Выполнил

студент КТбо1-4 \_\_\_\_\_\_\_\_\_\_\_\_\_ А. А. Воронов

Принял

ст. преподаватель каф. САПР \_\_\_\_\_\_\_\_\_\_\_\_\_ В. И. Данильченко

Таганрог 2024

ЛАБОРАТОРНАЯ РАБОТА №6

по дисциплине «Алгоритмизация и программирование»

**«Алгоритмы сортировки»**

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

(ФГАОУ ВО «ЮФУ»)

1. **Цель лабораторной работы:** Изучить алгоритмы сортировки на примере сортировки пузырьком, расческой и шейкерной сортировки.
2. **Задача:** Написать программу, которая считывает данные из исходного файла и сортирует их тремя разными способами: сортировкой пузырьком, сортировкой расческой и шейкерной сортировкой. Программа должна измерить время работы каждой сортировки и построить график алгоритмической сложности для каждой сортировки. Необходимо сравнить полученные временные результаты с теоретической оценкой сложности алгоритма.
3. **Ход работы:** я создал CLR проект в Visual Studio 2022. Далее я организовал 5 файлов, которые составили структуру многофайлового проекта: заголовочные файлы «Timr.h», «AlgSort.h», конструктор Windows Forms «AlgSort.h(конструктор)», а также 2 основных файла — «Timr.cpp» и «AlgSort.cpp». Внутри «Timr.cpp» я реализовал программный код сортировок, которые требуется проанализировать. Перед выполнением и после выполнения каждой сортировки я замерял текущее время от начала работы программы в миллисекундах при помощи библиотеки «ctime». Для удобства обработки входных массивов чисел я использовал контейнер «vector».

В файле «Timr.cpp» сперва я симулировал ручное заполнение файла числами — сделал генератор случайных чисел и поместил их в файл. Затем я считал из файла числа и записал их в вектор. При запуске сортировки алгоритм каждый раз копировал вектор в еще один массив и сортировал его отдельно — вектор оставался неизменным и передавался в следующий алгоритм сортировки. В конце программа возвращает массив с тремя числами. Каждое число — время сортировки для каждого алгоритма.

В файле «AlgSort.cpp» реализовано построение графика. Массив из файла «Timr.cpp» передается в «AlgSort.cpp». В программе реализован цикл — с каждой итерацией увеличивается количество исходных элементов в массиве для сортировки. Это значение передается назад в «Timr.cpp» и производится сортировка. На основе полученных данных о времени строится график.

**Программный код «Timr.cpp»:**

#include <iostream>// лаба: несколько сортировок, посмореть время работы каждой и построить график по времени, посмотреть асимптотику

#include <vector>

#include <ctime>

#include <fstream>

#include <string>

#include "Timr.h"

using namespace std;

void swap(int\* xp, int\* yp);

void BubbleSort(vector<int>& arr, int n);

void CombSort(vector<int>& arr, int n);

void ShakeSort(vector<int>& arr, int n);

void reader(fstream& input\_f, vector<int>& arr, int n);

void swap(int\* xp, int\* yp) {

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

void BubbleSort(vector<int>& arr, int n) {

int i, j;

for (i = 0; i < n - 1; i++)

for (j = 0; j < n - i - 1; j++)

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

swap(&arr[j], &arr[j + 1]);

};

void CombSort(vector<int>& arr, int n) {

int sm;

float shrink = 1.3;

int gap = n;

bool sorted = false;

while (!sorted) {

gap = floor(gap / shrink); // ??? floor

if (gap <= 1) {

sorted = true;

gap = 1;

}

for (int i = 0; i < n - gap; i++) {

sm = gap + i;

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

int temp = arr[i];

arr[i] = arr[sm];

arr[sm] = temp;

sorted = false;

}

}

}

};

void ShakeSort(vector<int>& arr, int n) {

bool swapped = true;

int start = 0;

int end = n - 1;

while (swapped) {

swapped = false;

for (int i = start; i < end; ++i) {

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

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

swapped = true;

}

}

if (!swapped)

break;

swapped = false;

--end;

for (int i = end - 1; i >= start; --i)

{

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

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

swapped = true;

}

}

++start;

}

}

// From input file to vector

void reader(fstream& input\_f, vector<int>& arr, int n) {

string str;

arr = {};

input\_f.seekg(0); //!!!!!!!!!!

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

getline(input\_f, str);

int tmp = stoi(str);

arr.push\_back(tmp);

}

}

double\* Timr(double n) {

srand(time(0));

double times[3] = {};

vector<int> arr;

arr.reserve(n);

fstream input\_f;

input\_f.open("input.txt", ios::out | ios::in);

// File random generation

for (int i = 0; i < n; i++) { input\_f << rand() % 1000 << endl; }

// Base array

reader(input\_f, arr, n);

// Bubble Sort

reader(input\_f, arr, n);

double start1 = clock();

BubbleSort(arr, arr.size());

double end1 = clock();

double duration1 = end1 - start1;

times[0] = duration1;

// Shake Sort

reader(input\_f, arr, n);

double start2 = clock();

ShakeSort(arr, arr.size());

double end2 = clock();

double duration2 = end2 - start2;

times[1] = duration2;

// Comb Sort

reader(input\_f, arr, n);

double start3 = clock();

CombSort(arr, arr.size());

double end3 = clock();

double duration3 = end3 - start3;

times[2] = duration3;

input\_f.close();

return times;

}

**Программный код «AlgSort.cpp»:**

#include "AlgSort.h"

#include "Timr.h"

#include <cmath>

using namespace System;

using namespace System::Windows::Forms;

[STAThreadAttribute]

void main(array<String^>^ args) {

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

S2L2SortGraphic::AlgSort form;

Application::Run(% form);

}

System::Void S2L2SortGraphic::AlgSort::построитьГрафикToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

a = 1;

b = pow(10, 3);

h = 100;

x = a;

this->chart->Series[0]->Points->Clear();

this->chart->Series[1]->Points->Clear();

this->chart->Series[2]->Points->Clear();

while (x <= b) {

double\* Y = Timr(x);

y = Y[0];

this->chart->Series[0]->Points->AddXY(x, y);

y = Y[1];

this->chart->Series[1]->Points->AddXY(x, y);

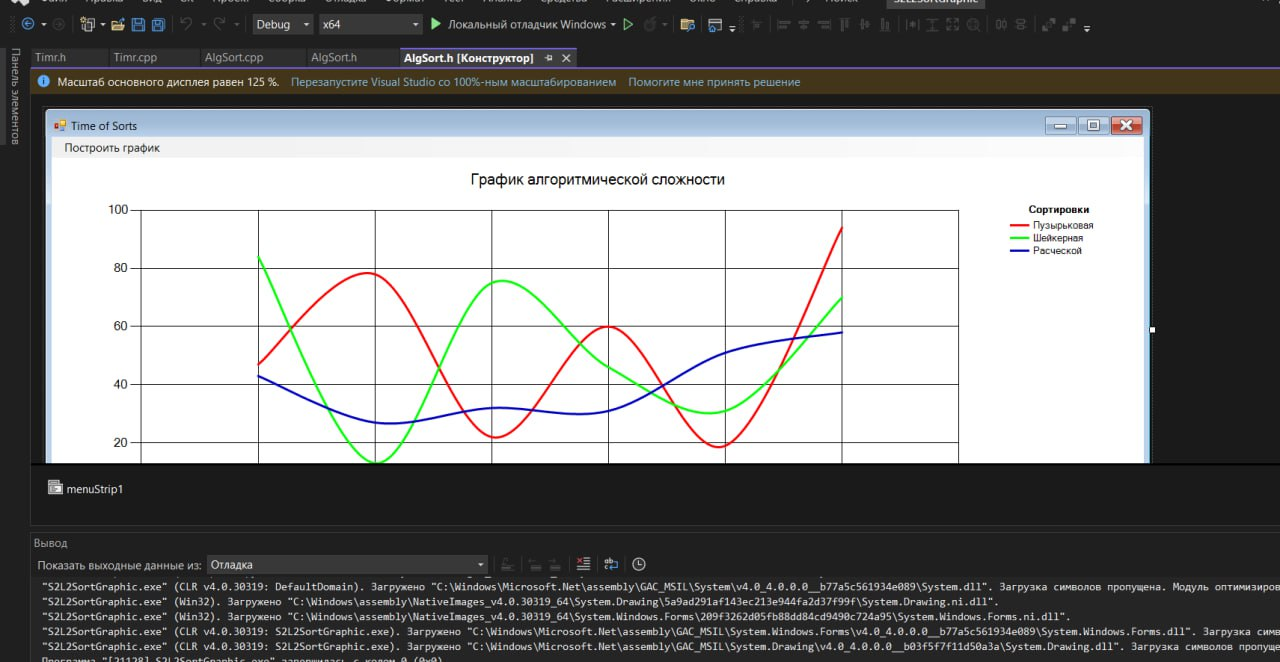
y = Y[2];

this->chart->Series[2]->Points->AddXY(x, y);

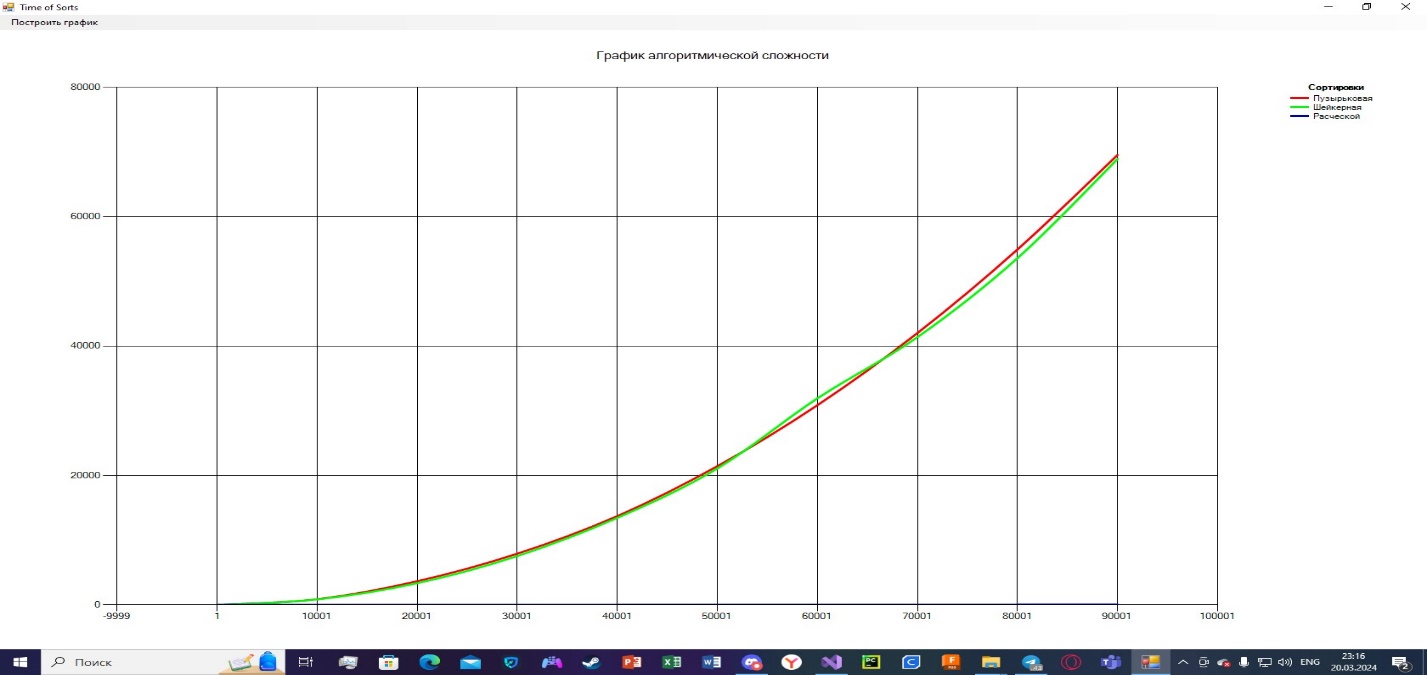
x += h;

}

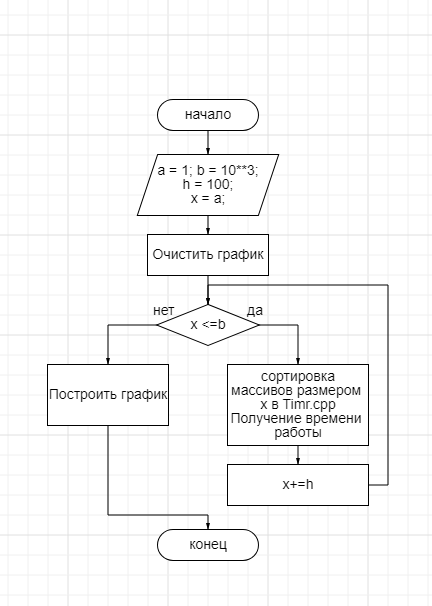
}**Вид «AlgSort.h(конструктор)»**



**Процесс работы программы:**



1. **Блок-схема:**

****

**5. Вывод:** в ходе работы были выполнены все цели и задачи, были реализованы алгоритмы сортировки пузырьком, шейкерной сортировки и сортировки расческой. В проекте использованы Windows Forms. Построен график алгоритмической сложности, который совпадает с теоритической.

**6. Источники:**

<https://sortvisualizer.com>

<https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?view=netdesktop-8.0>

https://learn.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170