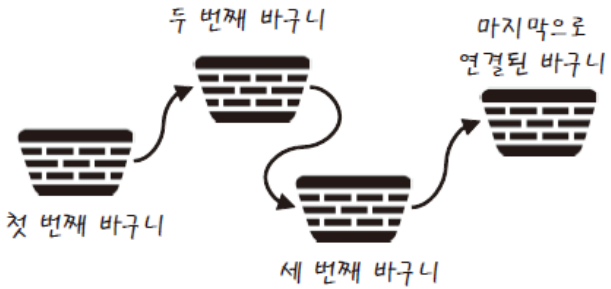
연결 리스트(Linked List) 2

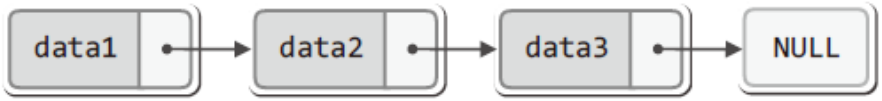
Linked! 무엇을 연결하겠다는 뜻인가!

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

일종의 바구니, 연결이 가능한 바구니



▶ [그림 04-1 : 노드의 표현]



▶ [그림 04-2: 노드의 연결]

예제 LinkedRead.c의 분석을 시도 바랍니다!

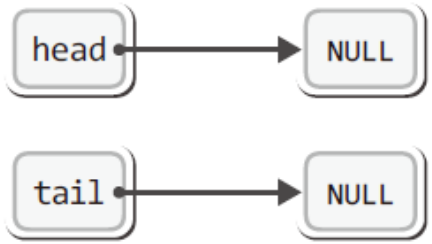
```c
#include <stdio.h>
#include <stdlib.h>

typedef struct _node
{
	int data;
	struct _node * next;
} Node;

int main(void)
{
	Node * head = NULL;    // NULL 포인터 초기화
	Node * tail = NULL;
	Node * cur = NULL;

	Node * newNode = NULL;
	int readData;

	/**** 데이터를 입력 받는 과정 ****/
	while(1)
	{
		printf("자연수 입력: ");
		scanf("%d", &readData);
		if(readData < 1)
			break;

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

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

		tail = newNode;
	}
	printf("₩n");

	/**** 입력 받은 데이터의 출력과정 ****/
	printf("입력 받은 데이터의 전체출력! ₩n");
	if(head == NULL)
	{
		printf("저장된 자연수가 존재하지 않습니다. ₩n");
	}
	else
	{
		cur = head;
		printf("%d  ", cur->data);   // 첫 번째 데이터 출력

		while(cur->next != NULL)    // 두 번째 이후의 데이터 출력
		{
			cur = cur->next;
			printf("%d  ", cur->data);
		}
	}
	printf("₩n₩n");

	/**** 메모리의 해제과정 ****/
	if(head == NULL)
	{
		return 0;    // 해제할 노드가 존재하지 않는다.
	}
	else
	{
		Node * delNode = head;
		Node * delNextNode = head->next;
		printf("%d을(를) 삭제합니다. ₩n", head->data);
		free(delNode);    // 첫 번째 노드의 삭제

		while(delNextNode != NULL)    // 두 번째 이후의 노드 삭제 위한 반복문
		{
			delNode = delNextNode;
			delNextNode = delNextNode->next;

			printf("%d을(를) 삭제합니다. ₩n", delNode->data);
			free(delNode);    // 두 번째 이후의 노드 삭제
		}
	}
	return 0;
}
```

예제 LinkedRead.c의 분석: 초기화

```
typedef struct _node
{
    int data;
    struct _node * next;
} Node;

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

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

LinkedRead.c의 일부

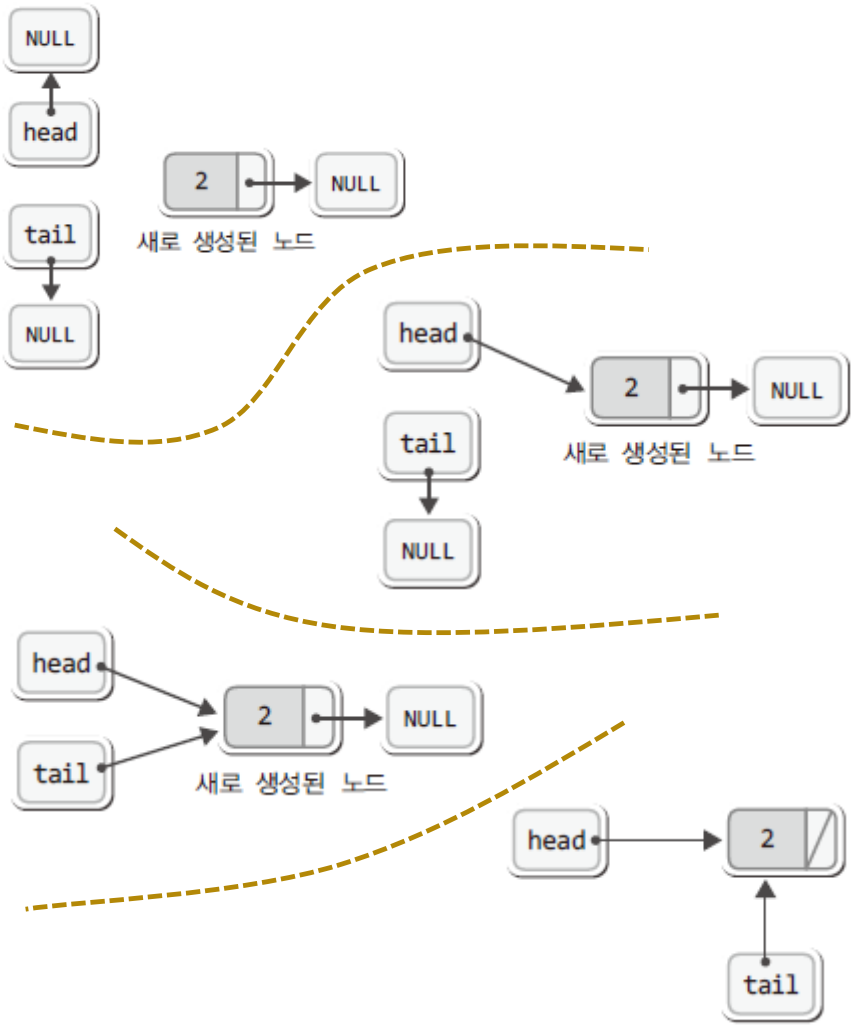
- head, tail, cur이 연결 리스트의 핵심!
- head와 tail은 연결을 추가 및 유지하기 위한것
- cur은 참조 및 조회를 위한것



예제 LinkedRead.c의 분석: 삽입 1회전

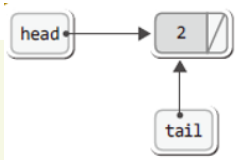
```
while(1)
{
    printf("자연수 입력: ");
    scanf("%d", &readData);
    if(readData < 1)
        break;

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

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

    tail = newNode;
}
```

LinkedRead.c의 일부



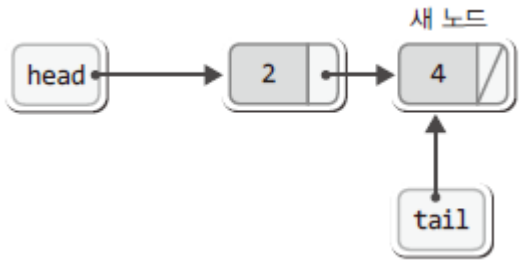
예제 LinkedRead.c의 분석: 삽입 2회전

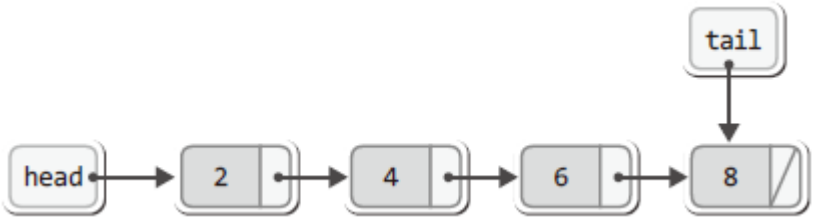
```
while(1)
{
    printf("자연수 입력: ");
    scanf("%d", &readData);
    if(readData < 1)
        break;

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

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

    tail = newNode;
}
```



tail->next는 앞선 node의 다음 node 연결 pointer

LinkedRead.c의 일부



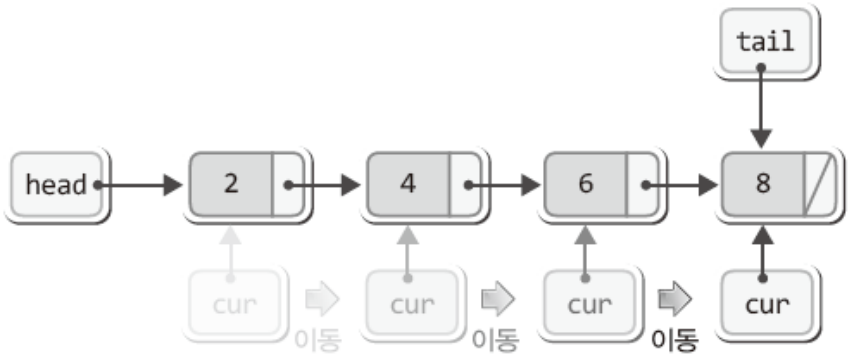


다수의 노드를 저장한 결과

예제 LinkedRead.c의 분석: 데이터 조회

전체 데이터의 출력 과정

```
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);
    }
}
```



LinkedRead.c의 일부

예제 LinkedRead.c의 분석: 데이터 삭제

```
if(head == NULL)      전체 노드의 삭제 과정
{
    return 0;
}
else
{
    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);
    }
}
```

LinkedRead.c의 일부