

## Chapter 06

# 반복문의 기본, for문



# 목차

1. 단순 for문
2. 중첩 for문
3. 다양한 for문의 형태

01

단순 for문

# 1. 단순 for문

## 1. for문의 이해

- For문은 여러 번 수행해야 할 작업을 한 번에 해결해주는 반복문
- 주로 for문과 while문이 사용
- 반복문이란 말 그대로 문장을 '반복'하게 만들어주는 것

“ 안녕하세요? 빙글빙글 for 문을 공부중입니다. ^^ ”

“ 안녕하세요? 빙글빙글 for 문을 공부중입니다. ^^ ”

“ 안녕하세요? 빙글빙글 for 문을 공부중입니다. ^^ ”

“ 안녕하세요? 빙글빙글 for 문을 공부중입니다. ^^ ”

“ 안녕하세요? 빙글빙글 for 문을 공부중입니다. ^^ ”

# 1. 단순 for문

## 1. for문의 이해

### 기본 6-1 같은 문장을 반복 출력하는 예

6-1.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     printf("안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^\\n");
06     printf("안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^\\n");
07     printf("안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^\\n");
08     printf("안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^\\n");
09     printf("안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^\\n");
10 }
```

—— 내용을 출력한다.

#### 실행 결과

```
안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^
안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^
안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^
안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^
안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^
```

# 1. 단순 for문

## 1. for문의 이해

### 기본 6-2 기본 for문 사용 예

6-2.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int i;
06
07     for(i=0; i < 5 ; i++)
08     {
09         printf("안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^\\n");
10     }
11 }
```

—— for문을 사용해서  
다섯 번 반복한다.

#### 실행 결과

```
안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^
안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^
안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^
안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^
안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^
```

# 1. 단순 for문

## 2. for문의 개념과 활용

- for문의 형식을 보면 괄호 안에 초기값, 조건식, 증감식이 세미콜론(;)으로 구분됨
- 그리고 중괄호({ }) 안에 반복할 문장들이 있음
- 반복할 문장이 하나뿐이라면 중괄호를 생략해도 무관
- [그림 6-1]을 통해 반복되는 순서를 보면 ❶, ❷를 수행한 뒤 ❸, ❹, ❷ 순서로 반복할 문장들을 계속 수행

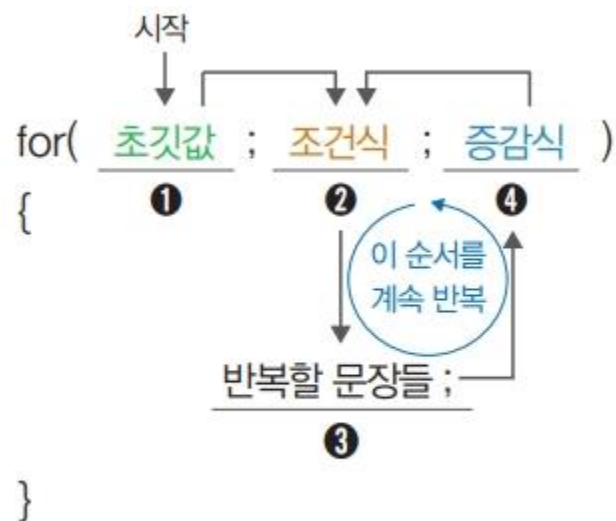


그림 6-1 for문의 기본 형식

# 1. 단순 for문

## 2. for문의 개념과 활용



그림 6-2 for문의 개념과 실제 사용

- 1 for문을 사용하려면 무조건 변수를 하나 준비
- 2 사용할 변수의 초기값을 0으로 설정
- 3 5가 될 때까지를 검사하기 위해 조건을  $i < 5$ 로 설정
- 4  $i++$ 는  $i=i+1$ 과 동일한 역할
- 5 실제로 반복되는 내용



# 1. 단순 for문

## 2. for문의 개념과 활용

- [그림 6-2]의 화살표를 따라 계속 수행하면 초깃값은 한 번만 실행되고 나머지가 계속 반복

초깃값 → 조건식 → 반복할 문장들 → 증감식 → 조건식 → 반복할 문장들 → 증감식 → 조건식 → 반복할  
문장들 → 증감식 → 조건식 ...

- [그림 6-1]과 비교하면 조건식, 반복할 문장들, 증감식만 빙글빙글 돌면서 반복되는 구조

# 1. 단순 for문

## 2. for문의 개념과 활용

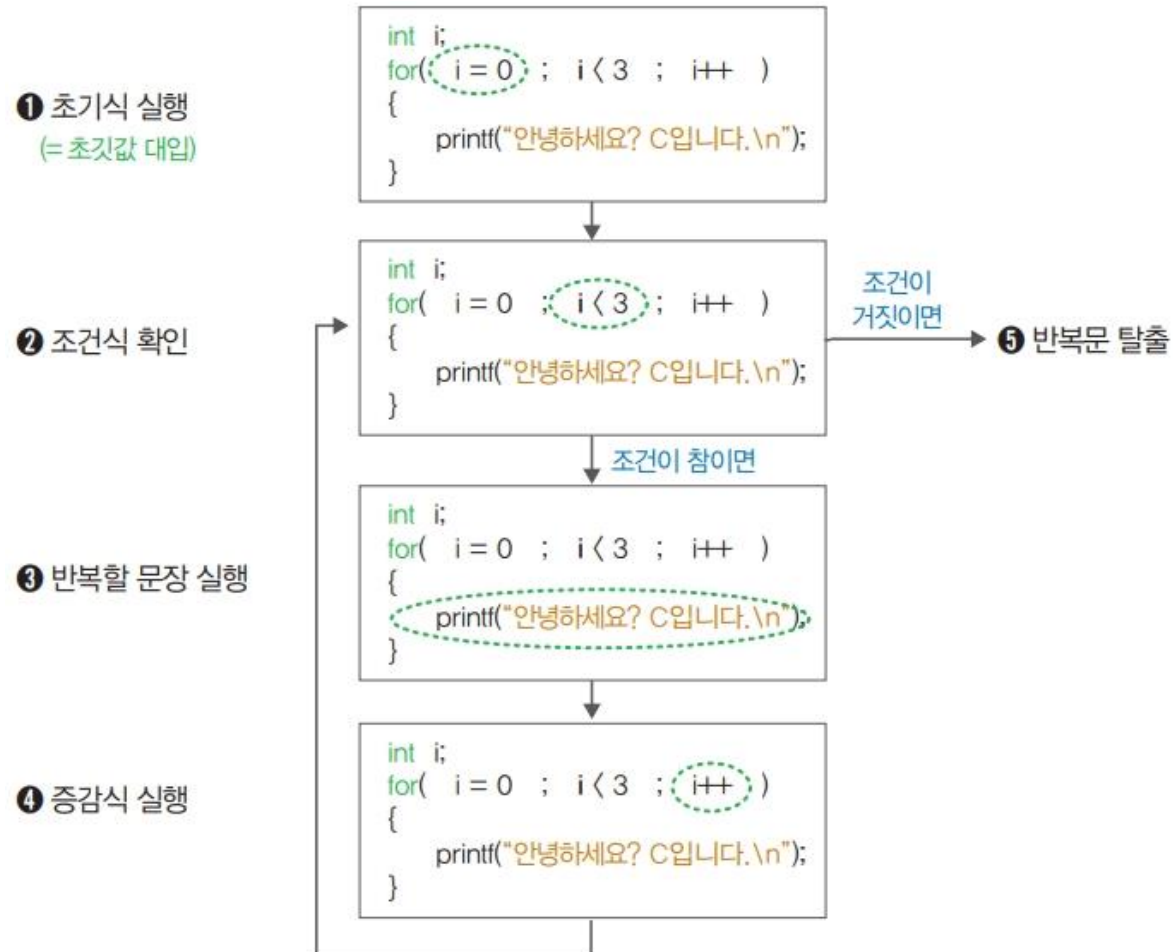


그림 6-3 for문이 반복되는 순서

# 1. 단순 for문

## 2. for문의 개념과 활용

▶ 제1회: ❶ 초기식을 수행한다(현재  $i=0$ ).

▶ 제2회: ❷ 조건식을 확인한다. 현재  $i$  값이 0이므로  $i < 3$ 은 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  $i < 3$ 은 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  $i < 3$ 은 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  $i < 3$ 이 거짓이다.

▶ 제12회: 조건식이 거짓이므로 ❺ 반복문을 탈출해 반복문 블록({ }) 밖의 내용을 수행한다.

# 1. 단순 for문

## 2. for문의 개념과 활용

- 초기식이  $i = 0$ 이 아니라  $i = 3$ 이라면 조건식이  $i < 3$ 이므로 거짓이 되어 바로 ❸를 수행하므로 printf문은 한 번도 수행되지 않음
- 조건식을 확인한 후 증감식( $i++$ )을 바로 수행하지 않고 반복할 문장을 먼저 수행한 후 증감식을 수행

```
int i;  
for( i=0; i < 3; i++ )  
    printf("빙글빙글 for문입니다. ^\n");
```

==

```
int i;  
for( i=0; i < 3; i++ )  
{  
    printf("빙글빙글 for문입니다. ^\n");  
}
```

# 1. 단순 for문

## 2. for문의 개념과 활용

기본 6-3 for문과 블록 사용 예

6-3.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int i;
06     for( i=0; i < 3; i++ )
07     {
08         printf("안녕하세요? \n");
09         printf("##또 안녕하세요?## \n");
10     }
11
12     printf("\n\n");
13
14     for( i=0; i < 3; i++ )
15         printf("안녕하세요? \n");
16         printf("##또 안녕하세요?## \n");
17 }
```

블록을 사용한 for문이다.

블록을 사용하지 않은 for문이다.

### 실행 결과

```
안녕하세요?
##또 안녕하세요?##
안녕하세요?
##또 안녕하세요?##
안녕하세요?
##또 안녕하세요?##
```

```
안녕하세요?
안녕하세요?
안녕하세요?
##또 안녕하세요?##
```

# 1. 단순 for문

## 2. for문의 개념과 활용

기본 6-4 for문 사용 예 1

6-4.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int i;
06
07     for( i=5; i > 0; i-- )      ----- 초깃값, 조건식, 증감식을 수정한다.
08     {
09         printf("%d : 안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^\\n", i);
10     }
11 }
```

### 실행 결과

```
5 : 안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^
4 : 안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^
3 : 안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^
2 : 안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^
1 : 안녕하세요? 빙글빙글 for문을 공부 중입니다. ^^
```

# 1. 단순 for문

## 2. for문의 개념과 활용

기본 6-5 for문 사용 예 2

6-5.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int i;
06
07     for( i=1; i <= 5; i++ ) ----- i 값이 1부터 5까지 변경된다.
08     {
09         printf("%d \n", i);
10     }
11 }
```

실행 결과

1  
2  
3  
4  
5

# 1. 단순 for문

## 2. for문의 개념과 활용

기본 6-6 for문을 활용하지 않고 합계 구하기

6-6.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int hap;
06
07     hap = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10;
08
09     printf(" 1에서 10까지의 합: %d \n", hap);
10 }
```

----- hap에 1부터 10까지 더해서 입력한다.

실행 결과

1에서 10까지의 합: 55

- 반복적인 덧셈을 수행할 때는 for문을 활용해야 함



# 1. 단순 for문

## 2. for문의 개념과 활용

기본 6-7 for문을 활용하여 합계 구하기 1

6-7.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int hap;           — 합계를 누적할 변수를 선언한다.
06     int i;             — 1부터 10까지 변하는 변수를 선언한다.
07
08     for( i=1; i <= 10; i++ ) { — for문에 의해 1부터 10까지 열 번 반복된다.
09         hap = hap + i;       — hap 변수에 1부터 10까지 반복해서 누적된다.
10     }
11
12     printf(" 1에서 10까지의 합: %d \n", hap);
13 }
```

### 실행 결과 ▼

오류 목록					
전체 솔루션					
1 오류 1 경고 0 메시지 빌드 - IntelliSense 검색 오류 목록					
코드	설명	프로젝트	파일	줄	비표시 오류(Suppress)
C6001	초기화되지 않은 메모리 'hap'을(를) 사용하고 있습니다.	Sample	Sample.c	9	
C4700	초기화되지 않은 'hap' 지역 변수를 사용했습니다.	Sample	Sample.c	9	

# 1. 단순 for문

## 2. for문의 개념과 활용

기본 6-8 for문을 활용하여 합계 구하기 2

6-8.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int hap = 0;          ----- 합계를 누적할 변수를 선언하고 0으로 초기화한다.
06     int i;
07
08     for( i=1; i <= 10; i++ ) {
09         hap += i;          ----- hap 변수에 1부터 10까지 반복해서 누적한다.
                             hap = hap+i와 동일하다.
10     }
11
12     printf(" 1에서 10까지의 합: %d \n", hap);
13 }
```

실행 결과

1에서 10까지의 합: 55

# 1. 단순 for문

## 2. for문의 개념과 활용

- [기본 6-8]에서 주의할 것은 변수  $i$ 와  $hap$ 의 값
- [기본 6-8]의 5행에서  $hap$ 을 0으로 초기화하고 8행에서  $i$  값을 1로 초기화
- 그리고 첫 번째 반복에서  $i$  값 1을 현재  $hap$ 의 값인 0과 더해 다시  $hap$ 에 넣음
- 두 번째 반복에서는  $i$  값 2를 현재  $hap$ 의 값인 1과 더해 다시  $hap$ 에 넣음
- 이런 과정이 열 번 반복되면 1~10을 더한 값이  $hap$ 에 들어감

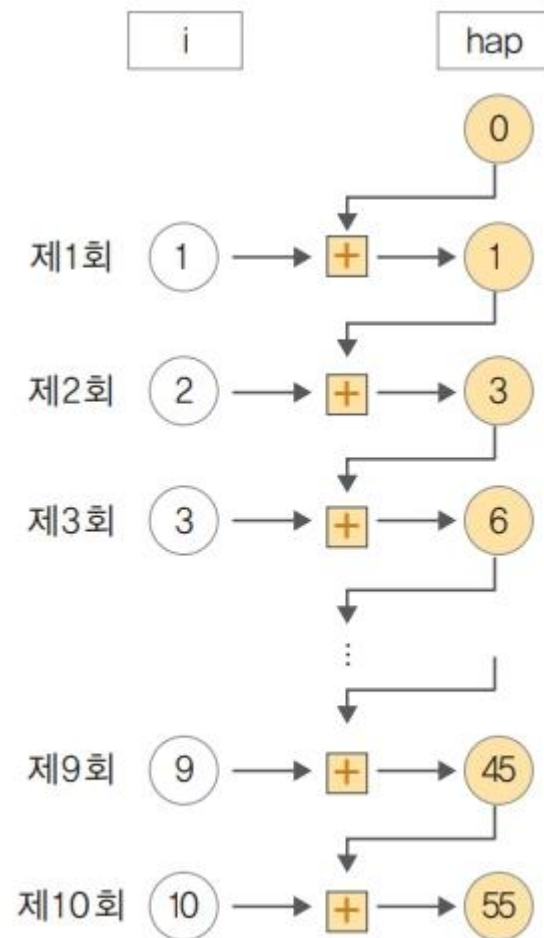


그림 6-4 변수  $i$ 와  $hap$ 의 변화

# 1. 단순 for문

## 2. for문의 개념과 활용

응용 6-9 for문을 활용하여 합계 구하기 3

6-9.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int hap = 0;
06     int i;
07
08     for( __1__ ; i <= 1000; __2__ ) {
09         hap = hap + i;
10     }
11
12     printf(" 500에서 1000까지의 홀수의 합: %d \n", hap);
13 }
```

—— i를 501부터 2씩 증가시킨다.

실행 결과

500에서 1000까지의 홀수의 합: 187500

2=+1 극5 2+=1 2 105=1 1 135

# 1. 단순 for문

## 2. for문의 개념과 활용

기본 6-10 for문을 활용하여 합계 구하기 4

6-10.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05     int hap=0;           —— 합계를 누적할 변수를 선언하고 0으로 초기화한다.
06     int i;               —— 1씩 증가시킬 변수이다.
07     int num;             —— 입력받을 최종값이다.
08
09     printf(" 값 입력 : ");
10     scanf("%d", &num);   —— 최종값을 입력한다.
11
12     for( i=1; i <= num; i++ ) { —— 1부터 최종값까지 1씩 증가시키며 반복한다.
13         hap = hap + i;
14     }
15
16     printf(" 1에서 %d까지의 합: %d \n", num, hap);
17 }
```

실행 결과

값 입력 : 100

1에서 100까지의 합: 5050

# 1. 단순 for문

## 2. for문의 개념과 활용

응용 6-11 for문을 활용하여 합계 구하기 5

6-11.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05     int hap=0;
06     int i;
07     int num1, num2, num3;
08
09     printf(" 시작값, 끝값, 증가값 입력 : ");
10     scanf("%d %d %d", &num1, &num2, &num3);
11
12     for( __1__ ) {
13         hap = hap + i;
14     }
15
16     printf(" %d에서 %d까지 %d씩 증가한 값의 합: %d \n", num1, num2, num3, hap);
17 }
```

입력받을 변수 3개를 선언한다.

공백 문자로 구분해서 3개의 수를 입력받는다.

시작값은 num1, 끝값은 num2, 증가값은 num3이다.

**실행 결과**

시작값, 끝값, 증가값 입력 : 2 300 3  
2에서 300까지 3씩 증가한 값의 합: 15050

εwnu+τ=τ :Zwnu => τ :Jwnu=τ ■ 〰음

# 1. 단순 for문

## 2. for문의 개념과 활용

기본 6-12 for문을 사용한 구구단 프로그램

6-12.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05     int i;
06     int dan;
07
08     printf(" 몇 단 ? ");
09     scanf("%d", &dan);
10
11     for( i=1; i <= 9; i++ ) {
12         printf(" %d X %d = %d \n", dan, i, dan*i);
13     }
14 }
```

—— 계산한 단을 입력받을 변수를 선언한다.

—— 계산할 단을 입력받는다.

—— 1부터 9까지 반복하며 입력한 단을 출력한다.

### 실행 결과

몇 단 ? 7  
7 X 1 = 7  
7 X 2 = 14  
7 X 3 = 21  
7 X 4 = 28  
7 X 5 = 35  
7 X 6 = 42  
7 X 7 = 49  
7 X 8 = 56  
7 X 9 = 63

02

중첩 for문



## 2. 중첩 for문

### 1. 중첩 for문의 개념

- 중첩 for문은 for문 내부에 또 다른 for문이 들어 있는 형태

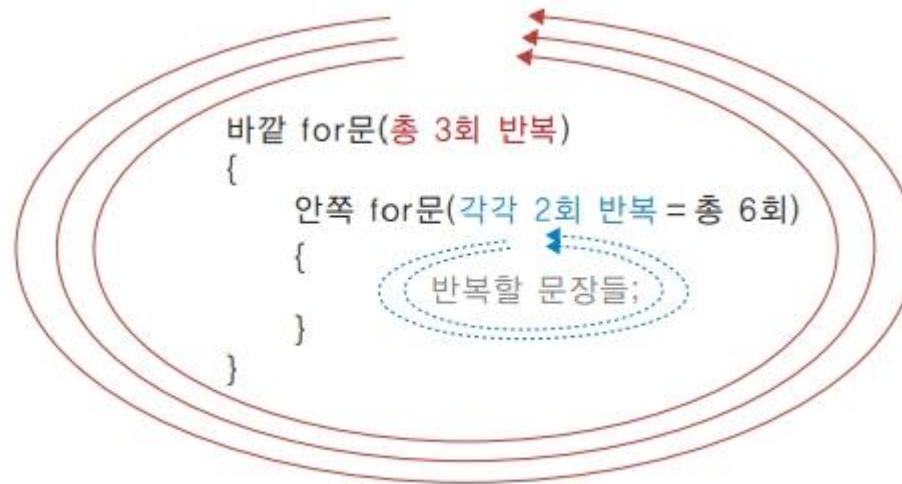


그림 6-5 중첩 for문의 동작 개념

## 2. 중첩 for문

### 1. 중첩 for문의 개념

- 중첩 for문은 for문 내부에 또 다른 for문이 들어 있는 형태

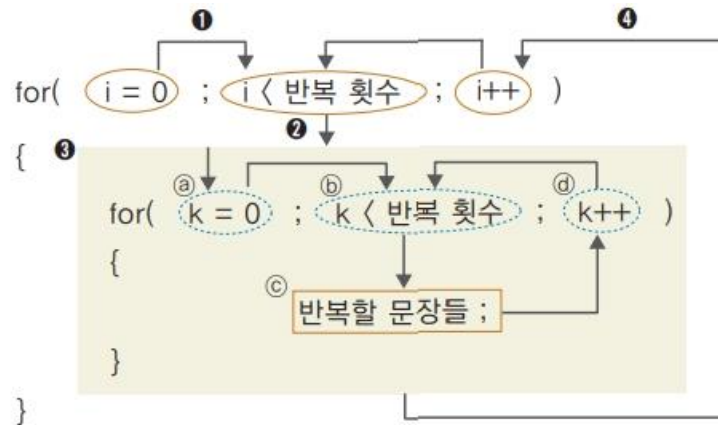


그림 6-6 중첩 for문의 작동 방식

- 예를 들어 바깥 for문의 반복 횟수를 3, 안쪽 for문의 반복 횟수를 2라고 가정하면 다음과 같은 순서로 작동

① → ② → ③ → (a → b → c → d → b → c → d → b → 안쪽 for문을 빠져나감) → ④ → ②  
→ ③ → (a → b → c → d → b → c → d → b → 안쪽 for문을 빠져나감) → ④ → ② → ③ →  
(a → b → c → d → b → c → d → b → 안쪽 for문을 빠져나감) → ④ → ② → 바깥 for문을 빠져나감

## 2. 중첩 for문

### 1. 중첩 for문의 개념

기본 6-13 중첩 for문 사용 예 1

6-13.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int i, k;          ----- 반복할 변수 i, k를 선언한다.
06
07     for( i=0; i < 3; i++ ) ----- 바깥 for문을 세 번 반복한다.
08     {
09         for( k=0; k < 2; k++ ) ----- 안쪽 for문을 두 번 반복한다.
10         {
11             printf("중첩 for문입니다. (i값: %d, k값: %d)\n", i, k);
12         }              ----- i와 k 값을 총 여섯(3×2) 번 출력한다.
13     }
14 }
```

## 2. 중첩 for문

### 1. 중첩 for문의 개념

- 중첩 for문의 실행 횟수는 '바깥 for문 반복 횟수 × 안쪽 for문 반복 횟수'

❶ 외부 for문 제1회: 7행의  $i$ 를 0으로 초기화하면  $i < 3$ 이 참이므로 바깥 for문 수행  
내부 for문 제1회: 9행의  $k$ 를 0으로 초기화하면  $k < 2$ 가 참이므로 안쪽 for문 수행  
11행의 `printf( )`를 실행해서 '중첩 for문입니다.' 출력  
내부 for문 제2회: 9행의  $k++$ 로  $k$ 를 1로 증가시키면  $k < 2$ 가 참이므로 안쪽 for문 수행  
11행의 `printf( )`를 실행해서 '중첩 for문입니다.' 출력  
내부 for문 제3회: 9행의  $k++$ 로  $k$ 를 2로 증가시키면  $k < 2$ 가 거짓이므로 안쪽 for문 종료

❷ 외부 for문 제2회: 7행의  $i++$ 로  $i$ 를 1로 증가시키면  $i < 3$ 이 참이므로 바깥 for문 수행  
내부 for문 제1회: 9행의  $k$ 를 0으로 초기화하면  $k < 2$ 가 참이므로 안쪽 for문 수행  
11행의 `printf( )`를 실행해서 '중첩 for문입니다.' 출력  
내부 for문 제2회: 9행의  $k++$ 로  $k$ 를 1로 증가시키면  $k < 2$ 가 참이므로 안쪽 for문 수행  
11행의 `printf( )`를 실행해서 '중첩 for문입니다.' 출력  
내부 for문 제3회: 9행의  $k++$ 로  $k$ 를 2로 증가시키면  $k < 2$ 가 거짓이므로 안쪽 for문 종료

❸ 외부 for문 제3회: 7행의  $i++$ 로  $i$ 를 2로 증가시키면  $i < 3$ 이 참이므로 바깥 for문 수행  
내부 for문 제1회: 9행의  $k$ 를 0으로 초기화하면  $k < 2$ 가 참이므로 안쪽 for문 수행  
11행의 `printf( )`를 실행해서 '중첩 for문입니다.' 출력  
내부 for문 제2회: 9행의  $k++$ 로  $k$ 를 1로 증가시키면  $k < 2$ 가 참이므로 안쪽 for문 수행  
11행의 `printf( )`를 실행해서 '중첩 for문입니다.' 출력  
내부 for문 제3회: 9행의  $k++$ 로  $k$ 를 2로 증가시키면  $k < 2$ 가 거짓이므로 안쪽 for문 종료

❹ 외부 for문 제4회: 7행의  $i++$ 로  $i$ 를 3으로 증가시키면  $i < 3$ 이 거짓이므로 바깥 for문 종료

## 2. 중첩 for문

### 1. 중첩 for문의 개념

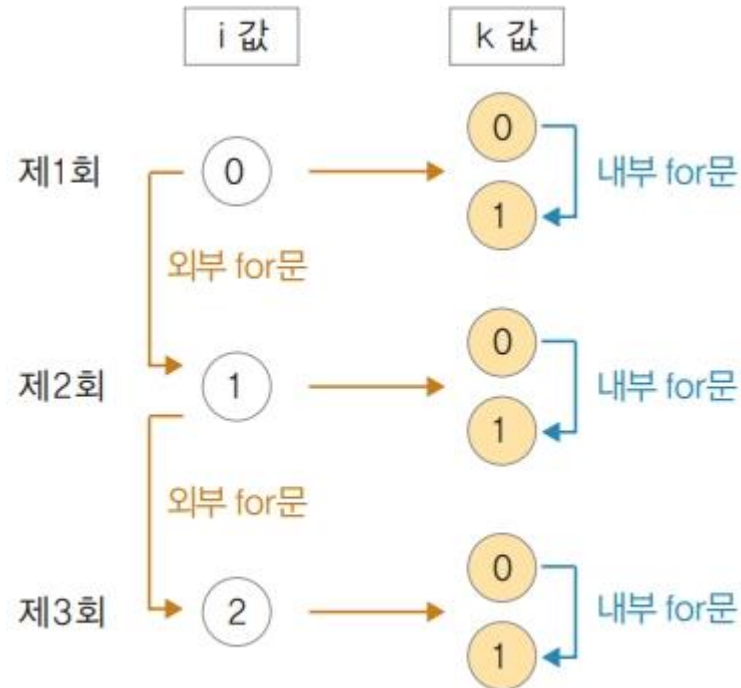


그림 6-7 중첩 for문에서 i 값과 k 값의 변화

## 2. 중첩 for문

### 2. 중첩 for문의 활용

- 지금까지 연습한 중첩 for문을 활용하여 구구단을 2단부터 9단까지 출력
- [그림 6-8]을 보고 변수 i와 k를 추출



그림 6-8 구구단에서 변수 i와 k의 추출

## 2. 중첩 for문

### 2. 중첩 for문의 활용

응용 6-14 중첩 for문 사용 예 2

6-14.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int i, k;
06
07     for( i = 2; i <= 9; i ++ ) ----- 2~9단까지 반복한다.
08     {
09         for( __1__ ) ----- 각 단의 뒷자리는 1~9를 반복한다.
10         {
11             printf(" %d X %d = %d \n", i, k, i*k); ----- 구구단을 출력한다.
12         }
13         printf("\n"); ----- 각 단이 끝나면 한 줄을 띄운다.
14     }
15 }
```

++ k : 6 => k : 1 = k 1 7 8 9

## 2. 중첩 for문

### 2. 중첩 for문의 활용

#### 실행 결과

```
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
2 X 7 = 14
2 X 8 = 16
2 X 9 = 18

3 X 1 = 3
3 X 2 = 6
  ⋮
  ⋮
```

```
  ⋮
  ⋮
8 X 7 = 56
8 X 8 = 64
8 X 9 = 72

9 X 1 = 9
9 X 2 = 18
9 X 3 = 27
9 X 4 = 36
9 X 5 = 45
9 X 6 = 54
9 X 7 = 63
9 X 8 = 72
9 X 9 = 81
```



## 2. 중첩 for문

### 2. 중첩 for문의 활용

- [응용 6-14]의 실제 수행 결과는 세로로 출력되므로 위아래로 스크롤해야 전체 결과를 볼 수 있음

2~9로 증가 후 종료(안쪽 for문: 변수 k)

1증가      1증가      1증가      1증가

② × 1 = 2	③ × 1 = 3	④ × 1 = 4	⑨ × 1 = 9	제1회
2 × 2 = 4	3 × 2 = 6	4 × 2 = 8	9 × 2 = 18	제2회
2 × 3 = 6	3 × 3 = 9	4 × 3 = 12	9 × 3 = 27	제3회
2 × 4 = 8	3 × 4 = 12	4 × 4 = 16	9 × 4 = 36	제4회
2 × 5 = 10	3 × 5 = 15	4 × 5 = 20	9 × 5 = 45	제5회
2 × 6 = 12	3 × 6 = 18	4 × 6 = 24	9 × 6 = 54	제6회
2 × 7 = 14	3 × 7 = 21	4 × 7 = 28	9 × 7 = 63	제7회
2 × 8 = 16	3 × 8 = 24	4 × 8 = 32	9 × 8 = 72	제8회
2 × 9 = 18	3 × 9 = 27	4 × 9 = 36	9 × 9 = 81	제9회

1~9로 증가 후 종료(바깥 for문: 변수 i)

그림 6-9 구구단에서 변수 i와 k의 추출(단, 가로 먼저 출력)

## 2. 중첩 for문

### 2. 중첩 for문의 활용

응용 6-15 중첩 for문 사용 예 3

6-15.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int i, k;
06
07     for( i = 1; i <= 9; i ++ )      —— 각 단의 뒷자리는 1~9를 반복한다.
08     {
09         for( k = 2; k <= 9; k ++ )  —— 2~9단까지 반복한다.
10         {
11             printf("%2dX%2d=%2d", i, k ); —— 각 단별로 한 줄씩 출력한다.
12         }
13         printf("\n");                —— 각 단의 한 줄을 출력한 후 다음 줄로 넘긴다.
14     }
15 }
```

03

## 다양한 for문의 형태

### 3. 다양한 for문의 형태

#### 1. 여러 개의 초깃값과 증감식을 사용하는 for문

- for문에 들어가는 초깃값이 꼭 하나일 필요는 없음
- 단, 여러 개를 초기화할 때는 콤마(,)로 각 각을 구분
- for( 초깃값 1, 초깃값 2; 조건식; 증감식 1, 증감식 2 )

기본 6-16 for문의 다양한 활용 예 1

6-16.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int i, k;
06
07     for( i = 1, k = 1; i <= 9; i++, k++ )
08         printf(" %d X %d = %d \n", i, k, i*k);
09 }
```

----- 반복할 변수 i, k를 선언한다.

----- 초깃값과 증감식이 2개이다.

#### 실행 결과

1 X 1 = 1  
2 X 2 = 4  
3 X 3 = 9  
4 X 4 = 16  
5 X 5 = 25  
6 X 6 = 36  
7 X 7 = 49  
8 X 8 = 64  
9 X 9 = 81

### 3. 다양한 for문의 형태

#### 2. 초깃값과 증감식이 없는 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

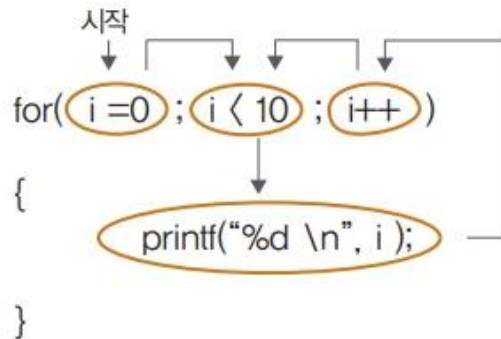
```
int i;  
i = 0;  
for( _____ ; i < 10 ; _____ )  
{  
    printf("%d \n", i);  
    i ++ ;  
}
```

- 0~9를 출력하는 기본 형식은 ❶
- 이때 ❷와 같이 초깃값인 i=0을 for문 밖으로 빼서 사용할 수도 있음
- ❸과 같이 초깃값과 증감식을 모두 빼서 사용할 수도 있음

### 3. 다양한 for문의 형태

#### 2. 초깃값과 증감식이 없는 for문

❶ 기본 형식



❷ 초깃값과 증감식 빼기

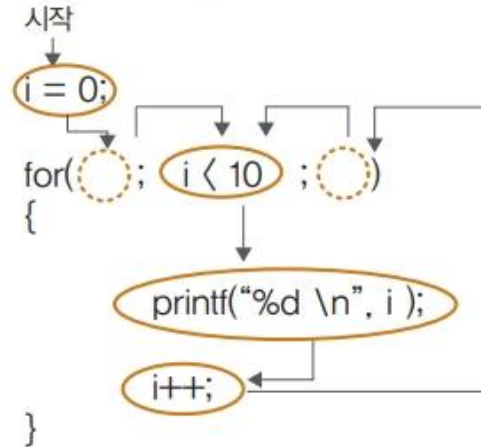


그림 6-10 두 소스의 비교

- 왼쪽(❶)과 오른쪽(❷) 소스의 실행 순서를 살펴보면 빈칸은 없는 것과 마찬가지이므로 결국 동일한 순서로 작동

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++ → ...

### 3. 다양한 for문의 형태

#### 2. 초깃값과 증감식이 없는 for문

기본 6-17 for문의 다양한 활용 예 2

6-17.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int i;
06     i = 0;
07     for( ; ; )
08     {
09         printf("%d \n", i);
10         i ++;
11     }
12 }
```

—— 초깃값, 조건식, 증감식이 아무것도 없다.

실행 결과

0  
1  
2  
3  
...  
...  
1000  
1001  
1002  
1003  
...  
...  
...  
무한반복

### 3. 다양한 for문의 형태

#### 2. 초깃값과 증감식이 없는 for문

응용 6-18 for문의 다양한 활용 예 3

6-18.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05     int a, b;
06
07     for( ; ; ) ——— 무한 루프가 발생한다.
08     {
09         printf("더할 두 수 입력 (멈추려면 Ctrl+C) : ");
10         scanf("%d %d", &a, &b); ——— 두 값을 입력받는다.
11
12         printf("%d + %d = %d \n", a, b, a+b); ——— 더하기 결과를 출력한다.
13     }
14 }
```

10.4 for



### 3. 다양한 for문의 형태

#### 2. 초깃값과 증감식이 없는 for문

##### 실행 결과

더할 두 수 입력 (멈추려면 Ctrl+C) : 11 33

$11 + 33 = 44$

더할 두 수 입력 (멈추려면 Ctrl+C) : 22 755

$22 + 755 = 777$

더할 두 수 입력 (멈추려면 Ctrl+C) :

- 7행은 for문 내부의 블록을 무한 루프에 빠지게 하는 구문
- 9행에서 설명문을 출력하고 10 행에서 두 수를 공백으로 구분해서 입력하면 12행에서 계산 결과가 출력
- Ctrl+C를 누를 때까지 반복

\*

예제 모음

## [예제모음 14] 구구단 출력 프로그램

**예제 설명** 중첩 for문을 사용하여 제목과 구구단을 출력하는 프로그램이다.

**실행 결과**

#제2단# #제3단# #제4단# #제5단# #제6단# #제7단# #제8단# #제9단#

```
2X 1= 2 3X 1= 3 4X 1= 4 5X 1= 5 6X 1= 6 7X 1= 7 8X 1= 8 9X 1= 9
2X 2= 4 3X 2= 6 4X 2= 8 5X 2=10 6X 2=12 7X 2=14 8X 2=16 9X 2=18
2X 3= 6 3X 3= 9 4X 3=12 5X 3=15 6X 3=18 7X 3=21 8X 3=24 9X 3=27
2X 4= 8 3X 4=12 4X 4=16 5X 4=20 6X 4=24 7X 4=28 8X 4=32 9X 4=36
2X 5=10 3X 5=15 4X 5=20 5X 5=25 6X 5=30 7X 5=35 8X 5=40 9X 5=45
2X 6=12 3X 6=18 4X 6=24 5X 6=30 6X 6=36 7X 6=42 8X 6=48 9X 6=54
2X 7=14 3X 7=21 4X 7=28 5X 7=35 6X 7=42 7X 7=49 8X 7=56 9X 7=63
2X 8=16 3X 8=24 4X 8=32 5X 8=40 6X 8=48 7X 8=56 8X 8=64 9X 8=72
2X 9=18 3X 9=27 4X 9=36 5X 9=45 6X 9=54 7X 9=63 8X 9=72 9X 9=81
```

## [예제모음 14] 구구단 출력 프로그램

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int i, k;
06
07     for( i = 2; i <= 9; i++ )
08         printf(" #제%d단#", i);      —— 맨 위에 단의 제목을 출력한다.
09
10     printf("\n\n");                  —— 두 줄을 띄운다.
11
12     for( i = 1; i <= 9; i++ )        —— 중첩 for문으로 구구단을 출력한다.
13     {
14         for( k = 2; k <= 9; k++ )
15         {
16             printf("%2dX%2d=%2d", k, i, k*i);
17         }
18         printf("\n");
19     }
20 }
```

## [예제모음 15] 아스키코드표 출력 프로그램

**예제 설명** for문과 if문을 사용하여 아스키코드의 0~127을 10진수, 16진수, 문자로 출력하는 프로그램이다.

### 실행 결과

-----  
10진수 16진수 문자  
-----

0	0	
1	1	┐
2	2	┌
3	3	
4	4	└
5	5	
6	6	-
7	7	
	⋮	
	⋮	

	⋮	
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7a	z
123	7b	{
124	7c	
125	7d	}
126	7e	~
127	7f	

## [예제모음 15] 아스키코드표 출력 프로그램

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int i;
06
07     for( i= 0; i < 128; i++ ) —— 0~127을 처리한다.
08     {
09         if( i%16 == 0 ) —— 16행마다 제목 줄을 출력한다.
10         {
11             printf("—————\n");
12             printf("10진수  16진수  문자 \n");
13             printf("—————\n");
14         }
15         printf("%5d %5x %5c\n", i, i, i); —— i 값을 10진수, 16진수, 문자로 출력한다.
16     }
17 }
```

## [예제모음 16] 입력한 문자를 거꾸로 출력

**예제 설명** 입력된 영문자나 숫자를 for문을 사용하여 반대 순서로 출력하는 프로그램이다.

**실행 결과**

영문자 및 숫자를 입력 (100자 이하) : IT\_CookBook\_C\_Language

입력한 문자열 ==> IT\_CookBook\_C\_Language

변환된 문자열 ==>egaugnaL\_C\_kooBkooC\_TI

## [예제모음 16] 입력한 문자를 거꾸로 출력

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <string.h>
03 #include <stdio.h>
04 void main( )
05 {
06     char str[100];           —— 입력받을 문자 배열이다.
07     int str_cnt;             —— 입력한 문자의 개수를 저장할 변수이다.
08     int i;
09
10     printf("영문자 및 숫자를 입력 (100자 이하) : ");
11     scanf("%s", str);        —— 최대 99자까지 문자를 입력한다.
12
13     printf("\n");
14     printf("입력한 문자열 ==> %s\n", str); —— 입력된 문자열을 출력한다.
15     printf("변환된 문자열 ==>");
16
17     str_cnt = strlen(str);    —— 입력된 문자의 개수를 계산한다.
18
19     for( i= str_cnt; i >=0; i-- ) —— 입력된 개수만큼 반대 순서로 출력한다.
20     {
21         printf("%c", str[i]);
22     }
23
24     printf("\n");
25 }
```



감사합니다!

