SWIFT
Data Structure - Dequeue

Bill Kim(김정훈) | ibillkim@gmail.com

목차

Dequeue

데크(Dequeue)는 Doubly-ended Queue의 약자로서 양쪽 끝에서 추가, 삭제가 가능한 선형 구조 형태의 자료구조입니다.

스택과 큐의 장점을 모아서 만들어진 형태입니다.

추가, 삭제가 되는 부분의 명칭을 보통 Front, Rear이라고 명칭합니다.

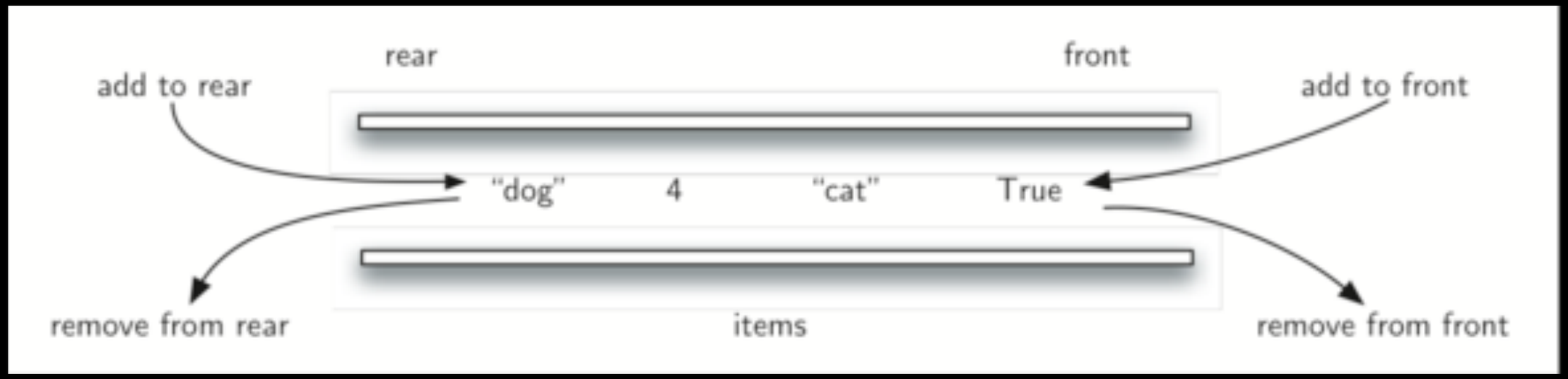
데크(Dequeue)에는 제약 조건을 걸어 사용 목적을 달리하는 구조 또한 있습니다.

그것은 바로 입력제한데크(Scroll)와 출력제한데크(Shelf)가 있습니다.

Concept

Dequeue의 기본 동작 흐름은 아래와 같습니다.

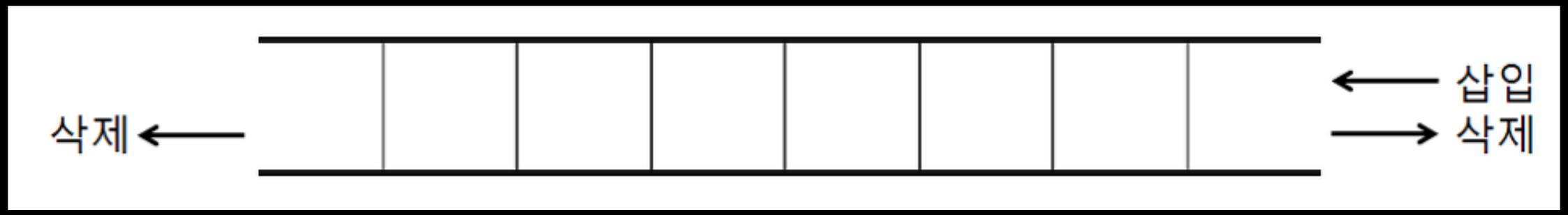
- 입력과 출력이 양방향 가능
- 입력과 출력의 순서가 맘대로 정할 수 있음
- Enqueue(추가) 및 Dequeue(삭제) 실행 속도는 O(1)



Concept

입력제한데크(Scroll)의 기본 동작 흐름은 아래와 같습니다.

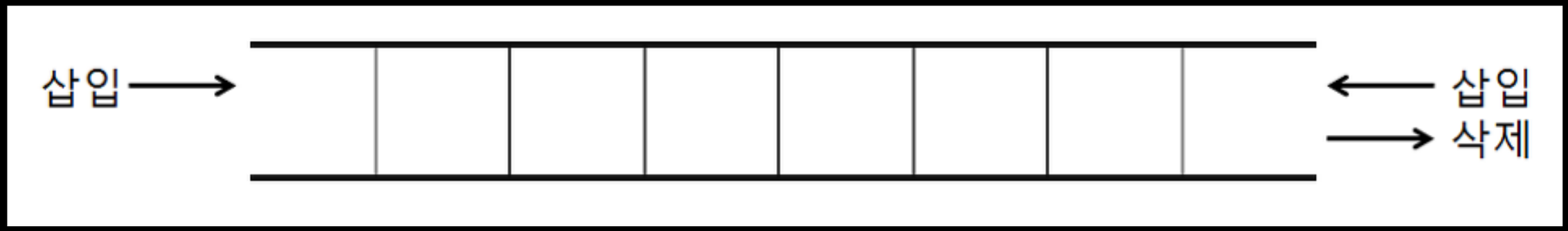
- 입력(추가)을 한곳에만 제한을 주는 데크
- 삭제는 양방향 가능



Concept

출력제한데크(Shelf)의 기본 동작 흐름은 아래와 같습니다.

- 삭제를 한곳에만 제한을 주는 데크
- 추가는 양방향 가능



Features

Dequeue의 특징을 살펴보면 아래와 같습니다.

- 데이터를 양쪽에서 삽입 한 곳에서는 삭제가 가능한 구조
- 데이터 입력 순서와 상관없이 출력 순서 조절 가능
- 스택과 큐의 장점을 모아서 만들어진 자료구조

Implementation

Swift를 활용하여 Dequeue를 구현해보겠습니다. 우선 필요한 메소드는 아래와 같습니다.

- init : 리스트를 초기화하는 함수
- enqueue : 데이터 입력
- enqueueFront : 앞쪽 방향에 데이터 입력
- dequeue : 첫번째 데이터 삭제
- dequeueBack : 뒤쪽 방향의 데이터 삭제
- peekFront : 첫번째 데이터 반환
- peekBack : 마지막 데이터 반환
- removeAll : 모든 데이터 삭제
- count : 현재 리스트의 크기를 반환
- isEmpty : 현재 리스트의 크기가 비어있는지 체크

Implementation

```
class Dequeue<T> {
    private var array = [T]()

    public init() {}

    public func enqueue(_ element: T) {
        array.append(element)
    }

    public func enqueueFront(_ element: T) {
        array.insert(element, at: 0)
    }

    public func dequeue() -> T? {
        if isEmpty {
            return nil
        } else {
            return array.removeFirst()
        }
    }

    public func dequeueBack() -> T? {
        if isEmpty {
            return nil
        } else {
            return array.removeLast()
        }
    }

    public func removeAll() {
        array.removeAll()
    }
}
```

Implementation

```
extension Dequeue {
    public func peekFront() -> T? {
        return array.first
    }

    public func peekBack() -> T? {
        return array.last
    }

    public var count: Int {
        return array.count
    }

    public var isEmpty: Bool {
        return array.isEmpty
    }
}
```

Implementation

데이터 반복(Iterator) 제어를 위하여 아래의 프로토콜들을 체택합니다.

```
public struct DequeueIterator<T> : IteratorProtocol {
    var currentElement: [T]

    public mutating func next() -> T? {
        if !self.currentElement.isEmpty {
            return currentElement.removeFirst()
        }else {
            return nil
        }
    }
}

extension Dequeue : Sequence {
    public func makeIterator() -> DequeueIterator<T> {
        return DequeueIterator(currentElement: self.array)
    }
}
```

Implementation

```swift
let dequeue = Dequeue<Int>()

dequeue.enqueue(1)
dequeue.enqueue(2)
dequeue.enqueue(3)
dequeue.enqueueFront(4)
dequeue.enqueueFront(5)

// 현재 데이터 카운트 : 5
print(dequeue.count)

for node in dequeue {
    print(node)
    // 5
    // 4
    // 1
    // 2
    // 3
}
```

Implementation

```
print(dequeue.peekFront()!) // 5

print(dequeue.dequeue()!) // 5
print(dequeue.dequeueBack()!) // 3

for node in dequeue {
    print(node)
    // 4
    // 1
    // 2
}

print(dequeue.peekBack()!) // 2
```

References

[1] 3. 데큐 구조체 : https://herohjk.com/16

[2] [선형구조]자료 구조의 개념 정리(리스트, 스택, 큐, 데크) : https://lee-mandu.tistory.com/462

[3] 자료구조] 큐와 데크란? : https://m.blog.naver.com/PostView.nhn?blogId=rbdi3222&logNo=220620826550&proxyReferer=https:%2F%2Fwww.google.com%2F

[4] 정보처리 자료구조 스택 큐 데크 : https://m.blog.naver.com/PostView.nhn?blogId=xpiart&logNo=220268937540&proxyReferer=https:%2F%2Fwww.google.com%2F

[5] [자료구조] 4. Deque : https://ieatt.tistory.com/7

Thank you!