

**Отчет по производственной практике**

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

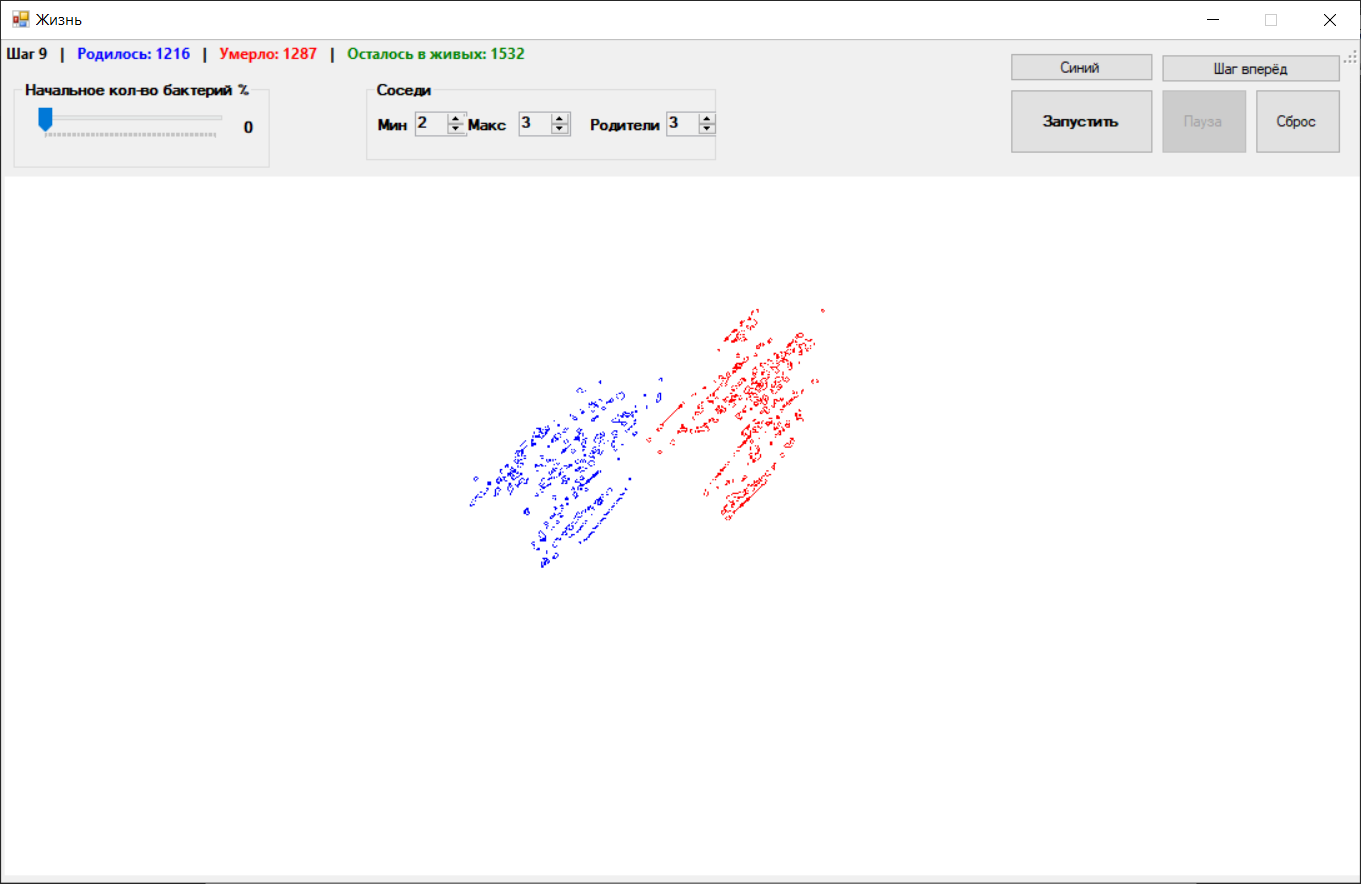
**Подготовил Минчев Сергей 4/11ИП**

**Игра «Жизнь»**

При запуске приложения появляется пустая плоскость – «вселенная», в которой будет развиваться жизнь, кнопка смены цвета «Красный/Синий», кнопки управления: «Запустить», «Пауза», «Шаг вперёд» и «Сброс», а также элементы, позволяющие изменить параметры жизни, а именно:

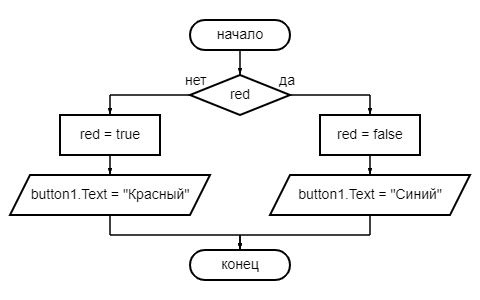
* процент заполнения вселенной живыми клетками перед началом игры;
* минимальное и максимальное кол-во «соседей» (если условие не соблюдается, клетка умирает);
* необходимое число «родителей» рядом друг с другом для того, чтобы в пустой клетке зародилась жизнь;

Выше расположена строка, отображающая номер шага, количество живых, родившихся и умерших клеток за этот шаг. Перед началом игры (а также во время паузы) у пользователя есть возможность нарисовать на поле линию – живые клетки.



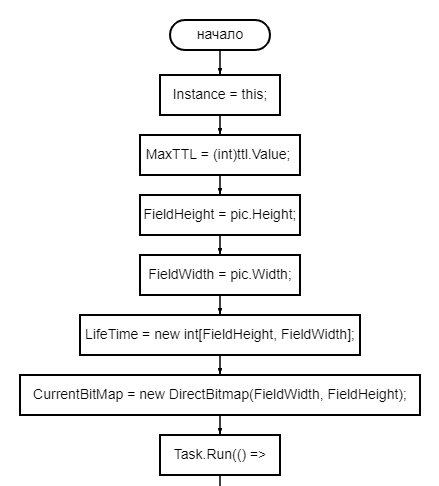
**Кнопка смены цвета «Красный/Синий»**  
**private void ColorButton\_Click(object sender, EventArgs e)**

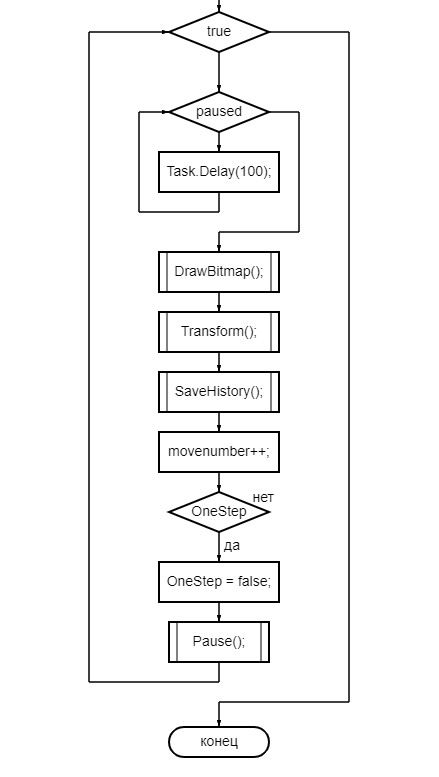
При нажатии меняется цвет, которым пользователь может нарисовать линию на поле. Текст кнопки меняется на название выбранного цвета.



**public MainForm()**

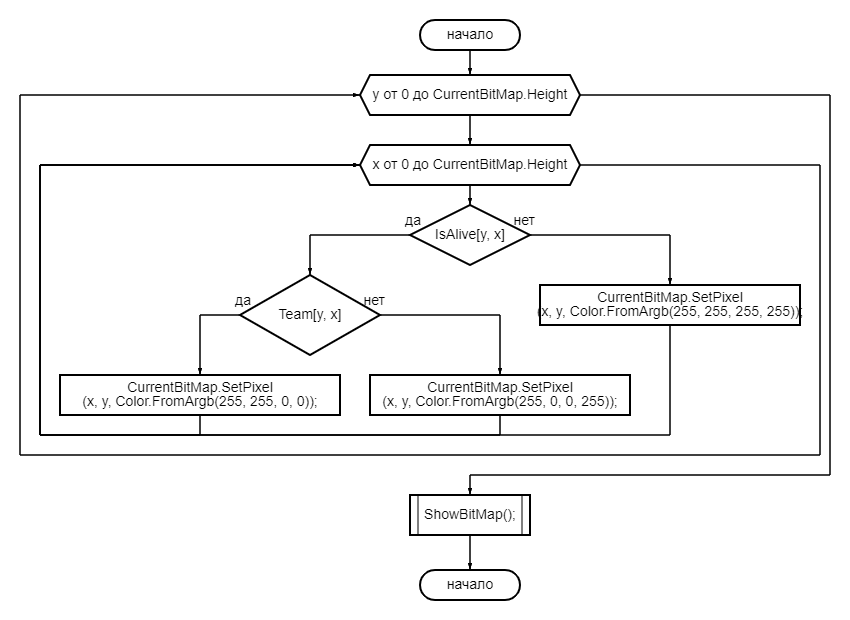
Запускаем задачу для изменения поля в фоновом режиме и бесконечный цикл. Для этого в задаче будут вызываться функции **DrawBitmap()** для рисования самого поля, **Transform()** для трансформации текущего состояния по правилам (условия для развития жизни) и **SaveHistory()** для сохранения истории каждого шага и возможности завершить игру, если конфигурация на очередном шаге в точности повторит себя же на одном из более ранних шагов.





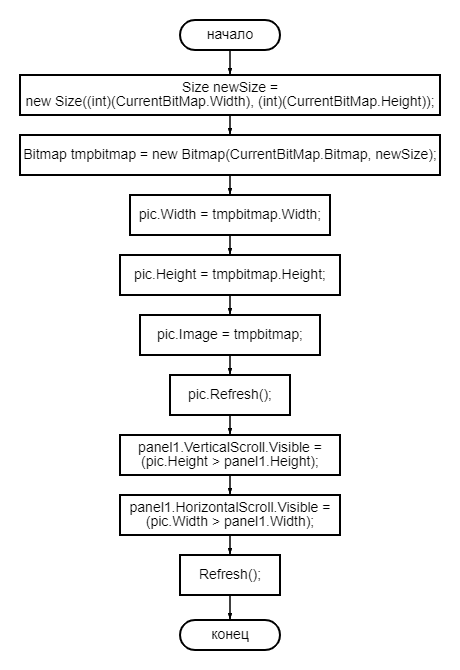
**Блок-схема и описание функции  
private void DrawBitmap()**

Функция проверяет состояние клеток (клеткой считается каждый пиксель поля) и красит его в соответствующий цвет. Затем вызывает функцию ShowBitmap() для отображения на экран.



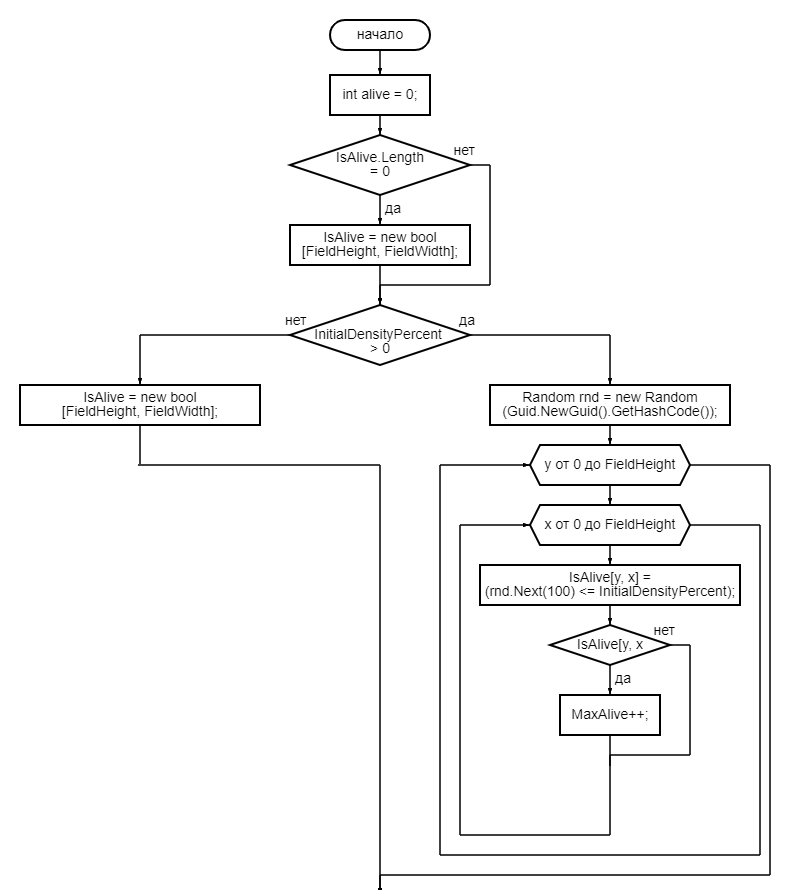
**Блок-схема и описание функции  
private void ShowBitMap()**

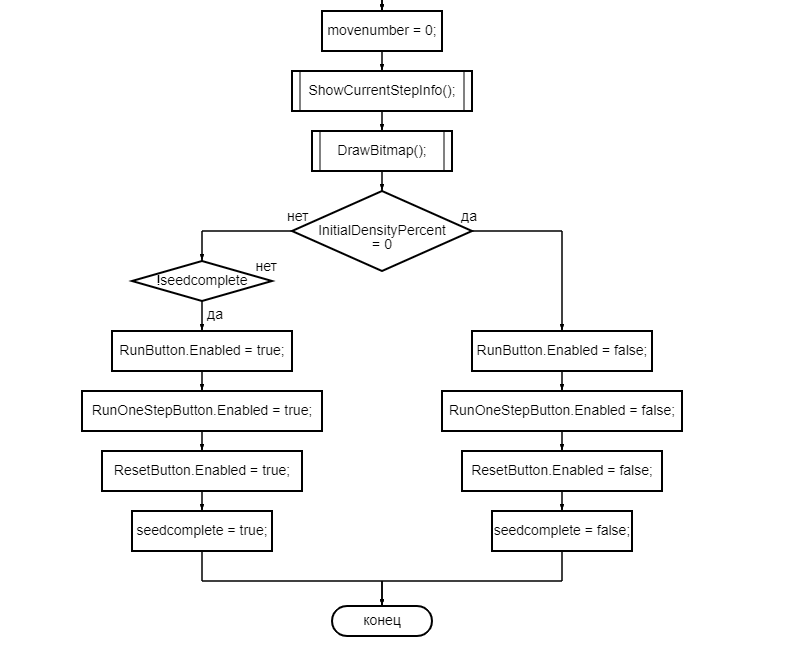
Выводит на экран картинку.



**Блок-схема и описание функции  
private void InitialSeed()**

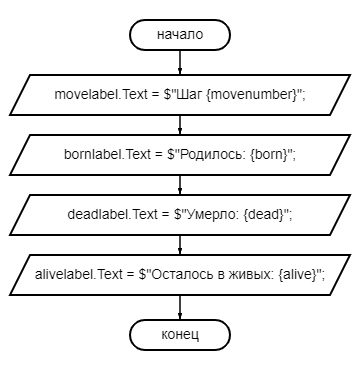
Заполняет поле на основе параметра «Начальное кол-во бактерий %». Если значение равно нулю, то остаётся пустое поле. Если нет – случайные клетки поля становятся живыми и окрашиваются в синий или красный цвет. Становятся доступны кнопки «Запустить», «Шаг вперёд», «Сброс».





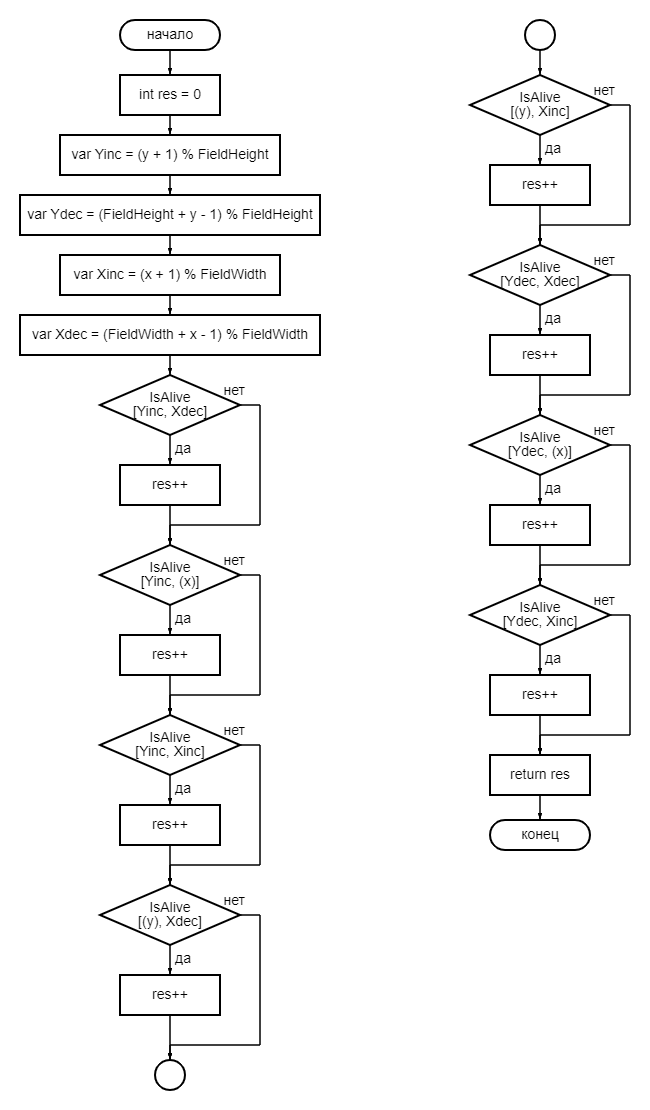
**Блок-схема и описание функции**  
**ShowCurrentStepInfo()**

Функция выводит на экран номер шага, число родившихся умерших и оставшихся в живых клеток.



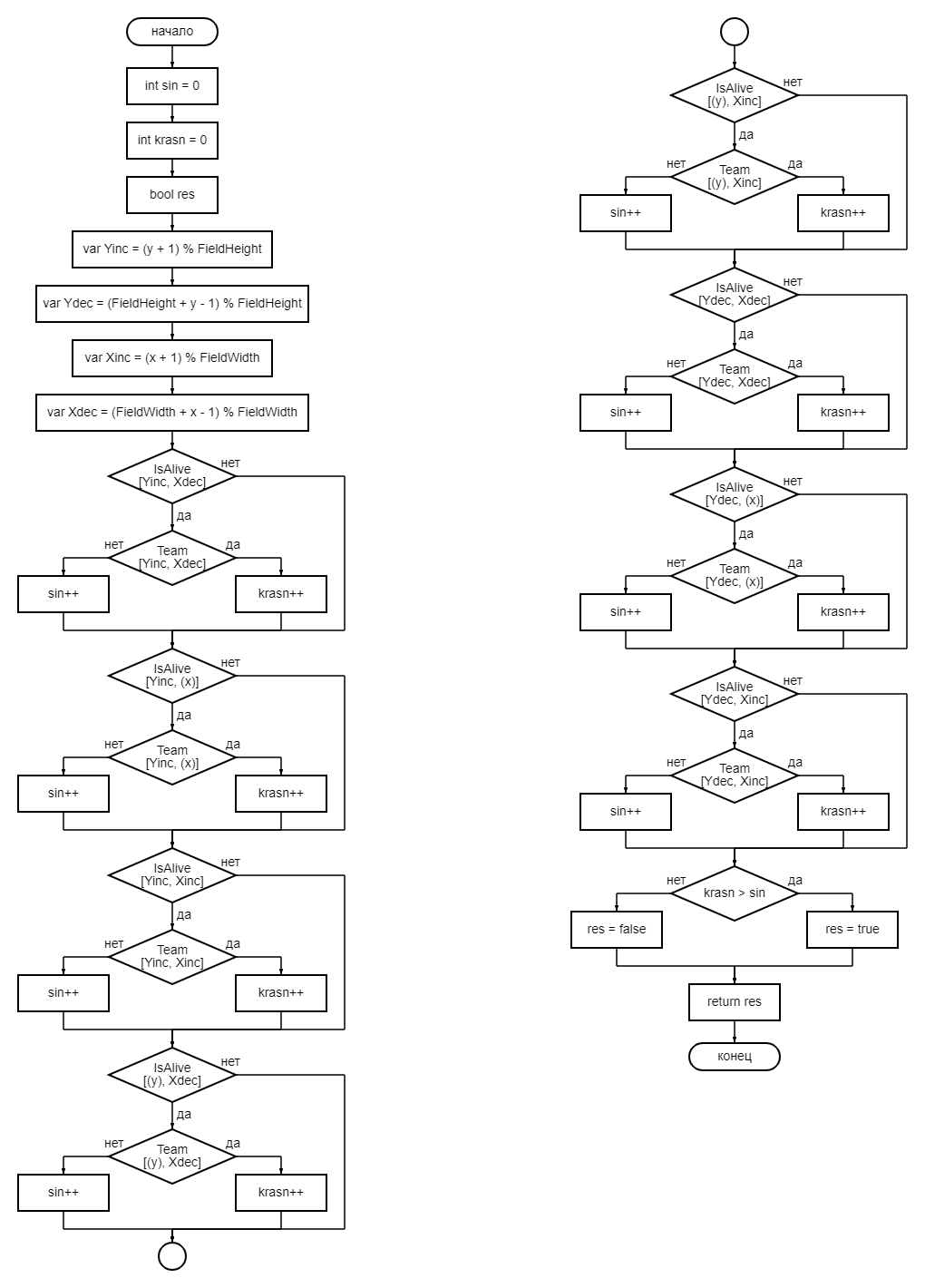
**Блок-схема и описание функции**  
**private int GetNeighborsMask(int y, int x)**

В функцию передаются координаты клетки. Функция проверяет состояние всех клеток вокруг (справа, слева, вверху, внизу, справа-вверху, слева-вверху, справа-внизу, слева-внизу). Если клетка живая, к счётчику прибавляется 1. Возвращает целое число от 0 до 8.



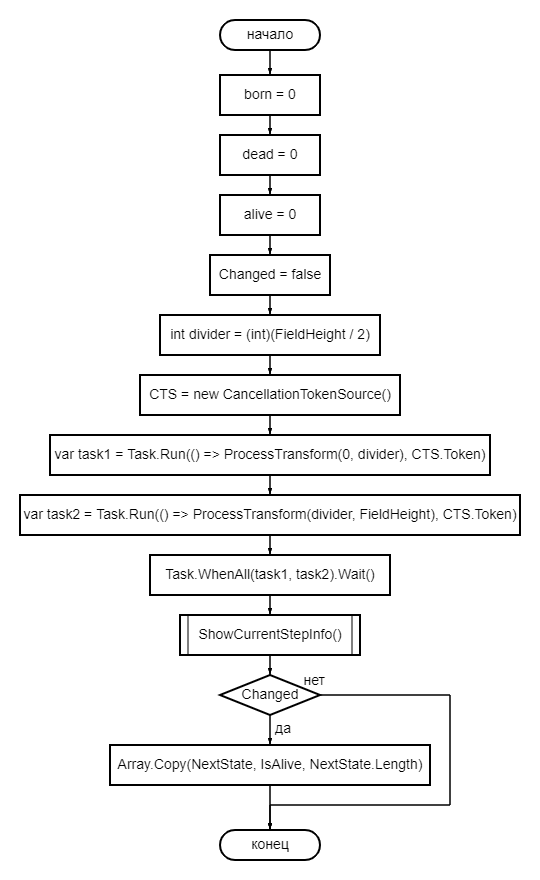
**Блок-схема и описание функции   
private int GetNeighborsTeam(int y, int x)**

Работает аналогично функции **GetNeighborsMask(int y, int x)**. Возвращает *true*, если большая часть живых клеток рядом красные и *false*, если большая часть клеток рядом синие.



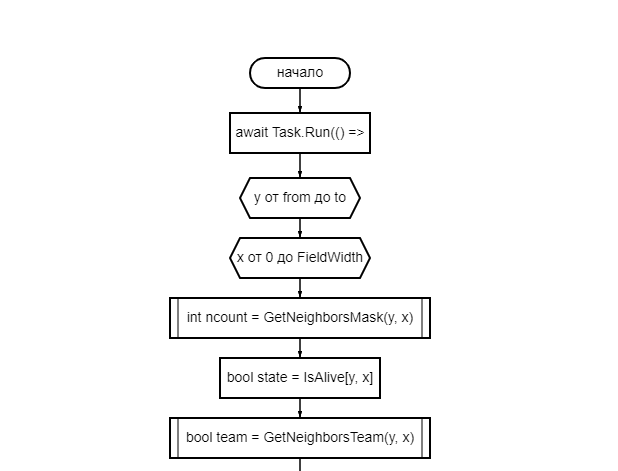
**Блок-схема и описание функции  
private void Transform()**

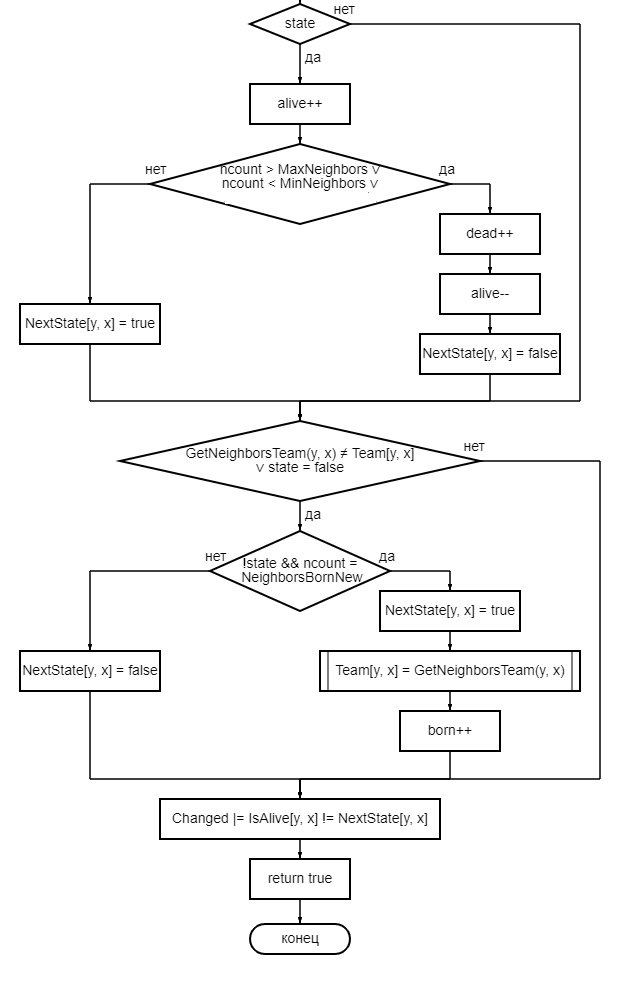
Вызывает фоновые задачи для трансформации поля ProcessTransform(int from, int to) по заданным правилам. После окончания выполнения задач обновляет строку с информацией о текущем шаге и, если произошли изменения, они сохраняются.



**Блок-схема и описание функции**   
**private async Task<bool> ProcessTransform(int from, int to)**

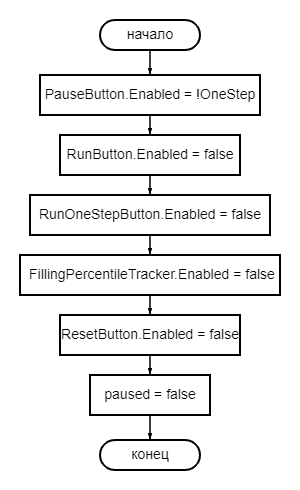
Фоновая задача для преобразования игрового поля. При помощи двух циклов (для координат x и y) функция проходит через каждую клетку игрового поля. Для каждой клетки функция получает количество её соседей (живых клеток рядом) – функция **GetNeighborsMask(y, x)**, цвет соседних клеток – функция **GetNeighborsTeam(y, x)**, текущее состояние клетки (true – жива, false – мертва). Если клетка жива, проверяется количество её соседей и при определённых условиях она умирает от одиночества или перенаселённости. Если клетка пустая, в ней зарождается жизнь при определённом числе соседей.





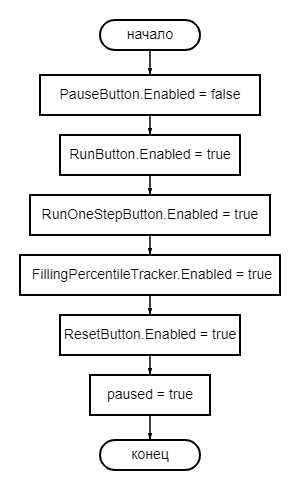
**Блок-схема и описание функции private void Play()**

Функция вызывается при нажатии на кнопку «Запустить» или «Шаг вперёд» (во втором случае OneStep принимает значение true). Становятся недоступными кнопки «Запустить», «Шаг вперёд», «Сброс» и ползунок «Начальное к-во бактерий». Доступность кнопки «Пауза» меняется в зависимости от флага OneStep (Пошаговое выполнение. Если включено – кнопка недоступна и наоборот). Переменная paused принимает значение false, симуляция жизни запускается.



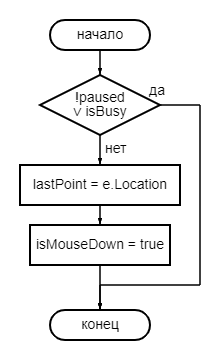
**Блок-схема и описание функции private void Pause()**

Функция вызывается при нажатии на кнопку «Пауза» или флагом OneStep. Переменной paused присваивается значение true, симуляция жизни приостанавливается. Становятся доступны кнопки «Запустить», «Шаг вперёд» и ползунок, блокируется кнопка «Пауза».



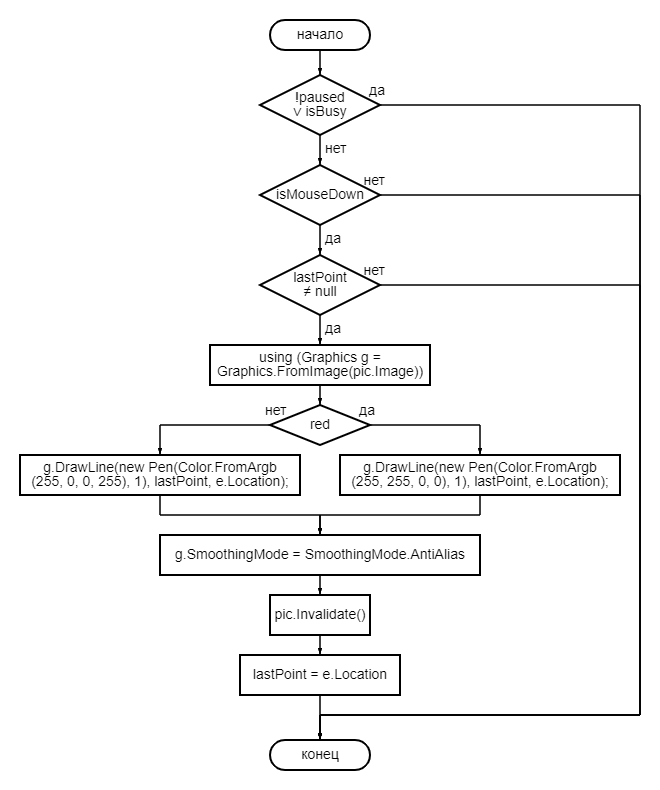
**Блок-схема и описание функции  
private void pic\_MouseDown(object sender, MouseEventArgs e)**

Вызывается при зажатии мышки на поле. Переменной IsMouseDown присваивается значение true для дальнейшего рисования.



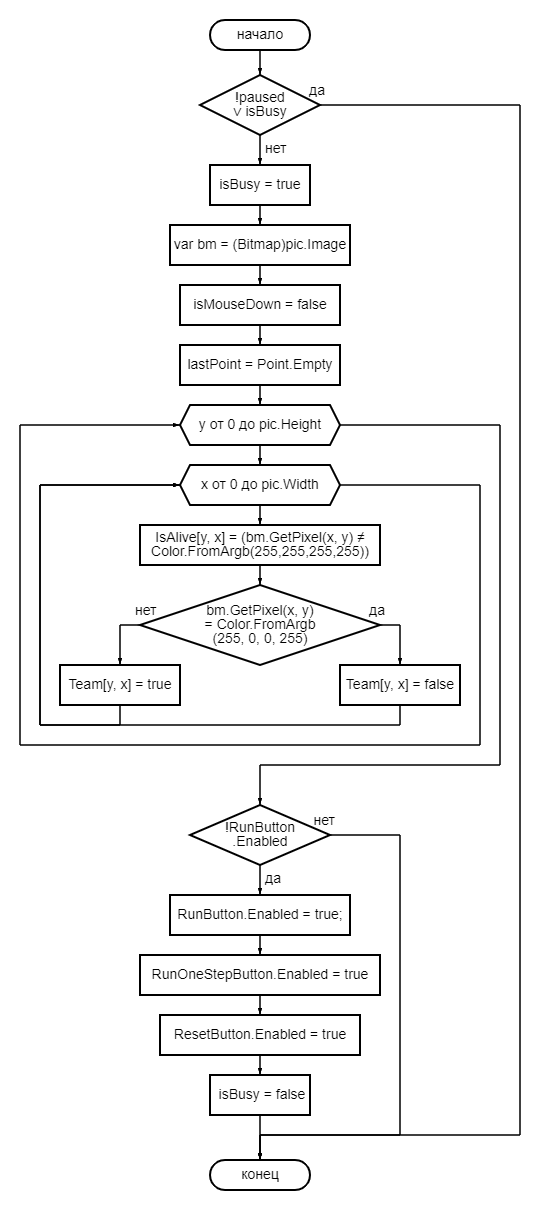
**Блок-схема и описание функции  
private void pic\_MouseMove(object sender, MouseEventArgs e)**

Вызывается перемещением зажатого (IsMouseDown = true) курсора по полю только во время паузы. Рисует линию.



**Блок-схема и описание функции  
private void pic\_MouseUp(object sender, MouseEventArgs e)**

Вызывается при отжатии мышки после окончания рисования линии на поле. IsMouseDown присваивается значение false. С помощью двух циклов (для координат x и y) проходим через все пиксели поля и меняем параметры клеток в зависимости от цвета пикселей. Делает доступными кнопки «Запустить», «Шаг вперёд» и «Сброс», если они были недоступны.



**Блок-схема и описание функции private void SaveHistory()**

Сохраняет конфигурацию каждого шага для того, чтобы при определённых условиях остановить игру (если не осталось живых клеток или конфигурация шагов полностью совпадает).

