

**Специальность: «информационные системы и программирование 09.02.07»**

**Подготовил Колтовой Илья Максимович 4/11ИП**

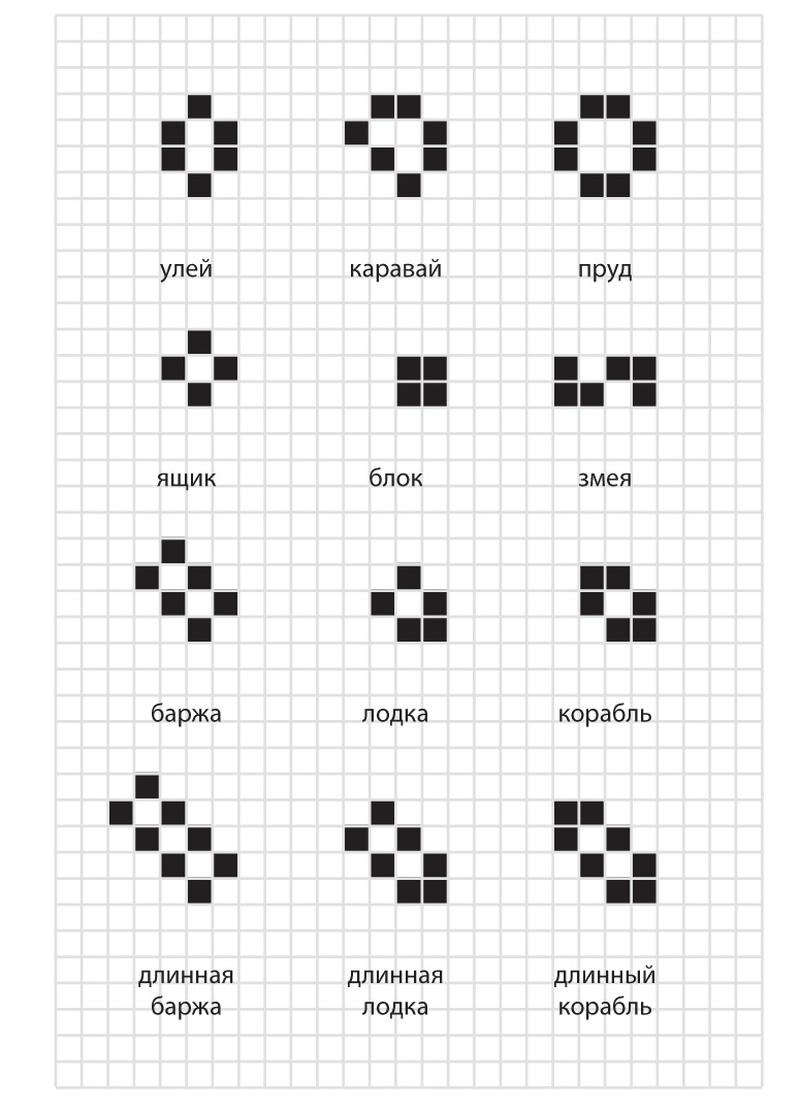
1. **Трактовка задачи**

**Задача № 1 «Игра «Жизнь»**

**Реализовать приложение по следующим правилам:**

Правила игры «жизнь»

1. Место действия этой игры — «вселенная» — это размеченная на клетки поверхность или плоскость — безграничная, ограниченная, или замкнутая (в пределе — бесконечная плоскость).
2. Каждая клетка на этой поверхности может находиться в двух состояниях: быть «живой» (заполненной) или быть «мёртвой» (пустой). Клетка имеет восемь соседей, окружающих её.
3. Распределение живых клеток в начале игры называется первым поколением. Каждое следующее поколение рассчитывается на основе предыдущего по таким правилам:
   1. в пустой (мёртвой) клетке, рядом с которой ровно три живые клетки, зарождается жизнь;
   2. если у живой клетки есть две или три живые соседки, то эта клетка продолжает жить; в противном случае, если соседей меньше двух или больше трёх, клетка умирает («от одиночества» или «от перенаселённости»).
4. Игра прекращается, если:
   1. на поле не останется ни одной «живой» клетки;
   2. конфигурация на очередном шаге в точности (без сдвигов и поворотов) повторит себя же на одном из более ранних шагов (складывается периодическая конфигурация);
   3. при очередном шаге ни одна из клеток не меняет своего состояния (складывается стабильная конфигурация; предыдущее правило, вырожденное до одного шага назад).



Предусмотреть заполнение клеток поля живыми клетками двух цветов (красный и синий) Красные считают белые и синие клетки свободными.

Синие считают белые и красные клетки свободными соответственно.

1. **Описание функций программы**
2. **Класс TheGameOfLife ()**

В классе инициализируются компоненты программы.

1. **Функция NeighborBlue ()**

Функция NeighborBlue () описывает логику поведения синих клеток, как по отношению к красным, так и условия их жизни.

1. **Функция NeighborRed ()**

Функция NeighborRed ()описывает логику поведения красных клеток, как по отношению к синим, так и условия их жизни.

1. **Метод LifeTick ()**

Метод отвечает за обновление поля каждый цикл таймера. Так же он отчищает поля, и создает новый массив, в котором будут содержаться живые клетки. Так же в методе содержаться циклы, которые отвечают клетки двух цветов. В том случае, если "рождается" новая клетка, то срабатывает переключатель для newGegeration, позволяющий увеличить отображаемый счетчик поколений на 1. После чего массив field обновляется данными из newField, происходит обновление LifeBox.

1. **Класс Start\_Click ()**

Кнопка «старт», включает в себя проверку на то, включен ли таймер, если таймер включен, то ничего сделать нельзя, если же таймер выключен, то на игровом поле в случайном порядке расставятся клетки красного цвета (это нужно для того, чтобы просто начать игру). Так же запуститься счетчик поколений клеток и запуститься таймер.

1. **Класс Stop\_Click ()**

Кнопка остановки игры, при нажатии на кнопку, все созданные клетки останавливаются (не происходит размножение и смерть клеток), так же останавливается таймер. В этом режиме программы, мы можем менять значения разрешения и плотности самих клеток.

1. **Функция MouseMove ()**

В этом классе описывается то, какими кнопками мы можем добавлять новые клетки, причем обоих цветов (красного или синего), так правой кнопкой мыши, мы можем добавить новы клетки, а левой кнопкой удалить клетки, причем как созданные нами, так созданные в случайном порядке.

1. **Функция ValidateMousePosition()**

Валидатор, который не дает программе выдать ошибку, при попытке пользователя вывести рисование клеток за границы игрового поля.

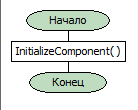
1. **Класс RedClick ()**

Кнопка смена цвета клеток, которые хочет нарисовать пользователь, при нажатии на кнопку, цвет клетки поменяется на красный. Причем клетки нельзя нарисовать пока не нажата кнопка.

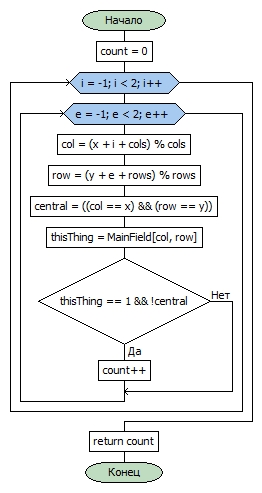
1. **Класс BlueClick ()**

Кнопка смена цвета клеток, которые хочет нарисовать пользователь, при нажатии на кнопку, цвет клетки поменяется на синий. Причем клетки нельзя нарисовать пока не нажата кнопка.

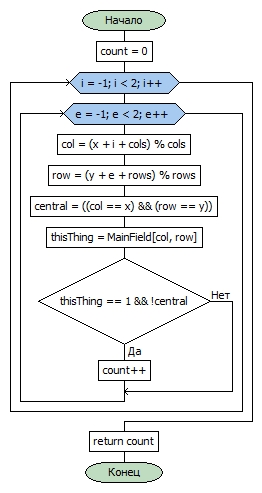
1. **Блок-схемы программы**
2. **Блок-схема класса TheGameOfLife ()**

****

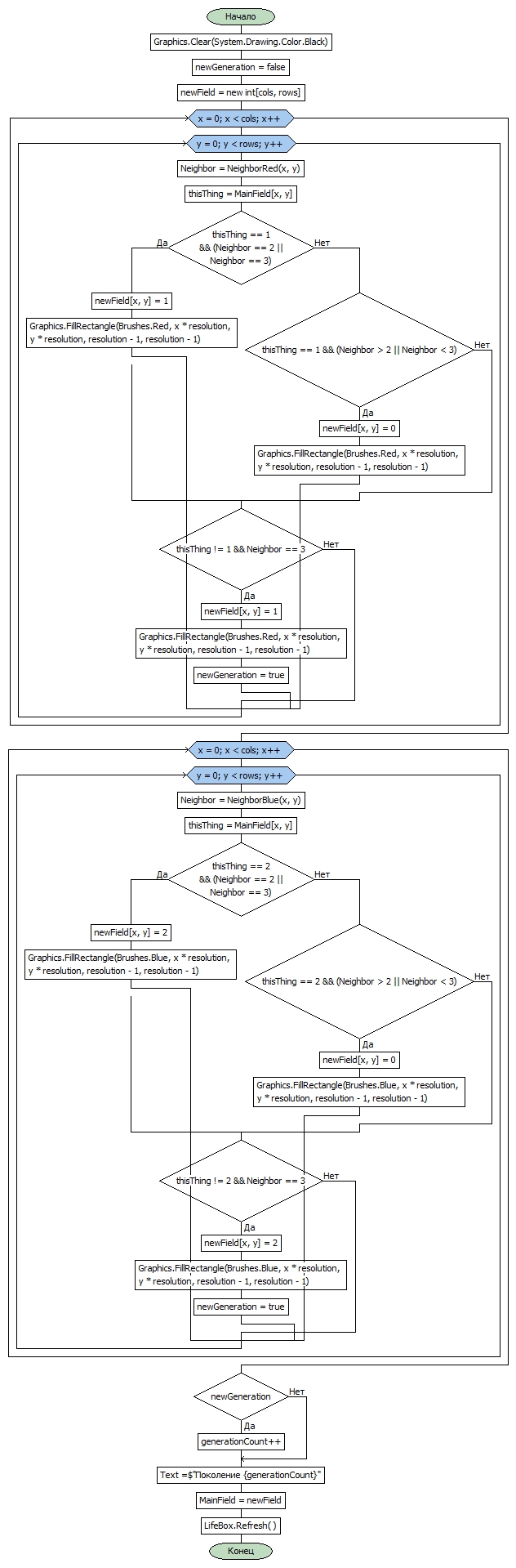
1. **Блок-схема функции NeighborBlue ()**

****

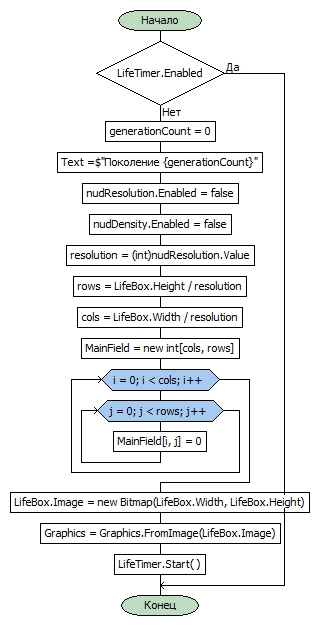
1. **Блок-схема функции NeighborRed ()**

****

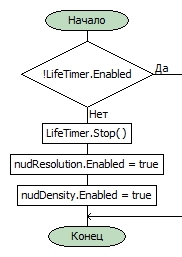
1. **Блок-схема класс LifeTick ()**

****

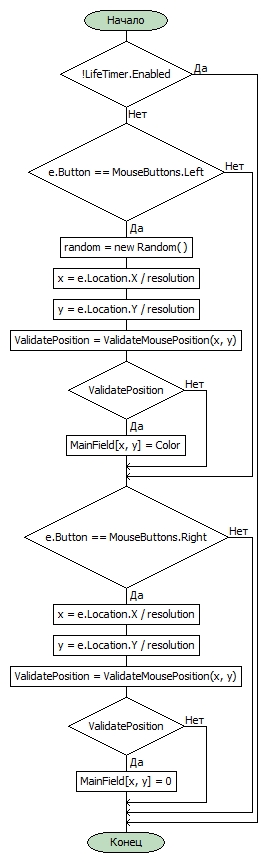
1. **Блок-схема класса Start\_Click ()**

****

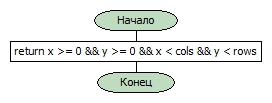
1. **Блок-схема класса Stop\_Click ()**

****

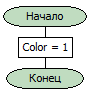
1. **Блок-схема функции MouseMove ()**

****

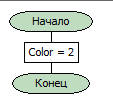
1. **Блок-схема функции ValidateMousePosition ()**

****

1. **Блок-схема класса RedClick ()**

****

1. **Блок-схема BlueClick ()**

****

1. **Код программы**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Drawing;

namespace Smertiorlife

{

public partial class TheGameOfLife : Form

{

private int generationCount = 0;

private Graphics Graphics;

private int resolution;

private int rows;

private int cols;

private int[,] MainField; //Принимает значения от 0 до 2

int Color = 1;

public TheGameOfLife()

{

InitializeComponent();

}

/// <summary>

/// Функции NeighborRed и NeighborBlue высчитывают количество количество соседних

/// клеток одного и того же цвета, принимая в входные параметры координаты челевой

/// клетки, возвращая количество соседей.

/// </summary>

private int NeighborRed(int x, int y)

{

int count = 0;// Счетчик соседей

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

{

for (int e = -1; e < 2; e++)

{

var col = (x + i + cols) % cols;

var row = (y + e + rows) % rows;

var central = ((col == x) && (row == y));

var thisThing = MainField[col, row];

if (thisThing == 1 && !central)

{

count++;

}

}

}

return count;

}

private int NeighborBlue(int x, int y)

{

int count = 0;

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

{

for (int e = -1; e < 2; e++)

{

var col = (x + i + cols) % cols;

var row = (y + e + rows) % rows;

var central = ((col == x) && (row == y));

var thisThing = MainField[col, row];

if (thisThing == 2 && !central)

{

count++;

}

}

}

return count;

}

private void LifeTick(object sender, EventArgs e)

{

Graphics.Clear(System.Drawing.Color.Black);

bool newGeneration = false;// переключатель, отображающий появление хотя бы одного поколения

var newField = new int[cols, rows]; // новое поле для положения клеток

for (int x = 0; x < cols; x++)

{

for (int y = 0; y < rows; y++)

{

var Neighbor = NeighborRed(x, y);

var thisThing = MainField[x, y];

if (thisThing == 1 && (Neighbor == 2 || Neighbor == 3))// если клетка живая и имеет 3 или 3 соседа

{

newField[x, y] = 1;

Graphics.FillRectangle(Brushes.Red, x \* resolution, y \* resolution, resolution - 1, resolution - 1);

// клетка остается

continue;

}

else if (thisThing == 1 && (Neighbor > 2 || Neighbor < 3))

{ // если меньше 2 или больше 3 соседей

newField[x, y] = 0;

Graphics.FillRectangle(Brushes.Red, x \* resolution, y \* resolution, resolution - 1, resolution - 1);

// клетка становится церного цвета

continue;

}

if (thisThing != 1 && Neighbor == 3)

{// если клетка не занята, и имеет 3 соседа, она становится занятой

newField[x, y] = 1;

Graphics.FillRectangle(Brushes.Red, x \* resolution, y \* resolution, resolution - 1, resolution - 1);

newGeneration = true; // это новое поколение

continue;

}

}

}

for (int x = 0; x < cols; x++)

{// тот же самый цикл, только для синих клеток с той же логикой

for (int y = 0; y < rows; y++)

{

var Neighbor = NeighborBlue(x, y);

var thisThing = MainField[x, y];

if (thisThing == 2 && (Neighbor == 2 || Neighbor == 3))

{

newField[x, y] = 2;

Graphics.FillRectangle(Brushes.Blue, x \* resolution, y \* resolution, resolution - 1, resolution - 1);

continue;

}

else if (thisThing == 2 && (Neighbor > 2 || Neighbor < 3))

{

newField[x, y] = 0;

Graphics.FillRectangle(Brushes.Blue, x \* resolution, y \* resolution, resolution - 1, resolution - 1);

continue;

}

if (thisThing != 2 && Neighbor == 3)

{

newField[x, y] = 2;

Graphics.FillRectangle(Brushes.Blue, x \* resolution, y \* resolution, resolution - 1, resolution - 1);

newGeneration = true;

continue;

}

}

}

if (newGeneration)// если есть хотя бы одна новая клетка

{

generationCount++;// увеличение счетчика

}

Text = $"Поколение {generationCount}";// отображение поколений

MainField = newField;// обновление поля

LifeBox.Refresh();// обновление графики

}

/// <summary>

/// кнопка старт, запускающая циклы

/// </summary>>

private void Start\_Click(object sender, EventArgs e)

{

if (LifeTimer.Enabled)

return;//защита от повторного нажатия

generationCount = 0;// счетчик поколений тановится равен 0

Text = $"Поколение {generationCount}";

nudResolution.Enabled = false;

nudDensity.Enabled = false;

resolution = (int)nudResolution.Value;

rows = LifeBox.Height / resolution;

cols = LifeBox.Width / resolution;

MainField = new int[cols, rows];

for (int i = 0; i < cols; i++) // Заполнение при старте пустым пространством

{

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

{

MainField[i, j] = 0;

}

}

LifeBox.Image = new Bitmap(LifeBox.Width, LifeBox.Height);

Graphics = Graphics.FromImage(LifeBox.Image);

LifeTimer.Start(); // Запуск игры и обновление

}

private void Stop\_Click(object sender, EventArgs e)

{

if (!LifeTimer.Enabled)

return;

LifeTimer.Stop(); //остановка игры

nudResolution.Enabled = true;

nudDensity.Enabled = true;

}

private void MouseMove(object sender, MouseEventArgs e)

{

if (!LifeTimer.Enabled)

return;

if (e.Button == MouseButtons.Left) // рисование клеток

{

Random random = new Random();

var x = e.Location.X / resolution;

var y = e.Location.Y / resolution;

var ValidatePosition = ValidateMousePosition(x, y);

if (ValidatePosition)

MainField[x, y] = Color;

}

if (e.Button == MouseButtons.Right) // удаление клеток

{

var x = e.Location.X / resolution;

var y = e.Location.Y / resolution;

var ValidatePosition = ValidateMousePosition(x, y);

if (ValidatePosition)

MainField[x, y] = 0;

}

}

private bool ValidateMousePosition(int x, int y) // защита

{

return x >= 0 && y >= 0 && x < cols && y < rows;

}

private void RedClick(object sender, EventArgs e) // позволяет рисовать красными клетками

{

Color = 1; //

}

private void BlueClick(object sender, EventArgs e) // позволяет рисовать синими клетками

{

Color = 2;

}

}

}