ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ УФИМСКИЙ УНИВЕРСИТЕТ НАУКИ И ТЕХНОЛОГИЙ

Кафедра ВМиК

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

по предмету **«Языки программирования»**

Выполнил: студент группы МО-203б

Ярошко Е. В.

Проверил:  
ст. преподаватель

Рипатти А. В.

**Уфа 2025 г.**

**Цель работы:**

Изучить работу с одномерными и многомерными массивами.

**Задачи работы:**

- освоить понятие перемножения прямоугольных матриц

- освоить понятие стохастической матрицы (матрицы переходных вероятностей для некоторой цепи Маркова)

**Задание:**

1. Реализовать перемножение прямоугольных матриц размеров NxM и MxK.

Числа N, M и K задаются пользователем в диапазоне от 1 до 10.

После этого программа генерирует матрицы размера NxM и MxK, заполненные случайными числами от 1 до 9.

В конце работы программа должна вывести матрицу размера NxK – результат перемножения двух матриц.

2. Стохастическая матрица (справа) – квадратная матрица A размера NxN, состоящая из неотрицательных чисел, сумма чисел в каждом столбце равна 1. Значение на пересечении i-й строки и j-го столбца задает вероятность перехода из состояния j в состояние i за один шаг.

Пусть x = (1/N, 1/N, …, 1/N)T – вектор начальных вероятностей системы (изначально мы можем оказаться равновероятно в одном из N состояний). По сути этот вектор – матрица размера Nx1.

Тогда A\*x – это вектор вероятностей оказаться в каждом из состояний после выполнения одного шага моделирования.

Нужно определить вероятности оказаться в каждом из N состояний через 220 шагов.

Число N задается пользователем в диапазоне от 2 до 10. После этого программа генерирует случайную стохастическую матрицу A размера NxN и выводит ее на экран.

Затем, возводя матрицу A в квадрат (перемножая A на саму себя), программы вычисляет матрицы A2, A4, A8, … A^(220). На экран выводится результат после каждого перемножения. В конце итоговая матрица умножается на x = (1/N, 1/N, …, 1/N)T, после чего результат выводится на экран.

**Теоретические основы:**

Для генерации случайных значений матрицы впервые используется библиотека random вместо rand() из библиотеки standard. Использование отдельной библиотеки гарантирует более качественную генерацию и случайную последовательность. Чтобы использовать генерацию, необходимо создавать объект random\_device, на основе которого создаётся новый объект генератора mt19937, основанный на алгоритме «Вихрь Мерсенна». Число 19937 в названии означает период генератора (2 в 19937-1 степени). Для задания границ генерации используется uniform\_real\_distribution.

Впервые в рамках лабораторной работы использовалось ключевое слово constexpr, которое позволяет вычислить значение переменной и встроить её непосредственно в код. Данный подход возможен потому, что все переданные аргументы уже известны на этапе компиляции программы.

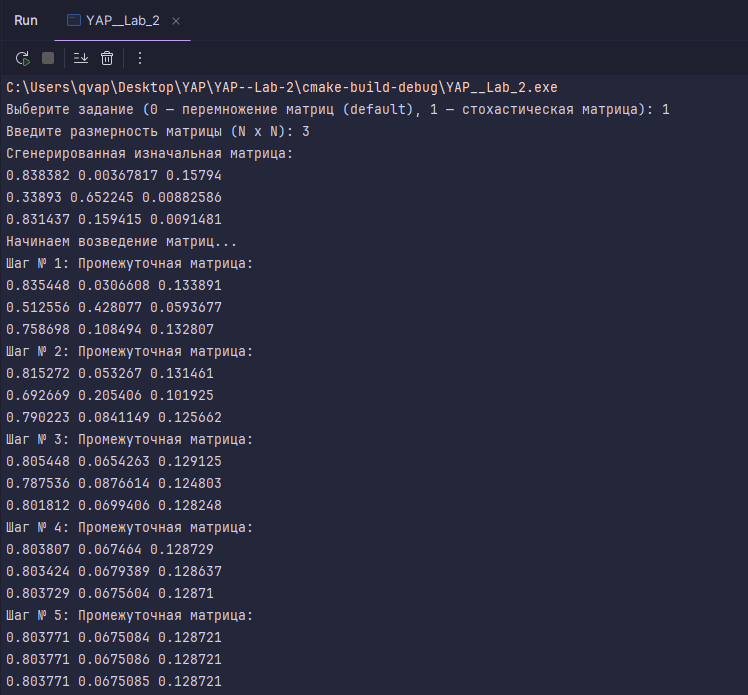
Для записи матрицы в виде списков данных используются вложенные списки (главный список — строки, вложенные списки — значения по столбцам). Для этого используются переменные vector<vector<int>> из библиотеки vector. Использование векторов позволяет создавать динамические списки с изменяемыми элементами.

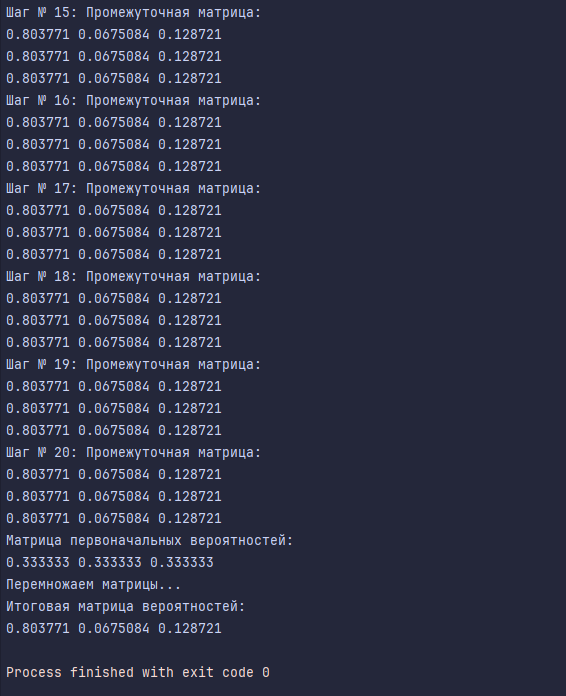
Во время выполнения лабораторной работы было замечено, что методические указания для работы имеют ошибки. Стохастическая матрица (справа) — матрица, где сумма чисел в каждой строке (не столбце!) равна 1. В соответствии с этим алгоритмы в работе скорректированы так, что вывод имеет предсказуемый результат (до внесения изменений значения стохастической матрицы при возведении в степень переполнялись). Также важен порядок умножения матриц: в соответствии с марковскими процессами, итоговым результатом должно быть умножение матрицы начальных вероятностей на стохастическую матрицу, а не наоборот.

**Листинг программы:**

*#include* <iostream>  
*#include* <vector>  
*#include* <locale>  
*#include* <windows.h>  
*#include* <random>  
  
using namespace std;  
  
*// Заполняет матрицу рандомными числами от 0 до 9*void *fill\_matrix*(*vector*<*vector*<double>> &*matrix*) {  
 constexpr int steps = (9.0 - 0.0) / 1.0;  
 *random\_device* rd;  
 *mt19937* gen*(*rd()*)*;  
 *uniform\_int\_distribution*<int> dist*(*0, steps*)*;  
 for (*vector*<double> &i : *matrix*) {  
 for (double &j : i) {  
 j = 0.0 + dist(gen) \* 1.0;  
 }  
 }  
}  
  
*// Заполняет стохастическую матрицу числами, суммирующимися в единицу*void *fill\_stochastic\_matrix*(*vector*<*vector*<double>> &*matrix*) {  
 *random\_device* rd;  
 *mt19937* gen*(*rd()*)*;  
  
 if (*matrix*.*size*() != *matrix[*0*]*.*size*()) {  
 cout *<<* "Матрица не является квадратной." *<< endl*;  
 return;  
 }  
  
 *// Для стохастической матрицы сумма элементов по СТРОКАМ должна быть равна 1* for (int i = 0; i < *matrix*.*size*(); ++i) { *// перебираем строки* double temp = 0.0;  
 for (int j = 0; j < *matrix[*i*]*.*size*() - 1; ++j) { *// перебираем столбцы, кроме последнего  
 uniform\_real\_distribution*<double> dist*(*0.0, 1.0 - temp*)*;  
 const double generated\_number = dist(gen);  
 *matrix[*i*][*j*]* = generated\_number;  
 temp += *matrix[*i*][*j*]*;  
 }  
 *// Последний элемент строки заполняем так, чтобы сумма стала ровно 1  
 matrix[*i*][matrix[*i*]*.*size*() - 1*]* = 1.0 - temp;  
 }  
}  
  
*// Перемножает две матрицы  
vector*<*vector*<double>> *multiply\_matrix*(const *vector*<*vector*<double>> &*matrix1*, const *vector*<*vector*<double>> &*matrix2*) {  
 *// Проверка на возможность перемножения матриц* if (*matrix1[*0*]*.*size*() != *matrix2*.*size*()) {  
 cout *<<* "Матрицы не могут быть перемножены!" *<< endl*;  
 return *{}*;  
 }  
  
 *// Создание матрицы на вывод  
 vector*<vector<double>> result*(matrix1*.*size*(), *vector*<double>*(matrix2[*0*]*.*size*()*))*;  
  
 *// Перемножение* for (int i = 0; i < *matrix1*.*size*(); ++i) {  
 for (int j = 0; j < *matrix2[*0*]*.*size*(); ++j) {  
 for (int k = 0; k < *matrix1[*0*]*.*size*(); ++k) {  
 result*[*i*][*j*]* += *matrix1[*i*][*k*]* \* *matrix2[*k*][*j*]*;  
 }  
 }  
 }  
  
 return result;  
}  
  
*// Проверяет данные матрицы на соответствие условиям*void *check\_matrix\_parameters*(double &*rows*, double &*columns*) {  
 if (*rows* < 1.0 || *rows* > 10.0 || *columns* < 1.0 || *columns* > 10.0) {  
 constexpr double default\_parameters[4] = {3.0, 4.0, 4.0, 6.0};  
 cout *<<* "Некорректные параметры, используются параметры по умолчанию: " *<<* default\_parameters[0] *<<* " строк, "  
 *<<* default\_parameters[1] *<<* " столбцов" *<< endl*;  
 *rows* = default\_parameters[0];  
 *columns* = default\_parameters[1];  
 }  
}  
  
void *display\_matrix*(const *vector*<*vector*<double>> &*matrix*) {  
 for (const *vector*<double> &i : *matrix*) {  
 for (const double &j : i) {  
 cout *<<* j *<<* " ";  
 }  
 cout *<< endl*;  
 }  
}  
  
*// Задание 1*void *manipulating\_matrix*() {  
 double row1 = 3.0;  
 double column1 = 4.0;  
 double row2 = 4.0;  
 double column2 = 6.0;  
  
 cout *<<* "Введите количество строк и столбцов для первой матрицы (через пробел от 1 до 10): " *<< endl*;  
 cin *>>* row1 *>>* column1;  
 *check\_matrix\_parameters*(row1, column1);  
  
 cout *<<* "Введите количество строк и столбцов для второй матрицы (через пробел от 1 до 10): " *<< endl*;  
 cin *>>* row2 *>>* column2;  
 *check\_matrix\_parameters*(row2, column2);  
  
 *vector*<vector<double>> matrix1*(*row1, *vector*<double>*(*column1*))*;  
 *vector*<vector<double>> matrix2*(*row2, *vector*<double>*(*column2*))*;  
 cout *<<* "Заполняем матрицу рандомными значениями..." *<< endl*;  
 *fill\_matrix*(matrix1);  
 *fill\_matrix*(matrix2);  
 cout *<<* "Введены значения:" *<< endl*;  
 cout *<<* "Первая матрица: " *<<* row1 *<<* " строк, " *<<* column1 *<<* " столбцов" *<< endl*;  
 *display\_matrix*(matrix1);  
 cout *<<* "Вторая матрица: " *<<* row2 *<<* " строк, " *<<* column2 *<<* " столбцов" *<< endl*;  
 *display\_matrix*(matrix2);  
  
 cout *<<* "Перемножаем матрицы..." *<< endl*;  
 const *vector*<*vector*<double>> result = *multiply\_matrix*(matrix1, matrix2);  
  
 cout *<<* "Новая (перемноженная) матрица:" *<< endl*;  
 *display\_matrix*(result);  
}  
  
*// Задание 2*void *manipulating\_stochastic\_matrix*() {  
 int N = 2;  
  
 cout *<<* "Введите размерность матрицы (N x N): ";  
 cin *>>* N;  
  
 if (N < 2 || N > 10) {  
 cout *<<* "Введены неверные значения. Возвращаем размерность по умолчанию: 2" *<< endl*;  
 N = 2;  
 }  
  
 *vector*<vector<double>> matrix*(*N, *vector*<double>*(*N*))*;  
 *fill\_stochastic\_matrix*(matrix);  
  
 cout *<<* "Сгенерированная изначальная матрица:" *<< endl*;  
 *display\_matrix*(matrix);  
  
 cout *<<* "Начинаем возведение матриц..." *<< endl*;  
 for (int i = 0; i < 20; ++i) {  
 matrix *= multiply\_matrix*(matrix, matrix);  
 cout *<<* "Шаг № " *<<* i + 1 *<<* ": Промежуточная матрица:" *<< endl*;  
 *display\_matrix*(matrix);  
 }  
  
 *// Вектор-строка начальных вероятностей (равномерное распределение)* const *vector*<vector<double>> initial\_probabilities*(*1, *vector*<double>*(*N, 1.0 / N*))*;  
 cout *<<* "Матрица первоначальных вероятностей: " *<< endl*;  
 *display\_matrix*(initial\_probabilities);  
  
 cout *<<* "Перемножаем матрицы..." *<< endl*;  
 *// Вектор-строка умножается на стохастическую матрицу (строки на столбцы)* const *vector*<*vector*<double>> result = *multiply\_matrix*(initial\_probabilities, matrix);  
  
 cout *<<* "Итоговая матрица вероятностей:" *<< endl*;  
 *display\_matrix*(result);  
}  
  
int *main*() {  
 *SetConsoleCP*(CP\_UTF8);  
 *SetConsoleOutputCP*(CP\_UTF8);  
 *setlocale*(LC\_ALL, ".UTF8");  
  
 char choice = '0';  
  
 cout *<<* "Выберите задание (0 — перемножение матриц (default), 1 — стохастическая матрица): ";  
 cin *>>* choice;  
  
 if (choice == '1') {  
 *manipulating\_stochastic\_matrix*();  
 } else {  
 *manipulating\_matrix*();  
 }  
  
 return 0;  
}

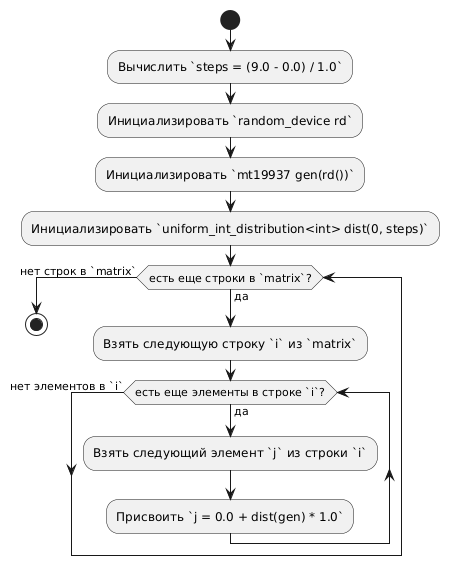
**Запуск программы:**

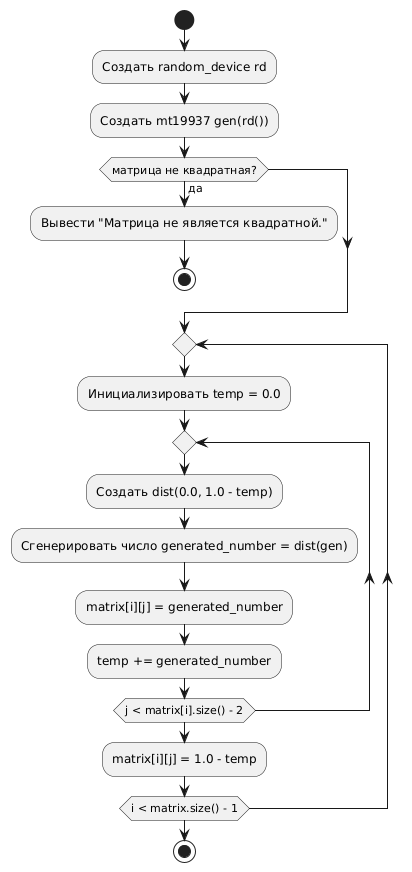


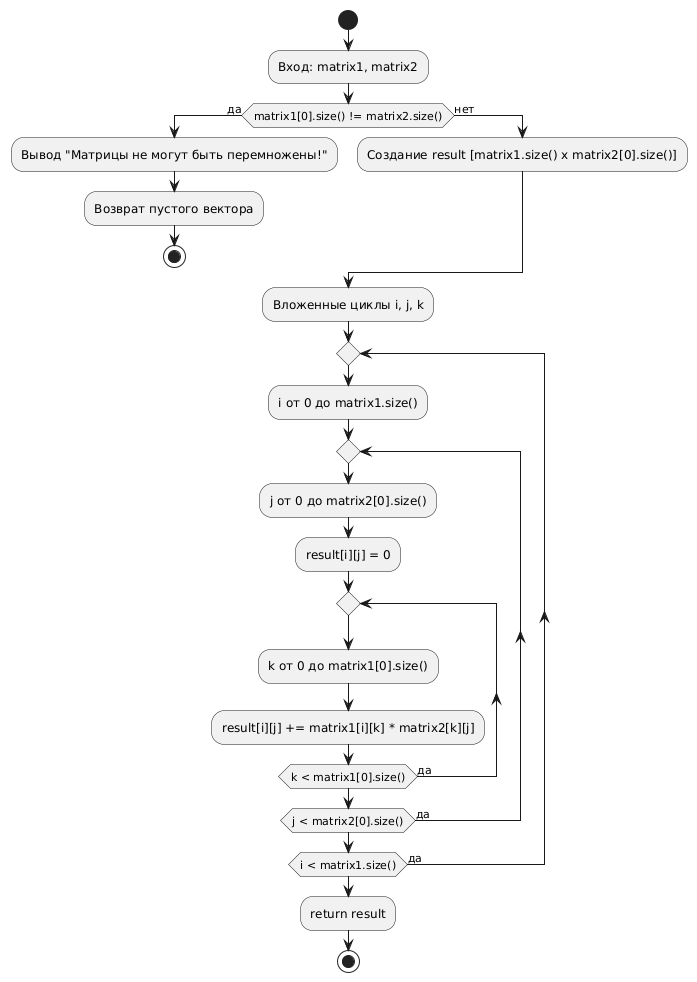


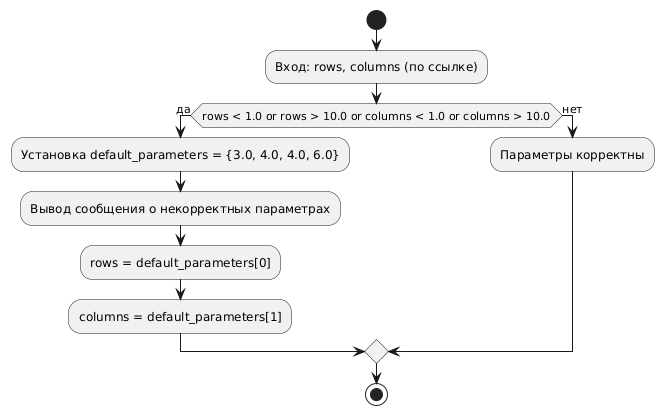
**Блок-схемы:**

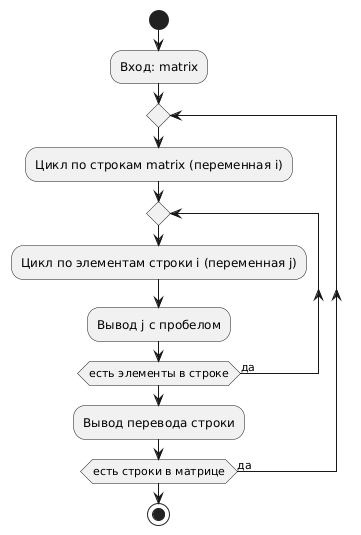
fill\_matrix:



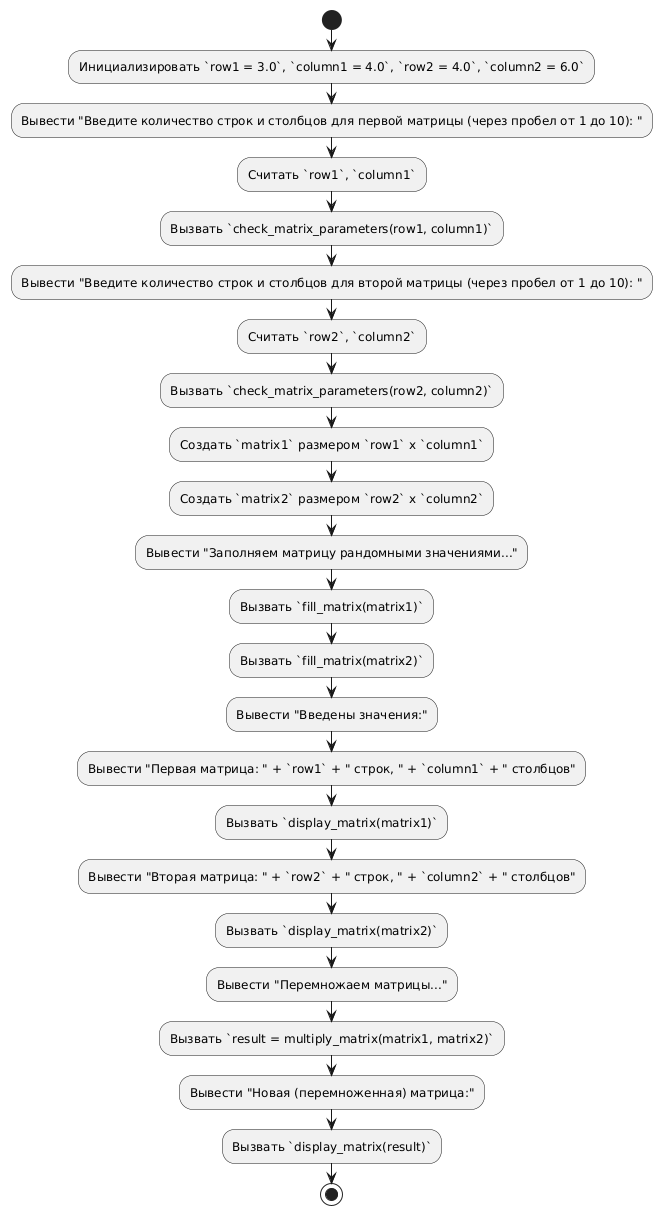
fill\_stochastic\_matrix:

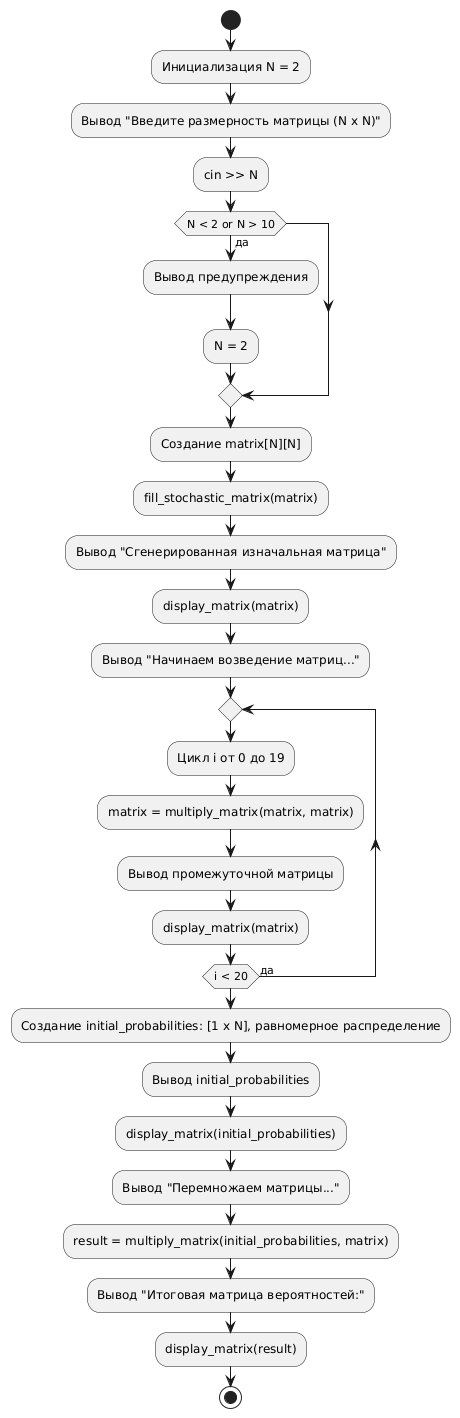
multiply\_matrix:

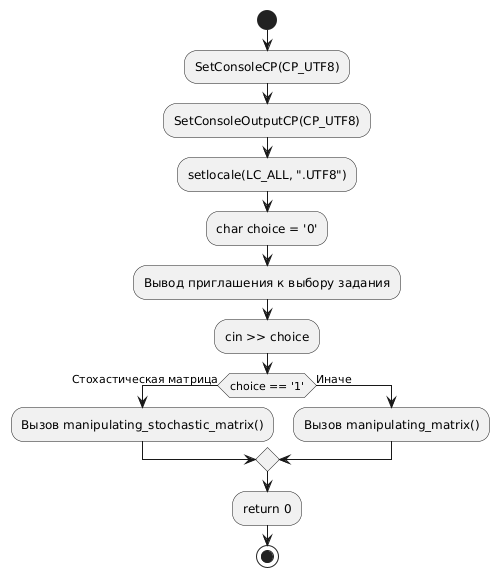
check\_matrix\_parameters:

display\_matrix:

manipulating\_matrix:



manipulating\_stochastic\_matrix:

main: