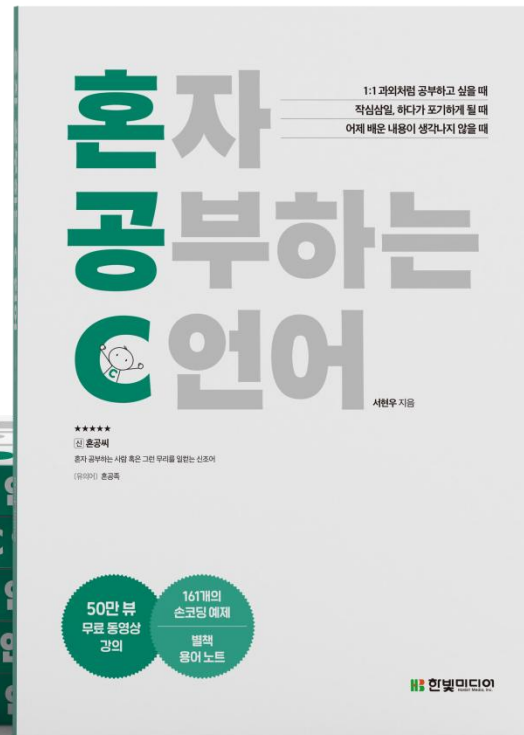


6장 반복문



06- 1

while문, for문, do ~ while문

❖ while문 (1/2)

while문을 사용한 반복문

소스 코드 예제6-1.c

```
01 #include <stdio.h>
```

```
02
```

```
03 int main(void)
```

```
04 {
```

```
05     int a = 1;
```

```
06
```

```
07     while (a < 10) // ① a가 10보다 작으므로 조건식은 참조건식은 참
```

```
08     {
```

```
09         a = a * 2; // ② a에 2를 곱해 a에 다시 저장
```

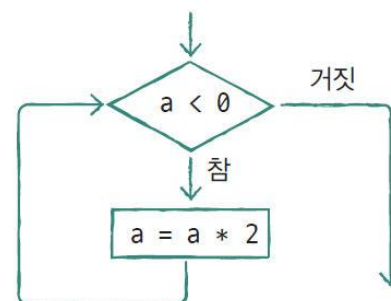
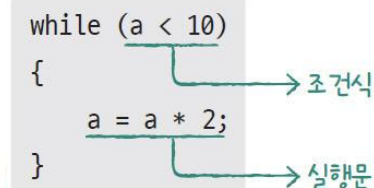
```
10     }
```

```
11     printf("a : %d\n", a); // a 값 출력
```

```
12
```

```
13     return 0;
```

```
14 }
```



실행결과
a : 16

06- 1

while문, for문, do ~ while문

❖ while문 (2/2)

- ①과 ②를 반복하는 과정

```

05    int a = 1;
06
07    while (a < 10)    // ①
08    {
09        a = a * 2;    // ②
10    }

```

5행 변수 선언 시
a = 1로 초기화

	while문 1회	while문 2회	while문 3회	while문 4회	while문 5회
조건식 a < 10	a < 10 참	a < 10 참	a < 10 참	a < 10 참	a < 10 거짓
실행문 a = a * 2	1 * 2	2 * 2	4 * 2	8 * 2	
변수 a 값	2	4	8	16	

while문 밖으로 빠져나옴

06- 1

while문, for문, do ~ while문

❖ for문

for문을 사용한 반복문

소스 코드 예제6-2.c

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int a = 1;
06     int i;
07
08     for (i = 0; i < 3; i++)
09     {
10         a = a * 2;
11     }
12     printf("a : %d\n", a);
13
14     return 0;
15 }
```

① 초기식은 최초
한 번만 실행

② 조건식이 참이면
반복할 문장 실행

④ i를 1증가시키고
다시 조건 검사

③ 반복할 문장 실행 후
증감식으로 올라갑니다.

// for문을 빠져나오면 a 값 출력

실행결과

a : 8

06- 1

while문, for문, do ~ while문

❖ do~while문

do ~ while문을 사용한 반복문

소스 코드 예제6-3.c

01 #include <stdio.h>

02

03 int main(void)

04 {

05 int a = 1;

06

07 do

08 {

09 a = a * 2;

10 } while (a < 10);

11 printf("a : %d\n", a);

12

13 return 0;

14 }

a 값이
1, 2, 4, 8일 때
반복합니다.

do // 7 ~ 10행

{

a = a * 2;

} while (a < 10);

a 값이 16이 되면
조건식이 거짓이므로
반복을 끝냅니다.

// a의 값을 2배로 늘린다.

// a가 10보다 작으면 9행을 반복

// 반복이 끝난 후 a 값 출력

실행결과

X

a : 16

키워드로 끝내는 핵심 포인트

- ❖ **while문**은 반복 문장을 실행하기 전에 조건을 먼저 검사한다.
- ❖ **for문**은 반복 횟수가 정해진 경우 사용하면 편리하다.
- ❖ **do~while문**은 반복 문장을 실행한 후에 반복 조건을 검사한다.

표로 정리하는 핵심 포인트

표 6-1 세 가지 반복문

반복문 형식	실행 방식
<pre>while (조건식) { 실행문; }</pre>	<p>조건식이 참인 동안 실행문을 반복한다. 최초 조건식이 거짓이면 실행문은 한 번도 실행되지 않는다.</p>
<pre>for (초기식; 조건식; 증감식) { 실행문; }</pre>	<p>초기식은 최초 한 번 실행한다. 조건식을 검사하여 참이면 실행문 → 증감식 → 조건식을 반복한다.</p>
<pre>do { 실행문; } while (조건식);</pre>	<p>실행문을 수행한 후에 조건을 검사한다. 조건식이 참인 동안 실행문을 반복한다. 실행문은 조건식과 관계없이 최소 한 번은 실행된다.</p>

❖ 중첩 반복문 (1/2)

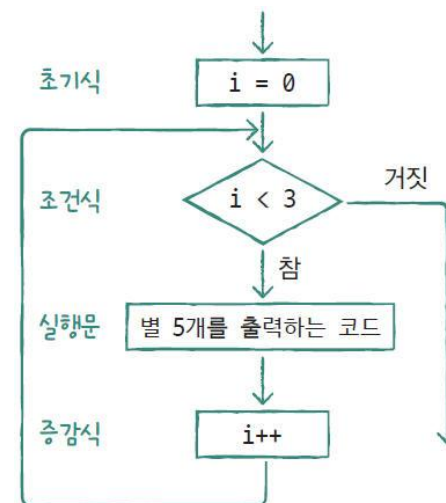
중첩 반복문을 사용한 별 출력

소스 코드 예제6-4.c

```

01 #include <stdio.h>
02
03 int main(void)
04 {
05     int i, j;                // 반복 횟수를 세기 위한 제어 변수
06
07     for (i = 0; i < 3; i++)   // i가 0부터 2까지 증가하면서 3번 반복
08     {
09         for (j = 0; j < 5; j++) // j가 0부터 4까지 증가하면서 5번 반복
10         {
11             printf("*");      // 별 출력문
12         }
13         printf("\n");         // 별을 5번 출력한 후에 줄을 바꾼다.
14     }
15
16     return 0;
17 }

```



실행결과

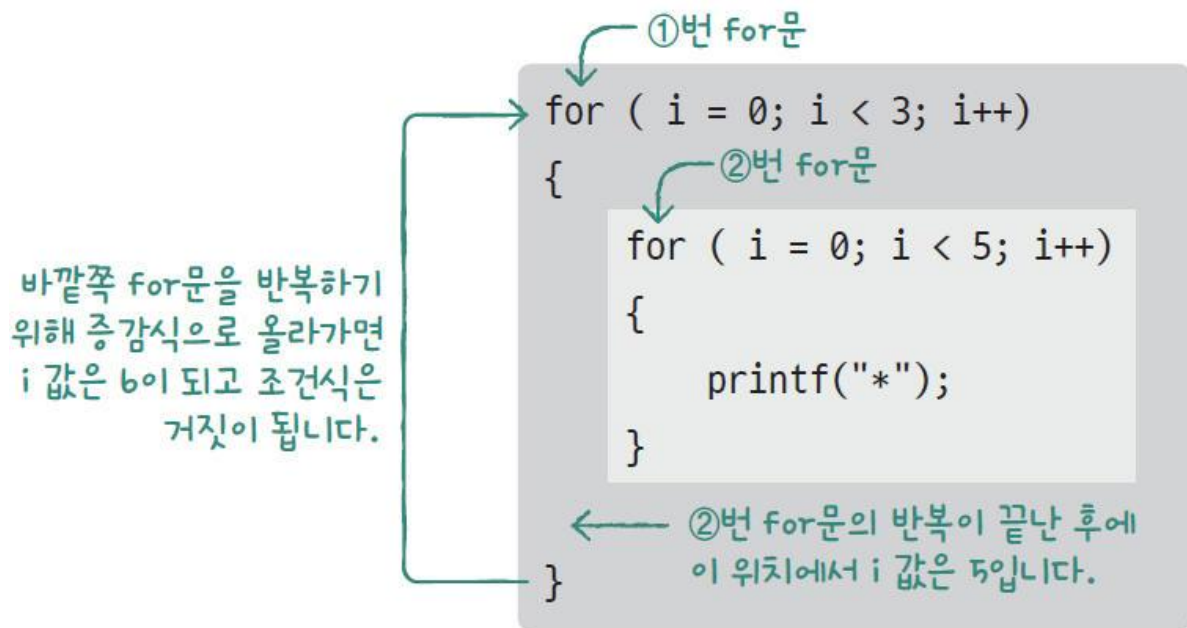
```

*****
*****
*****

```


❖ 중첩 반복문 (2/2)

- 각 반복문이 모두 다른 제어변수를 사용해야 한다.
- 같은 제어변수를 사용하면 5번만 반복한다.



❖ 구구단 출력 (1/3)

- 반복되는 기본 문장을 찾아 구현한다.

$2 * 1 = 2$ → 곱셈
수식
 $2 * 2 = 4$
 $2 * 3 = 6$
 $2 * 4 = 8$
 $2 * 5 = 10$
 $2 * 6 = 12$
 $2 * 7 = 14$
 $2 * 8 = 16$
 $2 * 9 = 18$ } $2*$ 가
9번 반복

```
printf("2 * %d = %d\n", 1, 2 * 1);
```

```
// 2 * 1 = 2를 출력
```

❖ 구구단 출력 (2/3)

- 원하는 횟수만큼 반복합니다.

```
for ( j = 0; j < 9; j++ )           // 출력문을 9번 반복
{
    printf("2 * %d = %d\n", 1, 2 * 1);    // 2 * 1 = 2를 출력
}
```

- 반복 규칙을 적용합니다.

```
for ( j = 1; j <= 9; j++ )           // 9번 반복에는 변함이 없다.
{
    printf("2 * %d = %d\n", j , 2 * j);    // j를 곱하는 값으로 활용
}
```

❖ 구구단 출력 (3/3)

- 1개 단을 출력하는 코드를 8번 반복하고 단이 바뀌도록 제어변수의 값을 조정한다.

```
for ( i = 2; i <= 9; i++ )           // i가 2에서 9까지 변하면서 8번 반복
{
    for ( j = 1; j <= 9; j++ )
    {
        printf("%d * %d = %d\n", i, j, i * j) ; // 반복 제어 변수 i와 j 활용
    }
}
```

↑
i가 2일 때 j는 1부터 9까지 변하면서 2단을 출력
i가 3일 때 j는 1부터 9까지 변하면서 3단을 출력
...
i가 9일 때 j는 1부터 9까지 변하면서 9단을 출력

❖ break 분기문 (1/2)

break를 사용한 반복문 종료 소스 코드 예제6-5.c

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int i;                // 반복 횟수를 세기 위한 제어 변수
06     int sum = 0;          // 1부터 10까지의 합을 누적할 변수
07
08     for (i = 1; i <= 10; i++)    // i는 1부터 10까지 증가하면서 10번 반복
09     {
10         sum += i;              // i 값을 sum에 누적
11         if (sum > 30) break;    // 누적한 값이 30보다 크면 반복문을 끝낸다.
12     }
13     printf("누적한 값 : %d\n", sum);
14     printf("마지막으로 더한 값 : %d\n", i);
15
16     return 0;
17 }
```

실행결과

누적한 값 : 36
마지막으로 더한 값 : 8

❖ break 분기문 (2/2)

안쪽 for문 하나만 탈출한다!

```
while ( ... )  
{  
    for ( ... )  
    {  
        ...  
        if ( 조건식 ) break;  
    }  
    ← 안쪽 for문 하나만 탈출합니다!  
}
```

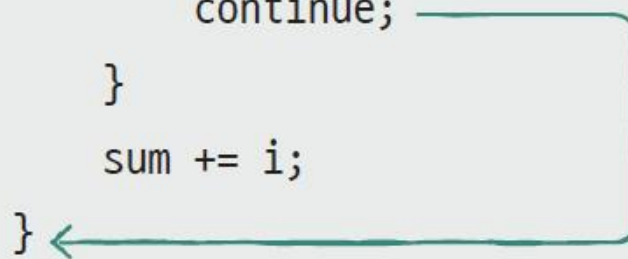
반복문 블록 전체를 벗어난다!

```
while (조건식)  
{  
    ...  
    if (조건식)  
    {  
        ...  
        break;  
    }  
    ← 반복문 블록 전체를 벗어납니다!  
}
```

❖ continue 분기문

- 반복문의 일부를 건너뛴다.

```
for ( i = 1; i <= 100; i++)  
{  
    if ( (i % 3) == 0 )  
    {  
        continue;  
    }  
    sum += i;  
}
```



i가 3의 배수면 `sum += i` 문장을 건너뛰고
블록 끝으로 간 후에 다시 반복합니다.

❖ 무한 반복문

```
while (1)
{
    printf("Be happy!\n");
}
```

```
for (;;)
{
    printf("Be happy!\n");
}
```

■ break로 무한 반복문 탈출

```
count = 0;
while (1)
{
    printf("Be happy!\n");
    count++;
    if (count == 5) break;
}
```

// 반복될 때마다 1씩 증가
// count의 값이 5가 되면 break로 반복 종료

키워드로 끝내는 핵심 포인트

- ❖ 중첩 반복문은 반복문의 실행문으로 반복문을 사용한다.
- ❖ `break`와 `continue`로 반복문의 실행 방식을 바꿀 수 있다.

표로 정리하는 핵심 포인트

표 6-2 중첩 반복문과 분기문

중첩 반복문 예	<pre>for (i = 0; i < 10; i++) { for (j = 0; j < 10; j++) { 반복할 문장; } }</pre>	<p>i-for문이 10번 반복되고 j-for문이 10번 반복되므로 반복할 문장은 100번 반복된다.</p>
분기문 사용 예	<pre>while (1) { if (조건식1) break; if (조건식2) continue; 반복할 문장; }</pre>	<p>조건식 1이 참이면 반복문을 끝낸다.</p> <p>조건식 2가 참이면 반복할 문장을 건너뛰고 처음부터 다시 반복한다.</p>