## Лабораторная работа 1

## Построение и исследование характеристик датчиков базовых случайных величин

***Базовой*** случайной величиной (БСВ) в статистическом моделировании называют непрерывную случайную величину $z$, равномерно распределенную на интервале (0,1). Ее плотность распределения вероятностей (**п.р.в**.) имеет вид:

$f(t) = 1,\ 0 < t < 1,$

Математическое ожидание (м.о.) и дисперсия БСВ составляют

$$M(z) = \frac{1}{2},$$

$$D(z) = \frac{1}{12},$$

соответственно.

БСВ моделируется на ЭВМ с помощью *датчиков* БСВ. Датчик БСВ – это устройство или программа, выдающая по запросу одно или несколько независимых значений $z_1$ , ..., $z_n$ БСВ.

Датчики БСВ могут быть трех типов: табличные, физические и программные.

***Табличный*** датчик БСВ – это просто таблица случайных чисел. Основной недостаток такого датчика – ограниченное количество случайных чисел в таблицах. А в статистическом эксперименте часто требуется не ограниченное заранее их количество.

***Физический*** датчик БСВ – это специальное радиоэлектронное устройство в ЭВМ, содержащее источник электронного шума. Шум преобразуется в случайные числа с распределением. Недостатки физического датчика БСВ: невозможность повторения каких-либо ранее полученных реализаций $z_1$, ... , $z_n$ без их предварительной записи в память ЭВМ, схемная нестабильность и сложность тиражирования датчика.

***Программный*** датчик БСВ обычно вычисляет значения $z_1$, $z_2$,..., по какой-либо рекуррентной формуле типа

$z_i = f(z_n),$

при заданном стартовом значении $z_0$.

Заданное значение $z_0$ полностью определяет всю последовательность реализаций $z_1$, $z_2$,..., поэтому $z$ часто называют *псевдослучайной* величиной. Но ее статистические свойства идентичны свойствам "чисто случайной" последовательности, что и обеспечивает успех статистического моделирования.

Программный датчик БСВ имеет следующие преимущества: простота создания датчика, простота применения, простота тиражирования, надежность, быстродействие, высокая точность достижения необходимых статистических свойств, сравнимая с точностью представления вещественных чисел, компактность, повторяемость, когда это нужно, любых

последовательностей случайных значений без их предварительного запоминания.

В дальнейшем мы будем рассматривать только программные датчики БСВ.

Имея датчик БСВ $z$, можно промоделировать любые случайные факторы: непрерывные или дискретные случайные величины (как простые, так и многомерные), случайные события, случайные процессы и поля и т.д. Для этого достаточно соответствующим образом преобразовать последовательность $z_1, z_2, \ldots$ . Поэтому БСВ $z$ и называют базовой.

Теоретически в качестве базовой можно было бы взять почти любую случайную величину (**с.в**.). Использование **с.в**. $z$ с распределением обусловлено технологическими соображениями: простотой и экономичностью датчика, простотой преобразования $z$ в другие случайные факторы, относительной простотой тестирования датчика.

**Метод середины квадрата**

Метод середины квадрата предложен для получения псевдослучайных чисел Д. фон Нейманом в 1946 г. Вот один из вариантов этого метода.

1. Возьмем произвольное 4-значное число.
2. Возведем полученное число в квадрат и, если необходимо, добавим к результату слева нули до 8-значного числа.
3. Возьмем четыре цифры из середины 8-значного в качестве нового случайного 4-значного числа.
4. Если нужны еще случайные числа, то перейдем к 2.

Например, если взять в качестве начального числа 1994, то из него получается следующая последовательность псевдослучайных чисел: 9760 2576 6357 4114 9249 5440 5936 2360 5696 4444 7491 1150 3225 4006 0480 2304 3084 5110 1121 2566 ...

Сам по себе метод середины квадрата не получил широкого распространения, так как выдает "больше чем нужно малых значений". Но открытый в нем принцип используется во многих, если не во всех, более поздних датчиках БСВ. Этот принцип состоит в вырезании нескольких цифр из результата какой-либо операции над числами.

**Мультипликативный конгруэнтный метод**

Так называемый мультипликативный конгруэнтный датчик БСВ задается двумя параметрами: модулем $m$ и множителем $k$. Обычно это достаточно большие целые числа.

При заданных $m$, $k$ числа $z_1, z_2, \ldots$, вычисляются по рекуррентной формуле:

$A_i = (kA_{i-1}) \bmod m,\ i = 1, 2, \ldots,$ (2.5)

$z_i = A_i / m,$

где $m$ − модуль, $k$ − множитель, $A_0$ − начальное значение, mod − операция вычисления остатка от деления $kA_{i-1}$ на $m$.

Таким образом, $A_1$ определяется как остаток от деления $kA_0$ на $m$; $A_2$ - как остаток от деления $kA_1$ на $m$ и т.д. Поскольку все числа $A_i$ − это остатки от деления на $m$, то $0 \le A_i < m$. Разделив последнее неравенство на $m$, видим, что $0 \le A_i / m < 1$, т. е. $0 \le z_i < 1$.

Из неравенства $0 \le A_i < m$ вытекает также, что датчик (2.5) дает периодическую последовательность $A_i$. Действительно, число всех возможных остатков от 0 до $m$ - 1 равно $m$ и, рано или поздно, на каком-то шаге $i$ обязательно появится значение $A_i$, уже встречавшееся ранее. С этого момента последовательность $A_i$ “зациклится".

Длина периода $T$ будет не больше $m$ - 1. Например, если встретится остаток $A_i = 0$, то далее, согласно (2.5), будет $A_{i+1} = 0$, $A_{i+2} = 0$, ... , т.е. длина периода $T$ = 1. Ненулевых же остатков в интервале $0 \le A_i < m$ всего $m$ - 1, и, если все они войдут в период, будет $T = m - 1$. Это имеет место, например, при $m$ = 13, $k$ = 7; в этом случае ряд $A_i$ выглядит так:

1, 7, 10, 5, 9, 11, 12, 6, 3, 8, 4, 2, 1, 7,... .

_____/

$T = m - 1 = 12$

Поскольку в качестве случайной можно использовать лишь подпоследовательность $A_i$ внутри одного периода, то параметры датчика выбирают так, чтобы длина периода $T$ была максимальной. С учетом ограничения $T \le m - 1$ модуль $m$ берут максимально возможным. Чтобы упростить вычисление остатков по (2.5), для двоичных ЭВМ часто берут $m = 2^n$. Рекомендуется также брать достаточно большой множитель $k$, причем взаимно простой с $m$.

В можно найти подробные рекомендации по выбору параметров $m$, $k$ и начального значения $A_0$ . Заметим, однако, что в настоящее время не известны правила, которые гарантировали бы высокое качество датчика без его специального статистического тестирования.

Датчик (2.5) называют мультипликативно-конгруэнтным потому, что он использует две основные операции − умножение (англ. multiplication) и вычисление остатка (в теории чисел − получение конгруэнтного числа). Можно было бы поэтому перевести его название и как "множительно-остатковый датчик".

Обратим внимание также и на то, что операция вычисления остатка воплощает здесь упоминавшийся в п. 2.2 неймановский принцип вытаскивания цифр. Это становится очевидным, если записывать числа в системе счисления с основанием $m$. Тогда операция $X \bmod m$ означает выбор последней цифры из числа $X$. Для $m = 2^n$ операция $X \bmod m$

**Тестирование равномерности**

Обозначим равномерное распределение вероятностей на интервале (0,1) через $R[0,1]$. Тогда утверждение, что БСВ $z$ имеет распределение $R[0,1]$, можно кратко записать в виде $z \sim R[0,1]$.

С помощью статистических тестов проверяют два свойства датчика, делающих его точной моделью идеальной БСВ, − это *равномерность* распределения чисел $z_i$, выдаваемых датчиком на интервале (0,1), и их статистическая *независимость*. При этом числа $z_i$ рассматривают как реализации некоторой **с.в.**, т.е. как статистическую выборку.

Достаточно простым методом проверки равномерности распределения является частотный тест. Он основан на законе больших чисел и выполняется по следующему алгоритму.

1. Разобьем интервал (0,1) на $K$ равных отрезков (например, $K = 10$).
2. Сгенерируем $n$ чисел $z_1,..., z_n$ с помощью тестируемого датчика БСВ (например, $n = 100$).
3. Подсчитаем, сколько чисел попало в каждый из $k$ отрезков, т.е. найдем числа попаданий $n_1,...,n_k$.
4. Рассчитаем относительные частоты попаданий в отрезки: $\hat{p} = \frac{n_1}{n},..., \hat{p} = \frac{n_k}{n}$
5. Построим гистограмму частот $\hat{p}_1,..., \hat{p}_k$ на $K$ отрезках интервала (0,1).
6. Повторим действия (2) – (5) для большего значения $n$ (например, для $n$ =10 000).
7. Оценим по полученным гистограммам сходимость каждой частоты $\hat{p}_i$ к вероятности $p = 1/K$ того, что БСВ попадет в $i$-й отрезок. Согласно закону больших чисел должно быть

$$\hat{p}_i \xrightarrow{p} \frac{1}{K}, \qquad n \to \infty \quad (2.6)$$

Это значит, что высоты столбиков во второй гистограмме должны в целом быть ближе к уровню $1/K$, чем в первой.

Тестирование датчика на равномерность можно совместить с оцениванием **м.о**. и дисперсии **с.в.** Оценки $\hat{M}$ и $\hat{D}$ для **м.о.** и дисперсии рассчитываются соответственно по формулам:

$$\hat{M} = \frac{1}{n}\sum_{i=1}^{n} z_i \qquad (2.7)$$

$$\hat{D} = \frac{1}{n}\sum_{i=1}^{n} z_i^2 - (\hat{M})^2 \qquad (2.8)$$

С ростом $n$ оценки $\hat{M}$ и $\hat{D}$ должны сходиться по вероятности к точным значениям $M(z) = 1/2$, $D(z) = 1/12 = 0.08333...$ .

**Тестирование независимости**

Простейшую проверку статистической независимости реализаций $z_1$, $z_2$, ..., можно осуществить, оценивая корреляцию между числами $z_i$ и $z_{i+s}$, отстоящими друг от друга на шаг $s$ >1.

Для вывода формулы, по которой можно рассчитать коэффициент корреляции чисел $z_i$ и $z_{i+s}$, рассмотрим две произвольные **с.в**. $x$, $y$. Коэффициент корреляции определяется для них формулой:

$$R(x,y) = \frac{M(xy) - M(x)M(y)}{\sqrt{D(x)D(y)}} \qquad (2.9)$$

Если известно, что $x$, $y \sim R[0,1]$, то $M(x) = M(y) = 1/2$ и $D(x) = D(y) = 1/12$, то есть (2.9) принимает вид:

$R(x,y) = 12\, M(xy) - 3.$ (2.10)

Условимся рассматривать пару чисел $(z_i , z_{i+s})$ как реализацию пары **с.в.** $(x,y)$. Тогда в выборке $z_1,..., z_n$ имеем всего $n - s$ реализаций этой пары:

$(z_1 , z_{1+s}),(z_2 , z_{2+s}),... ,(z_n , z_{n+s})$.

По ним можно рассчитать оценку $R'$ коэффициента корреляции $R(x,y)$, заменяя в (1.10) **м.о.** $M(xy)$ соответствующим средним арифметическим:

$$R' = 12 \frac{1}{n-s} (\sum_{i=1}^{n-s} z_i z_{i+s}) - 3 \qquad (2.11)$$

С ростом $n$ оценка $R'$ должна приближаться к нулю, в противном случае датчик БСВ не отвечает требованию независимости.

Конечно, если $R'$ сходится к нулю, то это еще не гарантирует наличие независимости, но все же один из тестов оказывается успешно выдержанным. При желании всегда можно продолжить испытания датчика другими методами.

**Задание.**

Написать программы, реализующие рассмотренные методы построения датчиков случайных величин.

Выполнить статистическое исследование датчиков.

Сравнить результаты.

## Лабораторная работа 2
## Имитация случайных событий

### Имитация случайного события

Пусть некоторое событие А происходит с вероятностью $P_A$. Требуется воспроизвести факт наступления события А. Поставим в соответствие событию А событие В, состоящее в том, что $x$ меньше либо равно $P_A$, где х здесь и в дальнейшем – случайное число (СЧ) с равномерным на интервале (0,1) законом распределения. Вычислим вероятность события В:

$$P(B) = \int_0^P 1 dy = P_A$$

Таким образом, события А и В являются равновероятными. Отсюда следует процедура имитации факта появления события А. Она сводится к

проверке неравенства $X_A$ меньше, либо равно Р, а алгоритм заключается в следующем:

1. С помощью датчика случайных чисел (СЧ) получают СЧ *X*;
2. Проверяют выполнение неравенства Х меньше, либо равно $P_A$;
3. Если оно выполняется, то событие А – произошло, если нет – то произошло $\overline{A}$

**Имитация сложного события**

Имитация сложного события, состоящего, например, из двух независимых элементарных событий А и В, заключается в проверке неравенств:

$$\left.\begin{array}{l} x_1 \le P_A \\ x_2 \le P_A \end{array}\right\},$$

где $P_A$ и $P_B$ – вероятности событий А и В, а $x_1$ и $x_2$ – СЧ с равномерным законом распределения.

В зависимости от исхода проверки неравенств (аналогично алгоритму 4.2.1.) делается вывод какой из вариантов:

$AB, A\overline{B}, \overline{A}B, \overline{A}\overline{B}$ имеет место.

**Имитация сложного события, состоящего из зависимых событий.**

В случае, когда сложное событие состоит из элементарных зависимых событий А и В имитация сложного события производится с помощью проверки следующих неравенств:

$$\left.\begin{array}{l} x_1 \le P_A \\ x_2 \le P_{B/A} \end{array}\right\} \quad \left.\begin{array}{l} x_1 > P_A \\ x_2 \le P_{B/\overline{A}} \end{array}\right\} \quad \left.\begin{array}{l} x_1 \le P_A \\ x_2 > P_{B/A} \end{array}\right\} \quad \left.\begin{array}{l} x_1 > P_A \\ x_2 > P_{B/\overline{A}} \end{array}\right\}$$

В зависимости от того, какая из этих четырех систем неравенств выполняется, делается вывод о том, какой из этих четырех возможных исходов $AB, A\overline{B}, \overline{A}B, \overline{A}\overline{B}$ имеет место.

В качестве исходных данных задаются $P_A, P_B$ и условная вероятность $P_{B/A}$, вероятность $P_{B/\overline{A}}$ может быть вычислена. По формуле полной вероятности:

$$P(A) = P(A/B) \cdot P(B) + P(A/\overline{B}) \cdot P(\overline{B}),$$

где

$P(\overline{B}) = 1 - P(B)$, отсюда легко выразить $P(A/\overline{B})$

**Имитация событий, составляющих полную группу**

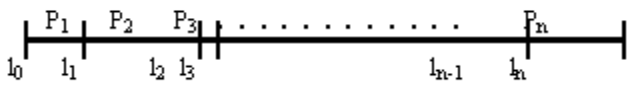
Пусть событие $A_i$ (i=1,n) составляют полную группу, тогда их вероятности $P_i$, таковы что:

$$\sum_{i=1}^{n} P_i = 1$$

Имитация факта появления одного из событий $A_i$ (i=1,n) сводится к проверке следующих неравенств:

$$\sum_{i=0}^{K-1} P_i \le x < \sum_{i=0}^{K} P_i, \qquad K = \overline{1,n}, \qquad P_0 = 0$$

Выполнение К-го неравенства эквивалентно выполнению события $A_K$. Описанный алгоритм называют иногда алгоритмом “розыгрыша по жребию”. Его можно интерпретировать как установление номера К-го отрезка длинной $P_K$, на который пало СЧ x, при условии разбиения отрезка единичной длины на отрезки с длинами $P_1,P_2,...P_n$ (рис 4.3.)




$$l_k = \sum_{i=0}^{K} P_i$$

Задание

Написать программы, реализующие рассмотренные методы имитации случайных событий.
Провести статистические исследования полученных результатов..
Проверить гипотезы о соответствии полученных результатов требуемым.

## Лабораторная работа 3

## Имитация непрерывных случайных величин (метод обратных функций)

Случайные числа с заданным законом распределения вероятностей, как правило, формируются в результате преобразования случайных равномерно распределенных чисел R[0,1]. В настоящее время известно много процедур, позволяющих имитировать непрерывные и дискретные вероятностные распределения. Рассмотрим содержание двух наиболее распространенных на практике процедур.

**Процедура 1.** (для непрерывных распределений). Пусть имеется непрерывная случайная величина X, распределенная с постоянной плотностью в интервале (0, 1), которая описывается плотностью распределения

$$f(x) = \begin{cases} 1, & x \in [0,1) \\ 0, & x \notin [0,1) \end{cases}$$

Требуется путем функционального преобразования $Y=\varphi(X)$ получить случайную величину с заданной функцией распределения $G(y)$. Покажем, что для этого надо подвергнуть равномерно распределенную случайную

величину X функциональному преобразованию

$$Y = G^{-1}(x),$$

где $G^{-1}$ – функция, обратная требуемой функции распределения $G(y)$.

Поскольку функция распределения непрерывна и монотонна, то и обратная функция $G^{-1}$ также непрерывна и монотонна. В этом случае функция распределения случайной величины Y определяется так:

$$P\{Y < y\} = P\{X < G(y)\} = \int_{-\infty}^{G(y)} f(x)dx = \int_{-\infty}^{G(y)} 1 \cdot dx = G(y).$$

Следовательно, для получения значения $y$ непрерывной случайной величины Y нужно выполнить следующее:

1. Получить значение случайной величины $X$, распределенной равномерно на интервале (0, 1).
2. Найти обратную функцию $G^{-1}(x)$ по отношению к требуемой функции распределения $G(y)$ и вычислить значение случайной величины Y по формуле:

$$y = G^{-1}(x).$$

**Задание**

Написать программу реализующую метод формирования непрерывной случайной величины.
Выполнить статистическое исследование (построение гистограммы, точечных, интервальных оценок)
Проверить гипотезы о соответствии закона распределения полученной случайной величины требуемому.

## Лабораторная работа 4
## Имитация дискретных случайных величин с заданным законом распределения

Функция распределения $G(y)$ дискретной случайной величины $Y$ представляет собой ступенчатую функцию; вероятность $P\{Y=y_i\}=p_i$ $(i=1, \ldots, n)$ равна величине скачка функции распределения $G(y)$ в точке $y_i$. Таким образом, участок оси ординат от 0 до 1 можно разбить на $n$ непересекающихся отрезков:

$$\Delta_1=(0, p_1);\ \Delta_2=(p_1, p_1+p_2); \ldots$$

$$\Delta_i=(p_1+ \ldots+p_{i-1}, p_1+ \ldots+p_i); \ldots \Delta_n=((\sum_{i=1}^{n-1} p_i, 1).$$

При таком разбиении длина i-го отрезка $\Delta_i$ равна $p_i$ $(i=1, \ldots, n)$. Способ получения дискретной случайной величины $Y$.

1. Разбить интервал (0,1) на непересекающиеся участки $\Delta_i$ $(i=1, \ldots, n)$ длиной $p_1, p_2, \ldots, p_n$.

2. Получить значение случайной величины $X$, распределенной равномерно на интервале (0, 1).
3. Определить, какому из интервалов $\Delta_i$ принадлежит значение случайной величины $x$. Если $x \in \Delta_i$, то случайная величина $Y=y_i$.

**Задание**

Написать программу реализующую метод формирования дискретной случайной величины.
Выполнить статистическое исследование (построение гистограммы, точечных, интервальных оценок)
Проверить гипотезы о соответствии закона распределения полученной случайной величины требуемому.

## Лабораторная работа 5
## Имитация систем случайных величин

Дискретный двумерный вектор CDCB задается двумерным законом распределения, т.е.

а) матрицей вероятностей $\|P_{ij}\|, i=\overline{1,n}, j=\overline{1,m}$, где $P_{ij}$ – вероятность совместного появления i-ого и j-ого значений соответственной первой и второй компоненты, причем:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} P_{ij} = 1.$$

б) двумя векторами возможных значений первой и второй компоненты $\{A_i\}$, $\{B_i\}$, $i=\overline{1,n}, j=\overline{1,m}$.

Получение значений двумерной дискретной системы случайных величин может осуществляться по следующему алгоритму.

Вычисляют суммы $q_i = \sum_{j=1}^{m} P_{ij}$, $l_k = \sum_{i=1}^{k} q_i$, $k=\overline{1,n}$.

Если $X$ - равномерно распределенное случайное число из интервала (0,1) такое, что $l_{k-1} < x \le l_k$, то считают, что $x_1$ компонента двумерной дискретной случайной величины получила k-ое значение.

Выбирают k-ую строку $\|P_{ij}\|$, вычисляют $r_S = \sum_{j=1}^{s} P_{kj}$.

Если вновь полученное с помощью датчика случайных чисел $X$ такое, что вторая компонента получила S-е значение.

Замечание: В алгоритме используется правило “розыгрыша по жребию”, однако надо иметь в виду, что $r_S \neq 1$.

Правило получения значений $y_1, \ldots, y_n$, системы случайной величины $(Y_1, \ldots, Y_n)$ сводится к следующему:

1. «Разыгрывается» значение $x_1$ случайной величины $X_1$, распределенное равномерно в интервале (0,1) и по функции распределения $G_1(y_1)$ получаем $y_1$ – значение случайной величины $Y_1$: $y_1 = G_1^{-1}(x_1)$, где $G_1^{-1}(x_1)$ – функция, обратная $G_1(y_1)$.
2. «Разыгрывается» значение $x_2$ случайной величины $X_2$, распределенное равномерно в интервале (0,1) и по функции распределения $G_{2/1}(y_2/y_1)$ получаем $y_2$ – значение случайной величины $Y_2$: $y_2 = G_{2/1}^{-1}(x_2 / y_1)$, где $G_{2/1}^{-1}(x_2/y_1)$ – функция, обратная $G_{2/1}(y_2/y_1)$. В качестве аргумента $y_1$ функции распределения $G_{2/1}(y_2/y_1)$ берется то значение $y_1$, которое было получено в пункте 1.
3. «Разыгрывается» значение $x_3$ случайной величины $X_3$, распределенное равномерно в интервале (0,1) и по функции распределения $G_{3/1,2}(y_3/y_1,y_2)$ получаем $y_3$ – значение случайной величины $Y_3$: В качестве аргументов $y_1$, $y_2$ функции распределения $G_{3/1,2}(y_3/y_1,y_2)$ берутся значения $y_1$ и $y_2$, которые были получены в пункте 1 и 2. И так далее.

**Задание**

Написать программу реализующую метод формирования двумерной случайной величины.
Выполнить статистическое исследование полученной величины (построение эмпирической матрицы распределения, гистограммы составляющих вектора, вычисление точечных, интервальных оценок, коэффициент корреляции)
Проверить гипотезы о соответствии полученных оценок характеристик случайной величины требуемым.

## Лабораторная работа 6
## Построение имитационных моделей

### Структура модели СМО (Q-схемы)

Для детального ознакомления с технологией машинной имитации рассмотрим Q-схему достаточно общего вида (рис.6.1). Q-схема содержит три фазы обслуживания и источник заявок.

Первая фаза содержит 2 однотипных канала $К_{11}$ и $К_{12}$ и общий входной накопитель заявок Н1. В случае заполнения накопителя Н1 заявки источника получают отказ (дисциплина отказа заявкам на входе фазы I).

Вторая фаза также содержит два однотипных канала $К_{21}$,$К_{22}$ и общий входной накопитель Н2. В случае заполнения накопителя Н2 заявки блокируются в первой фазе. Это означает, что если какой -либо канал $К_{21}$ или $К_{22}$ в некоторый момент модельного времени завершил обслуживание заявки и в этот момент каналы второй фазы заняты и накопитель заполнен, то

обслуженная заявка не покидает систему, что имеет место в случав отказа, а блокируется в канале первой фазы. Заявка сохраняется каналом первой фазы до тех пор, пока в накопителе Н2 не освободится по крайней мере одна позиция.

Третья фаза содержит только один канал $К_{31}$ и накопитель Н3 емкостью, равной нулю. При занятом канале $К_{31}$ заявки блокируются во второй фазе.

Для описания имитационной модели Q-схемы введем следующие переменные *[2]:*

$t_n$ - текущее значение модельного времени;

$t_m$ -время появления очередной заявки на выходе источника;

$t_{kj}$ -время окончания обслуживания каналом j k-й фазы К очередной заявки;

$Z_{kj}$ ($t_n$) - состояние канала j фазы k в момент t;

$L_i$ - емкость накопителя i-й фазы;

$Z_i$ - состояние накопителя i-й фазы;

$N_1$ - количество потерянных заявок;

$N_3$ - количество обслуженных системой заявок;

P - вероятность отказа (потери) заявки системой;

$\Delta t$ - интервал продвижения модельного времени в сплошном моделировании.

Каждый из каналов Q-схемы может находиться в следующих состояниях:

1) канал свободен (0);

2) канал занят обслуживанием (I);

3) канал заблокирован (хранит уже обслуженную заявку)(2). Текущее состояние Z накопителя Н равно количеству заявок , хранящемуся в накопителе в текущий момент модельного времени t.

## Алгоритм моделирования Q-схемы

Процедура моделирования начала обслуживания заявки каждым элементарным каналом $К_{ij}$ - сводится к следующему.

Выполняется обращение к генератору случайных чисел. Генератор формирует интервал обслуживания заявки каналом $К_{ij}$, закон распределения , длительности которого должен соответствовать закону F распределения времени обслуживания заявок каналом $К_{ij}$. Вычисляется время окончания обслуживания $t_{ij}=t_n+\tau_{ij}$, где $t_n$ - текущий момент модельного времени. Канал $К_{ij}$ переходит в состояние "занят обслуживанием".

Когда модельное время достигает значения $t_{ij}$, соответствующего моменту завершения обслуживания каналом $К_{ij}$, моделируется процесс передачи заявки с выхода канала $К_{ij}$ в накопитель $Н_{i+1}$ следующей фазы или в каналы фазы i+1, если ёмкость $L_{i+1}$ накопителя $Н_{1+1}$ равна нулю. Если фаза i+1 может принять заявку, то канал $К_{ij}$ переводится в состояние "свободен". В этом случае количество заявок в накопителе $Н_{i+1}$ фазы i+1 увеличивается на 1, а канал $К_{ij}$ может принять заявку из накопителя $Н_i$ своей фазы. Канал $К_{ij}$

переходит в состояние "занят обслуживанием", а количество заявок в накопителе $H_i$ уменьшается на единицу. Если фаза i+1 заявку принять не может (накопитель и каналы заняты обслуживанием заявок), канал $K_{ij}$ переводится в состояние "заблокирован".

Укрупнённая схема алгоритма моделирования Q-схемы, построенного по принципу последовательного просмотра состояний модели через фиксированный временной интервал Δt, представлена на рис.3.2. Такой метод управления модельным временем называется моделированием с постоянным шагом и состоит в том, что после каждого просмотра состояния модели, модельное время $t_n$ увеличивается на интервал Δt. Наращивание модельного времени $t_n = t_n + \Delta t$ выполняется блоком 10. Момент завершения моделирования Q-схемы может быть зафиксирован по числу просмотров N, по длине интервала времени моделирования T или по количеству обслуженных заявок N1. Проверка соответствующих условий выполняется блоком 3.

Работа вспомогательных блоков - ввода исходных данных *I,* установки начальных условий 2, обработки *II* и вывода результатов моделирования 12 - не отличается по своей сути от аналогичных блоков, используемых в алгоритмах вычислений на ЭВМ. Поэтому остановимся более детально на работе той части моделирующего алгоритма, которая отражает специфику моделирования подхода (блоки 4-9). Детализированные схемы алгоритмов этих блоков приведены на рис 3.3 - рис 3.8. На этих и последующих схемах моделирующих алгоритмов Q-схем приняты следующие обозначения: ZN(1) = z , Z(i, J) = $z_{ij}$, TM=$t_m$, TN=$t_n$, T(i,J) = $t_{ij}$, LO(I) =$L_i$, PO = P.

Процедура формирования времени завершения обслуживания заявок каналами $K_{ij}$ оформлена в виде подпрограммы WORK (T(K,J)). Процедура генерирует, $T_{kj}$ - длительность интервала обслуживания очередной заявки и формирует время завершения обслуживания $t(K,j) = t_n + T_{kj}$. Окончание обслуживания заявки в некотором канале $K_{ij}$ в момент времени $t_n$ может вызвать процесс распространения изменений состояний элементов ("особых состояний") системы в направлении противоположном движению заявок в системе, поэтому все H (накопители) и K (каналы) системы должны просматриваться при моделировании, начиная с обслуживающего канала последней фазы по направлению к накопителю 1-й фазы (см. рис. . ).

## Алгоритм формирования очередного состояния Q-схемы в дискретный фиксированный момент модельного времени

Рассмотрим реализацию основных блоков моделирующего алгоритма. Это блоки 9, 8..... 4, которые имитируют формирование заявок источником и их обслуживание в каналах 1-й, 2-й и 3-й фаз модели.

Рассмотрим состояние модели Q-схемы на стационарном участке моделирования. Пусть после очередного выполнения блока 10 модельное время приняло значение $t_n$.

Блок 4 (рис.6.3) имитирует завершение обслуживания заявок каналом $К_{31}$ третьей фазы. Блок 4.1 проверяет состояние канала $К_{31}$ и, если канал находится в состоянии "занят обслуживанием" ("I"), то в блоке 4.2 проверяется время $Т_{31}$ завершения обслуживания каналом $К_{31}$. Если это время меньше или совпадает с текущим модельным временем $t_n$, то это означает, что в момент $t_n$ на выходе $К_{31}$ появляется очередная заявка. В этом случае в блоке 4.3 увеличивается на 1 количество обслуженных заявок N3, а в блоке 4.4 канал $К_{31}$ переводится в состояние "свободен" ("O").

Блок 5 (рис.6.4), имитирует завершение обслуживания заявок каналами 2-й фазы и передачу обслуженных заявок в 3-ю фазу. Блоки 5.1, 5.9 и 5.10 составляют цикл просмотра каналов 2-й фазы. Блоки 5.2 и 5.3 проверяют состояние и время завершения обслуживания заявки каждым из каналов. Если для некоторого канала j его состояние $Z_{2j}=0$, т.е. он находится в состоянии "занят обслуживанием" или "заблокирован", $Т_{2j}<=Т_n$, то это означает, что канал $К_{2j}$ хранит ранее заблокированную заявку ( $Z_{2j} = 2$ и $Т_{2j} < Т_n$) или именно в момент $Т_n$ он завершил обслуживание ($Т_{2j} = Т_n$, $Z_{2j} < 1$). В этих случаях блок 5.4 проверяет состояние канала 3.1 3-й фазы. Если этот канал не свободен ($Z_{31}\neq0$), то блок 5.5 переводит канал $К_{ij}$ в состояние "заблокирован" (или подтверждает ранее установленное состояние "заблокирован"). Если $Z_{31}=0$, то блок 5.6 формирует новое время завершения обслуживания заявки каналом $К_{31}$, блок 5.7 переводит канал $К_{31}$ в состояние "занят обслуживанием", а блок 5.8 освобождает канал $К_{2j}$.

Блок 6 (рис.6.5) имитирует процесс передачи заявок из накопителя Н2 второй фазы в каналы $К_{21}$, $К_{22}$. Блоки 6.1, 6.7, 6.8 составляют цикл просмотра состояния каналов второй фазы. Блок 6.2 проверяет состояние накопителя Н2. Если накопитель Н2 содержит хотя бы одну заявку ($ZN(2)\neq0$), выполняется переход к блоку 6.3, который проверяет состояние очередного канала 2-й фазы. Если j-й канал свободен ($Z(2,j)=0$), то в блоке 6.4 вычисляется время завершения обслуживания заявки каналом $К_{2j}$, блок 6.5 переводит канал $К_{2j}$ в состояние "занят обслуживанием", а блок 6.6 уменьшает на единицу количество заявок в накопителе Н2. Если при выполнении блока 6.2 оказывается, что накопитель Н2 заявок не содержит ($ZN(2)=0$), то выполняется переход к блоку 7.

Блок 7 (рис.3.6) воспроизводит процесс передачи заявок из каналов 1-й фазы в накопитель и каналы 2-й фазы. Блоки 7.1, 7.15, 7.16 составляют цикл просмотра состояния каналов 1-й фазы. Если при выполнении блоков 7.2,7.3 оказывается, что некоторый канал $К_{1j}$ хранит заявку в состоянии "заблокирован" или выработал заявку в момент $t_n$, выполняется переход к блокам 7.4, 7.5, 7.6, 7.7, составляющим цикл просмотра состояния каналов 2-й фазы. Если в результате выполнения в цикле блока 7.5 находится некоторый канал $К_{2i}$ в состоянии "свободен" ($Z(2,i)=0$), то выполняются блоки 7.8, 7.9, 7.10. Эти блоки формируют время завершения обслуживания заявки каналом $К_{2i}$, канал $К_{2i}$ переводится в состояние "занят обслуживанием", а канал $К_{1j}$ переводится в состояние "свободен".

Если в результате просмотра каналов 2-Й фазы все каналы оказываются занятыми, в блоке 7.11 проверяется состояние накопителя Н2 .Если накопитель содержит свободные позиции (ZN(2)<L(2)), выполняются блоки 7.12, 7.14, увеличивающие на 1 количество заявок в накопителе Н2 и переводящие канал $К_{ij}$ в состояние "свободен". Если накопитель Н2 полностью заполнен, выполняется блок 7.18, переводящий канал $К_{ij}$ в состояние "заблокирован".

Детальный алгоритм блока 8 приведен на рис. 6.7. Блок имитирует процесс передачи заявок из накопителя Н1-й фазы в каналы 1-й фазы. Структура алгоритма полностью аналогична блоку 6.

Блок 9 (рис.6.8) воспроизводит поступление заявок из источника U на вход 1-й фазы. Если при выполнении 9.1 удовлетворяется $T_m \leq T_n$, то это означает, что в момент $t_n$ на выходе источника сформирована очередная заявка. Блоки 9.2, 9.6, 9.7 составляют цикл просмотра состояния каналов 1-й фазы. Если в результате просмотра блок 9.3 обнаружит свободный канал $К_{1j}$, выполняются блоки 9.4, 9.5. Они формируют время завершения обслуживания заявки $T_{ij}$ каналом $К_{ij}$ и переводят канал $К_{ij}$ в состояние "занят обслуживанием".

Если свободных каналов в 1-й фазе нет, то анализируется состояние накопителя Н (9.7). Если накопитель содержит свободную позицию (ZM(1)<L(1)), блок 9.9 увеличивает на 1 количество заявок в накопителе. Если накопитель заполнен, блок 9.10 увеличивает на 1 количество заявок получивших отказ. Во всех случаях в блоке 9.11 вычисляется момент времени t поступления очередной заявки источника на вход системы.

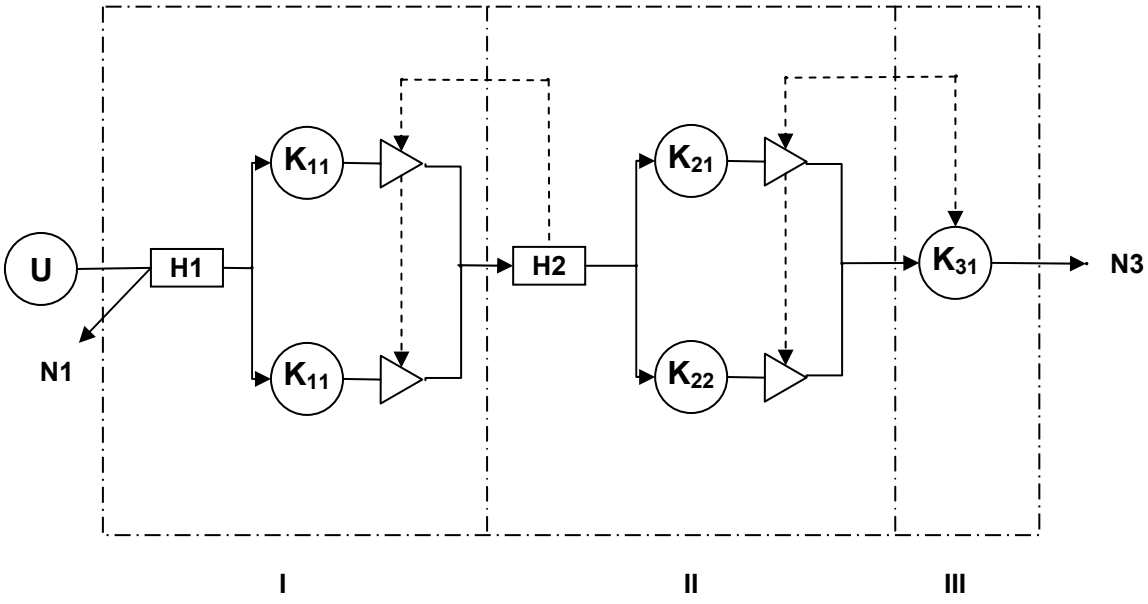



Рис. 6.1. Структура Q-схемы

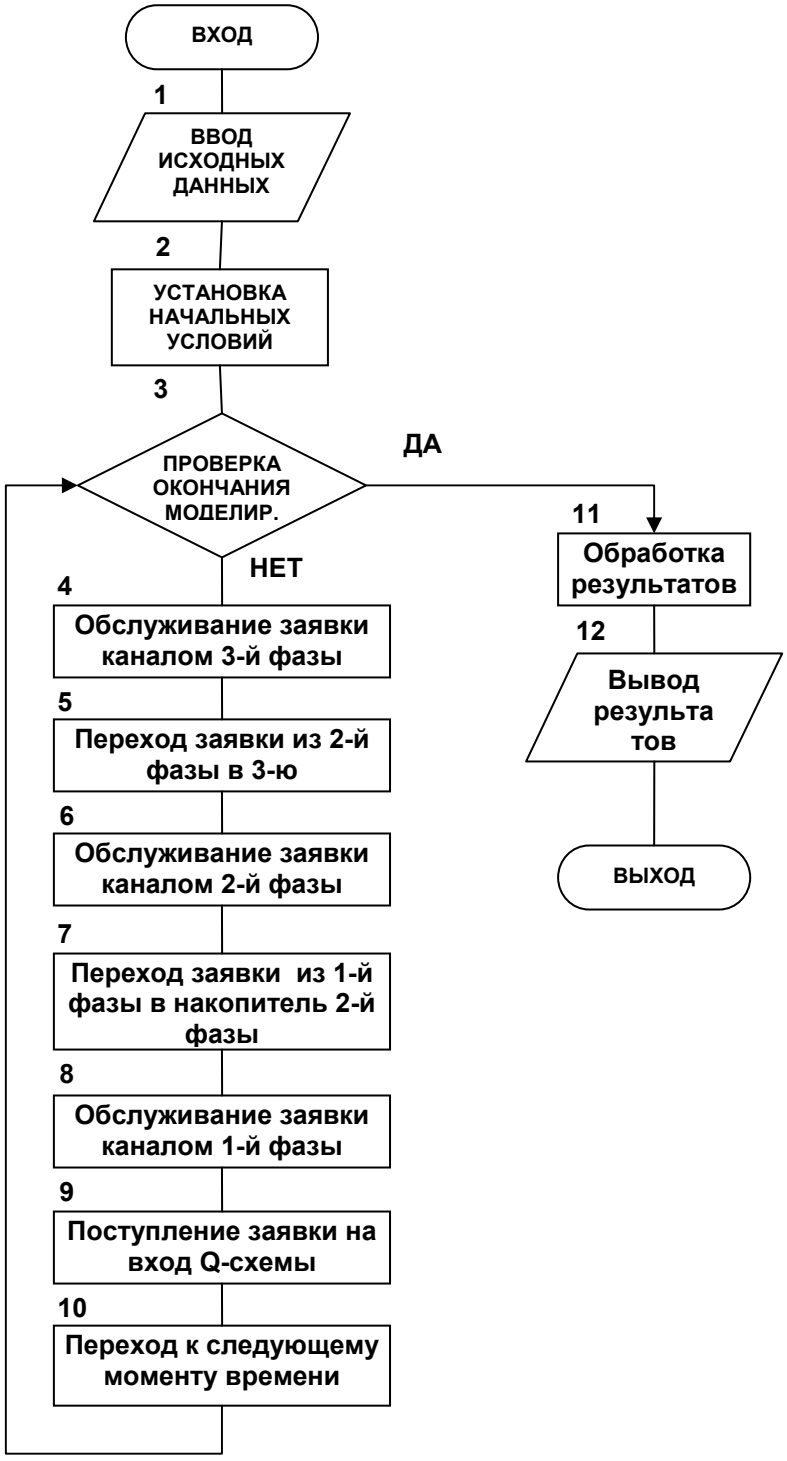



Рис. 6.2. Укрупненный алгоритм модели Q-схемы

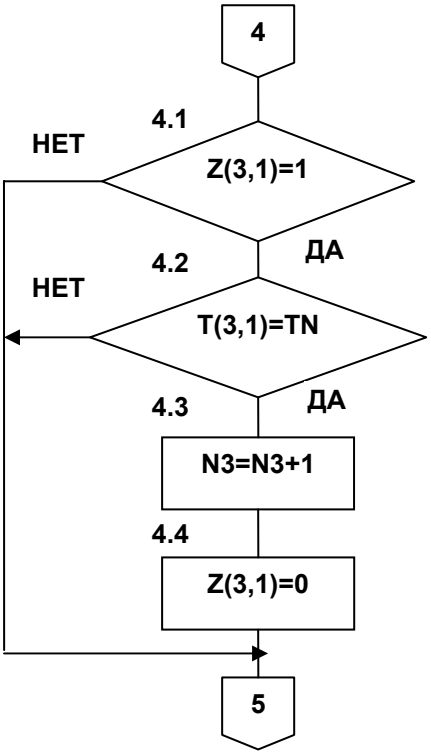



Рис. 6.3 Алгоритм блока 4

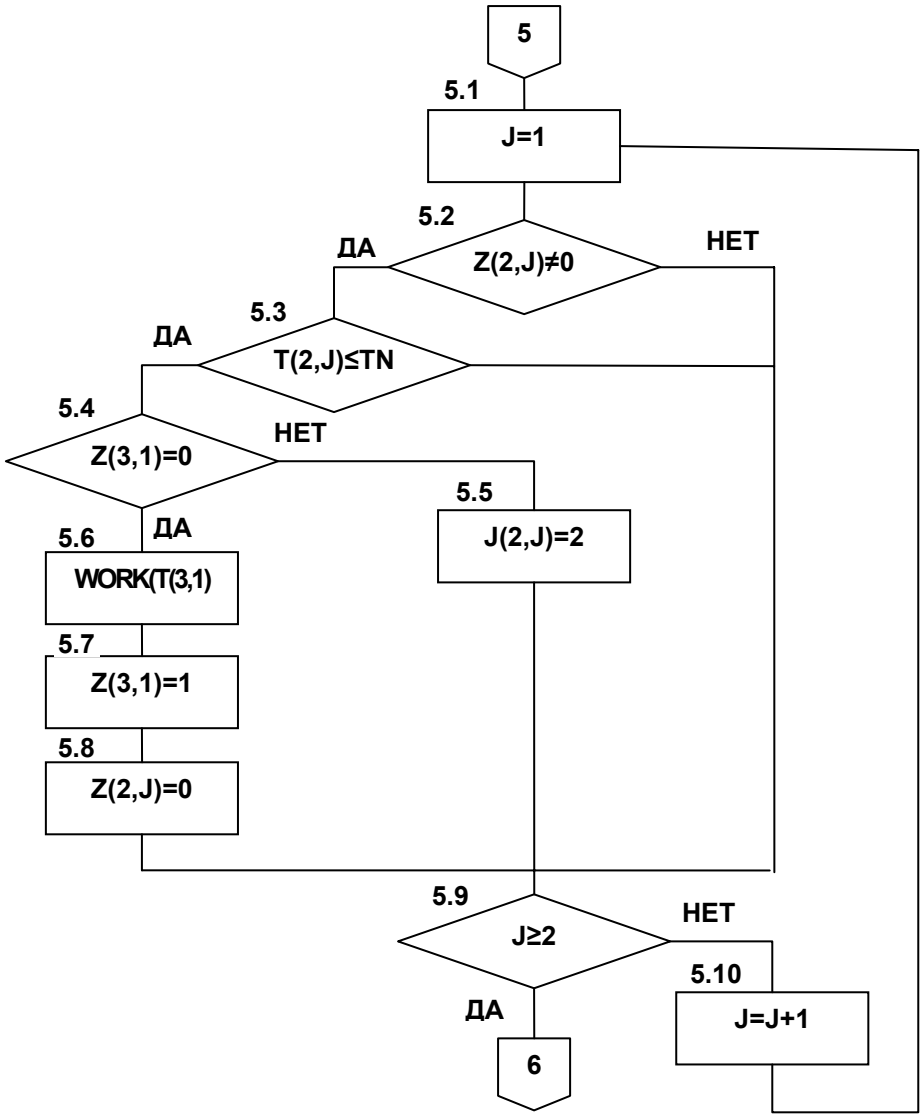



Рис. 6.4. Алгоритм блока 5

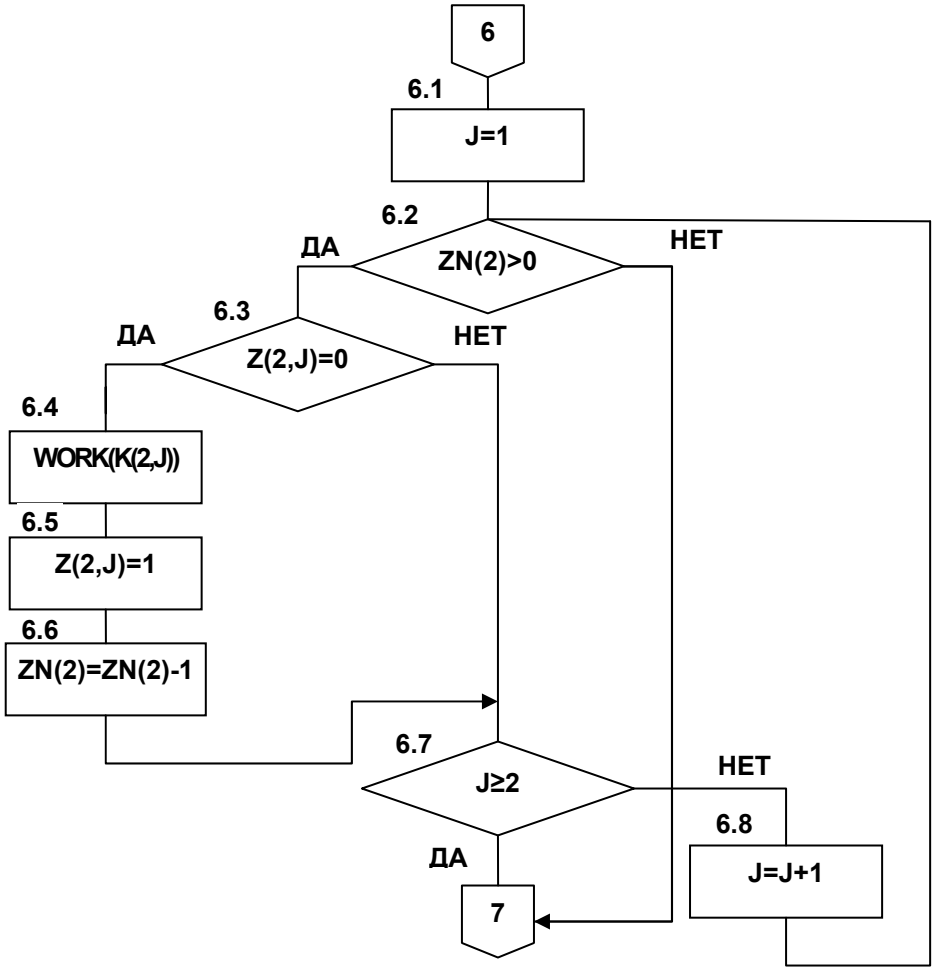



Рис. 6.5. Алгоритм блока 6

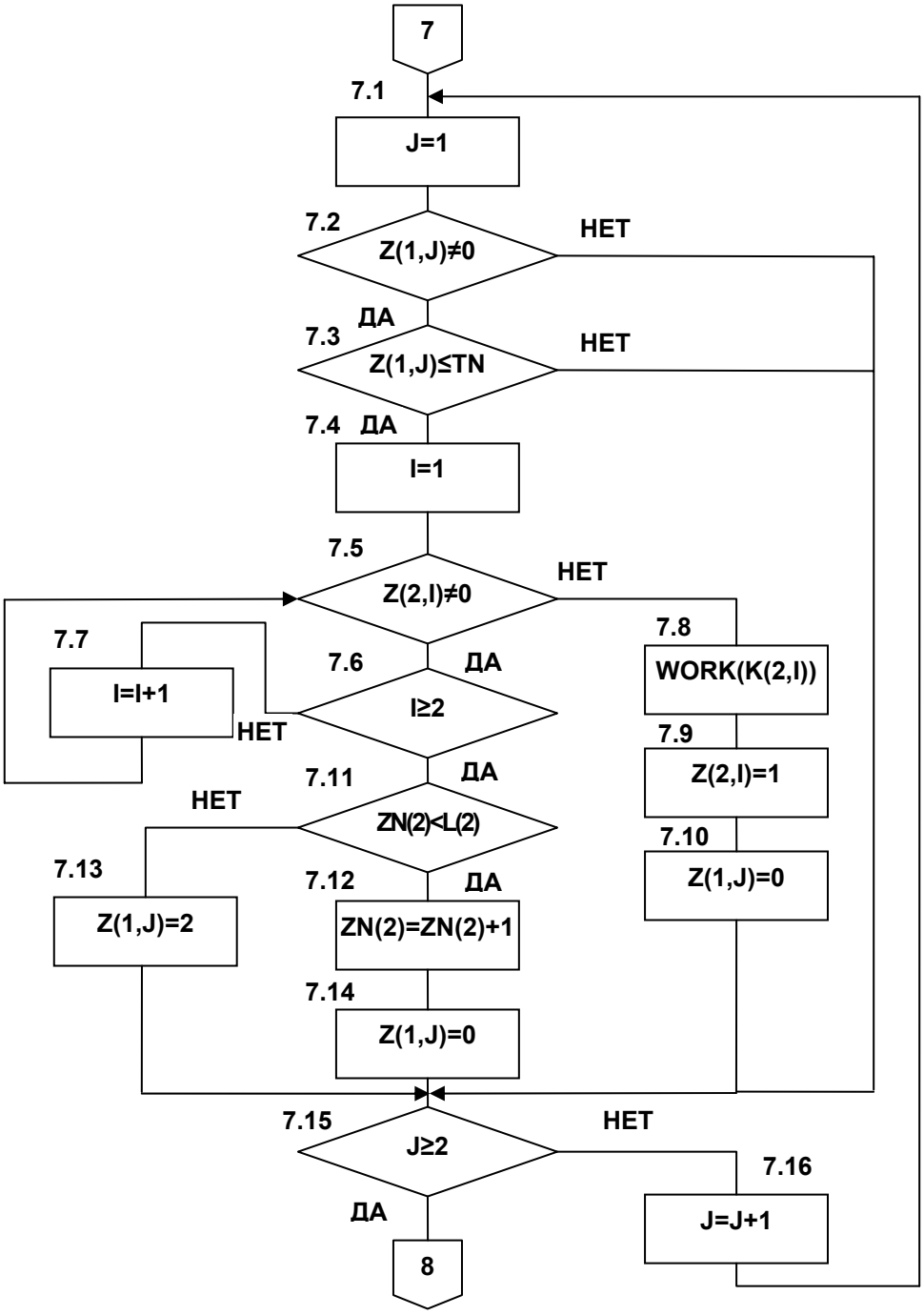



Рис. 6.6. Алгоритм блока 7

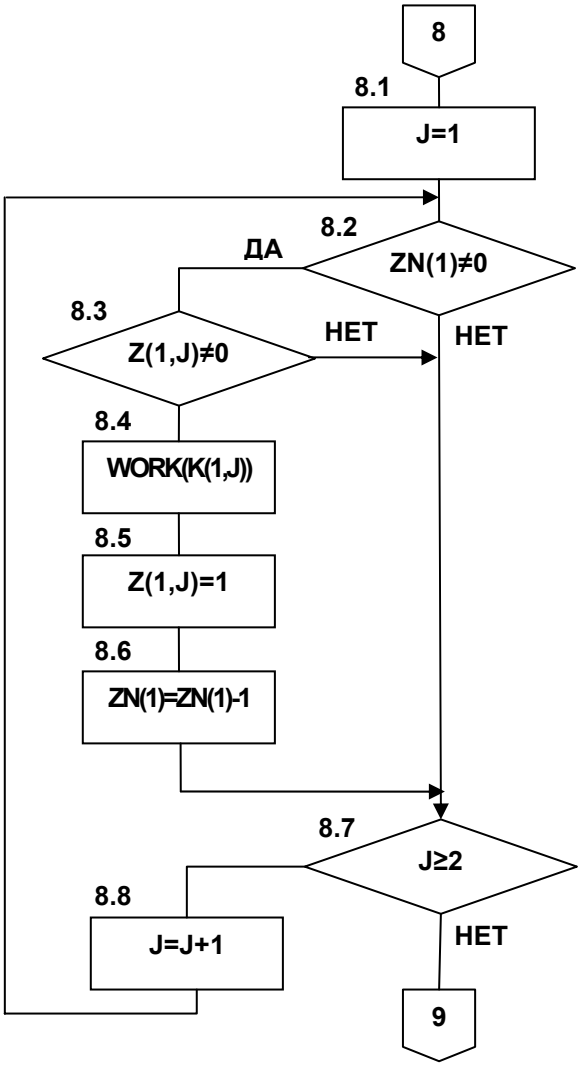



Рис. 6.7. Алгоритм блока 8.

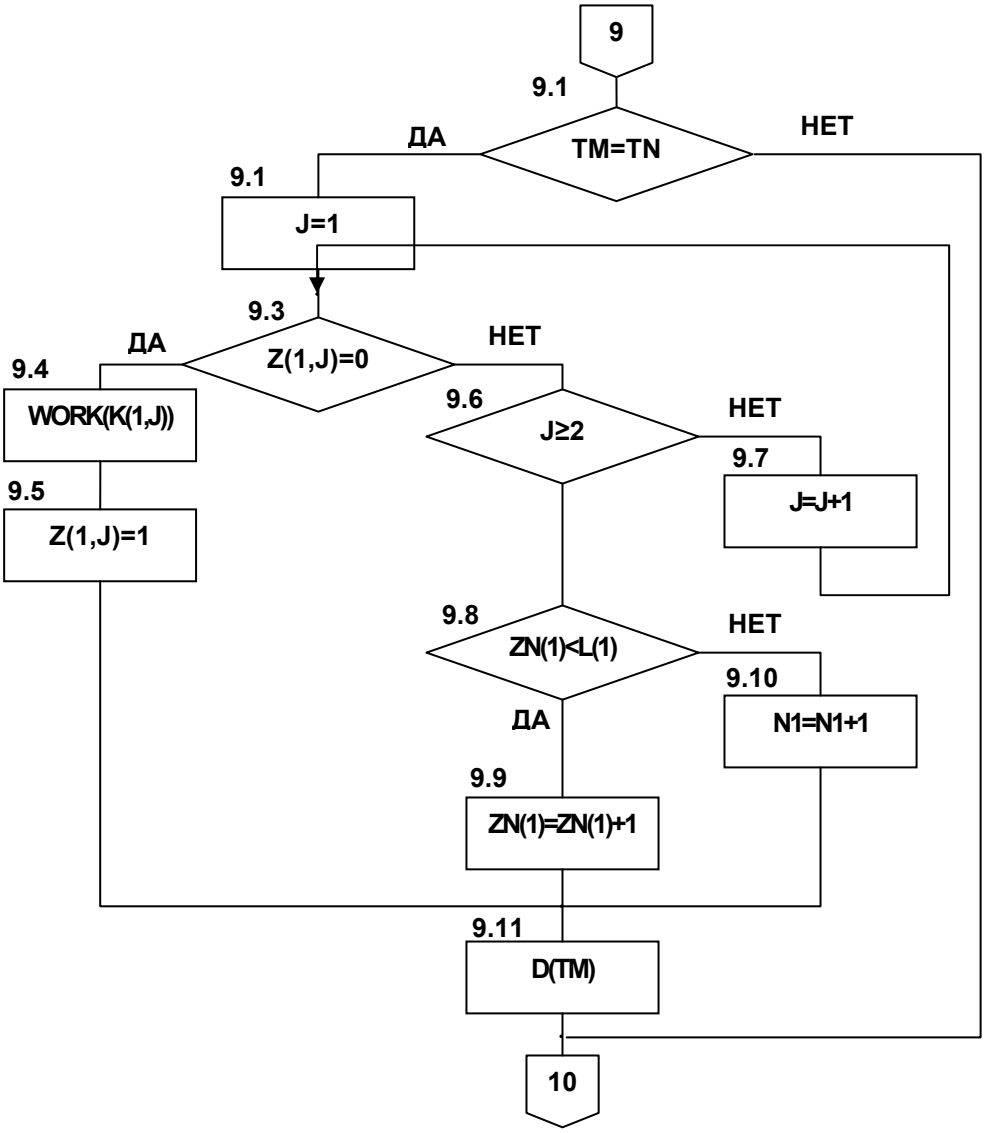



Рис. 6.8. Алгоритм блока 9.