# 연결 리스트

배열처럼 여러 개의 데이터를 담을 수 있으나 중간의 데이터를 삭제하거나 중간에 데이터를 삽입하기 쉬운 자료 구조이다. 또한 길이를 늘리거나 줄일 수도 있다. 연결 리스트의 각 요소를 노드라 부르며 노드는 데이터와 다음 노드의 포인터로 구성되어 있는 것이 특징이다.

## 예제

### 단순 연결 리스트

single_list.cpp

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

// 노드 구조체
struct Node
{
    int value;
    Node *next;
};
Node *head;

// 연결 리스트 초기화 - 머리를 할당한다.
void InitList()
{
    head=(Node *)malloc(sizeof(Node));
    head->next=NULL;
}

// Target 다음에 노드를 삽입한다.
Node *InsertNode(Node *Target,Node *aNode)
{
    Node *New;

    New=(Node *)malloc(sizeof(Node));
    *New=*aNode;

    New->next=Target->next;
    Target->next=New;
    return New;
}

// Target 다음 노드를 삭제한다.
bool DeleteNode(Node *Target)
{
    Node *Del;
```

```
    Del=Target->next;
    if (Del==NULL) {
        return false;
    }
    Target->next=Del->next;
    free(Del);
    return true;
}

// 연결 리스트의 모든 노드와 머리를 해제한다.
void UnInitList()
{
    while (DeleteNode(head)) {;}

    free(head);
    head=NULL;
}

void main()
{
    int i;
    Node *Now,Temp;

    InitList();

    // 다섯 개의 노드 삽입
    Now=head;
    for (i=1;i<=5;i++) {
        Temp.value=i;
        Now=InsertNode(Now,&Temp);
    }

    // 두 번째 노드 삭제
    DeleteNode(head->next);

    // 순회하면서 출력
    for (Now=head->next;Now;Now=Now->next) {
        printf("%d\t",Now->value);
    }
    printf("\n");

    UnInitList();
}
```

## 이중 연결 리스트

## 원형 연결 리스트

# 참고

- 위키피디아
- 김상형 씨 홈페이지