

# Основы MATLAB

## Структурное программирование

### Интегрированные математические пакеты

Юдинцев В. В.

Кафедра теоретической механики

13 сентября 2024 г.



САМАРСКИЙ УНИВЕРСИТЕТ  
SAMARA UNIVERSITY

# Содержание

1 Циклы

2 Векторизация

3 Операторы ветвления

4 Задачи

# Циклы

# Оператор цикла for

```
1 for count = start:step:final  
2     команды  
3     команды  
4 end
```

```
1 a = 0;  
2 for k = 1:10  
3     a = sin(k*pi)+a;  
4 end
```

# Оператор цикла while

Общий вид:

```
1 while условие  
2     команды  
3     команды  
4 end
```

Пример

```
1 while abs(xErr) < 0.001  
2     x1 = ...;  
3     x2 = ...;  
4     xErr = getError(x1, x2);  
5 end
```

# Операторы сравнения

---

Функция	Синтаксис
Равно	$x==y$
Не равно	$x\neq y$
Меньше	$x < y$
Больше	$x > y$
Меньше или равно	$x \leq y$
Больше или равно	$x \geq y$

---

# Продолжение выполнения цикла: continue

- **break** – прерывание цикла.
- **continue** – продолжение.

```
1 while условие
2     команда 1
3     if условие
4         команда 2
5         continue;
6     end
7     команда 3
8 end
```

“Команда 3” в седьмой строке не будет исполнена, если условие в строке 3 будет выполнено.

# Принудительный выход из блока: break

```
1 while условие
2     команда 1
3     if условие
4         команда 2
5         break ;
6     end
7     команда 3
8 end
9 ...
```

Если условие в строке 3 будет выполнено, то после команды 2 и выхода из цикла, программа продолжит работу со строки 9.

# Векторизация

# Функция `linspace`

Функция `linspace` создает последовательность из  $n$  точек в интервале от  $a$  до  $b$ , включая границы

```
1 >> linspace(5,10,5)
2
3 ans =
4
5      5.0000    6.2500    7.5000    8.7500    10.0000
```

# Векторы, как аргументы функции

Большинство встроенных функций MATLAB корректно обрабатывают аргументы – векторы (матрицы).

```
1 >> sin(1:5)
2 ans =
3     0.8415    0.9093    0.1411   -0.7568   -0.9589
```

# Векторы, как аргументы функции

Вариант 1: функция работает только со скалярным аргументом

```
1 function f = myfun(x)
2 f=exp(-x)*sqrt((x^2+1)/(x^4+0.1))
```

Вариант 2: функция адаптирована для векторного аргумента

```
1 function f = myfun(x)
2 f=exp(-x).*sqrt((x.^2+1)./(x.^4+0.1))
```

Вызов функции: `myfun([0.1 0.2 0.3])`

# Функция arrayfun

Функция `arrayfun` позволяет применить функцию к элементам массива.

```
1 function f = myfun(x)
2 f=exp(-x)*sqrt((x^2+1)/(x^4+0.1))
```

Необходимо применить функцию `myfun` к каждому элементу массива *a*:

```
1 >> a = 1:5
2
3 ans =
4      1 2 3 4 5
```

```
1 >> arrayfun(myfun, a)
2
3 ans =
4      0.4960    0.0754    0.0175    0.0047    0.0014
```

# Функция arrayfun

Для функции нескольких аргументов:

```
1 function res = mean_function(a, b)
2     res = (a + b)/2;
```

```
1 >> a = [1 2 3 4];
2 >> b = [2 3 4 5];
3 >> arrayfun(mean_function, a, b)
4
5 ans =
6     1.5000    2.5000    3.5000    4.5000
```

# Векторизация

Дан массив из 6 векторов расположенных в матрице  $3 \times 6$  в столбцах:

```
1 >> a = [1 2 3 4 5 6 ;  
2           7 8 9 10 11 12 ;  
3           1 2 3 4 8 1 ];
```

Модуль векторов можно вычислить, используя оператор цикла:

```
1 vnorm = zeros( 1, size(a,2) );  
2  
3 for i = 1:size(a,2)  
4     vnorm(i) = sqrt(a(1,i)^2+a(2,i)^2+a(3,i)^2);  
5 end
```

# Векторизация

Дан массив из 6 векторов расположенных в матрице  $3 \times 6$  в столбцах:

```
1 >> a = [1 2 3 4 5 6 ;  
2      7 8 9 10 11 12 ;  
3      1 2 3 4 8 1];
```

Эффективнее использовать встроенные функции:

```
1 vnorm = sqrt( sum(a.*a, 1) )
```

# Операторы ветвления

# Оператор if

```
1 if varA>5  
2     команды , выполняемые если varA >5  
3 end
```

Больше выбор:

```
1 if условие  
2     команды  
3 elseif условие  
4     команды  
5 elseif условие  
6     команды  
7 else  
8     команды  
9 end
```

# Оператор switch

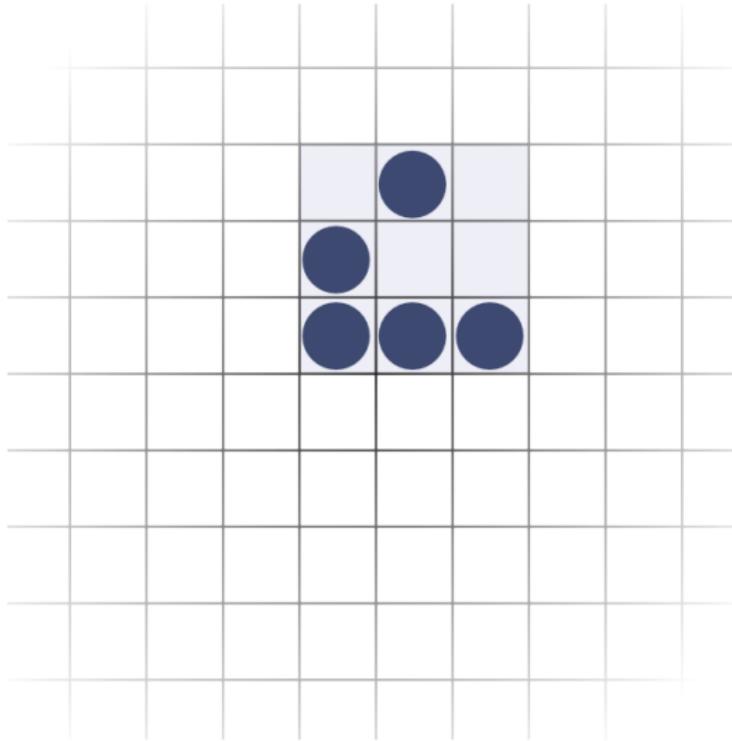
```
1 switch varA
2 case 1
3     команды если varA=1
4 case 2
5     команды если varA=2
6 case 3
7     команды если varA=3
8 otherwise
9     команды
10 end
```

# Задачи

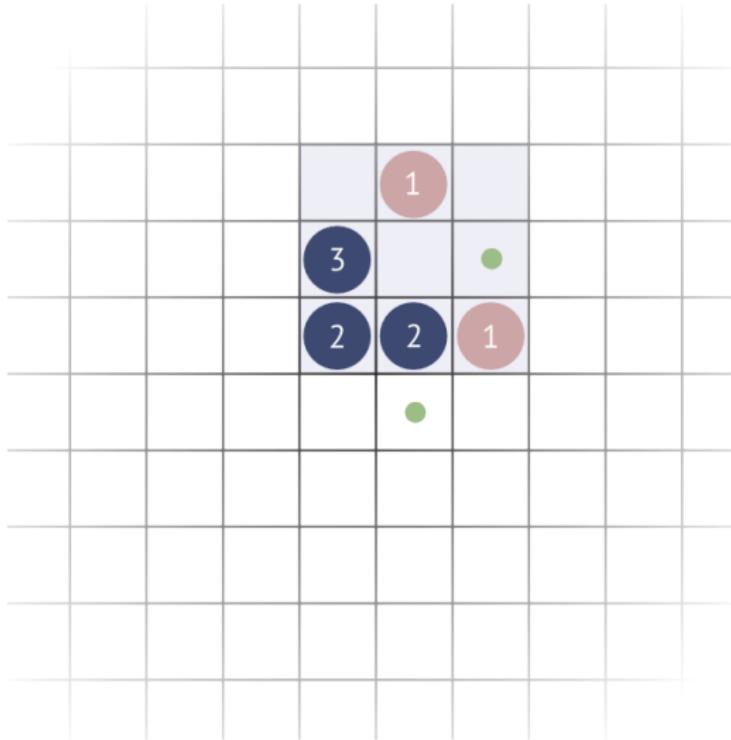
# Игра “Жизнь” (автор Дж. Конвей)

- ① Дано бесконечное поле на плоскости, разбитое на квадратные ячейки.
- ② Каждая ячейка может быть пустой или занятой клеткой.
- ③ Клетка умирает если вокруг неё меньше 2 или больше 3 соседей (занято меньше 2 или больше 3 смежных ячеек).
- ④ Клетка рождается в пустой ячейке если вокруг неё ровно 3 соседа.
- ⑤ Рождение и смерть клеток происходит одновременно.

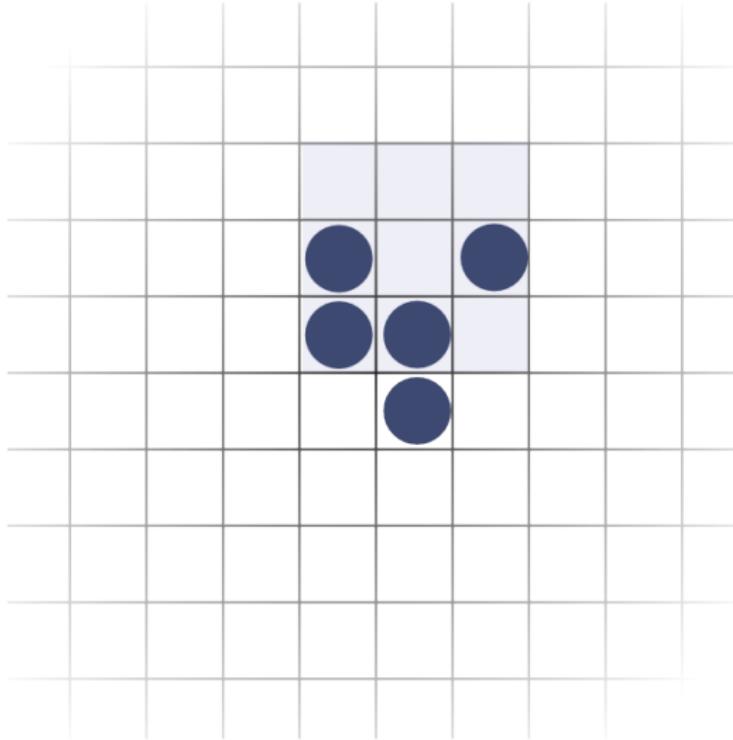
# Глайдер



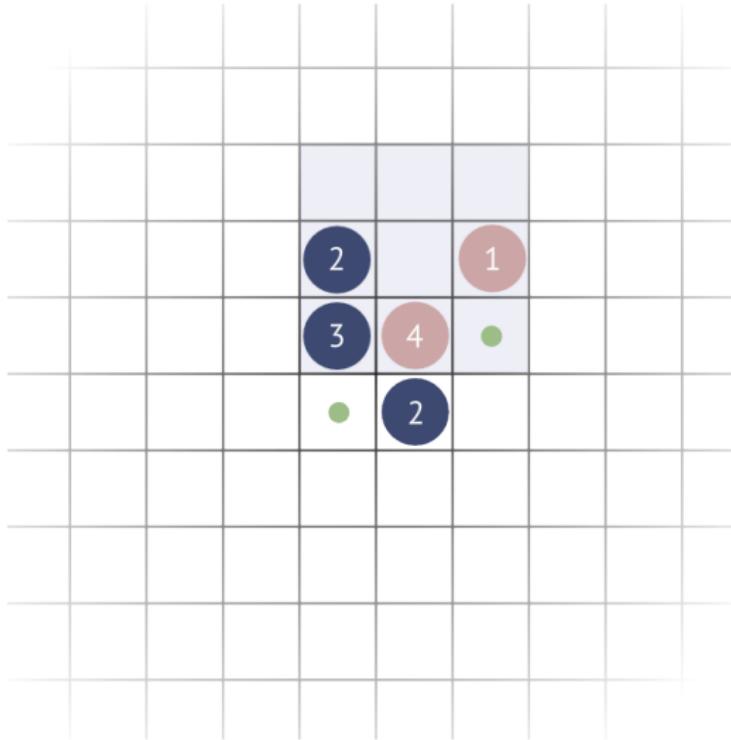
# Глайдер



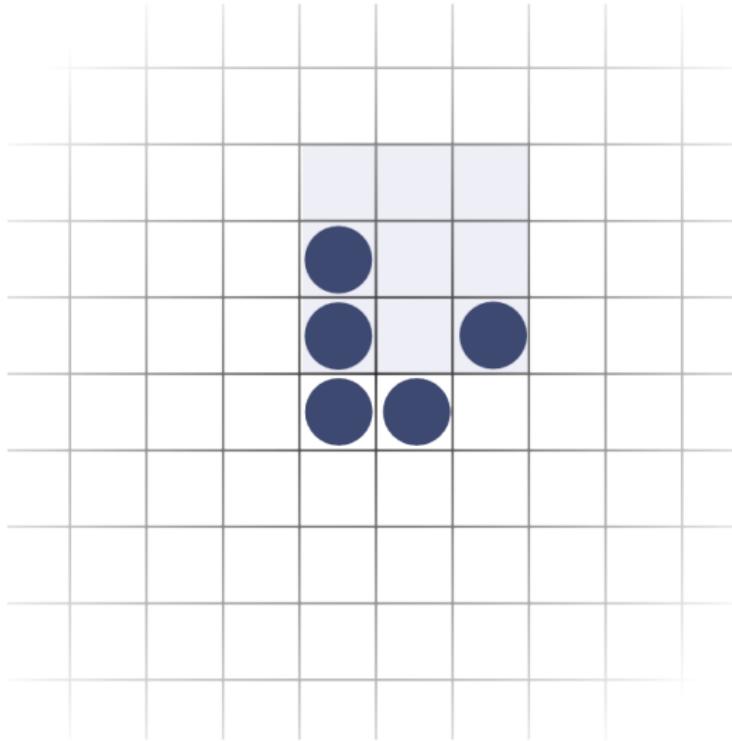
# Глайдер



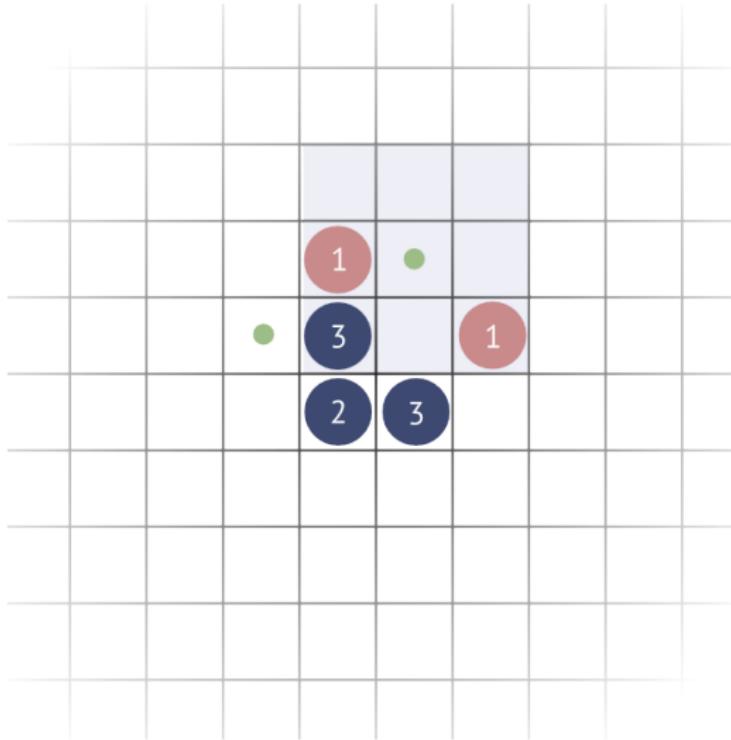
# Глайдер



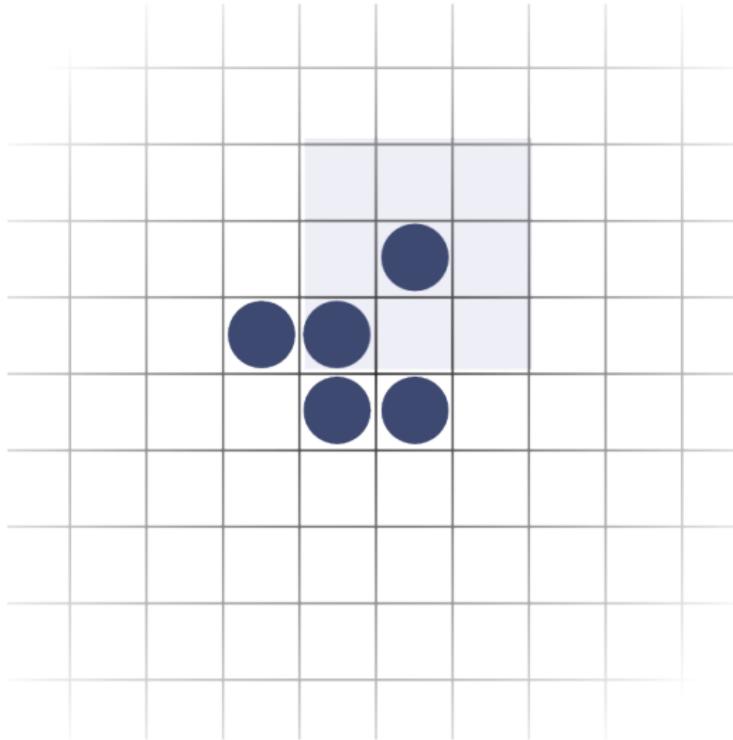
# Глайдер



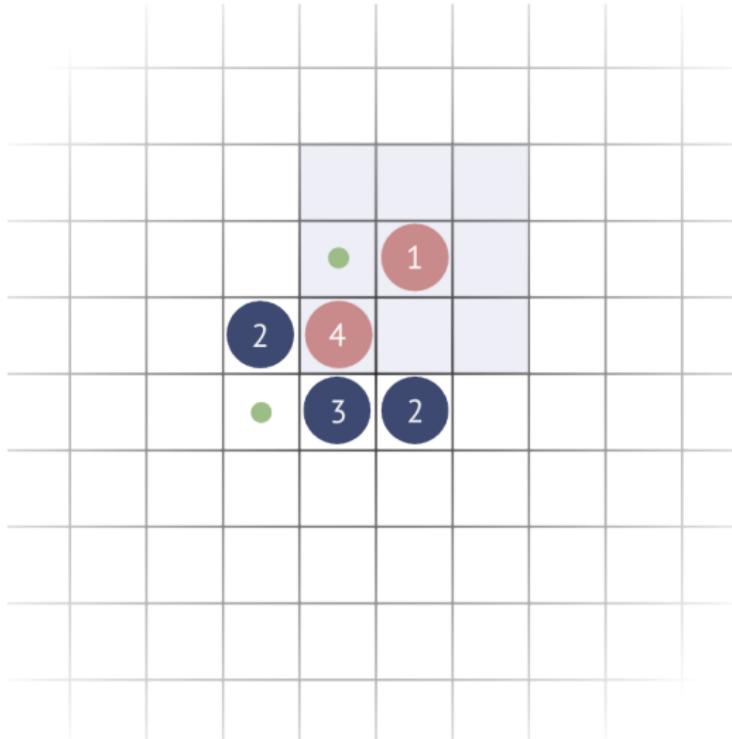
# Глайдер



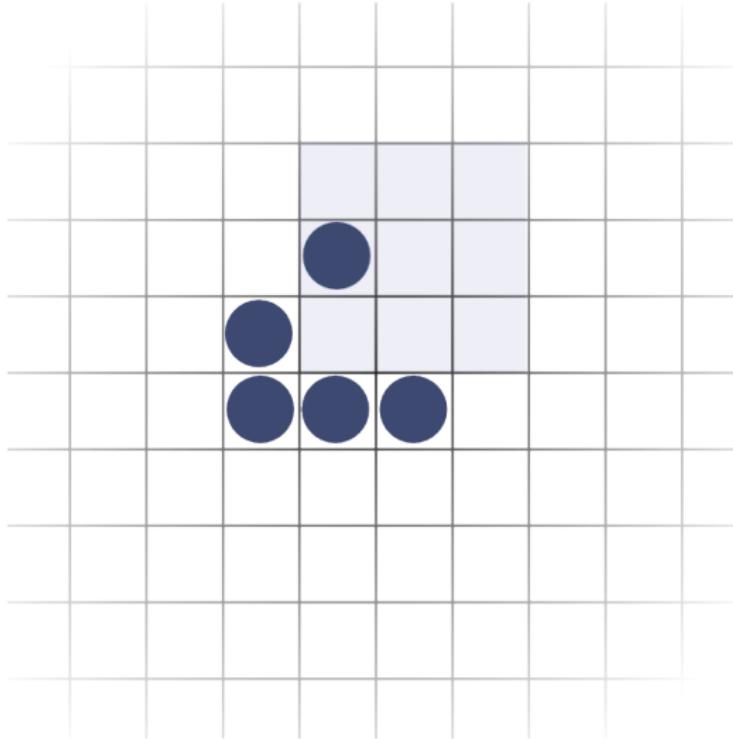
# Глайдер



# Глайдер



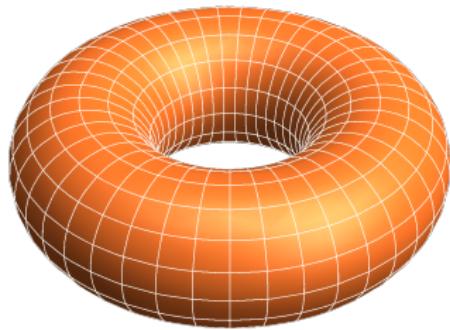
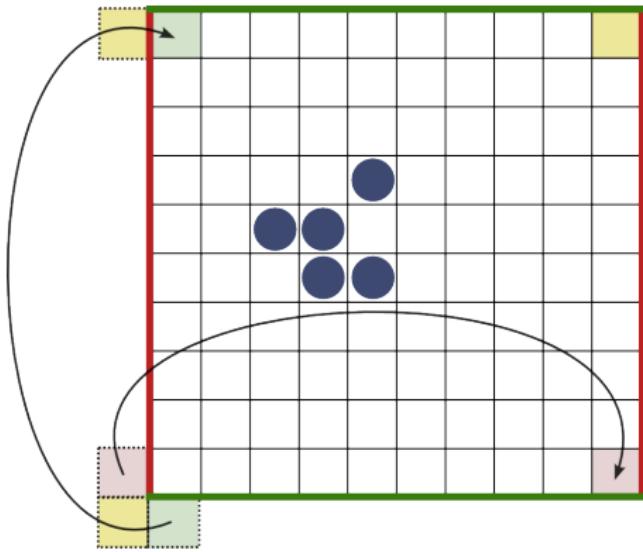
# Глайдер





# Задание

- 1 Напишите программу игры “Жизнь” на бесконечном поле.
- 2 Напишите программу игры “Жизнь” на торе (замкнутое пространство).



# Типы данных

Колония клеток (список клеток) задаётся матрицей, состоящей из двух столбцов. Каждая строка матрицы – это пара координат клетки:

$$\text{colony} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 3 & 2 \\ 2 & 3 \end{bmatrix}$$

```
1 colony = [1, 1; 2, 1; 3, 1; 3, 2; 2, 3];
```

Одна клетка задаётся матрицей-строкой из двух элементов – координат клетки:

```
1 cell = [1, 1];
```

# Функция `cell_in_colony.m`

```
1 function res = cell_in_colony(cell, colony)
2     ...
```

Аргументы функции:

- `cell` – строка с двумя координатами клетки ( $x, y$ )
- `colony` – матрица  $n \times 2$  (два столбца) с координатами клеток колонии

Результат работы функции:

- 1 – клетка принадлежит колонии
- 0 – клетка не принадлежит колонии

# Функция `get_neighbours_cells.m`

Функция для клетки с координатами `cell` определяет список координат, граничащих клеток (пустых или занятых):

```
1 function neighbours = get_neighbours_cells(cell)
2     ...
```

-1,1	0,1	1,1		
-1,0	x,y	1,0		
-1,-1	0,-1	1,-1		

x-1 y+1	x y+1	x+1 y+1		
x-1 y	x,y	x+1 y		
x-1 y-1	x y-1	x+1 y-1		

# Функция count\_cell\_neighbours.m

Функция определяет количество соседей у клетки `cell` в колонии `colony`:

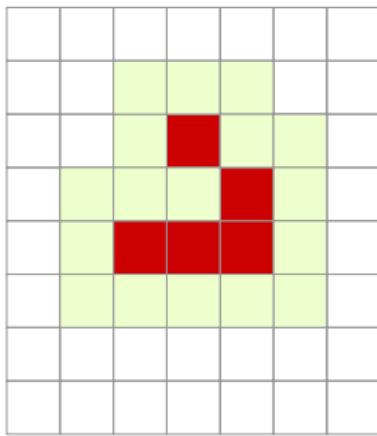
```
1 function count = count_cell_neighbours(cell, colony)
2     ...
```

- ➊ Определяется список координат клеток-соседей (пустых или занятых).
- ➋ Для если клетка из этого списка занята (принадлежит колонии), то счётчик соседей клетки `cell` увеличивается на единицу.

# Функция `get_colony_area.m`

Функция возвращает список клеток, принадлежащих и граничащих с колонией.

```
1 function cells = get_colony_area(colony)
2     ...
```



Клетки, граничащие с колонией



Клетки, принадлежащие колонии

# Функция `next_generation.m`

Функция возвращает матрицу координат клеток следующего поколения для колонии `colony`:

```
1 function next_gen = next_generation(colony)
2     ...
```

Пример работы функции

```
1 colony = [1, 1; 2, 1; 3, 1; 3, 2; 2, 3];
2 colony = next_generation(colony);
3 colony =
4
5     1    2
6     2    0
7     2    1
8     3    1
9     3    2
```