

# 제 7 장 반복

---

- 01 반복 개요와 while 문
- 02 do while 문과 for 문
- 03 분기문
- 04 중첩된 반복문



# 학습목표

- ▶ 반복문에 대하여 이해하고 구현할 수 있다.
  - while 문의 구조를 이해하고 필요한 반복의 구현이 가능
  - do while 문의 구조를 이해하고 필요한 반복의 구현이 가능
  - for 문의 구조를 이해하고 필요한 반복의 구현이 가능
  - 반복문 내부에서의 break와 continue의 기능
  - 의도적인 무한반복과 반복의 종료
- ▶ 중첩된 반복에 대하여 다음을 이해하고 구현할 수 있다.
  - 외부 제어변수와 내부 제어변수 변화를 이해
  - 구구단 구현
  - 입력의 종료를 알리는 방식과 구현

# 반복

- 반복
  - 순환 또는 루프(loop)라는 표현도 함께 사용
  - 반복몸체(repetition body)
    - 반복 조건을 만족하면 일정하게 반복되는 블록
- while, do while, for 세 가지 종류의 반복 구문

```
while ( <반복조건> )  
{  
    //반복몸체(loop body);  
    <해야할 일>;  
}
```

조건이 참(0이 아닌 값)이어야  
반복몸체를 실행

```
do  
{  
    //반복몸체(loop body);  
    <해야할 일>;  
} while ( <반복조건> );
```

조건이 참(0이 아닌 값)이어야  
반복몸체를 다시 실행

```
for ( <초기화>; <반복조건>; <증감> )  
{  
    //반복몸체(loop body);  
    <해야할 일>;  
}
```

# 특정 문자열을 여러 번 반복해 출력

## 반복 제어 변수의 중요성

- 반복 횟수에 따른 제어 변수의 값과 조건식의 결과를 살피는 것이 필요

실습예제 7-3
Prj03
03whilebasic.c
특정 문자열을 여러 번 반복해 출력
난이도: ★

```

01 #include <stdio.h>
02
03 int main(void)
04 {
05     int count = 1;
06
07     while (count <= 3)
08     {
09         printf("반복, 재미있네요!\n");
10         count++;
11     };
12     printf("\n제어 변수 count => %d\n", count);
13
14     return 0;
15 }

```

while 반복문체가 두 문장이므로 반드시 블록이 필요하며, 실수로 블록이 빠지면 논리 오류가 발생하여 무한 반복이 발생한다.

반복문체로 제어변수 count를 1 증가시키는데, 결과값이 연산에 참여하지 않으므로 ++count도 가능하고, 결국 count += 1도 가능

while 문이 종료된 이후의 count 값은 3이 아니라 4라는 사실에 주의, 그러므로 출력 값은 4

결과

반복, 재미있네요!  
반복, 재미있네요!  
반복, 재미있네요!

제어 변수 count => 4

반복횟수	변수 count 값	조건식	조건식 평가	반복문체
1	1	count <= 3 1 <= 3	while (1)	printf("C 언어 재미있네요!\n"); count++;
2	2	count <= 3 2 <= 3	while (1)	printf("C 언어 재미있네요!\n"); count++;
3	3	count <= 3 3 <= 3	while (1)	printf("C 언어 재미있네요!\n"); count++;
4	4	count <= 3 4 <= 3	while (0) while 종료	실행되지 못함

# while 반복으로 표준입력 실수를 모두 더하기

실습예제 7-4

Prj04

04whilesum.c

while 반복으로 표준입력 실수를 모두 더하기

난이도: ★

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03
04 int main()
05 {
06     double number = 1, sum = 0;
07     while (number != 0.0)
08     {
09         printf("실수 입력 >> ");
10         scanf("%lf", &number);
11         sum += number;
12     }
13
14     printf("합 = %.2f\n", sum);
15
16     return 0;
17 }
```

초기 값은 1이지만 첫 while 조건을  
통과하면서 새 표준입력 값이 대입됨

표준입력 값이 0이면 while 문을 종료

표준입력 값이 저장된 number를 계속 더하기

결과

```
실수 입력 >> 5.9
실수 입력 >> -3.4
실수 입력 >> 5
실수 입력 >> 0
합 = 7.50
```

논리 오류로 무한 반복 발생

```
int count = 1;
while (count <= 3)
    printf("C 언어 재미있네요!\n");
count++;
```

==

논리 오류로 무한 반복 발생

```
int count = 1;
while (count <= 3)
    printf("C 언어 재미있네요!\n");
count++;
```

블록 처리로 원하는 구문 처리

```
int count = 1;
while (count <= 3)
{
    printf("C 언어 재미있네요!\n");
    count++;
}
```

그림 7-12 while 블록의 중요성

# LAB 놀이 공원에서 키가 130 센티미터 이하인 정원 채우기

- 반복 while 문을 사용, 키가 130센티미터 이하인 어린이 정원 채우기
  - 정원은 매크로 상수 MAX로 정하고
    - 표준입력으로 어린이의 키를 입력 받아
    - 조건에 만족하면 입장할 수 있는 정원의 수를 하나씩 증가
    - 정해진 정원이 모두 차면 정원을 출력하고 프로 그램을 종료

Lab 7-1

lab1max.c

난이도: ★

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 #define MAX 4
04
05 int main()
06 {
07     int num = 0;
08     double height = 0;
09     while ( )
10     {
11         printf("키 입력 >> ");
12         scanf("%lf", &height);
13         if (height <= 130)
14             ;
15     }
16
17     printf("정원 %d명 완료!\n", num);
18
19     return 0;
20 }
```

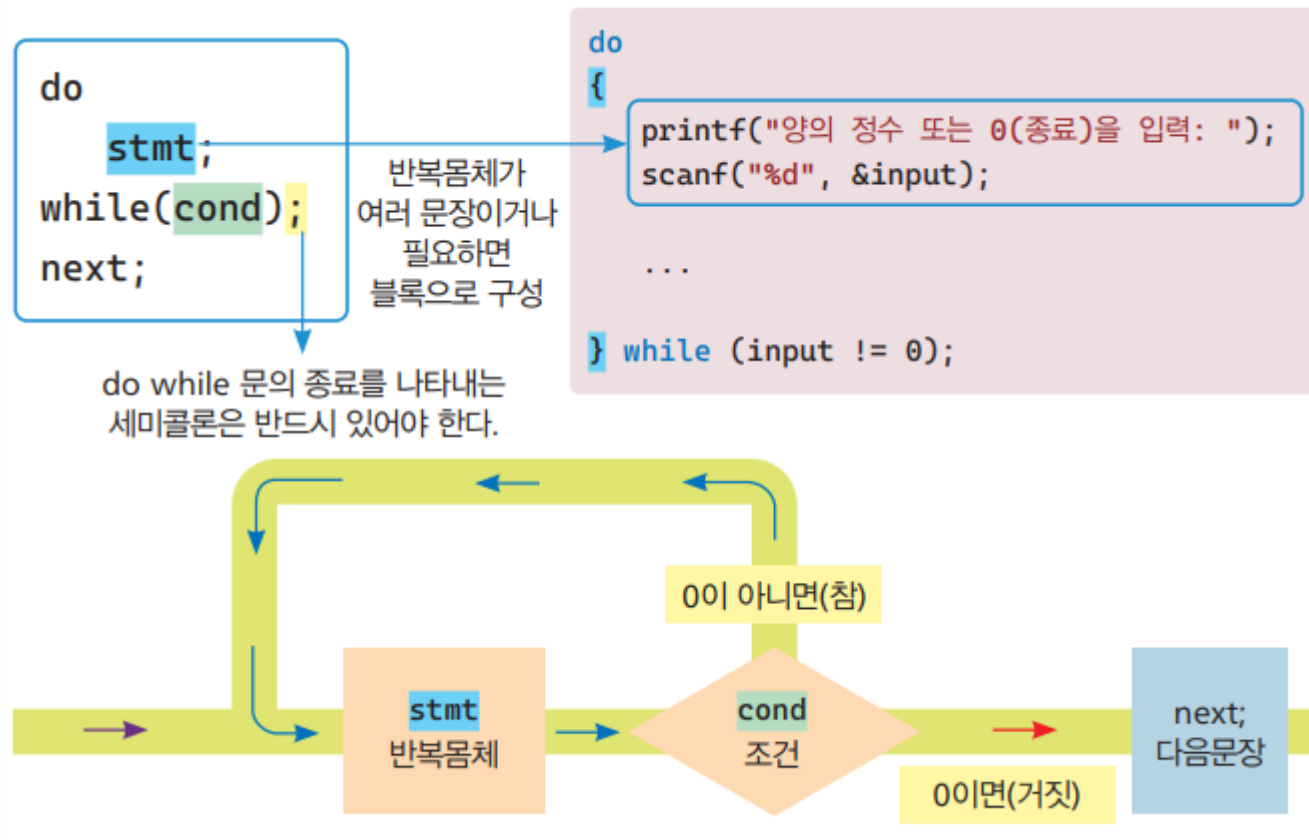
정답

```
09 while (num < MAX)
14     num++;
```

정원을 매크로 상수로 정의

# do while 문 구조와 제어흐름

- do while 문은 반복문체 수행 후에 반복 조건을 검사
  - while 문은 반복 전에 반복 조건을 평가
  - 반복 조건을 나중에 검사해야 하는 반복에 적합



# 센티널 값 검사에 유용

- 입력 후에 반복 검사를 진행하는 처리 과정
  - do while 문으로 구현이 적합
  - 센티널 값(sentinel value)
    - 반복의 종료를 알리는 특정한 자료 값

실습예제 7-5

Prj0505dowhile.c메뉴 주문 반복난이도: ★

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03
04 int main(void)
05 {
06     int input;
07     do
08     {
09         printf("[0]종료 [1]아메리카노 [2]카페라떼 [3]카푸치노 \n");
10         printf("주문할 커피 또는 종료(0)를 입력 >> ");
11         scanf("%d", &input);
12     } while (input != 0); //while (input);
13
14     return 0;
15 }
```

조건식 (input != 0)을 사용하므로 0이 아니어야 9번 줄로 이동하여 반복하며, 0이면 반복을 종료 하고, 조건식 (input != 0)은 (input)과 같음

결과

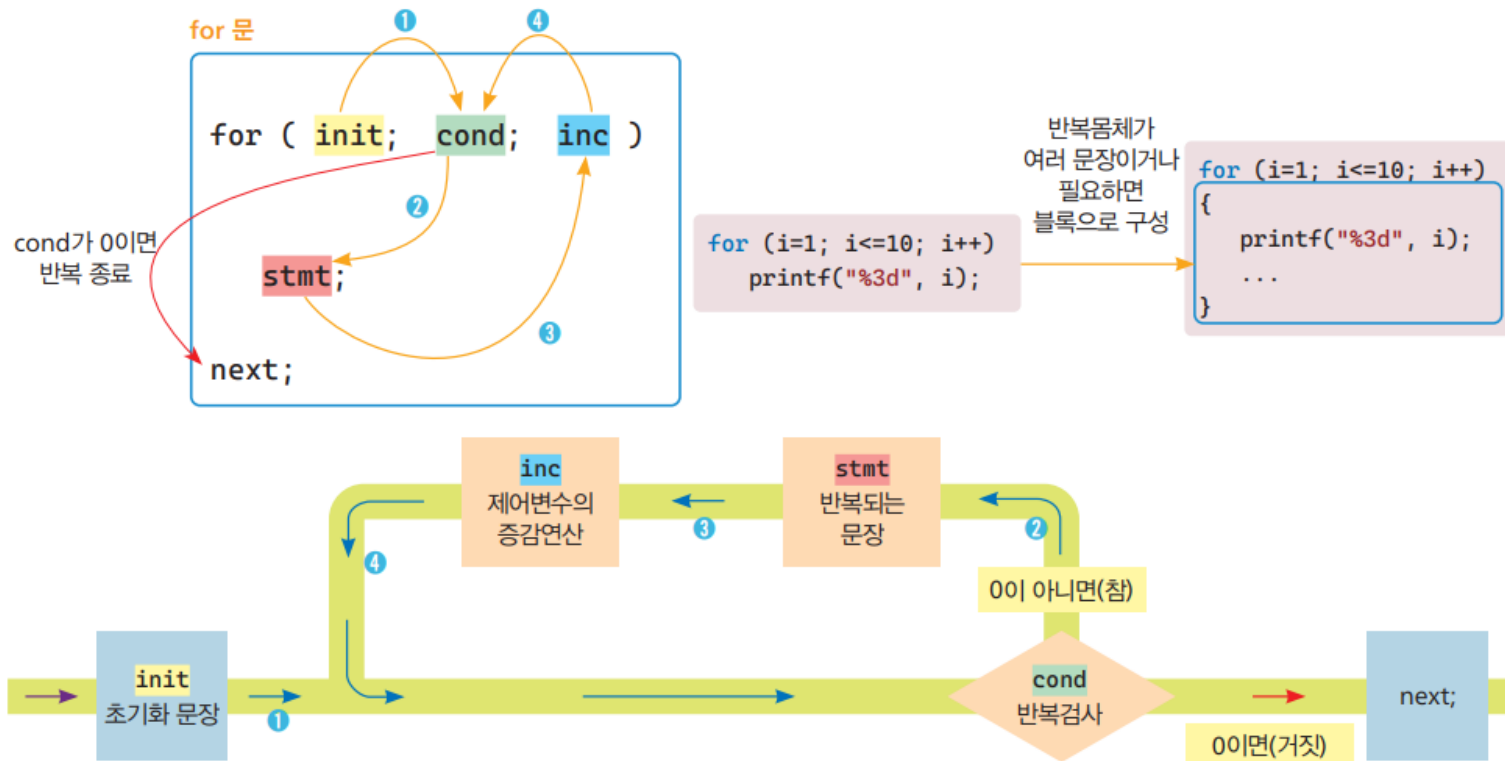
[0]종료 [1]아메리카노 [2]카페라떼 [3]카푸치노  
주문할 커피 또는 종료(0)를 입력 >> 2  
[0]종료 [1]아메리카노 [2]카페라떼 [3]카푸치노  
주문할 커피 또는 종료(0)를 입력 >> 3  
[0]종료 [1]아메리카노 [2]카페라떼 [3]카푸치노  
주문할 커피 또는 종료(0)를 입력 >> 0



# for 문 구조와 제어 흐름

- 반복문 **for (init; cond; inc) stmt;**

- init: 주로 초기화(initialization)
- cond: 반복 조건을 검사
- inc: 주로 반복을 결정하는 제어 변수의 증감(increment)을 수행



# for 구문으로 일정 횟수 반복

- 2개의 세미콜론은 반드시 필요
- 반복조건 cond를 아예 제거하면 반복은 무한히 계속

실습예제 7-6

Prj06

06forbasic.c

for 구문으로 일정 횟수 반복

난이도: ★

```
01 #include <stdio.h>
02 #define MAX 5
03
04 int main(void)
05 {
06     int i;
07     for (i = 1; i <= MAX; i++)
08         printf("반복 %d\n", i);
09
10     printf("\nfor 종료 이후 i => %d\n", i);
11
12     return 0;
13 }
```

조건식  $i \leq \text{MAX}$ 는 전처리 수행 후, MAX가 5로 대체되어  $i \leq 5$ 가 되며,  $i$ 가 5보다 큰 6인 경우 조건식이 거짓이 되어 반복을 종료

증감의  $i++$ 는 반복문체인 8번 줄의 문장이 실행된 이후 실행

결과

반복 1  
반복 2  
반복 3  
반복 4  
반복 5

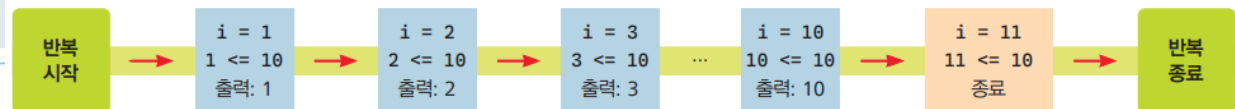
for 종료 이후  $i \Rightarrow 6$

초기화 문장은 단 한번만 실행된다.

```
for (int i = 1; i <= 10; i++)
    printf("%3d: ", i);
```

이 부분은  $++i$ ,  $i = i+1$ ,  $i += 1$  모두 가능하다.

```
int i = 0;
while (i <= 10)
{
    printf("%3d", i);
    i++;
}
```



# 다양한 for 문

- 섭씨 온도 celsius를 12.46으로 시작하여 3개의 화씨 온도를 각각 출력

실습예제 7-7

Prj07      07forcel2far3.c      for 문으로 3개의 섭씨 온도를 화씨 온도로 변환      난이도: ★

```
01  #include <stdio.h>
02  #define MAX 3
03  #define INCREMENT 10
04
05  int main(void)
06  {
07      double celsius = 12.46;
08
09      printf("섭씨(C)   화씨(F)\n");
10      for (int i = 1; i <= MAX; i++, celsius += INCREMENT)
11      {
12          printf("%.2f %.2f\n", celsius, 9.0 / 5 * celsius + 32);
13      }
14
15      return 0;
16  }
```

매크로 상수 3을 정의, MAX는 반복 횟수 값으로 지정

for 문 초기화는 int i=1과 같이 변수 선언과 초기화도 가능, 변수 i는 for 문 내부에서만 사용 가능한 변수

이 부분은 여러 문장을 콤마로 나열이 가능하며, 제어변수도 1 증가시키고, 섭씨 온도도 증가분인 INCREMENT(10)만큼 증가시킴

결과	섭씨(C)	화씨(F)
	12.46	54.43
	22.46	72.43
	32.46	90.43

# 반복 조건에서의 주의



## TIP 반복 조건에서의 주의

반복 조건에서 등호나 부등호의 ==나 != 또는 대입연산자 =의 사용은 주의를 필요로 한다. 비교연산자 ==와 !=에서 피연산자로 실수는 가급적 사용하지 않도록 하자. 예를 한 가지 들자면, 다음과 같이 0.0에서 0.1씩 증가시켜 1.0까지 10회를 반복하고자 하는 경우, 반복 조건을 `d != 1.0`으로 하면 실수 연산의 오차로 인해 조건식 `d != 1.0`이 항상 참인 결과로 반복이 무한히 계속될 수 있다. 다음의 왼쪽 소스는 무한히 반복되나 오른쪽과 같이 `d <= 1.0`으로 조건을 검사하면 0, 0.1, 0.2, ..., 1.0까지 출력된다.

```
for (double d = 0.0; d != 1.0; d += 0.1)
    printf("%f ", d);
```

```
for (double d = 0.0; d <= 1.0; d += 0.1)
    printf("%f ", d);
```

그림 7-19 실수의 연산에서 != 또는 ==의 문제

또한 대입연산자 =과 등호 연산 ==도 서로 혼동되지 않도록 유의하자. 다음 왼쪽 소스는 1 2 3 4 5가 출력되나 오른쪽 소스는 대입연산자인 =으로 잘못 사용하여 출력되는 것이 하나도 없다.

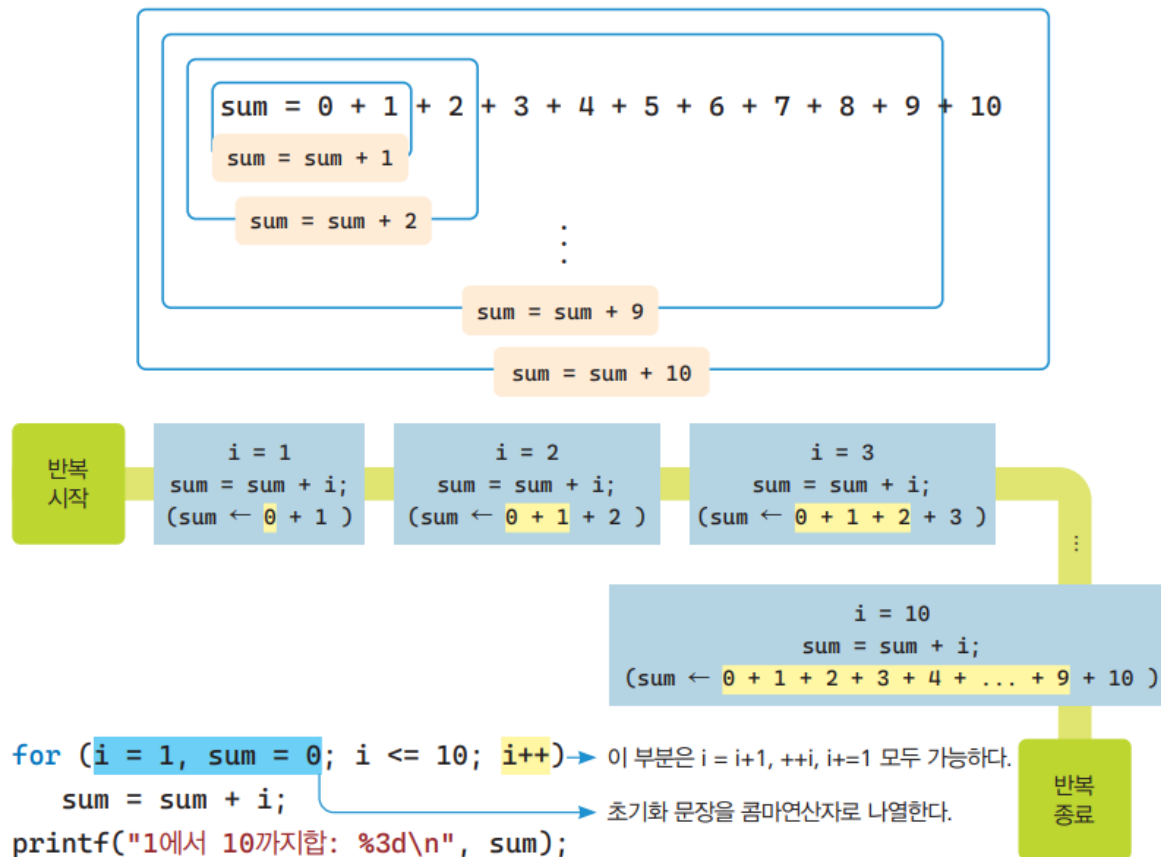
```
int i = 1;
while (!(i == 6))
{
    printf("%d ", i++);
}
```

```
int i = 1;
while (!(i = 6))
{
    printf("%d ", i++);
}
```

그림 7-20 ==를 =로 잘못 사용

# for 문의 합 구하기

- for 문을 이용하여 1에서 10까지 합을 구하는 모듈
  - 제어 변수 i를 이용하여 1부터 10까지 순회
  - 순회하는 제어 변수 i 값을 계속 합하여 변수 sum에 누적



# 1에서 10까지의 합을 구하는 다양한 for 구문

실습예제 7-9

Prj09

09forsum.c

1에서 10까지의 합을 구하는 다양한 for 구문

난이도: ★

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int i, sum;
06     for (i = 1, sum = 0; i <= 10; i++) //++i도 가능
07         sum += i; // sum = sum + i;
08     printf("1 ~ 10 합: %d\n", sum);
09
10     for (i = 1, sum = 0; i <= 10; )
11         sum += i++;
12     printf("1 ~ 10 합: %d\n", sum);
13
14     for (i = 0, sum = 0; i <= 9; )
15         sum += ++i;
16     printf("1 ~ 10 합: %d\n", sum);
17
18     for (i = 1, sum = 0; i <= 10; sum += i++); //반복문체가 없는 for 문
19     printf("1 ~ 10 합: %d\n", sum);
20     for (i = 0, sum = 0; i <= 9; sum += ++i); //반복문체가 없는 for 문
21     printf("1 ~ 10 합: %d\n", sum);
22
23     return 0;
24 }
```

반복문체인 sum += i는 들여쓰기가 반드시 필요하며, sum = sum + i의 축약

증감부분이 비어 있어도, 앞에 세미콜론은 반드시 필요

덧셈에 참여하는 값은 반복 조건을 통과한 값보다 1이 큰 정수이다.

for 문의 반복문체는 따로 없으므로, 뒤에 for 문을 종료하는 세미콜론 ;이 반드시 필요하다.

결과

```
1 ~ 10 합: 55
1 ~ 10 합: 55
1 ~ 10 합: 55
1 ~ 10 합: 55
1 ~ 10 합: 55
```

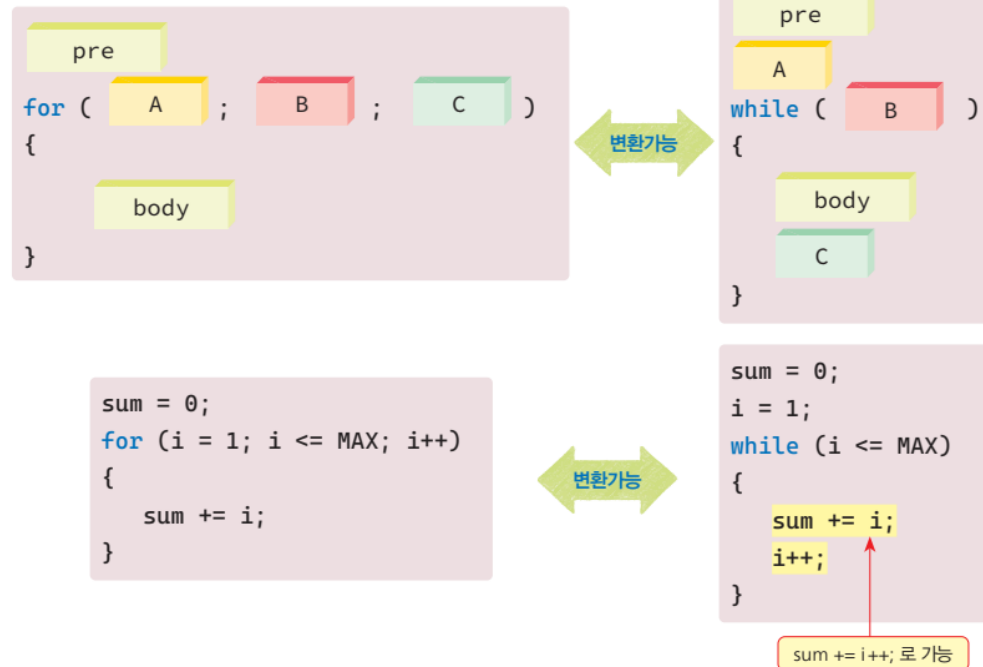
# for 문과 while 문의 비교

- for 문

- 주로 반복 횟수를 제어하는 제어 변수를 사용
- 초기화와 증감 부분이 있는 반복문에 적합

- while 문

- 구조가 간단하므로 다양한 구문에 이용
- 반복 횟수가 정해지지 않고 특정한 조건에 따라 반복을 결정하는 구문에 적합
- for 문과 while 문은 서로 변환이 가능



# 1에서부터 표준입력한 양수까지의 합을 구하는 for와 while

실습예제 7-10

Prj10

10inputsum.c

1에서부터 표준입력한 양수까지의 합을 구하는 for와 while

난이도: ★

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03
04 int main(void)
05 {
06     int i, sum, max;
07     printf("양의 정수 입력 >> ");
08     scanf("%d", &max);
09
10     for (i = 1, sum = 0; i <= max; i++) //++i도 가능
11         sum += i; // sum = sum + i;
12     printf("\nfor 문으로 구한 1에서 %d까지 합: %3d\n", max, sum);
13
14     i = 1, sum = 0;
15     while (i <= max)
16     {
17         sum += i; // sum = sum + i;
18         i++; // ++i도 가능
19     }
20     printf("\nwhile 문으로 구한 1에서 %d까지 합: %3d\n", max, sum);
21
22     return 0;
23 }
```

조건식  $i \leq \text{max}$ 로, 증감은  $i++$ 로, 증감은 1만 증가시키면 되므로,  $++i$ ,  $i += 1$ ,  $i = i + 1$  도 가능

반복문체의 두 번째 문장인  $i++$ 는 for 문에서 증감에 있던 문장으로, 증감은 1만 증가시키면 되므로,  $++i$ ,  $i += 1$ ,  $i = i + 1$  도 가능

결과

양의 정수 입력 >> 15

for 문으로 구한 1에서 15까지 합: 120

while 문으로 구한 1에서 15까지 합: 120



# LAB 백 단위 정수의 세 개의 각 자릿수 출력

Lab 7-2

lab2digit.c

난이도: ★

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03
04 int main(void)
05 {
06     int input = 0, result = 0, digit = 0;
07     int devider = 100;
08
09     printf("양의 정수[100~999] 입력 : ");
10     scanf("%d", &input);
11     result = input;
12     do
13     {
14         digit = ;
15         result %= devider;
16         printf("%3d단위 출력: %d\n", devider, digit);
17         ;
18     } while (devider >= 1);
19
20     return 0;
21 }
```

정답

```
14     digit = result / devider;
17     devider /= 10;
```

양의 정수[100~999] 입력 : 853  
100단위 출력: 8  
10단위 출력: 5  
1단위 출력: 3

# 반복의 중단 break

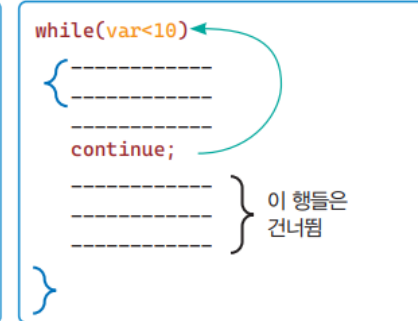
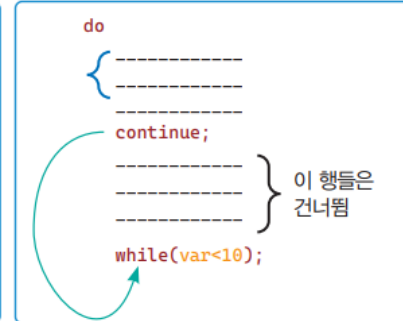
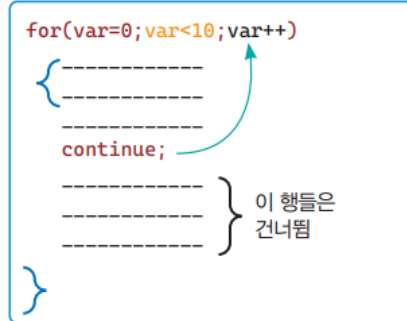
- break 문장
  - 반복 내부에서 반복을 종료

실습예제 7-11	Prj11	11break.c	반복된 정수의 16진수 변환과 break로 종료	난이도: ★
<pre>01  #define _CRT_SECURE_NO_WARNINGS 02  #include &lt;stdio.h&gt; 03 04  int main(void) 05  { 06      int input; 07      while (1) 08      { 09          printf("양의 정수 또는 0[종료] 입력 후 [Enter] &gt;&gt; "); 10          scanf("%d", &amp;input); 11          if (input == 0) 12              break; 13          printf("입력한 정수 %d: 16진수 %#x\n", input, input); 14      } 15      puts("종료"); 16 17      return 0; 18  }</pre> <p>while (1) 에서 조건식 1이 항상 참이므로 무한반복</p>				
결과	<p>양의 정수 또는 0[종료] 입력 후 [Enter] &gt;&gt; 15 입력한 정수 15: 16진수 0xf 양의 정수 또는 0[종료] 입력 후 [Enter] &gt;&gt; 16 입력한 정수 16: 16진수 0x10 양의 정수 또는 0[종료] 입력 후 [Enter] &gt;&gt; 0 종료</p>			

# 3으로 나누어지지 않는 정수 출력

## • 반복의 계속 continue

- continue는 자신이 속한 가장 근접한 반복에서 다음 반복을 실행



실습예제 7-12

Prj12

12continue.c

3으로 나누어지지 않는 정수 출력

난이도: ★

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     const int MAX = 15;
06
07     printf("1에서 %d까지 정수 중에서 3으로 나누어 떨어지지 않는 수\n", MAX);
08     for (int i = 1; i <= MAX; i++)
09     {
10         if (i % 3 == 0) // (!(i % 3))
11             continue;
12         printf("%3d", i);
13     }
14     puts("");
15
16     return 0;
17 }
```

continue를 만나면 실행되지 않고 다음 반복을 위해 i++로 이동

조건식 (i % 3 == 0)은 3으로 나누어 떨어지면 참, 떨어지지 않으면 거짓으로, (!(i % 3))으로도 가능

continue를 만나지 않으면 이 출력문이 실행

```
while ( cond1 )
{
    ...
    ② continue;

    for (init; cond2; inc )
    {
        ...
        ① continue;

        stmt1;
        stmt2;
    }
}
```

바로 위 ① continue를 만나면 실행되지 않는 부분

바로 위 ② continue를 만나면 실행되지 않는 부분

결과

1에서 15까지 정수 중에서 3으로 나누어 떨어지지 않는 수  
1 2 4 5 7 8 10 11 13 14

# goto와 무한 반복

- goto 문

- 레이블(label)이 위치한 다음 문장으로 실행 순서를 이동하는 문장
- 사용하지 않는 것이 바람직

- 무한 반복

실습예제 7-13

Prj1313goto.cgoto를 사용해 1에서 10까지의 정수 출력난이도: ★

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

문장 goto 문은 무조건 label이 있는 곳으로 이동하여 실행

goto 이후에 이동할 레이블을 기술, 변수 count가 11이 되면 if의 조건식이 거짓이 되어 12번 줄로 이동

결과1 2 3 4 5 6 7 8 9 10  
종료

무한반복

```
for ( ; ; )
{
    ...
}
```

무한반복

```
for ( ; 1 ; )
{
    ...
}
```

무한반복

```
while ( 1 )
{
    ...
}
```

무한반복

```
do
{
    ...
} while ( 1 )
```

오류

```
while ( )
{
    ...
}
```

오류

```
do
{
    ...
} while ( )
```

# 반복된 정수의 8진수와 16진수 변환과 break로 종료

- 양의 정수를 입력하면 입력된 정수와 함께 8진수와 16진수 형태를 출력
  - 다시 다음 입력을 받아 출력이 반복
  - 0 또는 음수인 0 이하의 정수를 입력하면 종료

실습예제 7-14	Prj14	14octhex.c	반복된 정수의 8진수와 16진수 변환과 break로 종료	난이도: ★
	<pre>01  #define _CRT_SECURE_NO_WARNINGS 02  #include &lt;stdio.h&gt; 03 04  int main(void) 05  { 06      int input; 07      do 08      { 09          printf("양의 정수 또는 음수나 0[종료] 입력 후 [Enter] &gt;&gt; "); 10          scanf("%d", &amp;input); 11          if (input &lt;= 0) 12              break; 13          printf("정수 %d: 8진수 %#o  16진수 %#x\n", input, input, input); 14      } while (1); 15      return 0; 16  }</pre> <p>break는 while 문을 종료</p> <p>조건식에 1이나 참을 의미하는 값을 넣으면 무한 반복</p>			
결과	<pre>양의 정수 또는 음수나 0[종료] 입력 후 [Enter] &gt;&gt; 8 정수 8: 8진수 010  16진수 0x8 양의 정수 또는 음수나 0[종료] 입력 후 [Enter] &gt;&gt; 16 정수 16: 8진수 020  16진수 0x10 양의 정수 또는 음수나 0[종료] 입력 후 [Enter] &gt;&gt; -1</pre>			

# LAB 양의 정수의 소수 여부를 판단해 출력

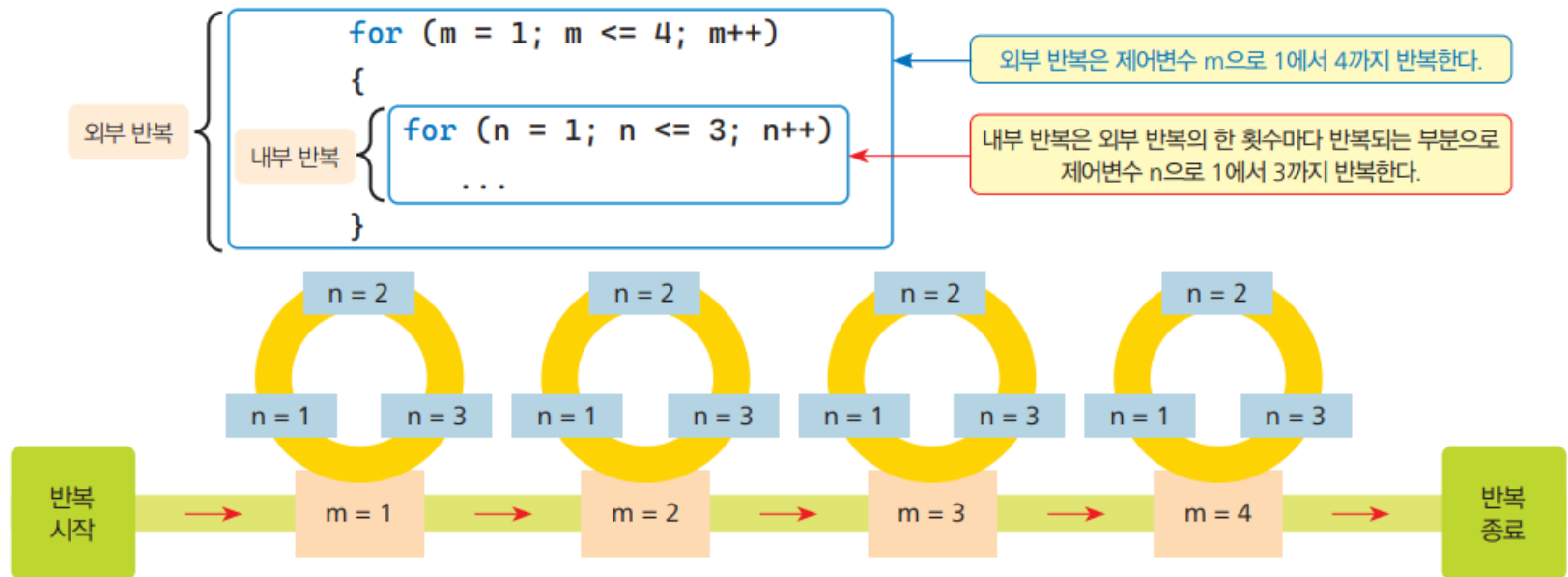
- 표준입력된 정수가 소수(prime number)인 지를 출력
  - 소수는 2부터 자기 자신 이전의 정수로 나누어지지 않는 수

Lab 7-3	lab3primebreak.c	난이도: ★
	<pre>01  #define _CRT_SECURE_NO_WARNINGS 02  #include &lt;stdio.h&gt; 03 04  int main() 05  { 06      int num, j; 07      printf("2 이상 양의 정수 입력 &gt;&gt; "); 08      scanf("%d", &amp;num); 09 10      for (j = 2; j &lt; num; j++) 11          if ( ) 12              break; 13 14      if ( ) 15          printf("%d 소수이다.\n", num); 16      else 17          printf("%d 소수가 아니다.\n", num); 18 19      return 0; 20  }</pre>	
정답	<pre>11      if (num % j == 0) 14      if (j == num)</pre>	

# 중첩된 for 문

## 중첩된 반복문

- for 문 내부에 for 문이 존재
- 제어 변수는 m, n
  - 외부 for 문의 제어 변수는 m이며, 내부 for 문의 제어 변수는 n
  - 외부 반복에서 m은 1에서 4까지 반복
  - 각각의 m에 대해, 내부 반복에서 n이 1에서 3까지 반복



# 중첩 반복

- 내부 반복과 외부 반복에서 각각의 변수 값의 변화를 이해
  - 외부 반복에서 1에서 5까지,
  - 내부 반복에서 1에서 7까지 반복하면서
  - 각각의 변수 값을 출력하는 예제 프로그램

실습예제 7-15

Prj15

15nestedloop.c

내부 반복과 외부 반복에서 각각의 변수 값의 변화를 이해

난이도: ★

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int m, n;
06     for (m = 1; m <= 5; m++)
07     {
08         printf("m = %-2d\n", m);
09         for (n = 1; n <= 7; n++)
10             printf("n = %-3d", n);
11         puts("");
12     }
13
14     return 0;
15 }
```

외부 반복의 for 문으로 1에서 5까지 반복

내부 반복의 for 문으로 1에서 7까지 반복

결과

m = 1  
n = 1 n = 2 n = 3 n = 4 n = 5 n = 6 n = 7  
m = 2  
n = 1 n = 2 n = 3 n = 4 n = 5 n = 6 n = 7  
m = 3  
n = 1 n = 2 n = 3 n = 4 n = 5 n = 6 n = 7  
m = 4  
n = 1 n = 2 n = 3 n = 4 n = 5 n = 6 n = 7  
m = 5  
n = 1 n = 2 n = 3 n = 4 n = 5 n = 6 n = 7



# 내부 반복이 외부 반복에 의존

- 외부 반복에서 변수  $i$ 는 1에서 5까지 반복
  - 내부 반복에서 제어 변수  $j$ 는 1에서 외부 반복의 제어 변수  $i$ 까지 반복

실습예제 7-16

Prj16

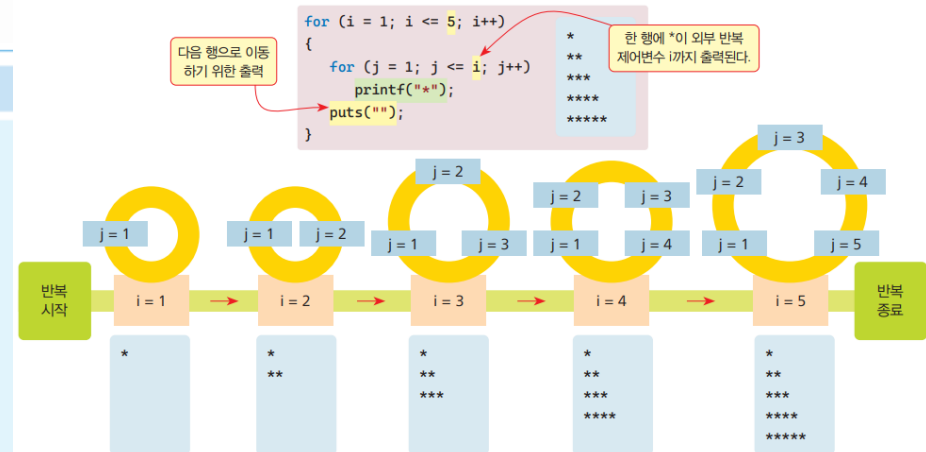
16triangle.c

별을 삼각형 모양으로 출력

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     const int MAX = 5;
06     int i, j;
07
08     for (i = 1; i <= MAX; i++)
09     {
10         for (j = 1; j <= i; j++)
11             printf("*");
12         puts("");
13     }
14
15     return 0;
16 }
```

결과

```
*
**
***
****
*****
```



내부 반복의 for 문으로 1에서  $i$ (외부 반복의 제어 변수)까지 반복,  $i$ 는 반복에 따라 1에서 5까지 변화

# 삼중 중첩 반복

- 양의 정수를 입력 받아 합을 출력
  - 0 또는 음수를 입력할 때까지 계속 수행하는 프로그램
  - 센티널 값인 0 또는 음수를 입력하면 프로그램이 종료

실습예제 7-17

Prj17

17loopsum.c

1에서 입력된 정수까지의 합 구하기

난이도: ★★

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03
04 int main(void)
05 {
06     int input, sum, i, j;
07     do
08     {
09         printf("양의 정수 또는 0(종료)을 입력: ");
10         scanf("%d", &input);
11
12         for (i = 1; i <= input; i++)
13         {
14             for (j = 1, sum = 0; j <= i; j++)
15             {
16                 printf("%d", j);
17                 j == i ? printf(" = ") : printf(" + ");
18                 sum += j;
19             }
20             printf("%d\n", sum);
21         }
22     } while (input > 0);
23
24     puts("종료합니다.");
25
26     return 0;
27 }
```

연산식 (j == i)는 j가 마지막이면 참, 반복의 중간이면 거짓임. 중간이면 j 값을 한 줄에 출력한 이후에 + 연산자 출력, 마지막이면 = 를 출력

입력 정수가 양수이면 반복하여 계속 실행하고, 0이나 음수이면 종료

결과

```
양의 정수 또는 0(종료)을 입력: 7
1 = 1
1 + 2 = 3
1 + 2 + 3 = 6
1 + 2 + 3 + 4 = 10
1 + 2 + 3 + 4 + 5 = 15
1 + 2 + 3 + 4 + 5 + 6 = 21
1 + 2 + 3 + 4 + 5 + 6 + 7 = 28
양의 정수 또는 0(종료)을 입력: 3
1 = 1
1 + 2 = 3
1 + 2 + 3 = 6
양의 정수 또는 0(종료)을 입력: 0
종료합니다.
```

# LAB 구구단 출력

- 중첩된 반복 for 문을 사용하여 2단부터 9단까지의 구구단을 출력

Lab 7-4

lab4mtable.c

난이도: ★

```
01  #include <stdio.h>
02  #define MAX 9
03
04  int main(void)
05  {
06      for (int i = 2; i <= MAX; i++)
07      {
08          printf("%5d단 출력\n", i);
09          for (int j = 2; j <= MAX; j++)
10              printf( );
11          printf( );
12      }
13
14      return 0;
15  }
```

정답

```
10      printf("%d*d = %2d ", i, j, i * j);
11      printf("\n");
```

감사합니다.