Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа №15

По дисциплине «Основы алгоритмизации и программирования»

На тему « Анализ алгоритмов сортировки»

Выполнил:

Студент 1 курса 8 группы

Лужецкий Владислав Константинович

Преподаватель: асс. Андронова М.В.

2024, Минск

Оглавление

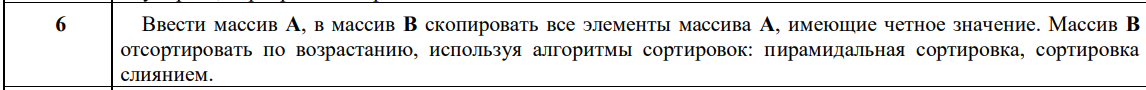
[Основное задание(Вариант 6) 2](#_Toc166194335)

[Дополнительное задание 1 4](#_Toc166194336)

[Дополнительное задание 2 5](#_Toc166194337)

[Дополнительное задание 3 6](#_Toc166194338)

# Основное задание(Вариант 6)



#include <iostream>

#include <ctime>

#include <chrono>

using namespace std;

void heapify(int A[], int pos, int n)

{

int t, tm;

while (2 \* pos + 1 < n)

{

t = 2 \* pos + 1;

if (2 \* pos + 2 < n && A[2 \* pos + 2] >= A[t])

t = 2 \* pos + 2;

if (A[pos] < A[t])

{

tm = A[pos];

A[pos] = A[t];

A[t] = tm;

pos = t;

}

else break;

}

}

void piramSort(int A[], int n)

{

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

heapify(A, i, n);

while (n > 0)

{

int tm = A[0];

A[0] = A[n - 1];

A[n - 1] = tm;

n--;

heapify(A, 0, n);

}

}

void insOrd(int m[], int sm, int em, int e)

{ // m[]  массив чисел; sm  индекс 1-ого элемента левой части; cm  индекс

//последн. элемента левой части; em  индекс последн. элемента правой части

int buf, i = sm;

while (i <= em && m[i] < e)

{

if (i - 1 >= sm)

m[i - 1] = m[i];

i++;

} m[i - 1] = e;

}

int\* merge(int m[], int sm, int cm, int em)

{

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

{

if (m[i] > m[cm + 1])

{

int buf = m[i];

m[i] = m[cm + 1];

insOrd(m, cm + 1, em, buf);

}

}

return m;

}

int\* sortMerge(int m[], int lm, int sm = 0)

{

if (lm > 1)

{

sortMerge(m, lm / 2, sm);

sortMerge(m, lm - lm / 2, sm + lm / 2);

merge(m, sm, sm + lm / 2 - 1, sm + lm - 1);

};

return m;

}

int main()

{

//по коду массив заполняется рандомными элементами от 1 до 50.

// Далее копируется во 2 массив.

// После сортируется двумя разными способами и считает время

// Считаю, что строить графики в Excel бессмысленно, потому что

// пирамидальная сортировка быстрее при любом size в +- 4-8 раз

srand(time(NULL));

setlocale(0, "");

int size;

cout << "\nВведите размерность массива:";

cin >> size;

int\* ArrayA = new int[size];

int\* ArrayB = new int[size]; //2 массив нужен для 2 сортировки.

cout << "\nИсходный массив: ";

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

{

ArrayA[i] = rand() % 50 + 1;

cout << ArrayA[i] << ' ';

}

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

{

ArrayB[i] = ArrayA[i];

}

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

piramSort(ArrayA, size);

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

auto duration = chrono::duration\_cast<chrono::microseconds>(endTime - startTime).count();

cout << "\nВремя выполнения пирамидальной сортировкой: " << duration << " микросекунд." << endl;

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

{

cout << ArrayA[i] << ' ';

}

auto startTime1 = chrono::high\_resolution\_clock::now();

sortMerge(ArrayB, size);

auto endTime1 = chrono::high\_resolution\_clock::now();

auto duration1 = chrono::duration\_cast<chrono::microseconds>(endTime1 - startTime1).count();

cout << "\nВремя выполнения сортировкой слиянием: " << duration1 << " микросекунд." << endl;

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

{

cout << ArrayB[i] << ' ';

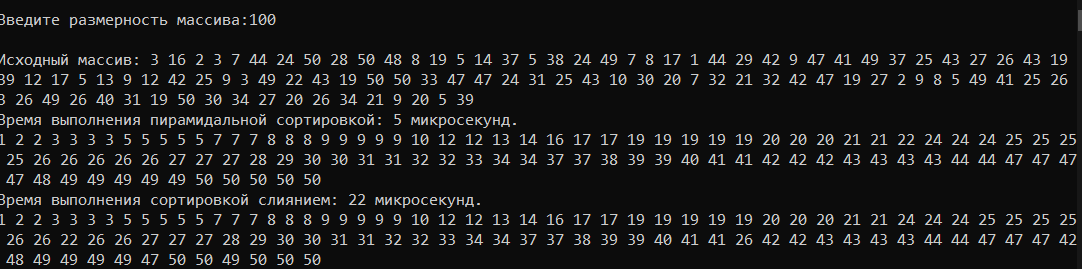
}

delete[] ArrayA;

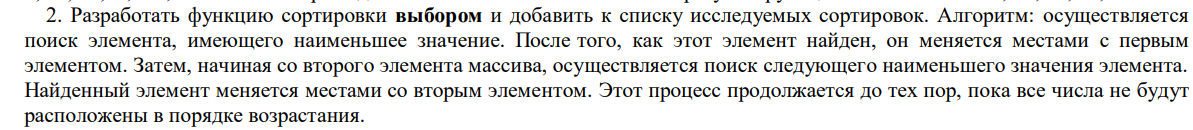
delete[] ArrayB;

return 0;

}



# Дополнительное задание 1



int tempD=0, min =0;

cout << "\n\nМассив D: "; //выбор

auto startArrayD = high\_resolution\_clock::now();

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

{

min = i;

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

{

if (arrayD[j] < arrayD[min])

min = j;

}

if (i != min)

{

tempD = arrayD[i];

arrayD[i] = arrayD[min];

arrayD[min] = tempD;

}

}

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

{

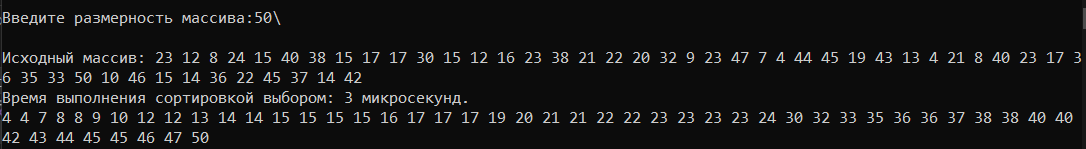
cout << arrayD[i] << ' ';

}

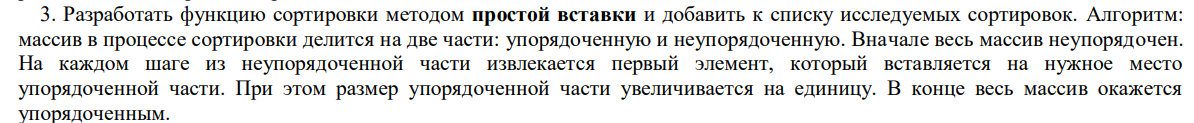
auto stopArrayD = high\_resolution\_clock::now();

auto differenceArrayD = duration\_cast<microseconds>(stopArrayD - startArrayD);

cout << "\nВремя сортировкой выбором: " << differenceArrayD.count() << " микросекунд.";



# Дополнительное задание 2



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

int tempC, current\_element;

for (int i = 1; i < size; i++)

{

current\_element = ArrayA[i];

for (int j = i - 1; j >= 0 && ArrayA[j] > current\_element; j--)

{

tempC = ArrayA[j + 1];

ArrayA[j + 1] = ArrayA[j];

ArrayA[j] = tempC;

}

}

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

auto duration = chrono::duration\_cast<chrono::microseconds>(endTime - startTime).count();

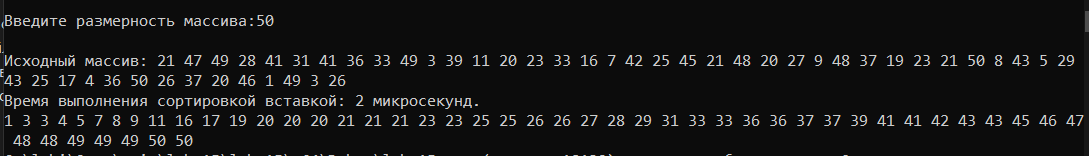
cout << "\nВремя выполнения сортировкой вставкой: " << duration << " микросекунд." << endl;

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

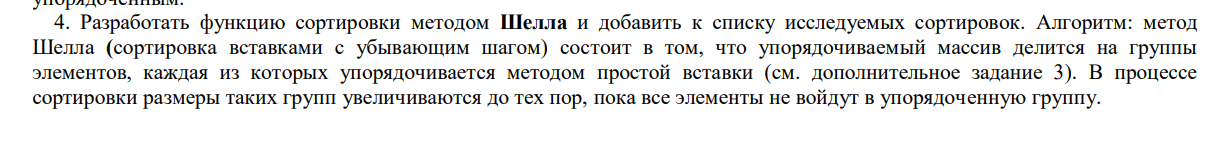
{

cout << ArrayA[i] << ' ';

}



# Дополнительное задание 3



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

for (int gap = size / 2; gap > 0; gap /= 2) {

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

int temp = ArrayA[i];

int j;

for (j = i; j >= gap && ArrayA[j - gap] > temp; j -= gap) {

ArrayA[j] = ArrayA[j - gap];

}

ArrayA[j] = temp;

}

}

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

auto duration = chrono::duration\_cast<chrono::microseconds>(endTime - startTime).count();

cout << "\nВремя выполнения сортировкой Шелла: " << duration << " микросекунд." << endl;

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

{

cout << ArrayA[i] << ' ';

}

