

#include <iostream>

#include <chrono> // Для вымярэння часу выканання сарціровак

#include <cstdlib> // Для функцыі rand() і srand()

#include <ctime> // Для ініцыялізацыі генератара выпадковых лікаў

using namespace std;

// Функцыя для сартавання масіва метадам бурбалкі

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

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]);

}

}

}

}

// Функцыя для хуткага сартавання (сартаванні Хоара)

void quickSort(int arr[], int low, int high) {

if (low < high) {

// Разбіццё масіва

int pivot = arr[(low + high) / 2];

int i = low - 1;

int j = high + 1;

while (true) {

do {

++i;

} while (arr[i] > pivot);

do {

--j;

} while (arr[j] < pivot);

if (i >= j) break;

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

}

// Рэкурсіўнае сартаванне двух падмасіваў

quickSort(arr, low, j);

quickSort(arr, j + 1, high);

}

}

int main() {

int n;

cout << "Увядзіце памер масіва A: ";

cin >> n;

// Генеруем масіў A

int\* A = new int[n]; // Вылучэнне памяці пад масіў A

srand(time(0)); // Ініцыялізацыя генератара выпадковых лікаў

cout << "Масіў A:\n";

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

A[i] = rand() % 100; // Генерацыя выпадковых лікаў ад 0 да 99

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

}

cout << endl;

// Знаходзім максімальны элемент у масіве A і яго індэкс

int maxElement = A[0];

int maxIndex = 0;

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

if (A[i] > maxElement) {

maxElement = A[i];

maxIndex = i;

}

}

// Ствараем масіў B і пераносім няцотныя элементы правей максімальнага элемента з масіва A

int\* B = new int[n]; // Вылучэнне памяці пад масіў B

int countB = 0; // Пераменная для падліку элементаў у масіве B

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

if (i % 2 != 0) {

B[countB] = A[i];

countB++;

}

}

// Вымяраем час выканання сартавання бурбалкай

auto start = std::chrono::high\_resolution\_clock::now(); // Пачатак замеру часу

// Сартуем масіў B па змяншэнні з дапамогай метаду бурбалкі

bubbleSort(B, countB);

auto end = std::chrono::high\_resolution\_clock::now(); // Канец замеру часу

std::chrono::microseconds duration = std::chrono::duration\_cast<std::chrono::microseconds>(end - start); // Вылічэнне працягласці часу ў мікрасекундах

std::cout << "Сартаванне метадам бурбалкі заняла: " << duration.count() << " мікрасекунд\n"; // Выснова працягласці часу выканання сартавання

// Выводзім адсартаваны масіў B

cout << "Масіў B (адсартаваны метадам бурбалкі):\n";

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

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

}

cout << endl;

// Капіяваны масіў A у масіў B і пераносім няцотныя элементы правей максімальнага элемента з масіва A

countB = 0; // Скідаем лічыльнік элементаў у масіве B

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

if (i % 2 != 0) {

B[countB] = A[i];

countB++;

}

}

// Вымяраем час выканання сартавання Хоара

start = std::chrono::high\_resolution\_clock::now(); // Пачатак замеру часу

// Сартуем масіў B па змяншэнні з дапамогай сартавання Хоара

quickSort(B, 0, countB - 1);

end = std::chrono::high\_resolution\_clock::now(); // Канец замеру часу

duration = std::chrono::duration\_cast<std::chrono::microseconds>(end - start); // Вылічэнне працягласці часу ў мікрасекундах

std::cout << "Сартаванне метадам Хоара заняла: " << duration.count() << " мікрасекунд\n"; // Выснова працягласці часу выканання сартавання

// Выводзім адсартаваны масіў B

cout << "Масіў B (адсартаваны сартаваннем Хоара):\n";

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

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

}

cout << endl;

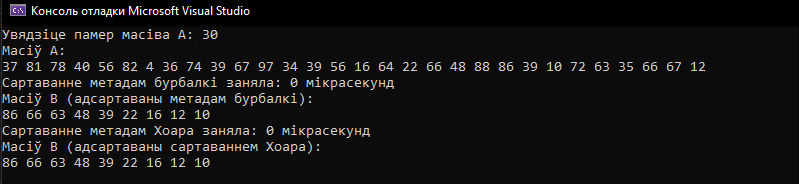
// Вызваляем выдзеленую памяць для масіваў A і B

delete[] A;

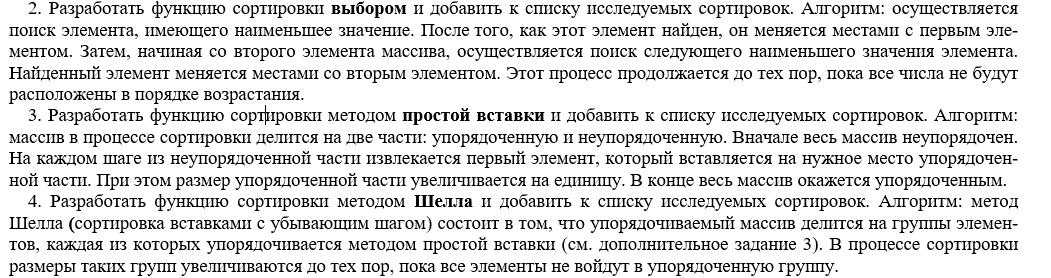
delete[] B;

return 0;

}



Дап. заданні



#include <iostream>

#include <chrono> // Для вымярэння часу выканання сарціровак

#include <cstdlib> // Для функцыі rand() і srand()

#include <ctime> // Для ініцыялізацыі генератара выпадковых лікаў

using namespace std;

// Функцыя для сартавання масіва метадам выбару

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]);

}

}

// Функцыя для сартавання масіва метадам простай устаўкі

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;

}

}

// Функцыя для сартавання масіва метадам Шэла

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() {

int n;

cout << "Увядзіце памер масіва A: ";

cin >> n;

// Генеруем масіў A

int\* A = new int[n]; // Вылучэнне памяці пад масіў A

srand(time(0)); // Ініцыялізацыя генератара выпадковых лікаў

cout << "Масіў A:\n";

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

A[i] = rand() % 100; // Генерацыя выпадковых лікаў ад 0 да 99

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

}

cout << endl;

// Знаходзім максімальны элемент у масіве A і яго індэкс

int maxElement = A[0];

int maxIndex = 0;

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

if (A[i] > maxElement) {

maxElement = A[i];

maxIndex = i;

}

}

// Ствараем масіў B і пераносім няцотныя элементы правей максімальнага элемента з масіва A

int\* B = new int[n]; // Вылучэнне памяці пад масіў B

int countB = 0; // Пераменная для падліку элементаў у масіве B

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

if (i % 2 != 0) {

B[countB] = A[i];

countB++;

}

}

// Вымяраем час выканання сартавання выбарам

auto start = chrono::high\_resolution\_clock::now(); // Пачатак замеру часу

selectionSort(B, countB);

auto end = chrono::high\_resolution\_clock::now(); // Канец замеру часу

chrono::microseconds duration = chrono::duration\_cast<chrono::microseconds>(end - start); // Вылічэнне працягласці часу ў мікрасекундах

cout << "Сартаванне метадам выбару заняла: " << duration.count() << " мікрасекунд" << endl;

// Выводзім адсартаваны масіў B

cout << "Масіў B (адсартаваны метадам выбару):\n";

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

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

}

cout << endl;

// Капіяваны масіў A у масіў B і пераносім няцотныя элементы правей максімальнага элемента з масіва A

countB = 0; // Скідаем лічыльнік элементаў у масіве B

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

if (i % 2 != 0) {

B[countB] = A[i];

countB++;

}

}

// Вымяраем час выканання сартавання метадам простай устаўкі

start = chrono::high\_resolution\_clock::now(); // Пачатак замеру часу

insertionSort(B, countB);

end = chrono::high\_resolution\_clock::now(); // Канец замеру часу

duration = chrono::duration\_cast<chrono::microseconds>(end - start); // Вылічэнне працягласці часу ў мікрасекундах

cout << "Сартаванне метадам простай устаўкі заняла: " << duration.count() << " мікрасекунд" << endl;

// Выводзім адсартаваны масіў B

cout << "Масіў B (адсартаваны метадам простай устаўкі):\n";

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

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

}

cout << endl;

// Капіяваны масіў A у масіў B і пераносім няцотныя элементы правей максімальнага элемента з масіва A

countB = 0; // Скідаем лічыльнік элементаў у масіве B

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

if (i % 2 != 0) {

B[countB] = A[i];

countB++;

}

}

// Вымяраем час выканання сартавання метадам Шэла

start = chrono::high\_resolution\_clock::now(); // Пачатак замеру часу

shellSort(B, countB);

end = chrono::high\_resolution\_clock::now(); // Канец замеру часу

duration = chrono::duration\_cast<chrono::microseconds>(end - start); // Вылічэнне працягласці часу ў мікрасекундах

cout << "Сартаванне метадам Шэла заняла: " << duration.count() << " мікрасекунд" << endl;

// Выводзім адсартаваны масіў B

cout << "Масіў B (адсартаваны метадам Шэла):\n";

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

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

}

cout << endl;

// Вызваляем выдзеленую памяць для масіваў A і B

delete[] A;

delete[] B;

return 0;

}

