ВЫСШАЯ ШКОЛА ЭКОНОМИКИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Департамент программной инженерии

Алгоритмы и структуры данных

Семинар №5. 2021-2022 учебный год

Нестеров Роман Александрович, *ДПИ ФКН и НУЛ ПОИС*

Бессмертный Александр Игоревич, *ДПИ ФКН*

Higher School of Economics

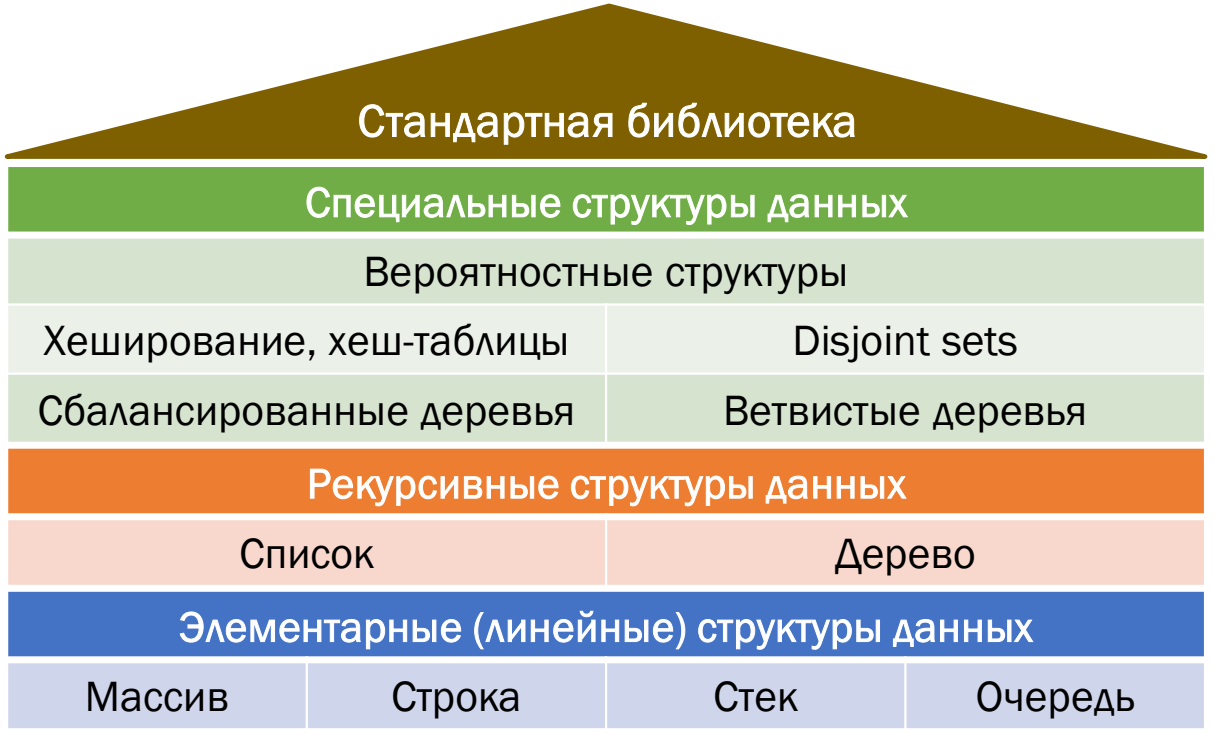
Faculty of Computer science

Sapienza é nella dell’esperienza

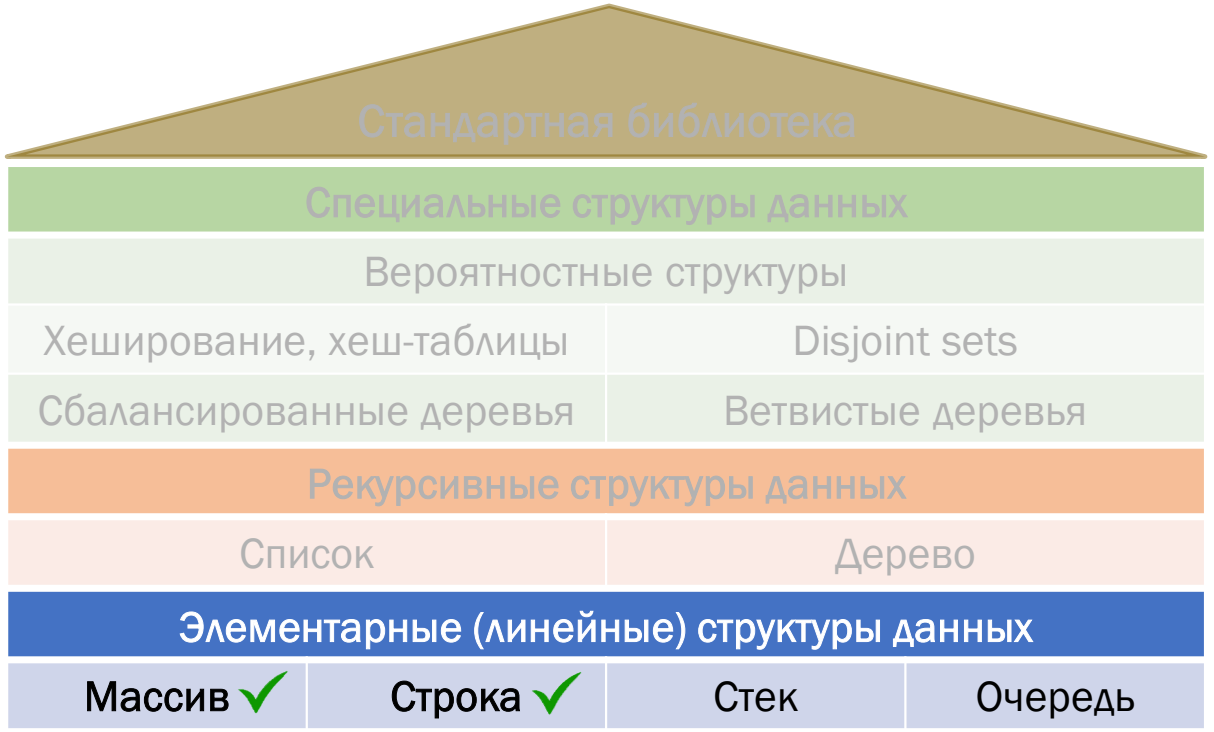
План

- Стек
- Очередь
- Двусторонняя очередь
- Шаблонный класс для стека и обработка исключений

Где мы?


Стандартная библиотека
Специальные структуры данных
Вероятностные структуры
Хеширование, хеш-таблицы
Disjoint sets
Сбалансированные деревья
Ветвистые деревья
Рекурсивные структуры данных
Список
Дерево
Элементарные (линейные) структуры данных
Массив
Строка
Стек
Очередь

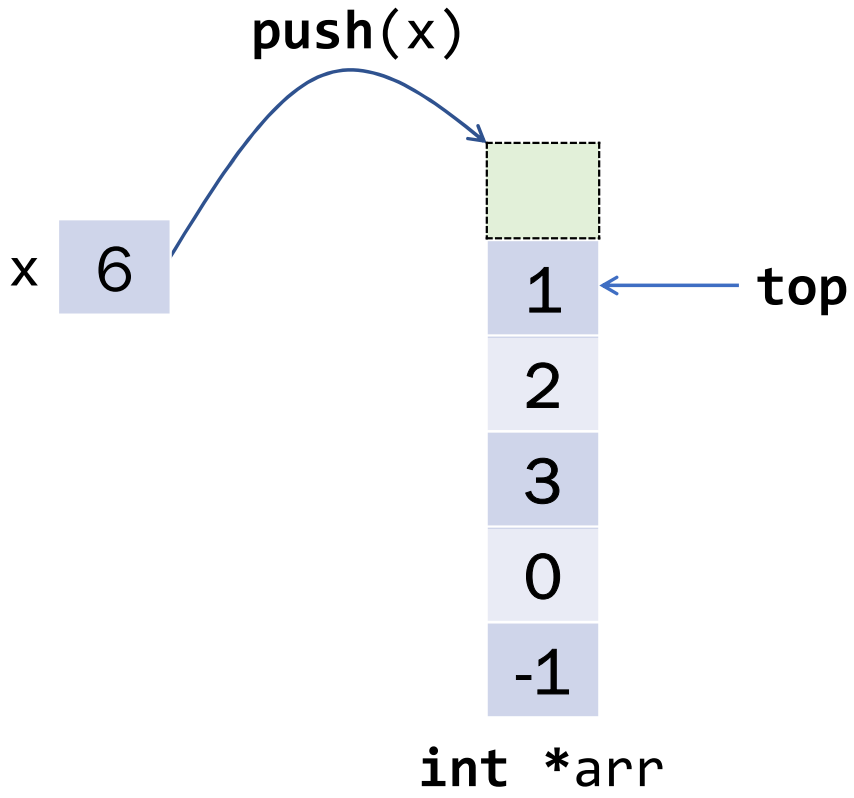
Где мы?


Стандартная библиотека
Специальные структуры данных
Вероятностные структуры
Хеширование, хеш-таблицы
Disjoint sets
Сбалансированные деревья
Ветвистые деревья
Рекурсивные структуры данных
Список
Дерево
Элементарные (линейные) структуры данных
Массив
Строка
Стек
Очередь

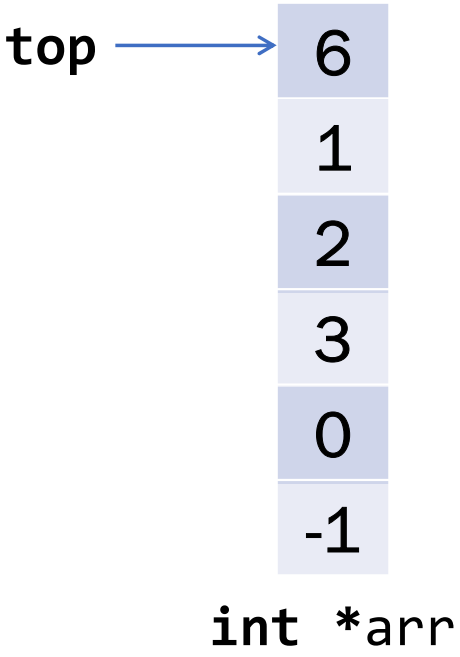
Стек

Last In, First Out

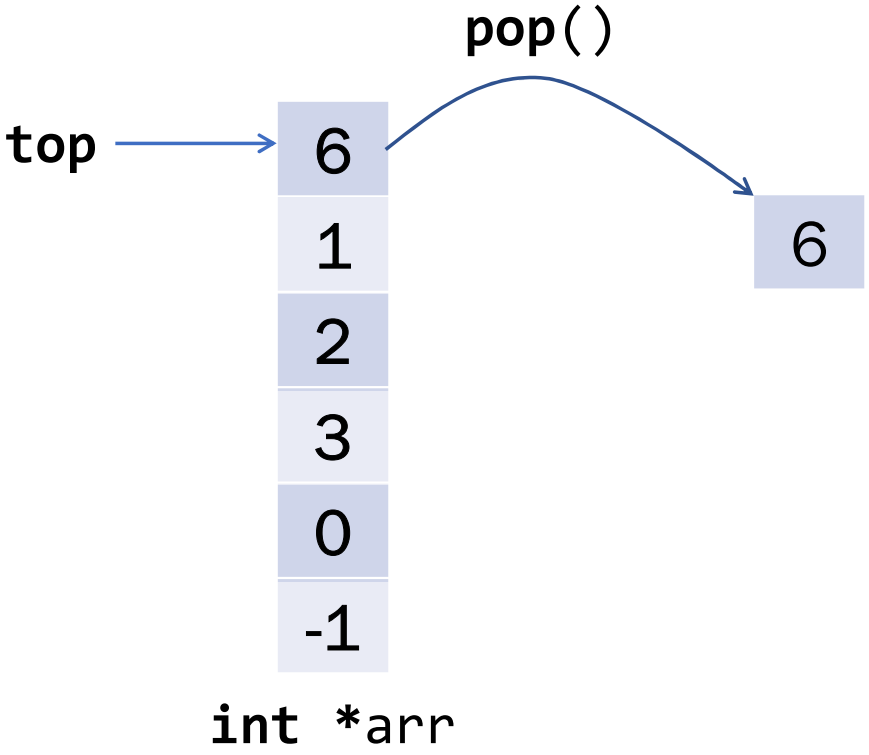
Базовые операции над стеком



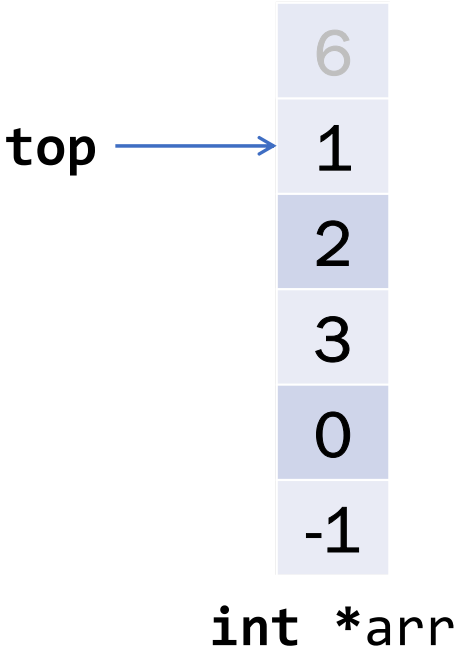
Базовые операции над стеком



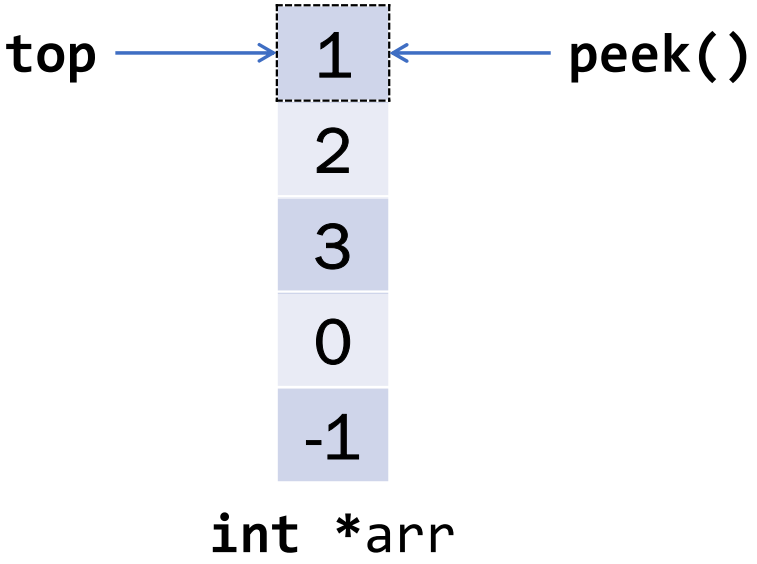
Базовые операции над стеком



Базовые операции над стеком

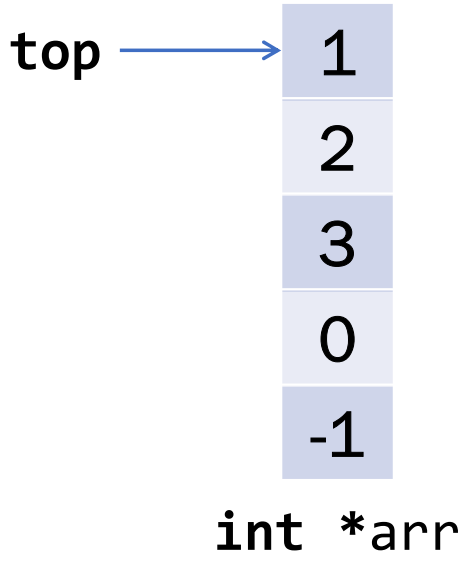


Базовые операции над стеком



Базовые операции над стеком

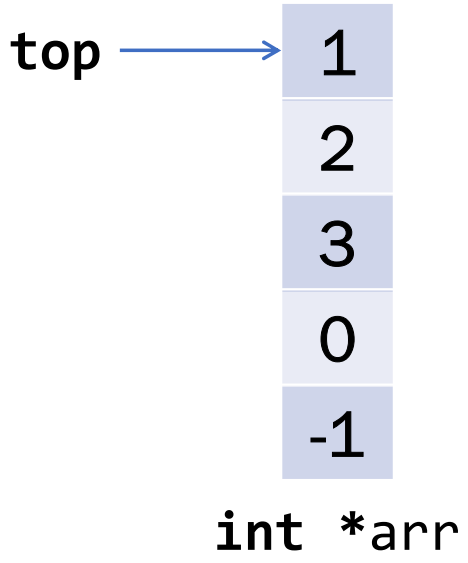
- Вставка элемента в стек **`push(x)`**
- Удаление головного элемента **`pop()`**
- Чтение головного элемента **`peek()`**



Базовые операции над стеком

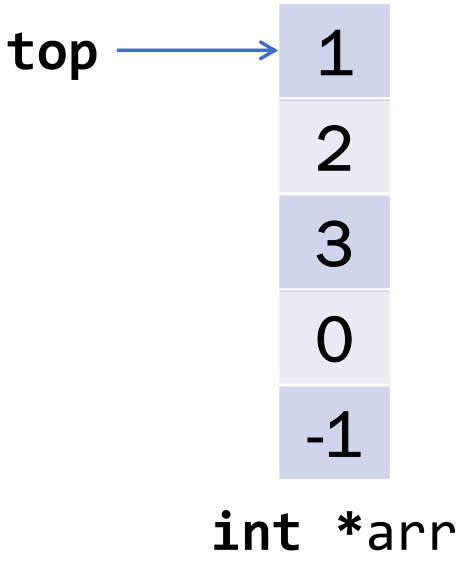
- Вставка элемента в стек **`push(x)`**
- Удаление головного элемента **`pop()`**
- Чтение головного элемента **`peek()`**

Указатель на голову стека **`top`** является подвижным.



Базовые операции над стеком

- Ошибка вставки в стек
  `StackOverflow`
- Ошибка удаления головного элемента
  `StackUnderflow`
- Ошибка чтения головного элемента
  `StackIsEmpty`



Стек вызовов рекурсивной функции

fact(6)

```
int fact(int x) {
    if (x == 0) {
        return 1;
    }

    return x * fact(x - 1);
}
```

Стек вызовов рекурсивной функции

```
int fact(int x) {
    if (x == 0) {
        return 1;
    }

    return x * fact(x - 1);
}
```

fact(6)

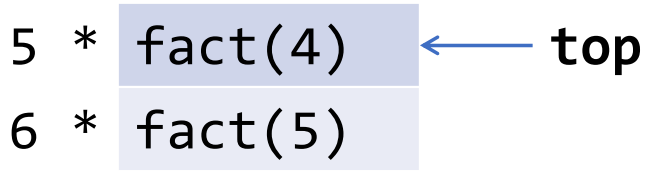
6 * fact(5) ← top

Стек вызовов рекурсивной функции

```
int fact(int x) {
    if (x == 0) {
        return 1;
    }

    return x * fact(x - 1);
}
```

fact(6)



Стек вызовов рекурсивной функции

```
int fact(int x) {
    if (x == 0) {
        return 1;
    }

    return x * fact(x - 1);
}
```

fact(6)

fact(0) = 1

1 * fact(0)

2 * fact(1)

3 * fact(2)

4 * fact(3)

5 * fact(4)

6 * fact(5)

Стек вызовов рекурсивной функции

```
int fact(int x) {
    if (x == 0) {
        return 1;
    }

    return x * fact(x - 1);
}
```

fact(6)

fact(0) = 1

1 * fact(0) = 1

2 * fact(1) = 2

3 * fact(2)

4 * fact(3)

5 * fact(4)

6 * fact(5)

Стек вызовов рекурсивной функции

```
int fact(int x) {
    if (x == 0) {
        return 1;
    }

    return x * fact(x - 1);
}
```

fact(6)

```
fact(0) = 1
  1 * fact(0) = 1
  2 * fact(1) = 2
  3 * fact(2) = 6
  4 * fact(3) = 24
  5 * fact(4) = 120
  6 * fact(5) = 720
```

Стек вызовов рекурсивной функции

```
int fact(int x) {
    if (x == 0) {
        return 1;
    }

    return x * fact(x - 1);
}
```

```
fact(6) = 720

fact(0) = 1
  1 * fact(0)  = 1
  2 * fact(1)  = 2
  3 * fact(2)  = 6
  4 * fact(3)  = 24
  5 * fact(4)  = 120
  6 * fact(5)  = 720
```

Применение стека

- Синтаксический анализ и вычисление значений выражений
- Конвертация выражений из одной записи в другую
  `(1 + 2) * 3 -> 1 2 + 3 *`
- Алгоритмы поиска с возвратом
- `...`

Применение стека

- Синтаксический анализ и вычисление значений выражений
- Конвертация выражений из одной записи в другую
  `(1 + 2) * 3 -> 1 2 + 3 *`
- Алгоритмы поиска с возвратом
- `...`
- Проверка правильной расстановки скобок

Проверка расстановки скобок

На вход подается строка, состоящая из открывающих и закрывающих скобок ‘(‘ и ‘)’. Требуется проверить, что скобки расставлены верно.

str	(	(	)	)	(	)

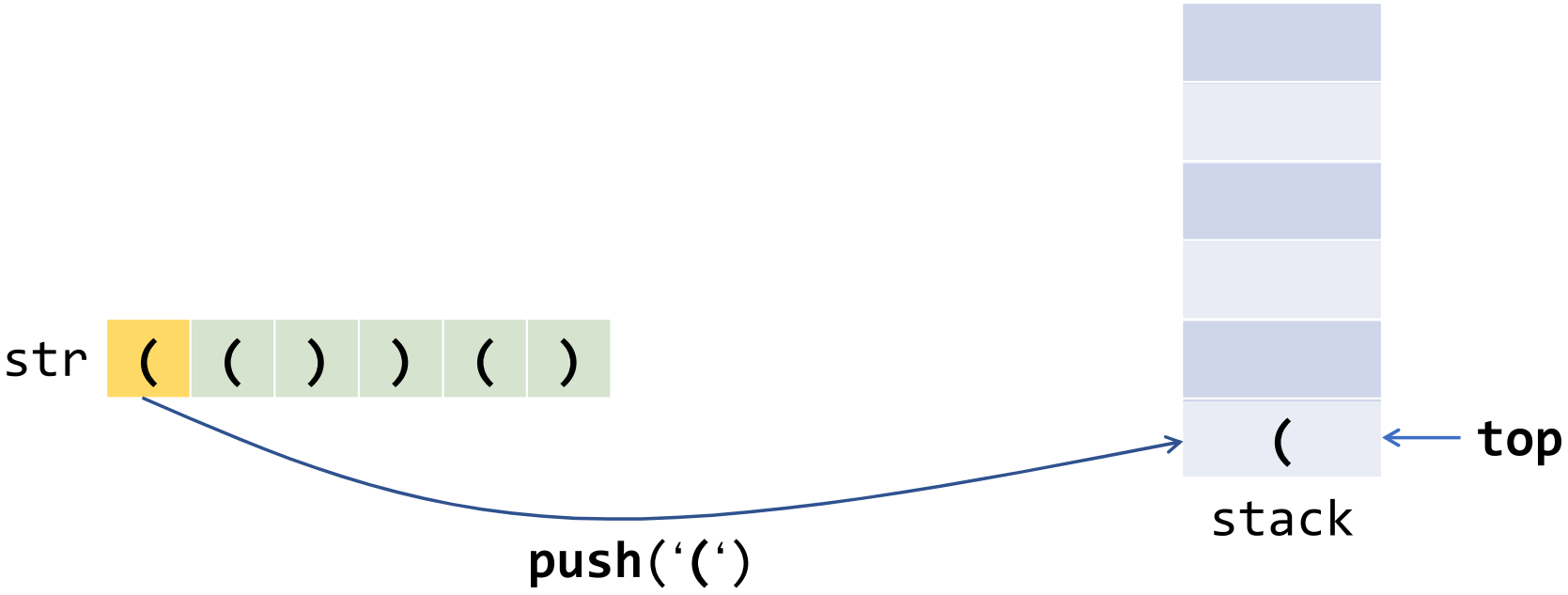
Проверка расстановки скобок

На вход подается строка, состоящая из открывающих и закрывающих скобок ‘(‘ и ‘)’. Требуется проверить, что скобки расставлены верно.

str	(	(	)	)	(	)

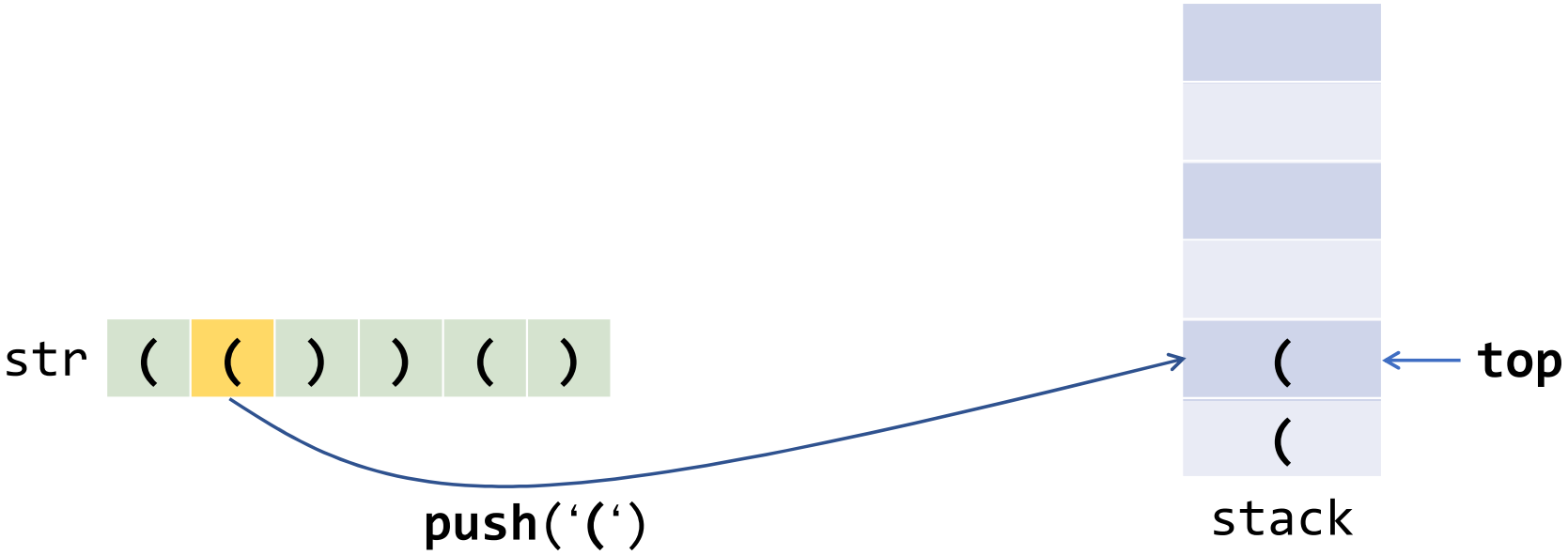
Проверка расстановки скобок

На вход подается строка, состоящая из открывающих и закрывающих скобок ‘(‘ и ‘)’. Требуется проверить, что скобки расставлены верно.



Проверка расстановки скобок

На вход подается строка, состоящая из открывающих и закрывающих скобок ‘(‘ и ‘)’. Требуется проверить, что скобки расставлены верно.

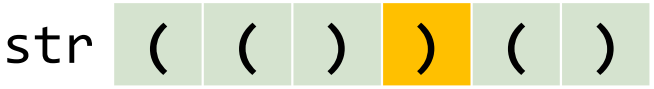


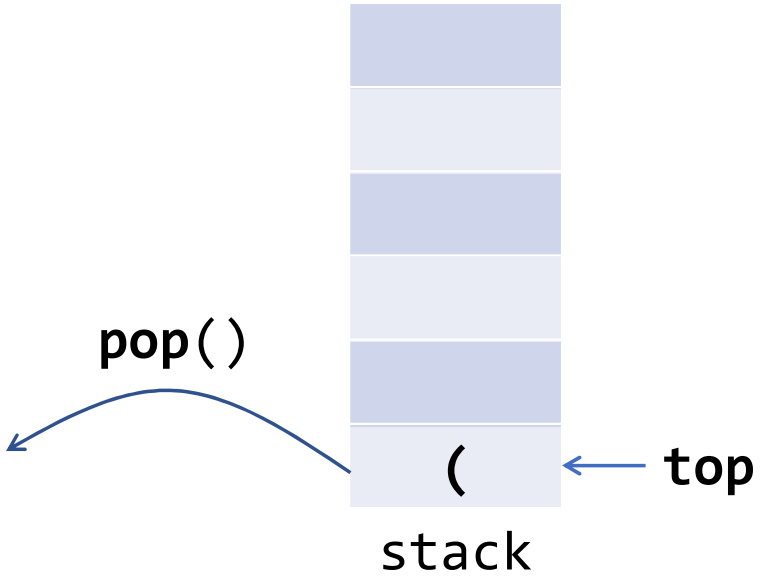
Проверка расстановки скобок

На вход подается строка, состоящая из открывающих и закрывающих скобок ‘**(**‘ и ‘**)**’. Требуется проверить, что скобки расставлены верно.

Проверка расстановки скобок

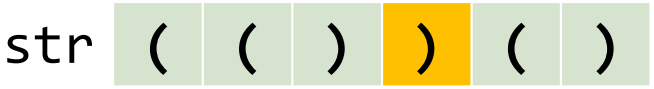
На вход подается строка, состоящая из открывающих и закрывающих скобок ‘(‘ и ‘)’. Требуется проверить, что скобки расставлены верно.

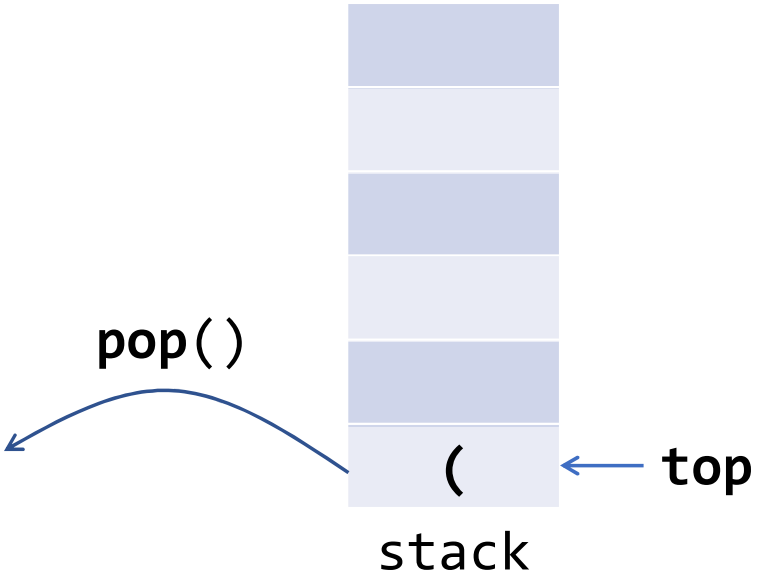




Проверка расстановки скобок

На вход подается строка, состоящая из открывающих и закрывающих скобок ‘(‘ и ‘)’. Требуется проверить, что скобки расставлены верно.





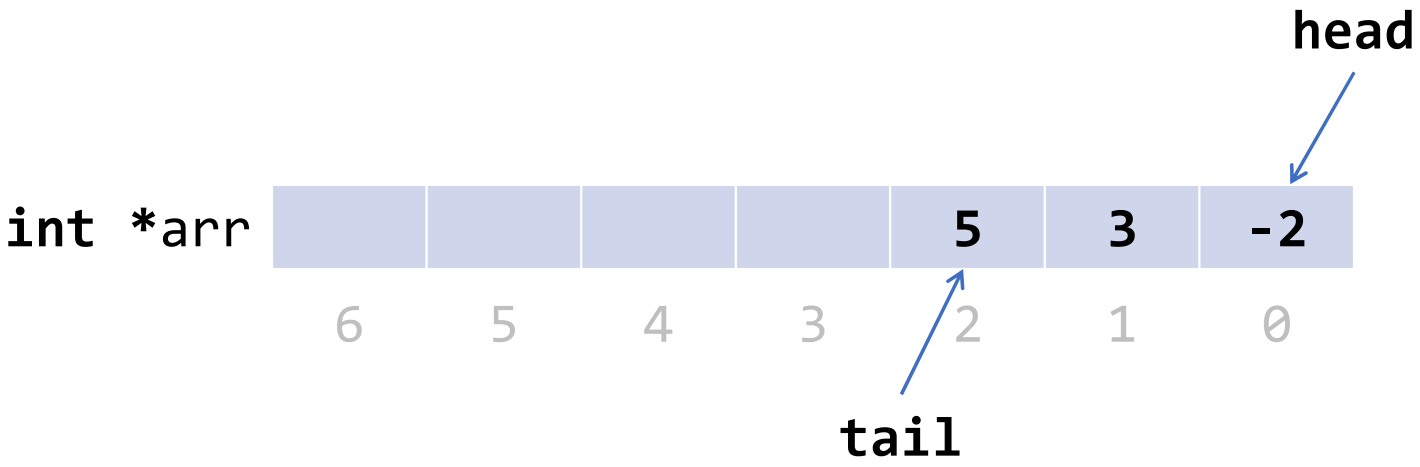
Если в стеке остаются символы, то скобки расставлены не верно.

Очередь

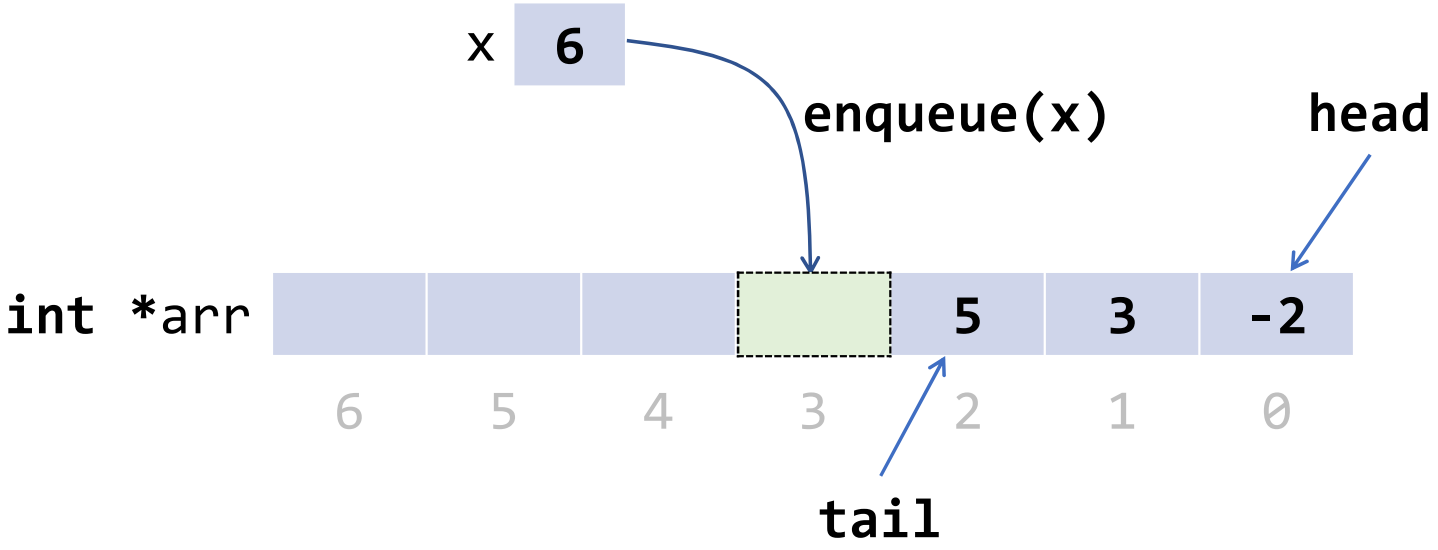
Базовые операции с очередью

- Вставка элемента в хвост очереди **`enqueue(x)`**
- Удаление элемента из головы очереди **`dequeue()`**
- Чтение головного элемента очереди **`front()`**
- Чтение хвостового элемента очереди **`rear()`**

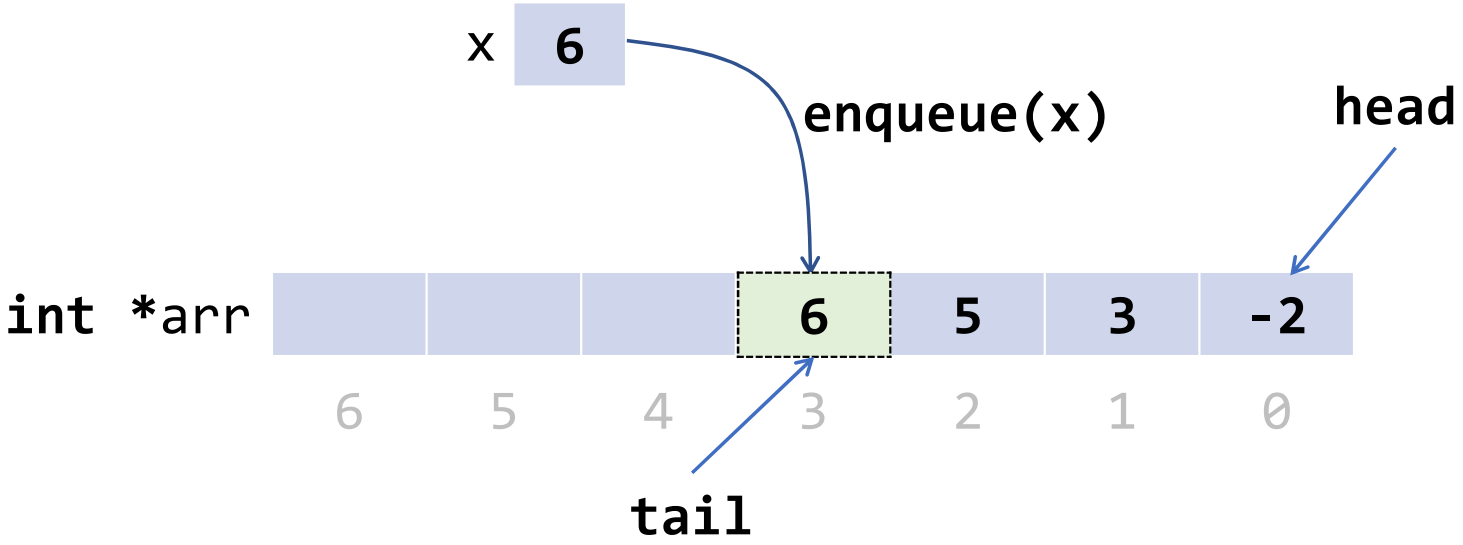
Базовые операции с очередью



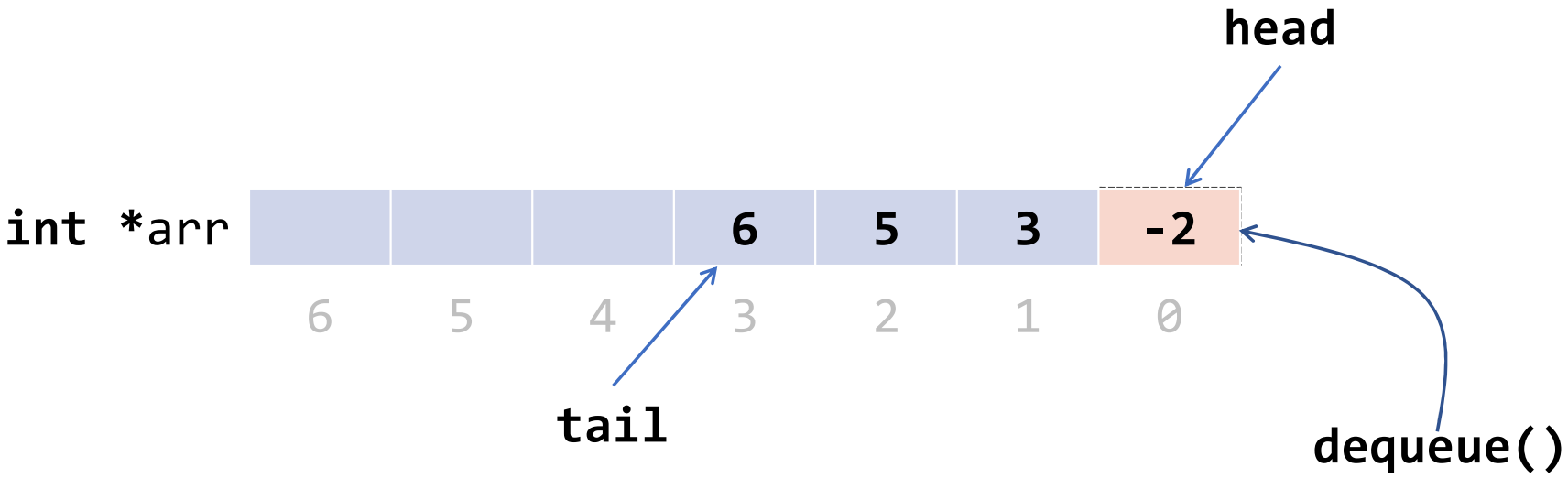
Базовые операции с очередью



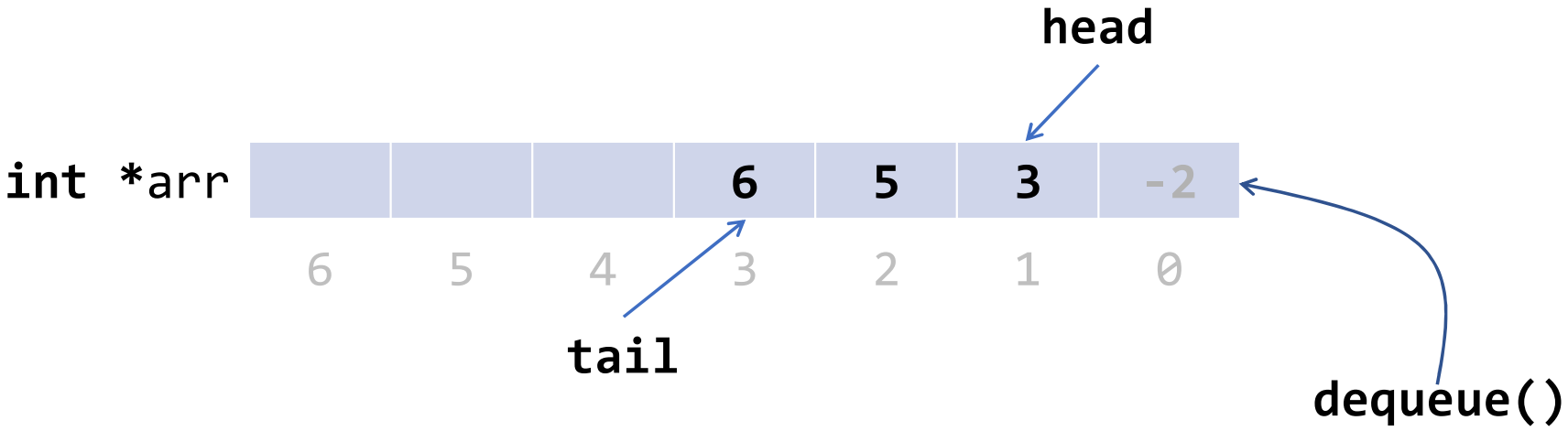
Базовые операции с очередью



Базовые операции с очередью

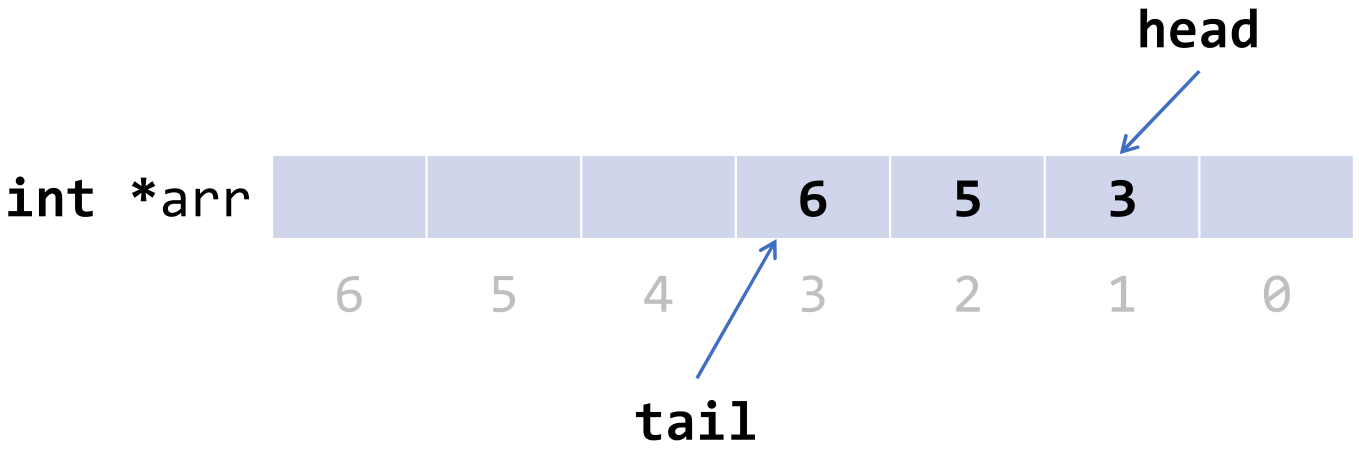


Базовые операции с очередью

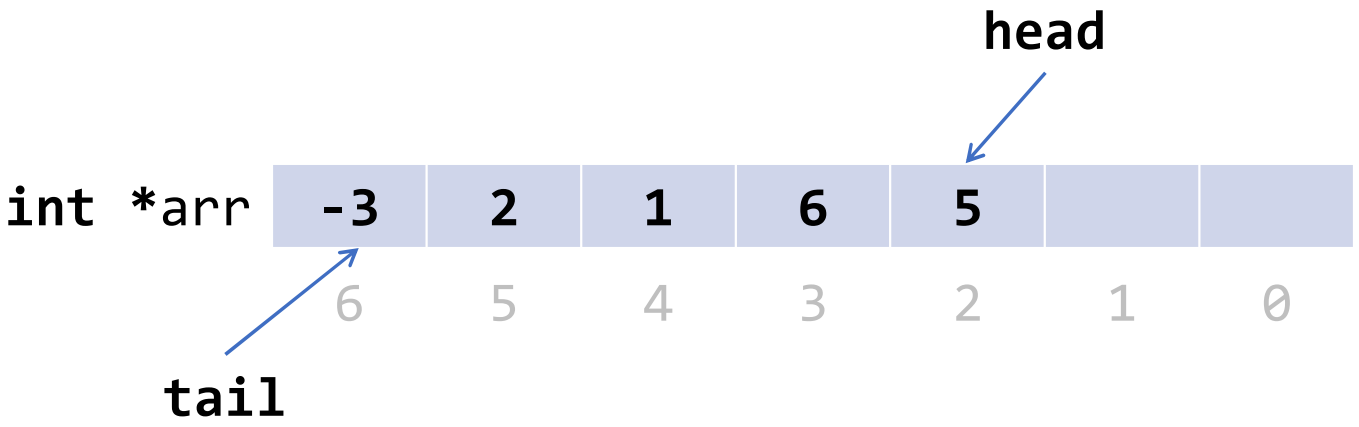


Базовые операции с очередью

Указатель и на голову, и на хвост очереди являются **подвижными**.

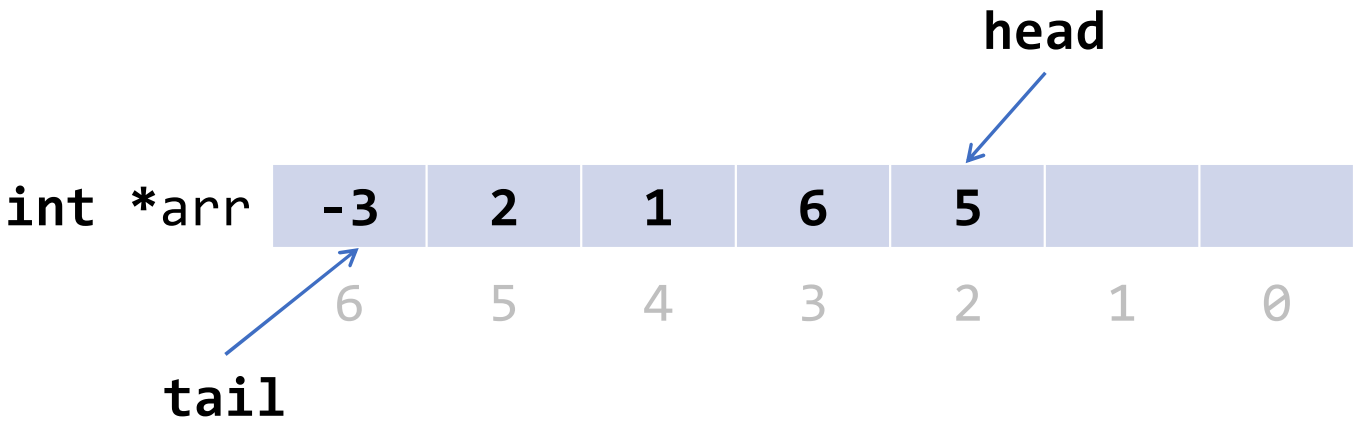


Очередь. Проблема заполнения



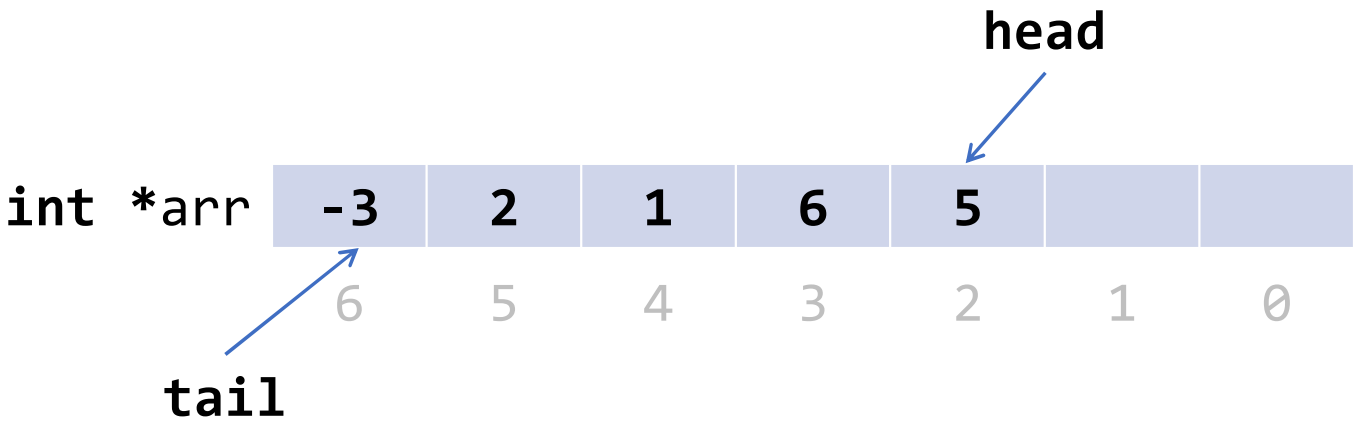
Очередь. Проблема заполнения

Проблема: Указатель на хвост достиг конца массива.



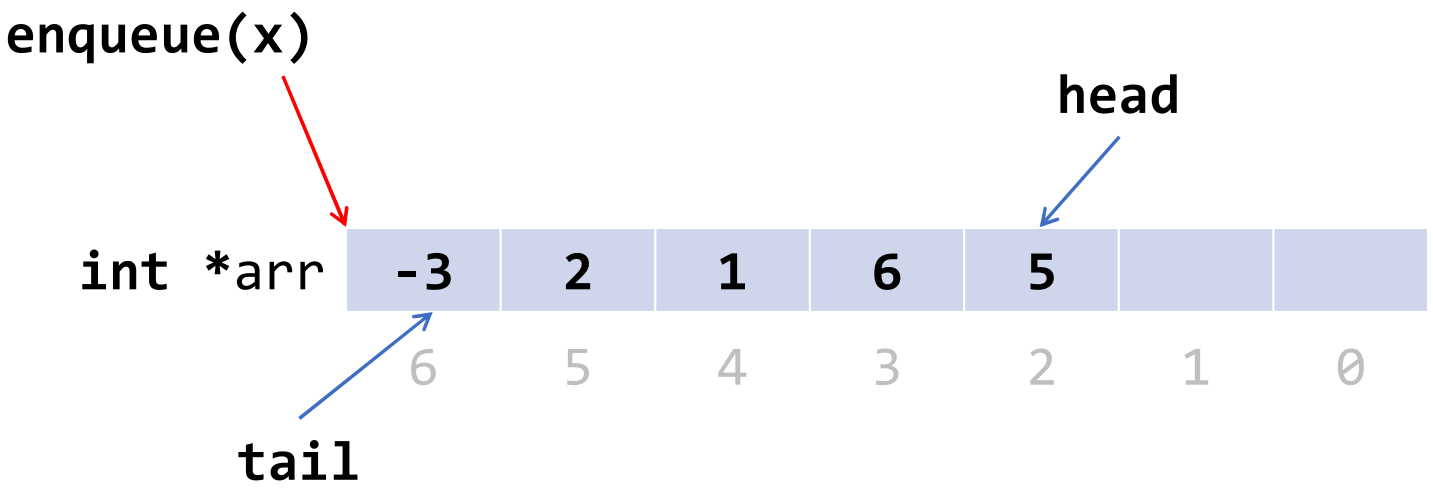
Очередь. Проблема заполнения

Проблема: Указатель на хвост достиг конца массива.
Решение: «Закольцевать» массив.



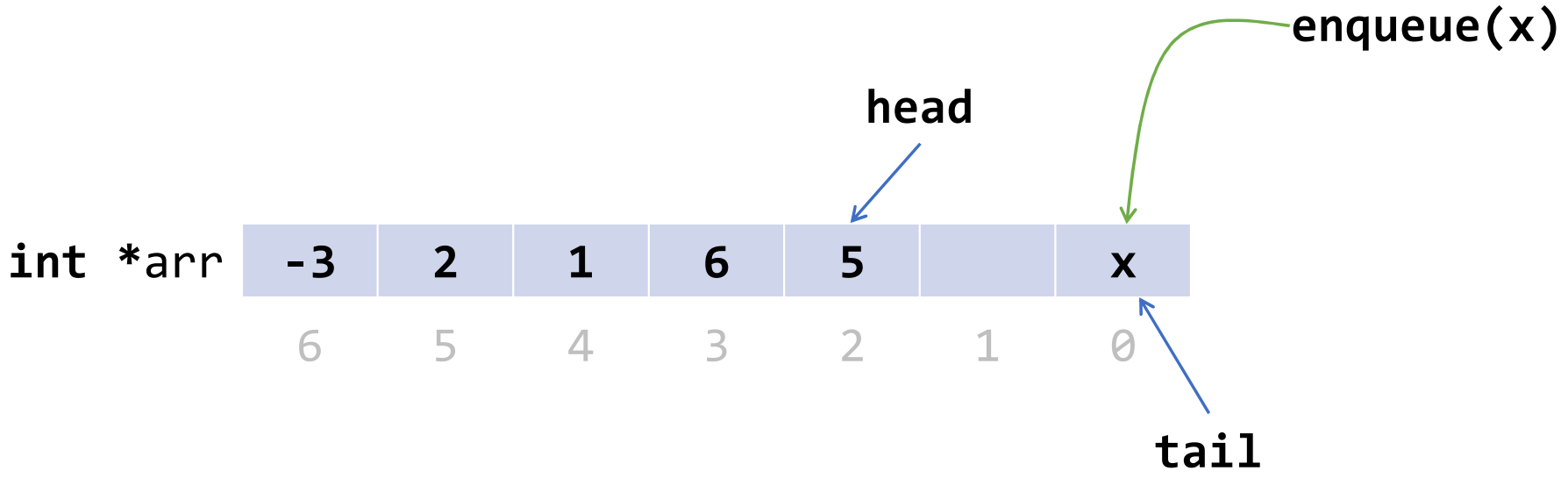
Очередь. Проблема заполнения

Проблема: Указатель на хвост достиг конца массива.
Решение: «Закольцевать» массив.



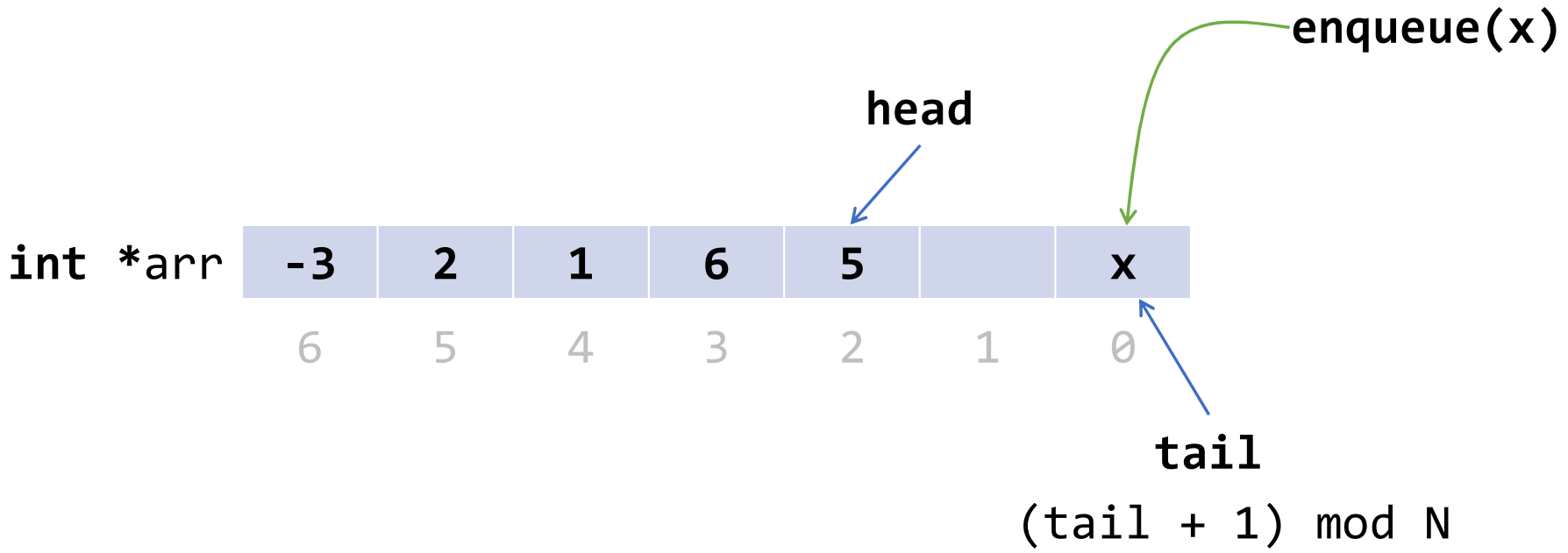
Очередь. Проблема заполнения

Проблема: Указатель на хвост достиг конца массива.
Решение: «Закольцевать» массив.



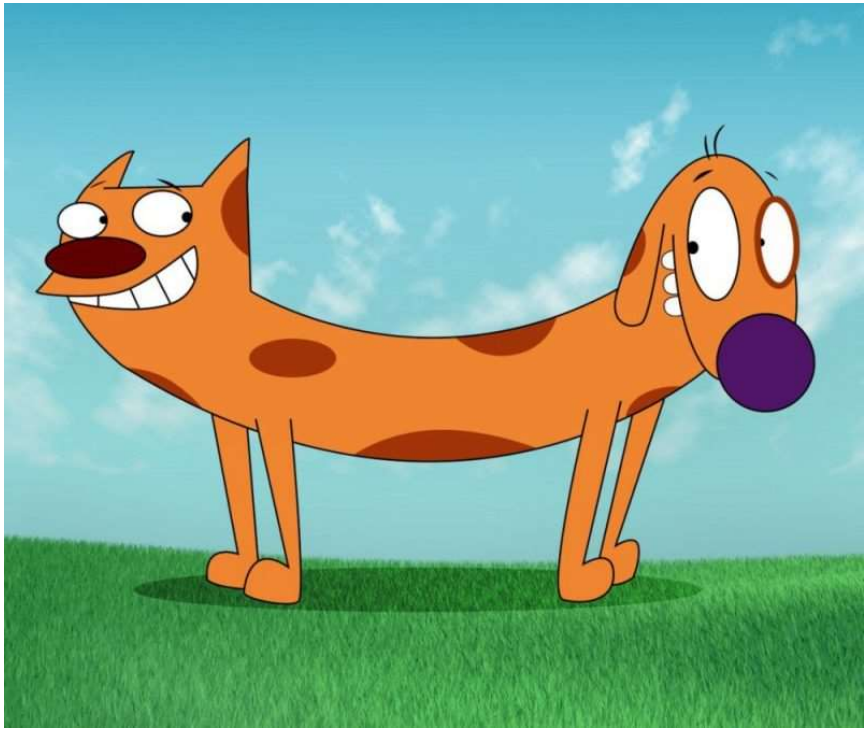
Очередь. Проблема заполнения

Проблема: Указатель на хвост достиг конца массива.
Решение: «Закольцевать» массив.

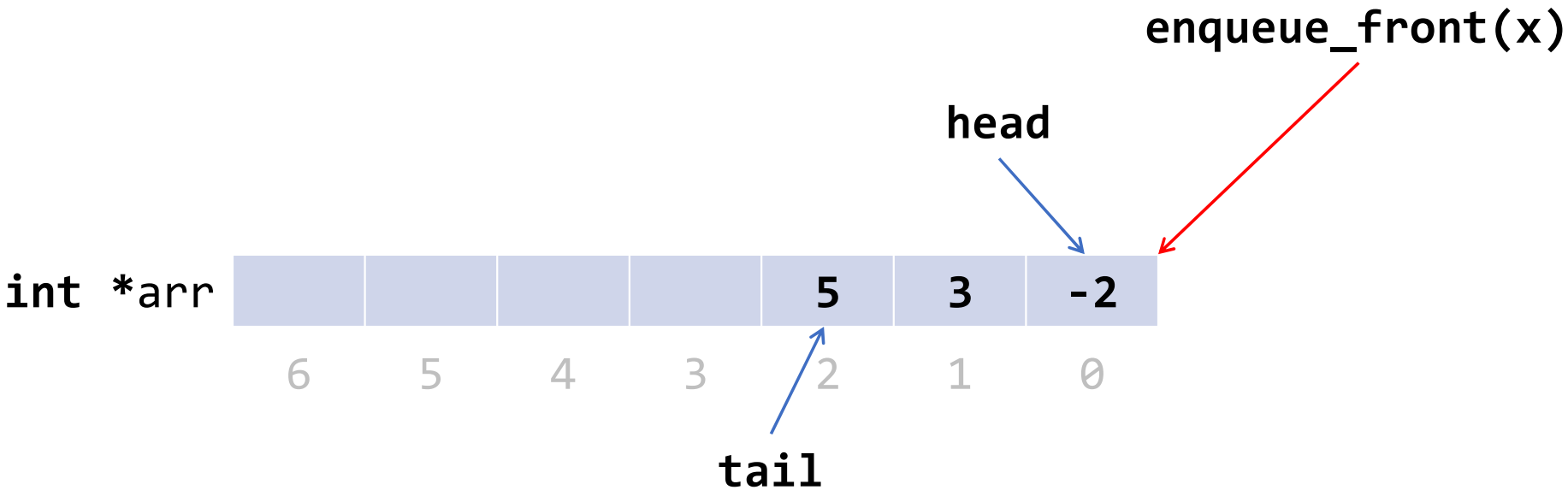


Двусторонняя очередь

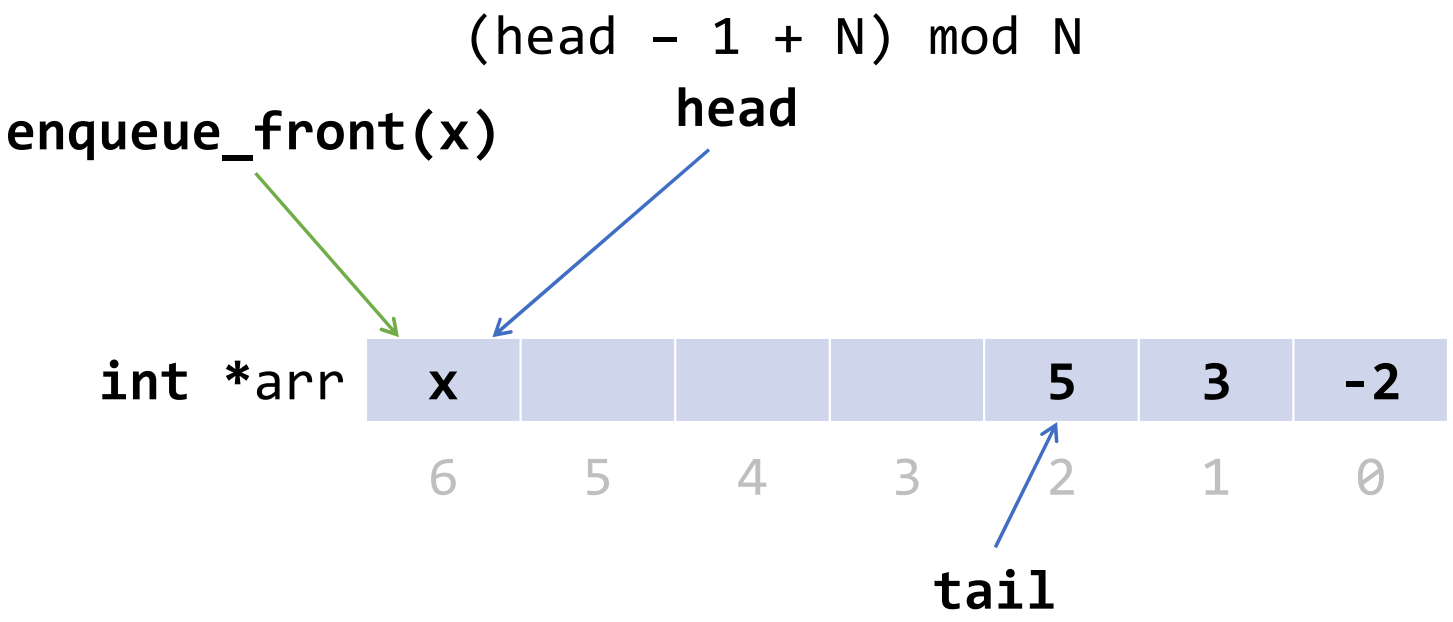
- Вставка элемента в хвост
  `enqueue_rear(x)`
- Вставка элемента в голову
  `enqueue_front(x)`
- Удаление элемента из головы
  `dequeue_front()`
- Удаление элемента из хвоста
  `dequeue_rear()`



Двусторонняя очередь



Двусторонняя очередь



Применение очередей

- Разграничение доступа к разделяемым ресурсам
- Асинхронный обмен сообщениями
- Диспетчеры операционной системы
- ...

Стек и очередь для ассоциативных операций

Стек с поддержкой минимума

Минимум является ассоциативной операцией:

`min(a, b, c) = min(min(a, b), c)`.

В стеке храним пару значений: элемент и минимум.

6	**1 = min(1, 6)**
1	**1**

`1 6 9 -8 3 0 1`

Стек с поддержкой минимума

Минимум является ассоциативной операцией:

`min(a, b, c) = min(min(a, b), c)`.

В стеке храним пару значений: элемент и минимум.

1	-8 = min(-8, 1)
0	-8 = min(-8, 0)
3	-8 = min(-8, 3)
-8	-8 = min(1, -8)
9	1 = min(1, 9)
6	1 = min(1, 6)
1	1

`1 6 9 -8 3 0 1`

Очередь на двух стеках

- **`enqueue(x)`** выполняется в первый стек
- **`dequeue()`** выполняется из второго стека

`1 -2 3 5 0 -4 2`

5	-2 = min(-2, 5)
3	-2 = min(-2, 3)
-2	-2 = min(1, -2)
1	1

`stack_enq`

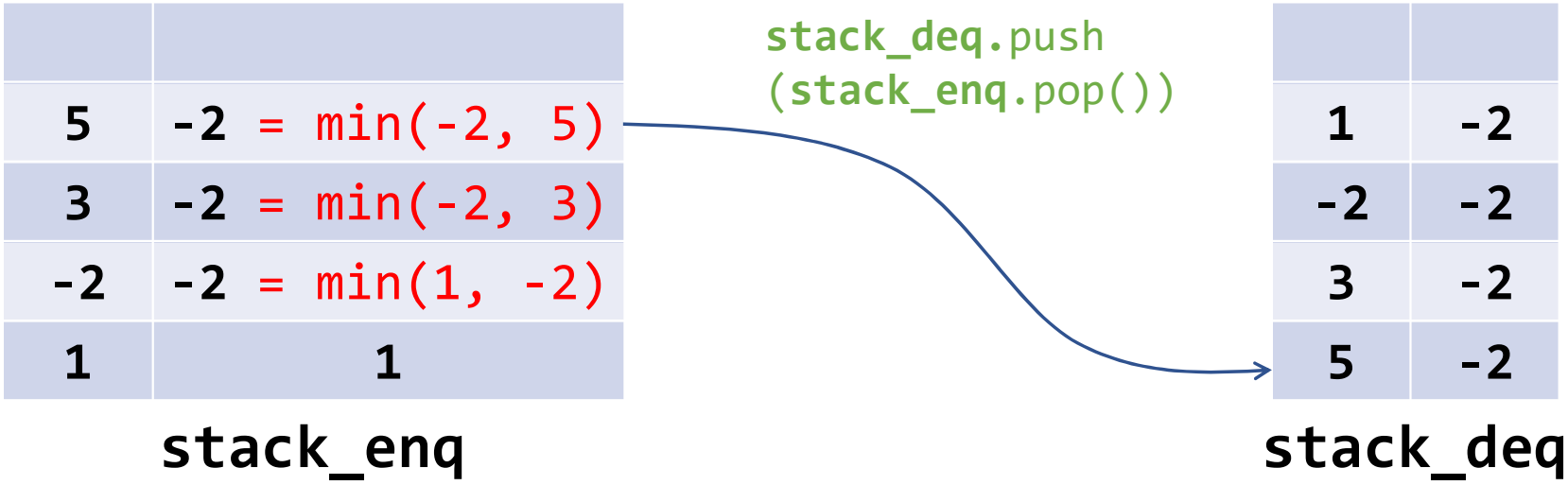
```
stack_enq.push(1)
stack_enq.push(-2)
stack_enq.push(3)
stack_enq.push(5)

dequeue() - ???
```

Очередь на двух стеках

- **`enqueue(x)`** выполняется в первый стек
- **`dequeue()`** выполняется из второго стека

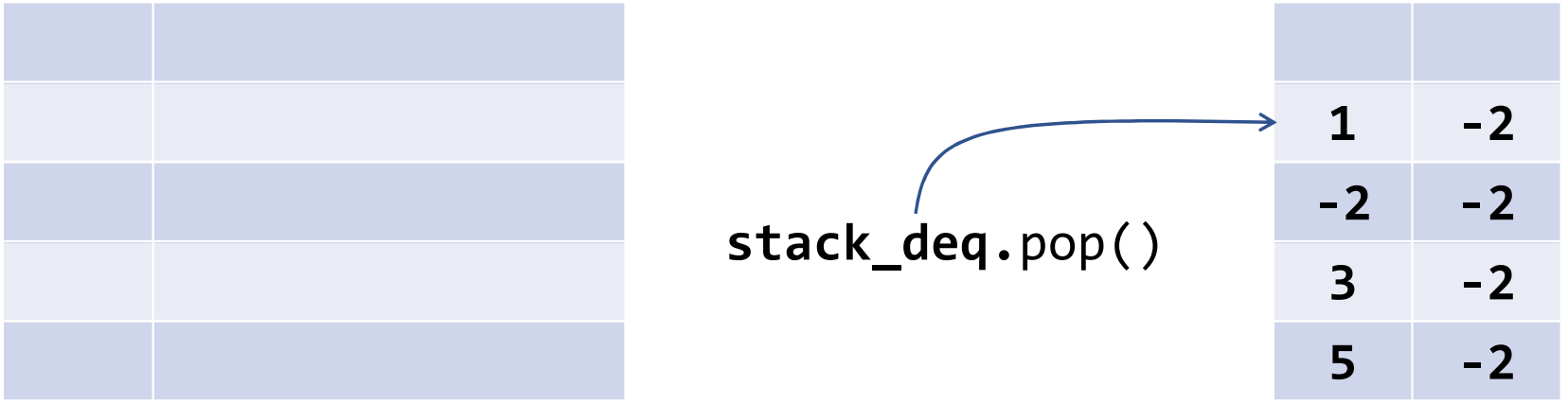
`1 -2 3 5 0 -4 2`



Очередь на двух стеках

- **`enqueue(x)`** выполняется в первый стек
- **`dequeue()`** выполняется из второго стека

`1 -2 3 5 0 -4 2`



Очередь на двух стеках

- **`enqueue(x)`** выполняется в первый стек
- **`dequeue()`** выполняется из второго стека

`1 -2 3 5 0 -4 2`

2	-4 = min(-4, 2)
-4	-4 = min(0, -4)
0	0

`stack_enq`

-2	-2
3	-2
5	-2

`stack_deq`

Очередь на двух стеках

- **`enqueue(x)`** выполняется в первый стек
- **`dequeue()`** выполняется из второго стека

`1 -2 3 5 0 -4 2`

```
min = min(stack_enq.peek(), stack_deq.peek())
```

2	-4 = min(-4, 2)
-4	-4 = min(0, -4)
0	0

`stack_enq`

-2	-2
3	-2
5	-2

`stack_deq`

Очередь на двух стеках

Можно использовать и для других **ассоциативных** операций:

- Максимум
- Наибольший общий делитель
- Наименьшее общее кратное
- . . .

Шаблонный класс стека и обработка исключений