Chapter 04.
연결리스트2

4주차

메모리 할당의 필요성

◆ 배열의 특성
- 메모리를 정적으로 확보(실행 전에 컴파일 시에 결정)해야 하므로
- 충분히 잡으려고 너무 많이 잡으면 메모리 낭비
- 부족하게 잡으면 추가되는 데이터를 처리 못함.

◆ 메모리를 동적으로(실행시간에) 잡을 수 있는 방법 필요

자기참조 구조체
(자신에 대한 포인터 타입 항목을 포함한 구조체)

```
struct _node
{
    int data;

    struct _node * next;
};

typedef struct _node Node;
```

데이터부:
노드가 가져야할 내용.
다양한 타입의 여러 항목 수용 가능.

링크부:
이전/다음 노드와의 연결에 사용

의미 동일!

```
typedef struct _node
{
    int data;

    struct _node * next;
} Node;
```

메모리 할당

- malloc : 요구한 byte수 만큼 메모리 할당 받음.
- free : 할당 받았던 메모리를 되돌려 줌.
- malloc과 free는 반드시 짝을 맞춰 사용.
- malloc에서 받은 포인터를 free함수의 인자로 주면 됨.

[함수 protptype]

```
void *malloc( size_t );
void free( void * );
```

[사용 예]

```
int *pInt = NULL;
pInt = (int *)malloc(sizeof(int) * 10);
free(pInt);
```

자기참조 구조체 이용한 프로그램 (LinkedRead.c)의 결과

```
자연수 입력: 3
자연수 입력: 4
자연수 입력: 2
자연수 입력: 0
3 4 2
3을 삭제
4을 삭제
2을 삭제
```

LinkedRead.c
(include, 구조체 정의)

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#include <stdlib.h>

#define TRUE	1
#define FALSE	0

typedef struct _node
{
	int data;	// 데이터를 담을 공간
	struct _node * next;	// 연결의 도구!
} Node;
```

LinkedRead.c
(main함수 1/4 : 리스트 초기화)

```
int main(void)
{
        Node * head = NULL;
        Node * tail = NULL;
        Node * cur = NULL;

        Node * newNode = NULL;
        int readData;
```

첫 노드, 마지막 노드 포인터.
빈 list일 때는 둘 다 NULL.

List의 노드를 참조할 때 사용.

LinkedRead.c
(main함수 2/4 : 노드 삽입)

```
while(1)
{
    printf("자연수 입력: ");
    scanf("%d", &readData);
    if(readData < 1)
        break;

    newNode = (Node*)malloc(sizeof(Node));
    newNode->data = readData;
    newNode->next = NULL;

    if(head == NULL)
        head = newNode;
    else
        tail->next = newNode;

    tail = newNode;
}
```

새 노드 준비

List의 첫 노드 삽입

List의 두 번째 이후 노드 삽입

LinkedRead.c
(main함수 3/4 : 데이터 조회)

```
if (head == NULL)
{
    printf("저장된 자연수가 존재하지 않습니다. ₩n");
}
else
{
    cur = head;
    printf("%d ", cur->data);
    while (cur->next != NULL)
    {
        cur = cur->next;
        printf("%d ", cur->data);
    }
}
printf("₩n");
```

노드가 없을 때의 처리

처음부터 끝까지 노드 운행.

LinkedRead.c
(main함수 4/4 : 데이터 삭제)

```
if (head != NULL) {
    Node * delNode = head;
    Node * delNextNode = head->next;
    printf("%d을 삭제₩n", head->data);
    free(delNode);
    while (delNextNode != NULL) {
        delNode = delNextNode;
        delNextNode = delNextNode->next;
        printf("%d을 삭제₩n", delNode->data);
        free(delNode);
    }
}
```

delNode 를 삭제하기 전에 그 다음 노드를 보관해 두기.

delNode 메모리 해제.