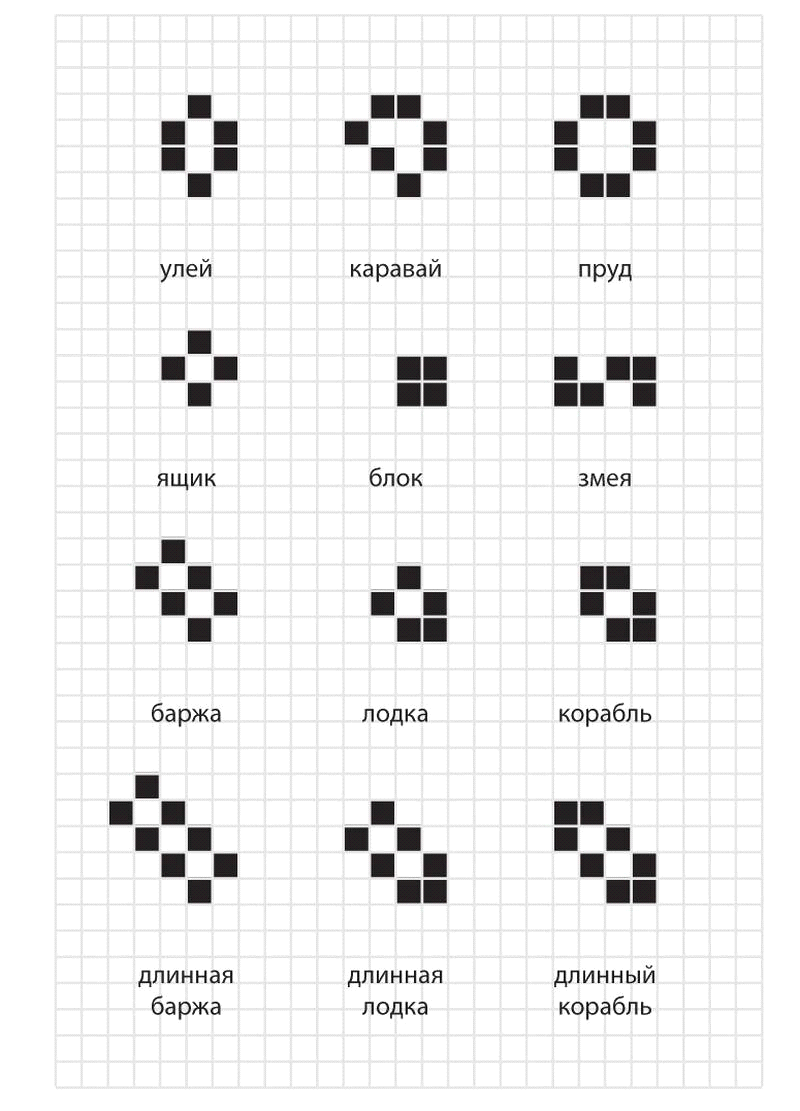


**Специальность : «Информационные системы и программирование 09.02.07»**

**Подготовил Грушин Антон Юрьевич 4/11ИП**

* Задача № 1 «Игра «Жизнь»
* Реализовать приложение по следующим правилам:
* Правила игры «жизнь»
* ⦁ Место действия этой игры — «вселенная» — это размеченная на клетки поверхность или плоскость — безграничная, ограниченная, или замкнутая (в пределе — бесконечная плоскость).
* ⦁ Каждая клетка на этой поверхности может находиться в двух состояниях: быть «живой» (заполненной) или быть «мёртвой» (пустой). Клетка имеет восемь соседей, окружающих её.
* ⦁ Распределение живых клеток в начале игры называется первым поколением. Каждое следующее поколение рассчитывается на основе предыдущего по таким правилам:
* ⦁ в пустой (мёртвой) клетке, рядом с которой ровно три живые клетки, зарождается жизнь;
* ⦁ 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»).
* ⦁ Игра прекращается, если:
* ⦁ на поле не останется ни одной «живой» клетки;
* ⦁ конфигурация на очередном шаге в точности (без сдвигов и поворотов) повторит себя же на одном из более ранних шагов (складывается периодическая конфигурация);
* ⦁ 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.

* **Описание функций программы**
* **Класс LIFE()**

В классе инициализируются компоненты программы.

* **Функция Blue()**

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

* **Функция Red()**

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

* **Метод timer1\_Tick()**

Метод отвечает за обновление поля каждый цикл таймера. Так же он отчищает поля , и создает новый массив Ю 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 pictureBox1.

* **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 что бы просто начать игру). Так же запуститься счетчик поколений клеток и запуститься таймер.

* **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.

* **Функция pictureBox1\_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.

* **Функция MousePosition()**

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

* **Класс bred\_Click()**

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.

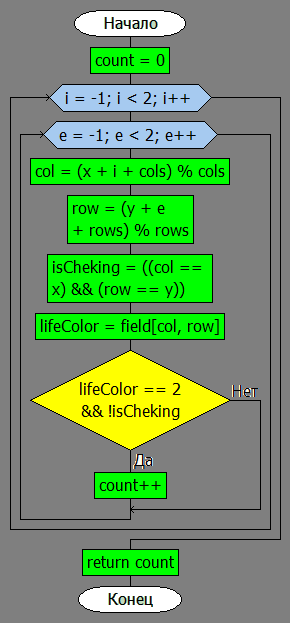
* **Класс bblue\_Click()**

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.

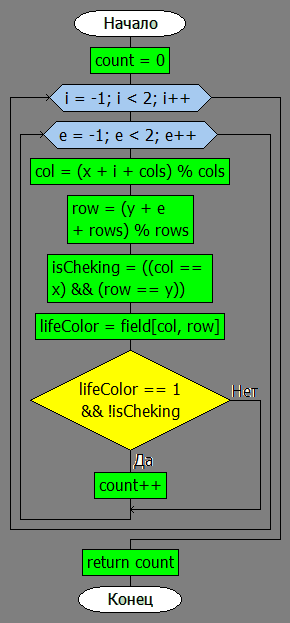
* **Блок-схемы программы**
* **Блок-схема класса LIFE()**



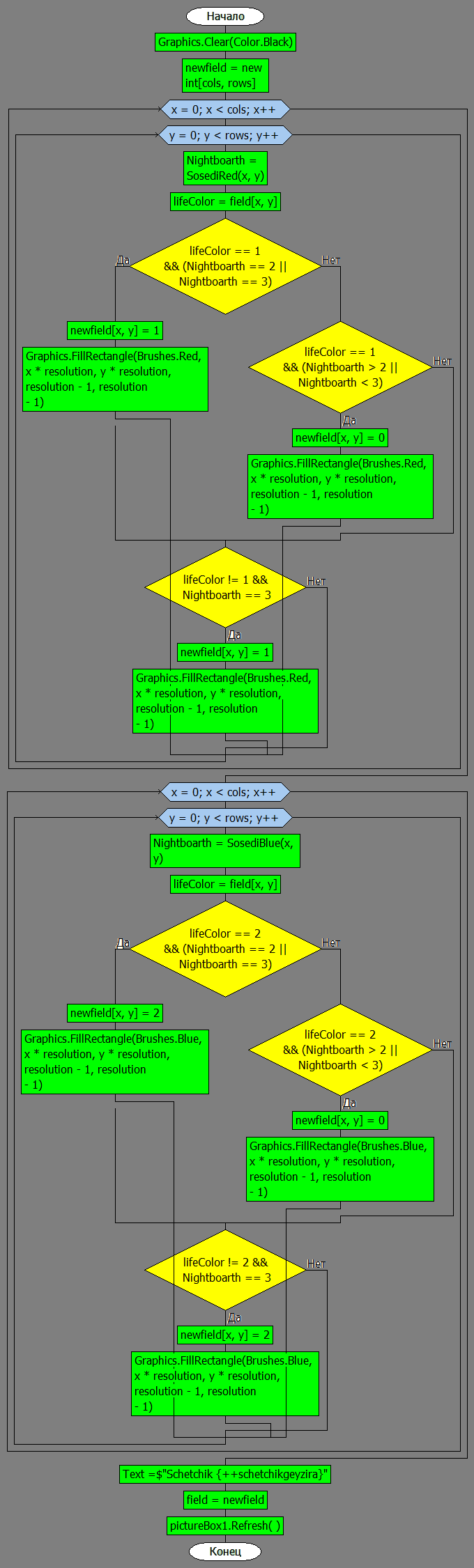
* **Блок-схема функции Blue()**



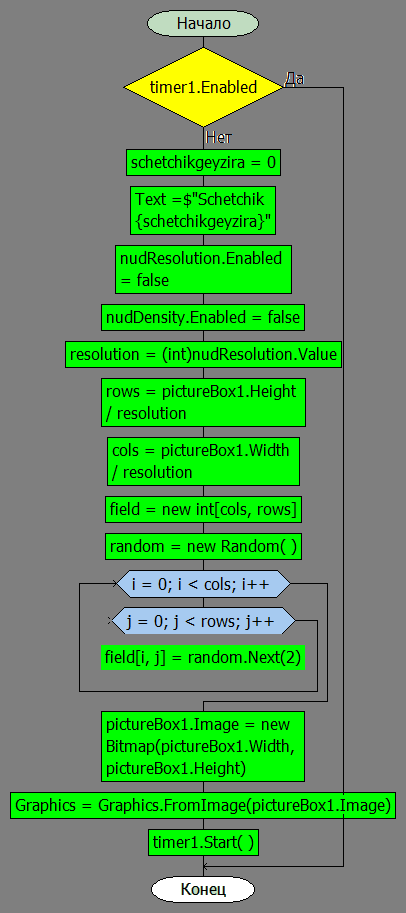
* **Блок-схема функции Red()**



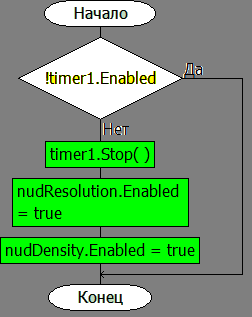
* **Блок-схема класс timer1\_Tick()**



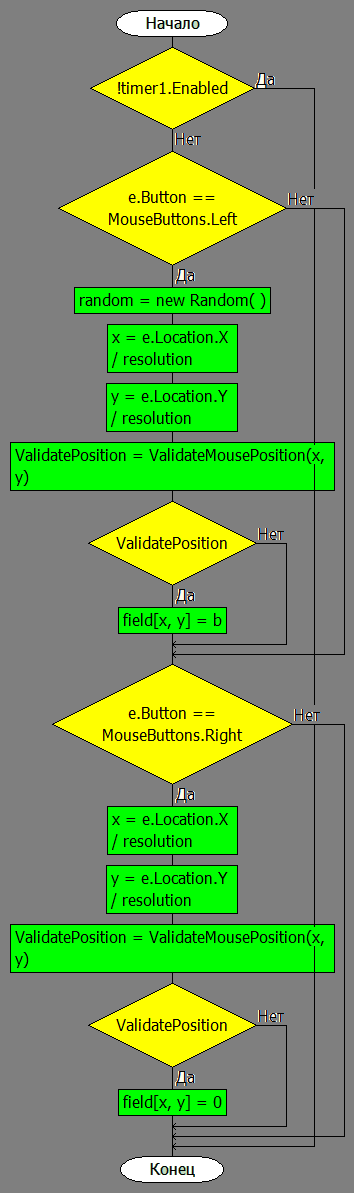
* **Блок-схема класса Start\_Click()**



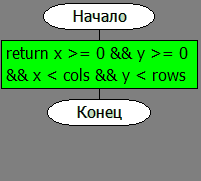
* **Блок-схема класса Stop\_Click()**



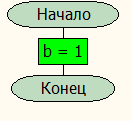
* **Блок-схема функции pictureBox1\_MouseMove()**



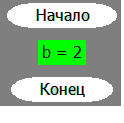
* **Блок-схема функции MousePosition()**



* **Блок-схема класса bred\_Click()**



* **Блок-схема bblue\_Click()**



* **Код программы**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

{

public partial class LIFE : Form

{

private int schetchikgeyzira = 0;

private Graphics Graphics;

private int resolution;

private int rows;

private int cols;

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

int b = 0;

public LIFE()

{

InitializeComponent();

}

private int Red(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 isCheking = ((col == x) && (row == y));

var lifeColor = field[col, row];

if (lifeColor == 1 && !isCheking)

{

count++;

}

}

}

return count;

}

private int Blue(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 isCheking = ((col == x) && (row == y));

var lifeColor = field[col, row];

if (lifeColor == 2 && !isCheking)

{

count++;

}

}

}

return count;

}

private void timer1\_Tick(object sender, EventArgs e)

{

Graphics.Clear(Color.Black);

bool newGeneration = false;

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 Nightboarth = Red(x, y);

var lifeColor = field[x, y];

if (lifeColor == 1 && (Nightboarth == 2 || Nightboarth == 3))

{

newfield[x, y] = 1;

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

continue;

}

else if (lifeColor == 1 && (Nightboarth > 2 || Nightboarth < 3))

{

newfield[x, y] = 0;

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

continue;

}

if (lifeColor != 1 && Nightboarth == 3)

{

newfield[x, y] = 1;

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

newGeneration = true;

continue;

}

}

}

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

{

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

{

var Nightboarth = Blue(x, y);

var lifeColor = field[x, y];

if (lifeColor == 2 && (Nightboarth == 2 || Nightboarth == 3))

{

newfield[x, y] = 2;

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

continue;

}

else if (lifeColor == 2 && (Nightboarth > 2 || Nightboarth < 3))

{

newfield[x, y] = 0;

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

continue;

}

if (lifeColor != 2 && Nightboarth == 3)

{

newfield[x, y] = 2;

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

newGeneration = true;

continue;

}

}

}

if (newGeneration)

{

schetchikgeyzira++;

}

Text = $"Schetchik {schetchikgeyzira}";

field = newfield;

pictureBox1.Refresh();

}

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

{

if (timer1.Enabled)

return;

schetchikgeyzira = 0;

Text = $"Schetchik {schetchikgeyzira}";

nudResolution.Enabled = false;

nudDensity.Enabled = false;

resolution = (int)nudResolution.Value;

rows = pictureBox1.Height / resolution;

cols = pictureBox1.Width / resolution;

field = new int[cols, rows];

Random random = new Random();

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

{

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

{

field[i, j] = random.Next(2);

}

}

pictureBox1.Image = new Bitmap(pictureBox1.Width, pictureBox1.Height);

Graphics = Graphics.FromImage(pictureBox1.Image);

timer1.Start();

}

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

{

if (!timer1.Enabled)

return;

timer1.Stop();

nudResolution.Enabled = true;

nudDensity.Enabled = true;

}

private void pictureBox1\_Mouse(object sender, MouseEventArgs e)

{

if (!timer1.Enabled)

return;

if (e.Button == MouseButtons.Left)

{

Random random = new Random();

var x = e.Location.X / resolution;

var y = e.Location.Y / resolution;

var ValidatePosition = MousePosition(x, y);

if (ValidatePosition)

field[x, y] = b;

}

if (e.Button == MouseButtons.Right)

{

var x = e.Location.X / resolution;

var y = e.Location.Y / resolution;

var ValidatePosition = MousePosition(x, y);

if (ValidatePosition)

field[x, y] = 0;

}

}

private bool MousePosition(int x, int y)

{

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

}

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

{

b = 1;

}

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

{

b = 2;

}

private void GameLife\_Load(object sender, EventArgs e)

{

}

private void splitContainer1\_Panel1\_Paint(object sender, PaintEventArgs e)

{

}

}

}