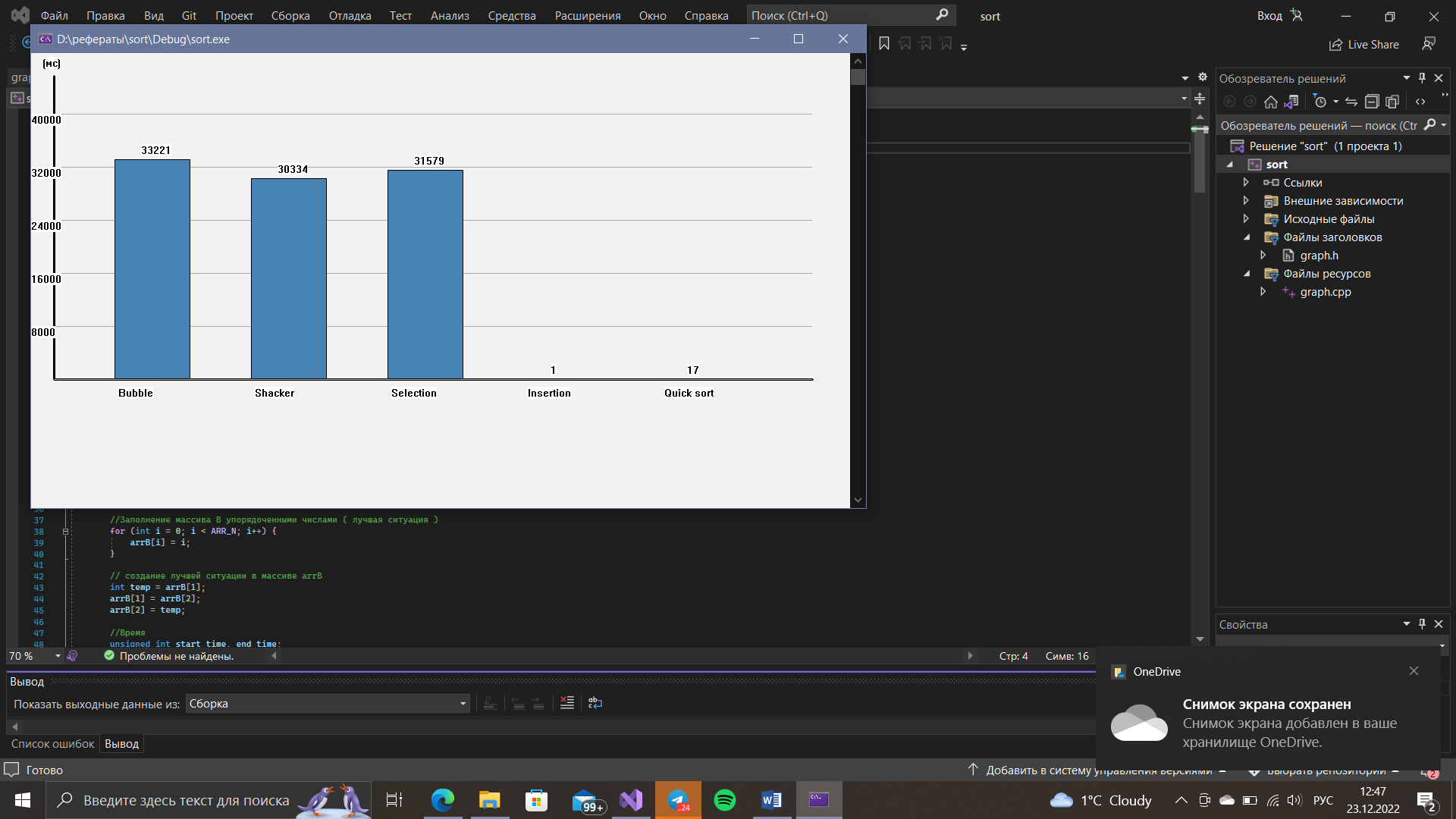
DrawQuick(time\_quick,arrTime[4]);

ReleaseDC(hwnd, hdc);

cin.ignore();

}

Результат действий модуля graph: 



**ВЫВОД**

По данным графиков видно, что лучше всех себя показала сортировка вставками(insertion) в обоих случаях, время ее выполнения занимает 1мс. Не настолько быстрой, но все же достаточно эффективной оказалась быстрая сортировка(quick), ее время не превосходит 35мс при случайной ситуации. Самая медленная сортировка – сортировка пузырьком(bubble). Сортировки шейкером(shacker) и выбором(selection) показали не самые быстрые, примерно одинаковые результаты в обеих ситуациях. Таким образом, мы видим, что выбор сортировок довольно велик, но разница в их производительности существенна.