**리스트란?**

: 리스트는 다른 나중에 배울 Stack, Queue, Tree, Graph 등과 같은 다른 자료구조 구현에 활용될 기초 자료구조입니다.

**리스트의 특징**

* 자료를 순서대로 한 줄로 저장하는 자료구조.
* 여러 자료가 일직선으로 서로 연결된 선형 구조입니다.
* 제일 처음 데이터를 가리켜서 Head라고 합니다.
* 제일 마지막 데이터를 가리켜서 Tail이라고 합니다.

**추상 자료형(ADT)**  
구체적인 기능의 완성과정을 언급하지 않고, 순수하게 기능이 무엇인지를 나열한 것

**지갑의 추상 자료형**

∙ 카드의 삽입

∙ 카드의 추출(카드를 빼냄)

∙ 동전의 삽입

∙ 동전의 추출(동전을 빼냄)

∙ 지폐의 삽입

∙ 지폐의 추출(지폐를 빼냄)

**자료형 Wallet의 정의**

typedef struct \_wallet {

int coin100Num; // 100원짜리 동전의 수

int bill5000Num; // 5,000원짜리 지폐의 수

} Wallet;

\*완전한 자료형의 정의로 인식되기 위해서는 해당 자료형과 관련이 있는 연산이 함께 정의되어야 한다

int TakeOutMoney(Wallet \* pw, int coinNum, int billNum); // 돈 꺼내는 연산

void PutMoney(Wallet \* pw, int coinNum, int billNum); // 돈 넣는 연산

\*자료구조의 내부 구현을 모르고도 해당 자료구조의 활용이 가능하도록 ADT를 정의하는 것이 옳다. main 함수를 먼저 접하게 되면, 구현할 자료구조를 구성하는 함수들을 잘 이해할 수 있다.

**리스트의 구분**

∙ 순차 리스트 배열을 기반으로 구현된 리스트

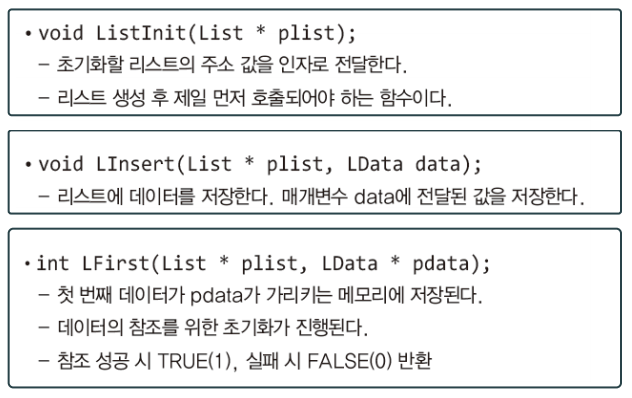
∙ 연결 리스트 메모리의 동적 할당을 기반으로 구현된 리스트

**리스트의 특징**

∙ 저장 형태 데이터를 나란히(하나의 열로) 저장한다.

∙ 저장 특성 중복이 되는 데이터의 저장을 허용한다.

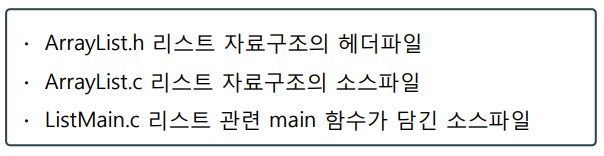
**ADT 정의**



LData는 저장 대상의 자료형을 결정할 수 있도록 typedef로 선언된 자료형의 이름이다.



리스트의 ADT를 기반으로 main 함수를 정의해봅시다



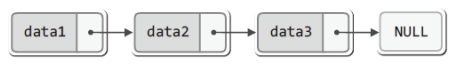
**연결 리스트**

typedef struct \_node {

int data; // 데이터를 담을 공간

struct \_node \* next; // 연결의 도구!

} Node;



typedef struct \_node {

int data;

struct \_node \* next;

} Node;