Оглавление

Введение

Актуальность темы. В исследованиях по теории вероятностей заметное место занимают вероятностные задачи комбинаторного характера. Одним из интенсивно развивающихся направлений таких исследований является изучение различных схем размещения частиц по ячейкам.

Изучаемая в дипломной работе схема представляет собой схему размещения неразличимых частиц по неразличимым ячейкам. Данная работа посвящается рассмотрению вопроса нахождения числа размещений в общей и частной (без пустых ячеек) схемах размещения неразличимых частиц по неразличимым ячейкам.

Научная новизна. В указанной общей схеме и в частной схеме без пустых ячеек проводится ряд исследований:

- для общего числа исходов частной схемы выписана рекуррента и по ней получена явная точная формула для него; выявлена связь чисел исходов в общей и частной схемах;
- описан случайный процесс последовательного размещения по одной частице по ячейкам, на основании которого приводится алгоритм решения задачи перечислительной комбинаторики представления всех возможных исходов размещения фиксированного числа частиц по ячейкам в схеме и получения распределений их вероятностей;
- предложены разные способы моделирования состояний схемы и приближенного нахождения числа ее исходов методом стохастического моделирования.

Цели работы:

- подробное рассмотрение вышеуказанных исследований;
- разработка вычислительных алгоритмов их компьютерной реализации;
- создание стандартных программ, реализующие эти алгоритмы.

ГЛАВА 1

Обзорная часть

§1. Обзор.

По близкой тематике была изучена следующая литература:

1. Джеймс Андерсон "Дискретная математика и комбинаторика";
2. Колчин, Севастьянов "Случайные размещения";
3. В.Липский "Комбинаторика для программистов";
4. Н.Я.Виленкин "Популярная комбинаторика";
5. Гульден Я., Джексон Д."Перечислительная комбинаторика ";
6. Ерош И.Л. "Дискретная математика. Комбинаторика";
7. Кофман А. "Введение в прикладную комбинаторику ";
8. Носов В.А. "Комбинаторика и теория графов";
9. Плотников А.Д. "Дискретная математика";
10. Риордан Дж."Введение в комбинаторный анализ";
11. Стенли Р. "Перечислительная Комбинаторика ";
12. Холл М. "Комбинаторика".

В дальнейшем, в этом пункте, будет вкратце рассмотрена каждая из выше представленных книг с целью краткого описания тех схем размещений, которые там рассматриваются.

Начнем наш обзор с книги Джеймса Андерсона "Дискретная математика и комбинаторика". Ниже представлена таблица, в которой содержатся все формулы, по котором находится количество способов размещения частиц по ячейкам в разных схемах размещения, представленные в данной книге.

	Размещение n частиц по k ячейкам	Количество способов
(1)	частицы — различимые, ячейки — различимые, n_i объектов в i-ой ячейке	$C(n; n_1, n_2, \ldots, n_k)$
(2)	частицы — различимые, ячейки — различимые, ящики могут быть пустыми	k^n
(3)	частицы — различимые, ячейки — различимые, ящики не могут быть пустыми	$k!S_k^{(n)}$
(4)	частицы — неразличимые, ячейки — различимые, ящики могут быть пустыми	$C(n+k-1, n) =$ $= \frac{n+k-1}{n!(k-1)!}$
(5)	частицы — неразличимые, ячейки — различимые, ящики не могут быть пустыми	$C(n-1, k-1) =$ $= \frac{n-1}{(n-k)!(k-1)!}$
(6)	частицы — различимые, ячейки — неразличимые, ящики могут быть пустыми	$S_1^{(n)} + S_2^{(n)} + \ldots + S_k^{(n)}$
(7)	частицы — различимые, ячейки — неразличимые, ящики не могут быть пустыми	$S_k^{(n)}$

Здесь $\{S_k^{(n)} : 0 \leq k \leq n\}$ — множество чисел Стирлинга второго рода (количество неупорядоченных разбиений n-элементного множества на k непустых подмножеств).

Числа Стирлинга второго рода определены соотношениями

$S_0^{(n)} = 0$при всех n ≥ 1;

$S_n^{(n)} = 1$ при всех n ≥ 0;

$S_k^{(n+1)} = S_{k-1}^n + kS_k^{(n)}$

Рассмотрим каждую формулу в таблице отдельно.

Формула (1). Если имеется n различимых объектов и k различимых ящиков, то число способов поместить n_i объектов в i-ый ящик для всех $1 \leq i \leq k$, где $n = n_1 + n_2 + n_3 + \ldots + n_k$, определяется по формуле

$$C(n; n_1, n_2, \ldots, n_k) = \frac{n!}{n_1!n_2!n_3! \ldots n_k!}$$

В этом случае ящик может быть пустой.

Формула (2). Здесь количество объектов, помещаемых в каждый ящик, не фиксировано, поэтому существуют k возможностей выбора ящика для размещения каждого из n объектов. Поэтому, согласно комбинаторному принципу умножения существуют k^n способов разместить объекты в ящиках.

Формула (6). Прежде чем рассматривать формулу (3), рассмотрим формулу (6) и (7). В (6) формуле снимается ограничение на то, что ни один из ящиков не может быть пустым, как в формуле (3) и (7). Тогда возможны ситуации:

1) все n объектов помещены в один ящик, что можно осуществить $S_1^{(n)}$ способами;

2) все n объектов помещены в два ящика, что можно осуществить $S_2^{(n)}$ способами;

...

k) все n объектов помещены в k ящиков, что можно осуществить $S^(n)_k$ способами, и т.д.

Итого, имеется

$$S_1^{(n)} + S_2^{(n)} + S_3^{(n)} + \ldots + S_k^{(n)}$$

способов размещения n объектов в k ящиках при условии, что ящики могут быть пустыми.

Формула (7). $S_k^{(n)}$ — число возможных способов размещения n различимых объектов в k неразличимых ящиках, ни один из которых не может быть пустым.

Формула (3). Используя формулу (7) мы можем найти количество способов размещения n различимых объектов в k различимых ящиках, ни один из которых не может быть пустым. Необходимо перейти от неупорядоченных ящиков к упорядоченным. При переходе от упорядоченных ящиков к неупорядоченным нужно делить соответствующее количество способов размещения на $k!$. Поэтому при обратном переходе нужно умножать на $k!$. Таким образом, существуют $k!S^{(n)}k$ способов размещения n различимых объектов в k неразличимых ящиках, ни один из которых не может быть пустым.

Как можно увидеть из таблицы, схема размещения неразличимых частиц по неразличимым ячейкам в этой книге не рассматривалось, но есть некоторая формула (5), являющаяся лишь частью рекуррентной формулы, которая будет непосредственно рассматриваться в основной части работы, и по которой будет составлен алгоритм. Далее в книге рассматривается числа Стирлинга второго рода и некоторая теорема, где они используются. Вот как звучит теорема: "Существуют $S_k^{(n)}$ способов размещения n различимых частиц по k неразличимым ячейкам, когда ни одна из ячеек не может быть пустой, где $\{S_k^{(n)} : 0 \leq k \leq n\}$ — множество чисел Стирлинга второго рода.". Как можно видеть, здесь речь идет о формуле (7). В данной теореме доказывается соотношение $f(n+1,k) = f(n,k-1) + kf(n,k)$, где $f(n,k)$ — количество способов размещения n различимых частиц по k неразличимым ячейкам, когда не одна из ячеек не может быть пустой. Как в дальнейшем будет видно, доказываемое соотношение похожа на формулу вычисления числа исходов в общей схеме, полученная пересчетом из числа исходов частной схемы без пустых ячеек.

Убеждаемся, что в данной книге нет даже упоминания о схеме размещения неразличимых частиц по неразличимым ячейка.

Далее рассмотрим книгу В.Липский "Комбинаторика для программистов". Здесь автор рассматривает задачу "определения числа способов размещения некоторых объектов в каком-то количестве «ящиков» так, чтобы были выполнены заданные ограничения" следующим образом: даны множества X, Y, причем $|X| = n$, $|Y| = m$. Сколько существует функций $f : X \to Y$, удовлетворяющих заданный ограничениям? Элементы множества X соответствуют объектам, элементы множества Y — ящикам, а каждая функция $f : X \to Y$ определяет некоторое размещение, указывая для каждого объекта $x \in X$ ящик

$f(x) \in Y$, в котором данный объект находится. Автор рассматривает три случая размещения частиц по ячейкам. Вкратце расскажем о них.

Первый случай, если не накладывается никаких ограничений на размещения. Тогда имеет место формула m^n.

Второй случай, если каждый ящик содержит не более одного объекта — такие размещения соответствуют взаимно однозначным функциям. Через $[m]_n$ обозначается число всех взаимно однозначных функций из n-элементного множества в m-элементное множество. Тогда имеет место формула $[m]_n = m(m-1)\dots(m-n+1)$, в которой полагается $[m]_0 = 1$.

Третий случай, если мы размещаем n объектов по m ящикам так, чтобы каждый ящик содержал бы последовательность, а не множество, как прежде, помещенных в нем объектов. Размещения такого типа будем называть упорядоченными размещениями n объектов по m ящикам. Число таких упорядочений обозначается через $[m]_n$ и имеет место формула $[m]_n = m(m+1)\dots(m+n-1)$.

Опять же, никого упоминания о нашей схеме.

Следующая книга — Н.Я.Виленкин "Популярная комбинаторика". Здесь рассматриваются схемы (2),(4) и (5) из выше показанной таблицы, поэтому рассматривать подробно их не будем. Из этого делаем вывод. что и в этой книге нашей схемы нет.

В книге Риордана Дж. "Введение в комбинаторный анализ" рассказывается о нахождение числа размещений тех же схем, что и у Джеймса Андерсона, но можно добавить еще две схемы. Первая схема: число способов размещения n различимых частиц по m различимым ячейкам, так чтобы p ячеек были заняты, а $m-p$ свободны, равно $(m)_p S(n,p)$, где $S(n,p)$ — числа Стирлинга второго рода. Вторая схема: число возможных размещений n объектов спецификации $(1^{n_1}2^{n_2}\dots)$ по m различным ячейкам равно

$$m^{n_1}\left(\begin{array}{c} m+1 \\ 2 \end{array}\right)^{n_2} m^{n_1}\left(\begin{array}{c} m+2 \\ 3 \end{array}\right)^{n_3}\dots.$$

Как можно видеть здесь также не рассматривается наша схема.

В книге Ероша И.Л. "Дискретная математика. Комбинаторика" рассматривается схема (2).

Также были просмотрены выпуски журнала "Теория вероятности и ее применение" с 1964-2005 года. В работах журнала рассматривается другая схема. Например, в заметке В.А.Иванова и С.Ю.Теребулина "Некоторые предельные теоремы в неравновероятной схеме размещения частиц комплектами" [ТВП 1978 т.23 в.3] рассматривается схема размещения частиц комплектами: имеется N ячеек, в которые бросают независимо n' групп по m частиц в каждой группе; частицы каждой группы размещаются в ячейках по одной так, что все возможные размещения имеют одинаковую вероятность $\begin{pmatrix} N \\ m \end{pmatrix}^{-1}$. В своей заметки авторы также утверждают, что в ряде приложений это предположение о равновероятности размещений комплектов не выполняется, а вместо него возникает условие: вероятность того, что частицы одного комплекта займут ячейки с номерами $i_1 < i_2 < \dots i_m$, равна

$$P_{i_1 \dots i_m} = p_{i_1} \dots p_{i_m} (\sum_{j_1 < \dots < j_m} p_{j_1} \dots p_{j_m})^{-1}$$

(Здесь $p_1, \dots, p_N$ — некоторые заданные числа, которые в дальнейшем для удобства мы нормируем условием $\sum_{i=1}^{N} p_i = 1$.) Этому вопросу и посвящена заметка, о схеме размещения неразличимых частиц по неразличимым ячейкам речи не идет.

Как видно, в рассмотренных книгах нет упоминаний о схема размещения неразличимых частиц по неразличимым ячейкам. Следовательно, можно сделать вывод, основываясь на данной литературе, что наша схема несет в себе научную новизну.

§2. Вид результата размещения частиц по ячейкам в схеме.

Результаты размещения в схеме совпадают со вторыми маркировками заполнений ячеек в схеме размещения неразличимых частиц по различимым ячейкам, т.е. описываются векторами вида $\overrightarrow{\mu_r} = (\mu_0, \mu_1, \dots, \mu_r)$, где r — число частиц, n —число ячеек, μ_i — число ячеек, содержащих ровно по i частиц после

размещения r частиц по n ячейкам, так, что

$$\sum_{i=0}^{r} \mu_i = n; i = \overline{0,r}; \sum_{i=1}^{r} i\mu_i = r$$

Другой вид результата — вариационный ряд заполнений ячеек вида ($\underbrace{0,\ldots,0}_{\mu_0}$, $\underbrace{1,1,\ldots,1}_{\mu_1},\ldots\underbrace{r}_{\mu_r}$). В частной схеме $\mu_0 = 0$.

§3. Изменение вектора $\overrightarrow{\mu_k}$ при добавлении одной частицы.

Опишем отличие вектора $\overrightarrow{\mu}_{k+1}$ от вектора $\overrightarrow{\mu_k}$ при размещении по n ячейкам схемы после k еще одной частицы. Оказывается что при этом в векторе $\overrightarrow{\mu_k}$ будут изменяться на единицу ровно две соседних компоненте, причем левая будет уменьшаться на единицу, а правая — увеличивается на единицу. В силу неотрицательности компонент векторов $\overrightarrow{\mu_k}$ и $\overrightarrow{\mu}_{k+1}$ число разных векторов $\overrightarrow{\mu}_{k+1}$, получающихся из данного вектора $\overrightarrow{\mu_k}$ будет равняться числу его ненулевых компонент, причем это число в векторе $\overrightarrow{\mu}_{k+1}$ по сравнению с вектором $\overrightarrow{\mu_k}$ при попадании добавленной частицы в ячейку с i частицами в зависимости от значений компонент вектора $\overrightarrow{\mu_k}$

а) не меняется при $\mu_i > 1, \mu_{i+1} > 0$;

б) увеличивается на 1 при $\mu_i > 1, \mu_{i+1} = 0$;

в) уменьшается на 1 при $\mu_i = 1, \mu_{i+1} > 0$.

Очевидно, что из разных векторов $\overrightarrow{\mu_k}$ при добавлении одной частицы могут получаться одинаковые векторы $\overrightarrow{\mu_{k+1}}$. Это происходит, например, когда векторы $\{\overrightarrow{\mu_k}\}$ отличаются друг от друга только компонентами μ_0 на единицу.

Пример 1.

Пусть $n = 4$, векторы $\overrightarrow{\mu_5} = (0,3,1,0,0,0)$ и $\overrightarrow{\mu_5} = (1,2,0,1,0,0)$ приведут к вектору $\overrightarrow{\mu_6} = (0,3,0,1,0,0,0)$ при попадании частицы в первом случае в ячейку с двумя частицами, а во втором — в пустую.

ГЛАВА 2

Точное вычисление чисел исходов в общей и частной схемах размещения.

§1. Связь чисел исходов в общей и частной схемах размещения.

Покажем, как число исходов $N^* = N^*(r, n)$ в общей схеме получается пересчетом из числа исходов $N = N(r, n)$ частной схемы (без пустых ячеек). Для этого предложим следующую процедуру: к данным r частицам добавим n частиц, получим $r^* = r + n$ частиц, которые разместим по n ячейкам без пустых ячеек $N(r^*, n) = N(r+n, n)$ способами. После этого вынем из каждой ячейки по одной частице, от чего число размещений не меняется и снова равно $N(r+n, n)$. Отсюда получаем, что

$$N^*(r, n) = N(r + n, n). \tag{1}$$

§2. Рекуррента для числа $N = N(r, n)$

Для числа $N = N(r, n)$ строим рекурренту по принципу деления всей совокупности N исходов частной схемы на две, численности которых зависят от меньших значений параметров, чем исходные r и n. К первой совокупности отнесем те исходы из $N(r, n)$, в которых минимальное заполнение ячеек равно 1 (их число $N(r-1, n-1)$, и оно получается, если в любую ячейку положить одну частицу, а остальные $(r-1)$ разместить по по той же схеме по остальным $(n-1)$ ячейкам), а ко второй — те исходы, в которых минимальное заполнение ячеек > 1 (их число $N(r-n, n)$, и они получаются, если во все ячейки положить

по одной частице, а остальные $(r-n)$ частиц разместить по n ячейкам по той же схеме). Таким образом. приходим к рекурренте

$$N(r,n) = N(r-1,n-1) + N(r-n,n). \tag{2}$$

где числа $N(r,n)$ обладают при r,$n \geq 0$ следующими очевидными свойствами: 1)$N(r,r) = 1$; 2)$N(r,n) = 0$ при $r < n$; 3)$N(r,0) = 0$; 4)$N(r,1) = 1$. Приведем пример использования рекурренты.

Пример 2. Пусть $r = 8$, $n = 3$. Тогда по (2) получаем: $N(8,3) = N(7,2) + N(5,3) = N(6,1) + N(5,2) + N(4,2) + N(2,3) = 1 + N(4,1) + N(3,2) + N(3,1) + N(2,2) + 0 = 1 + 1 + N(2,1) + N(1,2) + 1 + 1 + 0 = 1 + 1 + 1 + 0 + 1 + 1 + 0 = 5$. Получили, что число способов размещения неразличимых частиц по неразличимым ячейкам по рекурренте (2) равно 5. Визуально перебрав исходы схемы, можно получить разные заполнения ячеек без пустых с учетом того,что ячейки являются неразличимыми: $(1,1,6), (1,2,5), (1,3,4), (2,2,4), (2,3,3)$. Эти заполнения соответствуют векторам $\overrightarrow{\mu_8} = (\mu_0, \mu_1, \ldots, \mu_8)$: $(0,2,0,0,0,0,1,0,0)$, $(0,1,1,0,0,1,0,0,0), (0,1,0,1,1,0,0,0,0,0), (0,0,2,0,1,0,0,0,0)$,
$(0,0,1,2,0,0,0,0,0)$, то есть получаем 5 вариантов, как и по рекурренте (2).

Далее рассмотрим блок-схему алгоритма нахождения числа $N(r,n)$ способов размещения неразличимых частиц по неразличимым ячейка для любого r и n по рекурренте (2) и программу, написанную в пакете *Wolfram Mathematica 8* по алгоритму. Используя свойства чисел $N(r,n)$ 1)-4), называемые ниже предельными, производим по (2) такое число итераций по каждой ветви рекурренты (2), которое приводит к предельным значениям слагаемых. При этом в общем случае длины ветвей рекурренты до достижения предельных значений по свойства 1)-4) разные. В качестве исходных r и n возьмем значения из примера 1 и заодно проверим правильность работы программы.

<u>Блок-схема 1</u> (алгоритм нахождения N(r,n) по рекурренте (2)).

Программа 1 (нахождение $N(r,n)$ по рекурренте (2)).

```
Timing[r = 8; n = 3; y = 1;
  N = {{r, n}};
  NN = Table[0, {2}, {2}];
  NN = {{N[[1, 1]] - 1, N[[1, 2]] - 1}, {N[[1, 1]] - N[[1, 2]], N[[1, 2]]}};
  N = NN; y = 2;
  NN = Table[0, {100}]; k = 2; ChisloRazmeshenii = 0;
  For[v = 1, v <= 10^8, NN = Table[0, {y * 2}, {2}]; s = 0;
    For[i = 1, i <= y, If[i > 1, s + +];
      For[j = 1, j <= 2, NN[[i + s, j]] = N[[i, j]] - 1; j + +];
      For[j = 1, j <= 2,
        If[j == 1, NN[[i + 1 + s, j]] = N[[i, j]] - N[[i, 2]],
      NN[[i + 1 + s, j]] = N[[i, 2]]];
      j + +];
    i + +];
  N = NN;
  For[i = 1, i <= y * 2,
    If[N[[i, 1]] == N[[i, 2]] || N[[i, 2]] == 1, ChisloRazmeshenii + +];
  i + +];
  For[i = 1, i <= y * 2,
    If[N[[i, 1]] == N[[i, 2]] || N[[i, 2]] == 1 ||
     || N[[i, 1]] < N[[i, 2]] || N[[i, 2]] == 0,
    N[[i]] = {0, 0}];
  i + +];
  k = 0;
  For[i = 1, i <= y * 2, If[Not[N[[i]] == {0, 0}], k + +]; i + +];
  NNN = Table[0, {k}, {2}]; c = 1;
  For[u = 1, u <= k, o = c;
    For[i = o, i <= y * 2,
      If[Not[N[[i]] == {0, 0}], NNN[[u]] = N[[i]]; Break[]];
    i + +];
  c = i + 1; u + +];
  N = NNN; y = k; t = 1; e = 2;
```

$If[y == 1,$

If[N[[1, 1]] == N[[1, 2]] || N[[1, 2]] == 1 ||

$|| N[[1,1]] < N[[1,2]] \;||\; N[[1,2]] == 0, Break[]]];$

$If[N == \{\}, Break[]];$

$v++]$

Print["Число размещений с ", r, " частицами и ", n, " ячейками = ", ChisloRazmeshenii]]

Обозначения в программе:

r — количество частиц; n — количество ячеек; $y, k, i, j, u, t, e, c, o, s$ — счетчики;

$ChisloRazmeshenii$ — счетчик числа размещений (суммирует все единицы, полученные из свойств 1)-4));

N — числа $N(r, n)$, слагаемые рекурренты (2);

NN — массив полученных слагаемых рекурренты (2) на каждой итерации.

Результат работы программы: *"Число размещений с 8 частицами и 3 ячейками = 5".*

Как видно результат, полученный в примере 1, равен результату, полученному с помощью программы. Для того, чтобы не оставалось сомнений в правильности работы программы приведем еще пару результатов ее работы с небольшими значениями r и n, а также с большими r и n, чтобы убедиться в работоспособности программы при больших значениях. Для этого, мы меняем, на нужные нам, значения r и n в первой строке программы. Для начало возьмем небольшие значения,например $r = 10$ и $n = 4$. После подстановки этих значений в программы получили ответ "9", то есть 9 способов размещения. Этот ответ можно проверить вручную.

Далее подставим большие r и n, например 44 и 5 соответственно, и получим ответ 1602 способа размещения. Как видно программа не зацикливается для любого количества частиц и ячеек.

Ниже указан полный вывод всех слагаемых (до очистки и после).

Результаты работы программы на каждой итерации:

{{7, 2},{5, 3}} — слагаемые после проверки на свойства
{{6,1},{5,2},{4,2},{2,3}} — слагаемые до проверки на свойства
{{5,2},{4,2}} — слагаемые после проверки на свойства
{{4,1},{3,2},{3,1},{2,2}} — слагаемые до проверки на свойства
{{3,2}} — слагаемые после проверки на свойства
{{2,1},{1,2}} — слагаемые после проверки на свойства
{} — слагаемые после проверки на свойства
Число размещений с 8 частицами и 3 ячейками = 5

Как можно увидеть, в шестой строке вывода получились предельные значения
"$\{\{2,1\},\{1,2\}\}$", их программа удалила и получила пустой массив "{}" (седьмая строка вывода). Проверив на пустоту массив, цикл остановился, и программа вывела ответ: "*Число размещений с 8 частицами и 3 ячейками = 5*". Блок схема программы приведена в приложении.

Инструкция по использованию программы.

1. Открываем программу в пакете Wolfram Mathematica;
2. в первой строке вводим количество частиц r и количество ячеек n после знака равно;
3. нажимаем комбинацию клавиш Shift-Enter и получаем ответ.

На этом описание алгоритма вычисления числа размещений по рекурренте (2) можно закончить и перейти к алгоритму вычисления числа размещений по точной формуле.

§3. Точная формула для числа $N = N(r, n)$.

Для начало теоретически выведем явную формулу для числа $N(r, n)$. Ее вывод производится на основе рекурренты (2) и закономерностях ее применения при многократных итерациях с понижением значений параметров r и n

до получения предельных значений. Но есть некоторая особенность явной формулы — предельные значения по параметрам достигаются на одной и той же итерации по всем ветвям рекурренты (2), несмотря на то, что при использовании рекурренты (2) это не так, поскольку предельные значения определяются по свойствам 1)-4),то есть ветви рекурренты могут заканчиваться по свойствам 1)-4), мы убедились в этом на примере 1. Но допущение о достижении предельных значений на одной и той же итерации справедливо, если формально применять рекурренту (2) к числам $N(r,n)$, уже достигшим предельных значений, для чего доопределим свойства 1)-4) следующим способом:

$1a) N(r,r) = 1, r \geq 0;$

$2a) N(r,n) = 0, r < n;$

$3a) N(r,n) = 0, n \leq 0, r + n \neq 0;$

$4a) N(r,1) = 1.$

Это означает, что количество итераций по всем ветвям рекурренты (2) могут быть выровнены.

Введем индикаторную функцию:

$$I(z) = \begin{cases} 0, & \text{если } z < 0; \\ 1, & \text{если } z \geq 0; \end{cases}$$

Тогда мы можем представить наше число N с учетом свойств $1a) - 4a)$ и рекурренты (2) следующей формулой

$$N(r,n) = \sum_{j=1}^{2^m} I(r_{mj} - n_{mj}), \tag{3}$$

здесь

$$m = min\ i : \sum_{j=1}^{2^i} (n_{ij} - 1)(r_{ij} - n_{ij}) I(r_{ij} - n_{ij}) = 0, \tag{4}$$

r_{ij} и n_{ij} — значения параметров r и n соответственно в j-ом слагаемом по рекурренте (2) на i-ой итерации. Положим $r = r_{01}, n = n_{01}$.

С помощью формулы (4) можно найти число m, проверяя для каждого $i = = 1, 2, \ldots$ условие

$$\sum_{j=1}^{2^i} (n_{ij} - 1)(r_{ij} - n_{ij}) I(r_{ij} - nij) = 0, \tag{5}$$

при первом выполнении условия (5), получаем число m, которое равно последнему значению i.

Следовательно нам необходимо далее построить формулы, по которым будут вычисляться параметры r_{mj} и n_{mj}, которые,в свою очередь, участвуют в формуле (3). Для этого потребуется находить значения r_{ij} и n_{ij} и для $i < m$. При этом. так как рекуррента (2) разветвляется на два слагаемых, то на $i-$ой итерации $j = \overline{1, 2^i}$.

Далее представим схемы изменения каждого из параметров r и n в слагаемых при последовательных итерациях рекурренты (2), чтобы выявить некие закономерности при применении рекурренты. Начнем с параметра n_{ij}, обозначим эту схему, как схема 1.

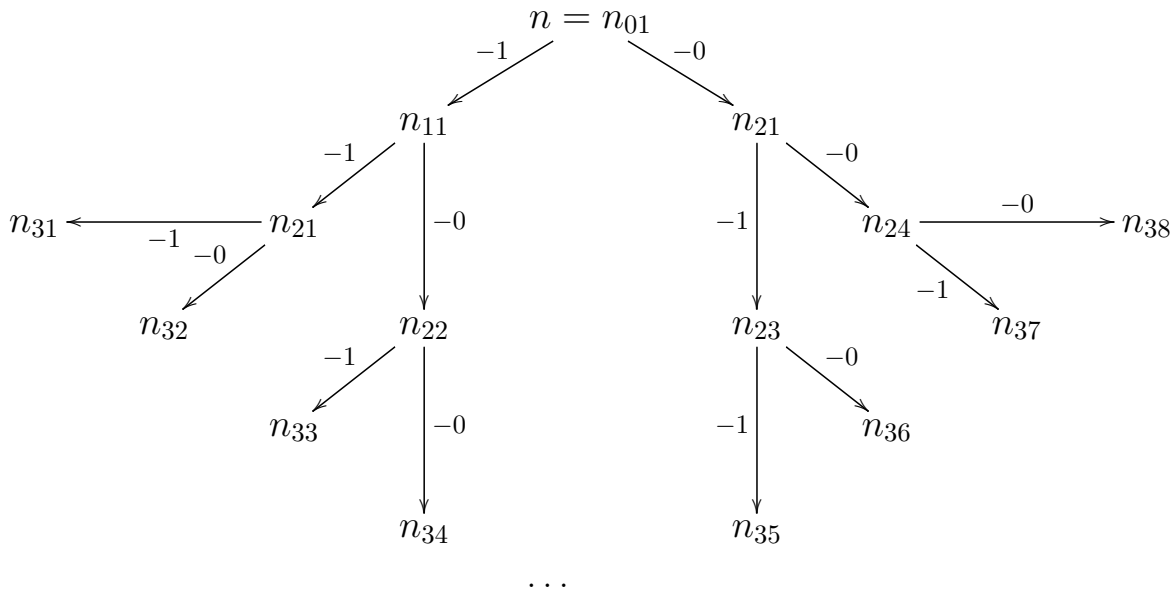


Схема 1

Здесь по вертикали (сверху вниз) представлены параметры n_{ij} по мере проведения итераций, а по дугообразной горизонтали — его последовательные обозначения в слагаемых рекурренты в порядке ее применения. На стрелках указаны действия для пересчета n_{ij}.

Данная схема представляет собой некий граф переходов из состояния n_{01} в другие возможные по рекурренте (2) состояния n_{ij}. Сопоставим всем правым ребрам нули, а левым — единицы. Отсюда видно, что каждый переход из состояния n_{01} в любое возможное состояние на $m-$ой итерации можно представить

как траекторию в виде последовательности нулей и единиц, где нули и единицы соответствуют последовательным ребрам графа от состояния n_{01} до состояния n_{mj}. Обозначим эту траекторию $(T_m) = (T_m)_2$, она является двоичной записью числа $(2^m - j)$, где j — номер состояния на $m-$ом уровне. Из схемы 1 следует, что для того, чтобы найти значения n_{mj} нужно из начального значения n вычесть столько единиц, сколько их в траектории графа от состояния $n = n_{01}$ до состояния n_{mj}, то есть число единичных разрядов двоичной записи числа $(2^m - j)$. Отсюда получаем следующую формулу

$$n_{mj} = n - S_{2^m - j}, \tag{6}$$

здесь S_z — сумма двоичных разрядов числа z.

Приведем пример применения формулы (6).

Пример 3. Возьмем для простоты понимания $m = 2$, следовательно, состояниям с номерами $j = \overline{1, 2^2} = \overline{1, 4}$ определяемым вторыми индексами параметра n числа $N(r, n)$, то есть состояниям $n_{21}, n_{22}, n_{23}, n_{24}$ будут соответствовать траектории $(11), (10), (01), (00)$ представляющие собой двоичные записи десятичных чисел $(2^m - j)$, где j — номер состояния на $m = 2-$ем уровне. Тогда $\{n_{2j}\}$ при $j = \overline{1, 4}$ будут принимать в порядке роста j значения $\{n-2, n-1, n-1, n\}$, что совпадает с результатом непосредственных вычислений по схеме 1 по указанным операциям на ребрах графа по каждой траектории.

Замечание 1. Можно определять значение n_{ij} при $i-$ой итерации, если в формуле (4) заменить фиксированное число m на любое $i \leq m$.

Замечание 2. Используя обозначение : $(T_i) = (T_i)_2$ — запись числа T_i в двоичном коде, $(T_i)_{10}$ — число в десятичном коде, соответствующее траектории (T_i) до $i-$ого уровня (включительно), получаем, что номер слагаемого на $i-$ой итерации, то есть второй индекс последнего состояния траектории (T_i) на $i-$ом уровне j удовлетворяет равенству:

$$j = 2^i - (T_i)_{10}. \tag{7}$$

Например, пусть $i = 2$. Тогда траектория

(11) пересекает уровень $i = 2$ на состоянии n_{2j}, где $j = 2^2 - (11)_{10} = 4 - 3 = 1$;

(10) пересекает уровень $i = 2$ на состоянии n_{2j}, где $j = 2^2 - (10)_{10} = 4 - 2 = 2$;

(01) пересекает уровень $i = 2$ на состоянии n_{2j}, где $j = 2^2 - (01)_{10} = 4 - 1 = 3$

и так далее.

Далее приведем соответствующую схему пересчета значений параметра r_{ij} по итерациям рекурренты (2) с указанием операций на стрелках переходов (ребрах графа).

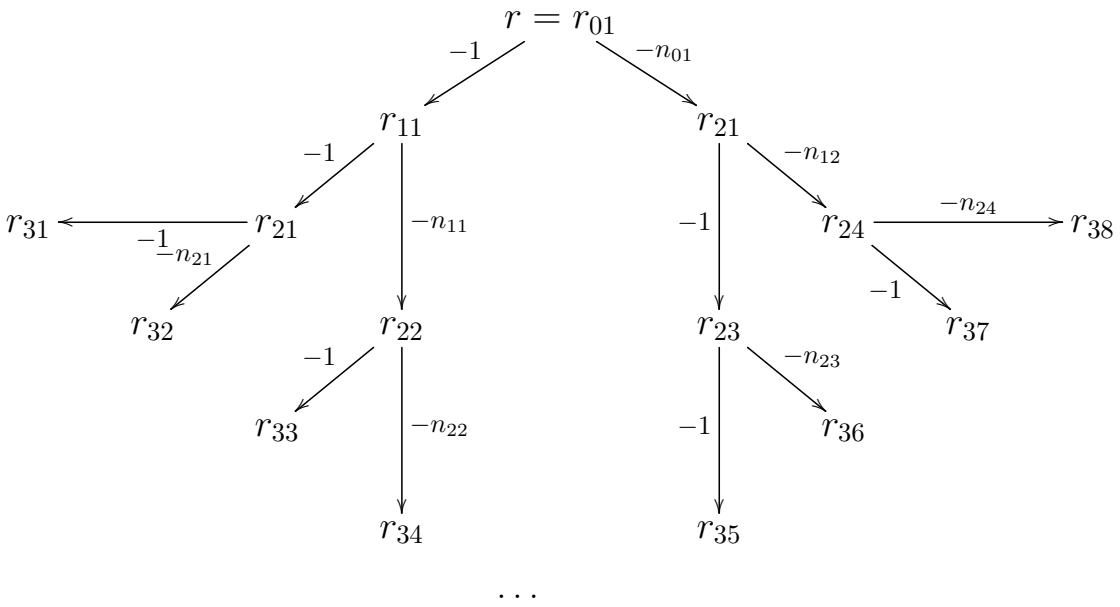


Схема 2

Аналогично, схема 2 представлена как граф переходов из состояния $r = r_{01}$ во все другие возможные состояния при итерациях по рекурренте (2) и будем ставить в соответствие всем правым ребрам графа нули, а левым — единицы. Также как и в случае схемы 1, можно представить переход из состояния r_{01} в любое $j-$ое состояние на $m-$ом уровне r_{mj} в виде траектории $(T_m) = (T_m)_2$ — последовательностью нулей и единиц в порядке итераций, то есть двоичной записью числа $(2^m - j)$. Из схемы 2 следует, что для того, чтобы найти значения r_{mj} нужно из начального значения r вычесть столько единиц, сколько их в траектории (T_m) от состояния $r = r_{01}$ до состояния r_{mj}, то есть число $S_{2^m - j}$ и число вида $(\sum_{k=1}^{L_m} n_{p_k - 1\ l_k})$, где L_m — число нулей в траектории (T_m), $\{p_k\}$ —

места нулей в траектории (T_m), а значение l_k вычисляется по (7), как места пересечения траектории (T_m) с уровнем $(p_k - 1)$, то есть по (7) получаем

$$l_k = 2^{p_k - 1} - (T_{p_k - 1})_{10},$$

здесь $(T_i)_{10}$ — десятичный код числа (T_i). Отсюда

$$r_{mj} = r - S_{2^m - j} - \sum_{k=1}^{L_m} n_{p_k - 1\ l_k} \tag{8}$$

где значения $n_{p_k - 1\ l_k}$ вычисляются по формуле (6).

Таким образом, мы получили формулы, с помощью которых можно найти числа исходов $N(r, n)$ в нашей схеме без пустых ячеек, а также число исходов $N^*(r, n)$, если в схеме допускаются пустые ячейки.

Замечание 4. Формула (8) дает общий вид выражения для r_{ij} при замене в ней m на i, а L_m на L_i.

Формулы (3)-(8) дают явное точное выражение для определения искомого числа исходов $N(r, n)$ в частной схеме размещения r неразличимых частиц по n неразличимым ячейкам без пустых ячеек. Тогда число исходов в общей схеме пересчитываются из числа $N(r, n)$ по формуле (1).

Далее рассмотрим, как, собственно, вычисляется число исходов $N(r, n)$ по выведенным выше формулам (3)-(8). Объединим схемы 1 и 2, пересчета параметров r_{ij} и n_{ij} соответственно, в одну общую схему 3. Будем рассматривать ее, как граф переходов из состояния r_{01}, n_{01} в другие возможные по рекурренте (2) состояния. На ребрах графа при левых и правых переходах будем указывать операции пересчета параметров r и n в порядке этих переходов.

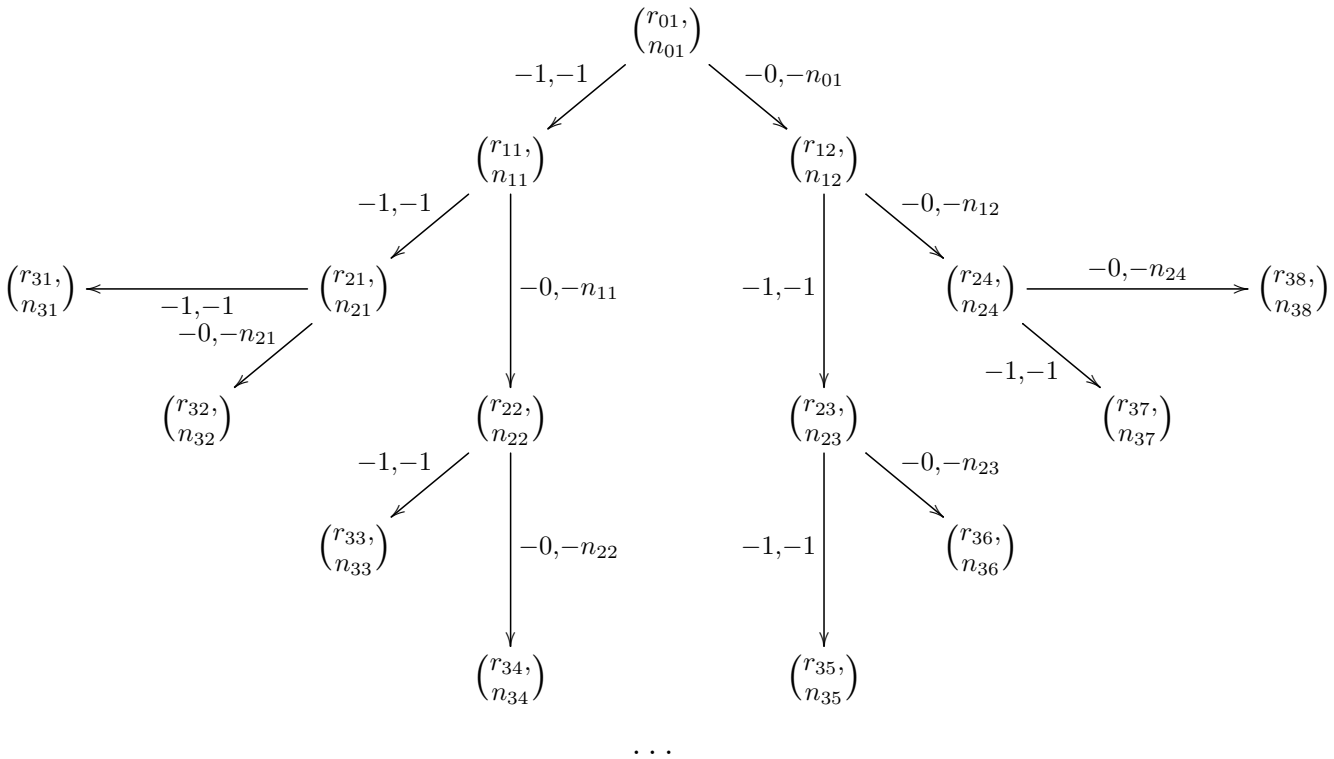


Схема 3

Начнем вычисление числа исходов с нахождения числа m — номера итерации, когда впервые по всем ветвям итераций рекурренты числа N достигают предельных значений по обобщенным свойствам чисел $N(r,n)$ $1a)-4a)$.

Оказывается, значение m можно определить непосредственно из данных r и n. Для этого введем понятие длины траектории в графе без выравнивания длины ветвей рекурренты, как число итераций до достижения параметрами r и n значений, при которых число N принимает предельное значение по свойствам $1)-4)$.

Длина траектории определяется значениями параметров r и n и скоростями их убывания по итерациям до выполнения впервые хотя бы одного из условий:$n=1$ или $r\le n$.

В качестве искомого m нас интересует длина наибольшей траектории. По левым сдвигам оба параметра убывают на любой итерации на единицу, а по правым — только первый параметр r, причем тем медленнее, чем меньше параметр n, то есть $r-n$. С учетом очевидного неравенства $r/n<(r-k)/(n-k), k>0$ при $r>n$ это значит, что траектория максимальной длины должна состоять

сначала из максимального $(n-2)$ левых переходов, не приводящих ее к концу, чтобы максимально уменьшить скорость убывания по итерациям параметра r, и далее подряд — из $[\frac{r-n+2}{2}]$, если число $(r-n+2$ —четно, или из $([\frac{r-n+2}{2}]+1)$, в противном случае, правых сдвигов, что записывается формулой $([\frac{r-(n-2)+1}{2}]-1)$, где $[z]$ — целая часть числа z. Отсюда получаем формулу, по которой из исходных значений r и n можно вычислить максимальную длину m

$$m = n-2+[\frac{r-(n-2)+1}{2}]-1 = n-3+[\frac{r-n+3}{2}] \qquad (9)$$

После нахождения числа mнужно подряд выписать все траектории $(T_m)_2$ от первой до 2^m-ой в виде последовательностей нулей, соответствующих правым сдвигам, и единиц, соответствующих левым сдвигам. По каждой $j-$ой траектории $(j = \overline{1,2^m})$ считается число нулей L_m и их места $\{p_k\}$ $(k = \overline{1,m})$ в траектории и по формулам (6) и (8) вычисляются значения параметров n_{mj} и r_{mj}, после чего, полученные n_{mj} и r_{mj} подставляем в формулу (3) и находим искомое число $N = N(r,n)$ с учетом их свойств 1a)-4a).

Рассмотрим конкретный пример.

Пример 4. Пусть $r = 7, n = 3$. Здесь сначала произведем непосредственный расчет числа N с выравниванием длин траекторий до m по свойствам $1a)-4a)$ по схемам 1,2,3 и по рекурренте (2). По (9) $m = 3-3+[\frac{7-3+3}{2}] = 3$. Тогда имеем

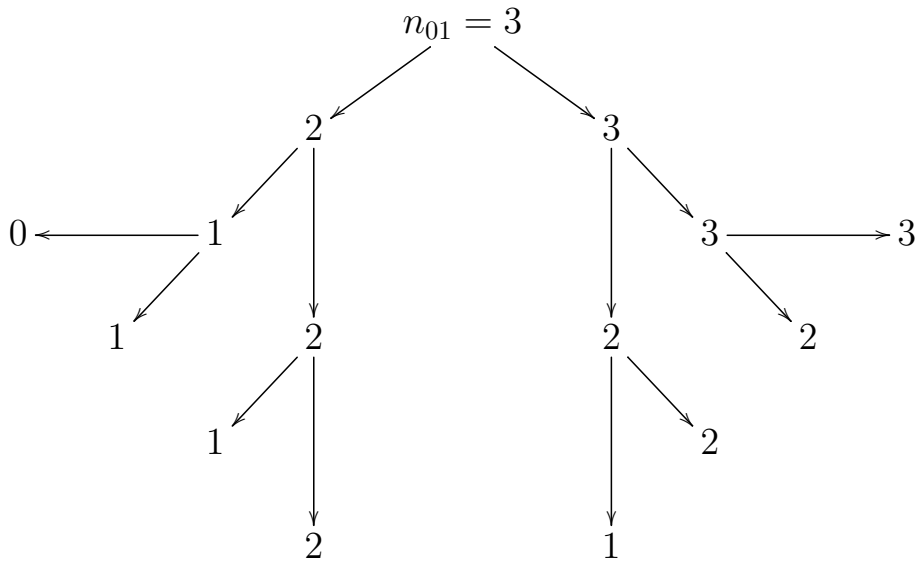


Схема 1

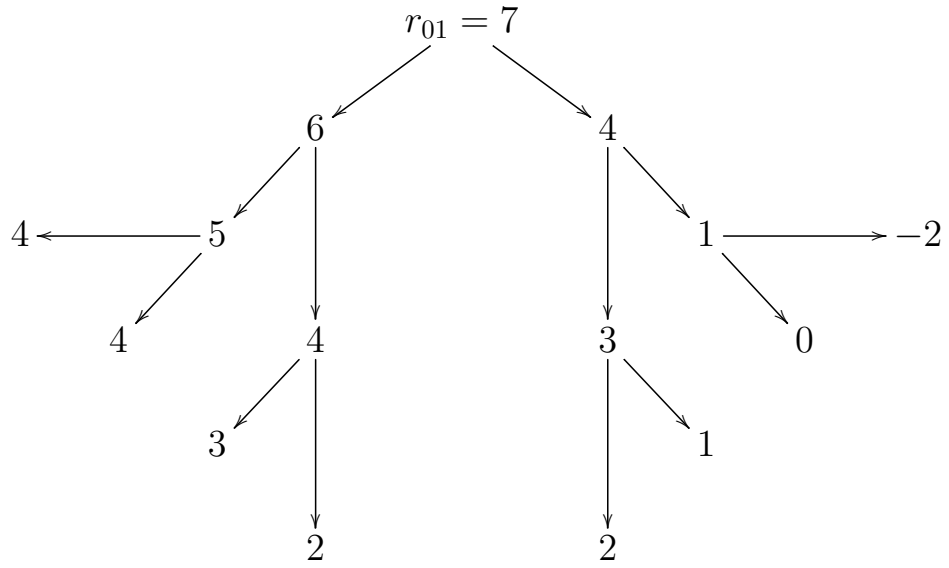


Схема 2

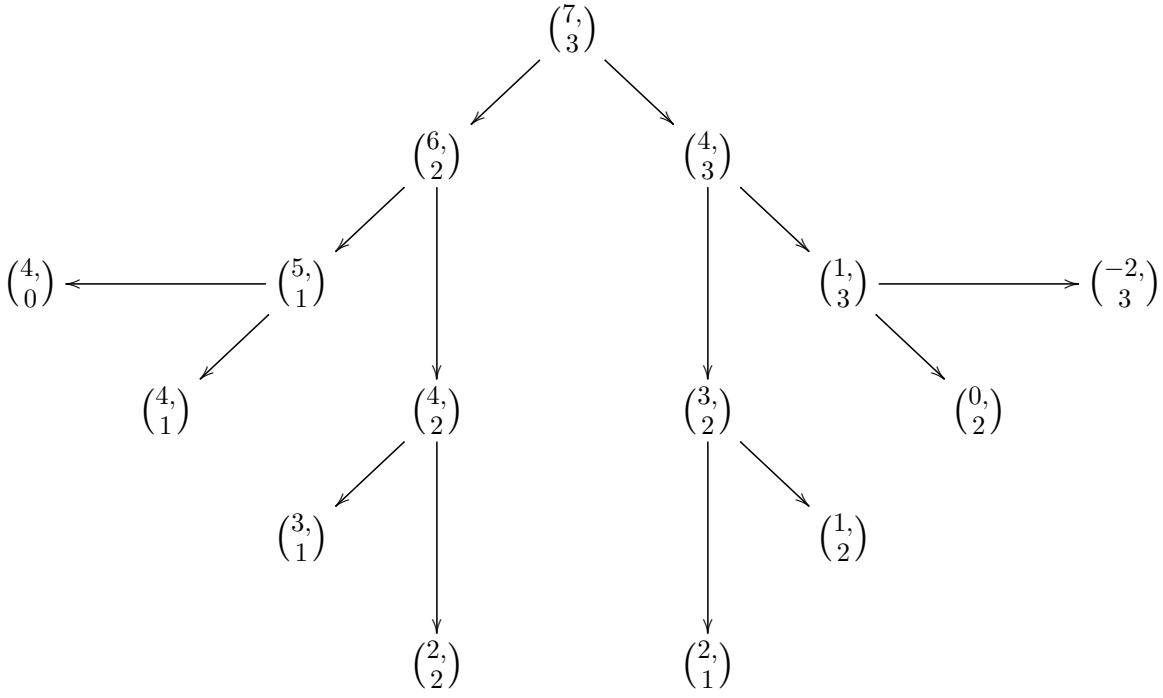


Схема 3

Отсюда получаем траектории $\{111, 110, 101, 100, 011, 010, 001, 000\}$ и по формуле (3) находим число исходов N как сумму чисел определенных свойствами $1a) - 4a)$: $N = 0 + 1 + 1 + 1 + 1 + 0 + 0 + 0 = 4$.

Можно взять те же значения $r = 7$ и $n = 3$ и найти N по рекурренте (2), а после сравнить результаты.

Пример 5. Пусть $r = 7, n = 3$. Тогда по (2) получаем:

$$N(7,3) = N(6,2) + N(4,3) = N(5,1) + N(4,2) + N(3,2) + N(1,3) =$$

$$= 1 + N(3,1) + N(2,2) + N(2,1) + N(1,2) + 0 = 1 + 1 + 1 + 1 + 0 + 0 = 4.$$

Как видим результат один и тот же.

Теперь произведем вычисление искомого числа N в примере по формулам (3)-(9). Начнем с вычисления всех значений $\{n_{ij}\}$ для $i = \overline{1,m} = \overline{1,3}; j = 1, 2^i$ по формуле (7):$\{n_{01}\} = 3; \{n_{1j}\} = \{n_{11} = 3 - s_1 = 2, n_{12} = 3 - s_0 = 3\} = = \{2,3\}; \{n_{2j}\} = \{n_{21} = 3 - s_{11} = 1; n_{22} = 3 - s_{10} = 2; n_{23} = 3 - s_{01} = 2; n_{24} = = 3 - s_{00} = 3\}; \{n_{3j}\} = \{n_{31} = 3 - s_{111} = 0; n_{32} = 3 - s_{110} = 1; n_{33} = 3 - s_{101} = = 1; n_{34} = 3 - s_{100} = 2; n_{35} = 3 - s_{011} = 1; n_{36} = 3 - s_{010} = 2; n_{37} = 3 - s_{001} = = 2; n_{38} = 3 - s_{000} = 3\}$, что совпадает с значениями $\{n_{ij}\}$ в схемах 1 и 3.

Далее по формуле (8) будем вычислять значения $\{r_{3j}\}$, предварительно определив для каждого $j = \overline{1,2^3} = \overline{1,8}$ все входящие в формулу (8) значения такие как $\{j\}, (T_3)_2, \{p_k\}; \{l_k\}$.Представим все расчеты $\{r_{3j}\}$ в виде таблицы с использованием вычисленных выше значений $\{n_{ij}\}$ для $i = 1, 2, 3$.

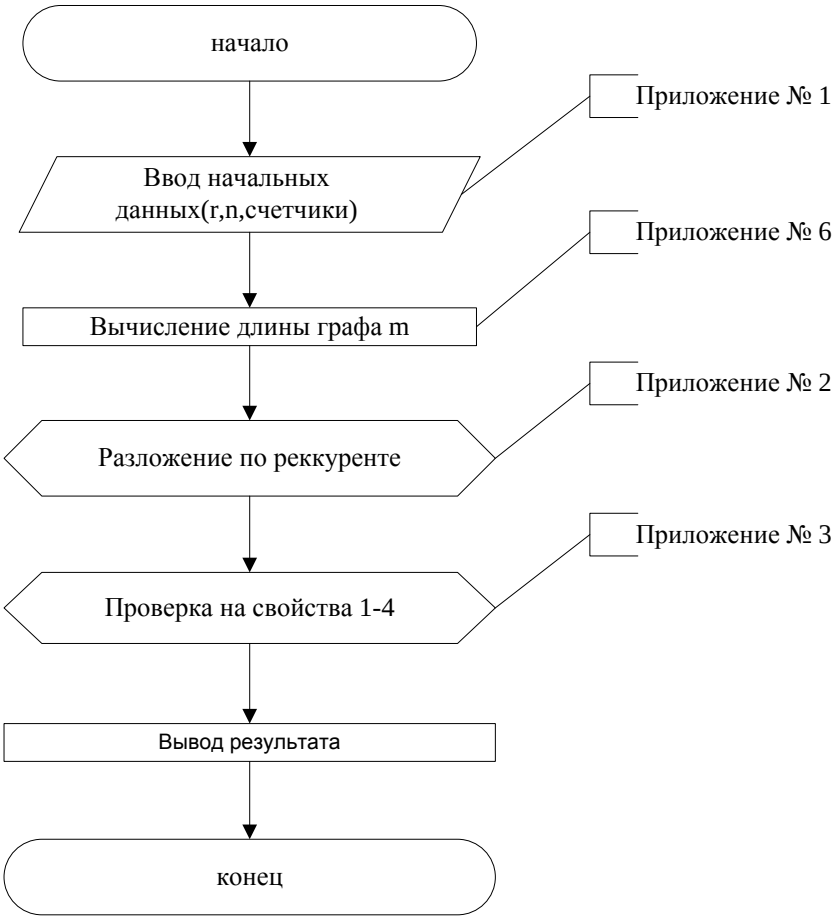
j	$(T_3)_2$	$\{p_k\};\{p_k-1\}$	$l_k=2^{p_k-1}-$ $-(T_{p_k-1})_{10}$	$r_{31}=r-S_{2^m-j}-$ $-\sum_{k=1}^{L_m} n_{p_k-1}l_k$	r_{31}
1	111	—	—	$r-S_{111}=7-3=4$	4
2	110	$p_1=3;$ $p_1-1=\underline{2}$	$2^2-(T_{11})_{10}=$ $=4-3=\underline{1}$	$r-S_{110}-n_{21}=$ $=7-2-1=4$	4
3	101	$p_1=2;$ $p_1-1=\underline{1}$	$2^1-(T_1)_{10}=$ $=2-1=\underline{1}$	$r-S_{101}-n_{11}=$ $=7-2-2=3$	3
4	100	$p_1=2;$ $p_1-1=\underline{1}$ $p_2=3;$ $p_2-1=\underline{2}$	$2^1-(T_1)_{10}=;$ $=2-1=\underline{1}$ $2^2-(T_{10})_{10}=$ $=4-2=\underline{2}$	$r-S_{100}-n_{11}-n_{21}=$ $=$7-1-2-2$=2$	2
5	011	$p_1=1;$ $p_1-1=\underline{0}$	$2^0=\underline{1}$	$r-S_{011}-n_{01}=$ $=7-2-3=2$	2
6	010	$p_1=1;;$ $p_1-1=\underline{0}$ $p_2=3;$ $p_2-1=\underline{2}$	$2^0=\underline{1};$ $2^2-(T_{01})_{10}=\underline{3}$	$r-S_{010}-n_{01}-n_{23}=$ $=7-3-2=2$	2
7	001	$p_1=1;;$ $p_1-1=\underline{0}$ $p_2=2;$ $p_2-1=\underline{1}$	$2^0=\underline{1};$ $2^2-(T_0)_{10}=\underline{2}$	$r-S_{001}-n_{01}-n_{12}=$ $=$7-1-3-3$=0$	0
8	000	$p_1=1;$ $p_1-1=\underline{0};$ $p_2=2;$ $p_2-1=\underline{1}$ $p_3=3;$ $p_3-1=\underline{2}$	$2^0=\underline{1};$ $2^1-(T_0)_{10}=\underline{2};$ $2^2-(T_{00})_{10}=\underline{4}$	$r-S_{000}-n_{01}-n_{12}-n_{24}=$ $=$7-0-3-3-3$=$-2	-2

Получаем $\{r_{31}\}=\{4,4,3,2,2,1,0,-2\}$, что совпадает с непосредственным расчетом $\{r_{3j}\}$ по схемам 2 и 3 и применением рекурренты (2) в примере 6. Далее по (3) и свойствам $1a)-4a)$ для $N(r,n)=N(7,3)$ имеем $N==0+1+1+1+1+0+0+0=4$.

Ниже приведена блок-схема алгоритма нахождения числа $N(r,n)$ по точной

формуле и программа, написанная в пакете *Wolfram Mathematica 8*, по этому алгоритму.

<u>Блок-схема 2</u> (алгоритм нахождения N(r,n) по точной формуле).



Программа 2 (нахождение $N(r,n)$ по точной формуле.)

Для начала в программе вводятся значения r и n из примеров 5 и 6 данного пункта, чтобы убедится в правильности работы программы. После этого возьмем значения побольше, чтобы убедится что программа работает и при более больших r и n.

$Timing[r = 7; n = 3; y = 1;$
$m = IntegerPart[n - 3 + ((r - n + 3)/2)];$
$N = \{\{r, n\}\}; NN = Table[0, \{2\}, \{2\}];$
$NN = \{\{N[[1, 1]] - 1, N[[1, 2]] - 1\}, \{N[[1, 1]] - N[[1, 2]], N[[1, 2]]\}\};$
$N = NN; y = 2; ChisloRazmeshenii = 0;$
$For[v = 1, v <= m - 1, N5 = Table[0, y * 2, 2]; s = 0;$
$For[i = 1, i <= y, If[i > 1, s + +];$
$For[j = 1, j <= 2, NN[[i + s, j]] = N[[i, j]] - 1; j + +];$
$For[j = 1, j <= 2,$
$If[j == 1, NN[[i+1+s, j]] = N[[i, j]]-N[[i, 2]], NN[[i+1+s, j]] =$
$= N[[i, 2]]];$
$j + +];$
$i + +];$
$N = NN; y = y * 2; v + +]$
$For[i = 1, i <= y,$
$If[N[[i, 1]] >= 0 \wedge N[[i, 1]] == N[[i, 2]]||$
$N[[i, 2]] == 1 \wedge N[[i, 1]] > N[[i, 2]], ChisloRazmeshenii + +];$
$i + +];$
Print["Число размещений с ", r, " частицами и ", n, " ячейками = ", ChisloRazmeshenii]]

Как видно результат программы совпадает с результатами примеров 5 и 6, следовательно с данными значениями программа работает правильно.

Обозначения в программе:

r — количество частиц; n — количество ячеек; y, v, i, j, s — счетчики; $Chislo-$
$-Razmeshenii$ — счетчик числа размещений (суммирует все единицы, получен-

ные из свойств 1a)-4a)); N — числа $N(r,n)$, слагаемые рекурренты (2); NN — массив полученных слагаемых рекурренты (2) на каждой итерации.

Результат работы программы: *"Число размещений с 7 частицами и 3 ячейками = 4"*.

Результаты программы на каждой итерации:

{{7,3}} — начальные значения

{{6,2},{4,3}} — слагаемые на первой итерации

{{5,1},{4,2},{3,2},{1,3}} — слагаемые на второй итерации

{{4,0},{4,1},{3,1},{2,2},{2,1},{1,2},{0,2},{-2,3}} — слагаемые на третьей итерации

Число размещений с 7 частицами и 3 ячейками = 4 — вывод ответа

Если сравнить результаты на каждой итерации, полученные в программе, с результатами в схеме 3 примера 6, то можно увидеть полное совпадение. Это означает, что программа работает правильно. Далее, также как и в предыдущем пункте, убедимся в правильности работы программы с более большими значениями. Для удобства сравнения возьмем, как и в предыдущий раз, $r = 44$ и $n = 5$. После окончания работы программы получаем результат:"Число размещений с 44 частицами и 5 ячейками = 1602". Этот результат совпадает с количеством размещений полученных в программе 1, при $r = 44$ и $n = 5$.

Инструкция по использованию программы.

1. Открываем программу в пакете Wolfram Mathematica;
2. в первой строке вводим количество частиц r и количество ячеек n после знака равно;
3. нажимаем комбинацию клавиш Shift-Enter и получаем ответ.

Далее переходим к вероятностному анализу схемы размещения неразличимых частиц по неразличимым ячейкам.

ГЛАВА 3

Вероятностный анализ схемы размещения неразличимых частиц по неразличимым ячейкам

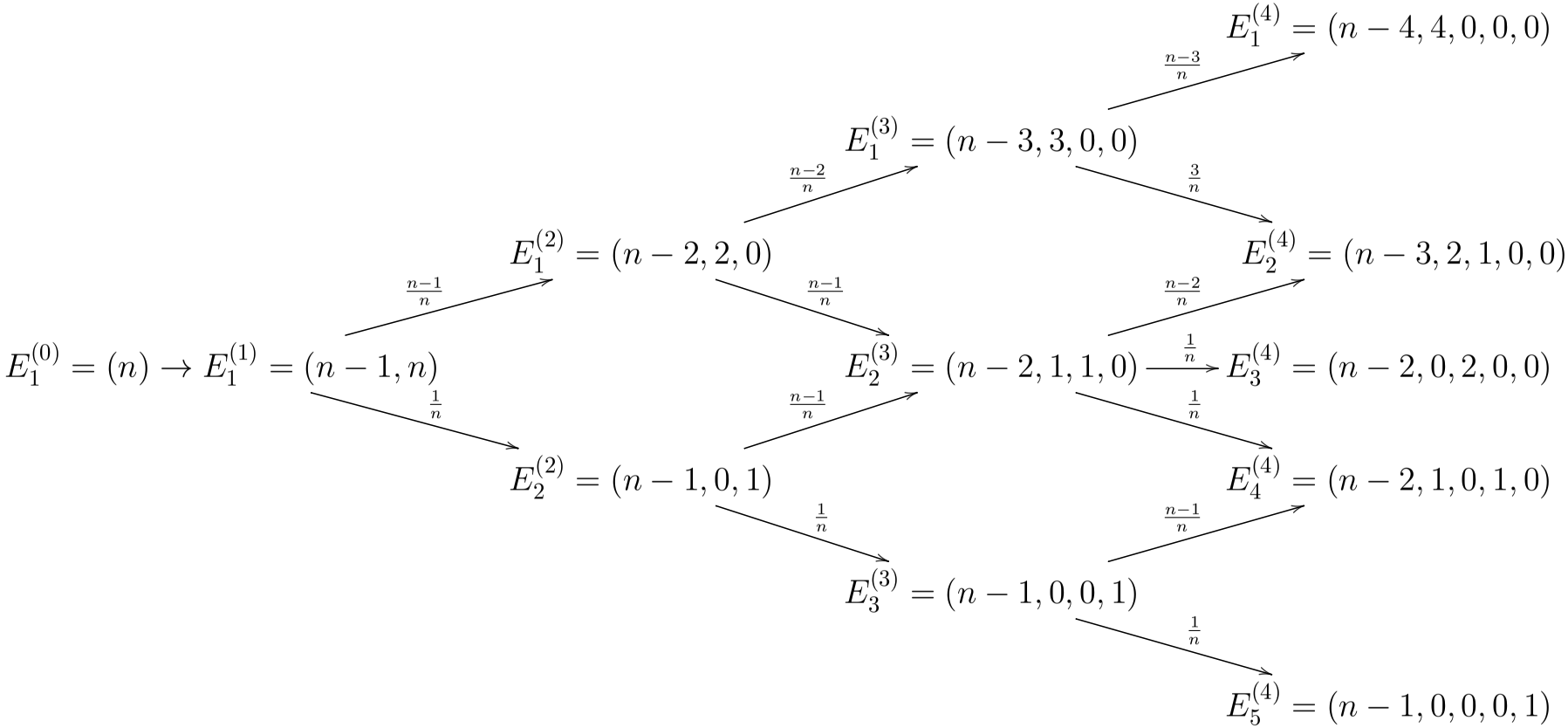
§1. Случайный процесс последовательного поединичного размещения частиц по ячейкам.

Для начало определим новый вектор $\overrightarrow{\eta_i}$, связанный с вектором $\overrightarrow{\mu_i}$, определенный во ведении. Вектор $\overrightarrow{\eta_i}$ — это, собственно, наглядный вид размещение частиц по ячейкам. Напомним, что вектор$\overrightarrow{\mu_i}$ имеет вид $\overrightarrow{\mu_i} = (\mu_0, \mu_1, \mu_2 \dots \mu_r)$, где r — количество частиц. Предположим имеется вектор $\overrightarrow{\mu_5} = (\mu_0 = 0, \mu_1 = 1, \mu_2 = 1, \mu_3 = 1, \mu_4 = 0, \mu_5 = 0)$, он соответствует вектору $\overrightarrow{\eta_5} = (1, 2, 3, 0, 0)$, то есть частицы размещаются именно так, ячейка с одной частицей, с двумя и с тремя, но нет пустой ячейки и ячеек с четырьмя и пятью частицами. Вектор $\overrightarrow{\eta_5}$ это наглядно показывает.

Далее построим случайный процесс, исходы которого на $i-$ом шаге описываются компонентами вектора μ_i. На каждом шаге будем добавлять по одной частицы в каждую из n неразличимую ячейку. Состояниями процесса будем считать численности их заполнений всех возможных уровней в возрастающем порядке после размещения i частиц, представляемые векторами μ_i нулевом начальном заполнении.

Процесс будем строить в виде графа переходов из состояния в состояние, где на каждом шаге будем добавлять по одной частице в каждую ячейку. Обозначать будем как $E_j^{(i)}$ — состояние номера j после размещения i частиц (на i-ом

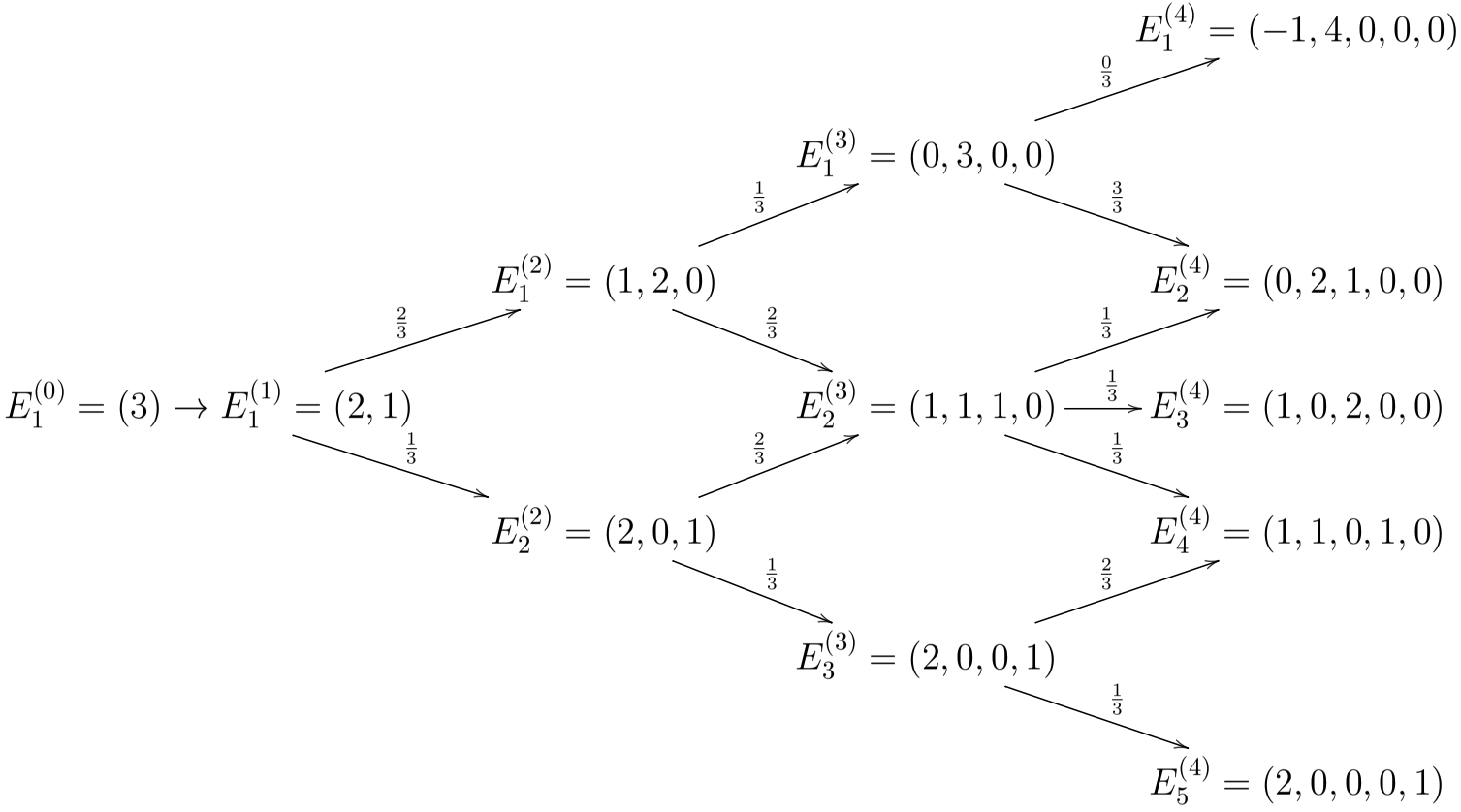
шаге размещения). Состояния $\{E_j^{(i)}\}$ нумеруются в порядке попадания следующей частицы в ячейки с растущем предварительным уровнем заполнения(то есть сначала — в одну из пустых ячеек, потом в одну из ячеек с одной частицей,потом — с двумя частицами и т.д.). Тогда состояния на каждом шаге будут упорядочены по принятому правилу. Вектор, описывающий состояние процесса на $i-$ом шаге состоит из $(i+1)$ компонент, задающих численности ячеек с уровнями заполнений от 0 до i в возрастающем порядке. Начальный вектор $\overrightarrow{\mu_0}$, соответствующий нулевому заполнению $\overrightarrow{\mu_0} = (n)$ и состоит из одной компоненты (все n ячеек пустые). Для наглядности на графе укажем вероятности переходов из состояния в состояние. Представим наглядно этот граф.


$E_1^{(0)} = (n) \to E_1^{(1)} = (n-1, n)$
$\frac{n-1}{n}$
$\frac{1}{n}$
$E_1^{(2)} = (n-2, 2, 0)$
$E_2^{(2)} = (n-1, 0, 1)$
$\frac{n-2}{n}$
$\frac{n-1}{n}$
$\frac{n-1}{n}$
$\frac{1}{n}$
$E_1^{(3)} = (n-3, 3, 0, 0)$
$E_2^{(3)} = (n-2, 1, 1, 0)$
$E_3^{(3)} = (n-1, 0, 0, 1)$
$\frac{n-3}{n}$
$\frac{3}{n}$
$\frac{n-2}{n}$
$\frac{1}{n}$
$\frac{1}{n}$
$\frac{n-1}{n}$
$\frac{1}{n}$
$E_1^{(4)} = (n-4, 4, 0, 0, 0)$
$E_2^{(4)} = (n-3, 2, 1, 0, 0)$
$E_3^{(4)} = (n-2, 0, 2, 0, 0)$
$E_4^{(4)} = (n-2, 1, 0, 1, 0)$
$E_5^{(4)} = (n-1, 0, 0, 0, 1)$

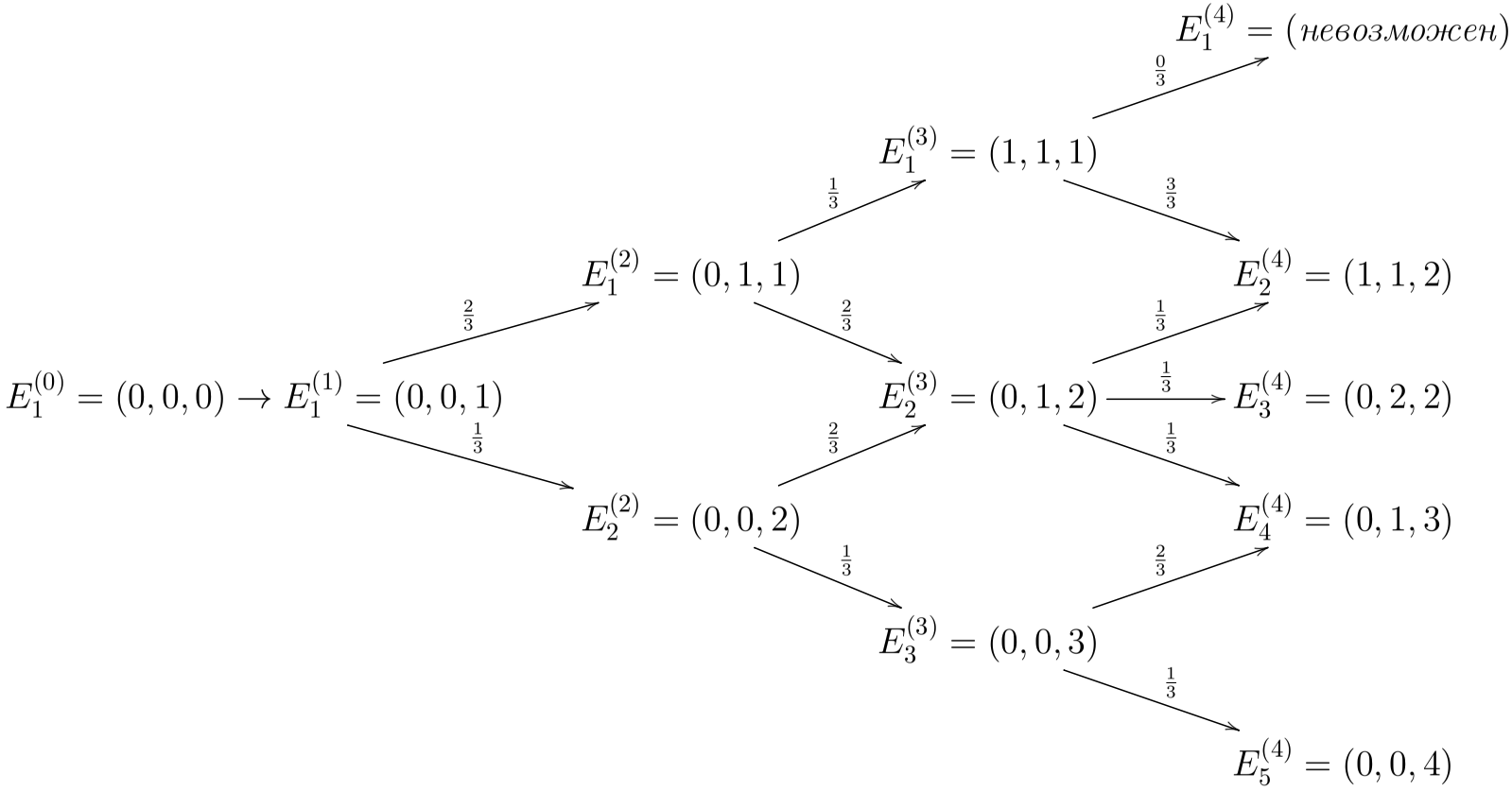
Приведем конкретный пример.

Пример 6.

Возьмем $n = 3$ и $r = 4$. Получаем следующий граф



И построим соответствующий граф, но теперь через вектор $\overrightarrow{\eta}$



По данным значениям r и n мы построили два графа перехода. Первый показывает с каким числом частиц в ячейках возможно размещение, а второй граф непосредственно показывает само размещение. Как можно заметить, переход из состояния $E_1^{(3)}$ в состояние $E_1^{(4)}$ в данном примере происходит с нулевой вероятностью. Это связано с тем, что состояние $E_1^{(3)} = (1, 1, 1)$, то есть все ячейки заполнены, и нет пустой, поэтому мы из состояния $E_1^{(3)}$ переходим только в состояние $E_2^{(4)}$ с вероятностью единицы. Также видно, что если у нас имеются две и более ячеек, заполненных одинаково, следующую частицу мы добавляем лишь в одну из них. Это происходит поскольку мы имеем дело с неразличимыми ячейками и частицами.

Ниже представлена блок-схема алгоритма построения графа случайного процесса последовательного поединичного размещения частиц по ячейкам для получения перечня всех N^* и программа, написанная в пакете *Wolfram Mathematica*, которая реализует данный алгоритм. Она будет основываться на втором графе, то есть она будет выводить ячейки с количеством частиц в них. Для этой программы будем брать 3 ячейки.

<u>Блок-схема 3</u> (построение графа случайного процесса последовательного поединичного размещения частиц по ячейкам для получения всех N*).

Программа 3(построение графа случайного процесса последовательного поединичного размещения частиц по ячейкам для получения перечня всех N^*.)

$Timing[E1 = \{\{0,0,0\}\}; r = 12$

$v = Length[E1]; z = 0; h = 0; p = 0; m = 1;$

$For[q = 1, q <= r, Eb = Table[0, \{1\}];$

$For[i = 1, i <= Length[E1], d = 0; l = Length[E1[[1]]] + 1;$

$For[j = 1, j <= Length[E1[[1]] - 1],$

$If[E1[[i, l-1]]! = E1[[i, l-2]], d = d + 1]; l--,$

$j++];$

$If[d == Length[E1[[1]]] - 1, z = z + 3];$

$If[(E1[[i, m]] == E1[[i, m+1]]||$

$E1[[i, m+1]] == E1[[i, m+2]]||$

$E1[[i, m]] == E1[[i, m+2]])\wedge$

$Not[E1[[i, m]] == E1[[i, m+1]] == E1[[i, m+2]]],$

$\mathrm{h} = \mathrm{h} + 2];$

$If[E1[[i, m]] == E1[[i, m+1]] == E1[[i, m+2]], p = p + 1];$

$i++];$

$Eb = Table[0, \{z + h + p\}]; u = 1;$

$For[i = 1, i <= Length[E1],$

$If[E1[[i, m]]! = E1[[i, m+1]]\wedge$

$E1[[i, m+1]]! = E1[[i, m+2]] \wedge E1[[i, m]]! = E1[[i, m+2]],$

$E2 = Table[0, \{Length[E1[[1]]]\}, \{Length[E1[[1]]]\}];$

$For[j = 0, j <= Length[E1[[1]]] - 1, E1[[i, j]] = E1[[i, j]] + 1;$

$E2[[j]] = E1[[i]]; E1[[i, j]] = E1[[i, j]] - 1,$

$j++]];$

$If[(E1[[i, m]] == E1[[i, m+1]]||$

$E1[[i, m+1]] == E1[[i, m+2]]||$

$E1[[i, m]] == E1[[i, m+2]])\wedge$

$Not[E1[[i, m]] == E1[[i, m+1]] == E1[[i, m+2]]],$

$E2 = Table[0, \{Length[E1[[1]]] - 1\}, \{Length[E1[[1]]]\}];$

$For[j = 1, j <= j <= Length[E1[[1]]] - 1,$

$If[E1[[i, 1]] == E1[[i, 2]], E1[[i, j+1]] = E1[[i, j+1]] + 1;$

$$
\begin{array}{l}
E2[[j]] = E1[[i]]; E1[[i, j+1]] = E1[[i, j+1]] - 1]; \\
If[E1[[i,1]]! = E1[[i,2]], E1[[i,j]] = E1[[i,j]] + 1; \\
E2[[j]] = E1[[i]]; E1[[i,j]] = E1[[i,j]] - 1]; \\
j++]]; \\
If[E1[[i,m]] == E1[[i,m+1]] \wedge \\
E1[[i,m+1]] == E1[[i,m+2]] \wedge E1[[i,m]] == E1[[i,m+2]], \\
E2 = Table[0, 1, \{Length[E1[[1]]]\}]; \\
E1[[i,3]] = E1[[i,3]] + 1; E2[[i]] = E1[[i]]; E1[[i,3]] = -1]; \\
For[j = 1, j <= Length[E2], Eb[[u]] = E2[[j]]; u++; j++]; \\
i++]; \\
E1 = Eb; \\
For[i = 0, i < Length[E1], \\
For[j = Length[E1], j > i, \\
If[j! = i \wedge Sort[E1[[i]]] == Sort[E1[[j]]], E1 = Delete[E1, j]], \\
j--], \\
i++]; \\
For[i = 1, i < Length[E1], E1[[i]] = Sort[E1[[i]]], i++]; \\
Print[E1]; h = 0; z = 0; p = 0, \\
q++]]
\end{array}
$$

Обозначения в программе:

$i, j, v, z, h, p, m, l, d, u, q$ — счетчики; $E1$ — массив с исходами графа, обновляется после каждого шага; Eb, $E2$ — промежуточные массивы с исходами графа; r — общее количество частиц, которые необходимо разместить по ячейкам.

Далее приведем результаты работы программы. Сначала проверим работу на **примере 6**

Результаты программы на каждой итерации (пример 6) :

{{0,0,0}}

{{0,0,1}}

{{0,1,1},{0,0,2}}

{{1,1,1},{0,1,2},{0,0,3}}

{{1,1,2},{0,2,2},{0,1,3},{0,0,4}}

Рассмотрим результаты. Первая строка содержит начальное состояние (размещение первого типа). Во второй мы кладем одну частицу в одну из ячеек, так как $\{\{0,0,0\}\}$ является размещением первого типа (все ячейки заполнены одинаково). Так как $\{\{0,0,1\}\}$ является размещением второго типа, следовательно, в третьей строке получается уже два варианта размещения. Первый из них получился путем добавления одной частицы в одну из пустых ячеек, а второй — добавлением в ячейку с одной частицей. Как можно предположить, в четвертой строке должно быть четыре исхода, так как оба предыдущих размещения относятся ко второму типа, то есть должны разлагаться на два новых исхода каждый, но по факту мы имеем лишь три. Это произошло потому, что в процессе разложения получилось два одинаковых исхода,в следствии чего программа удалила один. В последней строке находятся конечные исходы, в процессе их получения программа удалила повторяющиеся.

Результаты работы этой программы можно сравнить с результатами в **примере 6**, по второму графу. Видим, что результаты идентичны. Далее посмотрим какие результаты даст программы, если мы возьмем $r = 8$.

Результаты программы на каждой итерации при $r = 8$ ***:***

$$\{\{0,0,1\}\}$$
$$\{\{0,1,1\},\{0,0,2\}\}$$
$$\{\{1,1,1\},\{0,1,2\},\{0,0,3\}\}$$
$$\{\{1,1,2\},\{0,2,2\},\{0,1,3\},\{0,0,4\}\}$$
$$\{\{1,2,2\},\{1,1,3\},\{0,2,3\},\{0,1,4\},\{0,0,5\}\}$$
$$\{\{2,2,2\},\{1,2,3\},\{1,1,4\},\{0,3,3\},\{0,2,4\},\{0,1,5\},\{0,0,6\}\}$$
$$\{\{2,2,3\},\{1,3,3\},\{1,2,4\},\{1,1,5\},\{0,3,4\},\{0,2,5\},\{0,1,6\},\{0,0,7\}\}$$
$$\{\{2,3,3\},\{2,2,4\},\{1,3,4\},\{1,2,5\},\{1,1,6\},\{0,4,4\},\{0,3,5\},\{0,2,6\},\{0,1,7\},\{0,0,8\}\}$$

Полученные результаты можно проверить вручную и убедиться, что они верны.

Инструкция по использованию программы.

1. Открываем программу в пакете Wolfram Mathematica;

2. в первой строке вводим начальное состояние $E1$ после знака равно;

3. в r вводим количество итераций, то есть общее количество частиц, которые нужно разместить по ячейкам;

4. нажимаем комбинацию клавиш Shift-Enter и получаем ответ.

Далее рассмотрим распределения вероятностей векторов $\{\overrightarrow{\mu_r}\}$.

§2. Распределение вероятностей векторов $\{\overrightarrow{\mu_r}\}$ и N и N^*.

Для начала покажем программу, которая выводит наш граф, по которому можно определить все траектории до какого либо конечного состояния и вероятности перехода. Данная программа идентична предыдущей, за исключением вывода, поэтому блок-схему приводить нет смысла.

<u>Блок-схема 4</u> (построение графа случайного процесса последовательного поединичного размещения частиц по ячейкам).

Программа 4 (перечисление всех траекторий до данного состояния по графу).

$Timing[E1 = \{\{0,0,0\}\}; r = 4$
$v = Length[E1]; z = 0; h = 0; p = 0; m = 1; ho = Table[0, \{20\}];$
$For[q = 1, q <= r, Eb = Table[0, \{1\}]; h1 = Table[0, \{Length[E1]\}];$
$For[i = 1, i <= Length[E1], d = 0; l = Length[E1[[1]]] + 1;$
$For[j = 1, j <= Length[E1[[1]] - 1],$
$If[E1[[i, l-1]]! = E1[[i, l-2]], d = d + 1]; l--,$
$j++];$
$If[d == Length[E1[[1]]] - 1, z = z + 3;];$
$If[(E1[[i, m]] == E1[[i, m+1]]||$
$E1[[i, m+1]] == E1[[i, m+2]]||E1[[i, m]] == E1[[i, m+2]])\wedge$
$Not[E1[[i, m]] == E1[[i, m+1]] == E1[[i, m+2]]],$
$\mathrm{h} = \mathrm{h} + 2];$
$If[E1[[i, m]] == E1[[i, m+1]] == E1[[i, m+2]], p = p + 1];$
$i++];$
$Eb = Table[0, \{z + h + p\}]; u = 1; Print[Length[Eb]];$
$For[i = 1, i <= Length[E1],$
$If[E1[[i, m]]! = E1[[i, m+1]]\wedge$
$E1[[i, m+1]]! = E1[[i, m+2]] \wedge E1[[i, m]]! = E1[[i, m+2]],$
$E2 = Table[0, \{Length[E1[[1]]]\}, \{Length[E1[[1]]]\}];$
$For[j = 0, j <= Length[E1[[1]]]-1, E1[[i, j]] = E1[[i, j]]+1;$
$E2[[j]] = E1[[i]]; E1[[i, j]] = E1[[i, j]] - 1,$
$j++];$
$h1[[i]] = \{E1[[i]] -> "\frac{1}{3}" -> Sort[E2[[1]]],$
$E1[[i]] -> "\frac{1}{3}" -> Sort[E2[[2]]], E1[[i]] -> "\frac{1}{3}" ->$
$-> Sort[E2[[3]]]\}//MatrixForm];$
$If[(E1[[i, m]] == E1[[i, m+1]]||E1[[i, m+1]] == E1[[i, m+2]]||$
$E1[[i, m]] == E1[[i, m+2]]) \wedge Not[E1[[i, m]] ==$
$==\mathrm{E1[[i,\ m\ +\ 1]]} == \mathrm{E1[[i,\ m\ +\ 2]]}] ,$
$E2 = Table[0, \{Length[E1[[1]]] - 1\}, \{Length[E1[[1]]]\}];$
$For[j = 1, j <= Length[E1[[1]]] - 1,$
$If[E1[i, 1] == E1[[i, 2]], E1[[i, j+1]] = E1[[i, j+1]]+1;$

```
                    E2[[j]] = E1[[i]]; E1[[i, j + 1]] = E1[[i, j + 1]] − 1;
                    h1[[i]] = {E1[[i]] − > "2/3" − > Sort[E2[[1]]],
                    E1[[i]] − > "1/3" − > Sort[E2[[2]]]}//MatrixForm];
                If[E1[[i, 1]]! = E1[[i, 2]], E1[[i, j]] = E1[[i, j]] + 1;
                    E2[[j]] = E1[[i]]; E1[[i, j]] = E1[[i, j]] − 1;
                    h1[[i]] = {E1[[i]] − > "1/3" − > Sort[E2[[1]]],
                    E1[[i]] − > "2/3" − > Sort[E2[[2]]]}//MatrixForm];
            j + +]];
        If[E1[[i, m]] == E1[[i, m+1]] ∧ E1[[i, m+1]] == E1[[i, m+2]] ∧
            ∧E1[[i, m]] == E1[[i, m + 2]],
            E2 = Table[0, {i}, {Length[E1[[1]]]}];
            E1[[i, 3]] = E1[[i, 3]] + 1; E2[[i]] = E1[[i]];
            E1[[i, 3]] = E1[[i, 3]] − 1;
            h1[[i]] = {E1[[i]] − > "1" − > Sort[E2[[i]]]}//MatrixForm];
        For[j = 1, j <= Length[E2], Eb[[u]] = E2[[j]]; u + +; j + +];
    i + +];
    ho[[q − 1]] = h1//MatrixForm; E1 = Eb;
    For[i = 0, i < Length[E1],
        For[j = Length[E1], j > i,
            If[j! = i ∧ Sort[[E1][i]] == Sort[E1[[j]]], E1 = Delete[E1, j]],
        j − −],
    i + +];
    For[i = 1, i < Length[E1], E1[[i]] = Sort[E1[[i]]], i + +];
    h = 0; z = 0; p = 0,
q + +]
]
```

Данная программа делает то же самое, что и программа 2, но выводит не конечные исходы по порядку, а траектории до конечных исходов. Результат для четырех частиц:

Результат работы программы:

$$\left(\{0,0,0\} \to 1 \to \{0,0,1\} \right) \quad \left(\begin{array}{l} \{0,0,1\} \to \frac{2}{3} \to \{0,1,1\} \\ \{0,0,1\} \to \frac{1}{3} \to \{0,0,2\} \end{array} \right) \quad \left(\begin{array}{l} \{0,1,1\} \to \frac{1}{3} \to \{1,1,1\} \\ \{0,1,1\} \to \frac{2}{3} \to \{0,1,2\} \\ \{0,0,2\} \to \frac{2}{3} \to \{0,1,2\} \\ \{0,0,2\} \to \frac{1}{3} \to \{0,0,3\} \end{array} \right) \quad \left(\begin{array}{l} \{1,1,1\} \to 1 \to \{1,1,2\} \\ \{0,1,2\} \to \frac{1}{3} \to \{1,1,2\} \\ \{0,1,2\} \to \frac{1}{3} \to \{0,2,2\} \\ \{0,1,2\} \to \frac{1}{3} \to \{0,1,3\} \\ \{0,0,3\} \to \frac{2}{3} \to \{0,1,3\} \\ \{0,0,3\} \to \frac{1}{3} \to \{0,0,4\} \end{array} \right)$$

Вывод данной программы происходит таким образом, чтобы можно было проследить траектории из начального состояния в конечное наглядно. Например из состояние (1,1,2) в начальное можно попасть, с учетом единственности начального состояния (ищем траектории справа налево по обратным стрелкам в графе и перебираем их сверху вниз на каждом шаге), по следующим траекториям:

1ая траектория: $(1,1,2) \to (1,1,1) \to (0,1,1) \to (0,0,1) \to (0,0,0)$;
2ая траектория: $(1,1,2) \to (0,1,2) \to (0,1,1) \to (0,0,1) \to (0,0,0)$;
3я траектория: $(1,1,2) \to (0,1,2) \to (0,0,2) \to (0,0,1) \to (0,0,0)$.

Между стрелками в выводе результата стоят вероятности перехода из состояния в состояние, по которым мы будем вычислять распределение вероятностей, а этот граф поможет нам проверять результаты наглядно.

Инструкция по использованию программы.

1. Открываем программу в пакете Wolfram Mathematica;
2. в первой строке вводим начальное состояние $E1$ после знака равно;
3. в r вводим количество итераций, то есть общее количество частиц, которые нужно разместить по ячейкам;
4. нажимаем комбинацию клавиш Shift-Enter;
5. наводим курсор на строку $graf$, нажимаем комбинацию клавиш Shift-Enter и получаем ответ.

Как вычисляются числа N и N^* по данным графа, и как из полученных вероятностей перехода и траекторий получить распределение вероятностей? Ответы описаны ниже. Начнем с вычисление N и N^*.

Как видно, в графе на r-ом шаге мы получили всевозможные размещения. Количество размещений r неразличимых частиц по n неразличимым ячейкам на r-ом шаге и есть $N^* = N^*(r,n)$ — число исходов в общей схеме размещений. Число исходов $N = N(r,n)$ в той же схеме без пустых ячеек можно получить, отбраковывая из N^* состояний общей схемы состояния с нулевой первой компонентой числа пустых ячеек. То есть имеем численный метод вычисления чисел N и N^* путем построения графа состояний случайного процесса последовательного размещения частиц до $r-$ого шага.

Распределение вероятностей векторов $\overrightarrow{\mu_r}$ вычисляется по графу (п.1, схема 4) путем сложения по всем траекториям графа до данного состояния вектора $\overrightarrow{\mu_r}$ произведений вероятностей последовательных переходов указанных на ребрах графа.

Пример 7.Пусть $r = 4$. Тогда, например,

$P(E_3^{(4)}) = P(\overrightarrow{\mu_4} = (n-4,4,0,0,0)) = \frac{1}{n} \cdot \frac{n-1}{n} \cdot \frac{1}{n} + \frac{n-1}{n} \cdot \frac{n-1}{n} \cdot \frac{1}{n}$;
$P(E_2^{(4)}) = P(\overrightarrow{\mu_4} = (n-3,2,1,0,0)) = \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \frac{3}{n} + \frac{n-1}{n} \cdot \frac{2}{n} \cdot \frac{n-2}{n}$ и т.д. для всех четырех состояний.

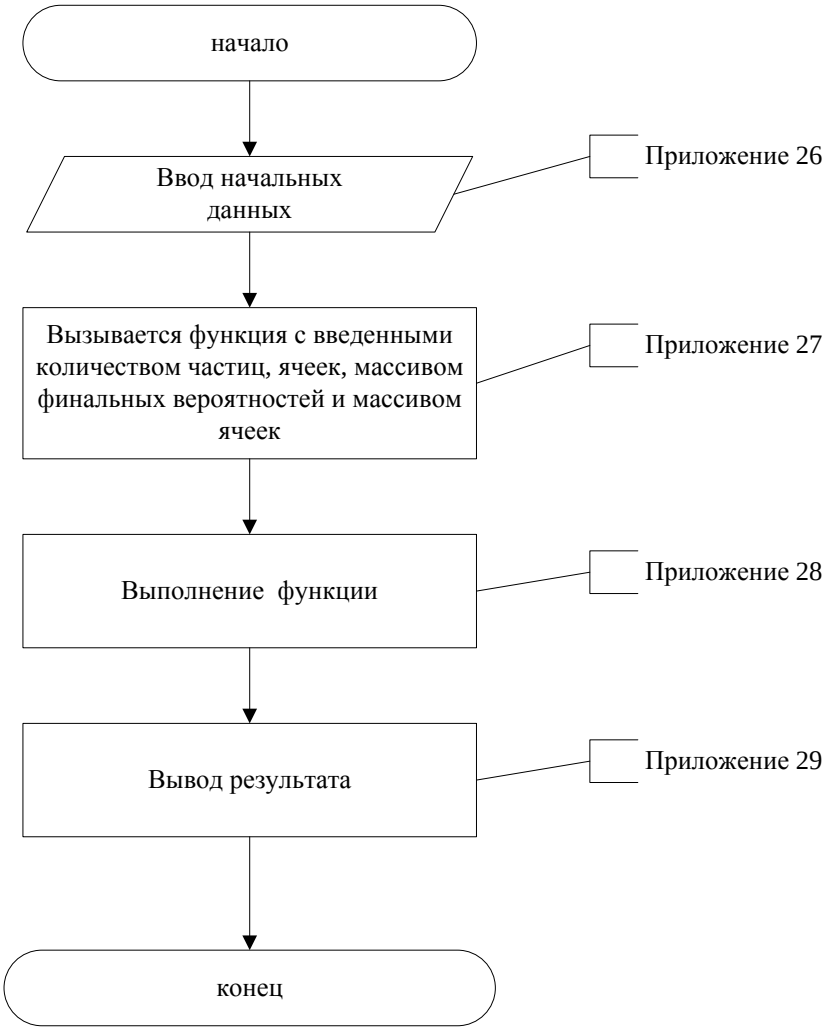
Поскольку дальше мы будем рассматривать программу подсчитывающую данные вероятности через граф 4, но через вектор $\overrightarrow{\eta_r}$, заменим в примере 7 вектор $\overrightarrow{\mu_r}$ на вектор $\overrightarrow{\eta_r}$:

Пример 7.Пусть $r = 4$. Тогда, например,

$P(E_3^{(4)}) = P(\overrightarrow{\eta_4} = (0,2,2)) = \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{1}{3} + \frac{2}{3} \cdot \frac{2}{3} \cdot \frac{1}{3} = \frac{2}{9} \approx 0,22$;
$P(E_2^{(4)}) = P(\overrightarrow{\eta_4} = (1,1,2)) = \frac{3}{3} \cdot \frac{1}{3} \cdot \frac{2}{3} + \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{2}{3} + \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{1}{3} = \frac{4}{9} \approx 0,44$;
и т.д. для всех четырех состояний.

Ниже представлена блок-схема алгоритма вычисления вероятностного распределения всех N исходов по траекториям графа и программа, написанная на языке программирования $C++$, осуществляющая данный алгоритм. Она считает $P(E_n^{(r)})$.

<u>Блок-схема 5</u> (алгоритм вычисления вероятностного распределения всех N исходов по траекториям граф).



Программа 5 (вычисление вероятностного распределения всех N исходов по траекториям графа).

```
include"stdafx.h"
include < stdlib.h >
include < math.h >
void gen(int num_cell, int part, int count, float prob, int* cells, float *final)
{
    if(count == part)
        return;
    int i = 0, j = 0, k;
    while(i < num_cell)
    {
        j = cells[i];
        k = 1;
        while(cells[i + 1] == j && i + 1 < num_cell)
        {
            k++;
            i++;
        }
        int *new_cells = new int[num_cell];
        for(int z = 0; z < num_cell; z++)
        {
            new_cells[z] = cells[z];
        }
        new_cells[i]++;
        float new_prob = prob * k/num_cell;
        int pointer = 0;
        for(int z = num_cell - 1, s = 1; z >= 0; z--, s*= 10)
        {
            pointer+= new_cells[z] * s;
        }
        final[pointer]+= new_prob;
        gen(num_cell, part, count + 1, new_prob, new_cells, final);
```

```
            free(new_cells);
            i++;
        }
}
int main()
{
    int num_cell, part;
    //get num_cell, part
    printf("Vvedite kolichestvo ycheek :");
    scanf_s("%d", &num_cell);
    printf("\n Vvedite kolichestvo chastiz :");
    scanf_s("%d", &part);
    printf("P(000) = 0");
    int *cells = new int[num_cell];
    int finals = pow(10.0, num_cell + 1);
    float *final = new float[finals];
    for(int z = 0; z < num_cell; z++)
    {
        cells[z] = 0;
    }
    for(int z = 0; z < finals; z++)
    {
        final[z] = 0;
    }
    gen(num_cell, part, 0, 1.0, cells, final);
    free(cells);
    //result
    for(int z = 0; z < finals; z++)
    {
        if(final[z] != 0)
            printf("\nP(0...%d) = %f", z, final[z]);
    }
    free(final);
```

$scanf\_s("\%d", \&part);$

$return\ 0;$

}

Обозначения в программе:

$i, k, j, z,$ — счетчики; num_cell —количество ячеек; $part$ — количество частиц; $count$ —счетчик ; $prob$ — вероятность перехода ; $final$ — конечные вероятности; $cells$ — ячейки; $pointer$ — счетчик.

Данная программа на каждой итерации также забрасывает в ячейку одну частицу, получая необходимые состояния. Одновременно с получением следующего состояния, программа считает вероятность перехода из предыдущего состояния в данное. При подсчете следующей вероятности учитывается предыдущая вероятность (то есть вероятность перехода в следующее состояние умножается на вероятность перехода в предыдущее состояние). Получив конечные состояния, программа складывает вероятности перехода в эти состояния и получает ответ.

Результат работы программы:

Следующий результат получен с использованием значений примера 7.

Введите количество ячеек : 3

Введите количество частиц : 4

$$P(0,0,4) = 0.037037$$

$$P(0,1,3) = 0.296296$$

$$P(0,2,2) = 0.222222$$

$$P(1,1,2) = 0.444444$$

Инструкция по использованию программы.

1. Открываем программу в Visual Studio;

2. запускаем программу, нажав на F5;

3. вводим количество ячеек, жмем Enter;

4. вводим количество частиц, жмем Enter;

5. получаем ответ;

6. для выхода из консольного окна набираем любую комбинацию символов и жмем Enter.

ГЛАВА 4

Исследование стохастической модели схемы

§1. Моделирование схемы размещения r неразличимых частиц по n неразличимым ячейкам.

Первый способ. В предыдущей программе мы нашли дискретное распределение вероятностей всех состояний на любом шаге. Значит по полученному распределению на r-ом шаге мы можем провести моделирование состояний методом маркировки, описанному в учебном пособии Н.Ю.Энатской и Е.Р.Хаккимуллина "Стохастическое моделирование" в 1 главе §4.

Замечание 5. Если при $r \geq n$ требуется смоделировать аналогичную схему без пустых ячеек, в графе перечисленных состояний $r-$ом шаге в (в схеме 4) исключаем состояния с нулевой компонентой μ_0 в векторе $\overrightarrow{\mu_r}$, а вероятности остальных состояний делим на единицу минус суммарная вероятность состояний с нулевой компонентой, а далее применяем метод маркировки.

Метод маркировки.

Правило. Для того чтобы разыграть дискретную случайную величину X, заданную законом распределения

X	x_1	x_2	$\ldots$	x_t
P	p_1	p_2	$\ldots$	p_t

надо, имея значения базовой случайной величины БСВ, с.в. $R \sim R[0,1]$: $s_1, s_2, \ldots$

1. разбить интервал $(0,1)$ оси Ox на t частичных интервалов: $\triangle_1 \ -(0; p_1)$, $\triangle_2 \ -(p_1; p_1+p_2), \ldots \ \triangle_t \ -(p_1+p_2+\ldots p_{t-1}; 1)$;

2. выбрать (например, из таблицы случайных чисел) случайное число s_i. Если s_i попало в частичный интервал $\triangle_i$, то разыгрываемая случайная величина X приняло возможное значение x_i.

Приведем пример использования метода маркировки конкретно в нашем случае. Возьмем конечные состояния и вероятности, полученные в программе 4 при $r = 4$ и $n = 3$.

Пример 8.

Начальные значения приведем в виде таблицы

X	{1,1,2}	{0,2,2}	{0,1,3}	{0,0,4}
P	0.037	0.222	0.296	0.444

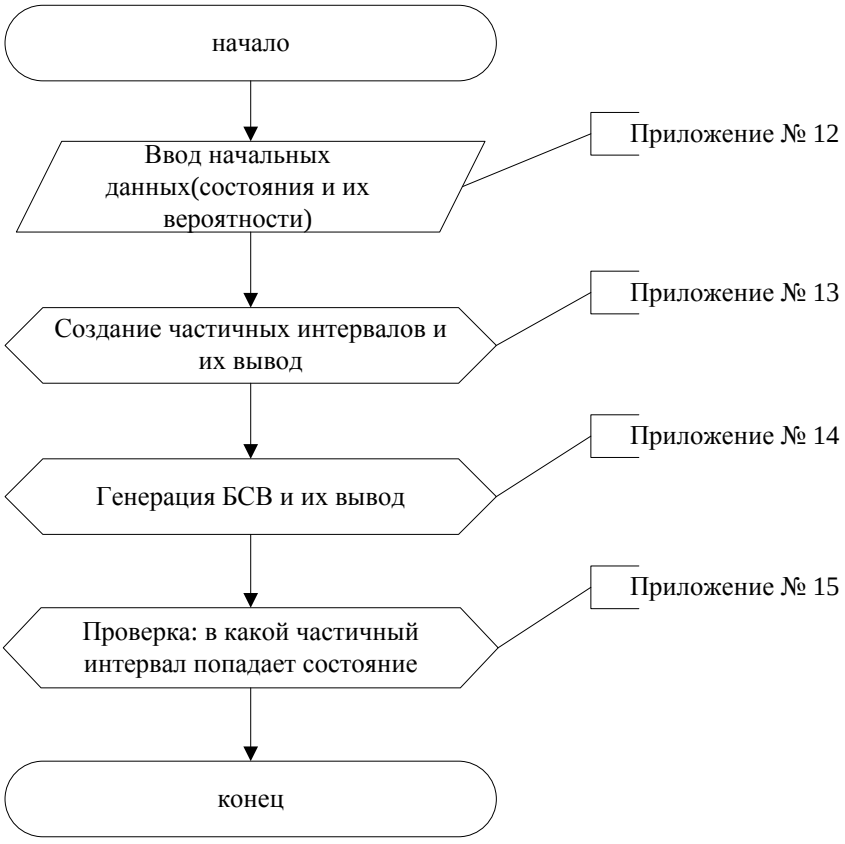
Разобьем интервал $(0, 1)$ на четыре частичных интервала так, как описано выше. Получаем: $\triangle_1 - (0; 0.037), \triangle_2 - (0.037, 0.259), \triangle_3 - (0.259, 0.555), \triangle_4 - (0.555, 1)$.

Далее возьмем шесть случайных чисел, например $0.304; 0.99; 0.347; 0.094; 0.327; 0.826$.

Случайное число $s_1 = 0.304$ принадлежит частичному интервалу $(0.259, 0.555)$, поэтому разыгрываемая случайная величина приняла возможное значение $x_1 = \{0, 1, 3\}$; $s_2 = 0.99 \in (0.555, 1) \Rightarrow x_2 = \{0, 0, 4\}$. Аналогично получаем остальные возможные значения. Получаем следующие разыгранные возможные значения: $\{0, 1, 3\}; \{0, 0, 4\}; \{0, 1, 3\}; \{0, 2, 2\}; \{0, 1, 3\}; \{0, 0, 4\}$.

Ниже представлена блок-схема алгоритма моделирование исходов схемы методом маркировки и программа, написанная на языке *JAVA*, реализующая данный алгоритм.

<u>Блок-схема 6(1)</u> (алгоритм моделирования исходов схемы методом маркировки).



Программа 6(1) (моделирование исходов схемы методом маркировки).

Программа выводит результат для конечных состояний, указанных в примере 8. Конечный результат может отличаться от результата, полученного в примере 8, так как каждый раз генерируются новые случайные величины s_i.

```
public class first {
    public static void main(String[] args) {
        String massivIsxodov[]={"{1,1,2}","{0,2,2}","{0,1,3}","{0,0,4}"};
        double massivVeroyatnostei[]={0.037037,0.222222,0.296296,0.444444};
        double massivBSV[]=new double[10];
        int p=massivVeroyatnostei.length+1;
        double KonciIntervalov[]=new double[p];
        KonciIntervalov[0]=0;
        for(int i=1;i<=massivVeroyatnostei.length;i++)
            KonciIntervalov[i] = KonciIntervalov[i − 1]+
            +massivVeroyatnostei[i − 1];
            System.out.print("(" + KonciIntervalov[i − 1] + ","+
            +KonciIntervalov[i] + ")" + ";");
        System.out.println();
        for(int i=0;i<=massivBSV.length-1;i++)
            {massivBSV[i] = (double)(Math.random() ∗ 1);
            System.out.print(massivBSV[i] + ";"); }
        System.out.println();
        for(inti = 0; i ≤ massivBSV.length − 1; i + +)
        {
            for(intj = 1; j ≤ massivVeroyatnostei.length; j + +)
            {
                if(KonciIntervalov[j − 1] ≤ massivIsxodov[i]&&
                massivIsxodov[i] ≤ KonciIntervalov[j])
                System.out.print(massivIsxodov[j-1]+";");
            }
        }}}
```

Обозначения в программе:

$massivIsxodov[]$ — массив конечных состояний; $massivVeroyatnostei[]$ — массив вероятностей полученных в предыдущей программе; $massivBSV[]$ — массив случайных чисел на отрезке $(0, 1)$; $KonciIntervalov[]$ — массив, содержащий концы отрезков; p, i, j — счетчики.

Результат работы программы:

$$(0.0, 0.037); (0.037, 0.259); (0.259, 0.555); (0.555, 1);$$
$$0.304; 0.99; 0.347; 0.094; 0.327; 0.826;$$
$$\{0, 1, 3\}; \{0, 0, 4\}; \{0, 1, 3\}; \{0, 2, 2\}; \{0, 1, 3\}; \{0, 0, 4\}$$

Рассмотрим вывод подробнее. В первой строке программа вывела полученные частичные интервалы. Во второй — случайные числа в интервале $(0, 1)$. Третья строка содержит результат — разыгранные возможные значения.

Далее посмотрим, что мы получим в результате работы программы, если возьмем конечные значения, полученные в **программе 4** при $r = 10$ и $n = 3$, при этом сгенерировав 10 случайных чисел s_i:

X	P
{3,3,4}	0.000051
{2,4,4}	0.001016
{2,3,5}	0.004572
{2,2,6}	0.012193
{1,4,5}	0.021338
{1,3,6}	0.012803
{1,2,7}	0.004572
{1,1,8}	0.036580
{0,5,5}	0.085353
{0,4,6}	0.128029
{0,3,7}	0.064015
{0,2,8}	0.256059
{0,1,9}	0.160037
{0,0,10}	0.213382

Результат программы:

1. (0.0,0.000051);(0.000051,0.001);(0.001,0.006);(0.006,0.018);(0.018,0.039);
   (0.039,0.052);(0.052,0.057);(0.057,0.093);(0.093,0.178);(0.178,0.306);
   (0.306,0.371);(0.371,0.627);(0.627,0.787);(0.787,1);
2. 0.743;0.995;0.822;0.021;0.949;0.378;0.738;0.27;0.579;0.045;
3. (0,1,9);(0,0,10);(0,0,10);(1,4,5);(0,0,10);(0,2,8);(0,1,9);(0,4,6);(0,2,8);(1,3,6)

Полученный результат при желании можно проверить вручную. Здесь первый пункт — полученные частичные интервалы, второй — 10 случайных чисел s_i, третий — разыгранные возможные значения.

Инструкция по использованию программы.

1. Открываем программу в Eclipse;
2. запускаем программу нажатием кнопки Run;
3. получаем ответ.

Второй способ. Известным из [15,гл.3,§1] способом моделирования заданное количество M_1 размещений r неразличимых частиц по n различимым ячейкам, выбираем из них M_2 повторяющихся, остальные маркируем по уровням заполнения в возрастающем порядке, отбраковывая одинаковые. Получаем L_1 исходов вида вектора $\overrightarrow{\mu_r}$. С отбракованными M_2 размещениями повторяем ту же процедуру, что и с исходными M_1 размещениями, отбраковывая из них M_3 повторяющихся, получаем L_2 исходов того же вида. Далее, заменяя M_i на M_{i+1}, а L_i на $L_{i+1}, i = 2, 3, \ldots$ повторяем то же k раз до тех пор, пока среди отбракованных вариантов не будет повторяющихся. Тогда, объединяя исходы вида вектора $\overrightarrow{\mu_r}$, получаем $L = \sum_{j=1}^{k} L_j$ исходов требуемой схемы.

Замечание 6. Если при $r \geq n$ требуется смоделировать исходы аналогичной схемы без пустых ячеек, то известным из [15,гл.3,§1] способом моделируем M_1 размещений r неразличимых частиц по n различимым ячейкам без пустых ячеек и применяем к ним второй способ.

Ниже представлена блок-схема алгоритма моделирования исходов схемы вторым способом, после которой рассмотрена программа по данному алгоритму, написанная в пакете *Wolfram Mathematica*, реализующая данный способ моделирования на компьютере.

<u>Блок-схема 6(2)</u> (алгоритм моделирования исходов схемы вторым способом).

Программа 6(2) (моделирование исходов схемы вторым способом.)

$n = 6; s = 8; M1 = 180; a = 10; Nomer = Table[0, n + s - 1];$
$MassivNomRk = Table[0, \{M1\}]; MassivNomRkpoR = Table[0, \{M1\}];$
$For[l = 0, l \leq M1 - 1, r = RandomReal[1, (n + s - 1)]; rk = r; rk = Sort[rk];$
$For[i = 0, i \leq (n + s - 2),$
$For[j = 0, j \leq (n + s - 2), If[r[[i]] == rk[[j]],$
$Nomer[[i]] = j],$
$j + +],$
$i + +];$
$MassivNomRkpoR[[l]] = Nomer; MassivNomR[[l]] =$
$= Sort[MassivNomRkpoR[[l]]],$
$l + +]$
$VibrElem = Table[0, \{Length[MassivNomRkpoR]\}, \{k\}];$
$For[i = 0, i \leq Length[MassivNomRkpoR] - 1,$
$b = Total[RandomInteger[1, 3]];$
$For[j = 0, j \leq s - 1, VibrElem[[i, j]] =$
$= MassivNomRkpoR[[i, j + b]],$
$j + +];$
$VibrElem[[i]] = Sort[VibrElem[[i]]],$
$i + +];$
$L = Table[0, \{a\}];$
$For[k = 0, k <= 10, e = 0; q = 1;$
$For[i = 0, i \leq Length[VibrElem] - 2,$
$For[j = q, j \leq Length[VibrElem] - 1,$
$If[VibrElem[[i]] == VibrElem[[j]], e = e + 1],$
$j + +]; q + +,$
$i + +];$
$M2 = Table[0, \{e\}]; q = 1; f = 1;$
$For[i = 0, i \leq Length[VibrElem] - 2,$
$For[j = q, j \leq Length[VibrElem] - 1,$
$If[VibrElem[[i]] == VibrElem[[j]],$
$M2[[f]] = VibrElem[[j]]; f + +],$
$j + +]; q + +,$

$i++];$

$For[i = 0, i \leq Length[VibrElem] - 2,$

$For[j = q, j \leq Length[VibrElem] - 1,$

$If[VibrElem[[i]] == VibrElem[[j]], e = e + 1;$

$VibrElem = Delete[VibrElem, j]],$

$j++]; q++,$

$i++];$

$NomSkrainNul =$

$= Table[0, \{Length[VibrElem]\}, \{Length[MassivNomR[[1]]]\}];$

$For[i = 0, i \leq Length[VibrElem] - 1,$

$For[j = 0, j \leq Length[VibrElem[[1]]] - 1,$

$NomSkrainNul[[i, VibrElem[[i, j]]]] = VibrElem[[i, j]],$

$j++],$

$i++];$

$NachNuli = Table[0,\ Length[NomSkrainNul]\}];$

$KonechNuli = Table[0, \{Length[NomSkrainNul]\}]; d = 0; c = 0;$

$For[i = 0, i \leq Length[NomSkrainNul] - 1,$

$For[j = 0, j \leq Length[NomSkrainNul[[1]]] - 1,$

$If[NomSkrainNul[[i, j]] == 0, d = d + 1, Break[]],$

$j++];$

$For[j = Length[NomSkrainNul[[1]]] + 1, j >= 0,$

$If[NomSkrainNul[[i, j]] == 0, c = c + 1, Break[]],$

$j--];$

$NachNuli[[i]] = d; KonechNuli[[i]] = c; d = 0; c = 0,$

$i++];$

$NomBezKrainNul = Table[0, \{Length[NomSkrainNul]\}];$

$For[i = 0, i \leq Length[NomSkrainNul] - 1,$

$Massiv = Table[0, \{Length[NomSkrainNul[[1]]] - NachNuli[[i]] -$
$KonechNuli[[i]]\}];$

$NomBezKrainNul[[i]] = Massiv,$

$i++];$

$For[i = 0, i \leq Length[NomSkrainNul] - 1,$

$For[j = 0, j <= Length[NomSkrainNul[[i]]] - NachNuli[[i]]-$

```
        - KonechNuli[[i]] - 1,
            NomBezKrainNul[[i, j]] =
            = NomSkrainNul[[i, j + NachNuli[[i]]]],
        j++],
    i++];
Ind = Table[0, {Length[NomBezKrainNul]}];
Rez = Table[-1, {Length[NomBezKrainNul]}, {n + k - 1}]; t = 0;
For[i = 0, i <= Length[NomBezKrainNul] - 1,
        For[j = 0, j <= Length[NomBezKrainNul[[i]]] - 1,
            If[NomBezKrainNul[[i, j]] != 0 && j <=
            <= Length[NomBezKrainNul[[i]]],
            t = t + 1, Rez[[i, p]] = t; t = 0; p++],
        j++];
        v = 1; t = 0; Ind[[i]] = p; p = 1,
    i++];
t = 1;
For[i = 0, i <= Length[NomBezKrainNul] - 1,
        For[j = Length[NomBezKrainNul[[i]]], j >= 0,
            If[NomBezKrainNul[[i, j]] != 0, t = t + 1, Break[]],
        j--];
        Rez[[i, Ind[[i]]]] = t; t = 1,
    i++];
For[i = 0, i <= Length[NomBezKrainNul] - 1,
        For[j = Length[Rez[[i]]], j >= 0,
            If[Rez[[i, j]] == -1, Rez[[i]] = Delete[Rez[[i]], j]],
        j--],
    i++];
For[i = 0, i <= Length[NomBezKrainNul] - 1,
        For[j = 0, j <= KonechNuli[[i]] - 1, Rez[[i]] = Append[Rez[[i]], 0],
        j++];
        For[j = 0, j <= NachNuli[[i]] - 1, Rez[[i]] = Prepend[Rez[[i]], 0],
        j++],
    i++];
```

$For[i = 0, i \leq Length[Rez] - 2,$
$For[j = q, j \leq Length[Rez] - 1,$
$If[Rez[[i]] = Rez[[j]], Rez = Delete[Rez, j]],$
$j++]; q++,$
$i++];$
$L[[w]] = Length[Rez]; VibrElem = M2; If[M2 == \{\}, Break[]],$
$k++]$
$KonechnoeL = \sum_{i=1}^{a} L[[i]]$

Обозначения в программе:

n — количество ячеек; s — количество частиц; $M1$ — количество размещений; $M2$ — повторяющиеся размещения; $a, i, j, e, b, q, k, v, p$ — счетчики; t — счетчик подряд идущих, вошедших или не вошедших в номера исходной выборки, номеров неравных 0; d — количество крайних нулей слева; c — количество крайних нулей справа; r — сгенерированные значения БСВ $(r_1, \ldots, r_{M1})$; rk — вариационный ряд; $Nomer$ — промежуточный массив номеров элементов значений БСВ; $MassivNomR$ — массив номеров элементов значений БСВ; $MassivNomRkpoR$ — массив номеров элементов исходной выборки r, принадлежащих множеству rk; $ViborElem$ — выбранные числа; L — массив L_i; $NomSkrainNul$ — массив выбранных чисел с крайними не выбранными; $NomBezkr$ массив выбранных чисел без крайних невыбранных; $NachNuli$ — количество крайних невыбранных чисел слева; $KonechNuli$ — количество крайних невыбранных чисел справа; $Massiv$ — промежуточный массив; Ind — массив счетчиков с количеством подряд идущих ненулевых элементов; Rez — массив исходов вида $\overrightarrow{\mu_r}$; $KonechnoeL$ — результирующее L.

Данная программа выводит конечное L, объединяющее в себе количества исходов каждого M_i. Результат получен для $n = 6$ и $k = 8$.

Результат работы программы:

$Konechnoe\ L = 16$

Инструкция по использованию программы.

1. Открываем программу Wolfram Mathematica;
2. вводим исходные данные: количество ячеек n и количество частиц s;
3. вводим количество $M1$ вариантов выбора разных номеров элементов выборки r;
4. запускаем программу нажатием комбинаций клавиш Shift-Enter;
5. получаем ответ.

§2. Приближенное вычисление чисел $N^* = N^*(r,n)$ и $N = N(r,n)$ методом стохастического моделирования.

Воспользуемся способом 2 (п.1) моделирования размещений соответственно в общей схеме, допускающее пустые ячейки с общим числом исходов N^* и в схеме без пустых ячеек с общим числом исходов N. Тогда можно считать, что $N^*/C^r_{n+r-1} \approx L/M_1$, откуда получаем $N^* \simeq LC^r_{n+r-1}/M_1$, а $N/C^{n-1}_{r-1} \approx L/M_1$, по замечанию 6. Тогда $N \approx LC^{n-1}_{r-1}/M_1$.

Ниже представлена небольшая программа, написанная в пакете *Wolfram Mathematica*, являющаяся логическим продолжением предыдущей программы и считающая приближенные значения чисел N и N^*.

Программа 7 (приближенное вычисление чисел N и N^*.)

$Print[$"N *s pustimi yacheikami*"$]$;
$NsPustimi = IntegerPart[KonechnoeL * ((n+k-1)!/(k! * (n-1)!))/M1]$
$Print[$"N *bez pustix yacheek*"$]$;
$NbezPustix = IntegerPart[KonechnoeL * ((k-1)!/((n-1)! * (k-n)!))/M1]$

Обозначения в программе:

$M1$ — заданное количество размещений r неразличимых частиц по неразличимым ячейкам; $NsPustimi$ — приближенное значение общего числа исходов в схеме, допускающей пустые ячейки; $NbezPustix$ — приближенное значение общего числа исходов в схеме без пустых ячеек.

Ниже представлен результат работы программы для $n = 6$ и $k = 8$.

Результат работы программы:

$N\ s\ pustimi\ yacheikami = 1470$

$N\ bez\ pustix\ yacheek = 24$

Инструкция по использованию программы.

1. Открываем программу Wolfram Mathematica;
2. запускаем программу нажатием комбинаций клавиш Shift-Enter;
3. получаем ответ.

Заключение

Рассмотрена схема размещения неразличимых частиц по неразличимым ячейкам. Проанализировав теоретические основы, были разработаны алгоритмы вычисления числа исходов N по рекурренте, и по изученным закономерностям работы рекурренты построен алгоритм числа исходов по точной формуле. По данным алгоритмам написаны и отлажены стандартные программы, реализующие их на компьютере. С помощью этих программ стало возможно вычисление числа исходов при большом количестве частиц и ячеек без трудоемких вычислений. Также был разработан алгоритм построения графа последовательного поединичного размещения частиц по ячейкам для получения перечня всех N^*. По этому алгоритму написана программа, облегчающая нахождение исходов для большого количества частиц, и программа, перечисляющая все траектории до данного состояния по полученному графу. Получив все траектории, стало возможным вычисление вероятностного распределения всех N исходов по ним. Вследствие чего, был разработан алгоритм вычисления вероятностного распределения, и написана программа, с помощью которой за несколько секунд и без громоздких вычислений можно получить эти распределения для всех N исходов. Получив эти распределения, используя алгоритм моделирования методом маркировки, была реализована программа, которая моделирует состояния схемы данным методом. После этого, по алгоритму моделирования состояний схемы вторым способом, описанному в дипломной работе, написана программа, реализующая этот алгоритм на компьютере и вычисляющая L исходов требуемой схемы за короткое время. Найдя L, мы вычисляем приближенные значения N^* и N методом, использующим стохастическое моделирование.

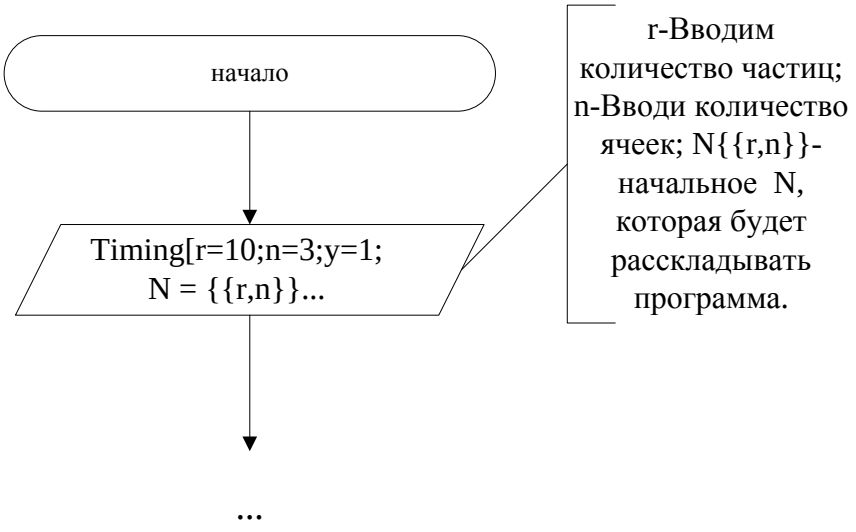
Все эти алгоритмы и программы способны облегчить решение многих задач, связанных с нашей схемой. Эта схема размещения неразличимых частиц по неразличимым ячейкам несет в себе научную новизну и является большим вкладом в фундаментальную теорию.

Библиографический список.

1. Андерсон Д. "Дискретная математика и комбинаторика".
2. Виленкин Н.Я. "Популярная комбинаторика".
3. Гульден Я., Джексон Д."Перечислительная комбинаторика ".
4. Ерош И.Л. "Дискретная математика. Комбинаторика".
5. Иванов В.А.,Теребулин С.Ю. "Некоторые предельные теоремы в неравновероятной схеме размещения частиц комплектами".
6. Колчин В.Ф., Севастьянов Б.А. "Случайные размещения".
7. Журнал им. А.Н.Колмогорова "Теория вероятностей и ее применение" выпуски 1964-2005гг.
8. Кофман А. "Введение в прикладную комбинаторику ".
9. Липский В. "Комбинаторика для программистов".
10. Носов В.А. "Комбинаторика и теория графов".
11. Плотников А.Д. "Дискретная математика".
12. Риордан Дж."Введение в комбинаторный анализ".
13. Стенли Р. "Перечислительная Комбинаторика ".
14. Холл М. "Комбинаторика".
15. Энатская Н.Ю., Хакимуллин Е.Р. " Стохастическое моделирование".
16. Энатская Н.Ю., Хакимуллин Е.Р. "Анализ схемы размещения неразличимых частиц по неразличимым ячейкам («Дискретная математика»)".

Приложение №1

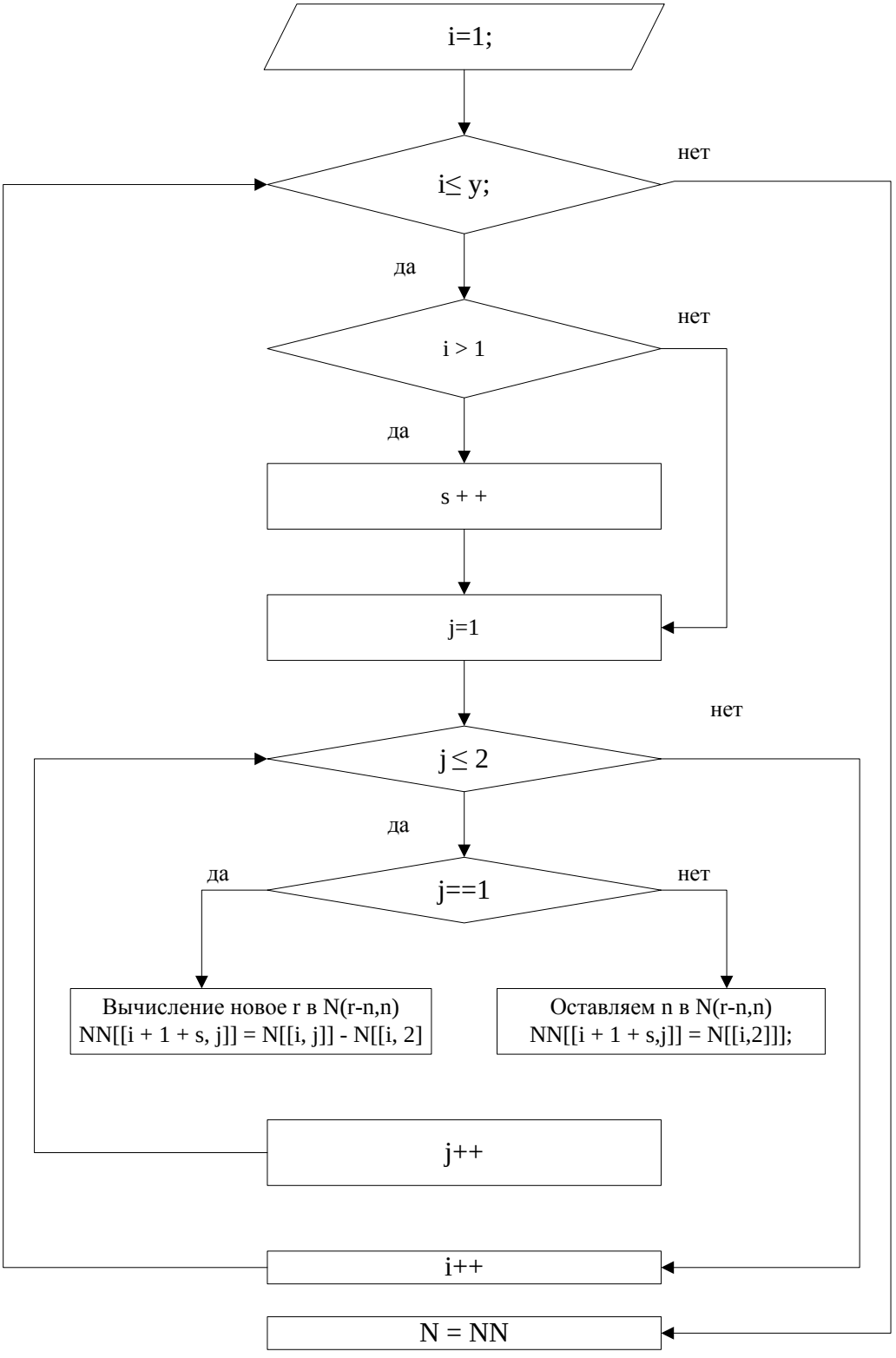
Ввод начальных данных для рекурренты и точной формулы.



...

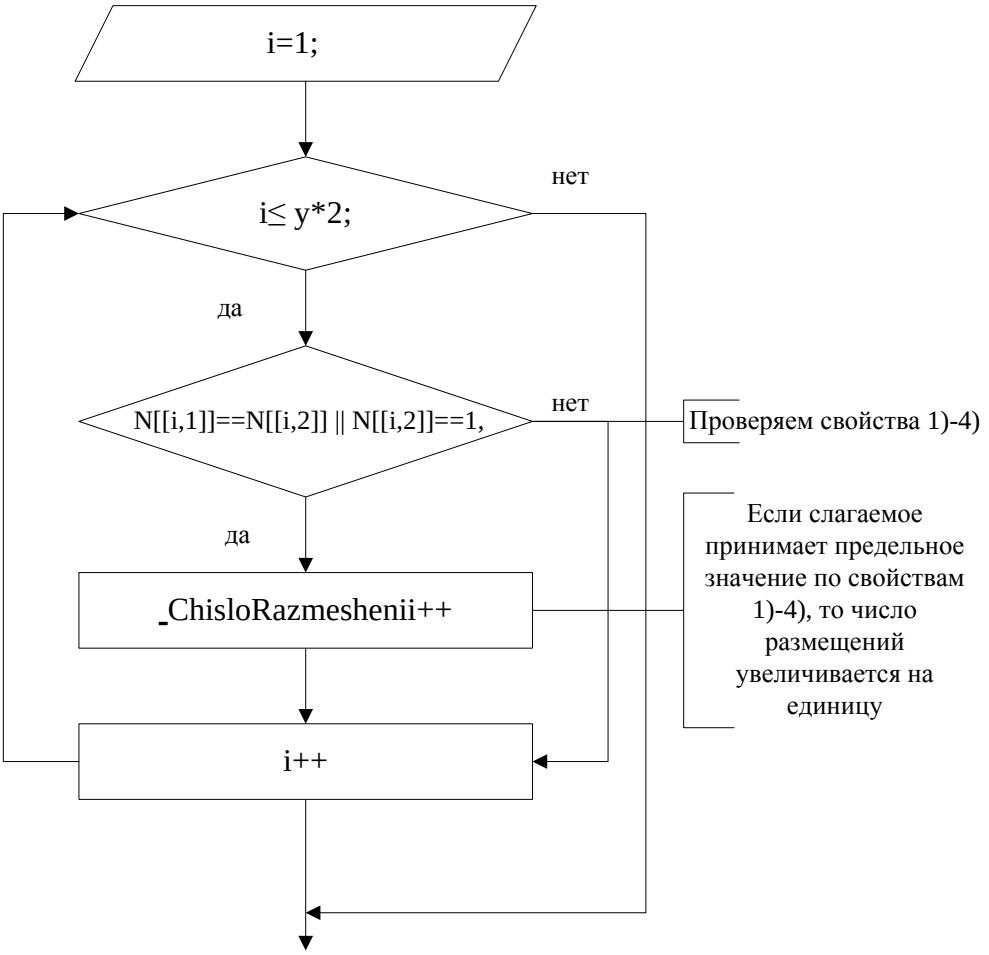
Приложение №2

Алгоритм разложение слагаемых по рекурренте.



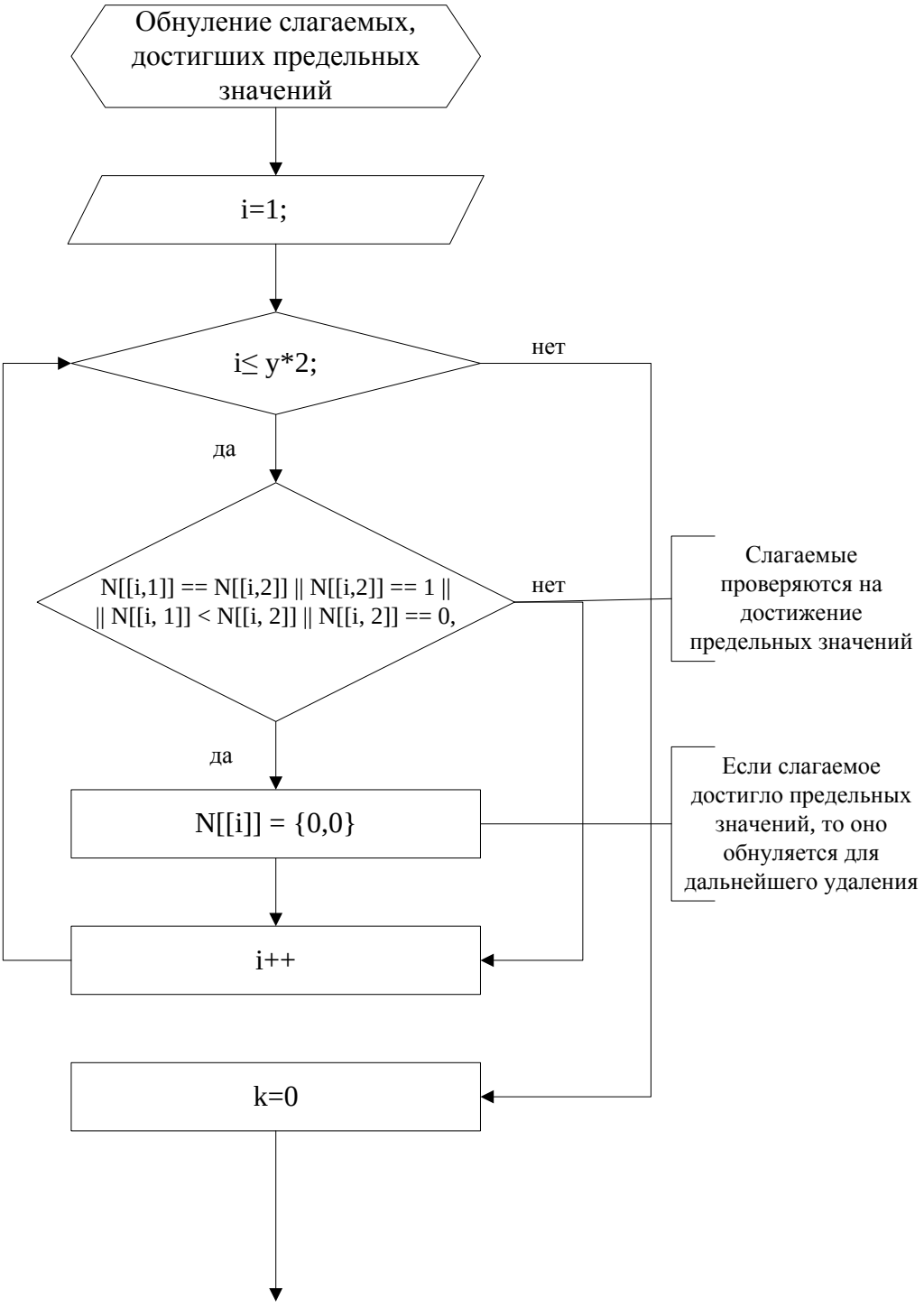
Приложение №3

Проверка на свойства 1)-4).

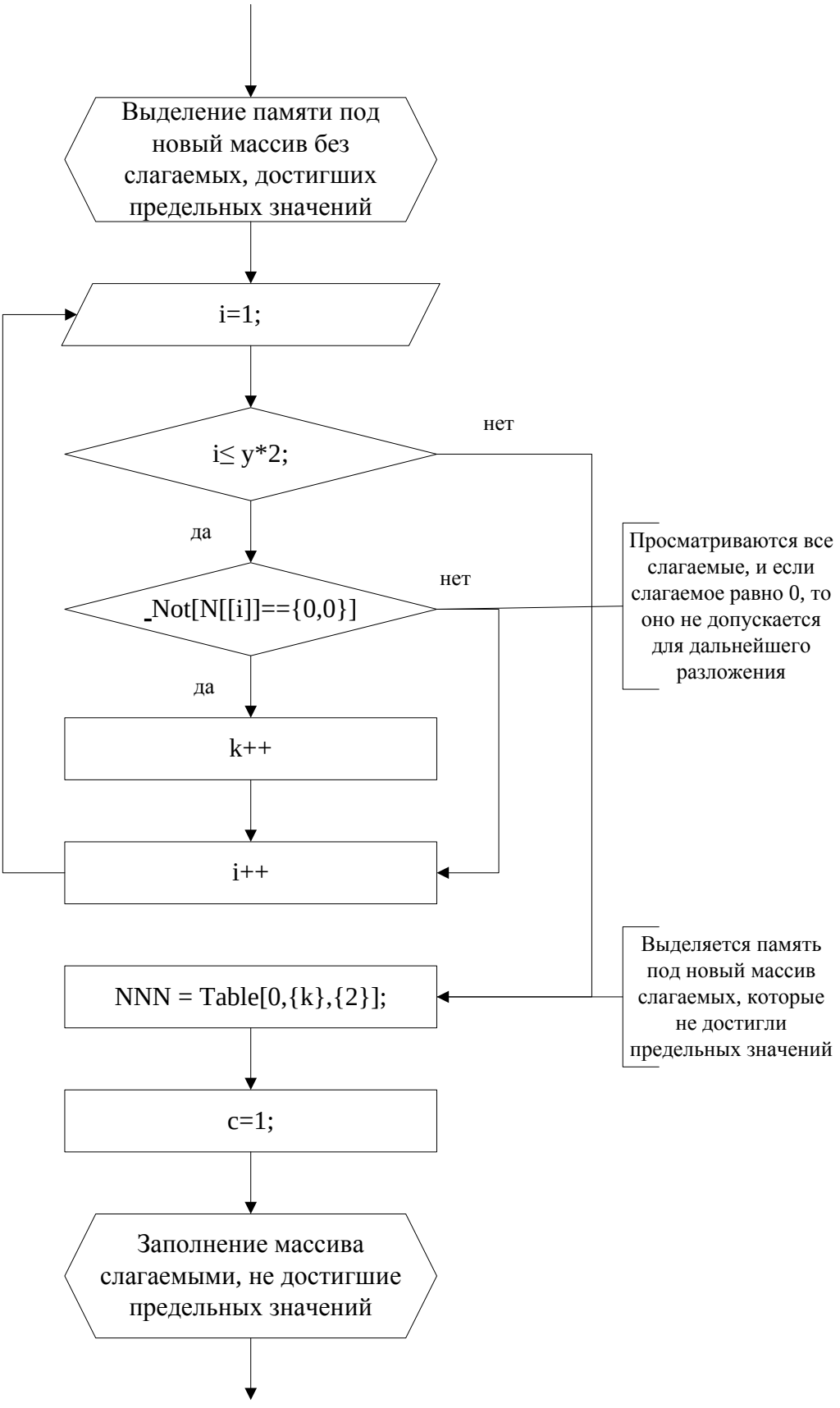


Приложение №4

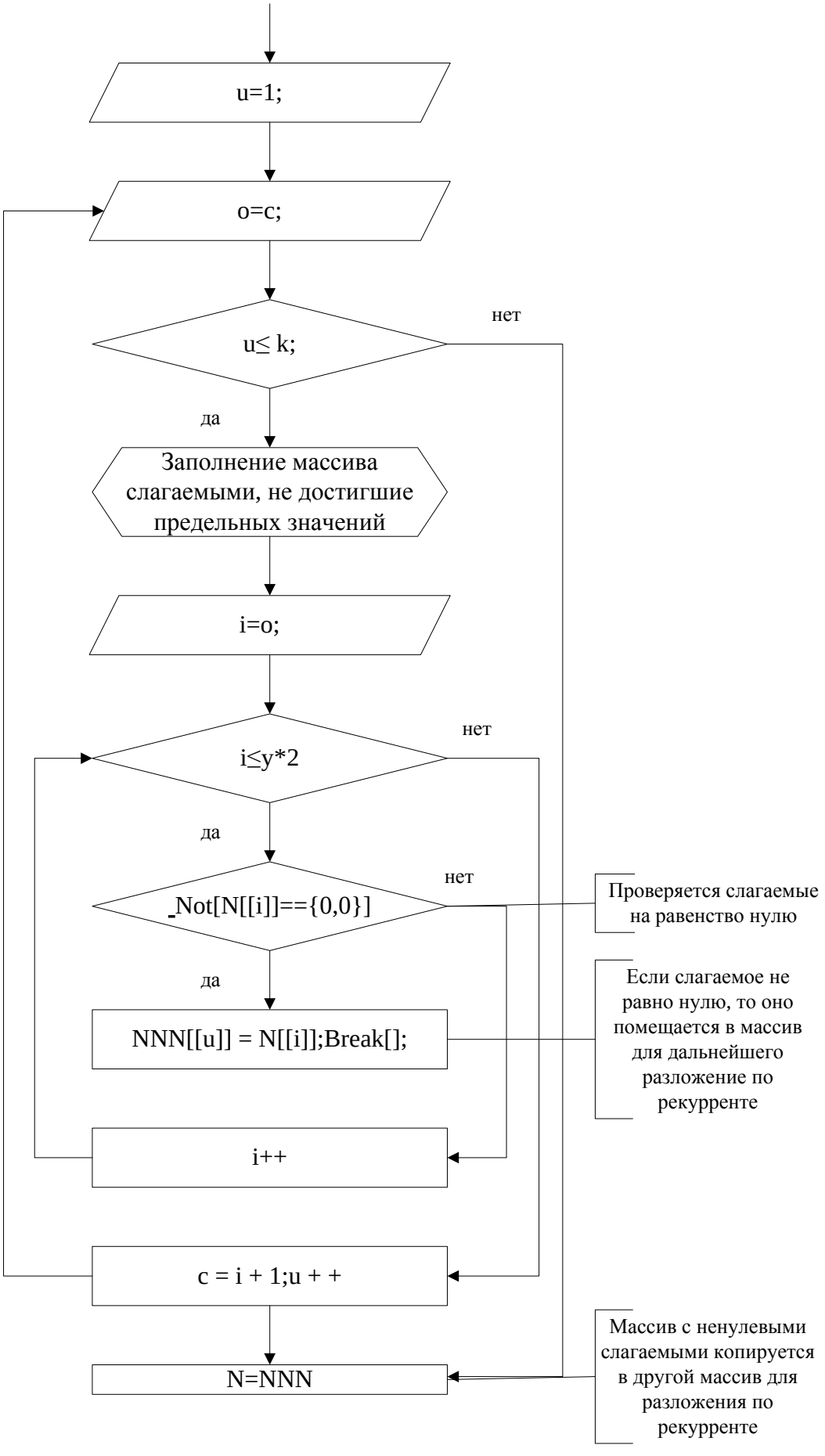
Удаление слагаемых, достигших предельных значений.



Приложение №4

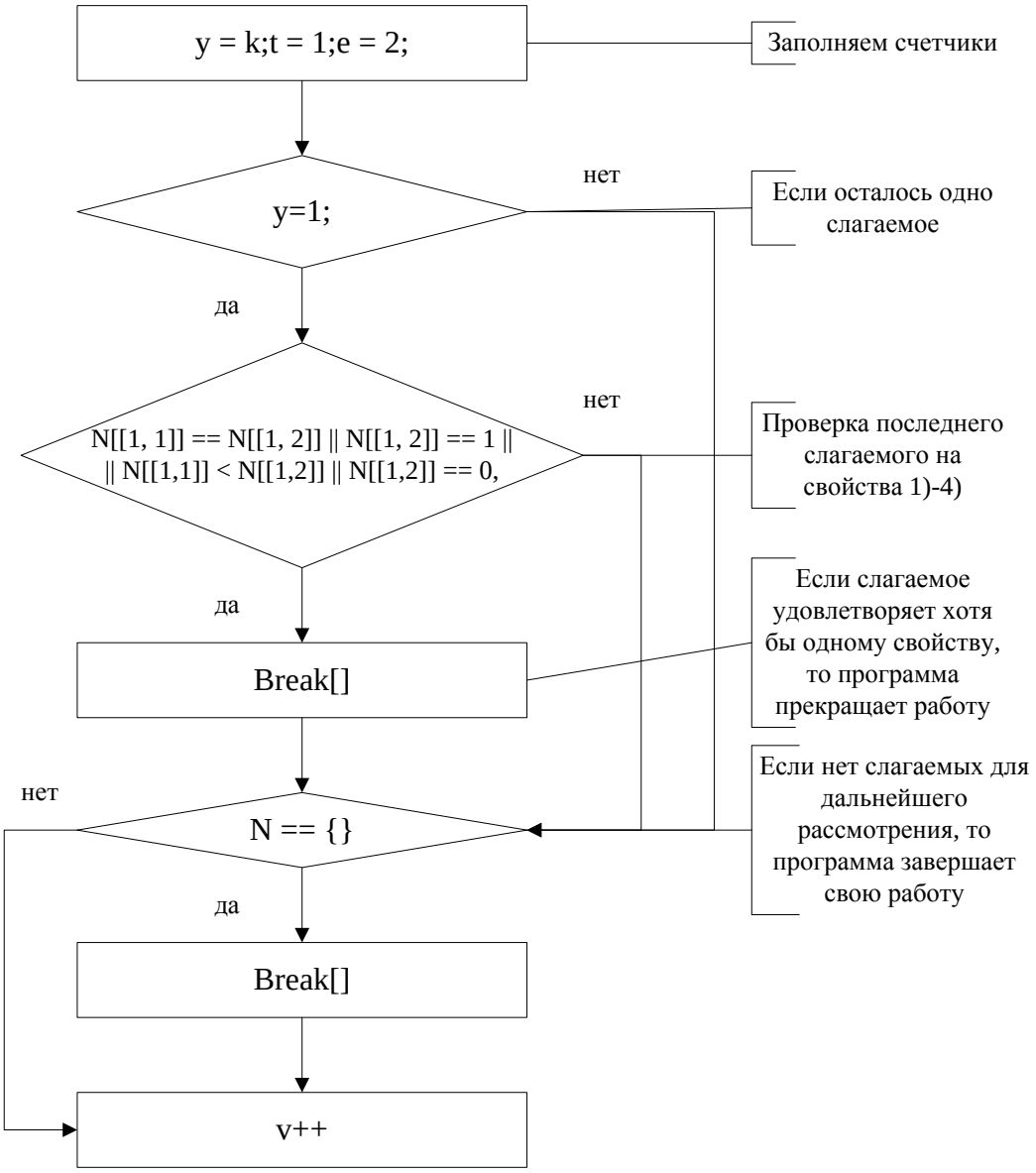


Приложение №4



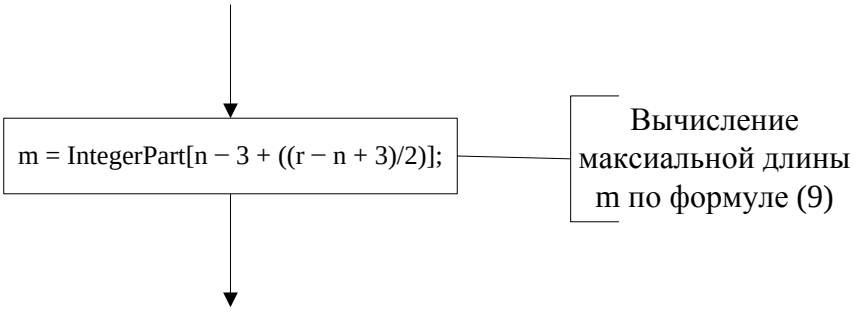
Приложение №5

Проверка: все ли слагаемые достигли предельных значений.



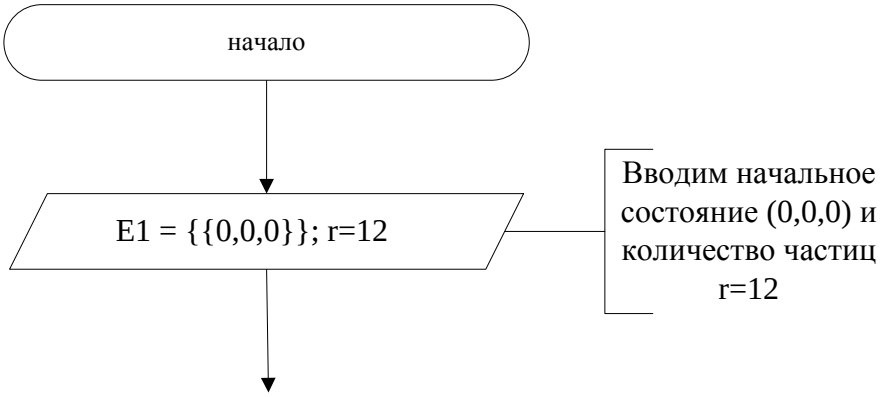
Приложение №6

Вычисление максимальной длины графа.



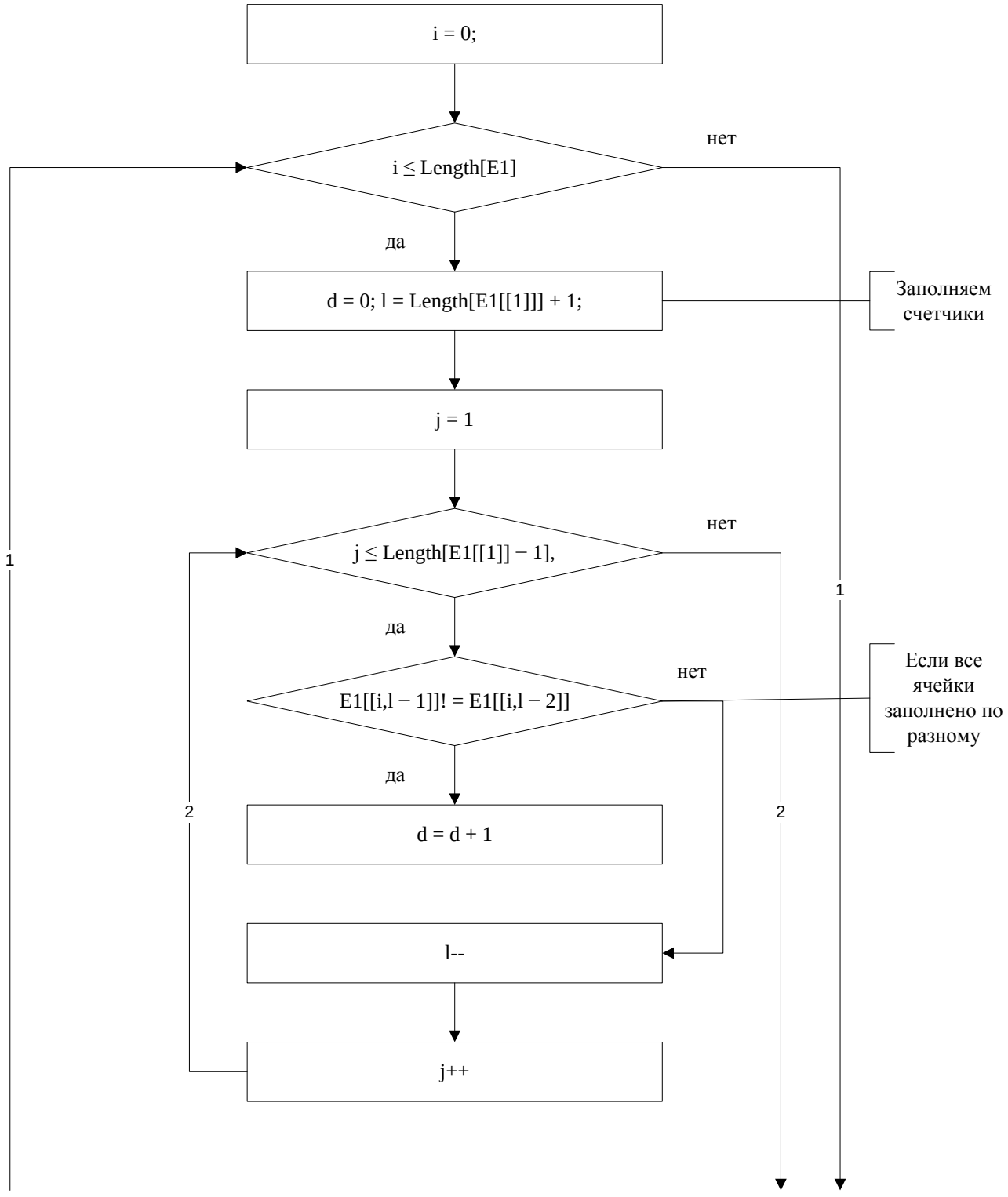
Приложение №7

Ввод начальных данных для получения состояний графа.

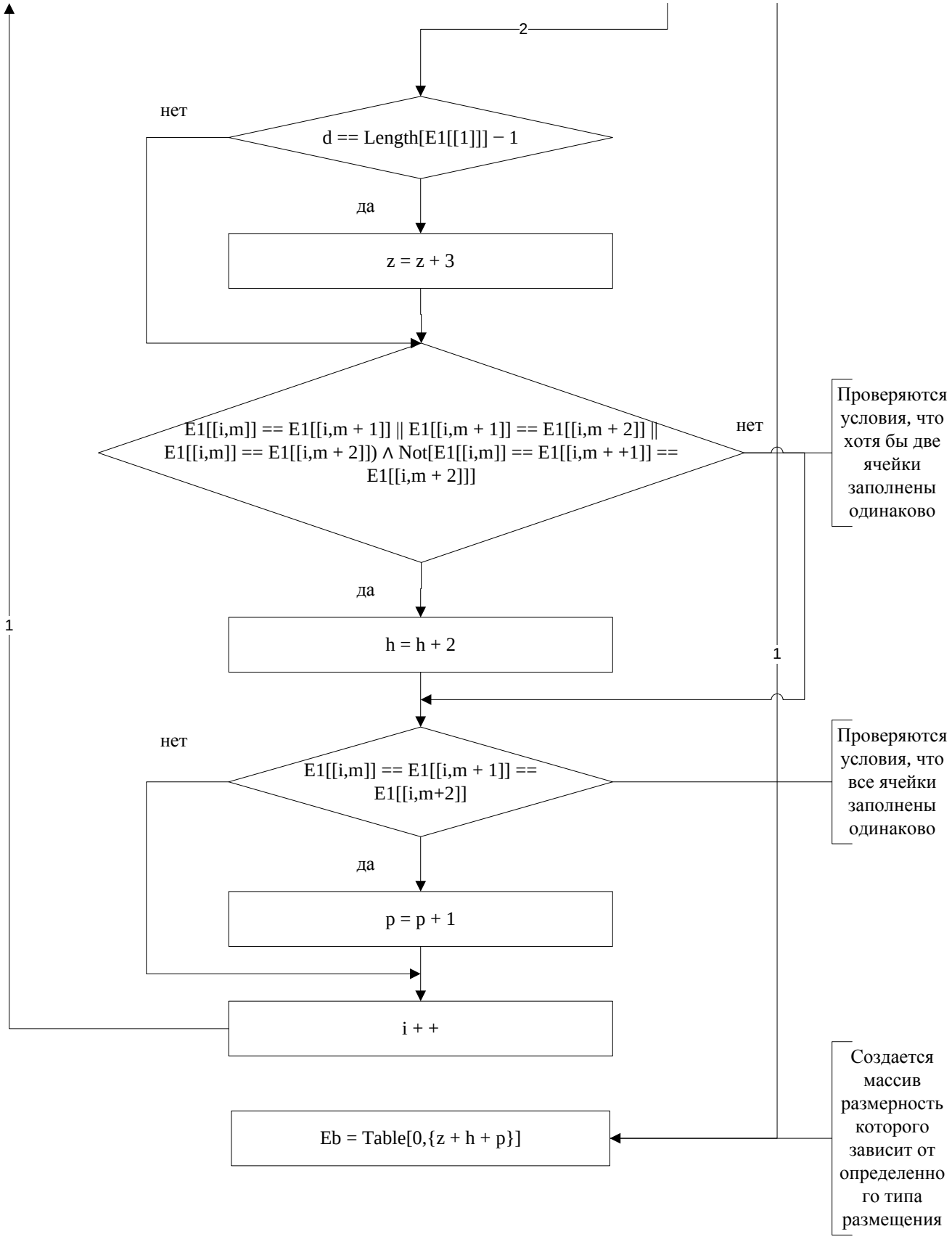


Приложение №8

Определение типа размещения.

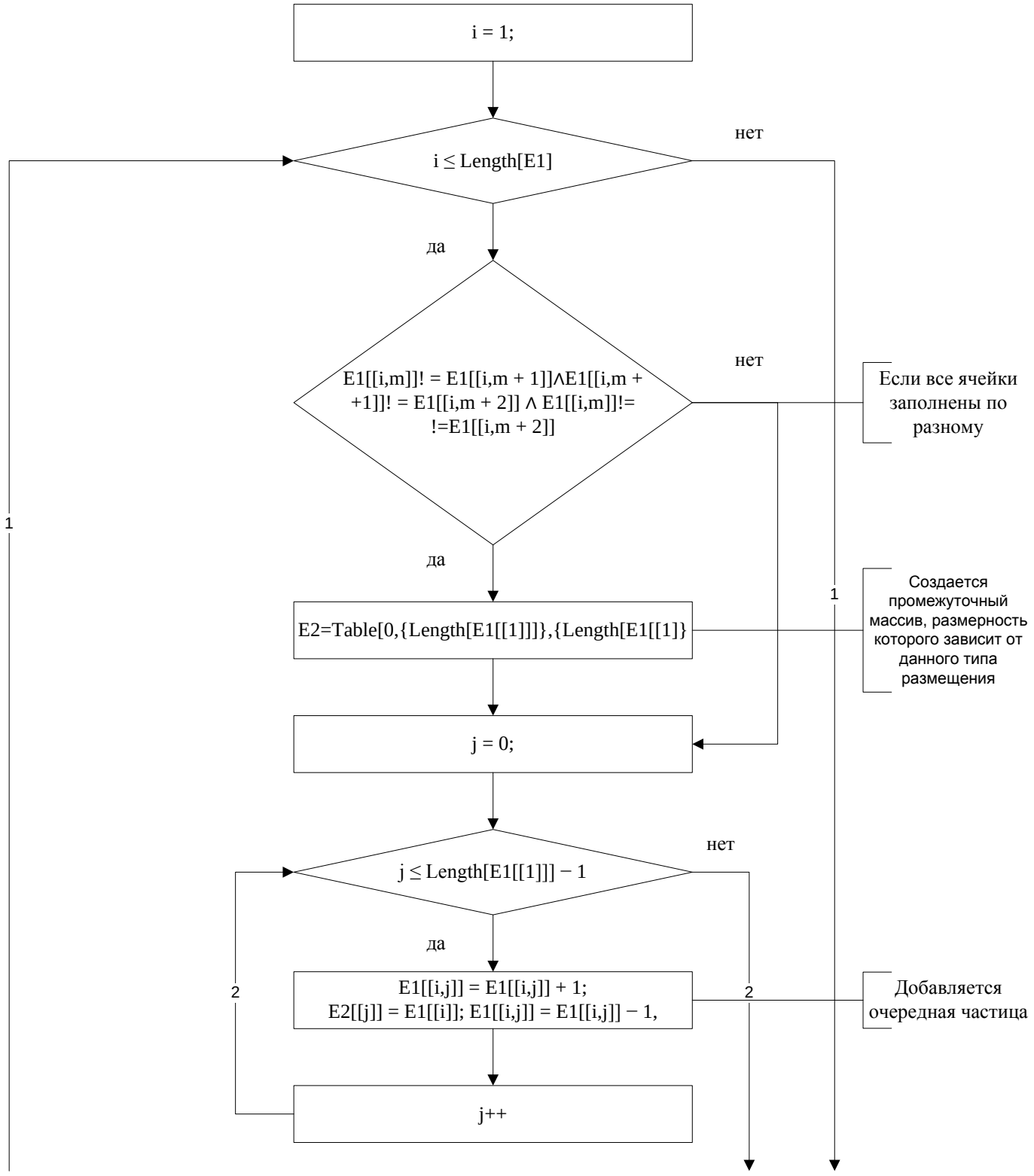


Приложение №8

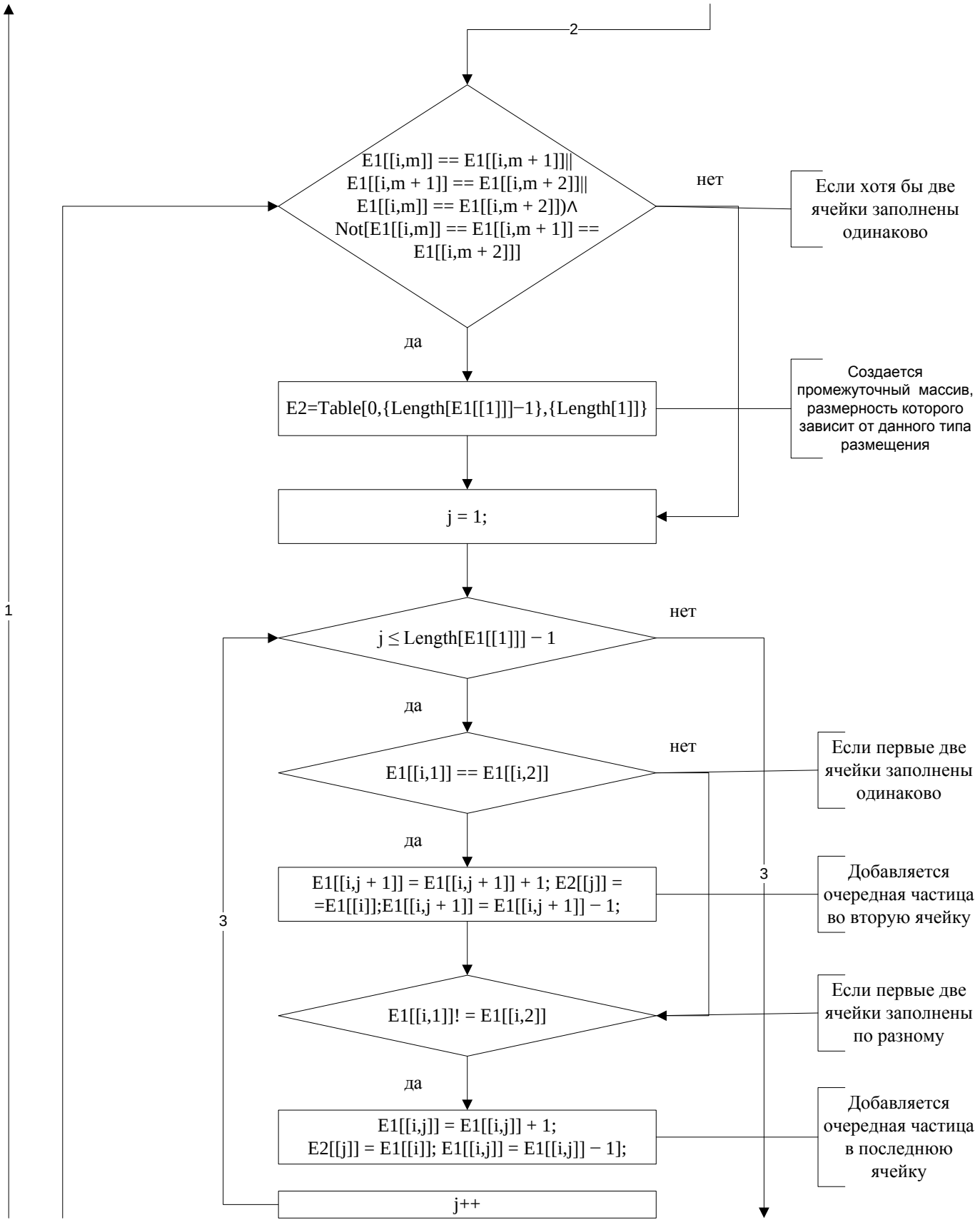


Приложение №9

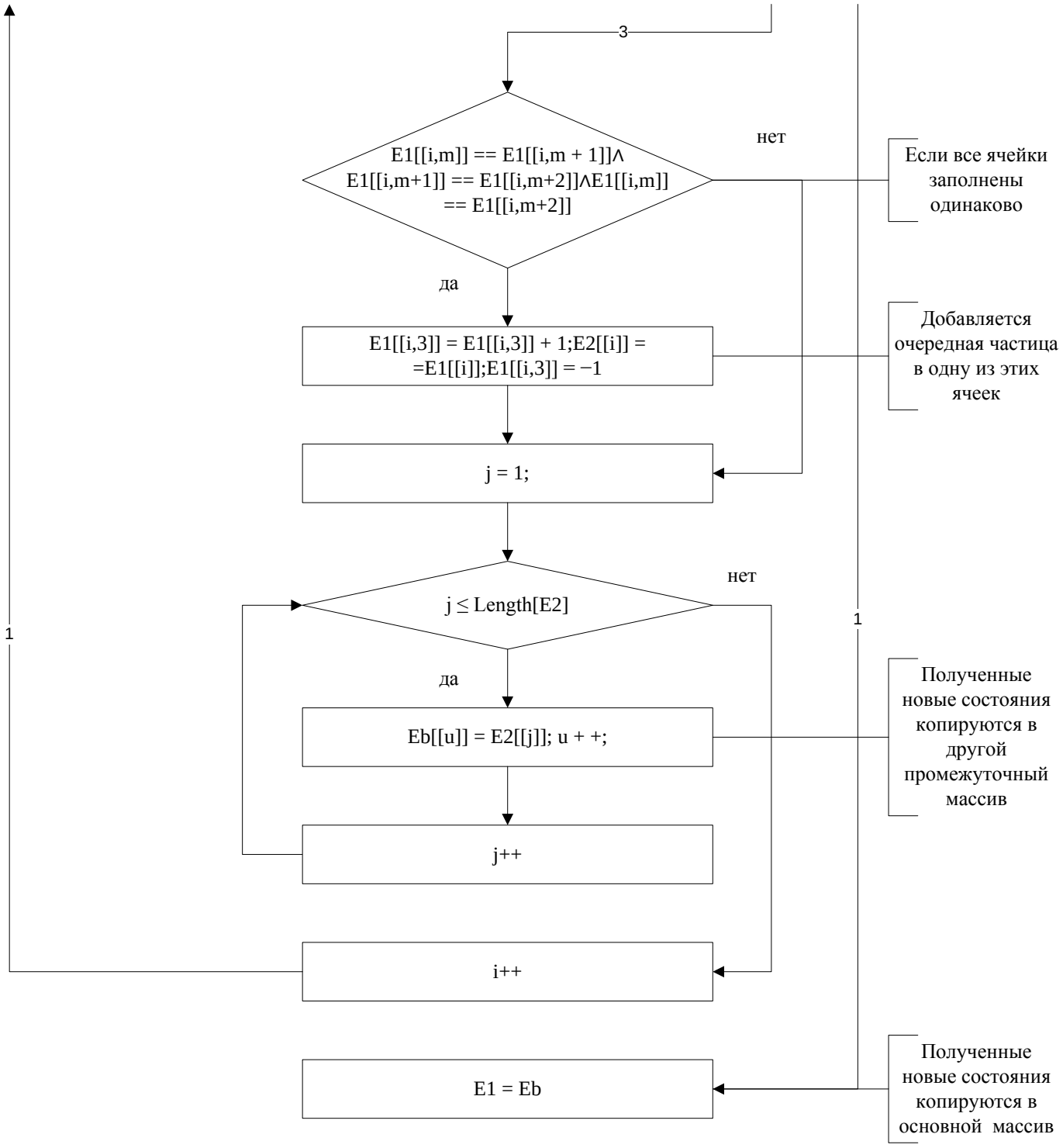
Добавление очередной частицы в графе.



Приложение №9

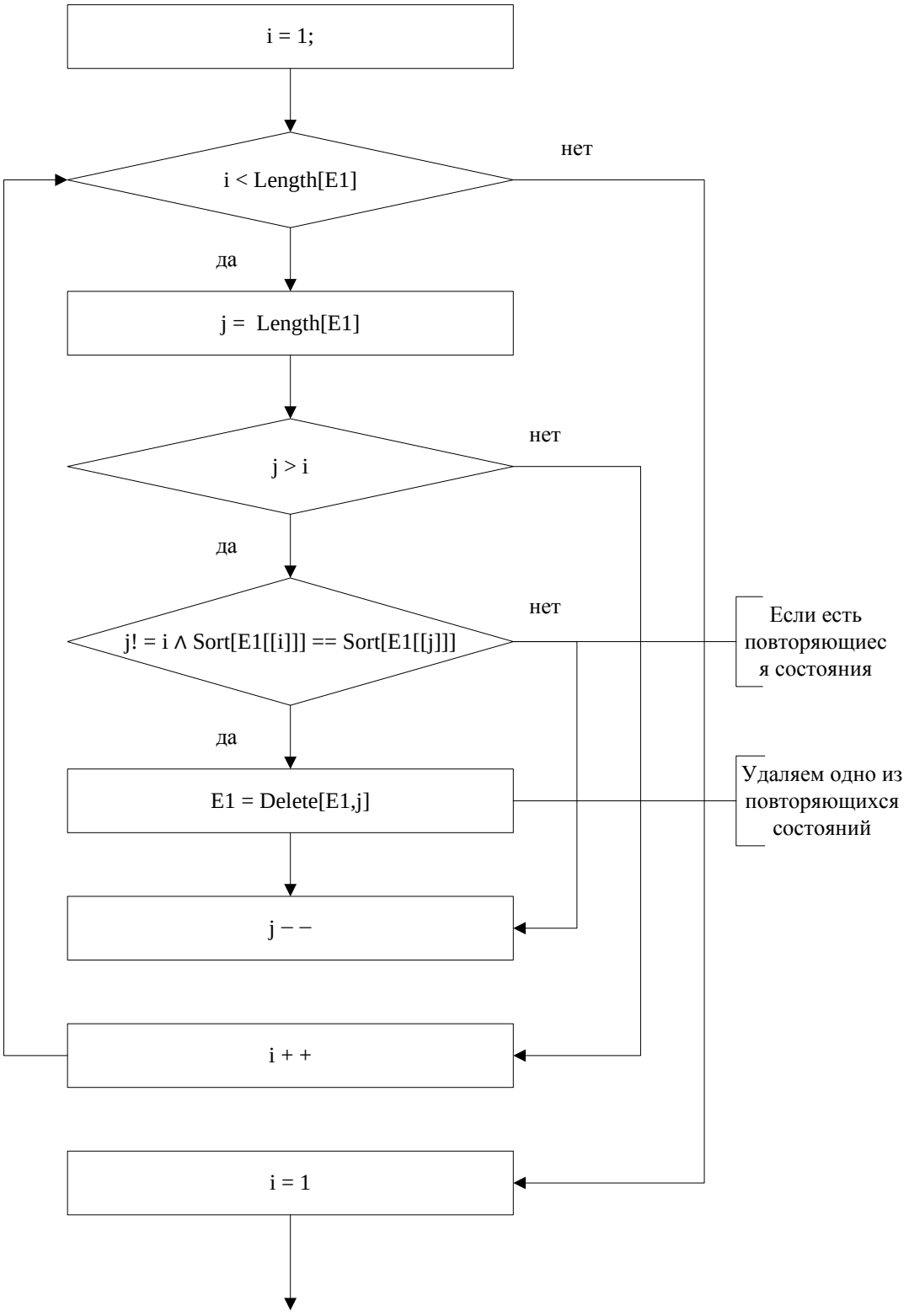


Приложение №9

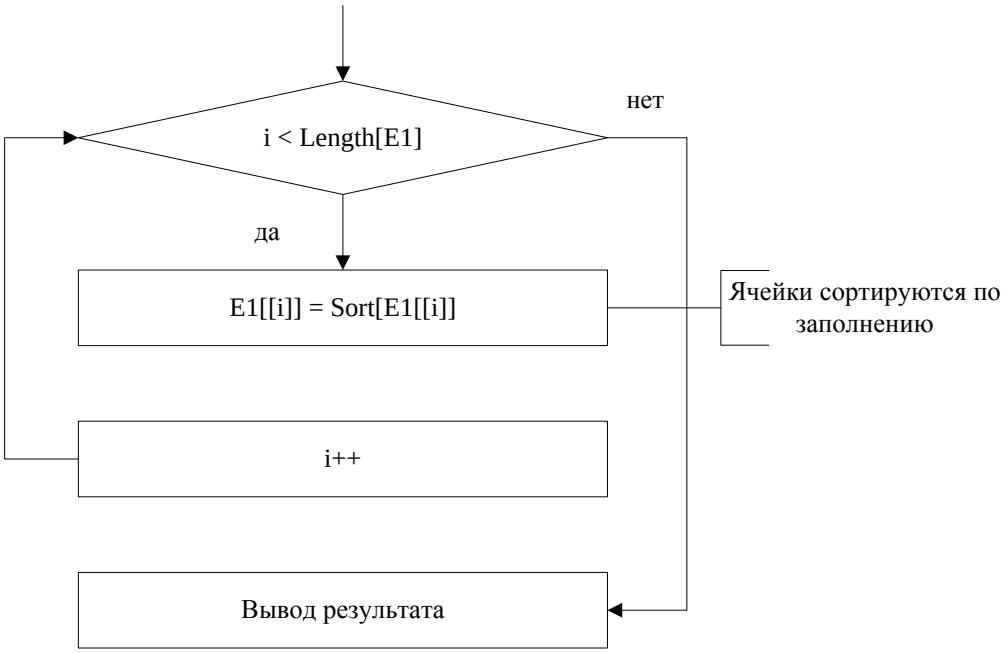


Приложение №10

Сортировка ячеек и удаление повторяющихся состояний.

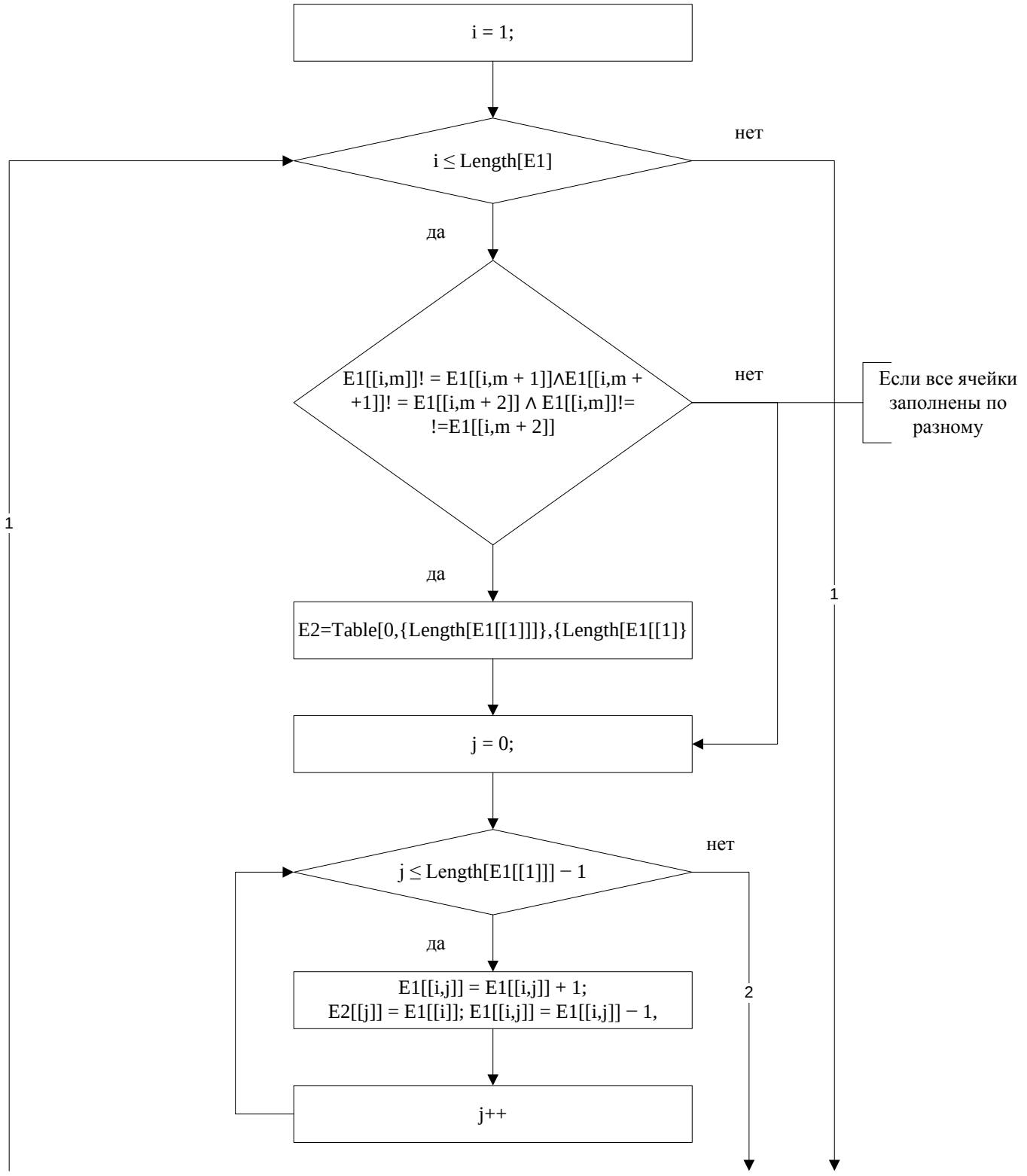


Приложение №10

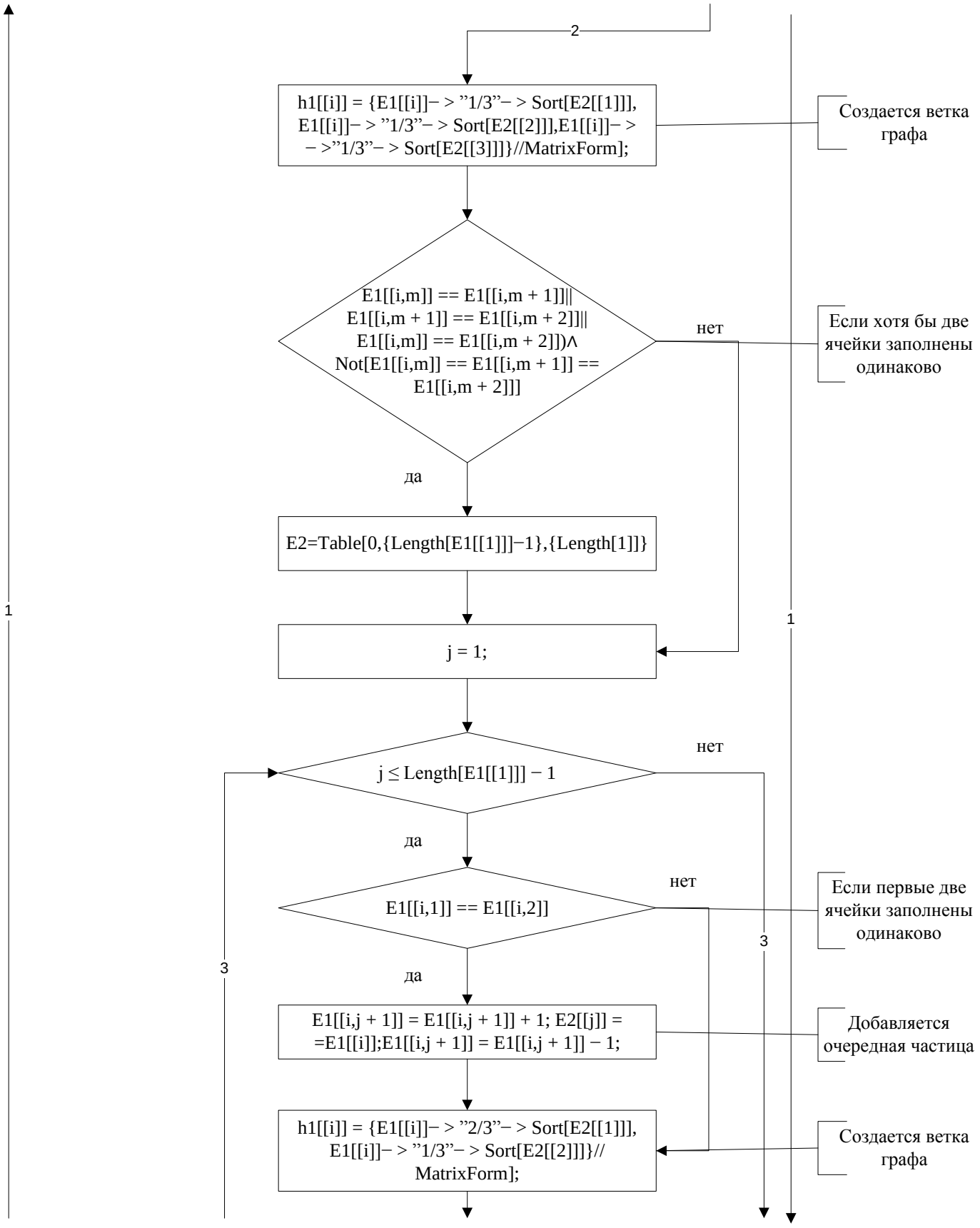


Приложение №11

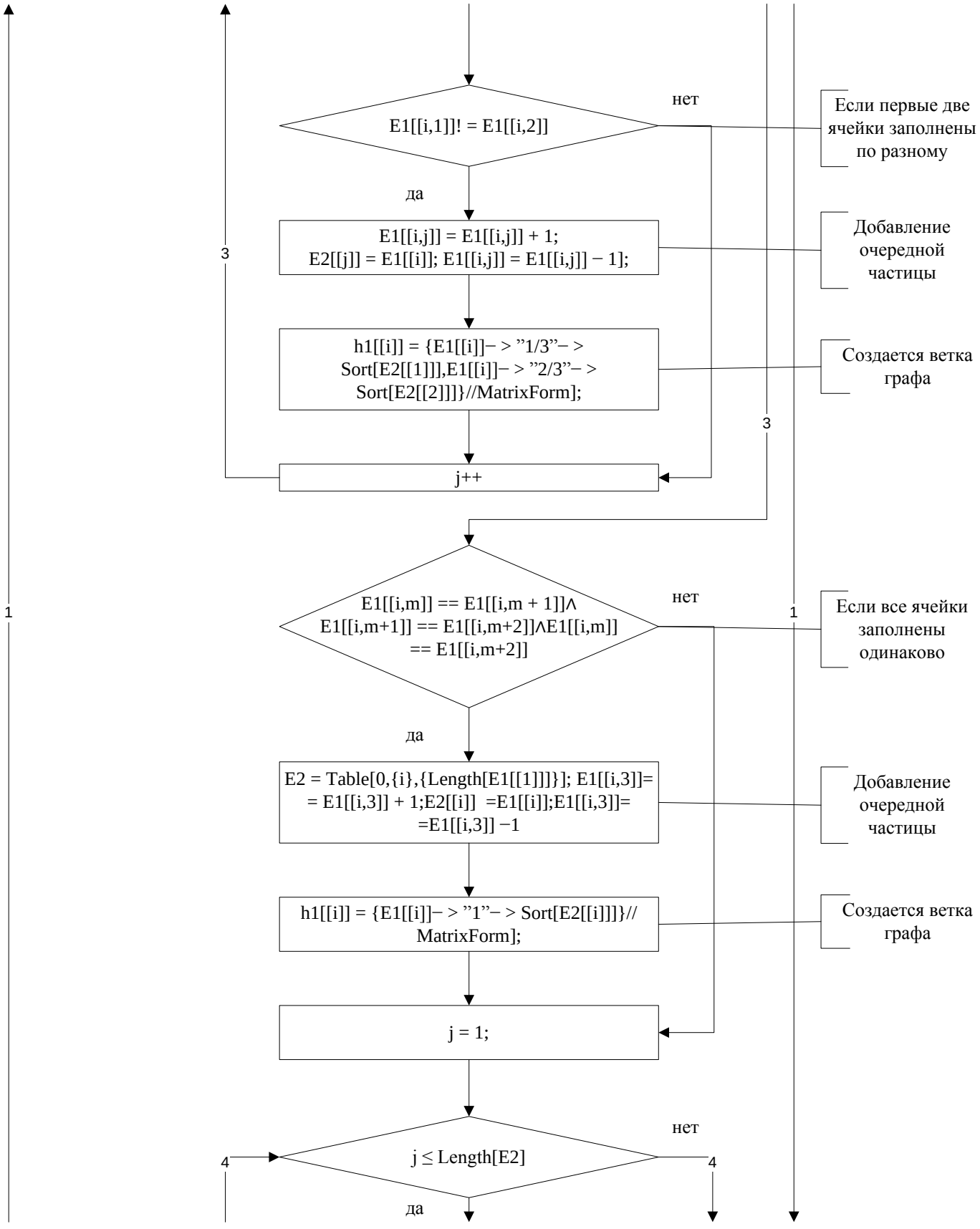
Добавление очередной частицы и создание ветки графа.



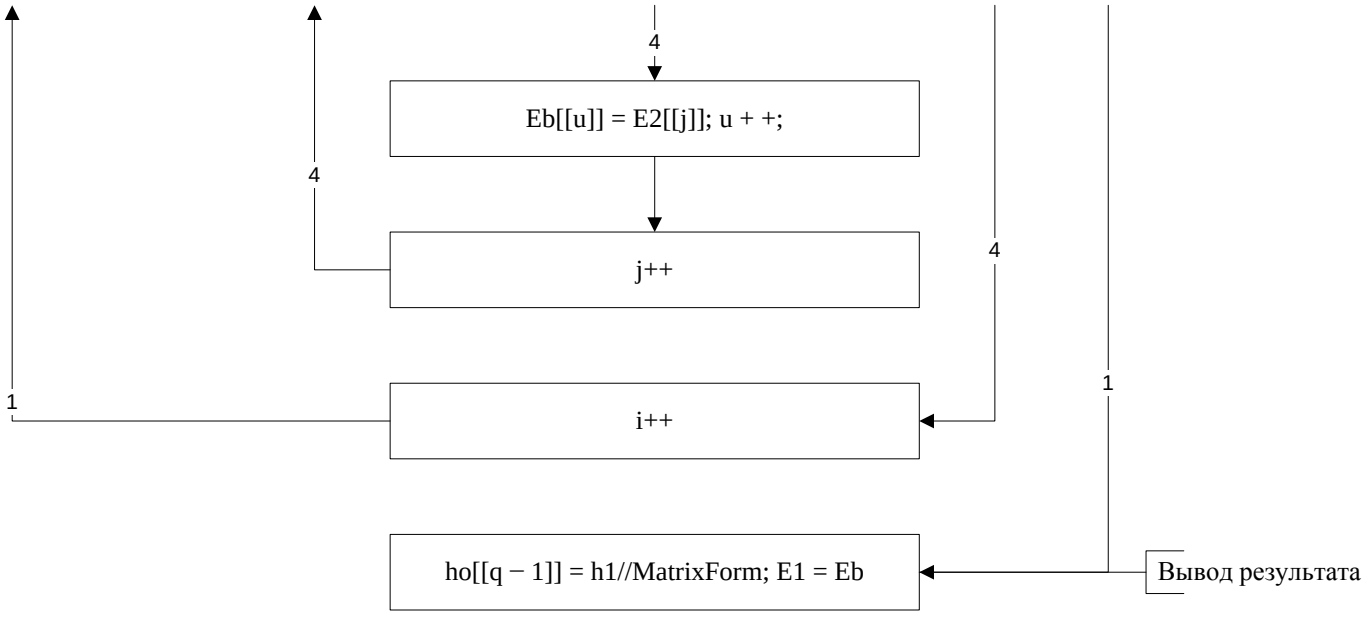
Приложение №11



Приложение №11

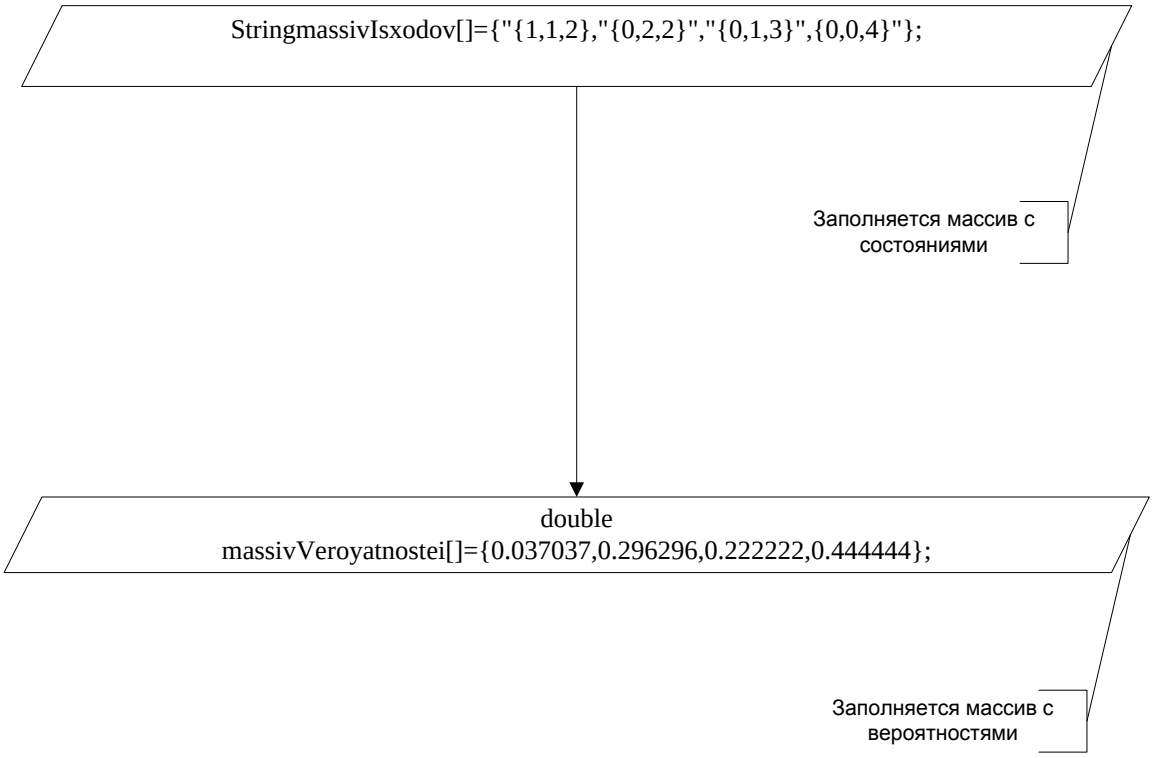


Приложение №11



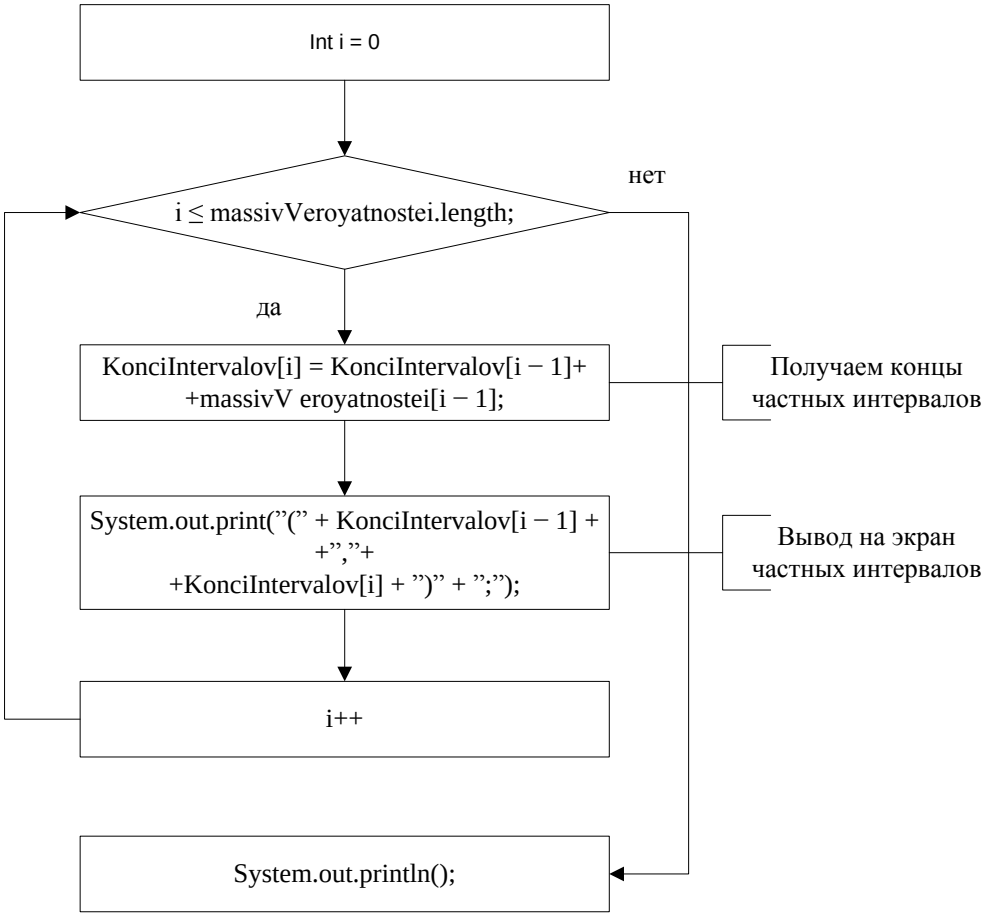
Приложение №12

Ввод начальных данных для моделирования схемы методом маркировки.



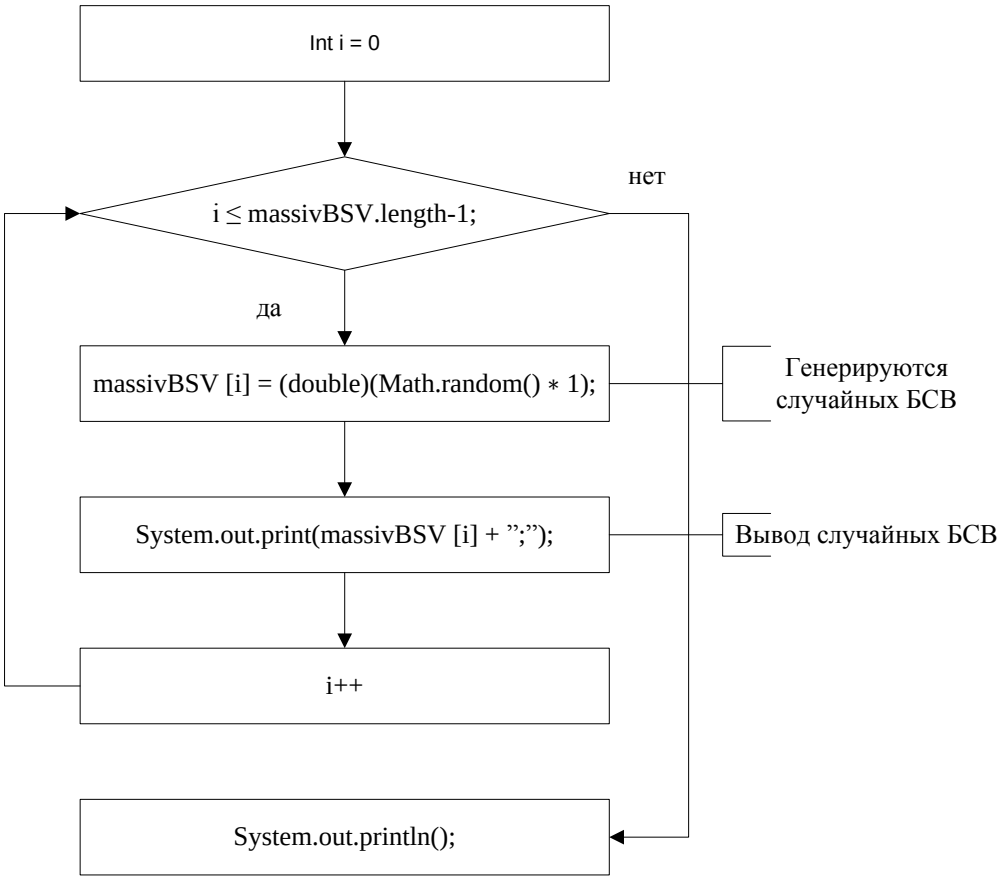
Приложение №13

Создание частичных интервалов и их вывод.



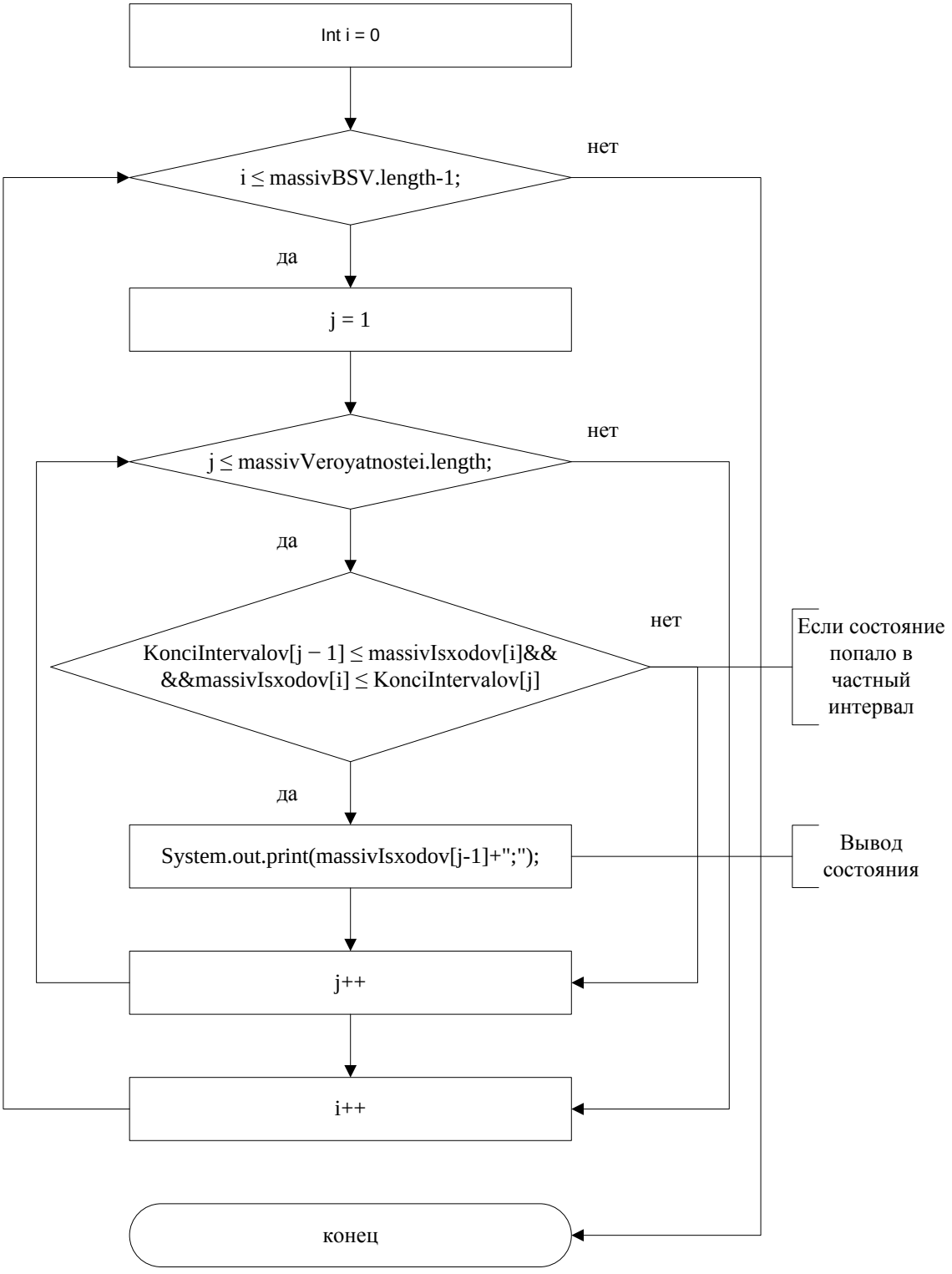
Приложение №14

Генерация БСВ и их вывод.



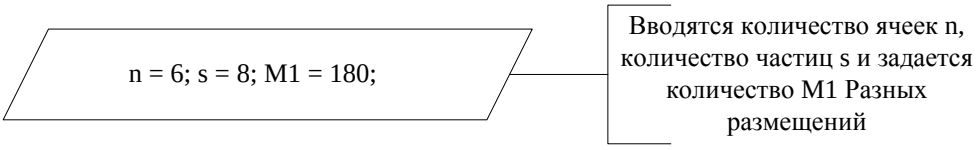
Приложение №15

Проверка: в какой частичный интервал попадает состояние.



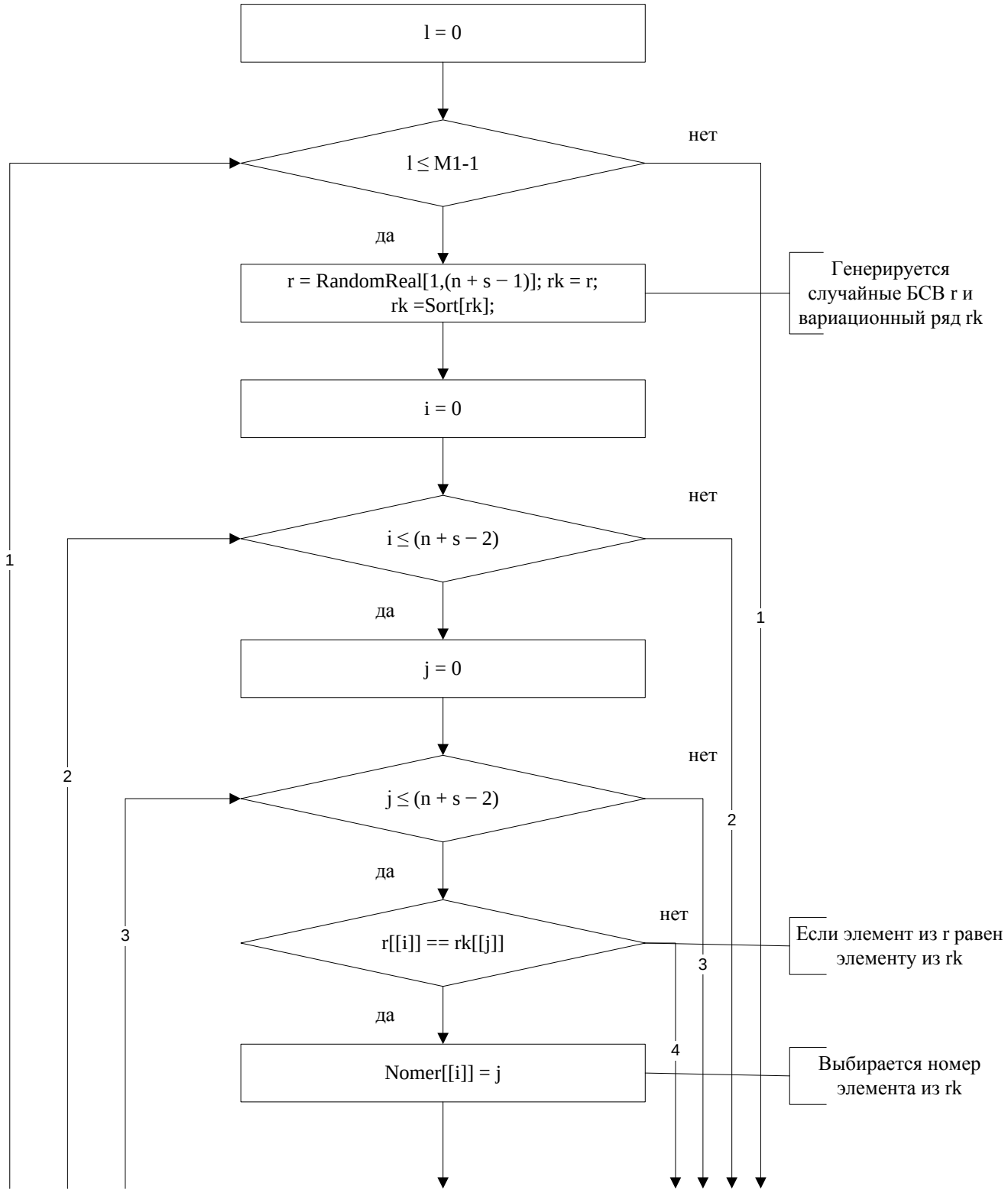
Приложение №16

Ввод начальных данных для моделирования схемы вторым способом.

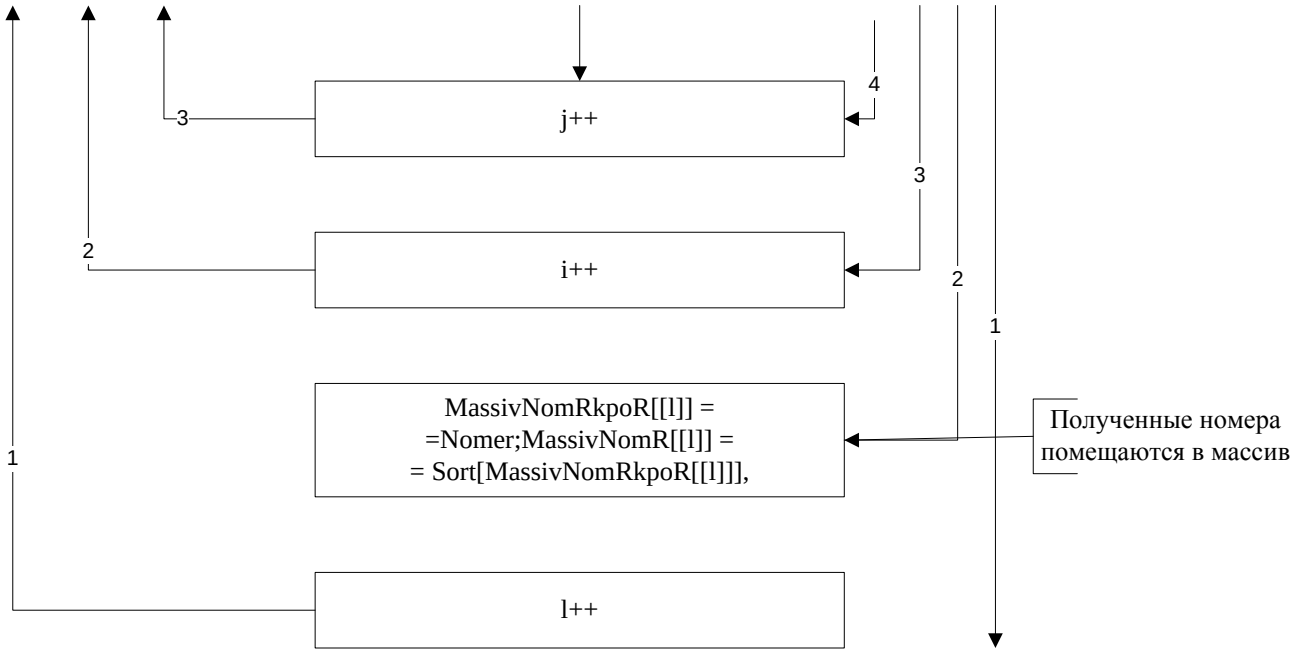


Приложение №17

Генерация БСВ и вариационного ряда.

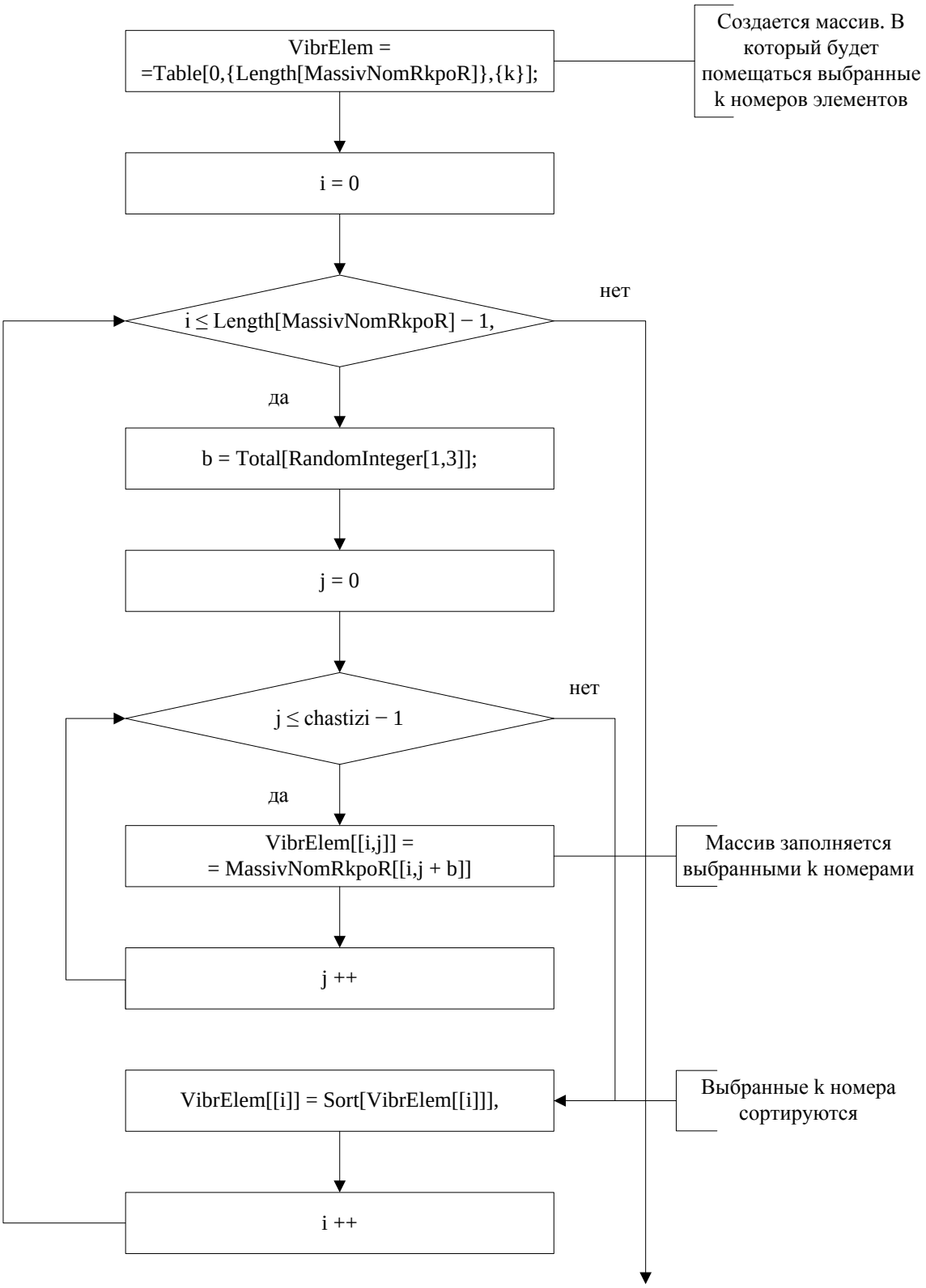


Приложение №17



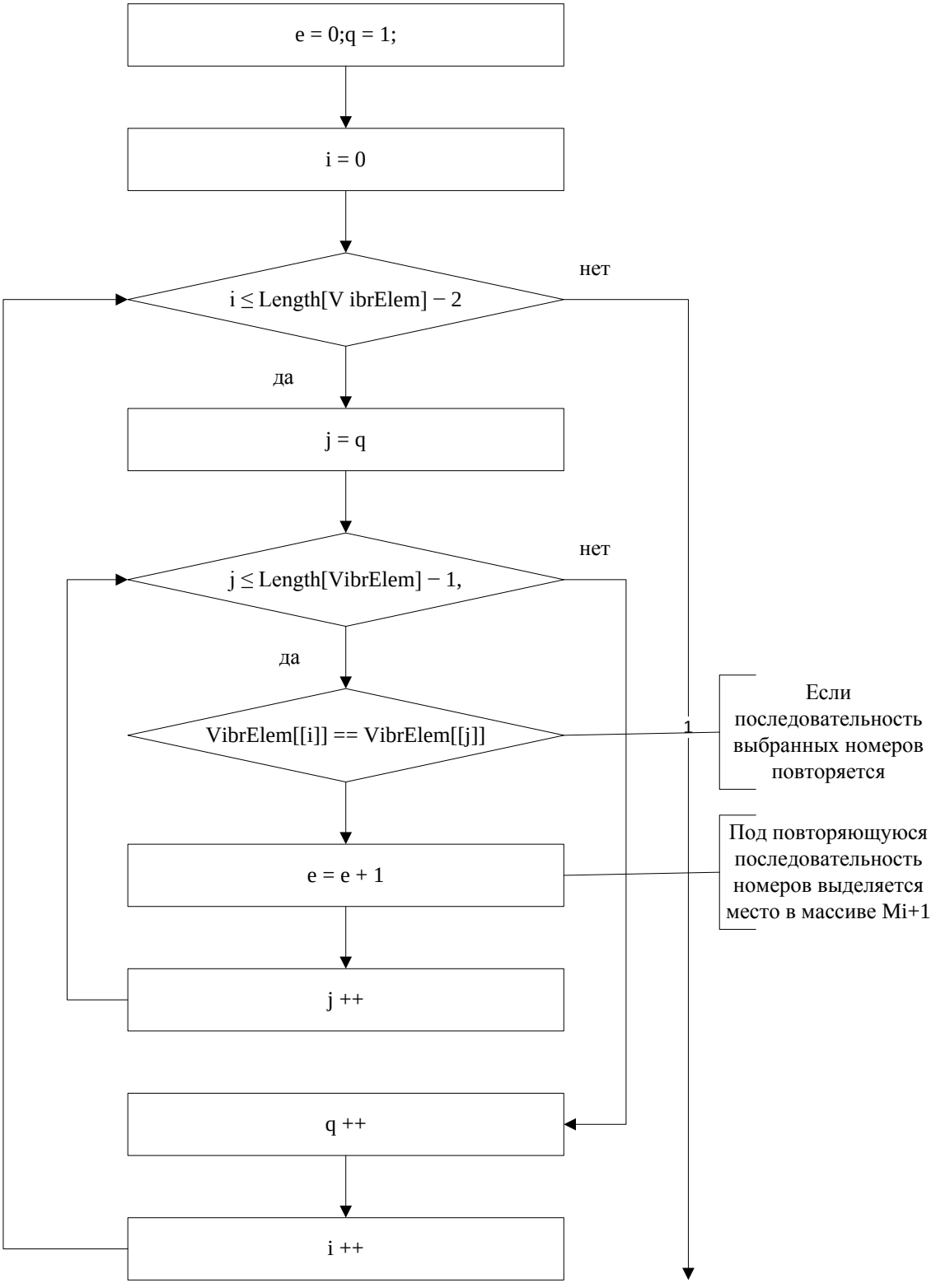
Приложение №18

Выбор k элементов.

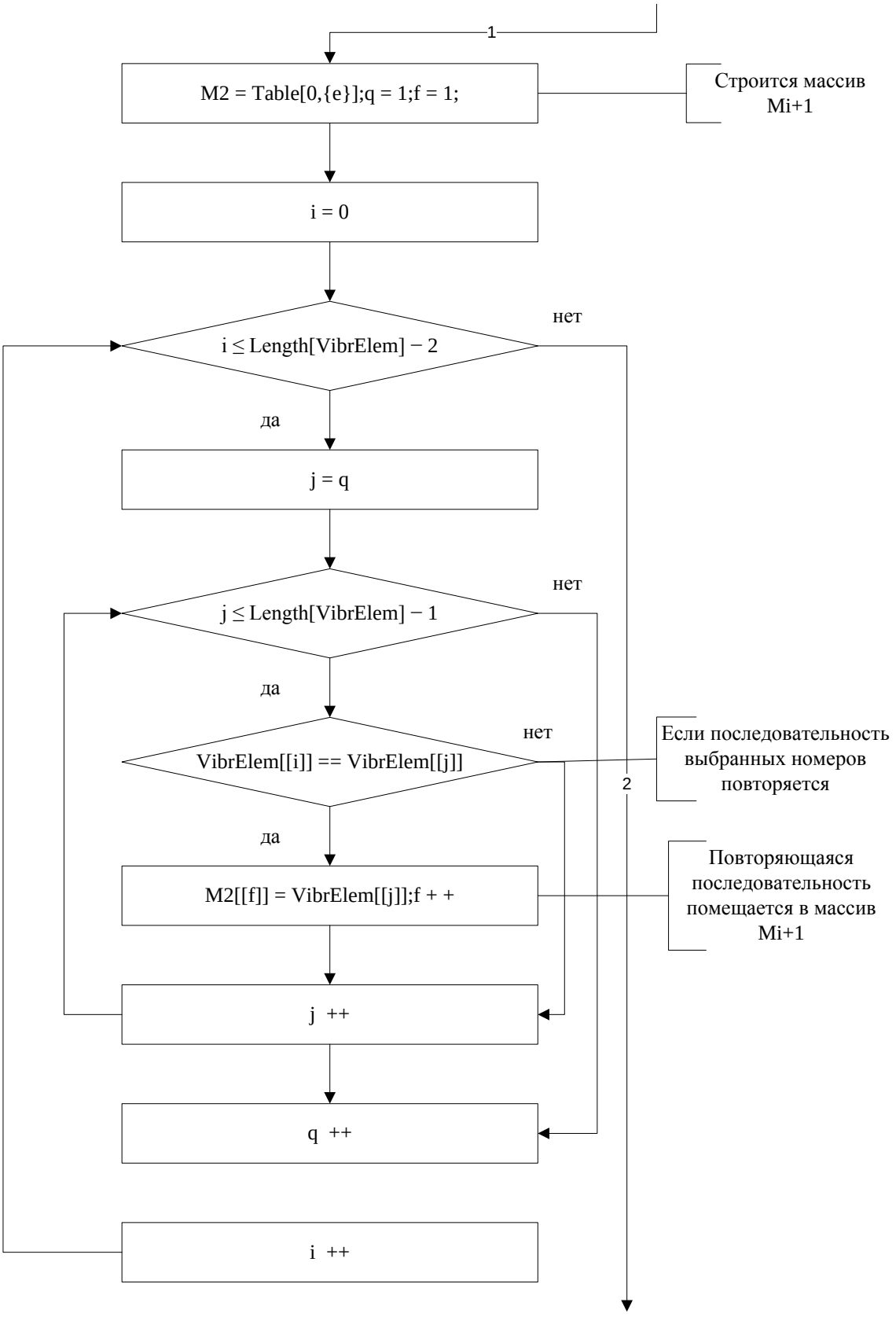


Приложение №19

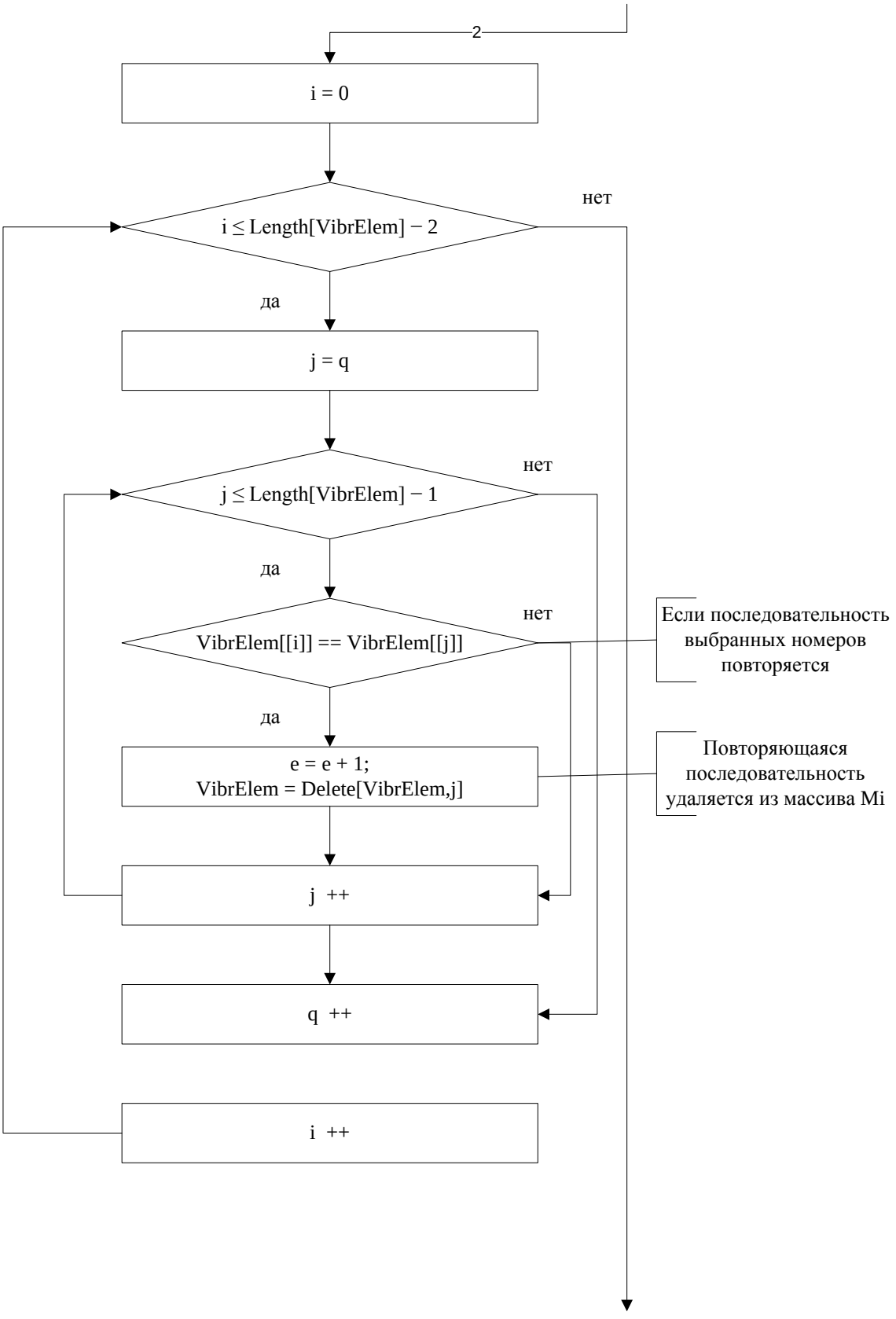
Внос повторяющихся выборок в Mi+1.



Приложение №19

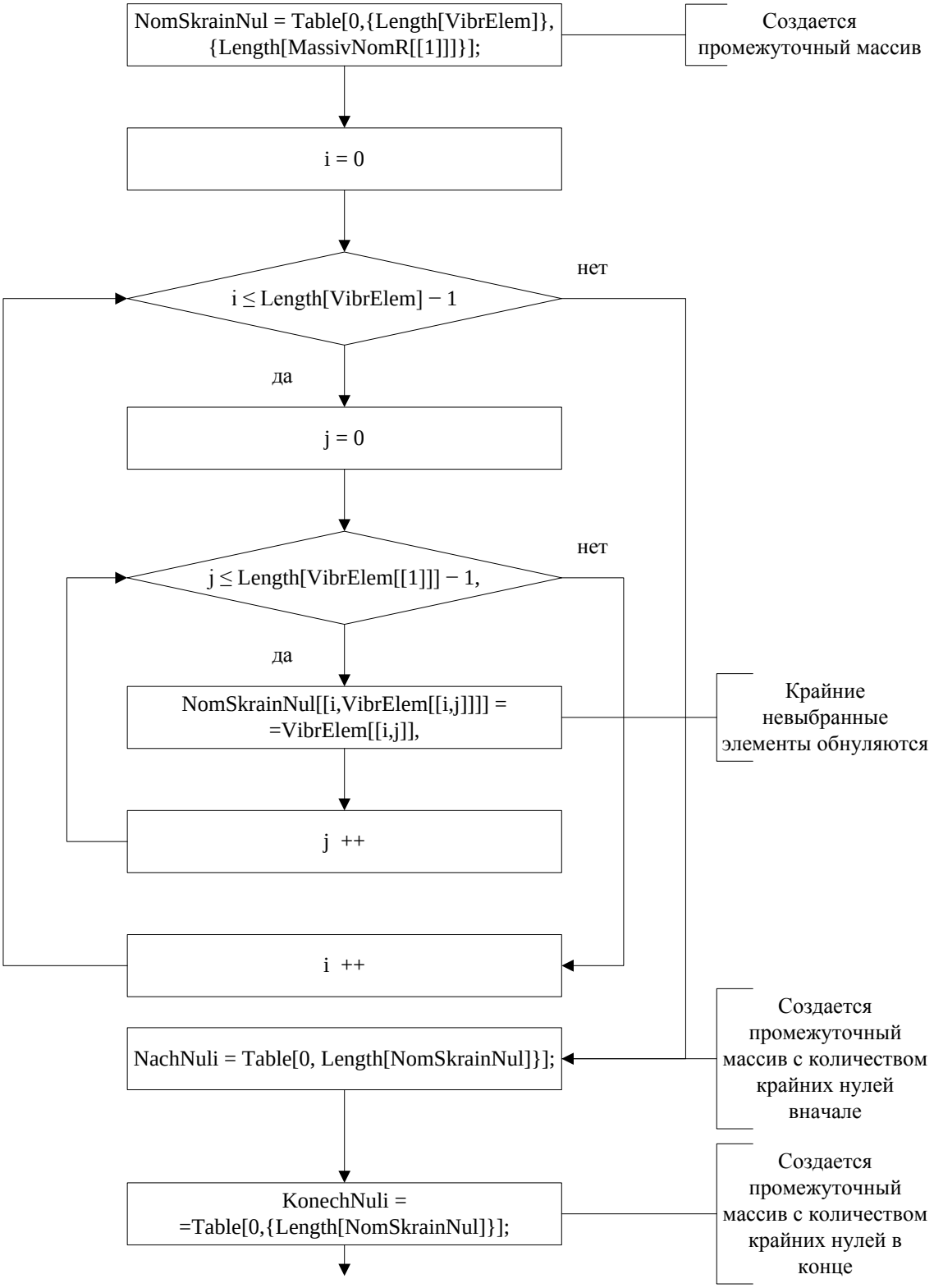


Приложение №19

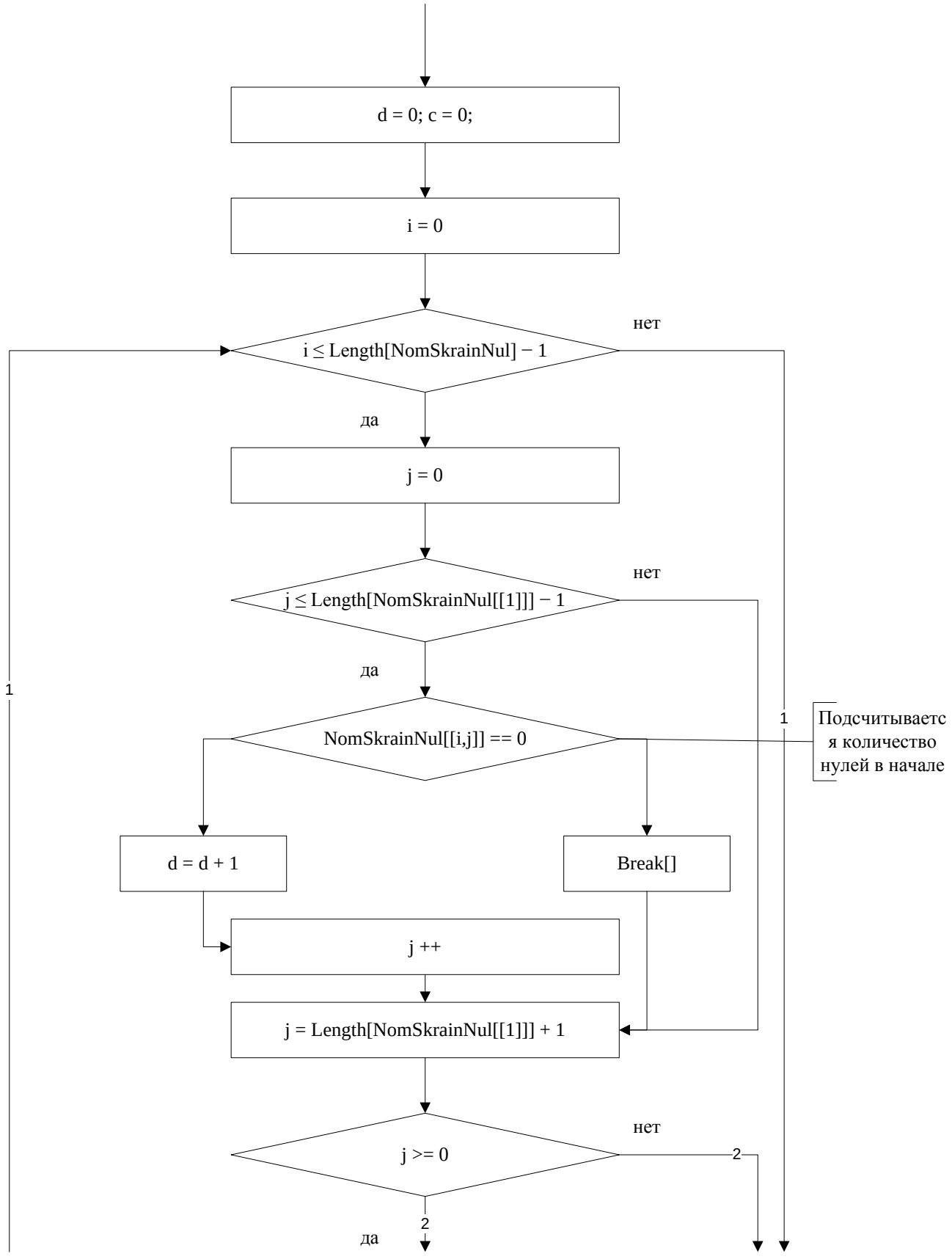


Приложение №20

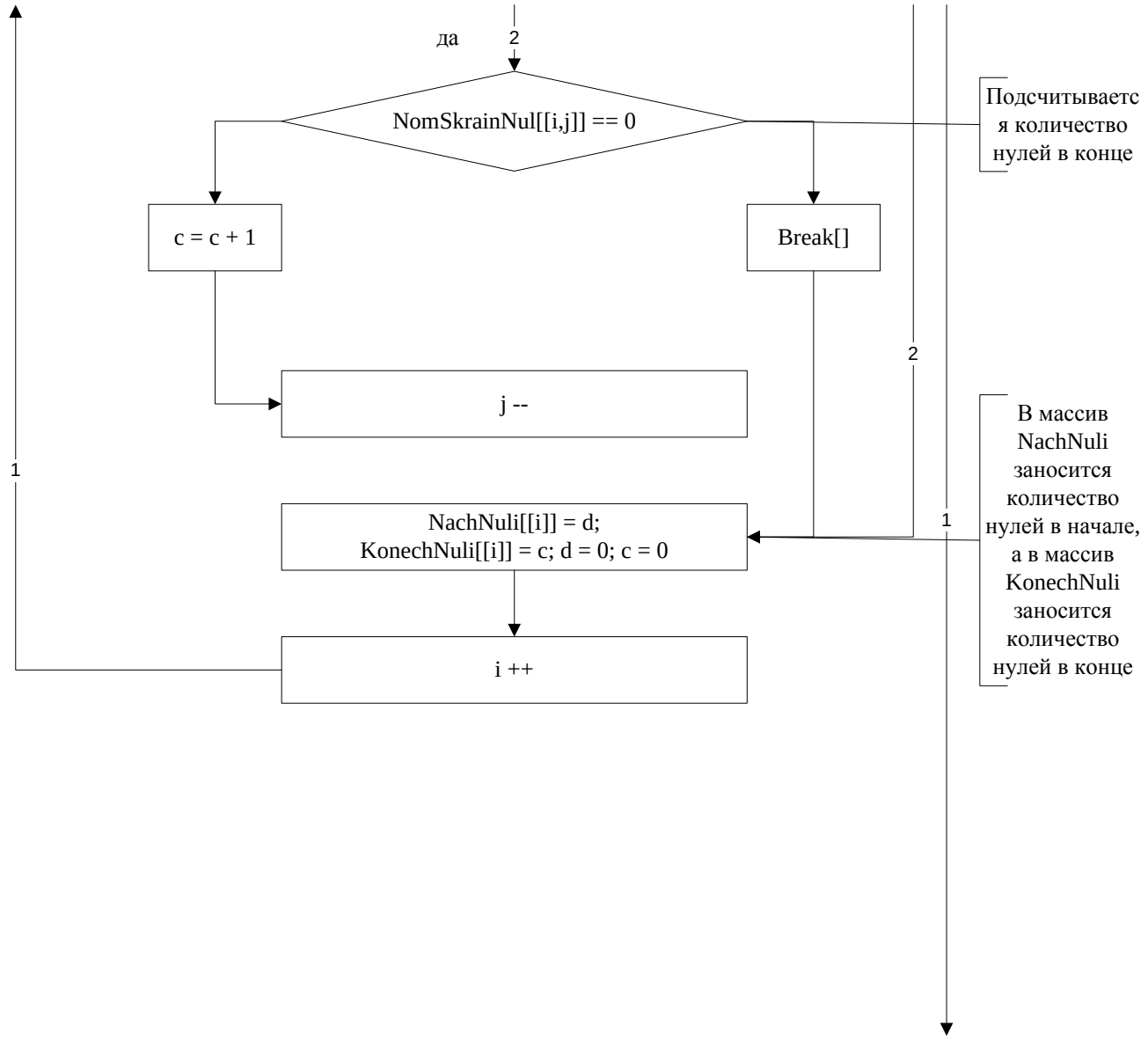
Обнуление крайних невыбранных элементов.



Приложение №20

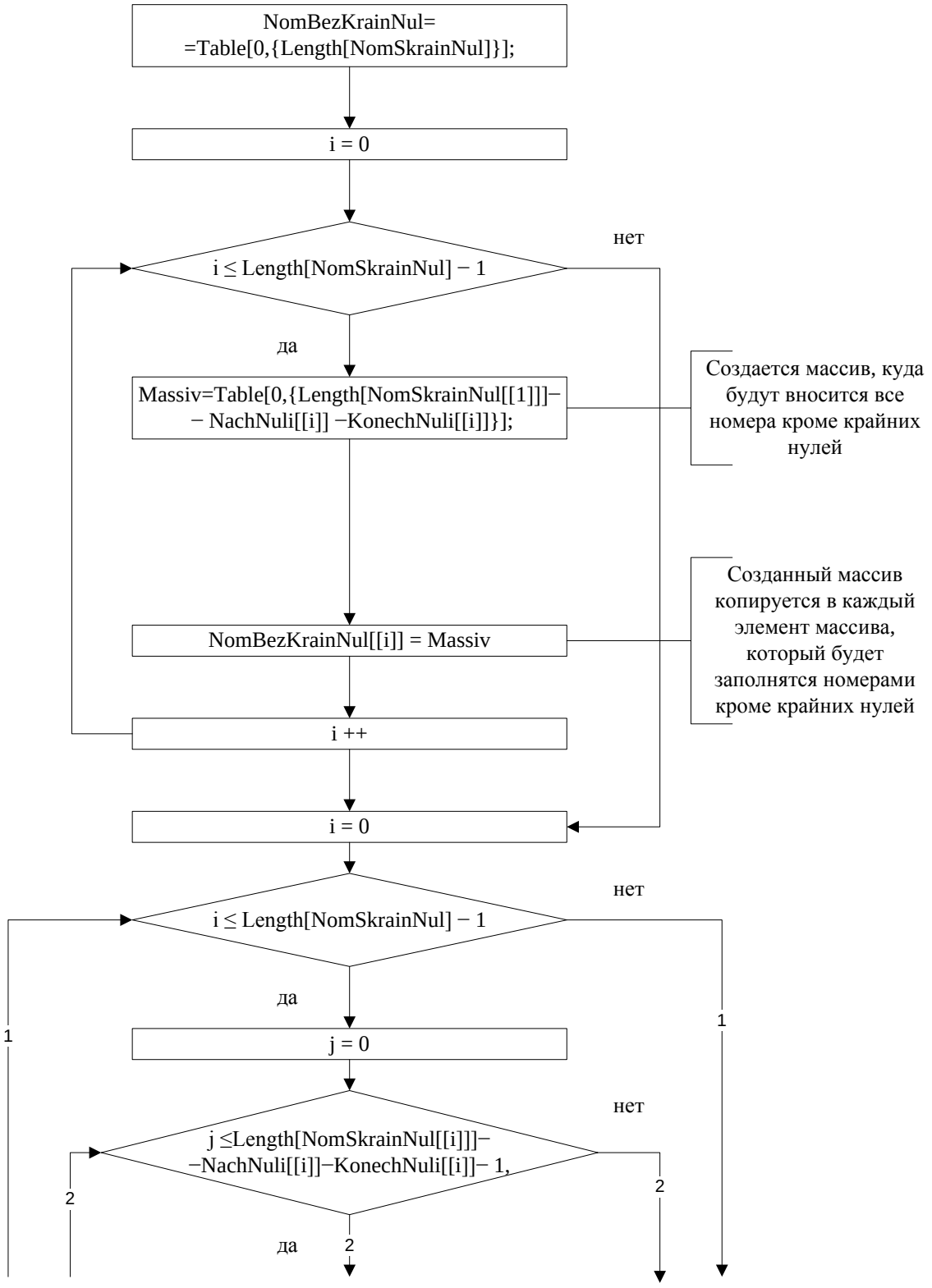


Приложение №20

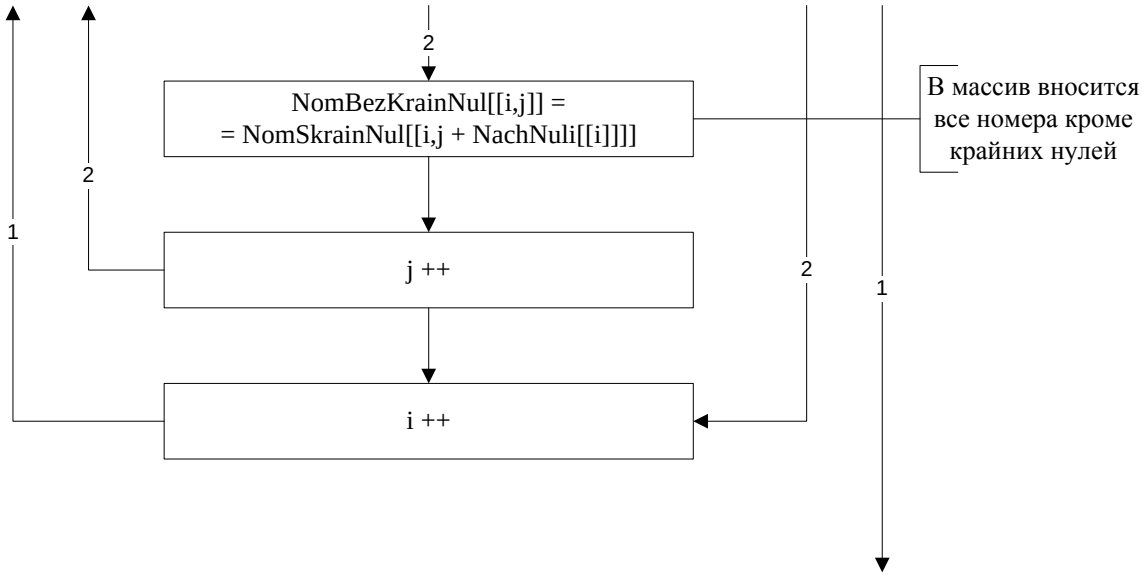


Приложение №21

Обнуление крайних невыбранных некрайних элементов.

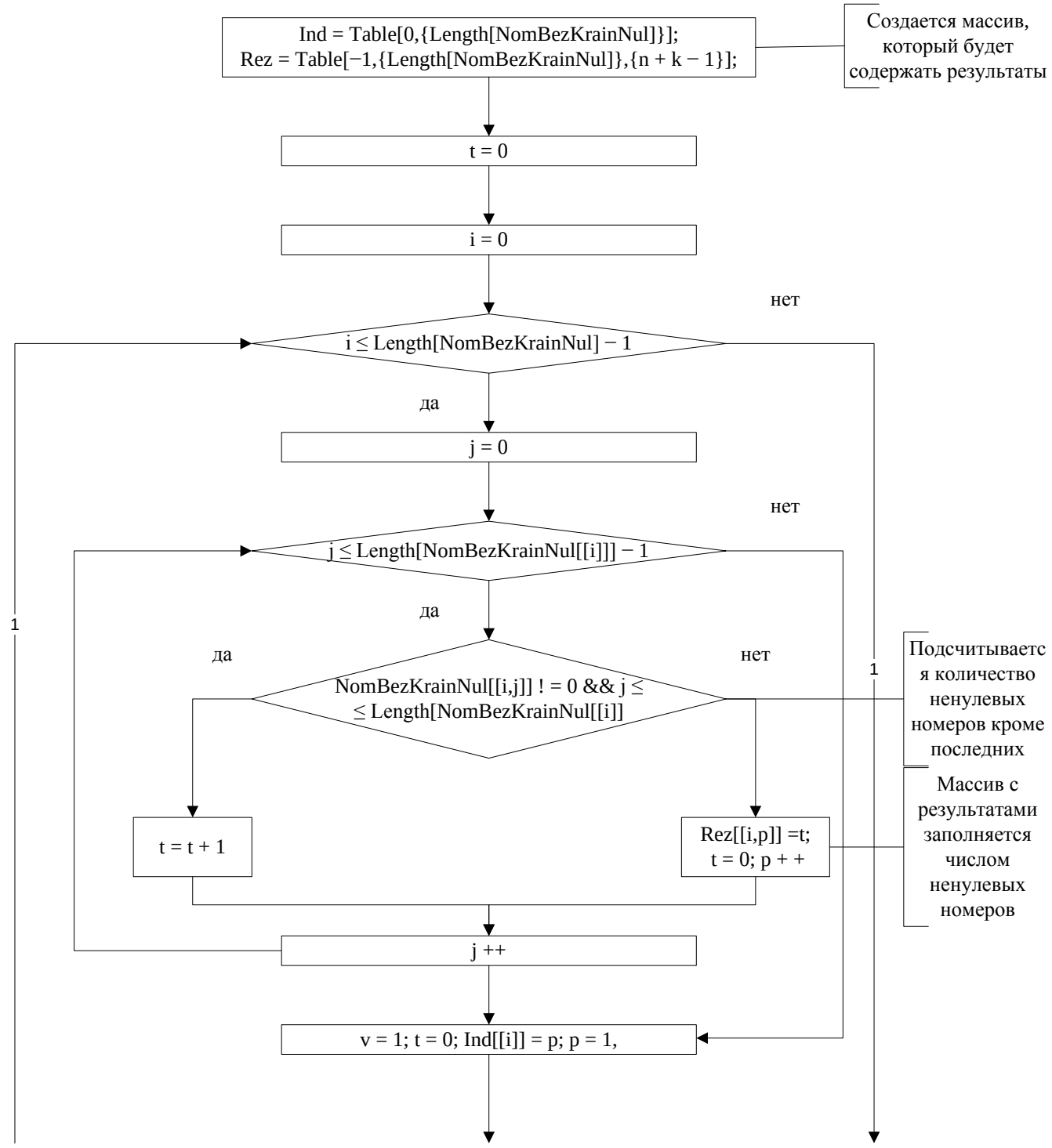


Приложение №21

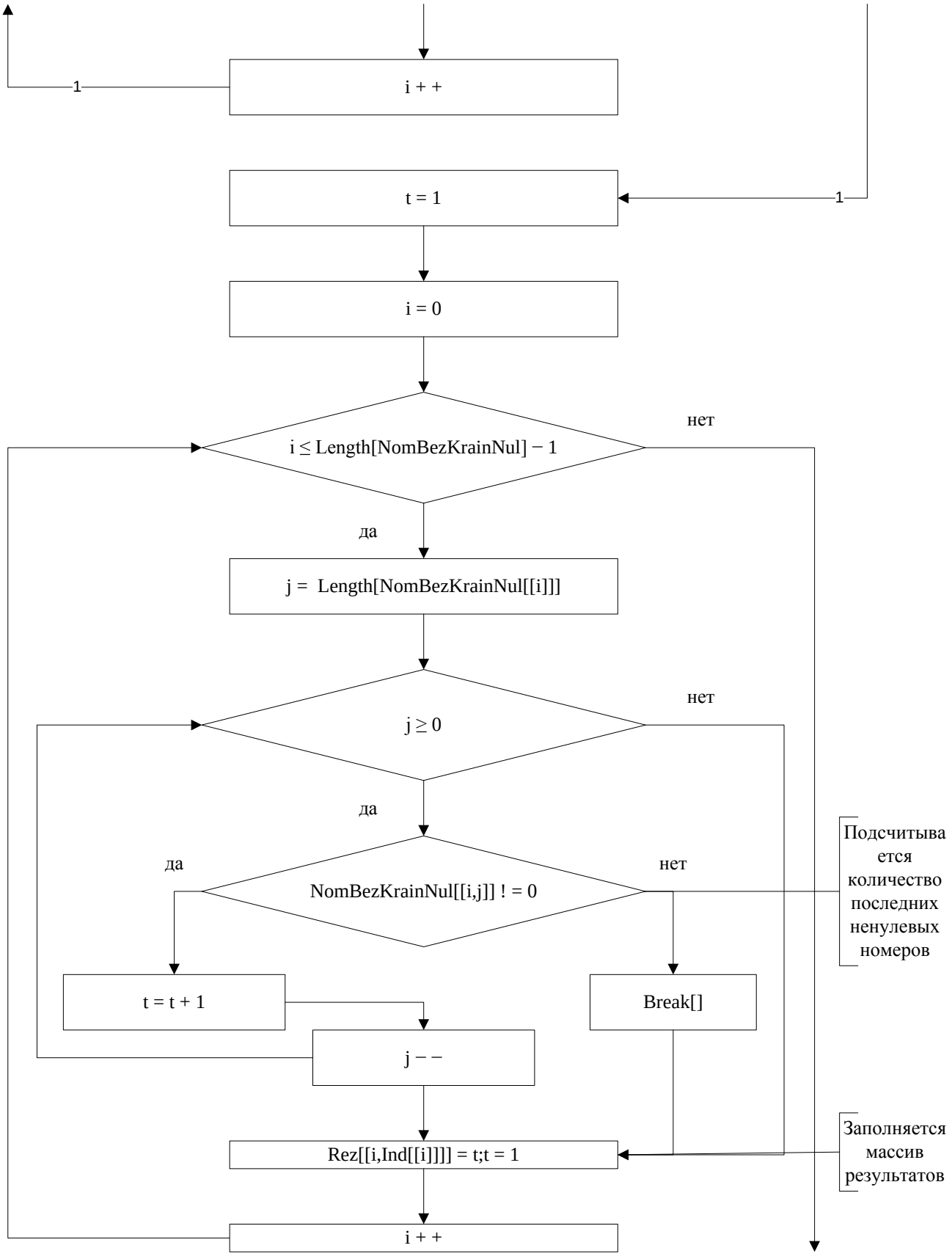


Приложение №22

Обработка выбранных элементов.

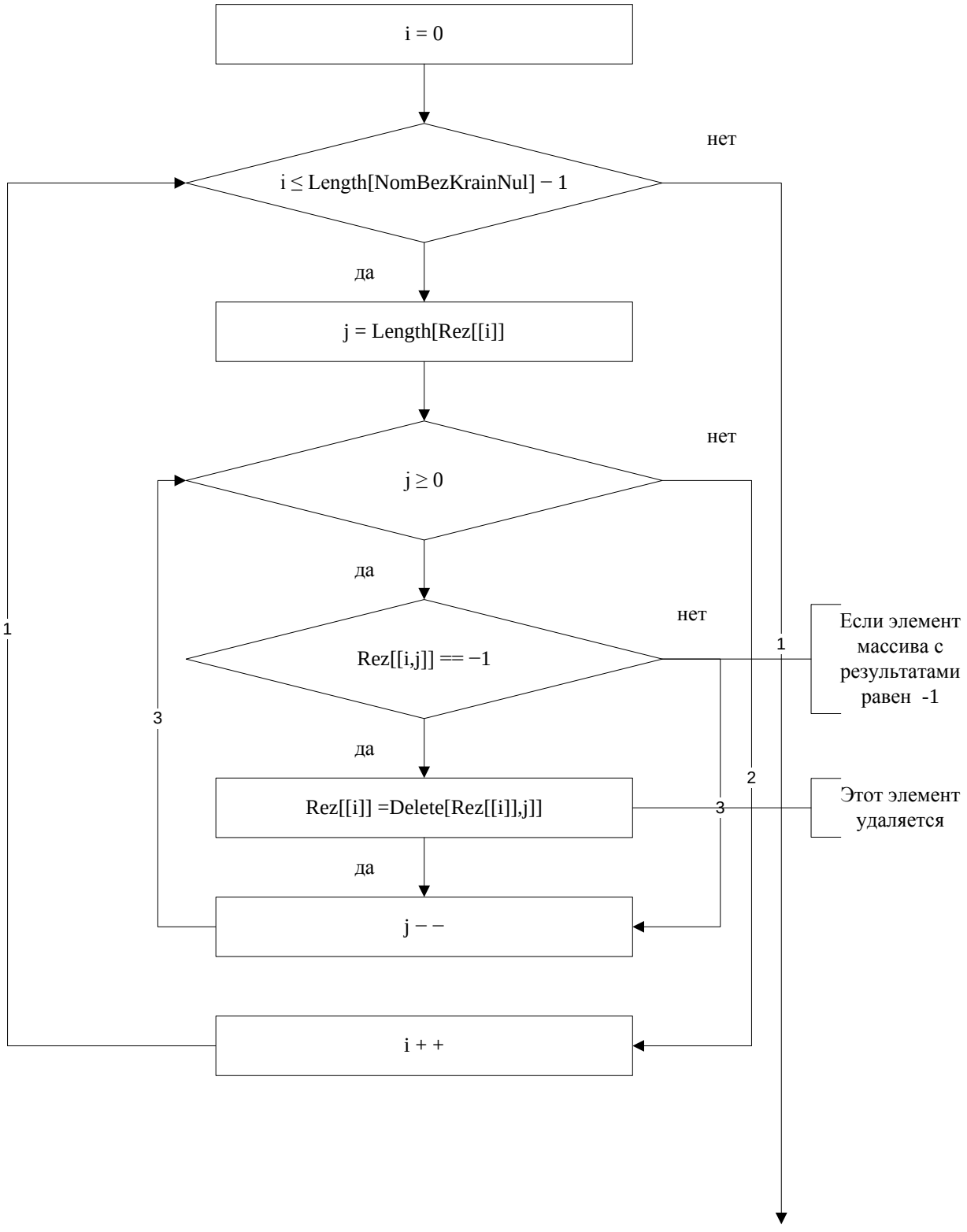


Приложение №22

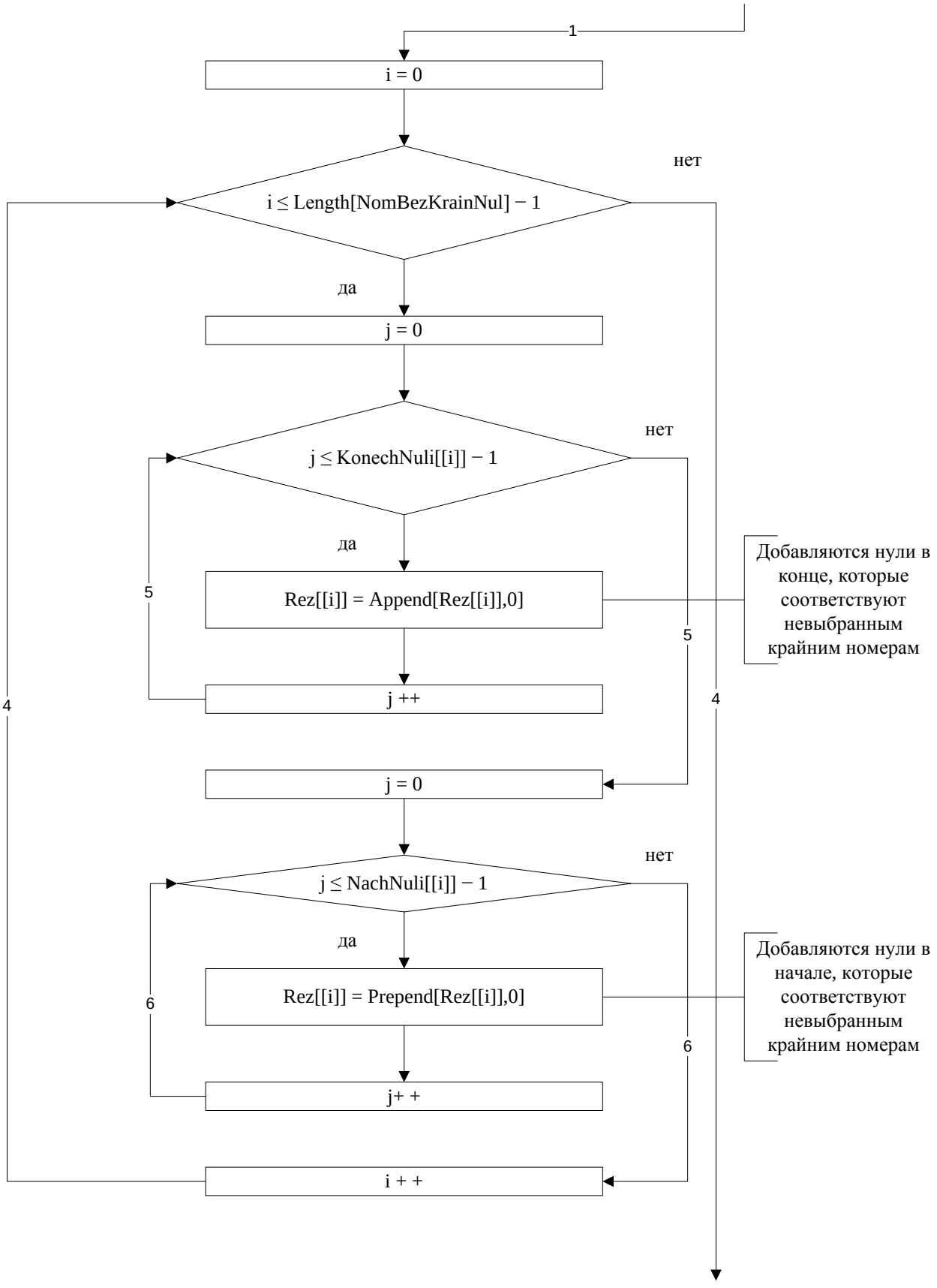


Приложение №23

Получение вектора μ.

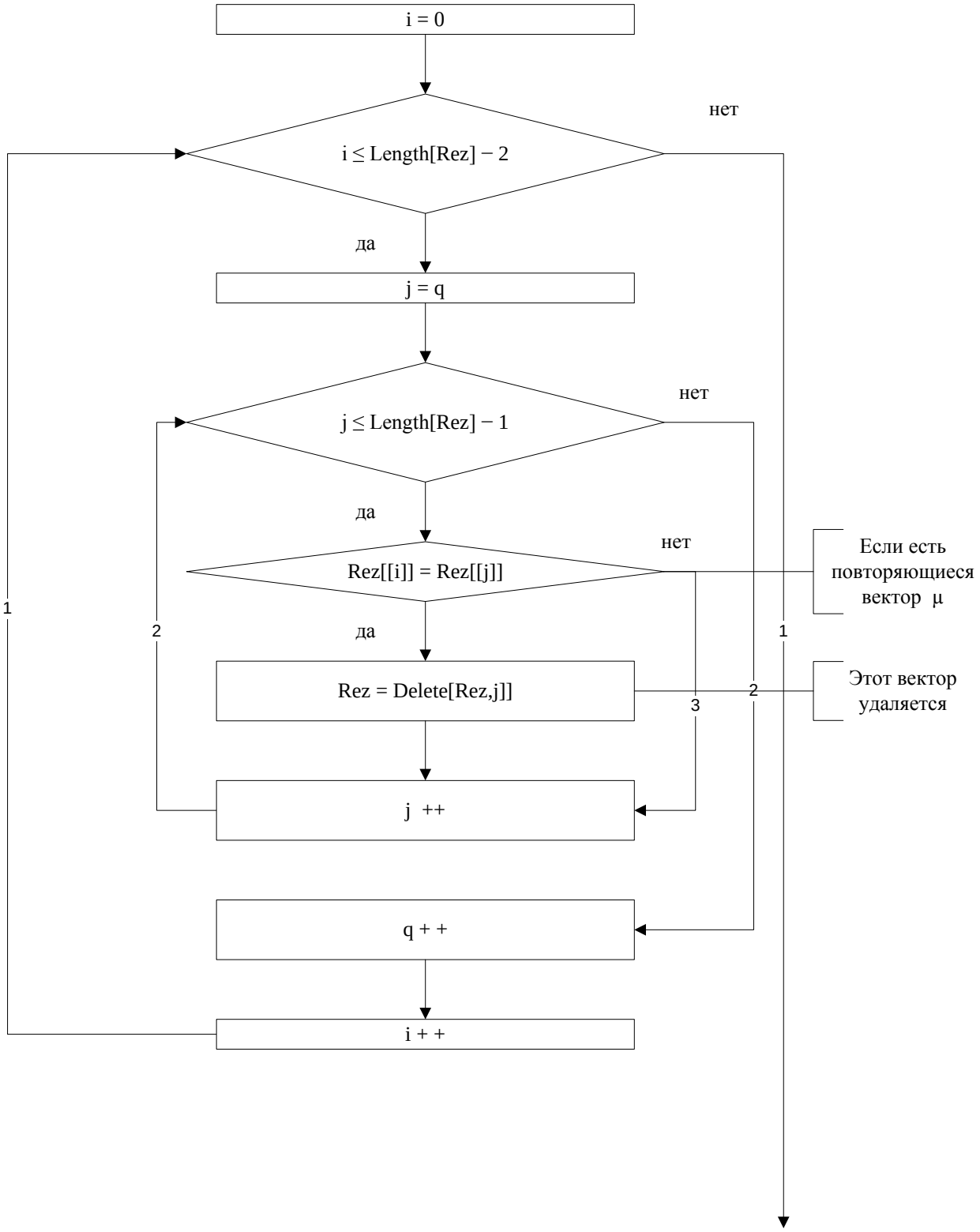


Приложение №23



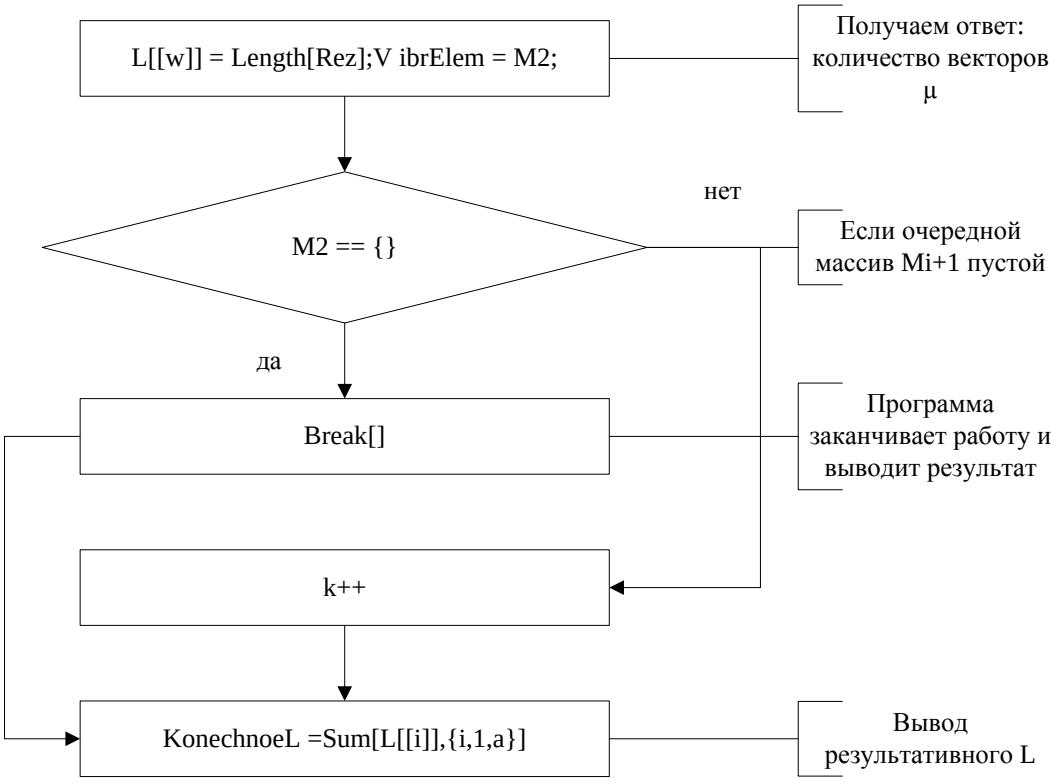
Приложение №24

Удаление повторяющихся исходов.



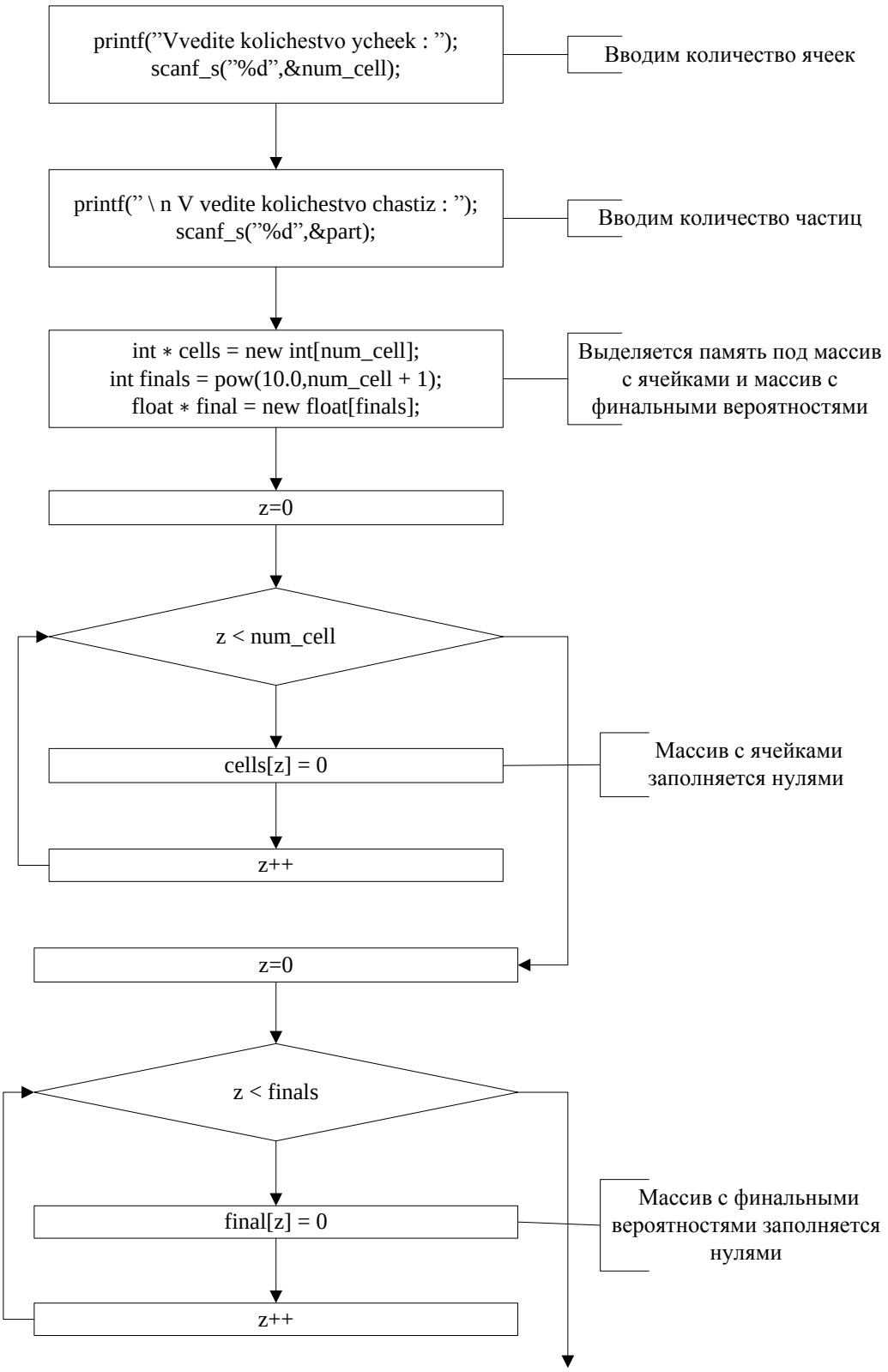
Приложение №25

Вывод результата в алгоритме моделирования схемы вторым способом.



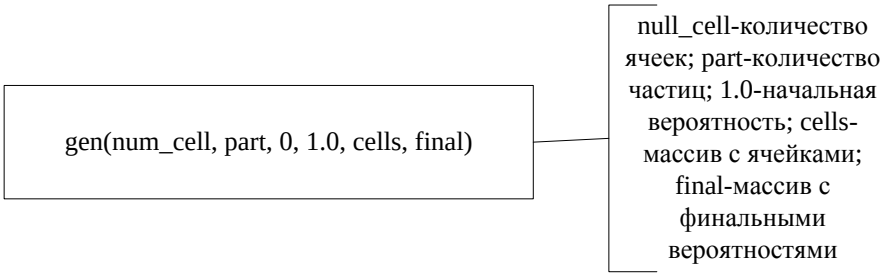
Приложение №26

Ввод начальных данных для программы 5

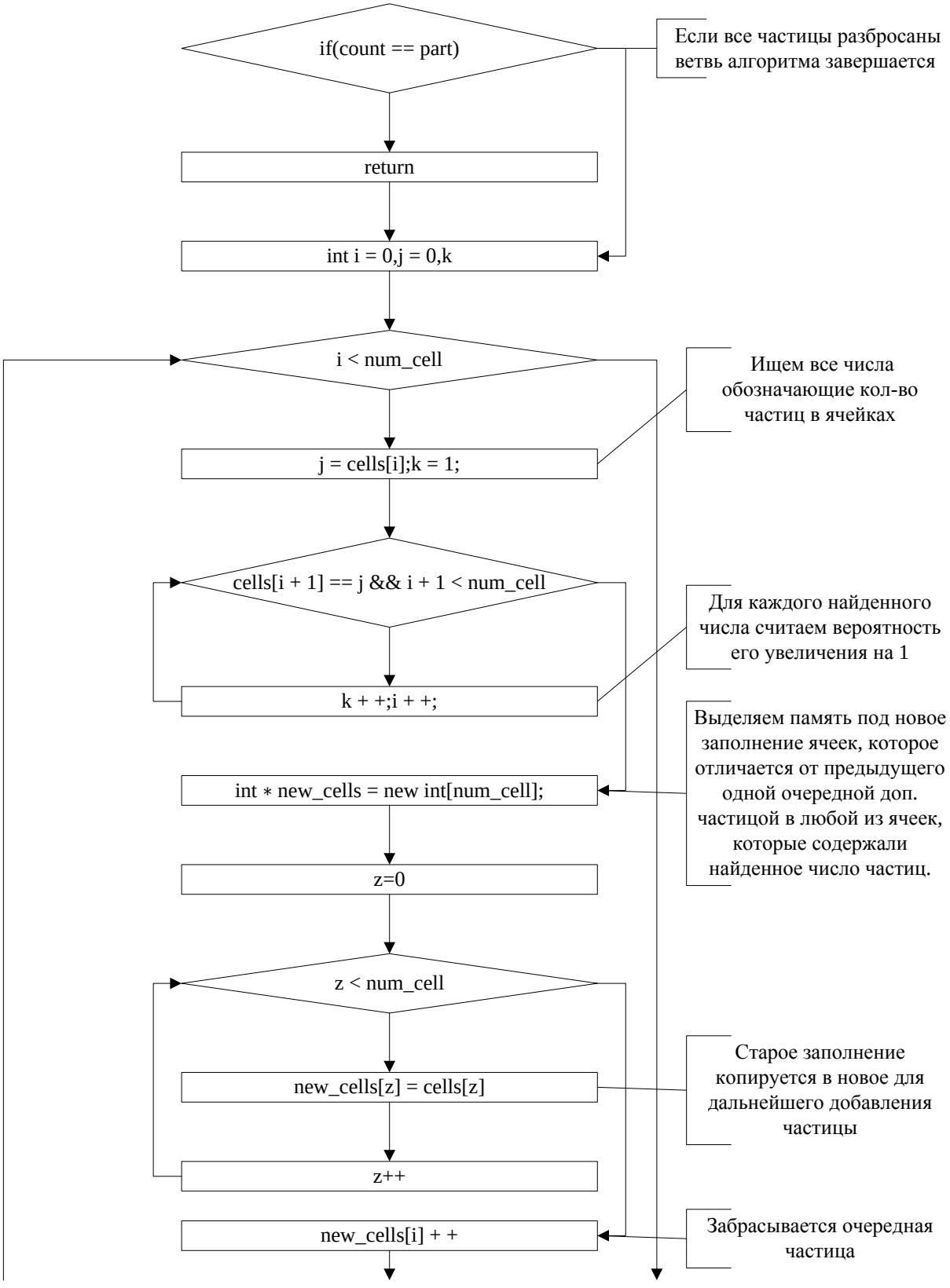


Приложение №27

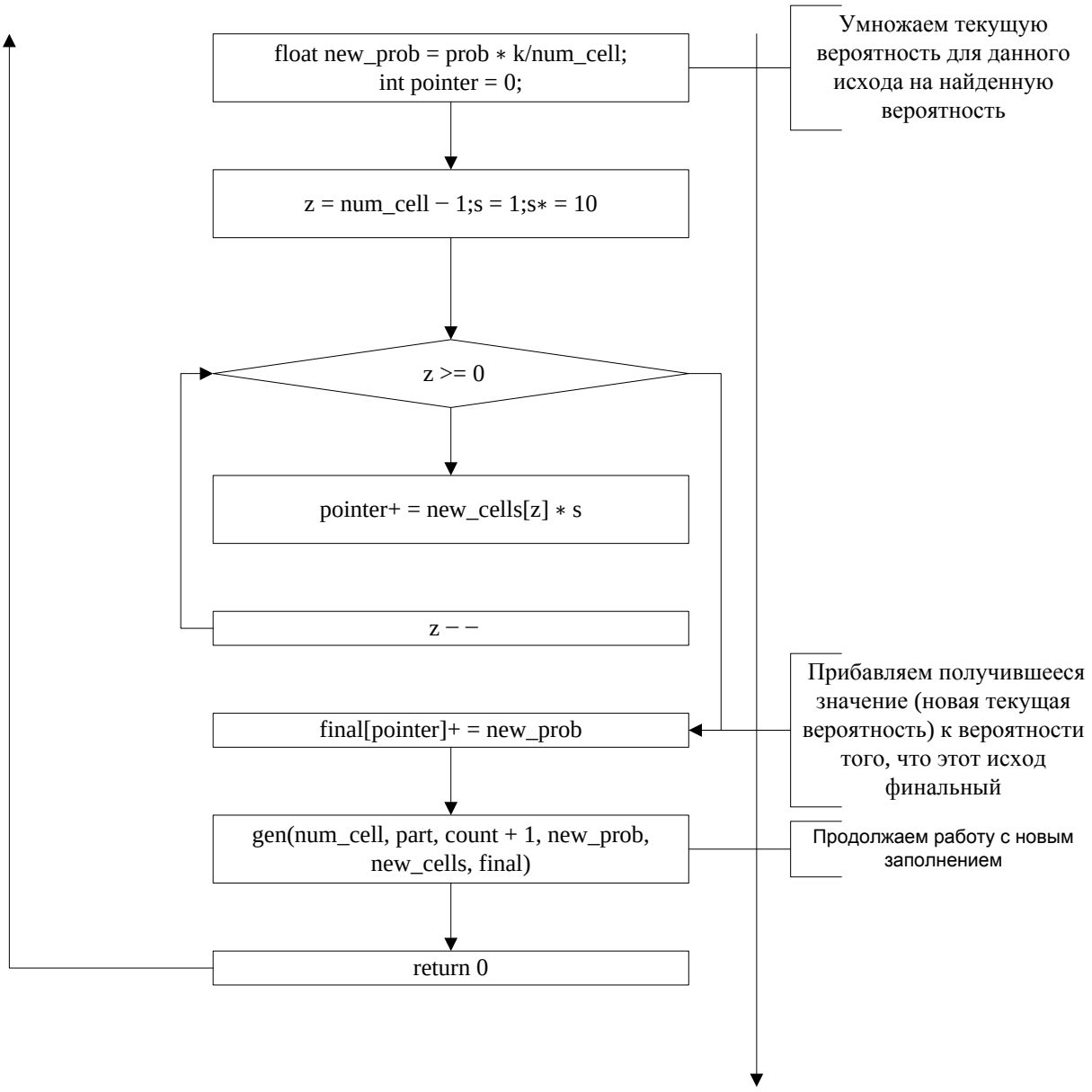
Вызов функции с начальными данными



Выполнение функции



Приложение №28



Приложение №29

Вывод результата работы программы 5