Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа №15

По дисциплине «Основы алгоритмизации и программирования»

На тему «**Анализ алгоритмов сортировок**»

Выполнил:

Студент 1 курса 10 группы

Мандрик Алексей Иванович

Преподаватель: асс. Андронова М.В.

Минск, 2024

В соответствии со своим вариантом написать программу для ***сортировок*** массивов указанными в таблице методами. Исходные массивы заполняются случайными числами.

Определить зависимость времени выполнения алгоритмов от количества элементов для каждого из алгоритмов. Выполнить моделирование для массивов размером 1000, 2000, 3000, 4000, 5000 (в зависимости от быстродействия компьютера размеры массивов можно увеличивать).

Произвести сравнение эффективности алгоритмов (построить графики в приложении Excel

**Вариант 8:**

****

**Код программы:**

#include <iostream>

#include <vector>

#include <algorithm>

#include <ctime>

#include <chrono>

#include <random>

// Функция для сортировки слиянием

void mergeSort(std::vector<int>& arr, int left, int right) {

if (left >= right) return;

int mid = left + (right - left) / 2;

mergeSort(arr, left, mid);

mergeSort(arr, mid + 1, right);

std::vector<int> temp(right - left + 1);

int i = left, j = mid + 1, k = 0;

while (i <= mid && j <= right) {

if (arr[i] <= arr[j]) {

temp[k++] = arr[i++];

}

else {

temp[k++] = arr[j++];

}

}

while (i <= mid) temp[k++] = arr[i++];

while (j <= right) temp[k++] = arr[j++];

for (int p = 0; p < k; ++p) {

arr[left + p] = temp[p];

}

}

// Функция для сортировки пузырьком

void bubbleSort(std::vector<int>& arr) {

int n = arr.size();

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

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

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

std::swap(arr[j], arr[j + 1]);

}

}

}

}

// Функция для генерации случайных чисел в диапазоне [min, max]

int getRandomNumber(int min, int max) {

static std::mt19937 mt(std::time(nullptr));

std::uniform\_int\_distribution<int> dist(min, max);

return dist(mt);

}

int main() {

setlocale(LC\_ALL, "Rus");

// Размеры массивов для моделирования времени выполнения

std::vector<int> sizes = { 1000, 2000, 3000, 4000, 5000 };

for (int size : sizes) {

std::vector<int> A(size), B(size), C;

// Заполнение массивов случайными числами

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

A[i] = getRandomNumber(1, 10000);

B[i] = getRandomNumber(1, 10000);

}

// Объединение массивов A и B в массив C

C.insert(C.end(), A.begin(), A.end());

C.insert(C.end(), B.begin(), B.end());

// Сортировка массива C слиянием

auto start\_merge = std::chrono::steady\_clock::now();

mergeSort(C, 0, C.size() - 1);

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

std::chrono::duration<double> merge\_time = end\_merge - start\_merge;

// Сортировка массива C пузырьком

auto start\_bubble = std::chrono::steady\_clock::now();

bubbleSort(C);

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

std::chrono::duration<double> bubble\_time = end\_bubble - start\_bubble;

// Вывод времени выполнения для каждого алгоритма и размера массива

std::cout << "Время выполнения для размера " << size << " элементов:" << std::endl;

std::cout << "Сортировка слиянием: " << merge\_time.count() << " секунд" << std::endl;

std::cout << "Сортировка пузырьком: " << bubble\_time.count() << " секунд" << std::endl;

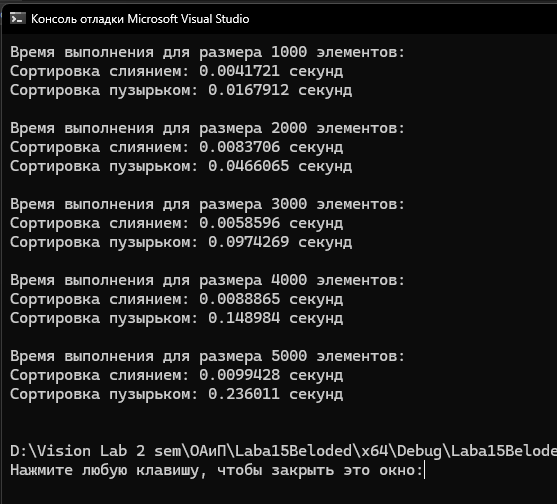
std::cout << std::endl;

}

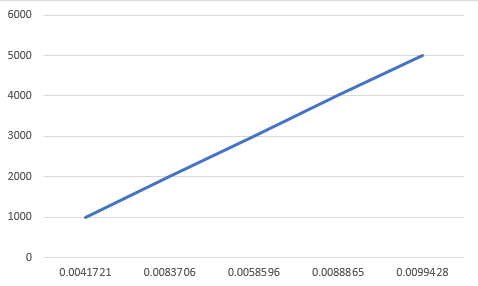
return 0;

}

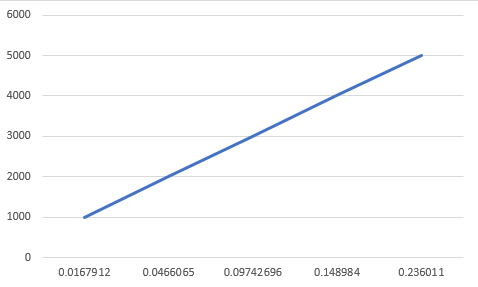
**Результат выполнения:**



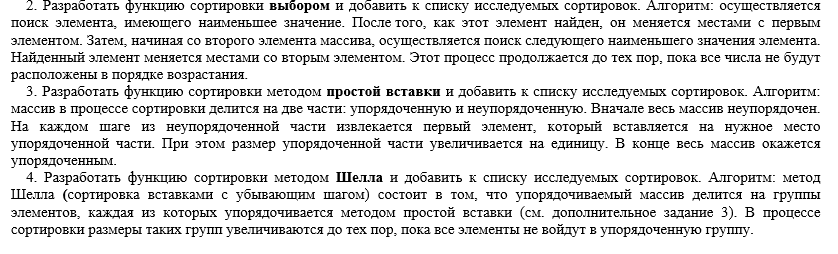
**Сортировка слиянием**

****

**Сортировка пузырьком**

****

**Дополнительные задания:**

****

**Код программы:**

#include <iostream>

#include <vector>

#include <algorithm>

#include <ctime>

#include <chrono>

#include <random>

// Функция для генерации случайных чисел в диапазоне [min, max]

int getRandomNumber(int min, int max) {

static std::mt19937 mt(std::time(nullptr));

std::uniform\_int\_distribution<int> dist(min, max);

return dist(mt);

}

// Функция для сортировки выбором

void selectionSort(std::vector<int>& arr) {

int n = arr.size();

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

int minIndex = i;

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

if (arr[j] < arr[minIndex]) {

minIndex = j;

}

}

std::swap(arr[i], arr[minIndex]);

}

}

// Функция для сортировки методом простой вставки

void insertionSort(std::vector<int>& arr) {

int n = arr.size();

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

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

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

j = j - 1;

}

arr[j + 1] = key;

}

}

// Функция для сортировки методом Шелла

void shellSort(std::vector<int>& arr) {

int n = arr.size();

for (int gap = n / 2; gap > 0; gap /= 2) {

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

int temp = arr[i];

int j;

for (j = i; j >= gap && arr[j - gap] > temp; j -= gap) {

arr[j] = arr[j - gap];

}

arr[j] = temp;

}

}

}

int main() {

setlocale(LC\_ALL, "Rus");

// Размеры массивов для моделирования времени выполнения

std::vector<int> sizes = { 1000, 2000, 3000, 4000, 5000 };

for (int size : sizes) {

std::vector<int> A(size);

// Заполнение массива случайными числами

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

A[i] = getRandomNumber(1, 10000);

}

// Сортировка выбором

auto start\_selection = std::chrono::steady\_clock::now();

selectionSort(A);

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

std::chrono::duration<double> selection\_time = end\_selection - start\_selection;

// Сортировка методом простой вставки

auto start\_insertion = std::chrono::steady\_clock::now();

insertionSort(A);

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

std::chrono::duration<double> insertion\_time = end\_insertion - start\_insertion;

// Сортировка методом Шелла

auto start\_shell = std::chrono::steady\_clock::now();

shellSort(A);

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

std::chrono::duration<double> shell\_time = end\_shell - start\_shell;

// Вывод времени выполнения для каждого алгоритма и размера массива

std::cout << "Время выполнения для размера " << size << " элементов:" << std::endl;

std::cout << "Сортировка выбором: " << selection\_time.count() << " секунд" << std::endl;

std::cout << "Сортировка методом простой вставки: " << insertion\_time.count() << " секунд" << std::endl;

std::cout << "Сортировка методом Шелла: " << shell\_time.count() << " секунд" << std::endl;

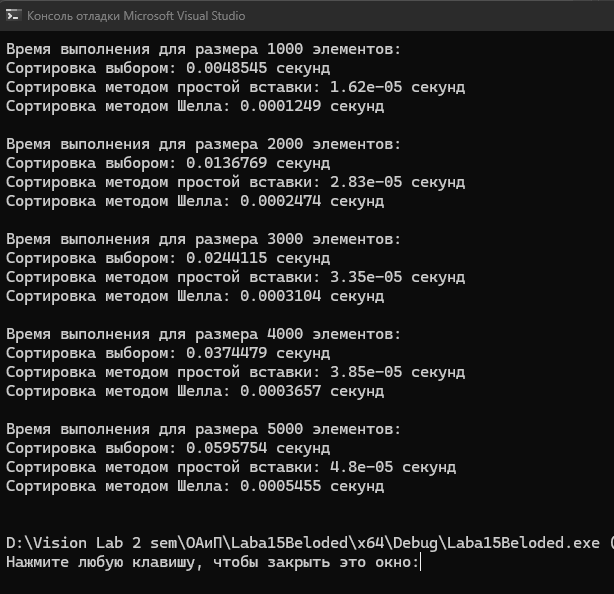
std::cout << std::endl;

}

return 0;

}

**Результат выполнения:**

****