

HW4. Doubly Linked List (DLL) Exercise

- DUE: 5/6 (수)

- 프로그램: Doubly Linked List 의 여러 함수들을 테스트
- 아래 함수들을 모두 테스트 해볼 것.

1) Menu

- | | | | |
|------------------|------------------|------------------|-------------------|
| 1. Insert-after | 2. Insertbefore | 3. InsertFirst | 4. InsertLast |
| 5. DeleteCurrent | 6. LocateCurrent | 7. UpdateCurrent | 8. Displaycurrent |
| 9. DisplayList | 10. Quit | | |

2) 2 개의 포인터 사용

- **Head:** - 리스트의 첫번째 노드
- **Current:** 현재위치 노드 가리킴.
메뉴의 모든 **action** 이후에는 해당 노드가 **current position** 이 됨.

3) 기본 알고리즘

```
1. Insert-after (data)      //   current node 뒤에 insert
- Create temp (data)                // 새로운 노드생성
- if (hed==NULL) head=temp          // empty 리스트인 경우
  elseif (current->next!=NULL)      // current 가 마지막 노드가 아니면
    insert in the middle            //
  else
    insert in the last               // 마지막노드 이후에 insert
  current = temp                    // insert 된 노드가 current node
```

```
2. Insertbefore (data)      //   current node 앞에 insert

- Create temp (data)                // 노드생성
- if (hed==NULL) head=temp          // empty list 인 경우
  elseif (current ==head)           // head node 앞에 insert
    insert in the front & head 이동
  else
    insert in the middle& last      // 이동 후 insert
  current = temp                    // temp 이동
```

3. InsertFirst (data) : 리스트의 맨 앞에 위치하고, head 이동

If (head ==NULL) head = temp // empty list 인 경우

 else

 head node 앞으로 temp node insert

 head = temp

current=temp // current node

4. InsertLast (data)

 - create new node //노드 생성

 - if (head==null) head=temp // empty list 인 경우

 else

 find last node & insert // p=head, p 이동

current = temp // current node

5. DeleteCurrent () // current node 삭제

 - if (head==NULL) print “List Empty”

 elseif (current ==head) // head node 삭제 & head 이동

 elseif (current->next ==0) //마지막노드 삭제

 else

 delete in the middle // middle 노드 삭제

6. LocateCurrent(nTH) // n 번째 위치(nth) 를 입력받고, 해당 데이터 출력

Ex) Enter a position to locate: 2

 ⇒ Address(2): 20

 - if (head==NULL) print “List Empty”

 else if (length >= nTH) // 현재리스트의 길이 ≥ 원하는 위치(nth)

 p=head & Find location // n번째 위치로 이동

 Print current-value; // 해당노드 데이터 출력

 current = p; // current는 새로운 위치

 else

 print "No such a line" // 입력받은 위치 > 전체 리스트길이

7. UpdateCurrent (newdata) : 현재 current 가 위치한 데이터의 값 교체

 current->value = newdata;

8. Displaycurrent() - 현재 **current** 가 위치한곳의 데이터 한개 출력
print current-value

9. DisplayList - 전체 리스트 출력

● 기타함수

Listlength: 전체 리스트 길이 반환

Is_empty: if (head=0) return 1, else return 0

```
Command: (1)insertAfter (2)insertBefore (3)insertFirst (4)insertLast (5)deleteCurrent
          (6)locateCurrent (7) updateCurrent (8) displayList (9) quit
==> 1
Enter a data to insert => 10
--- List ---
1 * 10
Command: (1)insertAfter (2)insertBefore (3)insertFirst (4)insertLast (5)deleteCurrent
          (6)locateCurrent (7) updateCurrent (8) displayList (9) quit
==> 2
Enter a data to insert => 20
--- List ---
1 * 20
2 : 10
Command: (1)insertAfter (2)insertBefore (3)insertFirst (4)insertLast (5)deleteCurrent
          (6)locateCurrent (7) updateCurrent (8) displayList (9) quit
==> 3
Enter a data to insert => 30
--- List ---
1 * 30
2 : 20
3 : 10
Command: (1)insertAfter (2)insertBefore (3)insertFirst (4)insertLast (5)deleteCurrent
          (6)locateCurrent (7) updateCurrent (8) displayList (9) quit
==> 4
Enter a data to insert => 40
--- List ---
1 : 30
2 : 20
3 : 10
4 * 40
Command: (1)insertAfter (2)insertBefore (3)insertFirst (4)insertLast (5)deleteCurrent
          (6)locateCurrent (7) updateCurrent (8) displayList (9) quit
==> 5
--- List ---
1 * 30
2 : 20
3 : 10
Command: (1)insertAfter (2)insertBefore (3)insertFirst (4)insertLast (5)deleteCurrent
          (6)locateCurrent (7) updateCurrent (8) displayList (9) quit
==> 6
Enter a position to locate => 5
No such a line
Command: (1)insertAfter (2)insertBefore (3)insertFirst (4)insertLast (5)deleteCurrent
          (6)locateCurrent (7) updateCurrent (8) displayList (9) quit
==> 6
Enter a position to locate => 2
2 * 20
Command: (1)insertAfter (2)insertBefore (3)insertFirst (4)insertLast (5)deleteCurrent
          (6)locateCurrent (7) updateCurrent (8) displayList (9) quit
==> 7
Enter a data to update => 22
--- List ---
1 : 30
2 * 22
3 : 10
Command: (1)insertAfter (2)insertBefore (3)insertFirst (4)insertLast (5)deleteCurrent
          (6)locateCurrent (7) updateCurrent (8) displayList (9) quit
==> 8
--- List ---
```