МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра № 304 «Вычислительные машины, системы и сети»

Программирование

Отчет по лабораторной работе № 3

Структуры

Выполнили студенты группы M3О-111Б-21

Багиров Эльдар

Нуриев Наиль

Проверила доцент, к.т.н., Дмитриева Е.А.

Москва 2022 г.

Оглавление

[Задание 3](#_Toc101373486)

[Структурные схемы алгоритмов 4](#_Toc101373487)

[Функция main 4](#_Toc101373488)

[Функция Errors 5](#_Toc101373489)

[Функция memory\_for\_matrix 6](#_Toc101373490)

[Функция input\_matrix 7](#_Toc101373491)

[Функция matrix\_from\_file 8](#_Toc101373492)

[Функция print\_matrix 9](#_Toc101373493)

[Функция maxim 10](#_Toc101373494)

[Функция product 11](#_Toc101373495)

[Код программы 12](#_Toc101373496)

[Тестирование программы 15](#_Toc101373497)

[Тестирование некорректных режимов 15](#_Toc101373498)

[Тестирование корректных режимов 17](#_Toc101373499)

[Вывод 20](#_Toc101373500)

# Задание

Кафедра: 302 Курс: ИНФОРМАТИКА

Практика

**ВАРИАНТ №3**

В процессе функционирования АСУ ВД в файле фиксируются данные о самолетах, совершивших посадку. Каждая запись имеет структуру типа:

**ТУ-154М Б-3726 131 12.1**

марка ЛА бортовой количество груз

номер пассажиров

1) подготовить программу, сортирующую записи с использованием индексной сортировки линейным методом в порядке возрастания числовой части бортовых номеров; определить суммарное количество пассажиров, вес грузов; результат печатать в виде таблицы;

2) обеспечить входной контроль бортового номера, количества пассажиров и веса груза, выполнить отладку и тестирование.

Чтение данных из файла производить с использованием функций ввода/вывода языка C++.

Алгоритм должен быть параметризован; обмен данными с подпрограммой должен осуществляться только через параметры; исходные данные хранятся в отдельном файле.

1. Структурные схемы алгоритмов

## Функция main

Главная (основная – main) функция

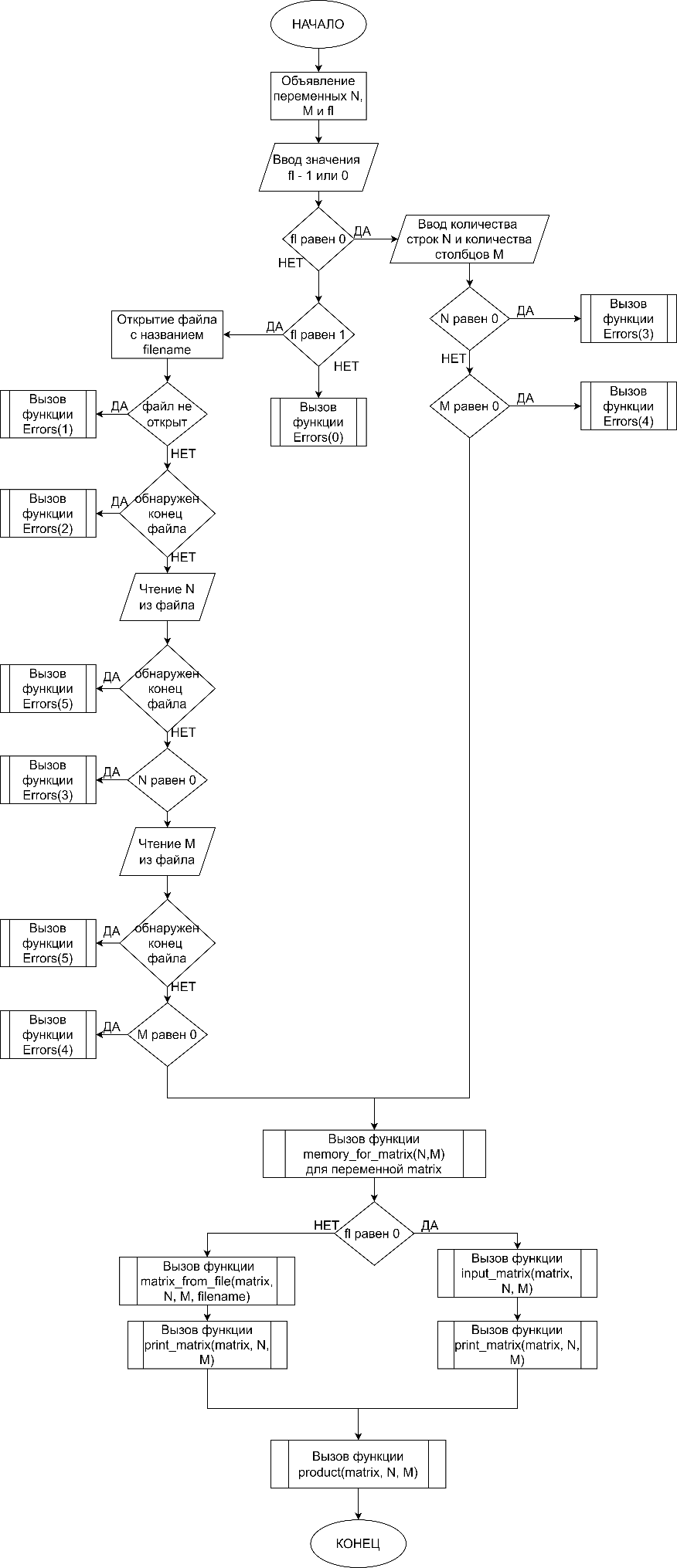


Рисунок 1. Структурная схема алгоритма функции main

## Функция Errors

Функция вывода ошибок по коду

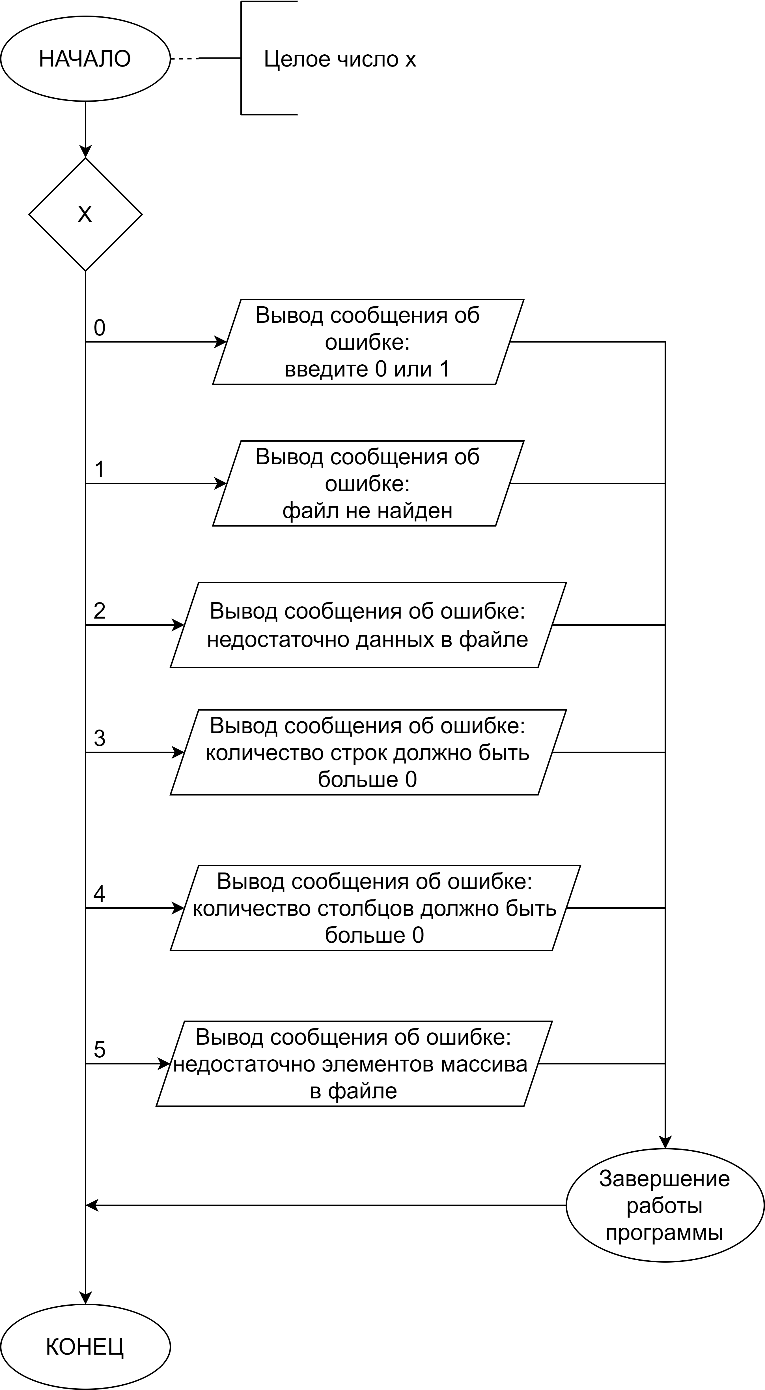


Рисунок 2. Структурная схема алгоритма функции Errors

## Функция memory\_for\_matrix

Функция выделения памяти для матрицы

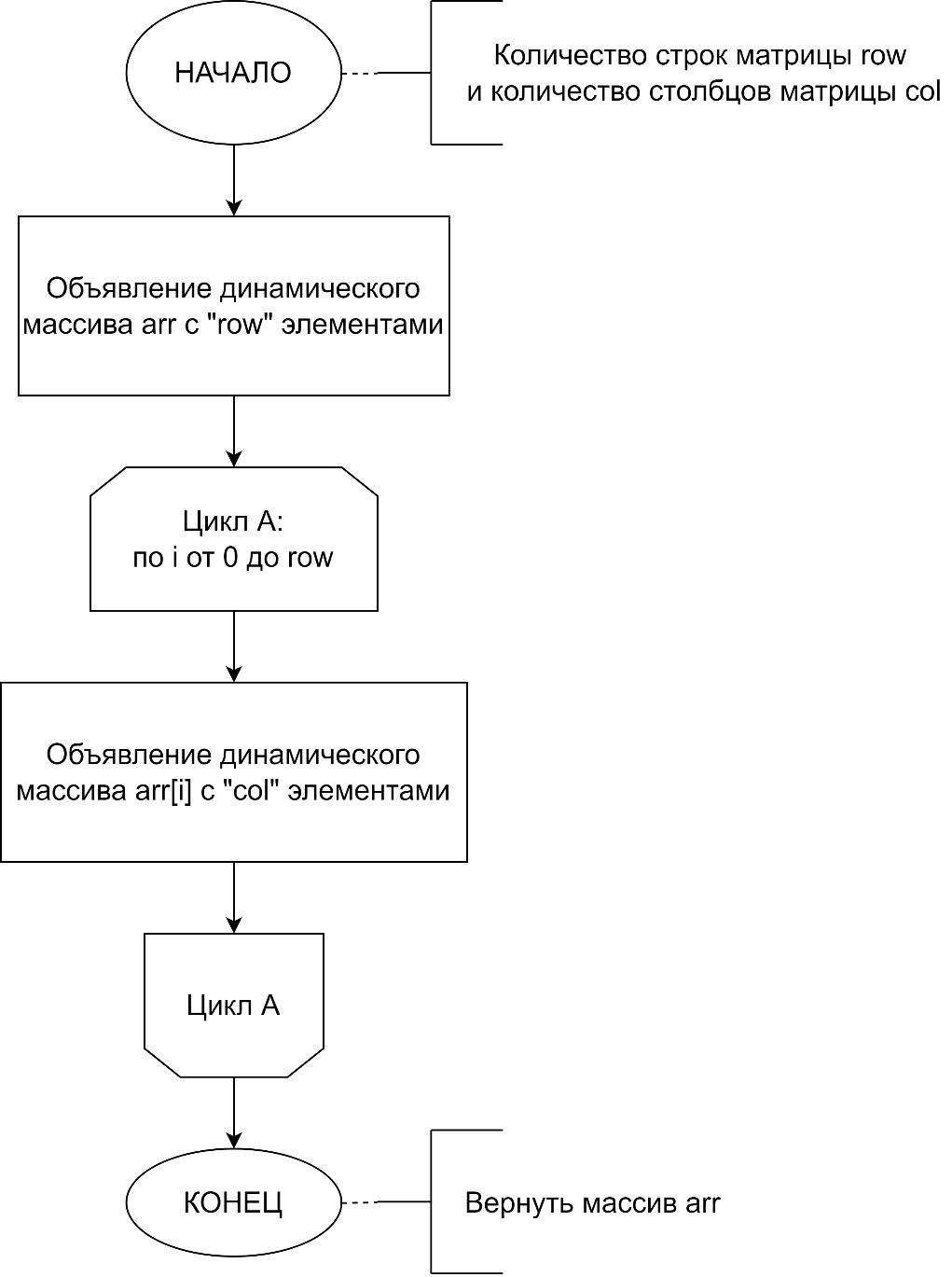


Рисунок 3. Структурная схема алгоритма функции memory\_for\_matrix

## Функция input\_matrix

Ввод матрицы из консоли

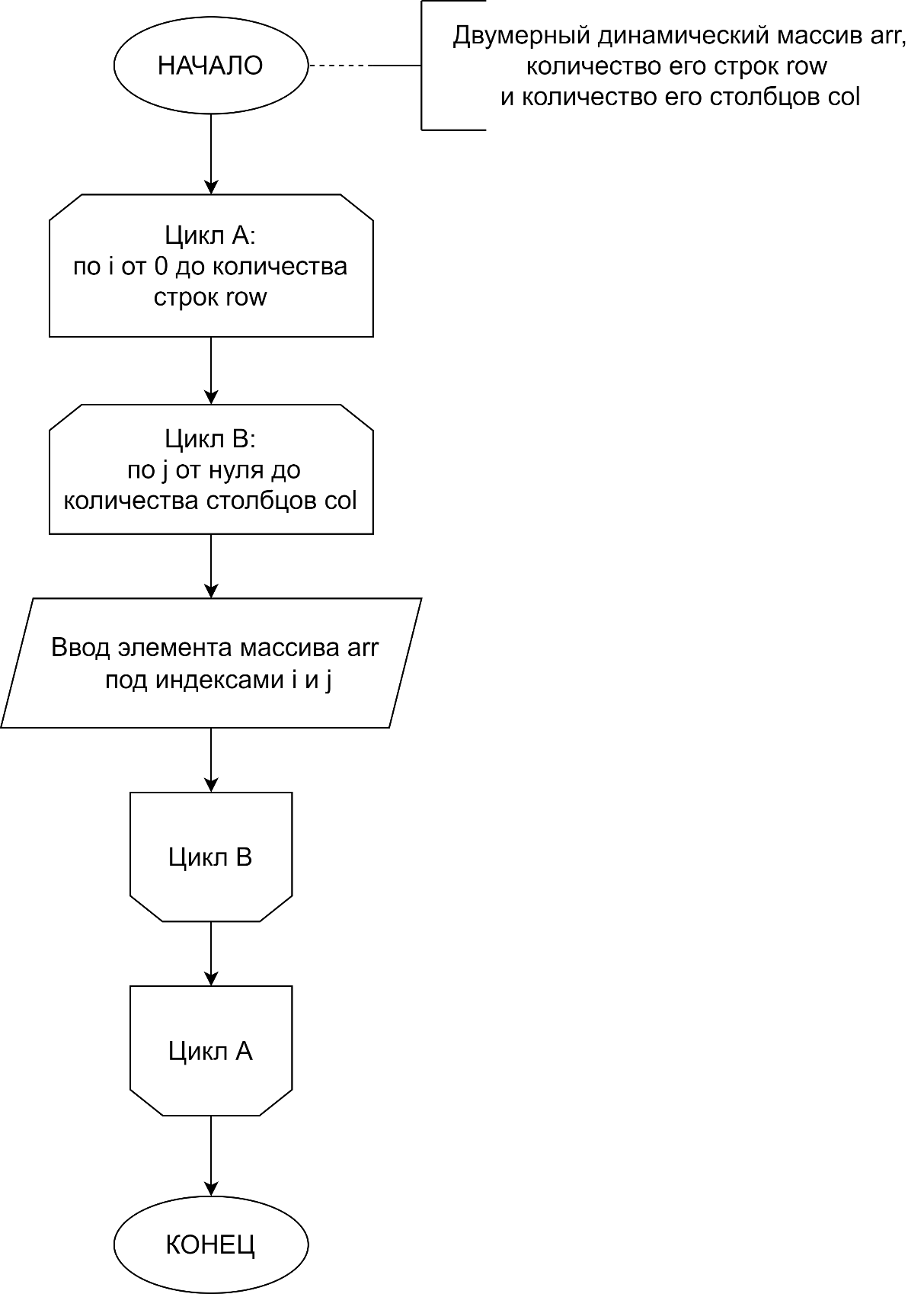


Рисунок 4. Структурная схема алгоритма функции input\_matrix

## Функция matrix\_from\_file

Ввод матрицы из файла

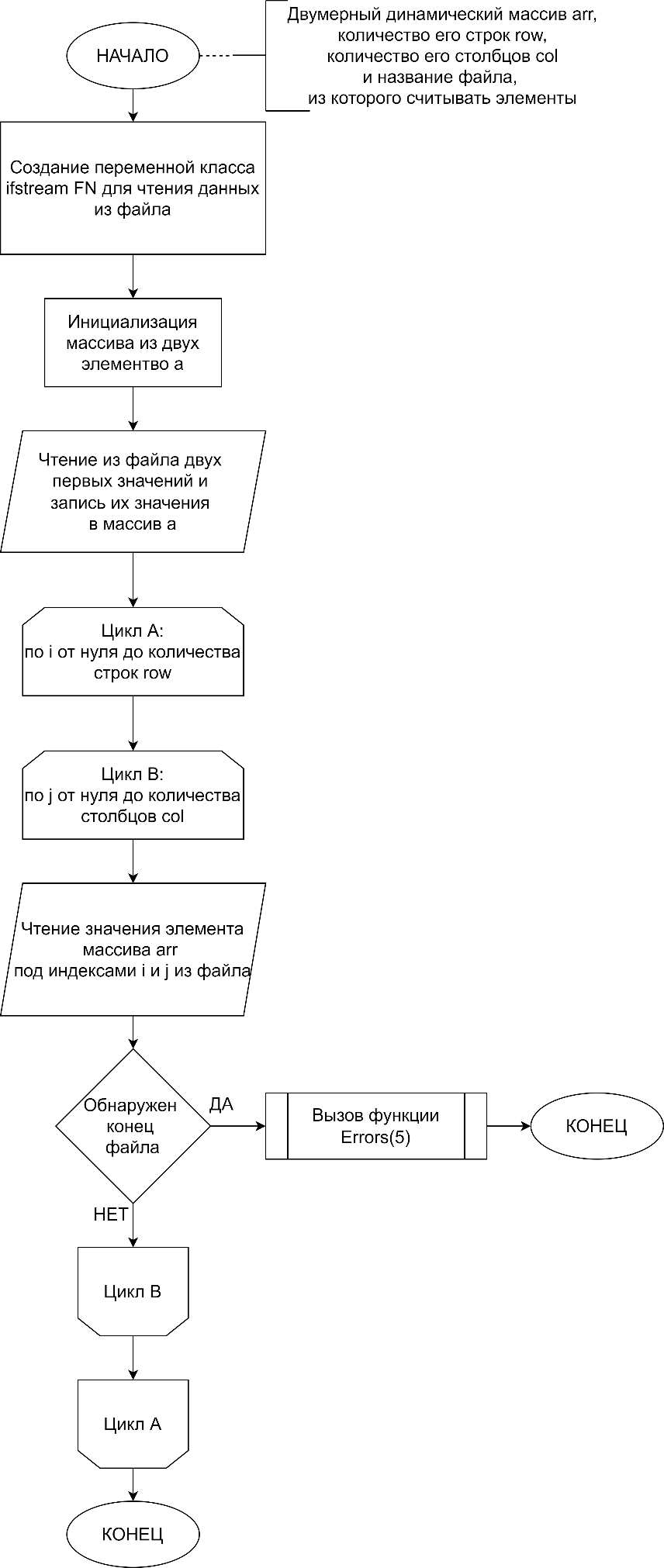


Рисунок 5. Структурная схема алгоритма функции matrix\_from\_file

## Функция print\_matrix

Печать матрицы

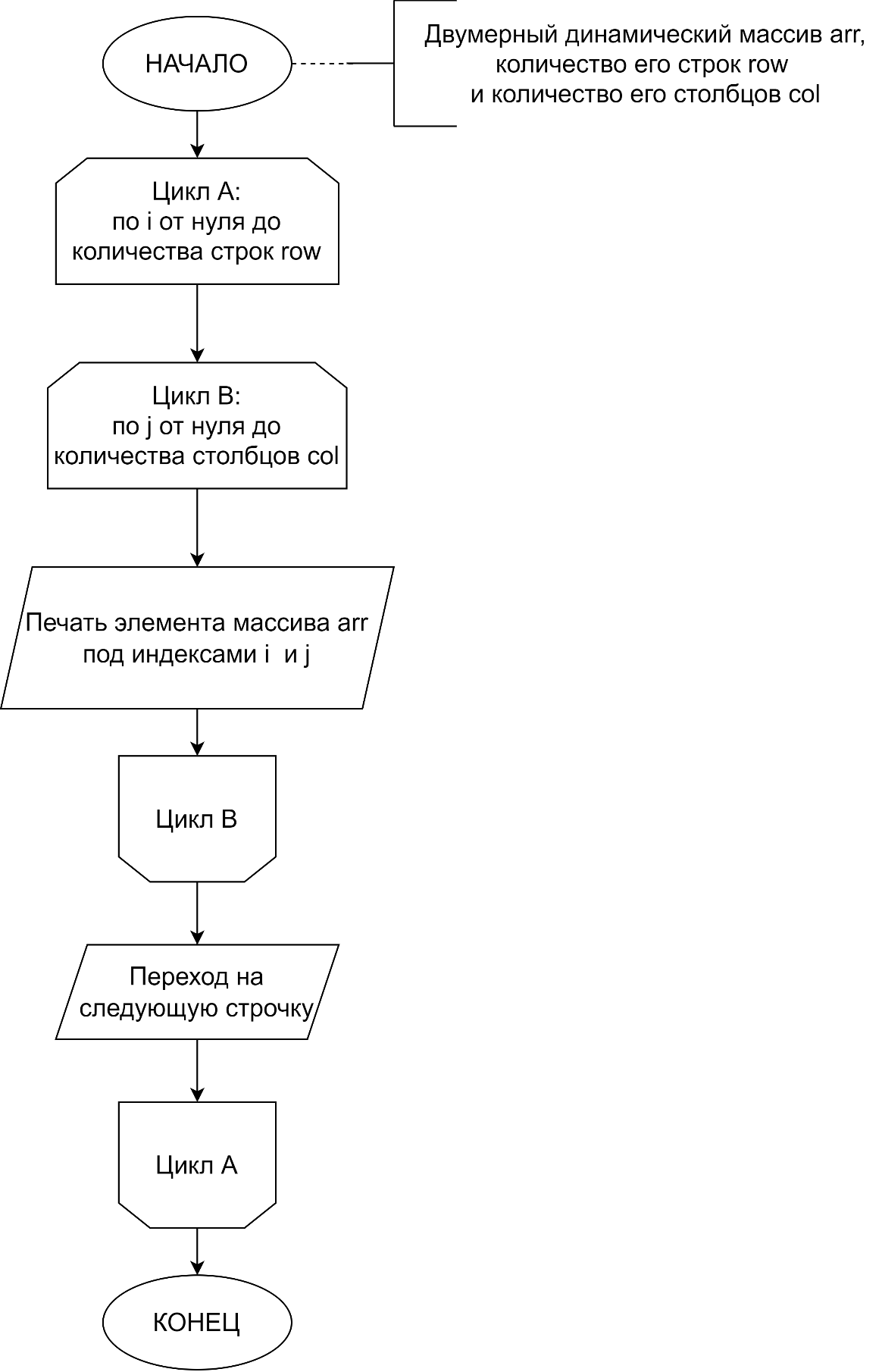


Рисунок 6. Структурная схема алгоритма функции print\_matrix

## Функция maxim

Нахождение максимума среди двух чисел

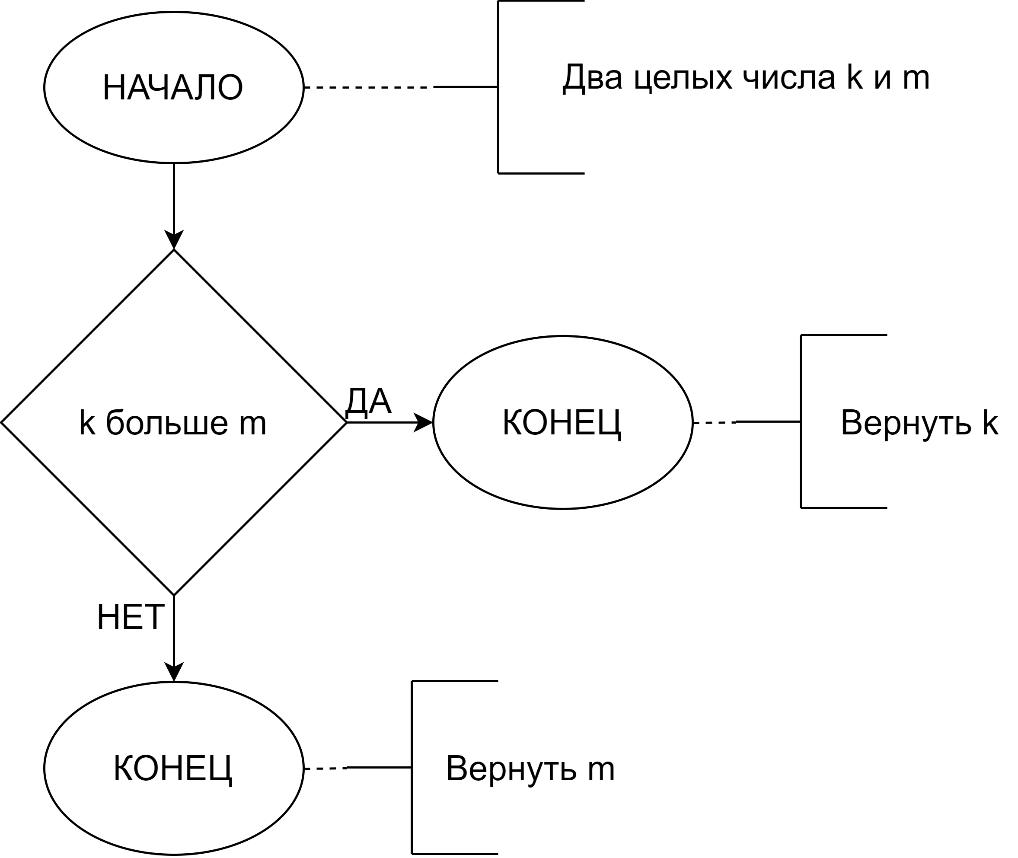


Рисунок 7. Структурная схема алгоритма функции maxim

## Функция product

Нахождение произведения чисел в строках матрицы

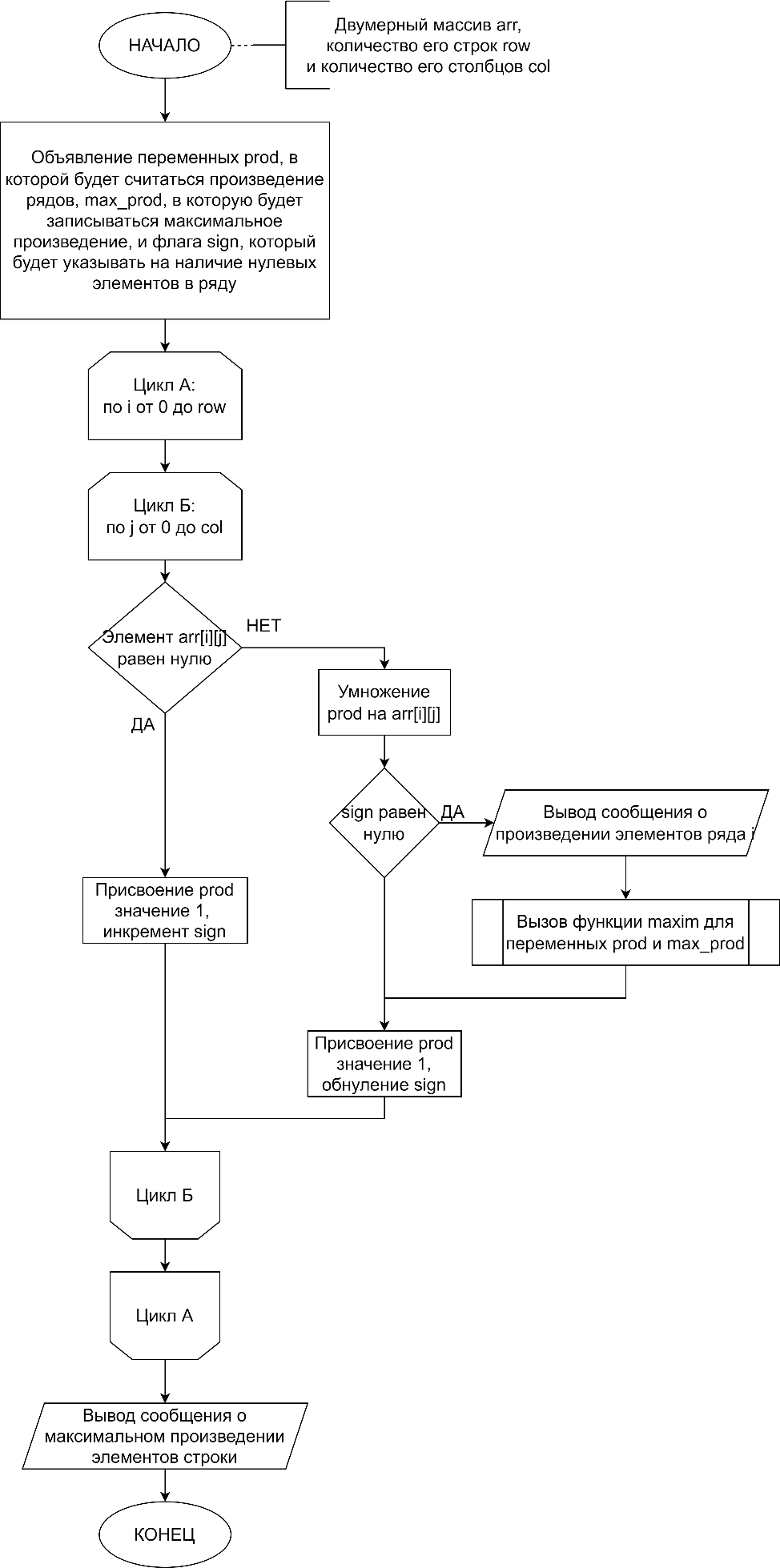


Рисунок 8. Структурная схема алгоритма функции product

# Код программы

#include <iostream>

#include <iomanip>

#include <fstream>

#include <regex>

using namespace std;

// псевдографика - заголовок таблицы

void HeadTable() {

cout << "\t"

<< char(218) << setfill(char(196)) << setw(20)

<< char(194) << setfill(char(196)) << setw(20)

<< char(194) << setfill(char(196)) << setw(20)

<< char(194) << setfill(char(196)) << setw(20)

<< char(194) << setfill(char(196)) << setw(20)

<< char(191) << endl;

cout << "\t"

<< char(179) << setfill(' ') << setw(19) << "#"

<< char(179) << setfill(' ') << setw(19) << "marka LA"

<< char(179) << setfill(' ') << setw(19) << "bort number"

<< char(179) << setfill(' ') << setw(19) << "pas"

<< char(179) << setfill(' ') << setw(19) << "massa"

<< char(179) << endl;

cout << "\t"

<< char(195) << setfill(char(196)) << setw(20)

<< char(197) << setfill(char(196)) << setw(20)

<< char(197) << setfill(char(196)) << setw(20)

<< char(197) << setw(20)

<< char(197) << setw(20) << char(180) << endl;

cout.setf(ios::showpoint | ios::fixed);

}

// псевдографика - завершение таблицы

void BottomTable() {

cout << "\t"

<< char(192) << setfill(char(196)) << setw(20)

<< char(193) << setfill(char(196)) << setw(20)

<< char(193) << setfill(char(196)) << setw(20)

<< char(193) << setfill(char(196)) << setw(20)

<< char(193) << setfill(char(196)) << setw(20)

<< char(217) << endl;

}

// псевдографика - тело таблицы

void BetweenTheRaws() {

cout << "\t"

<< char(195) << setfill(char(196)) << setw(20)

<< char(197) << setfill(char(196)) << setw(20)

<< char(197) << setfill(char(196)) << setw(20)

<< char(197) << setfill(char(196)) << setw(20)

<< char(197) << setfill(char(196)) << setw(20)

<< char(180) << endl;

}

// вывод строк таблицы с закрытием таблицы

void PrintTable(int n, char m[], char k[], int kol, double gruz, bool last) {

cout << "\t"

<< char(179) << setw(19) << setfill(char(255)) << n

<< setfill(char(255))

<< char(179) << setw(19) << setprecision(0) << m

<< setfill(char(255))

<< char(179) << setw(19) << setprecision(4) << k

<< setfill(char(255))

<< char(179) << setw(19) << setprecision(4) << kol

<< setfill(char(255))

<< char(179) << setw(19) << setprecision(1) << gruz

<< setfill(char(255)) << char(179) << endl;

if (last) {

BottomTable();

}

else {

BetweenTheRaws();

}

}

// ФУНКЦИЯ ДЛЯ ВЫВОДОВ ОШИБКИ ПО КОДАМ

void Errors(int x) {

setlocale(LC\_ALL, "rus");

switch (x) {

case 1:

cout << "Ошибка! Файл не найден!" << endl;

exit(1);

case 2:

cout << "Ошибка! Неверная марка ЛА" << endl << endl;

exit(2);

case 3:

cout << "Ошибка! Неверный бортовой номер" << endl << endl;

exit(3);

case 4:

cout << "Ошибка! Неверное количество пассажиров" << endl << endl;

exit(4);

case 5:

cout << "Ошибка! Неверное масса груза" << endl << endl;

exit(5);

}

}

// структура

struct aero {

char m[8];

char n[6];

int kol;

double gruz;

};

// проверка марки ЛА

bool valid\_aircrafttype(char candidate[]) {

regex r ("\\w{2}-\\d{3}\\w");

return regex\_match(candidate, r);

}

// проверка бортового номера

bool valid\_aircraftnum(char candidate[]) {

regex r("\\w-\\d+");

return regex\_match(candidate, r);

}

// проверка количества пассажиров

bool valid\_passangernum(int candidate) {

if (candidate >= 0) return true;

else return false;

}

// проверка количества груза

bool valid\_cargonum(double candidate) {

if (candidate >= 0) return true;

else return false;

}

// чтение структуры из файла

void str\_from\_file(const string file, aero\* mas) {

ifstream fil;

fil.open(file);

int M = 0;

fil >> M;

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

fil >> mas[i].m;

if (!valid\_aircrafttype(mas[i].m)) {

cout << endl << mas[i].m << endl;

Errors(2);

}

fil >> mas[i].n;

if (!valid\_aircraftnum(mas[i].n)) {

cout << endl << mas[i].n << endl;

Errors(3);

}

fil >> mas[i].kol;

if (!valid\_passangernum(mas[i].kol)) {

cout << endl << mas[i].kol << endl;

Errors(4);

}

fil >> mas[i].gruz;

if (!valid\_cargonum(mas[i].gruz)) {

cout << endl << mas[i].gruz << endl;

Errors(5);

}

}

}

// считывание бортовой номер

int num\_from\_struct(aero mas) {

int number = atoi(&mas.n[2]);

return number;

}

//линейная сортировка структуры по индексам

void sort\_the\_struct(aero\* mas, int M, int \*rar) {

int Temp;

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

for (int j = M - 1; j > i; j--) {

if (num\_from\_struct(mas[rar[j]]) < num\_from\_struct(mas[rar[j-1]])) {

Temp = rar[j];

rar[j] = rar[j - 1];

rar[j - 1] = Temp;

}

}

}

}

//const string filename = "file.txt";

const string filename = "file2.txt";

//const string filename = "wrong1.txt";

//const string filename = "wrong2.txt";

//const string filename = "wrong3.txt";

//const string filename = "wrong0.txt";

int main()

{

int pas = 0; // количество пассажиров

double allgruz = 0; // масса груза

ifstream fn; // создание переменной класса ifstream

fn.open(filename); // открытие файла

if (!fn.is\_open()) // ошибка не найденного файла

Errors(1);

int N; // количество строк данных в файле

fn >> N; // чтение количества строк данных в файле

int\* rar = new int[N]; // массив индексов для сортировки по индексам

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

rar[i] = i; // заполенение массива индексов

aero\* arr = new aero[N];

str\_from\_file(filename, arr);

HeadTable(); // печать оглавления таблицы

bool last = 0; // флаг конца таблицы

sort\_the\_struct(arr, N, rar); // сортировка структуры по индексам

int k;

for (int i = 0; i < N; i++) { // подсчет суммы и вывод таблицы

k = rar[i];

pas += arr[k].kol;

allgruz += arr[k].gruz;

PrintTable(i + 1, arr[k].m, arr[k].n, arr[k].kol, arr[k].gruz, last);

if (i == N - 2)

last = 1;

}

setlocale(LC\_ALL, "rus");

cout << "\n\tСуммарное количество пассажиров: " << pas << "\n\tСуммарный вес груза: " << setprecision(1) << allgruz << "\n\n";

system("pause");

}

# Тестирование программы

## Тестирование некорректных режимов

**Тест №1**

Цель теста: проверить работу программы при отсутствии файла

Исходные данные:

Ожидаемый результат: вывод сообщения об ошибке: «Ошибка! Файл не найден!»

Полученный результат:



Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

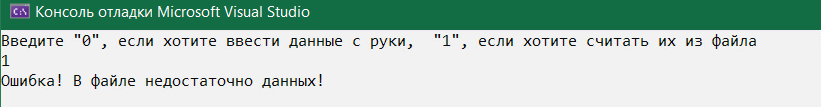
**Тест №2**

Цель теста: проверить работу программы при пустом файле

Исходные данные: (пустой файл)

Ожидаемый результат: вывод сообщения об ошибке: «Ошибка! В файле недостаточно данных!»

Полученный результат:



Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

**Тест №3**

Цель теста: проверить работу программы при отсутствии некоторых данных

Исходные данные: 1

Ожидаемый результат: вывод сообщения об ошибке: «Ошибка! В файле недостаточно данных!»

Полученный результат:



Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

**Тест №4**

Цель теста: проверить работу программы при нулевом количестве строк

Исходные данные: 0 5

Ожидаемый результат: вывод сообщения об ошибке: «Ошибка! Количество строк должно быть больше нуля!»

Полученный результат:



Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

**Тест №5**

Цель теста: проверить работу программы при нулевом количестве столбцов

Исходные данные: 6 0

Ожидаемый результат: вывод сообщения об ошибке: «Ошибка!Количество столбцов должно быть больше нуля!»

Полученный результат:



Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

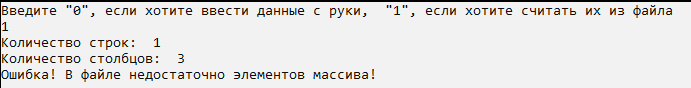
**Тест №6**

Цель теста: проверить работу программы при недостаточном количестве элементов массива

Исходные данные: 1 3 2 3

Ожидаемый результат: вывод сообщения об ошибке: «Ошибка! В файле недостаточно элементов массива!»

Полученный результат:



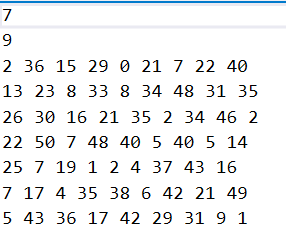
Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

## Тестирование корректных режимов

**Тест №1**

Цель теста: проверить работу программы в корректной области исходных данных

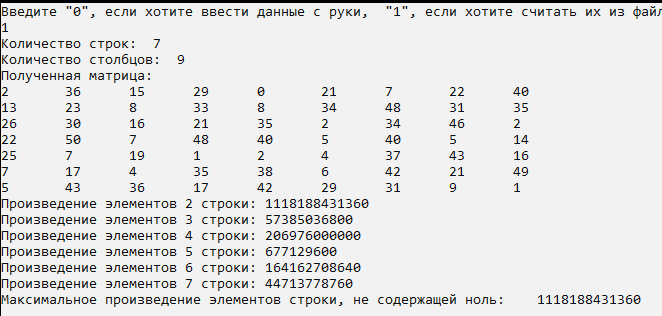
Исходные данные:



Ожидаемый результат:

Максимальное произведение элементов строки, не содержащей ноль: 1118188431360

Полученный результат:

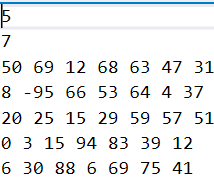


Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

**Тест №2**

Цель теста: проверить работу программы в корректной области исходных данных

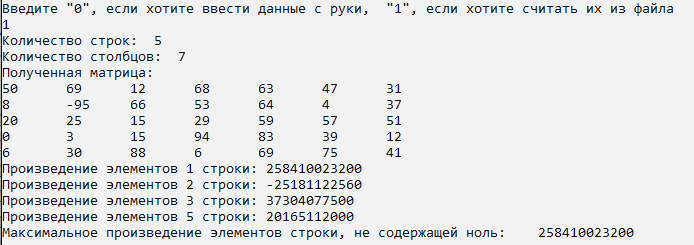
Исходные данные:



Ожидаемый результат:

Максимальное произведение элементов строки, не содержащей ноль: 258410023200

Полученный результат:



Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

**Тест №3**

Цель теста: проверить работу программы в корректной области исходных данных

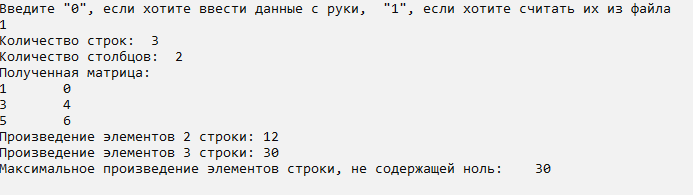
Исходные данные:



Ожидаемый результат:

Максимальное произведение элементов строки, не содержащей ноль: 30

Полученный результат:



Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

# Вывод

Был изучен параметризированный алгоритм работы с двумерным массивом, ввод данных в двумерный массив из консоли и из файла. Были составлены структурные схемы алгоритмов функций для дальнейшего написания кода. В результате проведения тестов ошибок в программе не обнаружено. Разработка программы завершена на основании того, что полученные результаты совпали с ожидаемыми, набор тестов считаем полным.