Изучаем новые структуры данных

Списки

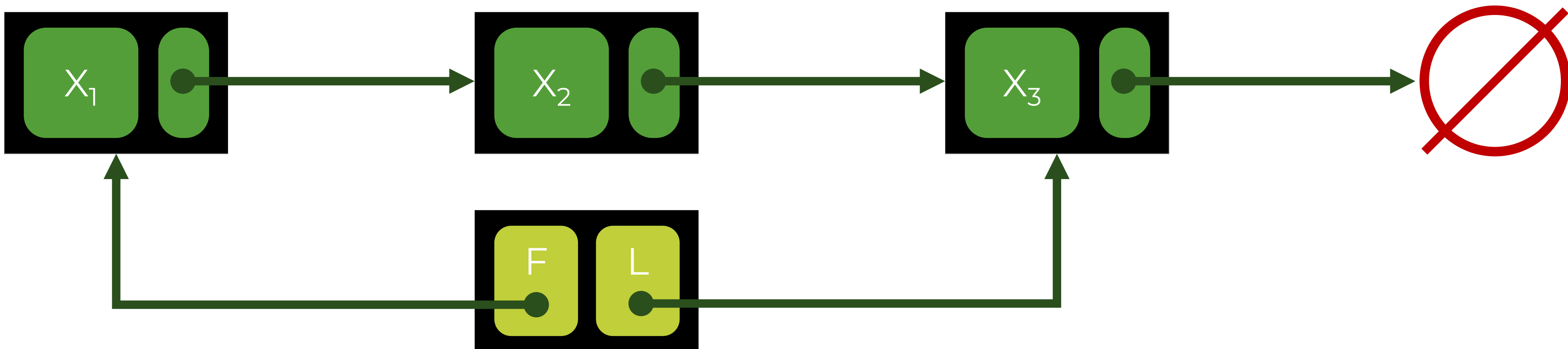
Что у нас сегодня?

1. Связный список
2. Двусвязный список
3. Стек
4. Очередь
5. Циклический массив
6. Замена рекурсии стеком
7. Разбор формул

Перейдем к
структурам данных

Связный список

Связный список — это структура данных, состоящая из узлов, каждый из которых содержит данные и ссылку («связку») на следующий узел списка.



Операции на связном списке

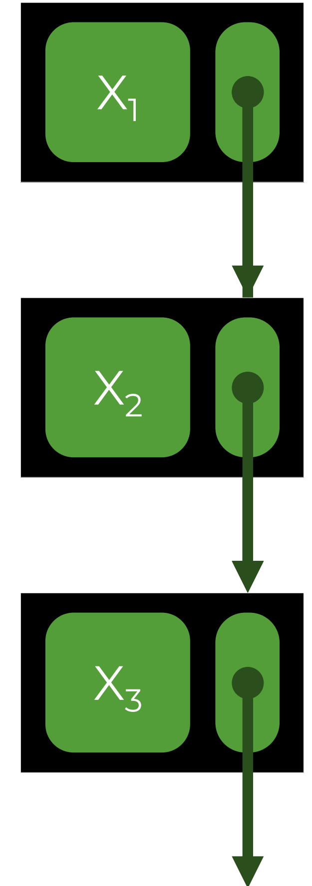
Операция	Массив	Дин. массив	Связный список
Индексация	O(1)	O(1)	O(N)
Вставка (append)	O(N)	O(1)*	O(1)
Вставка (insert)	O(N)	O(N)	**O(1) → | ← O(N)**
Удаление (pop_0)	O(N)	O(N)	O(1)
Удаление (pop_x)	O(N)	O(N)	**O(N)**
Вставка (индекс)	O(N)	O(N)	O(N)
Поиск	O(N)	O(N)	O(N)

Вставка в связный список

Асимптотика зависит от того, известен ли вам элемент, после которого вы вставляете новый, или нет.

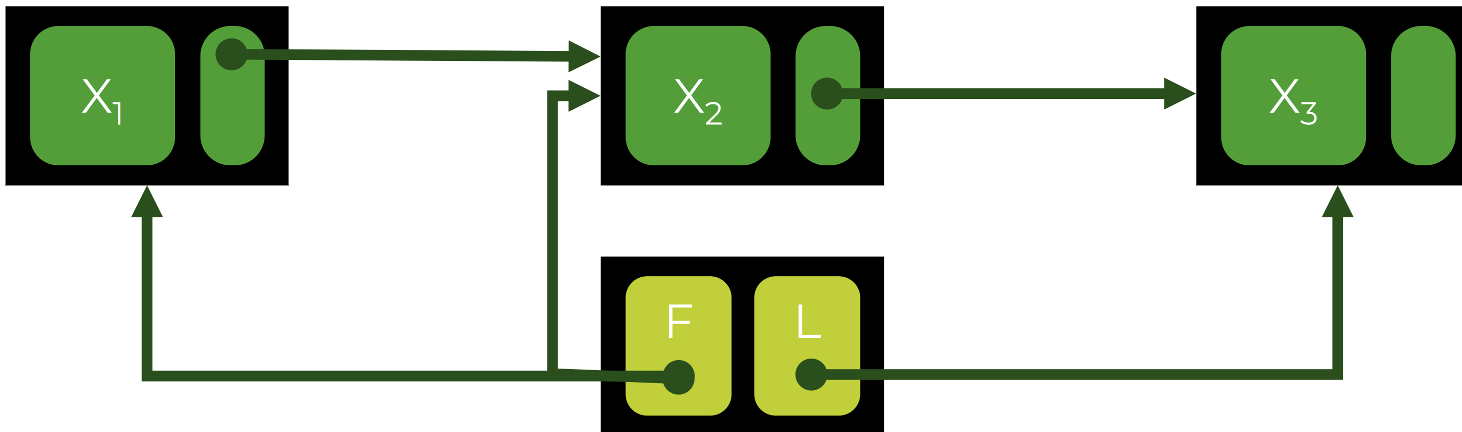
Если элемент **известен**, то достаточно создать новый узел и переопределить две связи. Асимптотика – **O(1)**.

Если нет, то предыдущий узел необходимо искать с первого элемента («головы») связного списка. Асимптотика – **O(N)** в общем случае.



Добавление в начало

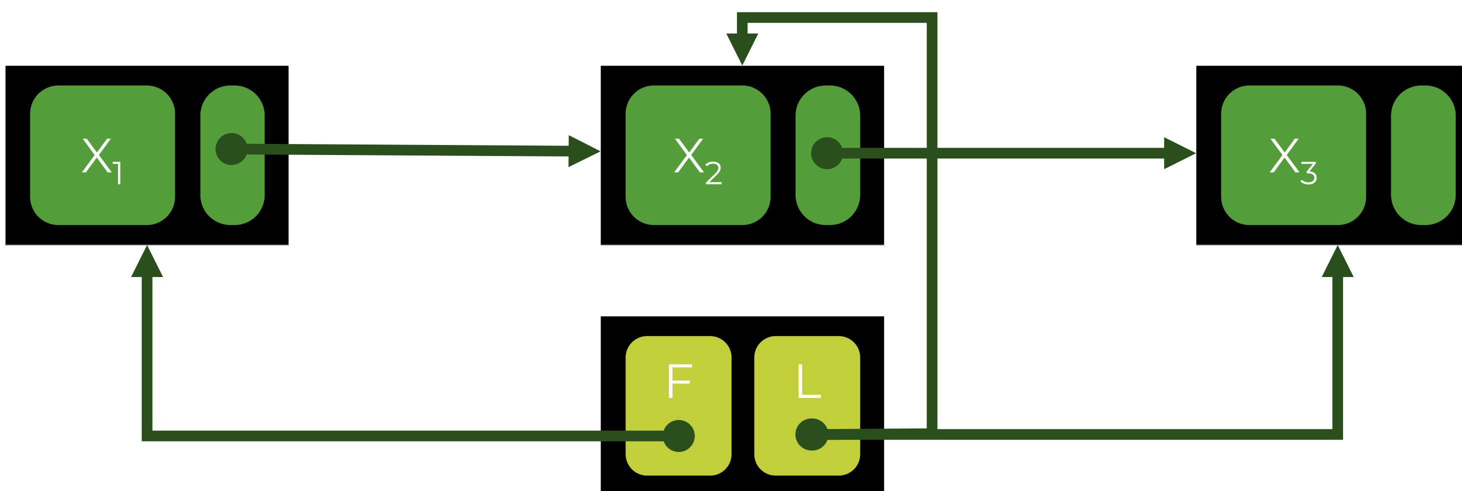
1. Создаем новый узел.
2. Создаем ссылку на старый первый элемент из нового первого элемента.
3. Меняем ссылку на первый элемент.



O(1)

Удаление из конца

1. Меняем ссылку на последний элемент.
2. Меняем указатель нового последнего элемента на **Null**.



Почему O(N)?

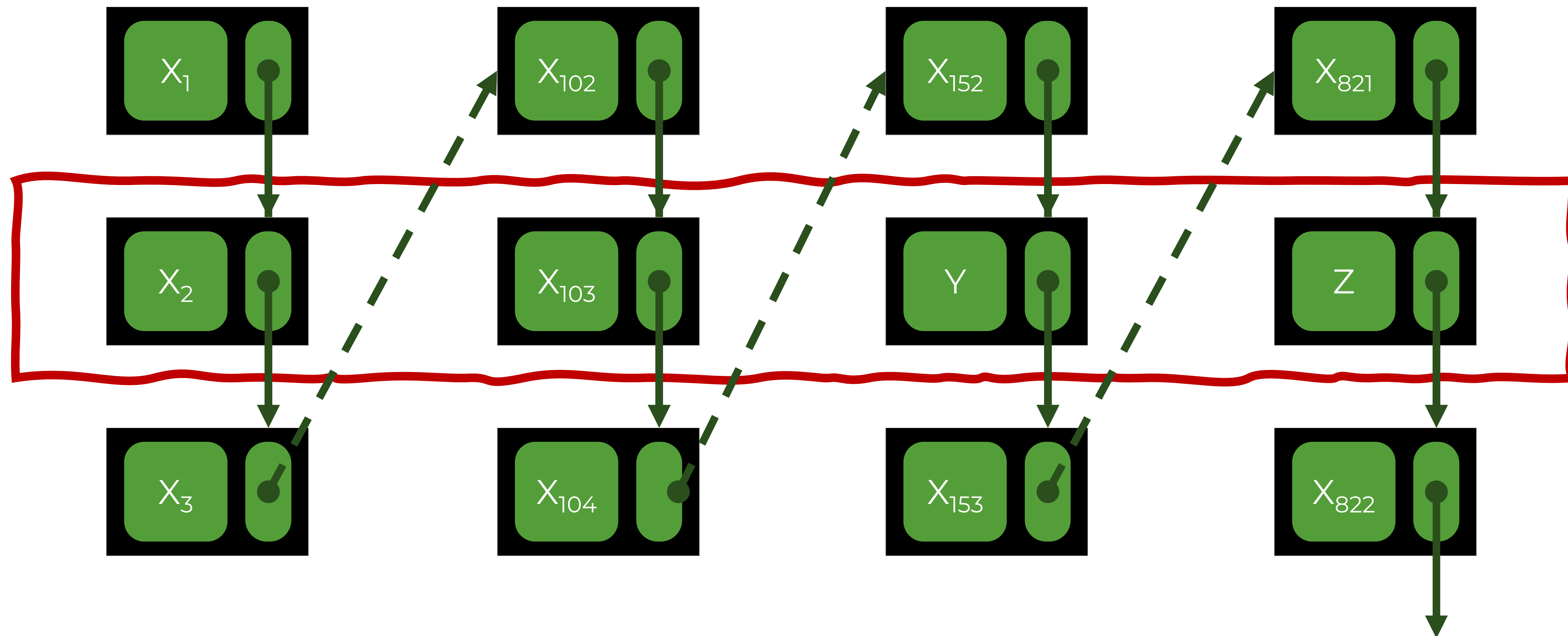
В чем преимущества связного списка?

Не требует фиксированной памяти

Параллелизуемый

Быстро добавляет и удаляет элементы с концов

Параллелизация



Уже спешим использовать вместо массивов?

Хаотично расположен в памяти

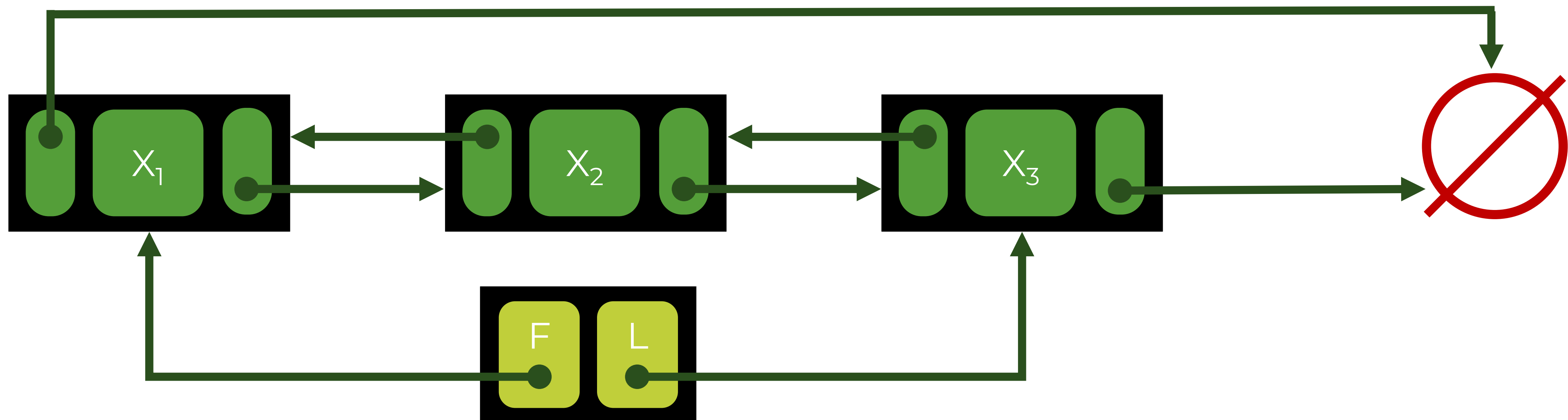
Трудно искать элементы

Индексировать тоже трудно

Пишем псевдокод!

Двусвязный список

Двусвязный список — это структура данных, состоящая из узлов, каждый из которых содержит данные и ссылку («связку») на следующий **и предыдущий** узлы списка.



Операции на двусвязном списке

Операция	Массив	Дин. массив	Двусвязный список
Индексация	O(1)	O(1)	O(N)
Вставка (append)	O(N)	O(1)*	O(1)
Вставка (insert)	O(N)	O(N)	**O(1)**
Удаление (pop_0)	O(N)	O(N)	O(1)
Удаление (pop_x)	O(N)	O(N)	**O(1)**
Вставка (индекс)	O(N)	O(N)	O(N)
Поиск	O(N)	O(N)	O(N)

Зачем нам нужен двусвязный список?

Если мы часто удаляем элементы не с начала

Если мы зачем-то часто смотрим на элементы, близкие к концу

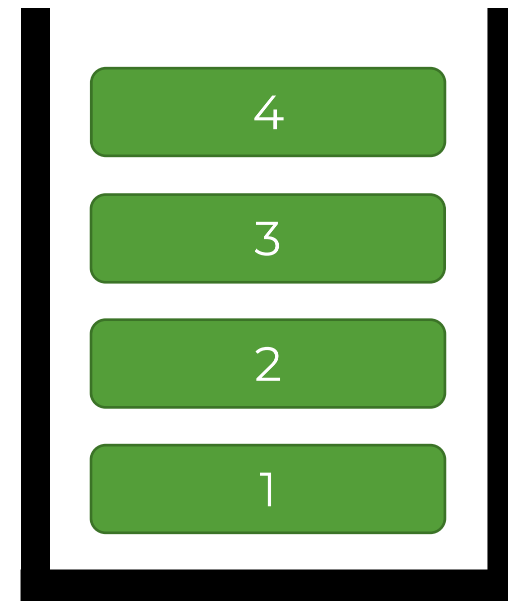
Пишем псевдокод!

стек overflow

Стек (stack)

Стек – это **абстрактная** структура данных, которая представляет собой набор элементов, упорядоченных по принципу **LIFO**.

Last **I**n – **F**irst **O**ut.



Какие операции нужны для стека?

Push O(1)

Pop O(1)

Peek?

Count?

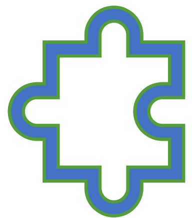
Isempty?

На какой структуре можно организовать стек?

Дин. массив

Связный список

Двусвязный список



Пишем псевдокод!

Зачем вообще нужен этот ваш стек?

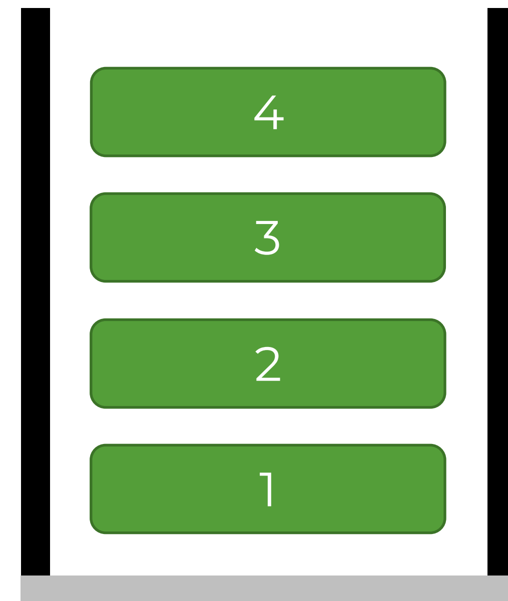
Вызов рекурсивных функций часто удобно осуществлять с помощью стека

Кроме того, он часто используется для обхода других, более сложных структур данных, например, дерево или граф

Очередь (queue)

Очередь – это **абстрактная** структура данных, которая представляет собой набор элементов, упорядоченных по принципу **FIFO**.

F*irst* **I**n – **F**irst **O**ut.



Какие операции нужны для очереди?

Enqueue O(1)

Dequeue O(1)

Peek?

Count?

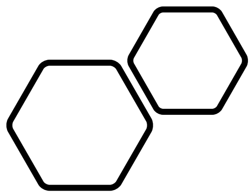
Isempty?

На какой структуре можно организовать очередь?

Связный список

Стек

Цикл. массив



Пишем псевдокод!

Зачем нужна очередь?

Когда нужно совершить какие-то действия в порядке их поступления.

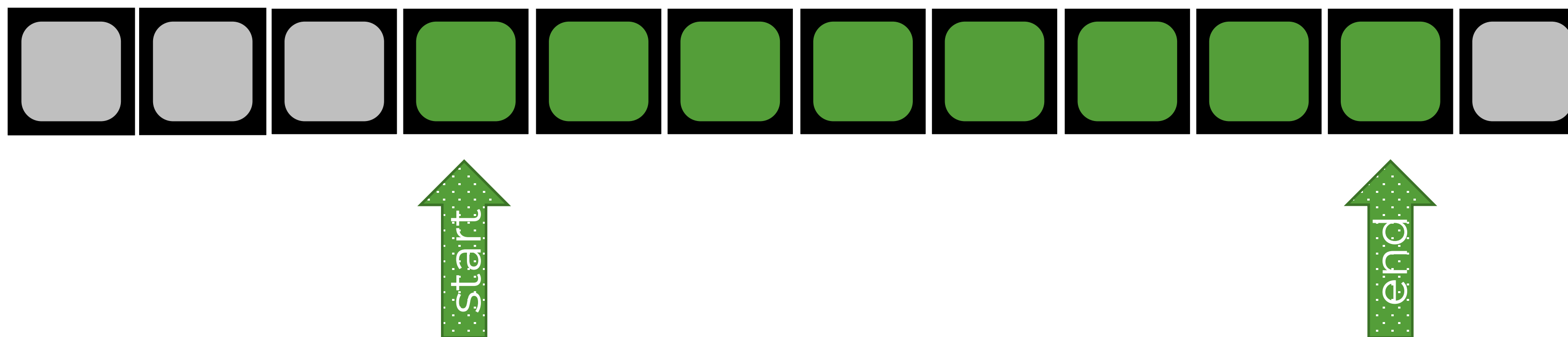
Кроме того, он часто используется для обхода других, более сложных структур данных, например, дерево или граф

Реализация очереди на стеке

Нам понадобится не один стек, а целых два!

Реализация очереди на массиве

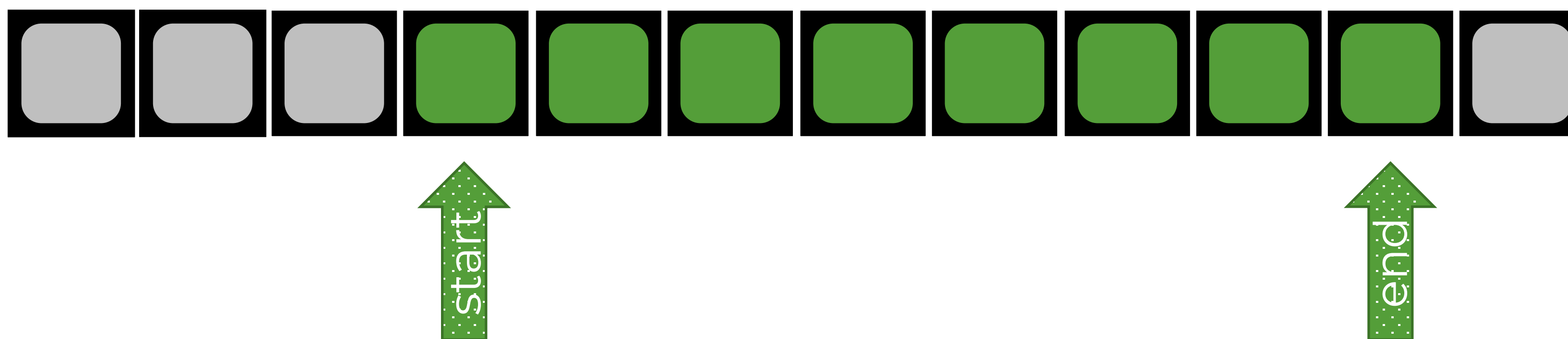
В реализации на массиве мы храним два указателя: **start** и **end**.



Реализация очереди на массиве

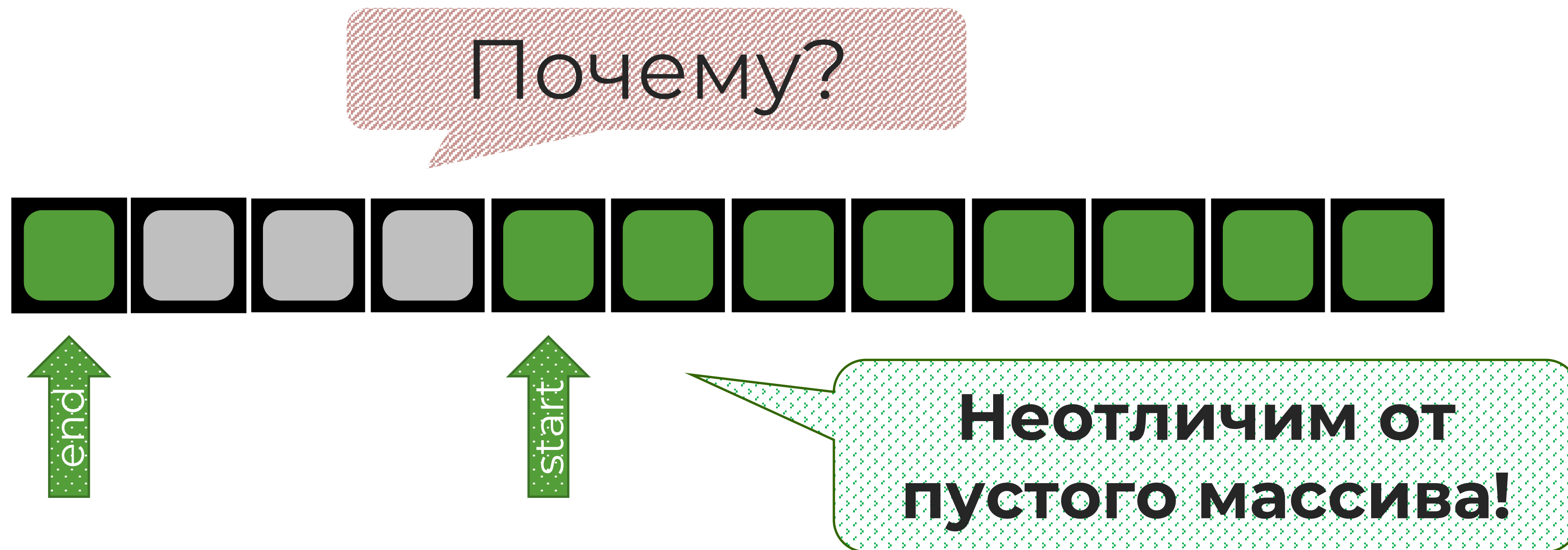
При удалении элемента из очереди в **Q[start]** записывается новый элемент очереди и **start** увеличивается на 1.

При добавлении **end** увеличивается на 1 и элемент записывается в **Q[end]**.



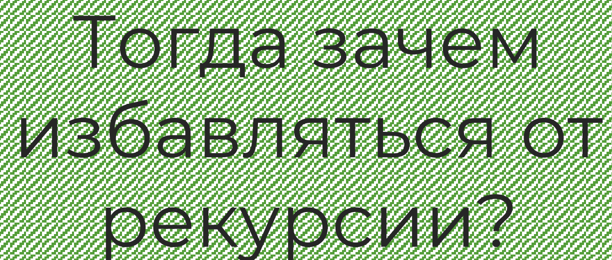
Циклический массив

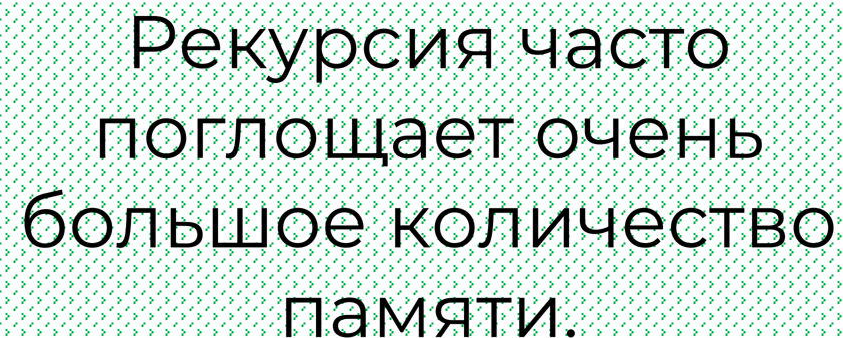
С таким алгоритмом одна из N ячеек всегда будет незанятой.



Замена рекурсии

Использовать рекурсию при написании кода очень удобно. Часто она помогает сделать код более легким для восприятия.





Хвостовая рекурсия

Компиляторы кода обычно умеют заменять рекурсию на цикл в случае хвостовой рекурсии. В некоторых компиляторах это гарантируется.

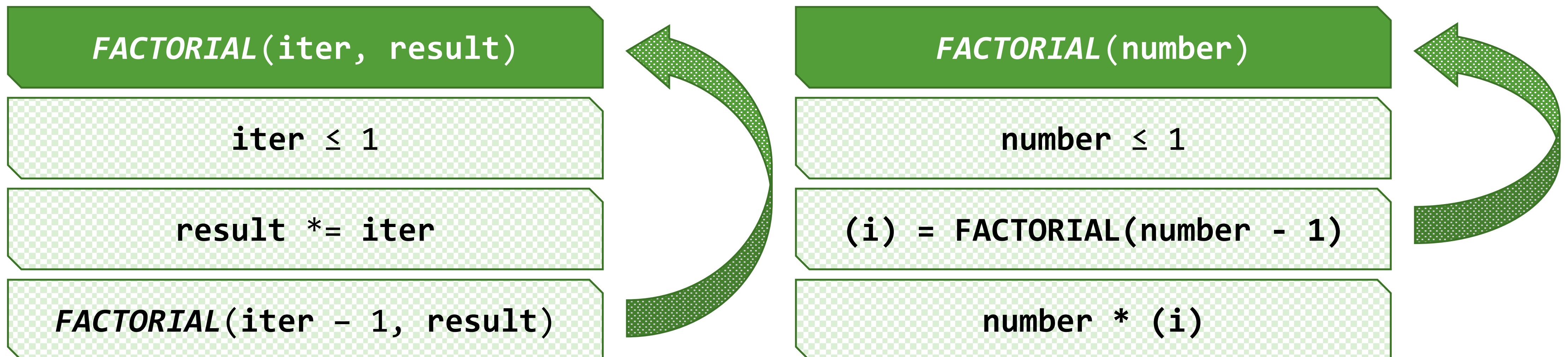
Что такое хвостовая рекурсия?

Случай рекурсии, когда любой рекурсивный вызов – **последняя** операция перед возвратом.

Примеры хвостовой рекурсии

```
int FACTORIAL(int iter, int result):
    if iter ≤ 1:
        return result;
    result *= iter
    return FACTORIAL(iter – 1, result);
```

```
int FACTORIAL(int number):
    if number ≤ 1:
        return 1;
    return number * FACTORIAL(number - 1);
```



Системный стек

Разберем на примере!

```
int FACTORIAL(int number):
    if number ≤ 1:
        return 1;
    return number * FACTORIAL(number - 1);
```

FACTORIAL(3) = 6

returns

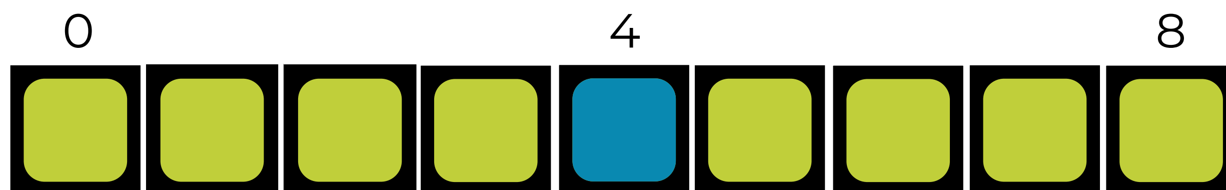
```
FACTORIAL()
number = 1
return 1
```

```
FACTORIAL()
number = 2
return 2 * 1 = 2
```

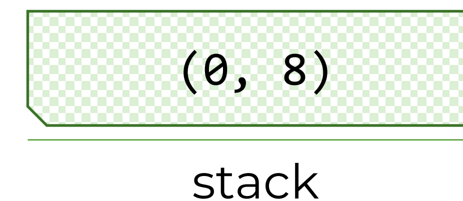
```
FACTORIAL()
number = 3
return 3 * 2 = 6
```

Используем стек вместо рекурсии

Рассмотрим на примере быстрой сортировки:

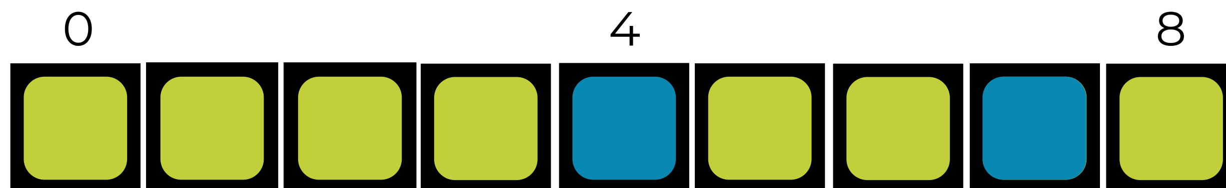


После нахождения положения опорного элемента нужно осуществить два вызова функции.

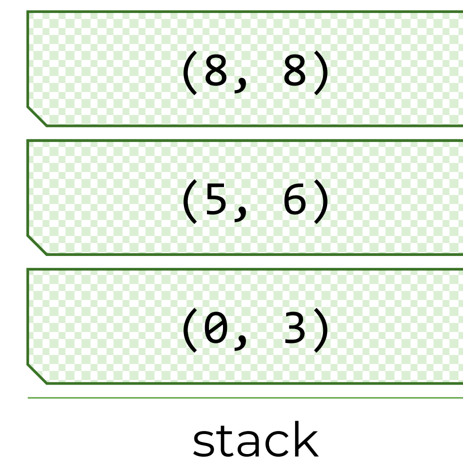


Используем стек вместо рекурсии

Кладем будущие вызовы в стек!

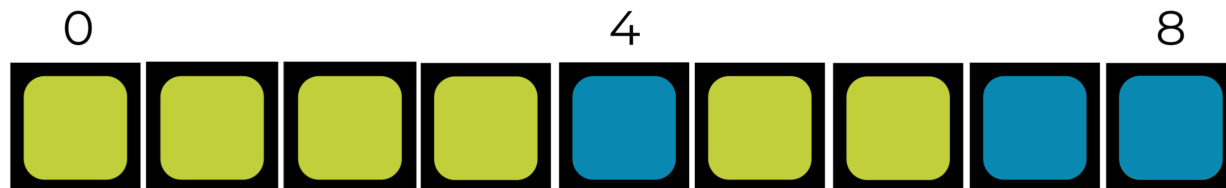


Если стек не пустой, продолжаем сначала.

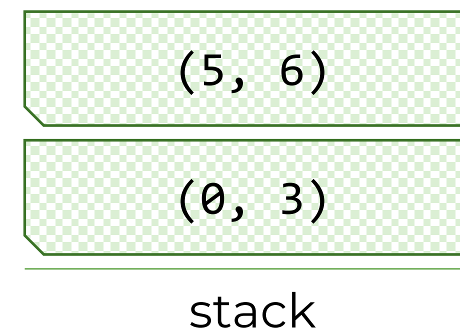


Используем стек вместо рекурсии

Короткие вызовы не пополняют стек, поэтому мы его опустошим.

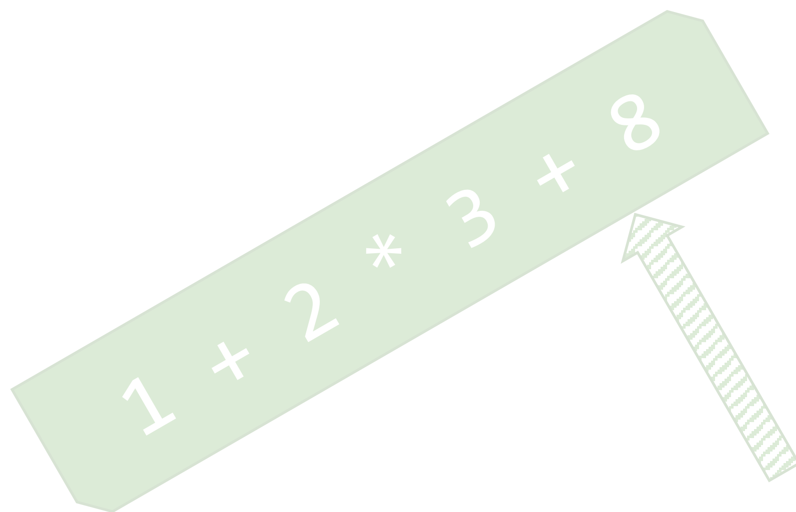


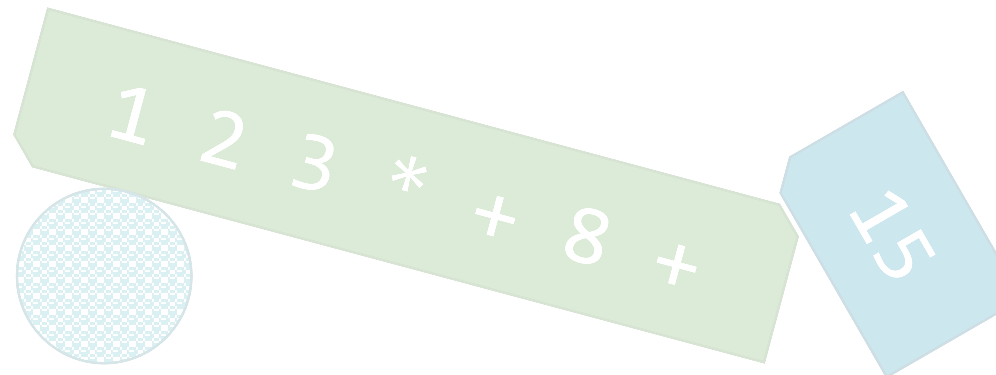
Вспомним, почему быстрая сортировка занимает O(logN) памяти в среднем случае



Задачи со стеком

Стек применяется во множестве других задач. Одна из классических задач – **парсинг формул**.





Обратная польская запись

Обычно при записи арифметических операций мы пользуемся **инфиксной записью**. Например:

1 * 2 + 4

Для обработки компьютером намного удобнее **постфиксная** запись следующего вида:

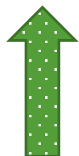
1 2 * 4 +

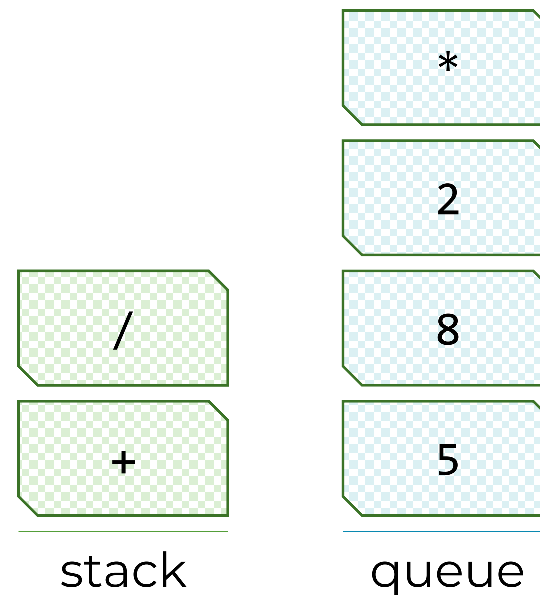
Ее принято называть **обратной польской записью**.

Перевод формулы в постфиксную запись

Вам понадобятся стек и очередь:

```
5 + 8 * 2 / ( 8 – 4 )
```

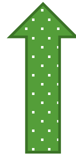




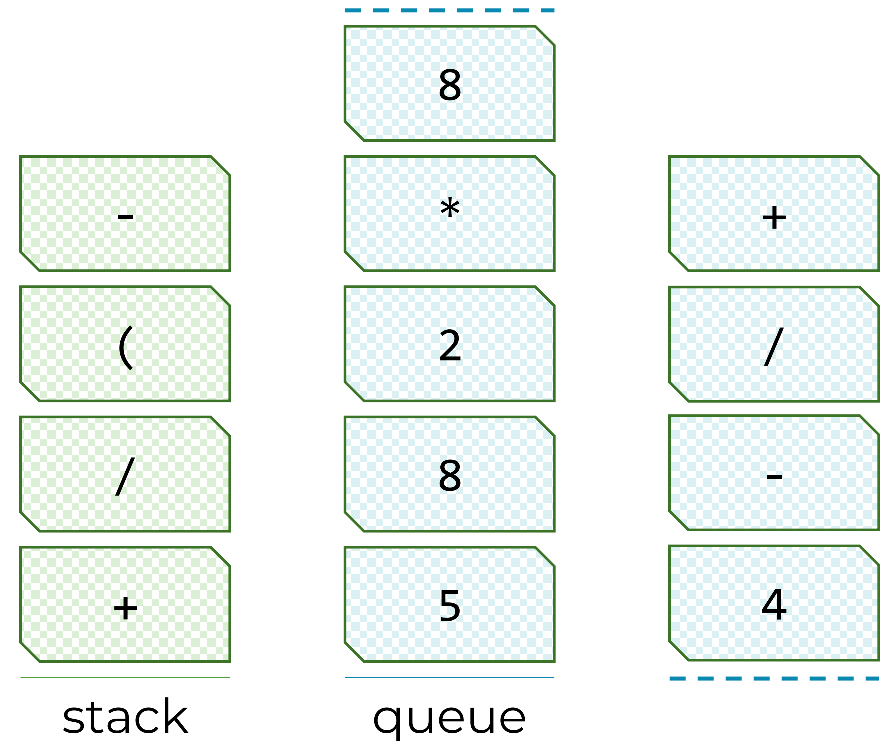
Перевод формулы в постфиксную запись

Вам понадобятся стек и очередь:

`5 + 8 * 2 / ( 8 – 4 )`

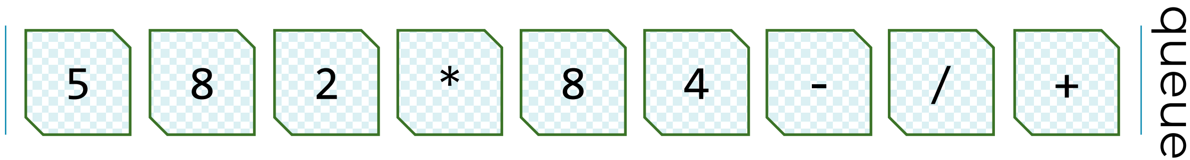


`5 8 2 * 8 4 - / +`



Разбор формулы по постфиксной записи

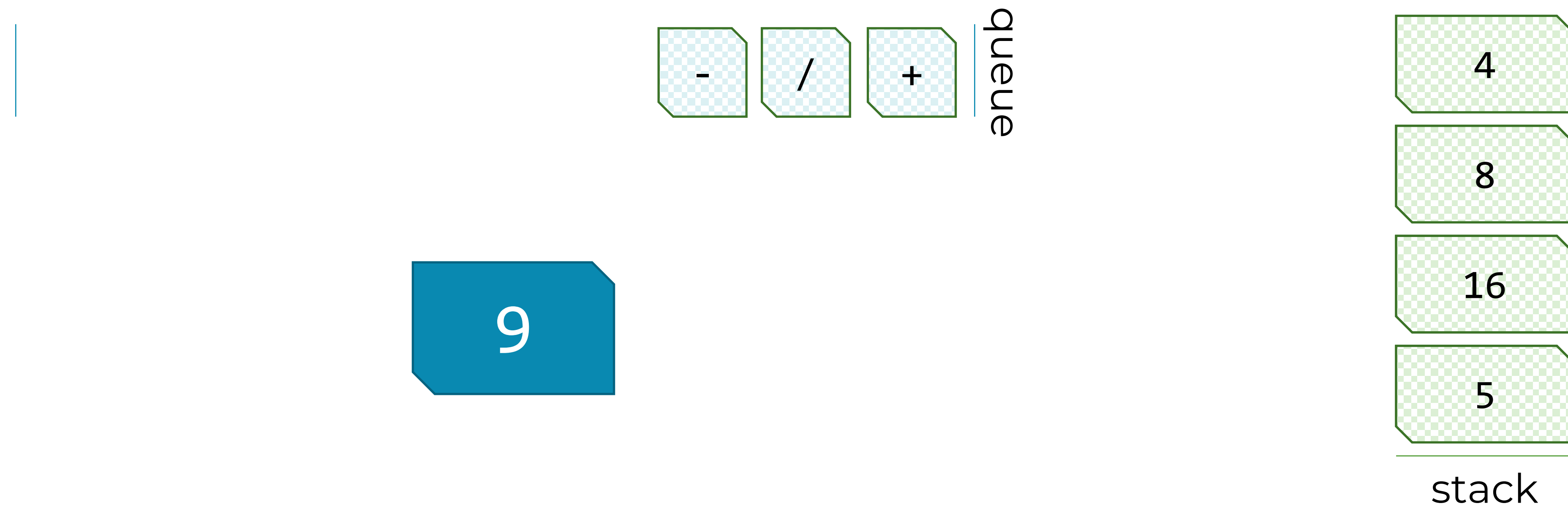
Вам понадобятся полученная очередь и стек:



stack

Разбор формулы по постфиксной записи

Вам понадобятся полученная очередь и стек:



Спасибо за внимание!