Министерство образования Тульской области

Государственное профессиональное образовательное учреждение
Тульской области
«Донской колледж информационных технологий»

РАЗРАБОТКА ОКОННОГО ПРИЛОЖЕНИЯ ДЛЯ РЕШЕНИЯ ТРАНСПОРТНОЙ ЗАДАЧИ

Курсовая работа

МДК. 01.02 Прикладное программирование

Студент группы 3-П-1 Е.Ю. Гавришин

Руководитель О.А. Сергеева

г. Донской
2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данное приложение предназначено для решения задачи оптимизации линейного программирования о поиске оптимального распределения однородных объектов из аккумулятора к приемникам с минимизацией затрат на перемещение. Для простоты понимания рассматривается как задача об оптимальном плане перевозок грузов из пунктов отправления в пункты потребления, с минимальными затратами на перевозки. Транспортная задача по теории сложности вычислений входит в класс сложности P. Когда суммарный объём предложений (грузов, имеющихся в пунктах отправления) не равен общему объёму спроса на товары (грузы), запрашиваемые пунктами потребления, транспортная задача называется несбалансированной (открытой).

Для классической транспортной задачи выделяют два типа задач: критерий стоимости (достижение минимума затрат на перевозку) или расстояний и критерий времени (затрачивается минимум времени на перевозку). Под названием транспортная задача, определяется широкий круг задач с единой математической моделью, эти задачи относятся к задачам линейного программирования и могут быть решены оптимальным методом. Однако, специальный метод решения транспортной задачи позволяет существенно упростить её решение, поскольку транспортная задача разрабатывалась для минимизации стоимости перевозок.

В основу логики приложения был положен алгоритм итерационного улучшения плана перевозок. Опорный план строится методом северо-западного угла. На каждом этапе максимально возможным числом заполняют левую верхнюю клетку оставшейся части таблицы. После нахождения опорного плана перевозок, применяется алгоритм его улучшения – метод потенциалов.

Так как доля языка программирования C# (произносится си шарп) в качестве языка для разработки прикладных программ в настоящее время

растёт с большой скоростью, приложение разработано на этом языке программирования. Язык был разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML. Переняв многое от своих предшественников — языков C++, Pascal, Модула, Smalltalk и, в особенности, Java — C#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов).

1. Задача:

«Имеется m пунктов отправления («поставщиков») и n пунктов потребления («потребителей») некоторого однородного товара. Разработать приложение, позволяющее найти план перевозок, при котором бы полностью удовлетворялся спрос всех потребителей, при этом хватало бы запасов поставщиков и суммарные транспортные расходы были бы минимальными.»

2. Постановка задачи:

Таблица 1 – Входная информация

Наименование	Идентификатор	Тип	
		Наименование	Служебное слово
Кол-во «пунктов отправления»	n	Целый	int
Кол-во «пунктов потребления»	m	Целый	int
Массив с количествами грузов в пунктах отправления	A	Вещественный	float[]
Массив с потребностями грузов в пунктах потребления	B	Вещественный	float[]
Массив коэффициентов (стоимостей)	c	Вещественный	float[,]

Таблица 2 – Выходная информация

Наименование	Идентификатор	Тип	
		Наименование	Служебное слово
Массив с распределенными «грузами»	X	Вещественный	float[,]?
Значение целевой функции (итоговая стоимость)	cost	Вещественный	float

3. Математическая формализация

Шаг 1. Определить модель задачи. Она является *открытой* при условии

$$\sum_{i=1}^{m} a_i \neq \sum_{k=1}^{n} b_k \quad (1)$$

и *закрытой,* если

$$\sum_{i=1}^{m} a_i = \sum_{k=1}^{n} b_k \quad (2)$$

Шаг 2. Построить исходный план перевозок по правилу северо-западного угла.

Шаг 3. Для текущего плана перевозок найти потенциалы путём решения системы линейных уравнений вида:

$$v_i + u_k = c_{ik} \quad (3)$$

и матрицу оценок, вычисляя её элементы по формуле вида:

$$d_{ik} = c_{ik} - (u_k + v_i). \quad (4)$$

Если матрица оценок не содержит отрицательных элементов, то получено оптимальное решение задачи. Записать оптимальный план перевозок и соответствующее ему значение функции цели (суммарные затраты). В противном случае выбрать для рассмотрения свободную клетку, которой соответствует наименьшая оценка.

Шаг 4. Для выбранной клетки построить цикл пересчета и найти величину λ груза, перераспределяемого по клеткам цикла.

Шаг 5. Построить новый план перевозок: добавить величину λ в положительных клетках цикла. Перейти к шагу 3.

При решении транспортной задачи, с достаточно большой вероятностью, можно получить *вырожденный план* перевозок.

Вырожденность в транспортной задаче — ситуация, когда в процессе решения транспортной задачи число базисных (занятых перевозками) ячеек транспортной таблицы меньше m + n - 1 (где m и n — число поставщиков и потребителей), и алгоритм решения впадает в бесконечный цикл из-за невозможности вычислить потенциалы или завершается с ошибкой.

Для решения этой проблемы в литературе предлагается несколько методов. Наиболее подходящим и удобным методом при создании приложения я посчитал *метод случайного выбора свободной ячейки* для включения в базис, который упомянут в учебнике Дж. Данцига [3: 312]. Данный метод не требует составления крайне сложного алгоритма для перемещения базисного нуля и выдает результат с вероятностью единица. Вычисление потенциалов, при использовании такого метода, может дать сбой (впасть в бесконечный цикл), и в этом случае случайный выбор следует повторить.

4. Контрольные примеры

Таблица 3 – Контрольный пример 1. Условие задачи

	85	62	90	60
120	7	4	15	9
80	11	2	7	3
100	4	5	12	8

Таблица 4 – Контрольный пример 1. Результат

	85	62	90	60	3
120		62		55	3
80			80		
100	85		10	5	

Целевая функция: Z = 1803

Таблица 5 – Контрольный пример 2. Условие задачи

	90	120	110	130
105	12	9	7	11
165	4	3	12	2
180	5	17	9	4

Таблица 6 – Контрольный пример 2. Результат

	90	120	110	130
105			105	
165		120		45
180	90		5	85

Целевая функция: Z = 2020

5. Блок-схема

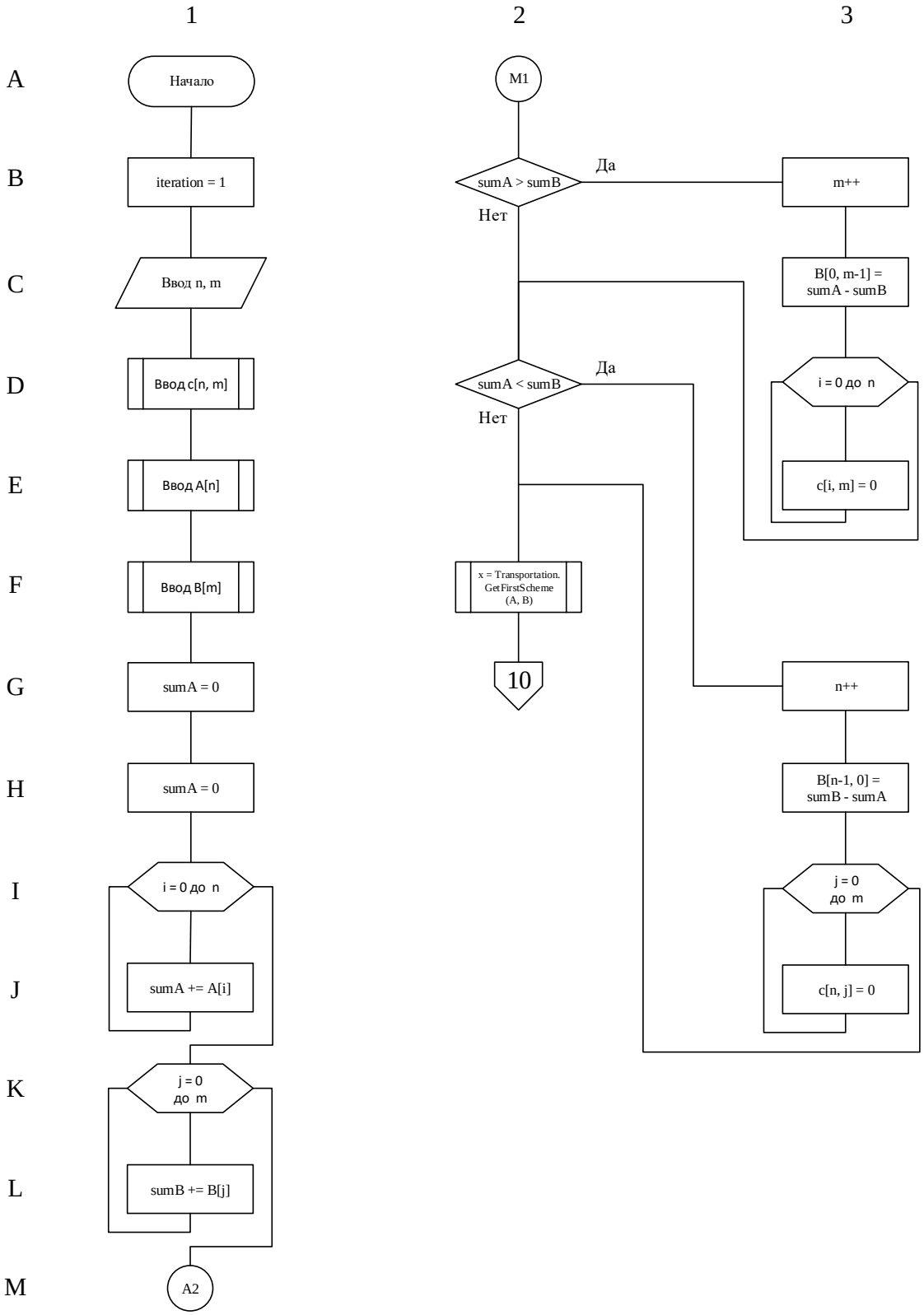

Рисунок 1. Блок-схема главной программы

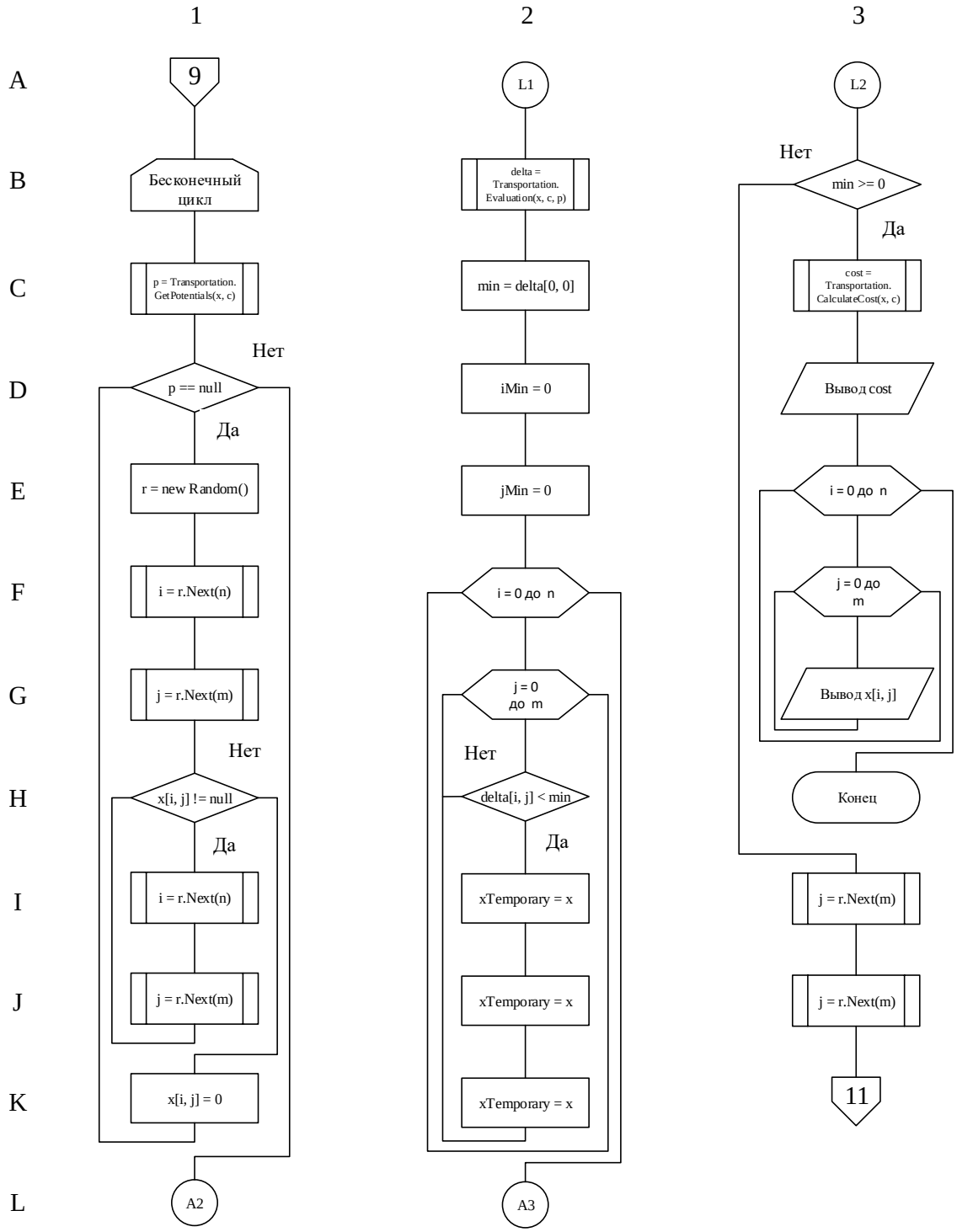

Рисунок 2. Блок-схема главной программы (продолжение)

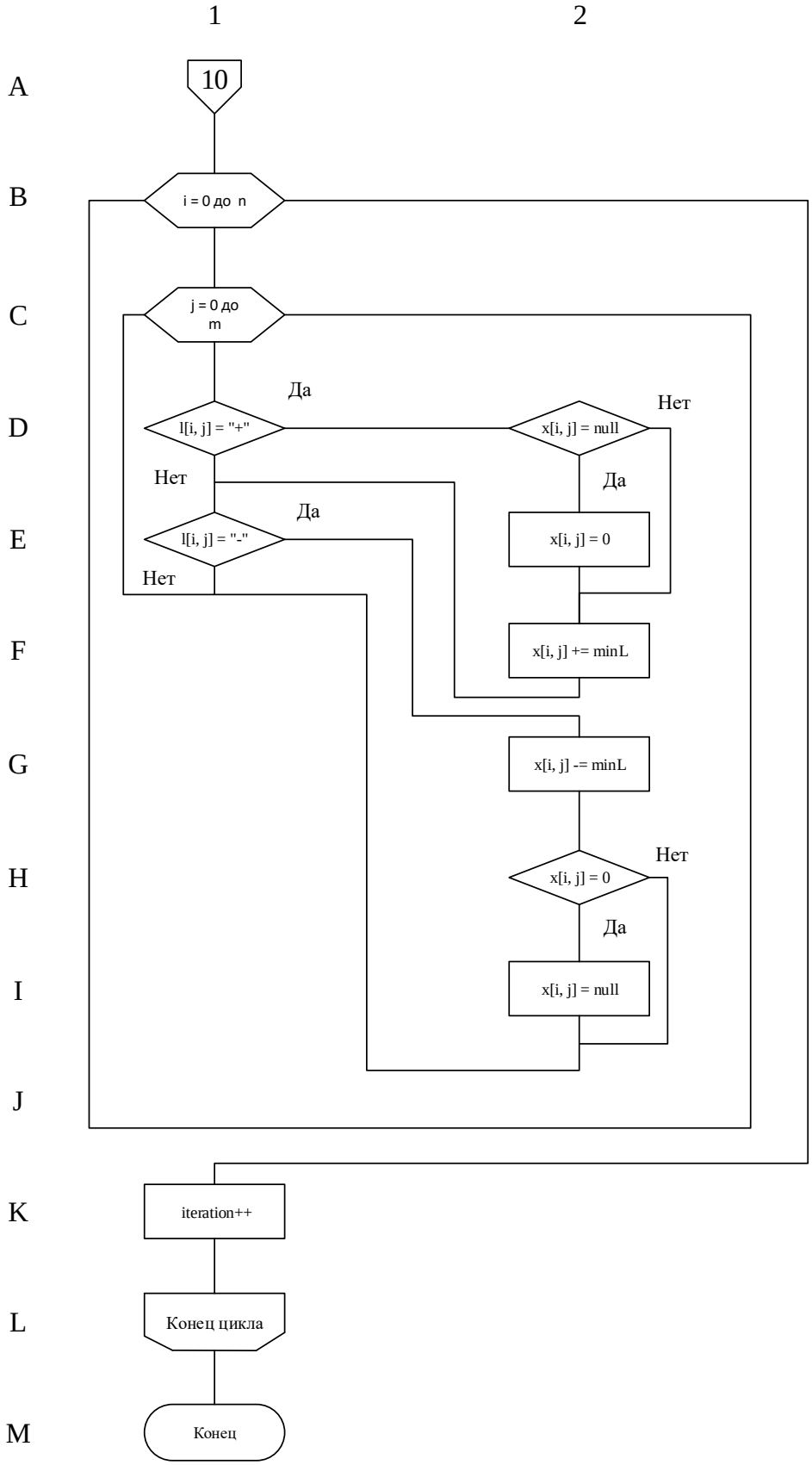

Рисунок 3. Блок-схема главной программы (продолжение)

Функция GetFirstScheme.

Входные параметры: double[] A, double[] B

Выходные параметры: double?[,] x

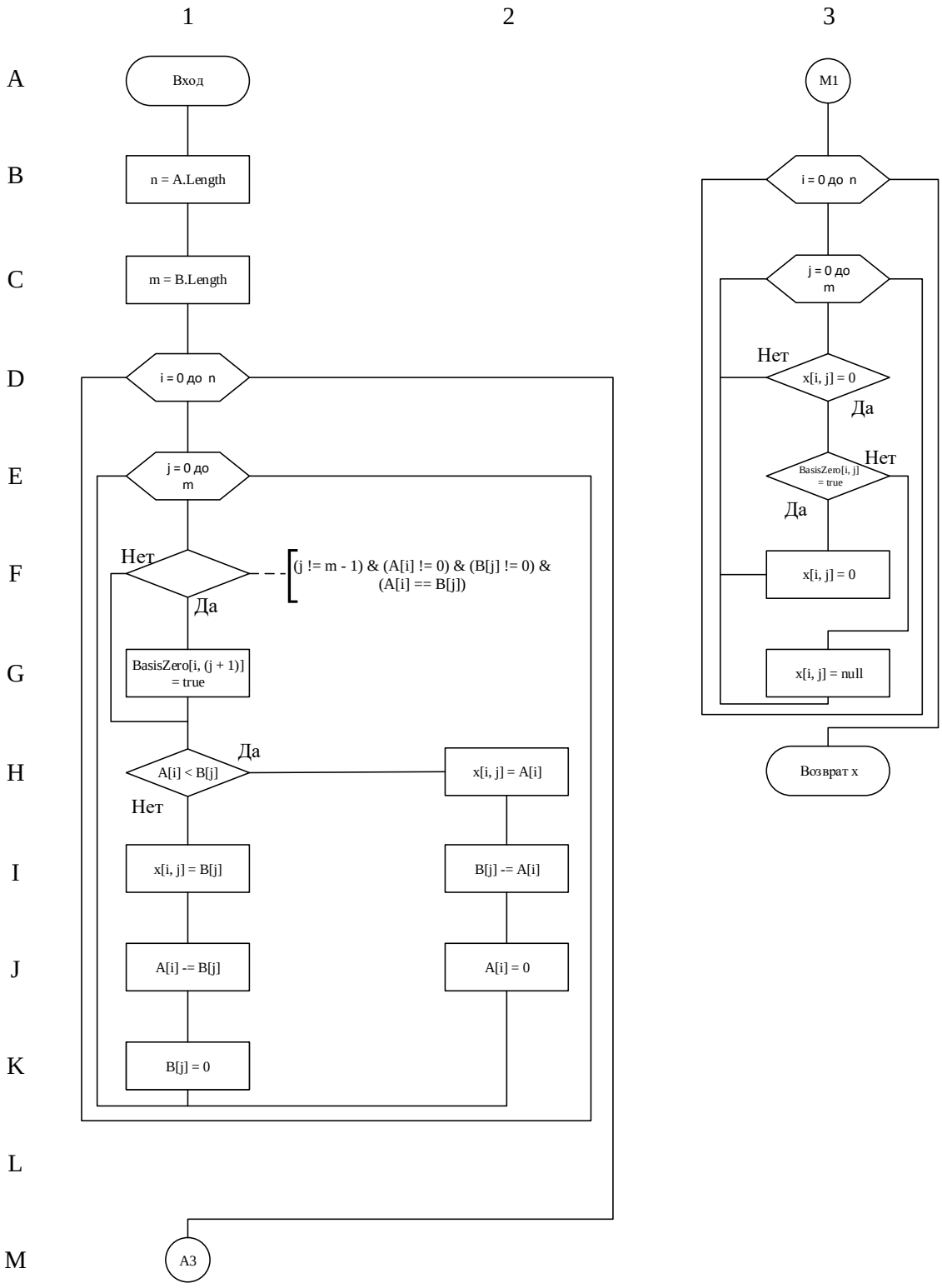

Рисунок 4. Блок-схема функции GetFirstScheme

Функция CalculateCost.

Входные параметры: double?[,] x, double[,] c

Выходные параметры: double z

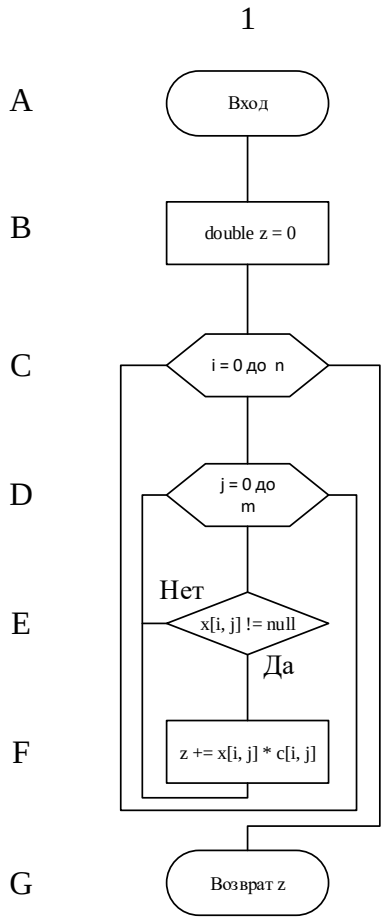

Рисунок 5. Блок-схема функции CalculateCost

Функция GetPotentials.

Входные параметры: double?[,] x, double[,] c

Выходные параметры: double?[][] P

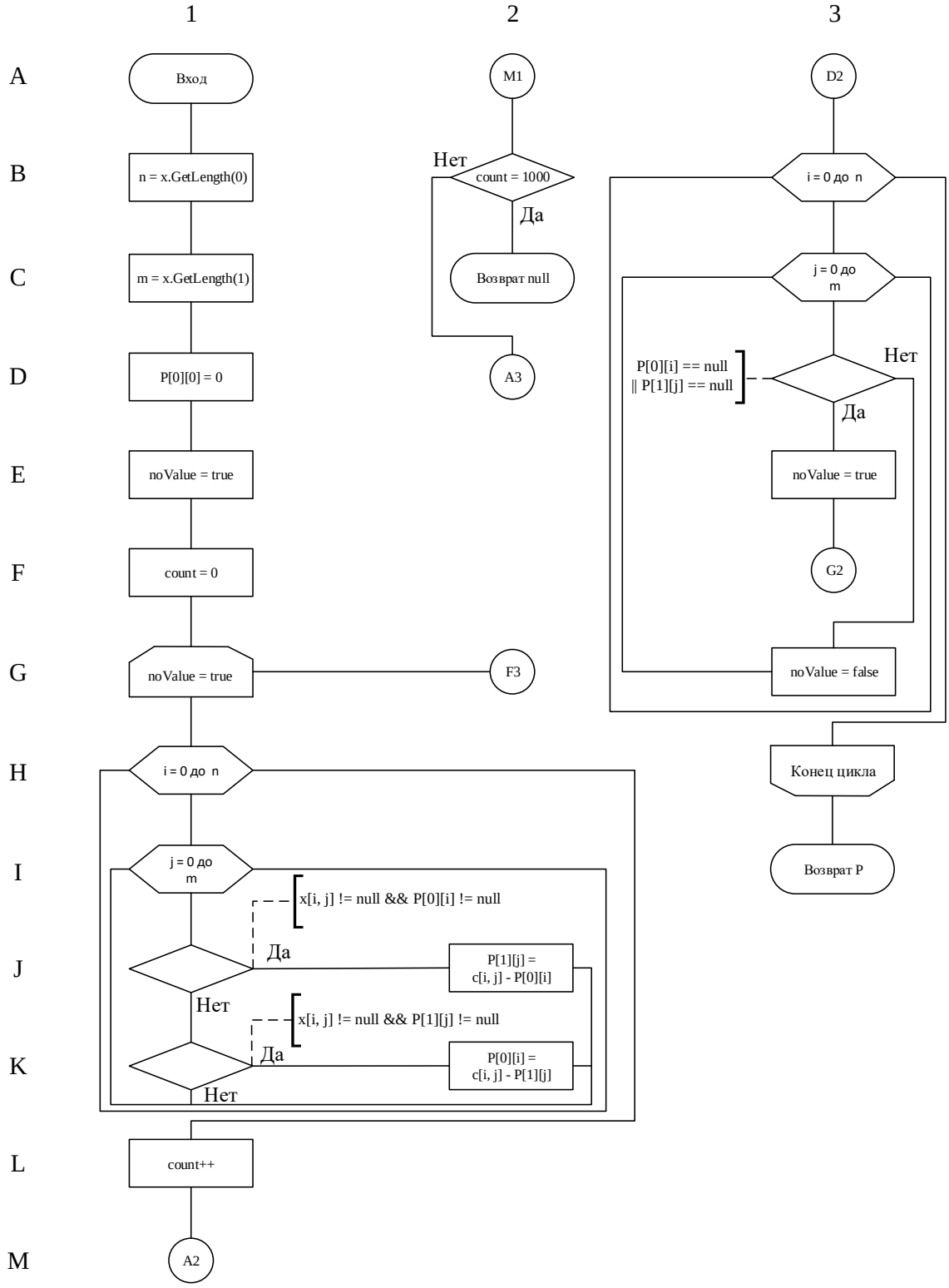

Рисунок 6. Блок-схема функции GetPotentials

Функция Evaluation.

Входные параметры: double?[,] x, double[,] c, double?[][] p

Выходные параметры: double?[,] d

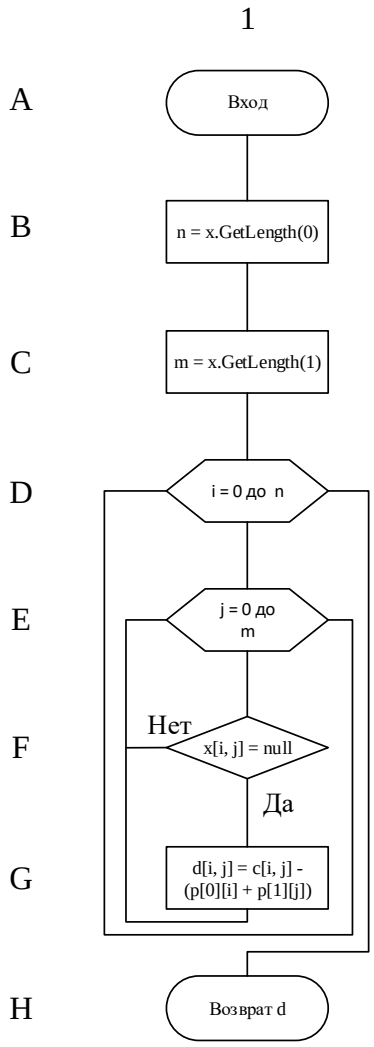

Рисунок 7. Блок-схема функции Evaluation

Функция CharForPutInColumn.

Входные параметры: int j, int n, string[,] sum

Выходные параметры: string

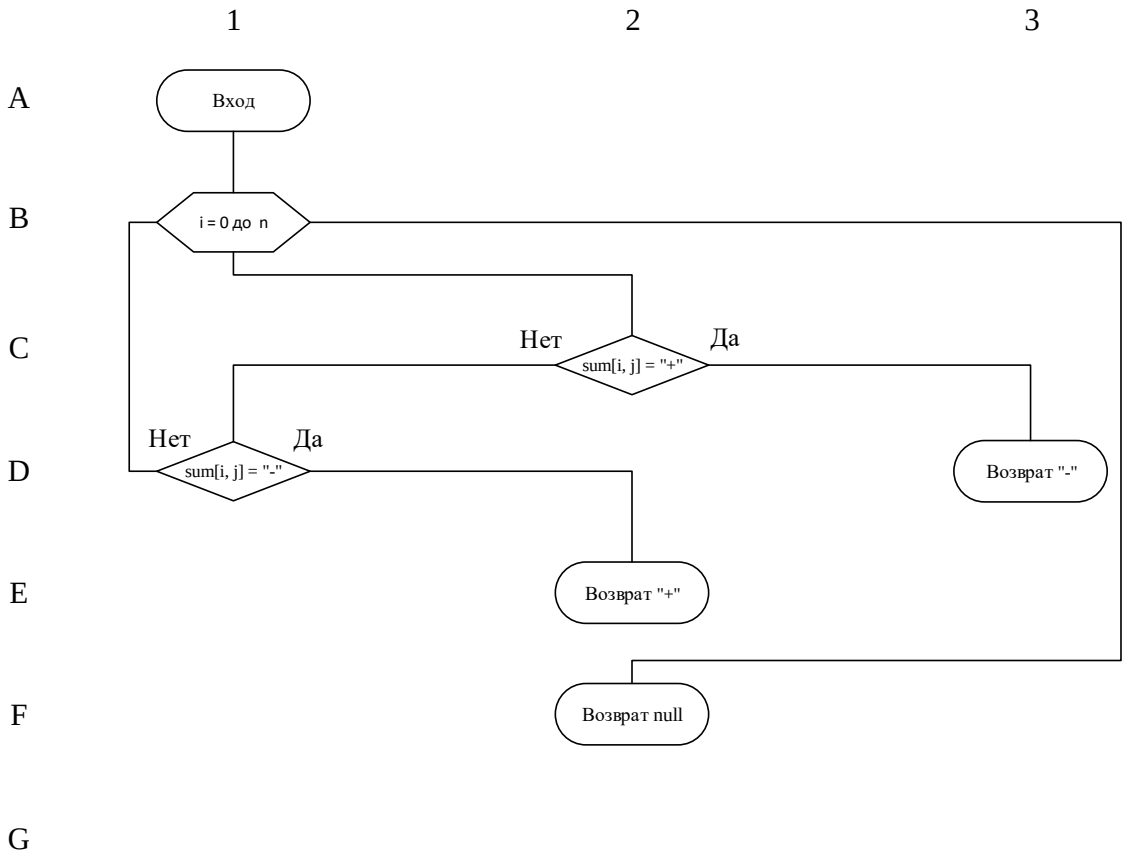

Рисунок 8. Блок-схема функции CharForPutInColumn

Функция CharForPutInRow.

Входные параметры: int i, int m, string[,] sum

Выходные параметры: string

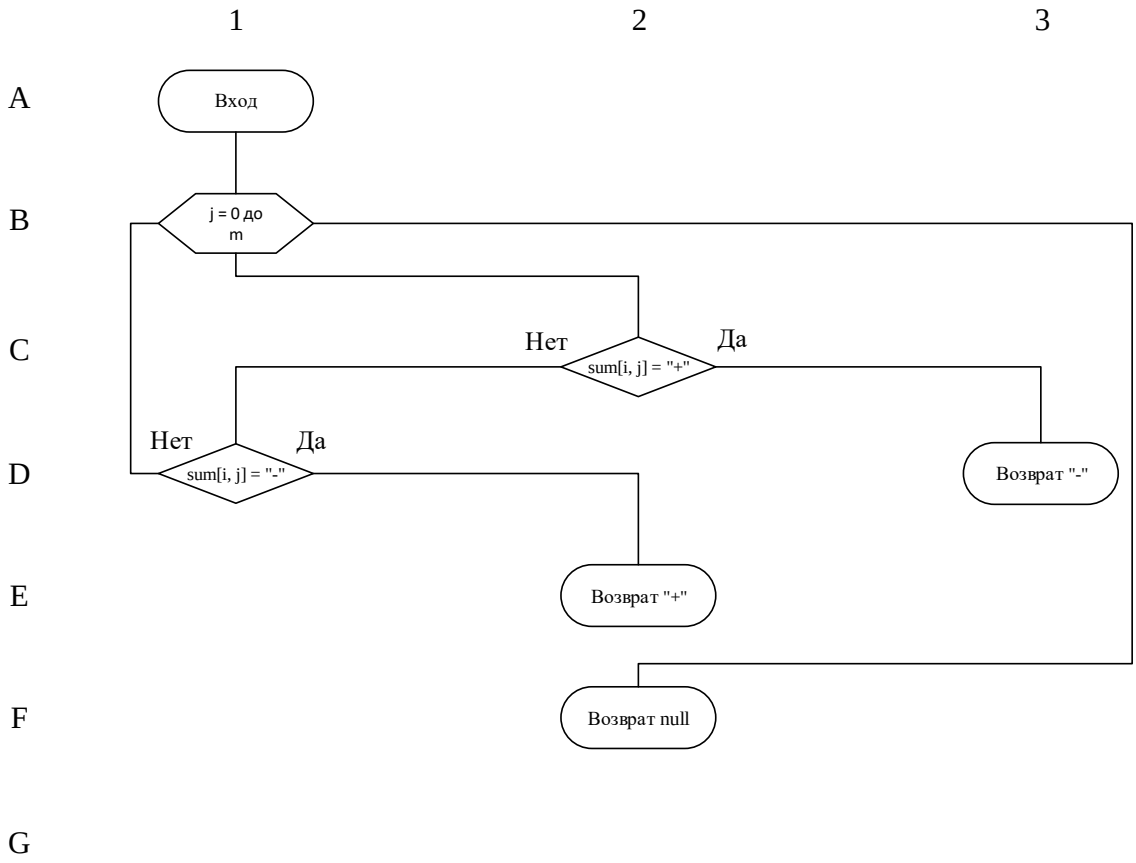

Рисунок 9. Блок-схема функции CharForPutInRow

Функция GetLambda.

Входные параметры: double?[,] x, int iMin, int jMin

Выходные параметры: string[,] symbol

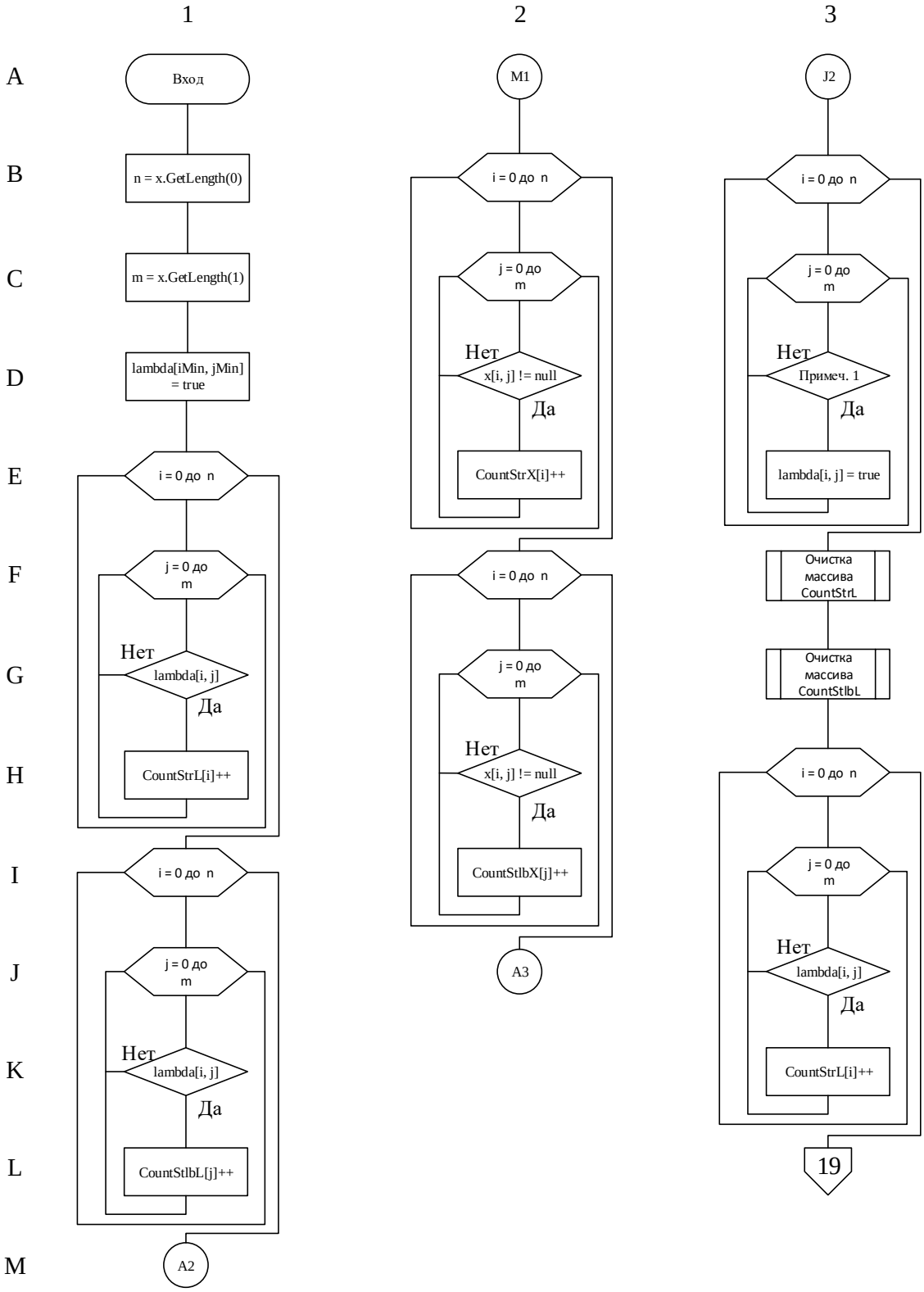

Рисунок 10. Блок-схема функции GetLambda

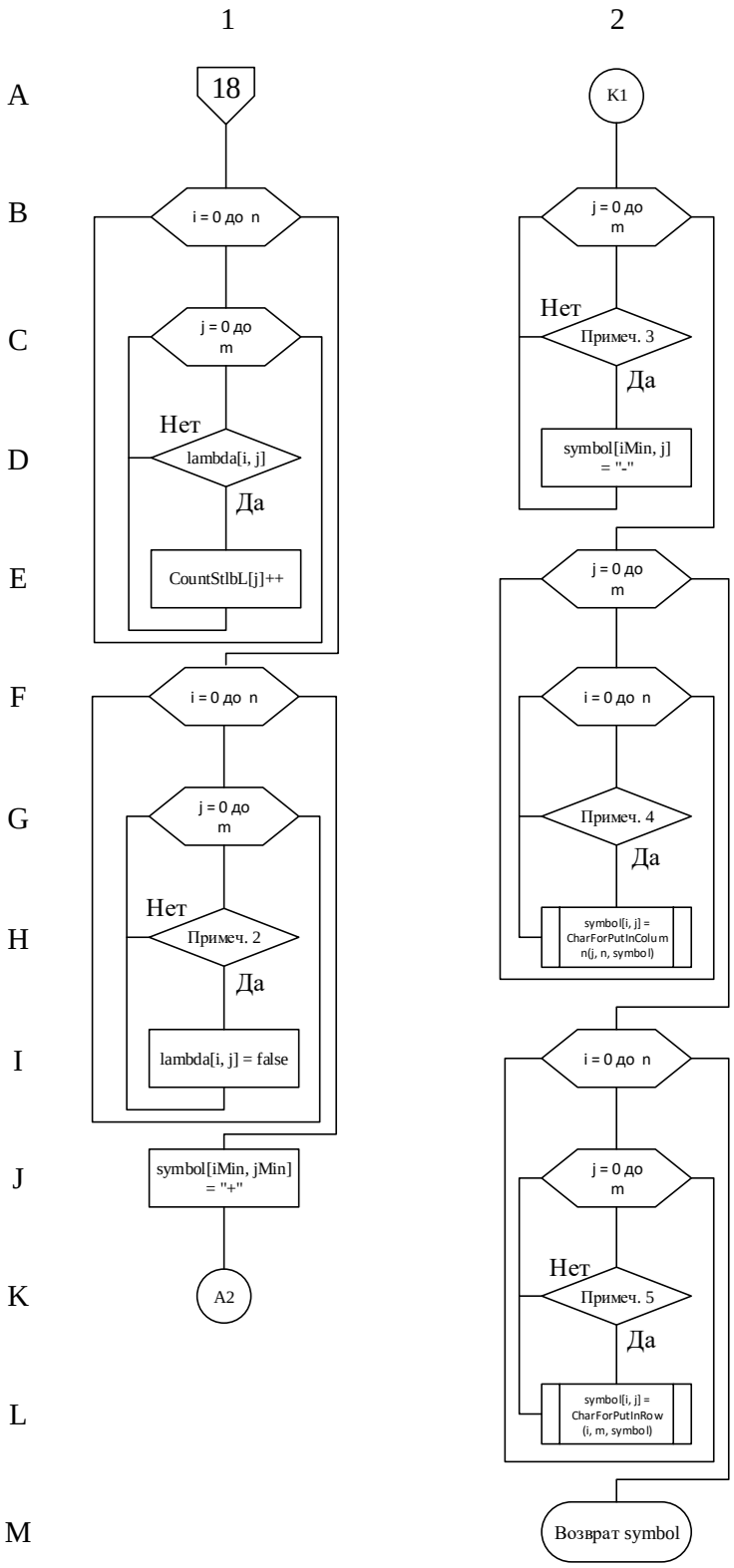

Рисунок 11. Блок-схема функции GetLambda (продолжение)

Функция MinOfLambda.

Входные параметры: string[,] MatrixOfLambda, double?[,] Units

Выходные параметры: double? min

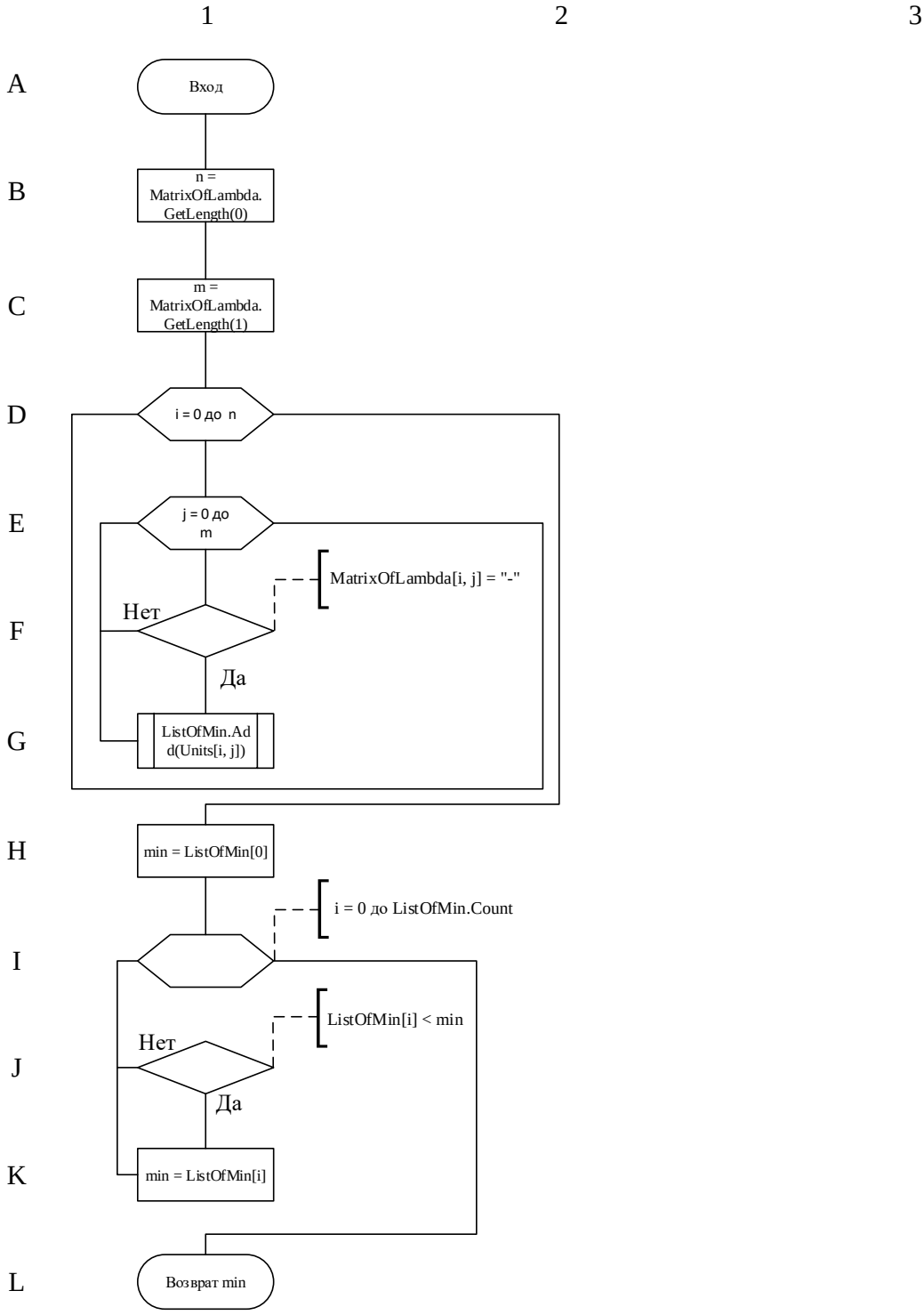

Рисунок 12. Блок-схема функции MinOfLambda

6. Примечания к символам блок-схем

1. x[i, j] != null && (CountStlbX[j] != 1 || CountStlbL[j] == 1) && (CountStrX[i] != 1 || CountStrL[i] == 1) && lambda[i, j] == false
2. (lambda[i, j]) && (CountStrL[i] != 2) && (CountStlbL[j] != 2)
3. lambda[iMin, j] && string.IsNullOrEmpty(symbol[iMin, j])
4. lambda[i, j] && string.IsNullOrEmpty(symbol[i, j])
5. lambda[i, j] && string.IsNullOrEmpty(symbol[i, j])

7. Листинг программы

Главная программа:

```
potencialBox.Items.Clear();
evaluationBox.Items.Clear();
#region Переменные
int iteration = 1;
double min;
int iMin;
int jMin;
int
   n = int.Parse(nA.Text),
   m = int.Parse(mB.Text);
double sumA = 0;
for (int i = 0; i < n; i++)
   sumA += Convert.ToDouble(TableA.Rows[i].Cells[0].Value);
double sumB = 0;
for (int j = 0; j < m; j++)
   sumB += Convert.ToDouble(TableB.Rows[0].Cells[j].Value);
if (sumA > sumB)
{
   m++;
   TableB.ColumnCount++;
```

```
                Price.ColumnCount++;
                Table.ColumnCount++;
                TableBforPrice.ColumnCount++;
                TableB.Rows[0].Cells[m - 1].Value = sumA - sumB;
                TableB.Rows[0].Cells[m-1].Style.BackColor=
TableBforPrice.Rows[0].Cells[m-1].Style.BackColor=
System.Drawing.Color.Plum;
                for (int i = 0; i < n; i++)
                    Price.Rows[i].Cells[Price.ColumnCount - 1].Value = 0;
            }
            else if (sumA < sumB)
            {
                n++;
                TableA.RowCount++;
                Price.RowCount++;
                Table.RowCount++;
                TableA.Rows[n - 1].Cells[0].Value = sumB - sumA;
                TableA.Rows[n-1].Cells[0].Style.BackColor=
System.Drawing.Color.Plum;
                for (int j = 0; j < m; j++)
                    Price.Rows[Price.RowCount - 1].Cells[j].Value = 0;
            }
            var c = new double[n, m];
```

```
for (int i = 0; i < n; i++)
    for (int j = 0; j < m; j++)
        c[i, j] = Convert.ToDouble(Price.Rows[i].Cells[j].Value);
var A = new double[n];
for (int i = 0; i < n; i++)
    A[i] = Convert.ToDouble(TableA.Rows[i].Cells[0].Value);
var B = new double[m];
for (int j = 0; j < m; j++)
    B[j] = Convert.ToDouble(TableB.Rows[0].Cells[j].Value);
#endregion
var x = Transportation.GetFirstScheme(A, B);
while (true)
{
    var p = Transportation.GetPotentials(x, c);
    var xTemporary = x;
    while (p == null)
    {
        Random r = new Random();
        int i = r.Next(n);
        int j = r.Next(m);
        while (xTemporary[i, j] != null)
        {
```

```
                i = r.Next(n);
                j = r.Next(m);
            }
            xTemporary[i, j] = 0;
            p = Transportation.GetPotentials(xTemporary, c);
            if (p == null)
                xTemporary = x;
            else
                x = xTemporary;
        }
        var delta = Transportation.Evaluation(x, c, p);
        potencialBox.Items.Add(string.Format("{0}ИТЕРАЦИЯ",
iteration));
        for (int i = 0; i < n; i++)
            potencialBox.Items.Add(string.Format("U[{0}] = {1}", i + 1,
p[0][i]));
        for (int j = 0; j < m; j++)
            potencialBox.Items.Add(string.Format("V[{0}] = {1}", j + 1,
p[1][j]));
          }
        evaluationBox.Items.Add(string.Format("{0}ИТЕРАЦИЯ",
iteration));
        for (int i = 0; i < n; i++)
```

```
                for (int j = 0; j < m; j++)
                    if (delta[i, j] != null)
                        evaluationBox.Items.Add(string.Format("D[{0},{1}]       =
{2}", i + 1, j + 1, delta[i, j]));
                min = Convert.ToInt32(delta[0, 0]);
                iMin = 0; jMin = 0;
                for (int i = 0; i < n; i++)
                    for (int j = 0; j < m; j++)
                        if (delta[i, j] < min)
                        {
                            min = Convert.ToInt32(delta[i, j]);
                            iMin = i;
                            jMin = j;
                        }
                if (min >= 0)
                {
                    cost.Text = Transportation.CalculateCost(x, c).ToString();
                    for (int i = 0; i < n; i++)
                        for (int j = 0; j < m; j++)
                            Table.Rows[i].Cells[j].Value = Convert.ToString(x[i, j]);
                    return;
                }
                var l = Transportation.GetLambda(x, iMin, jMin);
```

```
                var minL = Transportation.MinOfLambda(l, x);
                for (int i = 0; i < n; i++)
                    for (int j = 0; j < m; j++)
                        if (l[i, j] == "+")
                        {
                            if (x[i, j] == null)
                                x[i, j] = 0;
                            x[i, j] += minL;
                        }
                        else if (l[i, j] == "-")
                        {
                            x[i, j] -= minL;
                            if (x[i, j] == 0)
                                x[i, j] = null;
                        }
                iteration++;
                if (iteration == 200)
                {
                    MessageBox.Show(“Ошибка. Проверьте правильность условий”); return;
                }
            }
        }
```

Библиотека функций:

```
using System.Collections.Generic;
namespace Optimizer
{
    public class Transportation
    {
        public static double?[,] GetFirstScheme(double[] A, double[] B)
        {
            int n = A.Length, m = B.Length;
            double?[,] x = new double?[n, m];
            bool[,] BasisZero = new bool[n, m];
            for (int i = 0; i < n; i++)
                for (int j = 0; j < m; j++)
                {
                    if ((j != m - 1) && (A[i] != 0) && (B[j] != 0) && (A[i] == B[j]))
BasisZero[i, (j + 1)] = true;
                    if (A[i] < B[j])
                    {
                        x[i, j] = A[i];
                        B[j] -= A[i];
                        A[i] = 0;
                    }
                    else
```

```
            {
                x[i, j] = B[j];
                A[i] -= B[j];
                B[j] = 0;
            }
        }
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            if (x[i, j] == 0)
                if (BasisZero[i, j] == true)
                    x[i, j] = 0;
                else
                    x[i, j] = null;
    return x;
}
public static double CalculateCost(double?[,] x, double[,] c)
{
    double z = 0;
    for (int i = 0; i < x.GetLength(0); i++)
        for (int j = 0; j < x.GetLength(1); j++)
            if (x[i, j] != null) z += (double)x[i, j] * c[i, j];
    return z;
```

```
}
public static double?[][] GetPotentials(double?[,] x, double[,] c)
{
    int n = x.GetLength(0), m = x.GetLength(1);
    var P = new double?[2][] { new double?[n] , new double?[m] };
    P[0][0] = 0;
    bool noValue = true; int count = 0;
    startWhile: while (noValue)
    {
        for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)
            {
                if (x[i, j] != null && P[0][i] != null)
                    P[1][j] = c[i, j] - P[0][i];
                if (x[i, j] != null && P[1][j] != null)
                    P[0][i] = c[i, j] - P[1][j];
            }
        count++;
        if (count == 1000)
            return null;
        for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)
```

```
                if (P[0][i] == null || P[1][j] == null)
                {
                    noValue = true;
                    goto startWhile;
                }
                else noValue = false;
            }
            return P;
        }
        public static double?[,] Evaluation(double?[,] x, double[,] c, double?[][]
p)
        {
            int n = x.GetLength(0), m = x.GetLength(1);
            var d = new double?[n, m];
            for (int i = 0; i < n; i++)
                for (int j = 0; j < m; j++)
                    if (x[i, j] == null)
                        d[i, j] = System.Convert.ToInt32(c[i, j] - (p[0][i] + p[1][j]));
            return d;
        }
        private static string CharForPutInColumn(int j, int n, string[,] sum)
        {
            for (int i = 0; i < n; i++)
```

```
        if (sum[i, j] == "+")
            return "-";
        else if (sum[i, j] == "-")
            return "+";
    return null;
}
private static string CharForPutInRow(int i, int m, string[,] sum)
{
    for (int j = 0; j < m; j++)
        if (sum[i, j] == "+")
            return "-";
        else if (sum[i, j] == "-")
            return "+";
    return null;
}

public static string[,] GetLambda(double?[,] x, int iMin, int jMin)
{
    /*---------------Put lambda---------------*/
    #region InitializationVariables
    int n = x.GetLength(0), m = x.GetLength(1);
    var lambda = new bool[n, m];
```

```
lambda[iMin, jMin] = true;
var CountStrL = new int[n];
for (int i = 0; i < n; i++)
    for (int j = 0; j < m; j++)
        if (lambda[i, j])
            CountStrL[i]++;
var CountStlbL = new int[m];
for (int i = 0; i < n; i++)
    for (int j = 0; j < m; j++)
        if (lambda[i, j])
            CountStlbL[j]++;
var CountStrX = new int[n];
for (int i = 0; i < n; i++)
    for (int j = 0; j < m; j++)
        if (x[i, j] != null)
            CountStrX[i]++;
var CountStlbX = new int[m];
for (int i = 0; i < n; i++)
    for (int j = 0; j < m; j++)
        if (x[i, j] != null)
            CountStlbX[j]++;
#endregion
```

```
                for (int i = 0; i < n; i++)
                    for (int j = 0; j < m; j++)
                        if (x[i, j] != null && (CountStlbX[j] != 1 || CountStlbL[j] == 1)
&& (CountStrX[i] != 1 || CountStrL[i] == 1) && lambda[i, j] == false)
                            lambda[i, j] = true;
                #region ReCountLamdas
                System.Array.Clear(CountStrL, 0, CountStrL.Length);
                System.Array.Clear(CountStlbL, 0, CountStlbL.Length);
                for (int i = 0; i < n; i++)
                    for (int j = 0; j < m; j++)
                        if (lambda[i, j])
                            CountStrL[i]++;
                for (int i = 0; i < n; i++)
                    for (int j = 0; j < m; j++)
                        if (lambda[i, j])
                            CountStlbL[j]++;
                #endregion
                for (int i = 0; i < n; i++)
                    for (int j = 0; j < m; j++)
                        if ((lambda[i, j]) && (CountStrL[i] != 2) && (CountStlbL[j] !=
2))
                        {
                            lambda[i, j] = false;
```

```
            }
        /*---------------Put symbol---------------*/
        var symbol = new string[n, m]; // + or -
        symbol[iMin, jMin] = "+";
        for (int j = 0; j < m; j++)
            if (lambda[iMin, j] && string.IsNullOrEmpty(symbol[iMin, j]))
                symbol[iMin, j] = "-";
        for (int j = 0; j < m; j++)
            for (int i = 0; i < n; i++)
                if (lambda[i, j] && string.IsNullOrEmpty(symbol[i, j]))
                    symbol[i, j] = CharForPutInColumn(j, n, symbol);
        for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)
                if (lambda[i, j] && string.IsNullOrEmpty(symbol[i, j]))
                    symbol[i, j] = CharForPutInRow(i, m, symbol);
        return symbol;
    }
    public static double? MinOfLambda(string[,] MatrixOfLambda,
double?[,] Units)
    {
        int    n = MatrixOfLambda.GetLength(0),
               m = MatrixOfLambda.GetLength(1);
        var ListOfMin = new List<double?>();
```

```
            for (int i = 0; i < n; i++)
                for (int j = 0; j < m; j++)
                    if (MatrixOfLambda[i, j] == "-")
                        ListOfMin.Add(Units[i, j]);
            var min = ListOfMin[0];
            for (int i = 0; i < ListOfMin.Count; i++)
                if (ListOfMin[i] < min)
                    min = ListOfMin[i];
            return min;
        }
    }
}
```

8. Результаты работы программы

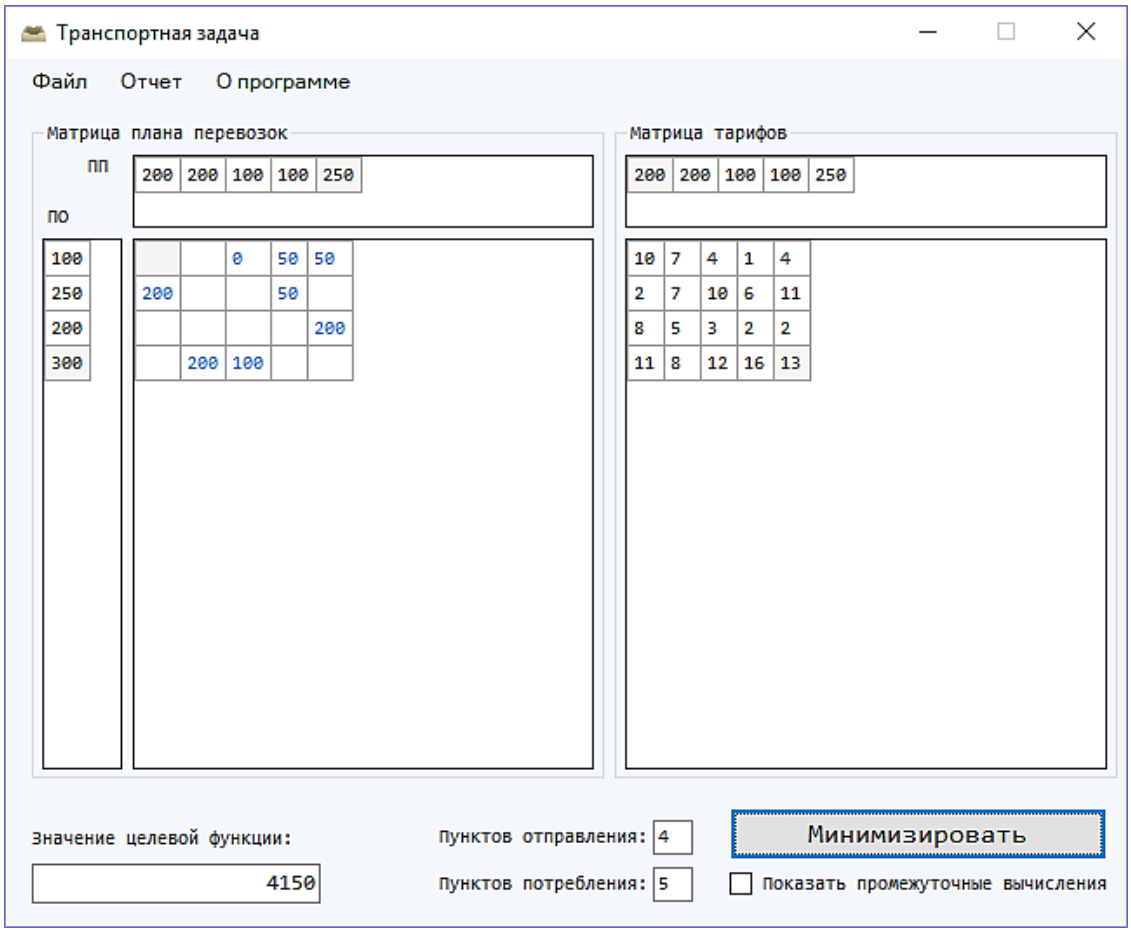

Рисунок 13. Работа приложения

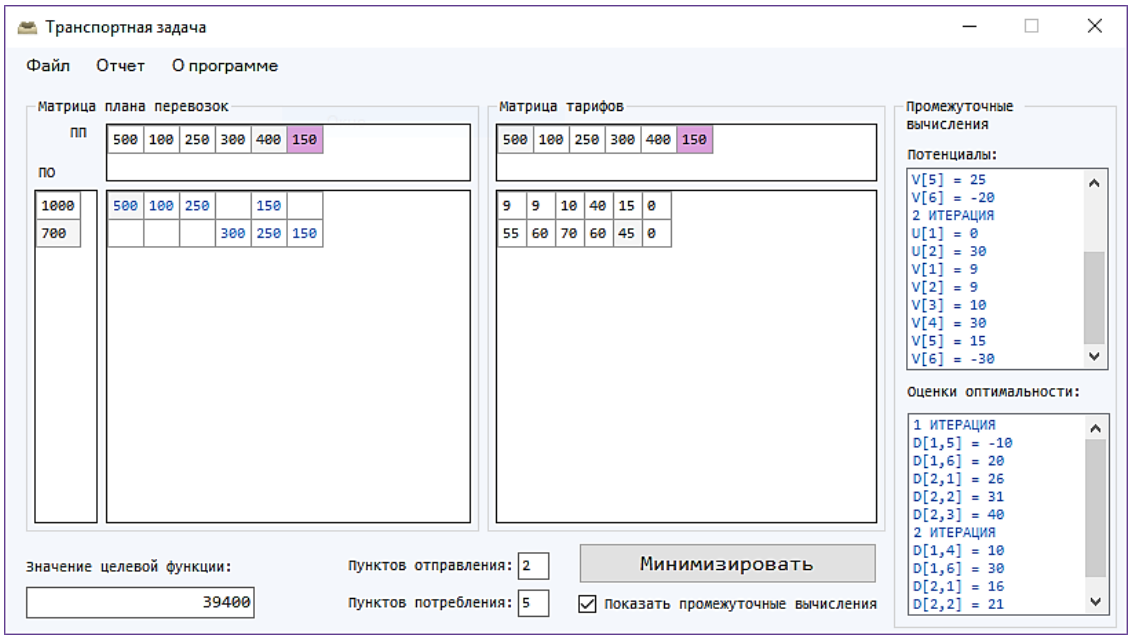

Рисунок 14. Работа приложения

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы я повысил опыт разработки прикладного программного обеспечения в среде разработки Microsoft Visual Studio, узнал больше о функциях и богатейших возможностях, поставляемых платформой .NET Framework, а также научился создавать собственные библиотеки, документировать их, повысил навык разработки алгоритмов, что является самой важной частью при разработке какого-либо прикладного приложения.

Кроме своей прямой функции – решение задачи, приложение предоставляет пользователю дополнительные функции, как вывод отчета о решении в документ Microsoft Office Word, его печать, отображение промежуточных вычислений, к которым придется прибегнуть, решая задачу вручную, а также возможность сохранять и открывать файлы, в которых сохраняются условия задачи.

Все части кода и алгоритмы были написаны и разработаны мной самостоятельно. Поставленная задача выполнена полностью.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Плотников А.Д. Математическое программирование. – 2-е изд., стер. – Минск: Новое знание, 2007. – 171с.
2. Соболь Б.В. Методы оптимизации: практикум. - Ростов н/Д: Феникс, 2009. – 380с.
3. Данциг Дж. Линейное программирование его обобщения и применения. – Москва: Прогресс, 1966. – 590с.
4. Пахомов Б. И. C# для начинающих. – СПб.: БХВ-Петербург, 2014. – 432с.
5. Зиборов В. В. Visual C# на примерах. – СПб.: БХВ-Петербург, 2013. – 480с.
6. Культин Н. Б. Microsoft Visual C# в задачах и примерах. – 2-е изд., исправл. – СПб.: БХВ-Петербург, 2015. – 320с.
7. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения. – Москва: ИПК Издательство стандартов № 2001, 1992. – 24 с.
8. https://ru.wikipedia.org/
9. https://cyclowiki.org/
10. https://msdn.microsoft.com/
11. https://mva.microsoft.com/