Министерство науки и высшего образования РФ
ФГАОУ ВО «Уральский федеральный университет
имени первого Президента России Б. Н. Ельцина»
ИРИТ-РТФ
Центр ускоренного обучения

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

по дисциплине «Прикладное программирование»

Тема: ***Приобретение навыков работы с матричным представлением данных, обработка матриц***

Студент группы РИЗ-200028у: И. С. Арсентьев
Преподаватель: О. Л. Чагаева,
ст. преподаватель

Екатеринбург 2022

СОДЕРЖАНИЕ

1 Постановка задачи

Цель: Задача заключается в том, что бы написать функции которые будут рассчитывать:

1. максимум матрицы,
2. минимум матрицы,
3. максимум верхнетреугольной части матрицы,
4. минимум верхнетреугольной части матрицы,
5. минимум нижнетреугольной части матрицы,
6. максимум нижнетреуголыюй части матрицы,
7. максимум главной диагонали матриц,
8. минимум главной диагонали матрицы,
9. минимум второстепенной диагонали матрицы,
10. среднее арифметическое значение элементов нижнетреугольной части матрицы,
11. среднее арифметическое значение элементов верхнетреугольной части матрицы,
12. среднее арифметическое значение элементов матрицы,
13. суммы строк матрицы,
14. минимальные значения строк,
15. максимальные значения строк,
16. минимальные значения столбцов,
17. максимальные значения столбцов,
18. средние арифметические значения столбцов,
19. средние арифметические значения строк,
20. сумма верхнетреугольной части матрицы,
21. сумма нижнетреугольной части матрицы,
22. элемент, наиболее близкий по значению к среднему арифметическому.

2 Описание работы

2.1. Выполнение поставленных задач.

```
#include <iostream>
#include <iomanip>
#include <stdlib.h>
#include <conio.h>
#define N 5 //пять столбцов,пять строк
using namespace std;
void main()
{
	setlocale(LC_ALL, "ru"); // русская кодировка в командной строке
	float m[N][N];
	//ввод прототипов функций
	float matMax(float(&arr)[N][N]);//максимум матрицы
	float matMin(float(&arr)[N][N]);//минимум матрицы
	float triUMax(float(&arr)[N][N]);//максимум верхнетреугольной части
матрицы
	float triUMin(float(&arr)[N][N]);//минимум верхнетреугольной части
матрицы
	float triDMax(float(&arr)[N][N]);//максимум нижнетреугольной части
матрицы
	float triDMin(float(&arr)[N][N]);//минимум нижнетреугольной части
матрицы
	float mDiagMax(float(&arr)[N][N]);//максимум главной диагонали матрицы
	float mDiagMin(float(&arr)[N][N]);//минимум главной диагонали матрицы
	float scDiagMax(float(&arr)[N][N]);//максимум второстепенной диагонали
матрицы
	float scDiagMin(float(&arr)[N][N]);//минимум второстепенной диагонали
матрицы
	float triDsred(float(&arr)[N][N]);//среднее арифметическое значение
элементов нижнетреугольной части матрицы
	float triUsred(float(&arr)[N][N]);//среднее арифметическое значение
элементов верхнетреугольной части матрицы
	float matSred(float(&arr)[N][N]);//среднее арифметическое значение
элементов матрицы
	float* rowSum(float(&arr)[N][N]);//сумма строк матрицы
	float* colSum(float(&arr)[N][N]);//сумма столбцов матрицы
	float* Minrow(float(&arr)[N][N]);//минимальные значения строк
	float* Maxrow(float(&arr)[N][N]);//максимальные значения строк
	float* Maxcol(float(&arr)[N][N]);//максимальные значения столбцов
	float* Mincol(float(&arr)[N][N]);//минимальные значения столбцов
	float* sredcol(float(&arr)[N][N]);//средние арифметические значения
столбцов
	float* sredrow(float(&arr)[N][N]);//средние арифметические значения
строк
	float triUsum(float(&arr)[N][N]);//сумма верхнетреугольной части
матрицы
	float triDsum(float(&arr)[N][N]);//сумма нижнетреугольной части матрицы
	float matnearSred(float(&arr)[N][N]);//элемент, наиболее близкий по
значению к среднему арифметическому
	int i, j;
	// заполнение исходного массива
	for (i = 0; i < N; i++)
	{
		for (j = 0; j < N; j++)
		{
			m[i][j] = rand() / 12.f;
		}
```

```
	}
	//вывод исходного массива и, последовательно всех функций
	cout << "Исходный массив" << endl;
	for (i = 0; i < N; i++)
	{
		for (j = 0; j < N; j++)
		{
			cout << setw(8) << setprecision(5) << m[i][j];
		}
		cout << endl;
	}
	cout << "максимум матрицы = " << matMax(m) << endl;
	cout << "минимум матрицы = " << matMin(m) << endl;
	cout << "максимум верхнетреугольной части матрицы = "<< triUMax(m) <<
endl;
	cout << "минимум верхнетреугольной части матрицы = "<< triUMin(m) <<
endl;
	cout << "минимум нижнетреугольной части матрицы = "<< triDMin(m) <<
endl;
	cout << "максимум нижнетреуголыюй части матрицы = "<< triDMax(m) <<
endl;
	cout << "максимум главной диагонали матрицы = " << mDiagMax(m) << endl;
	cout << "минимум главной диагонали матрицы = " << mDiagMin(m) << endl;
	cout << "максимум второстепенной диагонали матрицы = "<< scDiagMax(m)
<< endl;
	cout << "минимум второстепенной диагонали матрицы = "<< scDiagMin(m) <<
endl;
	cout << "среднее арифметическое значение элементов нижнетреугольной
части матрицы = "<< triDsred(m) << endl;
	cout << "среднее арифметическое значение элементов верхнетреугольной
части матрицы = "<< triUsred(m) << endl;
	cout << "среднее арифметическое значение элементов матрицы = "<<
matSred(m) << endl;
	cout << "суммы строк матрицы" << endl;
	for (int k = 0; k < N; k++)
	{
		cout << *(rowSum(m) + k) << endl;
	}
	cout << " суммы строк матрицы " << endl;
	for (int k = 0; k < N; k++)
	{
		cout << *(colSum(m) + k) << "  ";
	}
	cout << endl;
	cout << "минимальные значения строк" << endl;
	for (int k = 0; k < N; k++)
	{
		cout << *(Minrow(m) + k) << endl;
	}
	cout << "максимальные значения строк" << endl;
	for (int k = 0; k < N; k++)
	{
		cout << *(Maxrow(m) + k) << endl;
	}
	cout << "минимальные значения столбцов" << endl;
	for (int k = 0; k < N; k++)
	{
		cout << *(Mincol(m) + k) << "  ";
	}
	cout << endl;
	cout << "максимальные значения столбцов" << endl;
```

```
	for (int k = 0; k < N; k++)
	{
		cout << *(Maxcol(m) + k) << "  ";
	}
	cout << endl;
	cout << "среднее арифметические значения столбцов" << endl;
	for (int k = 0; k < N; k++)
	{
		cout << *(sredcol(m) + k) << "  ";
	}
	cout << endl;
	cout << "среднее арифметическое значений строк" << endl;
	for (int k = 0; k < N; k++)
	{
		cout << *(sredrow(m) + k) << endl;
	}
	cout << "сумма верхнетреугольной части матрицы = "
		<< triUsum(m) << endl;
	cout << "сумма нижнетреугольной части матрицы = "
		<< triDsum(m) << endl;
	cout << "элемент, наиболее близкий по значению к среднему
арифметическому = "
		<< matnearSred(m) << endl;
}
//максимум матрицы
float matMax(float(&arr)[N][N]) {
	float maxM = arr[0][0];
	for (int i = 0; i < N; i++)
	{
		for (int j = 0; j < N; j++)
		{
			if (arr[i][j] > maxM)
			{
				maxM = arr[i][j];
			}
		}
	}
	return maxM;
}
//минимум матрицы
float matMin(float(&arr)[N][N]) {
	float minM = arr[0][0];
	for (int i = 0; i < N; i++)
	{
		for (int j = 0; j < N; j++)
		{
			if (arr[i][j] < minM)
			{
				minM = arr[i][j];
			}
		}
	}
	return minM;
}
//максимум верхнетреугольной части матрицы
float triUMax(float(&arr)[N][N])
{
	int k = 0, j;
	float maxM;
	maxM = 0;
	for (int i = 0; i < N; i++)
```

```
	{
		j = k++;
		for (; j < N; j++)
		{
			if (arr[i][j] > maxM)
			{
				maxM = arr[i][j];
			}
		}
	}
	return maxM;
}
//минимум верхнетреугольной части матрицы
float triUMin(float(&arr)[N][N])
{
	int k = 0, j;
	float minM;
	minM = arr[0][0];
	for (int i = 0; i < N; i++)
	{
		j = k++;
		for (; j < N; j++)
		{
			if (arr[i][j] < minM)
			{
				minM = arr[i][j];
			}
		}
	}
	return minM;
}
//максимум нижнетреуголыюй части матрицы
float triDMin(float(&arr)[N][N])
{
	int k = 1, n;
	float minM;
	minM = arr[0][0];
	for (int i = 0; i < N; i++)
	{
		n = k++;
		for (int j = 0; j < n; j++)
		{
			if (arr[i][j] < minM)
			{
				minM = arr[i][j];
			}
		}
	}
	return minM;
}
//максимум главной диагонали матрицы
float triDMax(float(&arr)[N][N])
{
	int k = 1, n;
	float maxM;
	maxM = 0;
	for (int i = 0; i < N; i++)
	{
		n = k++;
		for (int j = 0; j < n; j++)
		{
```

```
                if (arr[i][j] > maxM)
                {
                      maxM = arr[i][j];
                }
          }
      }
      return maxM;
}
//максимум главной диагонали матрицы
float mDiagMax(float(&arr)[N][N])
{     float maxM;
      maxM = 0;
      for (int i = 0; i < N; i++)
      {
            if (arr[i][i] > maxM)
            {
                  maxM = arr[i][i];
            }
      }
      return maxM;
}
//минимум главной диагонали матрицы
float mDiagMin(float(&arr)[N][N])
{
      float minM;
      minM = arr[0][0];
      for (int i = 0; i < N; i++)
      {
            if (arr[i][i] < minM)
            {
                  minM = arr[i][i];
            }
      }
      return minM;
}
//минимум второстепенной диагонали матрицы
float scDiagMin(float(&arr)[N][N])
{
      float minM;
      minM = arr[N - 1][0];
      for (int i = N - 1; i >= 0; i--)
      {
            if (arr[i][N - i - 1] < minM)
            {
                  minM = arr[i][N - i - 1];
            }
      }
      return minM;
}
//максимум второстепенной диагонали матрицы
float scDiagMax(float(&arr)[N][N])
{
      float maxM;
      maxM = 0;
      for (int i = N - 1; i >= 0; i--)
      {
            if (arr[i][N - i - 1] > maxM)
            {
                  maxM = arr[i][N - i - 1];
            }
      }
```

```
	return maxM;
}
//среднее арифметическое значение элементов матрицы
float matSred(float(&arr)[N][N]) {
	float sred = 0;
	for (int i = 0; i < N; i++)
	{
		for (int j = 0; j < N; j++)
		{

			sred += arr[i][j];
		}
	}
	sred /= N * N;
	return sred;
}
//среднее арифметическое значение элементов верхнетреугольной части матрицы
float triUsred(float(&arr)[N][N])
{
	int k = 0, j;
	float sred = 0;

	for (int i = 0; i < N; i++)
	{
		j = k++;
		for (; j < N; j++)
		{
			sred += arr[i][j];
		}
	}
	sred /= (N + ((float)pow(N, 2) - N) / 2);
	return sred;
}
//среднее арифметическое значение элементов нижнетреугольной части матрицы
float triDsred(float(&arr)[N][N])
{
	int k = 1, n;
	float sred = 0;
	for (int i = 0; i < N; i++)
	{
		n = k++;
		for (int j = 0; j < n; j++)
		{
			sred += arr[i][j];
		}
	}
	sred /= (N + ((float)pow(N, 2) - N) / 2);
	return sred;
}
//суммы строк матрицы
float* rowSum(float(&arr)[N][N]) {
	float Sum;
	float narr[N];
	for (int i = 0; i < N; i++)
	{
		Sum = 0;
		for (int j = 0; j < N; j++)
		{
			Sum += arr[i][j];
		}
		narr[i] = Sum;
```

```
	}
	return narr;
}
//суммы столбцов матрицы
float* colSum(float(&arr)[N][N]) {
	float Sum;
	float narr[N];
	for (int i = 0; i < N; i++)
	{
		Sum = 0;
		for (int j = 0; j < N; j++)
		{
			Sum += arr[j][i];
		}
		narr[i] = Sum;
	}
	return narr;
}
//минимальные значения строк
float* Minrow(float(&arr)[N][N]) {
	float min;
	float narr[N];
	for (int i = 0; i < N; i++)
	{
		min = arr[i][0];
		for (int j = 0; j < N; j++)
		{
			if (arr[i][j] < min)
			{
				min = arr[i][j];
			}
		}
		narr[i] = min;
	}
	return narr;
}
//максимальные значения строк
float* Maxrow(float(&arr)[N][N]) {
	float max;
	float narr[N];
	for (int i = 0; i < N; i++)
	{
		max = 0;
		for (int j = 0; j < N; j++)
		{
			if (arr[i][j] > max)
			{
				max = arr[i][j];
			}
		}
		narr[i] = max;
	}
	return narr;
}
//максимальные значения столбцов
float* Maxcol(float(&arr)[N][N]) {
	float max;
	float narr[N];
	for (int i = 0; i < N; i++)
	{
		max = 0;
```

```
            for (int j = 0; j < N; j++)
            {
                  if (max < arr[j][i])
                  {
                        max = arr[j][i];
                  }
            }
            narr[i] = max;
      }
      return narr;
}
//минимальные значения столбцов
float* Mincol(float(&arr)[N][N]) {
      float min;
      float narr[N];
      for (int i = 0; i < N; i++)
      {
            min = arr[0][i];
            for (int j = 0; j < N; j++)
            {
                  if (min > arr[j][i])
                  {
                        min = arr[j][i];
                  }
            }
            narr[i] = min;
      }
      return narr;
}
//среднее арифметическое значения строк
float* sredrow(float(&arr)[N][N]) {
      float sred;
      float narr[N];
      for (int i = 0; i < N; i++)
      {
            sred = 0;
            for (int j = 0; j < N; j++)
            {
                  sred += arr[i][j];
            }
            sred /= N;
            narr[i] = sred;
      }
      return narr;
}
//среднее арифметическое значения столбцов
float* sredcol(float(&arr)[N][N]) {
      float sred;
      float narr[N];
      for (int i = 0; i < N; i++)
      {
            sred = 0;
            for (int j = 0; j < N; j++)
            {
                  sred += arr[j][i];
            }
            sred /= N;
            narr[i] = sred;
      }
      return narr;
}
```

```
//сумма верхнетреугольной части матрицы
float triUsum(float(&arr)[N][N])
{
	int k = 0, j;
	float sum = 0;
	for (int i = 0; i < N; i++)
	{
		j = k++;
		for (; j < N; j++)
		{

			sum += arr[i][j];

		}
	}
	return sum;
}
//сумма нижнетреугольной части матрицы
float triDsum(float(&arr)[N][N])
{
	int k = 1, n;
	float sum = 0;
	for (int i = 0; i < N; i++)
	{
		n = k++;
		for (int j = 0; j < n; j++)
		{

			sum += arr[i][j];

		}
	}
	return sum;
}
//элемент, наиболее близкий по значению к среднему арифметическому
float matnearSred(float(&arr)[N][N]) {
	float sred = 0, nsred, diff;
	for (int i = 0; i < N; i++)
	{
		for (int j = 0; j < N; j++)
		{

			sred += arr[i][j];
		}
	}
	sred /= N * N;
	diff = abs(arr[0][0] - sred);
	nsred = arr[0][0];
	for (int i = 0; i < N; i++)
	{
		for (int j = 0; j < N; j++)
		{
			if (abs(arr[i][j] - sred) < diff)
			{
				diff = abs(arr[i][j] - sred);
				nsred = arr[i][j];
			}
		}
	}
	return nsred;
}
```

Ниже будут приведены блок-схемы для каждого метода, реализованного в коде.

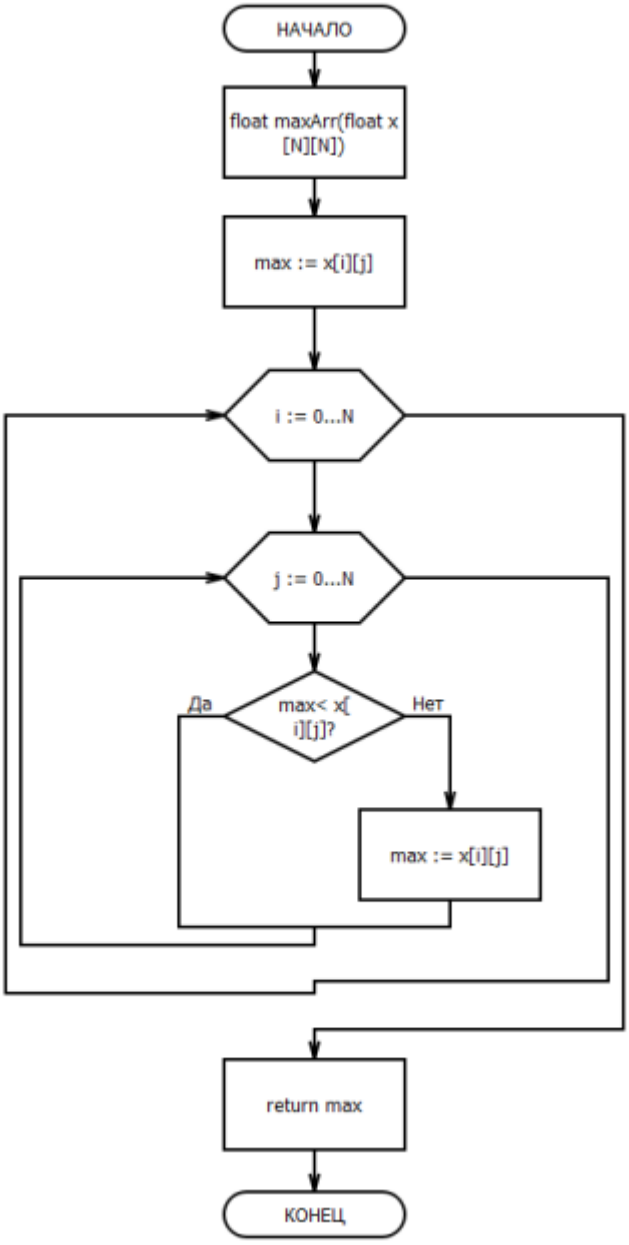

Рисунок 1 – Блок-схема для метода вычисления максимума матрицы

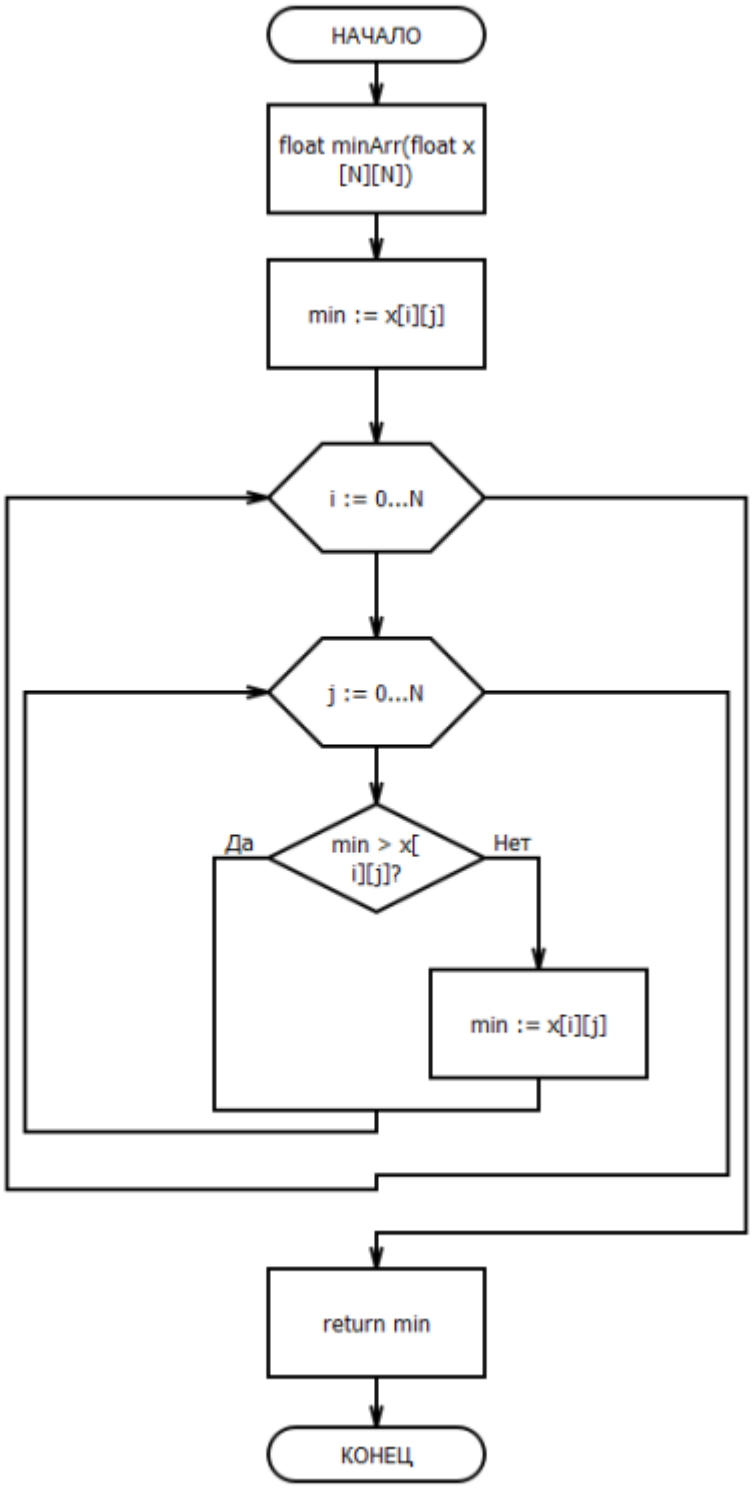

Рисунок 2 – Блок-схема для метода вычисления минимума матрицы

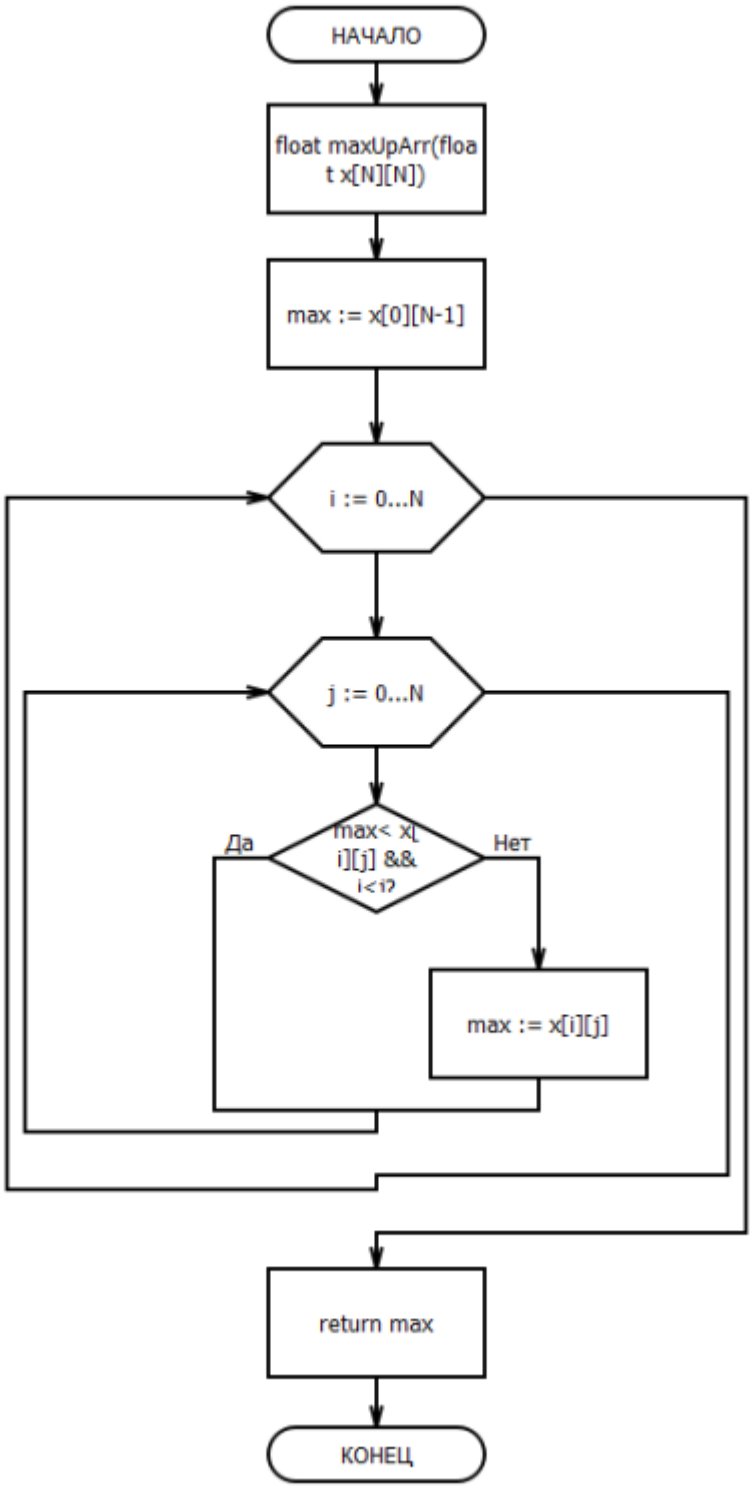

Рисунок 3 – Блок-схема для метода вычисления максимума верхнетреугольной матрицы

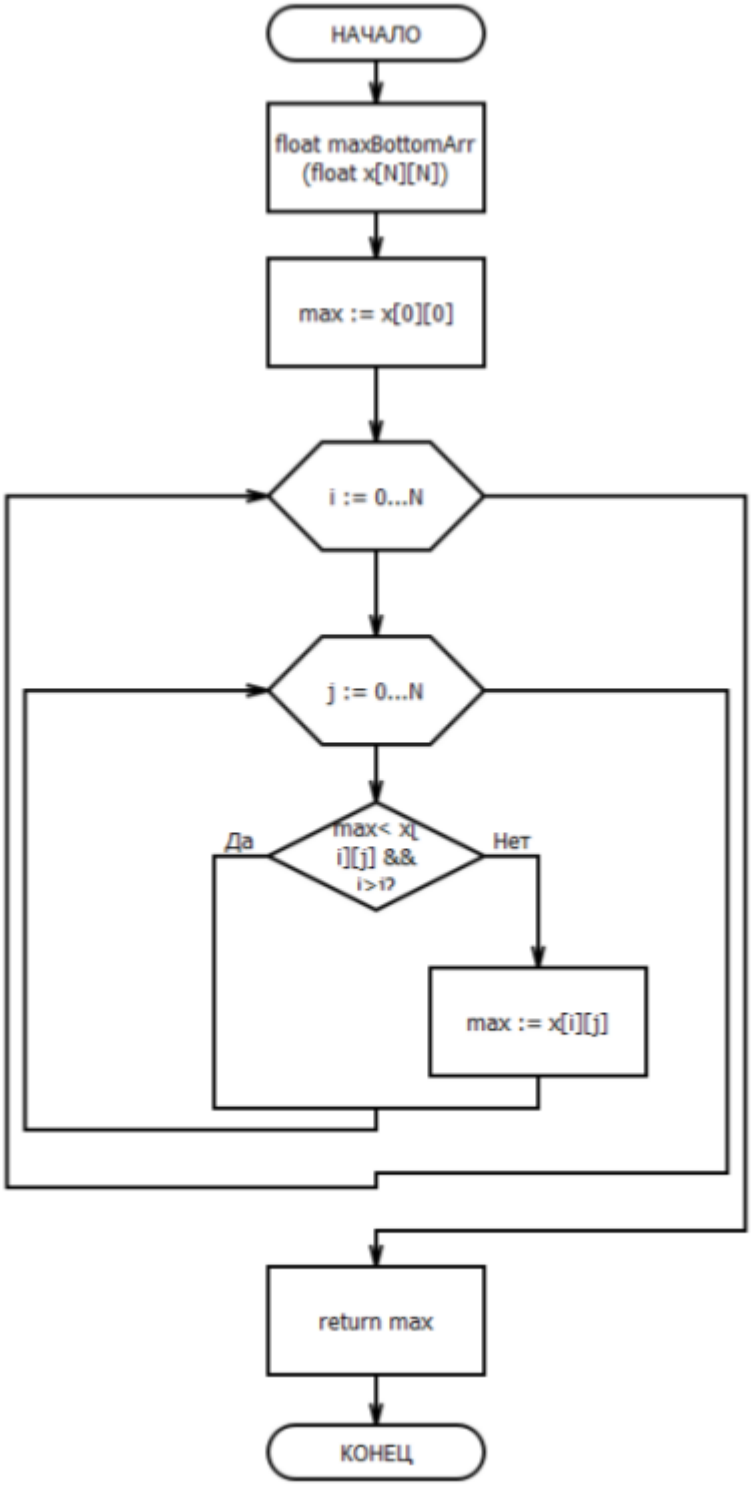

Рисунок 4 – Блок-схема для метода вычисления максимума нижнетреугольной матрицы

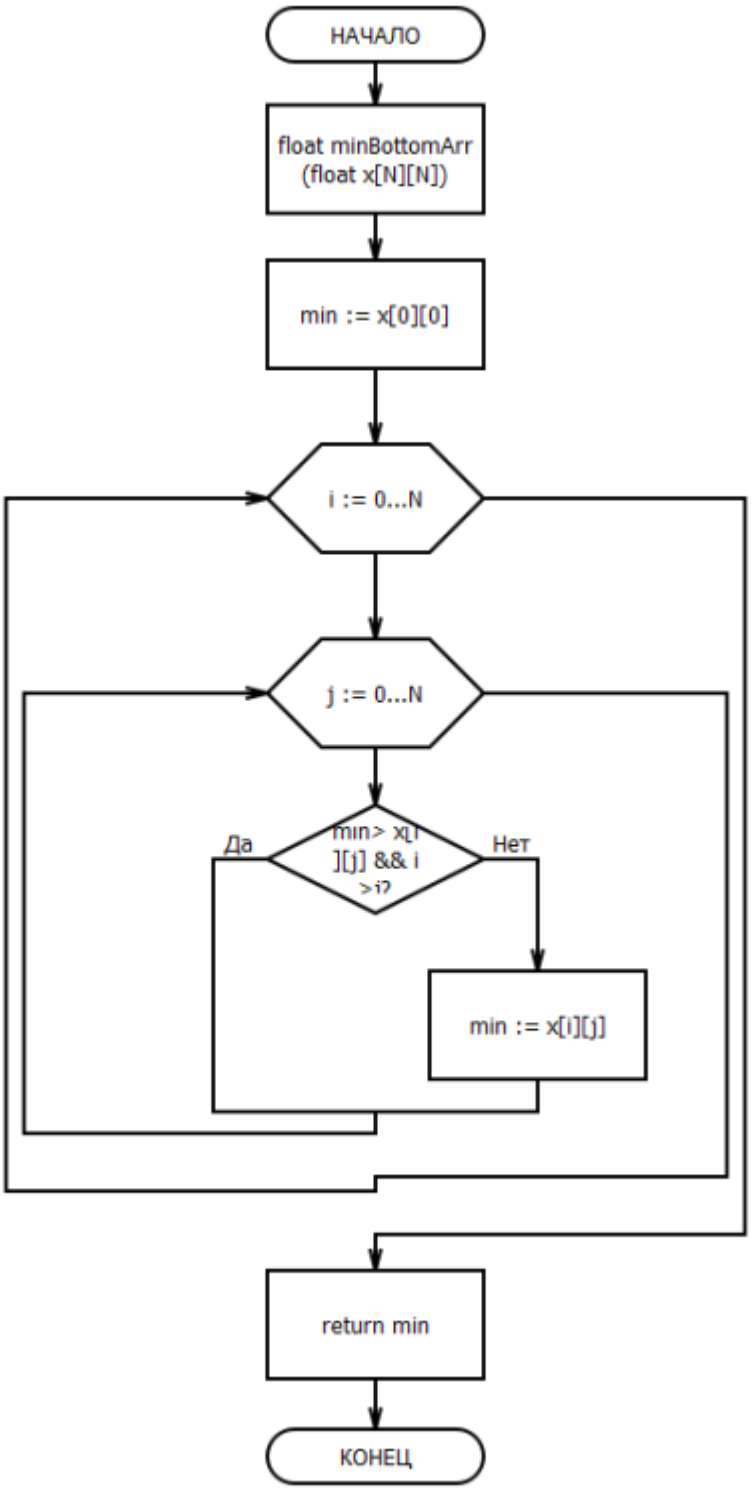

Рисунок 5 – Блок-схема для метода вычисления минимума нижнетреугольной матрицы

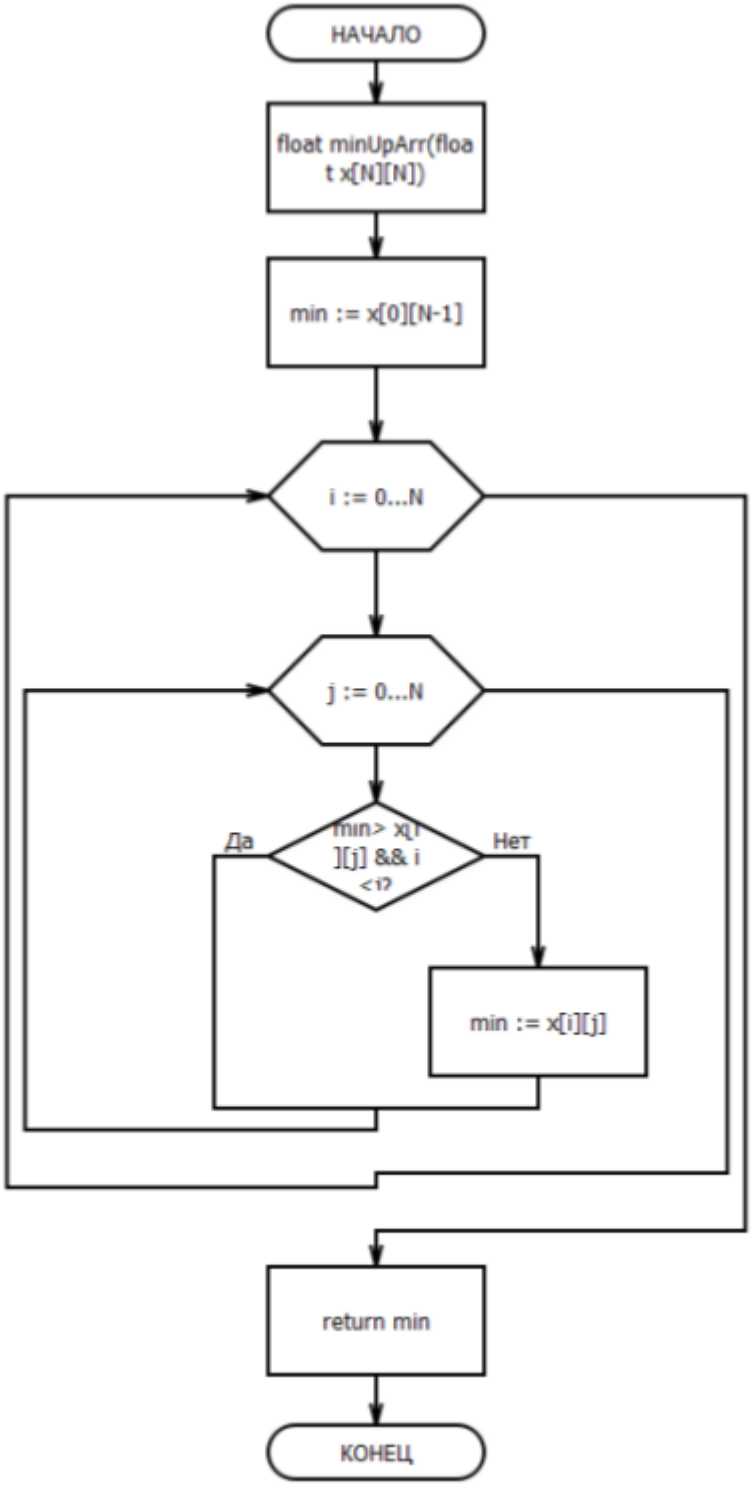

Рисунок 6 – Блок-схема для метода вычисления минимума главной диагонали матрицы

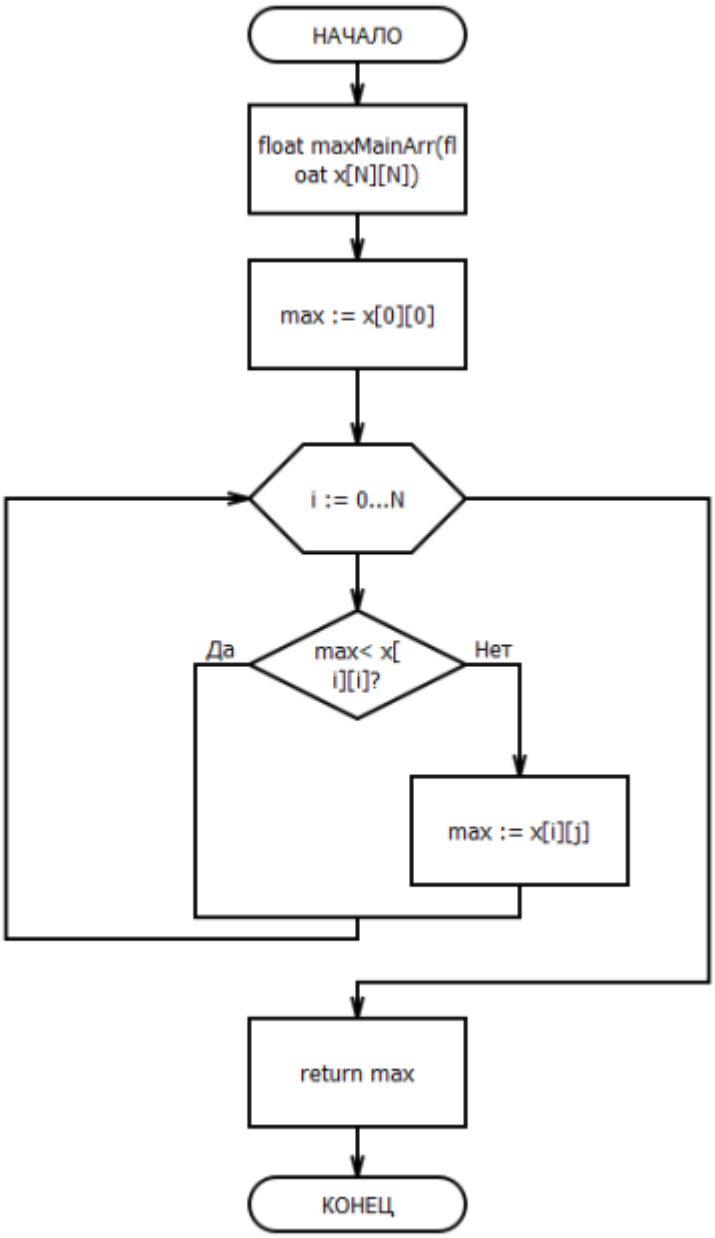

Рисунок 7 – Блок-схема для метода вычисления максимума главной диагонали матрицы

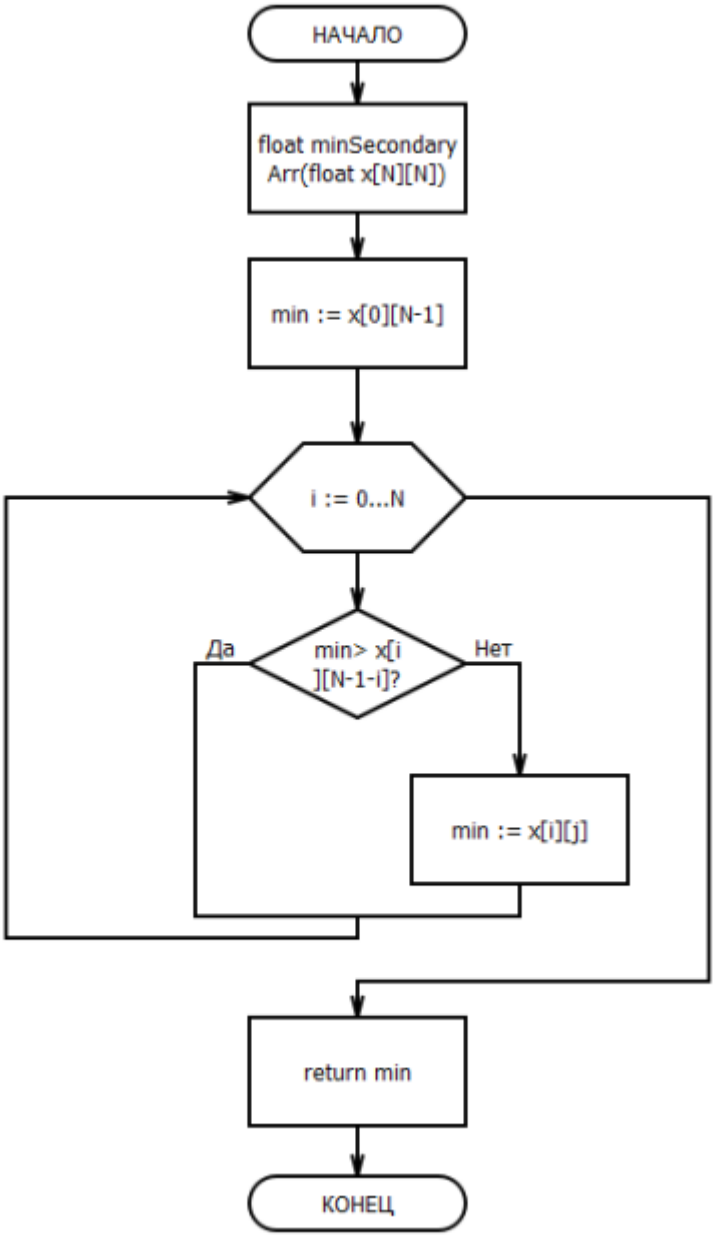

Рисунок 8 – Блок-схема для метода вычисления минимума второстепенной диагонали матрицы

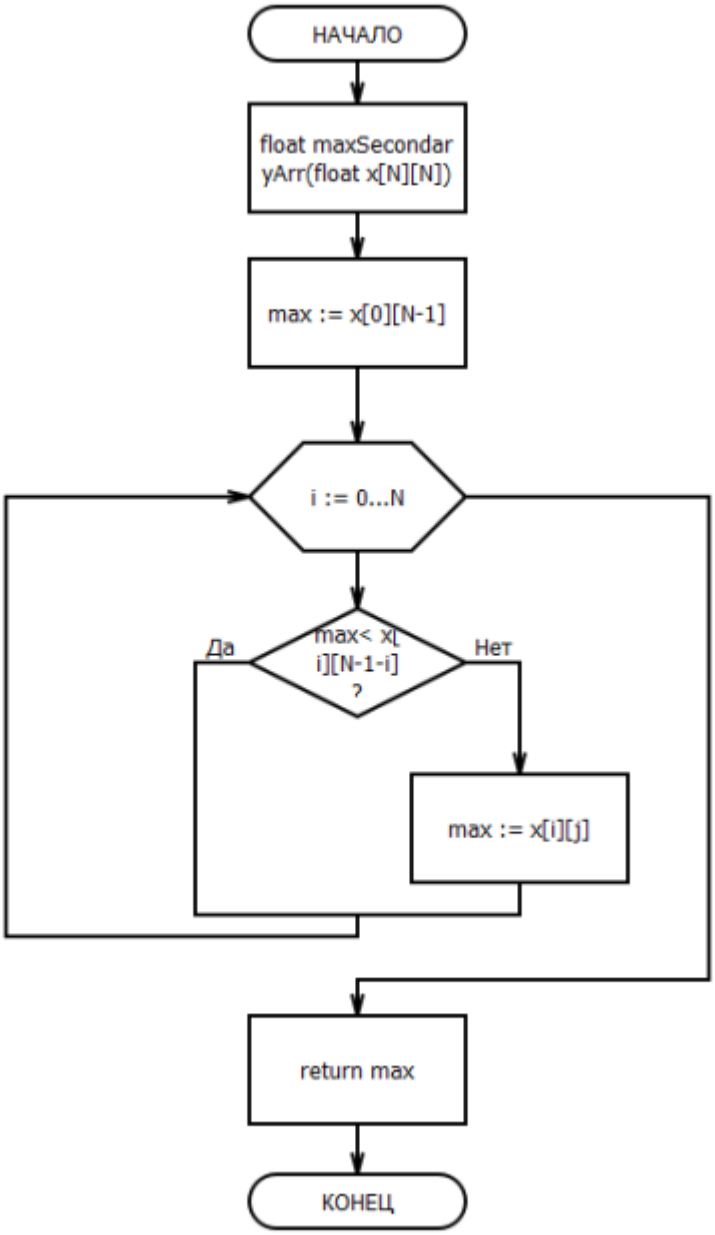

Рисунок 9 – Блок-схема для метода вычисления максимума второстепенной диагонали матрицы

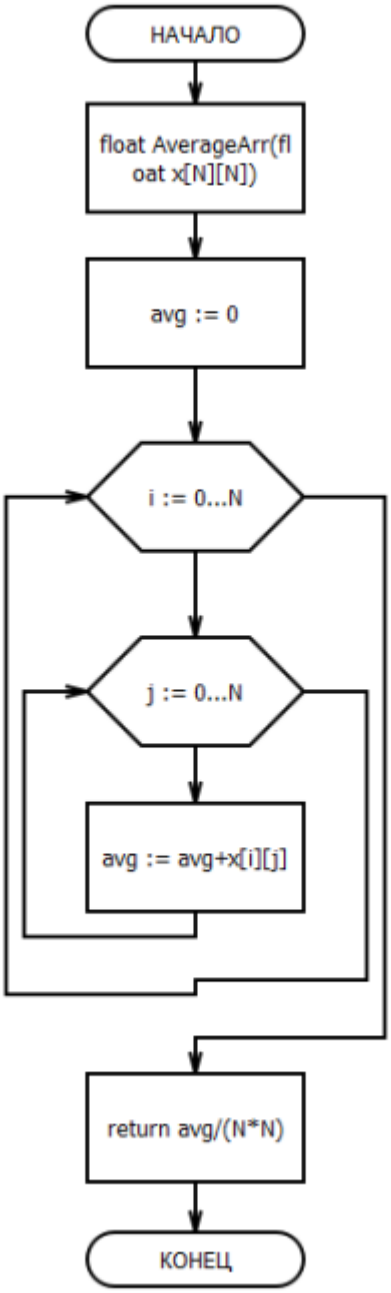

Рисунок 10 – Блок-схема для метода вычисления среднего арифметического значения элементов матрицы

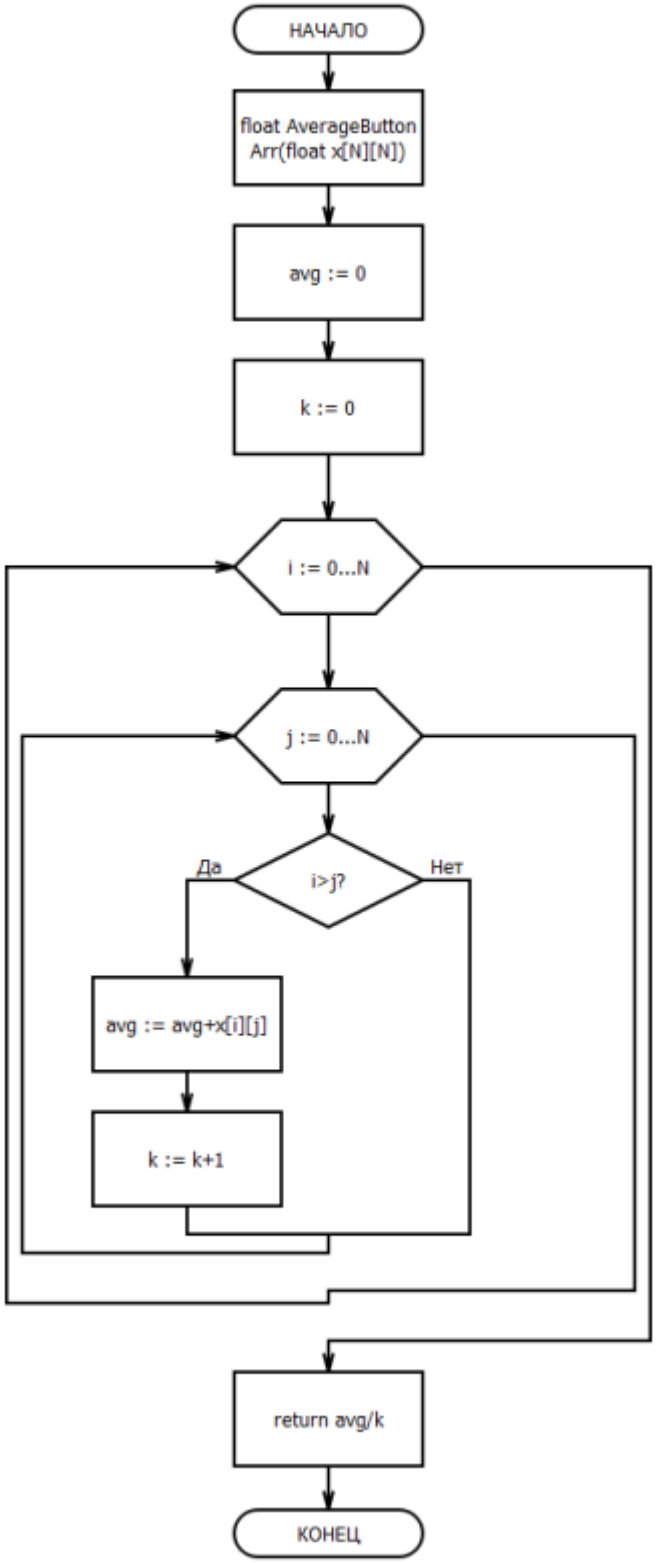

Рисунок 11 – Блок-схема для метода вычисления среднего арифметического значения элементов нижнетреугольной части матрицы

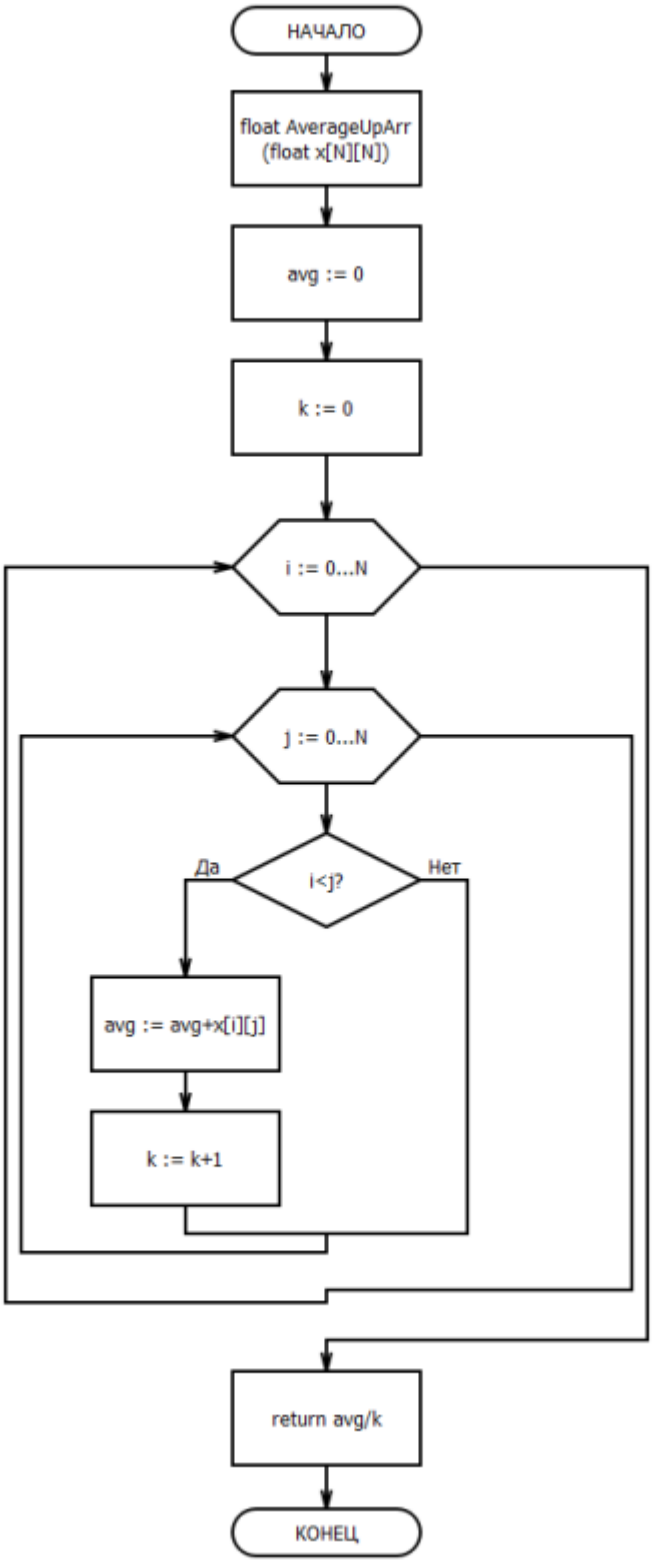

Рисунок 12 – Блок-схема для метода вычисления среднего арифметического значения элементов верхнетреугольной части матрицы

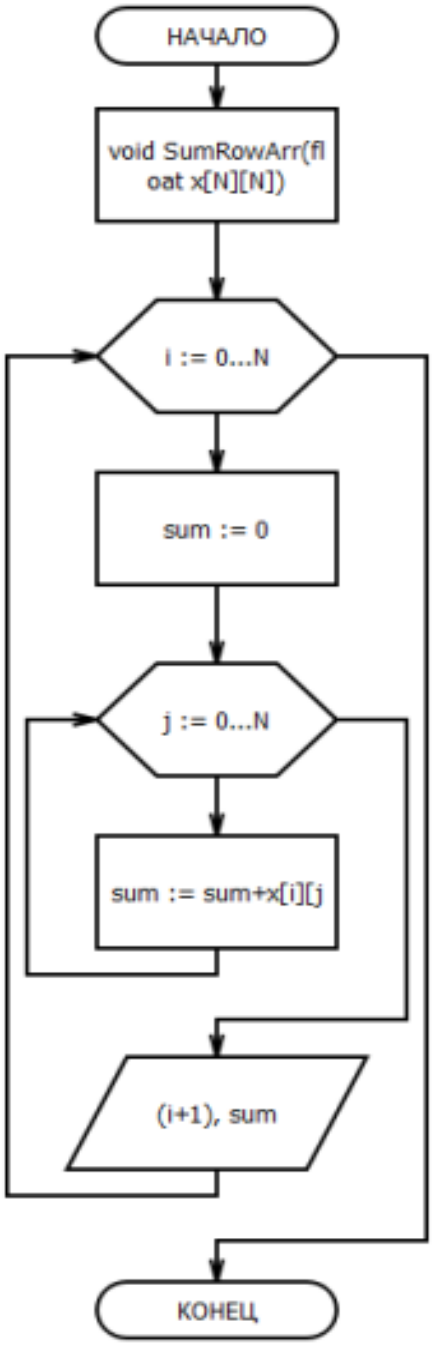

Рисунок 13 – Блок-схема для метода вычисления суммы строк матрицы

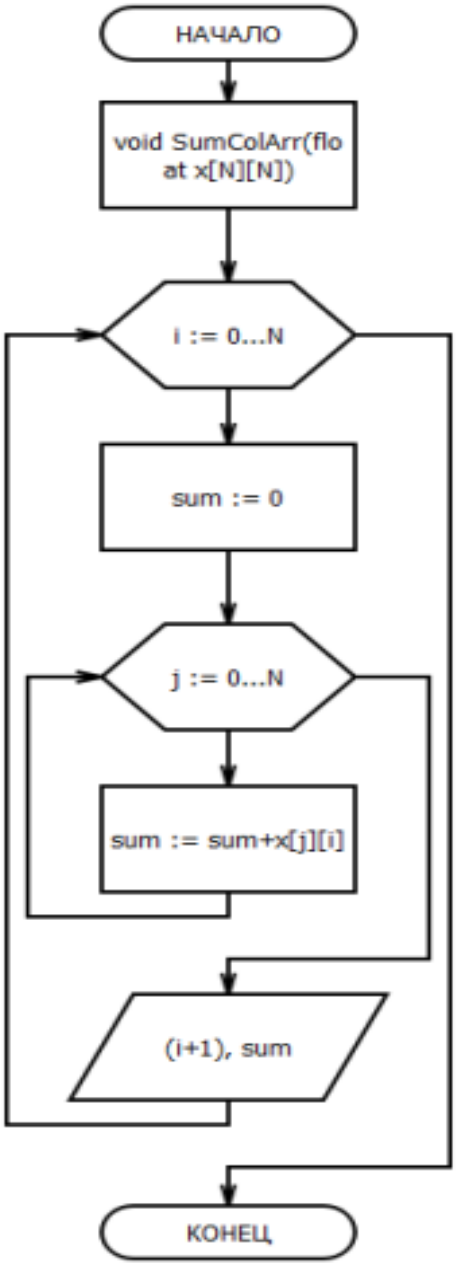

Рисунок 14 – Блок-схема для метода вычисления суммы столбцов матрицы

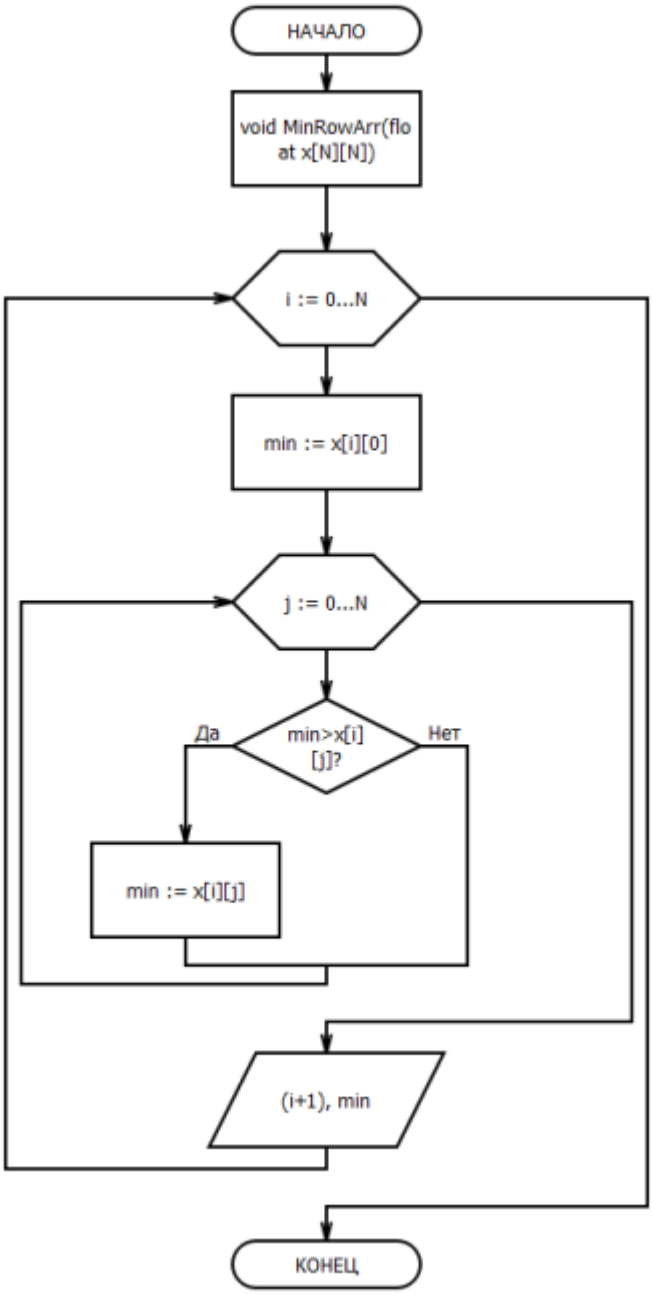

Рисунок 15 – Блок-схема для метода вычисления минимальных значений столбцов матрицы

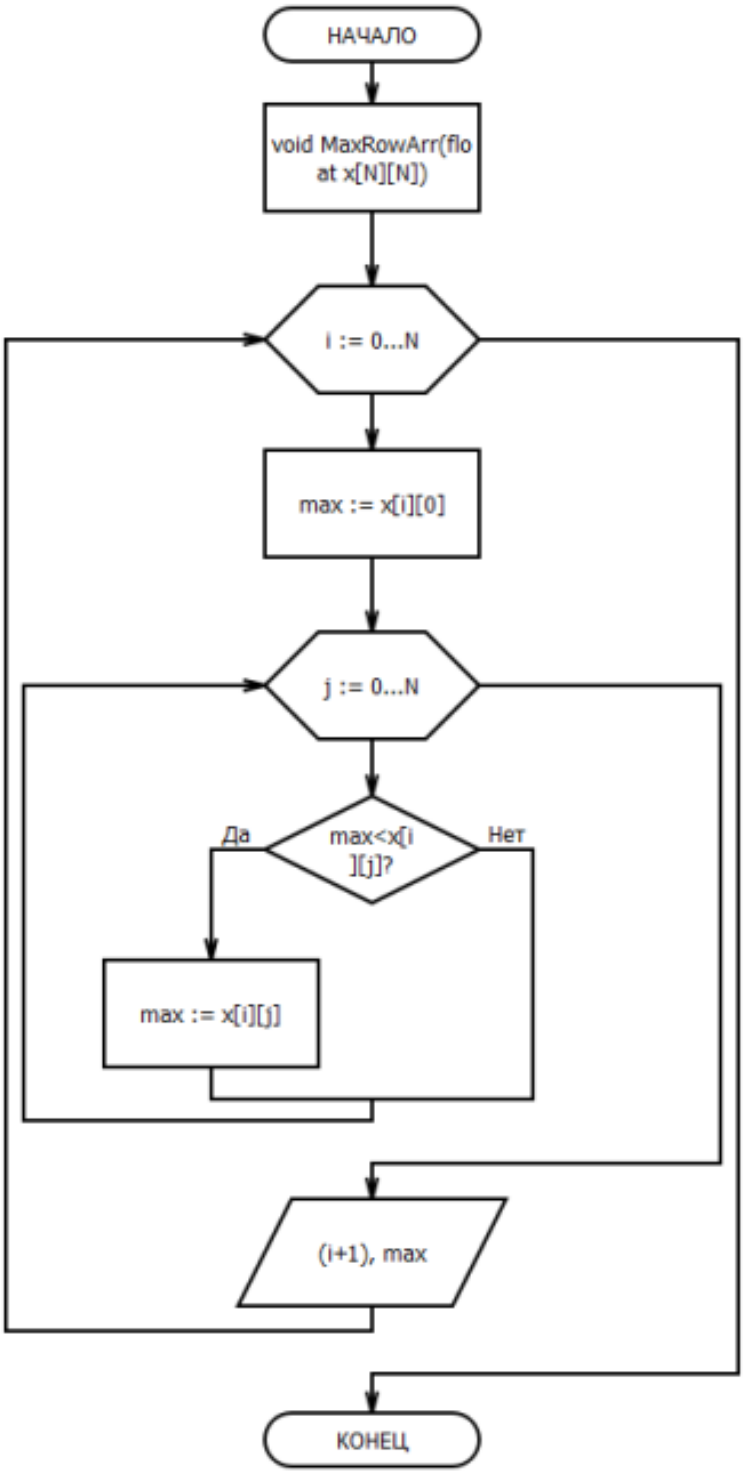

Рисунок 16 – Блок-схема для метода вычисления минимальных значений строк матрицы

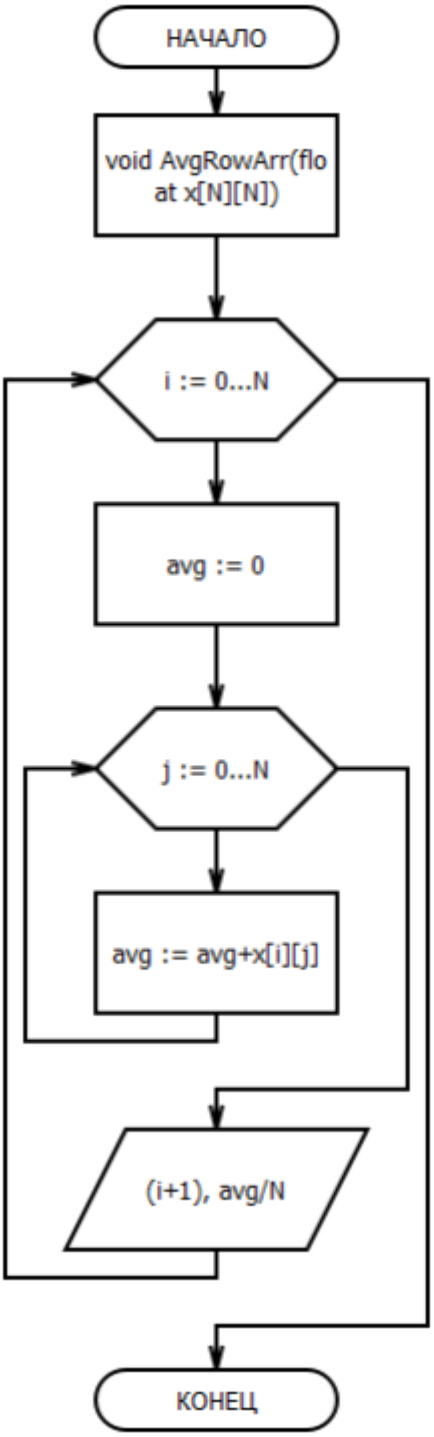

Рисунок 17 – Блок-схема для метода вычисления среднего арифметического значения строк матрицы

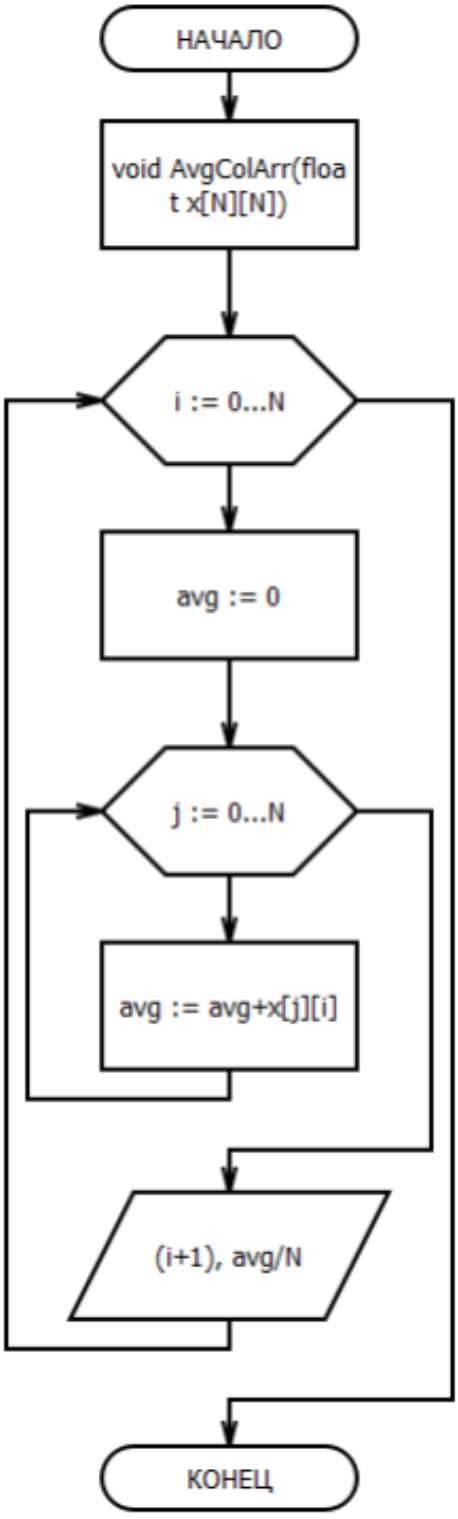

Рисунок 18 – Блок-схема для метода вычисления среднего арифметического значения столбцов матрицы

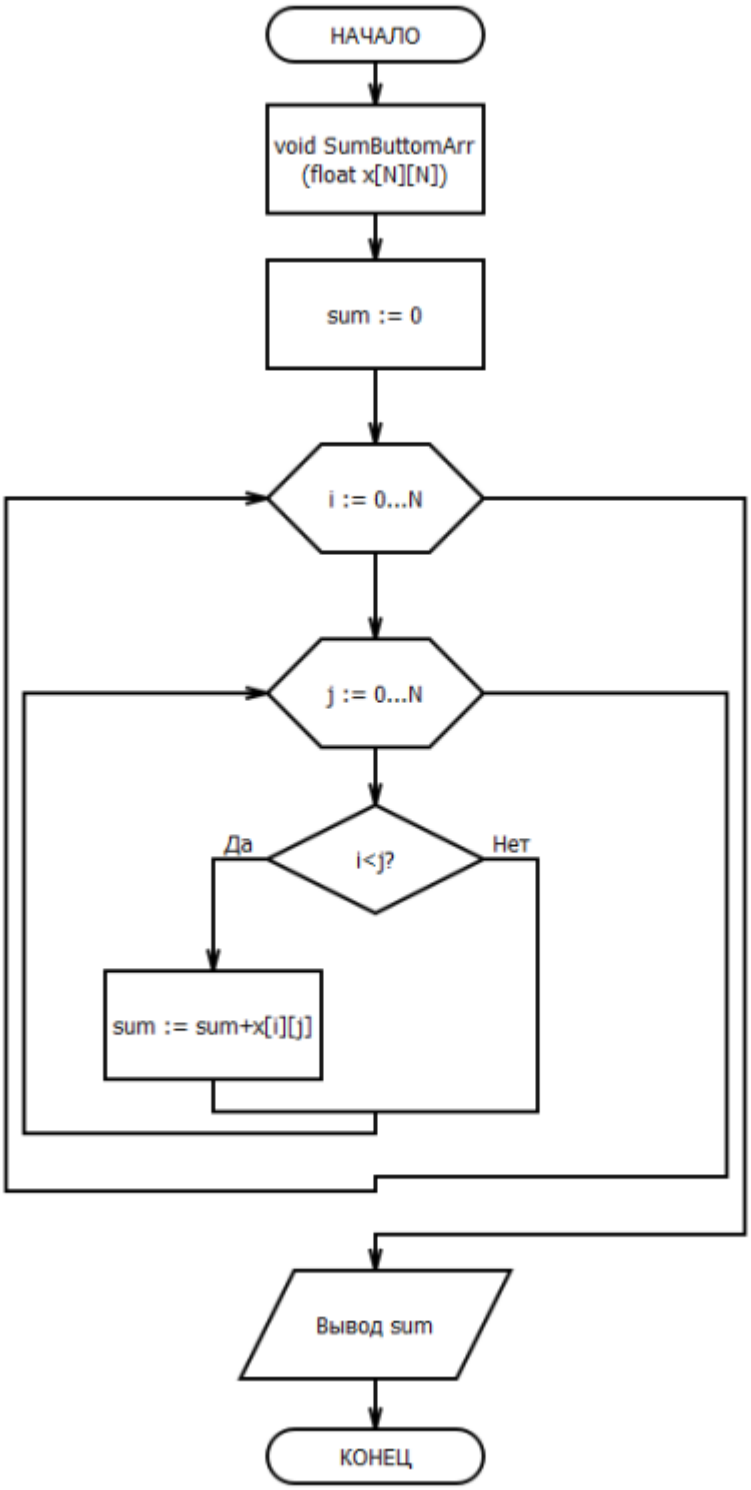

Рисунок 19 – Блок-схема для метода вычисления суммы нижнетреугольных частей матрицы

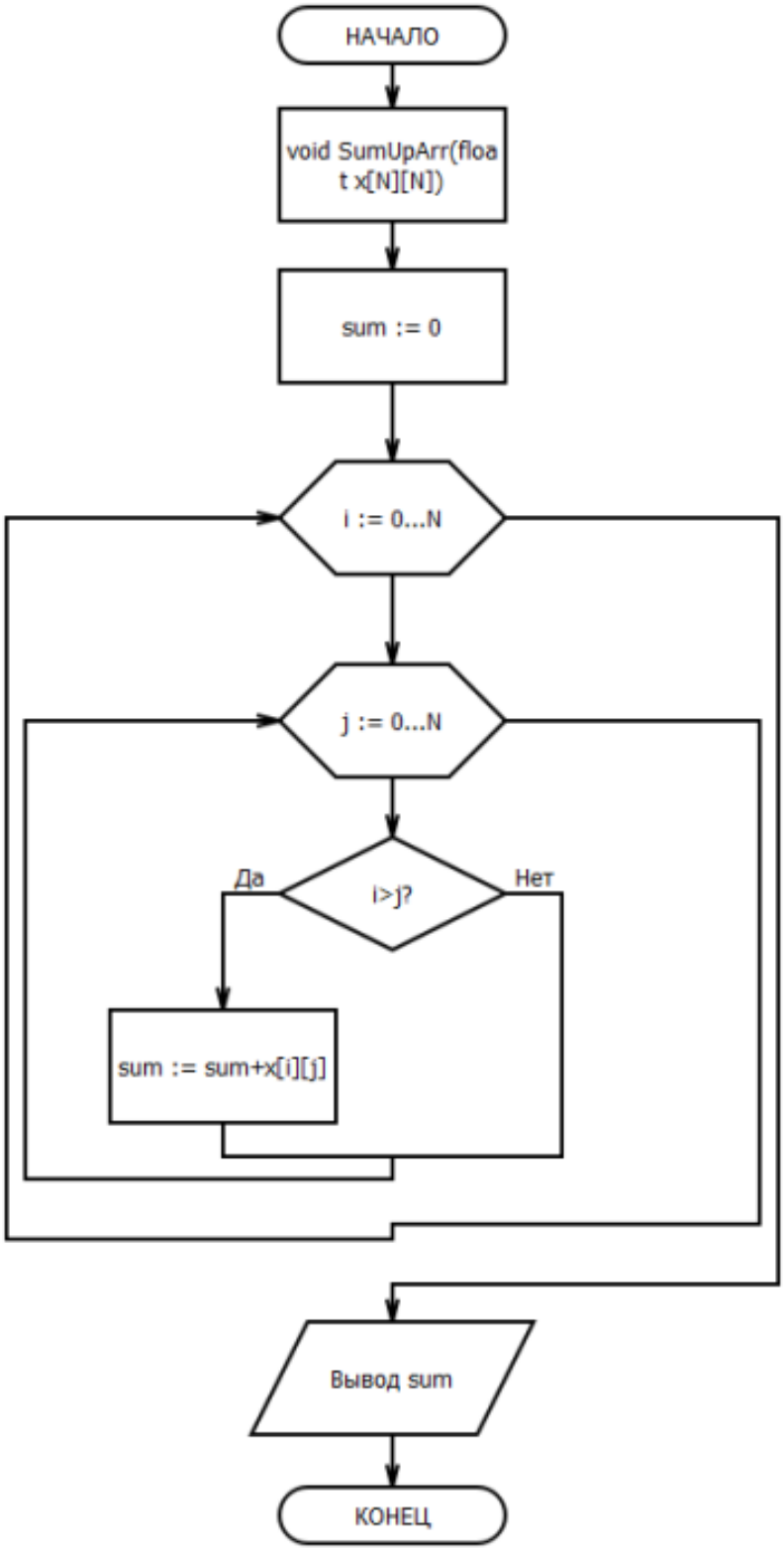

Рисунок 20 – Блок-схема для метода вычисления суммы верхнетреугольных частей матрицы

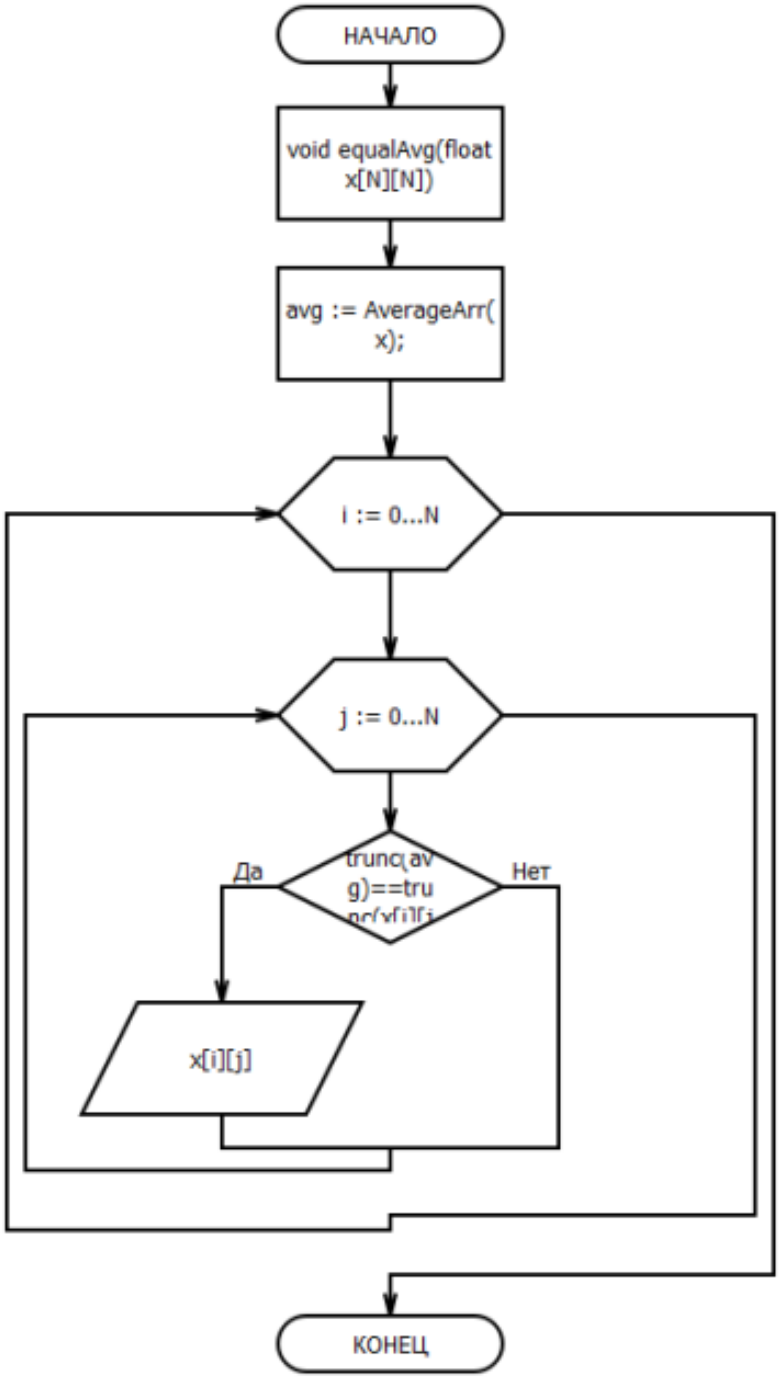

Рисунок 21 – Блок-схема для метода вычисления элемента, наиболее близкому к среднему арифметическому матрицы

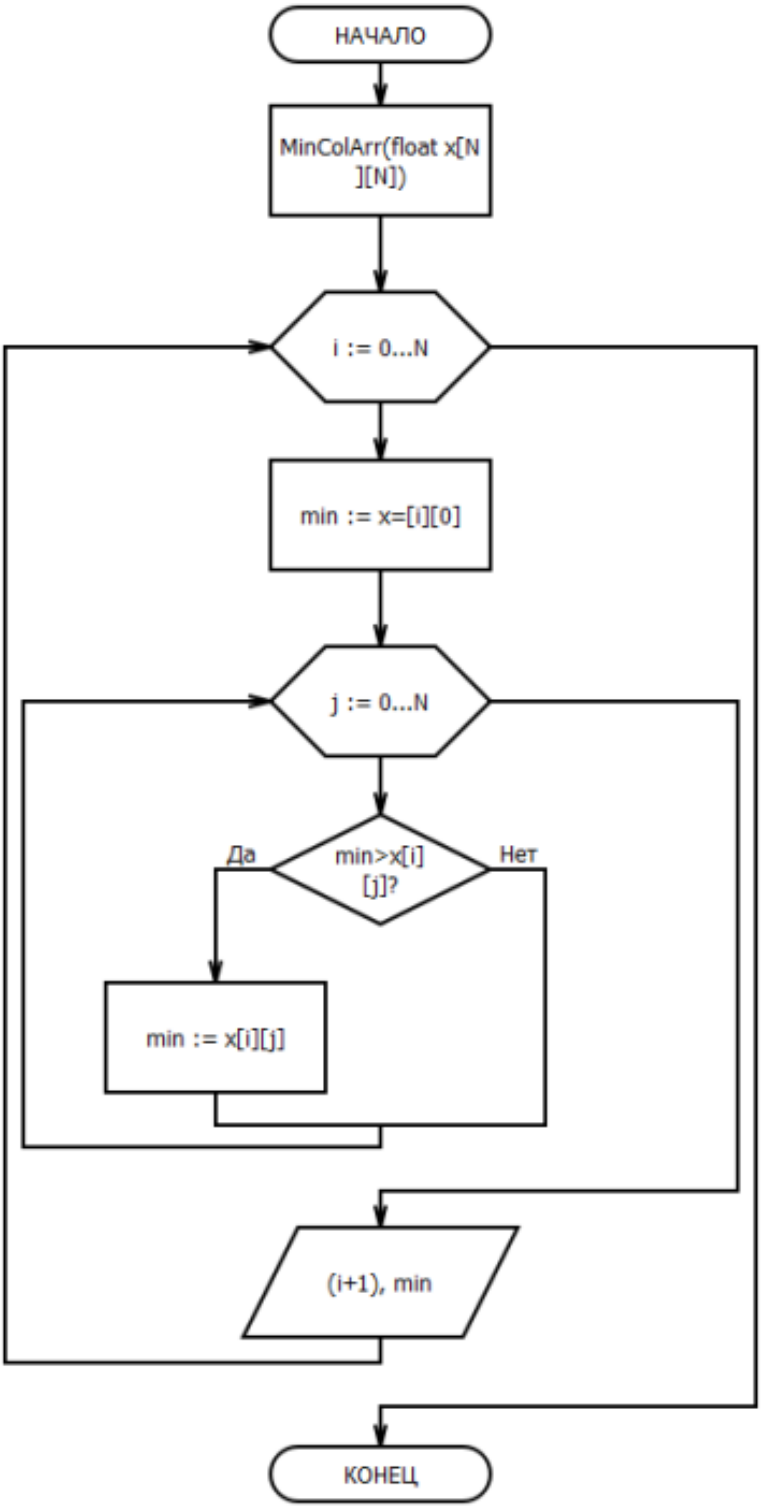

Рисунок 22 – Блок-схема для метода вычисления минимального значения столбцов матрицы

3 Выводы по лабораторной работе

Массив — это область памяти, где могут последовательно храниться несколько значений.

Массив создается почти так же, как и обычная переменная. Количество элементов массива задается при его объявлении заключается в квадратные скобки.

Массивы в памяти хранятся таким же образом, как переменная. Массив типа int из 10 элементов описывается с помощью адреса его первого элемента и количества байт, которое может вместить этот массив. Если для хранения одного целого числа выделяется 4 байта, то для массива из десяти целых чисел будет выделено 40 байт.