프로그래밍 기초

7장. 반복

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

반복문

- **반복**
  - 순환 또는 루프(loop)라는 표현도 함께 사용
  - 반복 몸체(repetition body)
    - ➢ 반복 조건을 만족하면 일정하게 반복되는 블록

- **for, while, do while 세 가지 종류의 반복 구문**

```
for(초기값; 반복조건; 증감)
{
    // 반복 몸체 (loop body)
    코드1;
    . . .

}
```

조건이 true(0이 아닌 값)인 경우, 반복 몸체 실행

```
while(반복조건)
{
    // 반복 몸체 (loop body)
    코드1;
    . . .

}
```

최소 1회 반복 몸체 실행

```
do
{
    // 반복 몸체 (loop body)
    코드1;
    . . .

} while(반복조건);
```

조건이 true(0이 아닌 값)인 경우, 반복 실행

반복문 활용

- for 문 - 정해진 횟수만큼 반복하는 구조
- while 문 - 특정한 조건이 만족되는 동안 반복을 계속하는 구조
  - 반복 횟수를 알기 어려운 경우에 사용

for 문

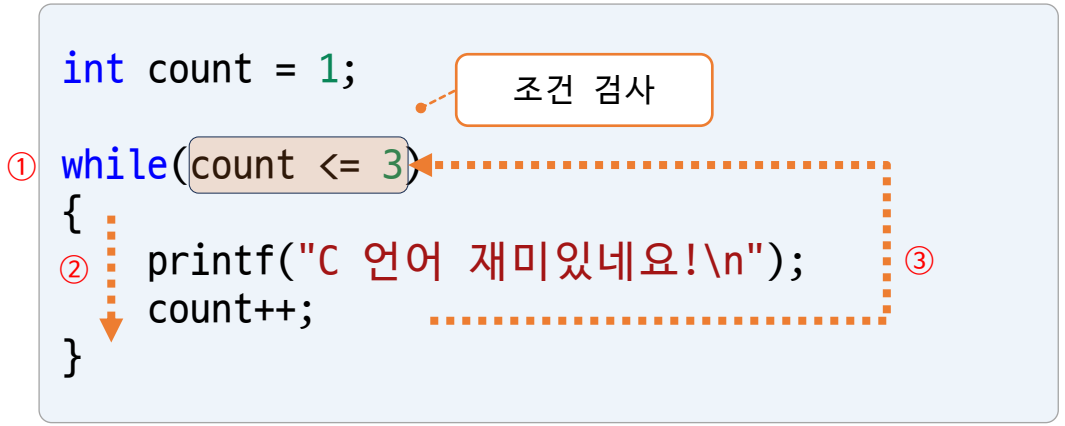
횟수 반복

while 문

조건 반복

while 문 구조와 제어 흐름

- **while(조건) 실행 흐름**
  - 1. while 문의 **조건**을 검사
  - 2. 조건이 true이면 while문 내부의 반복 몸체를 실행
    - 조건이 false이면 while문 내부로 진입하지 않음
  - 3. 반복 몸체를 실행 후 다시 while(조건)으로 이동
    - 반복 몸체(repetition body)는 필요하면 블록으로 구성



특정 문자열을 여러 번 반복해 출력

<03whilebasic.c>

```c
#include <stdio.h>

int main(void)
{
    int count = 1;

    while (count <= 3)
    {
        printf("반복, 재미있네요!\n");
        count++;
    }
    printf("\n제어 변수 count => %d\n", count);

    return 0;
}
```

반복 몸체(repetition body)가 두 문장이므로 반드시 블록이 필요

실행 결과

```
반복, 재미있네요!
반복, 재미있네요!
반복, 재미있네요!

제어 변수 count => 4
```

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

논리 오류로 무한 반복 발생

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

==

논리 오류로 무한 반복 발생

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

블록 처리로 원하는 구문 처리

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

그림 7-12 while 블록의 중요성

while 반복으로 표준 입력 실수를 모두 더하기 (실습)

<04whilenum.c>

```c
#include <stdio.h>

int main()
{
    double number = 1, sum = 0;

    while (number != 0.0)
    {
        printf("실수 입력 >> ");
        scanf("%lf", &number);
        sum += number;
    }

    printf("합 = %.2f\n", sum);
    return 0;
}
```

number의 초기값이 1
- while 문장 내부로 진입

number == 0.0인 경우, sum의 값 출력

실행 결과

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

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 받음
    - 조건에 만족하면 입장할 수 있는 정원의 수를 하나씩 증가
    - 정해진 정원이 모두 차면 정원을 출력하고 프로 그램을 종료

```c
#include <stdio.h>
#define MAX 4

int main()
{
    int num = 0;
    double height = 0;

    while (num < MAX)
    {
        printf("키 입력 >> ");
        scanf("%lf", &height);
        if (height <= 130)
            num++;
    }
    printf("정원 %d명 완료!\n", num);
    return 0;
}
```

num의 값과 매크로 상수값 비교

130 보다 작은 경우에만
- num 값을 증가

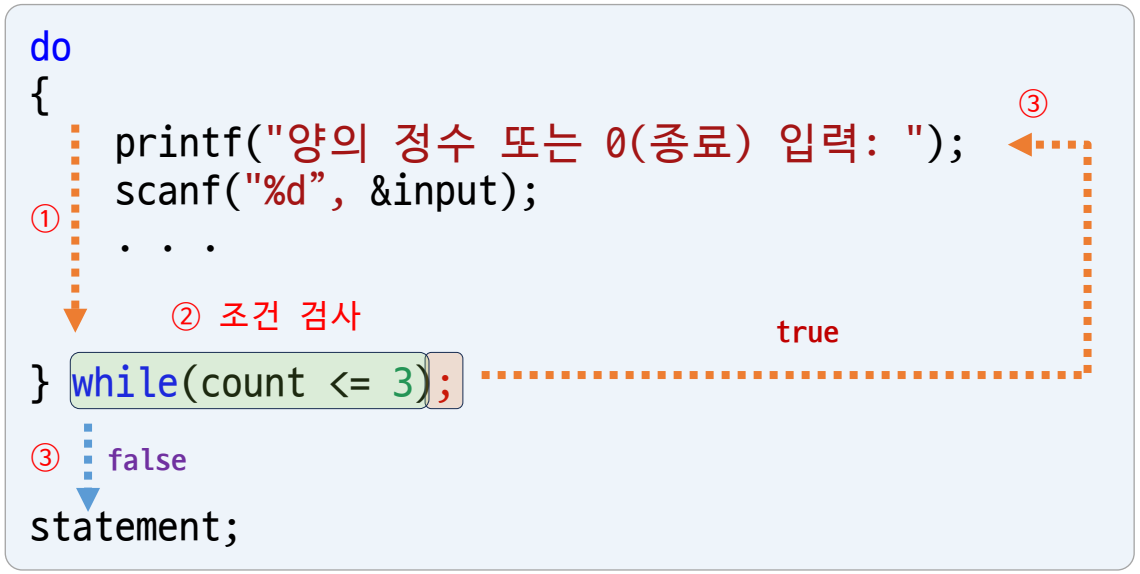
실행 결과

```
키 입력 >> 119.3
키 입력 >> 136.2
키 입력 >> 129
키 입력 >> 128
키 입력 >> 120
정원 4명 완료!
```

130보다 큼
- num 값이 증가하지 않음

do while 문 구조와 제어 흐름

- **do while 문은 반복 몸체 수행 후에 반복 조건을 검사**
  - 먼저 반복 몸체를 실행(①)한 이후에 반복 조건(cond)를 검사(②)
    - 조건이 true이면, 다시 반복 몸체를 실행
    - 조건이 false이면, do while 문을 종료
  - do while의 몸체는 최소 한 번은 실행
  - while(cond);
    - 반드시 while 이후에 세미콜론(;)이 필요



센티널 값 검사에 유용 (실습)

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

<05dowhile.c>

```c
#include <stdio.h>

int main(void)
{
    int input;
    do
    {
        printf("[0]종료 [1]아메리카노 [2]카페라떼 [3]카푸치노 \n");
        printf("주문할 커피 또는 종료(0)를 입력 >> ");
        scanf("%d", &input);
    } while (input != 0);

    printf("프로그램 종료\n");
    return 0;
}
```

input의 값이 0이면
- do while문 빠져 나감

실행 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
프로그램 종료
```

do while 예제: 숫자 맞추기

<05dowhile_guess.c>

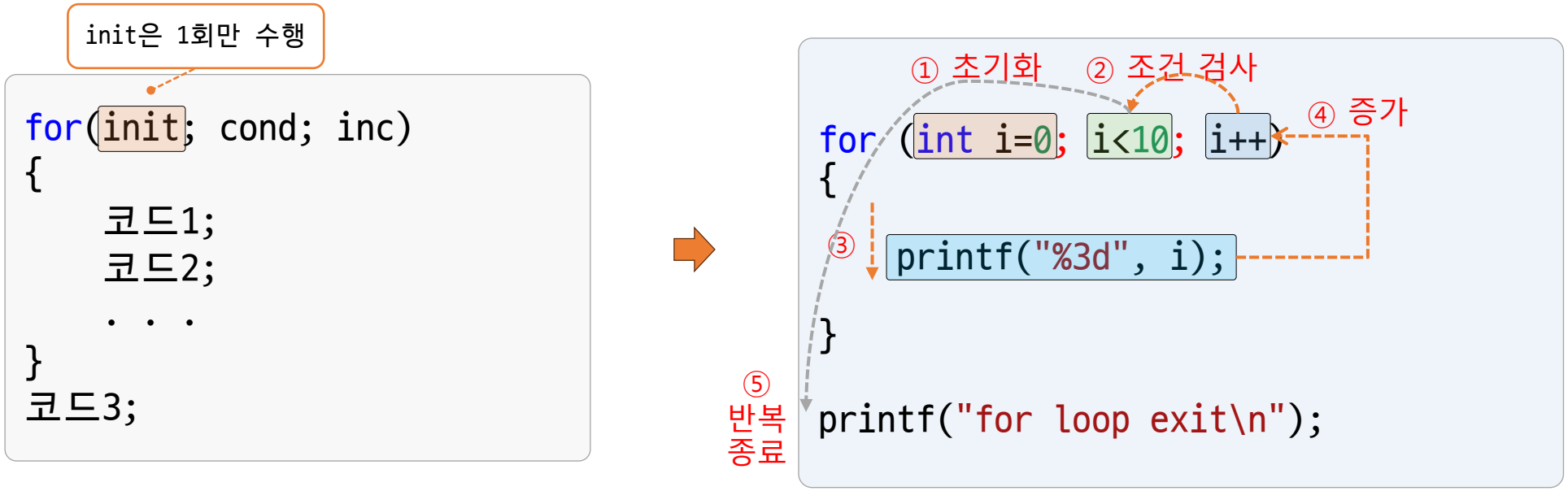
```c
#include <stdio.h>

int main()
{
    int answer = 59; // 정답
    int guess;
    int tries = 0; // 숫자 입력 횟수

    do {
        printf("숫자를 맞춰보세요: ");
        scanf("%d", &guess);
        tries++;

        if(guess > answer)
            printf("입력한 값이 정답보다 크다.\n");
        else if (guess < answer)
            printf("입력한 값이 정답보다 작다.\n");
        else
            printf("정답입니다.\n");
    } while (guess != answer);

    printf("시도 횟수: %d\n", tries);
    return 0;
}
```

실행 결과

```
숫자를 맞춰보세요: 27
입력한 값이 정답보다 작다.
숫자를 맞춰보세요: 80
입력한 값이 정답보다 크다.
숫자를 맞춰보세요: 57
입력한 값이 정답보다 작다.
숫자를 맞춰보세요: 59
정답입니다.
시도 횟수: 4
```

for 문 구조와 제어 흐름

- **반복문 for (init; cond; inc) stmt;**
  - init: 주로 제어 변수 초기화(initialization): 1회만 수행
  - cond: 반복 조건을 검사
  - inc: 주로 반복을 결정하는 제어 변수의 증가/감소(increment/decrement)를 수행



for 구문: 일정 횟수 반복 (실습)

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

<06forbasic.c>

```c
#include <stdio.h>
#define MAX 5

int main(void)
{
    int i;

    for (i = 1; i <= MAX; i++)
        printf("반복 %d\n", i);

    printf("for 종료 이후 i => %d\n", i);

    return 0;
}
```

초기화 문장(i = 1)은 1회만 수행

반복(5회) 종료 후 실행

실행 결과

```
반복 1
반복 2
반복 3
반복 4
반복 5
for 종료 이후 i => 6
```

for 문의 이해

- **1에서 10까지 출력하는 프로그램**
  - 반복횟수를 제어하는 제어변수 i를 1로 초기화
  - 조건검사 i <= 10를 이용하여 변수 i를 출력
    - ➢ 변수 i와 같이 반복의 횟수를 제어하는 변수를 제어변수

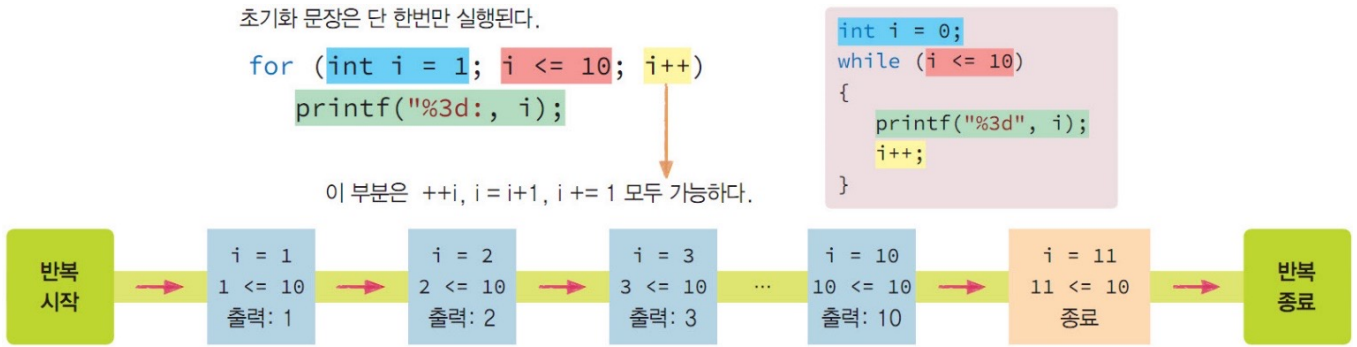


그림 7-17 1부터 10까지 출력하는 for문

표 7-2 for 문의 실행 과정

반복횟수	❶ 초기화	❷ 조건식			❸ 반복몸체		❹ 증감 연산이후
	int i = 1;	i	i < = 10	결과	printf("%3d", i);	출력	i++ 이후 i 값
1	int i = 1;	1	1 < = 10	1(참)	printf("%3d", i);	1	2
2	X	2	2 < = 10	1(참)	printf("%3d", i);	2	3
... (중간생략)							
9	X	9	2 < = 10	1(참)	printf("%3d", i);	9	10
10	X	10	10 < = 10	1(참)	printf("%3d", i);	10	11
11	X	11	11 < = 10	0(거짓)	실행 못함		

<06forbasic_10.c>

```c
#include <stdio.h>

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

    printf("\n");
    return 0;
}
```

① ② ④ ③

```
  1   2   3   4   5   6   7   8   9  10
```

for 증감 예제

- 증감: 2씩 증감

```c
#include <stdio.h>

int main()
{
    printf("Even number: ");

    for (int i = 0; i <= 10; i += 2)
    {
        printf("%d ", i);
    }
    printf("\n");
    return 0;
}
```

실행 결과

```
Even number: 0 2 4 6 8 10
```

- 증감: 1씩 감소

```c
#include <stdio.h>

int main()
{
    printf("Countdown: ");

    for (int i = 10; i > 0; i--)
    {
        printf("%d ", i);
    }
    printf("\n");
    return 0;
}
```

실행 결과

```
Countdown: 10 9 8 7 6 5 4 3 2 1
```

다양한 for 문

- 섭씨 온도 celsius를 12.46에서 10도씩 증가한 3개의 화씨(Fahrenheit) 온도를 각각 출력

<07forcel2far3.c>

```c
#include <stdio.h>
#define MAX 3
#define INCREMENT 10

int main(void)
{
    double celsius = 12.46;

    printf(" 섭씨(C) 화씨(F)\n");
    for (int i = 1; i <= MAX; i++, celsius += INCREMENT)
    {
        printf("%8.2f %8.2f\n", celsius, 9.0 / 5 * celsius + 32);
    }

    return 0;
}
```

권장 코드

```c
for (int i = 1; i <= MAX; i++)
{
    printf("%8.2f %8.2f\n", celsius, 9.0 / 5 * celsius + 32);
    celsius = celsius + INCREMENT;
}
```

증가 부분은 여러 문장을 콤마로 나열이 가능
i 증가 및 celsius 값을 10만큼 증가

실행 결과

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

반복 조건에서의 주의

- **실수 비교 연산**
  - != 연산을 <= 로 수정

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

실수 연산의 오차로 조건식 (d != 1.0)이 항상 참 -> 무한 반복 발생

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

```
0.000000 0.100000 0.200000 0.300000 0.400000 0.500000
0.600000 0.700000 0.800000 0.900000 1.000000
```

- **대입 연산자 사용: while 문 내부 실행 안됨**
  - 대입 연산자(=)를 비교 연산자(==)로 수정

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

!(i=6): false(0)이 됨 - while 내부 실행 안됨

권장 안함

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

```
1 2 3 4 5
```

권장 코드

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

```
1 2 3 4 5
```

2진수 출력 프로그램

<08forintbit.c>

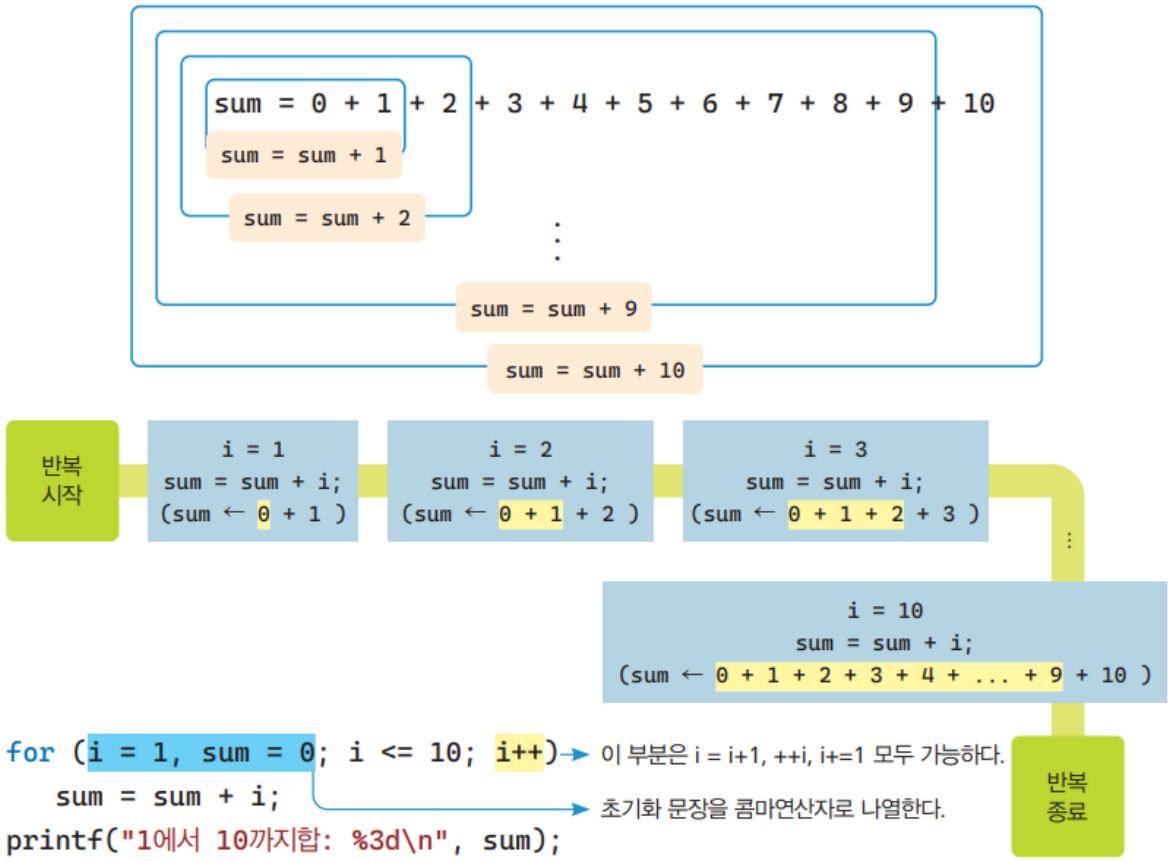
```c
#include <stdio.h>
#define TOTAL_BIT 32

int main(void)
{
    int num = 0;
    printf("10진수 정수를 입력하세요: ");
    scanf("%d", &num);

    printf("%d의 %d비트 내부 값:\n", num, TOTAL_BIT);

    for (int i = TOTAL_BIT - 1; i >= 0; i--)
    {
        printf("%d", num >> i & 1);
        if (i % 8 == 0)
            printf(" ");
    }
    printf("\n");
    return 0;
}
```

실행 결과

```
10진수 정수를 입력하세요: 15
15의 32비트 내부 값:
00000000 00000000 00000000 00001111
```

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

<09forsum.c>

```c
#include <stdio.h>

int main(void)
{
    int i, sum;
    for (i = 1, sum = 0; i <= 10; i++) //++i도 가능
        sum += i; // sum = sum + i;
    printf("1 ~ 10 합: %d\n", sum);

    for (i = 1, sum = 0; i <= 10; )
        sum += i++;
    printf("1 ~ 10 합: %d\n", sum);

    for (i = 0, sum = 0; i <= 9; )
        sum += ++i;
    printf("1 ~ 10 합: %d\n", sum);

    //반복 몸체가 없는 for 문
    for (i = 1, sum = 0; i <= 10; sum += i++);
    printf("1 ~ 10 합: %d\n", sum);

    //반복 몸체가 없는 for 문
    for (i = 0, sum = 0; i <= 9; sum += ++i);
    printf("1 ~ 10 합: %d\n", sum);

    return 0;
}
```

```c
int main(void)
{
    int sum = 0;
    int i = 0;

    for (i = 1; i <= 10; i++)
    {
        sum += i;
    }

    printf("1 ~ %d까지 합: %d\n", i - 1, sum);
    return 0;
}
```

변수 i, sum은 for문 바깥에 선언하면
-> for문 이후에도 변수 i, sum에 접근 가능

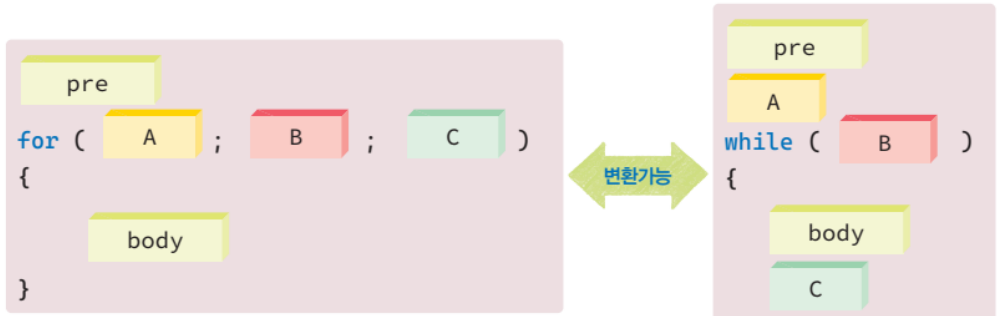
실행 결과

```
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**
  - 반복 횟수가 정해지지 않고 특정한 조건에 따라 반복을 결정하는 구문에 적합
  - for 문과 while 문은 서로 변환이 가능




```
sum = 0;
for (i = 1; i <= MAX; i++)
{
   sum += i;
}
```

변환가능

```
sum = 0;
i = 1;
while (i <= MAX)
{
   sum += i;
   i++;
}
```

sum += i++; 로 가능

1에서부터 입력한 양수까지의 합을 구하는 for와 while (실습)

<10inputsum.c>

```c
#include <stdio.h>

int main(void)
{
    int i, sum = 0, max;
    printf("양의 정수 입력 >> ");
    scanf("%d", &max);

    for (i = 1; i <= max; i++) //++i도 가능
        sum += i; // sum = sum + i;
    printf("\nfor 문으로 구한 1에서 %d까지 합: %3d\n", max, sum);

    i = 1, sum = 0;
    while (i <= max)
    {
        sum += i; // sum = sum + i;
        i++;      // ++i도 가능
    }
    printf("while 문으로 구한 1에서 %d까지 합: %3d\n", max, sum);

    return 0;
}
```

실행 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 (실습)

<lab2digit.c>

```c
#include <stdio.h>

int main(void)
{
    int input = 0, digit = 0;
    int divider = 100;

    printf("양의 정수[100~999] 입력 : ");
    scanf("%d", &input);
    do
    {
        digit = input / divider;
        input %= divider;
        printf("%3d단위 출력: %d\n", divider, digit);
        divider /= 10;
    } while (divider >= 1);

    return 0;
}
```

실행 결과

```
양의 정수[100~999] 입력 : 678
100단위 출력: 6
 10단위 출력: 7
  1단위 출력: 8
```

분기문

- **분기문**
  - 정해진 부분으로 바로 실행을 이동(jump)하는 기능을 수행
  - **break, continue, goto, return 문**

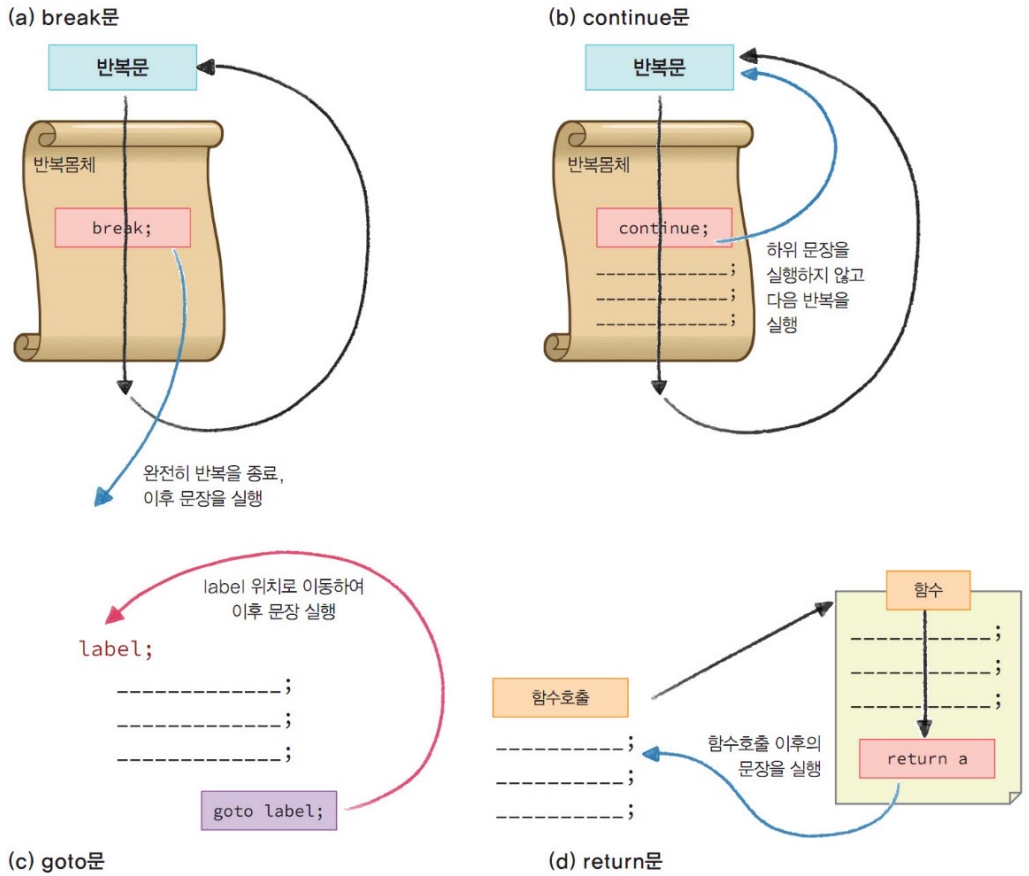
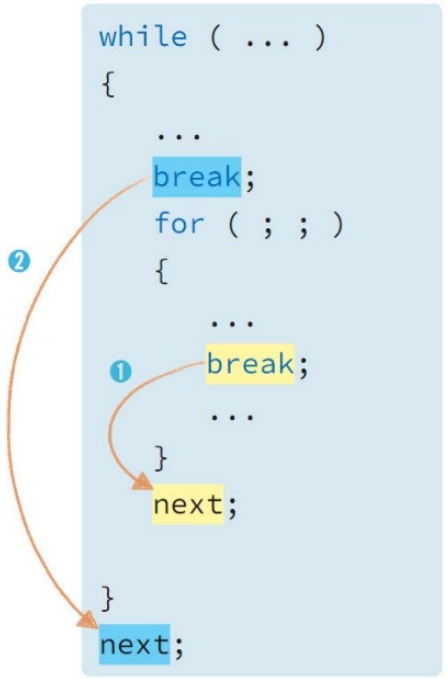


그림 7-26 여러 분기문의 개념

반복의 중단 break (실습)

- **break 문장**
  - 반복 내부에서 반복을 종료
  - 반복문이 중첩되어 있다면
    - ➢ break를 포함하는 가장 근접한 내부 반복을 종료
    - ➢ 하나의 반복문만 빠져나감



<11break.c>

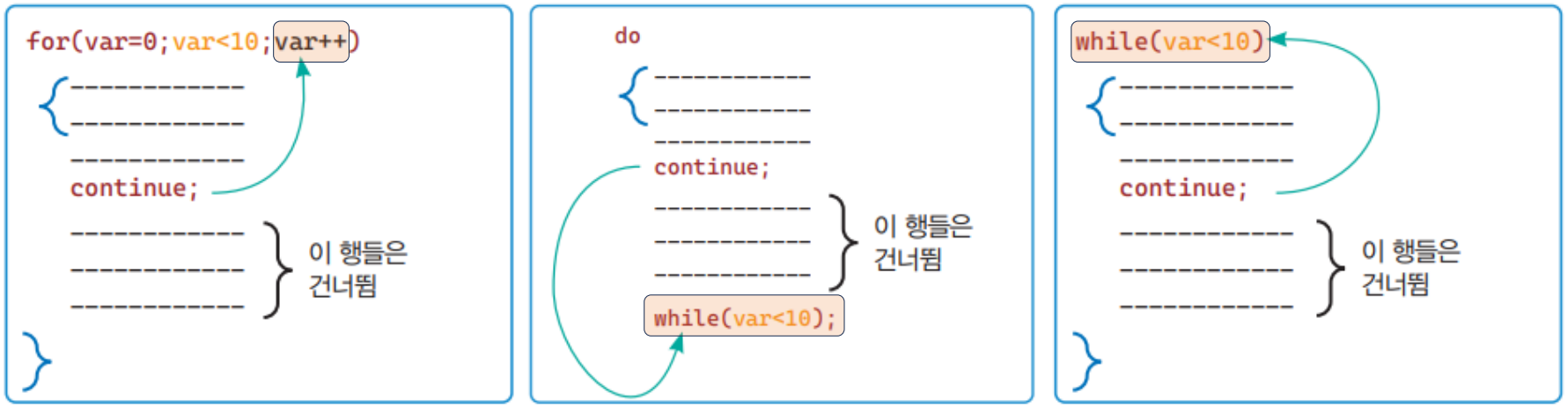
```c
#include <stdio.h>

int main(void)
{
    int input;
    while (1)
    {
        printf("양의 정수 또는 0[종료] 입력 후 [Enter] >> ");
        scanf("%d", &input);
        if (input == 0)
            break;
        printf("입력한 정수 %d: 16진수 %#x\n", input, input);
    }
    puts("종료");

    return 0;
}
```

while(1): 무한 반복

```
양의 정수 또는 0[종료] 입력 후 [Enter] >> 10
입력한 정수 10: 16진수 0xa
양의 정수 또는 0[종료] 입력 후 [Enter] >> 0
종료
```

반복의 계속: continue

- **continue 문**
  - continue 이후의 문장은 실행되지 않고 건너뜀
  - 반복문 while과 do while 반복 내부
    - ➢ continue를 만나면 **조건검사로 이동하여 실행**
  - 반복문 for 문
    - ➢ continue 문을 만나면 **증감 부분으로 이동하여 다음 반복 실행**



3으로 나누어지지 않는 정수 출력 (실습)

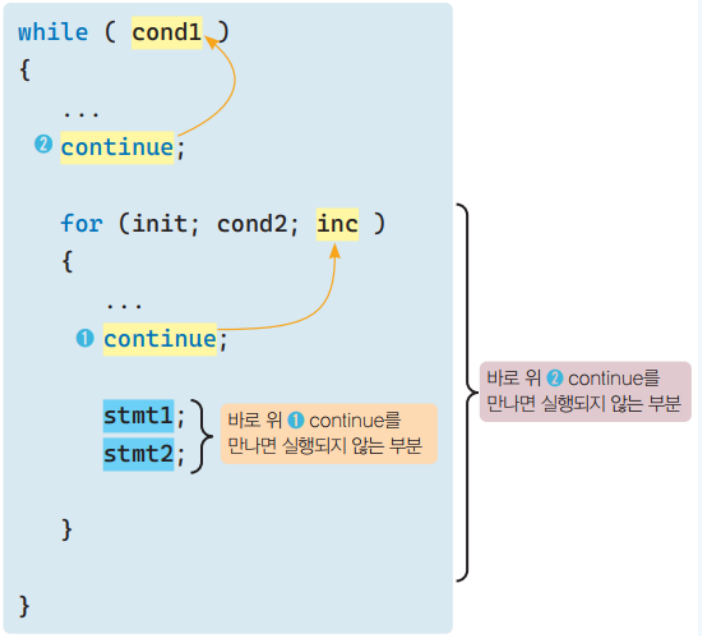
<12continue.c>

```c
#include <stdio.h>

int main(void)
{
    const int MAX = 15;

    printf("1에서 %d까지 정수 중에서 3으로 나누어 떨어지지 않는 수\n", MAX);
    for (int i = 1; i <= MAX; i++)
    {
        if (i % 3 == 0)
            continue;
        printf("%3d", i);
    }
    puts("");

    return 0;
}
```

i의 값이 3으로 나누어지면, 다음 반복을 위해 i++로 이동

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

continue는 자신이 속한 가장 근접한 반복문에서 다음 반복을 실행



goto 문

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

실습예제 7-13 | Prj13 | 13goto.c | goto를 사용해 1에서 10까지의 정수 출력 | 난이도: ★

```
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

결과

```
 1  2  3  4  5  6  7  8  9 10
종료
```

무한 반복(무한 루프)

- **무한 반복(infinite loop)**
  - 서버용 프로그램: 종료가 없이 항상 서비스 제공
  - 특정 메뉴를 선택했을 때 프로그램 종료

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

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

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

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

무한 반복 예제: 정수의 8진수와 16진수 변환과 break로 종료 (실습)

- **양의 정수를 입력하면 입력된 정수와 함께 8진수와 16진수 형태를 출력**
  - 0 또는 음의 정수를 입력하면 종료

<14octhex.c>

```c
#include <stdio.h>

int main(void)
{
    int input;
    while (1)
    {
        printf("양의 정수 입력 후 [Enter] (음수 또는 0: 종료) >> ");
        scanf("%d", &input);

        if (input <= 0)
            break;

        printf("정수 %d: 8진수 %#o 16진수 %#x\n", input, input, input);
    }
    return 0;
}
```

무한 반복 종료 조건

```
양의 정수 입력 후 [Enter] (음수 또는 0: 종료) >> 3
정수 3: 8진수 03  16진수 0x3
양의 정수 입력 후 [Enter] (음수 또는 0: 종료) >> 5
정수 5: 8진수 05  16진수 0x5
양의 정수 입력 후 [Enter] (음수 또는 0: 종료) >> 0
```

LAB 양의 정수의 소수 여부를 판단해 출력

- **입력된 정수가 소수(prime number)인 지를 출력**
  - 1과 자기 자신으로만 나누어지는 수 (2부터 자기 자신 이전의 정수로 나누어지지 않는 수)

<lab3primebreak.c>

```c
#include <stdio.h>

int main()
{
    int num, j;
    printf("2 이상 양의 정수 입력 >> ");
    scanf("%d", &num);

    for (j = 2; j < num; j++)
    {
        if (num % j == 0)
            break;
    }

    if (j == num)
        printf("%d 소수이다.\n", num);
    else
        printf("%d 소수가 아니다.\n", num);

    return 0;
}
```

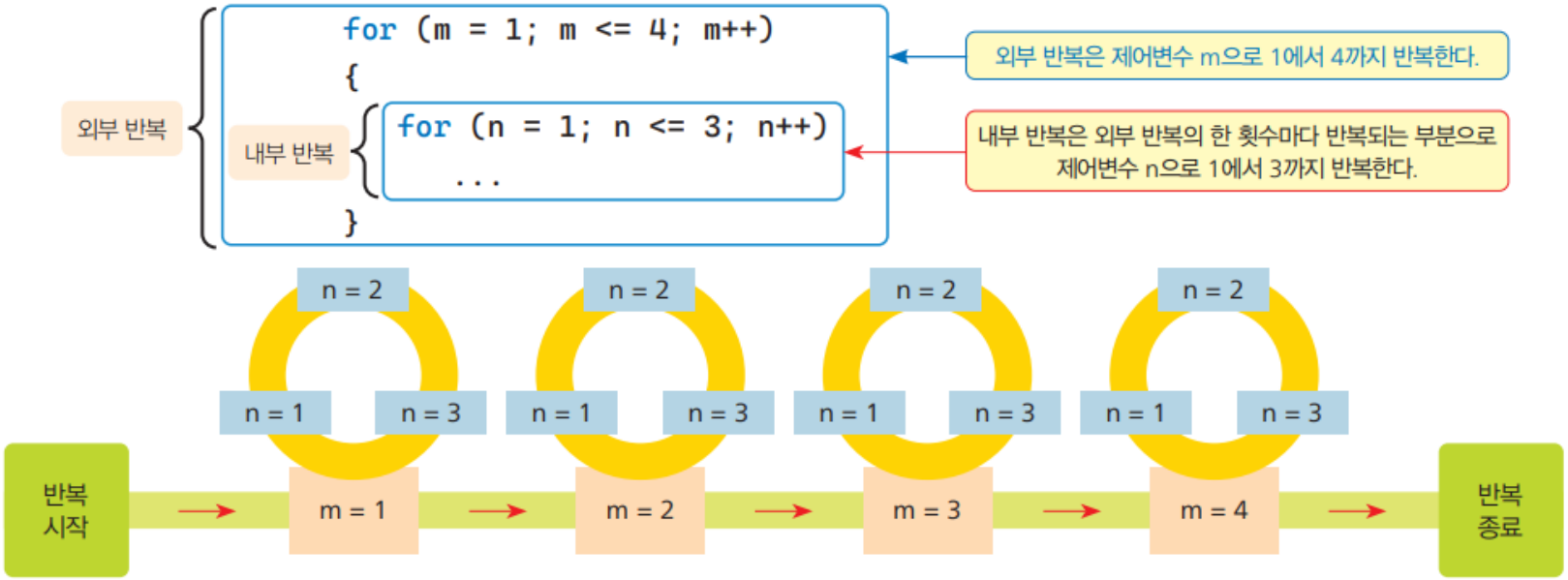
입력한 숫자가 j로 나누어 지면 반복문 종료

```
2 이상 양의 정수 입력 >> 13
13 소수이다.
```

```
2 이상 양의 정수 입력 >> 6
6 소수가 아니다.
```

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 반복 (총 반복 횟수: 5 x 7 = 35회 반복)
  - 각각의 변수 값을 출력하는 예제 프로그램

<15nestedloop.c>

```c
#include <stdio.h>

int main(void)
{
    int m, n;
    for (m = 1; m <= 5; m++)
    {
        printf("m = %-2d\n", m);
        for (n = 1; n <= 7; n++)
            printf("n = %-3d", n);
        puts("");
    }

    return 0;
}
```

m=1 일때
- n은 1~7까지 반복

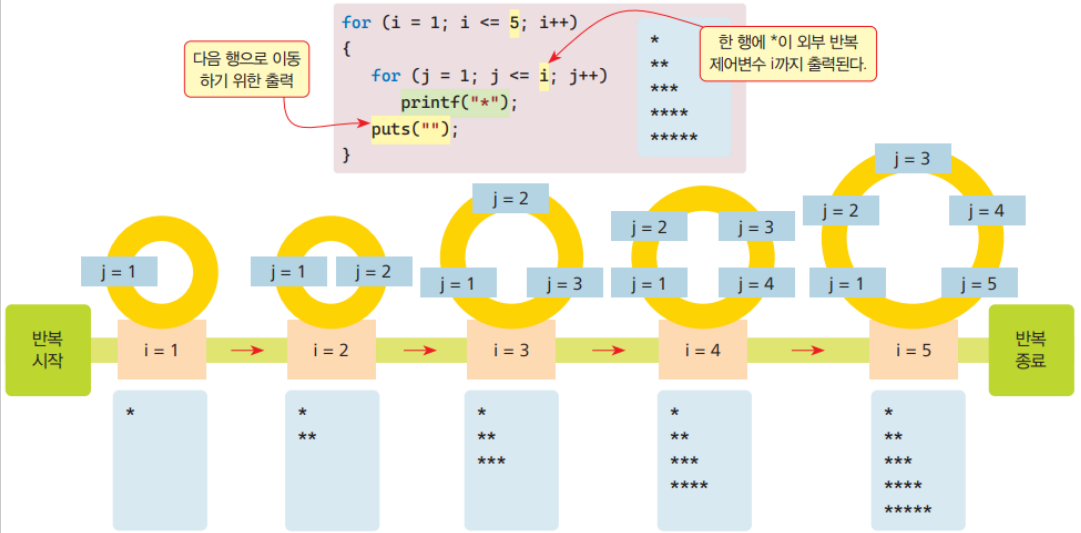
```
m = 1
n = 1  n = 2  n = 3  n = 4  n = 5  n = 6  n = 7
m = 2
n = 1  n = 2  n = 3  n = 4  n = 5  n = 6  n = 7
m = 3
n = 1  n = 2  n = 3  n = 4  n = 5  n = 6  n = 7
m = 4
n = 1  n = 2  n = 3  n = 4  n = 5  n = 6  n = 7
m = 5
n = 1  n = 2  n = 3  n = 4  n = 5  n = 6  n = 7
```

내부 반복이 외부 반복에 의존 (실습)

- **외부 반복에서 변수 i는 1 에서 5까지 반복**
  - 내부 반복에서 제어 변수 j는 1에서 외부 반복의 제어 변수 i까지 반복

<16triangle.c>

```c
#include <stdio.h>

int main(void)
{
    const int MAX = 5;
    int i, j;

    for (i = 1; i <= MAX; i++)
    {
        for (j = 1; j <= i; j++)
            printf("*");
        puts("");
    }

    return 0;
}
```




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

삼중 중첩 반복 (실습)

- **양의 정수를 입력 받아 합을 출력**
  - 0 또는 음수를 입력할 때까지 계속 수행하는 프로그램
  - 1부터 input 숫자까지의 합을 개선

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

<17loopsum.c>

```c
#include <stdio.h>

int main(void)
{
    int input = 0, sum=0;
    int i, j;
    do
    {
        printf("양의 정수 또는 0(종료)을 입력: ");
        scanf("%d", &input);
        for (i = 1; i <= input; i++)
        {
            for (j = 1, sum = 0; j <= i; j++)
            {
                printf("%d", j);
                j == i ? printf(" = ") : printf(" + ");
                sum += j;
            }
            printf("%d\n", sum);
        }
    } while (input > 0);

    puts("종료합니다.");
    return 0;
}
```

LAB 구구단 출력

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

```c
#include <stdio.h>
#define MAX 9

int main(void)
{
    for (int i = 2; i <= MAX; i++)
    {
        for (int j = 1; j <= MAX; j++)
        {
            printf("%d * %d = %2d  ", i, j, i * j);
        }
        printf("\n");
    }
    return 0;
}
```

i=2는 고정, j는 1~9까지 출력 및 계산

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

Lab. 구구단 세로 출력 (실습)

- 아래 출력 결과와 같이 구구단(multiplication table)을 세로로 출력하세요.
  - 중첩 for 문의 i, j 변경

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

Lab. 구구단 세로 출력 (실습)

<lab5multitable.c>

```c
#include <stdio.h>
#define MAX 9

int main()
{
    for (int i = 1; i <= MAX; i++)
    {
        for (int j = 2; j <= MAX; j++)
        {
            printf("%d * %d = %2d  ", j, i, j * i);
        }
        printf("\n");
    }
}
```

Questions?