Министерство образования и науки РФ

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа «Компьютерных технологий и информационных систем»

ОТЧЕТ

по дисциплине ««Теория и технология программирования»

**Лабораторная работа № 6**

**Выполнил:**

Cтудент гр. 5130902/40002 Г.Ю. Рюмин

**Проверил**

Ст. преподаватель А.М. Журавская

Санкт-Петербург

2025 г.

**1. Цель работы**

Цель задания – изучить методы сортировки данных.

**2. Задание**

Составить программу для сортировки массива данных методами: пузырьковой, отбора, вставки, Шелла и быстрой сортировки. Вывести на экран неупорядоченную (один раз) и упорядоченные (для каждого из методов) массивы данных. Составить сравнительную таблицу эффективности методов, в которой необходимо указать число сравнений и перестановок переменных в каждом методе сортировки.

Неупорядоченная матрица задается один раз случайным образом, далее она используется для каждого из методов сортировки.

|  |  |
| --- | --- |
| 27 | Упорядочить в каждом значении чисел матрицы цифры по убыванию, затем упорядочить данные в строках по возрастанию |

## **Код программы**

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <clocale>

#include <algorithm>

#include <string>

#include <sstream>

#include <climits>

#include <iomanip>

#include <vector>

using namespace std;

void bubbleSort(int arr[], int n, int& comp, int& swp);

void selectionSort(int arr[], int n, int& comp, int& swp);

void insertionSort(int arr[], int n, int& comp, int& swp);

void shellSort(int arr[], int n, int& comp, int& swp);

void quickSort(int arr[], int n, int& comp, int& swp);

int getIntInput(const string& prompt, int minVal, int maxVal) {

string input;

while (true) {

cout << prompt;

getline(cin, input);

if (input.empty()) {

cout << "Ошибка! Введите число." << endl;

continue;

}

bool valid = true;

size\_t start = 0;

if (input[0] == '-') start = 1;

for (size\_t i = start; i < input.size(); i++) {

if (!isdigit(input[i])) {

valid = false;

break;

}

}

if (!valid) {

cout << "Ошибка! Недопустимые символы." << endl;

continue;

}

try {

long num = stol(input);

if (num < minVal || num > maxVal) {

cout << "Ошибка! Число должно быть между " << minVal

<< " и " << maxVal << endl;

}

else {

return (int)num;

}

}

catch (...) {

cout << "Ошибка! Недопустимое значение." << endl;

}

}

}

int sortDigits(int num, void (\*sortFunc)(int[], int, int&, int&)) {

bool negative = num < 0;

num = abs(num);

vector<int> digits;

if (num == 0) digits.push\_back(0);

while (num > 0) {

digits.push\_back(num % 10);

num /= 10;

}

int comp = 0, swp = 0;

sortFunc(digits.data(), digits.size(), comp, swp);

reverse(digits.begin(), digits.end());

int result = 0;

for (int d : digits) {

result = result \* 10 + d;

}

return negative ? -result : result;

}

void bubbleSort(int arr[], int n, int& comp, int& swp) {

comp = swp = 0;

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

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

comp++;

if (arr[j] > arr[j + 1]) {

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

swp++;

swap(arr[j], arr[j + 1]);

}

}

}

}

}

void selectionSort(int arr[], int n, int& comp, int& swp) {

comp = swp = 0;

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

int minIndex = i;

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

comp++;

if (arr[j] < arr[minIndex]) {

minIndex = j;

}

}

if (arr[i] != arr[minIndex]) {

swp++;

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

}

}

}

void insertionSort(int arr[], int n, int& comp, int& swp) {

comp = swp = 0;

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

int key = arr[i];

int j = i - 1;

while (j >= 0) {

comp++;

if (arr[j] > key) {

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

swp++;

arr[j + 1] = arr[j];

}

j--;

}

else break;

}

if (arr[j + 1] != key) {

swp++;

arr[j + 1] = key;

}

}

}

void shellSort(int arr[], int n, int& comp, int& swp) {

comp = swp = 0;

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; j -= gap) {

comp++;

if (arr[j - gap] > temp) {

if (arr[j] != arr[j - gap]) {

swp++;

arr[j] = arr[j - gap];

}

}

else break;

}

if (arr[j] != temp) {

swp++;

arr[j] = temp;

}

}

}

}

int partition(int arr[], int low, int high, int& comp, int& swp) {

int pivot = arr[high];

int i = low - 1;

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

comp++;

if (arr[j] < pivot) {

i++;

if (arr[i] != arr[j]) {

swp++;

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

}

}

}

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

swp++;

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

}

return i + 1;

}

void quickSortHelper(int arr[], int low, int high, int& comp, int& swp) {

if (low < high) {

int pi = partition(arr, low, high, comp, swp);

quickSortHelper(arr, low, pi - 1, comp, swp);

quickSortHelper(arr, pi + 1, high, comp, swp);

}

}

void quickSort(int arr[], int n, int& comp, int& swp) {

comp = swp = 0;

quickSortHelper(arr, 0, n - 1, comp, swp);

}

void printMatrix(int\*\* matrix, int rows, int cols) {

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

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

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

}

cout << endl;

}

}

int main() {

setlocale(LC\_ALL, "Russian");

srand(time(0));

int rows = getIntInput("Введите количество строк (1-99): ", 1, 99);

int cols = getIntInput("Введите количество столбцов (1-99): ", 1, 99);

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

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

matrix[i] = new int[cols];

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

matrix[i][j] = rand() % 9000 + 1000;

}

}

cout << endl << "Исходная матрица:" << endl;

printMatrix(matrix, rows, cols);

struct {

string name;

void (\*func)(int[], int, int&, int&);

} sorts[] = {

{"Пузырьковая", bubbleSort},

{"Выбором", selectionSort},

{"Вставками", insertionSort},

{"Шелла", shellSort},

{"Быстрая", quickSort}

};

int total\_comp[5] = { 0 };

int total\_swp[5] = { 0 };

for (int s = 0; s < 5; s++) {

int\*\* copy = new int\* [rows];

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

copy[i] = new int[cols];

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

copy[i][j] = sortDigits(matrix[i][j], sorts[s].func);

}

}

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

int comp = 0, swp = 0;

sorts[s].func(copy[i], cols, comp, swp);

total\_comp[s] += comp;

total\_swp[s] += swp;

}

cout << endl << sorts[s].name << " сортировка: " << endl;

printMatrix(copy, rows, cols);

for (int i = 0; i < rows; i++) delete[] copy[i];

delete[] copy;

}

cout << endl << "Сравнительная таблица методов сортировки:" << endl;

cout << "-------------------------------------------------" << endl;

cout << "| Метод | Сравнения | Перестановки |" << endl;

cout << "-------------------------------------------------" << endl;

for (int s = 0; s < 5; s++) {

cout << "| " << left << setw(13) << sorts[s].name

<< " | " << setw(13) << total\_comp[s]

<< " | " << setw(13) << total\_swp[s] << " |" << endl;

}

cout << "-------------------------------------------------" << endl;

for (int i = 0; i < rows; i++) delete[] matrix[i];

delete[] matrix;

return 0;

}

**Пример работы программы**

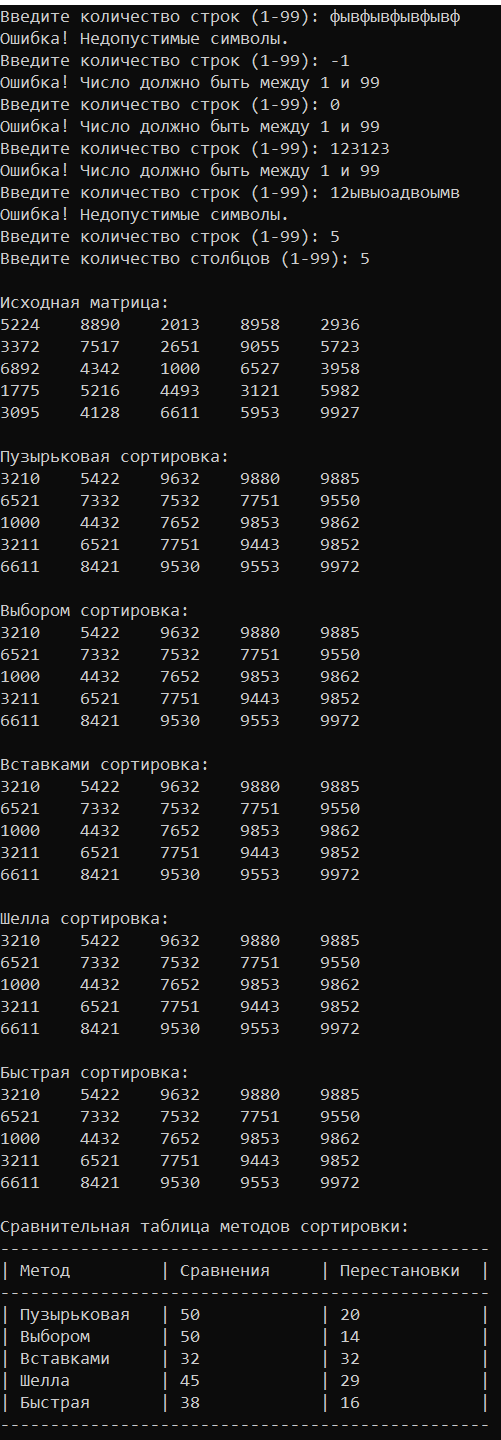


Рисунок 1 – Пример работы программы с некорректными данными, корректными данными, выводом таблицы сравнений и перестановок

## **Вывод**

В ходе лабораторной работы были изучены и реализованы методы сортировки: пузырьковая, отбором, вставками, Шелла и быстрая сортировка. Неупорядоченный массив был сгенерирован случайным образом и использован для всех методов. Упорядоченные массивы были выведены на экран, а также составлена сравнительная таблица эффективности, показавшая, что сортировка Шелла и быстрая сортировка являются наиболее эффективными благодаря меньшему числу сравнений и перестановок. Дополнительно была выполнена сортировка цифр в каждом значении матрицы по убыванию и упорядочивание строк по возрастанию. Работа позволила закрепить навыки реализации и анализа алгоритмов сортировки.