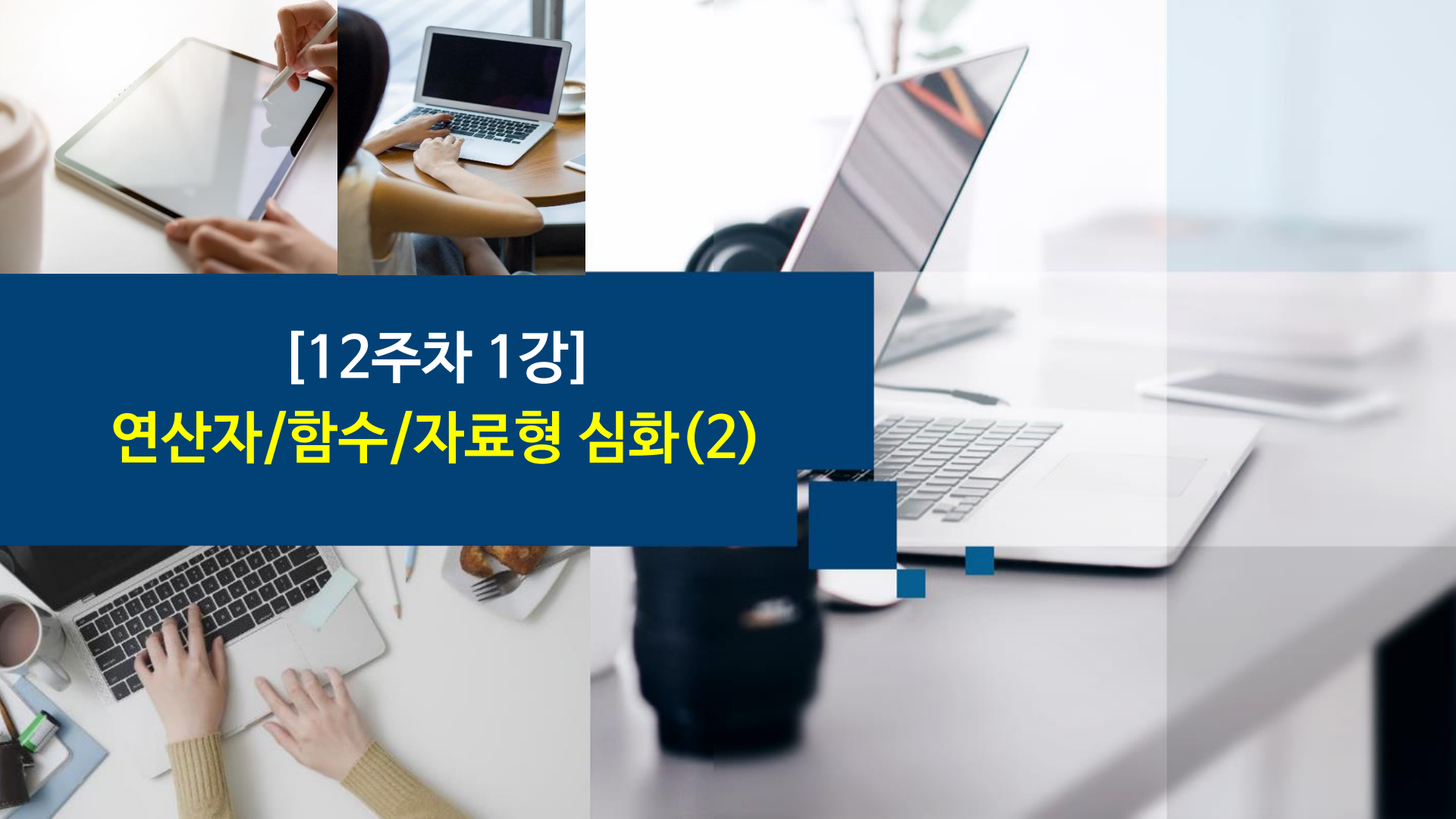


[12주차 1강]
연산자/함수/자료형 심화(2)

학습 내용

- 13.2 재귀함수

학습 목표

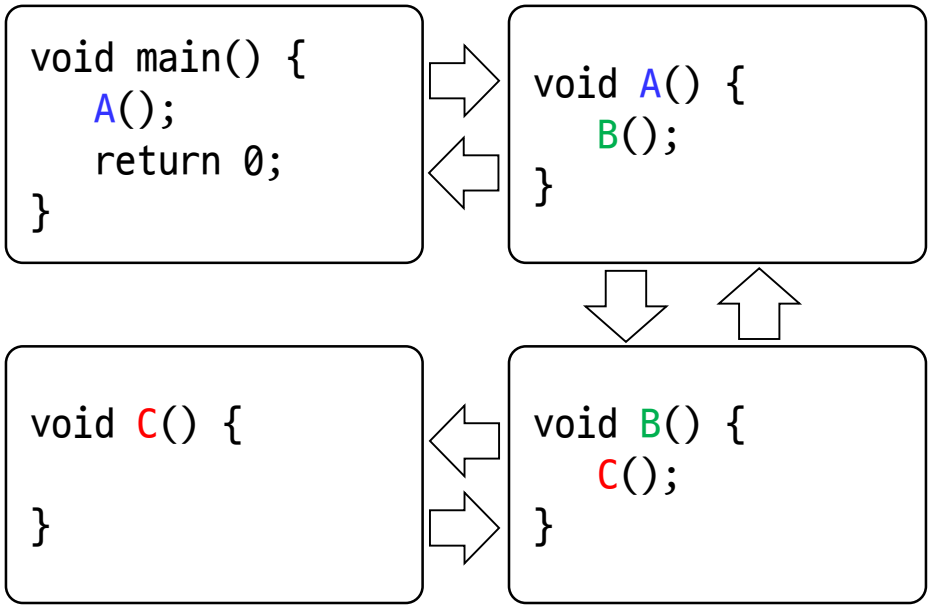
- 13.2 재귀함수의 개념을 이해하고 활용할 수 있다.

C프로그래밍및실습

1. 비트연산자
2. 재귀함수
3. 라이브러리 활용

함수 호출과 반환

- 다음 프로그램의 함수 호출과 반환 과정은?



함수 호출과 반환

- 출력문을 넣어 확인해보자.

```
void main() {
    A();
    return 0;
}
```

```
void A() {
    printf("A start\n");
    B();
    printf("A end\n");
}
```

```
void B() {
    printf("B start\n");
    C();
    printf("B end\n");
}
```

```
void C() {
    printf("C start\n");

    printf("C end\n");
}
```

출력 결과

```
A start
B start
C start
C end
B end
A end
```

재귀 함수 맛보기

- 다음 프로그램은 정상적인 프로그램일까?
  - ✓ dec( ) 함수에서 자기를 다시 호출한다??
  - ✓ 그럼 현재 수행 중이던 dec( ) 함수는 어떻게 되는 거지??
  - ✓ 놀랍게도(?) 정상적으로 컴파일도 되고 실행도 된다.

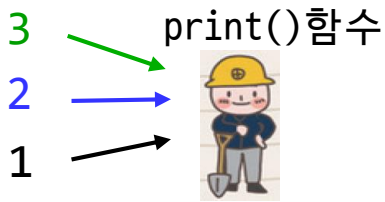
```
void dec(int x) {
    printf("x: %d\n", x);
    if( x > 1)
        dec(x-1); // 자기 자신 호출(?)
}
int main() {
    dec(3);
    return 0;
}
```

출력 결과

```
x: 3
x: 2
x: 1
```

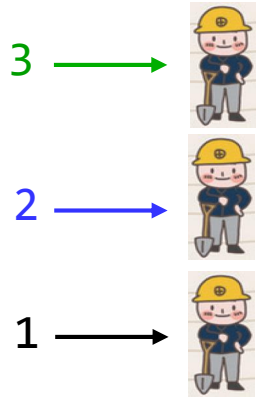
함수 호출 과정 들여다보기

- 아래 두 그림 중 어느 것이 print( ) 함수의 호출 과정을 더 정확하게 표현한 것일까?



1명의 일꾼(함수)이
3건의 요청을 처리??

```
void print(int x){
   printf("x: %d\n", x);
}
void main(){
   print(3);
   print(2);
   print(1);
}
```

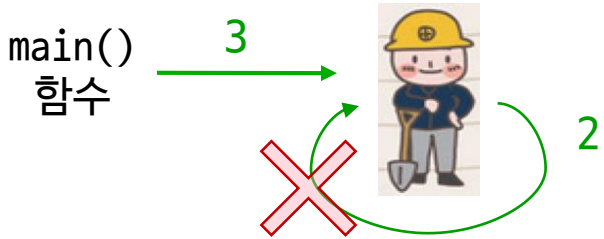


동일한 기능을 가진
3명의 일꾼(함수)이
각각 1건의 요청을 처리

재귀 함수 호출 과정

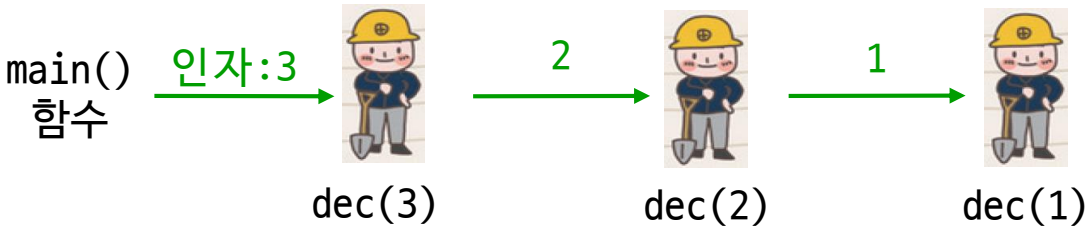
- 이전 프로그램의 함수 호출 과정을 그림으로 표현하면?

```
void dec(int x) {
    printf("x: %d\n", x);
    if( x > 1) dec(x-1);
}
void main()
{
    dec(3);
}
```



**자기 자신에게
일을 요청하는 것이 아님**

**일꾼(함수) dec가
자신과 동일한 기능을 가진
다른 일꾼 dec에게 요청**



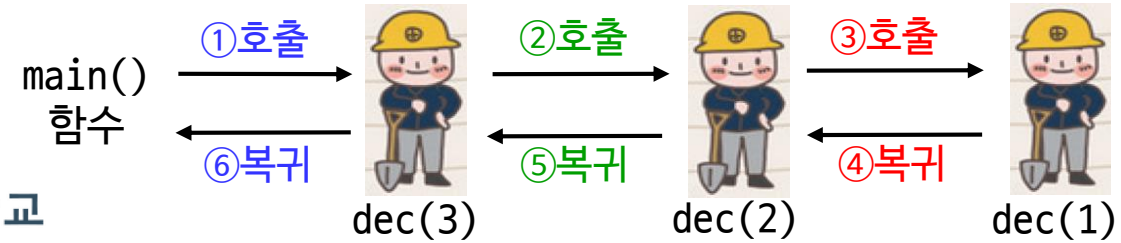
재귀 함수 동작 과정

- 동작 과정 파악을 위해 함수의 시작과 끝에 출력문 삽입

```
void dec(int x) {
    printf("+start: x=%d\n",x);
    if( x > 1) dec(x-1);
    printf("-end: x=%d\n",x);
}
void main() {
    dec(3);
    return 0;
}
```

출력 결과

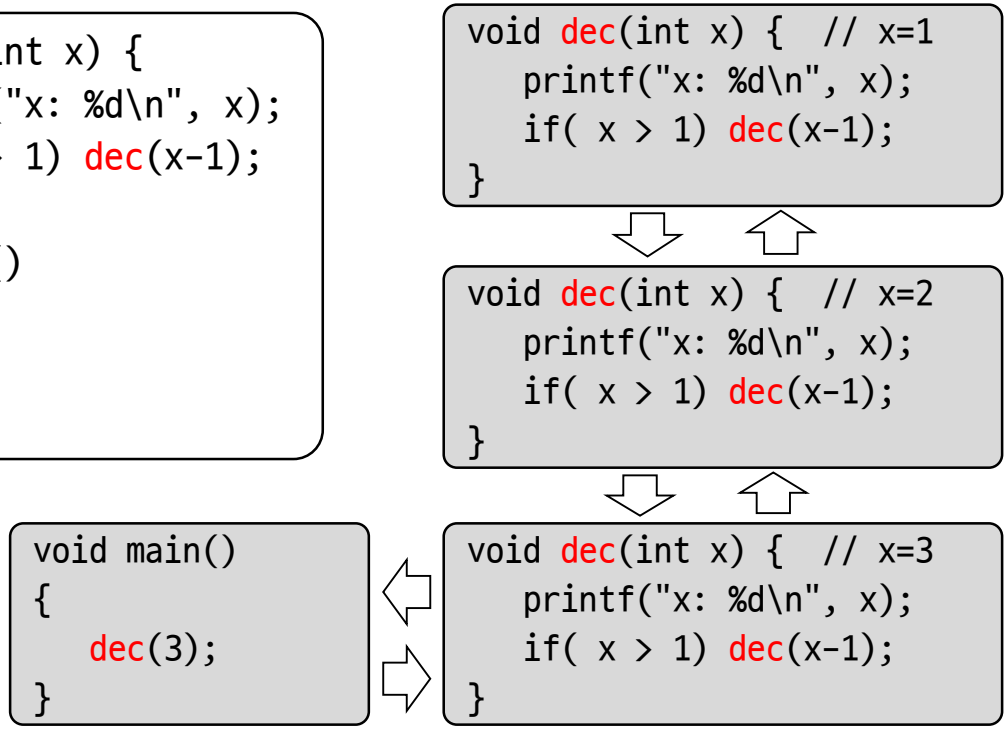
```
+start: x=3
+start: x=2
+start: x=1
-end: x=1
-end: x=2
-end: x=3
```



함수 정의와 호출의 의미

- 함수 정의는 틀에 해당하고, 호출이 되어야 실체를 가짐

```
void dec(int x) {
    printf("x: %d\n", x);
    if( x > 1) dec(x-1);
}
void main()
{
    dec(3);
}
```



재귀 함수

- 함수 내부에서 자기와 동일한 (이름의) 함수를 호출하는 함수

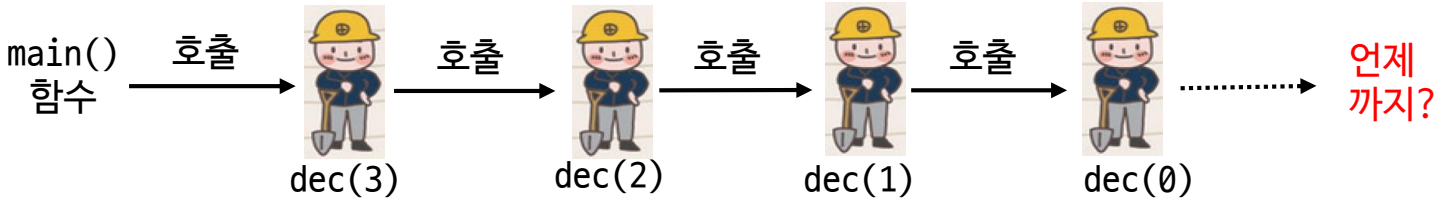
- 유사 개념: 점화식
  - ✓ $A_n = A_{n-1} + 2$ (등차 수열의 점화식)
  - ✓ A : 함수 이름에 해당
  - ✓ 첨자 n : 함수 인자에 해당

2. 재귀함수

주의사항: 재귀 호출의 종료 조건이 없으면 제대로 동작하지 않음

- 점화식에서 초기항이 없으면 정의되지 않는 것처럼
  - ✓ $A_n = A_{n-1} + 2$, $A_1 = 1$

```
void dec(int x){
    printf("x: %d\n", x);
    if( x > 1) dec(x-1);  // 없으면 런타임 오류 발생
}
int main(){
    dec(3);
    return 0;
}
```



※ 실습하기

[예제 13.3] 재귀함수를 사용하여 1부터 n까지의 합 계산

- sum(n) = 1+2+3+ … + (n−1) + n을 점화식으로 표현하면?
  - ✓ sum(n) = sum(n−1) + n (n 〉 1 인 경우)
  - ✓ sum(1) = 1; (n==1 인 경우)

```
int sum(int n) {
   int s;       // 합 저장
   if( n == 1)  s = 1;
   else   s = sum(n-1) + n;
   return s;
}
int main() {
   printf("%d", sum(10) );
   return 0;
}
```

```
// simple version

int sum(int n) {
   if( n == 1)  return 1;
   return sum(n-1) + n;
}
void main() {
   printf("%d", sum(10) );
   return 0;
}
```

※ 실습하기

[예제 13.4] 재귀함수를 이용하여 n!을 계산하는 프로그램을 작성하라.

- n! = n * (n−1) * (n−2) * ⋯ * 2 * 1

- 점화식으로 표현하면
  - ✓ n! = n*(n−1)! (n 〉 1 인 경우)
  - ✓ 1! = 1; (n==1 인 경우)

학습 정리

- **재귀 함수**란 자신과 동일한 함수를 호출하는 함수
- 함수를 정의하는 코드는 틀에 해당하고, 호출이 되어야 실체를 가짐
- 재귀 호출은 하나의 틀(함수)에서 자신과 모양이 동일한 또 다른 함수 실체를 만들어 내는 과정임
- 재귀 호출의 종료 조건이 없으면 제대로 동작하지 않음