ЛЕКЦИЯ № 4. Алгоритмы обработки двумерных массивов.

Цель лекции: *Знакомство с понятием матрицы, как двумерного массива. Приобретение навыков построения алгоритмов предназначенных для обработки матриц.*

7. Алгоритмы обработки матриц

Матрица– это двумерный массив, каждый элемент которого имеет два индекса: номер строки – `i`; номер столбца – `j`. Поэтому для работы с элементами матрицы необходимо использовать два цикла. Если значениями параметра первого цикла будут номера строк матрицы, то значениями параметра второго – столбцы (или наоборот). Обработка матрицы заключается в том, что вначале поочередно рассматриваются элементы первой строки (столбца), затем второй и т.д. до последней. Рассмотрим основные операции, выполняемые над матрицами при решении задач.

7.1. Алгоритм ввода-вывода матриц

Матрицы, как и массивы, нужно вводить (выводить) поэлементно. Блок-схема ввода элементов матрицы изображена на рис. 7.1. Вывод матрицы организуется аналогично вводу.

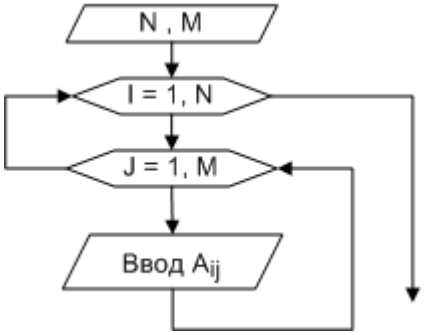

Рис. 7.1. Ввод элементов матрицы

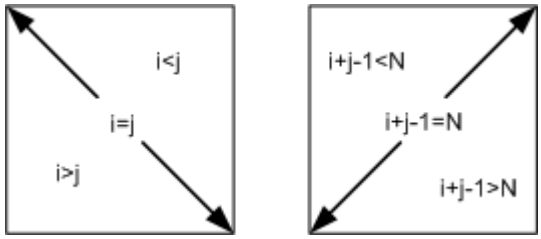

Рис. 7.2. Свойства элементов матрицы

Рассмотрим несколько задач обработки матриц. Для их решения напомним читателю некоторые свойства матриц (рис. 7.2):

- если номер строки элемента совпадает с номером столбца (`i=j`), это означает что элемент лежит на главной диагонали матрицы;
- если номер строки превышает номер столбца (`i>j`), то элемент находится ниже главной диагонали;
- если номер столбца больше номера строки (`i<j`), то элемент находится выше главной диагонали.
- элемент лежит на побочной диагонали, если его индексы удовлетворяют равенству `i+j-1=n`;
- неравенство `i+j-1<n` характерно для элемента находящегося выше побочной диагонали;
- соответственно, элементу лежащему ниже побочной диагонали соответствует выражение `i+j-1>n`.

7.2. Примеры алгоритмов обработки матрицами

ПРИМЕР 7.1. Найти сумму элементов матрицы, лежащих выше главной диагонали (рис 7.3).

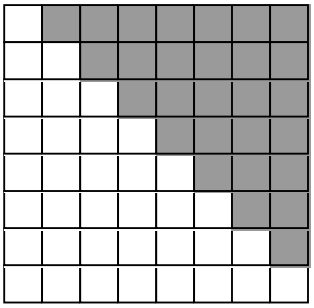

Рис. 7.3. Рисунок к условию задачи из примера 7.1

Алгоритм решения данной задачи (рис. 7.4) построен следующим образом: обнуляется ячейка для накапливания суммы (переменная `S`). Затем с помощью двух циклов (первый по строкам, второй по столбцам) просматривается каждый элемент матрицы, но суммирование происходит только в том случае если, этот элемент находится выше главной диагонали, то есть выполняется свойство `i<j`.

На рисунке 7.5 изображен еще один вариант решения данной задачи. В нем проверка условия *i<j* не выполняется, но, тем не менее, в нем так же суммируются элементы матрицы, находящиеся выше главной диагонали. Для того чтобы понять, как работает этот алгоритм, вернемся к рисунку 7.3. В первой строке заданной матрицы необходимо сложить все элементы, начиная со второго. Во второй – все, начиная с третьего, в `i`−й строке процесс начнется с `(i+1)`−го элемента и так далее. Таким образом, первый цикл работает от 1 до `N`, а второй от `i+1` до `M`. Предлагаем читателю самостоятельно составить программу, соответствующую описанному алгоритму.

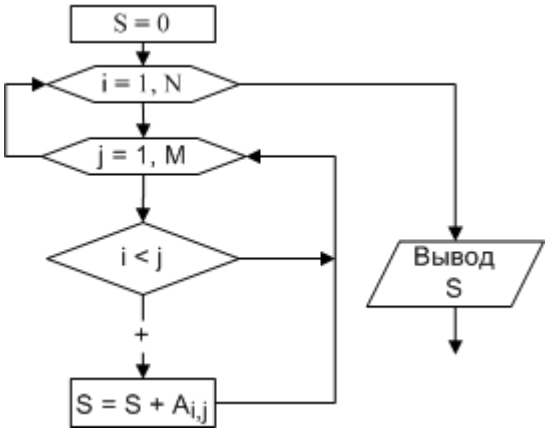

Рис. 7.4. Блок-схема примера 7.1 (алгоритм 1)

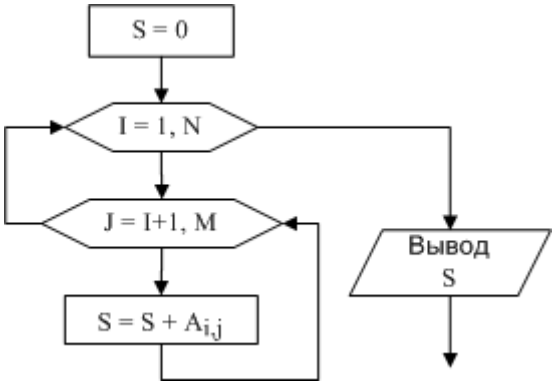

Рис. 7.5. Блок-схема примера 7.1 (алгоритм 2)

ПРИМЕР 7.2. Вычислить количество положительных элементов квадратной матрицы, расположенных по ее периметру и на диагоналях. Напомним, что в квадратной матрице число строк равно числу столбцов.

Прежде чем преступить к решению задачи рассмотрим рисунок 7.6, на котором изображена схема квадратных матриц различной размерности. Из условия задачи понятно, что не нужно рассматривать все элементы заданной

матрицы. Достаточно просмотреть первую и последнюю строки, первый и последний столбцы, а так же диагонали. Все эти элементы отмечены на схеме, причем черным цветом выделены элементы, обращение к которым может произойти дважды. Например, элемент с номером (1,1) принадлежит как к первой строке, так и к первому столбцу, а элемент с номером (N,N) находится в последней строке и последнем столбце одновременно. Кроме того, если N – число нечетное (на рисунке 7.6 эта матрица расположена слева), то существует элемент с номером (N/2+1, N/2+1), который находится на пересечении главной и побочной диагоналей. При нечетном значении N (матрица справа на рис. 7.6) диагонали не пересекаются.

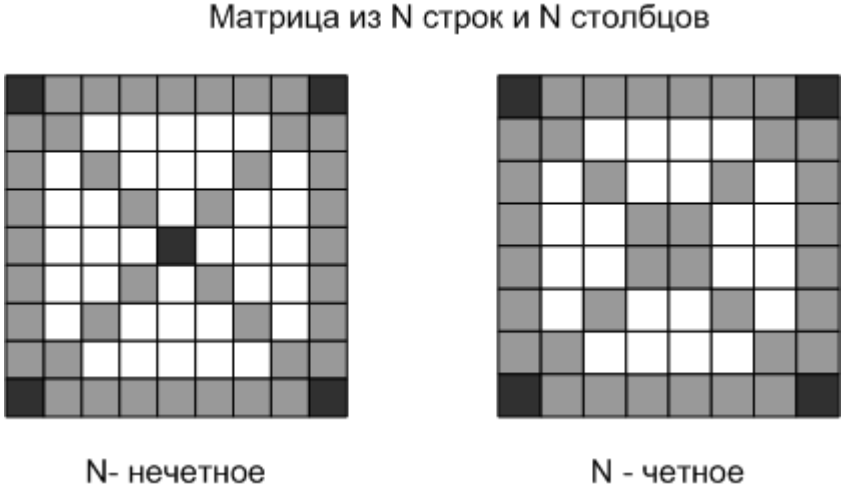

Рис. 7.6. Рисунок к условию задачи из примера 7.2

Итак, разобрав подробно постановку задачи, рассмотрим алгоритм ее решения. Для обращения к элементам главной диагонали вспомним, что номера строк этих элементов всегда равны номерам столбцов. Поэтому, если параметр *i* изменяется циклически от 1 до N, то $A_{i,i}$ – элемент главной диагонали. Воспользовавшись свойством, характерным для элементов побочной диагонали получим: i+j-1=n→j=n-i+1, следовательно, для i=1,2,…,n элемент $A_{i,n-i+1}$ – элемент побочной диагонали. Элементы, находящиеся по периметру матрицы записываются следующим образом: $A_{1,i}$ – первая строка, $A_{N,i}$ – последняя строка и соответственно $A_{i,1}$ – первый столбец, $A_{i,N}$ – последний столбец.

Блок–схема описанного алгоритма изображена на рис. 7.7. В блоке 1 организуется цикл для обращения к диагональным элементам матрицы. Причем в блоках 2–3 подсчитывается количество положительных элементов на главной диагонали, а в блоках 5–6 на побочной. Цикл в блоке 6 задает изменение параметра i от 2 до N-1. Это необходимо для того, чтобы не обращать к элементам, которые уже были рассмотрены: A_{11}, A_{1N}, $A_{N,1}$ и $A_{N,N}$. Блоки 7–8 подсчитывают положительные элементы в первой строке, 9 и 10 – в последней строке, 11 и 12 – в первом столбце, а 13 и 14 в последнем. Блок 15 проверяет, не был ли элемент, находящийся на пересечении диагоналей, подсчитан дважды. Напомним, что это могло произойти только в том случае, если N – нечетное число и этот элемент был положительным. Эти условия и проверяются в блоке 16, который уменьшает вычисленное количество положительных элементов на единицу.

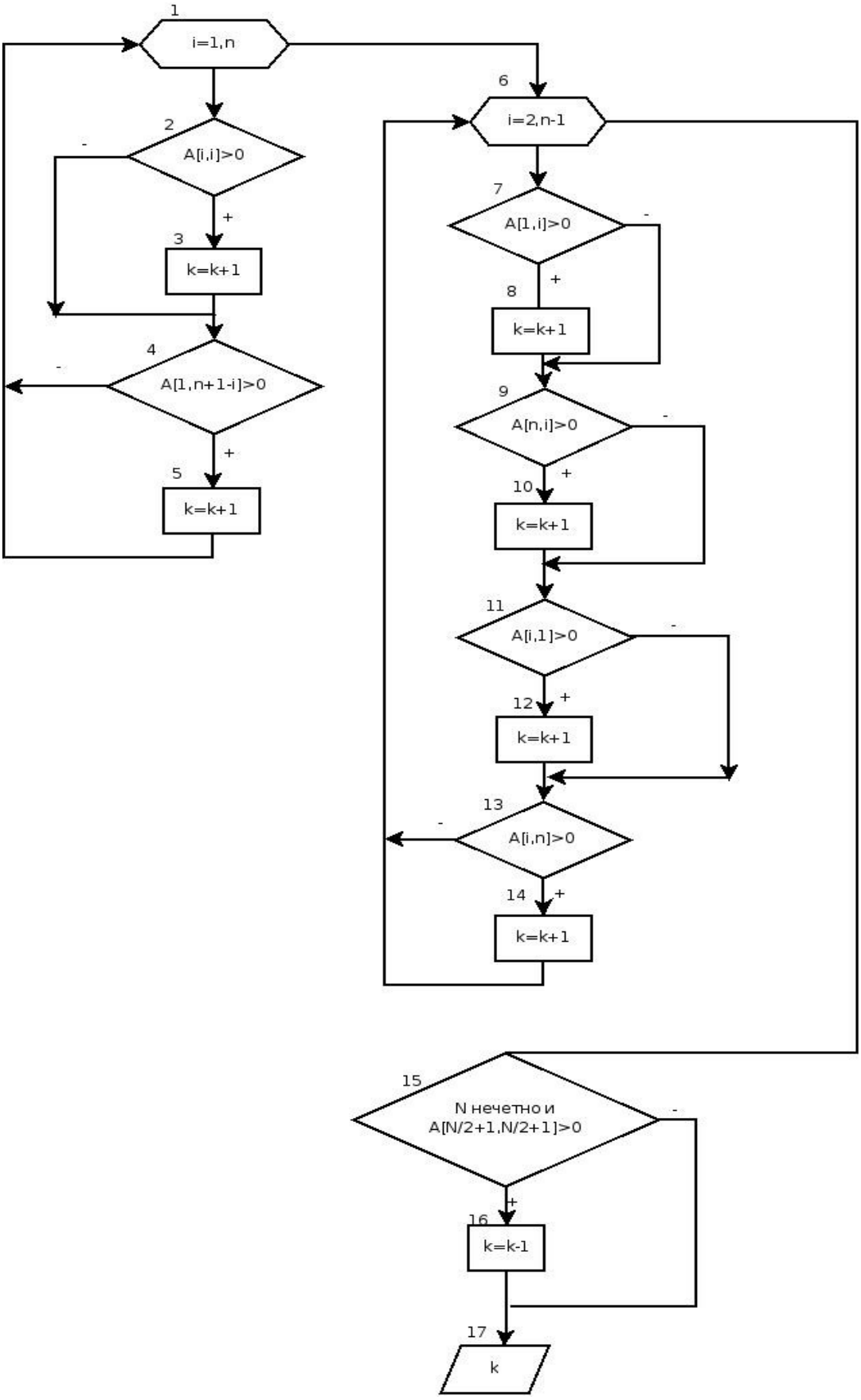

Рис. 7.7. Блок–схема к примеру 7.2

ПРИМЕР 7.3. Проверить, является ли заданная квадратная матрица единичной. Единичной называют матрицу, у которой элементы главной диагонали – единицы, а все остальные – нули.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Решать задачу будем так. Предположим, что матрица единичная (FL=ИСТИНА) и попытаемся доказать обратное. Если окажется, что хотя бы один диагональный элемент не равен единице или любой из элементов вне диагонали не равен нулю, то матрица единичной не является (FL=ЛОЖЬ). Воспользовавшись логическими операциями все эти условия можно соединить в одно и составить блок–схему (рис. 7.8).

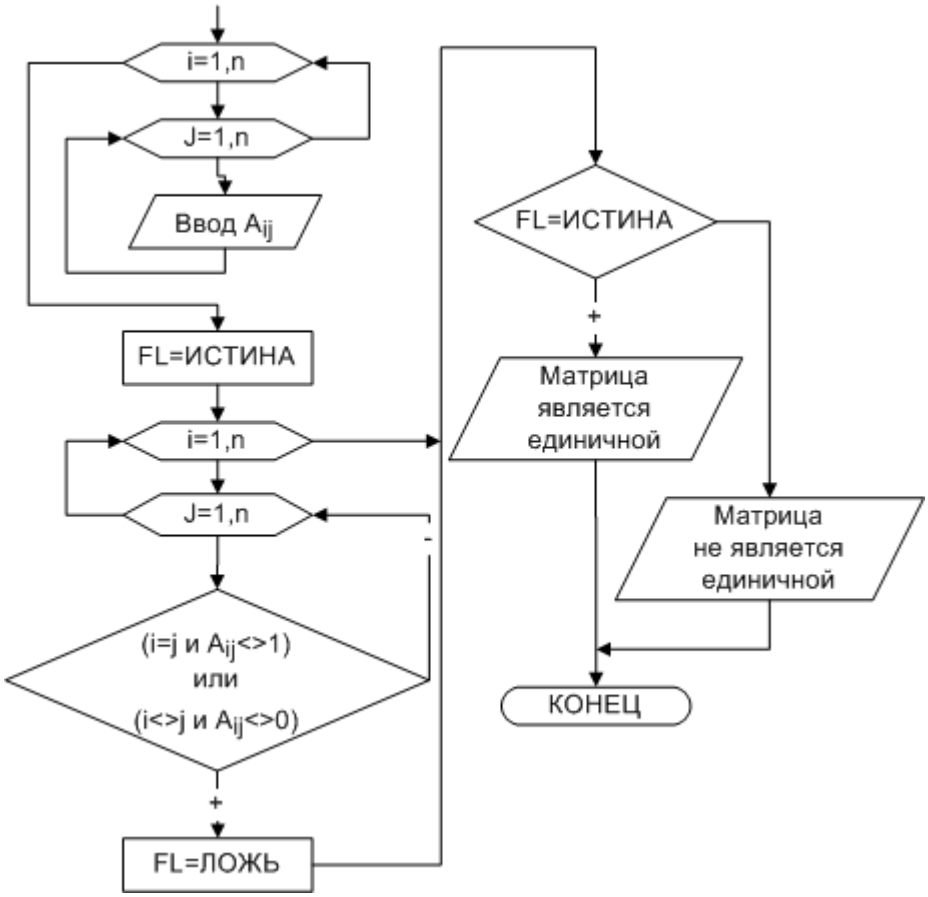

Рис. 7.8. Блок–схема к примеру 7.3

ПРИМЕР 7.4. Преобразовать исходную матрицу так, чтобы первый элемент каждой строки был заменен средним арифметическим элементов этой строки.

Для решения данной задачи необходимо найти в каждой строке сумму элементов, которую разделить на их количество. Полученный результат записать в первый элемент соответствующей строки. Блок-схема алгоритма решения приведена на рис. 7.9.

ПРИМЕР 7.5. Задана матрица $A_{n,m}$. Сформировать вектор P_m, в который записать номера строк максимальных элементов каждого столбца.

Алгоритм решения этой задачи следующий: для каждого столбца матрицы находим максимальный элемент и его номер, номер максимального элемента j–го столбца матрицы записываем в j−–й элемент массива P.

Блок-схема алгоритма приведена на рис. 7.10.

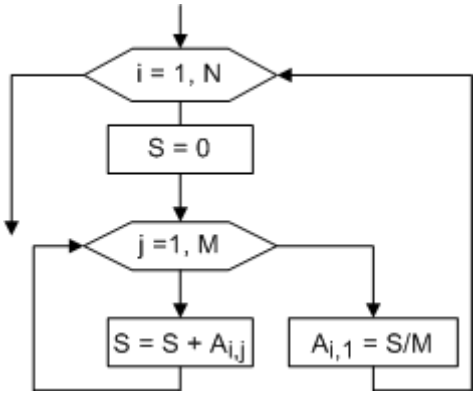

Рис. 7.9. Блок–схема алгоритма примера 7.4

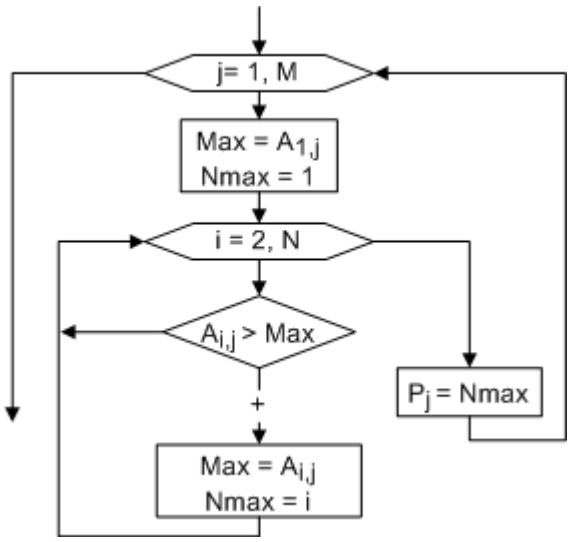

Рис. 7.10. Блок-схема алгоритма примера 7.5

ПРИМЕР 7.6. Написать программу умножения двух матриц $A_{n,m}$ и $B_{m,l}$.

Например, необходимо перемножить две матрицы

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{bb} \\ b_{31} & b_{32} \end{pmatrix}.$$

Воспользовавшись правилом «строка на столбец», получим матрицу:

$$\begin{pmatrix} a_{11}\cdot b_{11}+a_{12}\cdot b_{21}+a_{13}\cdot b_{31} & a_{11}\cdot b_{12}+a_{12}\cdot b_{22}+a_{13}\cdot b_{32} \\ a_{21}\cdot b_{11}+a_{22}\cdot b_{21}+a_{23}\cdot b_{31} & a_{21}\cdot b_{12}+a_{22}\cdot b_{22}+a_{23}\cdot b_{32} \\ a_{31}\cdot b_{11}+a_{32}\cdot b_{21}+a_{33}\cdot b_{31} & a_{31}\cdot b_{12}+a_{32}\cdot b_{22}+a_{33}\cdot b_{32} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{pmatrix}.$$

В общем виде формула для нахождения элемента C_{ij} матрицы имеет вид:

$$C_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj}, \quad \text{где } i=1,N \text{ и } j=1,L.$$

Обратите внимание, что проводить операцию умножения можно только в том случае, если количество строк левой матрицы совпадает с количеством столбцов правой. Кроме того, $A \times B \neq B \times A$.

Блок-схема, изображенная на рис. 7.11, реализует расчет каждого элемента матрицы C в виде суммы по вышеприведенной формуле.

ПРИМЕР 7.7. Поменять местами n-й и l-й столбцы матрицы $A_{k,m}$. Блок-схема приведена на рис. 7.12.

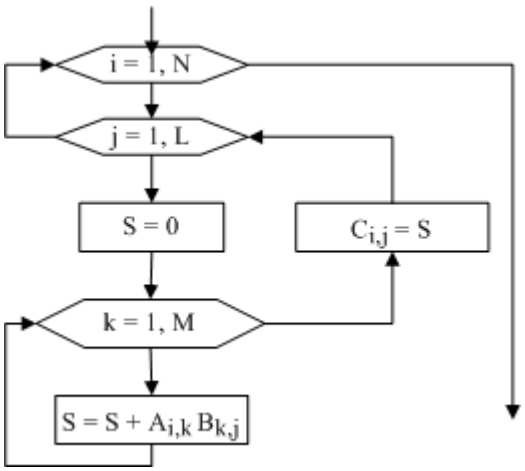

Рис. 7.11. Алгоритм умножения двух матриц

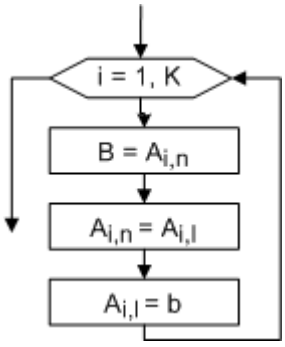

Рис. 7.12. Блок-схема алгоритма примера 7.7

ПРИМЕР 7.8. Преобразовать матрицу $A_{m,n}$ таким образом, чтобы каждый столбец был упорядочен по убыванию. Алгоритм решения этой задачи сводится к тому, что уже известный нам по предыдущей главе алгоритм упорядочивания элементов в массиве выполняется для каждого столбца матрицы. Блок-схема приведена на рис. 7.13.

ПРИМЕР 7.9. Преобразовать матрицу $A_{m,n}$ так, чтобы строки с нечетными индексами были упорядочены по убыванию, с четными – по возрастанию. Блок-схема приведена на рис. 7.14.

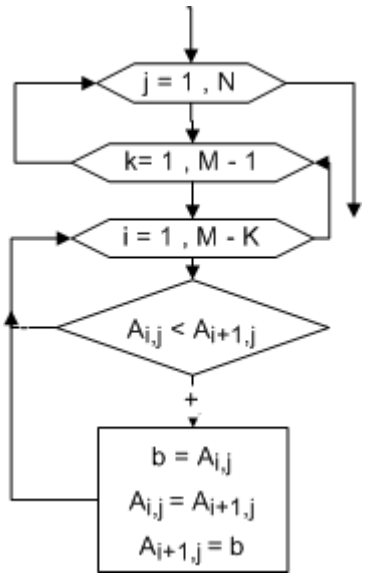

Рис 7.13. Блок-схема алгоритма примера 7.8

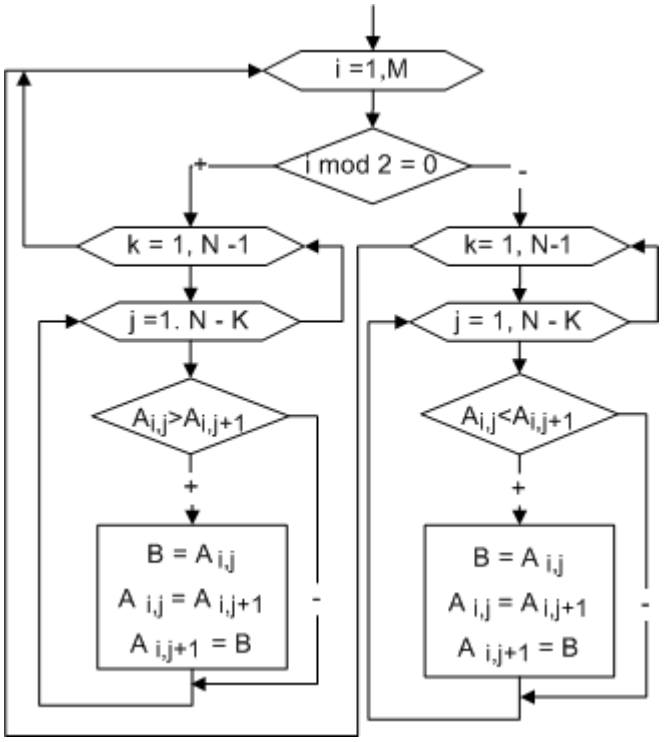

Рис. 7.14. Блок-схема алгоритма примера 7.9