

Основы MATLAB

Лекция 2

Юдинцев В. В.

Кафедра теоретической механики

Самарский университет

<http://yudintsev.info>

17 сентября 2016 г.

Содержание

- 1 Файловая система
- 2 Файл-скрипты
- 3 Файл-функции
- 4 Циклы
- 5 Операторы ветвления
- 6 Обработка исключительных ситуаций
- 7 Задачи

Файловая система

Типы файлов

- *.`m` (текстовые) содержат тексты программ, определения функций.
- *.`mat` (бинарные) содержат значения переменных.
- *.`mx` (бинарные). MEX-файлы – динамически подключаемые библиотеки

Скрипты

- Любую последовательность команд в MATLAB можно оформить в виде m файла.
- По умолчанию все переменные, объявленные внутри файл-скрипта, являются глобальными.

Функции

- Файл-функция содержит определение одной или нескольких функций.
- По-умолчанию все переменные, объявленные внутри файл-функции, являются локальными.
- Файл-функция является самостоятельным программным модулем, который связан с другими модулями и головной программой через входные и выходные параметры.

Создание т-файлов

- При создании файл-функций и файл-скриптов следует избегать перекрытия имён других функций.
- Для проверки имени можно использовать функцию `exist`:

```
1 >> exist atan  
2 ans =  
3      5
```

- Если имя не занято, то функция `exist` возвращает 0 (5 – если имя занято встроенной функцией).

Файл-скрипты

Файл-скрипт

- Файл-скрипт не имеет входных и выходных аргументов.
- Работает с данными из рабочей области.
- Все переменные, объявленные в файл-скрипте, являются глобальными.
- В процессе выполнения не компилируются.
- Представляют собой зафиксированную в виде файла последовательность операций.

Структура файла-скрипта

Файл fscript.m

```
1 % Описание кода ,  
2 % которое можно увидеть  
3 % напечатав help fscript в окне команд  
4 x=1:0.1:10;  
5 y=sin(x);
```

Выполнение файла-скрипта

- Вызов из командной строки (command window)
- Запуск из редактора при помощи сочетания **Ctrl** **Enter**
- Файл-скрипт можно для удобства разделить на ячейки (секции) при помощи удвоенного знака **%%**

```
%% секция 1
```

```
код ...
```

```
%% секция 2
```

```
код ...
```

- Код в каждой секции может быть выполнен при помощи

Ctrl**Enter**

Файл-функции

Объявление функции

File New Function

```
1 % комментарии к функции,
2 % которые будут выводится
3 % по команде
4 % help func_name
5 function [out1,out2] = func_name(in1,in2)
6 %
7 % тело функции
8 %
9 out1 = ...
10 out2 = ...
11
```

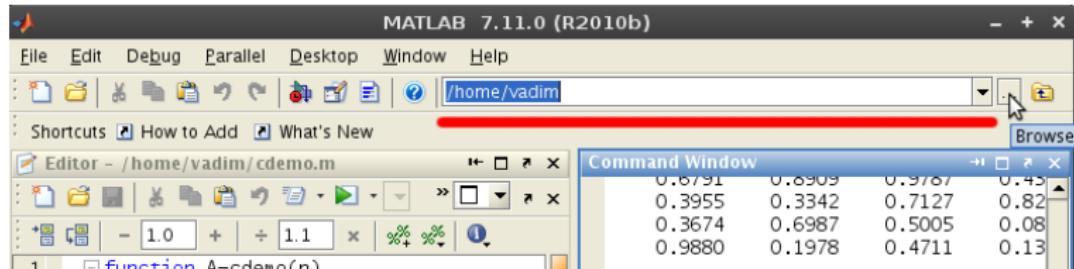
Глобальные переменные в функциях

- По умолчанию все переменные внутри функции являются локальными.
- Для того, чтобы несколько функций использовали одну переменную, её необходимо объявить глобальной.

```
1 function res = func_name(in1,in2)
2     global G;
3     res=G*in1+in2;
```

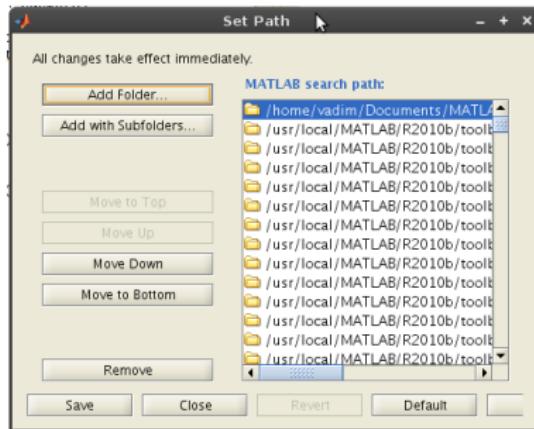
Выполнение функций и файл-скриптов

Имя файла и имя объявленной в нем функции предпочтительно делать одинаковыми. Каталог, в котором содержатся вызываемые функции, должен быть текущим или добавлен в пути поиска.



Выполнение функций и файл-скриптов

File Set Path



Оператор return

- Функция прекращает работу после выполнения последнего оператора.
- Принудительно завершить функцию можно оператором `return`.

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

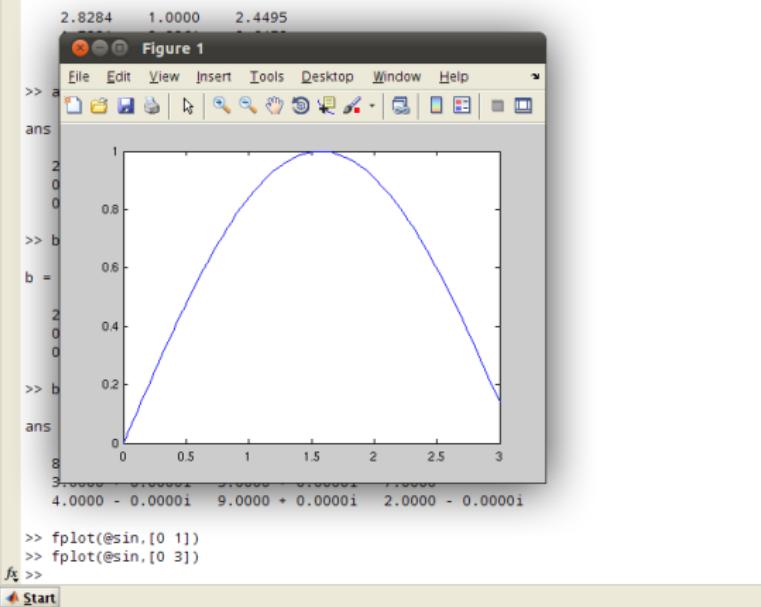
```
1 function f = myfun(x)
2 f=exp(-x)*sqrt((x^2+1)/(x^4+0.1))
```

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])`

Быстрое построение графика функции



Функция `fplot`

```

1 fplot('myfun',[0 4])
2 fplot(@myfun,[0 4])

```

первый аргумент: имя или ссылка на функцию, второй аргумент: диапазон изменения аргумента функции для построения графика

Внутренние функции

- Файл-функция вместе с определением основной функции может содержать определения вспомогательных функций, доступных к вызову только из основной функции.

```
1 function f = myfun(x)
2     f1=infun(x);
3     f=f1+cos(x);
4 % Внутренняя функция
5 function f = infun(x)
6     a=3;
7     f=sin(x^3);
```

- Переменные, используемые в подфункциях **локальные**.

Вложенные функции

- Вложенная функции определяется в теле основной функции.
Файл `myfun.m`:

```
1 function f = myfun(x)
2     f1=infun(x);
3     f=f1+cos(x);
4     function f = infun(x)
5         f=sin(x);
6     end
7 end
```

- Из вложенной функции `infun` доступны локальные переменные всех функций верхнего уровня и наоборот.

Функции с переменным количеством аргументов

- `varargin` – список ячеек с параметрами функции;
- `length(varargin)` – количество переданных аргумента;
- `varargin{1}` – первый аргумент;
- `varargin{2}` – второй аргумент.

```
1 function res = fun(varargin)
2     if length(varargin)<2
3         error('Недостаточное количество аргументов');
4     end
5     ...
```

Функции с переменным количеством аргументов

Аргумент `varargin` может быть указан после перечня обязательных аргументов

```
1 function res = fun(a1,a2,varargin)
2     if length(varargin)<4
3         error('Недостаточное количество аргументов');
4     end
5     ...
```

Функция в качестве аргумента

Использование функции `feval`

Передача имени функции строкой:

```
1 >> x = 1;
2 >> feval('sin',x)
3 ans =
4     0.8415
```

Передача ссылки на функцию (это работает быстрее):

```
1 >> feval(@sin,x)
2 ans =
3     0.8415
```

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	$x \sim= y$
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.

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
```

Функция в качестве аргумента

```
1 >> x = 1;
2 >> feval('sin',x)
3 ans =
4 0.8415
```

```
1 >> feval(@sin,x)
2 ans =
3 0.8415
```

Обработка исключительных ситуаций

Операторы try, catch

```
1 try
2     C = [A; B];
3 catch err
4     if (strcmp(err.identifier, 'MATLAB:catenate:
5         dimensionMismatch'))
6         msg = 'Текст сообщения';
7         error('MATLAB:myCode:dimensions', msg);
8     else
9         rethrow(err);
10    end
11 end
```

ME.**identifier** – идентификатор сообщения об ошибке.
ME.**message** – текст сообщения об ошибке.

Предупреждения (warning)

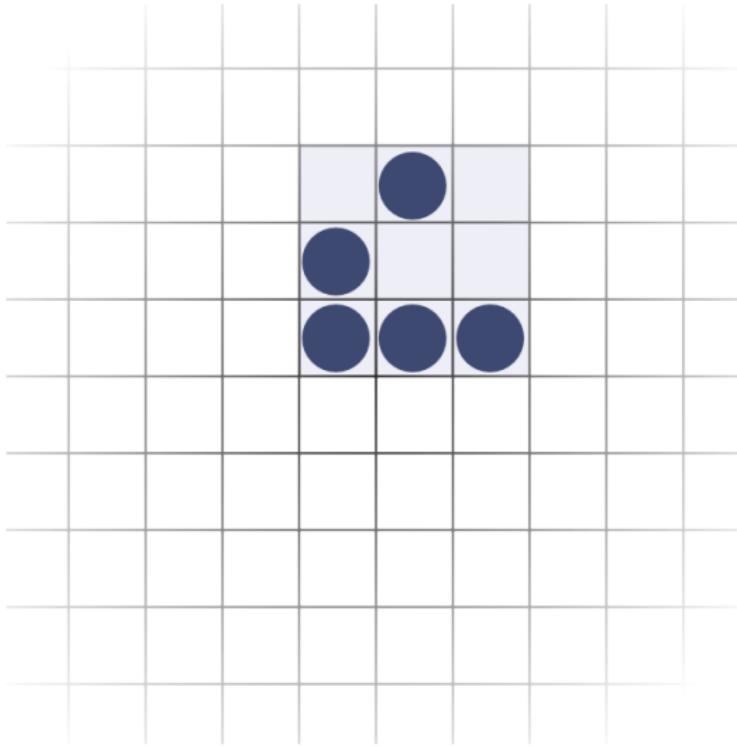
```
1 function res = fun(a, b)
2
3 if a<0
4     warning('a должен быть больше нуля! a=%s', ... num2str(a));
5 else
6     ...
7 end
```

Задачи

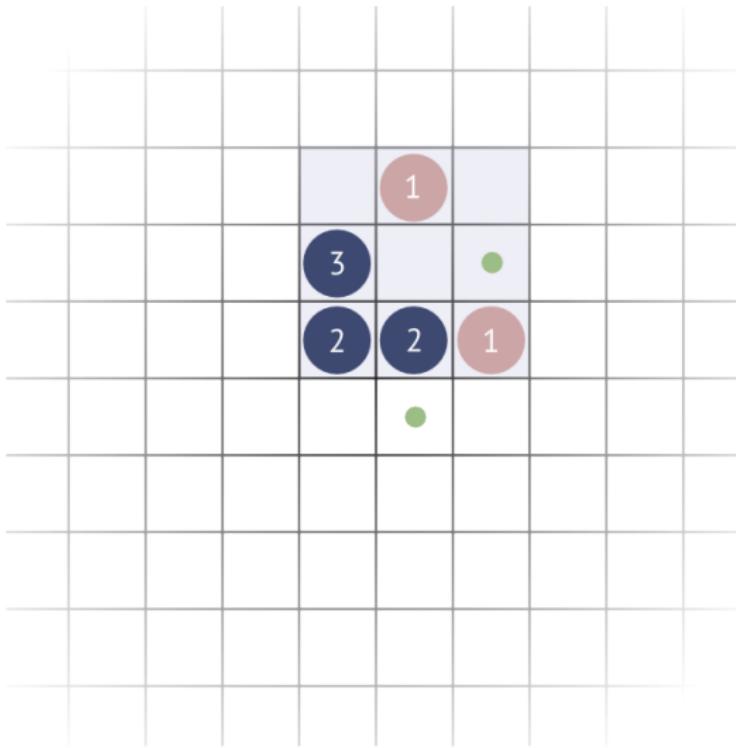
Игра “Жизнь” (автор Дж. Конвей)

- 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.

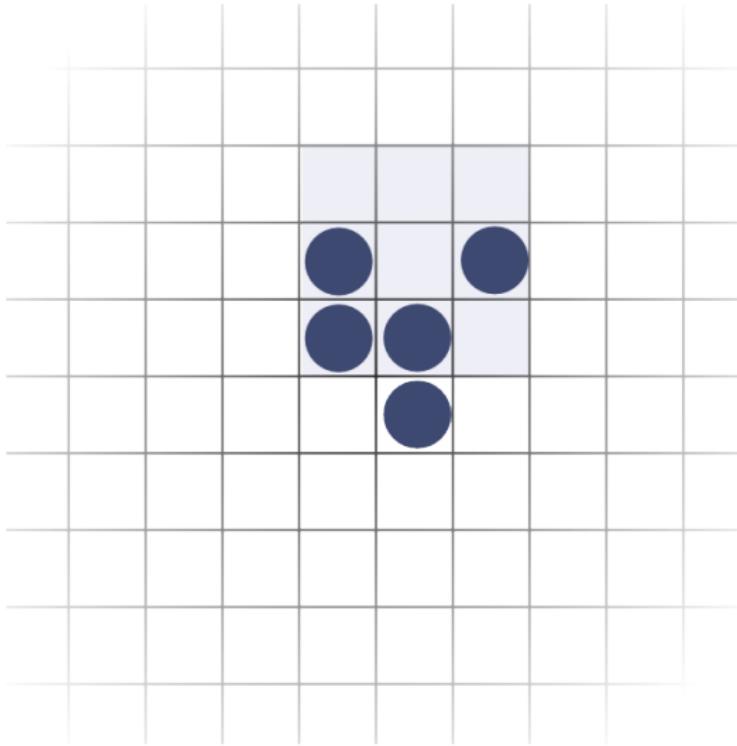
Глайдер



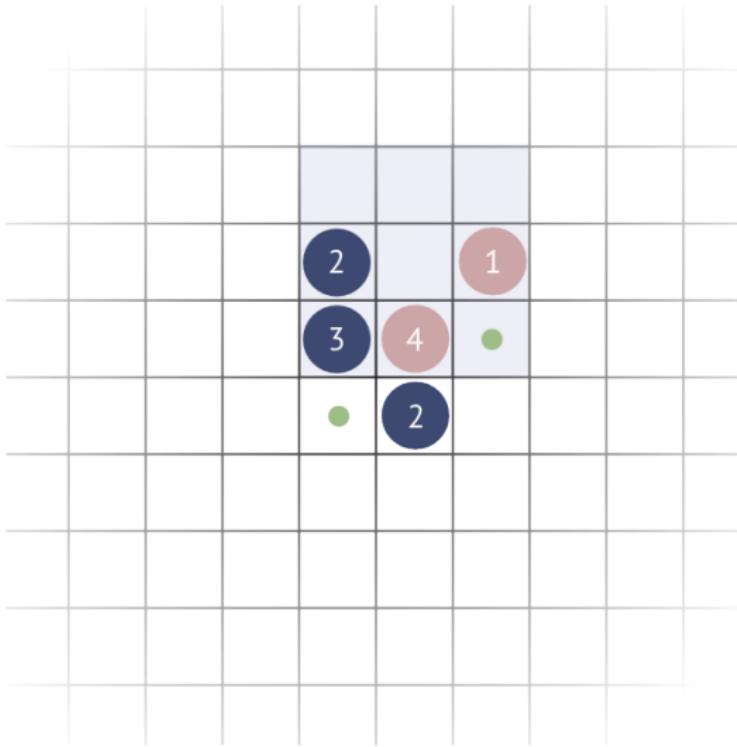
Глайдер



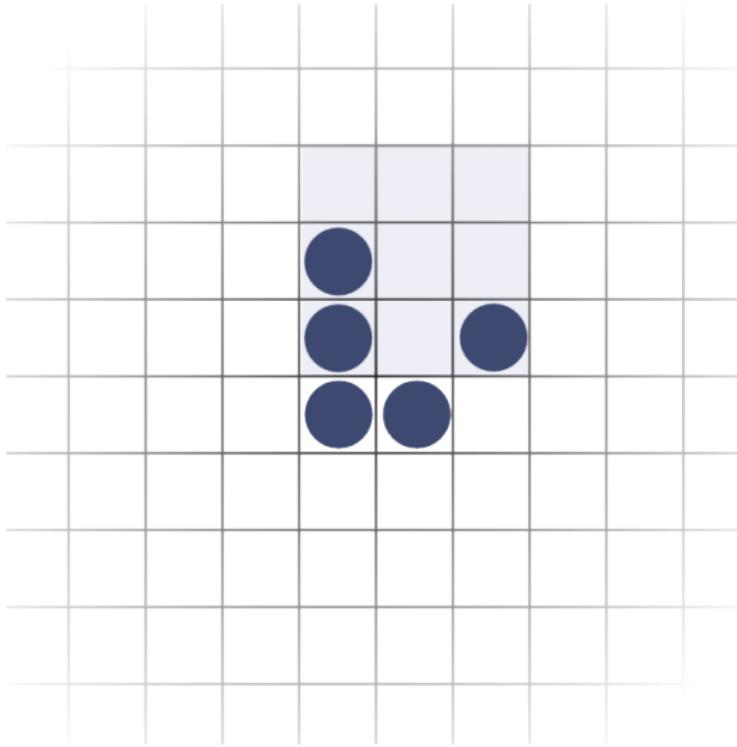
Глайдер



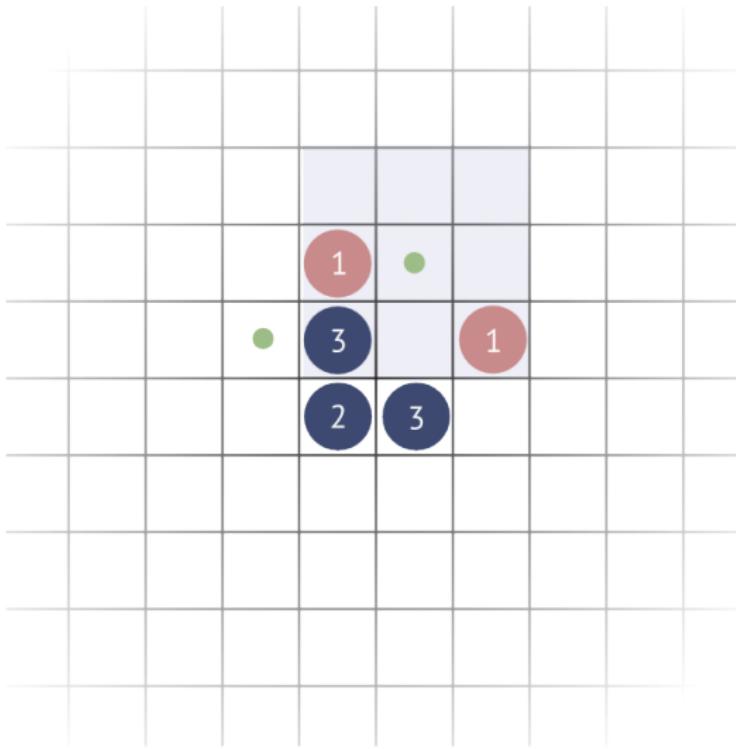
Глайдер



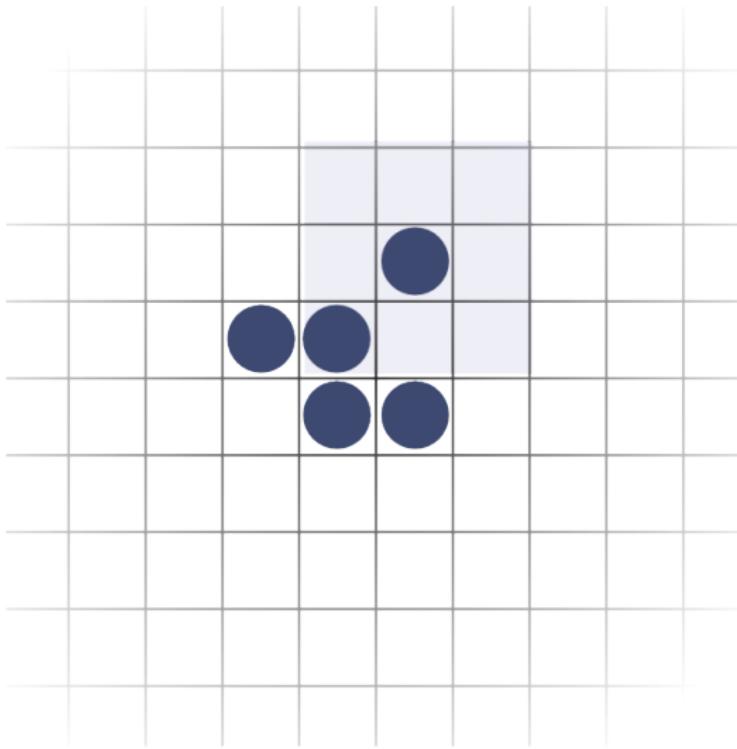
Глайдер



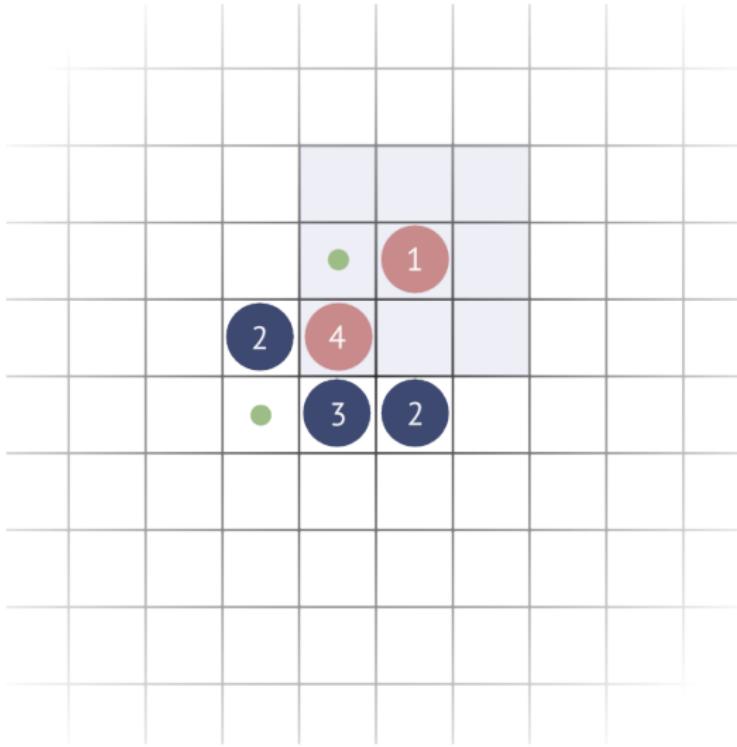
Глайдер



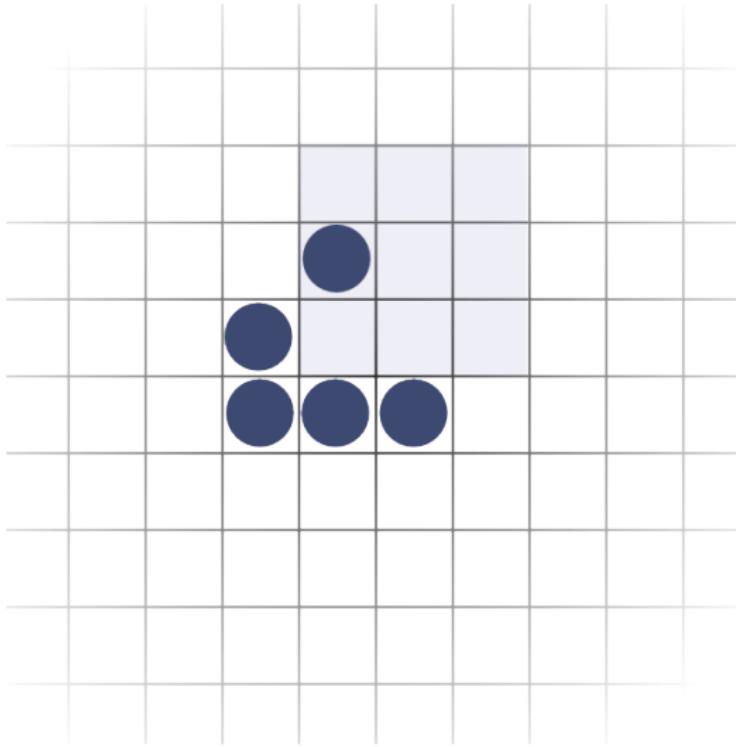
Глайдер



Глайдер



Глайдер



Задание

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

