Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

Лабораторная работа 15

По дисциплине “Основы алгоритмизации и программирования”

На тему “Анализ алгоритмов сортировок ”

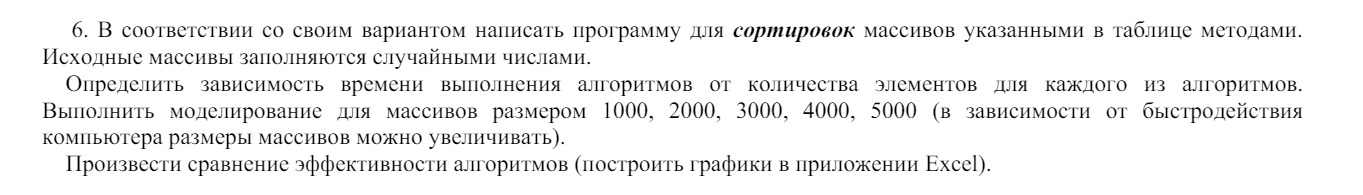
Выполнил:

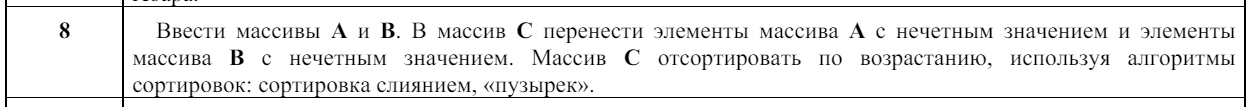
Студент 1 курса 6 группы

Кравченко Сергей Сергеевич

Преподаватель: асс. Андронова М.В.

2024





#include <iostream>

#include <vector>

#include <ctime>

#include <algorithm>

using namespace std;

// Генерация случайных чисел в диапазоне [min, max]

int getRandomNumber(int min, int max) {

return min + rand() % (max - min + 1);

}

// Заполнение массива случайными числами

void fillArrayWithRandomNumbers(vector<int>& arr, int size, int min, int max) {

arr.clear();

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

arr.push\_back(getRandomNumber(min, max));

}

}

// Функция для вывода массива

void printArray(const vector<int>& arr) {

for (int num : arr) {

cout << num << " ";

}

cout << endl;

}

// Функция для сортировки массива методом слияния

void mergeSort(vector<int>& arr) {

if (arr.size() <= 1) return;

vector<int> left(arr.begin(), arr.begin() + arr.size() / 2);

vector<int> right(arr.begin() + arr.size() / 2, arr.end());

mergeSort(left);

mergeSort(right);

merge(left.begin(), left.end(), right.begin(), right.end(), arr.begin());

}

// Функция для сортировки массива методом пузырька

void bubbleSort(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]);

}

}

}

}

int main() {

setlocale(LC\_ALL, "RUS");

// Инициализация генератора случайных чисел

srand(time(0));

// Размеры массивов для моделирования

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

for (int size : sizes) {

// Создание массивов А и В

vector<int> A, B;

fillArrayWithRandomNumbers(A, size, 1, 1000);

fillArrayWithRandomNumbers(B, size, 1, 1000);

// Создание массива C из нечетных элементов массивов А и В

vector<int> C;

for (int num : A) {

if (num % 2 != 0) {

C.push\_back(num);

}

}

for (int num : B) {

if (num % 2 != 0) {

C.push\_back(num);

}

}

// Сортировка массива C слиянием и вывод времени выполнения

vector<int> mergeSortC = C;

clock\_t startTime = clock();

mergeSort(mergeSortC);

clock\_t endTime = clock();

double mergeSortTime = double(endTime - startTime) / CLOCKS\_PER\_SEC;

// Сортировка массива C пузырьком и вывод времени выполнения

vector<int> bubbleSortC = C;

startTime = clock();

bubbleSort(bubbleSortC);

endTime = clock();

double bubbleSortTime = double(endTime - startTime) / CLOCKS\_PER\_SEC;

// Вывод результатов для текущего размера массива

cout << "Размер массива: " << size << endl;

cout << "Время сортировки слиянием: " << mergeSortTime << " секунд" << endl;

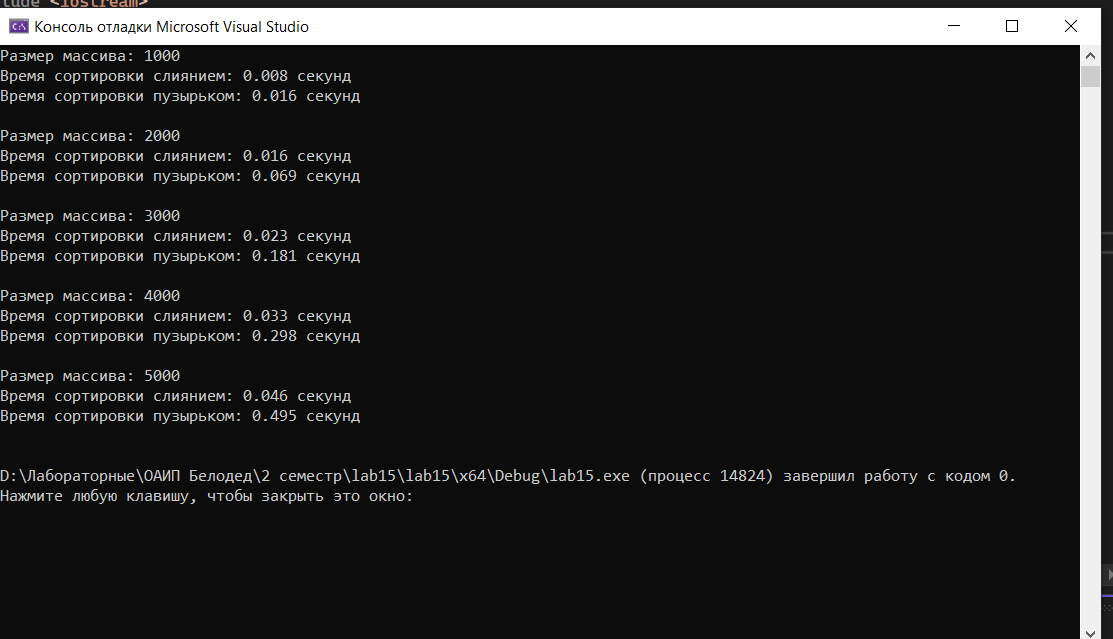
cout << "Время сортировки пузырьком: " << bubbleSortTime << " секунд" << endl;

cout << endl;

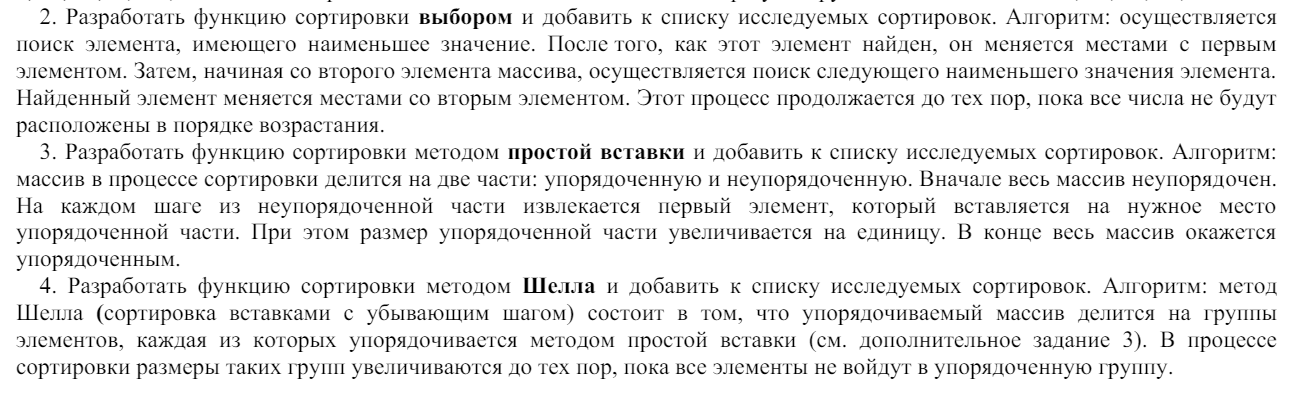
}

return 0;

}



Дополнительные задания



#include <iostream>

#include <chrono> // Для измерения времени выполнения сортировок

#include <cstdlib> // Для функций rand() и srand()

#include <ctime> // Для инициализации генератора случайных чисел

using namespace std;

// Функция для сортировки массива методом выбора

void selectionSort(int arr[], int n) {

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;

}

}

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

}

}

// Функция для сортировки массива методом вставки

void insertionSort(int arr[], int n) {

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 массива методом Шелла

void shellSort(int arr[], int n) {

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");

int n;

cout << "Введите размер массива A: ";

cin >> n;

// Генерируем массив A

int\* A = new int[n]; // Выделение памяти под массив A

srand(time(0)); // Инициализация генератора случайных чисел

cout << "Массив A:\n";

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

A[i] = rand() % 100; // Генерация случайных чисел от 0 до 99

cout << A[i] << " ";

}

cout << endl;

// Находим максимальный элемент в массиве A и его индекс

int maxElement = A[0];

int maxIndex = 0;

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

if (A[i] > maxElement) {

maxElement = A[i];

maxIndex = i;

}

}

// Создаем массив B и переносим нечетные элементы правее максимального элемента из массива A

int\* B = new int[n]; // Выделение памяти под массив B

int countB = 0; // Переменная для подсчета элементов в массиве B

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

if (i % 2 != 0) {

B[countB] = A[i];

countB++;

}

}

// Измеряем время выполнения сортировки выбором

auto start = chrono::high\_resolution\_clock::now(); // Начало измерения времени

selectionSort(B, countB);

auto end = chrono::high\_resolution\_clock::now(); // Конец измерения времени

chrono::microseconds duration = chrono::duration\_cast<chrono::microseconds>(end - start); // Вычисление продолжительности времени в микросекундах

cout << "Сортировка методом выбора заняла: " << duration.count() << " микросекунд" << endl;

// Выводим отсортированный массив B

cout << "Массив B (отсортированный методом выбора):\n";

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

cout << B[i] << " ";

}

cout << endl;

// Копируем массив A в массив B и переносим нечетные элементы правее максимального элемента из массива A

countB = 0; // Сбрасываем счетчик элементов в массиве B

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

if (i % 2 != 0) {

B[countB] = A[i];

countB++;

}

}

// Измеряем время выполнения сортировки методом простой вставки

start = chrono::high\_resolution\_clock::now(); // Начало измерения времени

insertionSort(B, countB);

end = chrono::high\_resolution\_clock::now(); // Конец измерения времени

duration = chrono::duration\_cast<chrono::microseconds>(end - start); // Вычисление продолжительности времени в микросекундах

cout << "Сортировка методом простой вставки заняла: " << duration.count() << " микросекунд" << endl;

// Выводим отсортированный массив B

cout << "Массив B (отсортированный методом простой вставки):\n";

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

cout << B[i] << " ";

}

cout << endl;

// Копируем массив A в массив B и переносим нечетные элементы правее максимального элемента из массива A

countB = 0; // Сбрасываем счетчик элементов в массиве B

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

if (i % 2 != 0) {

B[countB] = A[i];

countB++;

}

}

// Измеряем время выполнения сортировки методом Шелла

start = chrono::high\_resolution\_clock::now(); // Начало измерения времени

shellSort(B, countB);

end = chrono::high\_resolution\_clock::now(); // Конец измерения времени

duration = chrono::duration\_cast<chrono::microseconds>(end - start); // Вычисление продолжительности времени в микросекундах

cout << "Сортировка методом Шелла заняла: " << duration.count() << " микросекунд" << endl;

// Выводим отсортированный массив B

cout << "Массив B (отсортированный методом Шелла):\n";

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

cout << B[i] << " ";

}

cout << endl;

// Освобождаем выделенную память для массивов A и B

delete[] A;

delete[] B;

return 0;

}

