**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**"Уфимский государственный авиационный технический университет"**

**Кафедра** Высокопроизводительных вычислительных технологий и систем

**Дисциплина:** Математическое моделирование

**Отчет по лабораторной работе № 1**

**Тема:** «Имитационное моделирование сложных систем с помощью клеточных автоматов»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа ПМ-453 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Шамаев И.Р. |  |  |  |
| Принял | Лукащук В.О. |  |  |  |

**Уфа 2023**

**Цель работы:** получить навык имитационного моделирования сложных динамических систем с использованием клеточных автоматов на примере моделей биологических систем.

**Задание**:

**Задача I. Клеточный автомат "Жизнь"**

Выполнить программную реализацию клеточного автомата, функционирующего в соответствии со следующими правилами:

1. клетка может находиться в двух состояниях - пассивном и активном;
2. в качестве окрестности рассматривается восемь соседних клеток;
3. если в окрестности пассивной клетки две активных, то данная клетка также становится активной ("рождается");
4. если в окрестности активной клетки три или более активных клеток, то она становится пассивной ("умирает").

Реализовать алгоритм на клеточном пространстве ячеек. Начальное распределение активных и пассивных клеток – случайное, подчиняющееся равномерному закону распределения. Также подобрать начальные распределения, соответствующие стационарным и циклическим структурам (по три примера каждой структуры).

**Задача II. Клеточный автомат "Нейронная сеть"**

Данный автомат имитирует явления в однородной двумерной нейронной сети, состоящей из возбудимых элементов, и функционирует по следующим правилам:

1. клетка может находиться в трех состояниях: покоя, активном и состоянии восстановления;
2. в качестве окрестности рассматриваются восемь соседних клеток;
3. переход в состояние активности зависит от некоторого параметра, называемого *уровнем активатора*. В возбужденном состоянии клетки уровень активатора равен . В других состояниях он распадается на за такт;
4. если клетка была в покое и общее количество активатора в восьми соседних и в данной клетке превысило порог активации , то клетка возбуждается на тактов;
5. через тактов возбужденная клетка переходит в состояние восстановления на В тактов, а затем переходит в состояние покоя.

Реализовать алгоритм при следующих параметрах: клеточное пространство ячеек, . Начальное распределение состояния клеток задано плоским фронтом. Также имеется периодический источник возбуждения ( клетки) с периодом тактов. Выявить характер взаимодействия между собой различных фронтов возбуждения.

**Задача III. Клеточный автомат "Организмы - питательная среда"**

Клеточный автомат моделирует взаимодействие одноклеточных организмов с питательной средой и функционирует по следующим правилам:

1. клеточное пространство образует поле клеток;
2. окрестность клетки составляют восемь соседних клеток;
3. каждой клетке соответствует значение степени питательности раствора (энергоемкости), которое может изменяться от до ;
4. прирост питательности (энергоемкости) раствора клетки за такт времени выполняется следующим образом: при и при , где – скорость прироста питательности;
5. общий запас энергии питательного раствора определяется суммарной питательностью (энергией) всех клеток и не может быть более ;
6. клетка может быть свободной или содержать не более одного одноклеточного или другого живого организма;
7. отдельная особь одноклеточного черпает энергию из питательного раствора клетки, в которой она находится, снижая его питательность и повышая свой запас энергии на за такт;
8. максимально возможное количество энергии, запасаемое одноклеточным, не превышает ;
9. на свои нужды отдельная особь затрачивает энергии за такт;
10. особь всегда старается перейти на соседнюю свободную клетку, выбирая направление перехода случайным образом;
11. время жизни отдельной особи составляет тактов;
12. если время жизни особи превысило продолжительность жизни для данных организмов или запас энергии снизился до нуля, то особь умирает;
13. начиная с возраста тактов особь считается зрелой и может производить себе подобных, затрачивая энергии при каждом делении дополнительно. При этом старая особь переходит на свободную соседнюю клетку, а новая остается в старой. Если свободных клеток в окрестности нет, то деления не происходит;
14. начальное распределение особей по клеточному пространству подчинено равномерному закону распределения. Начальное число особей составляет максимально возможного, равного .

Реализовать алгоритм при следующих параметрах: клеточное пространство ячеек, . Выявить характерные зависимости в поведении колонии одноклеточных.

**Задача IV. Модифицированный клеточный автомат "Организмы - питательная среда"**

Выполнить модификацию алгоритма из задачи III, заменив правило на следующее: особь всегда старается перейти на соседнюю свободную клетку с наибольшим уровнем энергоемкости. Если ячейки в окрестности, имеют меньший запас энергии, то особь остается в прежней клетке.

*Как изменится поведение колонии одноклеточных? Какие явления самоорганизации в данном случае возникают?*

**Ход работы**.

Во всех заданиях решетка клеточного автомата считается замкнутой в тор.

Задача 1.

Код программы представлен в приложении, листинг 1. Активное состояние клетки показано белым цветом, пассивное – бордовым.

В начальный момент времени распределение активных и пассивных клеток – случайное, подчиняющееся равномерному закону распределения.

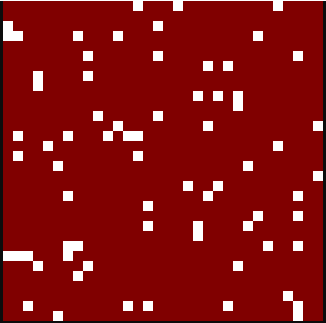


Рисунок . Начальное расположение клеток. 71 активных клеток.

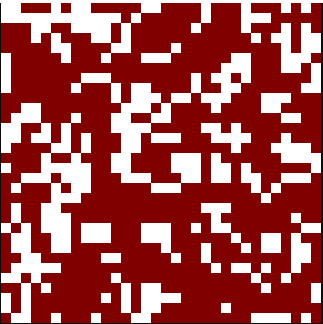


Рисунок . 5-я итерация. 331 активная клетка.

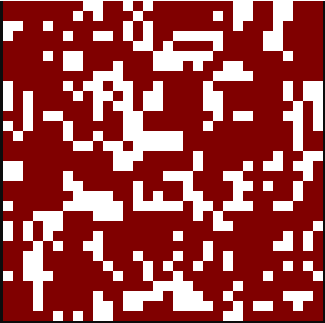


Рисунок . 10-я итерация. 299 активных клеток.

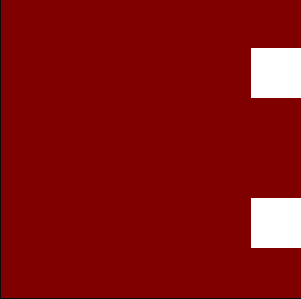


Рисунок . Пример статичной структуры

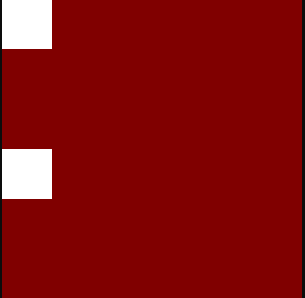
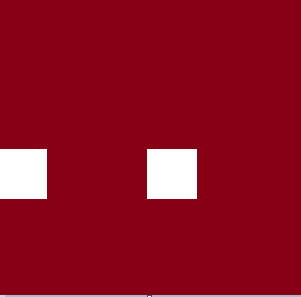
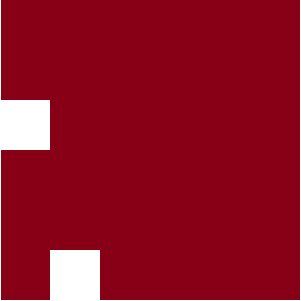
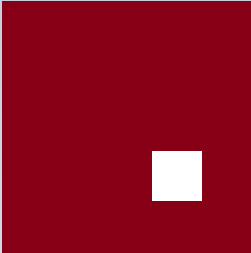
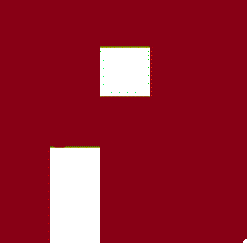
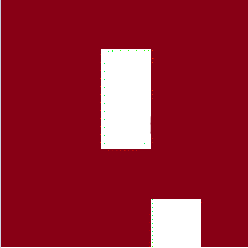
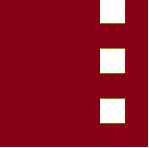


Рисунок . Пример статичной структуры.



Статическая структура



Пример циклической структуры

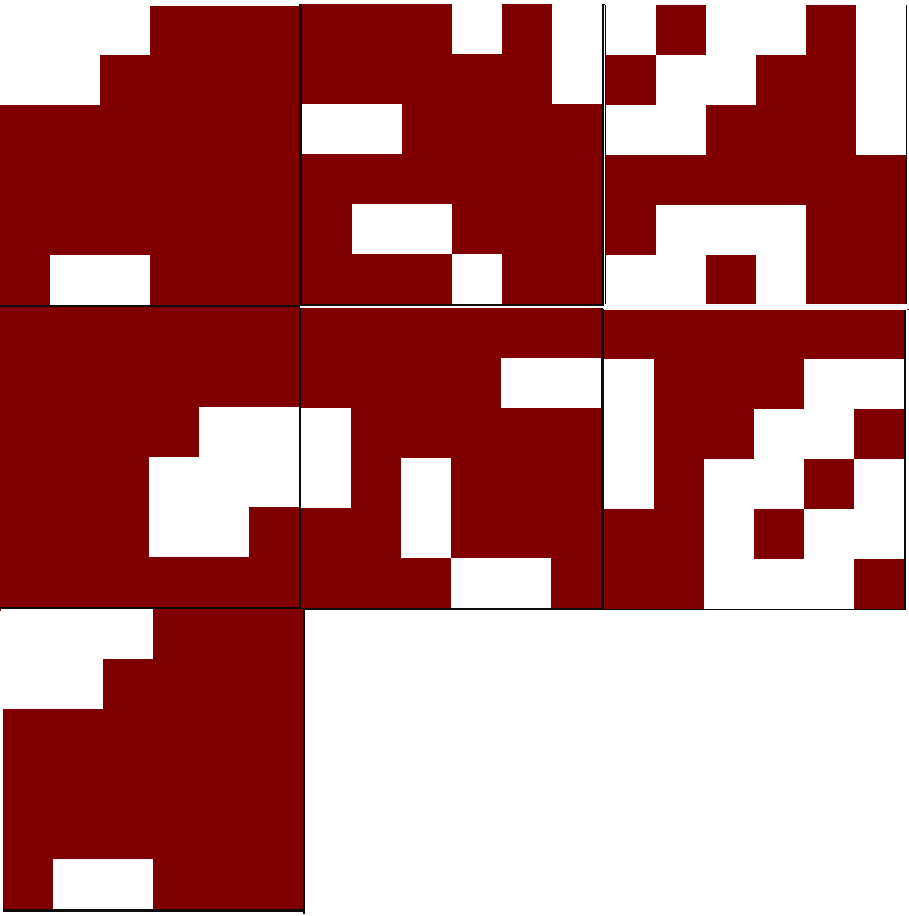


Рисунок 6. Пошаговый пример циклической структуры.

Задача 2.

Код программы представлен в приложении, листинг 2. Активное состояние клетки показано белым цветом, пассивное – бордовым, восстановление – синим.

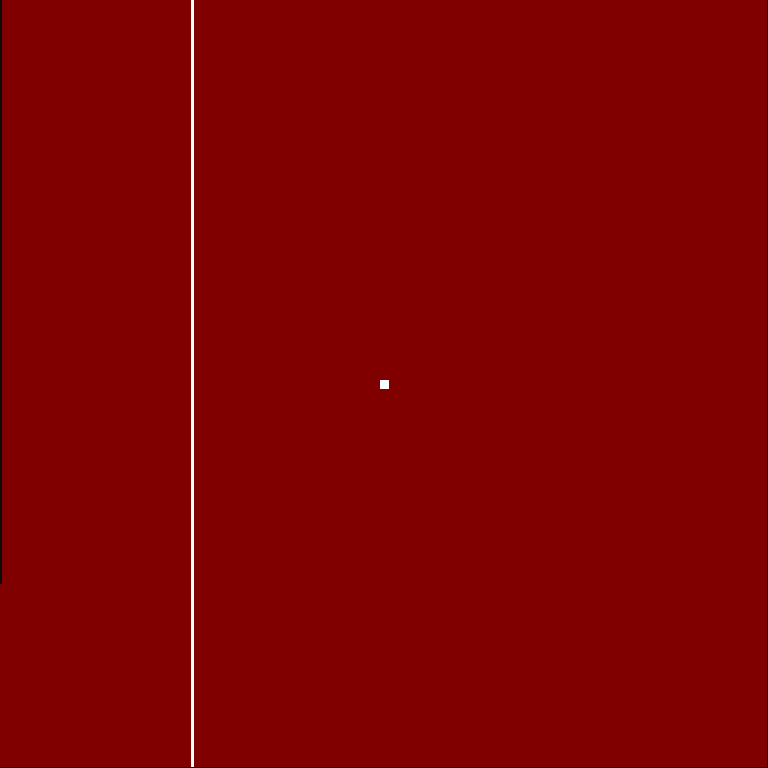


Рисунок 7. Начальное состояние клеточного автомата.

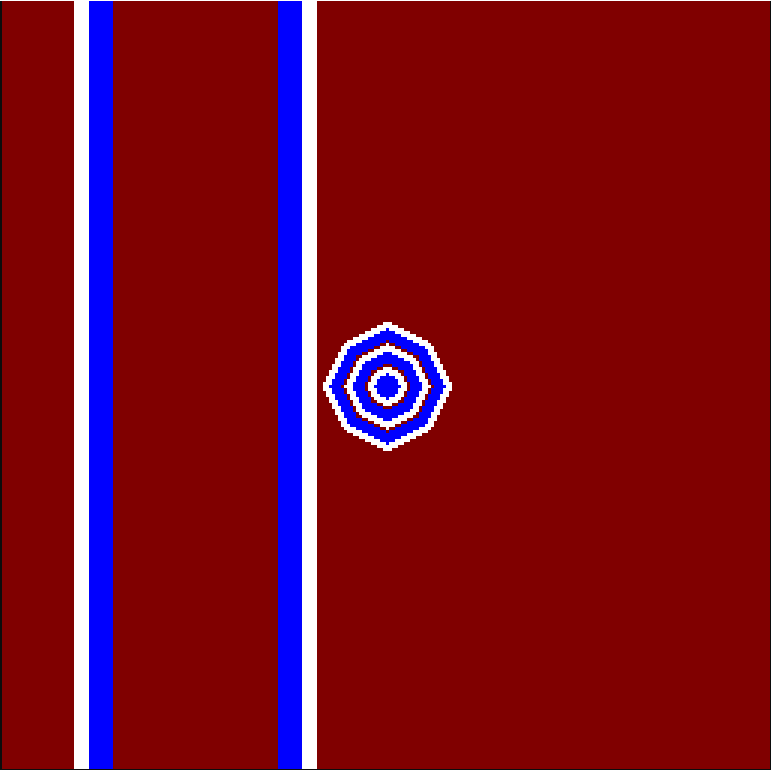


Рисунок 8. Состояние на 40-й итерации.

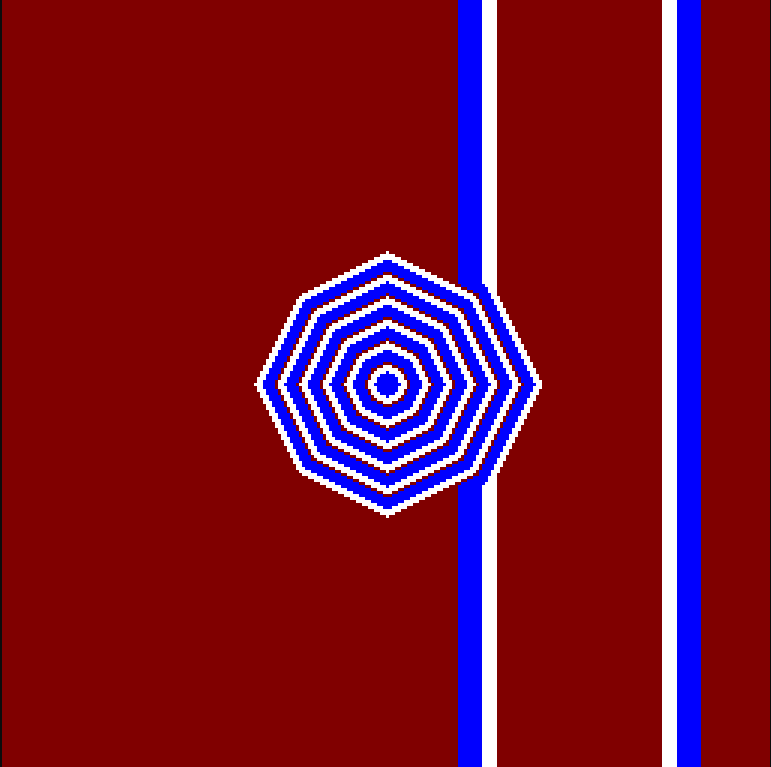


Рисунок 9. Состояние на 100-й итерации.

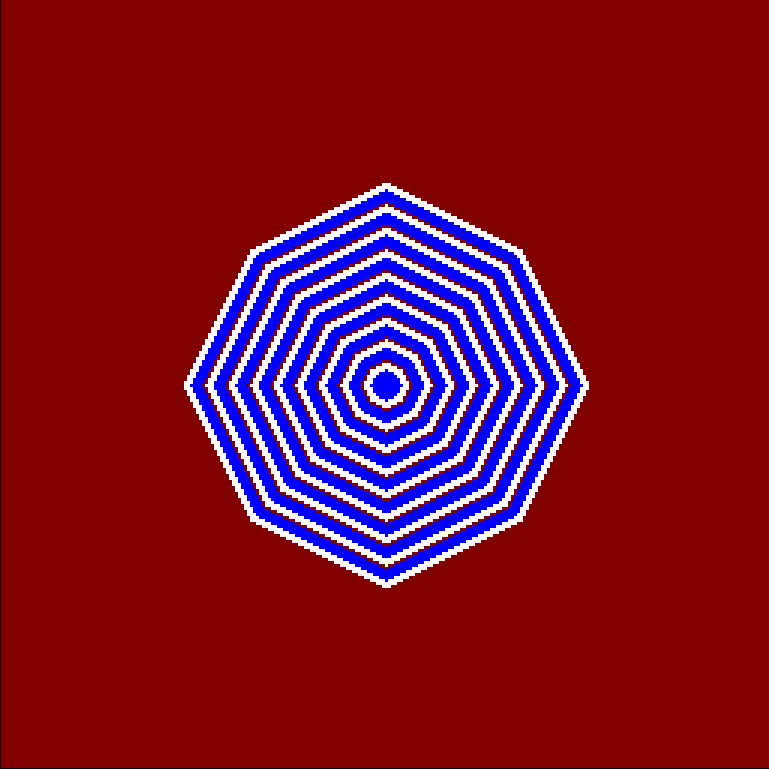


Рисунок 10. Состояние на 147-й итерации.

Периодический источник создает круговые волны. Плоский фронт поглотил в себе первую волну периодического источника, а также самого себя.

Задача 3.

Код программы представлен в приложении, листинг 3. Клетка, содержащая живой организм помечена белым цветом, свободная красным или зеленым, в зависимости от текущего значения степени питательности раствора в ней, чем его больше, тем она зеленее.

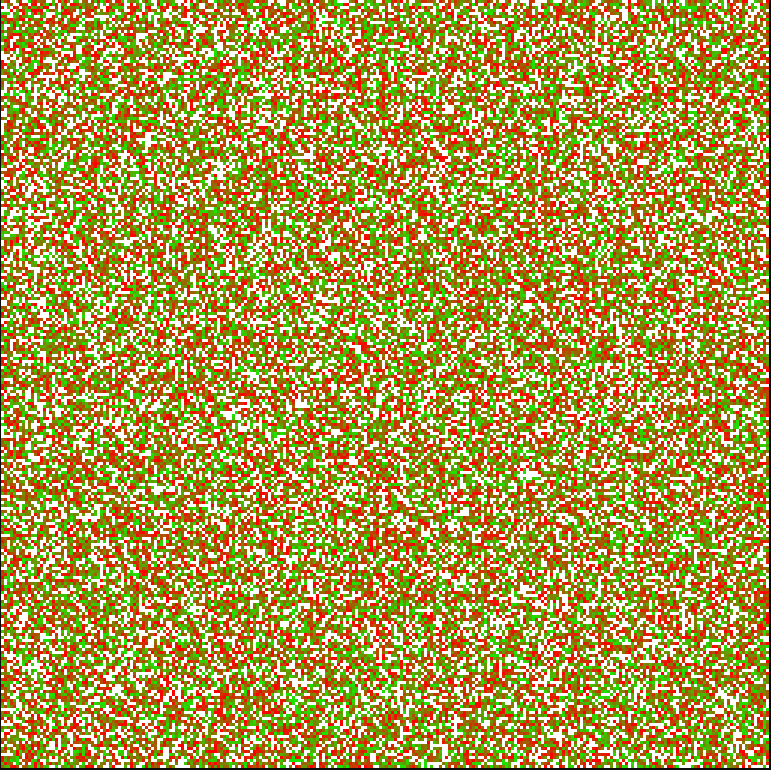


Рисунок . Начальное распределение клеточного автомата.

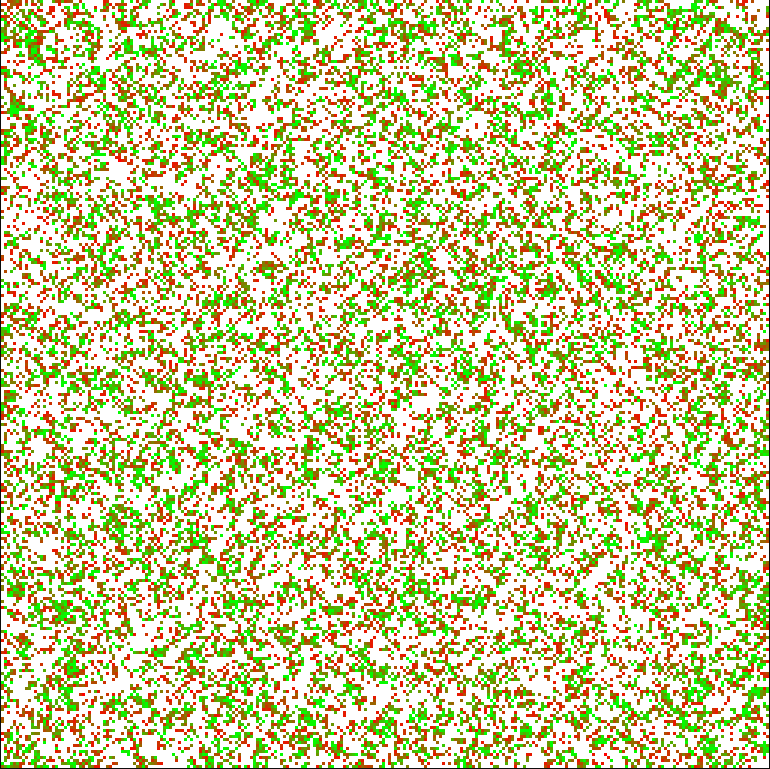


Рисунок 12. Состояние на 5 итерации. Живых организмов 38868

Такое резкое увеличение количества живых организмов связано с созреванием и способностью плодиться.

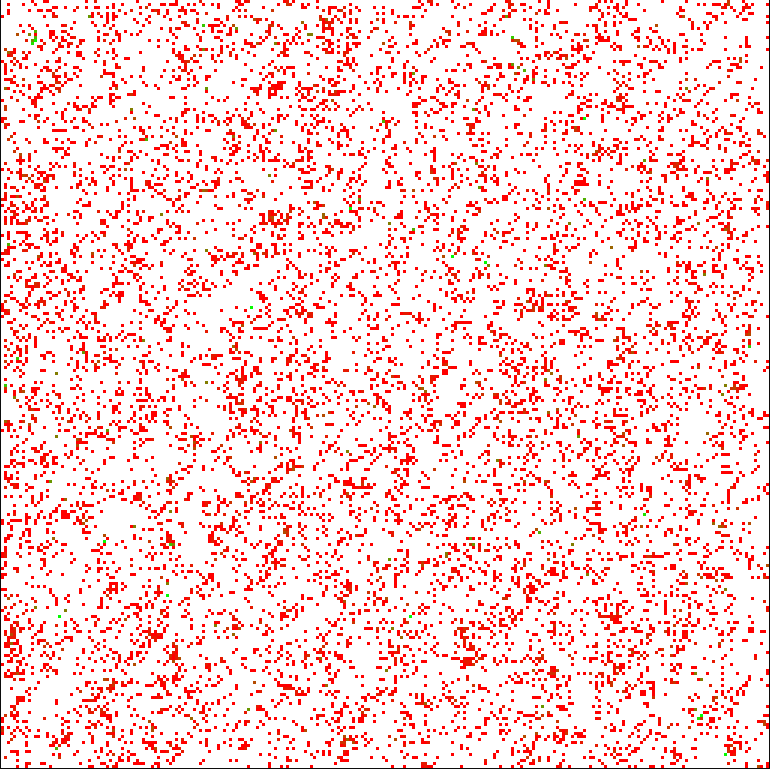


Рисунок 13. Состояние на 10-й итерации. Живых организмов 54745

Дальнейшее увеличение популяции обусловлено тем, что на 5 итерации не все клетки имели возможность оставить после себя потомка из-за необходимости «прыгнуть» на свободную клетку. Так же из-за недостаточной энергии, необходимой для размножения.

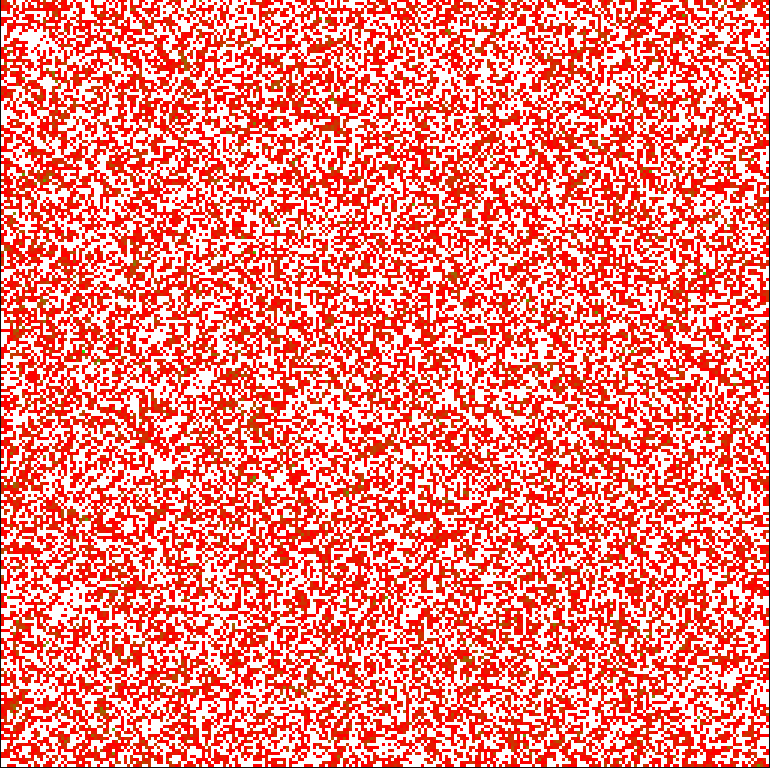


Рисунок 14. Состояние на 13-й итерации. Живых 30751

Уменьшение особей связано с нехваткой ресурсов.

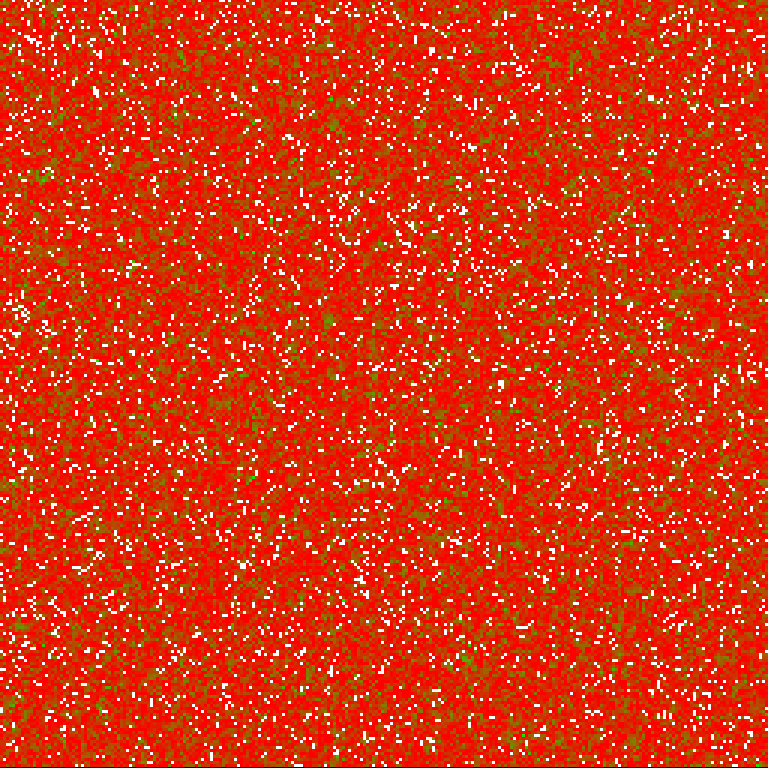


Рисунок 15. Состояние на 16-й итерации. Живых 4237

Резкое уменьшение связано с продолжительностью жизни особей в 15 тактов.

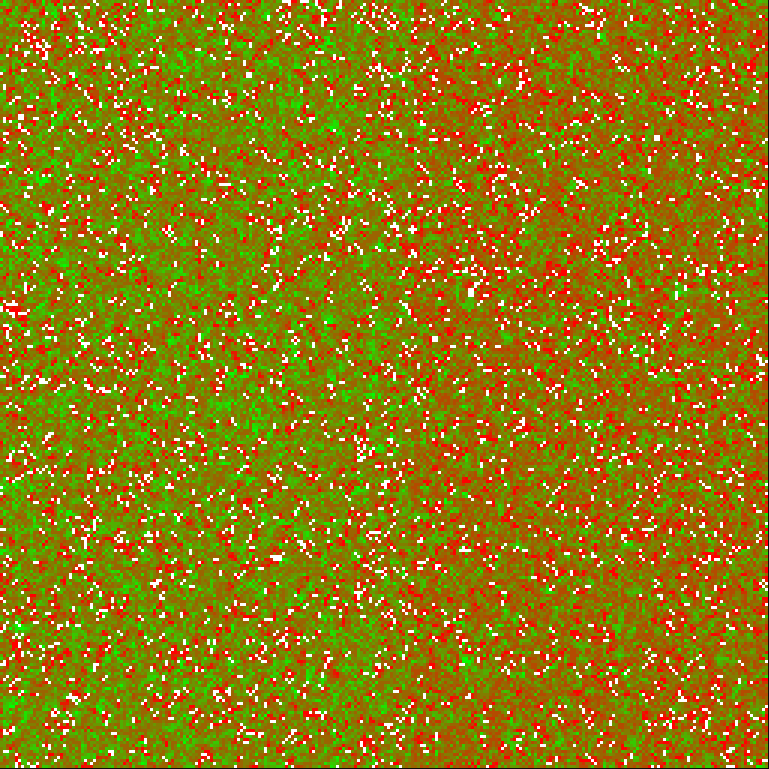


Рисунок 16. Состояние на 20-й итерации. Живых 3995

Заметно обогащение среды питательности, так же заметно, что оно происходит только с одной стороны, это связано с тем, что суммарная питательность всех клеток не может быть более .

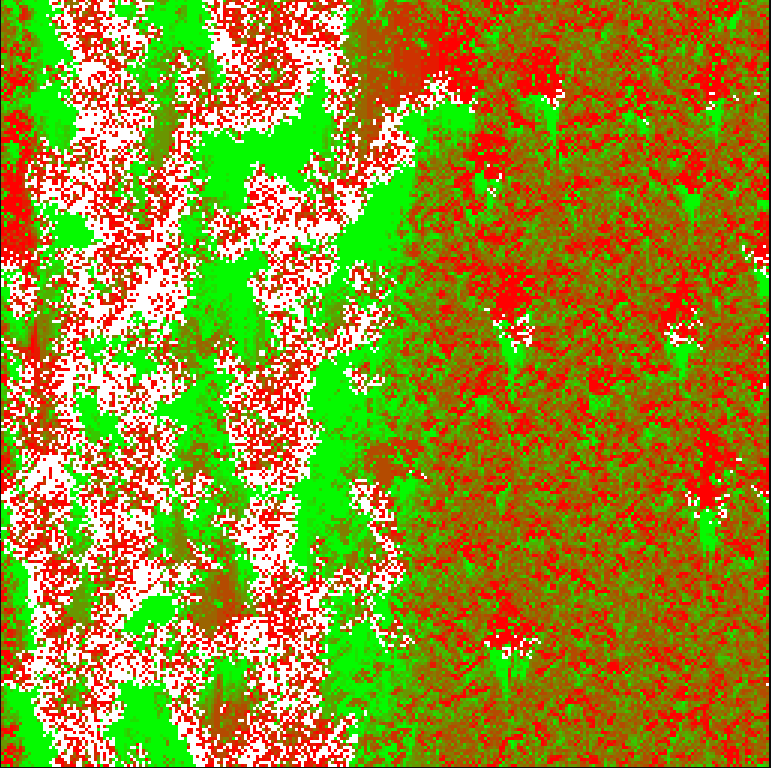


Рисунок 17. Состояние 71 итерации. Живых 11328.

Наблюдаются явление самоорганизации.

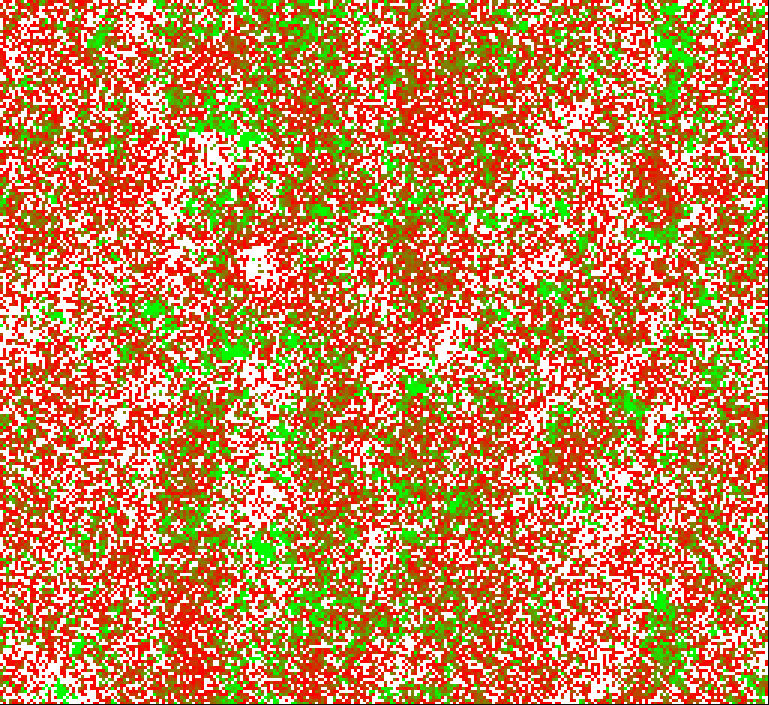


Рисунок 18. Состояние 155-й итерации. Живых 21519.

Клеточный автомат переходит в равновесное состояние.

**Вывод**.

В ходе лабораторной работы был получен навык моделирования имитации сложных динамических систем с использованием клеточных автоматов на примере моделей биологических систем.

Были найдены статические и циклические структуры для клеточного автомата жизнь.

В реализации клеточного автомата «Нейронная сеть» были получены следующие результаты:

* От периодического источника исходят круговые волны.
* При столкновении параллельных фронтов происходит гашение друг о друга.

Был реализован и проанализирован клеточный автомат «Организмы-питательная среда». При моделировании было выяснено, что без конкуренции между особями в клеточном автомате возникает явление самоорганизации: волна распространяется плоским фронтом.

**Приложение**

**Листинг 1**

#include "stdio.h"

#include "iostream"

#include <set>

#include <random>

#include <Windows.h>

const int N = 6; // Размерность

const unsigned int pixelScale = 50;//масштаб

const float A = 0.07;

inline int Tor(int x)

{

if (x < 0)

return x + N;

else

return x % N;

}

int f(int y, int yU, int yUR, int yR, int yDR, int yD, int yDL, int yL, int yUL)

{

int i = yU + yUR + yR + yDR + yD + yDL + yL + yUL;

if ((y == 0 && i == 2) || (y == 1 && (i <= 2)))

{

return 1;

}

return 0;

}

int main(int argc, char\* argv[])

{

//// Массив для хранения текущих состояний клеток

HWND myconsole = GetConsoleWindow(); // hwnd - Дескриптор - уникальный номер экземпляра окна программы(каждая программа при запуска получает его от windows автоматически), для того чтобы windows могла их различать

//Получите дескриптор контекста устройства

HDC mydc = GetDC(myconsole);

std::uniform\_int\_distribution<int> uid(0, N \* N); // Формирует равномерное (каждое значение одинаково вероятно) распределение целых чисел в выходном инклюзивно-эксклюзивном диапазоне.

std::random\_device device; // Формирует случайную последовательность с помощью внешнего устройства.

std::mt19937\_64 d(device()); // std::mt19937 - это генератор псевдослучайных чисел, для получения последовательности чисел его надо создать один раз, и потом использовать его вместе с каким-то случайным распределением чтобы получать случайные числа.

std::set<int> R; // Set – это множество, которое содержит несколько отсортированных элементов. При добавлении нового элемента в множество он сразу становится на свое место так, чтобы не нарушать порядка сортировки.

int cell[N][N];

int i = 0;

int cell\_temp[N][N];//Массив для хранения новых состояний клеток

char c;// Переменная,используемая при обработке нажатия кнопок

long iter = 0;// Счетчик итераций

for (int i = 0; i < N; i++)

for (int j = 0; j < N; j++)

{

cell[i][j] = 0;

}

//1!

do

{

auto tmp = uid(d); // auto - автоматическое определение типа данных компилятором

R.insert(tmp); // вставка

} while (R.size() != (int)(A \* N \* N));

// Начальное заполнение массива клеток

for (int i = 0; i < N; i++)

for (int j = 0; j < N; j++)

{

if (R.find(i \* N + j) != R.end()) // find - находит элемент с определенным ключом

{

cell[i][j] = 1;

}

}

// Итерации автомата

for (;;)

{

// Визуализация

int alive = 0;

for (i = 0; i < N; i++)

{

for (int j = 0; j < N; j++)

{

if (cell[i][j] == 1)

{

COLORREF COLOR1 = RGB(255, 255, 255);

for (size\_t k = i \* pixelScale + 250; k < i \* pixelScale + pixelScale + 250; k++)

{

for (size\_t l = j \* pixelScale; l < j \* pixelScale + pixelScale; l++)

{

SetPixel(mydc, k, l, COLOR1);

}

}

alive++;

}

if (cell[i][j] == 0)

{

COLORREF COLOR2 = RGB(128, 0, 0);

for (size\_t k = i \* pixelScale + 250; k < i \* pixelScale + pixelScale + 250; k++)

{

for (size\_t l = j \* pixelScale; l < j \* pixelScale + pixelScale; l++)

{

SetPixel(mydc, k, l, COLOR2);

}

}

}

}

}

std::cout << "Iter - " << iter << "; Alive - " << alive;

//Вычисление новых состояний клеток

for (i = 0; i < N; i++)

for (int j = 0; j < N; j++)

{

cell\_temp[i][j] = f(cell[i][j],

cell[Tor(i - 1)][j], cell[Tor(i - 1)][Tor(j + 1)], cell[i][Tor(j + 1)], cell[Tor(i + 1)][Tor(j + 1)],

cell[Tor(i + 1)][j], cell[Tor(i + 1)][Tor(j - 1)], cell[i][Tor(j - 1)], cell[Tor(i - 1)][Tor(j - 1)]);

}

//Перенос новых состояний в массив текущих состояний

for (i = 0; i < N; i++)

for (int j = 0; j < N; j++)

cell[i][j] = cell\_temp[i][j];

//Обработка нажатия кнопки <q>

c = getchar();

if (c == 'q') return 0;

iter++;

}

return 0;

}

**Листинг 2**

#include <Windows.h>

#include <iostream>

const int N = 256;

const unsigned int pixelScale = 3;//масштаб пикселей

const float A = 0.3;

const float P = 3;

const int T = 5;

const int B = 8;

inline int Tor(int x)

{

if (x < 0) return x + N;

else return x % N;

}

struct cell

{

int sost;

float level\_active;

int time\_takt;

};

cell y[N][N];

cell Y1[N][N];

cell AmbitCell(cell y, cell yU, cell yUR, cell yR, cell yDR, cell yD, cell yDL, cell yL, cell yUL, int time)

{

if (y.sost == 0)

{

y.level\_active = y.level\_active \* A;

float akt = y.level\_active + yU.level\_active + yUR.level\_active + yR.level\_active + yDR.level\_active + yD.level\_active + yDL.level\_active + yL.level\_active + yUL.level\_active;

if (akt >= P)

{

y.sost = 2;

y.time\_takt = time + T;

y.level\_active = 1.0;

}

}

if (y.sost == 1)

{

y.level\_active = y.level\_active \* A;

if (y.time\_takt == time)

{

y.sost--;

y.time\_takt = 0;

}

}

if (y.sost == 2)

{

y.level\_active = 1.0;

if (y.time\_takt == time)

{

y.sost--;

y.time\_takt = time + B;

}

}

return y;

}

int main(int argc, char\* argv[])

{

HWND myconsole = GetConsoleWindow();

RECT r;

GetWindowRect(myconsole, &r);

MoveWindow(myconsole, r.left, r.top, 1500, 900, TRUE);

HDC mydc = GetDC(myconsole);

int i = 0;

char c;

long iter = 0;

int flag = 0;

// Начальноезаполнениемассиваклеток

for (int i = 0; i < N; i++)

for (int j = 0; j < N; j++)

{

y[i][j].sost = 0;

y[i][j].level\_active = 0;

y[i][j].time\_takt = -1;

}

// плоский фронт

for (int i = N \* 0.25; i < N \* 0.25 + 1; i++)

for (int j = 0; j < N; j++)

{

y[i][j].sost = 2;

y[i][j].level\_active = 1.0;

y[i][j].time\_takt = T;

}

// Итерации автомата

for (;;)

{

//3\*3 источник

if (flag == 0)

{

for (int i = N / 2 - 1; i < N / 2 + 2; i++)

{

for (int j = N / 2 - 1; j < N / 2 + 2; j++)

{

y[i][j].sost = 2;

y[i][j].level\_active = 1.0;

y[i][j].time\_takt = iter + T;

}

}

}

flag++;

if (flag == 15)

flag = 0;

// Визуализация

int alive = 0;

for (i = 0; i < N; i++)

{

for (int j = 0; j < N; j++)

{

if (y[i][j].sost == 2)

{

COLORREF COLOR1 = RGB(255, 255, 255);

for (size\_t k = i \* pixelScale + 250; k < i \* pixelScale + pixelScale + 250; k++)

{

for (size\_t l = j \* pixelScale; l < j \* pixelScale + pixelScale; l++)

{

SetPixel(mydc, k, l, COLOR1);

}

}

alive++;

}

if (y[i][j].sost == 1)

{

COLORREF COLOR1 = RGB(0, 0, 255);

for (size\_t k = i \* pixelScale + 250; k < i \* pixelScale + pixelScale + 250; k++)

{

for (size\_t l = j \* pixelScale; l < j \* pixelScale + pixelScale; l++)

{

SetPixel(mydc, k, l, COLOR1);

}

}

}

if (y[i][j].sost == 0)

{

COLORREF COLOR2 = RGB(128, 0, 0);

for (size\_t k = i \* pixelScale + 250; k < i \* pixelScale + pixelScale + 250; k++)

{

for (size\_t l = j \* pixelScale; l < j \* pixelScale + pixelScale; l++)

{

SetPixel(mydc, k, l, COLOR2);

}

}

}

}

}

std::cout << "Iter - " << iter << "; Alive - " << alive;

//Вычисление новых состояний клеток

for (i = 0; i < N; i++)

for (int j = 0; j < N; j++)

{

Y1[i][j] = AmbitCell(y[i][j], y[Tor(i - 1)][Tor(j)],

y[Tor(i - 1)][Tor(j + 1)], y[Tor(i)][Tor(j + 1)],

y[Tor(i + 1)][Tor(j + 1)], y[Tor(i + 1)][Tor(j)],

y[Tor(i + 1)][Tor(j - 1)], y[Tor(i)][Tor(j - 1)],

y[Tor(i - 1)][Tor(j - 1)], iter);

}

for (i = 0; i < N; i++)

for (int j = 0; j < N; j++)

y[i][j] = Y1[i][j];

c = getchar();

if (c == 'q') return 0;

iter++;

}

return 0;

}

**Листинг 3**

#include <time.h>

#include <cstdlib>

#include <set>

#include <random>

#include <iostream>

#include <Windows.h>

const int N = 256;

const unsigned int pixelScale = 3;// масштаб пикселей

float MaxP = 0; // максимальная питательность среды

float NowP = 0; // текущая питательность среды

const float Pmax = 10;

const float r = 1;

const float A = 0.3;

const int L = 15;

const int T = 3;

const float dp = 5;

const float p1 = 35;

const float de = 2;

const float dr = 3;

inline int Tor(int x)

{

if (x < 0) return x + N;

else return x % N;

}

struct cell

{

float P = 0.0;

int sost = 0;

int nowBirth = 0;

int nextBirth = 0;

int jump = 0;

float nowEnergy = 0.0;

float nextEnergy = .0;

};

cell y[N][N];

cell Y1[N][N];

float Min(float a, float b)

{

if (a > b)

return b;

else

return a;

}

cell f(cell y, cell yU, cell yUR, cell yR, cell yDR, cell yD, cell yDL, cell yL, cell yUL, int time)

{

if ((NowP < MaxP) && (y.P < Pmax))

{

float buff = Min(Min(Pmax - y.P, MaxP - y.P), r);

y.P += buff;

NowP += buff;

}

if (y.sost == 1)

{

//7+8

//изменение энергии

if (y.nowEnergy <= (p1 - dp))

{

//Запас среды >=прибавки

if (y.P >= dp)

{

y.P -= dp;

NowP -= dp;

y.nowEnergy += dp;

}

else

{

y.nowEnergy += y.P;

NowP -= y.P;

y.P = 0;

}

}

else if (y.nowEnergy < p1)

{

if (y.P >= (p1 - y.nowEnergy))

{

y.P -= (p1 - y.nowEnergy);

NowP -= (p1 - y.nowEnergy);

y.nowEnergy = p1;

}

else

{

y.nowEnergy += y.P;

NowP -= y.P;

y.P = 0;

}

}

//9+11+12

if ((y.nowEnergy <= de) || ((time - y.nowBirth) >= L))

{

y.sost = 0;

y.nowEnergy = 0;

y.nowBirth = 0;

}

else { y.nowEnergy -= de; }

}

return y;

}

int main(int argc, char\* argv[])

{

HWND myconsole = GetConsoleWindow();

RECT r;

GetWindowRect(myconsole, &r);

MoveWindow(myconsole, r.left, r.top, 1500, 900, TRUE);

HDC mydc = GetDC(myconsole);

std::uniform\_int\_distribution<int> uid(0, N \* N);

std::random\_device device;

std::mt19937\_64 d(device());

srand(time(NULL));

std::set<int> R;

int i = 0;

char c;

// Счетчикитераций

long iter = 0;

//1!

do

{

auto tmp = uid(d);

R.insert(tmp);

} while (R.size() != (int)(A \* N \* N));

// Начальноезаполнениемассиваклеток

for (int i = 0; i < N; i++)

for (int j = 0; j < N; j++)

{

if (R.find(i \* N + j) != R.end())

{

y[i][j].sost = 1;

y[i][j].nowEnergy = rand() % int(p1);

}

y[i][j].P = rand() % int(Pmax);

MaxP += y[i][j].P;

y[i][j].nowBirth = 0;

y[i][j].jump = 0;

}

NowP = MaxP;

// Итерации автомата

for (;;)

{

// Визуализация

int alive = 0;

for (i = 0; i < N; i++)

{

for (int j = 0; j < N; j++)

{

if (y[i][j].sost == 1)

{

COLORREF COLOR1 = RGB(255, 255, 255);

for (size\_t k = i \* pixelScale + 250; k < i \* pixelScale + pixelScale + 250; k++)

{

for (size\_t l = j \* pixelScale; l < j \* pixelScale + pixelScale; l++)

{

SetPixel(mydc, k, l, COLOR1);

}

}

alive++;

}

if (y[i][j].sost == 0)

{

COLORREF COLOR2 = RGB(255 - (25 \* y[i][j].P), 25 \* y[i][j].P, 0);

for (size\_t k = i \* pixelScale + 250; k < i \* pixelScale + pixelScale + 250; k++)

{

for (size\_t l = j \* pixelScale; l < j \* pixelScale + pixelScale; l++)

{

SetPixel(mydc, k, l, COLOR2);

}

}

}

}

}

std::cout << "Iter - " << iter << "; Alive - " << alive;

//Вычисление новых состояний клеток

for (i = 0; i < N; i++) for (int j = 0; j < N; j++)

{

Y1[i][j] = f(y[i][j], y[Tor(i - 1)][Tor(j)],

y[Tor(i - 1)][Tor(j + 1)], y[Tor(i)][Tor(j + 1)],

y[Tor(i + 1)][Tor(j + 1)], y[Tor(i + 1)][Tor(j)],

y[Tor(i + 1)][Tor(j - 1)], y[Tor(i)][Tor(j - 1)],

y[Tor(i - 1)][Tor(j - 1)], iter);

}

//4!

for (i = 0; i < N; i++)

{

for (int j = 0; j < N; j++)

{

if (Y1[i][j].sost == 1)

{

int local\_x, local\_y;

bool flag\_equal = false;

std::set<int> S;

do

{

if (S.size() == 8)

{

local\_x = i;

local\_y = j;

flag\_equal = true;

Y1[local\_x][local\_y].jump = 1;

Y1[local\_x][local\_y].nextEnergy = Y1[i][j].nowEnergy;

Y1[local\_x][local\_y].nextBirth = Y1[local\_x][local\_y].nowBirth;

break;

} //Выход если некуда прыгать

auto direction = rand() % 8;

S.insert(direction);

switch (direction)

{

case 0:

{

local\_x = i;

local\_y = Tor(j - 1);

break;

}

case 1:

{

local\_x = Tor(i + 1);

local\_y = Tor(j - 1);

break;

}

case 2:

{

local\_x = Tor(i + 1);

local\_y = j;

break;

}

case 3:

{

local\_x = Tor(i + 1);

local\_y = Tor(j + 1);

break;

}

case 4:

{

local\_x = i;

local\_y = Tor(j + 1);

break;

}

case 5:

{

local\_x = Tor(i - 1);

local\_y = Tor(j + 1);

break;

}

case 6:

{

local\_x = Tor(i - 1);

local\_y = j;

break;

}

case 7:

{

local\_x = Tor(i - 1);

local\_y = Tor(j - 1);

break;

}

}

} while (Y1[local\_x][local\_y].jump != 0 || Y1[local\_x][local\_y].sost != 0);

if (!flag\_equal)

{

Y1[local\_x][local\_y].jump = 1;

Y1[local\_x][local\_y].nextBirth = Y1[i][j].nowBirth;

if (((iter - Y1[i][j].nowBirth) > T) && (Y1[i][j].nowEnergy > dr))

{

Y1[local\_x][local\_y].nextEnergy = Y1[i][j].nowEnergy - dr;

Y1[i][j].jump = 1;

Y1[i][j].nextEnergy = 0;

Y1[i][j].nextBirth = iter;

}

else

{

Y1[local\_x][local\_y].nextEnergy = Y1[i][j].nowEnergy;

Y1[i][j].sost = 0;

Y1[i][j].jump = 0;

Y1[i][j].nowEnergy = 0;

Y1[i][j].nowBirth = 0;

}

}

}

}

}

//5,6!

for (i = 0; i < N; i++)

{

for (int j = 0; j < N; j++)

{

if (Y1[i][j].jump == 1)

{

Y1[i][j].sost = 1;

Y1[i][j].jump = 0;

Y1[i][j].nowEnergy = Y1[i][j].nextEnergy;

Y1[i][j].nowBirth = Y1[i][j].nextBirth;

}

else

{

Y1[i][j].sost = 0;

Y1[i][j].nowEnergy = Y1[i][j].nextEnergy = 0;

Y1[i][j].nowBirth = Y1[i][j].nextBirth = 0;

}

}

}

for (i = 0; i < N; i++)

for (int j = 0; j < N; j++)

y[i][j] = Y1[i][j];

c = getchar();

if (c == 'q') return 0;

iter++;

}

return 0;

}