Банкузов Михаил 7 группа Лабораторная работа №17 Вариант 1



// Подключение необходимых библиотек

#include <iostream> // для ввода и вывода

#include <vector> // для работы с векторами

#include <algorithm> // для использования стандартных алгоритмов

#include <ctime> // для измерения времени выполнения

#include <cstdlib> // для генерации случайных чисел

// Функция для генерации случайных чисел от 0 до 99

int generateRandomNumber() {

return std::rand() % 100;

}

// 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 сортировки подсчетом

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

int maxElement = \*std::max\_element(arr.begin(), arr.end());

int minElement = \*std::min\_element(arr.begin(), arr.end());

int range = maxElement - minElement + 1;

std::vector<int> count(range, 0);

// Подсчет количества вхождений каждого элемента массива

for (int num : arr) {

count[num - minElement]++;

}

int idx = 0;

// Запись отсортированных элементов в массив

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

while (count[i] > 0) {

arr[idx++] = i + minElement;

count[i]--;

}

}

}

// Функция для сортировки массива методом сортировки Хоара (быстрой сортировки)

void quickSort(std::vector<int>& arr, int low, int high) {

if (low < high) {

int pivot = arr[high];

int i = low - 1;

// Перестановка элементов массива вокруг опорного элемента

for (int j = low; j <= high - 1; ++j) {

if (arr[j] <= pivot) {

i++;

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

}

}

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

int pivotIndex = i + 1;

// Рекурсивная сортировка двух подмассивов

quickSort(arr, low, pivotIndex - 1);

quickSort(arr, pivotIndex + 1, high);

}

}

int main() {

setlocale(LC\_ALL, "Russian"); // Установка локали

int n = 10000; // Размер массива

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

// Генерация случайных чисел для массива

std::generate(arr.begin(), arr.end(), generateRandomNumber);

// Измерение времени выполнения сортировки пузырьком

std::vector<int> arrBubbleSort(arr);

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

bubbleSort(arrBubbleSort);

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

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

// Восстановление исходного массива

std::generate(arr.begin(), arr.end(), generateRandomNumber);

// Измерение времени выполнения сортировки подсчетом

std::vector<int> arrCountingSort(arr);

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

countingSort(arrCountingSort);

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

double timeCountingSort = static\_cast<double>(endCountingSort - startCountingSort) / CLOCKS\_PER\_SEC;

// Восстановление исходного массива

std::generate(arr.begin(), arr.end(), generateRandomNumber);

// Измерение времени выполнения сортировки Хоара

std::vector<int> arrQuickSort(arr);

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

quickSort(arrQuickSort, 0, n - 1);

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

double timeQuickSort = static\_cast<double>(endQuickSort - startQuickSort) / CLOCKS\_PER\_SEC;

// Вывод результатов

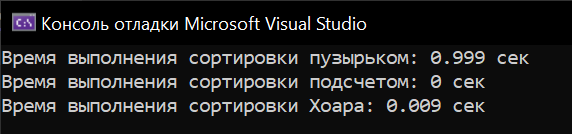
std::cout << "Время выполнения сортировки пузырьком: " << timeBubbleSort << " сек" << std::endl;

std::cout << "Время выполнения сортировки подсчетом: " << timeCountingSort << " сек" << std::endl;

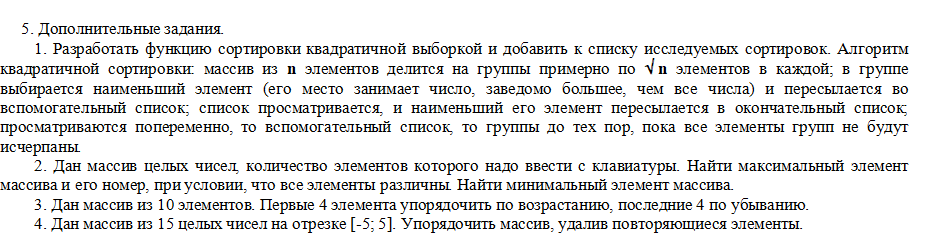
std::cout << "Время выполнения сортировки Хоара: " << timeQuickSort << " сек" << std::endl;

return 0;

}



Дополнительные задания



Задание 2

#include <iostream>

using namespace std;

int main() {

setlocale(LC\_ALL, "Rus");

int n;

cout << "Введите количество элементов массива: ";

cin >> n;

int max\_element = INT\_MIN; // начальное значение максимального элемента

int max\_index;

int min\_element = INT\_MAX; // начальное значение минимального элемента

int\* arr = new int[n];

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

cout << "Введите элемент массива #" << i + 1 << ": ";

cin >> arr[i]; //вводим размер массива

if (arr[i] > max\_element) {

max\_element = arr[i];

max\_index = i;

}

if (arr[i] < min\_element) {

min\_element = arr[i];

} //получаем максимальный элемент, его номер, а также мин. Элемем.

}

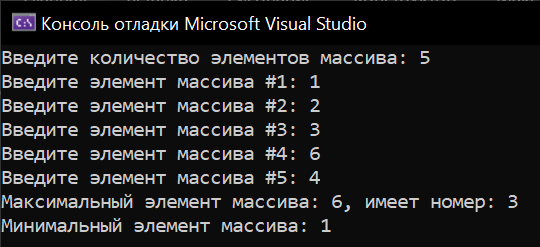
cout << "Максимальный элемент массива: " << max\_element << ", имеет номер: " << max\_index << endl;

cout << "Минимальный элемент массива: " << min\_element << endl;

delete[] arr;

return 0;

}



Задание 3

#include <iostream>

using namespace std;

void bubble\_sort(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]);

}

}

}

}

int main() {

int arr[10] = { 5, 3, 7, 1, 10, 8, 6, 4, 2, 9 };

bubble\_sort(arr, 4); // сортировка первых 4 элементов по возрастанию

bubble\_sort(arr + 6, 4); // сортировка последних 4 элементов по убыванию

for (int i = 0; i < 10; i++) {

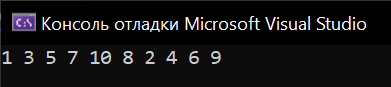
cout << arr[i] << " ";

}

cout << endl;

return 0;

}



Задание 4

#include <iostream>

using namespace std;

void bubble\_sort(int arr[], int n) { // сортировка пузырьком

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 remove\_duplicates(int arr[], int n) {

int k = 0;

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

if (arr[i] != arr[i + 1]) {

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

}

}

arr[k++] = arr[n - 1]; // последний элемент всегда уникальный, поэтому его нужно добавить в массив

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

arr[i] = 0; // заполняем нулями оставшуюся часть массива

}

}

int main() {

int arr[15] = { 5, -3, 4, -2, 0, 1, 3, -1, -5, -4, 2, 0, -2, 1, 5 };

bubble\_sort(arr, 15); // сортировка массива

remove\_duplicates(arr, 15); // удаление повторяющихся элементов

for (int i = 0; i < 15; i++) {

cout << arr[i] << " ";

}

cout << endl;

return 0;

}

