Банкузов Михаил 7 группа Лабораторная работа №16 Вариант 1



#include <iostream>

#include <vector>

#include <algorithm>

#include <ctime>

// Функция для сортировки массива методом пузырька

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 сортировки массива 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--;

}

arr[j + 1] = key;

}

}

int main() {

setlocale(LC\_ALL, "Russian");

int n;

std::cout << "Введите размер массива: ";

std::cin >> n;

// Генерация случайных чисел

std::srand(static\_cast<unsigned int>(std::time(nullptr)));

std::vector<int> arrA(n);

std::cout << "Массив A (случайные числа): ";

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

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

std::cout << arrA[i] << " ";

}

std::cout << std::endl;

// Создание массива B с элементами, имеющими четные индексы из массива A

std::vector<int> arrB;

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

if (i % 2 == 0) { // Проверка на четность индекса

arrB.push\_back(arrA[i]);

}

}

// Вывод массива B на экран

std::cout << "Массив B (с элементами с четными индексами): ";

for (int num : arrB) {

std::cout << num << " ";

}

std::cout << std::endl;

// Сортировка массива B методом пузырька

std::vector<int> arrB\_bubble = arrB; // Копия массива B для сортировки пузырьком

std::clock\_t startBubbleSort = std::clock();

bubbleSort(arrB\_bubble);

std::clock\_t endBubbleSort = std::clock();

// Вывод отсортированного массива B методом пузырька на экран

std::cout << "Отсортированный массив B (пузырек): ";

for (int num : arrB\_bubble) {

std::cout << num << " ";

}

std::cout << std::endl;

// Сортировка массива B методом простой вставки

std::vector<int> arrB\_insertion = arrB; // Копия массива B для сортировки простой вставкой

std::clock\_t startInsertionSort = std::clock();

insertionSort(arrB\_insertion);

std::clock\_t endInsertionSort = std::clock();

// Вывод отсортированного массива B методом простой вставки на экран

std::cout << "Отсортированный массив B (сортировка простой вставкой): ";

for (int num : arrB\_insertion) {

std::cout << num << " ";

}

std::cout << std::endl;

// Вычисление времени выполнения сортировок

double timeBubbleSort = static\_cast<double>(endBubbleSort - startBubbleSort) / CLOCKS\_PER\_SEC;

double timeInsertionSort = static\_cast<double>(endInsertionSort - startInsertionSort) / CLOCKS\_PER\_SEC;

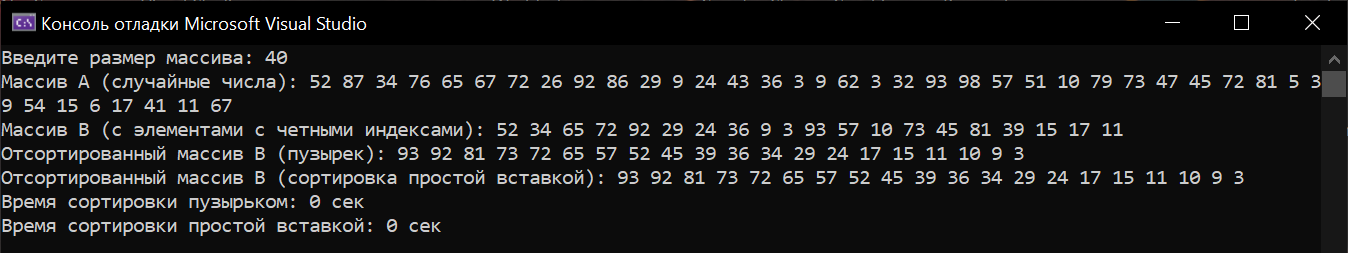
// Вывод времени выполнения сортировок на экран

std::cout << "Время сортировки пузырьком: " << timeBubbleSort << " сек" << std::endl;

std::cout << "Время сортировки простой вставкой: " << timeInsertionSort << " сек" << std::endl;

return 0;

}



Дополнительные задания

Вариант 10



#include <iostream>

#include <vector>

#include <algorithm>

#include <chrono>

#include <random>

// Функция для генерации случайных чисел

void generateRandomArray(std::vector<int>& arr, int size) { //функция для генерации случайных чисел

std::random\_device rd; // Источник энтропии для генератора случайных чисел

std::mt19937 gen(rd()); // Генератор случайных чисел

std::uniform\_int\_distribution<> dis(1, 100); // Равномерное распределение случайных чисел в диапазоне [1, 100]

arr.clear(); // Очистка массива

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

arr.push\_back(dis(gen));// Генерация случайного числа и добавление его в массив

}

}

// Функция для сортировки массива методом пузырька

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

for (int i = 0; i < arr.size() - 1; i++) {

for (int j = 0; j < arr.size() - i - 1; j++) {

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

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

}

}

}

}

// Функция для сортировки массива методом простой вставки

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

for (int i = 1; i < arr.size(); i++) {

int key = arr[i];

int j = i - 1;

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

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

j--;

}

arr[j + 1] = key;

}

}

// Функция для создания массива С

void createArrayC(const std::vector<int>& arrA, const std::vector<int>& arrB, std::vector<int>& arrC) {

arrC.clear();

int maxB = \*std::max\_element(arrB.begin(), arrB.end());

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

if (arrA[i] > maxB) {

arrC.push\_back(arrA[i]);

}

}

}

int main() {

setlocale(LC\_ALL, "Rus");

const int sizes[] = { 10, 100, 1000 }; // размеры массивов для моделирования

const int numSizes = sizeof(sizes) / sizeof(int); // количество размеров массивов для моделирования

std::vector<int> arrA, arrB, arrC; // массивы для моделирования

// Моделирование сортировки массивов указанными методами

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

generateRandomArray(arrA, sizes[i]);// генерация случайных массивов

generateRandomArray(arrB, sizes[i]);

auto start = std::chrono::high\_resolution\_clock::now(); // начало отсчета времени

bubbleSort(arrA); // сортировка массива методом пузырька

auto end = std::chrono::high\_resolution\_clock::now(); // конец отсчета времени

std::chrono::duration<double> diff = end - start; // вычисление разницы между началом и концом отсчета времени

std::cout << "Сортировка пузырьком для массива размером " << sizes[i] << " заняла " << diff.count() << " секунд." << std::endl; // вывод времени работы сортировки

start = std::chrono::high\_resolution\_clock::now(); // начало отсчета времени

insertionSort(arrB); // сортировка массива методом простой вставки

end = std::chrono::high\_resolution\_clock::now(); // конец отсчета времени

diff = end - start; // вычисление разницы между началом и концом отсчета времени

std::cout << "Сортировка втавкой для массива размером " << sizes[i] << " заняла " << diff.count() << " секунд." << std::endl;// вывод времени работы сортировки

}

// Создание и сортировка массива C

generateRandomArray(arrA, 10);

generateRandomArray(arrB, 10);

createArrayC(arrA, arrB, arrC);

std::sort(arrC.begin(), arrC.end(), std::greater<int>());

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

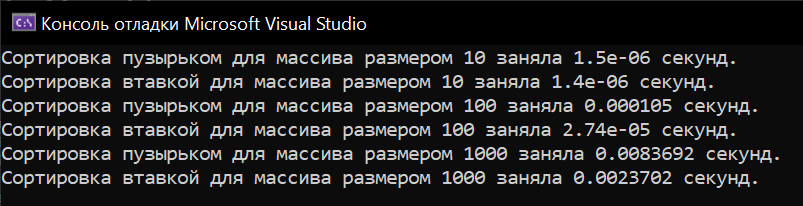
std::cout << arrC[i] << " ";

}

std::cout << std::endl;

return 0;

}



Вариант 5



#include <iostream>

#include <chrono>

#include <random>

using namespace std;

void bubbleSort(int arr[], int n) { //пузырьковая сортировка

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

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

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

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

}

void selectionSort(int arr[], int n) { //функция сортировк простой вставки

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

int min\_idx = i;

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

if (arr[j] < arr[min\_idx])

min\_idx = j;

swap(arr[min\_idx], arr[i]);

}

}

int main() {

setlocale(LC\_ALL, "Rus");

std::random\_device rd; // Источник энтропии для генератора случайных чисел

std::mt19937 gen(rd()); // Генератор случайных чисел

std::uniform\_int\_distribution<> dist(1, 100); // Равномерное распределение случайных чисел в диапазоне [1, 100]

int n; //размер массива

cout << "Введите размер массива: ";

cin >> n;

int\* arrA = new int[n]; //создание массива A i B

int\* arrB = new int[n];

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

arrA[i] = dist(gen);

arrB[i] = dist(gen);

}

//выводим массивы А и В

cout << "Массив A: ";

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

cout << arrA[i] << " ";

cout << endl;

cout << "Массив B: ";

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

cout << arrB[i] << " ";

cout << endl;

//создаем массив С и заполняем его четными числами из А и нечетными из В

int\* arrC = new int[n];

int index = 0;

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

if (arrA[i] % 2 == 0) {

arrC[index] = arrA[i];

index++;

}

}

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

if (arrB[i] % 2 == 1) {

arrC[index] = arrB[i];

index++;

}

}

cout << "Массив C: ";

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

cout << arrC[i] << " ";

cout << endl;

//оценка времени работы алгоритмов сортировки

auto start = std::chrono::high\_resolution\_clock::now();

bubbleSort(arrC, n);

auto end = std::chrono::high\_resolution\_clock::now();

auto duration = std::chrono::duration\_cast<std::chrono::nanoseconds>(end - start);

cout << "Время сортировки пузырьком: " << duration.count() << "ns" << endl;

start = std::chrono::high\_resolution\_clock::now();

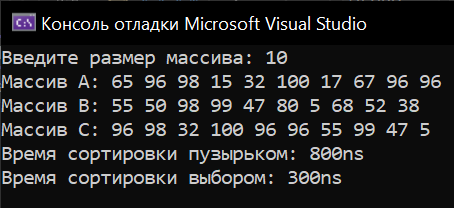
selectionSort(arrC, n);

end = std::chrono::high\_resolution\_clock::now();

duration = std::chrono::duration\_cast<std::chrono::nanoseconds>(end - start);

cout << "Время сортировки выбором: " << duration.count() << "ns" << endl;

}



Вариант 6



#include <iostream>

#include <algorithm>

#include <chrono>

#include <random>

using namespace std;

void copyEven(int\* a, int n, int\* b, int& m) { //функция копирования четных элементов в другой массив

m = 0;

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

if (a[i] % 2 == 0) {

b[m] = a[i];

m++;

}

}

}

void shellSort(int\* a, int n) { //функция сортировки Шелла

int h = 1;

while (h < n / 3) {

h = 3 \* h + 1;

}

while (h >= 1) {

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

for (int j = i; j >= h && a[j] < a[j - h]; j -= h) {

swap(a[j], a[j - h]);

}

}

h /= 3;

}

}

void selectionSort(int\* a, int n) {

{ //функция сортировк простой вставки

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

int minIndex = i;

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

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

minIndex = j;

}

}

if (minIndex != i) {

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

}

}

}

}

int main() {

setlocale(LC\_ALL, "Rus");

const int MAX\_N = 100000;

int a[MAX\_N], b[MAX\_N];

int n, m;

// Ввод размера массивов

cout << "Введите размер массива: ";

cin >> n;

// Заполнение исходного массива случайными числами

std::random\_device rd; // Источник энтропии для генератора случайных чисел

std::mt19937 gen(rd()); // Генератор случайных чисел

std::uniform\_int\_distribution<> dist(1, 100); // Равномерное распределение случайных чисел в диапазоне [1, 100]

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

a[i] = dist(gen);

}

//вывести исходный массив

cout << "Исходный массив: ";

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

cout << a[i] << " ";

}

cout << endl;

// Копирование четных элементов в другой массив

auto start = chrono::high\_resolution\_clock::now();

copyEven(a, n, b, m);

auto end = chrono::high\_resolution\_clock::now();

chrono::duration<double> duration = end - start;

cout << "Время копирования четных элементов: " << duration.count() << " секунд" << endl;

// Сортировка массива В с использованием сортировки Шелла

start = chrono::high\_resolution\_clock::now();

shellSort(b, m);

end = chrono::high\_resolution\_clock::now();

duration = end - start;

cout << "Время сортировки Шелла: " << duration.count() << " секунд" << endl;

// Вывод отсортированного массива В

cout << "Отсортированный массив В: ";

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

cout << b[i] << " ";

}

cout << endl;

// Сортировка массива В с использованием сортировки выбором

start = chrono::high\_resolution\_clock::now();

selectionSort(b, m);

end = chrono::high\_resolution\_clock::now();

duration = end - start;

cout << "Время сортировки выбором: " << duration.count() << " секунд" << endl;

// Вывод отсортированного массива В

cout << "Отсортированный массив В: ";

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

cout << b[i] << " ";

}

cout << endl;

return 0;

}

