구조체 변수와 포인터

```
struct point pos={11, 12};
struct point * pptr=&pos;
```

구조체 point의 포인터 변수 선언

```
(*pptr).xpos=10;
```

pptr이 가리키는 구조체 변수의 멤버 xpos에 접근

```
(*pptr).ypos=20;
```

pptr이 가리키는 구조체 변수의 멤버 ypos에 접근

구조체 포인터 변수를 대상으로 하는 포인터 연산 및 멤버의 접근방법

```
(*pptr).xpos=10;  ⟷  pptr->xpos=10;

(*pptr).ypos=20;  ⟷  pptr->ypos=20;
```

-> 연산자를 기반으로 하는 구조체 변수의 멤버 접근 방법

구조체 변수와 포인터 관련 예제

```
struct point
{
    int xpos;
    int ypos;
};

int main(void)
{
    struct point pos1={1, 2};
    struct point pos2={100, 200};
    struct point * pptr=&pos1;
    (*pptr).xpos += 4;
    (*pptr).ypos += 5;
    printf("[%d, %d] \n", pptr->xpos, pptr->ypos);
    pptr=&pos2;
    pptr->xpos += 1;
    pptr->ypos += 2;
    printf("[%d, %d] \n", (*pptr).xpos, (*pptr).ypos);
    return 0;
}
```

프로그래머들이 주로 사용하는 연산자이니
-> 연산자의 사용에 익숙해지자.

실행결과

```
[5, 7]
[101, 202]
```

메모리 영역별로 저장되는 데이터의 유형

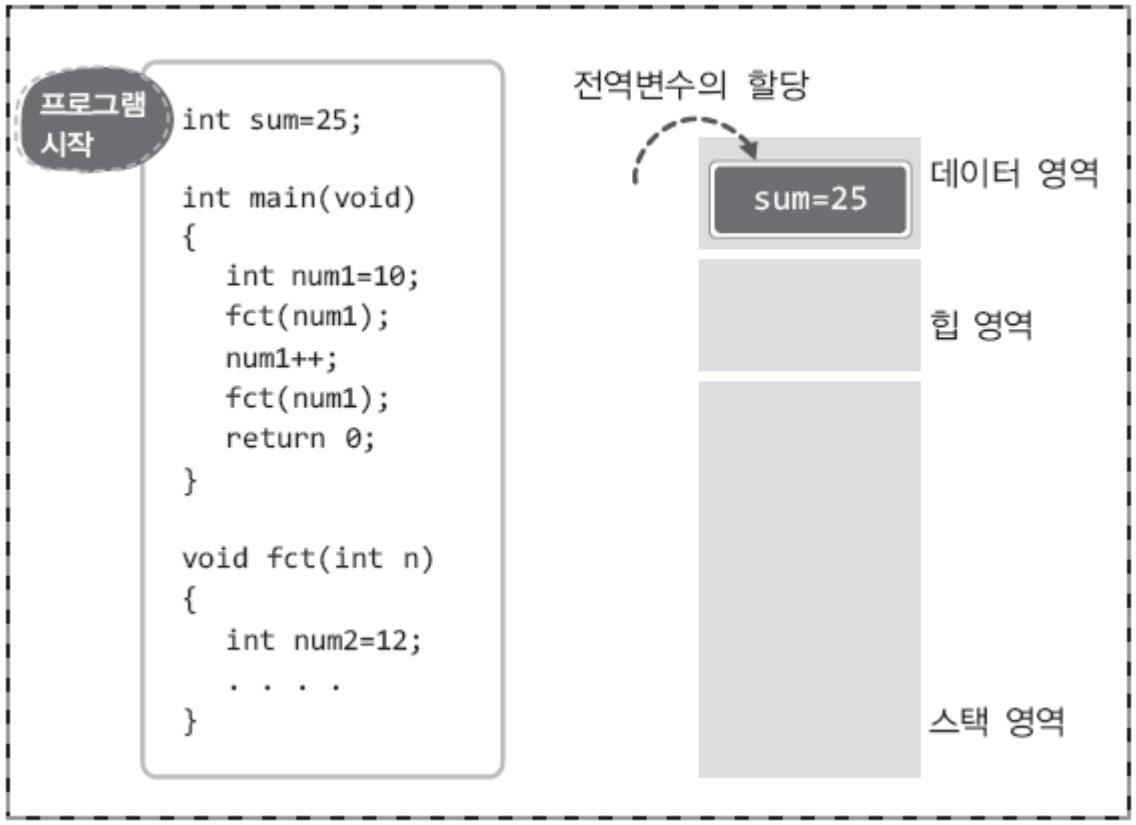
실행할 프로그램의 코드가 저장되는 메모리 공간.
CPU는 코드 영역에 저장된 명령문을 하나씩 가져다가 실행

전역변수와 static 변수가 할당되는 영역.
프로그램 시작과 동시에 할당되어 종료 시까지 남아있는 특징의 변수가 저장되는 영역

프로그래머가 원하는 시점에 메모리 공간에 할당 및 소멸을 하기 위한 영역

지역변수와 매개변수가 할당되는 영역
함수를 빠져나가면 소멸되는 변수를 저장하는 영역

프로그램의 실행에 따른 메모리의 상태 변화1

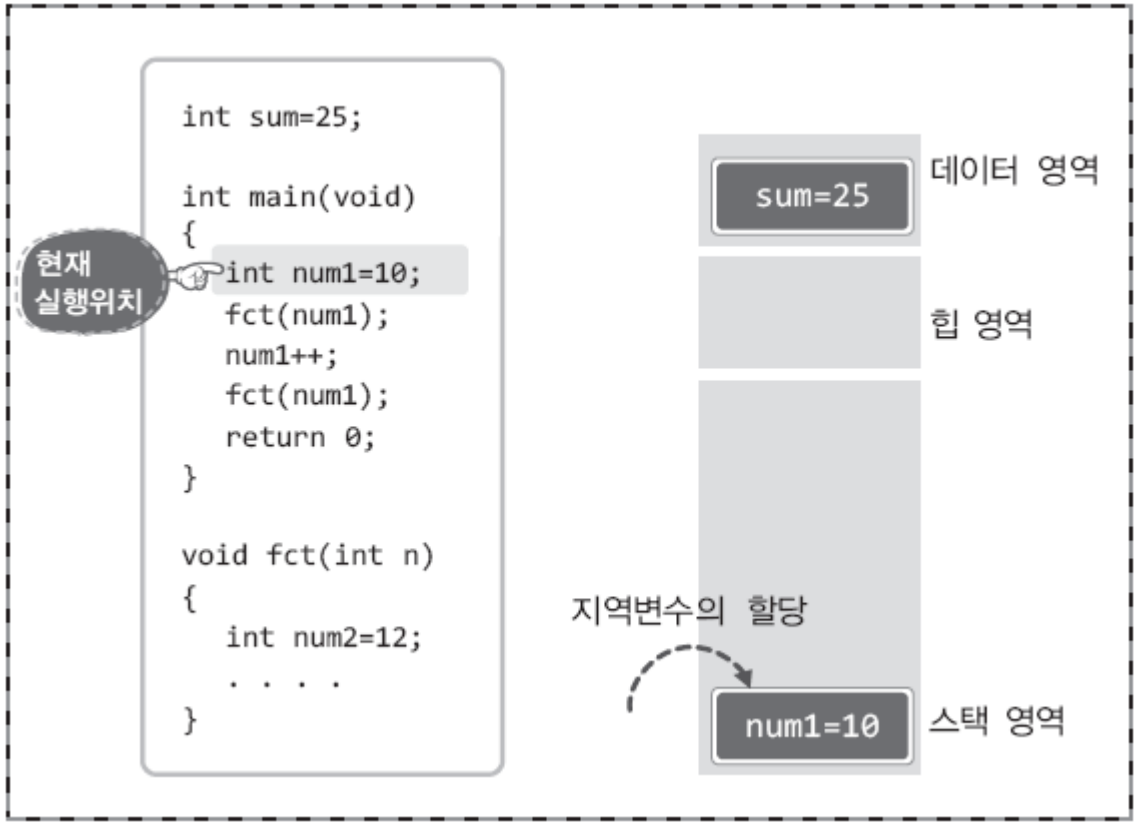


프로그램의 시작:

전역변수의 할당 및 초기화

실행의 흐름!

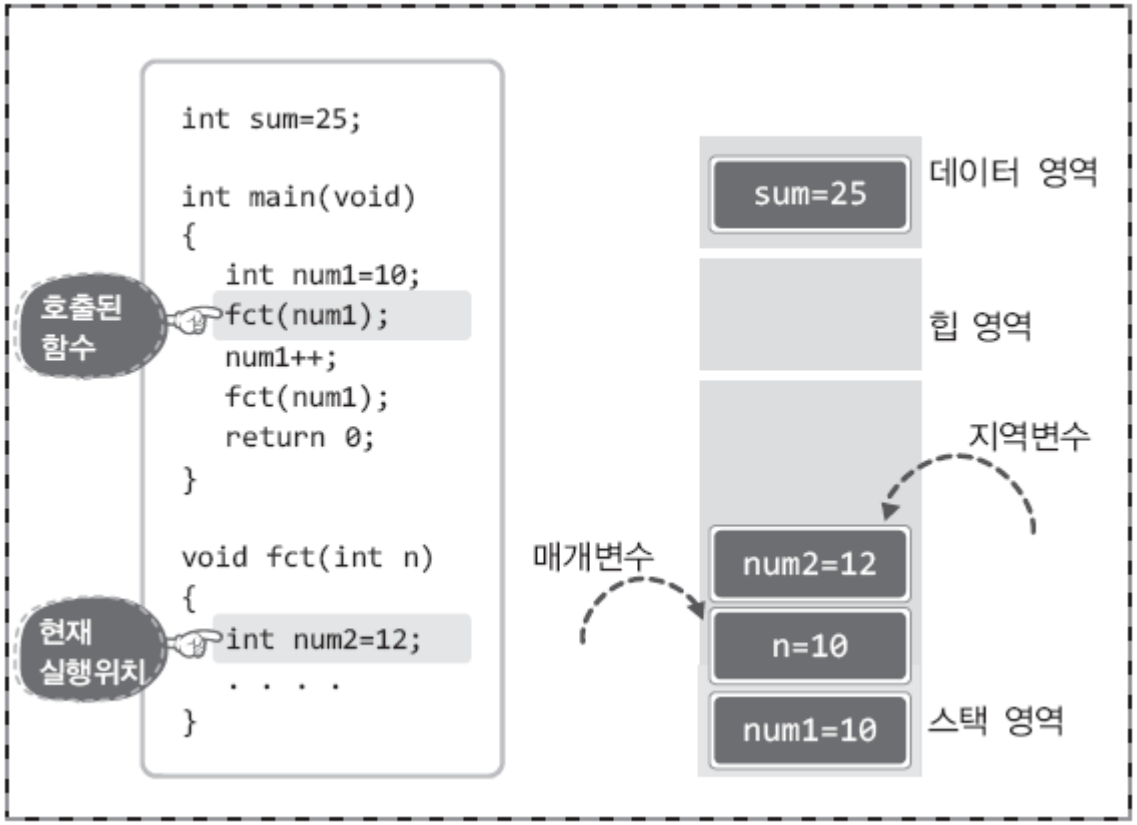
프로그램의 실행에 따른 메모리의 상태 변화2



main 함수의 호출 및 실행

실행의 흐름2

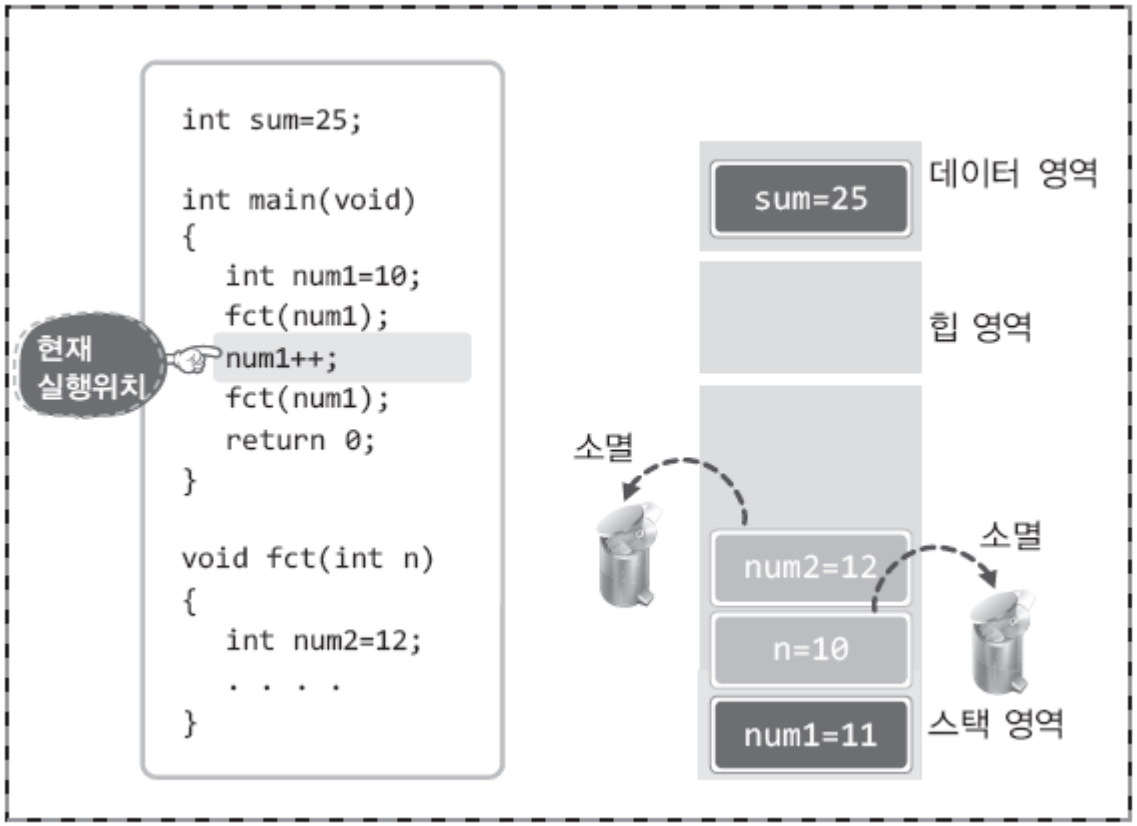
프로그램의 실행에 따른 메모리의 상태 변화3



fct 함수의 호출

실행의 흐름3

프로그램의 실행에 따른 메모리의 상태 변화4

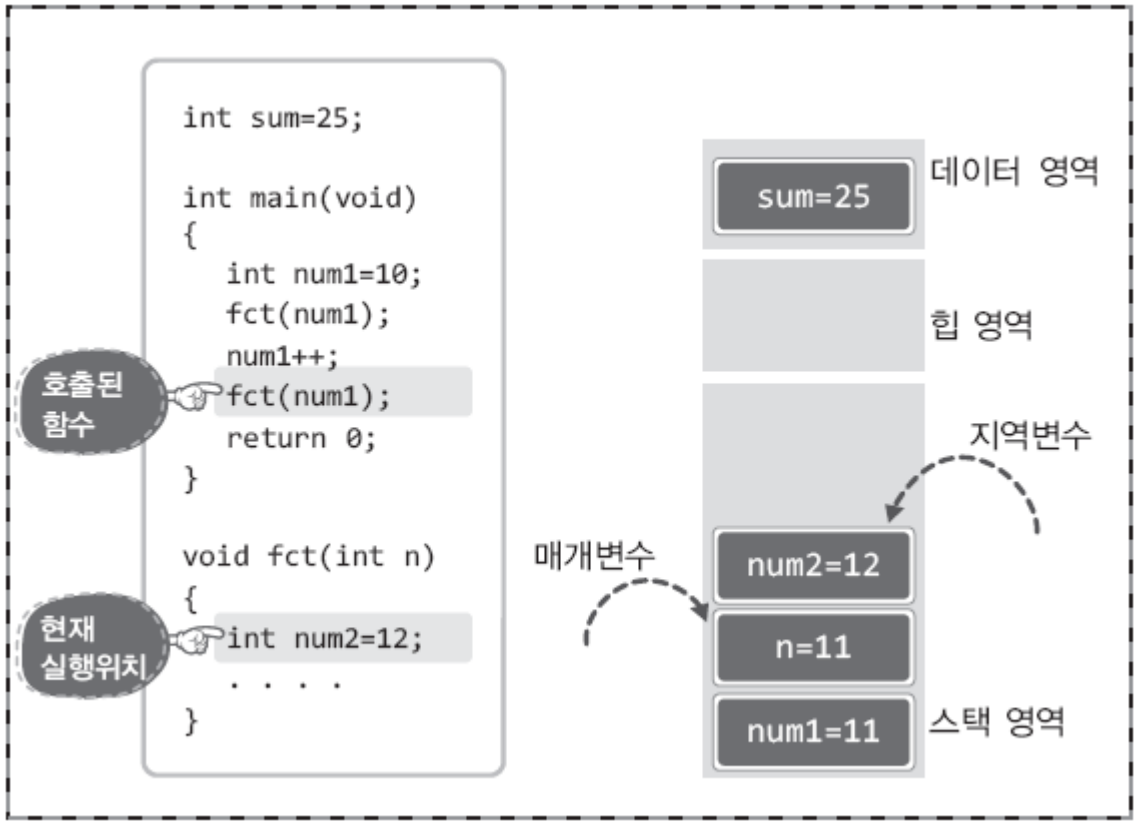


fct 함수의 반환

그리고 main 함수 이어서 실행

실행의 흐름4

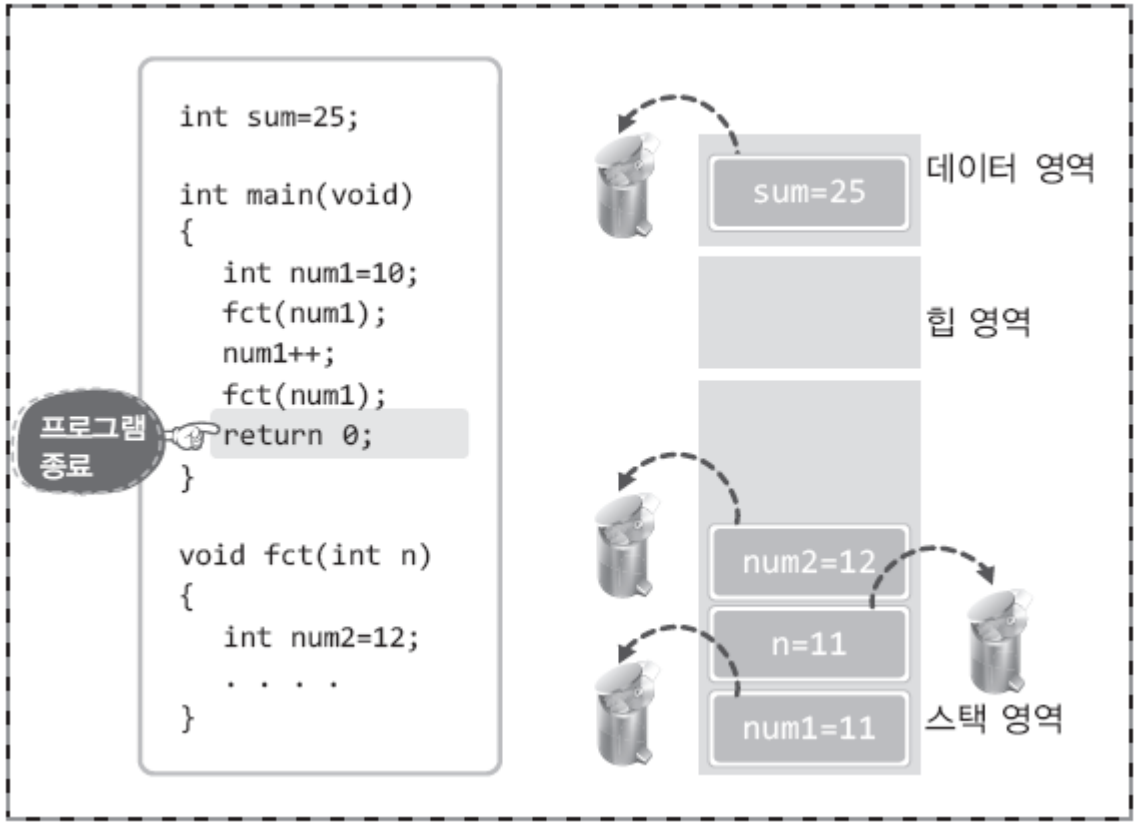
프로그램의 실행에 따른 메모리의 상태 변화5



fct 함수의 재호출 및 실행

실행의 흐름5

프로그램의 실행에 따른 메모리의 상태 변화6



fct 함수의 반환

및 main 함수의 반환

실행의 흐름6 (프로그램 종료)

함수의 호출순서가 main → fct1 → fct2이라면 스택의 반환은(지역변수의 소멸은) 그의 역순인 fct2 → fct1 → main으로 이루어진다는 특징을 기억하자!

헤더파일을 include 하는 두 가지 방법

표준 헤더파일의 포함

```
#include <헤더파일 이름>
```

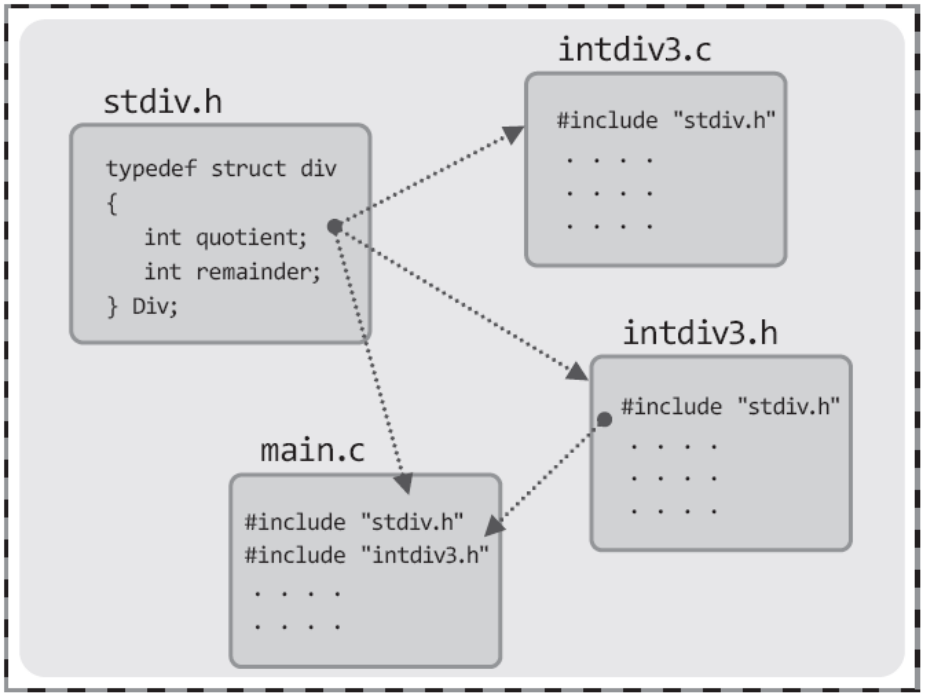
표준헤더 파일을 포함시킬 때 사용하는 방식이다. 표준헤더 파일이 저장된 디렉터리에서 헤더파일을 찾아서 포함을 시킨다.

프로그래머가 정의한 헤더파일의 포함

```
#include "헤더파일 이름"
```

프로그래머가 정의한 헤더파일을 포함시킬 때 사용하는 방식이다. 이 방식을 이용하면 이 문장을 포함하는 소스파일이 저장된 디렉터리에서 헤더파일을 찾게 된다.

헤더파일의 중복삽입 문제

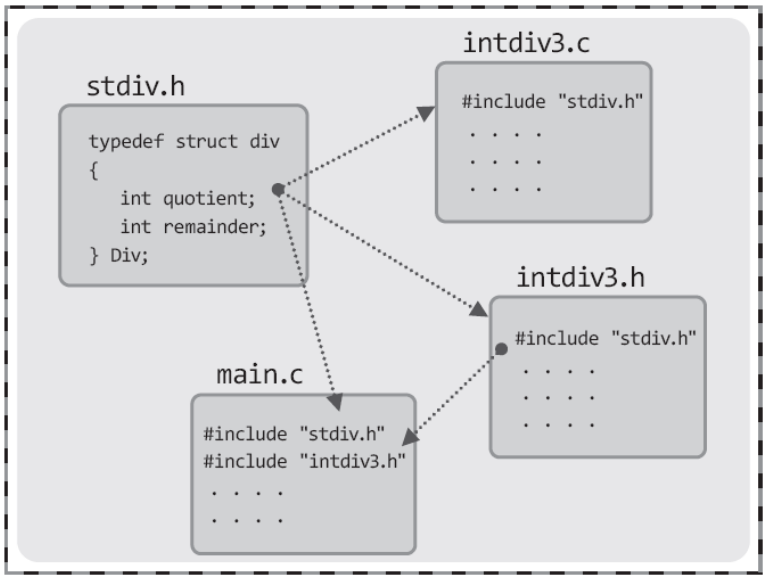


헤더파일을 직접적으로 또는 간접적으로 두 번 이상 포함하는 것 자체는 문제가 아니다. 그러나 두 번 이상 포함시킨 헤더파일의 내용에 따라서 문제가 될 수 있다.

일반적으로 선언(예로 함수의 선언)은 두 번 이상 포함시켜도 문제되지 않는다. 그러나 정의(예로 구조체 및 함수의 정의)는 두 번 이상 포함시키면 문제가 된다.

main.c는 결과적으로 구조체 Div의 정의를 두 번 포함하는 꼴이 된다! 그런데 구조체의 정의는 하나의 소스 파일 내에서 중복될 수 없다!

조건부 컴파일을 활용한 중복삽입 문제의 해결



중복 삽입문제의 해결책

```
#ifndef __STDIV2_H__
#define __STDIV2_H__

typedef struct div
{
    int quotient;     // 몫
    int remainder;    // 나머지
} Div;

#endif
```

매크로 __STDIV2_H__ 와 #ifndef의 효과가 main.c에서
어떻게 나타나는지 그려보자!

위와 같은 이유로 모든 헤더파일은 #ifndef~#endif로 감싸는 것이 안전하고 또 일반적이다!