**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Основы информатики»

Отчет по лабораторной работе №8

«Обработка и печать числовой матрицы»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-11 |  | преподаватель каф. ИУ5 |
| Михалёв Ярослав |  | Козлов А.Д. |
| Подпись и дата: |  | Подпись и дата: |
|  |  |  |

Москва, 2021 г.

Постановка задачи

**Цели работы:**

- создание двухмерных динамических массивов;

- обработка матриц;

- демонстрация технологии структурного программирования на примере программы сортировки строк матрицы;

- использование файлов для хранения исходных данных;

- форматированный вывод матриц на экран.

**Задание.**

Создать квадратную матрицу A размером N\*N (где N вводится с клавиатуры), имеющую элементы **A i,j = 1 / (j!)i .** В зависимости от размера матрицы и ширины поля вывода элемента матрицы, обеспечить удобное для пользователя отображение матрицы на экране. Оформить вывод матрицы размером N\*M на экран в виде функции с целью использования ее в последующих лабораторных работах для распечатки двумерных массивов.

Матрица должна передаваться в разрабатываемые функции через параметры.

Не изменяя кода функции вывода матрицы, распечатать матрицу в «научном» формате и в формате с фиксированной точкой с точностью 8 знаков после запятой.

Разработка алгоритма

**Описание входных, выходных и вспомогательных данных:**

Входные данные:

int N – размер матрицы A

Выходные данные:

double\*\* A – квадратная матрица N\*N

Вспомогательные данные:

int width – ширина консоли

**Список функций:**

double\*\* CreateMatrix(int N)

Создаёт единичную матрицу размером N - 1

Входные данные:

* int N – размер созданной матрицы

Выходные данные:

* double\*\* A – единичная матрица

void FillMatrix(double\*\* A, int N)

Заполняет матрицу по переданной ссылке

Входные данные:

* double\*\* A – ссылка на матрицу
* int N – размер матрицы A

void Clear(double\*\* A, int N)

Очищает из памяти двумерный динамический массив A

Входные данные:

* double\*\* A – динамический массив
* int N – размер динамического массива А

int GetConsoleWidth()

Описание: возвращает ширину консоли

Выходные данные:

* int width – ширина консоли

void PrintDinamicMatrix(double\*\* A, int N)

Выводит динамичный двумерный массив на экран по переданной ссылке

Входные данные:

* double\*\* A – ссылка на массив
* int N – размер массива

void PrintStaticMatrix(double B[10][10])

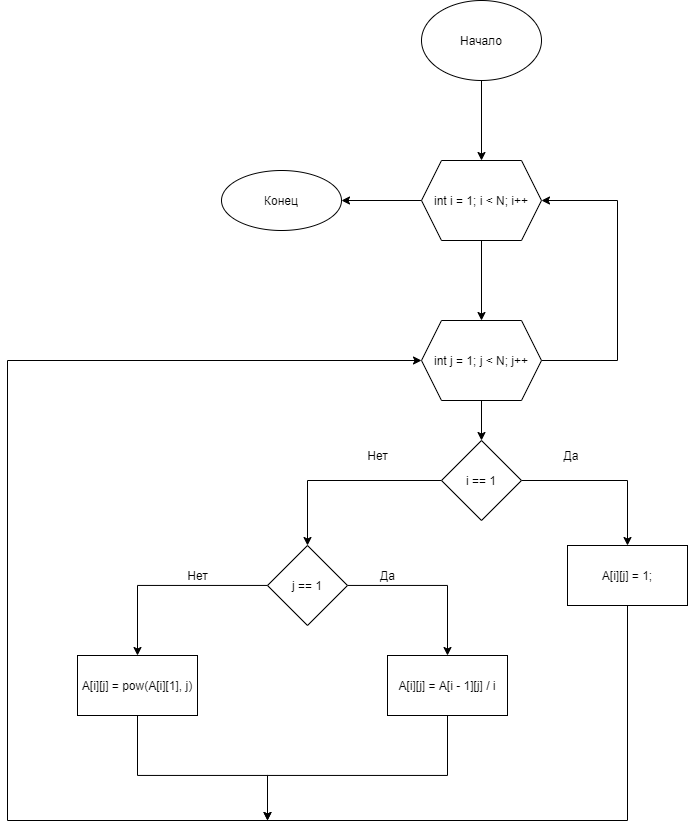
Выводит статический двумерный массив на экран

Входные данные:

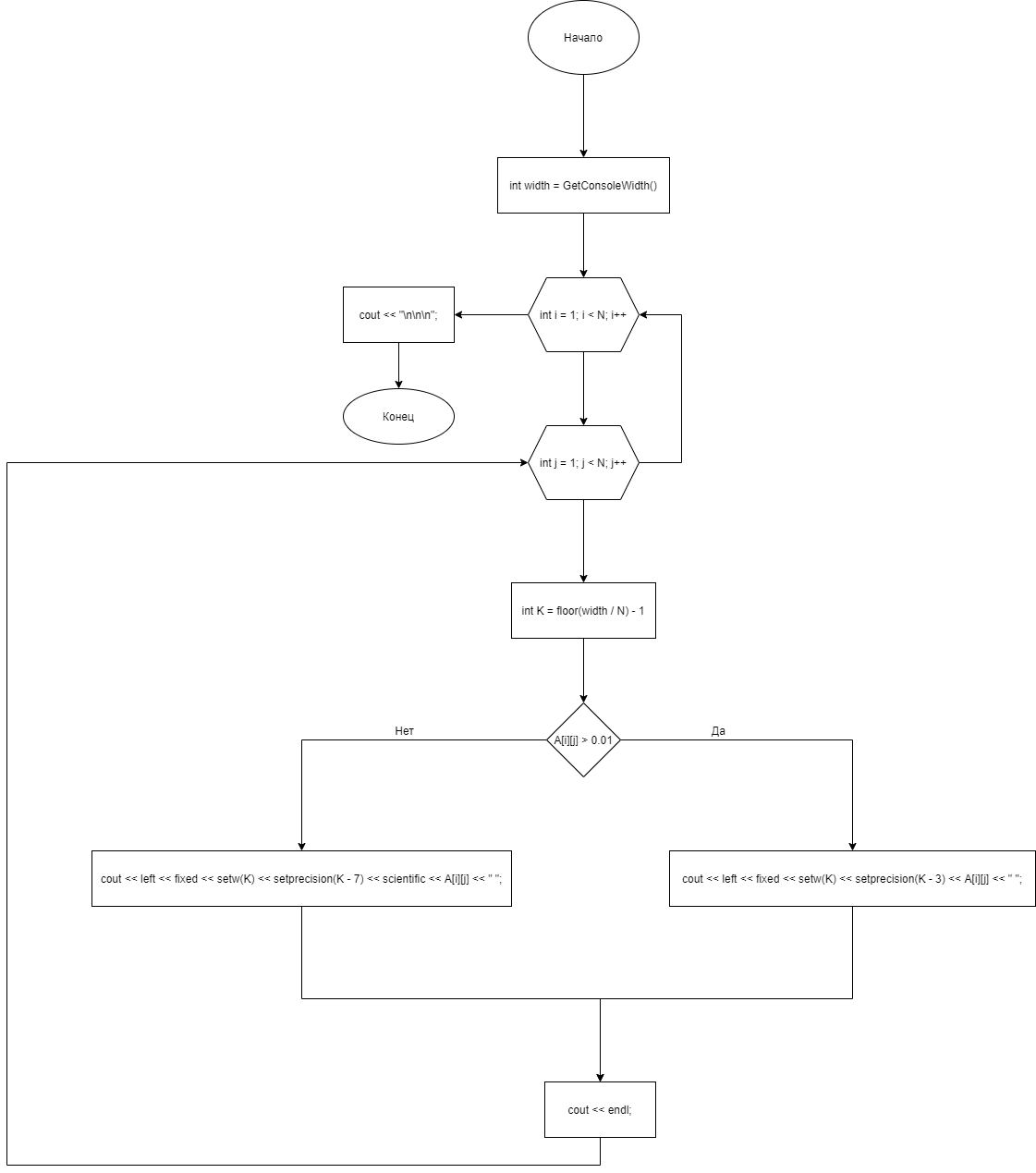
* double B[10][10] - статический двумерный массив 10 на 10

Схема алгоритма

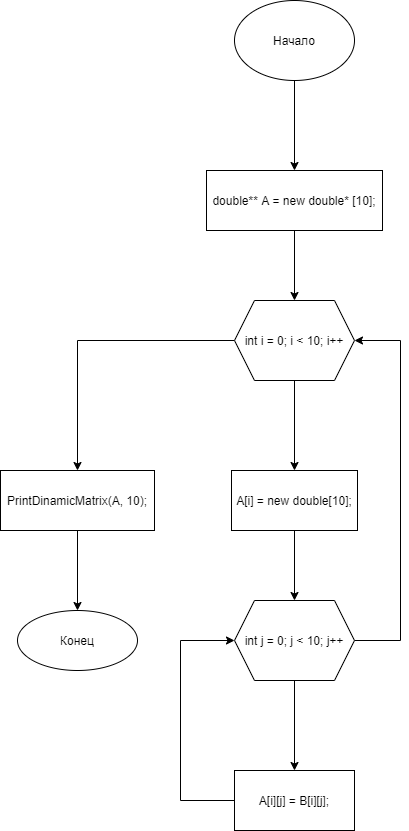
FillMatrix



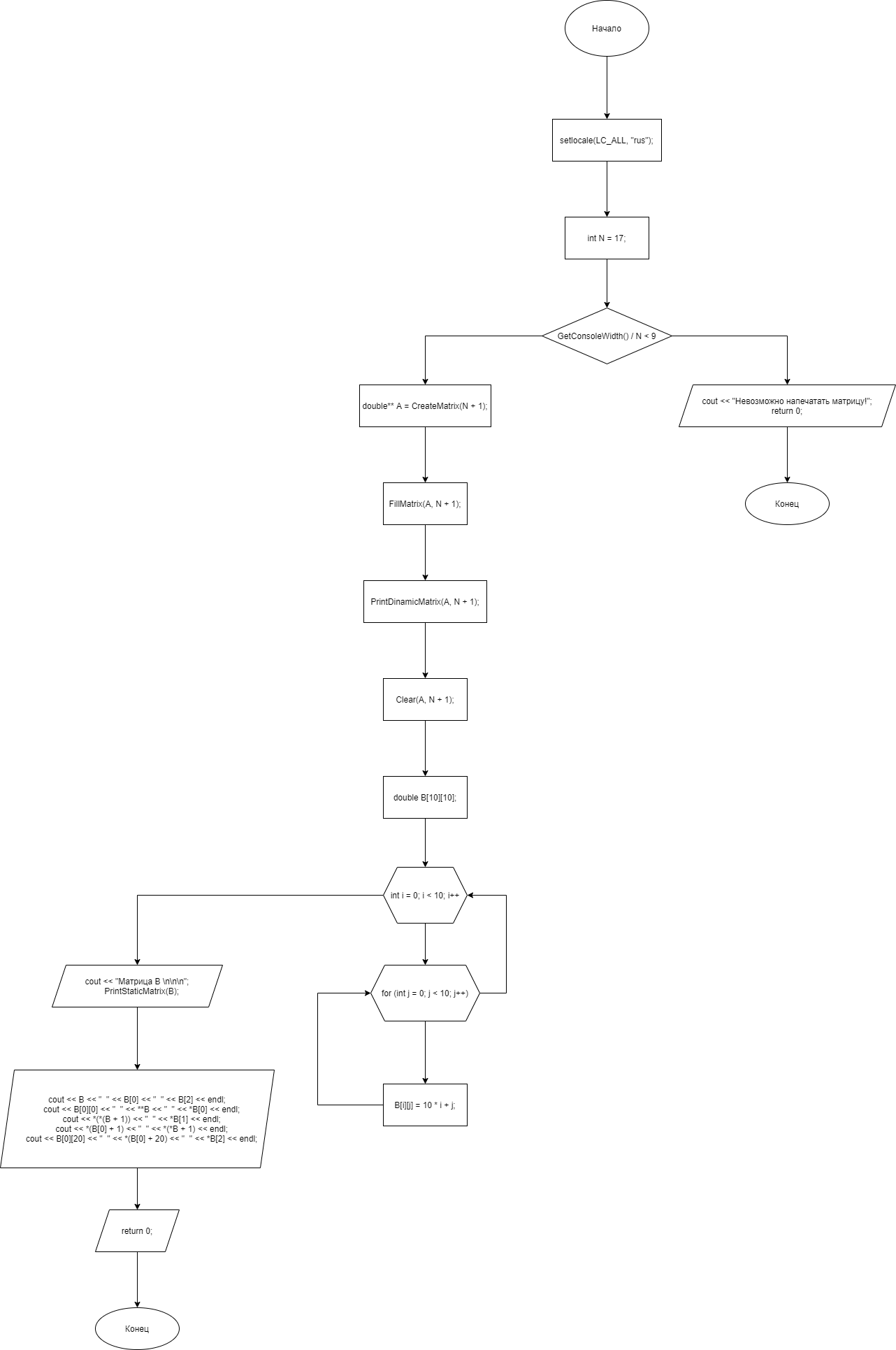
PrintDinamicMatrix



PrintStaticMatrix



Main



Текст программы

#include <iostream>

#include <iomanip>

#include <Windows.h>

**using** **namespace** std**;**

// Очищаем память

void Clear**(**double**\*\*** A**,** int N**)**

**{**

**for** **(**int i **=** 1**;** i **<** N**;** i**++)**

**delete[]** A**[**i**];**

**delete[]** A**;**

**}**

// Создаём единичную матрицу

double**\*\*** CreateMatrix**(**int N**)** **{**

double**\*\*** A **=** **new** double**\*** **[**N**];**

**for** **(**int i **=** 1**;** i **<** N**;** i**++)**

**{**

A**[**i**]** **=** **new** double**[**N**];**

**for** **(**int j **=** 1**;** j **<** N**;** j**++)**

A**[**i**][**j**]** **=** 1**;**

**}**

**return** A**;**

**}**

// Заполяем матрицу

void FillMatrix**(**double**\*\*** A**,** int N**)**

**{**

**for** **(**int i **=** 1**;** i **<** N**;** i**++)**

**{**

**for** **(**int j **=** 1**;** j **<** N**;** j**++)** **{**

**if** **(**i **==** 1**)**

**{**

A**[**i**][**j**]** **=** 1**;**

**}**

**else**

**{**

**if** **(**j **==** 1**)**

A**[**i**][**j**]** **=** A**[**i **-** 1**][**j**]** **/** i**;**

**else**

A**[**i**][**j**]** **=** pow**(**A**[**i**][**1**],** j**);**

**}**

**}**

**}**

**}**

// Вовращает ширину консоли

int GetConsoleWidth**()**

**{**

COORD**\*** Chars**;**

HANDLE hCon **=** GetStdHandle**(**STD\_OUTPUT\_HANDLE**);**

CONSOLE\_SCREEN\_BUFFER\_INFO csbInfo**;**

GetConsoleScreenBufferInfo**(**hCon**,** **&**csbInfo**);**

int width **=** csbInfo**.**srWindow**.**Right **-** csbInfo**.**srWindow**.**Left **+** 1**;**

**return** width**;**

**}**

// Выводим матрицу на экран

void PrintDinamicMatrix**(**double**\*\*** A**,** int N**)**

**{**

int width **=** GetConsoleWidth**();**

**for** **(**int i **=** 1**;** i **<** N**;** i**++)**

**{**

**for** **(**int j **=** 1**;** j **<** N**;** j**++)** **{**

int K **=** floor**(**width **/** N**)** **-** 1**;**

**if** **(**A**[**i**][**j**]** **>** 0.01**)**

cout **<<** left **<<** fixed **<<** setw**(**K**)** **<<** setprecision**(**K **-** 3**)** **<<** A**[**i**][**j**]** **<<** " "**;**

**else**

cout **<<** left **<<** fixed **<<** setw**(**K**)** **<<** setprecision**(**K **-** 7**)** **<<** scientific **<<** A**[**i**][**j**]** **<<** " "**;**

**}**

cout **<<** endl**;**

**}**

cout **<<** "\n\n\n"**;**

**}**

// Выводит статичную матрицу на экран

void PrintStaticMatrix**(**double B**[**10**][**10**])** **{**

double**\*\*** A **=** **new** double**\*** **[**10**];**

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

A**[**i**]** **=** **new** double**[**10**];**

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

A**[**i**][**j**]** **=** B**[**i**][**j**];**

**}**

**}**

PrintDinamicMatrix**(**A**,** 10**);**

Clear**(**A**,** 10**);**

**}**

int main**()**

**{**

setlocale**(**LC\_ALL**,** "rus"**);**

int N **=** 10**;**

**if** **(**GetConsoleWidth**()** **/** N **<** 9**)**

**{**

cout **<<** "Невозможно напечатать матрицу!"**;**

**return** 0**;**

**}**

double**\*\*** A **=** CreateMatrix**(**N **+** 1**);**

FillMatrix**(**A**,** N **+** 1**);**

PrintDinamicMatrix**(**A**,** N **+** 1**);**

Clear**(**A**,** N **+** 1**);**

double B**[**10**][**10**];**

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

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

B**[**i**][**j**]** **=** 10 **\*** i **+** j**;**

PrintStaticMatrix**(**B**);**

cout **<<** "\n\n\n"**;**

cout **<<** B **<<** " " **<<** B**[**0**]** **<<** " " **<<** B**[**2**]** **<<** endl**;**

cout **<<** B**[**0**][**0**]** **<<** " " **<<** **\*\***B **<<** " " **<<** **\***B**[**0**]** **<<** endl**;**

cout **<<** **\*(\*(**B **+** 1**))** **<<** " " **<<** **\***B**[**1**]** **<<** endl**;**

cout **<<** **\*(**B**[**0**]** **+** 1**)** **<<** " " **<<** **\*(\***B **+** 1**)** **<<** endl**;**

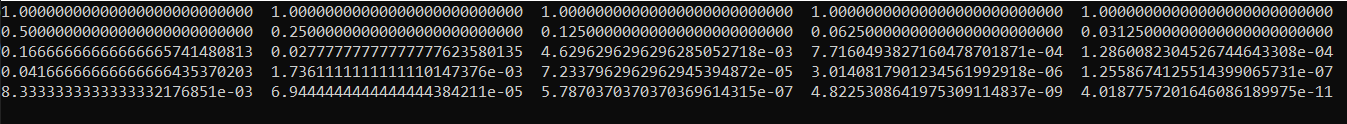
cout **<<** B**[**0**][**20**]** **<<** " " **<<** **\*(**B**[**0**]** **+** 20**)** **<<** " " **<<** **\***B**[**2**]** **<<** endl**;**

**return** 0**;**

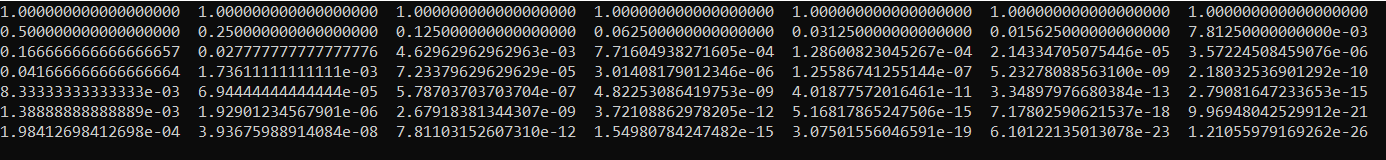
**}**

Анализ результатов

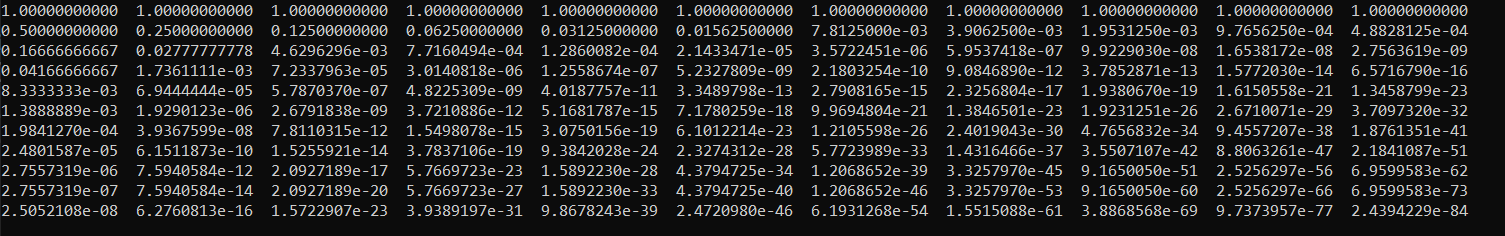
Для N = 5



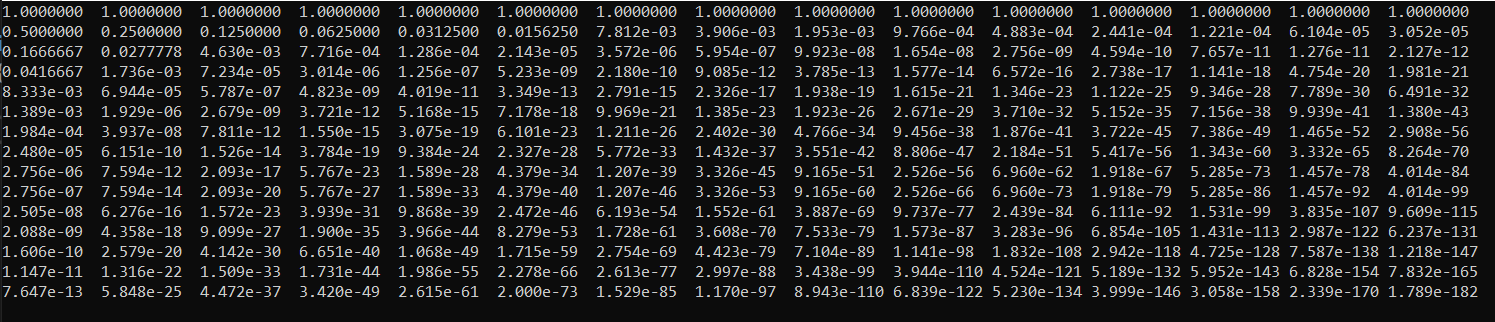
Для N = 7



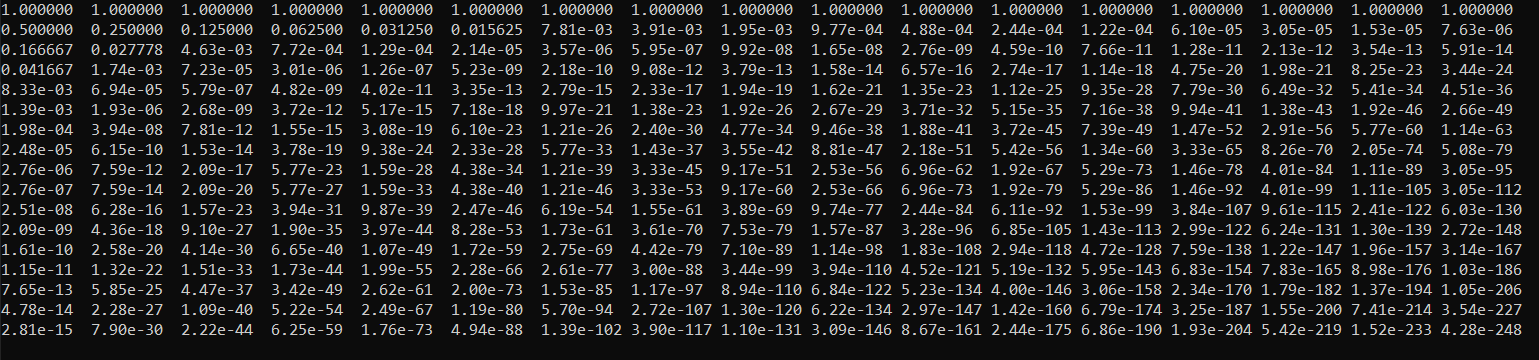
Для N = 11



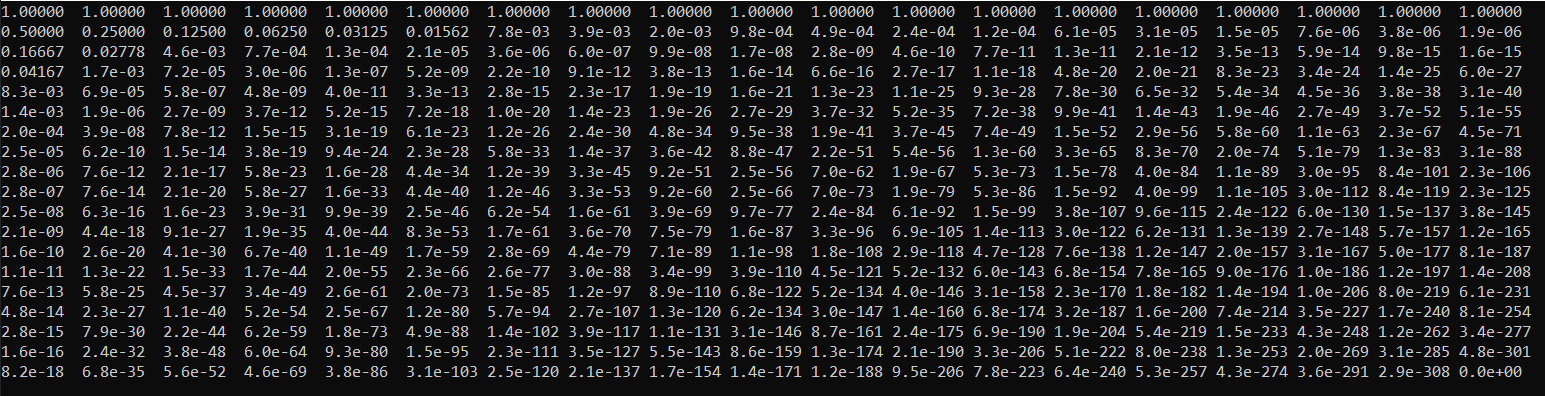
Для N = 15



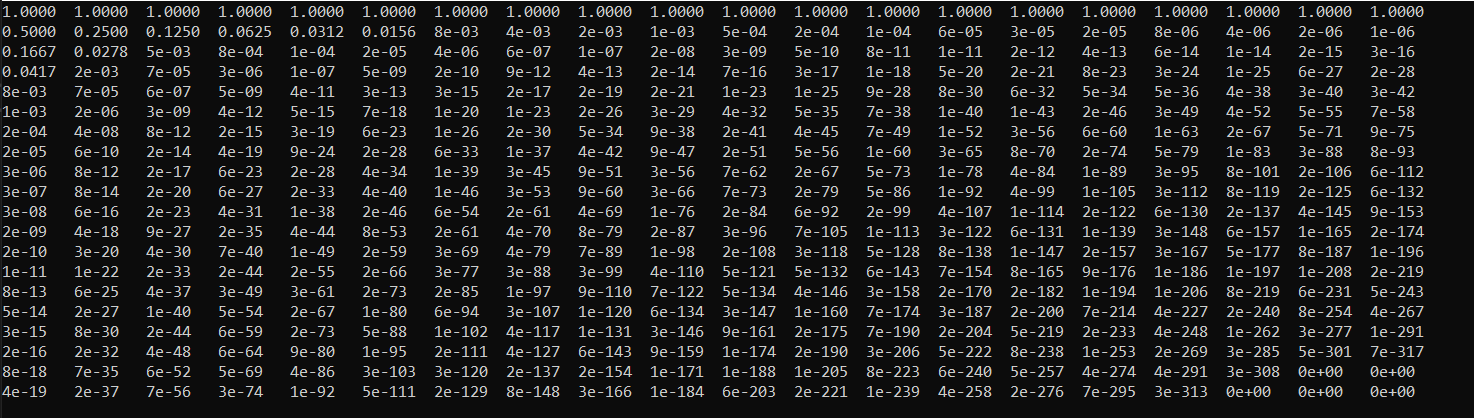
Для N = 17



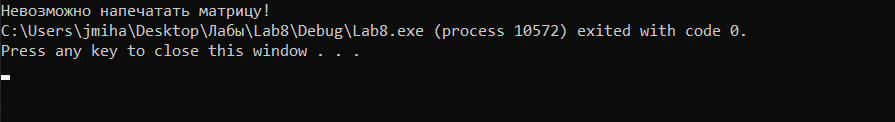
Для N = 19



Для N = 20



Для N > 20



Вывод

Я научился

* Создавать двумерные динамические массивы
* Обрабатывать матрицы
* Использовать текстовые файлы для хранения данных
* “Красиво” выводить матрицу на экран