광운대학교 KwangWoon University

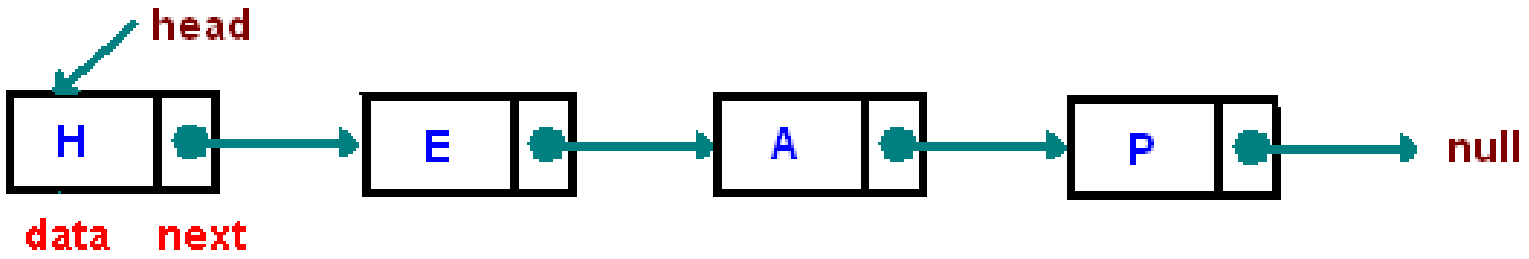
Linked List

Kwangwoon Univ.
Dept. of Computer engineering
Ki-Hoon Lee

조교 : 박효주 (gkwk011@naver.com)

Linked List란

- 노드들이 서로 연결된 형태의 자료구조
- 배열과 달리 저장할 수 있는 데이터의 양이 제한되어 있지 않아 삽입 / 삭제가 용이
- 가장 첫 노드를 가리키는 헤드(Head)가 반드시 필요



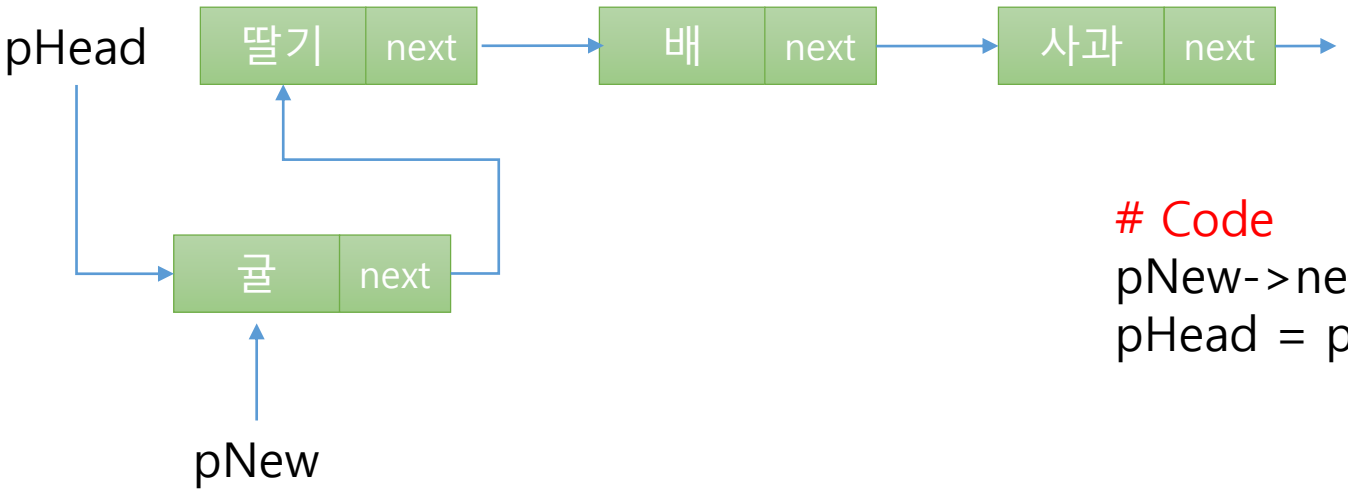
Insert

❖ 앞에 추가할 때

pHead: 가장 첫 노드를 가리키는 포인터
pNew: 새로 만드는 노드를 가리키는 포인터

Insert

❖ 앞에 추가할 때



Code

```
pNew->next = pHead;
pHead = pNew;
```

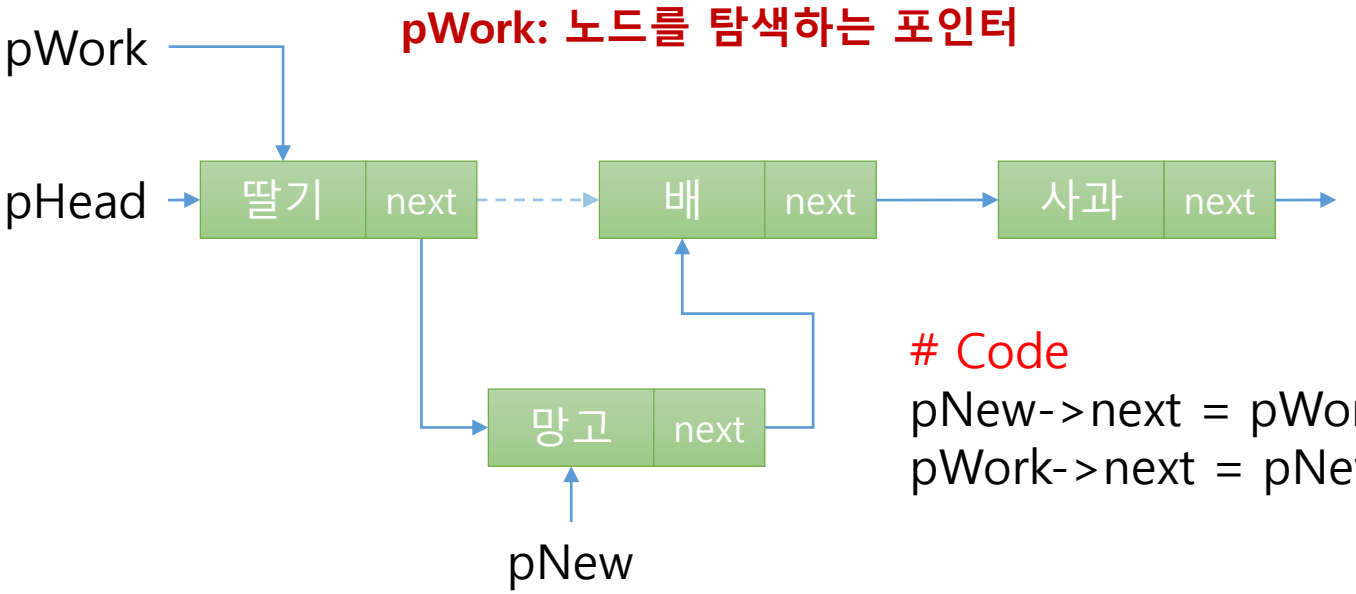
새로 들어오는 노드가 pHead가 가리키는 노드를 가리키고
pHead가 새로 들어오는 노드를 가리켜야 함

Insert

❖ 중간에 추가할 때

Insert

❖ 중간에 추가할 때



Code

```
pNew->next = pWork->next;
pWork->next = pNew;
```

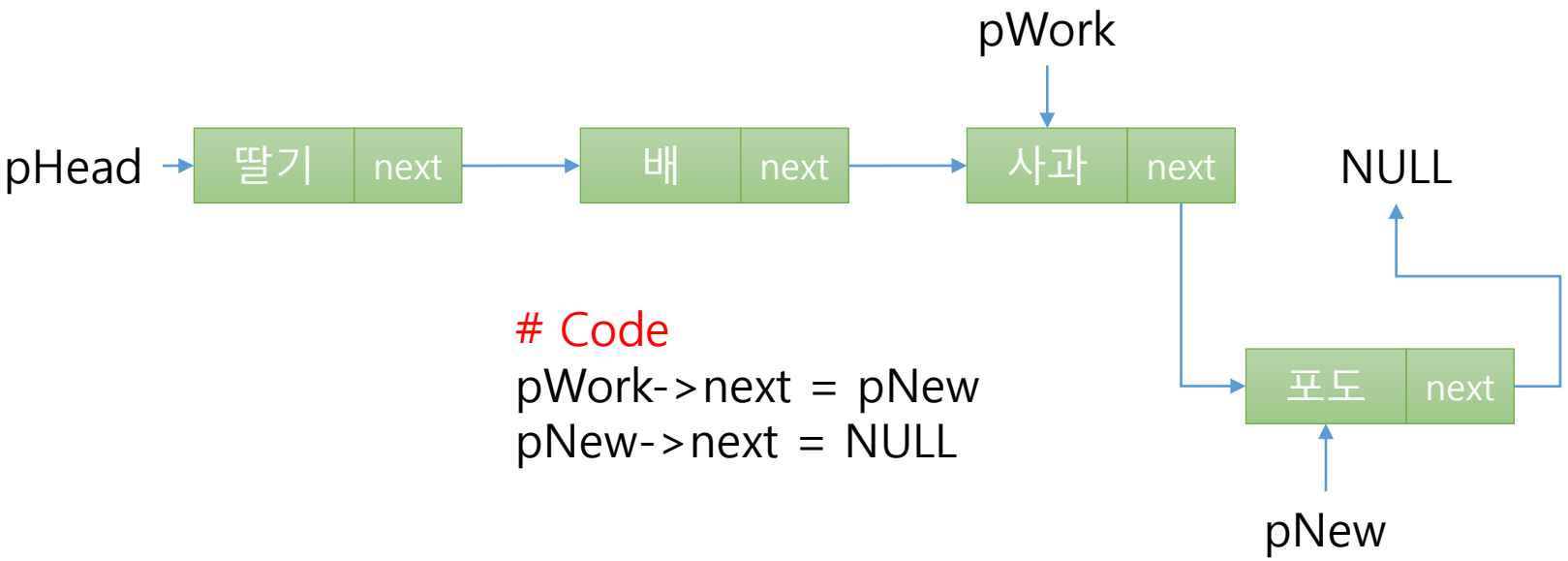
새로 들어오는 노드가 다음 노드를 가리키고
이전 노드가 새로 들어오는 노드를 가리켜야 함

Insert

❖ 마지막에 추가할 때

Insert

❖ 마지막에 추가할 때



Code
pWork->next = pNew
pNew->next = NULL

마지막 노드의 next가 새로 들어오는 노드를 가리키고
새로 들어오는 노드의 next가 NULL을 가리켜야 함

예제(정렬 삽입)

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

typedef struct node
{
    char name[32]; // 과일 이름
    struct node *next; // 다음 노드를 가리키는 포인터
}Node;

Node *pHead = NULL; // 헤드

void Insert(){
    char str[32];
    Node *pWork = NULL; // 탐색 노드
    Node *pPrev = NULL; // 탐색 노드의 이전 노드
    Node *pNew = NULL; // 새로 생성할 노드

    printf("과일이름: ");
    scanf("%s", str);

    // 노드 생성 및 초기화
    pNew = (Node *)malloc(sizeof(Node));
    strcpy(pNew->name, str);
```

```c
    if (pHead == NULL){
        // 노드가 하나도 존재하지 않을 경우
        pHead = pNew;
        pNew->next = NULL;
    }
    else{
        pWork = pHead;
        //전체 노드 탐색
        while (pWork != NULL){
            // pNew가 들어갈 위치의 다음 노드를 pWork가 가리키도록 함
            if (strcmp(pWork->name, pNew->name) > 0) break;
            pPrev = pWork;
            pWork = pWork->next;
        }
        if (pWork == pHead){
            // 가장 앞
            pNew->next = pHead;
            pHead = pNew;
        }
        else {
            // 중간 및 가장 마지막
            pPrev->next = pNew;
            pNew->next = pWork;
        }
    }
}
```

pPrev: pWork의 이전 노드를 기리키는 포인터(pWork가 pHead일 경우 NULL을 기리킴)

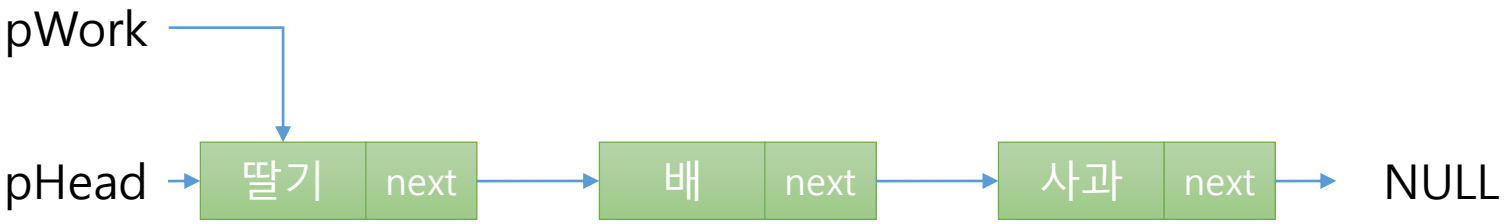
예제(정렬 삽입)

```
void Print(){
    Node *pWork = pHead;
    // 전체 노드 탐색
    while (pWork != NULL){
        // 존재하는 노드를 전부 찾아서 출력
        printf("%s -> ", pWork->name);
        pWork = pWork->next;
    } printf("NULL\n");
}
```

```
void main()
{
    int num;

    while (1){
        puts("======Menu=====");
        puts("1. Insert");
        puts("2. Print");
        puts("3. Delete");
        puts("4. Search");
        puts("5. Update");
        puts("===============");
        printf("입력 >> ");
        scanf("%d", &num);

        if (num == 1) Insert();
        else if (num == 2) Print();
        //else if (num == 3) Delete();
        //else if (num == 4) Search();
        //else if (num == 5) Update();
    }
    //free_linkedlist();
}
```

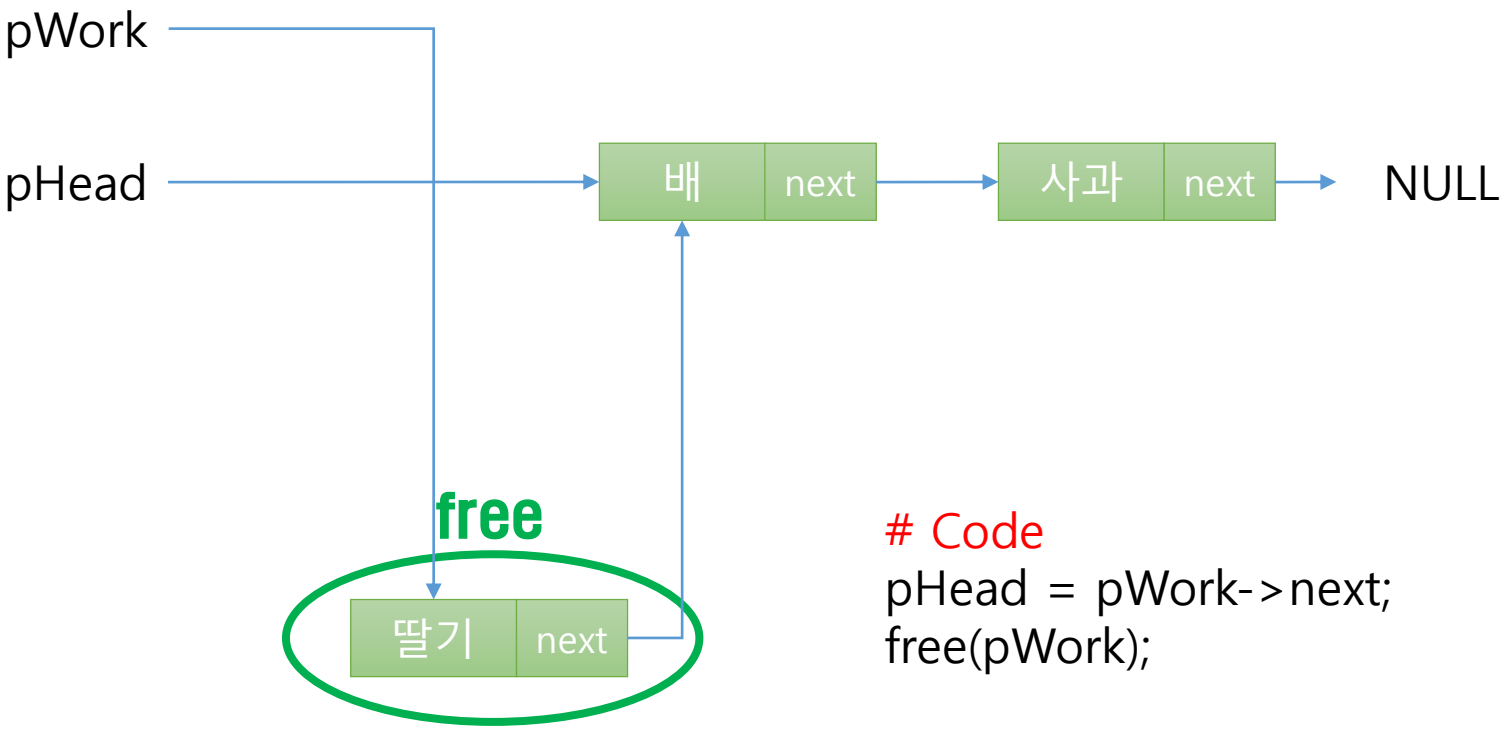
Delete

❖ 앞에 삭제할 때



Delete

❖ 앞에 삭제할 때

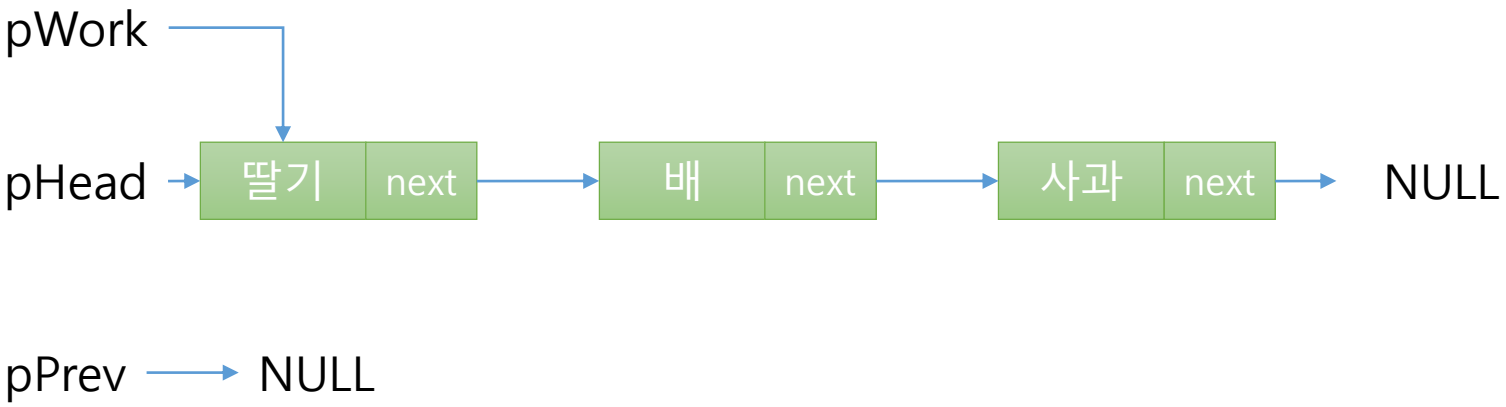


# Code
pHead = pWork->next;
free(pWork);

Head가 삭제될 노드의 다음 노드를 가리켜야 함

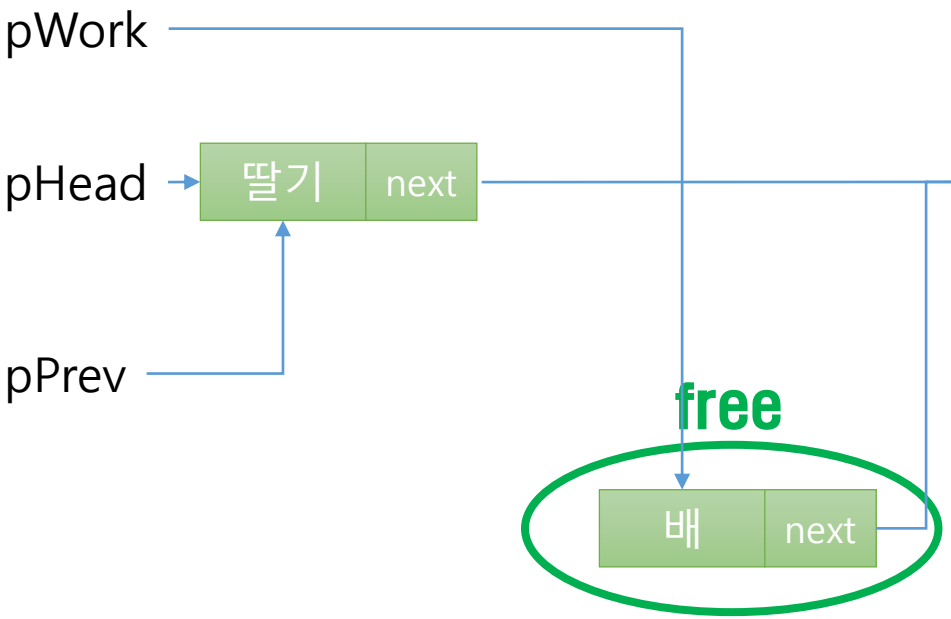
Delete

❖ 중간에 삭제할 때



Delete

❖ 중간에 삭제할 때



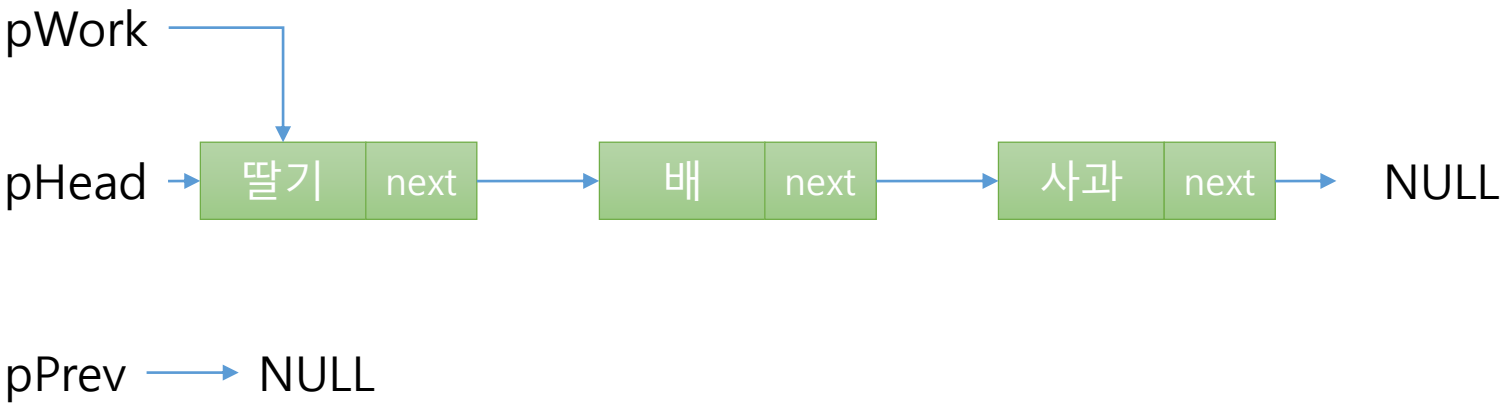
Code

```
pPrev->next = pWork->next;
free(pWork);
```

삭제될 노드의 이전 노드가 삭제될 노드의 다음 노드를 가리켜야 함

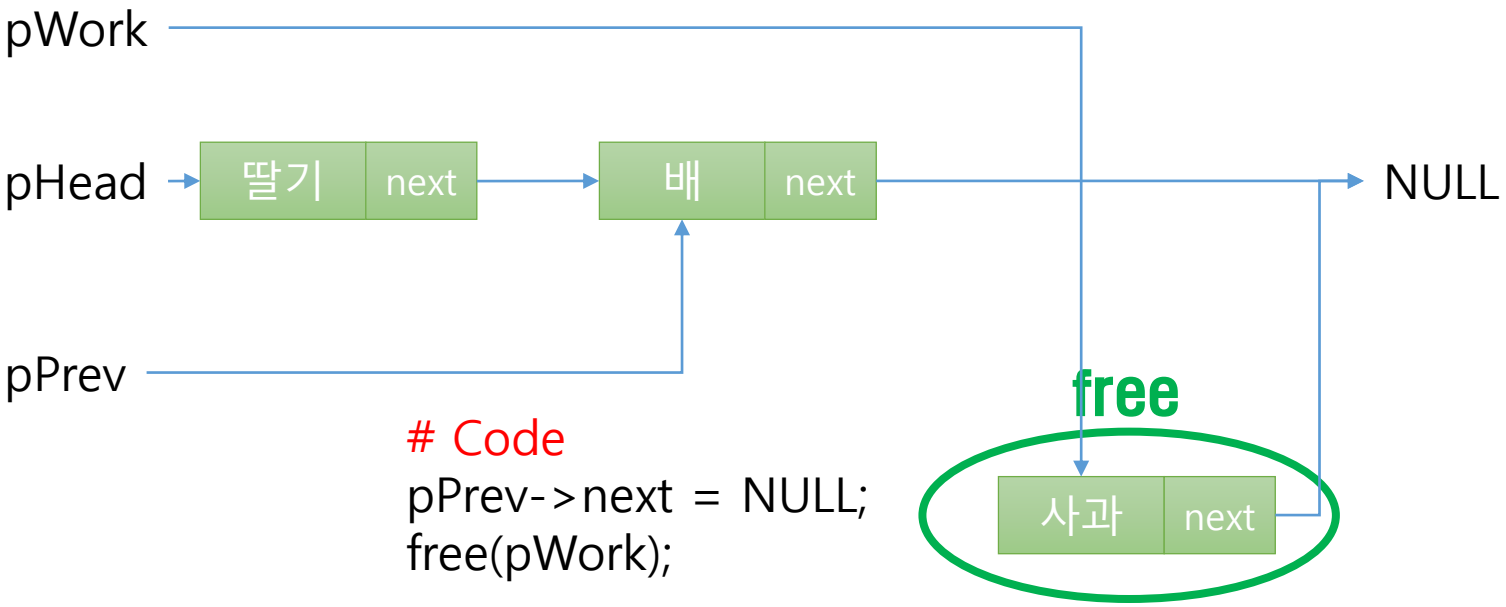
Delete

❖ 마지막에 삭제할 때



Delete

❖ 마지막에 삭제할 때



Code

```
pPrev->next = NULL;
free(pWork);
```

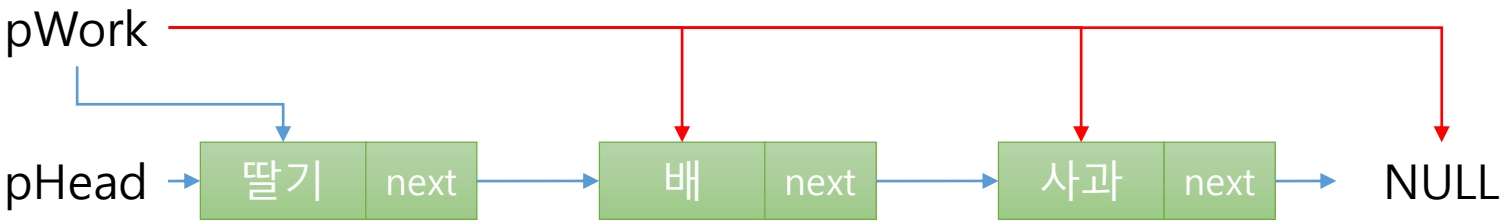
삭제될 노드의 이전 노드가 반드시 NULL을 가리켜야 함

예제(삭제)

❖ Delete 함수 추가

```
int Delete(){
    Node *pWork = pHead;
    Node *pPrev = NULL;
    char str[32];

    printf("과일이름: ");
    scanf("%s", str);

    // 전체 노드 탐색
    while (pWork != NULL){
        // 삭제할 노드를 찾은 경우
        if (!strcmp(pWork->name, str)){
            // 가장 첫 노드를 삭제해야 할 경우 pHead가 두번째 노드를 가리킴
            if (pWork == pHead) pHead = pWork->next;
            // 삭제할 노드의 이전 노드가 삭제할 노드의 다음 노드를 가리키도록 함
            else                pPrev->next = pWork->next;

            // 노드 삭제
            free(pWork);
            return 1; // 삭제 성공: 1 반환
        }
        pPrev = pWork;
        pWork = pWork->next;
    }
    puts("없는 과일입니다.\n");
    return 0; // 삭제 실패: 0 반환
}
```

Search



Code

```
pWork = pHead;
while(pWork != NULL){
        pWork = pWork->next;
}
```

pWork가 NULL일 때 까지 pWork의 next를 계속 찾아감

예제(검색)

❖ Search 함수 추가

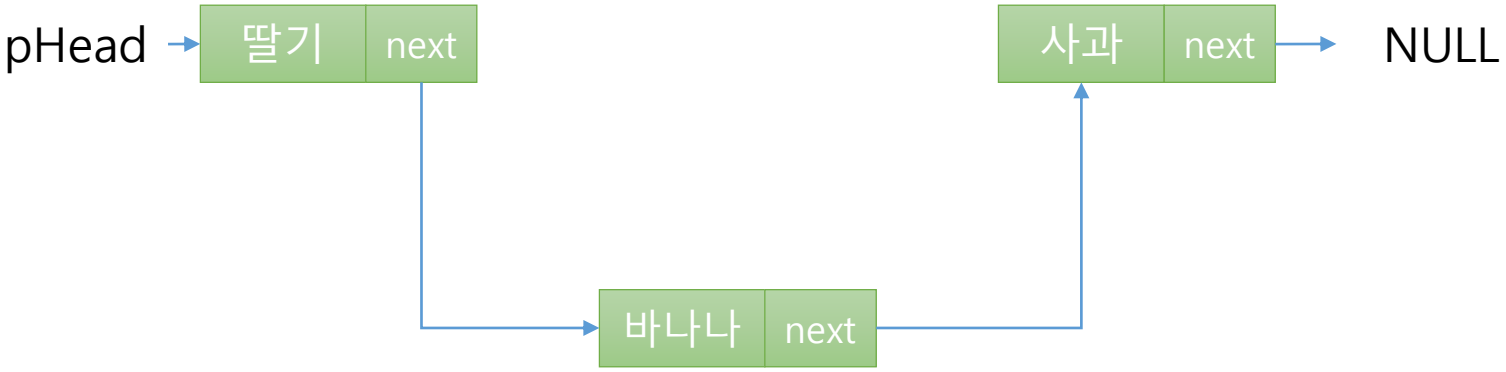
```
void Search(){
    Node *pWork = pHead;
    char str[32];
    int cnt = 0;

    printf("찾는 과일이름: ");
    scanf("%s", str);

    // 전체 노드 탐색
    while (pWork != NULL){
        // 키워드가 포함된 경우
        if (strstr(pWork->name, str)){
            if (cnt++ == 0) puts("과일\n========");
            printf("%s\n", pWork->name);
        }
        pWork = pWork->next;
    }
    if(cnt == 0) puts("목록이 없습니다.\n");
}
```

Update

Update

Delete(“배”)

Update



Insert("바나나")

예제(수정)

❖ Update 함수 추가

```
void Update(){
     printf("변경하려는 ");
     //삭제에 성공했을 경우
     if (Delete()){
          printf("새로운 ");
          Insert();
     }
}
```

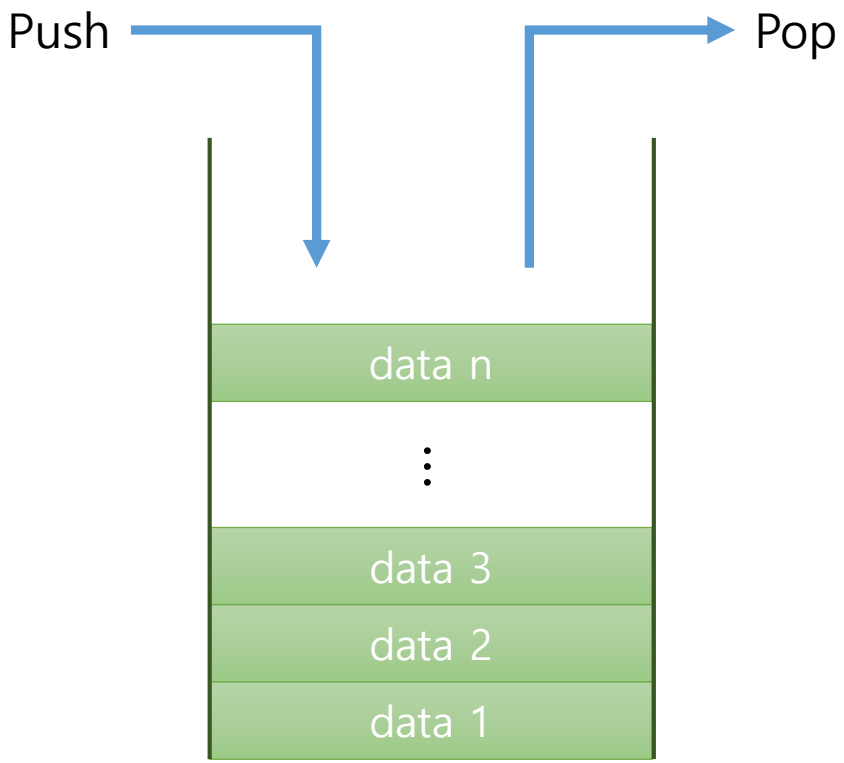
예제(메모리 해제)

❖ free_linkedlist 함수 추가

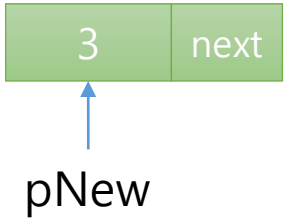
```
void free_linkedlist(){
    Node *temp;

    // 전체 노드 탐색
    while (pHead != NULL){
        temp = pHead;
        pHead = pHead->next;
        free(temp);
    }
}
```

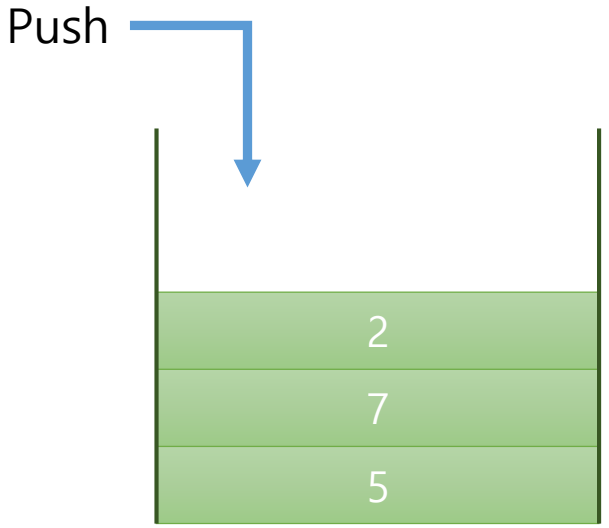
Stack이란

- LIFO (Last In First Out)으로 저장되는 자료구조
- 가장 마지막에 넣은 데이터부터 빠지게 된다.

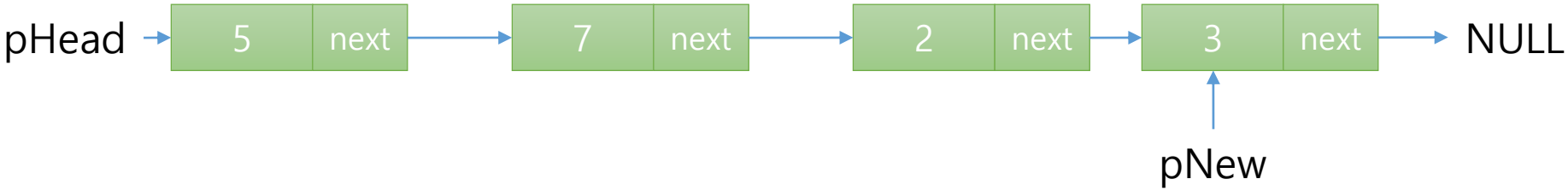


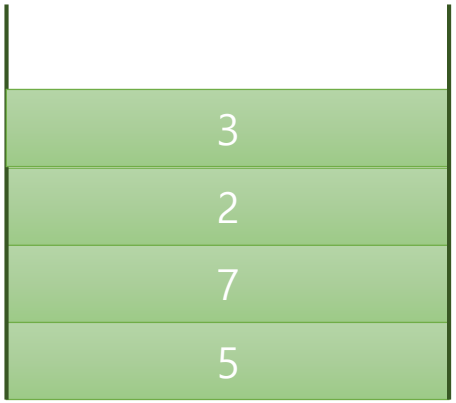
Push



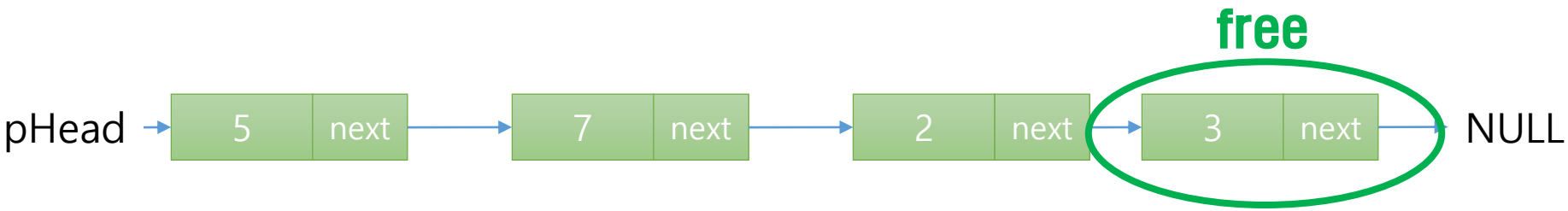


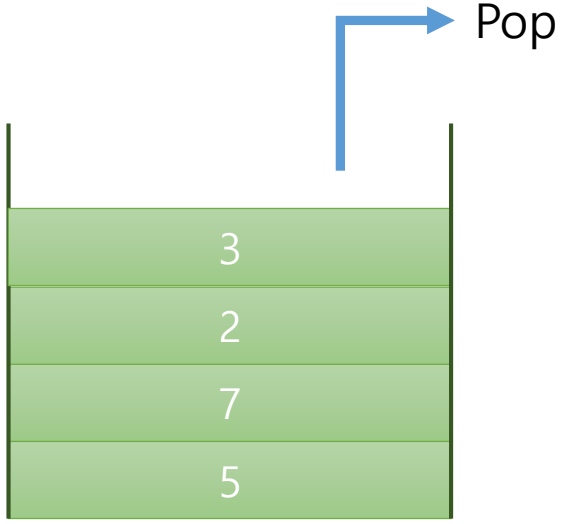
Push





Pop





Pop