

# 구조체와 공용체

제주대학교 컴퓨터공학과

변영철 교수

(ycb@jejunu.ac.kr)

# 이 장을 공부하면

- 코드를 묶어 추상화하듯이 변수들을 묶어 추상화할 수 있다.
- 구조체와 공용체를 이해할 수 있다.
- 열거형을 이해할 수 있다.
- 자료형 이름을 간단히 만들 수 있다.

# 추상화



- 코드를 묶어서 간단히 표현
  - 코드 추상화
  - 그 결과 함수가 만들어짐
- 가장 간단한 C 프로그램을 작성한 후 printf 문장을 추상화해 보자.

# 추상화



학번과 학점을 저장하는  
변수를 만들고 여기에 값을  
저장한 후 출력해보자.

```
#include <stdio.h>

void main() {
    int num;
    double grade;

    num = 2;
    grade = 2.7;

    printf("학번 : %d\n", num);
    printf("학점 : %.1lf\n", grade);

    getchar();

}
```

# 추상화



학번과 학점을 하나씩 더 추가  
하려면?

```
void main() {  
    int num;  
    double grade;  
    int num2;  
    double grade2;  
  
    num = 2;  
    grade = 2.7;  
    num2 = 3;  
    grade2 = 3.7;  
  
    printf("학번 : %d\n", num2);  
    printf("학점 : %.1lf\n", grade2);  
  
    getchar();  
}
```

# 추상화



- 관련된 변수 2개를 묶어서 간단히 표현할 수 있을까?
- 구조체 자료형

실행  
결과 학번 : 2  
학점 : 2.7

```
struct student
{
    int num;
    double grade;
};
```

// 구조체 선언

// int형 멤버

// double형 멤버

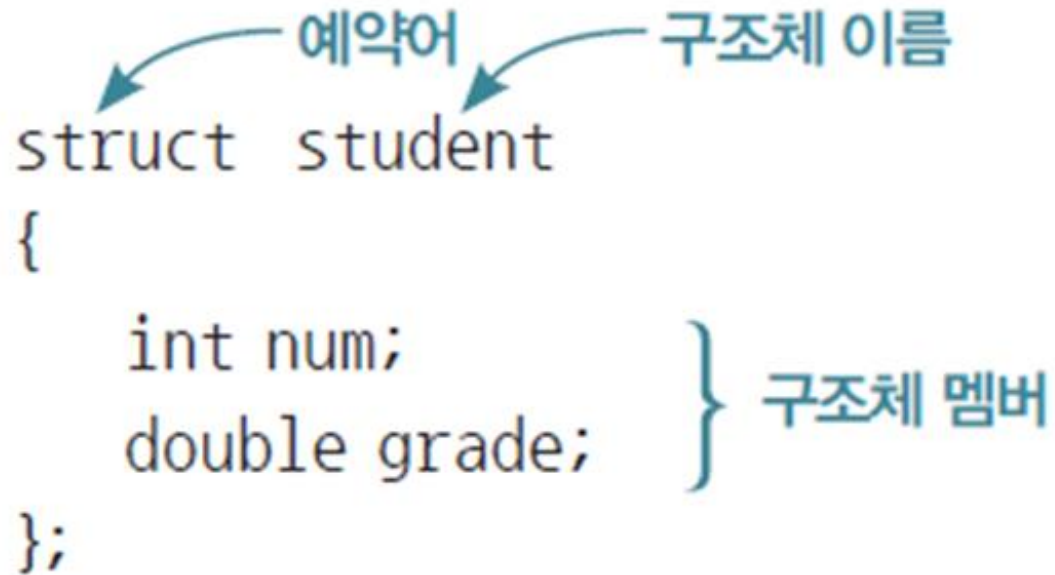
// 세미콜론 사용

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

    s1.num = 2;
    s1.grade = 2.7;
    printf("학번 : %d\n", s1.num);
    printf("학점 : %.1lf\n", s1.grade);

    return 0;
}
```

# 구조체 자료형 선언



The diagram shows a C struct declaration with annotations in Korean. The code is: `struct student { int num; double grade; };`. An arrow labeled '예약어' (keyword) points to 'struct'. Another arrow labeled '구조체 이름' (struct name) points to 'student'. A large closing brace '}' is labeled '구조체 멤버' (struct member), indicating the block of code between the opening and closing braces.

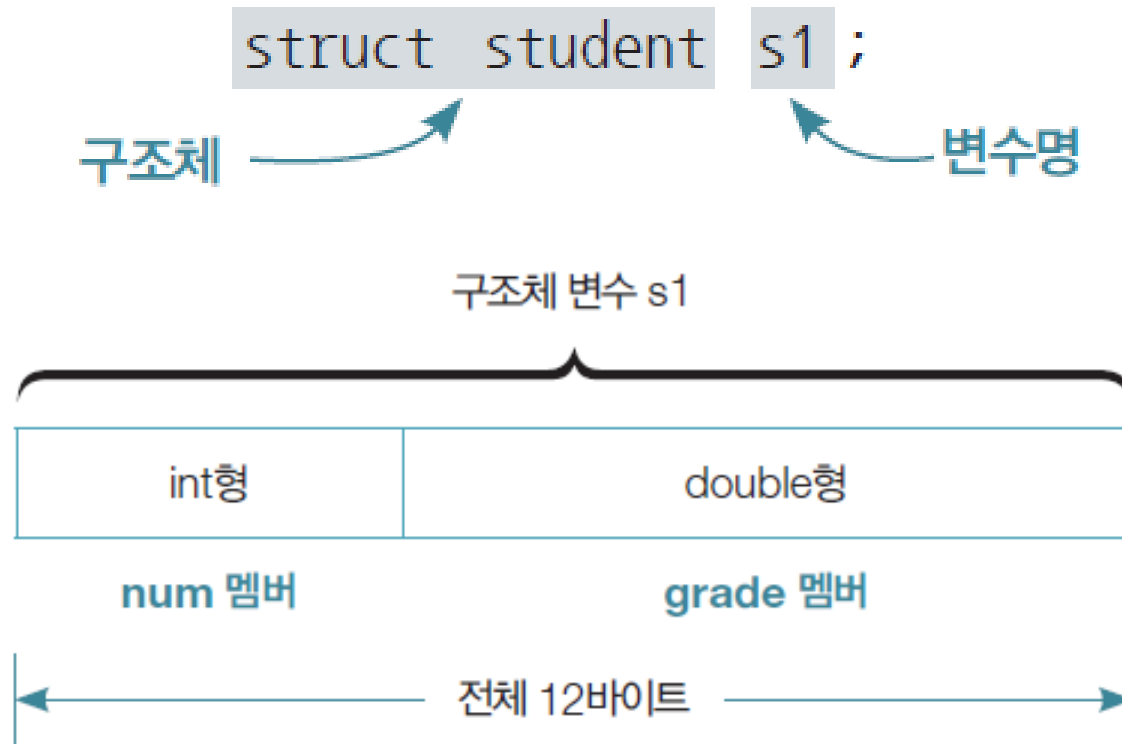
```
struct student
{
    int num;
    double grade;
};
```

예약어

구조체 이름

구조체 멤버

# 구조체 변수 정의





# 구조체 변수 접근

The diagram illustrates the syntax for accessing a member of a struct variable. The code snippet is `s1 . num = 2 ;`. Three labels with arrows point to specific parts of the code:   
- **구조체 변수명** (Struct variable name) points to `s1`.   
- **멤버명** (Member name) points to `num`.   
- **멤버접근 연산자** (Member access operator) points to the dot `.` between `s1` and `num`.

# 구조체 변수 접근

```
struct student {  
    int num;  
    double grade;  
};  
  
void main() {  
    struct student s1;  
    s1.num = 2;  
    s1.grade = 2.7;  
  
    printf("학번 : %d\n", s1.num);  
    printf("학점 : %.1lf\n", s1.grade);  
    getchar();  
}
```

# 이름 추가

```
#include <stdio.h>
#include <string.h>
struct student {
    char name[20];
    int num;
    double grade;
};
void main() {
    struct student s1;
    strcpy(s1.name, "Gildong Hong");
    s1.num = 2;
    s1.grade = 2.7;

    printf("이름 : %s\n", s1.name);
    printf("학번 : %d\n", s1.num);
    printf("학점 : %.1lf\n", s1.grade);
    getchar();
}
```

# 이름 추가

name

age

grade

char [20]

int

double

# 구조체 변수 초기화

```
#include <stdio.h>
#include <string.h>
struct student {
    char name[20];
    int num;
    double grade;
};
void main() {
    struct student s1;
    s1 = { "Gildong Hong", 2, 2.7 };

    printf("이름 : %s\n", s1.name);
    printf("학번 : %d\n", s1.num);
    printf("학점 : %.1lf\n", s1.grade);
    getchar();
}
```

# 구조체 배열

```
#include <stdio.h>
#include <string.h>
struct student {
    char name[20];
    int num;
    double grade;
};
void main() {
    struct student s[100];
    s[0] = { "Gildong Hong", 2, 2.7 };
    s[1] = { "Cheolsu Kim", 3, 3.7 };

    printf("이름 : %s\n", s[0].name);
    printf("학번 : %d\n", s[0].num);
    printf("학점 : %.1lf\n", s[0].grade);
    getchar();
}
```

# 배열 초기화

```
#include <stdio.h>
#include <string.h>
struct student {
    char name[20];
    int num;
    double grade;
};
void main() {
    struct student s[2] = {
        { "Gildong Hong", 2, 2.7},
        { "Cheolsu Kim", 3, 3.7}
    };

    printf("이름 : %s\n", s[0].name);
    printf("학번 : %d\n", s[0].num);
    printf("학점 : %.1lf\n", s[0].grade);
    getchar();
}
```

# 파라미터



**printf 문장들을 Print라  
는 함수로 코드 추상화  
해보자.**

```
void ShowInfo(struct student hakseng) {  
    printf("이름 : %s\n", hakseng.name);  
    printf("학번 : %d\n", hakseng.num);  
    printf("학점 : %.1lf\n", hakseng.grade);  
}  
  
void main() {  
    struct student s[2] = {  
        { "Gildong Hong", 2, 2.7},  
        { "Cheolsu Kim", 3, 3.7}  
    };  
    ShowInfo(s[0]);  
    ShowInfo(s[1]);  
    getchar();  
}
```

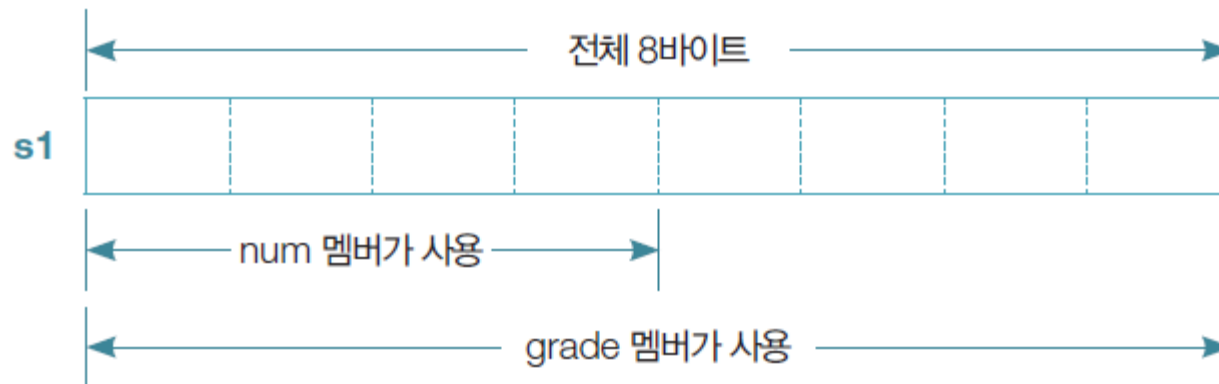


# 공용체

```
union student
{
    int num;
    double grade;
};
```

예약어      공용체 이름

공용체 멤버



# 열거형



**열거형을 쓰지 않을 경우 어떤 불편한 점이 있을까?**

**숫자를 쓸까, 기호를 쓸까?**

```
enum season {  
    SPRING, SUMMER, FALL, WINTER  
};
```

```
void main() {  
    int now = FALL;  
    switch (now) {  
        case SPRING:  
            printf("아름다운 벚꽃!\n");  
            break;  
        case SUMMER:  
            printf("시원한 바다!\n");  
            break;  
        case FALL:  
            printf("아름다운 단풍!\n");  
            break;  
        case WINTER:  
            printf("하얀 눈사람!\n");  
            break;  
    }  
}
```

# typedef 문



자료형 이름을 간단히  
만들어 변수 만들 때  
편함.

```
#include <stdio.h>
#include <string.h>
struct student {
    char name[20];
    int num;
    double grade;
};
typedef struct student Student;
void main() {
    Student s[2] = {
        { "Gildong Hong", 2, 2.7},
        { "Cheolsu Kim", 3, 3.7}
    };

    getchar();
}
```

# typedef 문

```
#include <stdio.h>
#include <string.h>
typedef struct {
    char name[20];
    int num;
    double grade;
} Student;

void main() {
    Student s[2] = {
        { "Gildong Hong", 2, 2.7},
        { "Cheolsu Kim", 3, 3.7}
    };

    getchar();
}
```