

개정3판

Visual
Studio
2017

쉽게 풀어쓴



언어

EXPRESS



천인국 지음

제13장 구조체



이번 장에서 학습할 내용

2

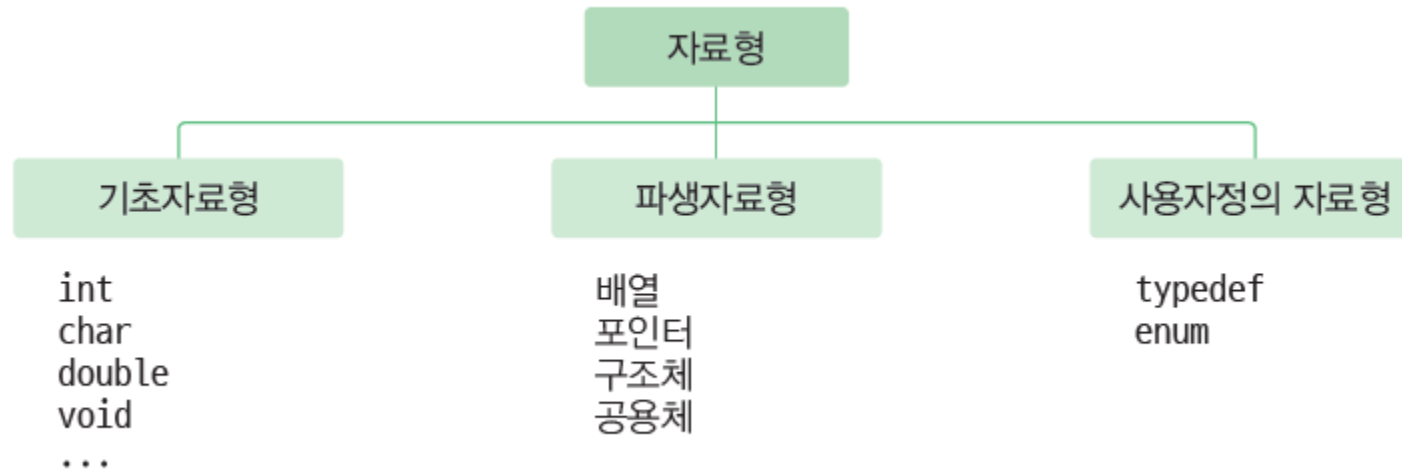
- 구조체의 개념, 정의, 초기화 방법
- 구조체와 포인터와의 관계
- 공용체와 typedef

구조체는 서로 다른
데이터들을 하나로
묶는 중요한
도구입니다.





자료형의 분류





구조체의 필요성

4

- 학생에 대한 데이터를 하나로 모으려면?



학번: 20100001(정수)
이름: "최자영"(문자열)
학점: 4.3(실수)
...



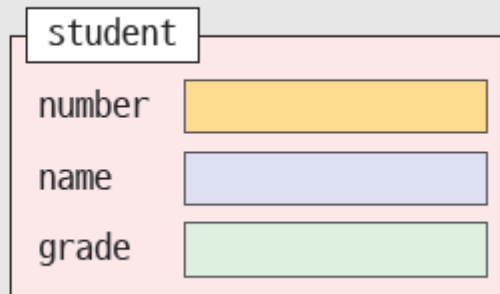
```
int number;  
char name[10];  
double grade;  
와 같이 개별 변수  
로 나타낼 수 있지  
만 묶을 수가 있나?
```



구조체의 필요성

5

```
struct student {  
    int number;    // 학번  
    char name[10]; // 이름  
    double grade;  // 학점  
};
```

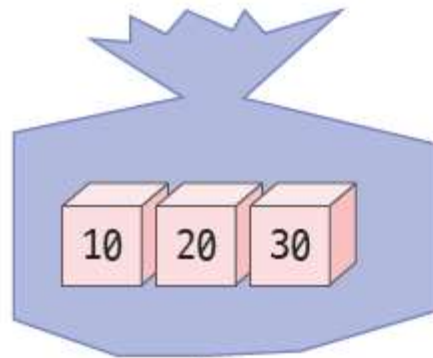


구조체를 사용
하면 변수들을
하나로 묶을 수
있습니다.

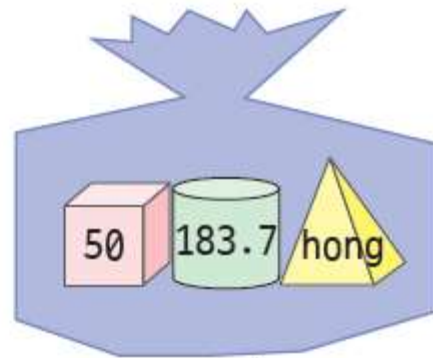


구조체와 배열

6



배열



구조체



중간 점검

7

1. 구조체와 배열의 차이점을 이야기해보라.
2. 복소수, 날짜, 화면의 좌표, 사각형 등을 표현하는데 필요한 데이터를 나열해보라.





구조체 선언

8

Syntax: 구조체 선언

예

```
struct student {  
    int number;    // 학번  
    char name[10]; // 이름  
    double grade;  // 학점  
};
```

구조체의 이름(태그)

구조체를 정의할 때 사용하는 키워드

구조체의 멤버

마지막에 세미콜론이 있어야 한다.



구조체 선언

9

- 구조체 선언은 변수 선언은 아님

구조체를 정의하는 것은 와플
이나 붕어빵을 만드는 틀을 정
의하는 것과 같다.



구조체

와플이나 붕어빵을 실제로 만
들리려 위해서는 구조체 변수
를 선언하여야 한다.



구조체 변수



구조체 선언의 예

10

```
// x값과 y값으로 이루어지는 화면의 좌표
struct point {
    int x;           // x 좌표
    int y;           // y 좌표
};
```

```
// 복소수
struct complex {
    double real;      // 실수부
    double imag;      // 허수부
};
```

```
// 날짜
struct date {
    int month;
    int day;
    int year;
};
```

```
// 사각형
struct rect {
    int x;
    int y;
    int width;
    int height;
};
```

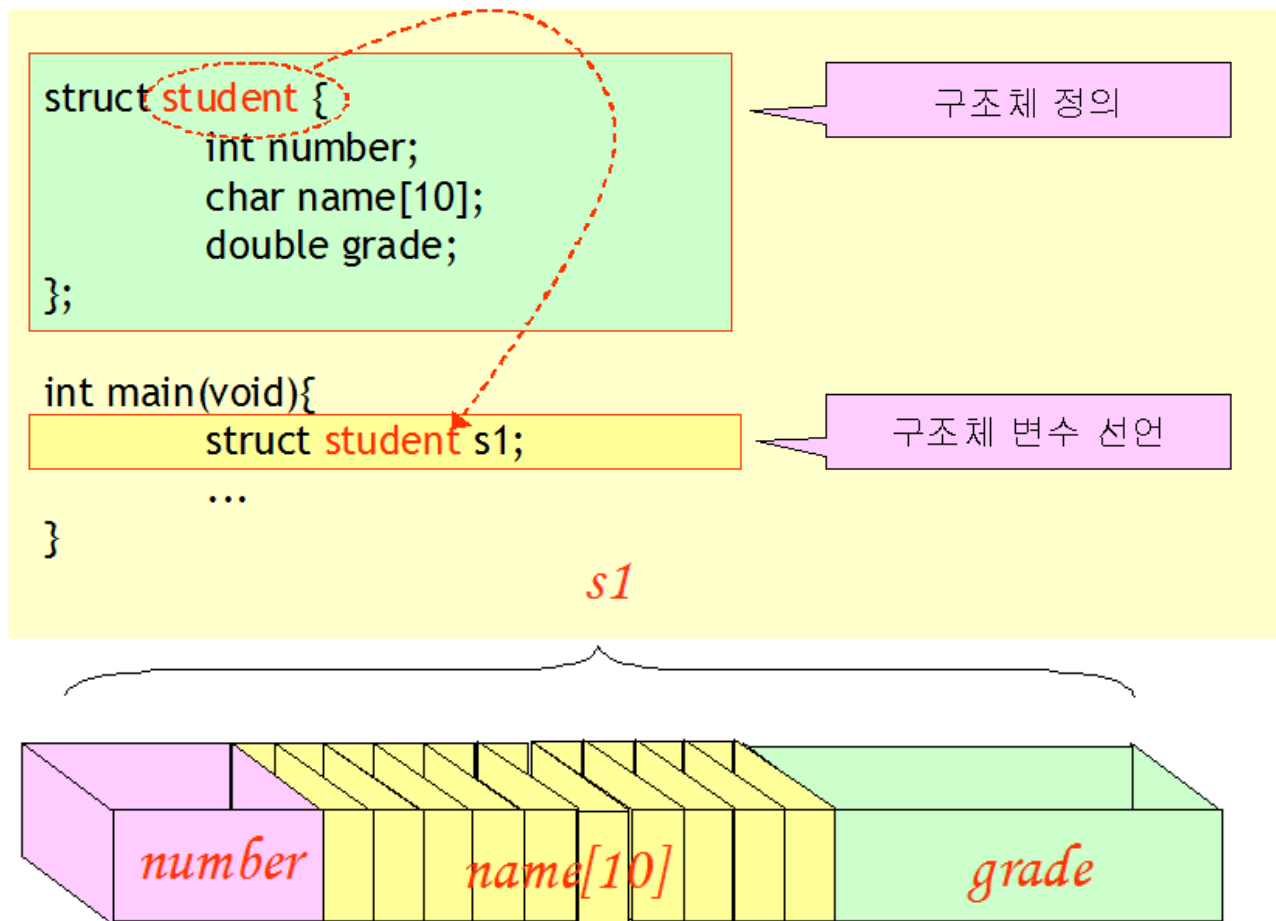
```
// 직원
struct employee {
    char name[20];    // 이름
    int age;           // 나이
    int gender;        // 성별
    int salary;        // 월급
};
```



구조체 변수 선언

11

- 구조체 정의와 구조체 변수 선언은 다르다.



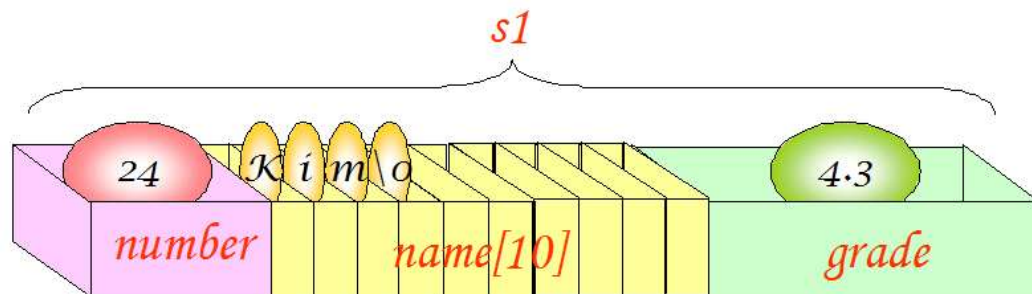


구조체의 초기화

12

- 중괄호를 이용하여 초기값을 나열한다.

```
struct student {  
    int number;  
    char name[10];  
    double grade;  
};  
struct student s1 = { 24, "Kim", 4.3 };
```





구조체 멤버 참조

13

Syntax: 구조체 멤버 접근

예 `s1.grade = 3.8;`

구조체 변수 구조체 멤버



.기호는
구조체에서
멤버를 참조할
때 사용하는
연산자입니다.



예제 #1

student1.c

14

...

```
struct student {  
    int number;  
    char name[10];  
    double grade;  
};
```

구조체 선언

```
int main(void)
```

```
{
```

```
    struct student s;
```

구조체 변수 선언

```
    s.number = 20190001;  
    strcpy(s.name, "홍길동");  
    s.grade = 4.3;
```

구조체 멤버 참조

```
    printf("학번: %d\n", s.number);  
    printf("이름: %s\n", s.name);  
    printf("학점: %f\n", s.grade);  
    return 0;
```

```
}
```

학번: 20190001

이름: 홍길동

학점: 4.300000



예제 #2

15

student2.c

```
struct student {  
    int number;  
    char name[10];  
    double grade;  
};
```

구조체 선언

학번을 입력하시오: 20190001
이름을 입력하시오: 홍길동
학점을 입력하시오(실수): 4.3
학번: 20190001
이름: 홍길동
학점: 4.300000

```
int main(void)  
{
```

```
    struct student s;
```

구조체 변수 선언

```
    printf("학번을 입력하시오: ");  
    scanf("%d", &s.number);
```

구조체 멤버의 주소 전달

```
    printf("이름을 입력하시오: ");  
    scanf("%s", s.name);
```

```
    printf("학점을 입력하시오(실수): ");  
    scanf("%lf", &s.grade);
```

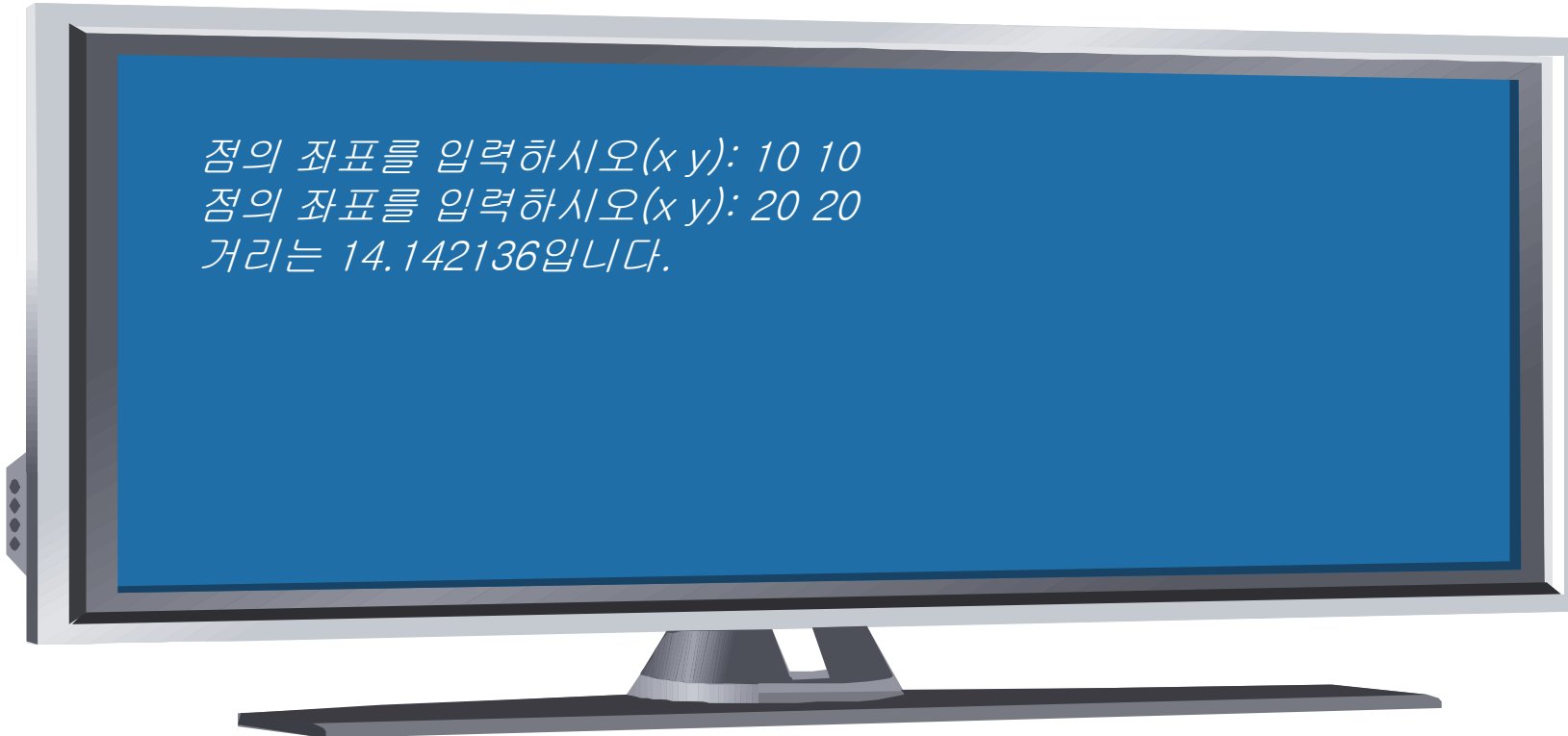
```
    printf("학번: %d\n", s.number);  
    printf("이름: %s\n", s.name);  
    printf("학점: %f\n", s.grade);  
    return 0;
```

```
}
```



lab: 2차원 공간 상의 점을 구조체로 표현하기

- 사용자로부터 두 점의 좌표를 입력받아서 두 점사이의 거리를 계산하여 보자. 점의 좌표를 구조체로 표현한다.





point.c

17

```
#include <math.h>
```

```
struct point {
```

```
    int x;
```

```
    int y;
```

```
};
```

```
int main(void)
```

```
{
```

```
    struct point p1, p2;
```

```
    int xdiff, ydiff;
```

```
    double dist;
```

```
    printf("점의 좌표를 입력하시오(x y): ");
```

```
    scanf("%d %d", &p1.x, &p1.y);
```

```
    printf("점의 좌표를 입력하시오(x y): ");
```

```
    scanf("%d %d", &p2.x, &p2.y);
```

```
    xdiff = p1.x - p2.x;
```

```
    ydiff = p1.y - p2.y;
```

```
    dist = sqrt((double)(xdiff * xdiff + ydiff * ydiff));
```

```
    printf("두 점사이의 거리는 %f입니다.\n", dist);
```

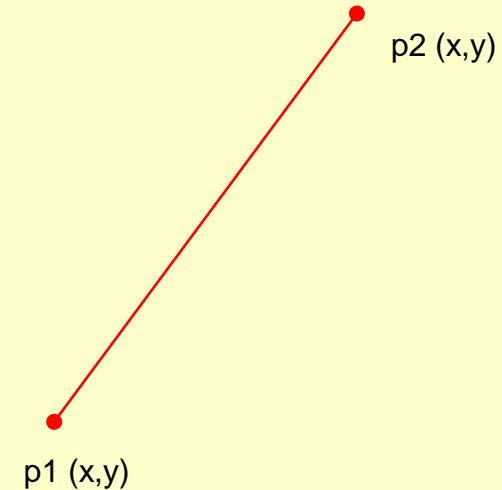
```
    return 0;
```

```
}
```

점의 좌표를 입력하시오(x y): 10 10

점의 좌표를 입력하시오(x y): 20 20

두 점사이의 거리는 14.142136입니다.





중간 점검

18

1. 구조체 안에 선언된 각각의 변수들을 _____이라고 한다.
2. 구조체의 선언에 사용하는 키워드는 _____이다.
3. 구조체의 태그는 왜 필요하며, 태그를 사용하는 경우와 사용하지 않은 경우가 어떻게 다른가?
4. 구조체의 선언만으로 변수가 만들어지는가?
5. 구조체의 멤버를 참조하는 연산자는 무엇인가?





구조체를 멤버로 가지는 구조체

19

```
struct date {                      // 구조체 선언
    int year;
    int month;
    int day;
};
```

```
struct student {                   // 구조체 선언
    int number;
    char name[10];
    struct date dob;              // 구조체 안에 구조체 포함
    double grade;
};

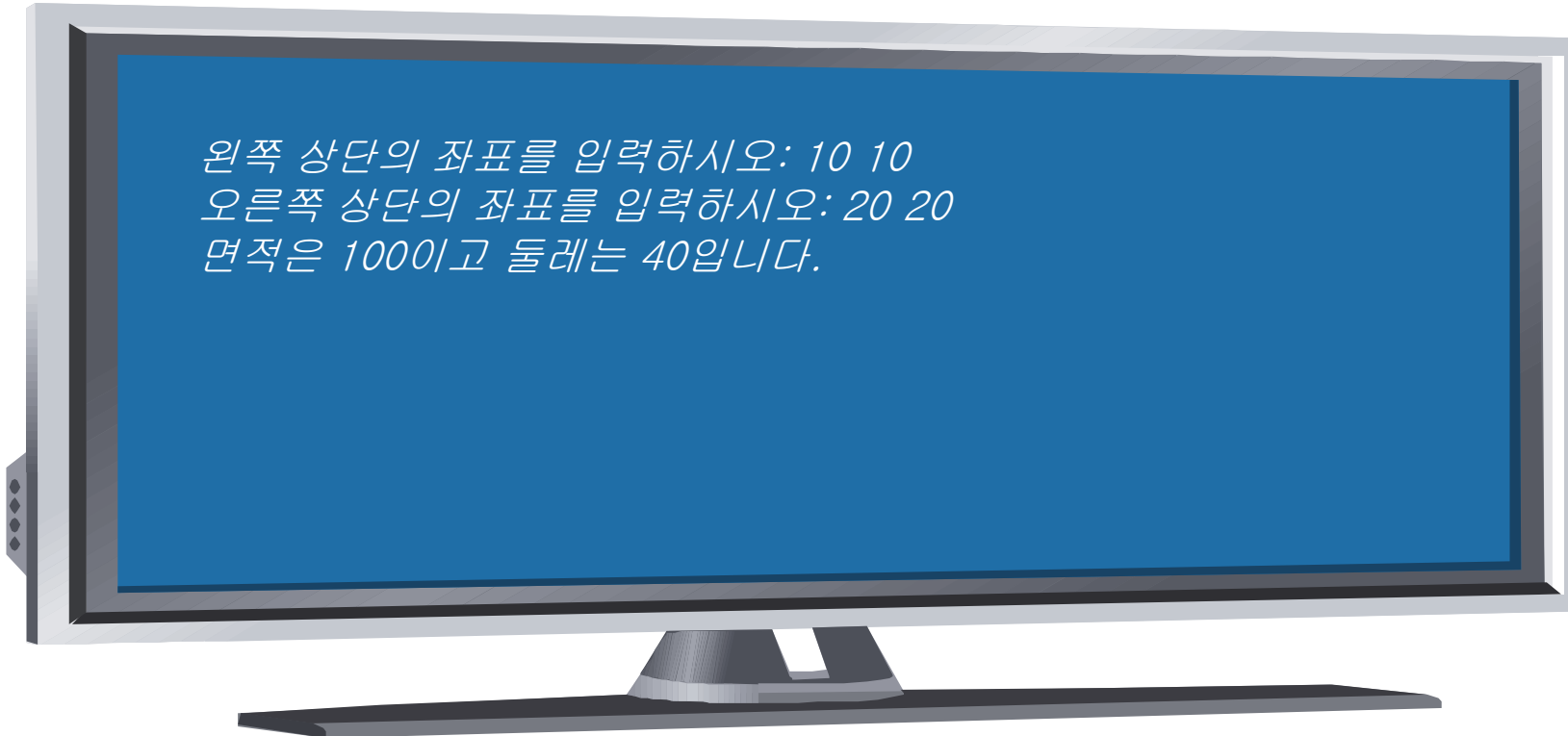
struct student s1;                // 구조체 변수 선언
```

```
s1.dob.year = 1983;               // 멤버 참조
s1.dob.month = 03;
s1.dob.day = 29;
```



lab: 사각형을 point 구조체로 나타내기

- 꼭지점의 좌표를 표시하는데 앞의 예제의 **point** 구조체를 사용하자.





예제

nested_struct.c

21

```
#include <stdio.h>

struct point {
    int x;
    int y;
};

struct rect {
    struct point p1;
    struct point p2;
};

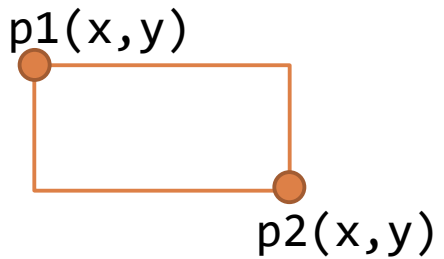
int main(void)
{
    struct rect r;
    int w, h, area, peri;
```





예제

22



```
printf("왼쪽 상단의 좌표를 입력하시오: ");  
scanf("%d %d", &r.p1.x, &r.p1.y);
```

```
printf("오른쪽 상단의 좌표를 입력하시오: ");  
scanf("%d %d", &r.p2.x, &r.p2.y);
```

```
w = r.p2.x - r.p1.x;  
h = r.p2.y - r.p1.y;
```

```
area = w * h;  
peri = 2 * w + 2 * h;  
printf("면적은 %d이고 둘레는 %d입니다.\n", area, peri);
```

```
return 0;
```

```
}
```



왼쪽 상단의 좌표를 입력하시오: 1 1
오른쪽 상단의 좌표를 입력하시오: 6 6
면적은 25이고 둘레는 20입니다.



구조체 변수의 대입과 비교

23

- 같은 구조체 변수끼리 대입은 가능하지만 비교는 불가능하다.

```
struct point {  
    int x;  
    int y;  
};  
  
int main(void)  
{  
    struct point p1 = {10, 20};  
    struct point p2 = {30, 40};  
  
    p2 = p1;                                // 대입 가능  
  
    if( p1 == p2 )                          // 비교 -> 컴파일 오류!!  
        printf("p1와 p2이 같습니다.");  
  
    if( (p1.x == p2.x) && (p1.y == p2.y) )  // 올바른 비교  
        printf("p1와 p2이 같습니다.");  
}
```



중간 점검

24

1. 구조체의 변수끼리 허용되는 연산에는 어떤 것들이 있는가?
2. 구조체 태그와 구조체 변수의 차이점은 무엇인가?
3. 구조체 멤버로 구조체를 넣을 수 있는가?
4. 구조체는 배열을 멤버로 가질 수 있는가?

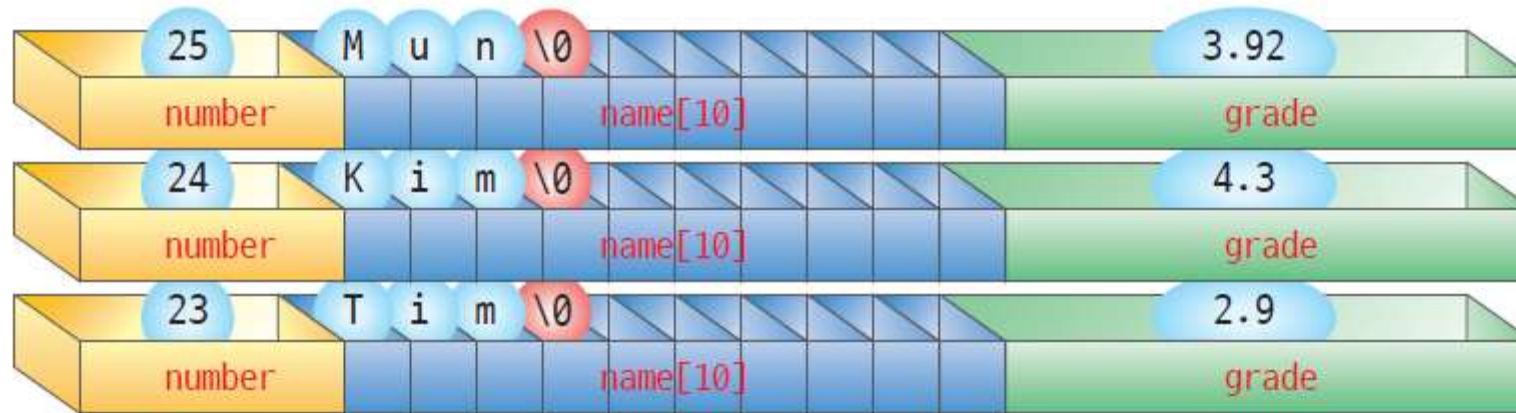




구조체 배열

25

- 구조체를 여러 개 모은 것





구조체 배열

26

```
struct student {  
    int number;  
    char name[20];  
    double grade;  
};  
  
int main(void)  
{  
    struct student list[100];    // 구조체의 배열 선언  
  
    list[2].number = 24;  
    strcpy(list[2].name, "홍길동");  
    list[2].grade = 4.3;  
}
```



구조체 배열의 초기화

27

```
struct student list[3] = {  
    { 1, "Park", 3.42 },  
    { 2, "Kim", 4.31 },  
    { 3, "Lee", 2.98 }  
};
```



예제

28

array_of_struct.c

```
#define SIZE 3

struct student {
    int number;
    char name[20];
    double grade;
};

int main(void)
{
    struct student list[SIZE];
    int i;

    for(i = 0; i < SIZE; i++)
    {
        printf("학번을 입력하시오: ");
        scanf("%d", &list[i].number);
        printf("이름을 입력하시오: ");
        scanf("%s", list[i].name);
        printf("학점을 입력하시오(실수): ");
        scanf("%lf", &list[i].grade);
    }

    for(i = 0; i < SIZE; i++)
        printf("학번: %d, 이름: %s, 학점: %f\n", list[i].number, list[i].name, list[i].grade);
    return 0;
}
```

학번을 입력하시오: 20190001
이름을 입력하시오: 홍길동
학점을 입력하시오(실수): 4.3
학번을 입력하시오: 20190002
이름을 입력하시오: 김유신
학점을 입력하시오(실수): 3.92
학번을 입력하시오: 20190003
이름을 입력하시오: 이성계
학점을 입력하시오(실수): 2.87
이름: 홍길동, 학점: 4.300000
이름: 김유신, 학점: 3.920000
이름: 이성계, 학점: 2.870000



중간 점검

29

1. 상품 5개의 정보를 저장할 수 있는 구조체의 배열을 정의해보라. 상품은 번호와 이름, 가격을 멤버로 가진다.

```
struct item {  
    int num;  
    char name[20];  
    int price;  
};  
  
struct item data[5];
```

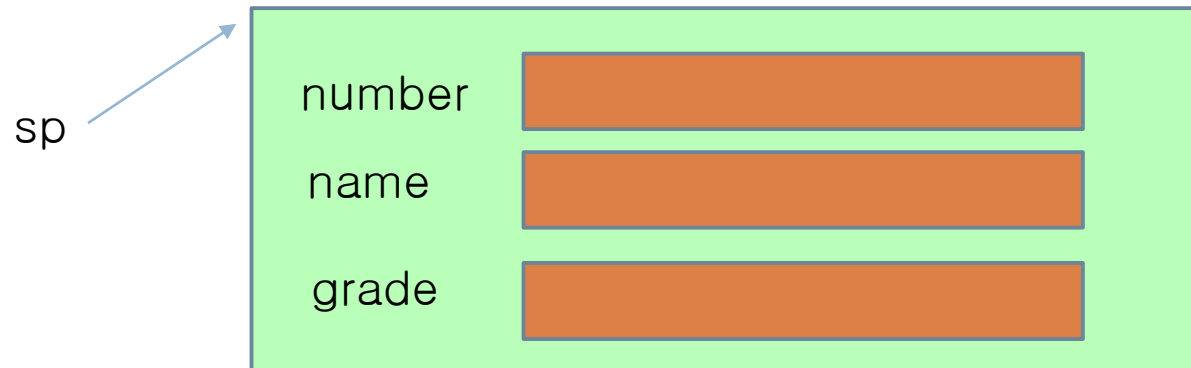




구조체와 포인터

30

1. 구조체를 가리키는 포인터
2. 포인터를 멤버로 가지는 구조체

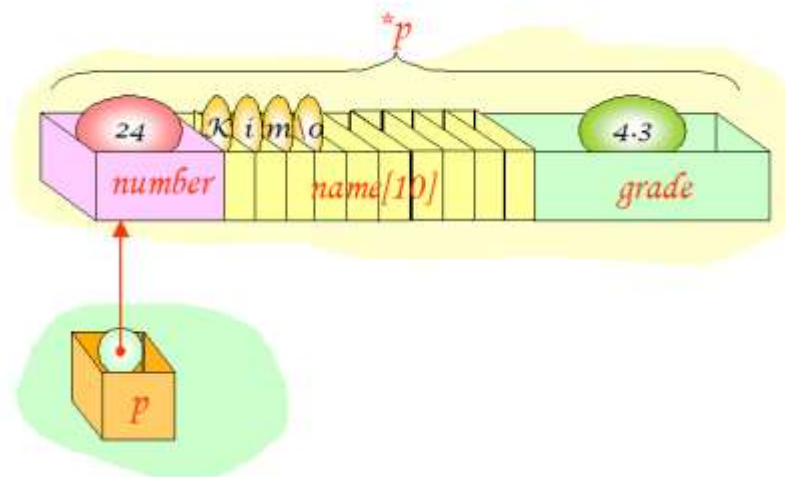




구조체를 가리키는 포인터

31

```
struct student *p;  
  
struct student s = { 24, "Kim", 4.3 };  
struct student *p;  
  
p = &s;  
  
printf("학번=%d 이름=%s 학점=%f \n", s.number, s.name, s.grade);  
printf("학번=%d 이름=%s 학점=%f \n", (*p).number, (*p).name, (*p).grade);
```





-> 연산자

32

- -> 연산자는 구조체 포인터로 구조체 멤버를 참조할 때 사용

```
struct student *p;
```

```
struct student s = { 24, "Kim", 4.3 };
```

```
struct student *p;
```

```
p = &s;
```

```
printf("학번=%d 이름=%s 키=%f \n", p->number, p->name, p->grade);
```




-> 연산자

33

p가 가리키는 구조체 변수

`(*p).number`

p가 가리키는 구조체 변수의 멤버 number

`p->number`

p가 가리키는 구조체 변수의 멤버 number



예제

pointer_to_st.c

34

// 포인터를 통한 구조체 참조

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

```
struct student {  
    int number;  
    char name[20];  
    double grade;  
};
```

```
int main(void)
```

```
{
```

```
    struct student s = { 20070001, "홍길동", 4.3 };
```

```
    struct student *p;
```

```
    p = &s;
```

```
    printf("학번=%d 이름=%s 학점=%f \n", s.number, s.name, s.grade);
```

```
    printf("학번=%d 이름=%s 학점=%f \n", (*p).number, (*p).name, (*p).grade);
```

```
    printf("학번=%d 이름=%s 학점=%f \n", p->number, p->name, p->grade);
```

```
    return 0;
```

```
}
```

학번=20190001 이름=홍길동 학점=4.300000

학번=20190001 이름=홍길동 학점=4.300000

학번=20190001 이름=홍길동 학점=4.300000




포인터를 멤버로 가지는 구조체

st_pointer.c

35

```
struct date {  
    int month;  
    int day;  
    int year;  
};
```

```
struct student {  
    int number;  
    char name[20];  
    double grade;  
    struct date *dob;  
};
```





포인터를 멤버로 가지는 구조체

36

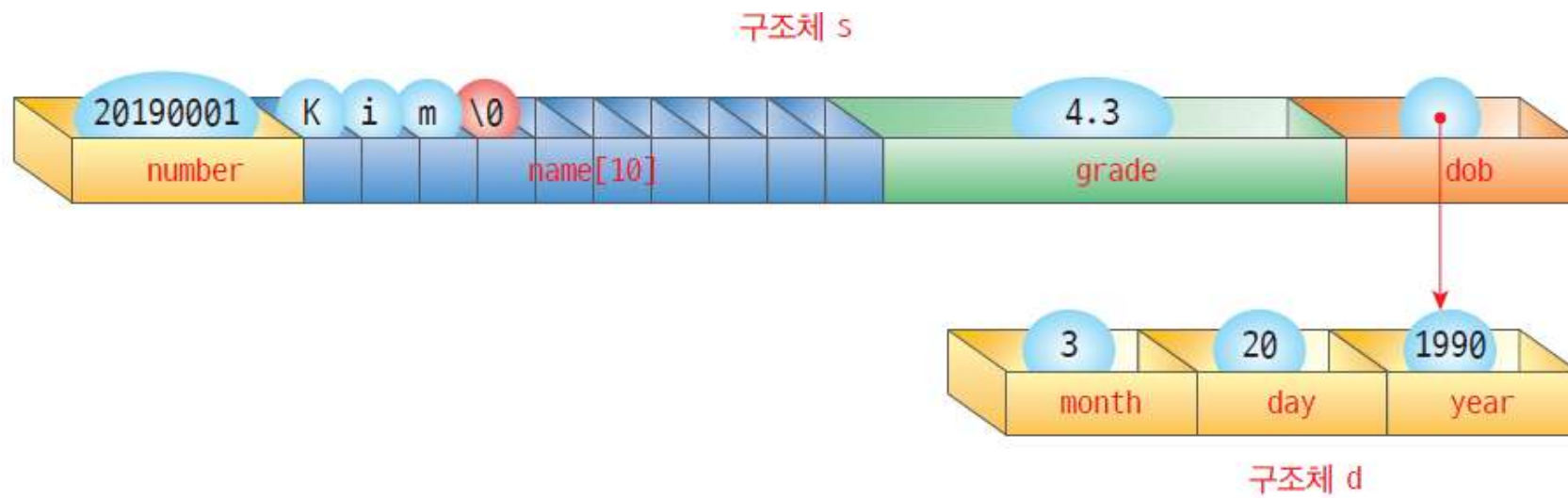
```
int main(void)
{
    struct date d = { 3, 20, 1980 };
    struct student s = { 20190001, "Kim", 4.3 };

    s.dob = &d;

    printf("학번: %d\n", s.number);
    printf("이름: %s\n", s.name);
    printf("학점: %f\n", s.grade);
    printf("생년월일: %d년 %d월 %d일\n", s.dob->year, s.dob->month, s.dob->day);

    return 0;
}
```

학번: 20190001
이름: Kim
학점: 4.300000
생년월일: 1990년 3월 20일





구조체와 함수

38

- 구조체를 함수의 인수로 전달하는 경우
 - ▣ 구조체의 복사본이 함수로 전달되게 된다.
 - ▣ 만약 구조체의 크기가 크면 그만큼 시간과 메모리가 소요된다.

구조체의 경우, 복사된다.

```
int equal(struct student s1, struct student s2)
{
    if( s1.number == s2.number )
        return 1;
    else
        return 0;
}
```

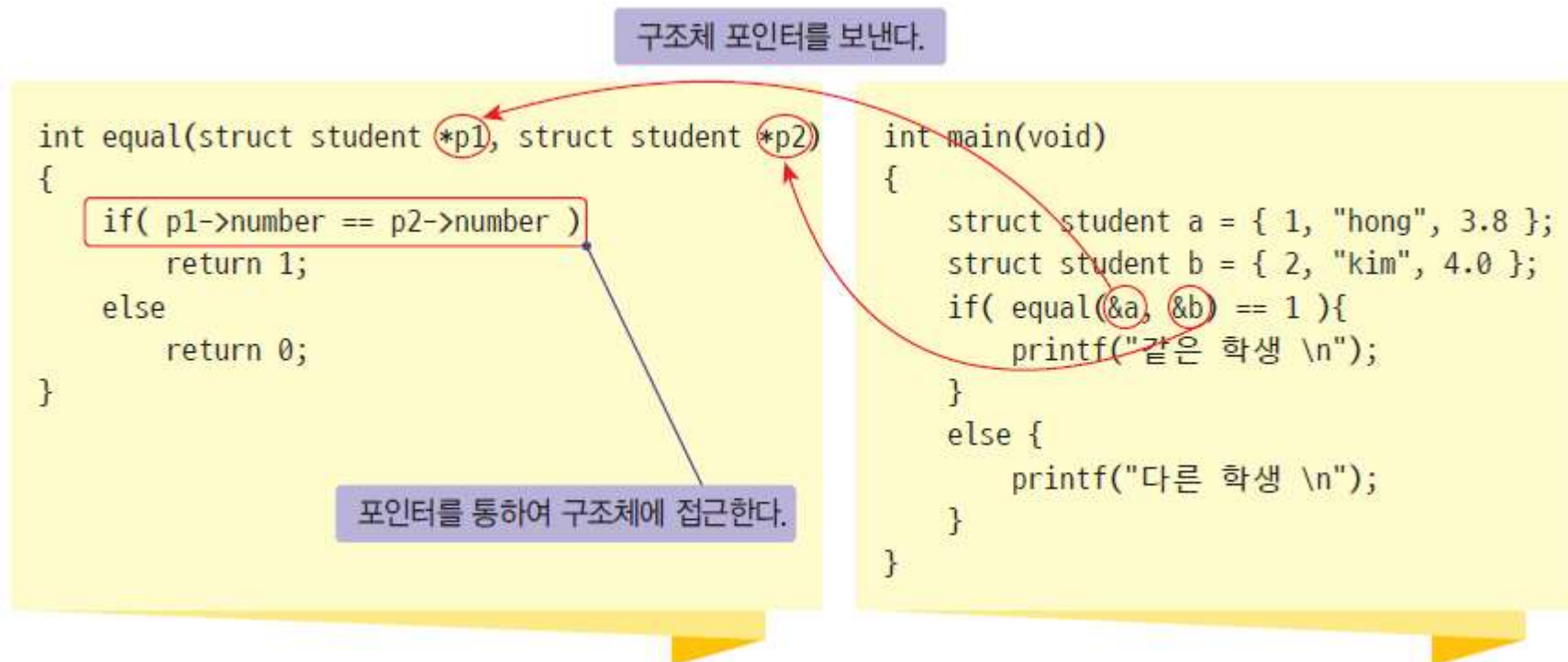
```
int main(void)
{
    struct student a = { 1, "hong", 3.8 };
    struct student b = { 2, "kim", 4.0 };
    if( equal(a, b) == 1 ){
        printf("같은 학생 \n");
    }
    else {
        printf("다른 학생 \n");
    }
}
```



구조체와 함수

39

- 구조체의 포인터를 함수의 인수로 전달하는 경우
 - ▣ 시간과 공간을 절약할 수 있다.
 - ▣ 원본 훼손의 가능성이 있다.





구조체를 반환하는 경우

40

- 복사본이 반환된다.

```
struct student create()
{
    struct student s;
    s.number = 3;
    strcpy(s.name, "park");
    s.grade = 4.0;
    return s;
}
```

구조체 s가 구조체 a로 복사된다.

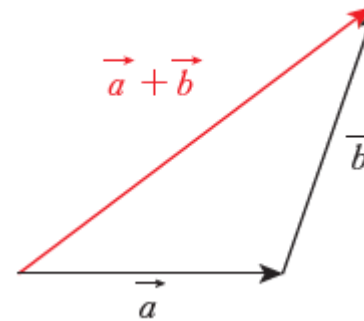
```
int main(void)
{
    struct student a;
    a = create();
    return 0;
}
```




lab: 벡터 연산

41

- 두 벡터의 합을 구하는 함수 `get_vector_sum()`를 제작하여 보자. 이 함수는 두개의 벡터를 인수로 받아서 덧셈을 하고 덧셈의 결과로 생성된 벡터를 반환한다.



벡터의 합은 (7.000000, 9.000000)입니다.



예제

vector.c

42

```
#include <stdio.h>

struct vector {
    float x;
    float y;
};

struct vector get_vector_sum(struct vector a, struct vector b);

int main(void)
{
    struct vector a = { 2.0, 3.0 };
    struct vector b = { 5.0, 6.0 };
    struct vector sum;

    sum = get_vector_sum(a, b);
    printf("벡터의 합은 (%f, %f)입니다.\n", sum.x, sum.y);

    return 0;
}
```



예제

43

```
struct vector get_vector_sum(struct vector a, struct vector b)
{
    struct vector result;

    result.x = a.x + b.x;
    result.y = a.y + b.y;

    return result;
}
```

벡터의 합은 (7.000000, 9.000000)입니다.

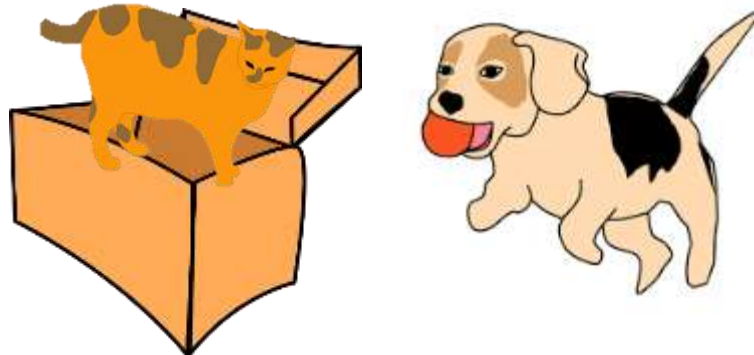


공용체

44

- 공용체(union)
 - ▣ 같은 메모리 영역을 여러 개의 변수가 공유
 - ▣ 공용체를 선언하고 사용하는 방법은 구조체와 아주 비슷

```
union example {  
    char c;        // 같은 공간 공유  
    int i;         // 같은 공간 공유  
};
```





예제

45

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

```
union example {  
    int i;  
    char c;  
};
```

공용체 선언

```
int main(void)  
{
```

```
    union example v;
```

공용체 변수 선언.

char 형으로 참조.

```
    v.c = 'A';
```

```
    printf("v.c:%c    v.i:%i\n", v.c, v.i );
```

```
    v.i = 10000;
```

int 형으로 참조.

```
    printf("v.c:%c    v.i:%i\n", v.c, v.i);
```

```
}
```

```
v.c:A v.i:-858993599
```

```
v.c:† v.i:10000
```



공용체에 타입 필드 사용

46

```
#include <stdio.h>
#include <string.h>
#define STU_NUMBER 1
#define REG_NUMBER 2

struct student {
    int type;
    union {
        int stu_number;           // 학번
        char reg_number[15];      // 주민등록번호
    } id;
    char name[20];
};
```



공용체에 타입 필드 사용

47

```
void print(struct student s)
{
    switch(s.type)
    {
        case STU_NUMBER:
            printf("학번 %d\n", s.id.stu_number);
            printf("이름: %s\n", s.name);
            break;
        case REG_NUMBER:
            printf("주민등록번호: %s\n", s.id.reg_number);
            printf("이름: %s\n", s.name);
            break;
        default:
            printf("타입오류\n");
            break;
    }
}
```



공용체에 타입 필드 사용

48

학번: 20190001

이름: 홍길동

주민등록번호: 860101-1056076

이름: 김철수

```
int main(void)
{
    struct student s1, s2;

    s1.type = STU_NUMBER;
    s1.id.stu_number = 20190001;
    strcpy(s1.name, "홍길동");

    s2.type = REG_NUMBER;
    strcpy(s2.id.reg_number, "860101-1056076");
    strcpy(s2.name, "김철수");

    print(s1);
    print(s2);
}
```




중간 점검

49

1. 공용체의 선언에 사용하는 키워드는 _____이다.
2. 공용체에 할당되는 메모리의 크기는 어떻게 결정되는가?

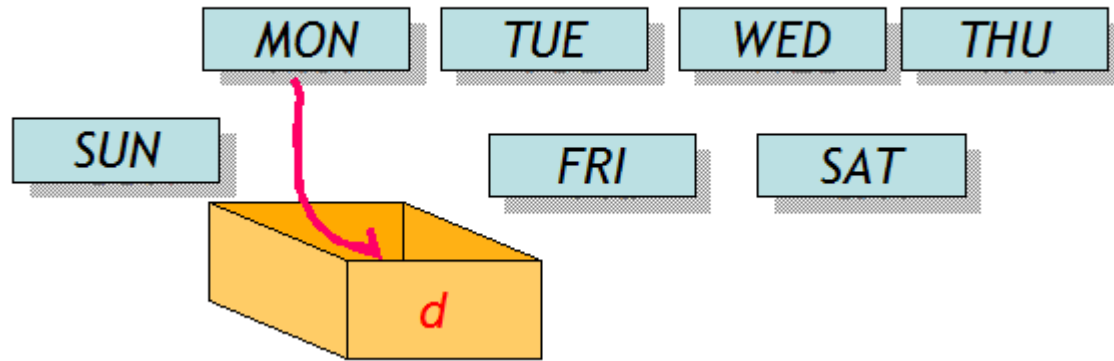




열거형

50

- 열거형(*enumeration*)이란 변수가 가질 수 있는 값들을 미리 열거해놓은 자료형
- (예) 요일을 저장하고 있는 변수는 { 일요일, 월요일, 화요일, 수요일, 목요일, 금요일, 토요일 } 중의 하나의 값만 가질 수 있다.





열거형의 선언

51

Syntax: 열거형 정의

예

```
enum days { SUN, MON, TUE, WED, THU, FRI, SAT };
```

열거형을 정의할 때
사용하는 키워드

열거형의 이름(태그)

열거형은 기호상수들이
모여 있는 자료형이다.

열거형 변수 선언

```
enum days today;  
today = SUN;    // OK!
```



열거형이 필요한 이유

52

- 다음과 같이 프로그램을 작성할 수 있다.
 - ▣ `int today;`
 - ▣ `today = 0; // 일요일`
 - ▣ `today = 1; // 월요일`
- 오류를 줄이고 가독성을 높여야 된다.
- 0보다는 **SUN**라는 기호상수가 더 바람직하다. 의미를 쉽게 알 수 있기 때문이다.
- `today`에 9와 같은 의미없는 값이 대입되지 않도록 미리 차단하는 것도 필요하다.



열거형 초기화

53

- 값을 지정하지 않으면 0부터 할당

```
enum days { SUN, MON, TUE, WED, THU, FRI, SAT }; // SUN=0, MON=1, ...  
enum days { SUN=1, MON, TUE, WED, THU, FRI, SAT }; // SUN=1, MON=2, ...  
enum days { SUN=7, MON=1, TUE, WED, THU, FRI, SAT=6 }; // SUN=7, MON=1, ...
```



열거형의 예

54

```
enum colors { white, red, blue, green, black };  
enum boolean { false, true };  
enum levels { low, medium, high };  
enum car_types { sedan, suv, sports_car, van, pickup, convertible };
```



예제

55

```
#include <stdio.h>
enum days { SUN, MON, TUE, WED, THU, FRI, SAT };
char *days_name[] = {
    "sunday", "monday", "tuesday", "wednesday", "thursday", "friday",
    "saturday" };

int main(void)
{
    enum days d;
    d = WED;
    printf("%d번째 요일은 %s입니다\n", d, days_name[d]);
    return 0;
}
```

3번째 요일은 wednesday입니다



열거형과 다른 방법과의 비교

56

| 정수 사용 | 기호 상수 | 열거형 |
|--|---|--|
| <pre>switch(code) { case 1: printf("LCD TV\n"); break; case 2: printf("OLED TV\n"); break; }</pre> | <pre>#define LCD 1 #define OLED 2 switch(code) { case LCD: printf("LCD TV\n"); break; case OLED: printf("OLED TV\n"); break; }</pre> | <pre>enum tvtype { LCD, OLED }; enum tvtype code; switch(code) { case LCD: printf("LCD TV\n"); break; case PDP: printf("OLED TV\n"); break; }</pre> |
| 컴퓨터는 알기 쉬우나 사람은 기억하기 어렵다. | 기호 상수를 작성할 때 오류를 저지를 수 있다. | 컴파일러가 중복이 일어나지 않도록 체크한다. |



중간 점검

57

1. 열거형의 선언에 사용하는 키워드는 _____이다.
2. 열거형은 어떤 경우에 사용되는가?
3. 열거형에서 특별히 값을 지정하지 않으면 자동으로 정수상수값이 할당되는가?

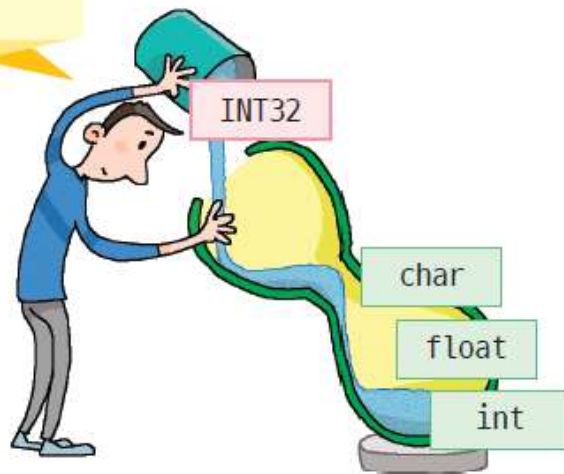




typedef의 개념

58

typedef은 기본 자료형에 새로운 자료형을 추가하는 것입니다.



지금부터 **INT32**이라는 새로운 타입을 사용할 수 있음을 알린다.





typedef

59

Syntax: typedef 정의

예 typedef unsigned char BYTE;

기존의 자료형인 unsigned char를 새로운 자료형 BYTE로 정의하는 것이다.

기존의 자료형

새로운 자료형



typedef의 예

60

```
typedef  unsigned char  BYTE;  
BYTE index;             // unsigned int index;와 같다.  
  
typedef  int            INT32;  
typedef  unsigned int   UINT32;  
  
INT32 i;                 // int i;와 같다.  
UINT32 k;               // unsigned int k;와 같다.
```



구조체로 새로운 타입 정의

61

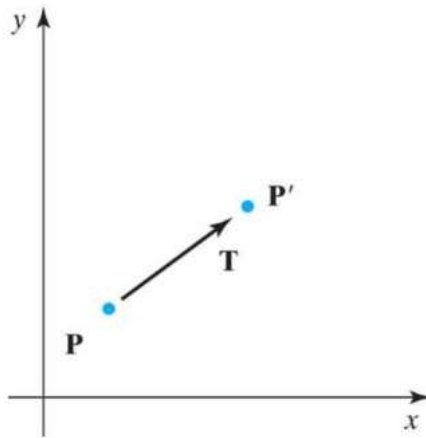
- 구조체로 새로운 타입을 정의할 수 있다.

```
struct point {  
    int x;  
    int y;  
};  
typedef struct point POINT;  
POINT a, b;
```



lab: 2차원 공간 상의 점을 POINT 타입으로 정의

- 2차원 공간에서의 점을 구조체로 표현한 다음에 이 구조체를 typedef을 이용하여 새로운 타입인 **POINT**로 정의한다



새로운 점의 좌표는(12, 13)입니다.



예제

typedef.c

63

```
#include <stdio.h>

typedef struct point {
    int x;
    int y;
} POINT;

POINT translate(POINT p, POINT delta);

int main(void)
{
    POINT p = { 2, 3 };
    POINT delta = { 10, 10 };
    POINT result;

    result = translate(p, delta);
    printf("새로운 점의 좌표는(%d, %d)입니다.\n", result.x, result.y);

    return 0;
}
```



예제

64

```
POINT translate(POINT p, POINT delta)
{
    POINT new_p;

    new_p.x = p.x + delta.x;
    new_p.y = p.y + delta.y;

    return new_p;
}
```

새로운 점의 좌표는 (12, 13)입니다.



typedef과 #define 비교

65

- 이식성을 높여준다.
 - ▣ 코드를 컴퓨터 하드웨어에 독립적으로 만들 수 있다
 - ▣ (예) `int`형은 2바이트이기도 하고 4바이트, `int`형 대신에 `typedef`을 이용한 `INT32`나 `INT16`을 사용하게 되면 확실하게 2바이트인지 4바이트인지를 지정할 수 있다.
- `#define`을 이용해도 `typedef`과 비슷한 효과를 낼 수 있다. 즉 다음과 같이 `INT32`를 정의할 수 있다.
 - ▣ `#define UINT32 unsigned int`
 - ▣ `typedef float VECTOR[2];` // `#define`으로는 불가능하다.
- 문서화의 역할도 한다.
 - ▣ `typedef`을 사용하게 되면 주석을 붙이는 것과 같은 효과



중간 점검

66

1. typedef의 용도는 무엇인가?
2. typedef의 장점은 무엇인가?
3. 사원을 나타내는 구조체를 정의하고 이것을 typedef을 사용하여 **employee**라는 새로운 타입으로 정의하여 보자.





mini project: 평점이 높은 학생 찾기

- 어느 학교나 학기가 끝나면 학과 내에서 가장 평점이 높은 학생을 선발하여서 장학금을 수여한다. 가장 평점이 높은 학생을 찾아서 학생의 이름과 학번, 평점을 화면에 출력하는 프로그램을 작성하여 보자.





```
#include <stdio.h>

struct student {
    int number;
    char name[20];
    double grade;
};

struct student list[] = {
    { 20120001, "홍길동, 4.2 },
    { 20120002, "김철수, 3.2 },
    { 20120002, "김영희, 3.9 }

};
```



```
int main(void)
{
    struct student super_stu;
    int i, size;

    size = sizeof(list)/sizeof(list[0]);
    super_stu = list[0];

    for(i=1; i< size; i++) {
        if( list[i].grade > super_stu.grade )
            super_stu = list[i];
    }
    printf("평점이 가장 높은 학생은 (이름%s, 학번%d, 평점%f)입니다\n",
           super_stu.name, super_stu.number, super_stu.grade);
}
```



도전문제

70

- 학생들에 대한 정보를 사용자로부터 받게끔 프로그램을 수정하라.
- 최대 평점의 학생을 찾는 부분을 함수 `get_max_stu()`로 독립시켜서 전체 프로그램을 다시 작성하여 보자.





Q & A

71

