Майер Р.В., Глазовский пединститут

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ:
4. ДИСКРЕТНО-ДЕТЕРМИНИРОВАННЫЕ МОДЕЛИ

В случае, когда исследуемый объект имеет конечный набор дискретных состояний, а переходы из одного состояния в другое происходят по определенным правилам, исключающим элемент случайности, используется **дискретно-детерминированный** или **автоматный подход** (F-схемы) [11]. Он позволяет создать компьютерные модели огромного множества объектов, которым присущи дискретный характер работы: автоматические устройства контроля, регулирования и управления, программируемые исполнители, системы коммутации, нейросети и т.д. Развитием автоматного подхода являются **метод клеточных автоматов** и **мультиагентный подход,** использующиеся для решения большого класса задач [1, 3, 5, 10, 12].

4.1. Сущность автоматного подхода

Автоматом обычно называют дискретное устройство, выполняющее заданную последовательность действий, в результате чего происходит преобразование информации, материальных объектов или энергии. Для изучения поведения автоматических устройств вводят понятие **абстрактного автомата,** -- воображаемого черного ящика, имеющим несколько внутренних состояний, вход и выход [11, 13]. Абстрактный автомат задается множеством внутренних состояний $Q=\{q_1,\ q_2,\ ...,\ q_k\}$, множествами входных и выходных символов (сигналов) $X=\{x_1,\ x_2,\ ...,\ x_n\}$ и $Y=\{y_1,\ y_2,\ ...,\ y_m\}$, а также функцией перехода Ψ и функцией выходов Θ. В некоторых случаях указывают начальное состояние z_0.

Функция перехода Ψ связывает внутреннее состояние устройства q^{t+1} в момент времени $t+1$ с внутренним состоянием q^t и входным символом x^t (сигналом) в предыдущий момент времени t: $q^{t+1}=\Psi(q^t,x^t)$. **Функция выходов** Θ связывает внутреннее состояние q^t устройства и входной сигнал x^t в момент t с выходным сигналом y^t в тот же момент времени: $y^t=\Theta(q^t,x^t)$. Компоненты $F=<X,\ Y,\ Q,\ \Psi,\ \Theta>$ однозначно

определяют **конечный автомат**, который можно рассматривать как математическую F-схему (от англ. finite automata).

Множество внутренних состояний Q образует **внутреннюю память** автомата. Если автомат имеет только одно внутреннее состояние, то он называется **автоматом без памяти**. Они задаются тройкой компонентов $< X,\ Y,\ \Theta >$. Такие автоматы не меняют своего поведения: выходной сигнал (символ) определяется только входным и не зависит от ранее поступивших сигналов (символов) в предыдущие моменты времени: $y_j^t = \Theta(x_i^t)$. К автоматам без памяти относятся: 1) цепь из источника, переключателей и лампочек; 2) комбинационная схема, состоящая из логических элементов И, ИЛИ, НЕ и не содержащая триггеров; 3) кодер, осуществляющий побуквенный перевод входного сообщения, и т.д.

Рассмотрим комбинационную схему с тремя входами и двумя выходами (рис. 1.1). Состояния ее выходов, описываются логическими функциями: y_1 = not(x_1 or x_2), y_2 = (x_1 or x_2) and x_3. Исследовать работу этой цифровой схемы можно с помощью программы ПР – 1 (Приложение). В ней перебираются всевозможные состояния входов x_1, x_2, x_3 и определяются состояния выходов y_1 и y_2. На экране монитора появляется **таблица истинности**.

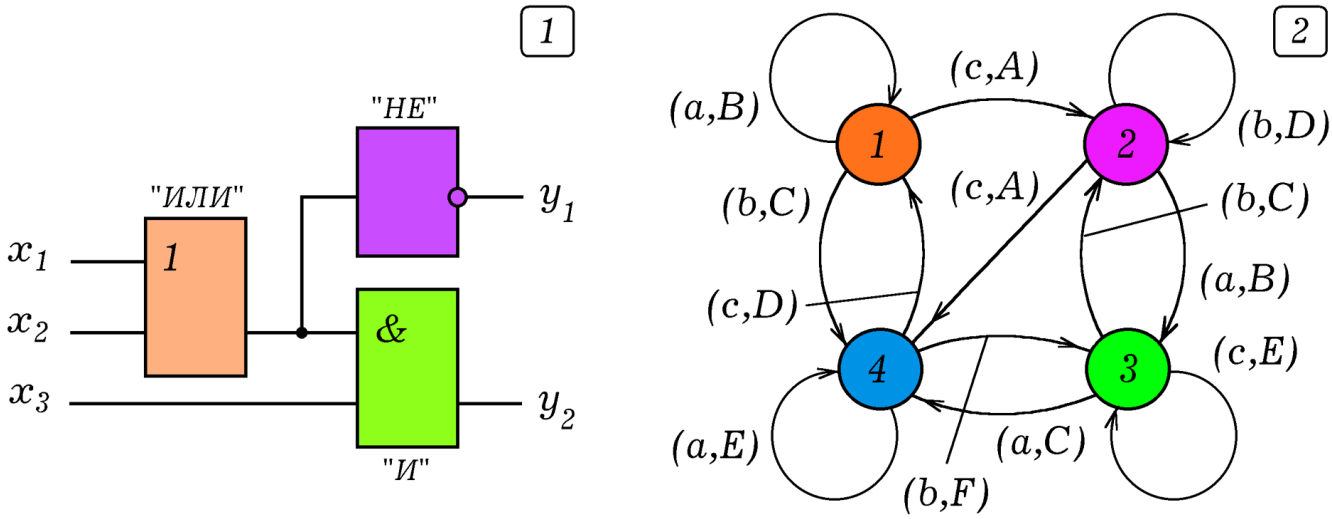

Рис. 1. Комбинационная схема. Диаграмма Мура автомата с памятью.

Если автомат имеет два или более внутренних состояния, то он называется **автоматом с памятью**. Рассмотрим автомат с четырьмя внутренними состояниями $Q = \{1,\ 2,\ 3,\ 4\}$, входным алфавитом $X = \{a,\ b,\ c\}$ и выходным алфавитом $Y = \{A,\ B,\ C,\ D,\ E,\ F\}$. Для задания правил переключения автомата будем использовать **диаграмму Мура**, представляющую собой ориентированный граф (рис. 1.2). Вершины графа изобра-

жают состояния автомата $Q = \{1,\ 2,\ 3,\ 4\}$, а ребра -- переходы из одного состояния в другое. Команда "2c $\rightarrow$ 4A" означает, что если автомат находится в состоянии 2 и на его вход приходит символ "c", то он переходит в состояние 4 и на его выходе появляется символ "A".

Промоделировать работу этого автомата на компьютере можно с помощью программы ПР – 2. Пусть на вход поступает последовательность символов "babacaabcbabcbcbaaccbcbabcbccaacbb". Программа последовательно перебирает входные символы x^t, на каждом временном шаге $t+1$ определяя новое состояние автомата Q^{t+1} и символ на его выходе y^{t+1}. Результаты выводятся на экран в виде таблицы.

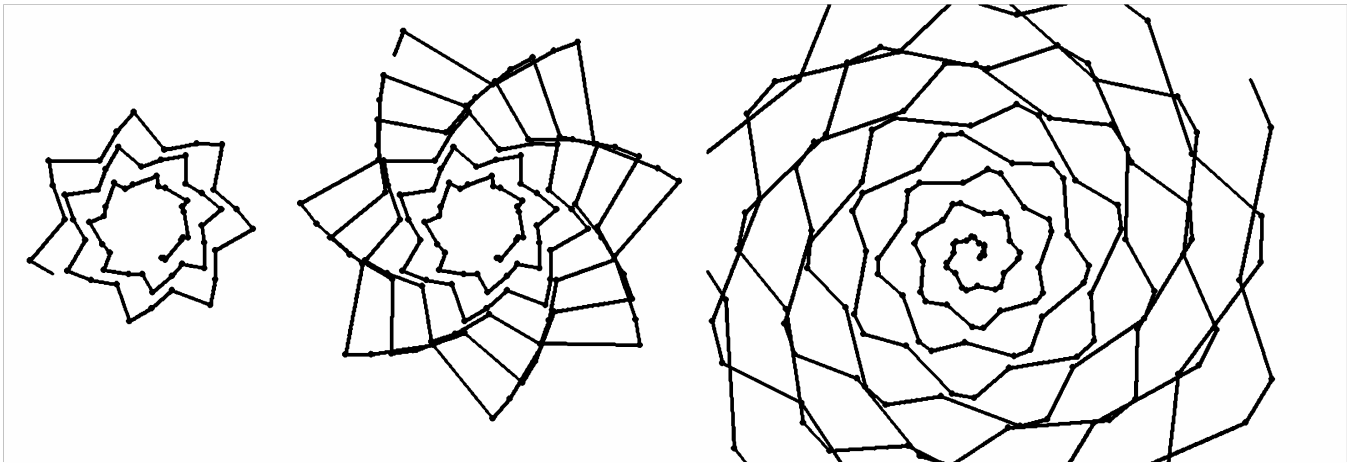

Рис. 2. Фигуры, которые рисует исполнитель "черепашка".

Поведение автомата может никак не зависеть от состояния входа, а определяться лишь номером шага (дискретным временем t) и заложенной в него программой. Такой автомат называется **исполнителем**. В качестве примера рассмотрим компьютерную модель движения черепашки [8], алгоритм функционирования которой содержит три вида команд: Повторить (количество повторений), Поворот (угол в градусах), Вперед (расстояние в условных единицах длины). Используется компьютерная программа ПР-3. Исполнение алгоритма

```
Повторить_40_раз{
    Поворот(-20);  Вперед(4*t);
    Поворот(60);   Вперед(5*t);
    Поворот(-100); Вперед(10);   }
```

приводит к тому, что "черепашка" рисует на плоскости сложную кривую. Примеры получающихся траекторий изображены на рис. 2.

4.2. Моделирование машин Поста и Тьюринга

При изучении теории алгоритмов, определении множества алгоритмически разрешимых задач и построении универсальных алгоритмических моделей обычно рассматривают абстрактные машины Поста и Тьюринга. Если доказать, что та или иная задача может быть решена с помощью такой гипотетической машины, то это означает, что она относится к классу алгоритмически разрешимых задач и решаема на вычислительной машине. Выполнение машинами Поста или Тьюринга соответствующего алгоритма возможно промоделировать на ЭВМ [6, 8].

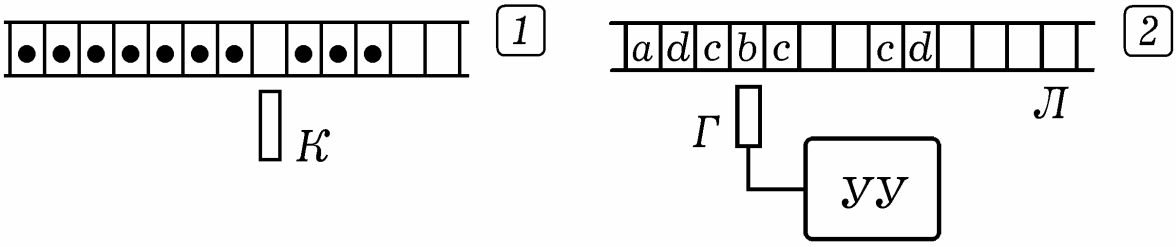

Рис. 3. Гипотетические машины Поста и Тьюринга.

Машина Поста состоит (МП) из ленты, разбитой на ячейки, и каретки, которая может считывать содержимое обозреваемой ячейки, перемещаться на одну ячейку влево или вправо, стирать и ставить метки (рис. 3.1). Для моделирования ее работы используется программа ПР – 4. Рассмотрим вычитание целых чисел с помощью МП. Применяемый алгоритм приведен ниже:

```
7                                  -- координата каретки
VVVVVV-VV-------------------------   лента
1 сместить влево, команда 2
2 если пусто -- команда 1, если метка -- команда 3
3 удалить метку, команда 4
4 сместить вправо, команда 5
5 если пусто -- команда 4, если метка -- команда 6
6 удалить метку, команда 7
7 сместить вправо, команда 8
8 если пусто -- команда 9, если метка -- команда 1
9 остановить МП.
```

В первых двух строчках указывается положение каретки и состояние ленты, на которой в унарной системе счисления записаны два числа (в данном случае 6 и 2). В программе ПР – 4 команды МП кодируются в формате: **kom[4]:='right'; k[4]:=5**;, что означает: "команда 4: сме-

стить каретку вправо, перейти к команде 5". Результат работы программы ПР - 4, моделирующей МП, выводится на экран в следующем виде:

```
V V V V V V - V V - - - - - - - - - - - - - - - - - - | 1 |
-----------M
V V V V V V - V V - - - - - - - - - - - - - - - - - - | 2 |
----------M
V V V V V - - V V - - - - - - - - - - - - - - - - - - | 3 |
----------M
V V V V V - - V V - - - - - - - - - - - - - - - - - - | 4 |
-----------M
V V V V V - - V V - - - - - - - - - - - - - - - - - - | 5 |
-----------M
```

Машина Тьюринга (МТ) состоит из бесконечной ленты, разбитой на ячейки, головки и управляющего устройства (рис. 3.2). Головка перемещается относительно ленты, стирает старые и ставит новые символы. Программа для МТ записывается в виде последовательности команд, представленных в общепринятом формате: "(Состояние q_1) (В ячейке символ S_1) $\rightarrow$ (Новое состояние q_2) (Напечатать символ S_2) (Сместить головку вправо (R), влево (L) или остановиться (S))".

Рассмотрим задачу: На ленте машины Тьюринга -- последовательность символов _ ABBAABAB _ _ _. Головка расположена напротив левого символа. Необходимо написать программу МТ, заменяющую символы "A" звездочками и группирующую их в правой части строки.

Программа для МТ может быть записана в виде таблицы:

q	_	A	B	*	|
1	1_R	2*R	1BR	1*R	1|S
2	3|R	2AR	2BR	2*R	3|R
3	4AL	3AR			
4	1_R	4AL	4BL	4*L	4|L

Из нее следует, что если машина находится в состоянии 2 и в обозреваемой ячейке видит символ "_", то она должна перейти в состояние 3, напечатать символ "|" и сместить головку на одну ячейку вправо. Для моделирования работы МТ используется программа ПР -- 5. Она позволяет решить разные задачи, но для этого требуется правильно закодировать программу МТ. Рассмотренная выше команда кодируется в виде **kom[2]:= '2_>3|R';**. Результат исполнения программы выводится на экран в виде:

```
_ A B B A A B A B _ _ _ _ _ _  q=1   k=1
==|
_ * B B A A B A B _ _ _ _ _ _  q=2   k=2
====|
_ * B B A A B A B _ _ _ _ _ _  q=2   k=3
======|
_ * B B A A B A B _ _ _ _ _ _  q=2   k=4
========|
```

Можно представить себе машину Тьюринга, в которой вместо ленты используется плоскость, разбитая на квадратные ячейки. Перемещающаяся головка считывает записанные в них символы, стирает их и печатает новые в соответствии с заложенной в нее программой. Такие воображаемые устройства, объединяющие в себе машину Тьюринга и муравья (термита), который ползает по плоскости, называются **тьюрмитами.**

Одним из наиболее известных тьюрмитов является **муравей Лэнгтона**, поведение которого описывается следующим алгоритмом [14]. Представим себе плоскость, разбитую на клетки, раскрашенные определенным образом в черный и белый цвет. В некоторой клетке находится муравей, который на каждом временном шаге должен перейти в одну из четырех смежных клеток. При этом реализуются следующие правила: 1) если муравей в черном квадрате, то он должен повернуть на 90^0 вправо, сделать квадрат белым и перейти в следующую клетку; 2) если муравей в белом квадрате, то он должен повернуть на 90^0 влево, сделать квадрат черным и перейти в следующую клетку. Этот простой алгоритм приводит к сложному движению муравья: после не очень продолжительного случайного блуждания, он начинает совершать упорядоченное периодическое движение, удаляясь от своего исходного положения.

4.3. Компьютерное моделирование нейросетей

Развитие информационных технологий, робототехники, искусственного интеллекта связано с проблемой построения и обучения нейросетей. Представим себе устройство с электронной "нервной системой", состоящей из отдельных блоков, моделирующих нейроны. Каждый такой искусственный нейрон должен состоять из суммирующего и порогового элементов. Нейрон возбуждается, когда сумма весов синапсов, на которые поступают импульсы раздражения, превышает его порог срабатывания.

Для исследования различных нейросетей используют понятие **формального нейрона**, под которым понимают гипотетический автомат с n входами x_1, x_2, ..., x_n и одним выходом y, характеризующийся **порогом срабатывания** h и **весами** ω_1, ω_2, ..., ω_n. Его выход возбужден $(y=1)$, когда сумма всех весов возбужденных входов превышает порог h:

$$\omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + ... + \omega_n x_n > h.$$

В противном случае выход не возбужден $(y=0)$. Это можно записать так:

$$y = \begin{cases} 1, & \textit{если} \quad \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + ... + \omega_n x_n > h \\ 0, & \textit{в противном случае.} \end{cases}$$

Если вес i-го входа положительный $(\omega_i > 0)$, то **вход возбуждающий**, если отрицательный $(\omega_i < 0)$, -- **вход тормозящий**. Выход искусственного нейрона может находиться в двух состояниях, поэтому он может разделять объекты только на два класса. Чтобы сымитировать работу нейрона на ЭВМ, достаточно найти взвешенную сумму его входов и использовать оператор условного перехода [6, 8].

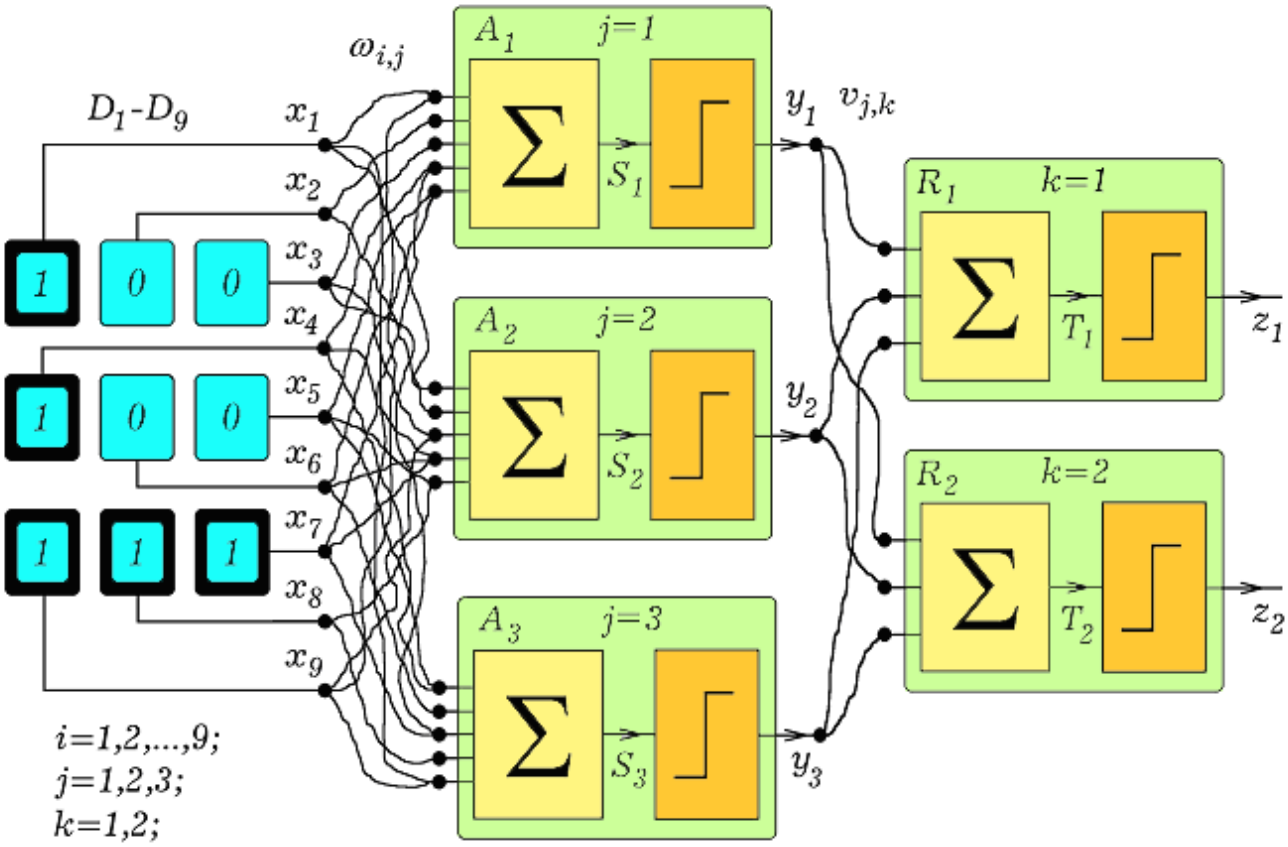

Рис. 3. Схема трехслойного персептрона.

Нейросеть представляет собой множество нейронов, соединенных так, что выход одного нейрона после разветвления присоединяется к входам других нейронов, причем каждый вход соединен не более чем с од-

ним выходом. Под **персептроном** понимают обучаемую нейросеть, состоящую из **датчиков, ассоциативных** и **реагирующих элементов** с заданной матрицей весовых коэффициентов. Рассмотрим трехслойный персептрон (рис. 3), имеющий слой сенсоров или датчиков D_i ($i = 1,2,...,9$), слой ассоциативных элементов A_j ($j=1,2,3$) и слой реагирующих элементов R_k ($k=1,2$). Если уровень воздействия на датчик превышает некоторое пороговое значение, то на его выходе появляется 1, а иначе -- 0. Ассоциативный элемент функционирует как формальный нейрон; веса его входов ω_{ij} принимают значения -1, 0 или 1. Реагирующие R--элементы работают так: когда сумма всех весов возбужденных входов положительна, на выходе 1, иначе, -- на выходе 0. Веса входов υ_{jk} реагирующего элемента могут принимать произвольные значения. Выход реагирующего элемента имеет два состояния, поэтому персептрон с двумя решающими элементами может классифицировать объекты на $2^2 = 4$ класса, соответствующие выходным сигналам 00, 01, 10, 11.

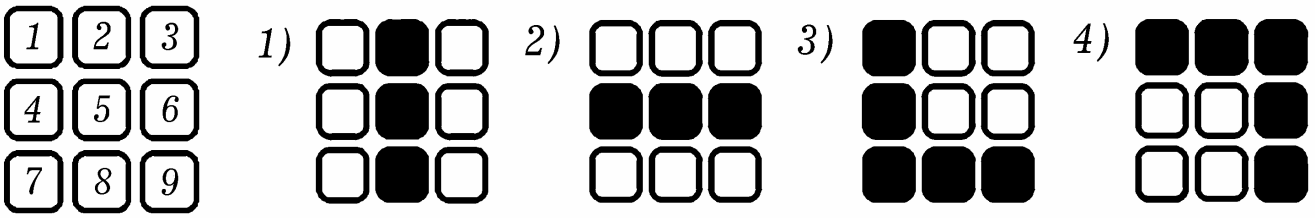

Рис. 4. Объекты, предъявляемые персептрону.

Создать персептрон не просто, но его можно достаточно легко промоделировать на компьютере. Рассмотрим программу ПР - 6, моделирующую работу трехслойного персептрона (рис. 3) и различающего четыре объекта, представленные на рис. 4. Учитывая расположение датчиков D_i ($i=1,\ 2,\ ...,\ 9$), предъявляемые персептрону объекты кодируются так: O_1 = (0, 1, 0, 0, 1, 0, 0, 1, 0), O_2 =(0, 0, 0, 1, 1, 1, 0, 0, 0), O_3 =(1, 0, 0, 1, 0, 0, 1, 1, 1), O_4 = (1, 1, 1, 0, 0, 1, 0, 0, 1). Персептрон будет правильно классифицировать эти объекты, если задать веса связей, соединяющих датчики с ассоциативными элементами, и ассоциативные элементы с реагирующими элементами, следующим образом:

$$\omega_{ij} = \begin{pmatrix} 0 & -1 & 1 & 1 & 0 & 1 & -1 & -1 & 0 \\ 0 & 1 & 1 & -1 & 0 & 1 & -1 & -1 & 0 \\ 1 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & 1 \end{pmatrix}, \quad \upsilon_{jk} = \begin{pmatrix} 2 & 2 & -2 \\ -2 & 2 & 2 \end{pmatrix}.$$

Порог срабатывания ассоциативных элементов h равен $0{,}5$. Подавая на входы сигналы, соответствующие перечисленным выше объектам, можно убедиться в правильной работе персептрона.

4.4. Метод клеточных автоматов

Одним из эффективных методов моделирования различных сред, полей, живых организмов является **метод клеточных автоматов** (КА). Он был предложен Джоном фон Нейманом и Конрадом Цусе в конце сороковых годов 20 века как способ создания универсальной вычислительной среды для изучения различных алгоритмов [1, 4, 12, 15]. Метод КА состоит в том, что рассматривается одно-, двух или трехмерная сетка, в узлах которой находятся автоматы. Каждый автомат взаимодействует (обменивается сигналами, информацией) только с соседними автоматами. Изменение состояния автомата в данном узле в следующий момент времени $t+1$ зависит от его предыдущего состояния и состояния его соседей в момент t, а также **правил переключения**, задаваемых программистом. Поведение системы автоматов зависит от этих правил и ее начального состояния.

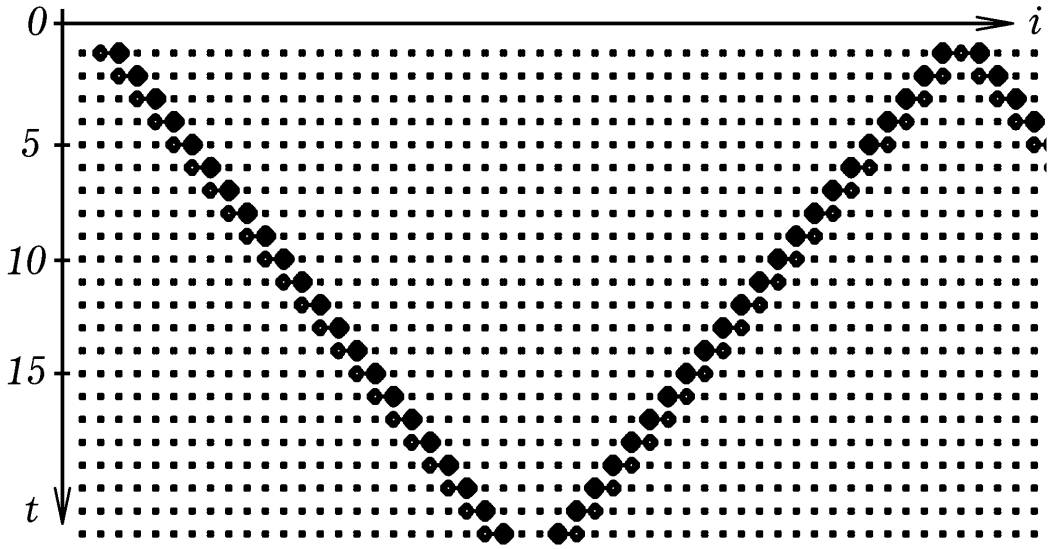

Рис. 5. Распространение импульса возбуждения в одномерной среде.

Рассмотрим использование метода КА для моделирования распространения импульса возбуждения по **одномерной активной среде** (например, нервному волокну). Мысленно заменим среду цепочкой одинаковых автоматов, каждый из которых может находиться в трех состояниях: 1) **покоя**, $s=0$; 2) **рефрактерности**, $s=1$; 3) **возбуждения**, $s=2$. Если эле-

мент находится в состоянии возбуждения, то он переводит соседние элементы, находящиеся в покое, в возбужденное состояние. Из возбужденного состояния элемент через один такт переходит в состоянии рефрактерности, в котором он не может быть возбужден, а затем через один такт -- в состояние покоя. Поведение элемента среды описывается правилом:

$$s_i^{t+1} = \begin{cases} 0, & \text{если} \quad s_i^t = 1, \\ 1, & \text{если} \quad s_i^t = 2, \\ 2, & \text{если} \quad s_i^t = 0 \quad \text{и} \quad (s_{i-1}^t = 2 \quad \text{или} \quad s_{i+1}^t = 2). \end{cases}$$

Для моделирования одномерной активной среды используется программа ПР-7. Она содержит цикл по времени t, в который вложен цикл по i. В нем перебираются все элементы среды и, в соответствии с указанным выше правилом, определяется состояние элемента на следующем временном слое $t+1$. Результаты ее работы представлены на рис. 5.

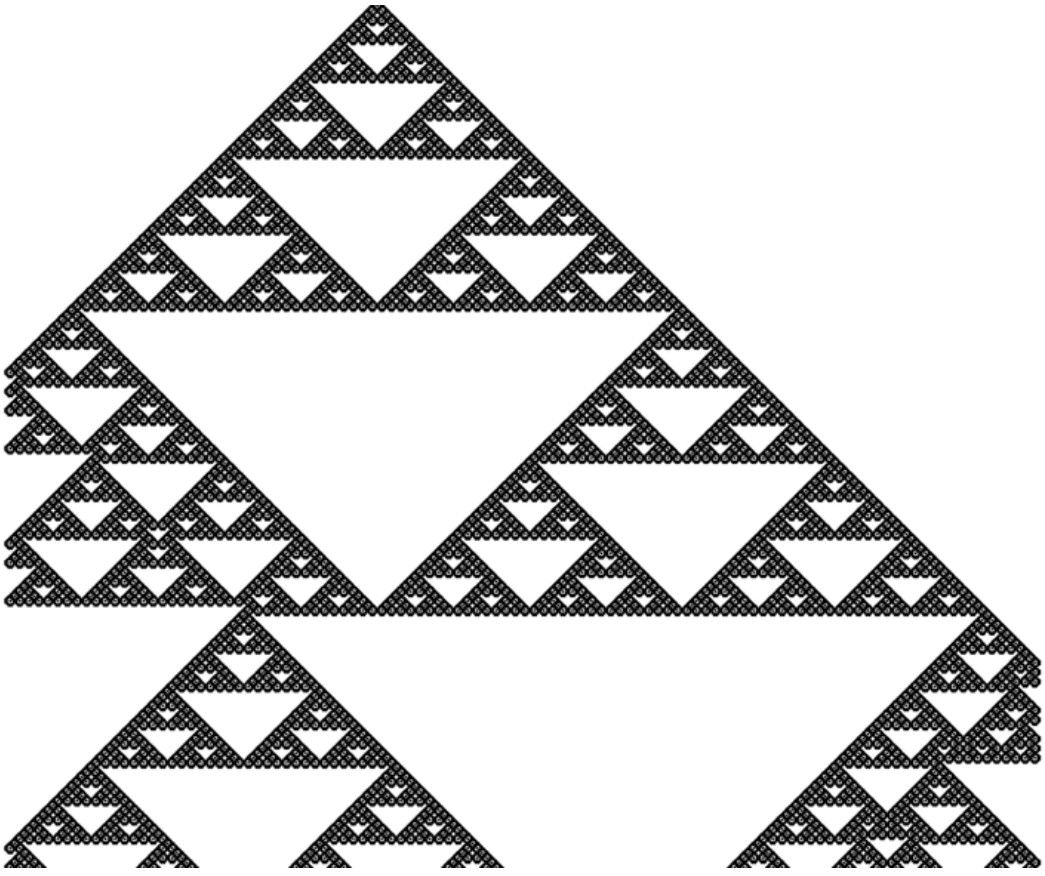

Рис. 6. Фрактал, нарисованный клеточным автоматом.

Немного изменив эту программу, можно получить фрактал, известный как **ковер Серпинского** (рис. 6). Для этого состояние каждой ячейки в момент $t+1$ должно определяться по формуле: $s_i^{t+1} = (s_{i-1}^t + s_{i+1}^t) \bmod 2$.

В ней суммируются состояния соседних ячеек на предыдущем временном шаге и определяется остаток от деления на 2 (программа ПР-8).

Классическим примером использования метода клеточных автоматов является **модель размножения бактерий** на плоской поверхности, известная как **игра "Жизнь"** [4, 12]. Она была создана Д.Х. Конуэйем в 1970 году, как имитация реальных процессов при зарождении и гибели колонии живых организмов. Плоская поверхность разбита на квадратные ячейки -- клетки, которые ведут себя как автоматы, способные находиться в двух состояниях: "живой" ($x_{ij}=1$) или "мертвый" ($x_{ij}=0$). При этом моделируются следующие биологические процессы: выживание, гибель, рождение. Конуэй подобрал правила переключения КА так, чтобы: 1) был исключен неограниченный рост популяции; 2) существовали начальные конфигурации живых клеток, при которых колония беспредельно развивалась; 3) существовали конфигурации живых клеток, при которых колония либо полностью исчезала, либо переходила в устойчивую конфигурацию, либо начинала совершать колебания.

Правила, сформулированные Конуэйем, выглядят так. Клетка оживает при наличии 3 живых соседей. Если живых соседей 4 и больше, она умирает от перенаселенности. Если живых соседей меньше 2, она умирает от одиночества. Эти правила можно записать в виде:

$$x_{ij}^{t+1}=\begin{cases}1, & \textit{если} \quad x_{ij}^{t}=0 \quad \textit{и} \quad s=3,\\ 1, & \textit{если} \quad x_{ij}^{t}=1 \quad \textit{и} \quad s=2 \quad \textit{или} \quad s=3,\\ 0, & \textit{если} \quad s<2 \quad \textit{или} \quad s>3.\end{cases}$$

Здесь s -- число "живых" соседей, которое вычисляется по формуле:

$$s=x_{i-1,j-1}+x_{i,j-1}+x_{i+1,j-1}+x_{i-1,j}+x_{i+1,j}+x_{i-1,j+1}+x_{i,j+1}+x_{i+1,j+1}.$$

Используется программа ПР--9. Для вычисления числа "живых" соседей и установления "жива" данная клетка или нет в момент $t+1$, используется процедура Raschet, а результат сохраняется в массиве x[i,j]. В массив x1[i,j] записываются состояния клеток на предыдущем временном шаге t. В начале программы следует задать исходное распределение "живых" клеток. Существуют различные начальные конфигурации живых клеток, обладающие замечательными свойствами. Так, на рис. 7 представлены две конфигурации, способные перемещаться, периодически повторяя свою форму: планер (рис. 7.1) и корабль (рис. 7.2). На рис. 7.3 изображена конфигурация из 5 клеток, демонстрирующая эффект резонанс-

ного возбуждения: при запуске программы популяция живых клеток начинает расти.

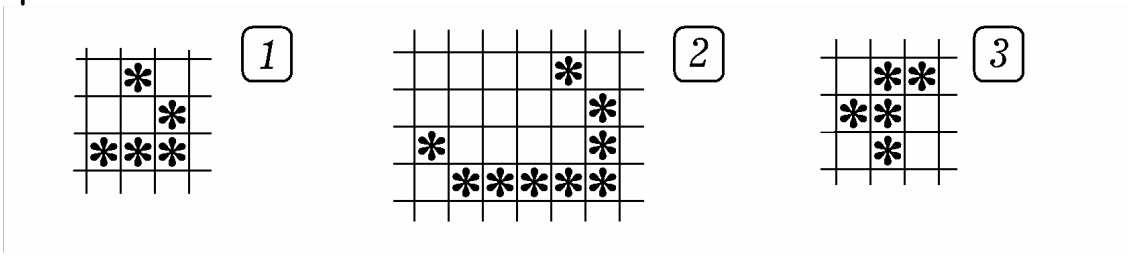

Рис. 7. Некоторые начальные конфигурации живых клеток.

Конвей предложил промоделировать зарождение жизни на Земле, задав такие правила переключения клеточных автоматов, при которых из хаотического расположения живых клеток получаются самовоспроизводящиеся структуры. Проблема создания **самовоспроизводящихся машин** исследовалась фон Нейманом [14]. Был предложен КА, который способен воспроизводить точную копию начального распределения возбужденных ячеек, моделирующих биологическую "клетку". Из исходной и получившейся клеток получаются еще 2 новые "клетки", затем еще 4 новые "клетки" и т.д. Доказано, что с помощью клеточного автомата "Жизнь" в принципе можно построить все дискретные элементы компьютера.

Развитием игры "Жизнь" является менее известный **клеточный автомат Инь-Ян** (женщина - мужчина), отражающий жизнь и размножение двуполых организмов [5]. Представим плоскость, разбитую на квадратные ячейки, которые могут находиться в одном из трех состояниях: 1) пустая (то есть мертвая) ячейка; 2) живая ячейка Инь; 3) живая ячейка Ян. Состояние каждой ячейки на следующем временном шаге $t+1$ определяется состоянием соседних восьми ячеек в момент t. Правила переключения автомата Инь-Ян заданы таким образом, чтобы популяции Инь и Ян противоборствовали, но не могли развиваться друг без друга. Они следующие: 1) рождение происходит, если у пустой ячейки ровно три живых неодинаковых соседа: в случае когда среди соседей только один Ян, рождается Ян; когда только один Инь, рождается Инь; 2) гибель от перенаселения или одиночества: если живая ячейка, имеет больше четырех или меньше двух соседей, она умирает от перенаселения или одиночества. 3) гибель в неравном противостоянии: если у живой ячейки ровно четыре соседа противоположного пола, то она умирает.

Более сложным применением клеточных автоматов в биологии является **модель Аква-Тор**, позволяющая исследовать поведение системы из двух популяций: травоядных -- рыб (зайцев) и хищников – акул (волков) [1]. Предполагается, что пищи для травоядных всегда хватает, а пи-

щей хищников являются травоядные. В названии присутствует слово тор, потому что в модели используется тороидальная поверхность -- замкнутый сам на себя прямоугольник, противоположные стороны которого склеены. В случае, когда какая-либо особь достигает верхнего края и покидает его, то точно такая же особь появляется на нижней стороне прямоугольной области. Аналогично, особь, пересекшая левую сторону прямоугольника, появляется на правой стороне и наоборот.

При этом реализуются правила: 1) особь случайно перемещается в соседнюю клетку; 2) через равные промежутки времени особь оставляет потомство в той клетке из которой она переместилась; 3) хищник перемещаясь в клетку с жертвой может с заданной вероятностью поглотить жертву; 4) особь живет ограниченное время, а затем умирает; 5) если хищник не находит пищи в течение заданного времени голодной смерти, он погибает. Компьютерная модель "Аква-Тор" позволяет изучить некоторые закономерности существования экосистемы Мирового океана, в частности исследовать влияние течения на распределение рыб, выявить условия стабильного существования популяций хищников и травоядных и т.д.

4.5. Понятие о мультиагентном подходе

Упомянутая выше компьютерная модель "Аква-Тор" может быть создана на основе **мультиагентного подхода** [1, 5]. Он состоит в том, что все моделируемые динамические объекты (люди, рыбы, автомашины и т.д.) являются независимыми агентами, функционирующими в виртуальном мире, состояние которого является результатом их взаимодействия. При этом они действуют автономно, у каждого из них нет информации о все системе, и отсутствуют агенты, полностью управляющие всей системой. В компьютерных моделях используются **программные агенты-автоматы**, функционирующие в соответствии с заданным алгоритмом. Так, например, акула 1, поглотив рыбу, на некоторое время переходит в другое состояние, соответствующее насыщению. Через некоторое время акула рождает новую акулу 2, которая ведет себя в соответствии с заложенной программой. Если акула 1 не находит пищи или становится старой, то она умирает. В общем случае агенты могут обмениваться информацией друг с другом, получать информацию об окружающей среде, а также имеют набор механизмов для изменения ее состояния. Каждый агент содержит мо-

дуль принятия решения, который на основе анализа входных данных выдает сигнал на ту или иную реакцию агента.

В качестве примера рассмотрим следующую задачу. В центре озера O находится лодка с исследователем, который хочет нарисовать карту береговой линии (рис. 8). В его распоряжении имеются 10 роботов-агентов, каждый из которых независимо от других доплывает до берега, устанавливает на нем синий флажок (т. A), а затем возвращается обратно к плоту, подплывая к нему на расстояние l (т. B), и сообщает координаты установленного флажка. После этого все повторяется снова. Над озером висит туман. Роботы, удаляясь от лодки, хаотически изменяют направление своего движения. Требуется промоделировать движение роботов-агентов.

При создании компьютерной модели каждый робот заменяется программным агентом, который может находиться в двух состояниях: 1) состояние 1 или режим поиска: агент удаляется от лодки с исследователем, случайным образом изменяя направление движения, пока не достигнет берега, на котором он установит флажок; 2) состояние 2: агент возвращается к лодке, чтобы, приблизившись на расстояние l, передать координаты установленного флажка. Поведение роботов-агентов моделируется программой ПР--10, результаты ее работы приведены на рис. 8.2.

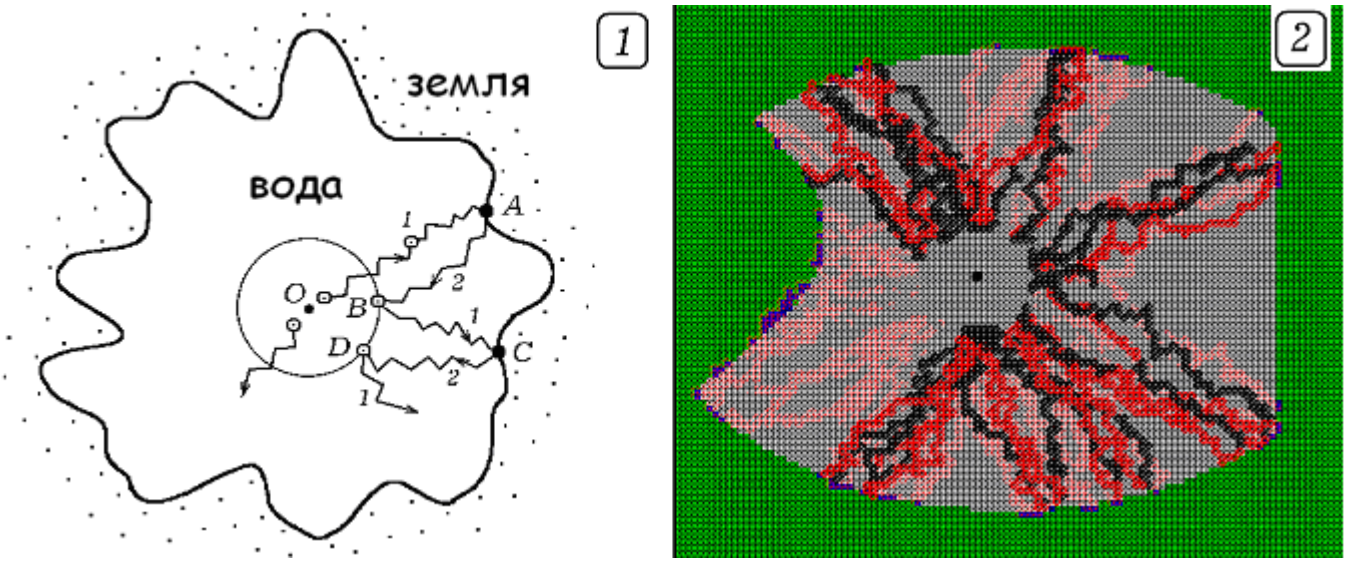

Рис. 8. К задаче об исследовании береговой линии.

Еще одним примером использования мультиагентного подхода является задача о поведении муравьев, которые сначала ищут пищу, а затем переносят ее в муравейник [14]. Представим себе несколько муравьев, выходящих из муравейника A в произвольных направлениях и изменяющих направление движения хаотическим образом. Они находятся в состоянии 1, которое будем называть режимом поиска. Каждый муравей на

своем пути оставляет индивидуальный след из особых веществ -- **феромонов**, который позволяет ему вернуться обратно в муравейник А. С течением времени феромоны испаряются и след исчезает.

Когда муравей номер 5 случайно обнаружил пищу в точке В, он переходит в состояние 2, которое назовем режимом транспортировки. Он берет небольшую часть пищи и, двигаясь по своему следу, переносит ее в муравейник А, снова оставляя феромоновый след. Достигнув муравейника А, он "сгружает" принесенную порцию пищи и переходит в состояние 3 -- режим порожнего движения от муравейника к пище.

Муравьи, находящиеся в А, узнают, что муравей номер 5 принес пищу. Они переходят в состояние 3 и устремляются по его феромоновому следу, выделяя свои феромоны. Поэтому пока по пути АВ происходит движение муравьев, переносящих пищу, феромоновый след не исчезает. Как только пища в В закончилась, муравьи перестают туда ходить, феромоновый след исчезает. Муравьи снова переходят в режим поиска (состояние 1), находят пищу в точке С и все повторяется снова.

Используя мультиагентный подход, можно написать компьютерную программу, моделирующую поведение колонии муравьев, которые ищут выход из воображаемого лабиринта. Когда речь идет о нахождении наикратчайшего пути между двумя вершинами размеченного графа, рассмотренный выше способ не является оптимальным, так как методы перебора или динамического программирования приводит к результату за меньшее число шагов. Однако можно представить ситуацию, когда исследователь с несколькими роботами-агентами оказался в реальном лабиринте, граф которого неизвестен. Если роботы будут вести себя подобно муравьям, которые ищут пищу и при этом оставляют феромоновый след, то им удастся найти выход из лабиринта.

Мультагентный подход позволяет исследовать коллективное поведение нескольких живых организмов, имеющих рецепторы, примитивную нервную систему и эффекторы, отвечающие за перемещение и их взаимодействие друг с другом и внешней средой. Искусственная нервная система каждого программного агента-организма при этом моделируется с помощью нейросети, которая способна обучаться. Каждый агент может иметь свою целевую функцию, определяющую его поведение. Подобные исследования позволяют установить минимальный набор качеств, которыми должны обладать организмы для того, чтобы они могли обучаться и приспосабливаться к тем или иным изменениям окружающей среды.

Мультиагентные системы, как правило, используются для решения проблем, которые не решаются с помощью монолитной или одноагентной системы. Среди них задачи, связанные с моделированием торговли, распространением товаров по сети дистрибъюторов, транспортных потоков, различных социальных процессов, информационных многопроцессорных систем, демографической ситуации, миграции людей и животных, распространения эпидемий и болезней при использовании биологического оружия, созданием экспертных систем поддержки принятия решения, эффекта виртуальной реальности в компьютерных играх и т.д. Эти исследования позволяют понять, как изменения индивидуального поведения агента влияют на поведение всей системы в целом, а также помогают решить различные задачи практического характера.

4.6. Изучение автоволновых процессов методом клеточных автоматов

Рассмотрим использование метода клеточных автоматов (КА) для моделирования реакции Белоусова–Жаботинского, сопровождающейся возникновением химических автоколебаний, круговых и спиральных автоволн и т.д. Представим **двумерную активную среду**, каждый элемент которой может находиться в трех различных состояниях: **покоя**, **возбуждения** и **рефрактерности**. При отсутствии внешнего воздействия элемент находится в состоянии покоя. В результате воздействия элемент переходит в возбужденное состояние, приобретая способность возбуждать соседние элементы. Через некоторое время после возбуждения элемент самостоятельно переключается в состояние рефрактерности, находясь в котором он не может быть возбужден. Затем элемент сам возвращается в исходное состояние покоя, снова приобретая способность переходить в возбужденное состояние. Промоделируем процессы, происходящие в двумерной активной среде при различных параметрах среды и начальном распределении возбужденных элементов [7].

Каждый элемент активной среды проходит одну и туже последовательность состояний, поэтому удобно использовать метод клеточных автоматов. Мысленно разобьем экран компьютера на элементы, определяемые индексами i, j и образующие двумерную сетку. Пусть состояние каждого элемента описывается фазой $y^t_{i,j}$ и концентрацией активатора $u^t_{i,j}$

(однокомпонентная модель), где t -- дискретный момент времени. Если элемент находится в покое, то будем считать, что $y_{i,j}^t = 0$. Если вследствие близости возбужденных элементов концентрация активатора $u_{i,j}^t$ достигает порогового значения h, то элемент возбуждается и переходит в состояние 1. Затем на следующем шаге он автоматически переключается в состояние 2, после этого -- в состояние 3 и т.д., оставаясь при этом возбужденным. Достигнув состояния r, элемент переходит в состояние рефрактерности. Через s $(s > r)$ шагов после возбуждения элемент возвращается в состояние покоя. Это так называемая обобщенная модель Винера-Розенблюта, которую математически можно выразить так:

$$y_{i,j}^{t+1} = \begin{cases} y_{i,j}^t + 1, & если \quad 0 < y_{i,j}^t < s, \\ 0, & если \quad y_{i,j}^t = s, \\ 0, & если \quad y_{i,j}^t = 0, \quad u_{i,j}^t < h, \\ 1, & если \quad y_{i,j}^t = 0, \quad u_{i,j}^t \geq h. \end{cases}$$

Будем считать, что при переходе из последнего состояния s в состояние покоя 0 концентрация активатора становится равной 0. При наличии соседнего элемента, находящегося в возбужденном состоянии, она увеличивается на 1. Если l ближайших соседей возбуждены, то на соответствующем шаге к предыдущему значению концентрации активатора прибавляется число возбужденных соседей: $u_{i,j}^{t+1} = u_{i,j}^t + l$. Можно ограничиться учетом восьми соседних элементов.

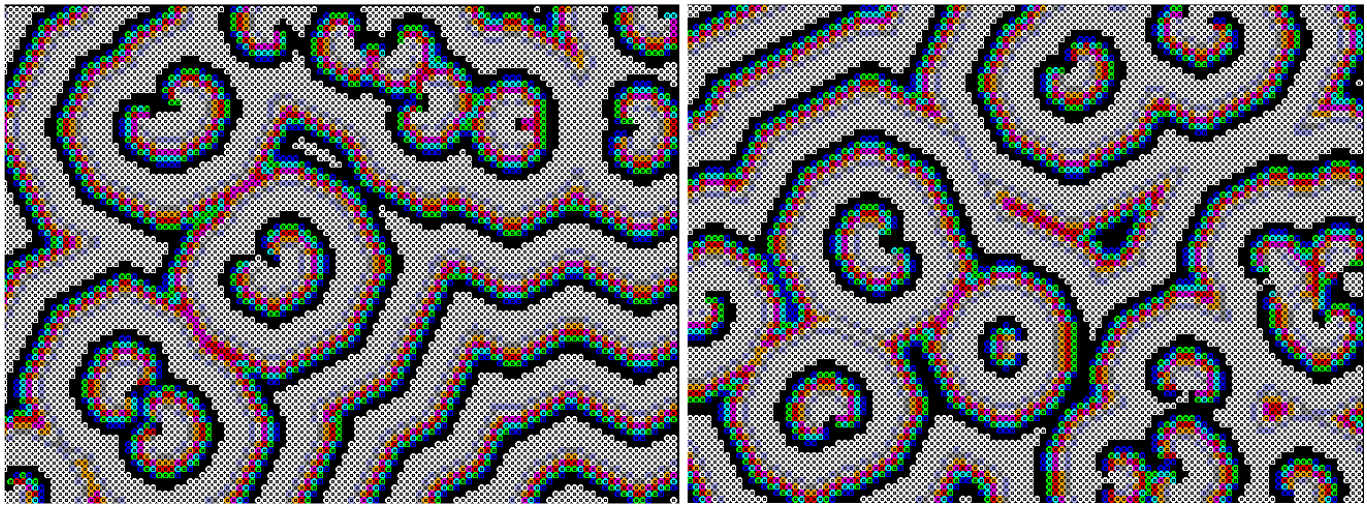

Рис. 9. Моделирование автоволн методом клеточных автоматов.

Рассмотренный алгоритм реализован в компьютерной программе ПР -- 11, написанной для хаотического начального распределения возбужденных элементов активной среды. При ее запуске через некоторое время на экране появляется совокупность упорядоченных автоволновых структур, которые очень похожи на структуры, наблюдаемые в опытах Белоусова-Жаботинского. Задавая различные начальные распределения возбужденных элементов среды, а также положение и частоту химических осцилляторов, можно промоделировать известные эксперименты с автоволнами [6, 7].

Автоволновые процессы связаны с диффузией реагирующих веществ (активаторов, ингибиторов и т.д.) и теплопроводностью среды. Диффузия и теплопроводность также могут быть промоделированы методом клеточных автоматов, пусть и не очень точно [2, 15]. Сопоставляя получающиеся результаты с решением диффуравнения теплопроводности методом конечных разностей [9], можно предположить, что метод конечных разностей является частным случаем метода клеточных автоматов. На эту мысль наводит и тот факт, что в ОЗУ цифровой ЭВМ для записи любого действительного числа отводится конечное количество ячеек памяти, и поэтому значение температуры в любом узле сетки изменяется дискретно, как и состояние ячейки в методе клеточных автоматов. Однако, дискретный характер функционирования клеточных автоматов -- это важное качество данной модели, в то время как ошибки вычислений, обусловленные дискретностью ЭВМ, являются нежелательным фактором, влияние которого стараются уменьшить, увеличивая разрядность памяти ЭВМ. Поэтому мы и в дальнейшем будем придерживаться общепринятого подхода и считать, что это два разных метода.

4.7. Другие примеры использования клеточных автоматов

Еще одним известным примером использования метода клеточных автоматов является модель **решеточного газа** [12, 15]. Представим себе однородную двумерную сетку по вертикальным и горизонтальным отрезкам которой могут перемещаться частицы. Будем считать, что в любой дискретный момент времени на каждом отрезке и в каждом узле находится не более одной частицы. Состояние s_{ij}^{t} каждого узла однозначно зада-

ется четырехразрядным двоичным числом. Запись $s_{ij}^{t} = (1101)$ означает, что в узел с координатами i, j в момент t входят три частицы, двигаясь по первому, второму и четвертому направлениям (рис. 10.1). Чтобы описать движение молекул газа, задают правила, описывающие движение и соударение частиц. Например, при соударении происходит изменение направления движения молекул в соответствии с правилом: (1010) $\rightarrow$ (0101), (0101) $\rightarrow$ (1010) (рис. 10.2 и 10.3).

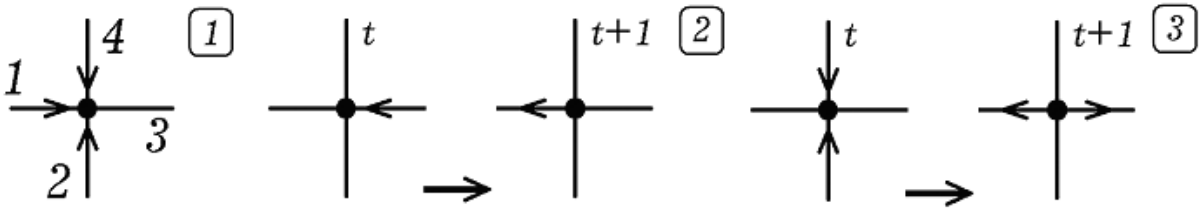

Рис. 10. К объяснению модели "решеточный газ".

На рис. 11, представленном в [15, с. 41] приведен результат моделирования вытекания решеточного газа из контейнера в большой сосуд. Получившаяся модель обратима: если на некотором шаге t поменять направления движения всех частиц на противоположные, то частицы газа вернутся обратно в сосуд, то есть система перейдет в исходное состояние. Это объясняется тем, что в этой модели отсутствуют потери информации, связанные с приближенными вычислениями и округлениями. Если же, поменяв направления скоростей, внести небольшую ошибку, например, добавить 1 молекулу, то система не вернется в исходное состояние.

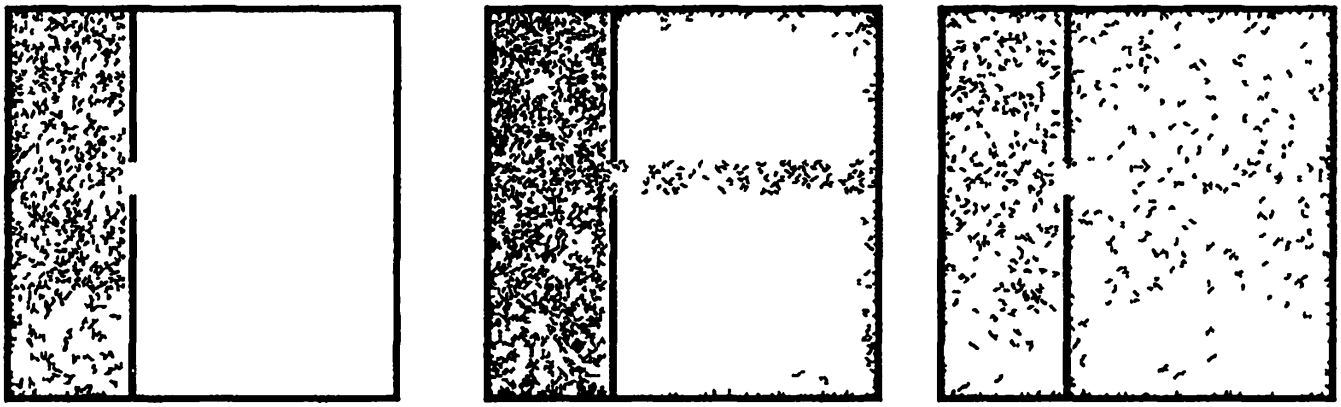

Рис. 11. Вытекание решеточного газа (из [15, с. 41]).

Модель "решеточный газ", реализованная на прямоугольной или треугольной решетке, позволяет промоделировать обтекание жидкостью препятствия, движение частиц снега в ветреную погоду, распространение

областей сжатия и разрежения в газообразной среде, распространение, отражение, преломление волн и другие явления [12, 15].

Еще одним интересным применением метода клеточных автоматов (КА) является моделирование течения вязкой жидкости или движения сыпучих материалов [2]. В книге [6] приведен пример программы, позволяющей промоделировать падение струи вязкой жидкости на препятствие и заполнение сосуда произвольной формы (рис. 12, 13).

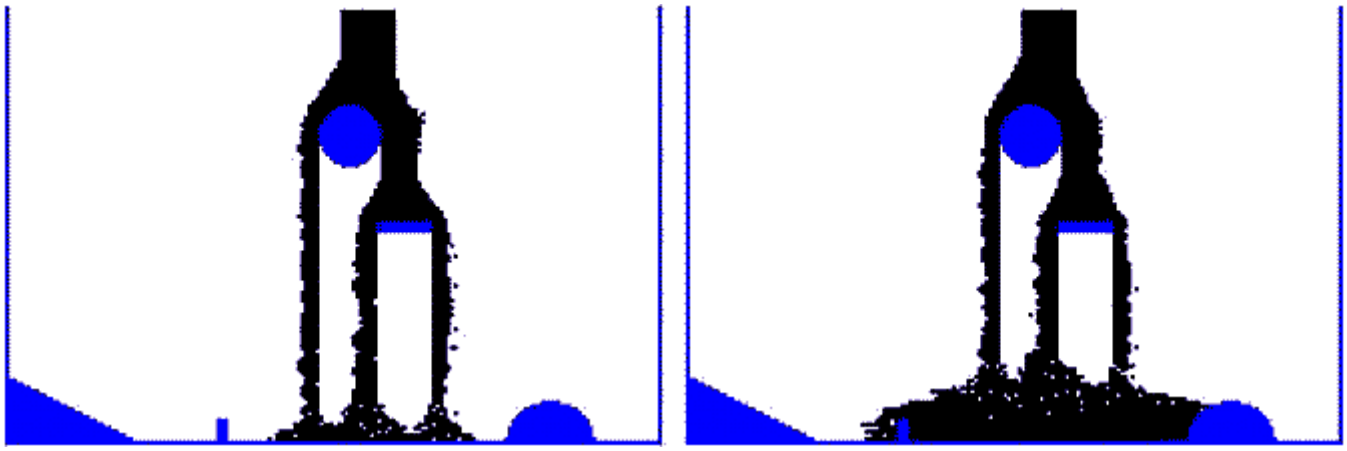

Рис. 12. Падение сыпучих материалов, вытекание вязкой жидкости.

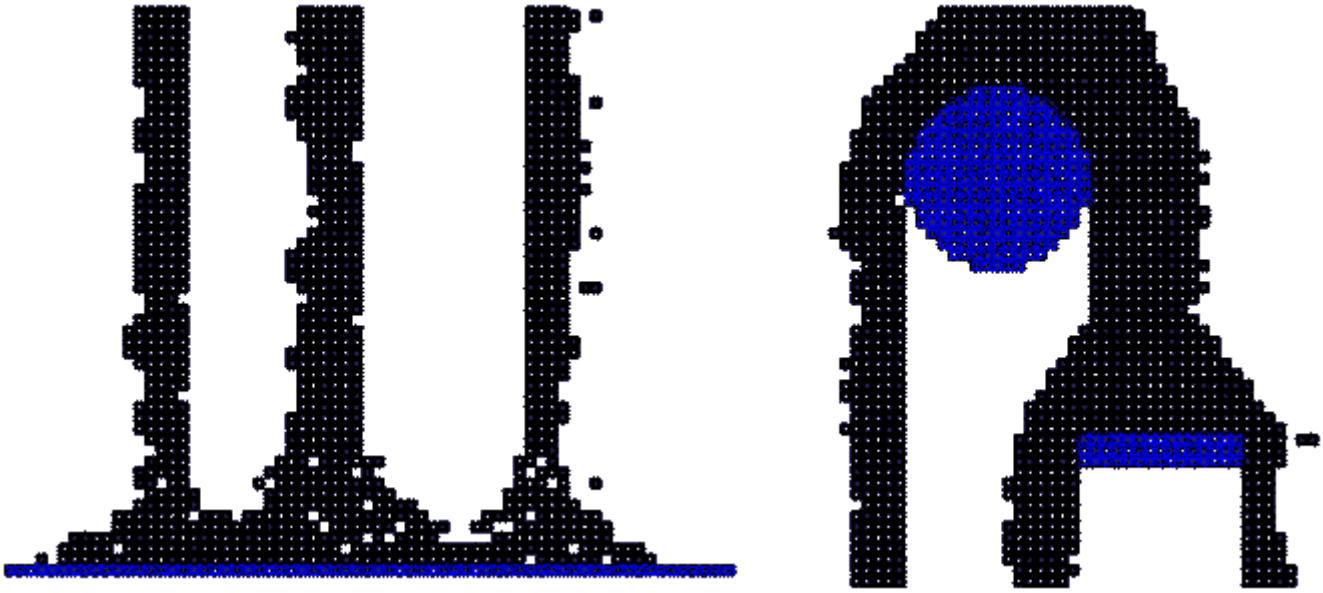

Рис. 13. Моделирование течения вязкой жидкости с помощью КА.

Метод клеточных автоматов позволяет промоделировать и другие явления. Например, в книге [15] проанализирована модель движения потока машин по дорогам, образующим двумерную сетку. Машины рассматриваются как частицы определенных размеров, которые движутся вдоль отрезков, соединяющих перекрестки. Модель позволяет изучить связь между плотностью автомобилей, пропускной способностью дороги и средней скоростью автомобилей.

Метод подвижных клеточных автоматов возник в результате объединения классического метода клеточных автоматов и метода дискретных элементов [14]. Он используется для решения задач механики деформируемого твердого тела, моделирования разрушения материала, возникновения повреждений, распространения трещин и перемешивания различных веществ. Сущность метода состоит в том, что объект рассматривается как совокупность взаимодействующих частиц, ведущих себя как клеточные автоматы. Каждая пара таких частиц в простейшем случае может находится в двух состояниях: 1) в связанном, когда они принадлежат одному телу; 2) в несвязанном, когда автоматы принадлежат различным телам или фрагментам тела.

Поведение составляющих тело частиц зависит от силы их взаимодействия и правила изменения их состояния. Перемещения частиц рассчитываются исходя из уравнений динамики с учетом состояния связи между частицами. При этом вместо статической сеточной концепции используется концепция соседей. Частицы могут менять своих соседей "отсоединяясь" от одних и "подсоединяясь" к другим. Метод **фронтальных клеточных автоматов** позволяет исследовать рост кристаллов [14].

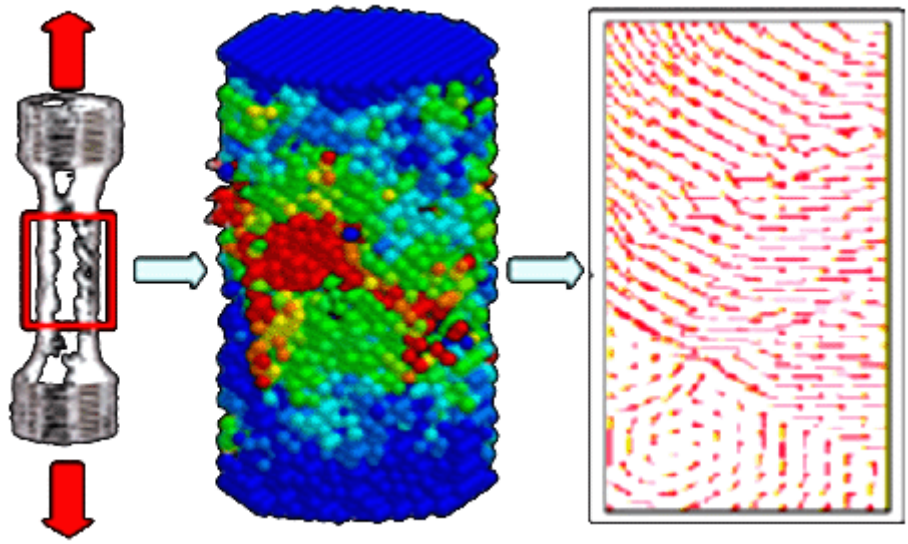

Рис. 14. К пояснению метода подвижных КА (из [14]).

Клеточные автоматы применяются для решения задач искусственного интеллекта и позволяют создать обучающуюся систему распознавания образов, которая при одних начальных условиях будет приводить к одному результату (аттрактору), а при других -- к другому. Они могут быть использованы для моделирования различных полей, сред, газодинамических и гидродинамических явлений, диффузии, процесса горения, некоторых химических превращений, распространения эпидемии, течений

жидкости со свободной границей, распространения тепловых потоков, роста и динамики магнитных или иных доменов, роста минеральных дендритов, динамики толпы людей или стаи животных, составления генетических алгоритмов и решения других задач. Так, можно промоделировать биологическую систему из клеток, условием выживания которых является выработка определенной последовательности выходных данных (реакции) на совокупность входных данных (раздражение), идущих от среды. Необходимо, чтобы такой автомат имел механизм случайного изменения правил выработки реакции и наследования. Эта модель позволяет понять условия возникновения жизнеспособных организмов.

Создатель игры "Жизнь" Конуэй утверждал, что все процессы во вселенной можно изучать методами клеточных автоматов, заложив в них соответствующие правила движения и взаимодействия элементарных частиц. Это справедливо лишь отчасти. Автоматный подход и, в частности, метод клеточных автоматов безусловно позволяет решить огромный класс задач, однако в ряде случаев целесообразно использовать другие методы моделирования.

Приложение

В приложении представлены тексты программ, позволяющие решить рассмотренные выше задачи. Они написаны в средах Borland Pascal 7.0, Free Pascal 1.0.10.

ПР-1.

```
uses crt;
var x1,x2,x3,y1,y2: boolean;
BEGIN
For x1:=false to true do
  For x2:=false to true do
    For x3:=false to true do begin
      y1:=not((x1)or(x2)); y2:=((x1)or(x2))and(x3);
      writeln(x1,' ',x2,' ',x3,' ',y1,' ',y2);
    end; ReadKey;
END.
```

ПР-2.

```
uses crt;
var i: integer; x,y,S,a,b,c: string;
    q: integer; label m;
```

```
BEGIN S:='babacaabcbabcbcbaaccbcbabcbccaacbb';
writeln('S=',S);q:=1;
For i:=1 to length(S) do begin x:=copy(S,i,1);
If (x='a')and(q=1) then begin q:=1; y:='B'; goto m; end;
If (x='c')and(q=1) then begin q:=2; y:='A'; goto m; end;
If (x='b')and(q=1) then begin q:=4; y:='C'; goto m; end;
If (x='b')and(q=2) then begin q:=2; y:='D'; goto m; end;
If (x='a')and(q=2) then begin q:=3; y:='B'; goto m; end;
If (x='c')and(q=2) then begin q:=4; y:='A'; goto m; end;
If (x='a')and(q=3) then begin q:=4; y:='C'; goto m; end;
If (x='c')and(q=3) then begin q:=3; y:='E'; goto m; end;
If (x='b')and(q=3) then begin q:=2; y:='C'; goto m; end;
If (x='a')and(q=4) then begin q:=4; y:='E'; goto m; end;
If (x='b')and(q=4) then begin q:=3; y:='F'; goto m; end;
If (x='c')and(q=4) then begin q:=1; y:='D'; goto m; end;
m: write(q,' ',y,' | '); end; ReadKey;
END.
```

ТР-3.

```
uses crt, graph;
var ugol,dlina,Gd,Gm,t: integer; alfa,x,y,x1,y1: real;
Procedure Napravo(ugol: integer);
begin alfa:=alfa+ugol*3.14/180; end;
Procedure Shag(dlina: integer);
begin circle(round(x),round(y),2);
  x1:=x+dlina*cos(alfa); y1:=y+dlina*sin(alfa);
  line(round(x+1),round(y+1),round(x1+1),round(y1+1));
  line(round(x),round(y),round(x1),round(y1)); x:=x1; y:=y1;
end;
BEGIN x:=320; y:=240;
Gd:=Detect; InitGraph(Gd, Gm, 'c:\bp\bgi');
Repeat inc(t); Napravo(-10); Shag(2*t); Napravo(60);
       Shag(3*t); Napravo(-100); Shag(30);
until t>20; Readkey; CloseGraph;
END.
```

ТР-4.

```
uses crt, dos;
const N=35; t=100;
var z,lenta: string; a, kom : array [1..N] of string;
    k, kk : array [1..N] of integer;
    x, p, i, ii : integer; Label m1, m2;
Procedure Programma;
Begin x:=7;
lenta:='VVVVVV-VV--------------------------------------';
kom[1]:='left';  k[1]:=2;
kom[2]:='if';    k[2]:=1; kk[2]:=3;
```

```
kom[3]:='erase'; k[3]:=4;
kom[4]:='right'; k[4]:=5;
kom[5]:='if';    k[5]:=4; kk[5]:=6;
kom[6]:='erase'; k[6]:=7;
kom[7]:='right'; k[7]:=8;
kom[8]:='if';    k[8]:=9; kk[8]:=1;
kom[9]:='stop';  k[9]:=0;
end;
Procedure Pechat;
begin writeln; p:=p+1; For i:=1 to N do write(a[i],' ');
  writeln('| ',p,' |');
  For i:=1 to x-1 do write('--'); write('M');
  delay(200); end;
BEGIN clrscr; Programma;
   For i:=1 to N do begin a[i]:=copy(lenta,i,1); end;
   Pechat; ii:=1; m2: If Keypressed then goto m1;
   If kom[ii]='stop' then goto m1;
   If kom[ii]='left'  then begin x:=x-1; Pechat;
                              ii:=k[ii]; goto m2; end;
   If kom[ii]='right' then begin x:=x+1; Pechat;
                              ii:=k[ii]; goto m2; end;
   If kom[ii]='erase' then begin a[x]:='-'; Pechat;
                              ii:=k[ii]; goto m2; end;
   If kom[ii]='metka' then begin a[x]:='V'; Pechat;
                              ii:=k[ii]; goto m2; end;
   If kom[ii]='if' then begin z:=a[x];
   If z='-' then ii:=k[ii] else
   ii:=kk[ii]; goto m2; end; writeln;
   writeln('OSHIBKA V STROKE', ii);
m1: writeln; writeln('KONEC RABOTI');
Repeat until KeyPressed;
END.
```

ПР-5.

```
uses crt, graph;
const N=50;
Var i,k,m,s,flag : integer;
    x1,x2,x4,x5,x6,q,lenta: string;
    kom,a : array[1..N] of string; Label m1;
BEGIN clrscr; lenta:='_ABBAABAB_______';
m:=1; q:= '1';
kom[1]:='1_>1_R';  kom[2]:='2_>3|R';
kom[3]:='3_>4AL';  kom[4]:= '4_>1_R';
kom[5]:='1A>2*R';  kom[6]:='2A>2AR';
kom[7]:='3A>3AR';  kom[8]:='4A>4AL';
kom[9]:='1B>1BR';  kom[10]:='2B>2BR';
kom[11]:='4B>4BL'; kom[12]:='1*>1*R';
```

```
kom[13]:='2*>2*R'; kom[14]:='4*>4*L';
kom[15]:='1|>1|S'; kom[16]:='2|>3|R';
kom[17]:='4|>4|L';
For i:=1 to N do a[i]:=copy(lenta,i,1);
Repeat flag:=0; s:=s+1;
  For i:=1 to N do begin
   x1:=copy(kom[i],1,1); x2:=copy(kom[i],2,1);
   x4:=copy(kom[i],4,1); x5:=copy(kom[i],5,1);
   x6:=copy(kom[i],6,1);
   If (flag=0)and(x1=q)and(x2=a[m]) then
     begin q:=x4; a[m]:=x5;
     If x6='R' then m:=m+1;
     If x6='L' then m:=m-1;
     If x6='S' then goto m1;
     flag:=1; end; end;
   m1: k:=k+1;
For i:=1 to 25 do write(a[i],' ');
writeln('   ',q,'   k=',k); delay(5);
For i:=1 to m-1 do write('=='); write('|'); writeln;
until x6='S'; Readkey;
END.
```

ПР-6.

```
uses crt;
const x: array[1..9] of integer=(0,0,0,
                                 1,1,1,
                                 0,0,0);
w: array[1..3,1..9] of real=((0,-1,1,1,0,1,-1,-1,0),
      (0,1,1,-1,0,1,-1,-1,0),(1,0,1,-1,0,0,1,-1,1));
v: array[1..2,1..3] of real=((2,2,-2),(-2,2,2));
var m,i,j,k,DV,MV,EC : integer;
    S,y : array[1..3] of real; T,z : array[1..2] of real;
BEGIN Clrscr;
For j:=1 to 3 do begin S[j]:=0;
    For i:=1 to 9 do S[j]:=S[j]+w[j,i]*x[i];
    If S[j]>0.5 then y[j]:=1 else y[j]:=0;
    Writeln('j= ',j,' | S= ',S[j]:2:1,' | y= ',y[j]:1:1);
  end;
For k:=1 to 2 do begin T[k]:=0;
    For j:=1 to 3 do T[k]:=T[k]+v[k,j]*y[j];
    If T[k]>0 then z[k]:=1 else z[k]:=0;
    Writeln('k= ',k,' | T= ',T[k]:2:1,
          ' | Vihod z= ',z[k]:1:0); end;
Readkey;
END.
```

ПР-7.

```
Uses crt,graph;
Var s,s1: array[0..60] of integer; Gd,Gm,i,j: integer;
BEGIN s[3]:=2; s[2]:=1; s[50]:=2;
Gd:= Detect; InitGraph(Gd, Gm, 'c:\bp\bgi');
Repeat inc(j);
For i:=1 to 59 do begin
  If ((s[i]=0)and((s[i-1]=2)or(s[i+1]=2))) then s1[i]:=2;
  If s[i]>0 then s1[i]:=s[i]-1; end; delay(200);
For i:=1 to 60 do begin s[i]:=s1[i];
  If s[i]=0 then circle(8*i,10*j,1);
  If s[i]=1 then circle(8*i,10*j,2);
  If s[i]=2 then circle(8*i,10*j,3); end;
until (KeyPressed)or(j>3000);
Repeat until KeyPressed; CloseGraph;
END.
```

ПР-8.

```
uses dos, crt, graph;
const  N=220;
var a,b: array[1..N] of word; Gd,Gm,i,j: integer;
Procedure Draw;
begin For i:=1 to N do begin
if a[i]=1 then circle(3*i,round(3*j),2); end; end;
BEGIN Gd:=Detect; InitGraph(Gd, Gm, 'c:\bp\bgi'); a[70]:=1;
Repeat Draw; inc(j); delay(500);
For i:=3 to N-2 do begin
If a[i-1]+a[i+1]=1 then b[i]:=1 else b[i]:=0; end;
For i:=3 to N-2 do a[i]:=b[i];
until keypressed; CloseGraph;
END.
```

ПР-9.

```
uses crt; const N=23;
type  z1 = record x,y,xy,x1,y1: real end;
     massiv = array[0..N+1,0..N+1] of integer;
var z,y,x,x1: massiv;  s,i,j,k,l,m: integer;
Procedure Print;
var  i,j: integer;
begin clrscr;
For i:=1 to N do begin For j:=1 to N do begin
 If x[i,j]=1 then Write(' *'); If x[i,j]=0 then Write('  ');
end; Writeln; end; end;
Procedure Oboznach;
var i,j: integer;
begin
 For i:=1 to N do For j:=1 to N do x1[i,j]:=x[i,j]; end;
```

```
Procedure Raschet;
var i,j: integer;
begin For i:=1 to N do For j:=1 to N do  begin
   S:=x1[i-1,j-1]+x1[i-1,j]+x1[i-1,j+1]+x1[i,j-1]+
       x1[i,j+1]+x1[i+1,j-1]+x1[i+1,j]+x1[i+1,j+1];
   If s=3 then x[i,j]:=1;
   If (s<2)or(s>3) then x[i,j]:=0;
end; end;
BEGIN For i:=1 to N do For j:=1 to N do x[i,j]:=0;
 x[6,3]:=1; x[7,4]:=1;
 x[5,5]:=1; x[6,5]:=1; x[7,5]:=1; Print;
 Repeat delay(300); Oboznach; Raschet; Print;
 until keypressed;
END.
```

ТТР-10.

```
Uses dos, crt, graph;
Const N=110; M=100; N_a=8;
type Agent = record x,y,sost: integer; end;
Var u: array [1..N,1..M] of integer;
i,j,a,b,t,Gd,Gm: integer;
Ag: array[1..N_a]of Agent; l,l1:real;
Procedure Smeshen;
begin Ag[i].x:=Ag[i].x+round(random(400)/100)-2;
Ag[i].y:=Ag[i].y+round(random(400)/100)-2;
end;
Procedure Draw1;
begin For i:=1 to N do For j:=1 to M do begin
if u[i,j]=0 then setcolor(white);
if u[i,j]=1 then setcolor(green);
if u[i,j]=2 then setcolor(blue);
rectangle(5*i,5*j,5*i+3,5*j+3); end; end;
Procedure Draw2;
begin For i:=1 to N_a do begin
if Ag[i].sost=2 then setcolor(red) else setcolor(black);
circle(5*Ag[i].x,5*Ag[i].y,2);
circle(5*Ag[i].x,5*Ag[i].y,1); end; end;
BEGIN Gd:= Detect; InitGraph(Gd, Gm, 'c:\bp\bgi');
  Randomize;
  For i:=1 to N do For j:=1 to M do begin
    If (i-55)*(i-55)/2+(j-50)*(j-50)>1700 then u[i,j]:=1;
    If (i<25*sin(j/20-0.5))or(i>98) then u[i,j]:=1; end;
  For i:=1 to N_a do begin Ag[i].sost:=1;
    Ag[i].x:=45+round(random(100)/10);
    Ag[i].y:=45+round(random(100)/10); end; Draw1;
Repeat inc(t);
  For i:=1 to N_a do begin
```

```
    If Ag[i].sost=1 then begin
      l:=sqr(Ag[i].x-50)+sqr(Ag[i].y-50);
      a:=Ag[i].x; b:=Ag[i].y; Smeshen;
      l1:=sqr(Ag[i].x-50)+sqr(Ag[i].y-50);
      If l1<l-3 then begin Ag[i].x:=a; Ag[i].y:=b; end;
    If u[Ag[i].x,Ag[i].y]>0 then begin Ag[i].sost:=2;
      u[Ag[i].x,Ag[i].y]:=2 end; end;
    If Ag[i].sost=2 then begin
      l:=sqr(Ag[i].x-50)+sqr(Ag[i].y-50);
      a:=Ag[i].x; b:=Ag[i].y; Smeshen;
      l1:=sqr(Ag[i].x-50)+sqr(Ag[i].y-50);
      If l1>l-3 then begin Ag[i].x:=a; Ag[i].y:=b; end; end;
    l1:=sqr(Ag[i].x-50)+sqr(Ag[i].y-50);
    If u[Ag[i].x,Ag[i].y]>0 then u[Ag[i].x,Ag[i].y]:=2;
    If l1<100 then Ag[i].sost:=1; end;
  If t>500 then begin Draw1; t:=0; end; Draw2; delay(50);
until KeyPressed; CloseGraph;
END.
```

ТТР-11.

```
uses dos, crt, graph;
const N=100; M=70; s=18; r=5; h=5;
var y, yy, u : array [0..N+1,0..M+1] of integer;
    ii, jj, j, t, Gd, Gm: integer; i: Longint;
label met;
BEGIN
Gd:= Detect; InitGraph(Gd, Gm, 'c:\bp\bgi'); Randomize;
For i:=1 to N do For j:=1 to M do begin
  y[i,j]:=round(random(400)/10);
  If y[i,j]>15 then y[i,j]:=0; end;
Repeat t:=t+1;
  For i:=1 to N do For j:=1 to M do begin
    If (y[i,j]>0) and (y[i,j]<s) then yy[i,j]:=y[i,j]+1;
    If y[i,j]=s then begin yy[i,j]:=0; u[i,j]:=0; end;
    If y[i,j] <> 0 then goto met;
    For ii:=i-1 to i+1 do For jj:=j-1 to j+1 do begin
      If (y[ii,jj]>0)and(y[ii,jj]<=r) then u[i,j]:=u[i,j]+1;
      If u[i,j]>=h then yy[i,j]:=1; end; met:
  end;
  For i:=1 to N do For j:=1 to M do begin y[i,j]:=yy[i,j];
    If y[i,j]<10 then setcolor(y[i,j]) else setcolor(white);
    circle(6*i-10,6*j,3); circle(6*i-10,6*j,2); end;
until KeyPressed; CloseGraph;
END.
```

ЛИТЕРАТУРА

1. Астафьев Г.Б., Короновский А.А., Храмов А.Е. Клеточные автоматы: Учебно-методическое пособие. -- Саратов: Изд-во ГосУНЦ "Колледж", 2003. -- 24 с.
2. Булавин Л.А., Выгорницкий Н.В., Лебовка Н.И. Компьютерное моделирование физических систем. -- Долгопрудный: Издательский Дом "Интеллект", 2011. -- 352 с.
3. Беркович С. Я. Клеточные автоматы как модель реальности: Поиски новых представлений физических и информационных процессов. -- М.: Изд-во МГУ, 1993. -- 112 с.
4. Гулд Х., Тобочник Я. Компьютерное моделирование в физике: В 2-х частях. Часть 2. -- М.: Мир, 1990. -- 400 с.
5. Задорожный В.Н., Юдин Е.Б. Мультиагентный подход в имитационном моделировании клеточных автоматов и сетевых структур. [Электронный ресурс] -- URL: http://www.gpss.ru/immod07/doklad/71.html (дата обращения 30.08.2012).
6. Майер Р.В. Задачи, алгоритмы, программы. [Электронный ресурс] -- URL: http://maier-rv.glazov.net, http://komp-model.narod.ru.
7. Майер Р.В. Компьютерное моделирование физических явлений. -- Глазов, ГГПИ: 2009. -- 112 с.
8. Майер Р.В. Теоретические основы информатики. Задачи и программы: Учебн. пособ. для студ. высш. учеб. заведений [Электронный ресурс]. -- URL: http://rmajer.narod.ru.
9. Поттер Д. Вычислительные методы в физике. -- М.: Мир, 1975. -- 392 с.
10. Рубанов В.Г., Филатов А.Г. Моделирование систем: Учебное пособие. -- Белгород: Изд–во БГТУ, 2006. -- 349 с.
11. Советов Б.Я., Яковлев С.А. Моделирование систем: Учеб для вузов -- М.: Высш. Шк., 2001. -- 343 с.
12. Тоффоли Т., Марголус Н. Машины клеточных автоматов. – М.: Мир, 1991. -- 280 с.
13. Цетлин М. Л. Исследования по теории автоматов и моделированию биологических систем. -- М.: Наука, 1969. -- 316 с.
14. Википедия [Электронный ресурс]: URL: http://ru.wikipedia.org (дата обращения 23.08.2012).
15. Chopard B., Droz M. Cellular Automata Modeling of Physical Systems. -- Cambridge: University Press. -- 341 p.