 Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа 14

По дисциплине «Основы алгоритмизации и программирования»

На тему «Хеш-таблицы c цепочками»

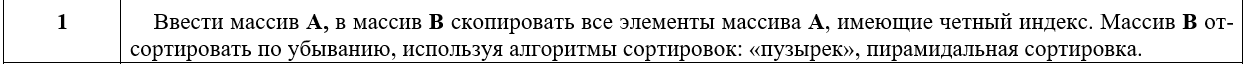
Выполнил:

Студент 1 курса 6 группы

Альшевская Алина Михайловна

Преподаватель: асс. Андронова М.В.

2024, Минск

Вариант 1



#include <iostream>

#include <vector>

#include <algorithm>

#include <Windows.h>

#include <chrono>

// Функция 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 пирамидной сортировки

void heapify(std::vector<int>& A, int n, int i) {

int largest = i;

int left = 2 \* i + 1;

int right = 2 \* i + 2;

if (left < n && A[left] > A[largest])

largest = left;

if (right < n && A[right] > A[largest])

largest = right;

if (largest != i) {

std::swap(A[i], A[largest]);

heapify(A, n, largest);

}

}

void heapSort(std::vector<int>& A) {

int n = A.size();

for (int i = n / 2 - 1; i >= 0; i--)

heapify(A, n, i);

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

std::swap(A[0], A[i]);

heapify(A, i, 0);

}

}

// Функция для заполнения массива случайными числами

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

for (int& i : arr) {

i = rand() % 100; // Генерируем случайное число от 0 до 99

}

}

int main() {

setlocale(LC\_ALL, "Russian");

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

for (int n : sizes) {

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

fillArray(A);

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

if (i % 2 == 0 ) {

B.push\_back(A[i]);

}

}

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

bubbleSort(B);

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

auto duration = std::chrono::duration\_cast<std::chrono::microseconds>(stop - start);

std::cout << "Пузырьковая сортировка для " << n << " элементов заняла " << duration.count() << " мс.\n" << std::endl;

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

heapSort(B);

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

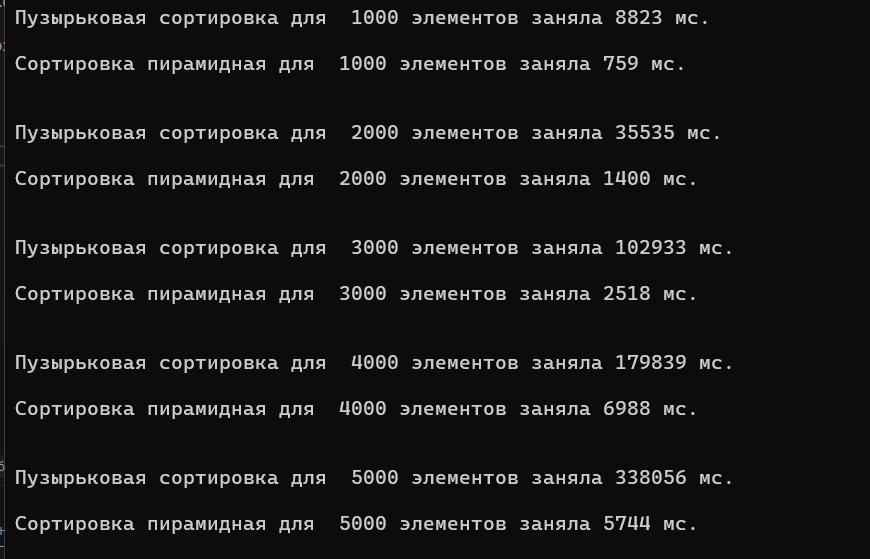
duration = std::chrono::duration\_cast<std::chrono::microseconds>(stop - start);

std::cout << "Сортировка пирамидная для " << n << " элементов заняла " << duration.count() << " мс.\n\n" << std::endl;

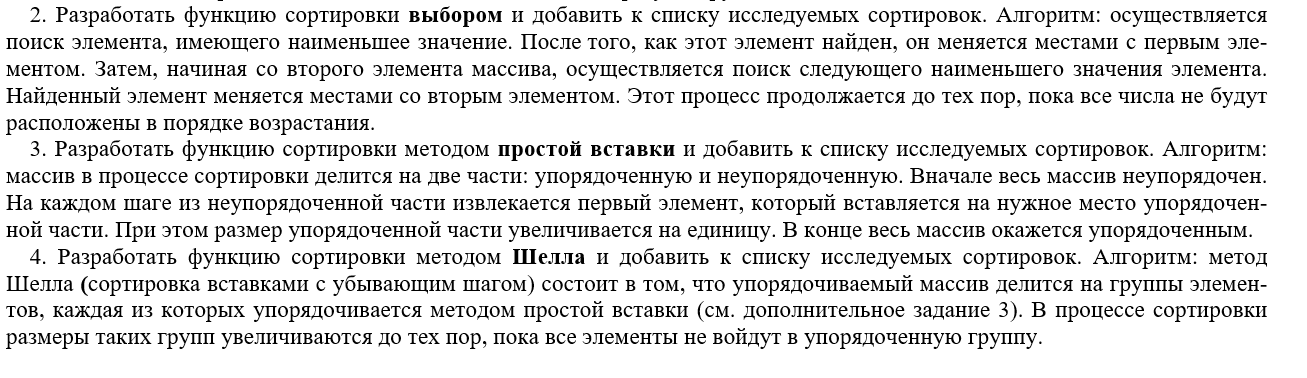
}

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;

}

