**Chapter 03 연결 리스트(Linked list)1**

**03-1 추상자료형: Abstract Data Type**

〮 추상 자료형(ADT): 구체적인 기능의 완성과정을 언급하지 않고, 순수하게 기능이 무엇 인지를 나열한 것 / (사용설명서라고 생각하면 됨)

〮 자료형의 정의 : 구조체(함수) ?

〮 리스트 자료구조 학습의 학습 순서

1. 리스트 자료구조의 ADT를 정의
2. ADT를 근거로 리스트의 자료구조를 활용하는 main함수 정의
3. ADT를 근거로 리스트 구현

**03-2 배열을 이용한 리스트의 구현**

〮 리스트의 이해

* 리스트 자료구조 특징 - > 데이터 나란히 저장/ 중복된 데이터의 저장을 막지 않음
* 구현방법에 따라 순차와 연결로 나뉨

순차 리스트: 배열을 기반으로 구현된 리스트

연결 리스트: 메모리의 동적 할당을 기반으로 구현된 리스트

〮 리스트의 ADT

* Operation:

void ListInit(List \* plist); // 리스트의 초기화

void Linsert(List \* plist, Ldata data); // 데이터 저장

int LFirst(list \* plist, Ldata \* pdata); //저장된 데이터 중 첫번째 데이터 반환 받는 함수

int LNext(list \* plist, Ldata \* pdata); // LF다음 두번째 이후 데이터 얻을 때 사용되는

LData LRemove(List \* plist); // 바로 이전에 참조된 데이터 삭제

int LCount(List \* plist); // 현재 리스트에 저장되어 있는 데이터 수 반환

〮 리스트의 ADT를 기반으로 정의된 main 함수

* *소스파일 참고 (ListMain.c)*
* 실행을 위해 필요한 파일들 ArrayList.h // ArrayList.c // ListMain.c

〮 리스트 배열 기반으로 구현하기

* 헤더파일에 자료구조 구현하고 main함수에서 구현된 자료구조 활용 (완전히 구분)
* *소스파일 참고 (ArrayList.h)*

1. 헤더파일의 정의 / 2. 삽입과 조회 / 3.삭제

〮 배열 기반의 리스트 구현 하나로 묶기

* *소스파일 참고 (ArrayList.c)*

〮 리스트에 구조체 변수 저장하기: Point라는 이름의 구조체 정의하기

* 구조체 Point -> 헤더파일에 저장 (Point.h)
* 구조체 Point 관련 함수들의 선언 및 정의 -> 소스파일에 저장 (Point.c)

관련 함수 : 초기화 / 출력 / 비교

* 배열 기반 리스트 두 파일에 담겨있는 코드를 Point 구조체 변수의 주소 값을 저장할 수 있도록 변경하기 (ArrayList.h / ArrayList.c)

헤더파일 ArrayList.h에서 typedef int LData; -> typedef Point\* LData; 로 변경

Point라는 구조체 등장했으니 다음의 헤더파일 선언이 추가되는 것은 당연

#include “Point.h” // ArrayList.h에 추가할 헤더파일 선언문

실행을 위한 파일 구성 (하나의 프로젝트 안에 모두 컴파일 되야 함)

- Point .h / Point.c :구조체 Point를 위한 파일

- ArrayList.h / Arraylist.c : 배열 기반 리스트

- PointListMain.c : 구조체 Point의 변수 저장

PointListMain.c

* *소스파일 참고*

〮 배열의 장점과 단점

* 단점: 배열의 길이 초기에 결정되어야 하며 변경 불가능 / 삭제 과정에서 데이터의 이동(복사)가 매우 빈번히 일어남
* 장점: 데이터의 참조가 쉬움