Списки на основе массива

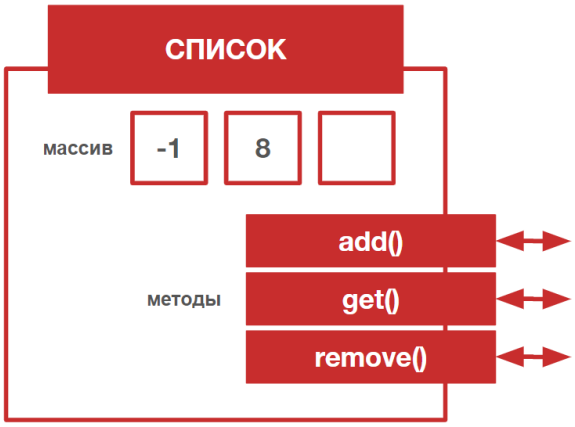
СПИСКИ

Список — это абстрактный тип данных, представляющий собой упорядоченную коллекцию элементов, с которой ассоциирован широкий спектр операций, таких как добавление, получение и удаление элементов. Это позволяет пользователям выполнять динамическое управление данными, в отличие от более статичных массивов.

Хотя списки и массивы подразумевают хранение последовательности элементов, ключевое отличие заключается в том, что массивы — это примитивная структура данных, обеспечивающая основу для хранения элементов в памяти. Массивы характеризуются фиксированным размером и прямым доступом к элементам по индексу, что делает их быстрыми и эффективными для определенных операций. Однако, их статичность ограничивает гибкость использования в динамических ситуациях.

В отличие от массивов, списки представляют собой более высокоуровневую абстракцию, включающую не только способ хранения элементов, но и комплекс операций управления этими элементами. Списки могут быть реализованы через различные структуры данных, в том числе через массивы, связные списки и другие, каждая из которых предлагает разные преимущества в зависимости от требуемых операций.

Реализация списков через классы, действующие как оболочки вокруг массивов, позволяет сочетать простоту и доступность массивов с дополнительными возможностями управления, такими как автоматическое расширение в зависимости от количества элементов. Эти классы-оболочки обеспечивают богатый набор операций для работы с данными, включая добавление элементов в конец или начало списка, вставку по индексу, удаление и доступ к элементам, а также многие другие функции, которые не так просто реализовать с использованием обычных массивов.



РЕАЛИЗАЦИЯ СПИСКА НА ОСНОВЕ МАССИВА

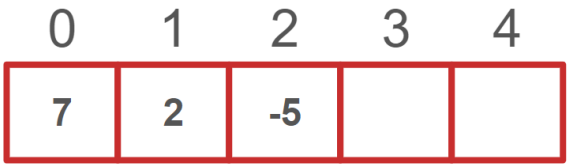
Для создания списка на основе массива, необходимо сначала определить начальный размер этого массива, который будет служить хранилищем элементов списка. Этот размер задаётся как стартовое значение, от которого будет начинаться работа списка. Важно также предусмотреть конструктор, который инициирует массив с этим начальным размером.

Однако, в процессе работы со списком, количество реально хранящихся в нем элементов может отличаться от физического размера массива. Для отслеживания актуального количества элементов в списке используется специальное поле, например, **`count`**. Это поле позволяет понимать, сколько элементов на данный момент находится в списке, независимо от его текущего физического размера.

```
class MyArrayList {

    // начальный размер массива
    private static int INITIAL_SIZE = 10;
    // массив для хранения элементов
    private int[] data;
    // количество элементов списка
    private int count = 0;

    public MyArrayList() {
        // создание массива
        this.data = new int[INITIAL_SIZE];
    }
}
```



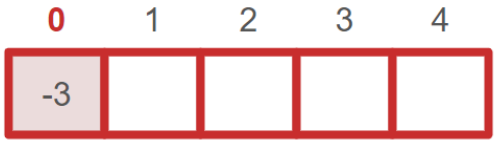
Длина массива имеет значение 5, при этом count должен быть равен 3

Такая структура предоставляет основу для дальнейшего расширения функциональности списка, включая добавление новых элементов, удаление существующих, доступ к элементам по индексу и другие операции, обычно ассоциируемые со списками. При этом, в зависимости от реализации, может потребоваться добавление логики для автоматического расширения массива, когда текущее количество элементов превышает его размер.

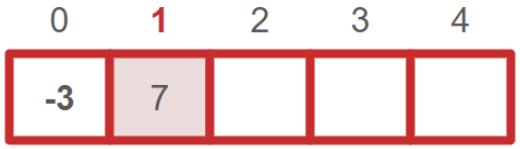
ДОБАВЛЕНИЕ ЭЛЕМЕНТА В СПИСОК

В реализации списка на основе массива, добавление нового элемента в конец списка осуществляется путем размещения элемента в первой свободной позиции массива, которая идентифицируется по значению переменной **`count`**, отражающей текущее количество элементов в списке.

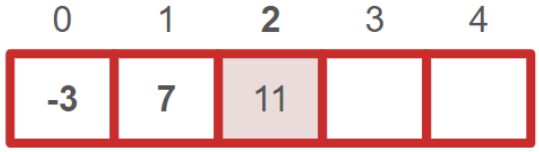
Пример работы алгоритма:



Значение count = 0. Новый добавляемый элемент -3 попадает в data[count] или data[0]. Увеличиваем count



Значение count = 0. Новый добавляемый элемент 7 попадает в data[count] или data[1]. Увеличиваем count



Значение count = 2. Новый добавляемый элемент 11 попадает в data[count] или data[2]. Увеличиваем count

Реализация алгоритма добавления:

```
public void add(int element) {
    data[count] = element;
    count++;
}
```

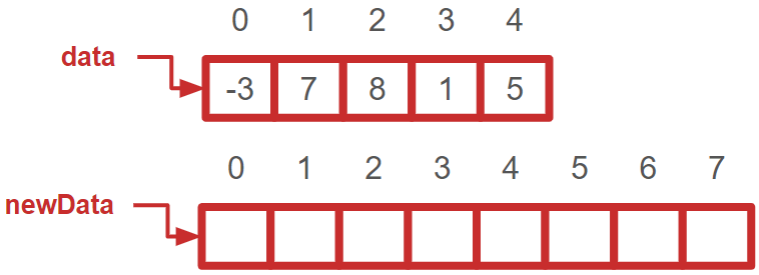
Этот подход обеспечивает простой и эффективный механизм для добавления элементов в конец списка, позволяя легко расширять его содержимое.

РАСШИРЕНИЕ СПИСКА

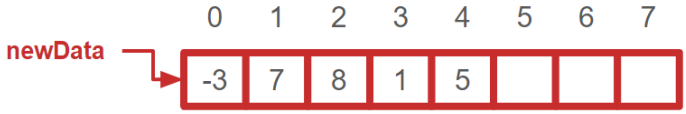
Когда размер массива, используемого для реализации списка, достигает своего предела, и необходимо добавить еще элементы, возникает необходимость в расширении этого

массива. Поскольку прямое изменение размера существующего массива невозможно в языках программирования, таких как Java, требуется создание нового массива большего размера и последующее копирование элементов из старого массива в новый.

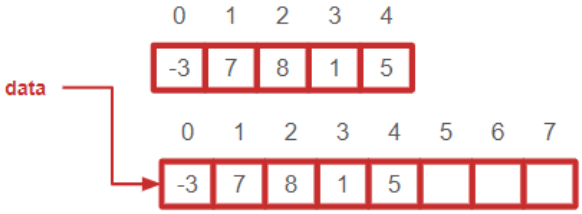
Сначала определяется новый размер массива (обычно в полтора раза больше текущего размера, чтобы уменьшить количество операций расширения при последующих добавлениях) и создается новый массив **`newData`** этого размера.



Далее все существующие элементы из старого массива **`data`** копируются в новый массив **`newData`**



После копирования всех элементов ссылка **`data`** обновляется так, чтобы она указывала на новый массив **`newData`**. Старый массив **`data`** становится недостижимым для последующего кода, и его память будет освобождена сборщиком мусора автоматически.



В результате этих действий размер хранилища списка увеличивается, и можно добавлять новые элементы без риска переполнения массива. Этот процесс обеспечивает управление размером внутреннего хранилища списка, делая его гибким и адаптируемым к различным сценариям использования.

УДАЛЕНИЕ ЭЛЕМЕНТА

При удалении элемента из списка, реализованного на основе массива, по его индексу требуется осуществить сдвиг оставшихся элементов влево, чтобы заполнить образовавшийся пробел и сохранить непрерывность данных в массиве.