МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСІТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТІТУТ»

КАФЕДРА МЕДИЧНОЇ КІБЕРНЕТИКИ ТА ТЕЛЕМЕДИЦИНИ

**Лабораторна робота № 13**

з дисципліни «Алгоритмізація та програмування»

на тему: “ Алгоритми сортування ”

Варіант 16

**Виконав:**

Студент групи БС-32

Хоміцевич Микита

**Перевірила:**

викладач

Матвійчук А.О.

Київ-2014

**Задание**



**□ Лабораторна робота виконана без зауважень**

**□ Лабораторна робота має зауваження:**

**□ присутні зауваження до блок-схем:**

**□ виконані не за стандартом**

**□ не відповідають коду**

**□ присутній код**

**□ інші зауваження:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**□присутні зауваження до коду:**

**□задача завдання вирішена хибно**

**□ код програми не компілюється**

**□ використано глобальні змінні**

**□ недостатня декомпозиція на функціонуванні користувача**

**□ статичні змінні при роботі з масивами**

**□ оформлення коду**

**□присутні зайві символи «{»та «}»**

**□інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**□невірні відповіді на запитання:**

**□№1 □№2 □№3**

**□маються інші зауваження: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Программный код**

#include <iostream>

#include <conio.h>

#include <locale.h>

#include <cmath>

#include <vector>

#include <ctime>

using namespace std;

void MAIN();

void quick\_sort(int\*\*, int, int, int);

void matrix\_input(int\*\*);

void matrix\_output(int\*\*);

void matrix\_delelte(int\*\*);

int main() {

MAIN();

\_getch();

}

void MAIN() {

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

int \*\*matrix = new int \*[10];

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

matrix[i] = new int[10];

cout << "\t" << " FIRST MATRIX " << endl;

matrix\_input(matrix);

matrix\_output(matrix);

cout << endl << endl;

//==============QUICK\_SORT=====================

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

for (int j = 0; j < 10; j += 10)

quick\_sort(matrix, i, 0, 9);

//==============QUICK\_SORT=====================

matrix\_output(matrix);

matrix\_delelte(matrix);

}

void matrix\_input(int \*\*matrix) {

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

for (int j = 0; j < 10; ++j)

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

}

void matrix\_output(int \*\*matrix) {

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

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

cout << matrix[i][j] << " ";

if (j == 9) {cout << endl;}

}

}

void matrix\_delete(int \*\*matrix) {

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

delete[] matrix[i];

delete[] matrix;

}

void quick\_sort(int \*\*matrix, int rows, int low, int high) {

int i = low, j = high, med\_ind = (low + high) / 2, med = matrix[rows][med\_ind];

do {

while (matrix[rows][i] < med) { ++i; }

while (matrix[rows][j] > med) { --j; }

if (i <= j) {

if (matrix[rows][i] > matrix[rows][j]) {

matrix[rows][i] = matrix[rows][i] ^ matrix[rows][j];

matrix[rows][j] = matrix[rows][i] ^ matrix[rows][j];

matrix[rows][i] = matrix[rows][i] ^ matrix[rows][j];

}

i++; j--;

}

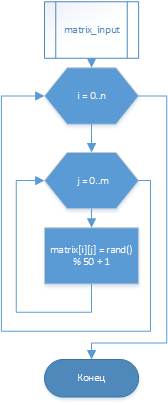
}

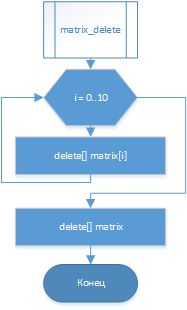
while (i < j);

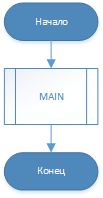
if (low < j) { quick\_sort(matrix, rows, low, j); }

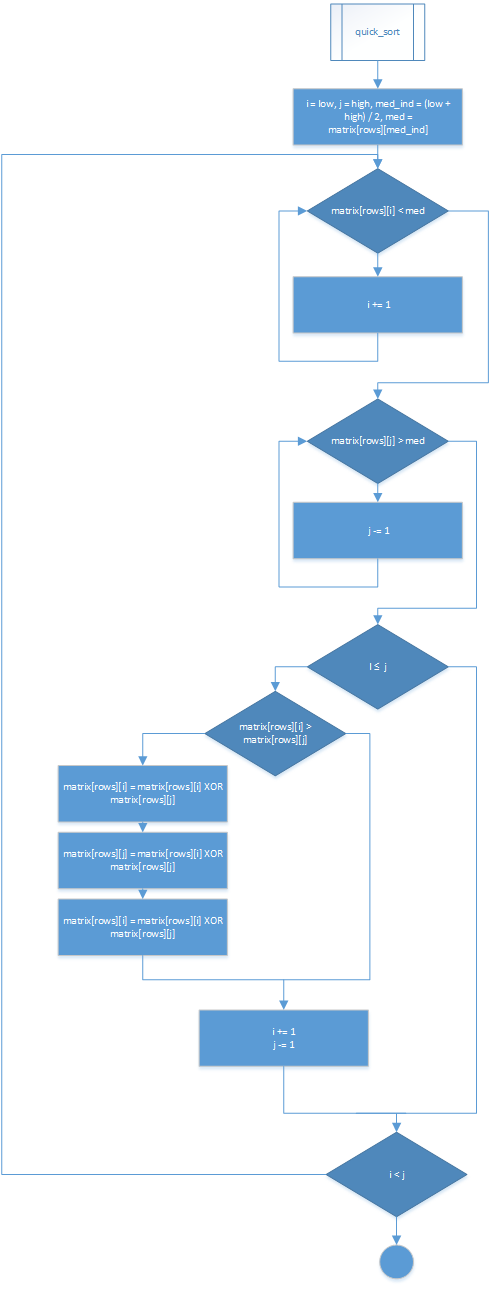
if (i < high) { quick\_sort(matrix, rows, i, high); }

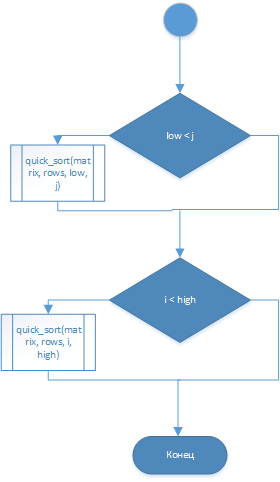
}

**Блок-схема**

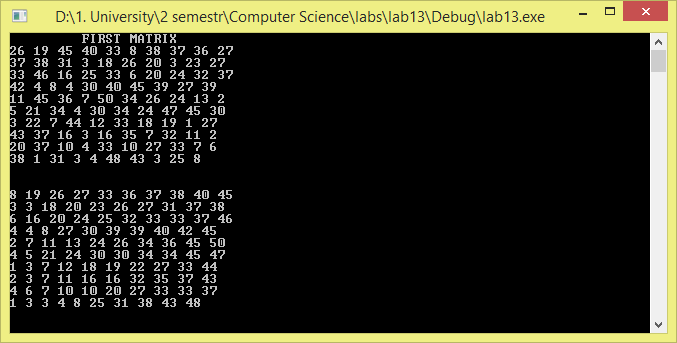
****

****

****

****

**Скриншот работы программы**

****

**Ответы на контрольные вопросы**

1. Что такое сортировка, какая основная цель сортировки?

Сортировка – это упорядочивание конкретных данных по заданному(ым) параметру(ам): по алфавиту, по увеличению, убыванию и т.д.

Основная цель сортировки – облегчить в дальнейшем работу с данными (для поиска или изменения данных).

1. Влияет ли тип данных на время выполнения алгоритма сортировки?

Да, влияет, т.к., к примеру, при использовании типа double процессору необходимо тратить проц.время для вычисления точности данного числа (кол-во знаков после запятой).

1. Почему не существует универсального алгоритма сортировки?

Каждая сортировка имеет как свою сильные стороны, так и слабые, поэтому, исходя из этого и учитывая специфику задачи, нельзя говорить про универсальную сортировку. Можно лишь сказать про сортировку, которая будет состоять из нескольких видов сортировок и при определенных условиях, будет работать какая-то одна или несколько выбраных сортировок.

1. Чем можно пояснить разнообразие алгоритмов сортировки?

Заданные данные можно сортировать разными способами, а эти способы так же можно улучшать, поэтому и существует такое кол-во видов сортировок.

1. Как определить, какому алгоритму сортировки отдать предпочтение при решении задачи?

Нужно учитывать несколько факторов:

* Параметры сортирования
* Насколько отсортированы данные по нашим параметрам
* Какое кол-во памяти должно использоваться
* Скорость работы алгоритма
* Объемы данных

1. Какие на сегодняшний день наиболее эффективные методы сортирования?

Нужно понимать, что есть два вида алгоритмов сортирования данных: устойчивые сортировки и неустойчивые.

Если говорить про устойчивые, то основными сортировками являются: merge sort,bucket sort

Популярные неустойчевые сортировки: shell sort, quick sort, heap sort

1. Почему алгоритмы быстрой сортировки не дают большого выйгрыша при малых размерах массива?

Потому что для использования быстрых сортировок нужны большие расходы на их использование в плане затрат памяти, доп.подготовки и сложности самих алгоритмов. А на малых массивах та же bubble sort будет работать быстрее, чем quick sort, так как в bubble нет никаких расходов памяти до работы самого алгоритма, просто идет обычный перебор.

1. За счет чего в алгоритмах быстрой сортировки происходит выйгрыш при выполнении операции сравнения и перестановок?

Выйгрыш быстрой сортировки по сравнению с простыми сортировками в том, что в quick sort идет доп. Расход памяти, но при этом идет сокращение кол-ва итераций (т.к. элементы передвигаются на большие расстояния). На больших значениях размера массива кол-во итераций очень важен, из-за этого и выходит, что быстрая сортировка быстро работает.

1. В чем преимущества и недостатки по отношению одной к другй след. Алгоритмов сортировок: пирамидальная сортировка, сортировка слиянием, сортировка Шелла, сортировка Хоара?

|  |  |  |  |
| --- | --- | --- | --- |
| Heap sort | Merge sort | Shell sort | Quick sort |
| advantages | | | |
| Превращение списка в кучу |  | Наиболее простая реализация |  |
|  |  |  |  |
| disadvantages | | | |
| Сложность алгоритма: O(n\*log^2(n)) |  |  | В худшем случае: O(n^2) |