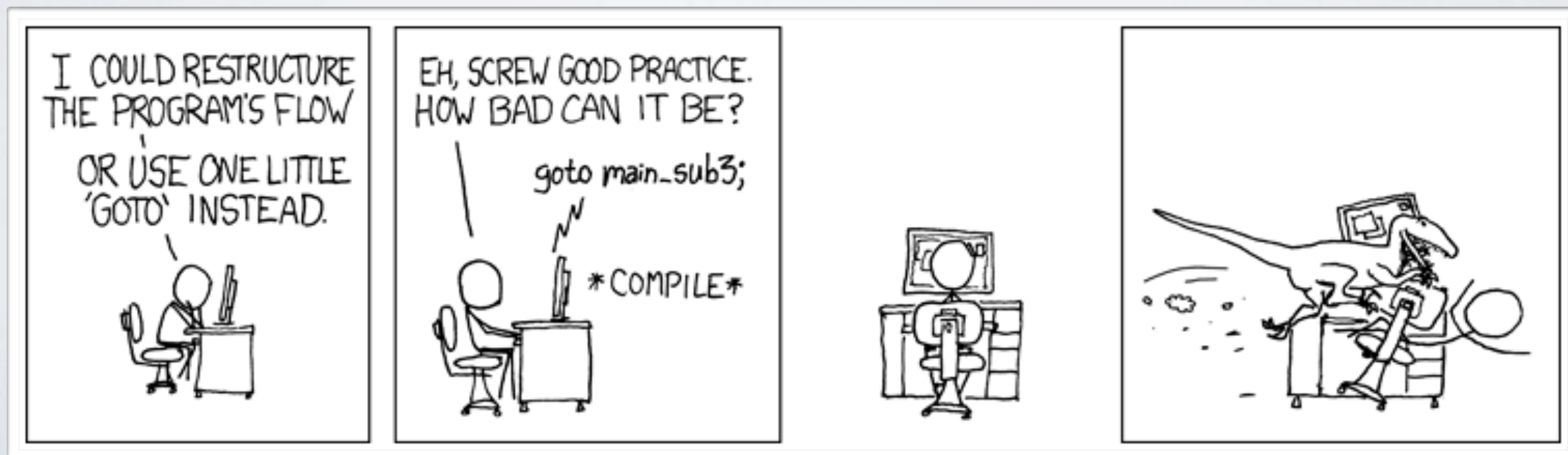


ОСНОВЫ ПРОГРАММНОГО КОНСТРУИРОВАНИЯ

Лекция № 6
10 октября 2016 г.





СОСТАВНЫЕ ТИПЫ ДАННЫХ

ЗАПИСИ

Объединяют разнотипные данные (элементы, поля), относящиеся к одному объекту из предметной области.

```
struct Point {  
    double x, y, z;  
};  
  
struct Planet {  
    const char *name;  
    double mass;  
    double radius;  
    /* .... */  
};
```

```
struct Order {  
    struct User *user;  
    struct Date date;  
    int nitems;  
    struct OrderItem *items;  
};  
  
struct Order o;  
o.nitems = 37;  
  
struct Order *po = &o;  
use(po->nitems);
```

ЗАПИСИ В ПАМЯТИ

Запись представляется участком памяти фиксированного размера. Элементы хранятся рядом друг с другом.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
x								y								z							

```
sizeof(struct Point) // => 24
offsetof(struct Point, x) // => 0
offsetof(struct Point, y) // => 8
offsetof(struct Point, z) // => 16
```

МАССИВЫ

- Множество однотипных данных:
 - Набор временных отсчетов.
 - Векторы и матрицы.
 - Набор строк.
- Элементы массива хранятся в смежных ячейках памяти.



ОДНОМЕРНЫЙ МАССИВ

```
int a[5];
```

Индекс	0	1	2	3	4
Адрес	A_0	?	?	?	?

- A_0 — адрес начала массива, s — размер элемента массива.
- $A(n)$ — адрес n -го элемента массива:
 - $A(n) = A_0 + n \cdot s$

ДВУМЕРНЫЙ МАССИВ

```
int a[2][3];
```

C-style

	j		
i	0	1	2
	3	4	5

или

FORTRAN-style

	j		
i	0	2	4
	1	3	5

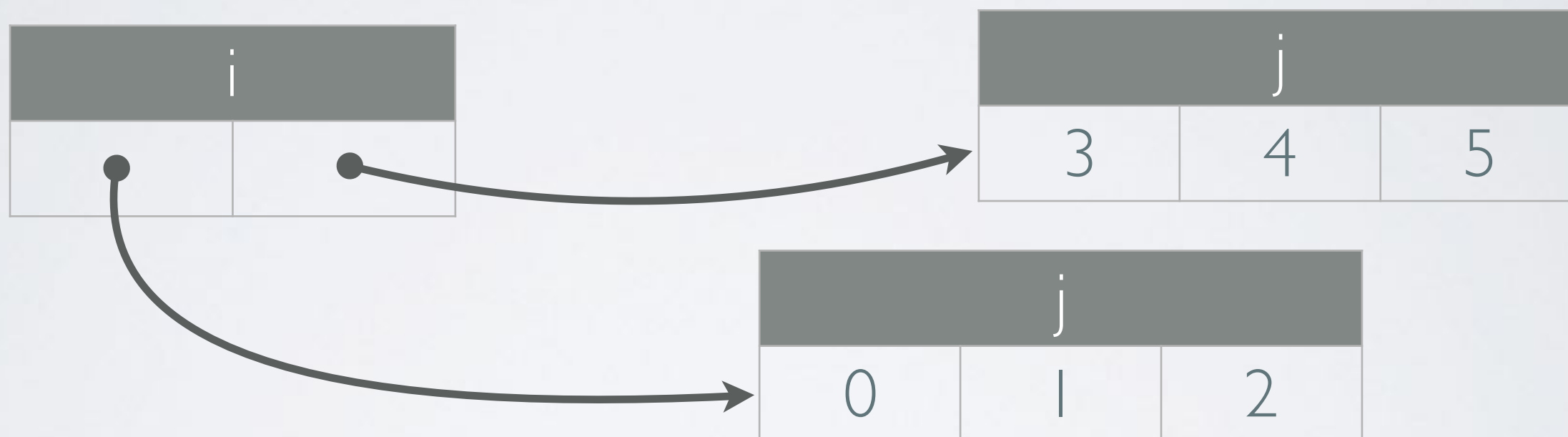
$$n = 3i + j$$

$$n = 2j + i$$

$$A(i, j) = A_0 + n \cdot s$$

МАССИВ МАССИВОВ

```
int a[2][3]; // Java
```

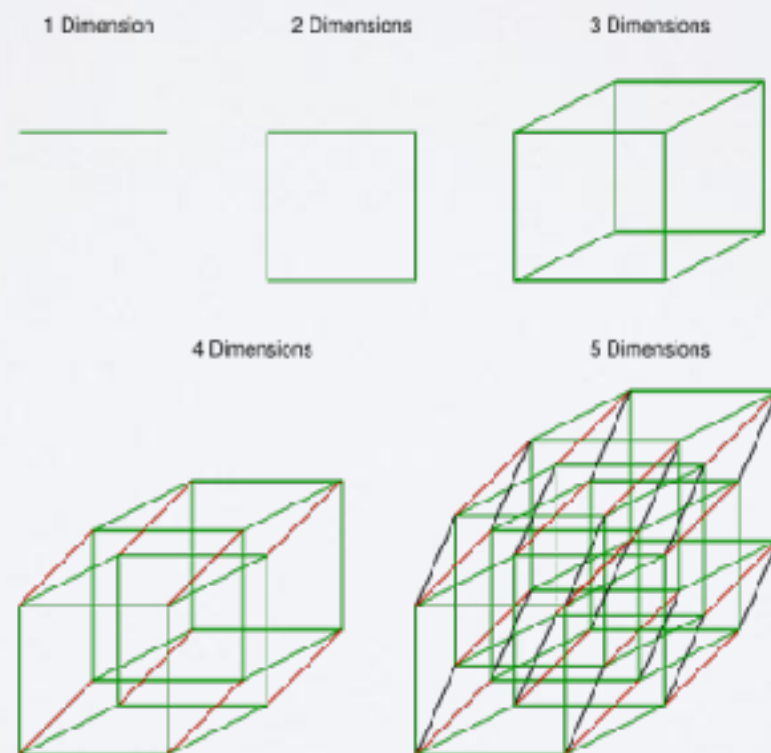


$$A(i, j) = *(A_0 + i \cdot s_p) + j \cdot s,$$

где s_p — размер указателя.

N-МЕРНЫЙ МАССИВ

- K размерностей: $A[N_K][N_{K-1}]....[N_1]$.
- $A(n_K, n_{K-1}, ..., n_1) = A_0 + s \cdot n(n_K, n_{K-1}, ..., n_1)$.



C-STYLE

- Уменьшаем порядок задачи на единицу ($K \rightarrow K-1$), «откусывая» последнюю размерность:
 - $n(n_K, n_{K-1}, \dots, n_1) = n(n_K, n_{K-1}, \dots, n_2, 0) + n_1$
 - $n(n_K, n_{K-1}, \dots, n_2, 0) = n(n_K, n_{K-1}, \dots, n_2) \cdot N_1$
- В итоге:
 - $n(n_K, n_{K-1}, \dots, n_1) = (((\dots(n_K \cdot N_{K-1} + n_{K-1}) \dots)) \cdot N_2 + n_2) \cdot N_1 + n_1$

FORTRAN-STYLE

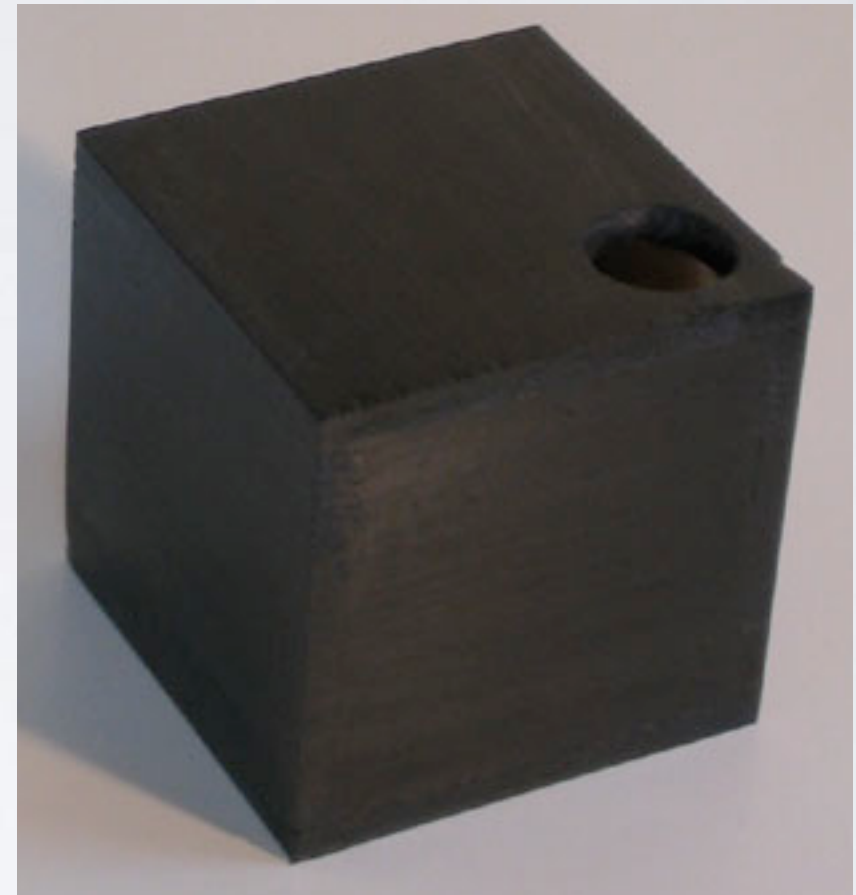
- «Откусываем» первую размерность:
 - $n(n_K, n_{K-1}, \dots, n_1) = n_K + n(0, n_{K-1}, \dots, n_1)$
 - $n(0, n_{K-1}, \dots, n_1) = N_K \cdot n(n_{K-1}, \dots, n_1)$
- В итоге:
 - $n(n_K, n_{K-1}, \dots, n_1) = n_K + N_K \cdot (n_{K-1} + N_{K-1} \cdot ((\dots (n_2 + N_2 \cdot n_1) \dots)))$

ДВА СПОСОБА СМОТРЕТЬ НА ДАННЫЕ



«Белый ящик»
(прозрачный)

Устройство



«Черный ящик»
(непрозрачный)

Операции

ОПЕРАЦИИ НАД ДАННЫМИ

- Записи:

- Прочитать элемент по имени.
- Изменить элемент по имени.
- ~~Добавить новый элемент.~~
- ~~Удалить элемент.~~

- Массив:

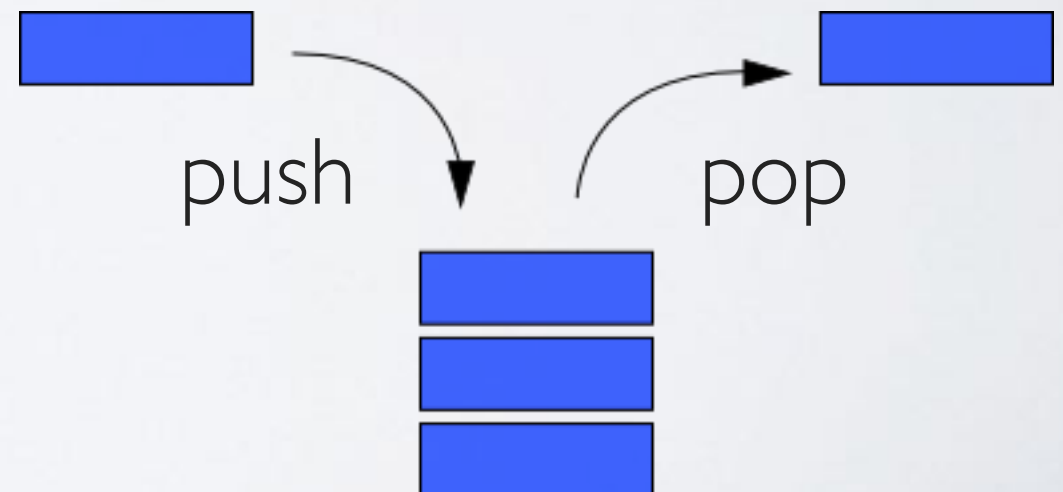
- Прочитать элемент по индексу.
- Изменить элемент по индексу.
- ~~Изменить размер массива.~~
- ~~Добавить элемент.~~
- ~~Удалить элемент.~~

«ПРОДВИНУТОСТЬ» СТРУКТУР ДАННЫХ

- Количество «степеней свободы» (возможных операций со структурой данных).
- Рост количества степеней свободы возможен:
 - за счет усложнения интерфейса;
 - за счет усложнения внутренней структуры;
 - за счет понижения быстродействия.

СТЕК

- Абстрактный тип данных.
- Базовые операции:
 - Вставка элемента (push).
 - Извлечение элемента (pop).
- Принцип LIFO (last-in, first-out) – «последним вошел, первым вышел».

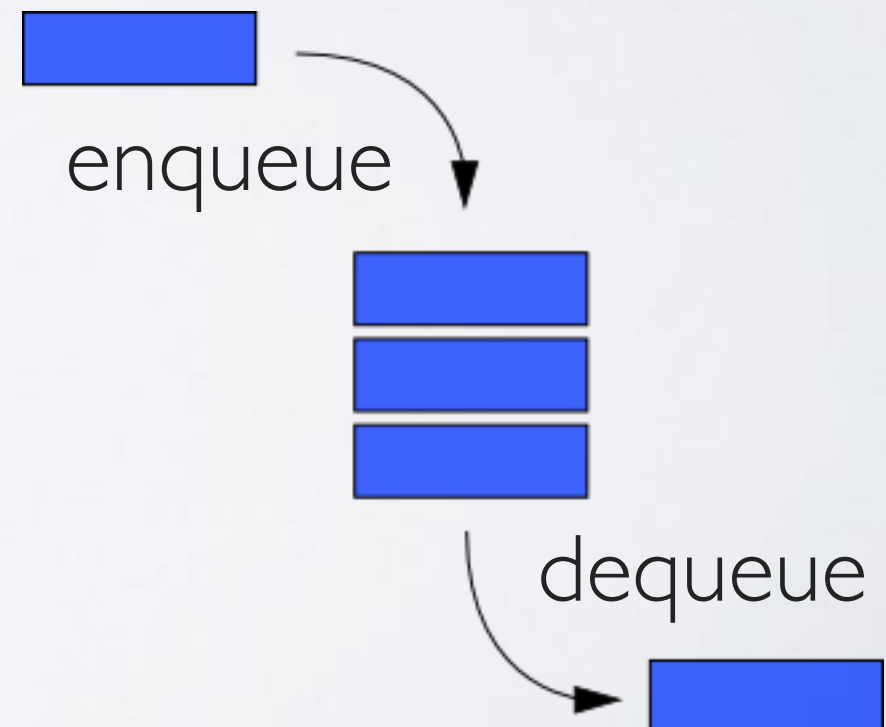


ПРИМЕНЕНИЯ СТЕКА

- Машинный стек.
- Вычисление выражений в обратной польской записи:
 - $(1 + 2 * 4 + 3) \rightarrow 1\ 2\ 4\ *\ +\ 3\ +$
 - Язык Forth, Postscript.
- Поиск пути в лабиринте.
- Задачи с иерархией элементов (обход дерева).

ОЧЕРЕДЬ

- Абстрактный тип данных.
- Базовые операции:
 - Вставка элемента (enqueue).
 - Извлечение элемента (dequeue).
- Принцип FIFO (first-in, first-out) – «первым вошел, первым вышел».



ПРИМЕНЕНИЕ ОЧЕРЕДЕЙ

Обработка отложенных запросов.

Порождение
запросов



ooooooooooooooooочередь



Обработка
запросов

СТЕК ЧЕРЕЗ СТАТИЧЕСКИЙ МАССИВ

- Статический массив достаточно большого размера.

```
int stack[99999];  
int sp = 0;
```

```
stack[sp] = 10;  
sp++;
```

```
stack[sp] = 20;  
sp++;
```

```
sp--;  
stack[sp]; // => 20
```

СТЕК ЧЕРЕЗ ДИНАМИЧЕСКИЙ МАССИВ

- Динамически расширяющийся массив.
- Храним N элементов в массиве размера M ($N \leq M$).
- Если при добавлении N становится больше M , то увеличиваем массив до размера $2M$, копируя содержимое.
- Если при удалении N становится меньше $M/4$, то уменьшаем массив до размера $M/2$, копируя содержимое.

СТЕК ЧЕРЕЗ ДИНАМИЧЕСКИЙ МАССИВ

- Временная сложность операции в стеке:
 - В лучшем случае потребуется просто изменить элемента массива: $O(1)$.
 - В худшем случае потребуется скопировать все N элементов и затем изменить один элемент: $O(N)$.
- Амортизационная сложность (среднее в худшем случае) операций добавления и удаления: $O(1)$.

ЗАДАЧКА О СЛОЖНОСТИ

- Докажите, что амортизационная временная сложность операции добавления в стек, реализованный с помощью динамически расширяющегося массива, равна $O(1)$.

2

2 7



2 7 1

2 7 1 3



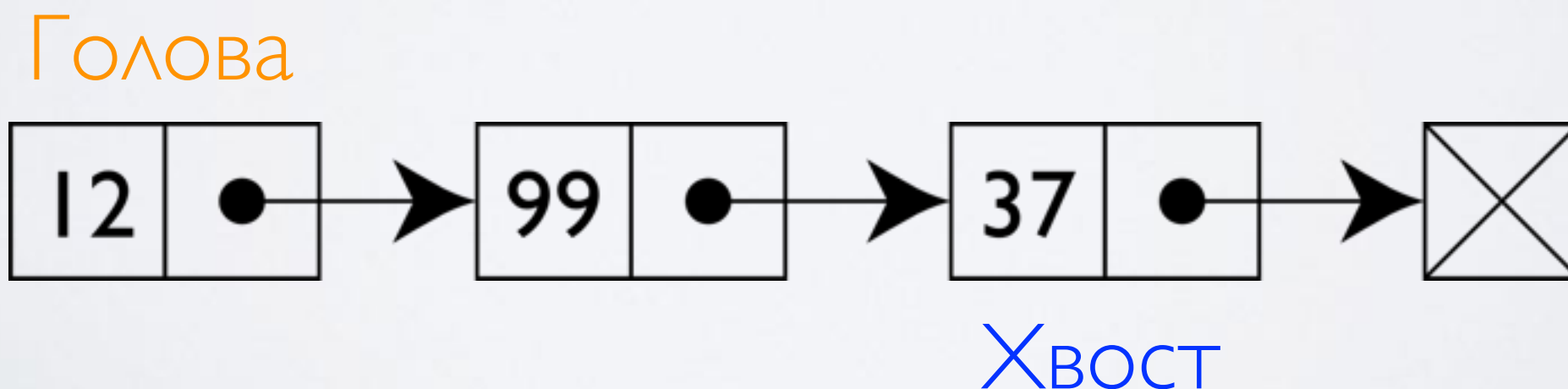
2 7 1 3 8

2 7 1 3 8 4

- Подсказка: оцените сложность добавления N элементов в стек.

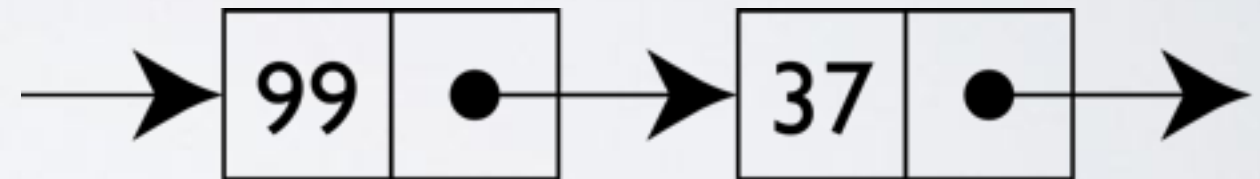
СВЯЗНЫЕ СПИСКИ

Связный список (linked list) – линейно упорядоченный набор элементов, каждый из которых содержит связь со следующим элементом.



ОПЕРАЦИИ СО СВЯЗНЫМ СПИСКОМ

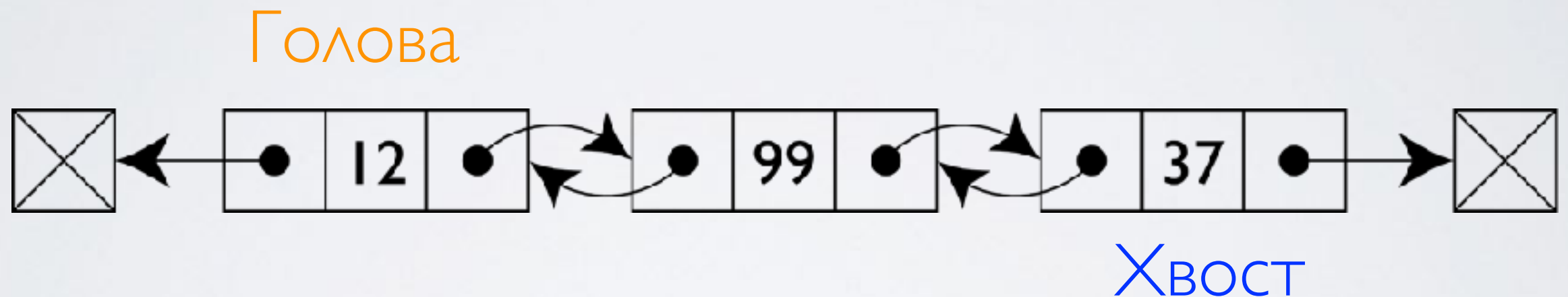
- Вставка элемента в голову.
- Вставка в середину.
- Удаление элемента.
- Обход списка.
- Поиск элемента по критерию.



СВОЙСТВА СВЯЗНОГО СПИСКА

- Динамическая структура данных.
- Вставка и удаление выполняются за $O(1)$.
- Медленный поиск по номеру (индексирование): $O(n)$.

ДВУСВЯЗНЫЙ СПИСОК



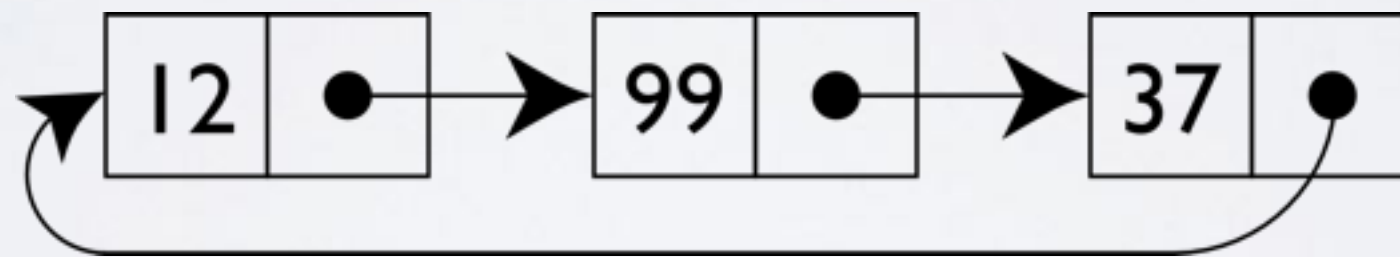
*Преимущество: одинаковая легкость
операций в обе стороны*

ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ СПИСКОВ

- Многочлены (символьная алгебра).
- Реализация стеков и очередей.
- Цепочки кластеров в файловых системах.

КОЛЬЦЕВОЙ СВЯЗНЫЙ СПИСОК

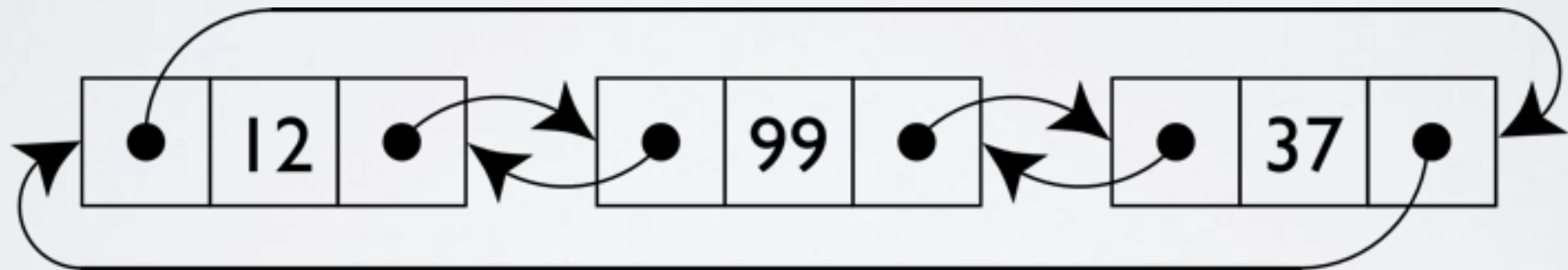
Голова?



ПРИМЕНЕНИЕ КОЛЬЦЕВЫХ СПИСКОВ

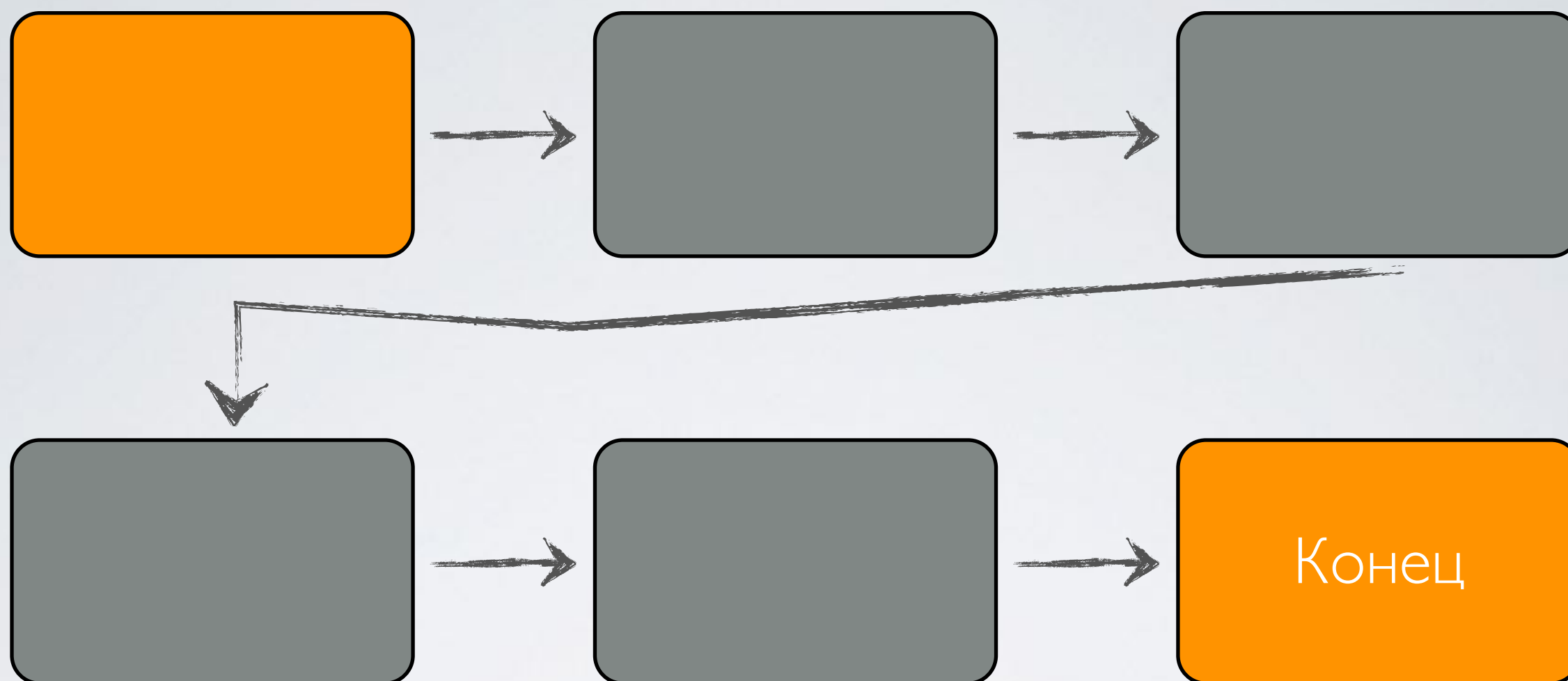
- Список вершин многоугольника.
- Список процессов в системе с разделением времени.
- Кольцо буферов ввода/вывода.

ДВУСВЯЗНЫЙ КОЛЬЦЕВОЙ СПИСОК



ИЕРАРХИЯ

- Абстрактные типы данных:
 - Последовательности (стек, очередь, дек)
 - Множества
 - Словари
 - Графы
- Структуры данных:
 - Массив
 - Списки
 - Деревья (дерево поиска, двоичная куча)
 - Хеш-таблицы



КОНЕЦ ШЕСТОЙ ЛЕКЦИИ

Чем больше степеней свободы, тем лучше.
Если не приходится за это слишком дорого платить.