

ОНЛАЙН-ОБРАЗОВАНИЕ

Не забыть включить запись!

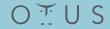




Меня хорошо слышно

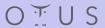


Напишите в чат, если есть проблемы! Ставьте если все хорошо



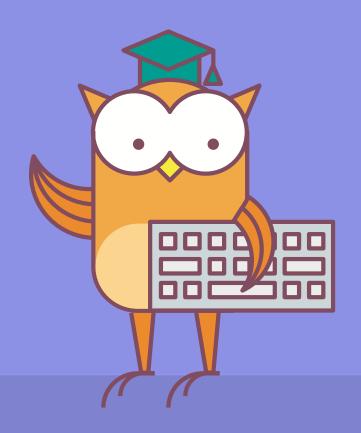
Добрый вечер!







Базовые структуры данных





- Массив
- Динамические массив
- Список
- Стек
- Очередь
- Очередь с приоритетами





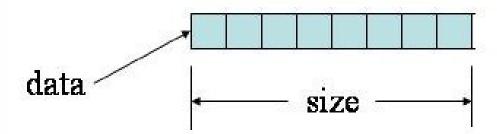
- Какие структуры данных как устроены
- В каком случае какую структуру применять



• Что такое массив?



- Элементы одного типа
- Доступ по индексу



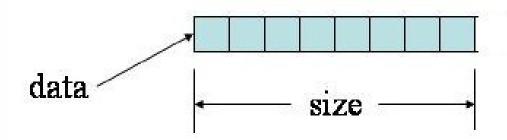


```
• • •
```

```
Java/C#
    int[] a = new int[10];
C++
    int a[10];
    int* a = new int[10];
```

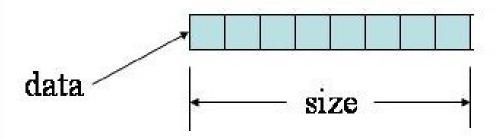


- Элементы одного типа
- Доступ по индексу
- Растет по мере необходимости



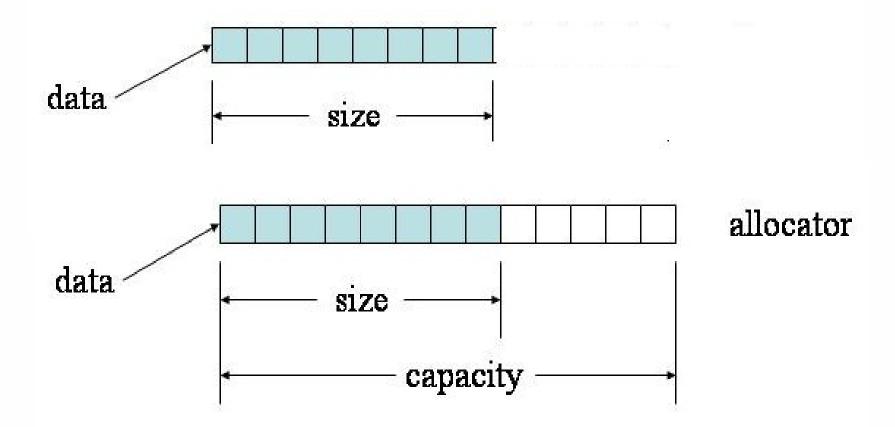


- в материалах к занятию класс DArray
- запустить, посмотреть код
- в чем неэффективность реализации?
- как улучшить?



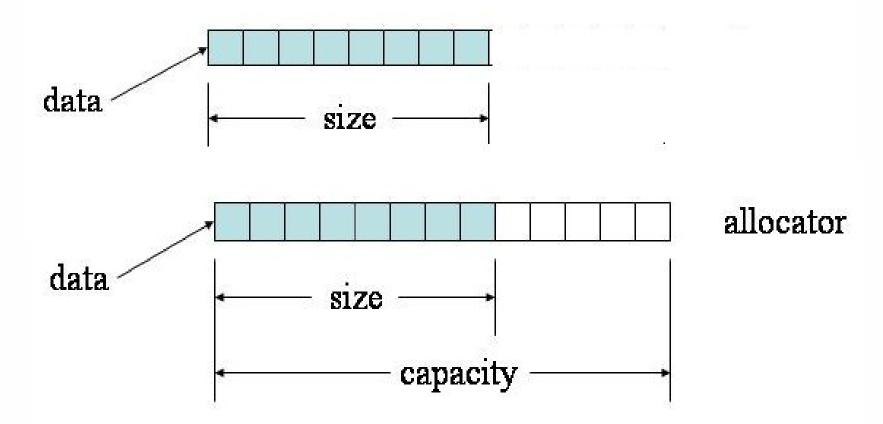


- Элементы одного типа
- Доступ по индексу
- Растет по мере необходимости
- Растет блоками по N элементов



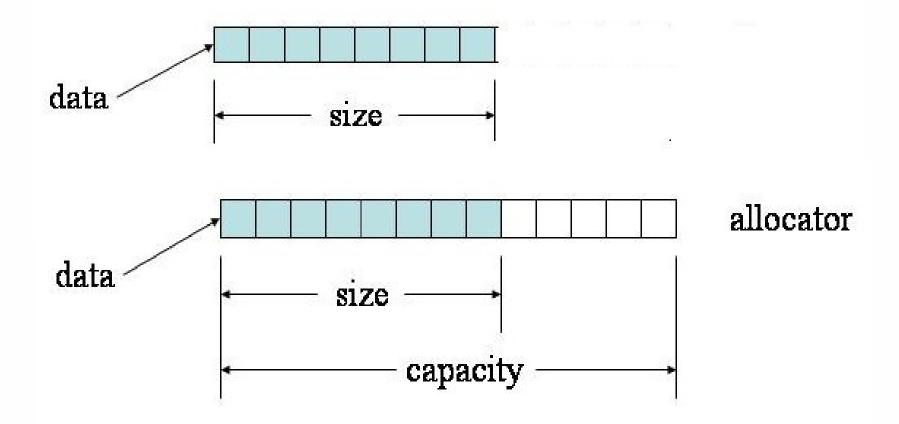


- Реализовать класс BArray
- Сравнить производительность с DArray
- Реализовать remove



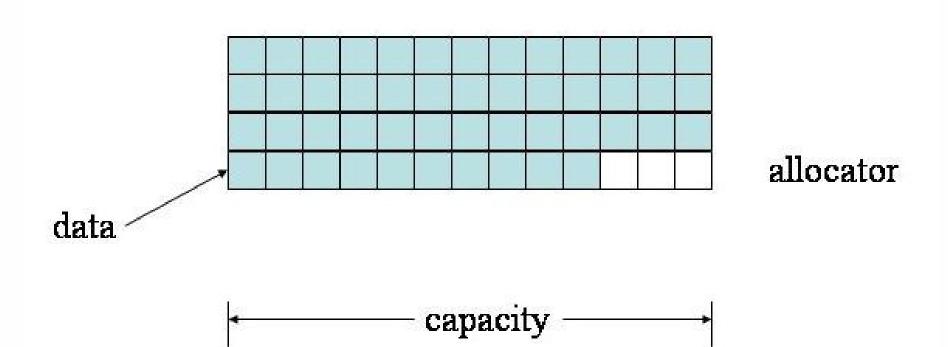


- В чем неэффективность?
- Как можно улучшить?





- Элементы одного типа
- Доступ по индексу
- Растет по мере необходимости
- Растет блоками по N элементов
- При расширении копируется только часть данных



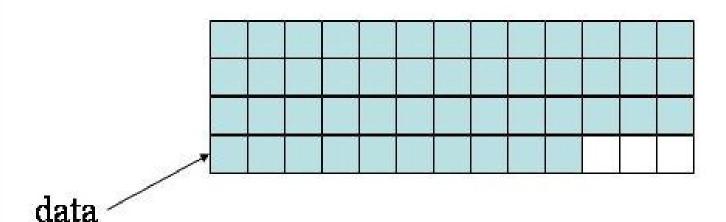


Массив массивов

```
public class IArray<T> {
              BArray<BArray<T>> _arr;
```



```
T at(int index) {
    int index1 = index /_delta;
    int index2 = index % _delta;
    return (T)_arr.get(index1).get(index2);
}
```

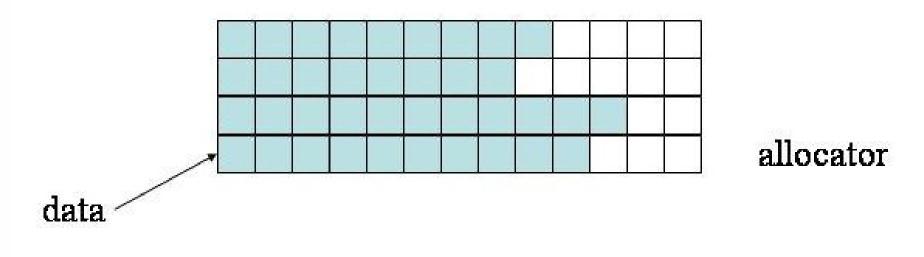


allocator

- capacity ----



- Элементы одного типа
- Доступ по индексу
- Растет по мере необходимости
- Растет блоками по N элементов
- При расширении копируется только часть данных
- Вставка в любое место массива одинаково эффективна



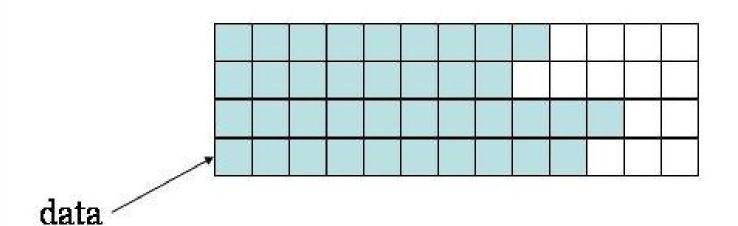
- capacity ----



- Структура данных
- item get()

```
public class IArray<T> {
          BArray<BArray<T>> _arr;
```

Какая реализация эффективнее и почему?

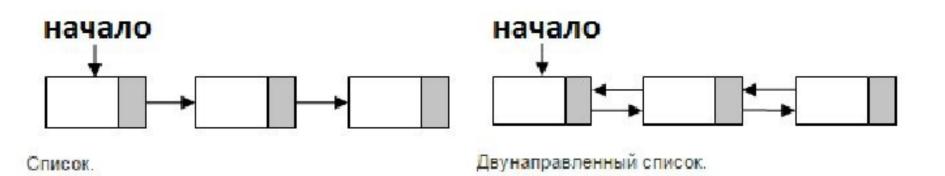


allocator

- capacity ----

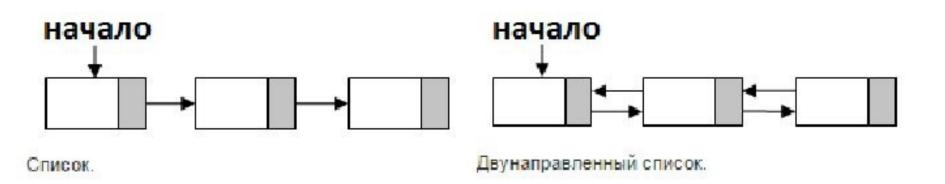


- Элементы одного типа
- Последовательный доступ
- Растет по мере необходимости
- При расширении данные не копируются
- Вставка в любое место списка одинаково эффективна



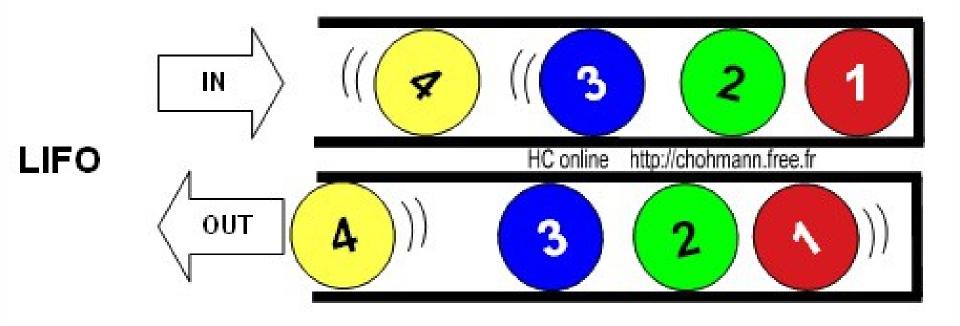


- в материалах к занятию класс OList
- запустить, посмотреть код
- в чем неэффективность реализации?
- как улучшить?



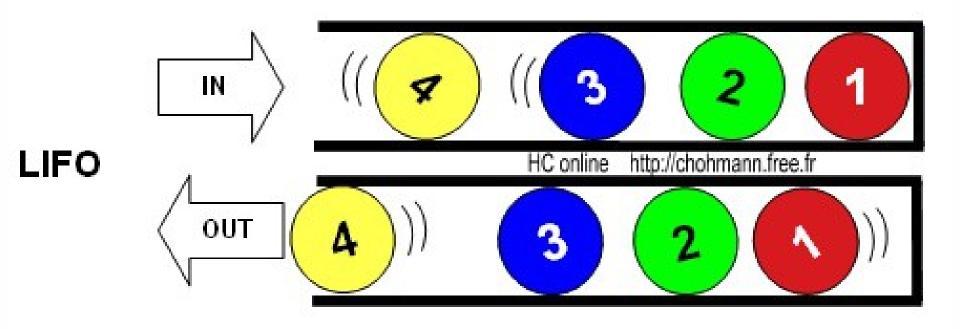


- Элементы одного типа
- Стратегия LIFO (Last In First Out)



Input sequence 1, 2, 3, $4 \neq 0$ utput sequence 4, 3, 2, 1

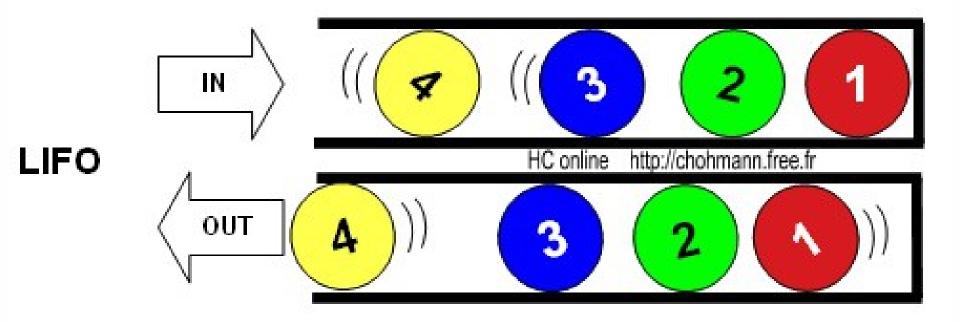
 На какой структуре данных эффективнее сделать стек?



Input sequence 1, 2, 3, $4 \neq 0$ utput sequence 4, 3, 2, 1



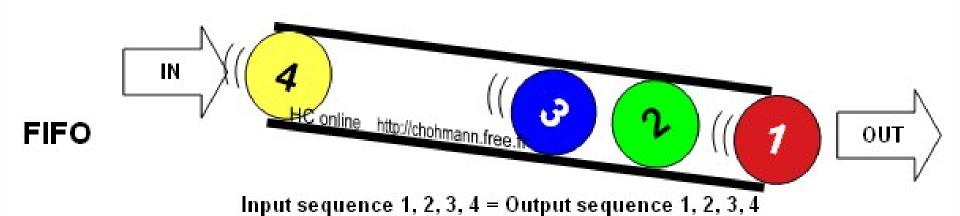
- push(item)
- item pop()



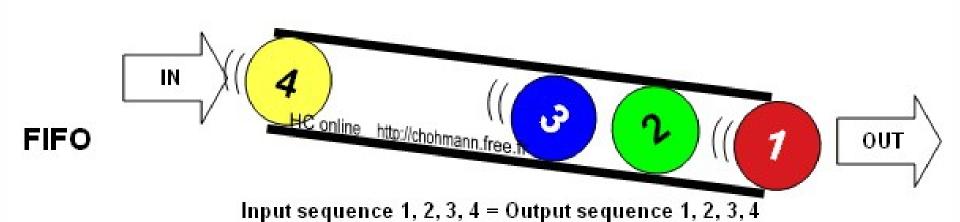
Input sequence 1, 2, 3, $4 \neq 0$ utput sequence 4, 3, 2, 1



- Элементы одного типа
- Стратегия FIFO (First In First Out)

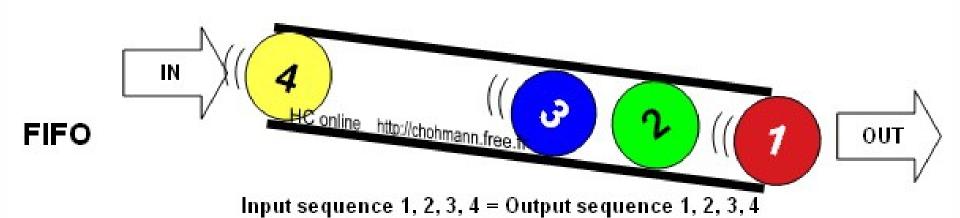


• На какой структуре данных эффективнее сделать очередь?



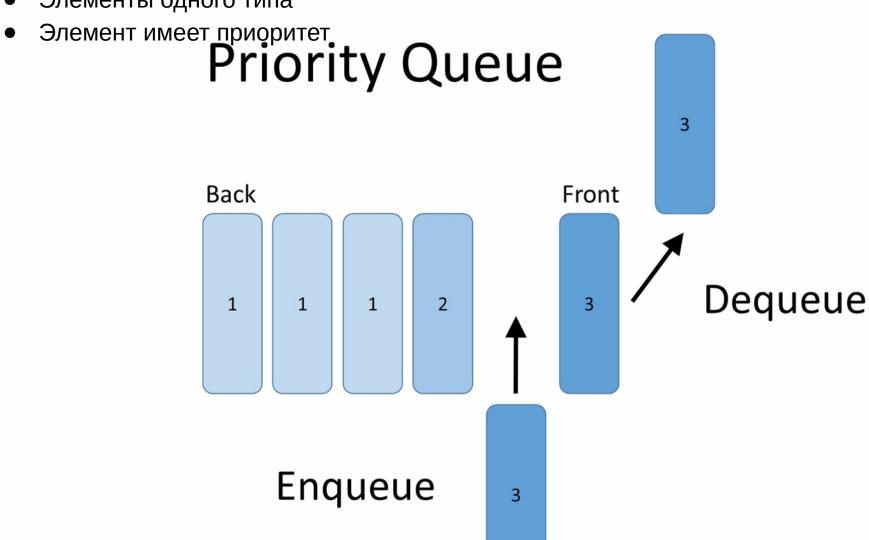


- enqueue(item)
- item dequeue()



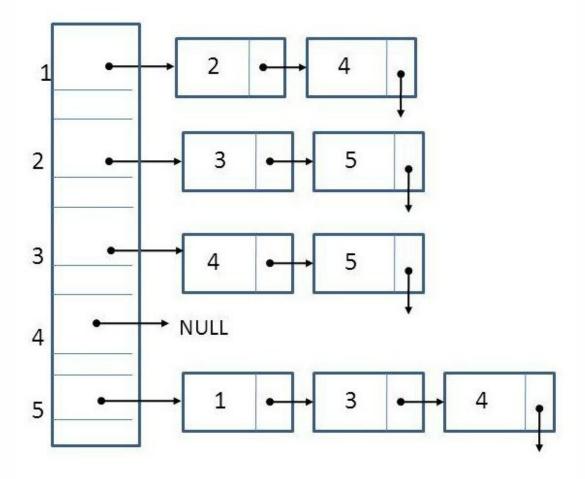


Элементы одного типа





- Элементы одного типа
- Элемент имеет приоритет



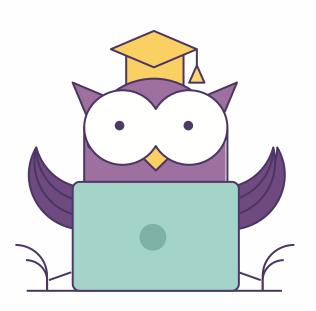


Список списков

```
public class PQueue<T> {
         SortedList<List<T>> _priority;
```



- Массив
- Динамические массив
- Список
- Стек
- Очередь
- Очередь с приоритетами





- IArray массив с быстрой вставкой в любое место
- PQueue очередь с приоритетами



Спасибо за внимание!

