Алгоритмы и структуры данных

Лекция 4.
Быстрая сортировка.
Элементарные структуры данных

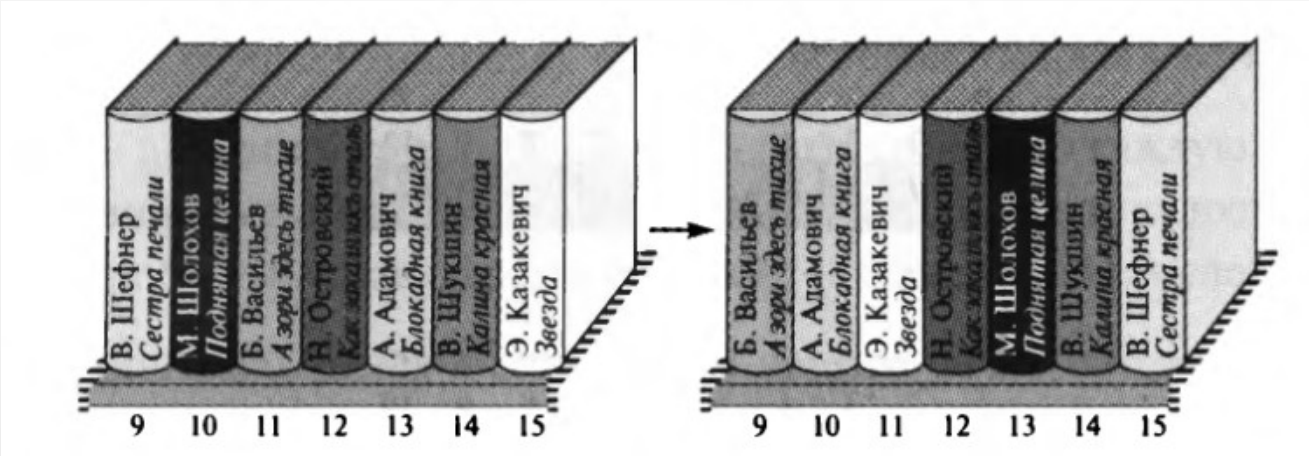
Быстрая сортировка

- Худшее время: $\Theta(n^2)$
- Лучшее время: $\Theta(n \lg n)$
- Среднее время: $\Theta(n \lg n)$
- Затраты памяти: $\Theta(n)$
- Использует принцип «разделяй и властвуй»
- работает «на месте»
- время работы в среднем и в худшем случае разное

Схема работы

- Разделение

Крайняя справа книга Казакевича выбрана в качестве опорной



«Разбиение» (partition) в быстрой сортировке

Схема работы

- Властвование

Осуществляется путём рекурсивной сортировки книг слева и справа от опорного элемента

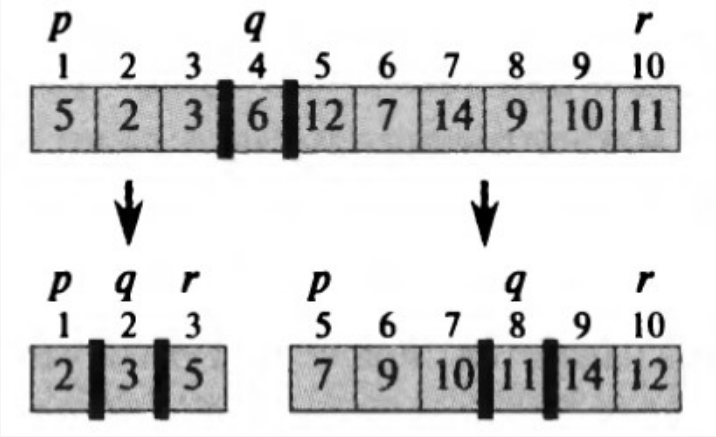
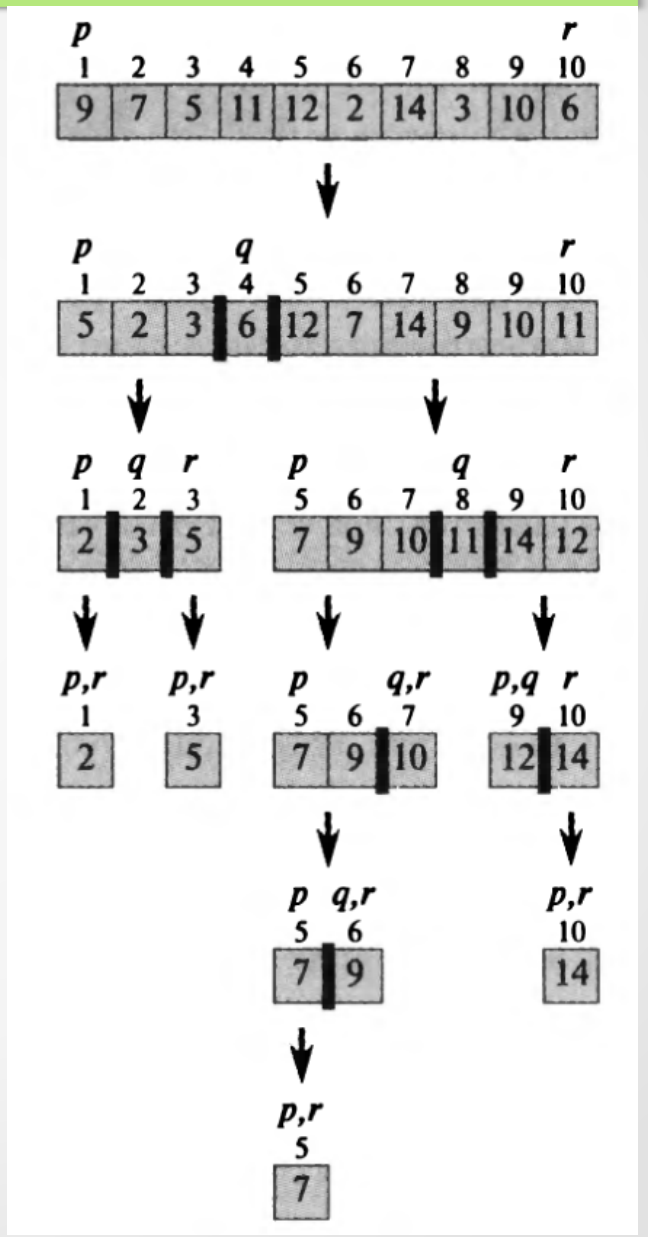


Схема работы

- Объединение

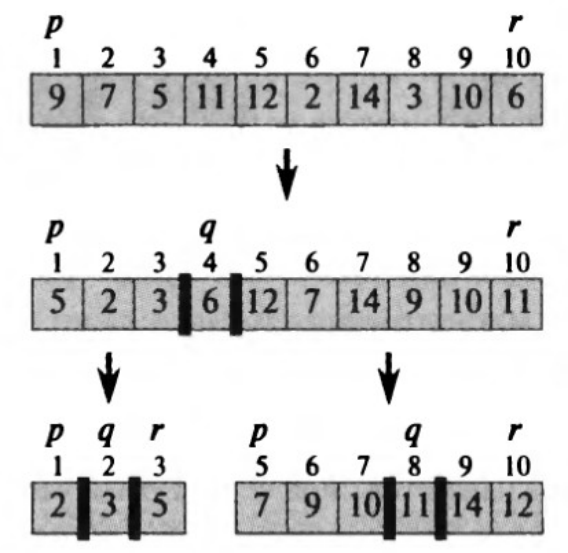
На этом этапе мы ничего не делаем!

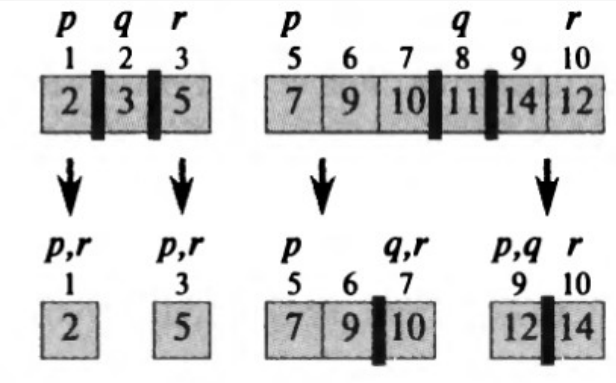
Почему?
Потому, что все книги с **p** по **r** уже отсортированы!

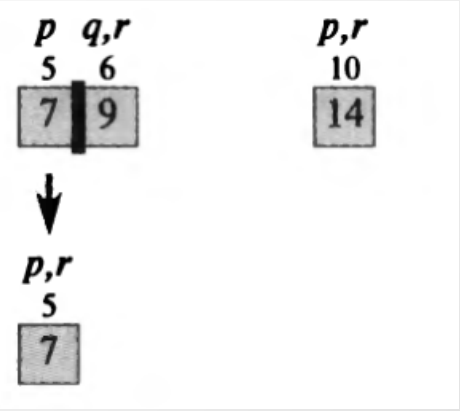


Быстрая сортировка на C++

```
27  void quicksort(int* a, int p, int r)
28  {
29      if (p>=r)
30          return;
31      int q = partition(a, p, r);
32      quicksort(a, p, q-1);
33      quicksort(a, q+1, r);
34  }
```







Разделение в быстрой сортировке

```
14  int partition(int* a, int p, int r)
15  {
16      int q = p;
17      for (int u = p; u<=r-1; ++u)
18          if (a[u]<=a[r])
19          {
20              swap(a, q, u);
21              ++q;
22          }
23      swap(a, q, r);
24      return q;
25  }
```

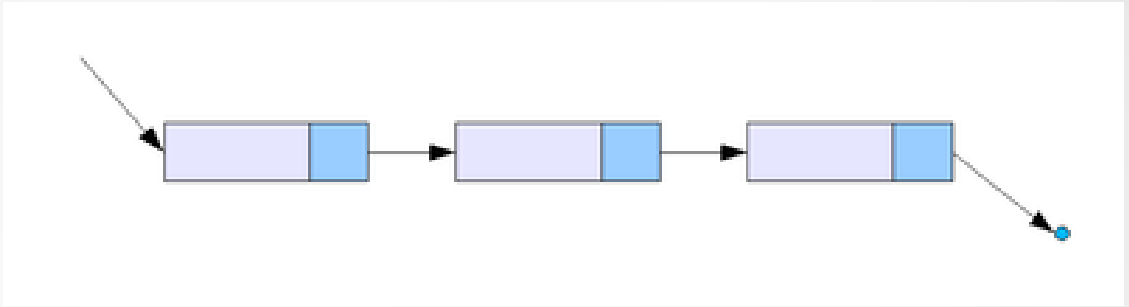
Индекс **q** в роли «границы». Всё, что слева от **q** будет меньше или равно опорному элементу.
Индекс **u** проходит от **u** до **r-1**, не доходя до **r**, т. к. **a[r]** — опорный элемент.
Теперь **q** — первый элемент из тех, которые больше опорного.
После прохода по всему массиву меняем опорный элемент с элементом **q**.
Такую операцию можно выполнить, т. к. нам не важно, будут ли отсортированы элементы слева и справа. Главное — что они разделены на две группы (больше и меньше).

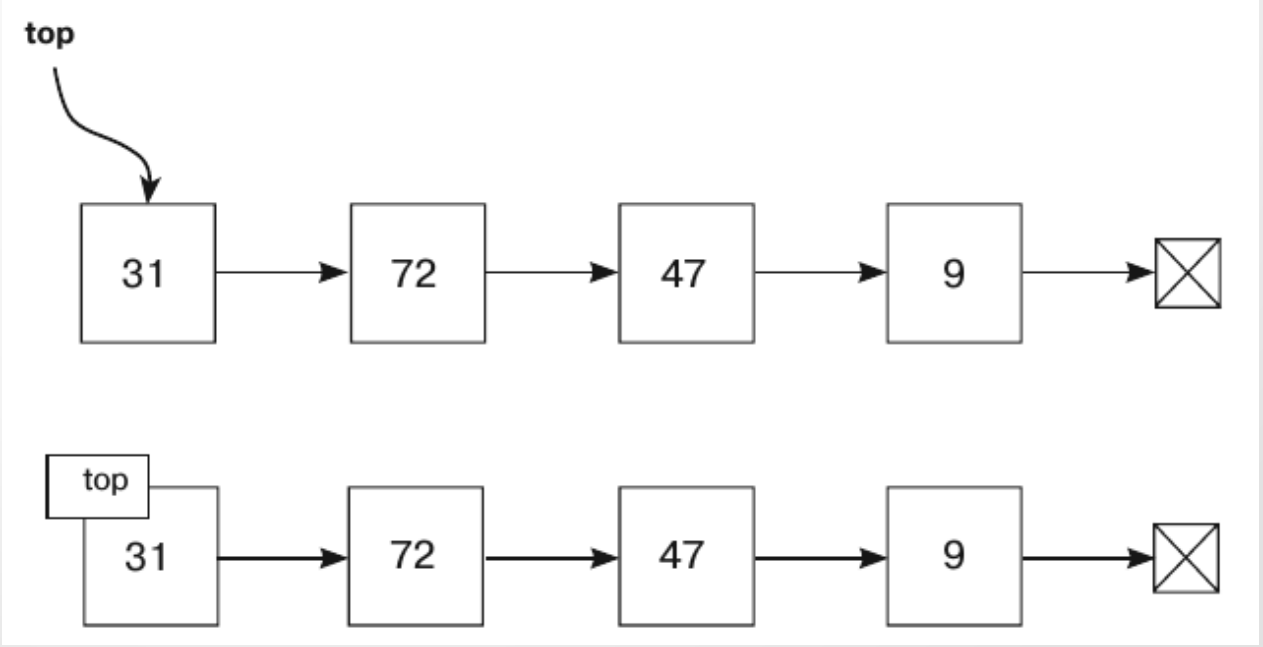
Элементарные структуры данных

Связный список

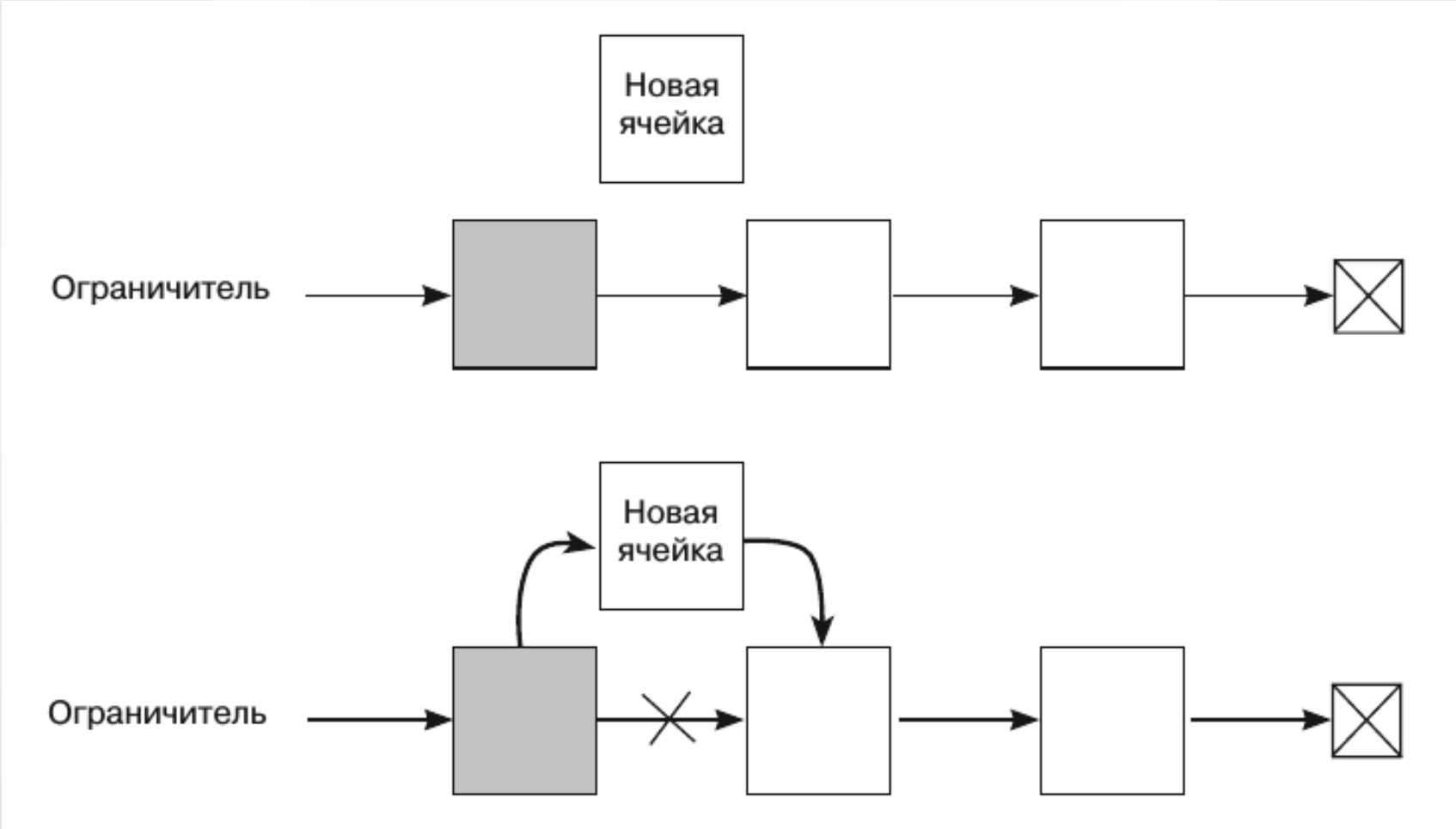
1.1 Однонаправленный связный список

```
struct list {
    int  value;
    list *next;
} stack;
```

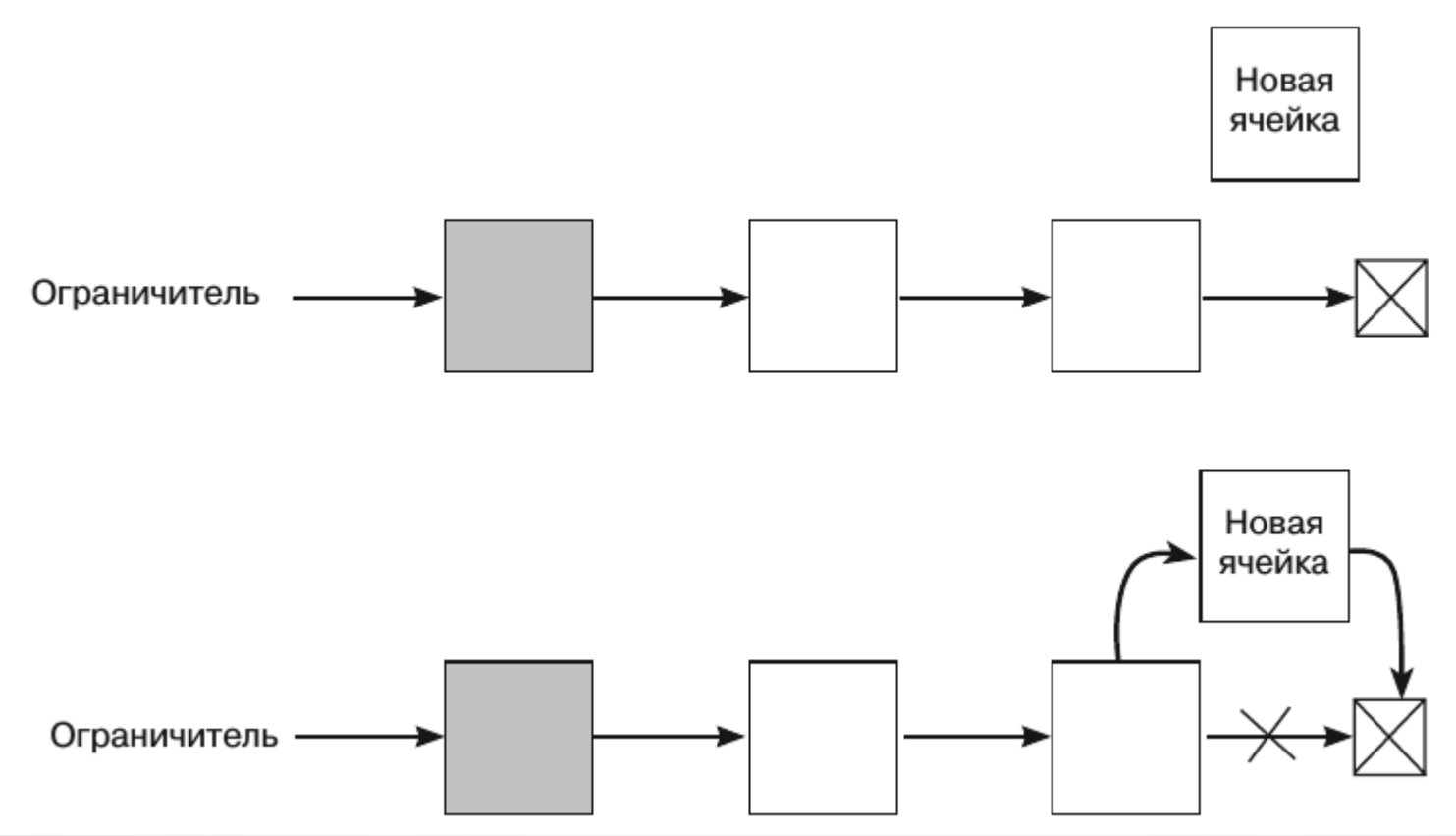




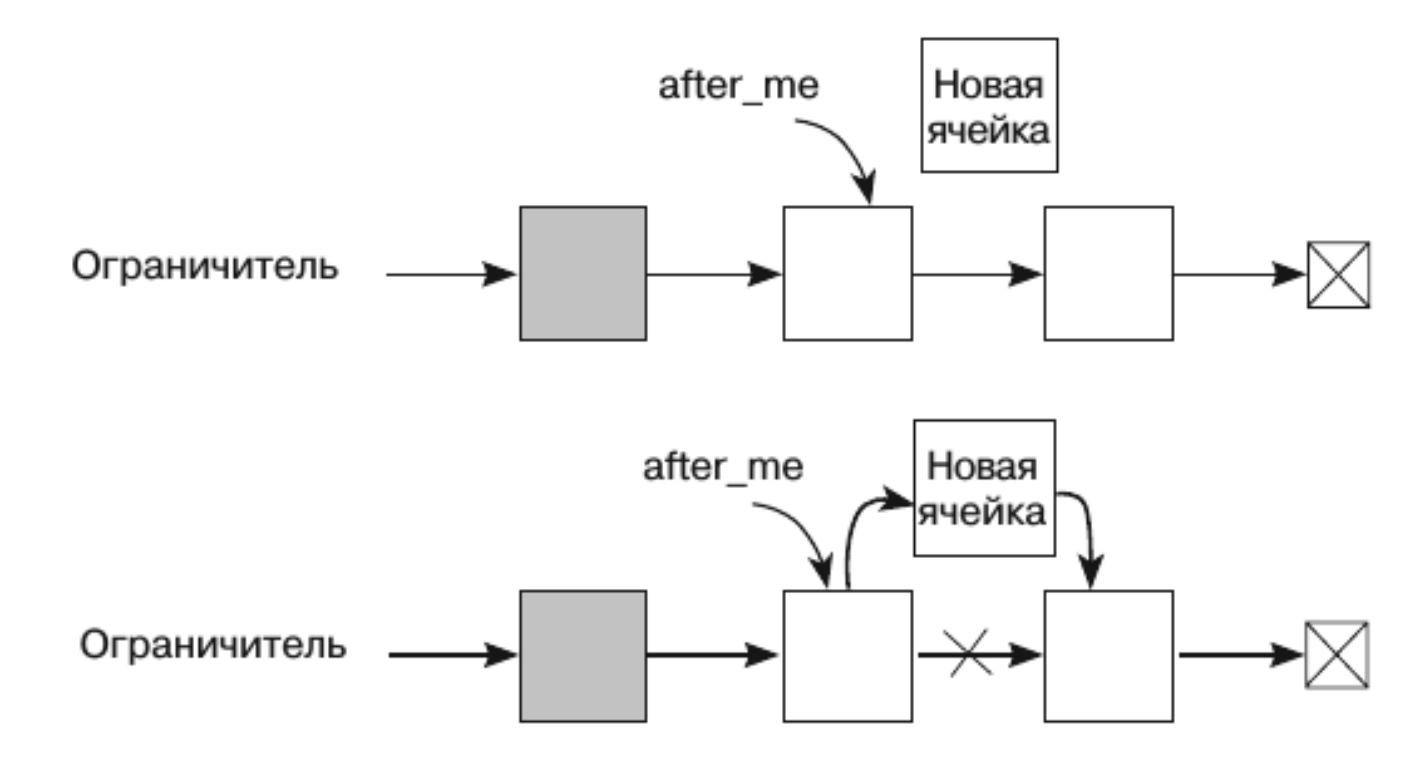
1.1 Добавление в начало



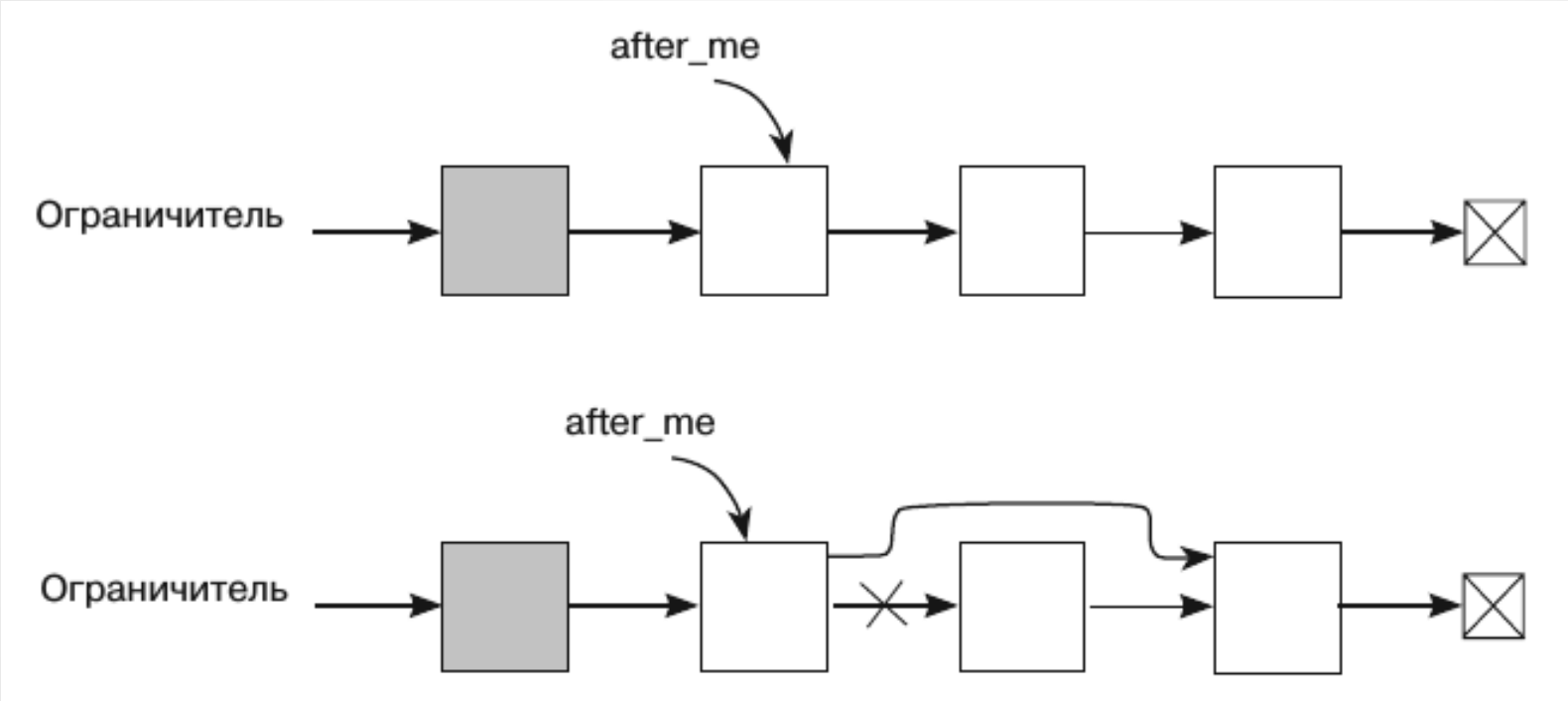
1.2 Добавление в конец



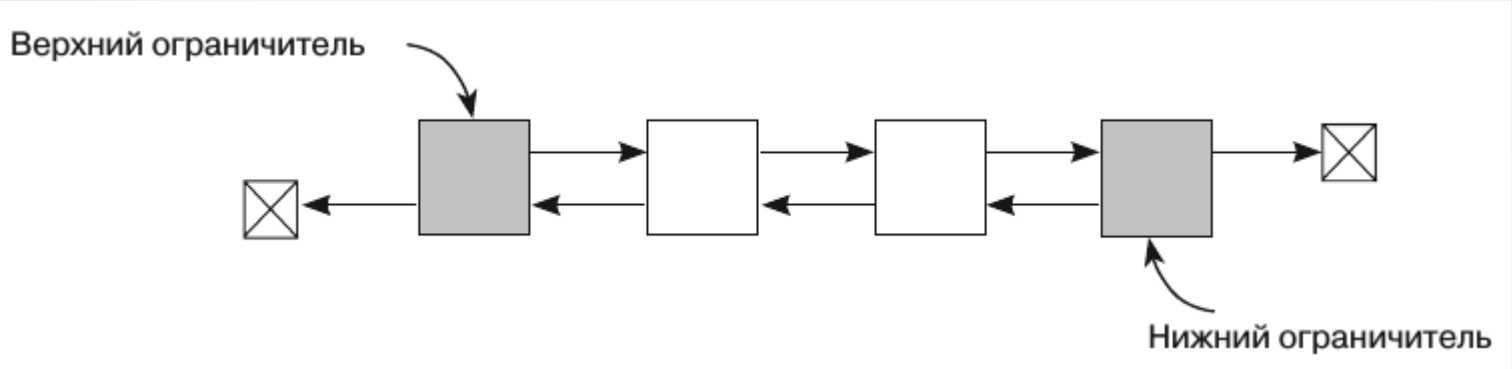
1.3. Вставка


after_me
Новая ячейка
Ограничитель
after_me
Новая ячейка
Ограничитель

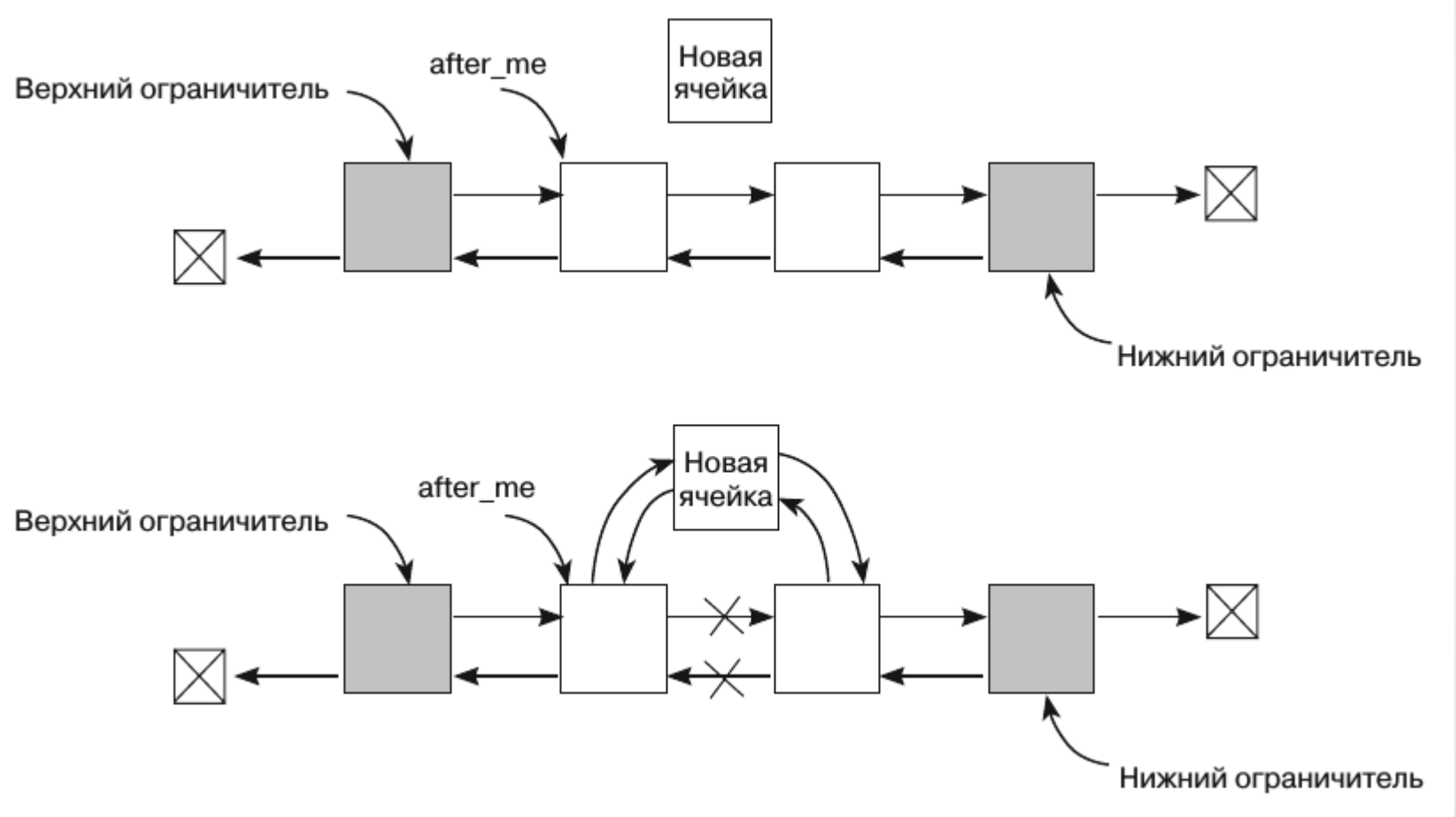
1.4. Удаление



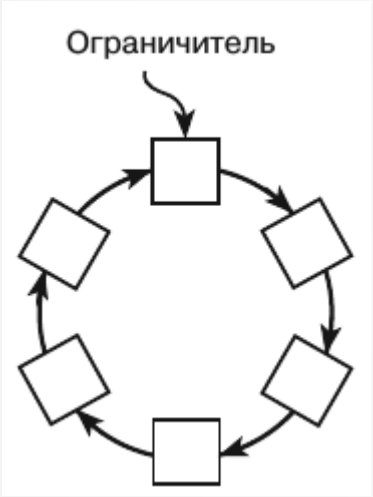
2. Двунаправленный связный список

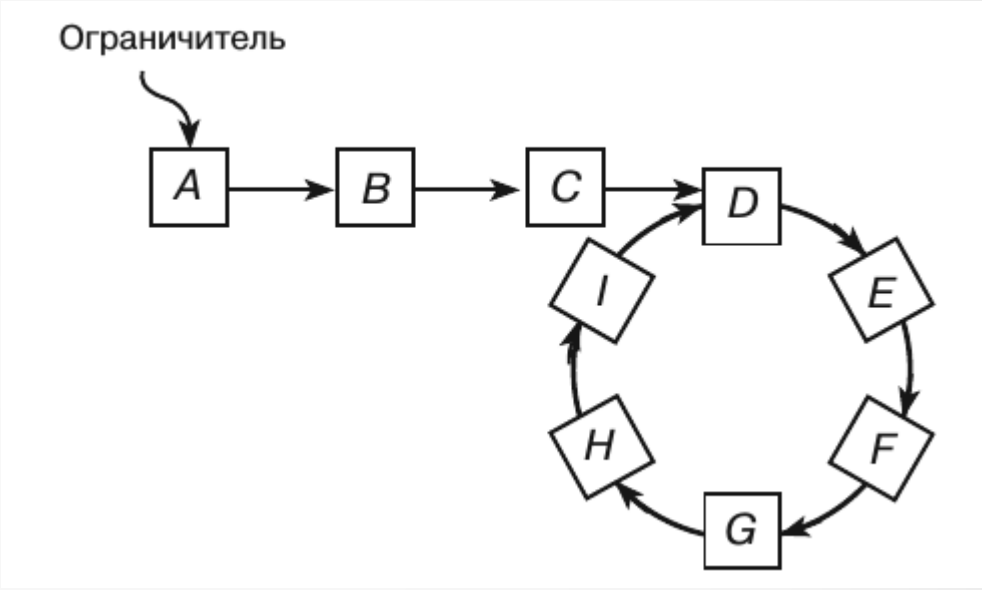


2.1. Вставка элемента

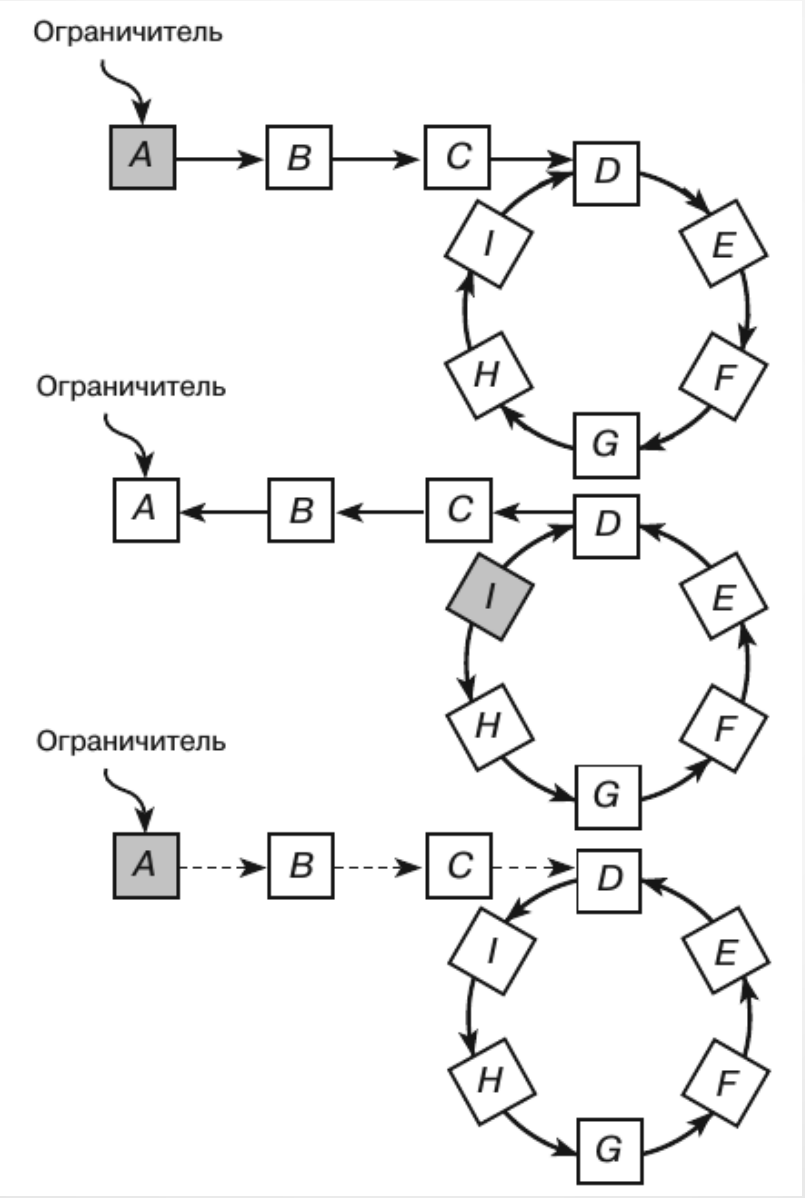


3. Циклический список





5. Реверсирование списка



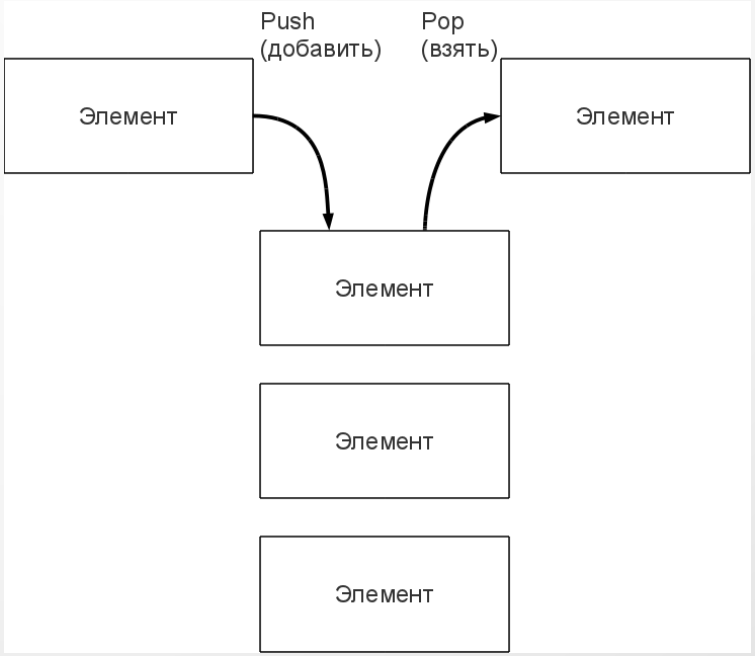
6 Операции со списками

- ✓создание;
- ✓печать (просмотр);
- ✓добавление элемента в конец;
- ✓извлечение элемента из начала;
- ✓проверка пустоты;
- ✓очистка.

Стек и очередь

1. Стек

Стек — это структура данных представляющая собой список элементов, организованных по принципу LIFO (англ. last in — first out, «последним пришёл — первым вышел»).



1.1 Операции со стеком

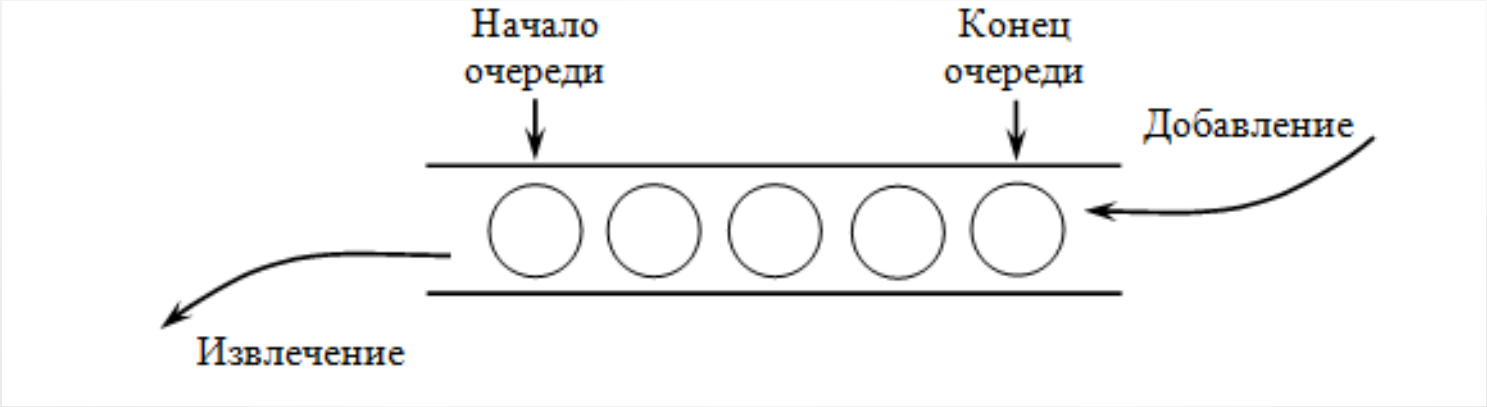
- извлечение из вершины стека (pop);
- добавление в вершину стека (push).

1.1 Операции со стеком

- ✓ создание стека;
- ✓ печать (просмотр) стека;
- ✓ добавление элемента в вершину стека (push);
- ✓ извлечение элемента из вершины стека (pop);
- ✓ проверка пустоты стека;
- ✓ очистка стека.

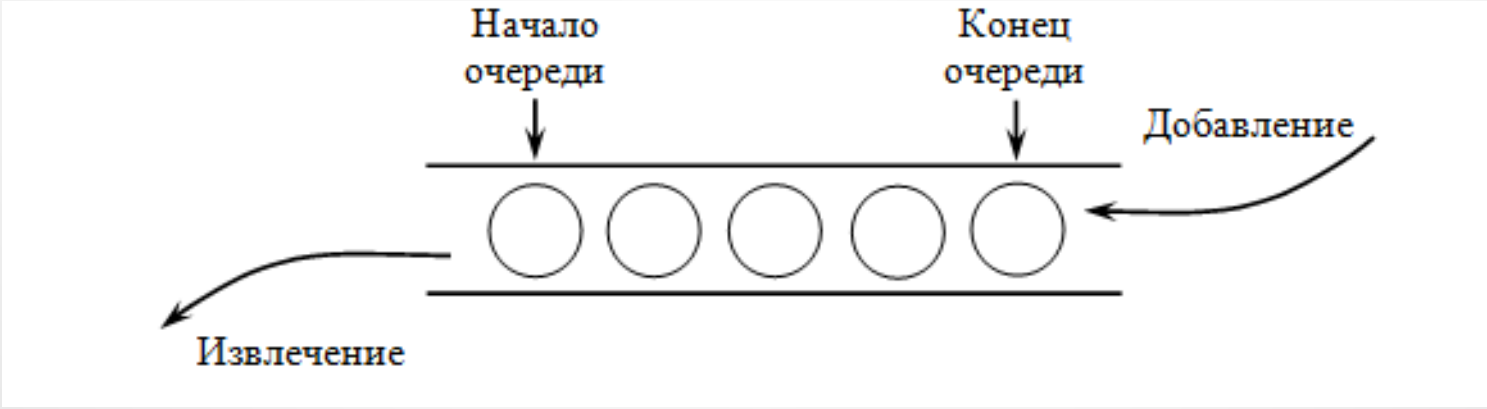
2. Очередь

Очередь — это структура данных представляющая собой список элементов, организованных по принципу FIFO (англ. first in — first out, «первым пришёл — первым вышел»).



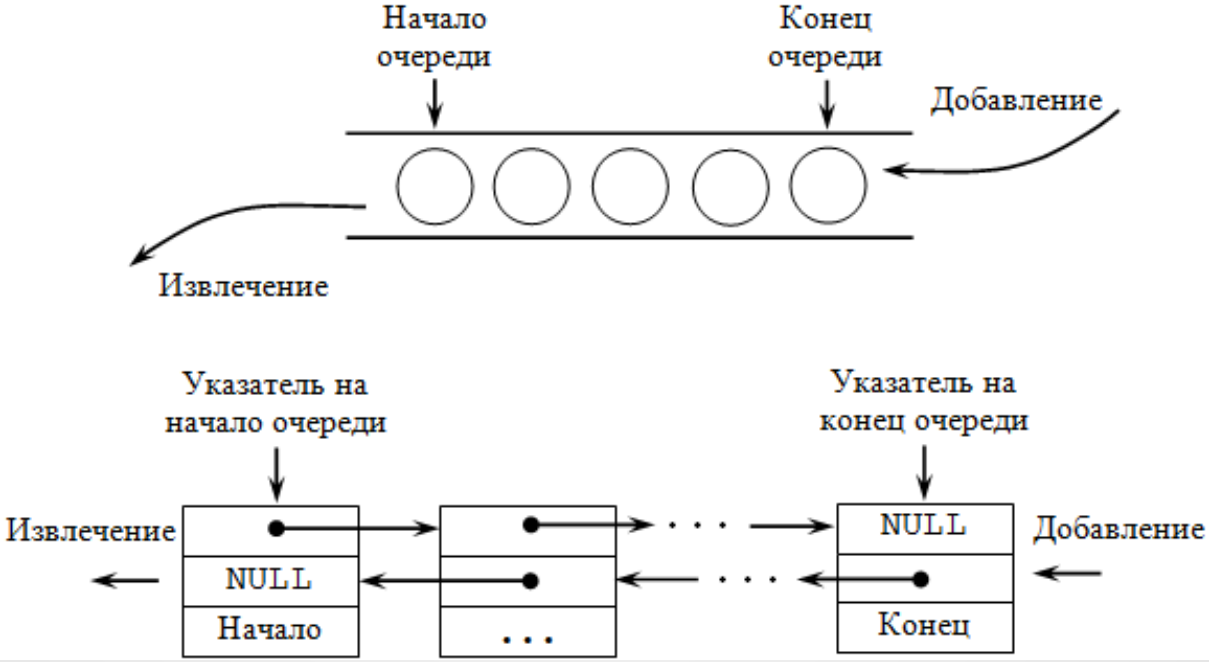
2. Очередь

В очереди доступны два элемента (две позиции): начало очереди и конец очереди. Поместить элемент можно только в конец очереди, а взять элемент только из ее начала. Примером может служить обыкновенная очередь в магазине.



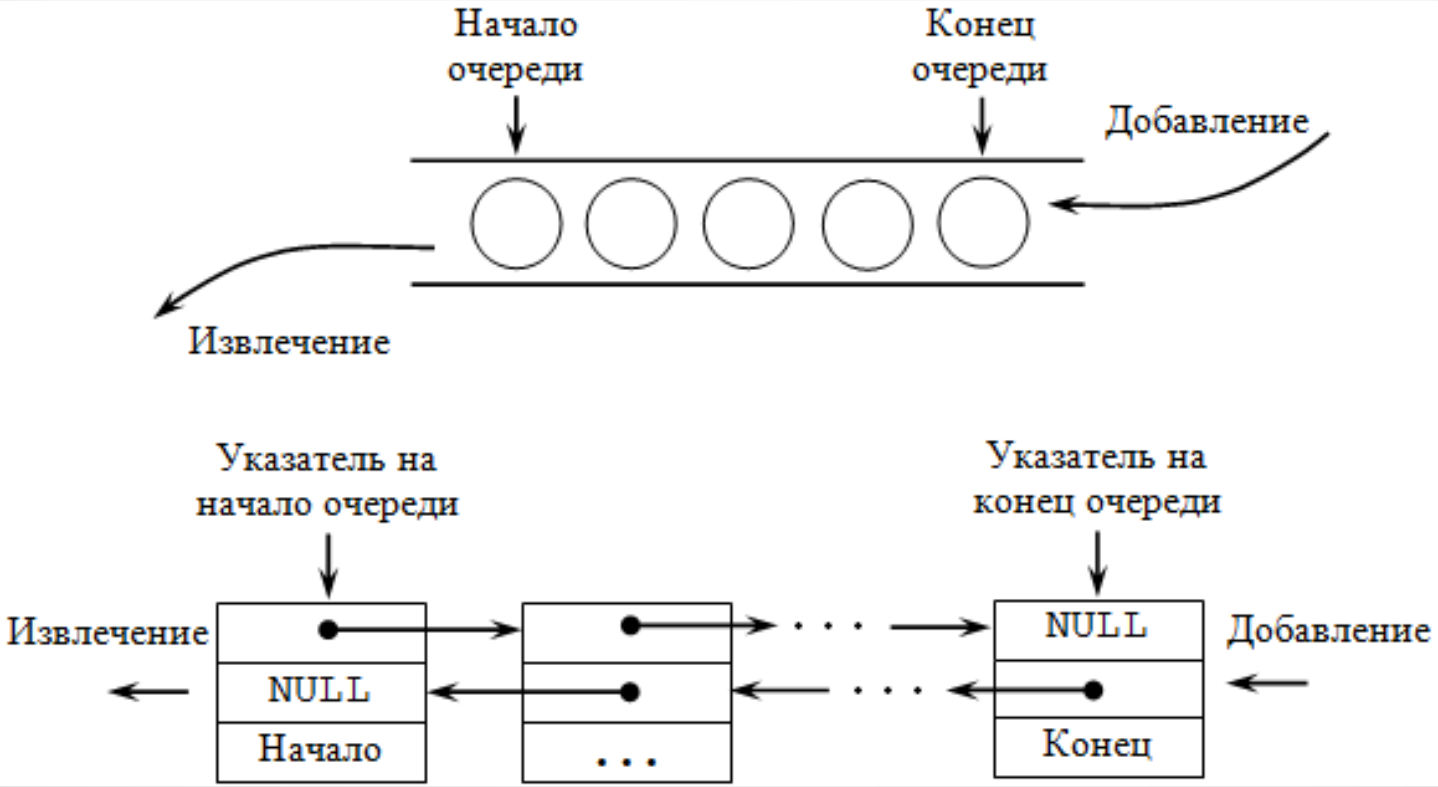
2.1. Организация очереди

Целесообразно хранить два указателя – один на начало списка (откуда извлекаем элементы) и один на конец списка (куда добавляем элементы). Если очередь пуста, то списка не существует, и указатели принимают значение NULL.



2.1. Организация очереди

Очередь можно организовать на основе двунаправленного связного списка



2.2 Операции с очередью

- ✓ создание очереди;
- ✓ печать (просмотр) очереди;
- ✓ добавление элемента в конец очереди;
- ✓ извлечение элемента из начала очереди;
- ✓ проверка пустоты очереди;
- ✓ очистка очереди.