자료구조 실습 보고서

[제 5 주]

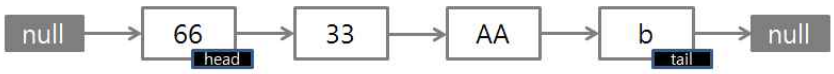
제출일 : 2018.4.4.수요일

학번 : 201604137

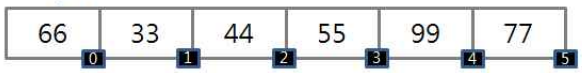
이름 : 김예지

[프로그램 설명서]

Linked List



Array List



LinkedList와 ArrayList를 비교하자면 LinkedList는 몇 개의 참조자만 바꿈으로써 새로운 자료의 삽입이나 기존 자료의 삭제를 위치에 관계없이 빠른 시간안에 수행 할 수 있다. ArrayList 같은 경우는 O(N)만큼의 연산 속도가 걸리기 때문에 자료의 최대 개수에 영향을 받지만, LinkedList는 그런 제약을 받지 않는다.LinkedList는 데이터를 저장하는 각 노드가 이전 노드와 다음 노드의 상태만 알고 있다고 보면 된다.

하지만, LinkedList의 단점도 존재한다. ArrayList에서는 무작위 접근(random access)이 가능하지만, LinkedList에서는 순차접근(sequential access) 만이 가능하다. 특히, 단순 LinkedList는 단방향성을 갖고 있기 때문에 인덱스를 이용하여 자료를 검색하는 애플리케이션에는 적합하지 않다.

[실행 결과 분석]

```java
@Override
public boolean add(String data) {
    if (!this.contains(data)) {
        Node newNode = new Node(data);
        newNode.setNext(this.head);
        this.head = newNode;
        this.size++;
        return true;
    } else {
        return false;
    }
}
```

1. add : 입력받은 data를 포함하고 있는지 확인한 다음 그 data를 가진 노드를 새로 생성한다. 그리고 head가 방금 만든 newNode를 가리키게 설정한다. 그리고 사이즈를 키움.

```java
@Override
public boolean remove(String data) {
    Node prevNode = null; // previosNode
    Node currNode = this.head; // currentNode
    boolean found = false;

    // 삭제하고자 하는 데이터가 있는지 확인하는 과정(데이터 존재하면 found==true
    while (currNode != null && !found) {
        if (currNode.getData().equals(data)) {
            found = true;
        } else {
            prevNode = currNode;
            currNode = currNode.getNext();
        }
    }

    // 삭제하고자 하는 데이터가 존재하는 경우 next변경
    if (!found) {
        return false;
    } else {
        if (currNode == this.head) {
            this.head = this.head.getNext();
        } else {
            prevNode.setNext(currNode.getNext());
        }
        this.size--;
        return true;
    }
}
```

2. remove : 삭제하게 되면 앞 노드(prevNode)와 현재 노드(currNode)같이 기준을 정해주고, 현재노드(현 위치)가 null이 아니고 데이터가 존재하지 않을 때까지 반복문을 돌려서 data와 같은 값이 있으면 그 값을 가져와서 currNode에 담는다. 찾는 값이 없다면 이전 노드에 현재노드를 담고 현재노드에 다음노드를 담는다.(다음노드를 검사하기 위해)
그리고 삭제하고자 하는 데이터가 존재하는 경우에는 두가지로 나뉘는데 (1)첫번째 노드에 찾는 값이 있을 때(=현재 위치가 헤드인데 그 다음값이 원하는 값일 때) 그 다음 값을 가져오고 (2)첫번째 노드가 아닐 때는 현재노드의 다음번노드(주소) 값을 이전노드에 담는다. 그리고 삭제했으니 사이즈를 줄여준다.

```
-
// 노드를 하나씩 움직이면서 원하는 데이터가 포함되어있는지 확인하는 과정
@Override
public boolean contains(String data) {
    Node searchNode = this.head;
    while (searchNode != null) {
        if (searchNode.getData().equals(data)) {
            return true;
        }
        searchNode = searchNode.getNext();
    }

    return false;
}
```

3.현재 노드를 헤드부터로 해준 후 순서대로 돌린다. 원하는 데이터가 있다면 true를 반환하고 그 다음 노드를 가리킨다.

```
// 노드를 움직이면서 노드에 있는 데이터를 반환해줌
public String checkNode() {
    String content = "";
    Node searchNode = this.head;
    while (searchNode != null) {
        content += searchNode.getData();
        content += " ";
        searchNode = searchNode.getNext();
    }
    return content;
}
```

4.프린트 해줄 때는 head부터 순서대로 null이 안나올 때 까지 그 순서의 값을 가져와서 content에 담고 content를 반환한다.

```
@Override
public boolean swap(int i, int j) {
    // 1.i,j가 size보다 큰지 확인 + 0번째 노드를 건드리려고 하는지 확인(head)
    if (i > size || j > size || i == 0 || j == 0) {
        return false;
    } else {
        // 값 추출부분
        String iData = "";
        String jData = "";
        Node searchNode = this.head;
        int k = 1;
        while (searchNode != null) {
            if (k == i) {
                iData = searchNode.getData();
            }
            if (k == j) {
                jData = searchNode.getData();
            }
            searchNode = searchNode.getNext(); // 다음노드
            k++;
            System.out.println(1);
        }
```

```
        // 추가부분
        searchNode = this.head;
        k = 1;
        while (searchNode != null) {
            if (k == i) {
                searchNode.setData(jData);
            }

            System.out.println(2);

        if (k == j) {
            searchNode.setData(iData);
        }

        searchNode = searchNode.getNext(); // 다음노드
        k++;
        System.out.println(3);
        }
    }

    return true;
}
```

5. swap하고 싶은 순서 값 i,j를 입력받고 노드들을 싹 돌면서 i,j번째 값을 찾아서 각각 iData, jData에 넣는다. 그리고 둘 다 찾을 때 까지 다음노드를 가져온다. 그리고 다시 처음 노드부터 돌면서 I번째에 jData를 담고, j번째에 iData를 담는다. 그러면 값이 뒤 바뀌어 있음.