Data Structure

Week 4

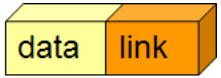
KyuDong SIM

1. 이번 주 실습 내용

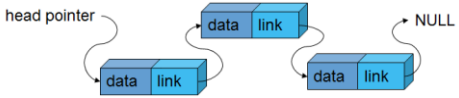
- Linked List

Linked List

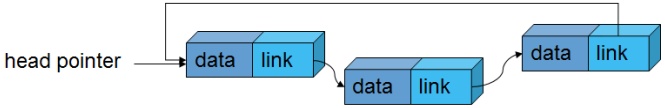
- 구성 요소
  - Node : Data & Link로 구성된 기본 단위
    - Data : 실제 데이터를 저장하는 영역
    - Link : 다른 Node의 위치를 저장하는 영역



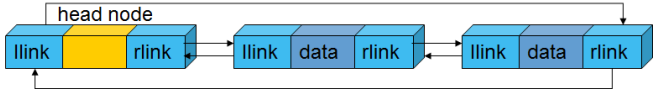
- 종류
  - Single Linked List
    - Node에 Data와 1개의 Link로 구성
    - Link는 다음 데이터의 위치 저장



  - Circular Linked List
    - Linked List가 원형 구조를 이룸
    - 마지막 Node의 Link가 처음 위치를 저장

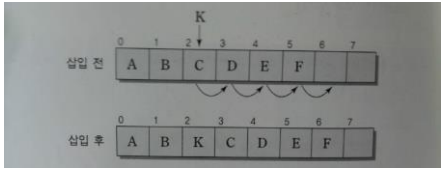


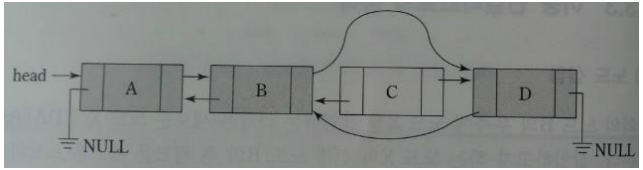
  - Doubly Linked List
    - Node에 Data와 2개의 Link로 구성
    - 2개의 Link는 각각 다음/이전 데이터 위치 저장



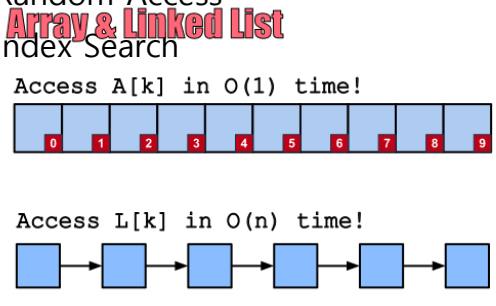
Linked List vs Array

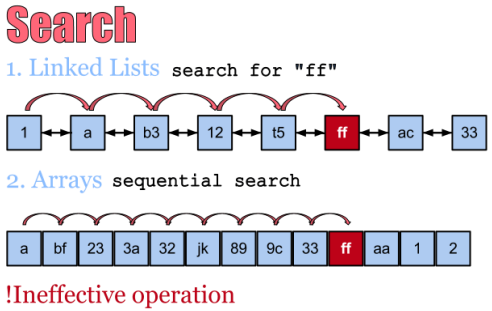
- Linked List
  - 장점
    - 동적인 크기 조작
    - 중간에 데이터를 끼어 넣는 작업
      - Link만 끊어주면 해결됨





  - 단점
    - Random Access
    - Index Search





Lab 2: Linked List

Insert a new node right after the node with the given key. If your list does not have any node with the given key, just pass.

Delete the node with the given key. If your list does not have any node with the given key, just pass.

Find the previous node of the node with the given key. If your list does not have any node with the given key, just pass.

Show the entire list

Lab 2: Linked List

INPUT

No duplicate keys in the list and the input

i x y : insert a new node with the key “x” after the node with the key “y”

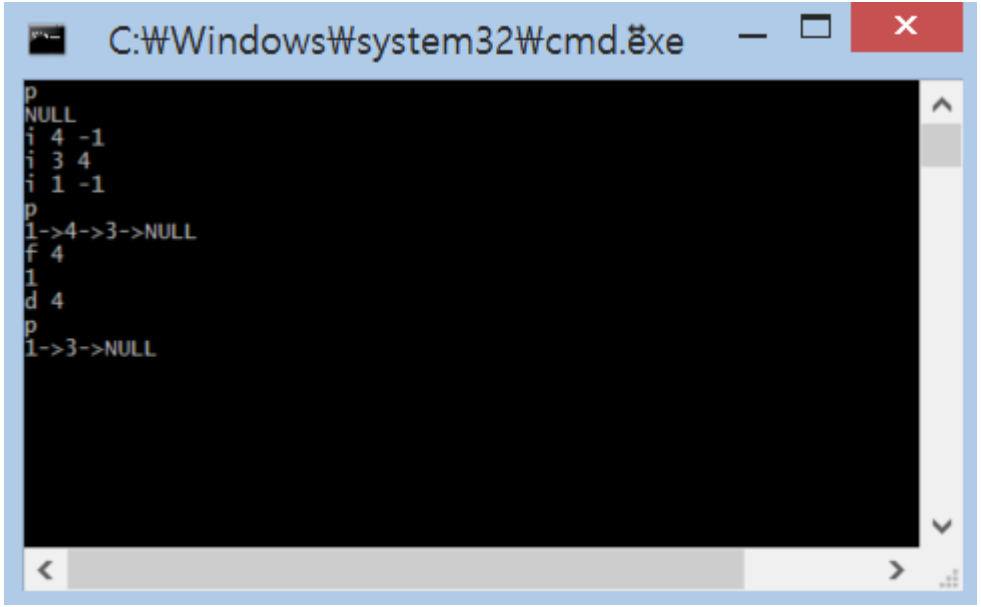
i x -1 : insert a new node with the key “x” before the first node in the list

d x : delete the node with the key “x”

f x : print the key of the previous node of the node with the key “x”

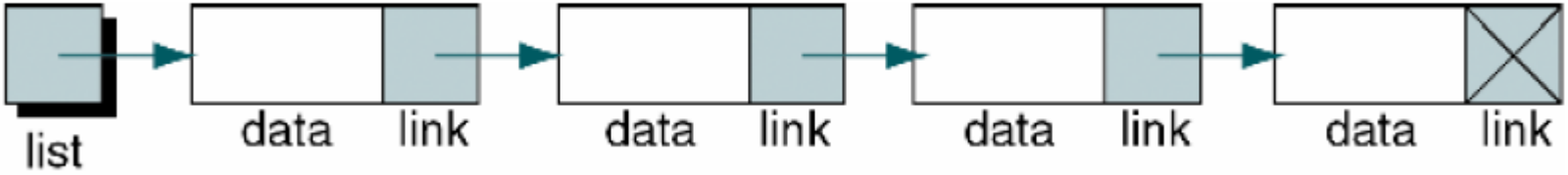
p : print the entire list from the beginning to the end

Lab 2: Linked List



Node, Pointer 선언

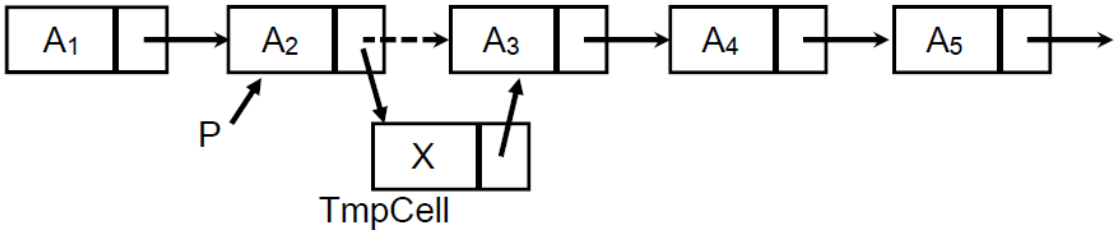
```
struct Node
{
    int data;
    struct Node* Next;
};

typedef struct Node *PtrToNode;
typedef PtrToNode List;
typedef PtrToNode Position;
```



Insert

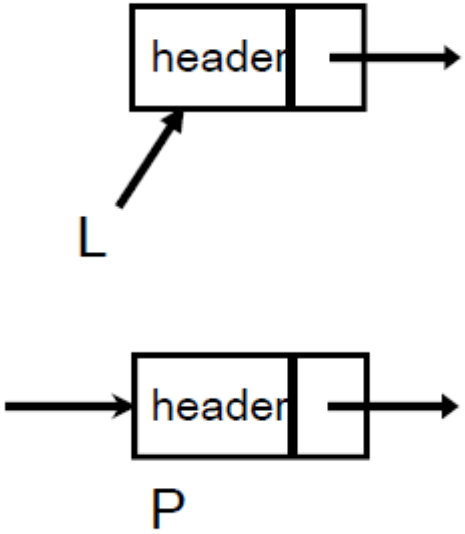
```
void Insert(int X, List L, Position P)
{
    Position TmpCell;

    TmpCell = (Position) malloc (sizeof(struct Node));
    if(TmpCell == NULL)
        printf("Out of space!");

    TmpCell->data = X;
    TmpCell->Next = NULL;

    TmpCell->Next = P->Next;
    P->Next = TmpCell;
}
```

P = L 이면 맨 앞에 Insert 가능



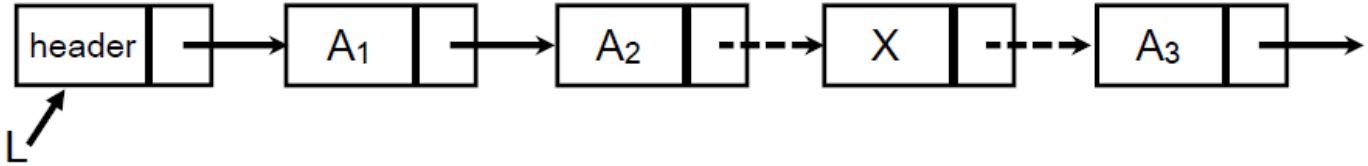
IsEmpty, IsLast

```
int IsEmpty(List L)
{
    return L->Next == NULL;
}

int IsLast(Position P, List L)
{
    return P->Next == NULL;
}
```



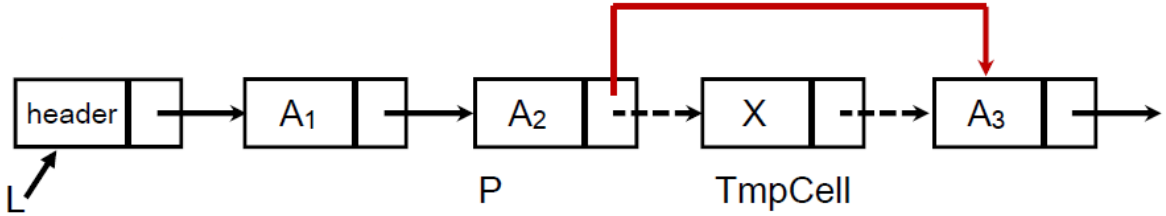
Find

```
/* return position ox X in L; NULL if not found */
Position Find(int X, List L)
{
    Position P;

    P = L->Next;
    while (P != NULL && P->data !=X )
        P = P->Next;

    return P;
}
```



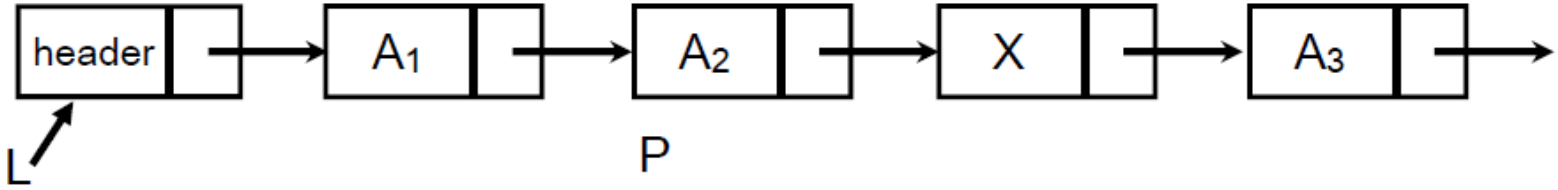
Delete

```
void Delete(int X, List L)
{
    Position P, TmpCell;

    P = FindPrevious(X, L);
    if( !IsLast(P, L))
    {
        TmpCell = P->Next;
        P->Next = TmpCell->Next;
        free(TmpCell);
    }
}
```



FindPrevious

```
Position FindPrevious(int X, List L)
{
    Position P;

    P = L;
    while(P->Next != NULL && P->Next->data !=X)
        P = P->Next;
    return P;
}
```



Insert Data, Print List

```
void DataInsert(List L, int target, int data)
{



}
```

Find 와 Insert 로 구현

```
void PrintList(List L)
{



}
```

Linkded List 출력

List 초기화

```
List L;
L = (List)malloc(sizeof(List));
L->Next = NULL;
```

반복 명령 수행

```
while (state)
{
    scanf("%c", &command);
    switch (command)
    {
    case 'i':
        /*Insert Data*/
        break;
    case 'd':
        /*Delete Data*/
        break;
    case 'f':
        /*Find Data*/
        break;
    case 'p':
        /*Print Data*/
        break;
    case 'q':
        /*Quit*/
        break;
    }
}
```

제출 및 알림

수업 중 확인 or 메일제출 (이름, 학번, 소스코드)

메일 제출 :

주소 : (89kdsim@naver.com)

기한 : ~2016-03-30