13 구조체와 공용체

구조체

❖ 구조체의 필요성

- 책에 대한 정보를 살펴보면 매우 다양

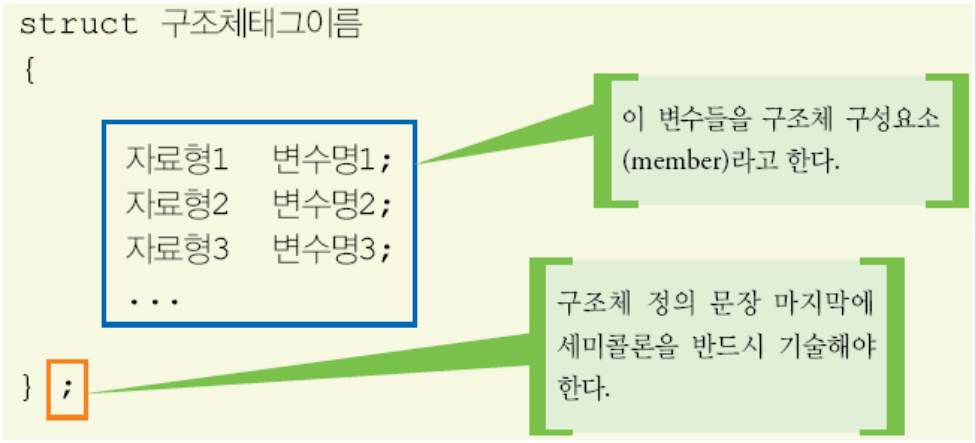
> **책 정보**
>
> **제목, 저자, 출판사, 페이지수, 가격, ISBN 등**

❖ struct

- 서로 다른 자료형의 변수들을 묶어서 만든 하나의 새로운 자료형을 구조체(struct)
- 구조체는 연관된 멤버로 구성되는 통합 자료형으로 대표적인 유도자료형

❖ 구조체 정의

- 중괄호 사이에 원하는 여러 개의 변수를 선언
- 중괄호 다음 마지막에 세미콜론(;)을 반드시 기술



구조체 정의와 변수 선언

- 구조체 자료형 정의
  - 구조체 멤버(구성요소, member)
    - 구조체 정의 구문에서는 각 구성요소의 초기값을 대입할 수 없음
    - 한 구조체 내부에서 선언되는 구조체 멤버의 이름은 모두 유일
  - 구조체의 정의는 구조체 struct book을 새로운 자료형으로 정의하는 구문
    - 구조체 정의는 구조체 변수를 선언하는 구문이 아님
- 구조체 변수 선언
  - 구조체 정의 구문은 변수의 범위와 비슷
    - 구조체 정의 구문의 위치에 따라 구조체 선언 시 자료형 이용의 범위가 정해짐

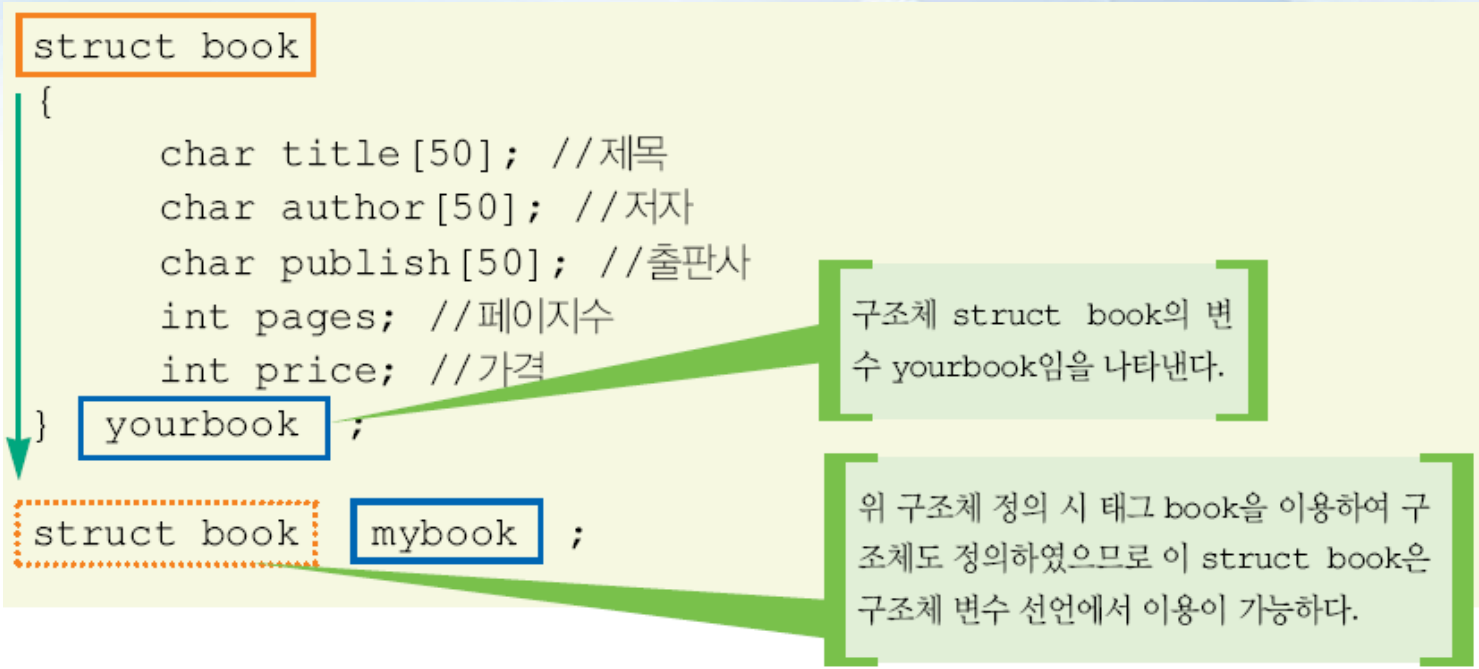
```
struct book
{
      char title[50]; //제목
      char author[50]; //저자
      char publish[50]; //출판사
      int pages; //페이지수
      int price; //가격
};

int main(void)
{
      struct book   mybook   ;
      ...
}
```

구조체 정의 문장 이후, 이 파일 내부에서는 이 구조체의 이용이 가능하다.

구조체 정의와 변수 선언

- 구조체 변수를 선언하는 다른 방법은 다음과 같이 구조체 정의와 변수 선언을 함께하는 방법



- 위 구문에서 변수 yourbook과 mybook은 모두 같은 구조체 struct book 자료형
- 위 구조체 정의 구문에서 구조체 태그 이름 book을 생략해도 구조체 변수 선언이 가능

자료형이 다른 구조체

- 다음 구문에서 구조체 변수 mybook과 yourbook은 자료형이 다른 변수
  - 즉 동일한 구조체 태그 이름으로 선언된 변수만이 동일한 자료형의 구조체가 됨
  - 자료형이 다르면 대입연산자를 사용 불가능

```
struct
{
      char title[50]; //제목
      char author[50]; //저자
      char publish[50]; //출판사
      int pages; //페이지수
      int price; //가격
} mybook ;
```

```
struct
{
      char title[50]; //제목
      char author[50]; //저자
      char publish[50]; //출판사
      int pages; //페이지수
      int price; //가격
} yourbook ;
```

typedef를 이용한 형 선언

❖ **typedef를 이용한 형 선언**

- 구조체 struct book이 정의된 상태에서 구조체 struct book 좀 더 간단하게 선언

❖ **typedef로 구조체 정의**

- 구조체 정의 자체를 typedef와 함께하는 방법
- 아래 typedef 구문에서 새로운 자료형으로 정의되는 키워드는 software로서,
  - 이 구문 이후에는 구조체를 선언할 때 software를 이용하여 형의 선언이 가능

```
struct book {
        char title[50]; //제목
        char author[50]; //저자
        char publish[50]; //출판사
        int pages; //페이지수
        int price; //가격
};

typedef  struct book  book ;
…
book yourbook;
book mybook;
```

```
typedef  struct
{
    char title[50]; //제목
    char company[50]; //제작회사
    char kinds[50]; //종류
    int size; //크기
    int price; //가격
}  software  ;
...
software  visualc;
software  turboc;
```

초기 값 지정

❖ 구조체 struct book 정의

- 구조체 정의에서는 구조체 멤버에 초기 값을 지정 불가능

```
struct book {
        ...
        int pages = 200;
        int price = 30000;
};
typedef struct book book;
```

구조체 정의에서 멤버의 초기 값을 저장하는 것은 허용되지 않는다.

❖ 구조체 변수 mybook에 초기 값을 대입

- 구조체 변수도 배열과 같이 중괄호를 이용하여 초기 값을 대입 가능

```
book mybook = {"C@PL.com", "강 환수", "학술정보", 530, 20000};
```

❖ 멤버 참조

- 구조체 멤버를 접근하기 위해서는 멤버 접근 연산자인 마침표(.)를 이용

```
구조체변수.구조체멤버
mybook.author
```

예제 소스

❖ structbook.c

- 구조체 정의, 변수 선언, 구조체 이용 프로그램

```
구조체 struct book의 사이즈는 sizeof (struct book) =

C 언어 국내 서적

저자 : 강 환수, 제목 : C@PL.com, 페이지수 : 530
출판사 : 인피니티북스, 가격 : 20000

저자 : Al Kelly, 제목 : A Book On C, 페이지수 : 830
출판사 : 홍릉과학출판사, 가격 : 24000

저자 : 강 환수, 제목 : C@PL.com, 페이지수 : 530
출판사 : 인피니티북스, 가격 : 20000
Press any key to continue
```

```c
////////////////////////////////////////////////////////////////////////////////
/* file : structbook.c */

#include <stdio.h>

int main(void)
{
    struct
    {
        char title[50];     //제목
        char author[50];    //저자
        char publish[50];   //출판사
        int pages;          //페이지수
        int price;          //가격
    } yours = {"A Book On C", "Al Kelly", "홍릉과학출판사", 830, 24000};

    struct book
    {
        char title[50];     //제목
        char author[50];    //저자
        char publish[50];   //출판사
        int pages;          //페이지수
        int price;          //가격
    } webook, mybook = {"C@PL.com", "강 환수", "학술정보", 530, 20000};

    printf("구조체 struct book의 사이즈는 sizeof (struct book) = %d\n\n", sizeof (struct book) );

    printf("C 언어 국내 서적\n\n");
    printf("저자 : %s, 제목 : %s, 페이지수 : %d\n출판사 : %s, 가격 : %d\n\n",
            mybook.author, mybook.title, mybook.pages, mybook.publish, mybook.price);
    printf("저자 : %s, 제목 : %s, 페이지수 : %d\n출판사 : %s, 가격 : %d\n\n",
            yours.author, yours.title, yours.pages, yours.publish, yours.price);

    webook = mybook;

    printf("저자 : %s, 제목 : %s, 페이지수 : %d\n출판사 : %s, 가격 : %d\n",
            webook.author, webook.title, webook.pages, webook.publish, webook.price);

    return 0;
}
////////////////////////////////////////////////////////////////////////////////
```

복소수 구조체

❖ 복소수를 표현하는 구조체 complex를 정의

- 복소수는 a+bi로 표현되며, 실수부 a와 허수부 b는 실수 값
- 구조체 struct complex는 다음과 같이 정의

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

구조체 주의

- 메인 함수 상단 전역 부분에서 **struct complex**를 정의
  - 메인 함수 내부에서 다시 같은 이름으로 struct complex를 정의하면서 변수 comp를 하나 선언
- **comp = comp2;**와 같은 대입문은 에러
  - 두 변수 comp1과 comp2는 다시 complex 유형으로 선언

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

typedef struct complex complex;

int main(void)
{
      struct complex {
            double real; //실수
            double img; //허수
      } comp;

      complex comp1 = {3, 4}, comp2 = {2, 5};

      comp.real = comp1.real;
      comp.img = comp2.img;
      //comp = comp2; //서로 자료형이 다르므로 에러 발생
      ...
}
```

위 구조체 complex와 아래의 구조체 complex는 다른 구조체이다.

구조체 포인터

- 구조체 변수를 가리키는 주소 값을 저장하려면 구조체 포인터 변수를 이용

```
struct univ {
        char title[50]; //이름
        char address[50]; //주소
        int students; //학생수
};

struct univ ku = {"한국대학교", "서울시 서초구", 5000};
struct univ *ptr = &ku;
```

- 변수 ku는 구조체 변수이고, 변수 ptr은 구조체를 가리키는 포인터

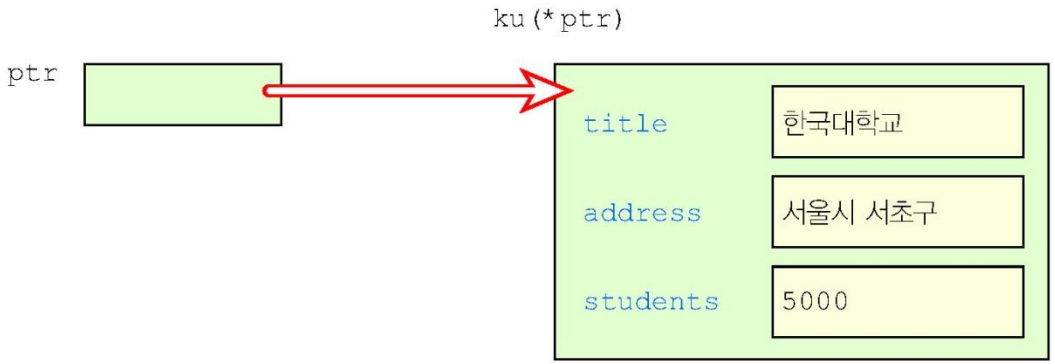


그림 16.6 구조체 변수와 구조체 포인터 변수

구조체 포인터 이용

❖ 연산자 ->

- 구조체 포인터 변수 ptr을 이용하여 구조체의 멤버를 참조하려면 연산자 ->를 이용
  - 연산자 ->는 구조체 멤버 참조 연산자로서 연산자 우선순위가 가장 높음

```
printf("학교 : %s, 주소 : %s, 학생수 : %d\n",
                    ptr->title, ptr->address, ptr->students);
```

❖ 간접 연산자 *

- 변수 ptr을 이용하여 구조체의 멤버 member를 참조하는 다른 방법은 (*ptr).member를 이용하는 방법

```
printf("학교 : %s, 주소 : %s, 학생수 : %d₩n",
          (*ptr).title, (*ptr).address, (*ptr).students);
```

- 위 구문에서 (*ptr).title은 *ptr.title과는 다른 의미이므로 반드시 괄호를 사용
- *ptr.title은 *(ptr.title)을 의미하므로 ptr이 포인터 변수이면 에러가 발생
  - 구조체의 멤버를 참조하는 연산자 ->와 .의 연산자 우선순위가 가장 높기 때문

포인터 변수

❖ 구조체 변수 및 포인터 변수를 이용하는 다음 4가지 구문을 잘 구별하도록

구 문	의 미
(*ptr).title	포인터 ptr이 가리키는 구조체의 멤버 title
ptr->title	위와 같은 의미로 포인터 ptr이 가리키는 구조체의 멤버 title
*ku.title	* (ku.title)을 의미하며, 구조체 변수 ku의 멤버 포인터 title이 가리키는 변수로, 이 경우는 구조체 변수 ku의 학교 이름의 첫 문자임
*ptr->title	* (ptr->title)을 의미하며, 포인터 ptr이 가리키는 구조체의 멤버 title이 가리키는 변수로 이 경우는 구조체 포인터 ptr이 가리키는 구조체의 학교 이름의 첫 문자임

표 16.1 구조체 변수를 이용한 멤버의 참조

구조체 배열

- 구조체 변수를 여러 개 선언하기 위해서 구조체 배열을 이용
  - 구조체 book으로 배열 clang[3]을 선언하여 구조체 원소 3개를 선언하는 구문

```
struct book {
        char author[50];
        char title[50];
        int pages;
};

struct book clang[3] =
      { {"Deitel", "C How To Program", 600},
        {"Al Kelly", "A Book On C", 700},
        {"Stephen Prata", "C Primer Plus", 800} };
```

- 다른 배열과 같이 구조체 배열도 첨자를 이용하여 각 원소를 참조하며, 첨자는 0부터 (배열크기-1)까지 가능
  - 첫 구조체 원소의 멤버를 출력하는 구문

```
printf("저자 : %s, 제목 : %s, 페이지수 : %d₩n",
        clang[0].author, clang[0].title, clang[0].pages);
```

구조체 인자, 값 전달

❖ 복소수 연산에 이용되는 함수를 구현

- 함수 paircomplex1()는 전달 인자 복소수의 켤레 복소수를 구하여 반환하는 함수
  - 그러므로 변수 pcomp에는 {3.4, -4.8}이 저장

```
complex comp = {3.4, 4.8};
complex pcomp;
pcomp = paircomplex1(comp);
```

❖ 구현

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

- 위와 같이 구조체는 함수의 전달 인자와 반환 값으로 이용이 가능
- 위 함수는 구조체 인자를 값에 의한 호출(call by value) 방식으로 이용
- 즉 함수 paircomplex1() 내부에서 지역 구조체 변수 com을 하나 만들어 실인자의 구조체 값을 모두 복사하는 방식으로 구조체 값을 전달

구조체 인자, 주소 전달

❖ 인자를 call by address로

- 이전 함수를 주소에 의한 호출(call by address) 방식으로 변환
- 다음 함수 paircomplex2()는 인자를 주소 값으로 저장하여, 실인자의 변수 comp의 값을 직접 수정하는 방식

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

- 위 함수를 호출하기 위해서는 comp의 주소 값을 이용해야 하므로 다음과 같이 호출

```
paircomplex2(&comp)
```

- 구조체가 많은 멤버를 가지거나, 큰 배열을 멤버로 가지는 경우, 구조체 자체를 인자로 전달하는 것은 매우 비효율
- 이러한 경우는 주소 값으로 전달하는 방식인 주소에 의한 호출 방식이 효율적

예제 소스

❖ complexfunction .c

- 함수 paircomplex1()과 paircomplex2()를 구하는 프로그램

```
복소수 =   3.4 +   4.8i
복소수 =   3.4 +  -4.8i
복소수 =   3.4 +   4.8i
Press any key to continue
```

그림 16.9 예제 complexfunction 결과

```
////////////////////////////////////////////////////////////
/* file : complexfunction.c */

#include <stdio.h>

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

complex paircomplex1(complex com);
void printcomplex(complex com);
void paircomplex2(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 = %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;
}
////////////////////////////////////////////////////////////
```

공용체

❖ 정의
- 공용체는 서로 다른 자료형을 동일한 저장 공간에 이용하는 자료형

❖ 공용체 union data를 정의하는 구문
- 구조체 정의에서 키워드 struct를 union으로 사용한 것과 비슷

```
union data {
        char ch;
        int cnt;
        double real;
};
```

- 공용체의 멤버는 모든 멤버가 동일한 저장 공간을 사용하므로 동시에 여러 멤버의 값을 저장하여 이용할 수 없으며, 마지막에 저장한 하나의 멤버의 자료 값만을 저장
- 공용체 union data 정의와 함께 변수 data1과 포인터 변수 pdata를 선언하는 구문

```
union data {
        char ch;
        int cnt;
        double real;
} data1, *pdata ;
```

공용체 메모리 내부

❖ 메모리

- 공용체 union data 자료형의 변수 data1은
  - 멤버의 유형이 char, int, double이므로
  - 멤버 중 가장 큰 크기인 double 형의 8바이트를 공용체의 저장공간으로 확보하여 세 개의 멤버가 함께 이용

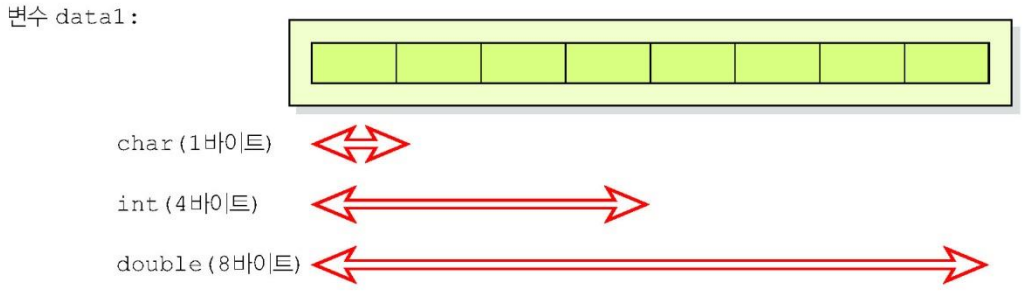


그림 16.10 공용체의 저장 공간

```
union data
{
        char ch;
        int cnt;
        double real;
} data1, *pdata ;
```

❖ 동일한 저장장소를 함께 이용

- 멤버가 char인 경우는 8바이트 중에서 첫 1바이트만 이용하고,
- int인 경우는 전체 공간의 첫 4바이트만 이용하며,
- double인 경우는 8바이트 공간을 모두 사용

공용체 이용

- 구조체와 같이 typedef를 이용하여 새로운 자료형으로 정의

```
typedef union data uniondata;
```

- 구조체의 초기값은 첫 멤버의 초기 값으로만 저장이 가능하고
  - 다른 동일한 변수의 값으로 초기화 가능

```
//uniondata data1 = {10};  //에러 발생
uniondata data2 = {'A'}; //첫 멤버인 char 유형으로만 초기화 가능
uniondata data3 = data2; //다른 변수로 초기화 가능
```

- 공용체 변수로 멤버를 접근하기 위해서는
  - 구조체와 같이 접근 연산자 '.'을 이용하며, 포인터인 경우는 연산자 '->'를 이용

```
pdata = &data2;
printf("%2c %2c₩n", pdata->ch, (*pdata).ch);
printf("%2c %2c₩n", data2.ch, data3.ch);
```

- 공용체 변수 data1의 멤버 ch에 문자 'a'를 저장하는 구문
  - 이 문장 이후에 멤버 cnt나 real을 출력하는 것은 의미가 없음

```
data1.ch = 'a';
printf("%c, %d, %6.2f\n", data1.ch, data1.cnt, data1.real);
```

열거형

❖ 키워드 enum을 이용하여 열거형을 정의

- 열거형 구문은 관련 있는 정수형 상수 목록 집합을 정의하는 구문

```
enum color {yellow, red, blue, magenta, green};
enum color col;
```

- 위 문장은 변수 col을 열거형 enum color로 선언하는 구문
- 열거형 enum color는 색상을 나타내는 5개의 상수 yellow, red, blue, magenta, green을 표현하는 의미
- color는 열거형 태그이름으로 다음과 같이 생략 가능

```
enum {yellow, red, blue, magenta, green} col;
```

❖ 정수 상수에 대응

- 5개의 상수 yellow, red, blue, magenta, green은
  - 각각 0에서부터 4까지의 정수 상수에 대응
- 정수 상수는 변수 col에 대입할 수 있고, 상수 0에 해당하는 값을 표현

```
col = yellow;
```

열거형 이용

- enum day7을 정의하고,
  - typedef를 이용하여 새로이 열거형 자료형 day를 정의하는 구문

```
enum day7 {sun, mon, tue, wed, thu, fri, sat};
typedef enum day7 day;
```

- 새로운 자료형 day는 enum day7로 변수 선언에 다음과 같이 이용할 수 있고, 변수를 선언하면서 초기 값으로 상수 fri를 대입하는 문장

```
day today = fri;
```

- 열거형 day는 7개의 상수 sun, mon, tue, wed, thu, fri, sat를 가지며, 각각 0에서부터 6까지의 정수 상수에 대응
- 열거형 enum pl 정의에서는 필요한 경우, 상수 값을 각 상수에 지정 가능

```
enum pl {c=1972, cpp=1983, java=1995, cs=2000};
typedef enum pl plang;
```

- 다음은 circle 상수는 0, tri는 3, rect는 4, star는 7, dia는 8로 정의
  - 즉 상수 값을 지정한 상수는 그 값으로, 지정되지 않은 상수는 그 이후로 1씩 증가한 상수 값으로 정의

```
enum shape {circle, tri=3, rect=4, star=7, dia};
typedef enum shape shape;
```

열거형 예제

```
enum Day7 {sun, mon, tue, wed, thu, fri, sat};
typedef enum Day7 Day;

Day week;

switch (week)
{
case sun : break;
case mon : break;


}
```

```
int week; // 1 ~ 7

switch (week)
{
case 1 : break;    // sun
case 2 : break;    // mon


}
```