Структурное программирование

Основы MATLAB

Юдинцев В. В.

Кафедра теоретической механики

23 февраля 2026 г.

САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

Содержание

Циклы

Оператор цикла for

```
for count = start:step:final
   команды
   команды
end
```

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

Итерации по вектору-строке

```
1 v = [1  2  3];
2 for  i=v
3     disp('i:');
4     disp(i);
5 end
```

Результат (тело цикла выполняется 3 раза):

```
1 i :
2         1
3
4 i :
5         2
6
7 i :
8         3
```

Итерации по вектору-столбцу

```
1 v = [1; 2; 3];
2 for i=v
3     disp('i:');
4     disp(i);
5 end
```

Результат (тело цикла выполняется 1 раз):

```
1 i :
2        1
3        2
4        3
```

Итерации по матрице

```
1  v = [1  2  3
2       4  5  6];
3  for  i=v
4       disp('i:');
5       disp(i);
6  end
```

Результат (тело цикла выполняется 3 раза):

```
1  i :
2          1
3          4
4  i :
5          2
6          5
7  i :
8          3
9          6
```

Оператор цикла while

Общий вид:

```
while условие
   команды
   команды
end
```

Пример

```
while abs(xErr)<0.001
   x1 = ...;
   x2 = ...;
   xErr = getError(x1,x2);
end
```

Операторы сравнения

Функция	Синтаксис
Равно	`x==y`
Не равно	`x~=y`
Меньше	`x<y`
Больше	`x>y`
Меньше или равно	`x<=y`
Больше или равно	`x>=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

```
>> linspace(5,10,5)

ans =

    5.0000    6.2500    7.5000    8.7500   10.0000
```

Векторы, как аргументы функции

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

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

Векторы, как аргументы функции

Вариант 1: функция работает только со скалярным аргументом

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

Вариант 2: функция адаптирована для векторного аргумента

```
function f = myfun(x)
  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.

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

Необходимо применить функцию `myfun` к каждоиу элементу массива a:

```
>> a = 1:5

ans =
        1  2  3  4  5
```

```
>> arrayfun(myfun, a)

ans =
    0.4960    0.0754    0.0175    0.0047    0.0014
```

Функция arrayfun

Для функции нескольких аргументов:

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

```
>> a = [1 2 3 4];
>> b = [2 3 4 5];
>> arrayfun(mean_function, a, b)

ans =
    1.5000    2.5000    3.5000    4.5000
```

Векторизация

Дан массив из 6 векторов расположенных в матрице 3×6 в столбцах:

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

Модуль векторов можно вычислить, используя оператор цикла:

```
vnorm = zeros( 1, size(a,2) );

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

Векторизация

Дан массив из 6 векторов расположенных в матрице 3×6 в столбцах:

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

Эффективнее использовать встроенные функции:

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

Операторы ветвления

Оператор if

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

Больше выбор:

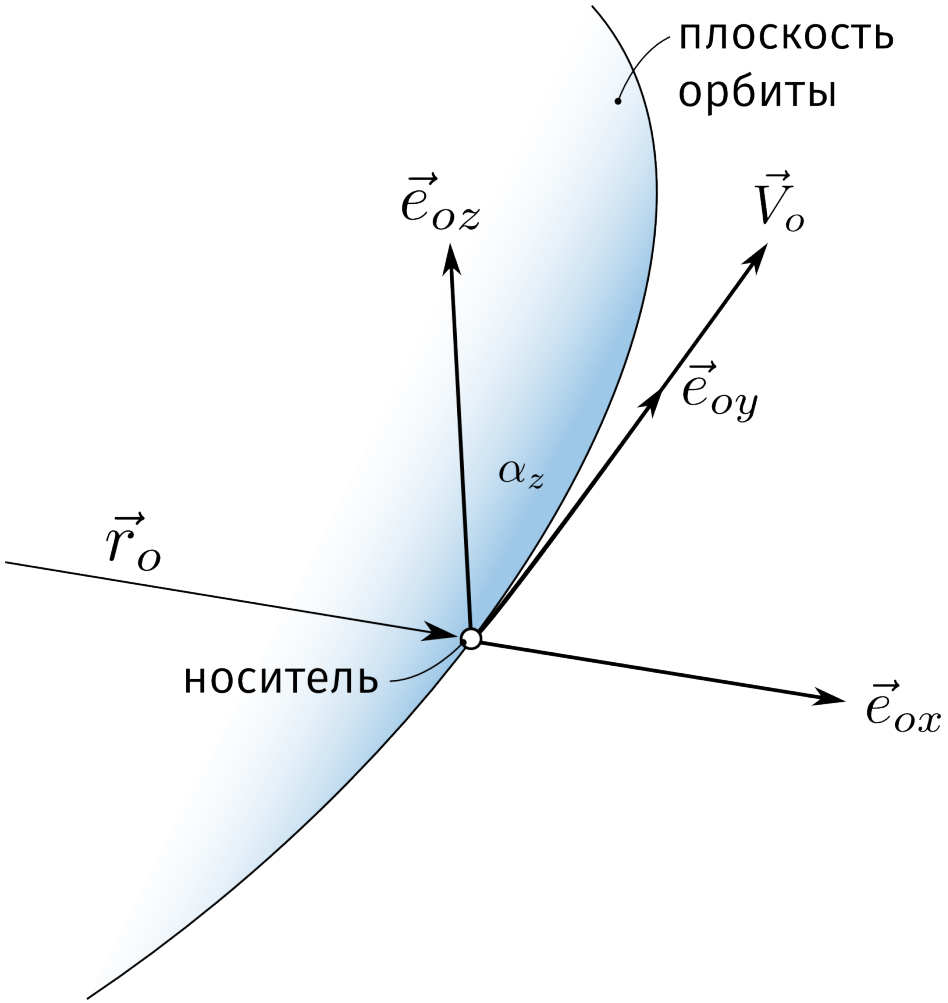
```
1 if условие
2   команды
3 elseif условие
4   команды
5 elseif условие
6   команды
7 else
8   команды
9 end
```

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=31
8  otherwise
9    команды
10 end
```

Пример использования векторизации

Направления осей $Cx_oy_oz_o$



Единичный вектор радиус-вектора носителя:

$$\vec{e}_{ox} = \vec{r}_0 / r_0.$$

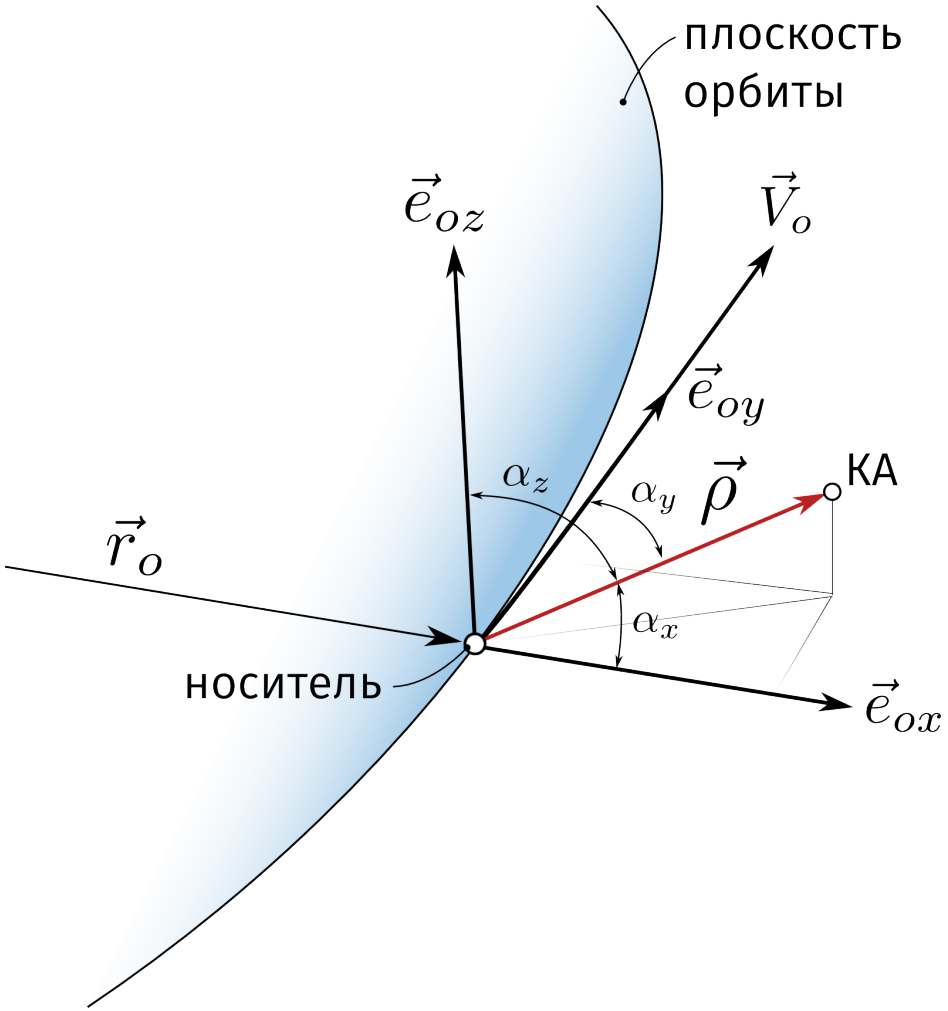
Единичный вектор нормали к орбите:

$$\vec{e}_{oz} = (\vec{r}_0 \times \vec{v}_0) / |\vec{r}_0 \times \vec{v}_0|.$$

Единичный вектор трансверсали:

$$\vec{e}_{oy} = (\vec{e}_{oz} \times \vec{e}_{ox}) / |\vec{e}_{oz} \times \vec{e}_{ox}|.$$

Положение КА относительно носителя



Радиус-вектор положения КА относительно носителя:

$$\vec{\rho} = \vec{r} - \vec{r}_0$$

Координаты КА относительно носителя в $Cx_oy_oz_o$:

$$x = \vec{\rho} \cdot \vec{e}_{ox} = \vec{\rho} \cos \alpha_x,$$
$$y = \vec{\rho} \cdot \vec{e}_{oy} = \vec{\rho} \cos \alpha_y,$$
$$z = \vec{\rho} \cdot \vec{e}_{oz} = \vec{\rho} \cos \alpha_z.$$

Вектор относительного положения

Матрица результатов интегрирования абсолютного движения двух материальных точек:

```
q =
v1x, v1y, v1z, x1, y1, z1, v2x, v2y, v2z, x2, y2, z2
v1x, v1y, v1z, x1, y1, z1, v2x, v2y, v2z, x2, y2, z2
 ...
```

Координатный столбец вектора положения второй точки относительно первой в проекциях на неподвижную систему координат:

```
rho = q(:,10:12)-q(:,1:3);
```

Орты орбитальной подвижной системы

```
q =
v1x, v1y, v1z, x1, y1, z1, v2x, v2y, v2z, x2, y2, z2
v1x, v1y, v1z, x1, y1, z1, v2x, v2y, v2z, x2, y2, z2
 ...
```

Координатные столбцы единичных векторов орбитальной подвижной системы координат, связанной с первым телом:

```
r = sqrt( sum( q(:,4:6).*q(:,4:6) ,2) );
ex0 = q(:,4:6)./repmat(r,1,3);
```

```
v = sqrt( sum( q(:,1:3).*q(:,1:3) ,2) );
ev0 = q(:,4:6)./repmat(v,1,3);
```

Орты орбитальной подвижной системы

```
q =
v1x, v1y, v1z, x1, y1, z1, v2x, v2y, v2z, x2, y2, z2
v1x, v1y, v1z, x1, y1, z1, v2x, v2y, v2z, x2, y2, z2
 ...
```

Единичный вектор нормали к плоскости орбиты:

```
ez0 = cross(ex0,ev0,2);
ez0 = ez0./repmat(sqrt(sum(ez0.*ez0,2)),1,3);
```

Единичный вектор трансверсали:

```
ey0 = cross(ez0,er0,2);
ey0 = ey0./repmat(sqrt(sum(ey0.*ey0,2)),1,3);
```

Относительное положение

Координатный столбец вектора положения второй точки относительно первой в проекциях на неподвижную систему координат:

```
1 rho = q(:,10:12)-q(:,1:3);
```

Тот же вектор в проекция на оси орбитальной подвижной системы:

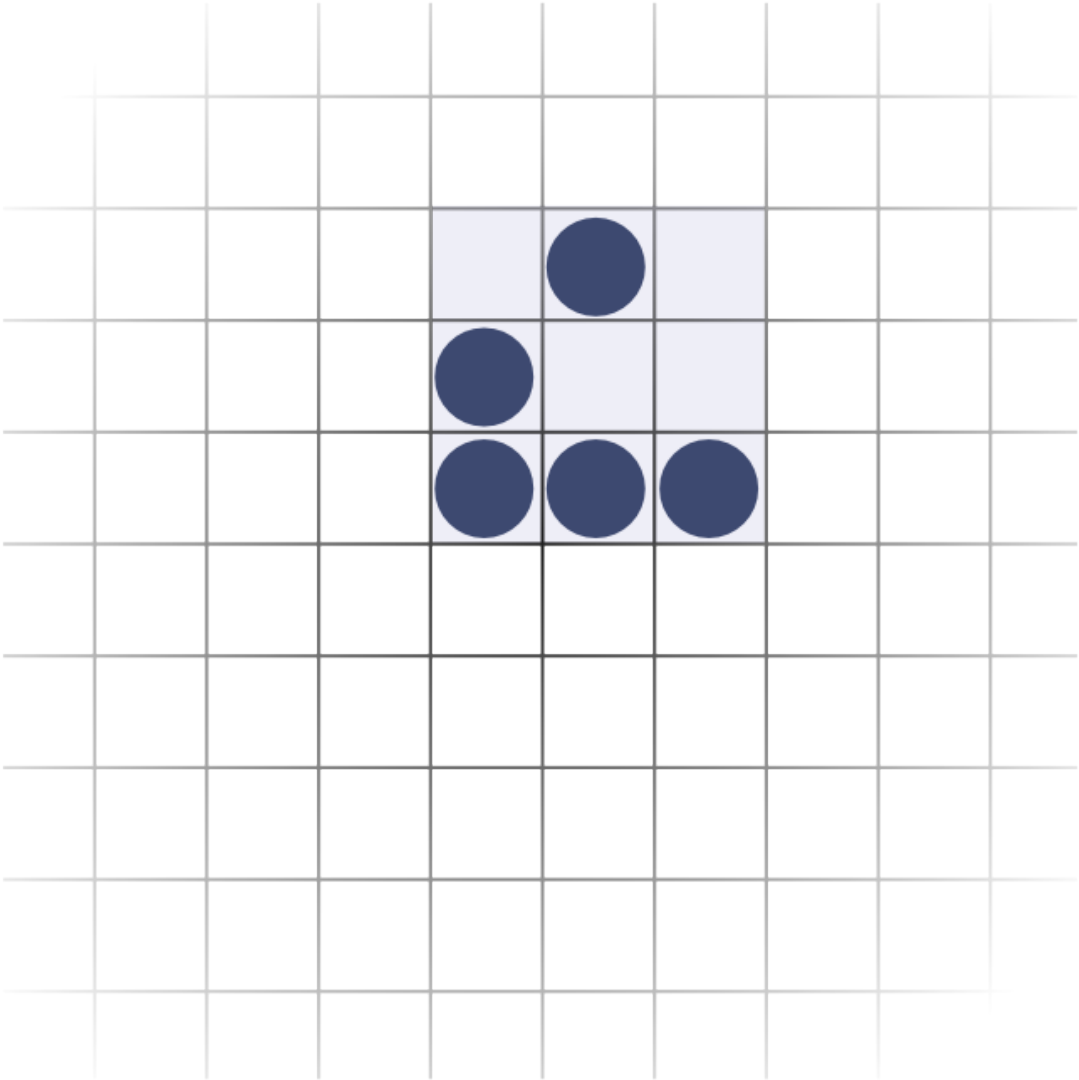
```
1 rho_o = [dot(rho,ex0,2),dot(rho,ey0,2),dot(rho,ez0,2)];
```

Задачи

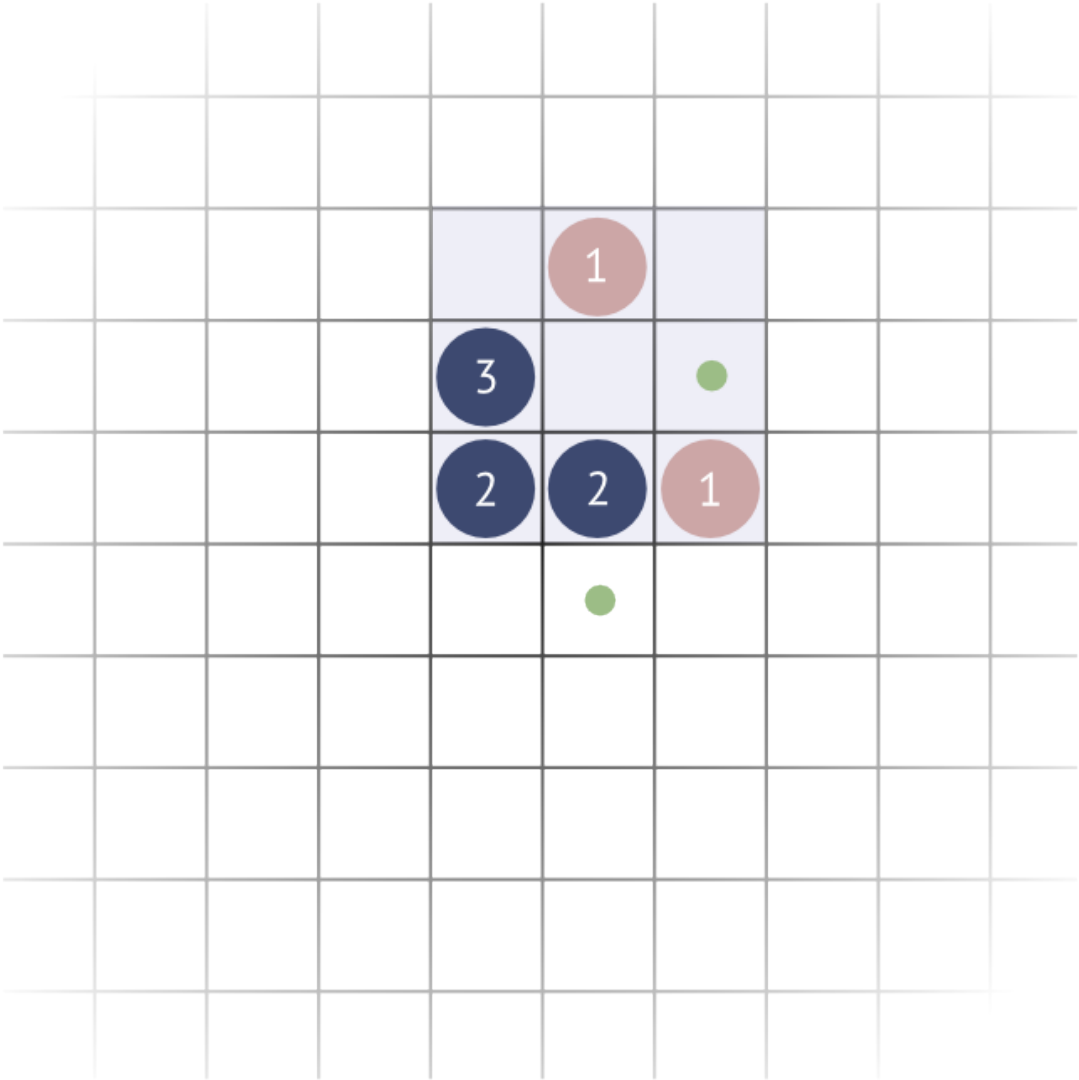
Игра “Жизнь” (автор Дж. Конвей)

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.

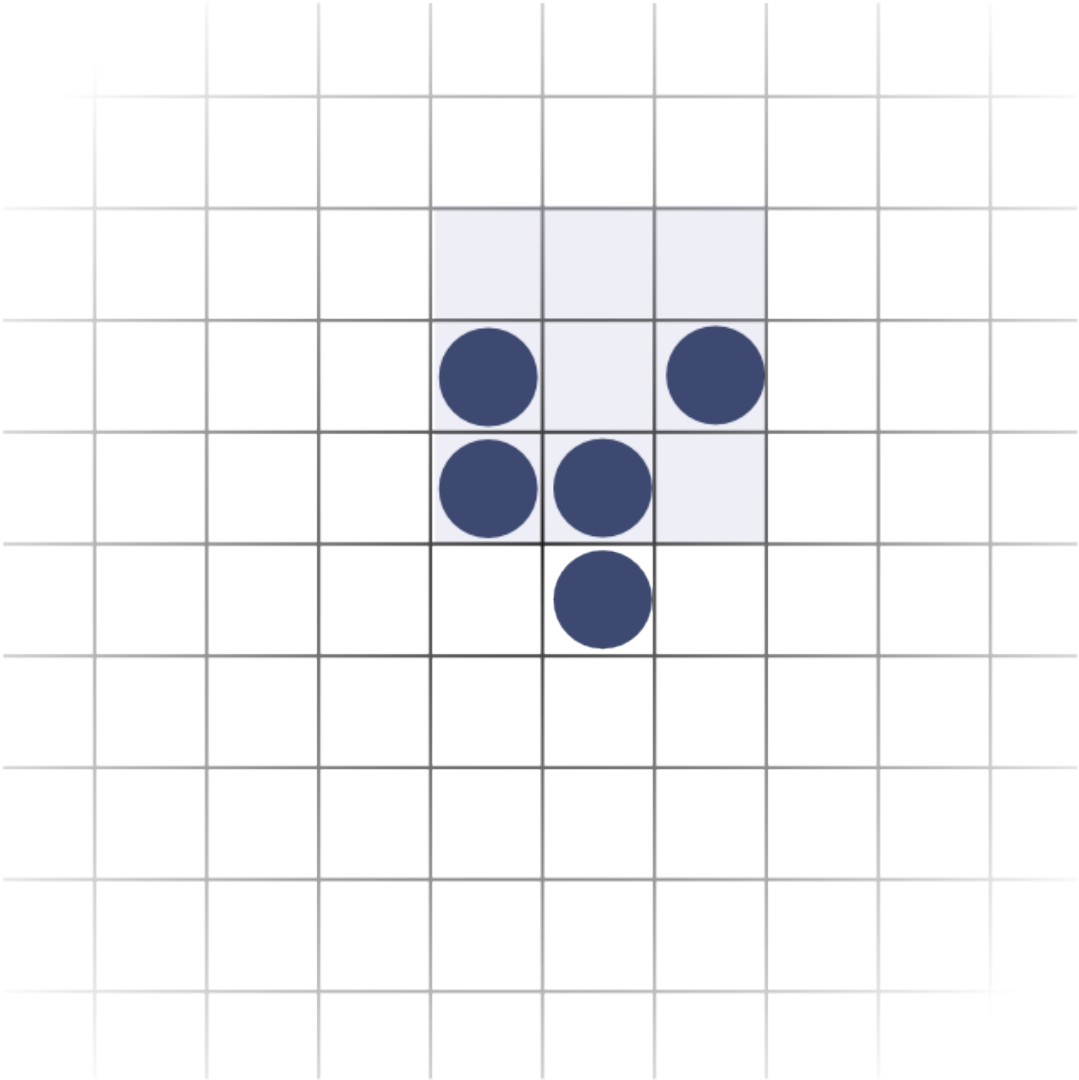
Глайдер



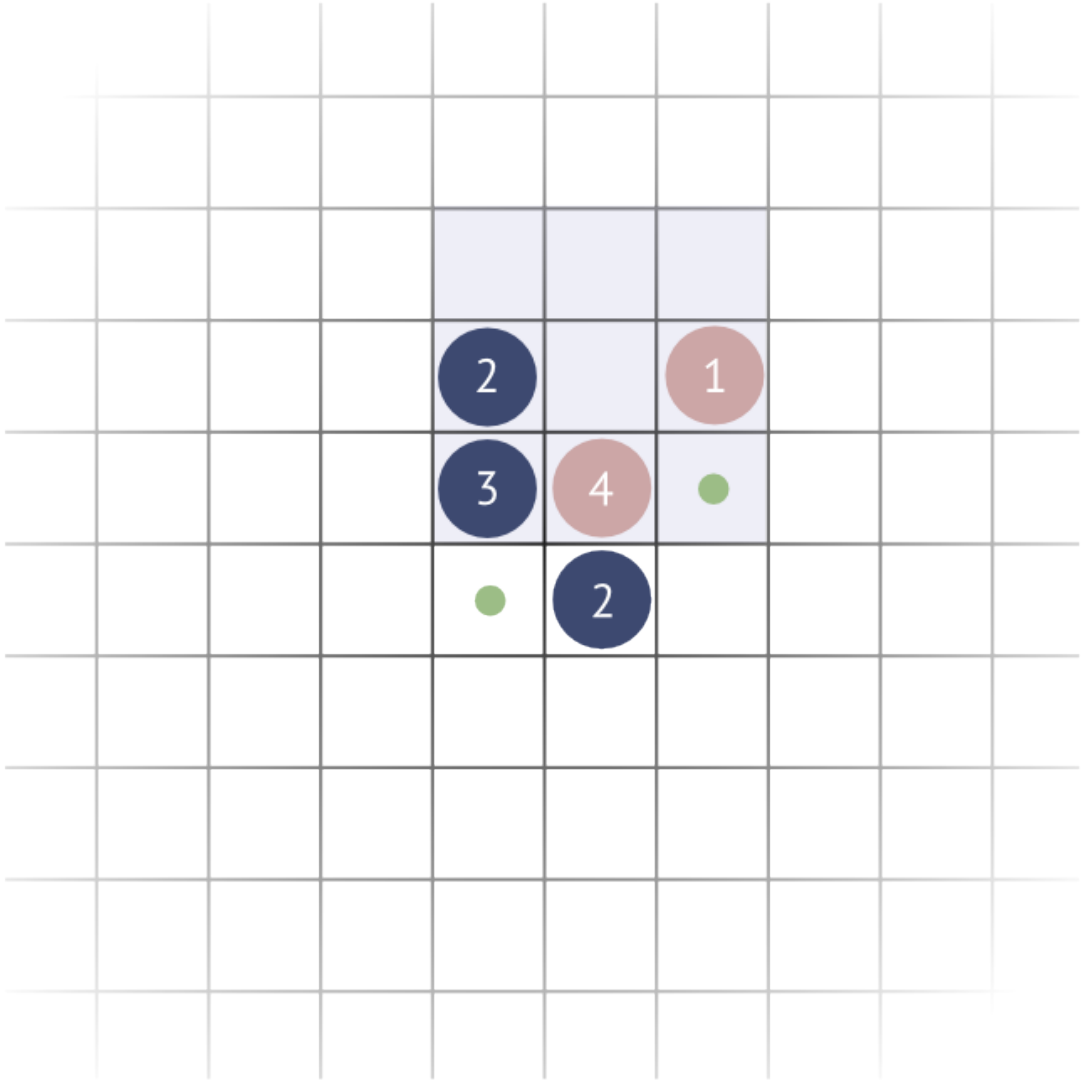
Глайдер



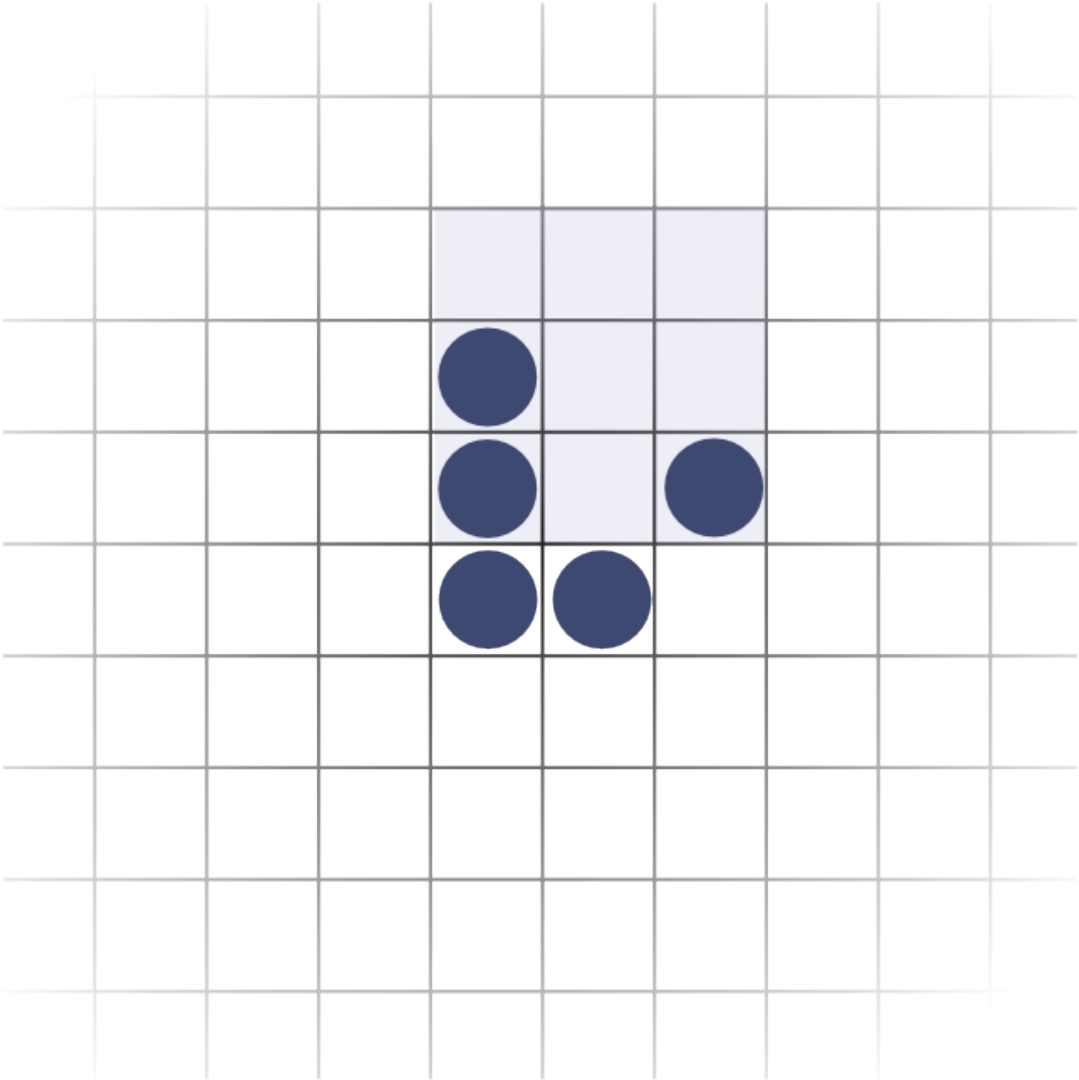
Глайдер



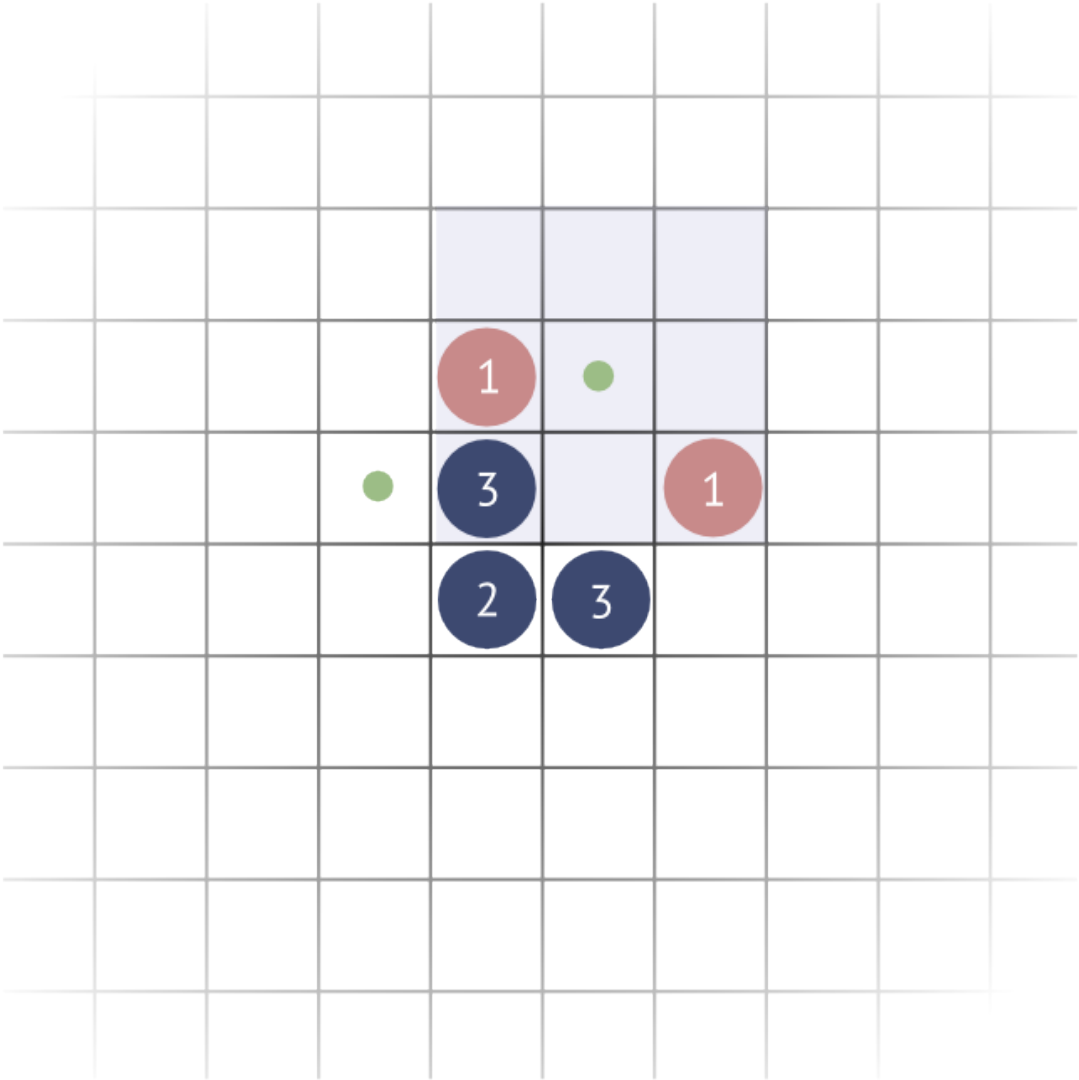
Глайдер



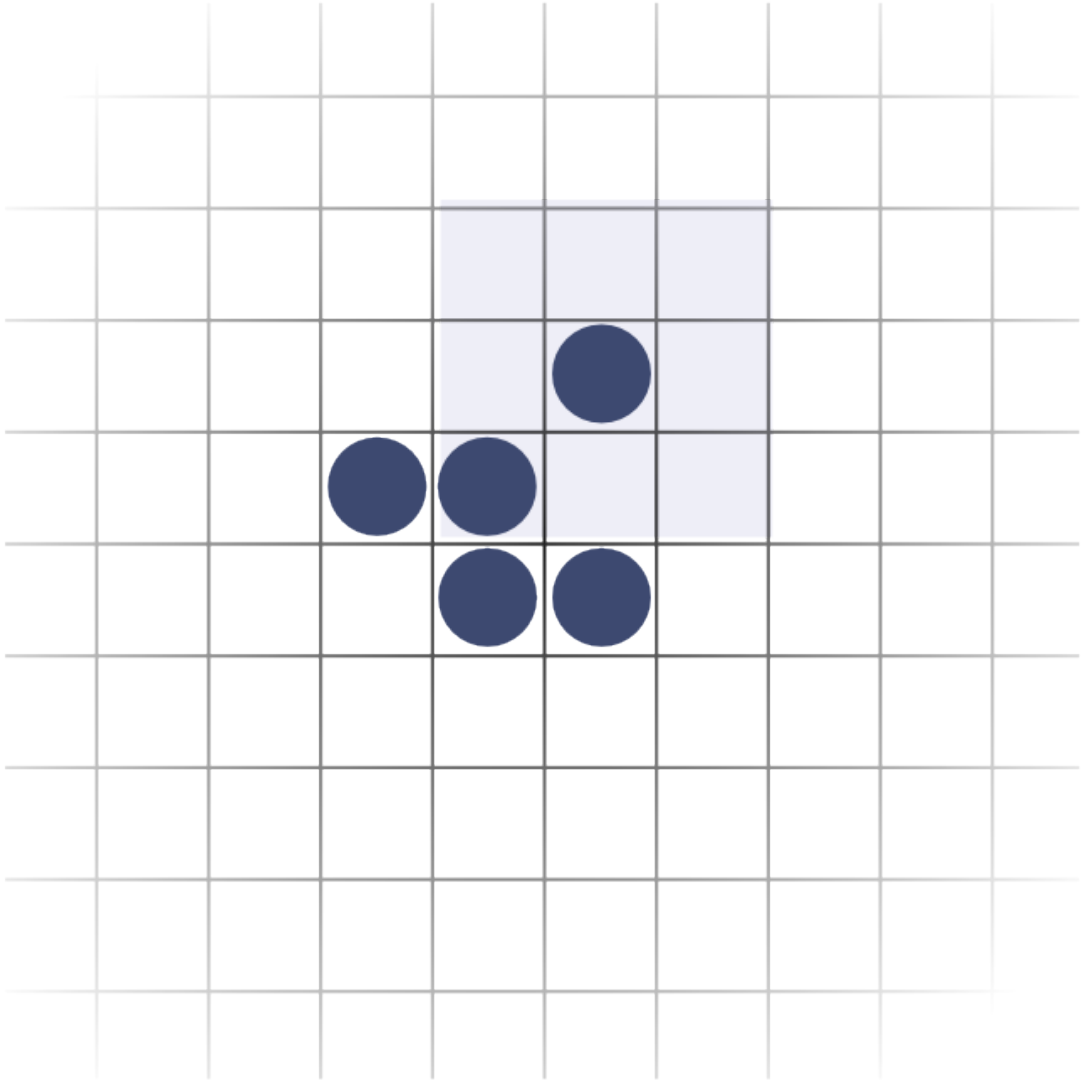
Глайдер



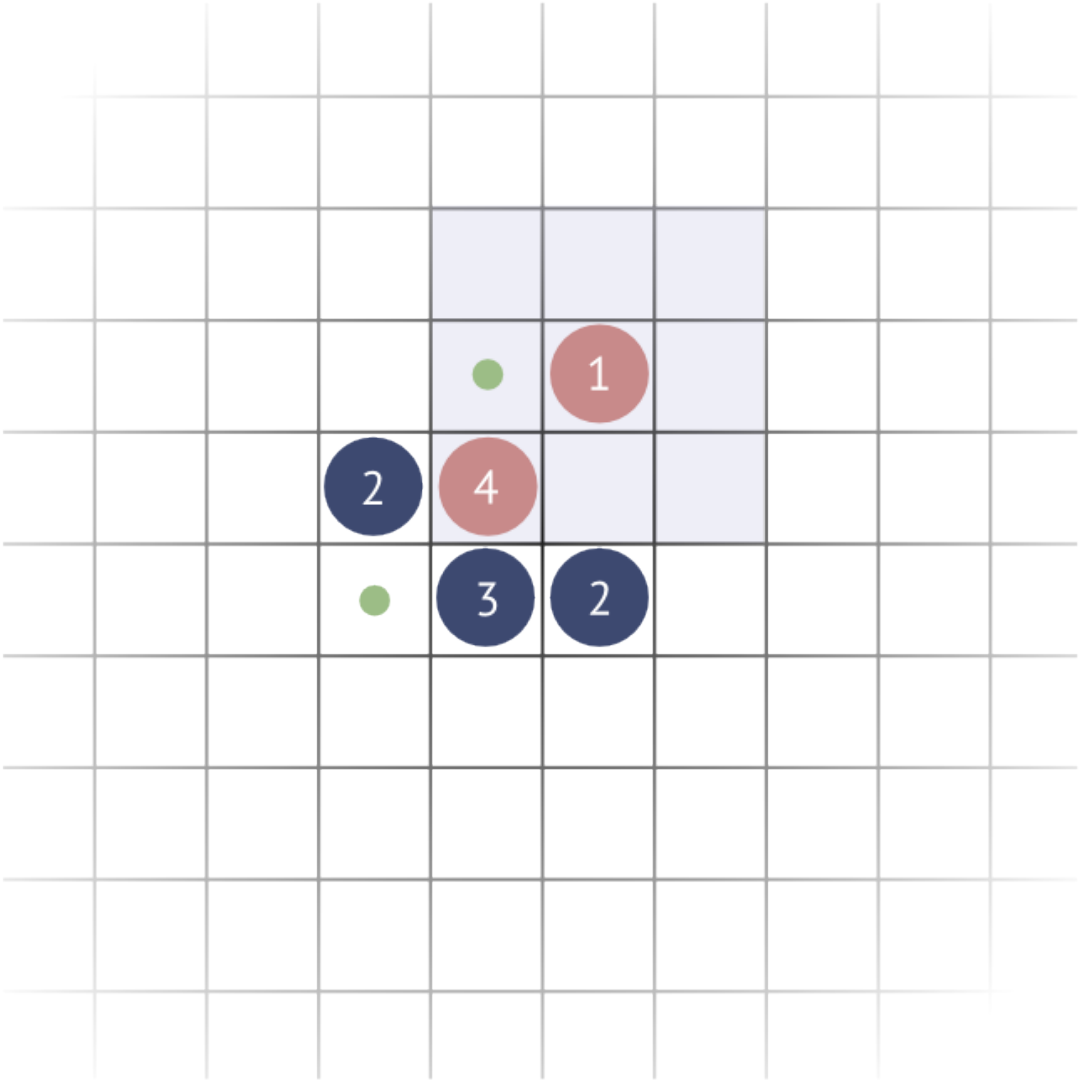
Глайдер



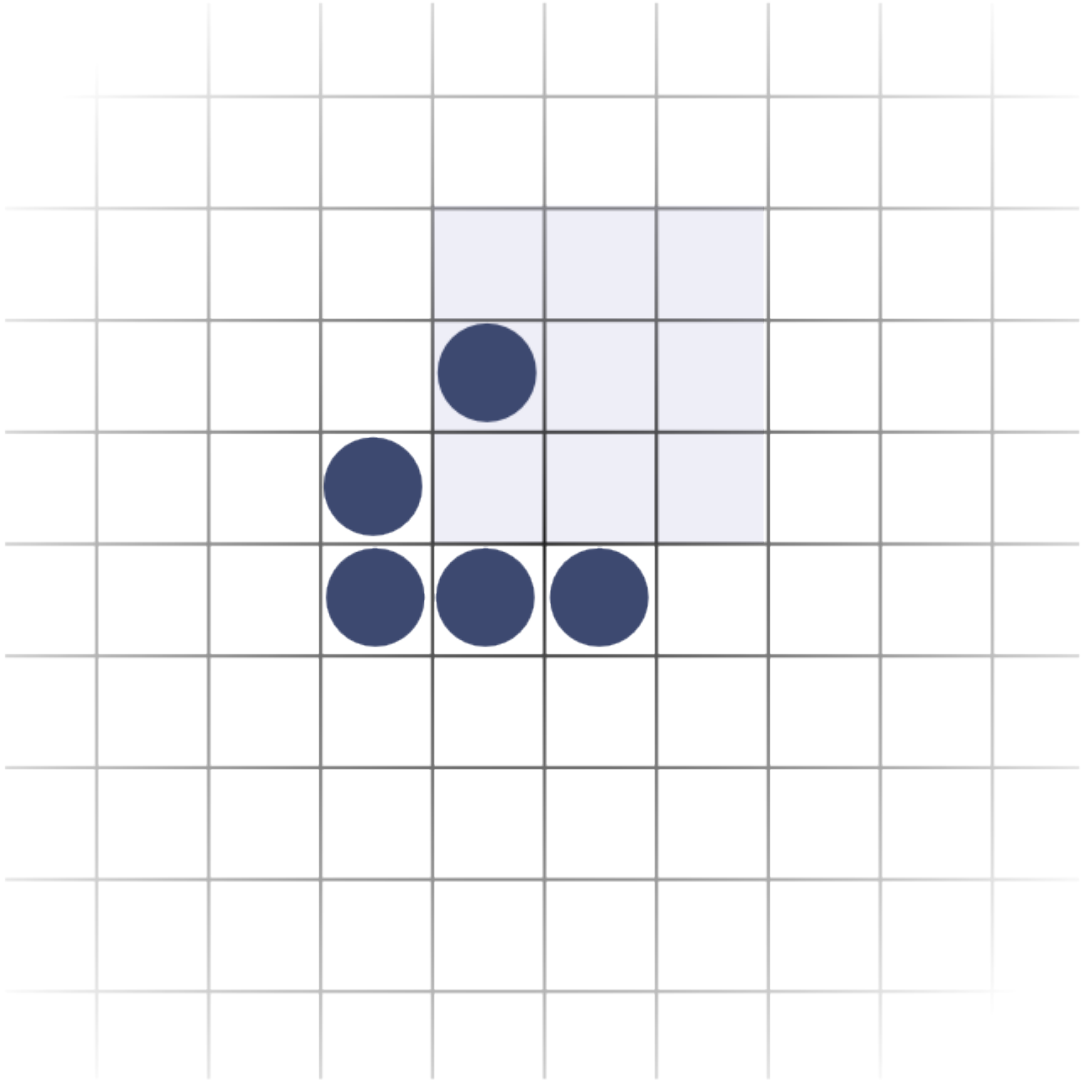
Глайдер



Глайдер

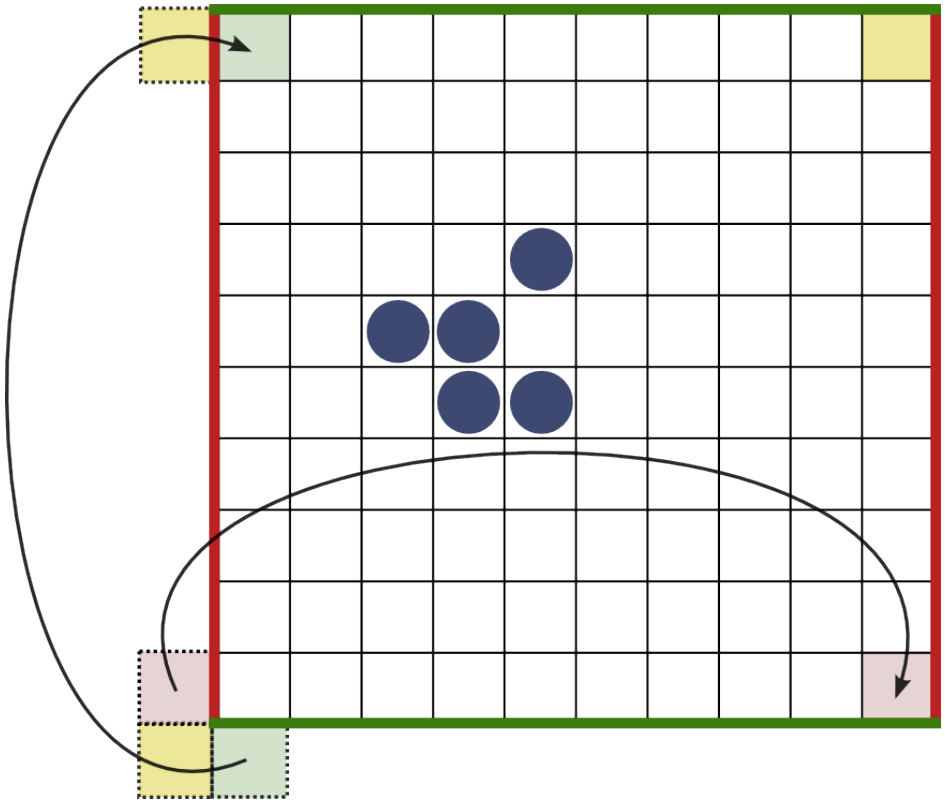


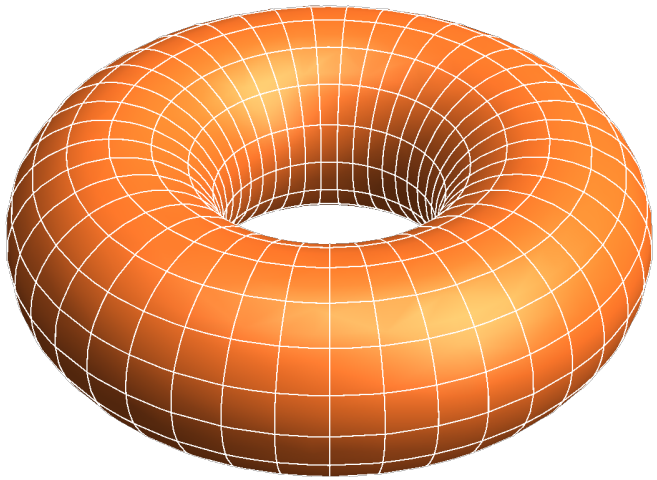
Глайдер



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`:

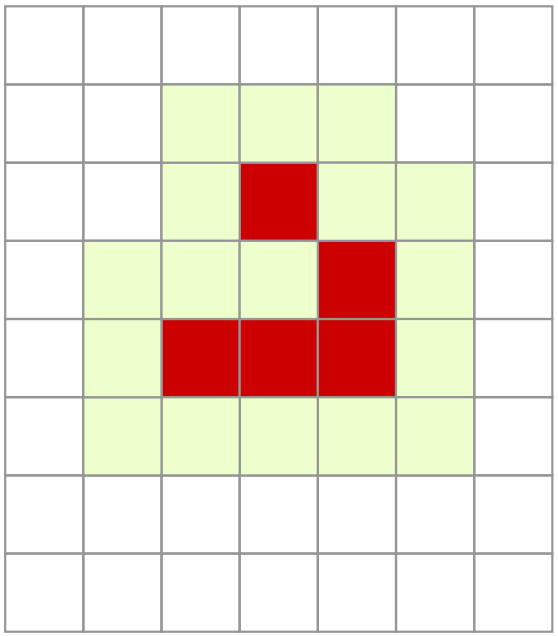
```
1 function count = count_cell_neighbours(cell, colony)
2     ...
```

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

Функция get_colony_area.m

Функция возвращает списко 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
```