WEEK 12

구조체와 공용체

학습목표

I. 구조체와 공용체 기본 개념 이해
II. 자료형 재정의를 위한 typedef 사용
III. 구조체 포인터와 배열을 활용

학습목차

Ⅰ. 구조체

1. 구조체의 개념과 정의
2. 구조체 변수 선언과 초기화
3. 구조체 활용

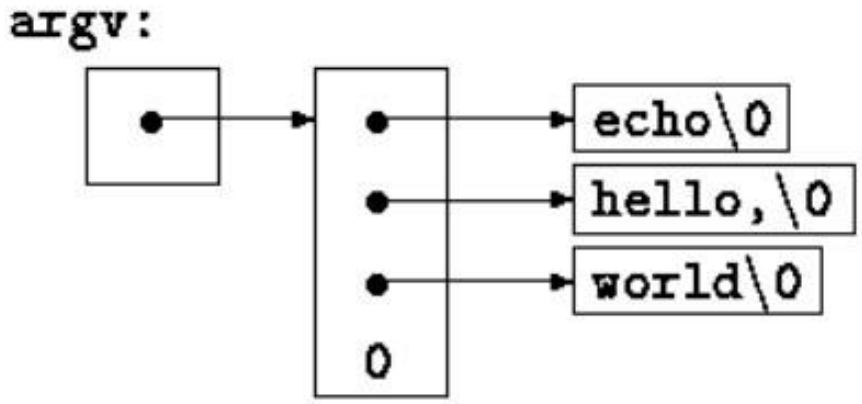
The simplest illustration is the program `echo`, which echoes its command-line arguments on a single line, separated by blanks. That is, the command

```
echo hello, world
```

prints the output

```
hello, world
```

By convention, `argv[0]` is the name by which the program was invoked, so `argc` is at least 1. If `argc` is 1, there are no command-line arguments after the program name. In the example above, `argc` is 3, and `argv[0]`, `argv[1]`, and `argv[2]` are `"echo"`, `"hello,"`, and `"world"` respectively. The first optional argument is `argv[1]` and the last is `argv[argc-1]`; additionally, the standard requires that `argv[argc]` be a null pointer.



[출처] The C Programming Language, 2nd Ed. Ritchie & kernighan

명령행 인자(command line arguments) 복습

```c
#include <stdio.h>

/* echo command-line arguments; 1st version */
main(int argc, char *argv[])
{
    int i;

    for (i = 1; i < argc; i++)
        printf("%s%s", argv[i], (i < argc-1) ? " " : "");
    printf("\n");
    return 0;
}
```

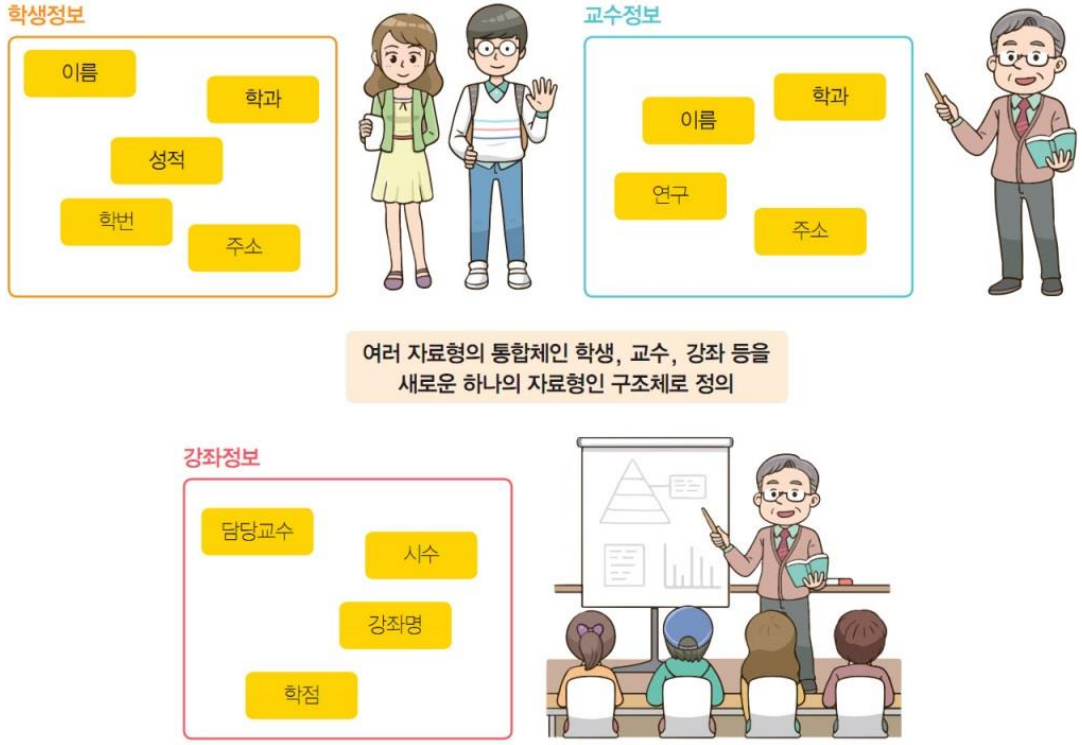
```c
#include <stdio.h>

/* echo command-line arguments; 2nd version */
main(int argc, char *argv[])
{
    while (--argc > 0)
        printf("%s%s", *++argv, (argc > 1) ? " " : "");
    printf("\n");
    return 0;
}
```

[출처] The C Programming Language, 2nd Ed. Ritchie & kernighan

1. 구조체의 개념과 정의

❖ 구조체(structure) 개념

- 연관성이 있는 서로 다른 개별적인 자료형의 변수들을 하나의 단위로 묶은 새로운 자료형
- 연관된 멤버로 구성되는 통합 자료형으로 대표적인 유도 자료형



[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 구조체의 개념과 정의

◆ 구조체 정의

❖ 키워드 struct 사용

- 키워드 struct 다음에 구조체 태그 이름을 기술
- 중괄호 사이에 원하는 멤버를 여러 개의 변수로 선언하는 구조
- 변수 선언에서 이용될 새로운 구조체 자료형을 정의하는 구문

❖ 구조체 멤버

- 구조체를 구성하는 하나 하나의 항목
- 초기값 대입 불가능

문자열 입출력 함수: stdio.h

```
struct 구조체태그이름
{
    자료형 변수명1;
    자료형 변수명2;
    ...
};
```

구조체 구성요소(struct member)라 한다.
초기값을 설정할 수 없다.

세미콜론은 반드시 필요하다.

```
struct account
{
    char name[10];      // 계좌주 이름
    int actnum;         // 계좌 번호
    double balance;     // 잔고
};
```

```
struct lecture
{
    char name[20];      // 강좌명
    int credit;         // 학점
    int hour;           // 시수
};
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 구조체 변수 선언과 초기화

❖ struct 구조체태그이름이 새로운 자료형으로 이용

```
struct   구조체태그이름   변수명;
struct   구조체태그이름   변수명1, 변수명2, 변수명3, ... ;
```

여러 변수의 선언도 가능하다.

```
struct account yours;
struct account act1, act2, act3;
```

❖ 구조체 정의와 변수 선언을 함께하는 방법

```
struct account
{
    char name[12];        //계좌주이름
    int actnum;           //계좌번호
    double balance;       //잔고
} myaccount;

struct account youraccount;
```

변수 myaccount는 struct account형 변수로 선언된다.

변수 youraccount도 struct account형 변수로 선언된다.

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 구조체 변수 선언과 초기화

❖ 변수 선언 시 중괄호를 이용한 초기화 지정이 가능

```
struct 구조체태그이름    변수명 = {초기값1, 초기값2, 초기값3, ... };
```

```
struct account
{
   char name[12];       //계좌주이름
   int actnum;          //계좌번호
   double balance;      //잔고
};

struct account mine = {"홍길동", 1001, 300000 };
struct account mine = {"한국인", 1001};
```

멤버 balance에는 double형 기본값인 0.00이 저장

그림 13-9 구조체 변수의 초기화

TIP 버전 C99 추가 기능

구조체 초기화에서 멤버의 순서와 관계없이 ".멤버이름 = 초기값"으로 지정된 멤버에 초기값을 저장(designated initializer)할 수 있다.

```
struct account me = { .name = "홍길동", .balance = 50000 };
printf("%s %d %.2f\n", me.name, me.actnum, me.balance);
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 구조체 변수 선언과 초기화

◆ 구조체 변수의 크기와 접근 연산자

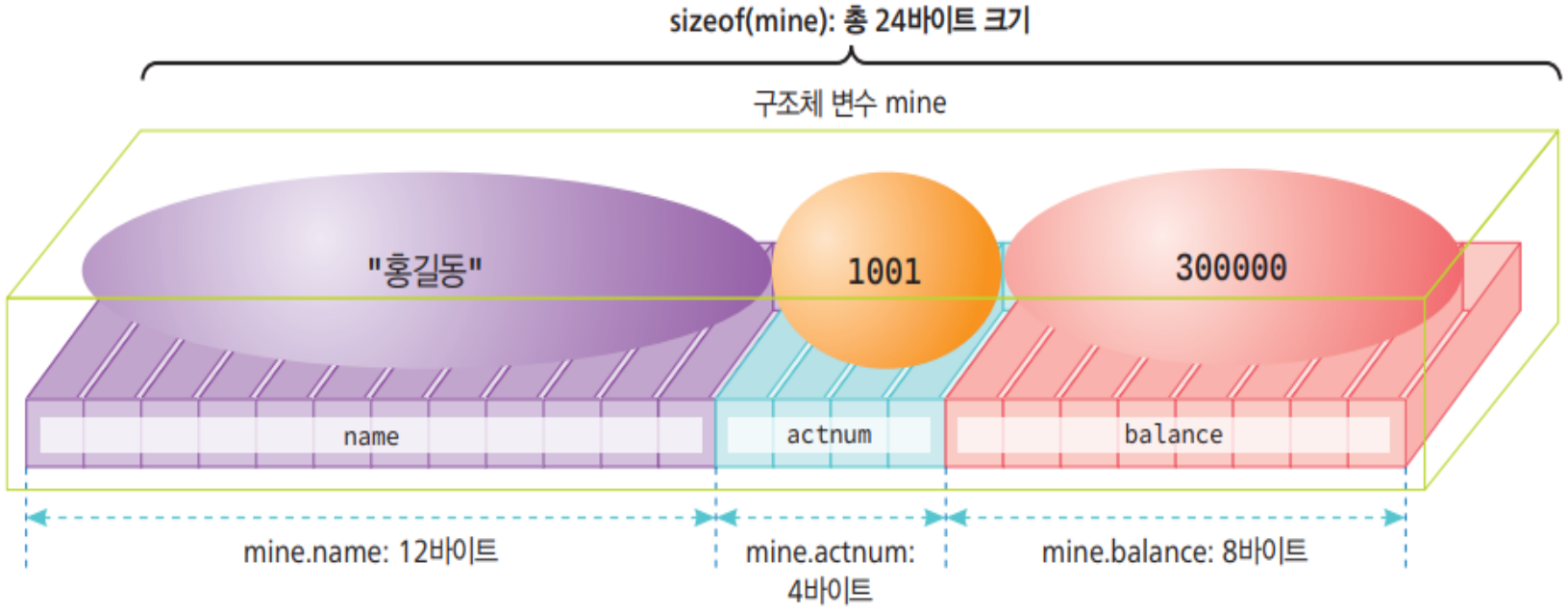
❖ 구조체 멤버 접근 방법

- 참조연산자 . 사용

```
구조체변수이름.멤버
mine.actnum = 1002;    mine.balance = 300000;
```

❖ sizeof

- 실제 구조체의 크기는 멤버의 크기의 합보다 크거나 같음.



[출처] 강환수 외, Perfect C 3판, 인피니티북스

실습예제 13-1 | Prj01 | 01structbasic.c | 구조체 정의와 구조체 변수 선언 | 난이도: ★

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

//은행 계좌를 위한 구조체 정의
struct account
{
    char name[12];     //계좌주 이름
    int actnum;        //계좌번호
    double balance;    //잔고
};

int main(void)
{
    //구조체 변수 선언 및 초기화
    struct account mine = { "홍길동", 1001, 300000 };
    struct account yours;

    strcpy(yours.name, "이동원");
    //strcpy_s(yours.name, 12, "이동원"); //가능
    //yours.name = "이동원"; //오류
    yours.actnum = 1002;
    yours.balance = 500000;

    printf("구조체 크기: %zu\n", sizeof(mine));
    printf("%s %d %.2f\n", mine.name, mine.actnum, mine.balance);
    printf("%s %d %.2f\n", yours.name, yours.actnum, yours.balance);

    return 0;
}
```

구조체 struct account 형인 mine을 선언하면서 초기화, 모든 멤버를 모두 초기화

연산자 sizeof(mine)로 변수 mine의 크기를 조회하여 출력

결과

```
구조체 크기: 24
홍길동 1001 300000.00
이동원 1002 500000.00
```

TIP 버전 C99 추가 기능

이미 선언된 구조체 변수에도 쉽게 멤버 값을 지정하는 방법을 제공한다. 이미 선언된 구조체 you에 초기화 방법과 같이 멤버에 일괄적으로 값을 대입한 후 타입 변환을 사용하여 대입한다.

```c
struct account you;
you = (struct account) { .name = "김파이", .balance = 70000 };
printf("%s %d %.2f\n", you.name, you.actnum, you.balance);
```

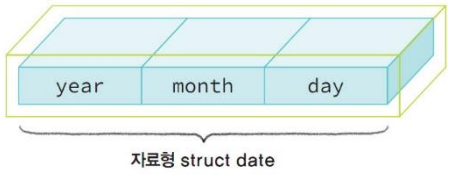
[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 구조체 활용

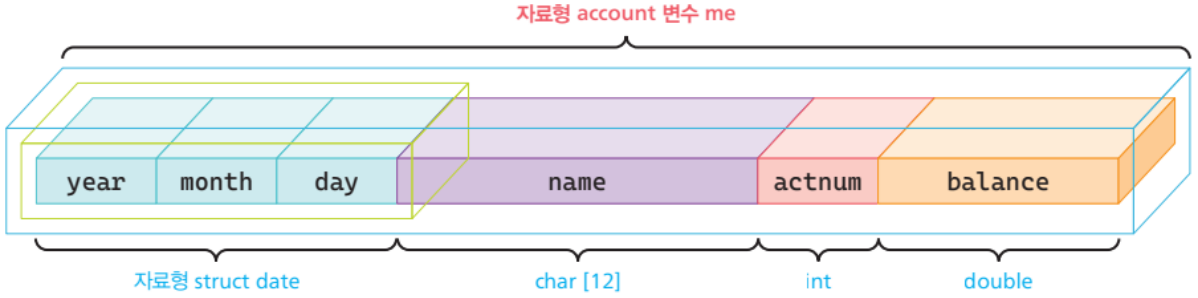
❖ 구조체 멤버로 다른 구조체 사용

- 구조체 멤버로 이미 정의된 다른 구조체 형 변수와 구조체 포인터 변수를 사용 가능

```
struct date
{
   int year;      //년
   int month;     //월
   int day;       //일
};
```




```
struct account
{
   struct date open;     //계좌 개설일자
   char name[12];        //계좌주 이름
   int actnum;           //계좌번호
   double balance;       //잔고
};
struct account me = {{2022, 3, 9}, "홍길동", 1001, 300000 };
```



[출처] 강환수 외, Perfect C 3판, 인피니티북스

Prj02 02nestedstruct.c 구조체 멤버로 다른 구조체 형 포함 난이도: ★

```c
#include <stdio.h>
#include <string.h>

//날짜를 위한 구조체
struct date
{
   int year;   //년
   int month;  //월
   int day;    //일
};

//은행계좌를 위한 구조체
struct account
{
   struct date open;    //계좌 개설일자
   char name[12];       //계좌주 이름
   int actnum;          //계좌번호
   double balance;      //잔고
};

int main(void)
{
   struct account me = { { 2022, 3, 9 }, "홍길동", 1001, 300000 };

   printf("구조체 크기: %zu\n", sizeof(me));
   printf("[%d. %d. %d]\n", me.open.year, me.open.month, me.open.day);
   printf("%s %d %.2f\n", me.name, me.actnum, me.balance);
}
```

구조체 멤버로 다른 구조체 변수를 포함

변수 open을 위한 {}는 생략 가능

중첩된 구조체를 접근하려면 접근연산자를 2번 사용

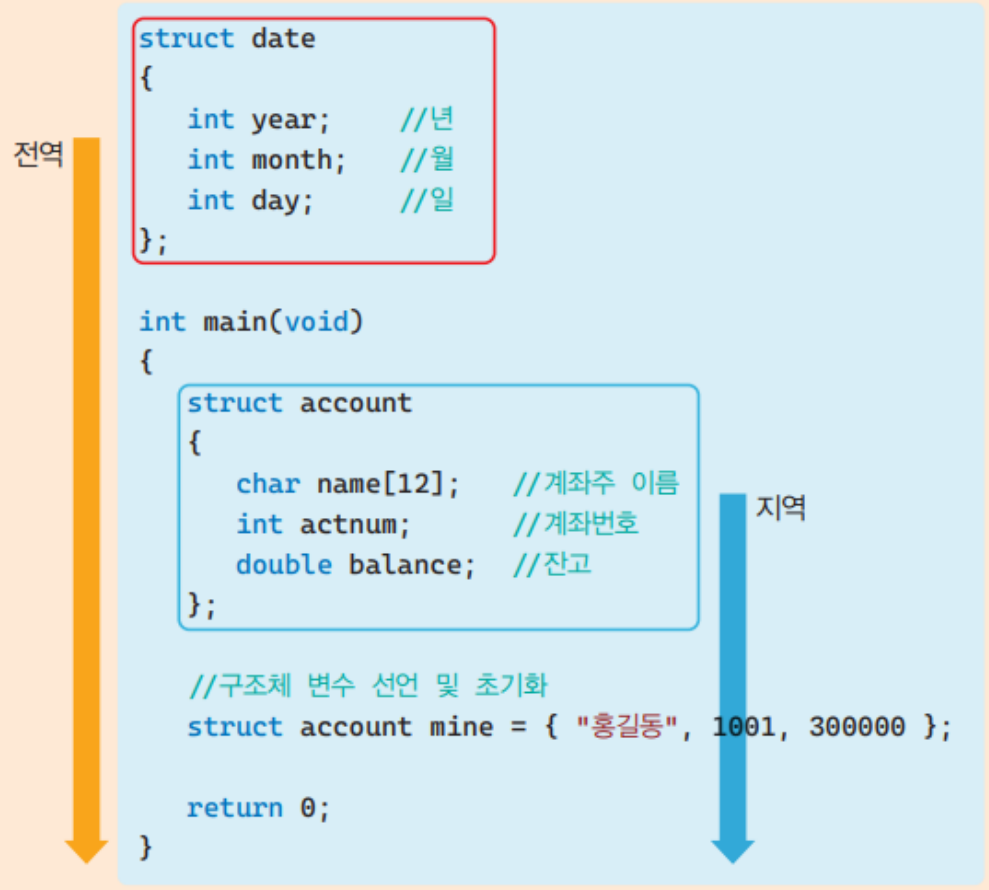
```
구조체 크기: 40
[2022. 3. 9]
홍길동 1001 300000.00
```

산술적인 크기인 36보다 큼

[출처] 강환수 외, Perfect C 3판, 인피니티북스

TIP 구조체 정의의 위치

구조체 정의는 변수의 선언처럼 그 정의 위치에 따라 구조체 자료형의 유효 범위가 결정된다. 즉 **구조체의 정의도 변수 선언처럼 유효범위는 전역(global) 또는 지역(local)으로 모두 가능**하다. 다음과 같이 main() 함수 외부 상단에서 정의된 구조체 struct date는 전역으로 이 파일의 이 위치 이후 모든 함수에서 사용 가능하다. 그러나 main() 함수 내부에서 정의된 구조체 struct account는 지역으로 이 위치 이후 함수 main() 내부에서만 사용 가능하다.




```
struct date
{
    int year;     //년
    int month;    //월
    int day;      //일
};

int main(void)
{
    struct account
    {
        char name[12];    //계좌주 이름
        int actnum;       //계좌번호
        double balance;   //잔고
    };

    //구조체 변수 선언 및 초기화
    struct account mine = { "홍길동", 1001, 300000 };

    return 0;
}
```

그림 13-14 구조체 정의의 유효 범위

[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 구조체 활용

❖ 구조체 변수의 대입과 동등 비교

```c
struct student
{
    int snum;           //학번
    char *dept;         //학과 이름
    char name[12];      //학생 이름
};
struct student hong = { 202200001, "컴퓨터정보공학과", "홍길동" };
struct student one;

one = hong;
```

```c
if ( one == hong ) //오류
    printf("내용이 같은 구조체입니다.\n");
```

동등 비교는 사용 불가능

```c
if (one.snum == hong.snum)
    printf("학번이 %d로 동일합니다.\n", one.snum);
```

구조체 멤버 하나 하나를 비교

```c
if (one.snum = = hong.snum && !strcmp(one.name, hong.name) && !strcmp(one.dept, hong.dept))
    printf("내용이 같은 구조체입니다.\n");
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

Prj03 03structcmp.c 구조체 정의와 비교 난이도: ★★

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

int main(void)
{
    //학생을 위한 구조체
    struct student
    {
        int snum;           //학번
        char* dept;         //학과 이름
        char name[12];      //학생 이름
    };
    struct student hong = { 202200001, "컴퓨터정보공학과", "홍길동" };
    struct student na = { 202200002 };
    struct student you = { 202200003 };

    //학생이름 입력
    scanf("%s", na.name);
    //na.name = "나한국"; //컴파일 오류 '식이 수정할 수 있는 lvalue여야 합니다.'
    //scanf("%s", na.dept); //실행 오류

}
```

변수 dept는 char *로 문자열 상수의 주소가 저장 가능하나, scanf()나 memcpy(), strcpy()로는 문자열 저장이 불가능

변수 name은 char 배열로 문자열 자체의 저장이 가능하므로 scanf()나 memcpy(), strcpy()의 사용도 가능

구조체 struct student 형인 na을 선언하면서 초기화, 첫 번째 멤버인 학번만 기술되었으므로 나머지는 각각 NULL과 NULL 문자로 초기화

[출처] 강환수 외, Perfect C 3판, 인피니티북스

```
23      na.dept = "컴퓨터정보공학과";
24      you.dept = "기계공학과";
25      memcpy(you.name, "홍길동", 7);
26      strcpy(you.name, "홍길동");
27      strcpy_s(you.name, 7, "홍길동");
28
29      printf("[%d, %s, %s]\n", hong.snum, hong.dept, hong.name);
30      printf("[%d, %s, %s]\n", na.snum, na.dept, na.name);
31      printf("[%d, %s, %s]\n\n", you.snum, you.dept, you.name);
32
33      struct student one;
34      one = you;
35      if (one.snum == you.snum)
36          printf("학번이 %d로 동일합니다.\n", one.snum);
37      //if ( one == bae ) //컴파일 오류
38      if (one.snum == you.snum && !strcmp(one.name, you.name) &&
39                                          !strcmp(one.dept, you.dept))
40          printf("내용이 같은 구조체입니다.\n");
41
42      return 0;
43  }
```

구조체 변수 one에 변수 you의 내용을 모든 비교하려면 각각의 멤버를 모두 비교

```
김현식
[202200001, 컴퓨터정보공학과, 홍길동]
[202200002, 컴퓨터정보공학과, 김현식]
[202200003, 기계공학과, 홍길동]

학번이 202200003로 동일합니다.
내용이 같은 구조체입니다.
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

Ⅰ. 구조체

1교시 수업을 마치겠습니다.

Ⅱ. 자료형 재정의와 공용체

1. 자료형 재정의
2. 구조체 자료형 재정의
3. 공용체 활용

1. 자료형 재정의

◆ 자료형 재정의 구문

❖ 키워드 typedef

- 이미 사용되는 자료형을 다른 새로운 자료형 이름으로 재정의

자료형 재정의 typedef 구문

```
typedef   기존자료유형이름   새로운자료형1, 새로운자료형2, ... ;
```

```
typedef int profit;
typedef unsigned int budget;
typedef unsigned int size_t;
typedef unsigned __int64 size_t;
```

여러 이름으로도 재정의할 수 있다.

새로운 이름 size_t도 자료형 unsigned int와 동일하게 이용 가능하다.

[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 자료형 재정의

◆ 자료형 재정의 목적

❖ 프로그램의 시스템 간 호환성과 편의성

- 시스템마다 자료형의 크기가 달라 문제 발생
  - 터보 C++ 컴파일러에서 자료형 int는 저장공간의 크기가 2바이트
  - Visual C++에서는 4바이트

Visual C++: 4바이트

```
int salary = 2000000;
```

Turbo C++: 2바이트

```
int salary = 2000000;
```

오버플로 발생(데이터 손실)

- typedef를 사용하여 새로운 자료형 myint를 정의하여 사용

이 typedef 문장만 수정하면 터보 C++에서 이 소스를 그대로 이용 가능하다.

Visual C++ 소스

```
typedef  int  myint;
...
myint salary = 2000000;
```

Turbo C++ 소스

```
typedef  long  myint;
...
myint salary = 2000000;
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 자료형 재정의

Prj05 05typedef.c 자료형 재정의 키워드 typedef 이용 난이도: ★

```c
#include <stdio.h>

//함수 외부에서 정의된 자료형은 이후 파일에서 사용 가능
typedef unsigned int budget;

int main(void)
{
   budget year = 24500000; //새로운 자료형 budget 사용

   //함수 내부에서 정의된 자료형은 함수 내부에서만 사용 가능
   typedef int profit;
   profit month = 4600000; //새로운 자료형 profit 사용
   printf("올 예산은 %d, 이달의 이익은 %d 입니다.\n", year, month);

   return 0;
}

void test(void)
{
   budget year = 24500000; //새로운 자료형 budget 사용

   //profit은 이 함수에서는 사용 불가, 컴파일 오류 발생
   //profit year;
}
```

budget은 int와 같은 자료형으로 변수 year를 budget으로 선언하면서 초기값 대입

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 구조체 자료형 재정의

❖ struct를 제거한 새로운 자료형

- typedef 사용하여 구조체를 한 단어의 새로운 자료형으로 정의하면 사용하기에 편리
- typedef 사용하여 구조체 struct date를 date로 재정의 가능
- date가 아닌 datetype 등 다른 이름으로도 재정의 가능

```
struct date
{
   int year;     //년
   int month;    //월
   int day;      //일
};

typedef struct date date;
```

자료형인 date는 struct date와 함께 동일한 자료유형으로 이용이 가능하다.

❖ 구조체 정의와 typedef를 함께 사용 가능

- 구조체 자료형 software 정의

```
typedef struct
{
   char title[30];     //제목
   char company[30];   //제작회사
   char kinds[30];     //종류
   date release;       //출시일
} software;
```

software는 변수가 아니라 새로운 자료형이다.

[출처] 강환수 외, Perfect C 3판, 인피니티북스

Prj06 06tdefstruct.c 문장 typedef를 이용하여 구조체 자료형을 다른 이름으로 재정의해 사용 난이도: ★

```c
#include <stdio.h>

struct date
{
   int year;      //년
   int month;     //월
   int day;       //일
};

//struct date 유형을 간단히 date 형으로 사용하기 위한 구문
typedef struct date date;

int main(void)
{
   //구조체를 정의하면서 바로 자료형 software로 정의하기 위한 구문
   typedef struct
   {
      char title[30];      //제목
      char company[30];    //제작회사
      char kinds[30];      //종류
      date release;        //출시일
   } software;

   software vs = { "비쥬얼스튜디오 커뮤니티", "MS", "통합개발환경",
                                                   { 2022, 8, 29 } };

   printf("제품명: %s\n", vs.title);
   printf("회사 : %s\n", vs.company);
   printf("종류 : %s\n", vs.kinds);
   printf("출시일: %d. %d. %d\n", vs.release.year,
                                   vs.release.month, vs.release.day);

   return 0;
}
```

date는 구조체 struct date 의 새로운 자료형

software는 변수가 아니라 구조체의 새로운 자료형

```
제품명: 비쥬얼스튜디오 커뮤니티
회사 : MS
종류 : 통합개발환경
출시일: 2022. 8. 29
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 공용체 활용

❖ union을 사용한 공용체 정의

- 서로 다른 자료형의 값을 동일한 저장공간에 저장하는 자료형
  - 공용체 변수의 크기는 멤버 중 가장 큰 자료형의 크기
- union을 struct로 사용하는 것을 제외하면 구조체 선언 방법과 동일

공용체 정의 및 변수 선언 구문

```
union 공용체태그이름
{
    자료형 멤버변수명1;
    자료형 멤버변수명2;
    ...
} [변수명1] [,변수명2];
```

공용체 구성요소인 멤버(struct member)이다.

세미콜론은 반드시 필요하다.

```
union data
{
    char ch;        //문자형
    int cnt;        //정수형
    double real;    //실수형
} data1;
```

```
union udata
{
    char name[4];   //char형 배열
    int n;          //정수형
    double val;     //실수형
};
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 공용체 활용

❖ 공용체의 멤버는 모든 멤버가 동일한 저장공간을 사용

- 동시에 여러 멤버의 값을 동시에 저장하여 이용 불가능
- 마지막에 저장된 하나의 멤버의 자료값만을 저장
- 공용체도 구조체와 같이 typedef를 이용하여 새로운 자료형으로 정의 가능

❖ 초기값

- 공용체 정의 시 처음 선언한 멤버의 초기값으로만 저장 가능
- 만일 다른 멤버로 초기값을 지정하면 컴파일 시 경고가 발생
- 초기값으로 동일한 유형의 다른 변수의 대입도 가능

```
typedef union data uniondata;

    uniondata data2 = {'A'};        //첫 멤버인 char형으로만 초기화 가능
    //uniondata data2 = {10.3};     //컴파일 시 경고 발생
```

warning C4244: '초기화중' : 'double'에서 'char'(으)로 변환하면서 데이터가 손실될 수 있습니다.

```
    uniondata data3 = data2;        //다른 변수로 초기화 가능
```

TIP 버전 C99 추가 기능

구조체처럼 공용체도 멤버의 순서와 관계없이 ".멤버이름 = 초기값"으로 지정된 멤버에 초기값을 저장(designated initializer)할 수 있다.

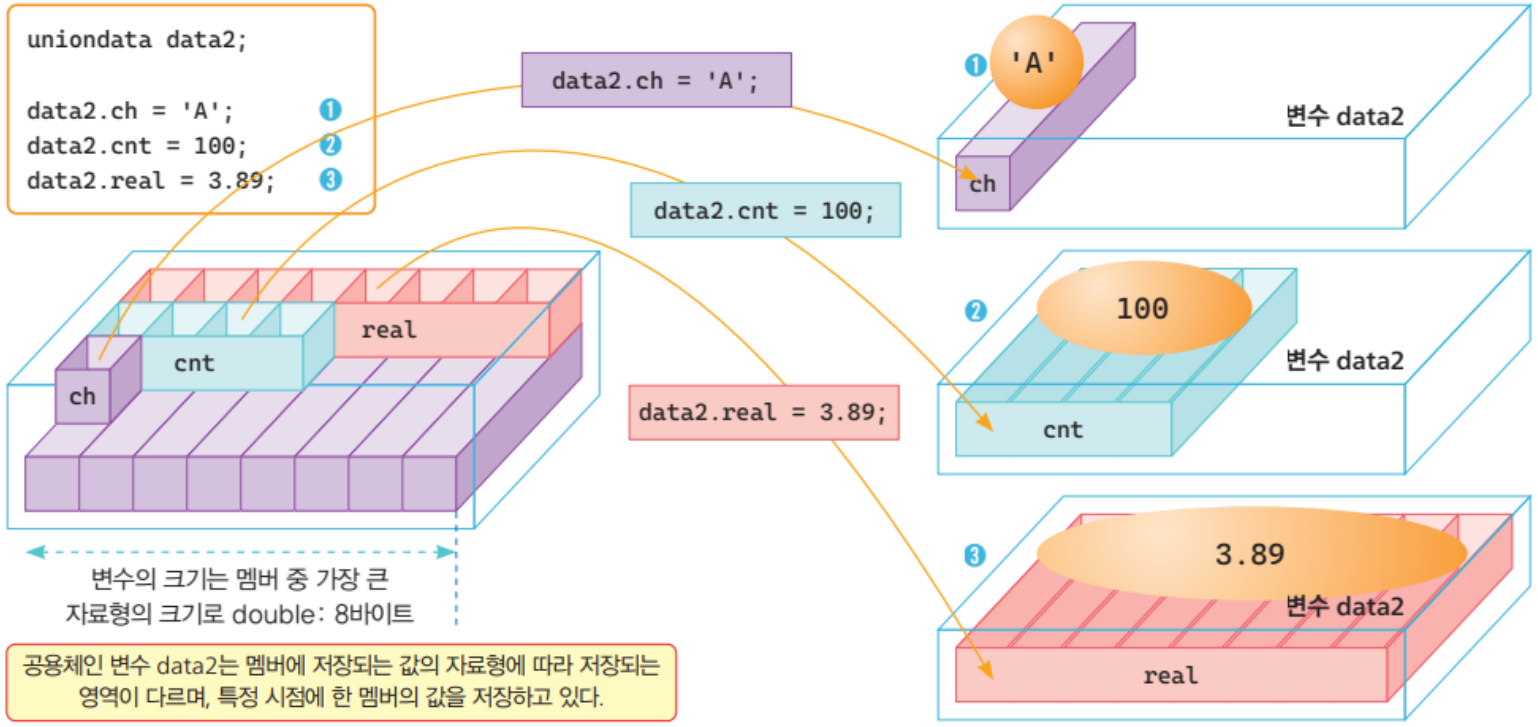
```
union data value = { .real = 3.98 };
printf("%.2f\n", value.real);
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 공용체 활용

❖ 접근연산자 .

- 공용체 변수로 멤버를 접근
- 공용체를 참조할 경우 정확한 멤버를 사용하는 것은 프로그래머의 책임



[출처] 강환수 외, Perfect C 3판, 인피니티북스

실습예제 13-4 | Prj04 | 04union.c | 공용체 정의와 변수 선언 및 사용 | 난이도: ★

```c
#include <stdio.h>

//유니온 구조체를 정의하면서 변수 data1도 선언한 문장
union data
{
    char ch;        //문자형
    int cnt;        //정수형
    double real;    //실수형
} data1;            //data1은 전역변수

int main(void)
{
    union data data2 = { 'A' };  //첫 멤버인 char형으로만 초기화 가능
    union data data3 = { 97.78 }; //컴파일 시 경고 발생
    union data data4 = data2;     //다른 변수로 초기화 가능
    data4.real = 3.78;

    printf("%zu %zu\n", sizeof(union data), sizeof(data3));
    printf("%c %c %f\n", data2.ch, data3.ch, data4.real);

    //멤버 ch에 저장
    data1.ch = 'a';
    printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);
    //멤버 cnt에 저장
    data1.cnt = 100;
    printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);
    //멤버 real에 저장
    data1.real = 3.156759;
    printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);

    return 0;
}
```

warning C4244: '초기화 중': 'double'에서 'char'(으)로 변환하면서 데이터가 손실될 수 있습니다.

초기화 값 97.78에서 정수인 97만 멤버 ch에 저장

공용체인 data1에서 자료형 int인 멤버 cnt에 정수 100을 저장, 이 이후로는 data1.cnt만 의미가 있음

결과

```
8 8
A a 3.780000
a 97 0.000000
d 100 0.000000
N -590162866 3.156759
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

Ⅱ. 자료형 재정의와 공용체

2교시 수업을 마치겠습니다.

Ⅲ. 구조체와 공용체의 포인터와 배열

1. 구조체 포인터
2. 공용체 포인터
3. 구조체 배열

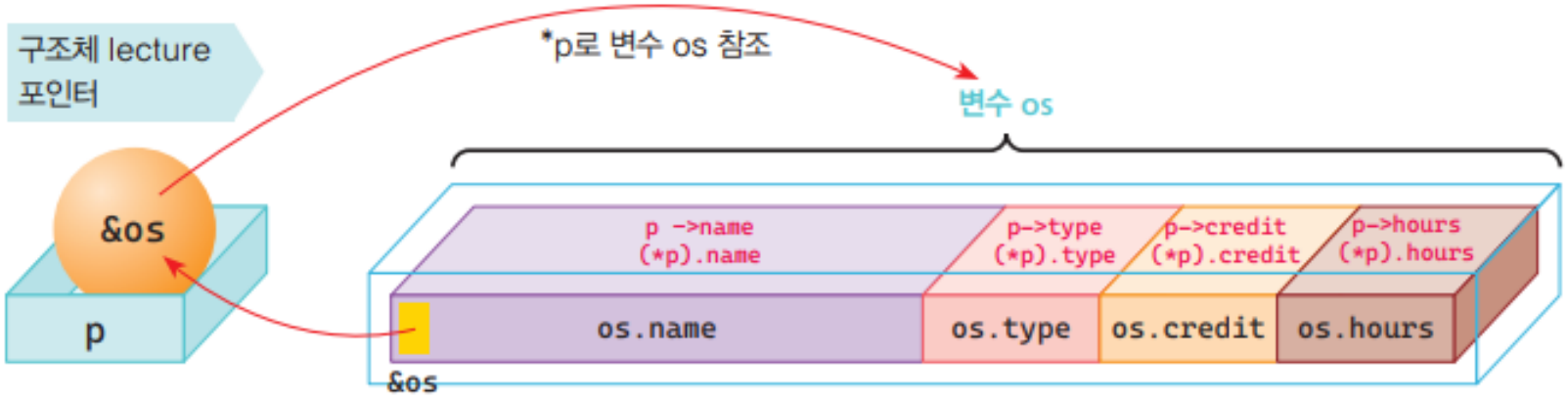
1. 구조체 포인터

❖ 구조체 포인터 변수

- 구조체의 주소값을 저장할 수 있는 변수
- 다른 포인터 변수와 사용 방법이 동일

```
struct lecture
{
    char name[20];          //강좌명
    int type;               //강좌구분
    int credit;             //학점
    int hours;              //시수
};
typedef struct lecture lecture;
lecture *p;
```

```
lecture os = {"운영체제", 2, 3, 3};
lecture *p = &os;
```



[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 구조체 포인터

◆ 포인터 변수의 구조체 멤버 접근

❖ 포인터 p의 구조체 멤버 접근연산자 ->

- p->name과 같이 사용
- (*p).name으로도 사용 가능

❖ 연산식 *p.name과 구분

- *(p.name) 의미
  - 접근연산자()가 간접연산자 (*)보다 우선순위가 빠르므로
  - p가 포인터이므로 p.name은 문법 오류가 발생

접근 연산식	구조체 변수 os와 구조체 포인터 변수 p인 경우의 의미
p->name	포인터 p가 가리키는 구조체의 멤버 name
(*p).name	
*p.name	*(p.name)이고 p가 포인터이므로 p.name 은 문법오류가 발생
*os.name	*(os.name)를 의미하며, 구조체 변수os의 멤버 포인터 name이 가리키는 변수로, 이 경우는 구조체 변수 os 멤버 강좌명의 첫 문자임, 다만 한글인 경우에는 실행 오류
*p->name	*(p->name)을 의미하며, 포인터 p이 가리키는 구조체의 멤버 name이 가리키는 변수로 이 경우는 구조체 포인터 p이 가리키는 구조체의 멤버 강좌명의 첫 문자임, 마찬가지로 한글인 경우에는 실행 오류

[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 구조체 포인터

◆ 멤버 접근연산자 ->, 구조체 멤버 접근연산자 .의 연산자 우선순위

- 다른 어떠한 연산자 우선순위보다 가장 높음
- 연산자 ->와 .은 우선순위 1위
  - 결합성은 좌에서 우이며
- 연산자 *은 우선순위 2
  - 결합성은 우에서 좌

[출처] 강환수 외, Perfect C 3판, 인피니티북스

Prj07 07structptr.c 구조체 포인터의 선언과 사용 난이도: ★★

```c
#include <stdio.h>

struct lecture
{
    char name[20];    //강좌명
    int type;         //강좌구분 0: 교양, 1: 일반선택, 2: 전공필수, 3: 전공선택
    int credit;       //학점
    int hours;        //시수
};
typedef struct lecture lecture;

//제목을 위한 문자열
char* head[] = { "강좌명", "강좌구분", "학점", "시수" };
//강좌 종류를 위한 문자열
char* lectype[] = { "교양", "일반선택", "전공필수", "전공선택" };

int main(void)
{
    lecture os = { "운영체제", 2, 3, 3 };
    lecture c = { "C프로그래밍", 3, 3, 4 };
    lecture* p = &os;

    printf("구조체 크기: %zu, 포인터 크기: %zu\n\n", sizeof(os), sizeof(p));
    printf("%10s %12s %6s %6s\n", head[0], head[1], head[2], head[3]);
    printf("%12s %10s %5d %5d\n", p->name, lectype[p->type],
                                                p->credit, p->hours);

    //포인터 변경
    p = &c;
    printf("%12s %10s %5d %5d\n", (*p).name, lectype[(*p).type],
                                        (*p).credit, (*p).hours);
    printf("%12c %10s %5d %5d\n", *c.name, lectype[c.type],
                                            c.credit, c.hours);

    return 0;
}
```

강좌구분이 2인 "전공필수"로 지정

문자열 "C프로그래밍"에서 첫 글자인 C만 참조하는 연산식

```
구조체 크기: 32, 포인터 크기: 8

      강좌명      강좌구분   학점   시수
    운영체제      전공필수      3      3
  C프로그래밍      전공선택      3      4
           C      전공선택      3      4
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 공용체 포인터

❖ 공용체 포인터 변수로 멤버를 접근

- 접근연산자 ->를 이용

❖ 공용체 포인터 p를 선언

- p->ch = 'a';
  - p가 가리키는 공용체 멤버 ch에 'a'를 저장
- p->cnt, p->real
- 각각 value.cnt, value.real을 참조

```
union data
{
    char ch;
    int cnt;
    double real;
} value, *p ;
```

변수 value는 union data형이며 p는 union data 포인터형으로 선언

```
p = &value;      //포인터 p에 value의 주소값을 저장
p->ch = 'a';     //value.ch = 'a';와 동일한 문장
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

Prj08 08 unionptr.c 공용체 정의와 변수 선언 및 사용 난이도: ★★

```c
#include <stdio.h>

int main(void)
{
    //공용체 union data 정의
    union data
    {
        char ch;
        int cnt;
        double real;
    };

    //유니온 union data를 다시 자료형 udata로 정의
    typedef union data udata;

    //udata 형으로 value와 포인터 p 선언
    udata value, *p;

    p = &value;
    p->ch = 'a';
    printf("%c %c\n", p->ch, (*p).ch);
    p->cnt = 100;
    printf("%d ", p->cnt);
    p->real = 3.14;

    return 0;
}
```

char, int, double 자료형 중 하나를 동시에 저장할 수 있는 8바이트 공간의 공용체 unin data를 정의하기 위한 문장 시작으로 6행에서 11행까지 정의, 이 공용체 정의로 이 위치 이후 모든 파일에서 공용체 unin data 사용 가능

변수 value와 p는 모두 함수 main() 내부에서만 사용이 가능한 지역변수

연산식 p->ch와 (*p).ch는 모두 value.ch를 참조

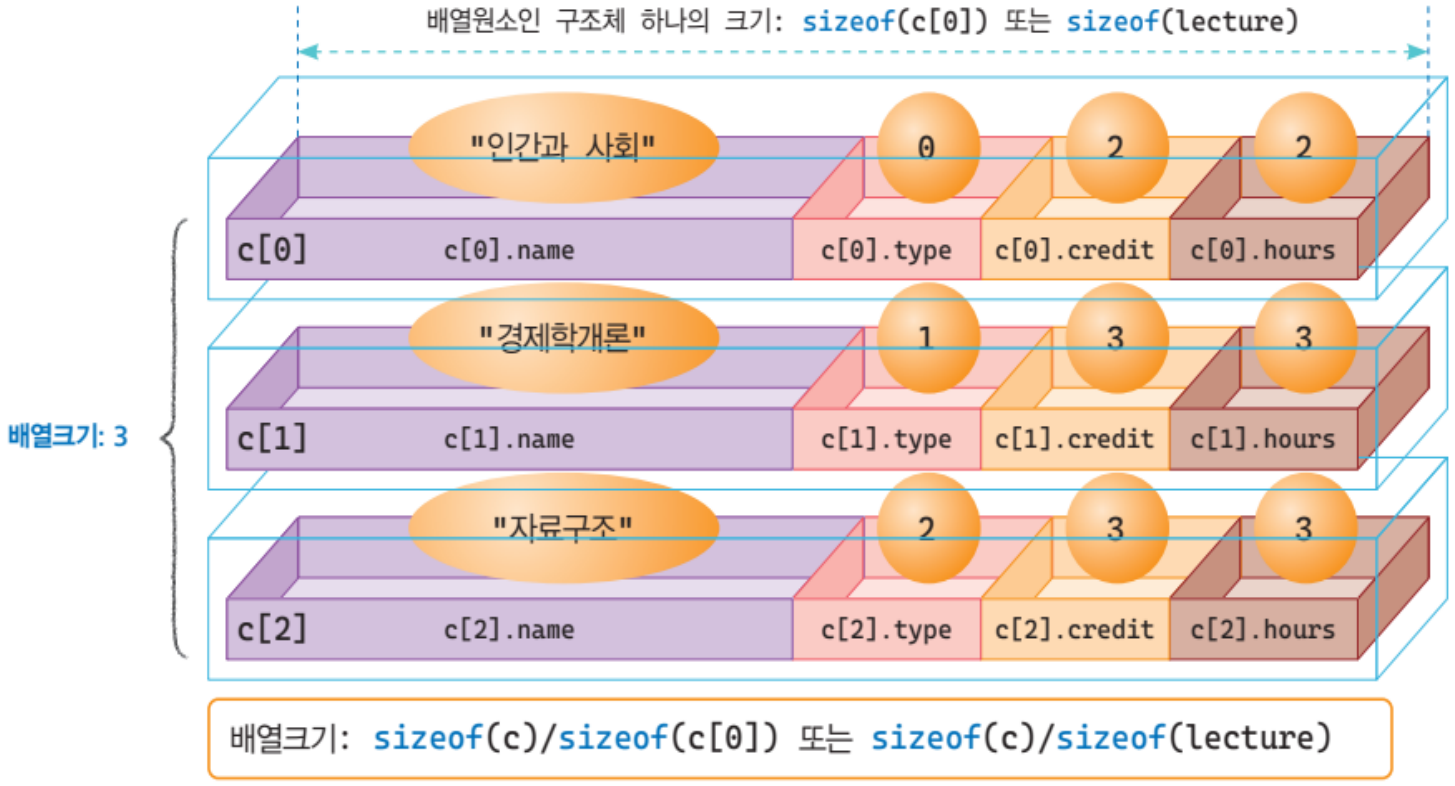
```
a a
100 3.14
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 구조체 배열

❖ 구조체 lecture 배열

```
lecture c[] = { {"인간과 사회", 0, 2, 2},
                {"경제학개론", 1, 3, 3},
                {"자료구조", 2, 3, 3}};
```



[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 구조체 배열

Prj09 09structary.c 구조체 배열을 선언한

```c
#include <stdio.h>

struct lecture
{
   char name[20];    //강좌명
   int type;         //강좌구분
   int credit;   //학점
   int hours;    //시수
};
typedef struct lecture lecture;

char* lectype[] = { "교양", "일반선택", "전공필수", "전공선택" };
char* head[] = { "강좌명", "강좌구분", "학점", "시수" };

int main(void)
{
   //구조체 lecture의 배열 선언 및 초기화
   lecture course[] = { { "인간과 사회", 0, 2, 2 },
                        { "경제학개론", 1, 3, 3 },
                        { "자료구조", 2, 3, 3 },
                        { "모바일프로그래밍", 2, 3, 4 },
                        { "고급 C프로그래밍", 3, 3, 4 } };

   int arysize = sizeof(course) / sizeof(course[0]);

   printf("배열크기: %d\n\n", arysize);
   printf("%12s   %12s %6s %6s\n", head[0], head[1], head[2], head[3]);
   printf("================================================\n");

   for (int i = 0; i < arysize; i++)
      printf("%16s %10s %5d %5d\n", course[i].name,
         lectype[course[i].type], course[i].credit, course[i].hours);

   return 0;
}
```

구조체 struct의 배열을 선언하면서 바로 초기값을 대입, 배열 크기는 지정하지 않고 초기값을 지정한 원소 수가 5

```
배열크기: 5

          강좌명      강좌구분     학점    시수
================================================
      인간과 사회          교양     2     2
       경제학개론      일반선택     3     3
         자료구조      전공필수     3     3
   모바일프로그래밍      전공필수     3     4
   고급 C프로그래밍      전공선택     3     4
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 구조체 배열

◆ 복소수를 위한 구조체

❖ 복소수 a + bi

- 구조체 complex

```
struct complex
{
        double real;   // 실수
        double img;    // 허수
};
typedef struct complex complex;
```

자료형 struct complex　　자료형 complex

자료형 struct complex와 complex
모두 복소수 자료형으로 사용 가능

[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 구조체 배열

◆ 복소수를 위한 구조체

실습예제 11-7 complexnumber.c

구조체를 사용하여 복소수를 표현, 함수의 인자와 반환형으로 사용

```c
// file: complexnumber.c

#include <stdio.h>

struct complex
{
    double real;  //실수
    double img;   //허수
};
typedef struct complex complex;

complex paircomplex1(complex com);
void paircomplex2(complex *com);
void printcomplex(complex com);

int main(void)
{
    complex comp = {3.4, 4.8};
    complex pcomp;

    printcomplex(comp);
    pcomp = paircomplex1(comp);
    printcomplex(pcomp);
    paircomplex2(&pcomp);
    printcomplex(pcomp);

    return 0;
}

void printcomplex(complex com)
{
    printf("복소수(a + bi) = %5.1f + %5.1fi \n", com.real, com.img);
}

complex paircomplex1(complex com)
{
    com.img = -com.img;
    return com;
}

void paircomplex2(complex *com)
{
    com->img = -com->img;
}
```

실행결과

```
복소수(a + bi) =   3.4 +   4.8i
복소수(a + bi) =   3.4 +  -4.8i
복소수(a + bi) =   3.4 +   4.8i
```

[출처] 강환수 외, C언어 스케치, 인피니티북스

Ⅲ. 구조체와 공용체의 포인터와 배열

3교시 수업을 마치겠습니다.