**3. 연결 리스트**

1. **추상 자료형(Abstract Data Type)**

추상 자료형은 컴퓨터 공학에서 흔히 등장하는 용어이다. 좁은 범위에서의 추상 자료형과 넓은 범위에서의 추상 자료형이 다르게 느껴질 수 있지만 큰 맥락 안에서는 같은 대상을 말하는 것이다.

1. **자료구조에서의 추상 자료형**

구현에 대한 구체적인 언급 없이 데이터의 집합과 그 활용방식이 드러난 자료집합을 가리켜 추상 자료형이라고 한다. 예를 들어 백원, 오백원, 천원의 정수 타입을 가지는 Wallet이라는 구조체를 선언하고 TakeOutMoney, PutMoney라는 함수의 매개변수에 Wallet 타입을 사용한다면 이는 추상 자료형이다.

자료구조의 학습 방법은 ADT의 정의 – ADT를 근거로 한 자료구조의 활용 – ADT를 근거로 자료구조 구현을 따른다.

**배열을 이용한 리스트의 구현**

리스트는 구현 방법에 따라 크게 두 가지로 나뉜다.

순차 리스트 – 배열을 기반으로 구현된 리스트

연결 리스트 – 메모리의 동적 할당을 기반으로 구현된 리스트

각종 자료구조의 ADT는 표준이 아니다. 사용 집단의 목적에 따라 ADT는 달라질 수 있다. 리스트의 가장 기본적인 특성은 리스트는 데이터를 나란히 저장한다는 특성이다. 그리고 중복을 허용한다. 리스트는 중복을 허용하지 않는 집합과는 다르다.

Void ListInit(List \* plist);

초기화할 리스트의 주소 값을 인자로 전달한다.

리스트 생성 후 제일 먼저 호출되어야 하는 함수이다.

Void LInsert(List \* plist, LData data);

리스트에 데이터를 저장한다. 매개변수 data에 전달된 값을 저장한다.

Int LFirst(List \* plist, LData \* pdata);

첫 번째 데이터가 pdata가 가리키는 메모리에 저장된다.

데이터의 참조를 위한 초기화가 진행된다.

참조 성공 시 TRUE(1), 실패 시 FALSE(0) 반환

Int LNext(List \* plist, LData \* pdata);

참조된 데이터의 다음 데이터가 pdata가 가리키는 메모리에 저장된다.

순차적인 참조를 위해 반복 호출이 가능하다.

참조를 새로 시작하려면 먼저 LFirst 함수를 호출해야 한다.

참조 성공 시 TRUE(1), 실패 시 FALSE(0) 반환

LData LRemove(List \* plist)

LFirst 또는 LNext 함수의 마지막 반환 데이터를 삭제한다.

삭제된 데이터는 반환된다.

마지막 반환 데이터를 삭제하므로 연이은 반복 호출을 허용하지 않는다.

Int LCount(List \* plist)

리스트에 저장되어 있는 데이터의 수를 반환한다.

위와 같은 헤더 파일의 선언이 가능하다.

어레이 리스트의 가장 큰 특징은 어레이의 총 길이가 정적으로 할당된 뒤에 배열을 사용한다는 것이다. 동적으로 할당할 수도 있지만 일반적으로 컴파일 시에 정적할당된 길이 만큼의 배열을 이용한다. 또 인자의 갱신 또는 삽입, 삭제 시에 해당 인자의 뒤에 있는 모든 인자들이 값을 한 칸씩 앞으로 이동해야 한다는 단점이 있다. 따라서 갱신이 잦은 프로그램의 자료구조로 사용하기에는 부적절하다. 그러나 검색과 리딩만을 목적으로 하는 데이터인 경우 정적으로 할당되어 빠르게 해당 인덱스를 찾아갈 수 있으므로 유용한 자료구조라고 할 수 있다.

ADT를 사용해 구조체로 초기 선언 이후 사용하는 방식도 자주 쓰인다

#define SIZE 100;

Typedef struct \_WALLET

{

Int arr[SIZE]

int dollar;

int coin;

} wallet;

등으로 선언해 사용할 수 있다.

wallet myWallet;은 달러와 코인이 든 정수형 어레이 리스트가 된다.