C Programming (CSE2035)
(Chap11. Structures and Unions)

Sungwon Jung, Ph.D.

Bigdata Processing & DB LAB
Dept. of Computer Science and Engineering
Sogang University
Seoul, Korea
Tel: +82-2-705-8930
Email : jungsung@sogang.ac.kr

Structures and functions

- 구조체 또한 구조체가 아닌 변수와 마찬가지로 함수의 인자로써 전달될 수 있다.

- 기존의 변수가 함수의 인자로써 전달되는 방법
  - 변수의 값이 함수의 인자로 전달되는 방법 (Call by value)
  - 변수의 주소를 함수의 인자로 전달하는 방법 (Call by reference)

```
void CallByValue(int a) {
…
}

void CallByReference(int *a) {
…
}
```

```
main()
{
        int  a;
        CallByValue(a);
        CallByReference(&a);
}
```

Structures and functions

- 구조체 형식의 변수가 함수의 인자로써 전달되는 방법
  - 구조체 변수의 내용이 함수의 인자로 전달되는 방법 (Call by value)
  - 구조체 변수의 주소를 함수의 인자로 전달하는 방법 (Call by reference)
  - 멤버의 내용이 함수의 인자로 전달되는 방법 (Call by value)
  - 멤버의 주소를 함수의 인자로 전달하는 방법 (Call by reference)
  - 구조체 변수를 인자로 전달하려면 호출하는 쪽(caller)이나 피호출자(callee) 모두 구조체 형식을 알아야 한다.

```
typedef struct {
        int i;
        …
}str;

void CallByValue2(str a) {
…
}

void CallByReference2(str *a) {
…
}
```

```
main()
{
        str a;
        CallByValue(a.i);
        CallByReference(&a.i);
        CallByValue2(a);
        CallByReference2(&a);
}
```

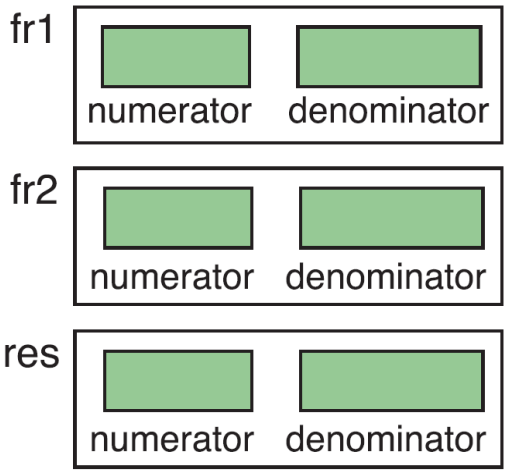
Structures and functions

- 구조체 멤버의 내용이 전달되는 것(Call by value)과 멤버의 주소가 전달되는 것(Call by reference)은 기존의 전달 방식과 같다.

```
…
res.numerator =
    multiply(fr1.numerator, fr2.numerator);
res.denominator =
    multiply(fr1.denominator, fr2.denominator);
…
```

```
// =============== multiply ==============
multiply (int x, int y)
{
    return x * y;
}   // multiply
```

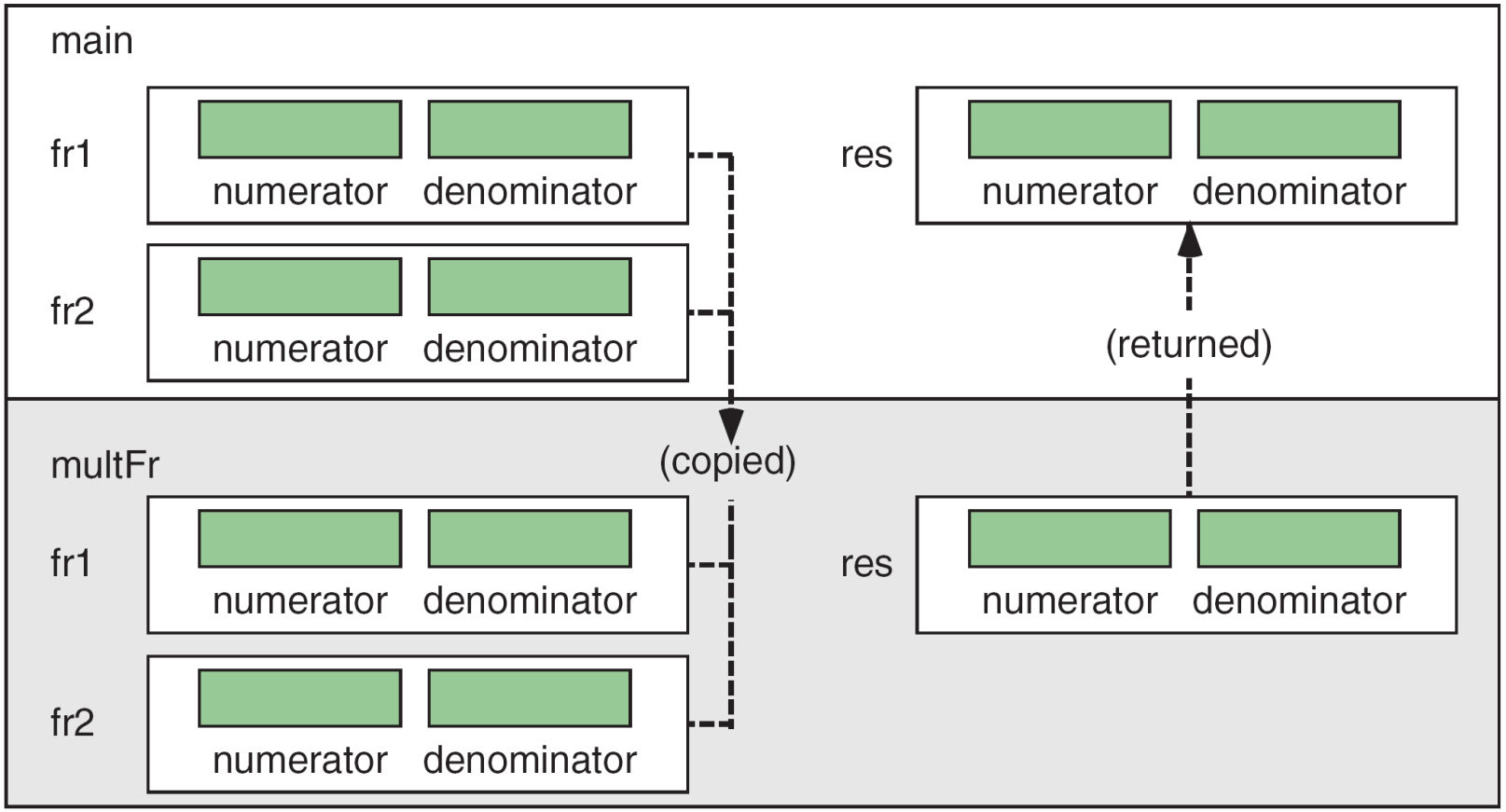
x y




```
typedef struct {
        int numerator;
        int denominator;
}FRACTION;
```

Structures and functions

- 구조체 변수의 내용 전달 또한 멤버의 전달 방식과 같으며, 전달되는 **type**만 구조체 **type**으로 바뀌었을 뿐이다.



Structures and functions

- 구조체 멤버의 내용 을 함수에 전달

```
FRACTION result;
FRACTION fr1;
FRACTION fr2;
result.numerator = multiply(fr1.numerator, fr2.numerator);
result.denominator = multiply(fr1.denominator, fr2.denominator);
```

fr1과 fr2의 멤버의 내용은 int형으로 함수에 전달된다.

- 구조체 변수의 내용을 함수에 전달

```
FRACTION multiply(FRACTION fr1, FRACTION fr2) {
        FRACTION result;
        result.numerator = fr1.numerator * fr2.numerator;
        result.denominator = fr1.denominator * fr2.denominator;
        return result;
}
```

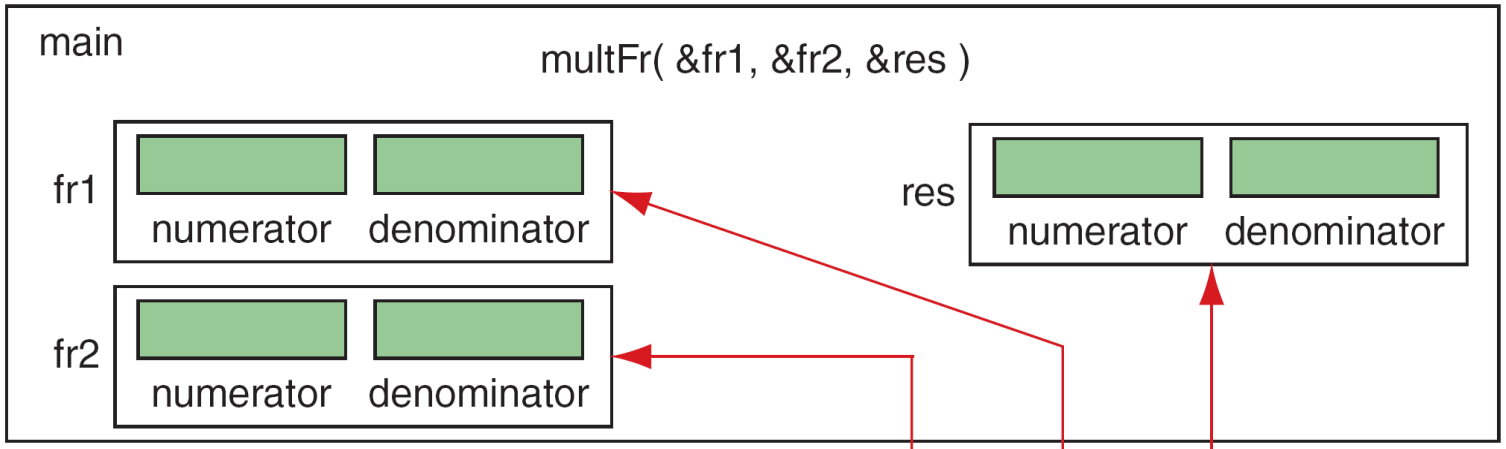
구조체 변수를 전달 받는 multiply함수.

```
FRACTION result;
FRACTION fr1;
FRACTION fr2;
result = multiply(fr1, fr2);
```

구조체 변수인 fr1과 fr2를 통째로 함수에 전달 받는다.

Passing structures through pointers

- 포인터를 통해 구조체 변수를 함수에 전달하는(Call by reference)방법은 구조체 변수의 크기가 매우 클 때 효과적이다.




```
void multFr (FRACTION *pFr1,
             FRACTION *pFr2,
             FRACTION *pRes)
{
   …
   pRes->numerator   = pFr1->numerator   * pFr2->numerator;
   pRes->denominator = pFr1->denominator * pFr2->denominator;
   return;
} // multFr
```

Passing structures through pointers

- 포인터를 통해 구조체 변수를 함수에 전달

```
#include <stdio.h>

// Global Declarations
typedef struct
        {
          int numerator;
          int denominator
        } FRACTION;

// Function Declarations
void getFr    (FRACTION* pFr);
void multFr   (FRACTION* pFr1, FRACTION* pFr2,
               FRACTION* pRes2);
void printFr  (FRACTION* pFr1, FRACTION* pFr2,
               FRACTION* pRes);
```

모든 함수들로부터 구조체가 보이게 하기 위해 전역으로 선언한다.

구조체가 포인터 형으로 전달된다.

Passing structures through pointers

- 구조체 포인터를 이용한 분수(fraction)들의 multiply계산을 구현한 프로그램

```
int   main (void)
{
// Local Declarations
   FRACTION fr1;
   FRACTION fr2;
   FRACTION res;

// Statements
   getFr    (&fr1);
   getFr    (&fr2);
   multFr   (&fr1, &fr2, &res);
   printFr  (&fr1, &fr2, &res);
   return 0;
}  // main
```

선언된 구조체들의 주소를 통한 참조 (Call by reference)

Passing structures through pointers

- **void getFr (FRACTION* pFr)**
  - 두 개의 int형 변수를 user로부터 입력 받아, 분수(fraction)를 pFr에 저장한다.
  - Parameter: pFr은 FRACTION 구조체를 가리키는 포인터

```
void getFr (FRACTION* pFr)
{
// Statements
   printf("Write a fraction in the form of x/y: ");
   scanf ("%d/%d", &pFr->numerator,
           &(*pFr).denominator);
   return;
}  // getFr
```

Passing structures through pointers

- **void multFr (FRACTION* pFr1, FRACTION* pFr2, FRATION* pRes)**
  - pFr1, pFr2 두 개의 분수(fraction)의 곱을 pRes에 저장한다.
  - Parameter: pFr1, pFr2, pRes는 FRACTION 구조체를 가리키는 포인터

```
void multFr  (FRACTION* pFr1, FRACTION* pFr2,
              FRACTION* pRes)
{
// Statements
   pRes->numerator    =
         pFr1->numerator * pFr2->numerator;
   pRes->denominator =
         pFr1->denominator * pFr2->denominator;
   return;
}  // multFr
```

Passing structures through pointers

- **void printFr (FRACTION* pFr1, FRACTION* pFr2, FRATION* pRes)**
  - pFr1, pFr2, pRes 세 개의 분수(fraction)의 값을 출력한다.
  - Parameter: pFr1, pFr2, pRes는 FRACTION 구조체를 가리키는 포인터

```
void printFr (FRACTION* pFr1, FRACTION* pFr2,
              FRACTION* pRes)
{
// Statements
   printf("\nThe result of %d/%d * %d/%d is %d/%d\n",
          pFr1->numerator, pFr1->denominator,
          pFr2->numerator, pFr2->denominator,
          pRes->numerator, pRes->denominator);
   return;
}  // printFr
```

```
Results:
Write a fraction in the form of x/y: 4/3
Write a fraction in the form of x/y: 6/7

The result of 4/3 * 6/7 is 24/21
```

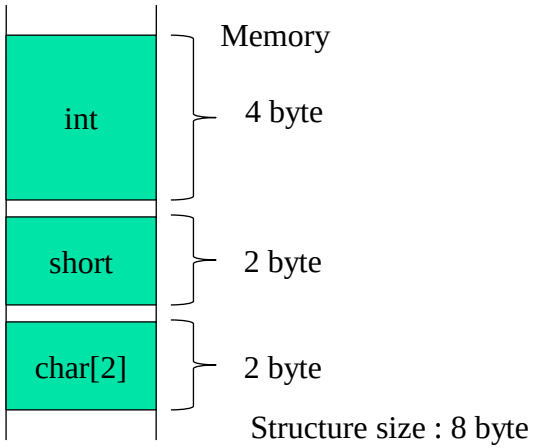
Unions

- 공용체(union)의 기본적인 구조와 사용법은 구조체(structure)와 동일하다.
- 공용체 변수는 구조체 변수와 달리 **struct**라는 키워드 대신 **union**이라는 키워드를 쓴다.
- 구조체 변수는 구성 멤버들에게 개별적인 메모리 공간을 할당한다.
- 공용체 변수는 크기가 가장 큰 한 멤버에게만 메모리 공간만을 할당하고 이 공간을 공용체 내의 모든 멤버들이 공유해서 사용한다.

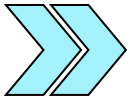
Unions

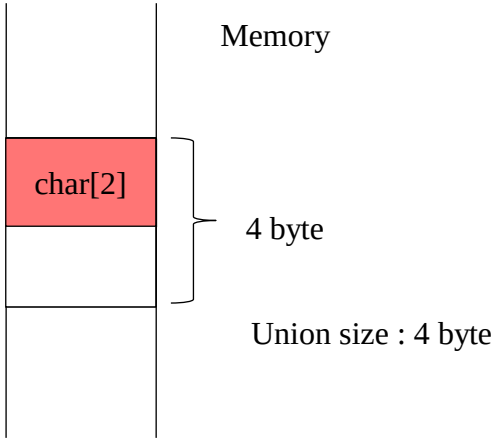
- 구조체와 공용체의 메모리 할당

```
struct {
        int I;
        short S;
        char C[2];
}s;
```




```
union {
        int I;
        short S;
        char C[2];
}u;
```





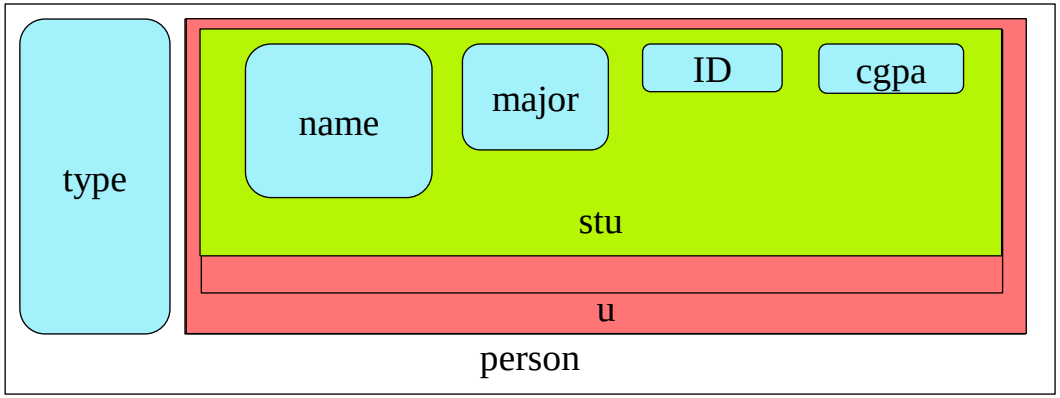
Unions

- **nested structure**와 비슷하게, 구조체가 공용체를 멤버로 갖는 것이 가능하고 공용체 또한 구조체를 멤버로 갖는 것이 가능하다.

```
typedef struct {
        char    name[20];
        char    mail[20];
        int     mobile;
}professor;

typedef struct {
        char    name[20];
        char    major[10];
        int     ID;
        float   cgpa;
}student;
```

```
typedef struct {
char    type;
        union {
                professor prof;
                student   stu;
        } u;
}person;
```



Unions

- 구조체 **person**은 학교에 등록된 사람을 저장하기 위한 데이터 타입으로써, 학생이나 교수를 저장할 수 있다.

- **person**의 멤버인 **u**는 학생/교수의 정보를 저장하기 위한 공용체로써 교수정보에는 이름, 메일주소, 이동전화 번호를 저장하고 학생정보에는 이름, 전공, **ID, CGPA**를 저장한다.

- **u**의 멤버 **prof**와 **stu**는 같은 메모리 공간을 공유해서 사용하므로 **person**이 가리키는 누군가는 학생이거나 교수여야 한다.

- 구조체 **person**은 저장된 데이터가 교수의 것인지 학생의 것인지 구분하기 위해 **char** 타입의 변수 **type**을 둔다. 저장된 데이터가 교수의 것인 경우 '**p**'를, 학생의 것인 경우 '**s**'를 저장한다.

Unions

- 예제 프로그램 – **nested structure, union**

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

typedef struct {
        char    name[20];
        char    mail[20];
        int     mobile;
}professor;

typedef struct {
        char    name[20];
        char    major[10];
        int     ID;
        float   cgpa;
}student;

typedef struct {
        char    type;
        union {
                professor       prof;
                student         stu;
        } u;
}person;
```

앞서 살펴본 데이터 구조

Unions

- 예제 프로그램 – **nested structure, union**

```
int main(void)
{
        int     i;
        person  data[2];

        data[0].type  = 'p';
        strcpy(data[0].u.prof.name, "Kim");
        strcpy(data[0].u.prof.mail, "kim@sogang.ac.kr");
        data[0].u.prof.mobile = 1234567;

        data[1].type = 's';
        strcpy(data[1].u.stu.major, "CS");
        strcpy(data[1].u.stu.name, "Chulsoo");
        data[1].u.stu.ID = 20091234;
        data[1].u.stu.cgpa = 3.4;
```

데이터 초기화

Unions

- 예제 프로그램 – **nested structure, union**

```
for(i = 0; i < 2; ++i) {
        switch(data[i].type) {
                case 'p':
                        printf("professor %s\n", data[i].u.prof.name);
                        printf("%s, %d\n", data[i].u.prof.mail, data[i].u.prof.mobile);
                        break;
                case 's':
                        printf("(%s) %d\n", data[i].u.stu.major, data[i].u.stu.ID);
                        printf("%s\nCGPA : %f\n", data[i].u.stu.name, data[i].u.stu.cgpa);
                        break;
        }
        printf("\n");
}
```

형식에 따라 출력

- 출력 결과

```
professor Kim
kim@sogang.ac.kr, 1234567

(CS) 20091234
Chulsoo CGPA : 3.400000
```