Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Иркутский национальный исследовательский технический университет»

|  |
| --- |
| Институт информационных технологий и анализа данных |
| (наименование) |

**О Т Ч Ё Т**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| о прохождении | | | | учебной практики |
|  | | | (вид практики: учебная/производственная) | |
| по получению первичных профессиональных умений и навыков, в том числе | | | | |
| практики первичных умений и навыков научно-исследовательской | | | | |
| деятельности | | | | |
| (тип практики: технологическая/научно-исследовательская работа/преддипломная и др.) | | | | |
| на | | Институт информационных технологий и анализа данных | | |
|  | (наименование профильной организации) | | | |

|  |  |
| --- | --- |
| Обучающегося | Солопова Д.Д., ИСТб-19-2 |
|  | (ФИО, группа, подпись) |

|  |  |
| --- | --- |
| Руководитель практики | |
| Осипова Е.А., доцент | |
| ( ФИО, должность, подпись) | |
| Допущен к защите | |
| Осипова Е.А., | |
| (ФИО, подпись, дата) | |
| Оценка по практике |  |
| Осипова Е.А., | |
| (ФИО, подпись, дата) | |

|  |  |  |
| --- | --- | --- |
| Содержание отчета на |  | стр. |
|  |  |  |

Иркутск 2020

Содержание

[Введение 3](#_Toc52212714)

[1 Постановка задачи 4](#_Toc52212716)

[2 Таблица спецификаций 4](#_Toc52212717)

[3 Блок-схема алгоритма решения задания 7](#_Toc52212718)

[4 Таблица тестов 18](#_Toc52212719)

[5 Исходный код 24](#_Toc52212720)

[6 Анализ вычислительной сложности алгоритма 34](#_Toc52212721)

[7 Результаты тестирования 37](#_Toc52212723)

[8 Заключение 45](#_Toc52212724)

[9 Список использованных источников 46](#_Toc52212725)

Введение

Целью практики является совершенствование навыков алгоритмизации, тестирования и программирования на языке высокого уровня, а также закрепление знаний, полученных при изучении дисциплин «Информатика» и «Программирование».

Основными задачами учебной практики являются приобретение практических навыков:

* рационального использования рабочего времени;
* культуры труда;
* разработки алгоритмов решения задач;
* использования различных методов тестирования алгоритмов;
* применения языка программирования высокого уровня;
* качественного выполнения заданий;

межличностных отношений при коллективном выполнении задания.

1 Постановка задачи

**Задача.** Вычислить обратную матрицу методом алгебраических дополнений. Размер матрицы задается пользователем. Реализовать два варианта ввода значений первоначальной матрицы: ввод значений пользователем вручную, случайная генерация элементов в заданном диапазоне.

# **2** **Таблица спецификаций**

Таблица 1 – Таблица спецификаций

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Имя** | **Назначение** | **Тип** | **ОДЗ** |
| **int main() – точка входа в программу** | | | | |
| 1 | size | Входные данные — размер функции | int | [2 ; 10] |
| 2 | сheck | Входные данные – флаг определения типа записи чисел в матрицу | int | [0; 1] |
| 3 | eps | Входные данные – количество знаков после запятой для вывода элементов матрицы на экран | int | [0 ; 5] |
| 4 | d\_min | Входные данные – минимальное значение диапазона случайных чисел | int | (-100000; 100000) |
| 5 | d\_max | Входные данные – максимальное значение диапазона случайных чисел | int | (-100000; 100000) |
| 6 | matrix | Входные данные – квадратная матрица размера size | double | (-100000; 100000)(не более 21 знак) |
| 7 | unMatrix | Выходные данные – квадратная матрица размера size, которая обратна матрице matrix | double | (-100000; 100000)(не более 21 знак) |
| **getIntValue() – функция возвращающая целое число** | | | | |
| 1 | zero | Входные данные – идентификатор характеризующий натуральность вводимых чисел(не включая 0) | bool | true/false |
| 2 | value | Выходные данные – результат корректного ввода целого числа | int | (-100000; 100000) |
| **getDoubleValue() – функция возвращающая вещественное число** | | | | |
| 1 | value | Выходные данные – результат корректного ввода вещественного числа | double | (-100000; 100000)(не более 21 знак) |
| **createMatrix() – функция создания квадратной матрицы** | | | | |
| 1 | size | Входные данные – число, характеризующая размер квадратной матрицы | int | [−2 147 483 648, +2 147 483 647] |
| 2 | user | Входные данные – флаг ввода(отвечает за то, каким образом данные будут поступать в матрицу: с помощью пользователя или рандомного заполнения) | bool | true/false |
| 3 | min | Входные данные – минимальное число из диапазона рандомных значений(если пользователь вводит значения в матрицу данный параметр вместе с max игнорируется) | int | [−2 147 483 648, +2 147 483 647] |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | max | Входные данные – максимальное число из диапазона рандомных чисел | int | [−2 147 483 648, +2 147 483 647] |
| 5 | matrix | Выходные данные – матрица размера size заполненная одним из допустимых способов: с помощью пользователя или рандомными числами из диапазона [min;max] | double | [1.17E-38…3,4E+38] |
| **showMatrixInConsole() – функция выводящая квадратную матрицу на консоль** | | | | |
| 1 | matrix | Входные данные – квадратная матрица | double | [1.17E-38…3,4E+38] |
| 2 | size | Входные данные – размер квадратной матрицы | int | [−2 147 483 648, +2 147 483 647] |
| 3 | eps | Входные данные – точность вывода элементов матрицы на экран | int | [−2 147 483 648, +2 147 483 647] |
| **getMatrixMinor() – функция поиска минора в квадратной матрице** | | | | |
| 1 | mas | Входные данные – матрица, в которой осуществляется поиск минора | double | [1.17E-38…3,4E+38] |
| 2 | p | Выходные данные – матрица, в которую будет записан минор | double | [1.17E-38…3,4E+38] |
| 3 | i | Входные данные – значение отвечающее на вопрос “по какой строке найти минор?” | int | [−2 147 483 648, +2 147 483 647] |
| 4 | j | Входные данные – значение отвечающее на вопрос “по какому столбцу найти минор?” | int | [−2 147 483 648, +2 147 483 647] |
| 5 | m | Входные данные – размер квадратной матрицы | int | [−2 147 483 648, +2 147 483 647] |
| **Determinant() – функция вычисляющая определитель квадратной матрицы** | | | | |
| 1 | mas | Входные данные – матрица определитель которой нужно найти | double | [1.17E-38…3,4E+38] |
| 2 | m | Входные данные – размер квадратной матрицы | int | [−2 147 483 648, +2 147 483 647] |
| 3 | d | Выходные данные – определитель квадратной матрицы | double | [1.17E-38…3,4E+38] |
| **transportantMatrix() – функция вычисляющая транспонированную матрицу** | | | | |
| 1 | mas | Входные данные – матрица, чью транспонированную матрицу нужно найти | double | [1.17E-38…3,4E+38] |
| 2 | s | Входные данные – размер квадратной матрицы | int | [−2 147 483 648, +2 147 483 647] |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | tr | Выходные данные – транспонированная матрица относительно данной | double | [1.17E-38…3,4E+38] |
| **unionMatrix() – функция вычисляющая матрицу, каждый элемент которой состоит из алгебраических дополнений данной матрицы** | | | | |
| 1 | mas | Входные данные – матрица, чьи алгебраические дополнения будут элементами результирующей матрицы | double | [1.17E-38…3,4E+38] |
| 2 | s | Входные данные – размер квадратной матрицы | int | [−2 147 483 648, +2 147 483 647] |
| 3 | unMatrix | Выходные данные – результирующая матрица, элементы которой равны алгебраическим дополнениям матрицы mas | double | [1.17E-38…3,4E+38] |
| **matrixValueMultiplication() – функция умножающая элементы матрицы на число** | | | | |
| 1 | matrix | Выходные данные – матрица, элементы которой нужно умножить на value | double | [1.17E-38…3,4E+38] |
| 2 | value | Входные данные – число, на которое нужно умножить элементы матрицы matrix | double | [1.17E-38…3,4E+38] |
| 3 | size | Входные данные – размер квадратной матрицы | int | [−2 147 483 648, +2 147 483 647] |
| **matrixMultiplication() – функция перемножающая две матрицы** | | | | |
| 1 | matrix1 | Входные данные – первая квадратная матрица участвующая в умножении | double | [1.17E-38…3,4E+38] |
| 2 | matrix2 | Входные данные – вторая квадратная матрица участвующая в умножении | double | [1.17E-38…3,4E+38] |
| 3 | size | Входные данные – размер квадратных матриц | int | [−2 147 483 648, +2 147 483 647] |
| 4 | mult | Выходные данные – матрица, полученная в результате умножения матрицы matrix1 на матрицу matrix2 | double | [1.17E-38…3,4E+38] |

3 Блок-схема алгоритма решения задания

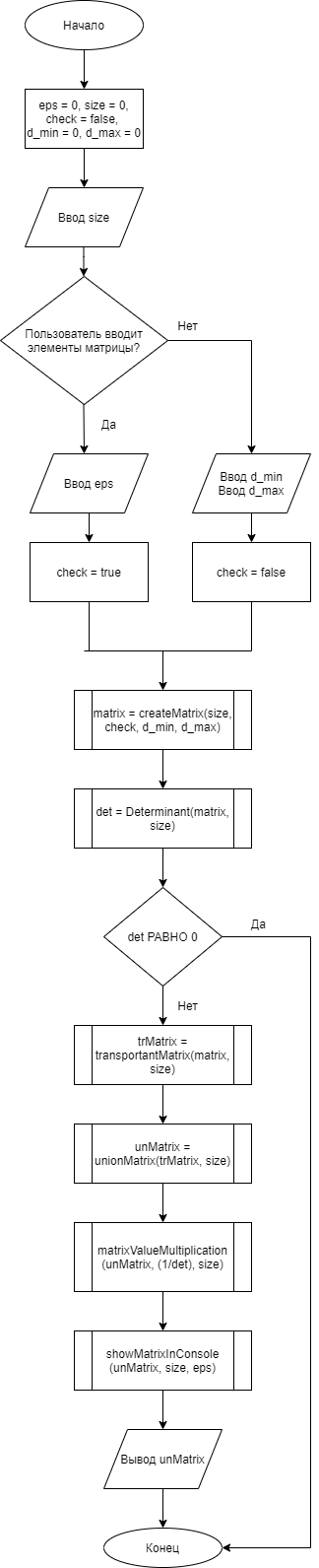


Рисунок 1 – Блок-схема алгоритма решения

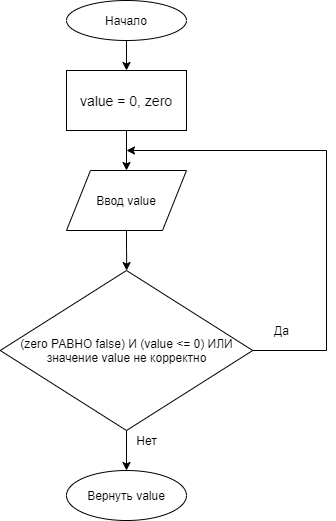


Рисунок 2 – Блок-схема функции getIntValue

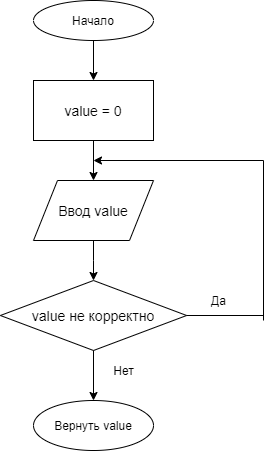


Рисунок 3 – Блок-схема функции getDoubleValue

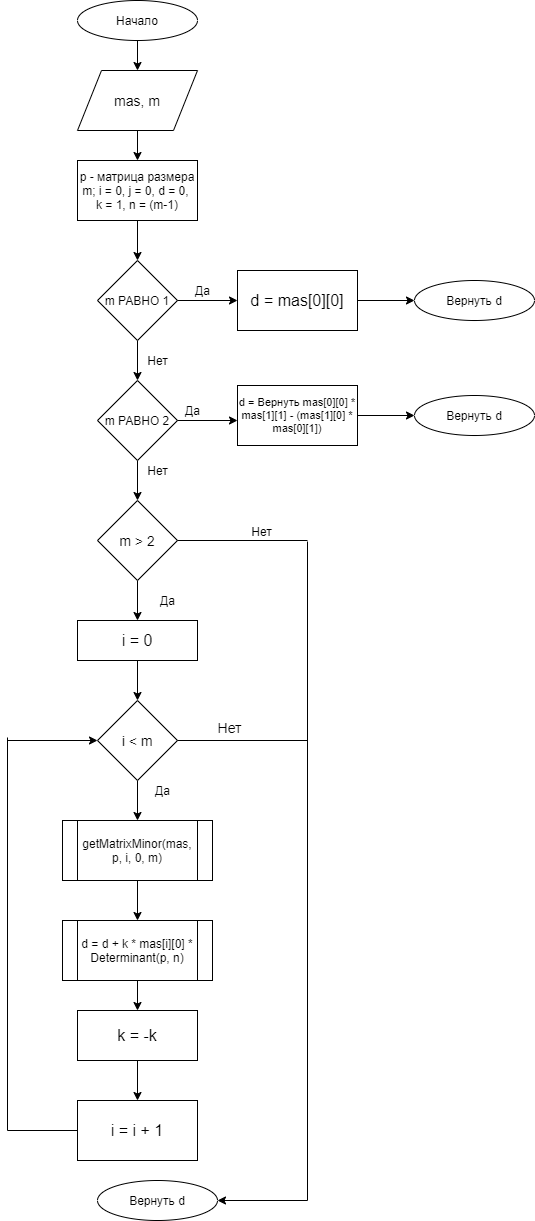


Рисунок 4 – Блок-схема функции Determinant

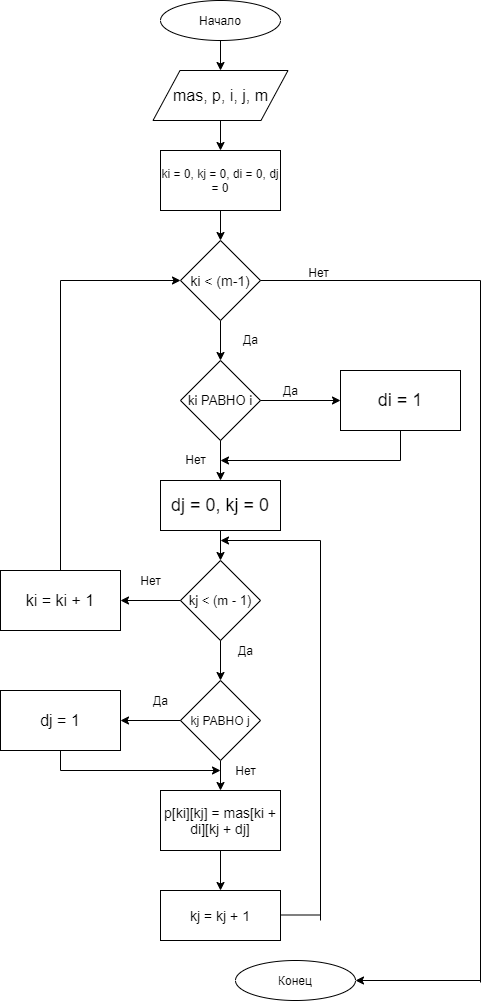


Рисунок 5 – Блок-схема функции getMatrixMinor

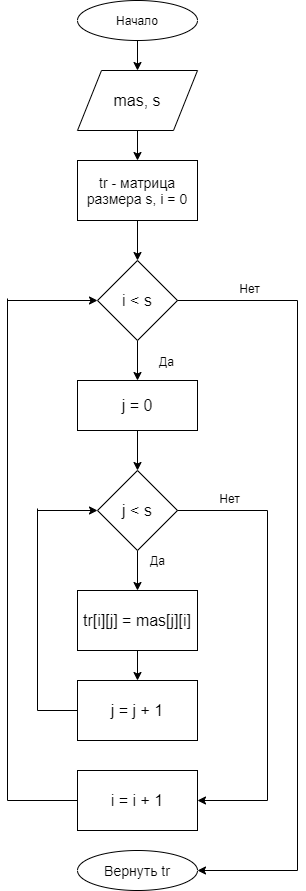


Рисунок 6 - Блок-схема функции transportantMatrix

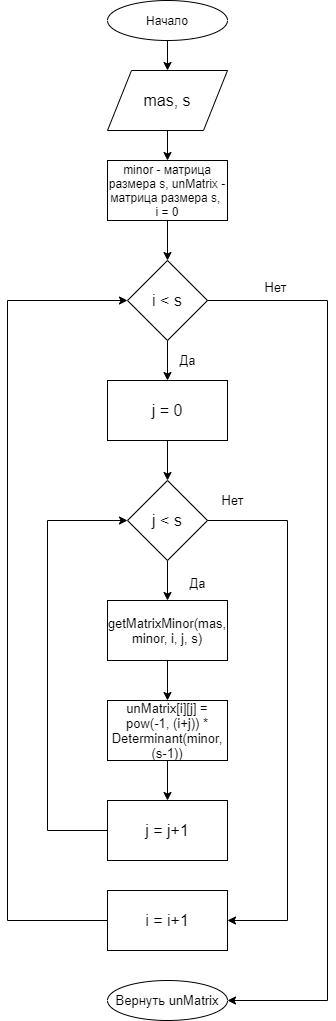


Рисунок 7 – Блок-схема функции unionMatrix

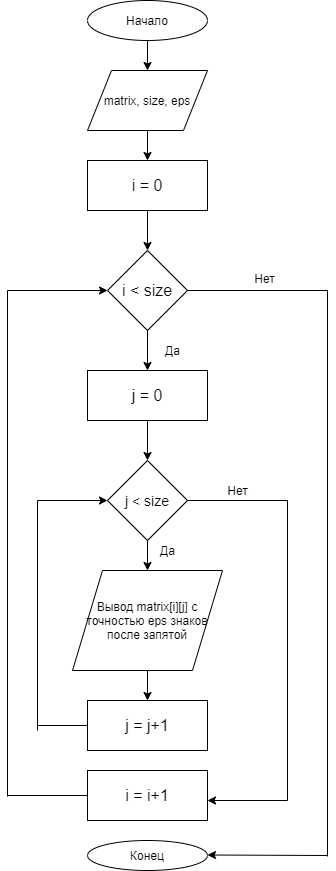


Рисунок 8 – Блок-схема функции showMatrix

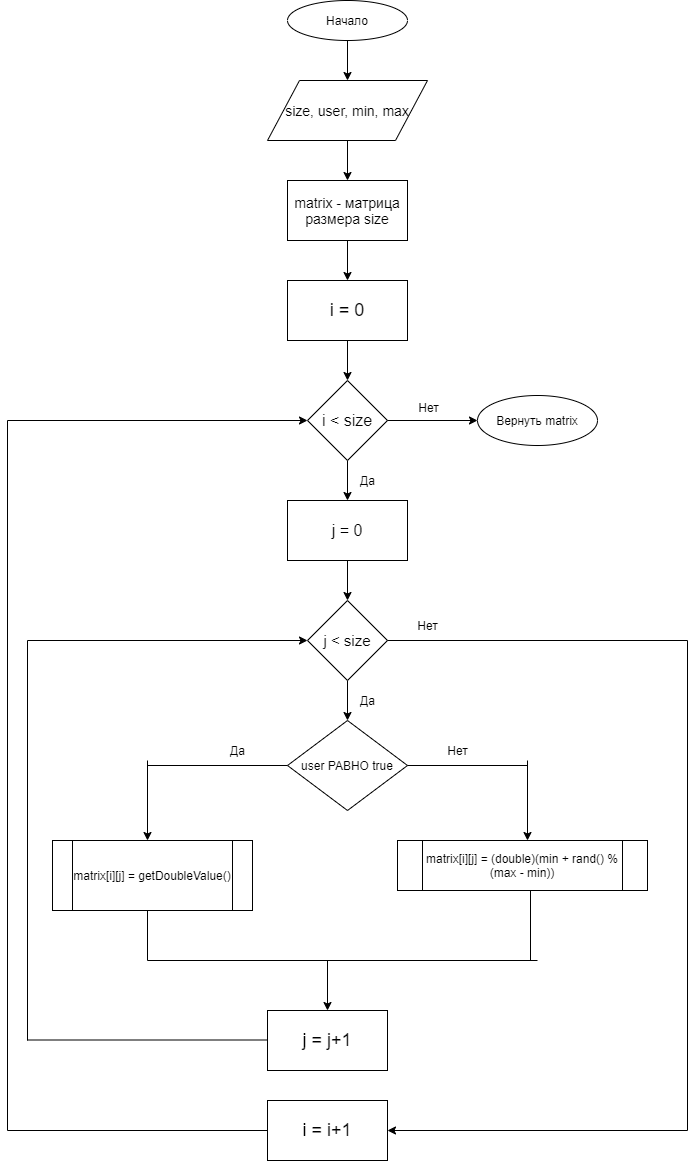


Рисунок 9 – Блок-схема функции createMatrix

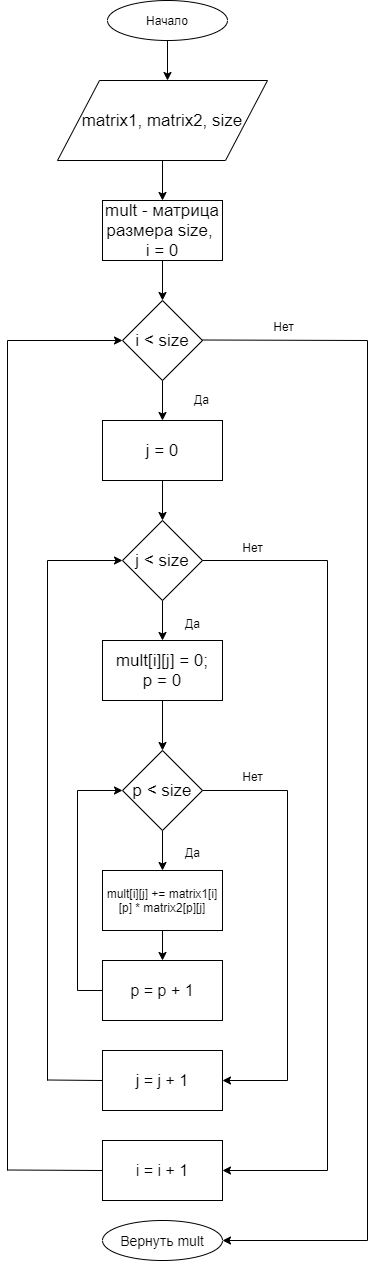


Рисунок 10 – Блок-схема функции matrixMultiplication

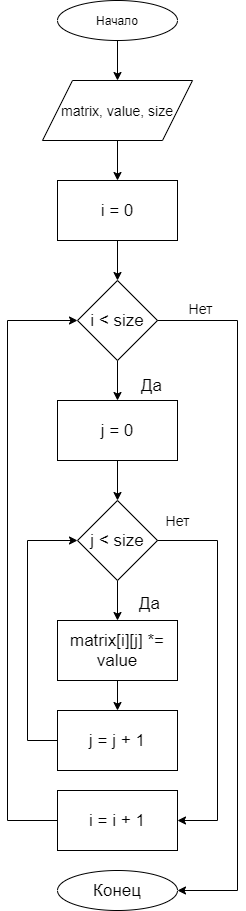


Рисунок 11 – Блок-схема функции matrixValueMultiplication

# **4** **Таблица тестов**

В соответствии с рекомендованной стратегией тестирования были определены правильные и неправильные классы эквивалентности для входных данных на основе анализа постановки задачи (см. таблицу 2). В связи с тем, что алгоритм реализует разветвляющийся процесс, в описании классов эквивалентности приводятся только значимые параметры, то есть те, которые влияют на результат функции. Например, eps используется только при выводе значений матрицы на экран и никак не влияет на значения матрицы. Таким образом, коэффициенты, не указанные в описании классов эквивалентности могут принимать любые значения (в ОДЗ).

\*Чтобы определить какой метод ввода значений в матрицу используется в каждом тесте, обратите внимание на переменную check. Если check равна false, то используется рандомное заполнение матрицы, иначе элементы матрицы вводятся пользователем. В случае, когда check равен true, то eps также вводится пользователем(это сделано для упрощения чтения когда check равен true в тестах не указано чему равно eps, а когда check равен false не указано чему равно eps, т.к. eps будет равен нулю).

Таблица 2 – Выделенные классы эквивалентности

|  |  |  |
| --- | --- | --- |
| № | Правильные классы эквивалентности | Неправильные классы эквивалентности |
| 1 | №1  2 <= size <= 10  сheck = 1,  0 <= eps <= 5  [значения матрицы введены успешно и корректно] | №2  size > 10 или size < 2  сheck = 1  0 <= eps <= 5  [значения матрицы введены успешно и корректно] |
| 2 | - | №3  2 <= size <= 10  сheck > 1 или сheck < 0  0 <= eps <= 5  [значения матрицы введены успешно и корректно] |
| 3 | - | №4  2 <= size <= 10  сheck = 1  eps < 0 или eps > 5  [значения матрицы введены успешно и корректно] |

Продолжение таблицы 2

|  |  |  |
| --- | --- | --- |
| 4 |  | №5  2 <= size <= 10  сheck = 1  0 <= eps <= 5  [значения матрицы введены не корректно] |
| 5 | №6  2 <= size <= 10  сheck = 0  eps = 0  (-100000 < d\_min < 100000)  (-100000 < d\_max < 100000)  d\_min < d\_max | №7  2 <= size <= 10  сheck = 0  eps = 0  значение d\_min вне ОДЗ  или значение d\_max вне ОДЗ  d\_min < d\_max |
| 6 | - | №8  2 <= size <= 10  сheck = 0  eps = 0  (-100000 < d\_min < 100000)  (-100000 < d\_max < 100000)  d\_min >= d\_max |
| 7 | №9  2 <= size <= 10  сheck = 0  eps = 0  (-100000 < d\_min < 100000)  (-100000 < d\_max < 100000)  d\_min < d\_max  [значения матрицы сгенерированы таким образом, что её определитель равен 0] | - |
| 8 | №10  2 <= size <= 10  сheck = 1  0 <= eps <= 5  [значения матрицы подобраны таким образом, что её определитель равен 0] | - |

На основе классов эквивалентности были разработаны тесты представленные в таблице 3. Следует отметить что в силу особенностей постановки задачи не на каждый правильный класс эквивалентности получился отдельный тест.

Таблица 3 – Таблица тестов

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Входные данные | Выходные данные |
| 1 | Штатный тест на срабатывание первого условия (правильный класс эквивалентности 1) | size = 2, check = 1,  eps = 5,  [матрица заполненная пользователем]:  17 10  10 10 | 0, 14286 -0,14286  -0,14286 0,24286 |
| 2 | Штатный тест на срабатывание второго условия (правильный класс эквивалентности 6) | size = 4, check = 0,  d\_min = 10,  d\_max = 100,  [матрица заполненная рандомно]:  72 27 37 33  43 55 12 88  81 47 44 30  81 88 52 30 | (для удобства все элементы округлены до 2-го знака)  -0,23 0,01 0,31 -0,10  -0,09 0,01 0,08 -0,01  0,43 -0,04 -0,55 0,19  0,10 0,01 -0,13 0,03 |
| 3 | Штатный тест на срабатывание третьего условия (правильный класс эквивалентности 9) | size = 2  сheck = 0  d\_min = 1  d\_max = 1  [матрица заполненная рандомно]:  1 1  1 1 | Определитель исходной матрицы равен 0. Нахождение обратной матрицы невозможно |
| 4 | Штатный тест на срабатывание четвёртого условия(правильный класс эквивалентности 10) | size = 3  сheck = 1  eps = 2  [матрица заполненная пользователем]:  1 2 3  4 5 6  7 8 9 | Определитель исходной матрицы равен 0. Нахождение обратной матрицы невозможно |
| 5 | Тест на ошибочную ситуацию в первом условии (неправильный класс эквивалентности 2) | size = 90  сheck = 1  eps = 5  [матрица заполненная пользователем]:  ….. | Ошибка: размера матрицы не может быть меньше 2 или больше 10 |

Продолжение таблицы 3

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | Тест на ошибочную ситуацию в первом условии(неправильный класс эквивалентности 3) | size = 3  check = 50  eps = 3  [матрица заполненная ?]:  …. | Ошибка: не корректный ответ на вопрос |
| 7 | Тест на ошибочкую ситуацию в первом условии(неправильный класс эквивалентности 4) | 2 <= size <= 10  сheck = 1  eps = -10  [матрица заполненная пользователем]:  ….. | Ошибка: количество знаков должно быть значением в диапазоне [0; 5] |
| 8 | Тест на ошибочную ситуацию в первом условии(неправильный класс эквивалентности 5) | size = 3  сheck = 1  eps = 2  [матрица заполненная пользователем]:  0 245 24151251  1 2415 215  2 3412 5151 | Ошибка: вводимое число должно быть в диапазоне (-100000 ; 100000) |
| 9 | Тест на ошибочную ситуацию во втором условии(неправильный класс эквивалентности 7) | size = 10  сheck = 0  eps = 0  d\_min = 500  d\_max = 500000  [матрица заполненная рандомно]:  …… | Ошибка: вводимое число должно быть в диапазоне (-100000 ; 100000) |

Продолжение таблицы 3

|  |  |  |  |
| --- | --- | --- | --- |
| 10 | Тест на ошибочную ситуацию во втором условии(неправильный класс эквивалентности 8) | size = 5  сheck = 0  eps = 0  d\_min = 5  d\_max = 5  [матрица заполненная рандомно]:  …. | Ошибка: минимальное значение диапазона рандомных чисел должно быть меньше максимального значения! |
| 11 | Предположение об ошибке(пользователь вводит в матрицу не корректное значение) | size = 2  check = 1  eps = 4  [матрица заполненная пользователем]:  1 hello  faw 39 | Ошибка: введена не корректная строка |
| 12 | Предположение об ошибке(пользователь вводит строку вместо размера) | size = “много”  check = 1  eps = 4  [матрица заполненная пользователем]:  …. | Ошибка: введена не корректная строка |
| 13 | Предположение об ошибке(пользователь вводит строку содержащей число в d\_max) | size = 2  check = 0  eps = 0  d\_min = 50  d\_max = 200fg  [матрица заполненная рандомно]:  …. | Ошибка: введена не корректная строка |

Так же в таблицу тестов было добавлено три теста, полученные методом предположения об ошибке. Случай, при котором пользователь вводит вместо числового значения строку или другие некорректные данные.

Стоит отметить, что вводя любое числовое значение в программе происходит проверка корректности ввода. Это происходит абсолютно для любой переменной: от size до последнего элемента массива matrix(при пользовательском вводе) или до d\_max(при заполнении матрицы рандомными значениями).

Чтобы провести тестирование по модели **белого ящика** построим граф потока управления на основе блок-схема алгоритма. Каждая инструкция представлена отдельным узлом, а ребра показывают направление выполнения алгоритма. Граф потока управления приведен на рисунке 12.

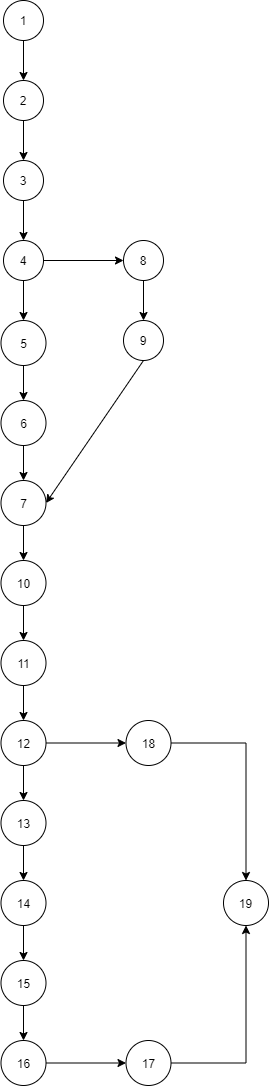


Рисунок 12 — Граф потока управления для алгоритма

В результате выполнения теста 1 выполняется проход по пути *1234567101112131415161719* В результате выполнения теста 2 1234897101112131415161719— путь. Теста 3 —*12348971011121819*. Теста 4 – 12345671011121819. Имеющийся набор тестов обеспечивает покрытие всех возможных путей, поэтому формировать дополнительные тесты не требуется.

5 Исходный код

В исходный алгоритм были внесены изменения для того, чтобы учесть проблемные ситуации выявленные в ходе разработки тестов.

Алгоритм был реализован на языке программирования C, код доступен по ссылке <https://repl.it/join/hspeoype-dansw>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

#include <stdbool.h>

#include <malloc.h>

#include <math.h>

#include <string.h>

#include <errno.h>

#define SIZE\_STR 8

#define SIZE\_STR\_DOUBLE 23

int getIntValue(bool zero) {

  size\_t length = 0;

char \*end = NULL;

char buf[SIZE\_STR] = "";

 int value = 0;

 bool t = true;

 while((fgets(buf, sizeof(buf), stdin)) && (t)){

length = strlen(buf);

t = false;

if (buf[length - 1] == '\n') {

buf[--length] = '\0';

errno = 0;

value = strtol(buf, &end, 10);

if (length == 0) {

fprintf(stderr, "\nОшибка: введена пустая строка.\n");

t = true;

}

if (errno != 0 || \*end != '\0') {

fprintf(stderr, "\nОшибка: введена некорректная строка, число должно быть целочисленным\n");

fprintf(stderr, "\t%s\n" , buf);

fprintf(stderr, "\t%\*c\n", (int)(end - buf) + 1, '^');

t = true;

}

}else {

scanf("%\*[^\n]");

scanf("%\*c");

fprintf(stderr, "Ошибка: запись числа должна содержать не более %d символов.\n", (SIZE\_STR - 2));

t = true;

}

if((!t) && ((value <= (int)(-100000)) || (value >= (int)(100000)))){

  fprintf(stderr, "Ошибка: вводимое число должно быть в диапазоне (-100000; 100000).\n");

  t = true;

  }

if((!t) && (value <= 0) && (!zero)){

  printf("\nОшибка: введите число больше 0\n");

  t = true;

  }

if(t){

  printf("Введите корректное значение: ");

  }

  }

return value;

}

double getDoubleValue(){

  setlocale(LC\_ALL, "rus");

  size\_t length = 0;

char \*end = NULL;

char buf[SIZE\_STR\_DOUBLE] = "";

 double value = 0;

 bool t = true;

 while((fgets(buf, sizeof(buf), stdin)) && (t)){

length = strlen(buf);

t = false;

if (buf[length - 1] == '\n') {

buf[--length] = '\0';

errno = 0;

value = strtod(buf, &end);

if (length == 0) {

fprintf(stderr, "\nОшибка: введена пустая строка.\n");

t = true;

}

if (errno != 0 || \*end != '\0') {

fprintf(stderr, "\nОшибка: введена некорректная строка\n");

fprintf(stderr, "\t%s\n" , buf);

fprintf(stderr, "\t%\*c\n", (int)(end - buf) + 1, '^');

t = true;

}

}else {

scanf("%\*[^\n]");

scanf("%\*c");

fprintf(stderr, "Ошибка: запись числа должна содержать не более %d символов\n", (SIZE\_STR\_DOUBLE - 2));

t = true;

}

if((!t) && ((value <= (double)(-100000)) || (value >= (double)(100000)))){

  printf("\nОшибка: вводимое число должно быть в диапазоне (-100000; 100000).\n");

  t = true;

  }

if(t){

  printf("Введите корректное значение: ");

  }

  }

return value;

}

double\*\* createMatrix(int size, bool user, int min, int max) {

  setlocale(LC\_ALL, "rus");

  if(!user)

    srand(time(NULL));

  double\*\* matrix = (double\*\*)malloc(size \* sizeof(double\*));

  int i = 0;

  while(i < size) {

    matrix[i] = (double\*)malloc(size \* sizeof(double));

    int j = 0;

    while(j < size) {

      if (user) {

        printf("Введите элемент матрицы [%d ; %d]: ", i, j);

        matrix[i][j] = getDoubleValue();

      }

      else

        matrix[i][j] = (double)(min + rand() % (max - min));

      j++;

    }

    i++;

  }

  return matrix;

}

void showMatrixInConsole(double\*\* matrix, int size, int eps) {

  if((matrix == NULL) || (size <= 0))

    return;

  int i = 0;

  while(i < size) {

    int j = 0;

    while(j < size){

      printf("%.\*lf ", eps, matrix[i][j]);

      j++;

    }

    printf("\n");

    i++;

  }

  printf("\n");

  return;

}

void getMatrixMinor(double \*\*mas, double \*\*p, int i, int j, int m) {

if(mas == NULL)

  return;

int ki = 0, kj = 0, di = 0, dj = 0;

di = 0;

while(ki < (m-1)) {

if (ki == i)

    di = 1;

dj = 0;

kj = 0;

while(kj < (m-1)) {

if (kj == j)

     dj = 1;

p[ki][kj] = mas[ki + di][kj + dj];

kj++;

}

ki++;

}

}

double Determinant(double \*\*mas, int m) {

if(mas == NULL)

  return 0;

int i, j, k, n;

double d;

double \*\*p;

p = (double\*\*)malloc(m \* sizeof(double\*));

i = 0;

while(i < m){

p[i] = (double\*)malloc(m \* sizeof(double));

i++;

}

j = 0;

d = 0;

k = 1;

n = (m - 1);

if (m == 1) {

d = mas[0][0];

int del\_item = 0;

  while(del\_item < m){

  free(p[del\_item]);

  del\_item++;

  }

  free(p);

return d;

}

if (m == 2) {

d = mas[0][0] \* mas[1][1] - (mas[1][0] \* mas[0][1]);

int del\_item = 0;

  while(del\_item < m){

  free(p[del\_item]);

  del\_item++;

  }

  free(p);

return(d);

}

if (m>2) {

  i = 0;

while(i < m){

getMatrixMinor(mas, p, i, 0, m);

d = d + k \* mas[i][0] \* Determinant(p, n);

k = -k;

i++;

}

}

int del\_item = 0;

while(del\_item < m){

free(p[del\_item]);

del\_item++;

}

free(p);

return(d);

}

double\*\* transportantMatrix(double\*\* mas, int s){

  if(mas == NULL)

    return NULL;

  double\*\* tr = (double\*\*)malloc(s \* sizeof(double\*));

  int i = 0;

  while(i < s){

    tr[i] = (double\*)malloc(s \* sizeof(double));

    int j = 0;

    while(j < s){

      tr[i][j] = mas[j][i];

      j++;

    }

    i++;

  }

  return tr;

}

double\*\* unionMatrix(double\*\* mas, int s){

  if(mas == NULL)

    return NULL;

  double\*\* minor = (double\*\*)malloc(s \* sizeof(double\*)),

    \*\*unMatrix = (double\*\*)malloc(s \* sizeof(double\*));

  int i = 0;

  while(i < s){

    minor[i] = (double\*)malloc(s \* sizeof(double));

    unMatrix[i] = (double\*)malloc(s \* sizeof(double));

    i++;

  }

  i = 0;

  while(i < s){

    int j = 0;

    while(j < s){

      getMatrixMinor(mas, minor, i, j, s);

      unMatrix[i][j] = pow(-1, (i+j)) \* Determinant(minor, (s-1));

      j++;

    }

    i++;

  }

  i = 0;

  while(i < s){

    free(minor[i]);

    i++;

  }

  free(minor);

  return unMatrix;

}

void matrixValueMultiplication(double\*\* matrix, double value, int size){

  if((matrix == NULL) || (size <= 0))

    return;

  int i = 0;

  while(i < size){

    int j = 0;

    while(j < size){

      matrix[i][j] \*= value;

      j++;

    }

    i++;

  }

  return;

}

double\*\* matrixMultiplication(double\*\* matrix1, double\*\* matrix2, int size){

  if((matrix1 == NULL) || (matrix2 == NULL) || (size <= 0))

    return NULL;

  double\*\* mult = (double\*\*)malloc(size \* sizeof(double\*));

  int i = 0;

  while(i < size){

    mult[i] = (double\*)malloc(size \* sizeof(double));

    int j = 0;

    while(j < size){

      mult[i][j] = 0;

      int p = 0;

      while(p < size){

        mult[i][j] += matrix1[i][p] \* matrix2[p][j];

        p++;

      }

      j++;

    }

    i++;

  }

  return mult;

}

int main() {

  setlocale(LC\_ALL, "rus");

  int size = 0;

  printf("Введите размер квадратной матрицы: ");

  while(((size = getIntValue(false)) < 2) || (size > 10)){

    printf("\nОшибка: размер матрицы не может быть меньше 2 или больше 10\n");

    printf("Введите корректное значение: ");

  }

  printf("Как реализовать заполнение матрицы? (0 - рандомное заполнение / 1 - с помощью пользователя): ");

  int check = 0;

  {

    check = getIntValue(true);

    while ((check > 1) || (check < 0)) {

printf("Ошибка: некорректный ответ на вопрос\n");

      printf("Введите корректный ответ на вопрос (0 или 1): ");

      check = getIntValue(true);

    }

  }

  int eps = 0;

  if(check){

    printf("Программа работает с действительными числами. \nВведите количество знаков после запятой\nдля корректного вывода чисел: ");

    eps = getIntValue(true);

    while((eps < 0) || (eps > 5)){

      printf("Ошибка: количество знаков должно быть значением в диапазоне [0; 5]\n");

      printf("Введите корректное положительное целое значение в диапазоне [0; 5]: ");

      eps = getIntValue(true);

    }

  }

  int d\_min = 0, d\_max = 0;

  if (!check) {

    while ((d\_min == d\_max) || (d\_max < d\_min)) {

      printf("Введите минимальное и максимальное значение диапазона рандомных чисел\n");

      printf("Минимальное значение: ");

      d\_min = getIntValue(true);

      printf("Максимальное значение: ");

      d\_max = getIntValue(true);

      if ((d\_min == d\_max) || (d\_max < d\_min))

        printf("\nОшибка: минимальное значение диапазона рандомных чисел должно быть меньше максимального значения!\n");

    }

  }

  double\*\* matrix = createMatrix(size, check, d\_min, d\_max);

  double det = Determinant(matrix, size);

  printf("Исходная матрица: \n");

  showMatrixInConsole(matrix, size, eps);

  printf("\nОпределитель исходной матрицы: %.\*lf\n\n", eps, det);

  if(!det){

    printf("Определитель исходной матрицы равен 0. Нахождение обратной матрицы невозможно");

    int del\_item = 0;

    while(del\_item < size){

      free(matrix[del\_item]);

      del\_item++;

    }

    free(matrix);

    return 1;

  }

  double\*\* trMatrix = transportantMatrix(matrix, size);

  printf("Транспонированная исходная матрица: \n");

  showMatrixInConsole(trMatrix, size, eps);

  double\*\* unMatrix = unionMatrix(trMatrix, size);

  printf("Матрица составленная из алгебраических дополнений транспонированной исходной матрицы: \n");

  showMatrixInConsole(unMatrix, size, eps);

  printf("\n\nОбратная матрица найденная методом алгебраических дополнений: \n");

  matrixValueMultiplication(unMatrix, (1/det), size);

  showMatrixInConsole(unMatrix, size, (eps == 0)? 5 : eps);

  double\*\* multiMatrix = matrixMultiplication(matrix, unMatrix, size);

  printf("\n\nПроверка произведения исходной матрицы на обратную(в произведении дают единичную матрицу): \n");

  int i = 0;

  while(i < size){

    int j = 0;

    while(j < size){

      multiMatrix[i][j] = (multiMatrix[i][j] == -0)? (-1)\*multiMatrix[i][j] : multiMatrix[i][j];

      j++;

    }

    i++;

  }

  showMatrixInConsole(multiMatrix, size, 0);

  int del\_item = 0;

  while(del\_item < size){

    free(matrix[del\_item]);

    free(trMatrix[del\_item]);

    free(unMatrix[del\_item]);

    free(multiMatrix[del\_item]);

    del\_item++;

  }

  free(matrix);

  free(trMatrix);

  free(unMatrix);

  free(multiMatrix);

  system("PAUSE");

  return 0;

}

6 Анализ вычислительной сложности алгоритма

Выполним асимптотическую оценку функции сложности алгоритма.

Опишем алгоритм в виде псевдокода:

Алгоритм включает в себя несколько функций.

Псевдкод функции getMatrixMinor():

1.ki = 0

2.kj = 0

3.while(ki < (m-1)){

4. if(ki == i)

5. di = 1

6. dj = 0

7. kj = 0

8. while(kj < (m-1)){

9. if(kj == j)

10. dj = 1

11. p[ki][kj] = mas[ki + di][kj + dj];

12. kj++;

13. }

14. ki++

15.}

Вычислим оценку сложности функции getMatrixMinor():

Определим операцию, которая выполняется большее количество раз: строки с 9 по 12

Сколько раз данная операция выполняется? Количество операций зависит от числа m. Первый цикл выполняется m раз (пока условие ki < (m-1) выполняется). Второй цикл выполняется так же m раз (пока kj < (m-1)). В таком случае, количество выполненных операций будет равно m\*m = m^2. Оценка сложности этой функции(меньшего алгоритма) равна O(n^2);

Псевдокод функции Determinant():

1. i = 0

2. j = 0

3. d = 0

4. k = 1

5. n = (m – 1)

6.if(m == 1){

7. d = mas[0][0]

8. return d;

9.}

10.if(m == 2){

11. d = mas[0][0] \* mas[1][1] - (mas[1][0] \* mas[0][1]);

12. return d;

13.}

14.if(m > 2){

15. while(i < m){

16. getMatrixMinor(mas, p, i, 0, m);

17. d = d + k \* mas[i][0]\*Determinant(p, n);

18. k = -k

19. i++

20.}

21.return d

Вычислим оценку сложности функции Determinant():

Определим операцию, которая выполняется большее количество раз:

Строчки кода с 16 по 19;

Сколько раз данные операции выполняется? Количество операций зависит от числа m.

Цикл выполняется ровно m+1 раз (пока i < m). Оценка сложности этой функции(подалгоритма) составляет O(n)

Псевдокод функции transportantMatrix():

1. i = 0

2. while(i < s){

3. j = 0

4. while(j < s){

5. tr[i][j] = mas[j][i];

6. j++;

7. }

8. i++;

9.}

10. return tr;

Вычислим оценку сложности функции transportantMatrix():

Определим операцию, которая выполняется большее кол-во раз: строки с 5 до 6 выполняются ровно (s+1)\*(s+1) раз. T(n) = (n+1)^2 = n^2 + 2n + 1; Оценка сложности этой функции(подалгоритма) составляет O(n^2)

Псевдокод функции unionMatrix():

1. i=0

2.while(i < s){

3. j = 0

4. while(j < s){

5. getMatrixMinor(mas, minor, i, j, s);

6. unMatrix[i][j] = pow(-1, (i+j))\*Determinant(minor, (s-1));

7. j++;

8. }

9. i++

10.}

11.return unMatrix;

Вычислим оценку сложности функции unionMatrix():

Определим самую большую по количеству выполнений операцию: это строки с 5 по 7 они выполняются ровно (s+1)(s+1) раз. T(n) = n^2 + 2n + 1; Оценка сложности функции(подалгоритма) равняется O(n^2);

Псевдокод функции matrixValueMultiplicaton():

1. i=0

2.while(i<size){

3. j = 0

4. while(j < size){

5. matrix[i][j] = matrix[i][j] \* value;

6. j++;

7. }

8.i++

9.}

Вычислим оценку сложности функции matrixValueMultiplication():

Определим самую большую по кол-ву выполнений операцию: это строки с 5 по 6 они выполняются ровно (s+1)\*(s+1) раз. T(n) = n^2 + 2n + 1; Оценка сложности функции(подалгоритма) равняется O(n^2);

7 Результаты тестирования

В таблице 4 приведены результаты тестов из таблицы 3.

Таблица 4

|  |  |  |  |
| --- | --- | --- | --- |
| № | Результат | Входные данные | Выходные данные |
| 1 | Тест пройден | size = 2, check = 1,  eps = 5,  [матрица заполненная пользователем]:  17 10  10 10 | 0, 14286 -0,14286  -0,14286 0,24286 |
| 2 | Тест пройден | size = 4, check = 0,  d\_min = 10,  d\_max = 100,  [матрица заполненная рандомно]:  13 94 41 34  61 13 95 79  35 61 18 64  89 10 57 66 | (для удобства все элементы округлены до 2-го знака)   * 1. -0.02 -0.01 0.02   2. -0.01 -0.00 0.00   0.01 0.01 -0.02 -0.00 |
| 3 | Тест пройден | size = 2  сheck = 0  d\_min = 1  d\_max = 1  [матрица заполненная рандомно]:  1 1  1 1 | Определитель исходной матрицы равен 0. Нахождение обратной матрицы невозможно |

Продолжение таблицы 4

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | Тест пройден | size = 3  сheck = 1  eps = 2  [матрица заполненная пользователем]:  1 2 3  4 5 6  7 8 9 | Определитель исходной матрицы равен 0. Нахождение обратной матрицы невозможно |
| 5 | Тест пройден | size = 90  сheck = 1  eps = 5  [матрица заполненная пользователем]:  ….. | Ошибка: размера матрицы не может быть меньше 2 или больше 10 |
| 6 | Тест пройден | size = 3  check = 50  eps = 3  [матрица заполненная ?]:  …. | Ошибка: не корректный ответ на вопрос |
| 7 | Тест пройден | size = 3  сheck = 1  eps = -10  [матрица заполненная пользователем]:  ….. | Ошибка: количество знаков должно быть значением в диапазоне [0; 5] |

Продолжение таблицы 4

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | Тест пройден | size = 3  сheck = 1  eps = 2  [матрица заполненная пользователем]:  0 245 24151251  1 2415 215  2 3412 5151 | Ошибка: вводимое число должно быть в диапазоне (-100000 ; 100000) |
| 9 | Тест пройден | size = 10  сheck = 0  eps = 0  d\_min = 500  d\_max = 500000  [матрица заполненная рандомно]:  …… | Ошибка: вводимое число должно быть в диапазоне (-100000 ; 100000) |
| 10 | Тест пройден | size = 5  сheck = 0  eps = 0  d\_min = 5  d\_max = 5  [матрица заполненная рандомно]:  …. | Ошибка: минимальное значение диапазона рандомных чисел должно быть меньше максимального значения! |
| 11 | Тест пройден | size = 2  check = 1  eps = 4  [матрица заполненная пользователем]:  1 hello  faw 39 | Ошибка: введена не корректная строка |
| 12 | Тест пройден | size = “abc”  check = 1  eps = 4  [матрица заполненная пользователем]:  …. | Ошибка: введена не корректная строка |
| 13 | Тест пройден | size = 2  check = 0  eps = 0  d\_min = 50  d\_max = 200fg  [матрица заполненная рандомно]:  …. | Ошибка: введена не корректная строка |

Результаты выполнения тестов с 1 по 13 приведены на рисунках с 13 по 25 соответственно.

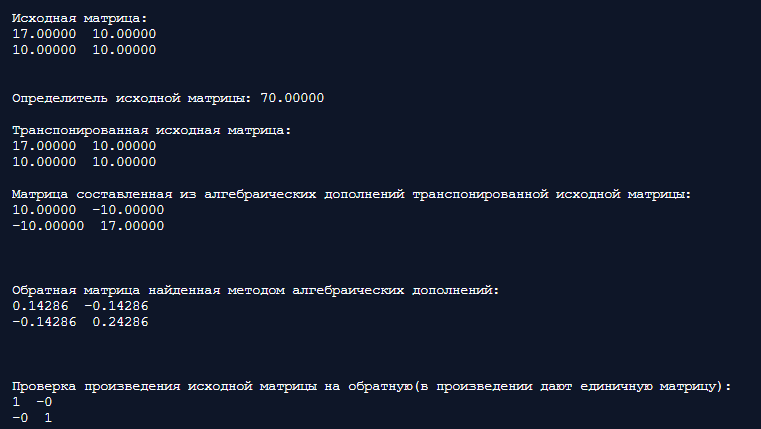


Рисунок 13 – результат выполнения теста 1

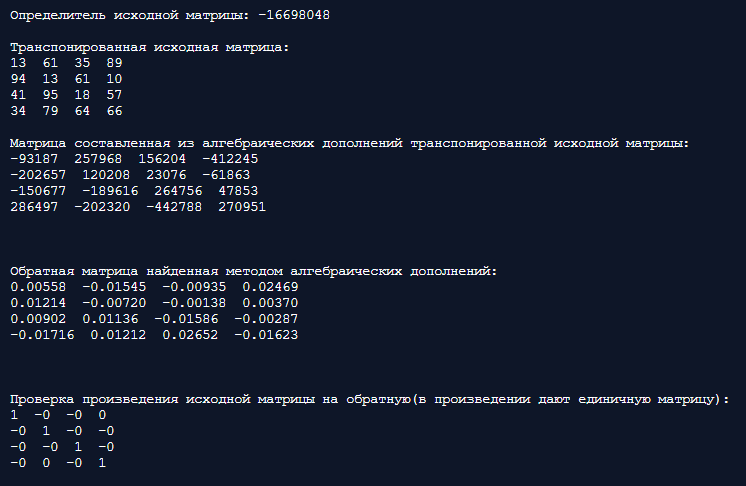


Рисунок 14 – Результат выполнения теста 2

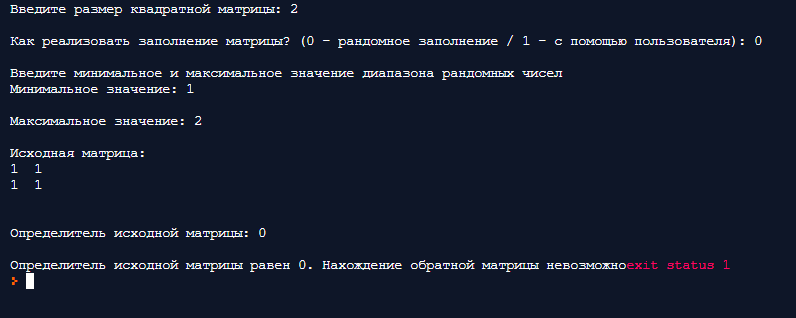


Рисунок 15 – Результат выполнения теста 3

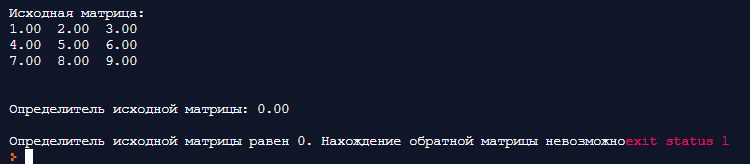


Рисунок 16 – Результат выполнения теста 4

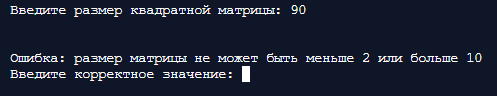


Рисунок 17 – Результат выполнения теста 5

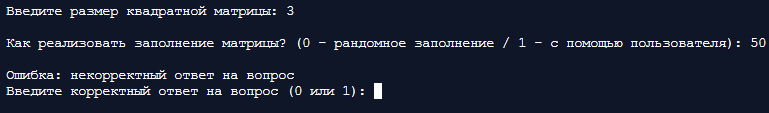


Рисунок 18 – Результат выполнения теста 6

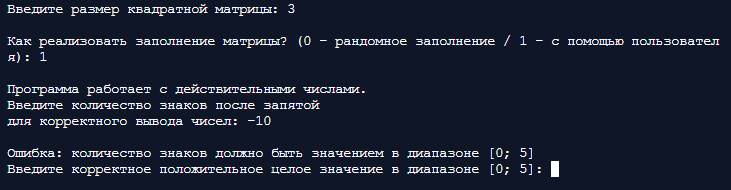


Рисунок 19 – Результат выполнения теста 7

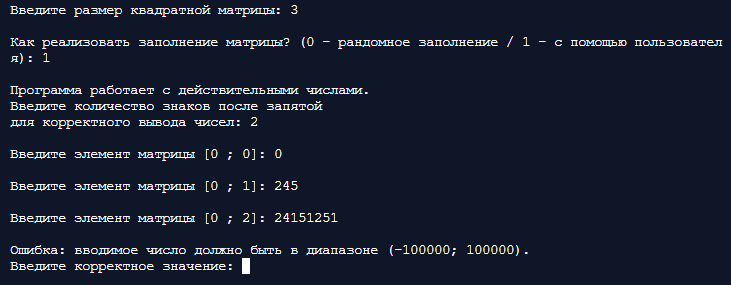


Рисунок 20 – Результат выполнения теста 8

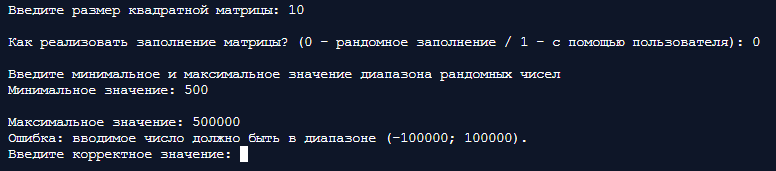


Рисунок 21 – Результат выполнения теста 9

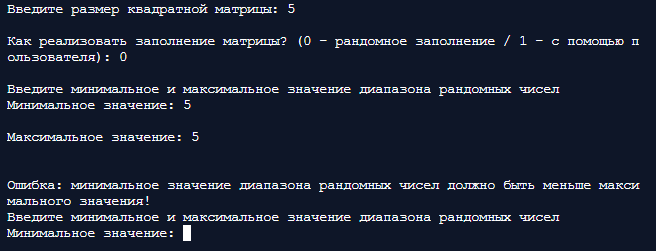


Рисунок 22 – Результат выполнения теста 10

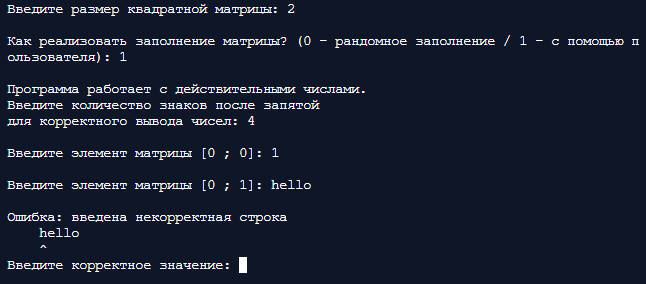


Рисунок 23 – Результат выполнения теста 11

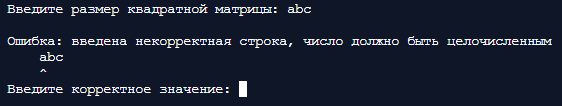


Рисунок 24 – Результат выполнения теста 12

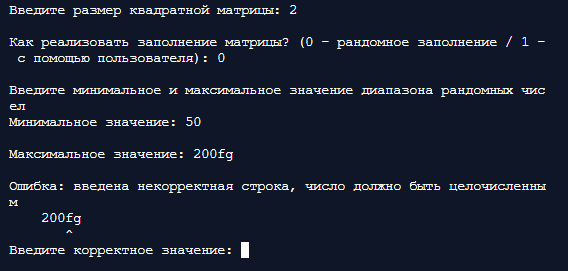


Рисунок 25 – Результат выполнения теста 13

8 Заключение

В процессе практики был разработан алгоритм решения индивидуального задания: «Вычислить обратную матрицу методом алгебраических дополнений. Размер матрицы задается пользователем. Реализовать два варианта ввода значений первоначальной матрицы: ввод значений пользователем вручную, случайная генерация элементов в заданном диапазоне.». С использованием методов классов эквивалентности и предположения об ошибке сформировано 13 тестов покрывающих все пути. Составленная на языке Си программа, реализующая разработанный алгоритм, прошла все 13 тестов успешно.

В ходе учебной практики были получены навыки составления блок-схемы алгоритма решения задачи, разработки тестов с использованием различных методов, а также применения языка программирования Си для реализации алгоритма решения задачи.

Все цели практики достигнуты.

9 Список использованных источников

1. Павловская Т.А. С и С++. Программирование на языке высокого уровня. СПб.: Питер, 2003. – 461с.

2. Программирование на языке Си [Электронный ресурс] URL: http://www.codenet.ru/progr/cpp/1/