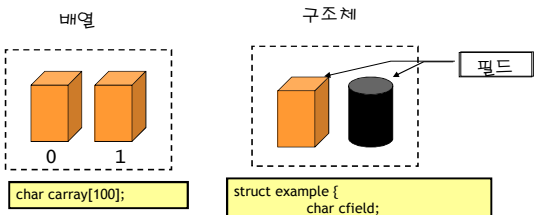
구조체

- 구조체(structure): 타입이 다른 데이터를 하나로 묶는 방법
- 배열(array): 타입이 같은 데이터들을 하나로 묶는 방법




```
char carray[100];
```

```
struct example {
        char cfield;
        int ifield;
        float ffield;
        double dfield;
};
struct example s1;
```


구조체의 사용예

- 구조체의 선언과 구조체 변수의 생성

```
struct person {
        char name[10];      // 문자배열로 된 이름
        int age;            // 나이를 나타내는 정수값
        float height;       // 키를 나타내는 실수값
};
struct person a;            // 구조체 변수 선언
```

- typedef을 이용한 구조체의 선언과 구조체 변수의 생성

```
typedef struct person {
        char name[10];      // 문자배열로 된 이름
        int age;            // 나이를 나타내는 정수값
        float height;       // 키를 나타내는 실수값
} person;
person a;                   // 구조체 변수 선언
```


구조체의 대입과 비교 연산

- 구조체 변수의 대입: 가능

```
struct person {
        char name[10];      // 문자배열로 된 이름
        int age;            // 나이를 나타내는 정수값
        float height;       // 키를 나타내는 실수값
};
main()
{
        person a, b;
        b = a;              // 가능
}
```

- 구조체 변수끼리의 비교: 불가능

```
main()
{
        if( a > b )
                printf("a가 b보다 나이가 많음");    // 불가능
}
```


자체참조 구조체

- **자체 참조 구조체**(self-referential structure): 필드중에 자기 자신을 가리키는 포인터가 한 개 이상 존재하는 구조체
- 연결 리스트나 트리에 많이 등장

```
typedef struct ListNode {
        char        data[10];
        struct ListNode *link;
} ListNode;
```

구조체 응용: 복소수

- 복소수를 표현하기 위한 구조체를 만든다.
  - 복소수: 실수부 + 허수부i
    - 실수부와 허수부는 모두 double로 처리
- 복소수 관련 함수들
  - 복소수 입력함수와 출력 함수
  - 복소수의 덧셈, 뺄셈, 곱셈
  - 복소수의 크기, 켤레복소수 등
- 파일은 어떻게 분리하나?
  - 복소수 구조체 선언 및 인라인 함수: Complex.h
  - 복소수 함수의 extern 선언: Complex.h
  - 복소수 함수 및 관련된 전역 변수: Complex.cpp
  - 복소수를 사용하여 처리하는 함수: AnyApp.cpp

- 복소수를 사용하는 문제
  - 이차방정식 근의 공식 문제
  - 함수 원형:
    - int solveQuadricFunc(double a, double b, double c, Complex& r1, Complex& r2);
    - 매개변수:
      - a, b, c: ax^2 + bx + c = 0
      - r1, r2: 근을 반환 받기 위한 매개변수
        - 자료형은? 포인터나 레퍼런스 형으로 할 것
    - 반환: 근의 개수
      - 2: 실근 2개
      - 1: 중근 1개
      - -2: 허근 2개
  - 파일: quadFunc.cpp (quadFunc.h)

구조체의 배열: 아이템 리스트

- 게임 아이템 리스트 프로그램
  - 게임 아이템을 구조체로 만든다. ➔ 이름??? GameItem
    - 아이템의 이름: char name[80];
    - 가격: int price;
    - 기타 추가하고 싶은 필드…
  - 아이템 리스트를 만든다.
    - 배열을 사용: GameItem ItemList[MAX_ITEMS];
    - 현재 아이템의 개수: int NumItems;
  - 아이템 리스트 관리 프로그램을 만든다.
    - 아이템 추가, 삭제
    - 아이템 리스트 초기화
    - 현재 아이템 리스트 출력
    - 아이템 리스트 파일 읽기, 파일 저장

- 파일 분리?
  - GameItem.h
    - 아이템 구조체 선언
    - Inline의 아이템 처리 함수
    - 아이템 처리 함수의 extern 선언
    - 필요시 default function parameter
  - GameItem.cpp
    - 전역 변수: 아이템 리스트 및 개수
    - 아이템 처리 함수의 구현 (함수 몸체)
  - AnyApp.cpp
    - 게임 아이템 리스트 테스트 프로그램
    - #include “GameItem.h”

- 게임 아이템 구조체

```
struct GameItem {
        char name[10];      // 문자배열로 된 이름
        int  price;         // 나이를 나타내는 정수값
        ...                 // 추가 필드
};
```

- 아이템 리스트
  - 전역 변수로 선언

```
GameItem  ItemList[MAX_ITEMS];      // 아이템 리스트
int  NumItems = 0;                  // 현재 아이템의 개수
```


- 아이템 리스트 처리 함수 설계
  - 아이템 삽입 함수: 무조건 리스트의 맨 뒤에 삽입한다고 가정
    - void addItem(GameItem item);
    - void addItem(char* name, int price);
  - 아이템 삭제 함수: 무조건 리스트의 맨 뒤의 항목 삭제 가정
    - void removeItem();
  - 아이템 리스트 초기화 함수: 현재 리스트의 개수를 0으로 초기화
    - void resetList();
  - 아이템 리스트 화면 출력 함수: 현재 리스트의 화면 출력
    - void printList();

- 파일 입출력
  - 파일 열기 → 읽기, 쓰기 → 파일 닫기
    - FILE *fp = fopen(filename, “r”); // “w”
    - fprintf(fp, “%20s %6d”, name, price);
    - fscan(fp, “%s%6d”, name, &price);
    - fclose(fp);
  - 파일 형식 결정
    - 아이템 개수 + 각 아이템 정보
- 아이템 리스트 파일 입출력 함수
  - 리스트 저장 함수
    - void storeItemList(char* filename);
  - 리스트 로드 함수
    - void storeItemList(char* filename);