|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_***ИУК «Информатика и управление»*\_\_**\_\_\_\_\_\_\_\_\_\_\_\_

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**Лабораторная работа №4**

**«Сортировки»**

**ДИСЦИПЛИНА: «Вычислительные алгоритмы»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-42Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_)  (Подпись) (Ф.И.О.) |
| Проверил(а): | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_Ткаченко. А. В.\_\_\_)  (Подпись) (Ф.И.О.) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |
| Калуга, 2024 г. | | |

# Цели работы:

формирование практических навыков разработки алгоритма сортировки.

**Задачи:**

1. Создать блок-схему алгоритма
2. Реализовать заданный алгоритм
3. Измерить время выполнения алгоритма
4. Исследовать вычислительную сложность алгоритма

**Вариант №17**

Написать программу, сортирующую методом вставок главную диагональ двумерного массива целых чисел. Массив считать из текстового файла, содержащего в первой строке кол-во строк и кол-во столбцов, а далее в каждой строке целые числа, соответствующие элементам строки массива.

**Результаты работы:**

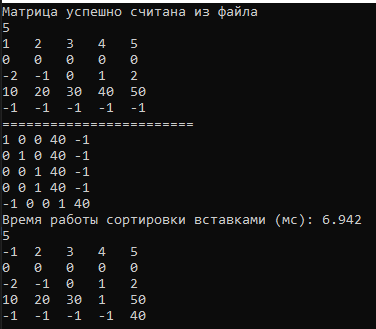
****

Рис. 1 Результаты работы сортировки

**Оценка сложности алгоритма:**

|  |  |  |
| --- | --- | --- |
| Алгоритм insertion\_sort()->void | Стоимость операции | Количество повторений |
| { | - | - |
| temp = array | c1 | 1 |
| for i <- 0 to dim: | c2 | n |
| { | - | - |
| temp[i] <- matrix[i][i] | c3 | n-1 |
| } | - | - |
| for i <- 0 to dim: | c4 | n |
| { | - | - |
| j <- i | c5 | n-1 |
| while temp[j] < temp[j-1] | c6 |  |
| { | - | - |
| if j = 0 then break | c7 |  |
| swap(temp[j], temp[j-1]) | |  | | --- | | c8 | |  |
| j <- j-1 | |  | | --- | | c9 | |  |
| } | - | - |
| } | - | - |
| for i<-0 to dim: | c10 | n |
| { | - | - |
| matrix[i][i] <- temp[i] | c11 | n-1 |
| } | - | - |
| } | - | - |

Табл. 1 Таблица оценки сложности алгоритма

tj - индексная переменная, число выполнения условий сравнения

**Блок-схемы алгоритмов:**

*Clear()->void:*



*Init()->void:*



*Insertion\_sort()->void:*



*Print()->void:*



*readMatrixFromFile(path: string)->void:*



*main()->void:*



**Листинг кода:**

#include <iostream>

#include <string>

#include <fstream>

#include <exception>

#include <iomanip>

#include <chrono>

template <class T>

class Matrix

{

T\*\* matrix = nullptr;

int dim = 0;

void clear()

{

if (this->matrix)

{

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

{

delete[] this->matrix[i];

}

delete[] this->matrix;

this->matrix = nullptr;

this->dim = 0;

}

}

void init()

{

if (!this->matrix)

{

this->matrix = new T\* [this->dim];

for (int i = 0; i < this->dim; i++)

{

this->matrix[i] = new T[this->dim];

}

}

}

private:

public:

Matrix() : dim{ 0 }, matrix{ nullptr } {};

~Matrix()

{

clear();

}

void insertion\_sort()

{

T\* temp = new T[this->dim];

for (int i = 0; i < this->dim; i++)

{

temp[i] = this->matrix[i][i];

}

for (int i = 1; i < this->dim; i++)

{

int j = i;

while (temp[j] < temp[j-1])

{

if (j == 0) break;

std::swap(temp[j], temp[j-1]);

j--;

}

}

for (int i = 0; i < this->dim; i++)

{

this->matrix[i][i] = temp[i];

}

delete[] temp;

}

void print()

{

std::cout << this->dim << std::endl;

for (int i = 0; i < this->dim; i++)

{

for (int j = 0; j < this->dim; j++)

{

std::cout << std::setiosflags(std::ios\_base::left) << std::setw(4) << matrix[i][j];

}

std::cout << std::endl;

}

}

void readMatrixFromFile(std::string path)

{

std::ifstream fin = std::ifstream(path);

if (fin.is\_open())

{

try

{

int rows = 0, cols = 0;

fin >> rows >> cols;

if (rows != cols)

{

std::cout << "Матрица не квадратная!" << std::endl;

throw new std::runtime\_error("Matrix not square");

}

this->dim = rows;

init();

for (int i = 0; i < this->dim; i++)

{

for (int j = 0; j < this->dim; j++)

{

fin >> this->matrix[i][j];

}

}

std::cout << "Матрица успешно считана из файла" << std::endl;

fin.close();

}

catch (...)

{

std::cout << "Ошибка парсинга!" << std::endl;

fin.close();

}

}

}

};

int main()

{

setlocale(0, "");

Matrix<int> m = Matrix<int>();

m.readMatrixFromFile("matrix.txt");

m.print();

std::cout << "========================" << std::endl;

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

m.insertion\_sort();

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

auto elapsed\_ms = ((std::chrono::nanoseconds)(end - begin)).count();

std::cout << "Время работы сортировки вставками (мс): " << elapsed\_ms / std::pow(10, 6) << std::endl;

m.print();

}

**Вывод:**

В ходе выполнения лабораторной работы были сформированы практические навыки разработки алгоритма сортировки.