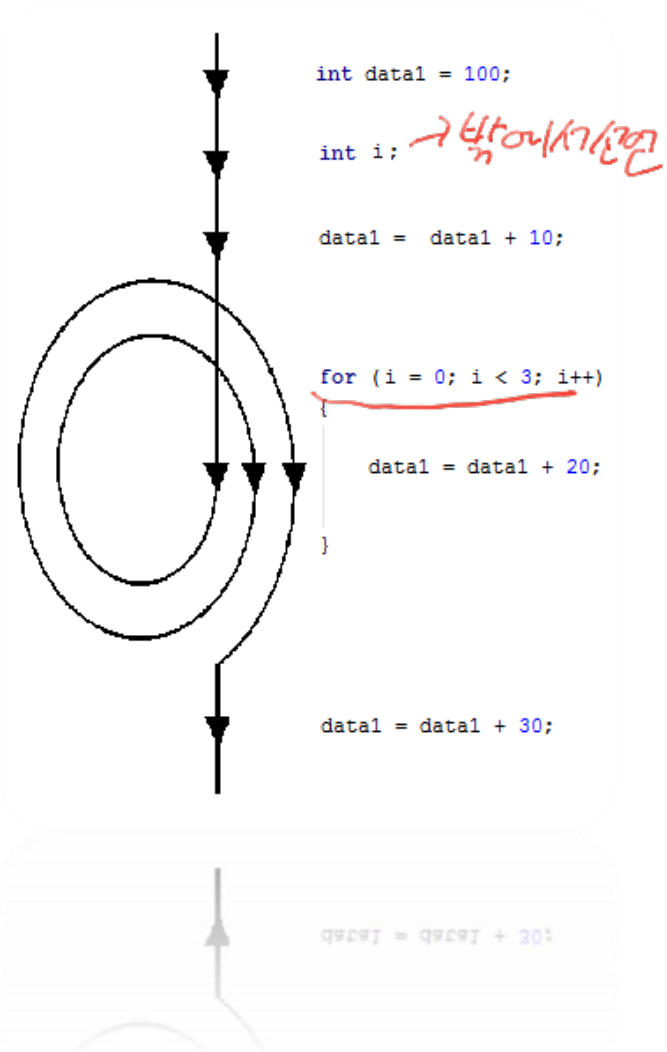
Looping Statements: For Loop

Section 1 Basic **for** Statements
Section 2 Nested **for** Loops
Section 3 Various Applications

0 Types of for Loops

4가지 기본 형태, 많이 쓰이는 것은 2가지

- **Traditional for-Loops**
  - 대개 수열과 함께 쓰이는 반복문으로
  - 0부터 1씩 증가하는 등차수열,
  - 또는 N에서 1씩 감소하는 등차수열로 가장 쉽게 널리 쓰인다.
  - 보통 그 수열의 수는 i, j, k로 쓰이며 Loop Counter가 중심
  - Loop Counter: i, j, k를 지칭하는 말



- **Iterator-based for-Loops**
  - for each: C에서 지원하지 않는다. 단, Modern C/C++은 지원

- **Vectorized for-Loops**
  - for all

- **Compound for-Loops**

- 두 번째부터는 다른 언어에서 배우도록 하자. **오늘은 4절 중에 1절만**

전통적 for 반복문의 형식

- **for 반복문의 구성 요소**
  - **초기화**: Loop Counter를 초기화
  - **조건식**: Loop Counter의 범위를 검사
  - **증감문**: Loop Counter에 변화

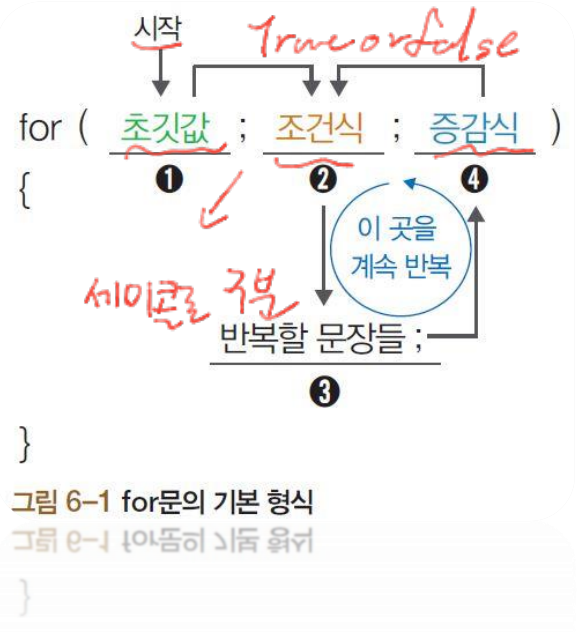


그림 6-1 for문의 기본 형식

- **for 반복문의 실행 절차**
  - 1. 초기화: ❶ 을 수행
  - 2. 조건 검사: ❷ 를 수행
  - 3. 명령 블록 실행: ❸ 의 명령들을 수행
  - 4. 증감문 실행: ❹ 수행
  - 5. 반복: 2로 돌아가 ❷를 통해 반복할 문장들을 계속 수행함.

- **for 반복문의 형식**
  - 괄호 안에 초깃값, 조건식, 증감식이 순서대로 입력되며 ";" 으로 구분됨.
  - 블록 { } 안에 반복할 문장이 나타나는데 ~~반복할 문장이 하나뿐이라면 중괄호를 생략해도 됨.~~

1 Basic for Statements

전통적 for 반복문의 형식

- **기본 구조**

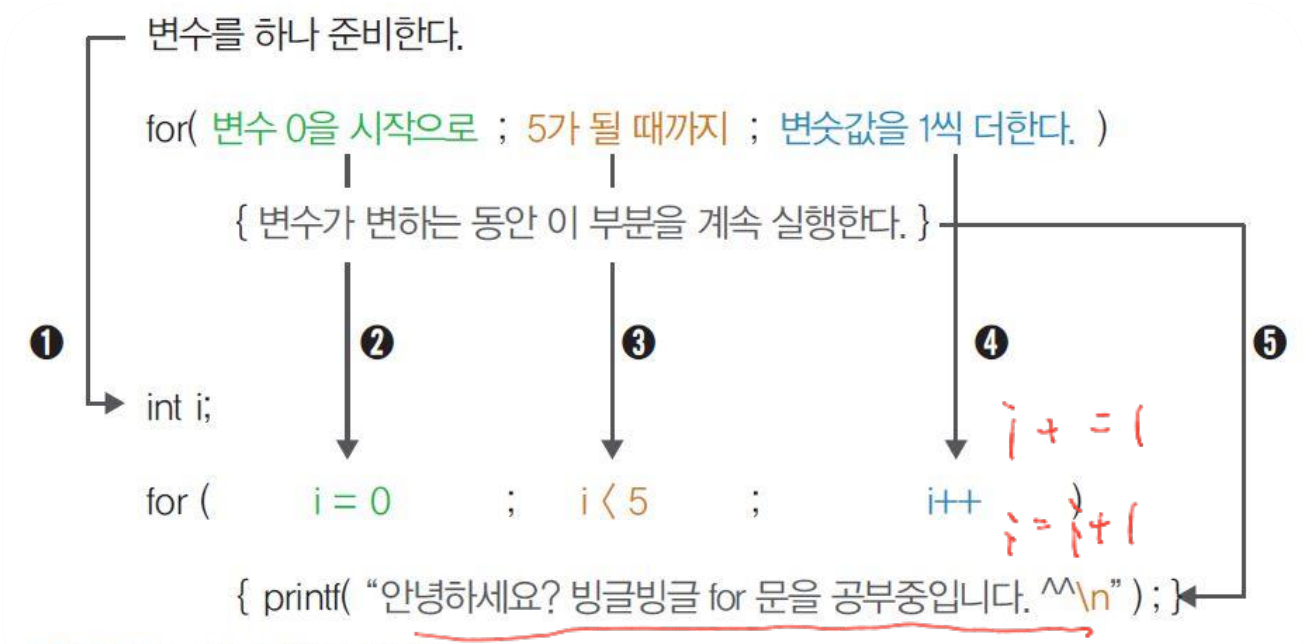


그림 6-2 for문의 개념과 실제 사용

- **참고 사항**
  - ❷ 사용할 변수의 초깃값을 0으로 설정. 꼭 0이 아니어도 되지만 보통 0부터 시작함.
  - 과거의 C에서는 초기화 구문에 변수의 선언을 허용하지 않았음.
  - 현재의 C는 일부 환경에서 초기화 구문에 변수를 선언하면서 초기화할 수 있음
  - for (int i = 0; ...

1 Basic for Statements

전통적 for 반복문의 형식

- 기본 구조

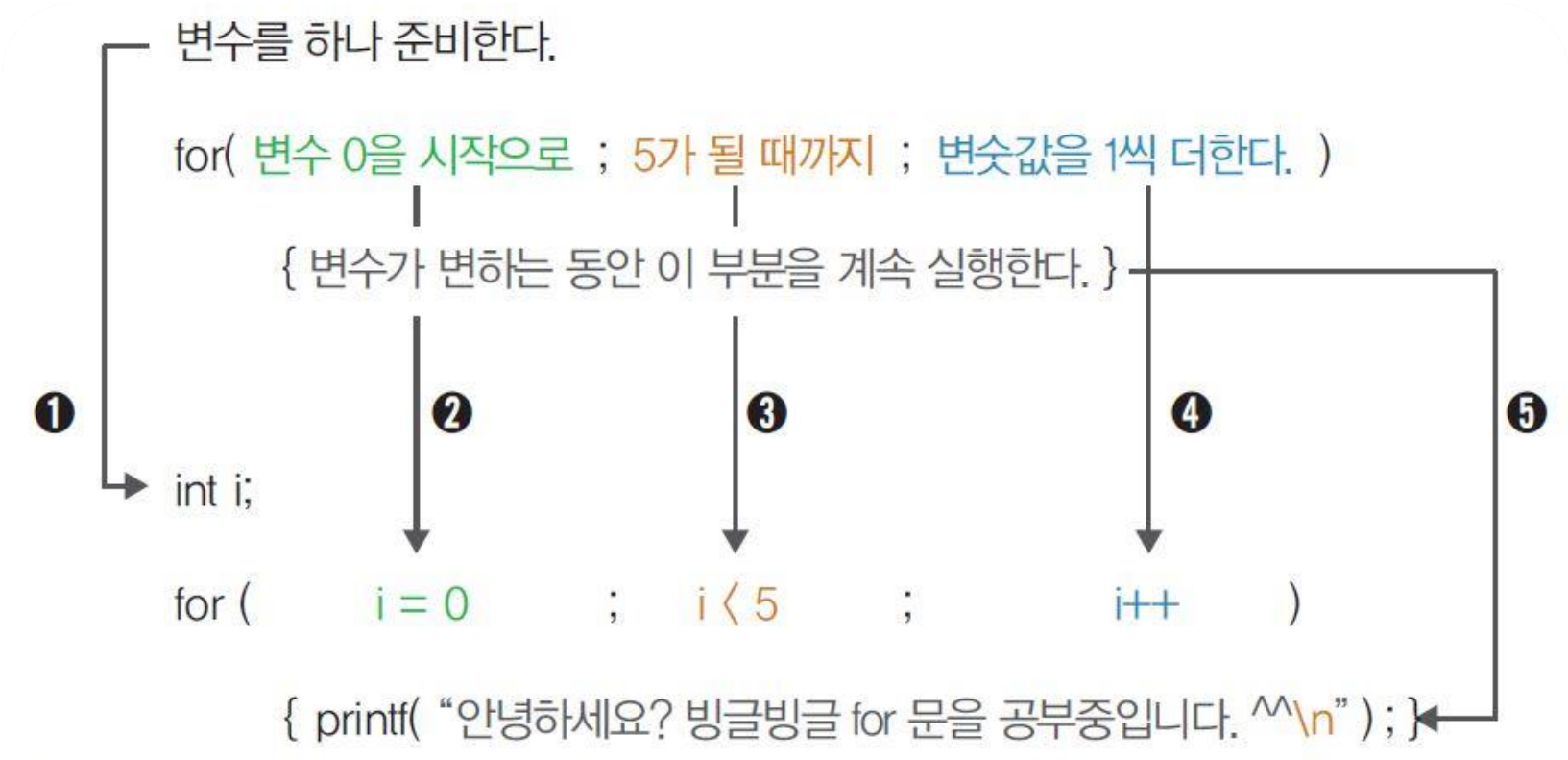


그림 6-2 for문의 개념과 실제 사용

전통적 for 반복문의 형식

- **참고 사항**
  - 과거의 C에서는 초기화 구문에 변수의 선언을 허용하지 않았음.
  - 현재의 C는 일부 환경에서 초기화 구문에 변수를 선언하면서 초기화할 수 있음
  - for (int i = 0; ...

- **❷ 사용할 변수의 초깃값을 0으로 설정. 꼭 0이 아니어도 되지만 보통 0부터 시작함.**
  - 왜 why?

- **자연수 (Natural Number) 의 정의**
  - 자연수의 숫자 범위: 1부터 무한대의 정수 ➔ 왜 why?

  - 컴퓨터공학에서 취급하는 자연수의 숫자 범위는?
  - 그리고 그 이유는?

전통적 for 반복문의 형식

- **❷ 사용할 변수의 초깃값을 0으로 설정.**
  - 꼭 0이 아니어도 되지만 보통 0부터 시작함
  - 0으로부터 시작되는 등차 수열
- **등차 수열을 Loop Counter로 사용시 반복 횟수 설정 방법**
  - ❸에서 다섯 번 실행해야 하므로 'i<=5'라고 생각하기 쉽지만,
  - ❷에서 초깃값이 0부터 시작하므로 'i<5'로 설정해야 0, 1, 2, 3, 4로 총 다섯 번 실행됨.
  - 만약 'i<=5'라고 쓰면 여섯 번 실행됨. ➔ { 0, 1, 2, 3, 4, 5, 6 }
- **❹ 'i++'는 'i=i+1'과 동일한 역할. i를 1 증가시킴.**
  - 이 부분은 for문의 명령어 블록의 실행이 종료될 때마다 수행함.
- **❺ 실제로 반복되는 명령어 블록.**
  - printf문 한 줄만 들어있지만, 복잡한 프로그램일수록 많은 내용이 들어감.
  - 또, 간소화를 위해 일부 내용을 증감식 구문으로 이동시킬 수도 있음

- **[그림 6-2]에서 화살표를 따라 계속 수행해 보면, 초깃값은 한 번만 실행되고 나머지가 계속 반복되는 구조다.**

전통적 for 반복문의 형식

- **❷ 사용할 변수의 초깃값을 0으로 설정.**
  - 꼭 0이 아니어도 되지만 보통 0부터 시작함
  - 0으로부터 시작되는 등차 수열

- **등차 수열을 Loop Counter로 사용시 반복 횟수 설정 방법**
  - ❸에서 다섯 번 실행해야 하므로 'i<=5'라고 생각하기 쉽다.
  - ❷에서 초깃값이 0부터 시작하므로 'i<5'로 설정해야 0, 1, 2, 3, 4로 총 다섯 번 실행됨.
  - 만약 'i<=5'라고 쓰면 여섯 번 실행됨. ➔ { 0, 1, 2, 3, 4, 5, 6 }

- **❹ 'i++'는 'i=i+1'과 동일한 역할. i를 1 증가시킴.**
  - 이 부분은 for문의 명령어 블록의 실행이 종료될 때마다 수행함.

- **❺ 실제로 반복되는 명령어 블록.**
  - printf문 한 줄만 들어있지만, 복잡한 프로그램일수록 많은 내용이 들어감.
  - 또, 간소화를 위해 일부 내용을 증감식 구문으로 이동시킬 수도 있음

전통적 for 반복문의 형식

코딩 규칙

- 명령 블록에 명령이 하나이더라도 반드시 { } 를 사용한다.
- for 뒤에 공백을 사용한다.
- ; 뒤에 공백을 사용한다.
-) 뒤에 공백을 사용하고 명령어 블록을 연다.

```
for (int i = 0; i < kSomeNumber; i++) {
    printf("I take it back\n");
}
```

for 반복문의 Flow Chart

1 Basic for Statements

전통적 for 반복문의 형식

- **구조 분석**

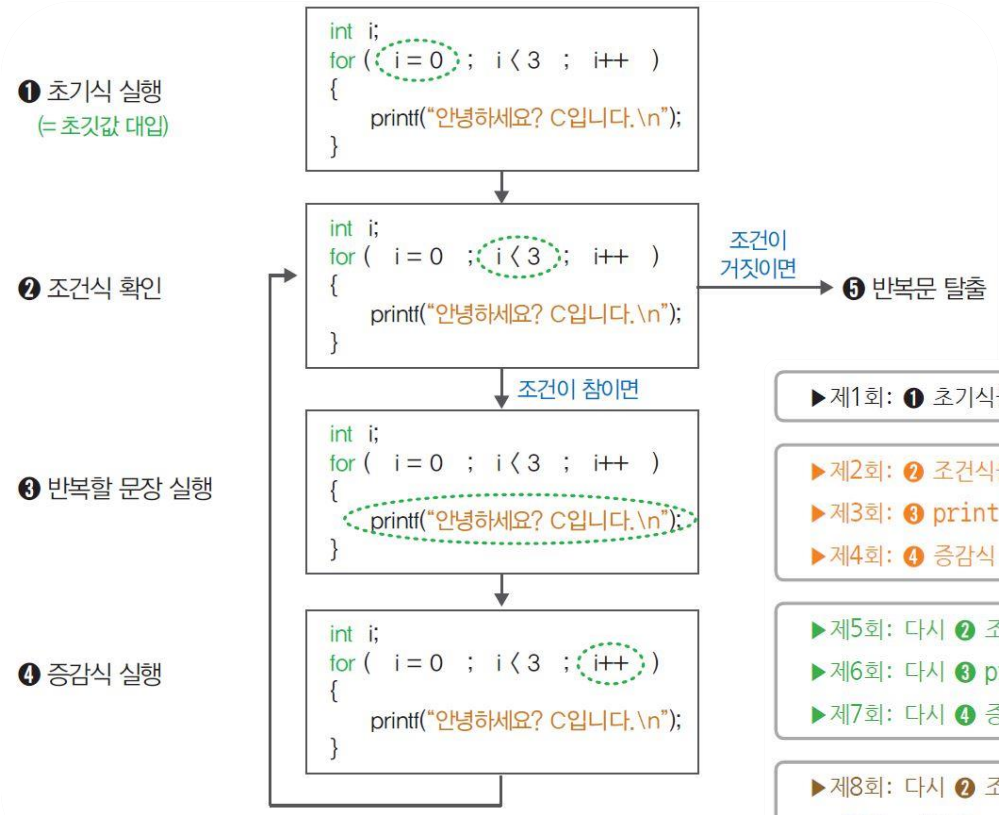


그림 6-3 for문이 반복되는 순서

▶제1회: ❶ 초기식을 수행한다(현재 i=0).

▶제2회: ❷ 조건식을 확인한다. 현재 i 값이 0이므로 i<3은 참이다.
▶제3회: ❸ printf문을 수행한다('안녕하세요? …'를 출력한다).
▶제4회: ❹ 증감식 i++를 수행해서 i 값을 1 증가시킨다(현재 i=1).

▶제5회: 다시 ❷ 조건식을 확인한다. 현재 i 값이 1이므로 i<3은 참이다.
▶제6회: 다시 ❸ printf문을 수행한다('안녕하세요? …'를 출력한다).
▶제7회: 다시 ❹ 증감식 i++를 수행해서 i 값을 1 증가시킨다(현재 i=2).

▶제8회: 다시 ❷ 조건식을 확인한다. 현재 i 값이 2이므로 i<3은 참이다.
▶제9회: 다시 ❸ printf문을 수행한다('안녕하세요? …'를 출력한다).
▶제10회: 다시 ❹ 증감식 i++를 수행해서 i 값을 1 증가시킨다(현재 i=3).

▶제11회: 다시 ❷ 조건식을 확인한다. 현재 i 값이 3이므로 드디어 i<3이 거짓이다.
▶제12회: 조건식이 거짓이므로 ❺ 반복문을 탈출하고 반복문 블록({ }) 밖의 내용을 수행한다.

2 Nested for Loops
반복문의 중첩

- **중첩 for문의 개념**
  - for문의 명령어 블록에 또 다른 for문이 들어 있는 형태
  - 참고: 총 반복 횟수 = 바깥 for문 Loop Counter 집합의 크기 x 안쪽 바깥 for문 LC 집합의 크기
    - m x n회 라고 하자

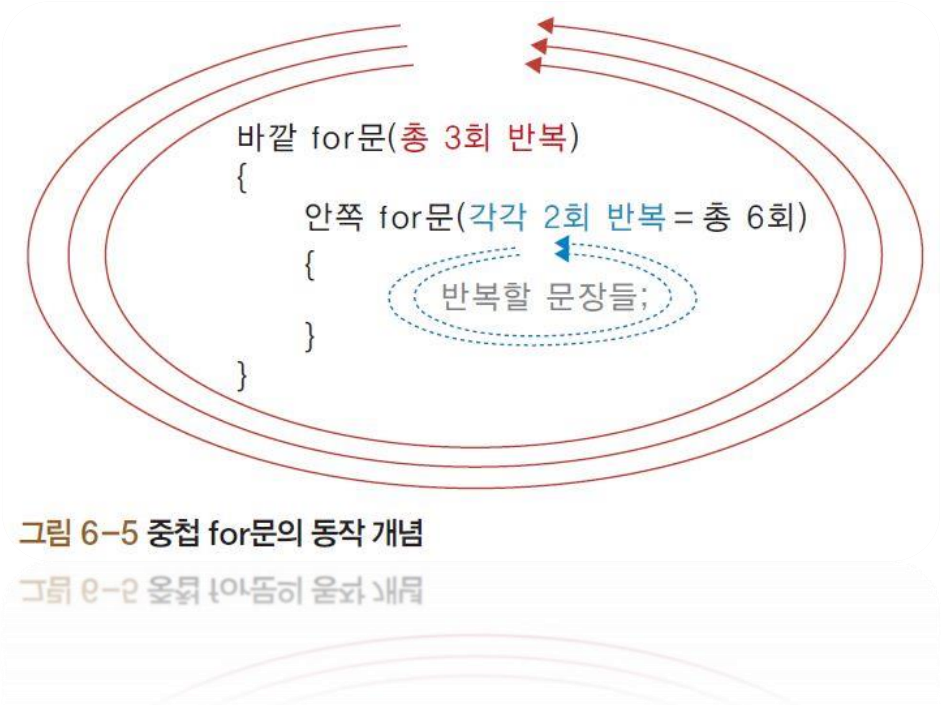


그림 6-5 중첩 for문의 동작 개념

- **n회 동작하는 블록을 m회 수행하기 때문임**

2 Nested for Loops
반복문의 중첩

- **동작 구조 분석**

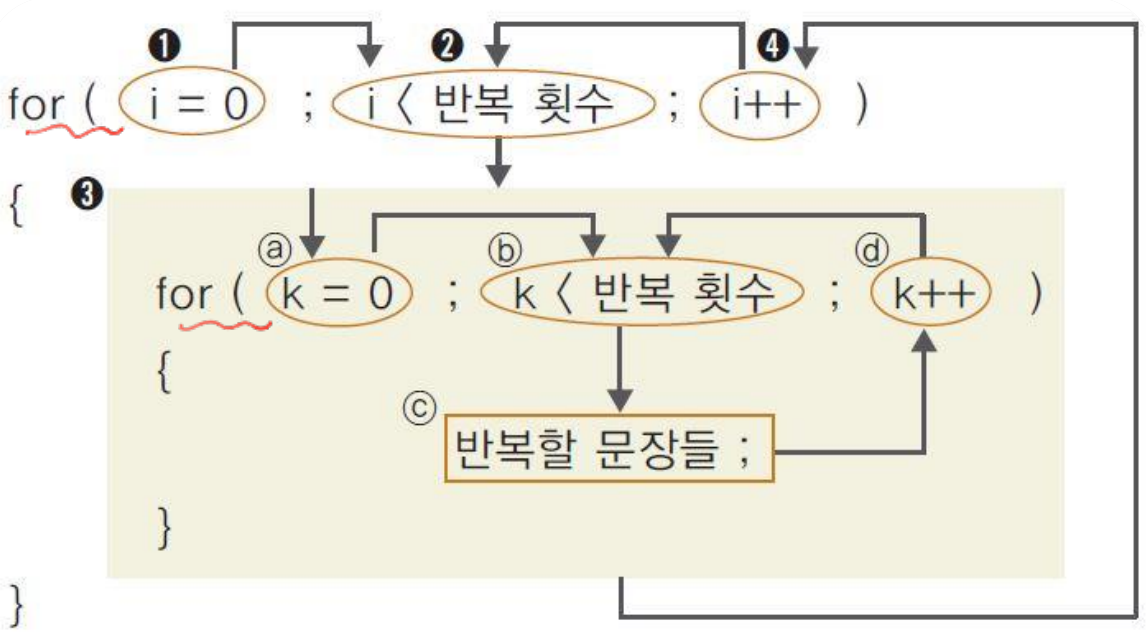


그림 6-6 중첩 for문의 작동 방식

❶ → ❷ → ❸ → (ⓐ → ⓑ → ⓒ → ⓓ → ⓑ → ⓒ → ⓓ → ⓑ → 안쪽 for문을 빠져나감) → ❹ → ❷ → ❸ → (ⓐ → ⓑ → ⓒ → ⓓ → ⓑ → ⓒ → ⓓ → ⓑ → 안쪽 for문을 빠져나감) → ❹ → ❷ → ❸ → (ⓐ → ⓑ → ⓒ → ⓓ → ⓑ → ⓒ → ⓓ → ⓑ → 안쪽 for문을 빠져나감) → ❹ → ❷ → 바깥 for문을 빠져나감

2 Nested for Loops
반복문의 중첩

- **코드 분석**

```
for (int i = 0; i < 3; i++) {
    for (int i = 0; i < 2; i++) {
        printf("I take it back\n");
    }
}
```

→ 밖의 반복문이 다 개친다.

```
for (int i = 0; i < 3; i++) {
    for (int k = 0; k < 2; k++) {
        printf("I take it back\n");
    }
}
```

- 뭐가 달라?

- 너희는 이렇게 안할 거 같아?

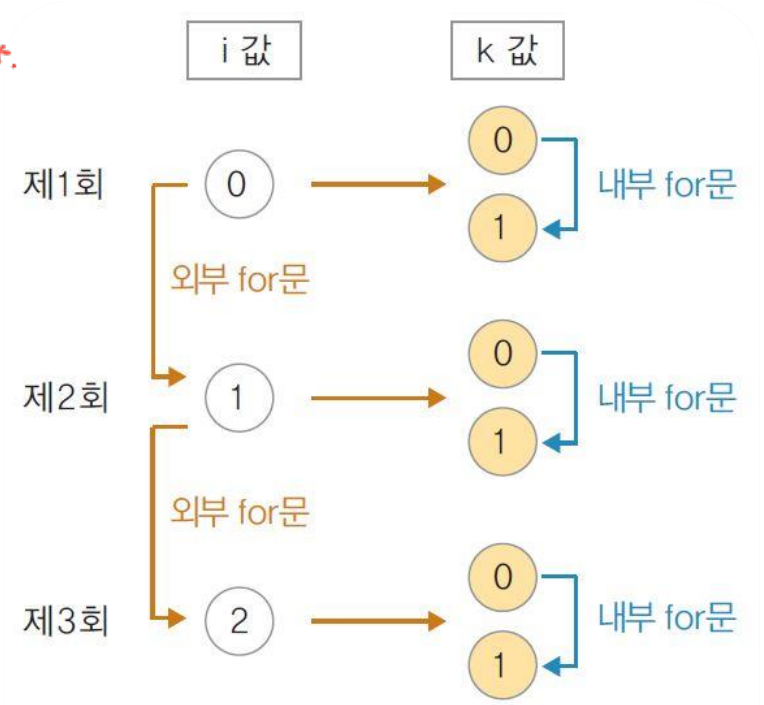


그림 6-7 중첩 for문에서 i 값과 k 값의 변화

2 Nested for Loops

반복문의 중첩 예시

- **구구단**

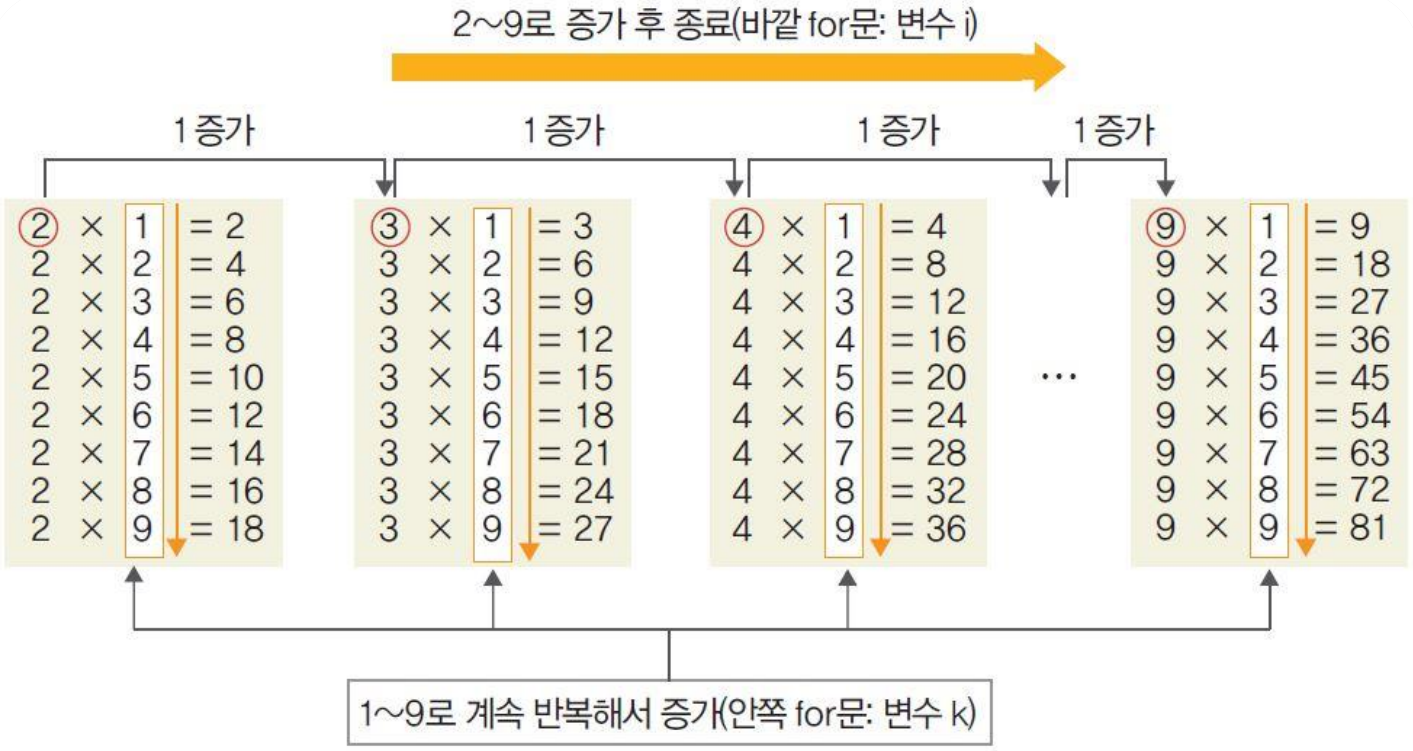


그림 6-8 구구단에서 변수 i와 k의 추출

- 이 형식을 지키면서 출력하려면 for 반복문을 어떻게 중첩시켜야 할까?
- 다음 페이지로 넘어가기 전에 줄바꿈, 긴 공백 등을 꼼꼼하게 종이에 써가면서 따져보자.

2 Nested for Loops

반복문의 중첩 예시

- **구구단**

그림 6-9 구구단에서 변수 i와 k의 추출(단, 가로 먼저 출력)

- 다음 페이지로 넘어가기 전에 줄바꿈, 긴 공백 등을 꼼꼼하게 종이에 써가면서 다시 따져보자.

2 Nested for Loops

반복문의 중첩 예시

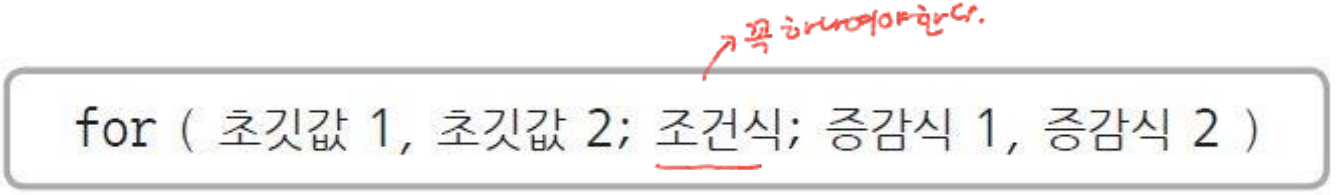
- **구구단 코드**

```
for (int i = _; i <= _; i++) {
    for (int k = _; k <= _; k++) {
        printf("%_d × %_d = %_d_", ____, ____, ____);
    }
    printf("\n");
}
```

- _의 빈 칸을 스스로 채워보고 실행시켜봐.

여러 개의 초기화식, 증감식을 사용하는 for 반복문

- **여러 개의 초깃값과 증감식을 사용하는 for문**
  - 초깃값이 하나일 필요 없음
  - 초깃값이 여러 개일 때는 콤마(,)로 구분
  - 증감식도 하나 이상 사용 가능



- **갑자기 등장한 , Operator의 역할**
  - "그리고"의 의미가 아님
  - 우리가 작성하는 명령과 연산의 결과가 나오는 과정을 디테일하게 파악하고 있어야 보인다.

  - , Operator의 결과물: 이전의 연산 결과를 _____ 한다.

3 Various Applications

초기화식이 없는 for 반복문

- 결과가 같은 다른 형식의 for문

❶ 기본 형식

```
int i;
for ( i = 0 ; i < 10 ; i ++ )
{
   printf ("%d \n", i) ;
}
```

❷ 초깃값 빼기

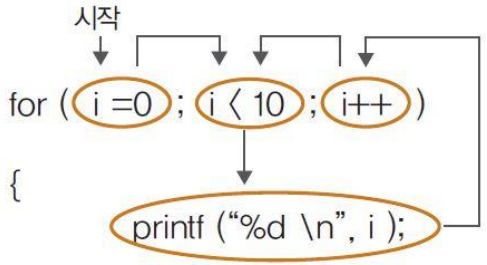
```
int i;
i = 0;
for ( ___ ; i < 10 ; i ++ )
{
   printf ("%d \n", i) ;
}
```

❸ 초깃값과 증감식 빼기

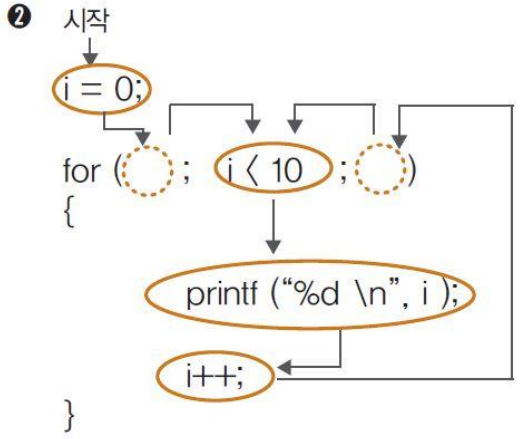
```
int i;
i = 0;
for ( ___ ; i < 10 ; ___ )
{
   printf ("%d \n", i) ;
   i ++ ;
}
```

- 초기식이 없더라도 그 자리는 반드시 세미콜론(;)으로 구분

❶



❷



❶ 시작 → i=0 → i<10 → printf( ) → i++ → i<10 → printf( ) → i++ → ···

❷ 시작 → i=0 → 빈칸 → i<10 → printf( ) → i++ → 빈칸 → i<10 → printf() → i++ → ···

초기화식이 없는 for 반복문

- **무조건 for 반복문**
  - 조건식란을 지우면 항상 참으로 인식된다.
  - 사용하지 마라.
  - 어쩔 수 없이 이렇게 만들거면 차라리 while을 써라.

Summary

Looping Statements with Progression

- **수열의 개념과 함께 이해해야 하는 반복문**
  - 가장 쉬운 수열은 정의역의 최소값에서 출발하는 등차수열
  - 예: { 0, 1, 2, 3, … }
  - 다른 말로 Loop Counter
  - for 반복문에서는 Loop Counter 값이 어떻게 변화하는지 잘 관찰해야 한다.

- **제어문은 동일하게 사용 가능하다**
  - 탈출문: break
  - 생략문: continue
  - 반환문: return

- **코딩 규칙 (공통)**