

[**ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ**](https://dogm.mos.ru/)

Центральный административный округ

Государственное бюджетное профессиональное образовательное учреждение города Москвы "Московский автомобильно-дорожный колледж им. А.А. Николаева"

**Производственная Практика**

**Подготовил Шайко Никита**

**Группа 4/11ИП**

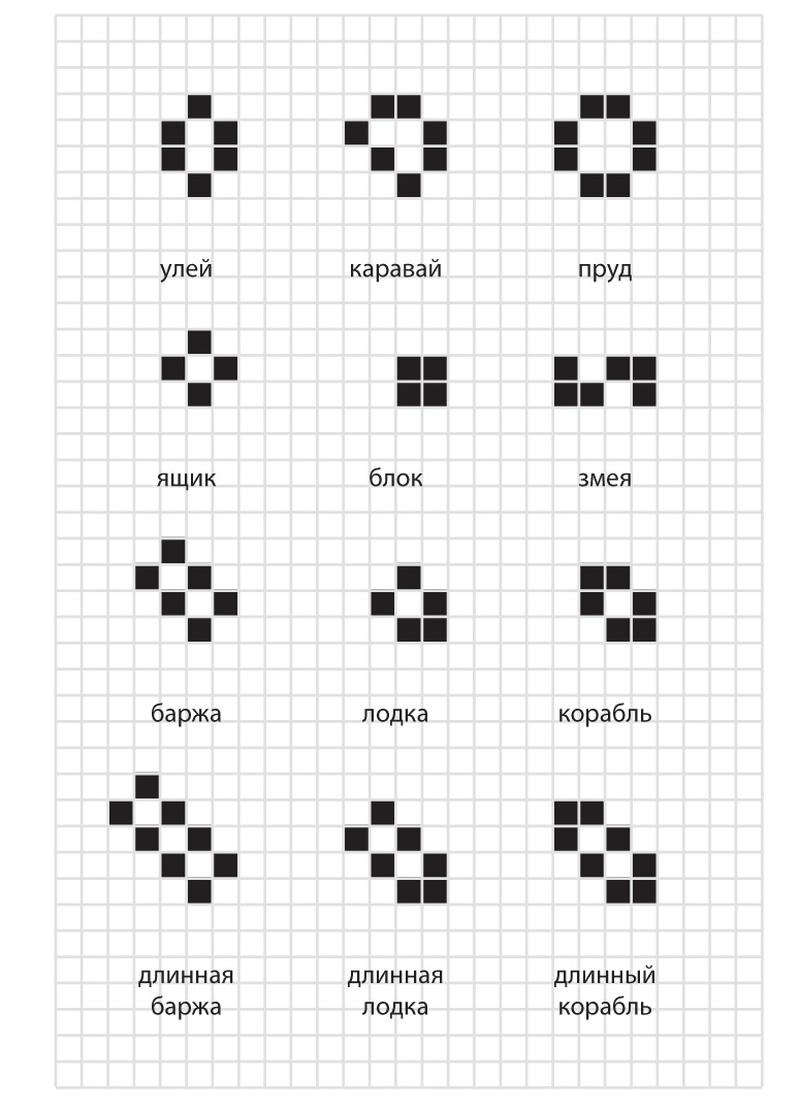
**Специальность: “Информационные системы и программирование (09.02.07)”**

**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.

**Описание функций программы**

1. **Класс gamelifetime ()**

В классе инициализируются компоненты программы.

1. **Функция neigborblue()**

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 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.

1. **Класс buttonStart\_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.

1. **Класс buttonStop\_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 программ, мы можем менять значения разрешения и плотности самих клеток.

1. **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.

1. **Функция ValidateMousePosition()**

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

1. **Класс buttonred\_Click()**

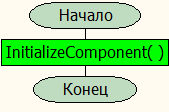
Кнопка смена цвета клеток , которые хочет нарисовать пользователь , при нажатии на кнопку , цвет клетки поменяется на красный. Причем клетки нельзя нарисовать пока не нажата кнопка.

1. **Класс buttonblue\_Click()**

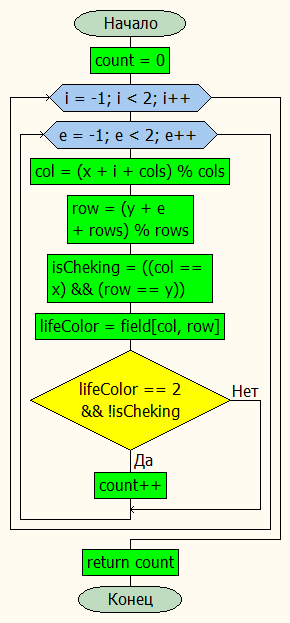
Кнопка смена цвета клеток , которые хочет нарисовать пользователь , при нажатии на кнопку , цвет клетки поменяется на синий . Причем клетки нельзя нарисовать пока не нажата кнопка.

**Блок-схемы программы**

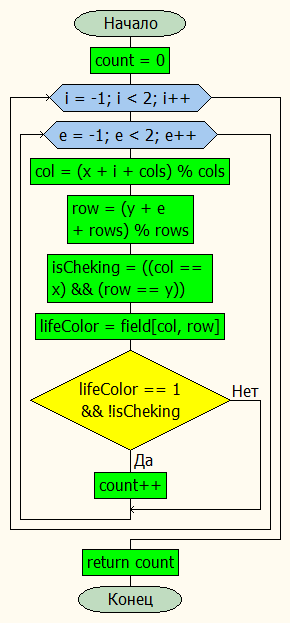
1. **Блок-схема класса gamelifetime ()**



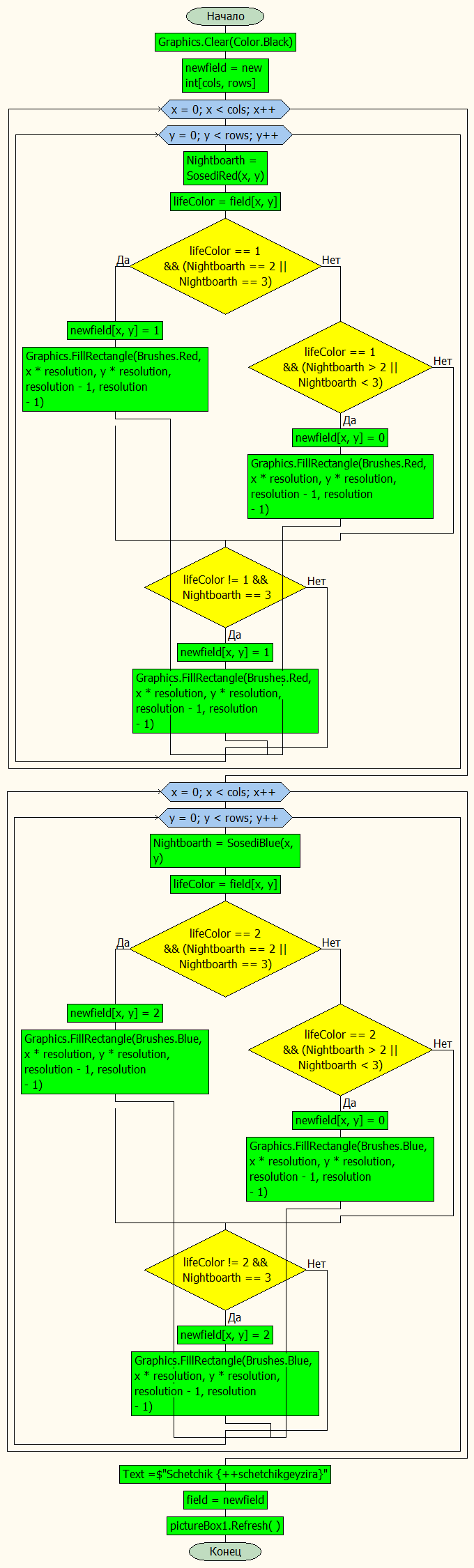
1. **Блок-схема функции neigborblue ()**



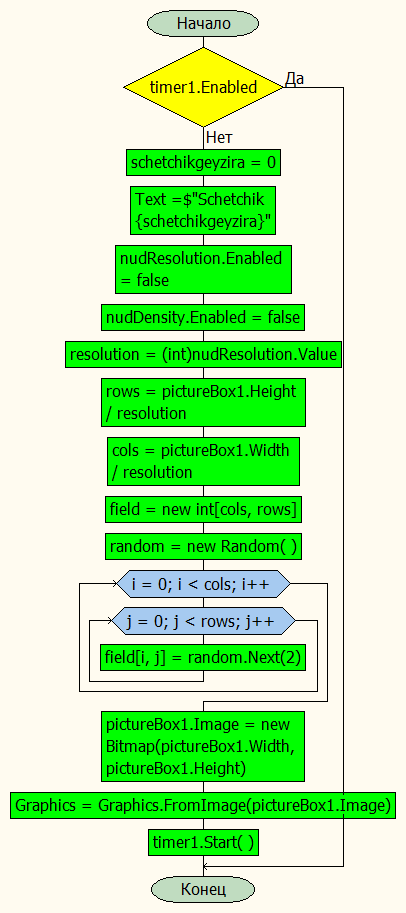
1. **Блок-схема функции neighborred()**



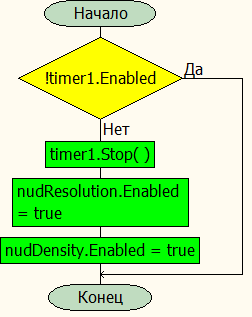
1. **Блок-схема класс timer1\_Tick()**



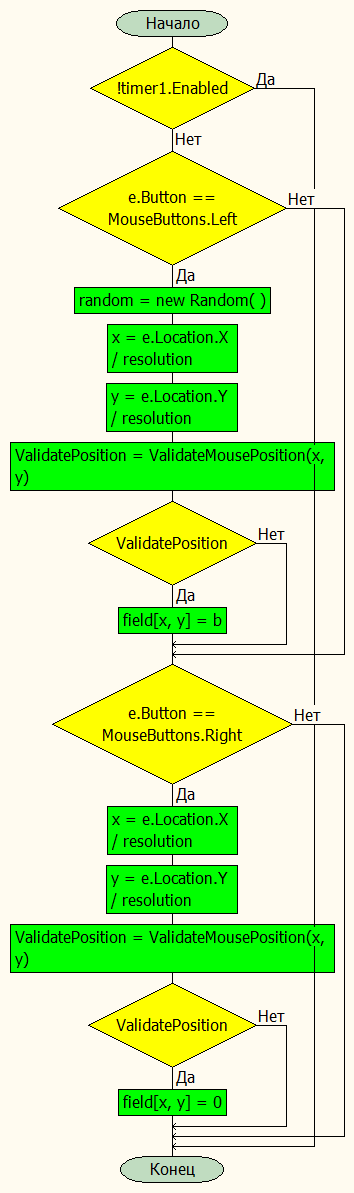
1. **Блок-схема класса buttonStart\_Click()**



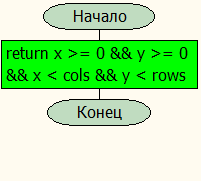
1. **Блок-схема класса buttonStop\_Click()**



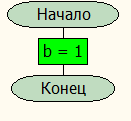
1. **Блок-схема функции pictureBox1\_MouseMove()**



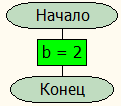
1. **Блок-схема функции ValidateMousePosition()**



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



1. **Блок-схема buttonblue\_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;

namespace Smertiorlife

{

public partial class gamelifetime : Form

{

private int schetchikgeyzira = 0;

private Graphics Graphics;

private int resolution;

private int rows;

private int cols;

private int[,] field;

int b = 0;

public gamelifetime()

{

InitializeComponent();

}

private int neighborred(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 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 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];

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

{

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

{

var Nightboarth = neighborred(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 = neighborblue(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 pictureBox1\_MouseMove(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 = ValidateMousePosition(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 = ValidateMousePosition(x, y);

if (ValidatePosition)

field[x, y] = 0;

}

}

private bool ValidateMousePosition(int x, int y)

{

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

}

private void buttonred\_Click\_1(object sender, EventArgs e)

{

b = 1;

}

private void buttonblue\_Click\_1(object sender, EventArgs e)

{

b = 2;

}

private void buttonStart\_Click\_1(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 buttonStop\_Click\_1(object sender, EventArgs e)

{

if (!timer1.Enabled)

return;

timer1.Stop();

nudResolution.Enabled = true;

nudDensity.Enabled = true;

}

}

}