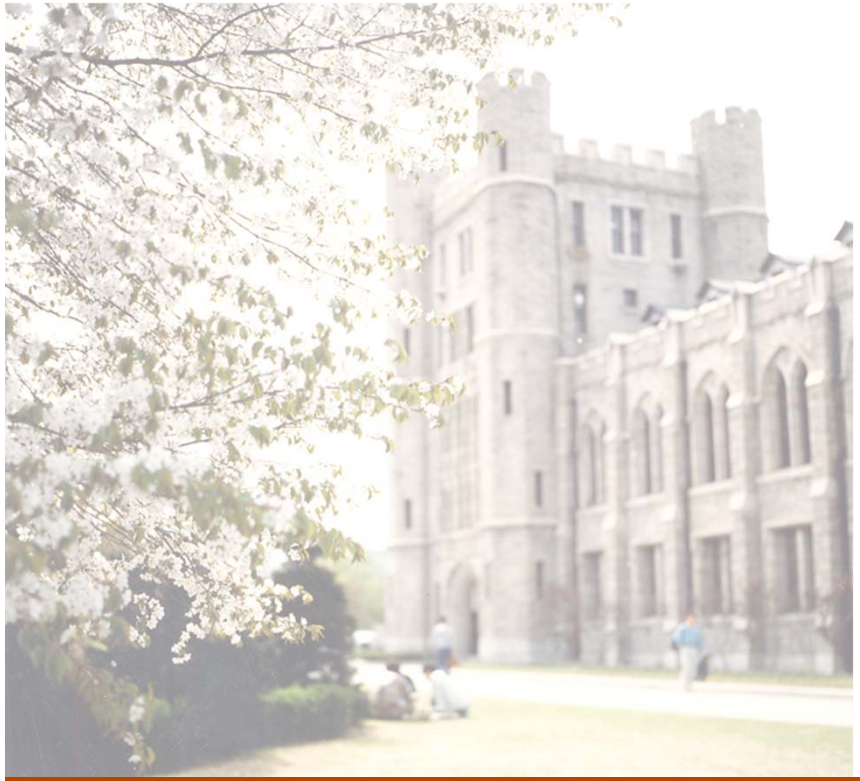
DAEGU UNIVERSITY

42경산 라피신(La piscine) 대비 사전 SW교육과정 - C



Heungwoo Nam

Computer Engineering
Daegu University
2023. 7. 13

Objective & Contents

❑ 수업목표

- 함수와 변수의 이해

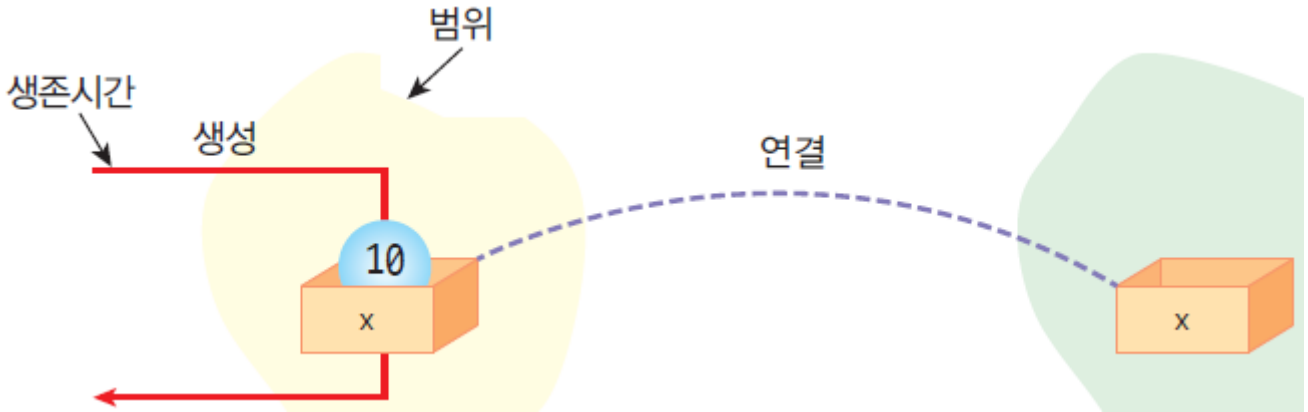
❑ Contents

- 9.1 변수의 속성
- 9.2 지역 변수
- 9.3 전역 변수
- 9.4 생존 기간
- 9.5 연결
- 9.8 순환 호출

9.1 변수의 속성

❑ 변수의 속성

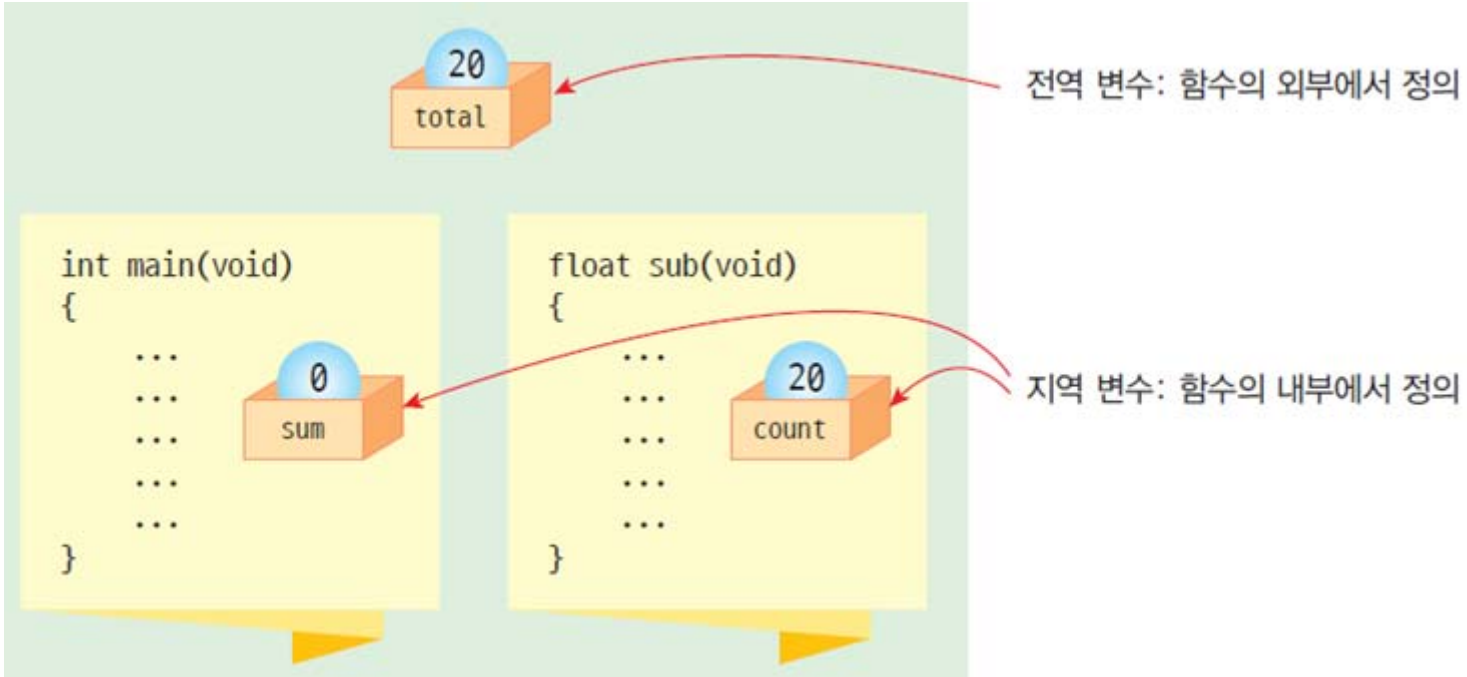
- 기본 속성: 이름, 타입, 크기, 값
- 추가 속성: 범위, 생존 시간, 연결
  - **범위(scope)** : 변수가 사용 가능한 범위
    - 블록 안에 정의되어 있으면 해당 블록으로 범위가 제한
    - 함수외부에서 정의되면 전체 파일에서 변수 사용 가능
  - **생존 시간(lifetime)** : 메모리에 존재하는 시간
    - 생존 시간은 변수에게 할당된 기억 공간의 유지기간 의미
  - **연결(linkage)** : 다른 영역에 있는 변수와의 연결 상태
    - 서로 다른 파일에 존재하는 변수들을 사용할 수 있도록 연결



9.1 변수의 속성

❑ 지역 변수와 전역 변수

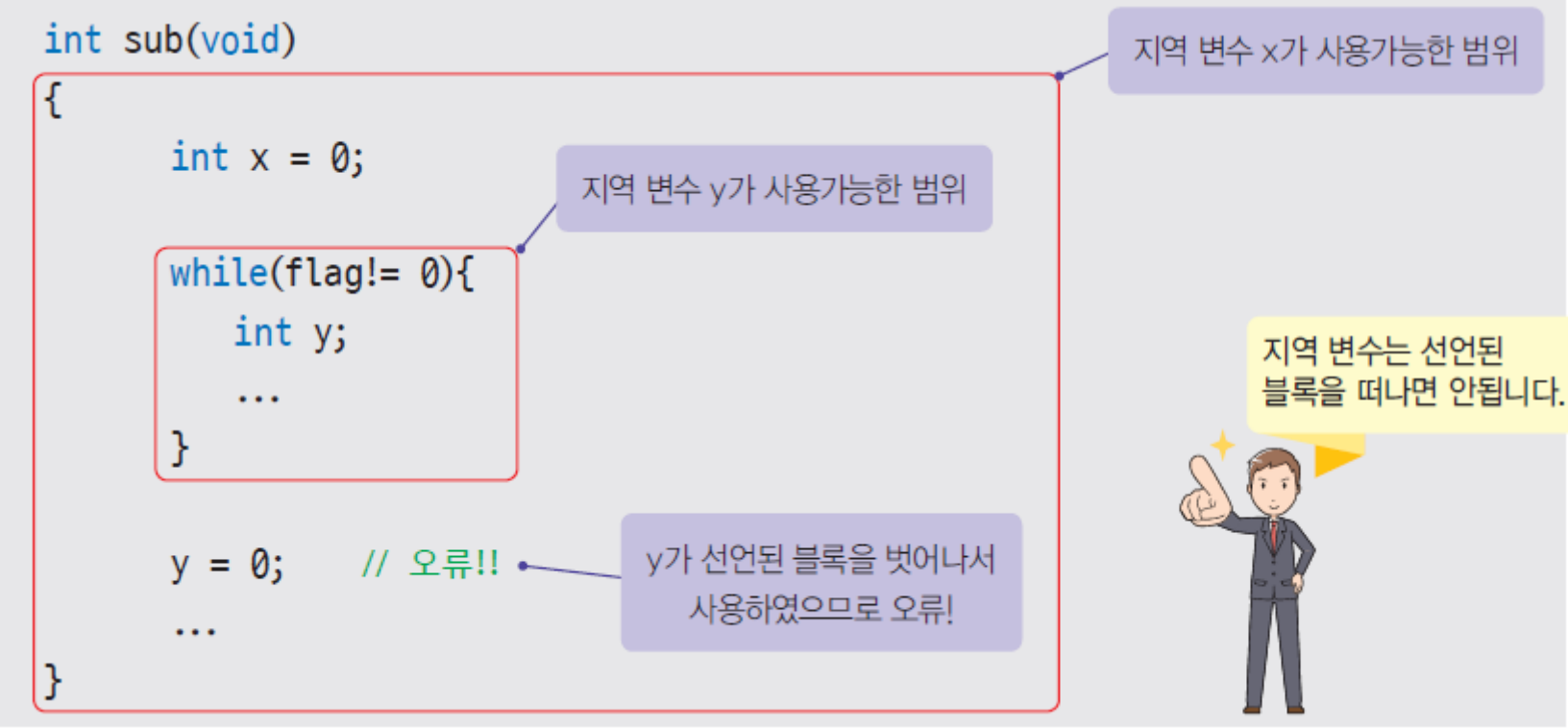
- 지역 변수(local variable): 함수 또는 블록 안에서 정의되는 변수. 지역 변수는 해당 블록이나 함수 안에서만 사용이 가능함.
- 전역 변수(global variable): 함수의 외부에서 선언되는 변수. 전역변수는 소스 파일의 어느 곳에서도 사용이 가능함.



9.2 지역 변수

❑ 지역 변수(local variable)

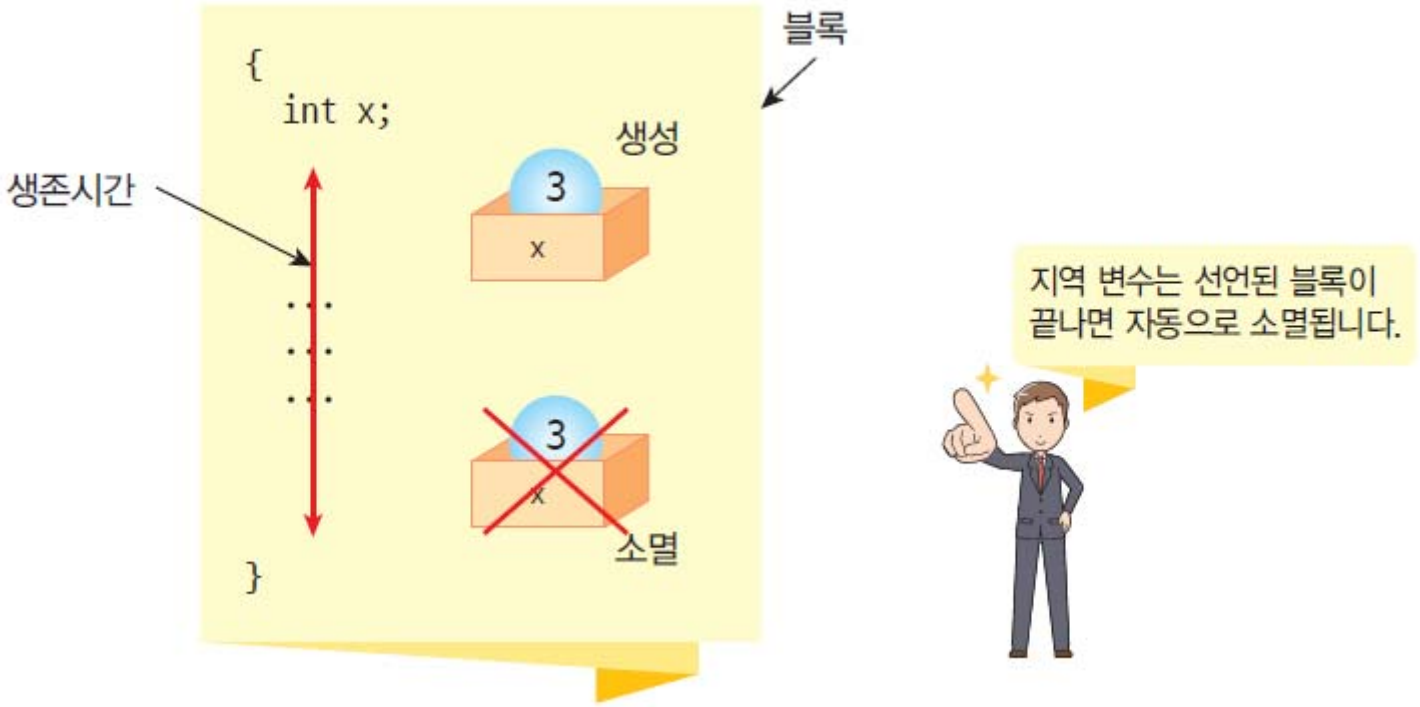
- 블록 안에 선언되는 변수
  - 블록이란 중괄호로 둘러싸인 영역 ({ }): 함수몸체, 반복문 등



9.2 지역 변수

❑ 지역 변수의 생존 기간

- 지역 변수는 변수가 선언된 블록이 시작할 때 시스템 스택(stack)에 생성되며 동시에 초기화됨
- 지역 변수에 할당된 메모리 공간은 블록 끝에서 반환됨.



9.2 지역 변수

❑ 지역 변수 예제

```
#include <stdio.h>

int main(void)
{
        int i;

        for(i = 0;i < 5; i++)
        {
                int temp = 1;
                printf("temp = %d\n", temp);
                temp++;
         }
         return 0;
}
```

블록이 시작할 때 마다 생성되어 초기화된다.

```
temp = 1
temp = 1
temp = 1
temp = 1
temp = 1
```

9.2 지역 변수

❑ 함수의 매개 변수

```
#include <stdio.h>
int inc(int counter);

int main(void)
{
        int i;

        i = 10;
        printf("함수 호출전 i=%d\n", i);
        inc(i);
        printf("함수 호출후 i=%d\n", i);
        return 0;
}
void  inc(int counter)
{
      counter++;
}
```

값에 의한 호출
(call by value)

매개 변수도 일종의
지역 변수임

```
함수 호출전 i=10
함수 호출후 i=10
```

9.3 전역 변수

❑ 전역 변수(global variable)

- 전역 변수는 함수 외부에서 선언되는 변수
- 전역 변수의 범위는 소스 파일 전체임

9.3 전역 변수

❑ 전역 변수(global variable)

- 전역변수 초기값과 생존기간

```
#include<stdio.h>
int A;
int B;
int add()
{
      return A + B;
}

int main()
{
      int answer;
      A = 5;
      B = 7;
      answer = add();
      printf("%d + %d = %d\n", A, B, answer);
      return 0;
}
```

전역 변수
초기값은 0

전역 변수의 범위

5 + 7 = 12

9.3 전역 변수

❑ 전역 변수(global variable)

- 전역변수 초기값

```
#include <stdio.h>

int counter;        ──→  전역 변수
                         초기값은 0
int main(void)
{
    printf("counter=%d\n", counter);
    return 0;
}
```

```
counter=0
```

9.3 전역 변수

❑ 전역 변수의 사용

- 스파게티 코드 (spaghetti code)
  - 전역변수들로 인하여 코드가 꼬이는 현상을 말함.

```
#include <stdio.h>
int x;
void sub();

int main(void)
{
      for(x=0; x<10; x++)
            sub();
}

void sub()
{
      for(x=0; x<10; x++)
            printf("*");
}
```

☞ 전역변수 사용의 기준

1) 거의 모든 함수에서 사용하는 공통적인 데이터는 전역 변수로 함.
2) 일부의 함수들만 사용하는 데이터는 전역 변수로 하지 말고 함수의 인수로 전달한다.

9.3 전역 변수

❑ 같은 이름의 전역 변수와 지역 변수

```
#include <stdio.h>

int sum = 1;          // 전역 변수

int main(void)
{

        int sum = 0;          // 지역 변수

        printf("sum = %d\n", sum);

        return 0;
}
```

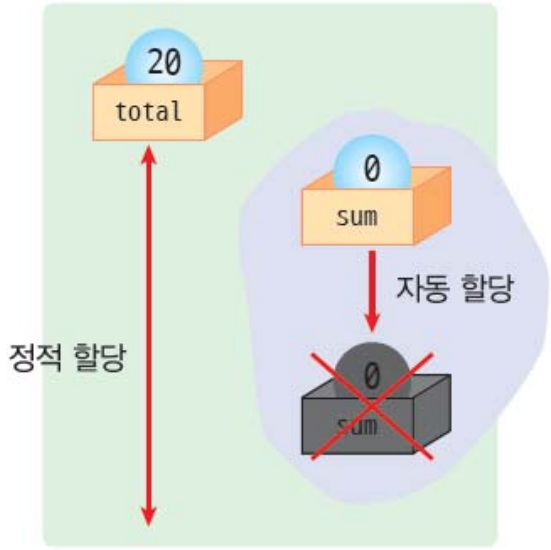
→ 지역변수와 전역변수가 이름이 같은 경우, 지역변수가 우선시 된다.

```
sum = 0
```

9.4 생존 시간

❑ 생존 시간에 따라서 변수를 분류하면 다음과 같음

- 정적 할당(static allocation)
  - 프로그램이 실행되는 동안에는 계속하여서 변수에 저장 공간이 할당되어 있는 방법
- 자동 할당(automatic allocation)
  - 블록이 시작되면서 변수에 저장 공간이 할당되고 블록이 종료되면 저장 공간이 회수되는 방법



9.4 생존 시간

❑ 생존 기간을 결정하는 요인

- 변수가 선언된 위치
  - 전역 변수: 정적 할당
  - 지역 변수: 자동 할당
    - 지역 변수 앞에 저장 유형 지정자를 붙이면 정적 할당 변경가능
- 저장 유형 지정자
  - 변수를 선언할 때 앞에 저장 유형을 지정하는 수식어를 붙일 수 있음.
  - 저장 유형 지정자
    - auto
    - register
    - static
    - extern

9.4 생존 시간

❑ 저장 유형 지정자: static

- 지역 변수처럼 블록에서 사용되지만 블록을 벗어나도 자동으로 제거되지 않는 변수

```
#include <stdio.h>

void sub() {
      static int scount = 0;
      int acount = 0;
      printf("scount = %d\t", scount);
      printf("acount = %d\n", acount);
      scount++;
      acount++;
}

int main(void) {
      sub();
      sub();
      sub();
      return 0;
}
```

정적 지역 변수로서 static을 붙이면 지역 변수가 정적 변수로 된다.

```
scount = 0 acount = 0
scount = 1 acount = 0
scount = 2 acount = 0
```

9.4 생존 시간

❑ lab: 한번만 초기화하기

- 정적 변수는 한번만 초기화하고 싶은 경우에도 사용됨

```
#include <stdio.h>
#include <stdlib.h>
void init();
int main(void)
{
        init();
        init();
        init();
        return 0;
}
void init() {
        static int inited = 0;
        if( inited == 0 ){
                printf("init(): 네트워크 장치를 초기화합니다. \n");
                inited = 1;
        }
        else  printf("init(): 이미 초기화되었으므로 초기화하지 않습니다. \n");
}
```

9.5 연결

❑ 연결(linkage)

: 다른 범위에 속하는 변수들을 서로 연결하는 것

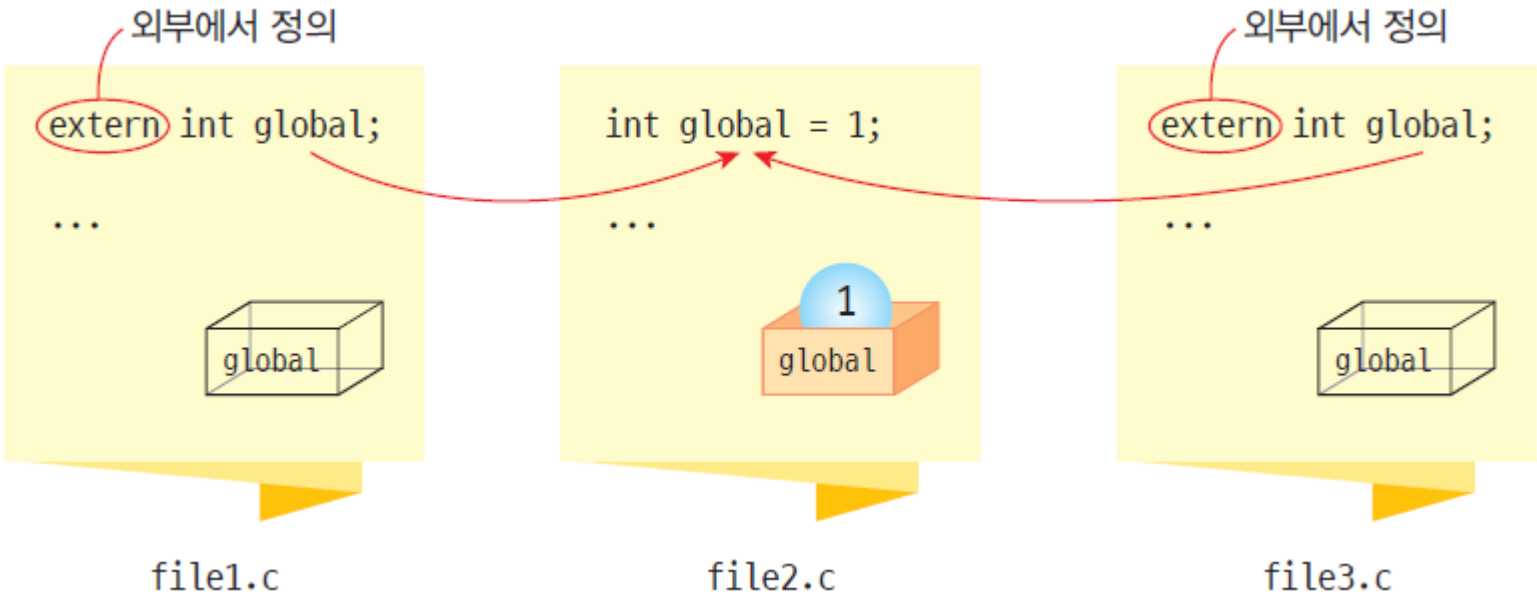
- 무연결(no linkage)
  - 지역변수로서 외부와 연결을 가지지 않음
- 외부 연결(external linkage)
  - 전역변수로서 여러 개의 소스 파일에 걸쳐서 사용됨
- 내부 연결
  - 전역변수로서 하나의 소스 파일에서만 사용이 가능함.

❑ 전역 변수만이 연결을 가질 수 있음.

9.5 연결

❑ 외부 연결

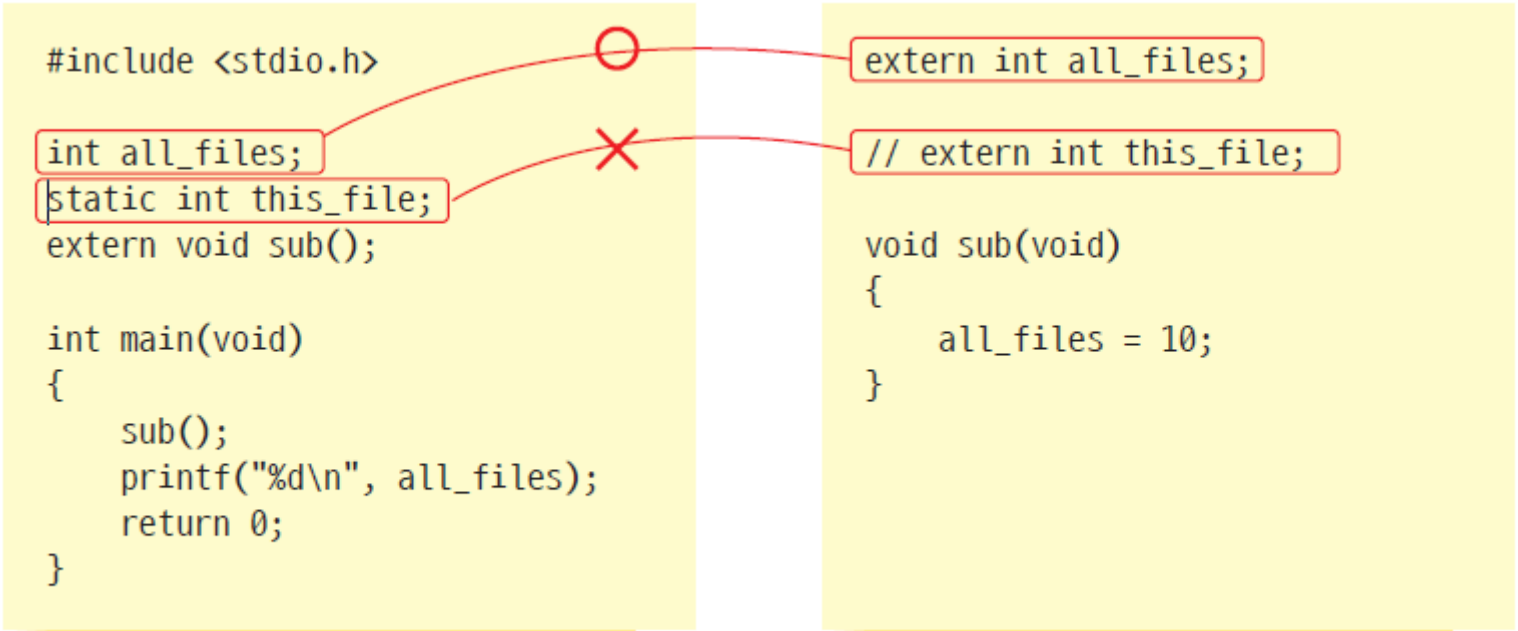
- 전역 변수를 extern을 이용하여서 서로 연결함.
  - 하나의 파일에서 전역 변수를 선언하고
  - 다른 파일에서는 extern 지정자를 사용하여 변수를 참조함
  - 변수를 초기화하는 것은 변수가 정의된 파일에서만 가능함.



9.5 연결

❑ 내부 연결

- 전역 변수 앞에 static이 붙으면 내부 연결이 됨.
  즉 하나의 소스 파일 안에서만 사용이 가능함.



```c
#include <stdio.h>

int all_files;
static int this_file;
extern void sub();

int main(void)
{
    sub();
    printf("%d\n", all_files);
    return 0;
}
```

linkage1.c

```c
extern int all_files;

// extern int this_file;

void sub(void)
{
    all_files = 10;
}
```

linkage2.c

실행결과

```
10
```

9.8 순환 호출

❑ 순환 (recursion)

- 함수는 자기 자신을 호출할 수 있는데, 이것을 순환이라고 함
- 예제) 팩토리얼 구하기
  - 팩토리얼 프로그래밍: (n-1)! 팩토리얼을 현재 작성중인 함수를 다시 호출하여 계산(순환 호출)

$$n! = \begin{cases} 1 & n=0 \\ n*(n-1)! & n \geq 1 \end{cases}$$

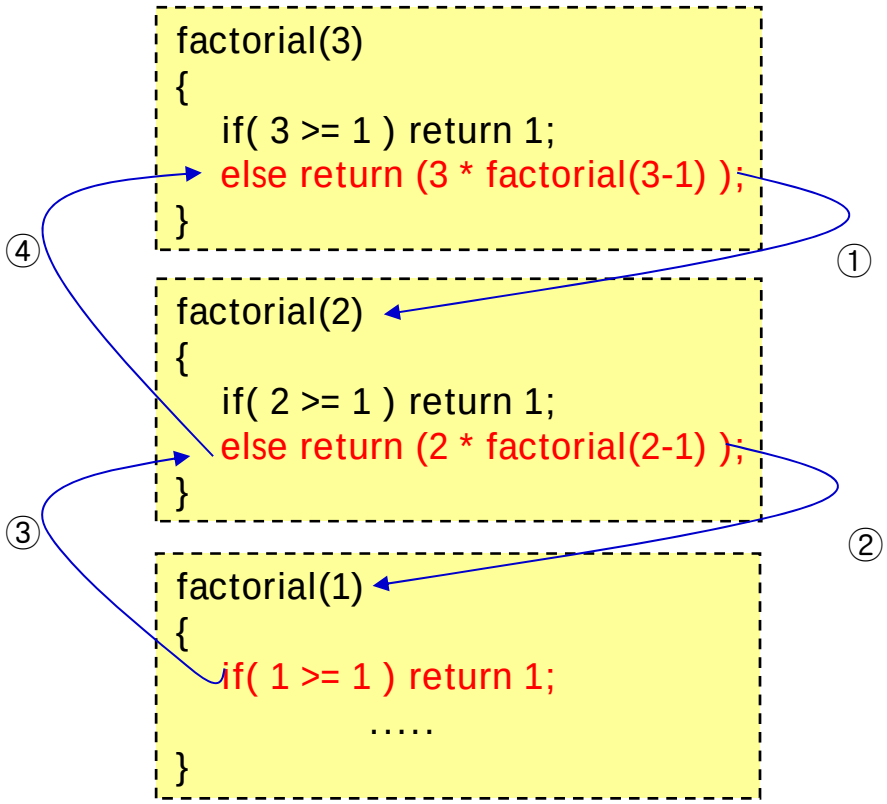
```
int factorial(int n)
{
    if( n <= 1 ) return(1);
    else return (n * factorial(n-1) );
}
```

9.8 순환 호출

❑ 순환 (recursion)

- 예제) 팩토리얼 구하기

```
factorial(3) = 3 * factorial(2)
             = 3 * 2 * factorial(1)
             = 3 * 2 * 1
             = 3 * 2
             = 6
```



9.8 순환 호출

❑ 순환 (recursion)

- 예제) 팩토리얼 구하기

```
// 재귀적인 팩토리얼 함수 계산
#include <stdio.h>

long factorial(int n)
{
        printf("factorial(%d)\n", n);

        if(n <= 1) return 1;
        else return n * factorial(n - 1);
}

int main(void)
{
        int x = 0;
        long f;

        printf("정수를 입력하시오:");
        scanf("%d", &n);
        printf("%d!은 %d입니다. ₩n", n, factorial(n));
        return 0;
}
```