



컴퓨터 프로그래밍

Computer Programming

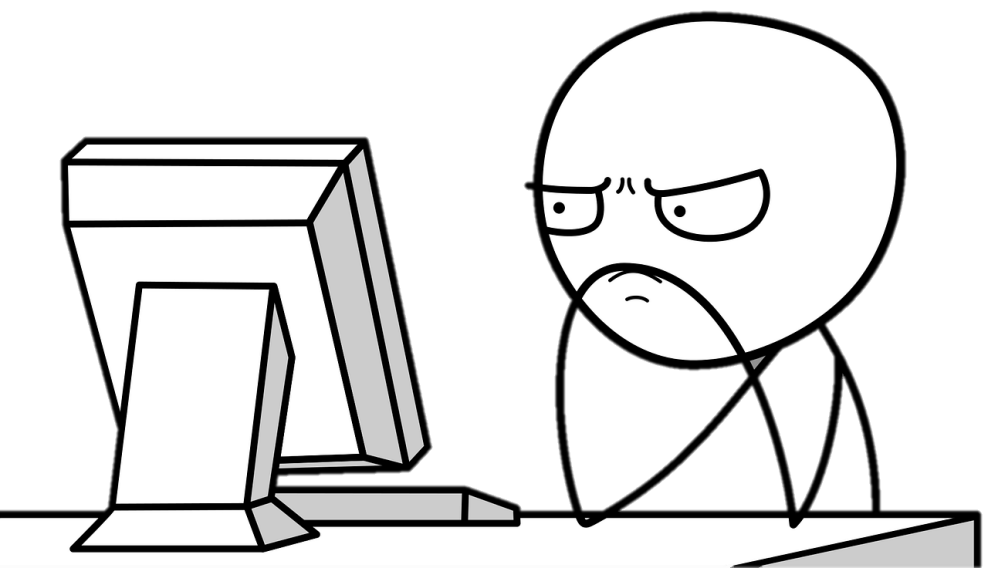
Assembly, C, C++, C#, Clojure, Erlang, Groovy, Java, JavaScript, Haskell, Ruby, Scala, Python, PHP, Perl, Lua, Swift, Kotlin, Rust, Go, JavaScript, TypeScript, Dart, Bash, AppleScript, and many others.

07 반복

목차

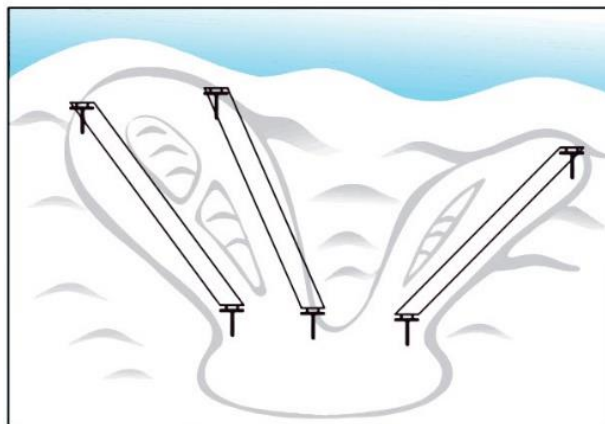
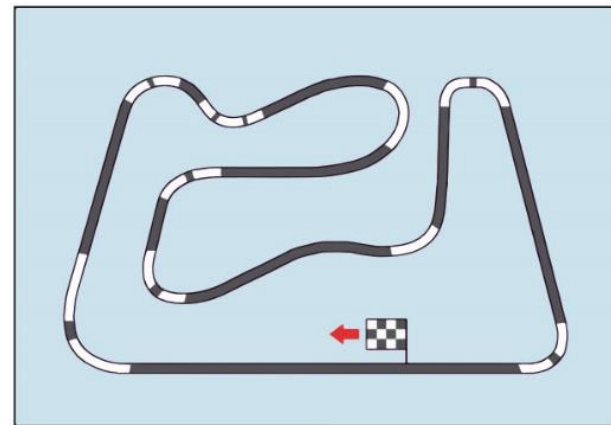
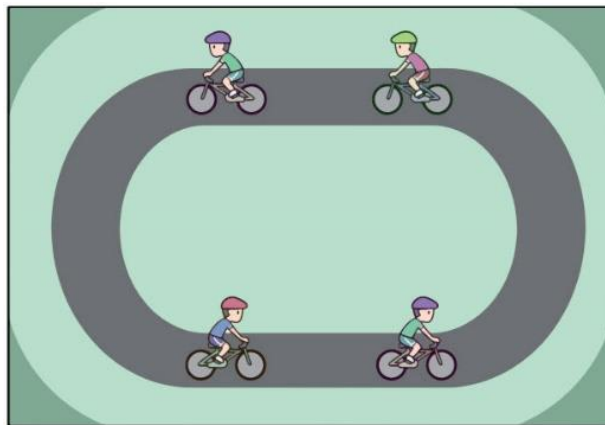
1. 반복과 while 문
2. do while 문과 for 문
3. 분기문
4. 중첩된 반복문

1. 반복과 while 문



반복

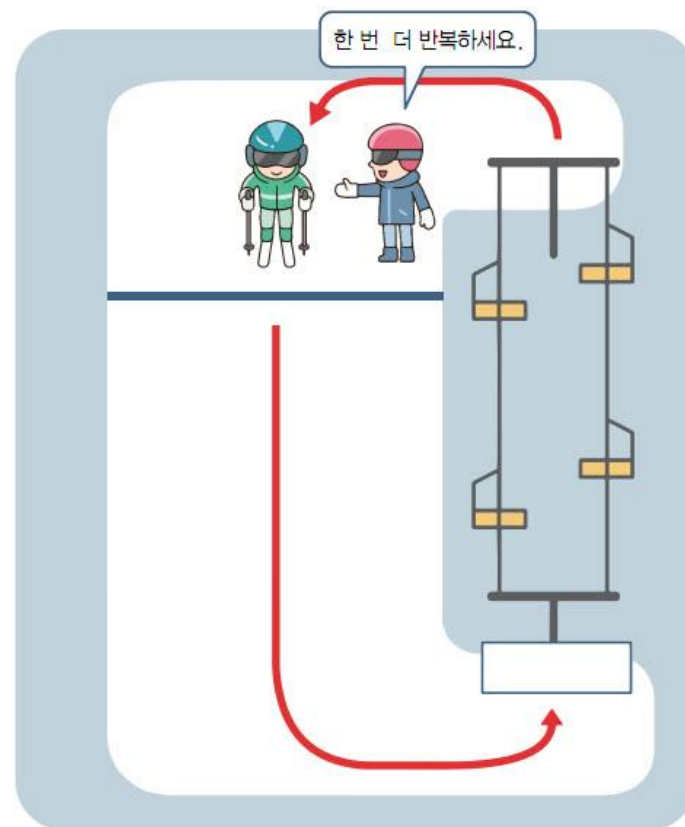
- 반복^{repetition}
 - 같거나 비슷한 일을 여러 번 수행하는 작업
- 순환^{loop, 루프}
 - 반복과 같은 의미
 - 롤러코스터의 원형 궤도처럼 원래 고리 또는 순환이라는 의미가 루프^{loop}



while 반복문

- 반복문 while은 단순한 숫자의 반복이 아니라 반복할 때마다 조건을 따지는 반복문
- 조건식이 반복몸체 앞에 위치

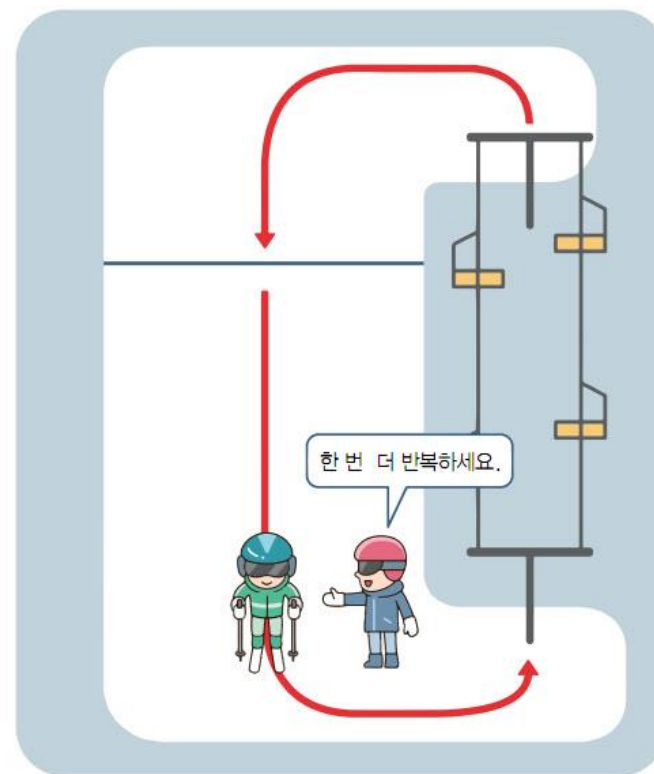
```
while ( <반복조건> )  
{  
    //반복몸체(loop body);  
    <해야할 일>;  
}
```



do while 반복문

- 조건식이 반복몸체 뒤에 위치하므로 처음에 조건을 검사할 수는 없음
- 무조건 한 번 실행 한 후 조건을 검사하고 이때 조건식이 참(0이 아니면)이면 반복을 더 실행

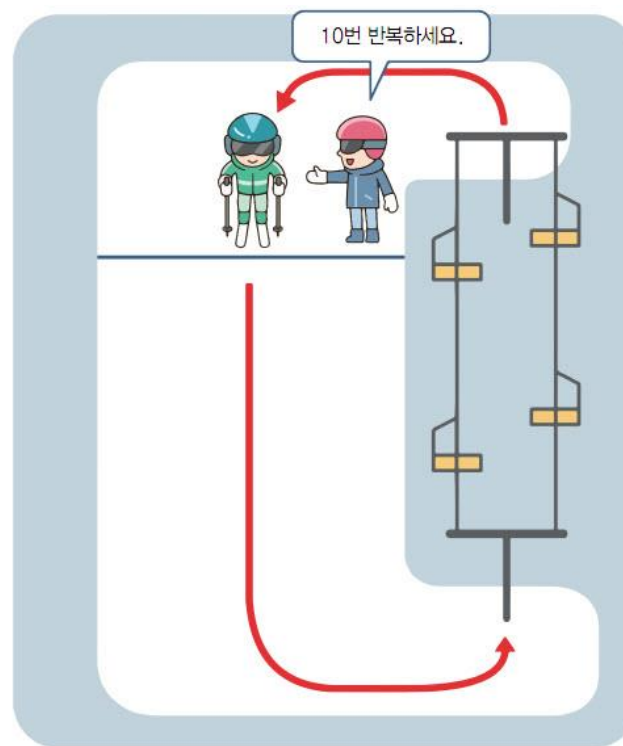
```
do  
{  
    //반복몸체(loop body);  
    <해야할 일>;  
} while ( <반복조건> );
```



for 반복문

- 반복문 for는 숫자로 반복하는 횟수를 제어하는 반복문
- 명시적으로 반복 횟수를 결정할 때 주로 사용

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



반복 구문의 필요성

- 동일하거나 또는 약간 다른 출력을 반복
 - 함수 printf() 호출을 여러 번 반복

```
C 언어 재미있네요!  
C 언어 재미있네요!  
C 언어 재미있네요!
```

```
1  
2  
3  
4  
5
```

while 문 구조와 제어흐름

- 문장 `while (cond) stmt;`
- `cond`를 평가하여 0이 아니면(참) 반복몸체인 `stmt1`를 실행
 - 다시 반복조건 `cond`를 평가하여 `while` 문 종료 시까지 반복
 - 반복은 `cond`가 0(거짓)이 될 때 계속
- 반복이 실행되는 `stmt`를 반복몸체(repetition body)라 부름
 - 필요하면 블록으로 구성
- `while` 문은 `for`나 `do while` 반복문보다 간단

while 문 구조와 제어흐름

while 문

```
while (cond)
    stmt;
next;
```

블록으로 구성 가능

```
int count = 1;
```

반복의 횟수를 결정하는
변수인 제어변수이다.

```
while (count <= 3)
```

```
{
```

```
    printf("C 언어 재미있네요!\n");
    count++;
```

```
};
```

이 부분을 반복몸체라 한다.

stmt
반복몸체

0이 아니면(참)

cond
반복검사

0이면(거짓)

next;
다음문장

Source Code #01: whilebasic.c

```
1 // file: whilebasic.c
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     int count = 1;
8
9     while (count <= 3)
10    {
11        printf("C 언어 재미있네요!\n");
12        count++;
13    };
14    printf("\n제어변수 count => %d\n", count);
15
16    return 0;
17 }
```

```
C 언어 재미있네요!
C 언어 재미있네요!
C 언어 재미있네요!
```

```
제어변수 count => 4
```

- 반복횟수를 제어하는 변수를 제어변수
- 조건식 (count <= 3)으로 while의 조건을 구성
- 반복몸체
 - 출력문 printf("C 언어 재미있네요!\n");
 - count++;를 삽입

```
printf("C 언어 재미있네요!\n");
printf("C 언어 재미있네요!\n");
printf("C 언어 재미있네요!\n");
```

```
int count = 1;

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

Source Code #02: whilenumber.c

```
1 // file: whilenumber.c
2
3 #include <stdio.h>
4 #define MAX 5
5
6 int main(void)
7 {
8     int n = 1;
9
10    while (n <= MAX)
11        printf("%d\n", n++);
12
13    printf("\n제어변수 n => %d\n", n);
14
15    return 0;
16 }
```

```
1
2
3
4
5
제어변수 n => 6
```

- 후위 증가연산자 `n++`의 연산값은 증가되기 이전 값이므로 반복이 시작된 1부터 5까지 출력
 - 연산식 `n++`에서 5가 출력되면, `n`이 6이 되고, while 조건식 (`6 <= 5`) 값이 0이 되어 while 문장을 빠져 나옴

```
int n = 1;
//정수값을 1씩 증가시키면서 출력 반복
printf("%d\n", n++);
printf("%d\n", n++);
printf("%d\n", n++);
printf("%d\n", n++);
printf("%d\n", n++);
```

```
#define MAX 5
...
int n = 1;
while (n <= MAX)
    printf("%d\n", n++);
```

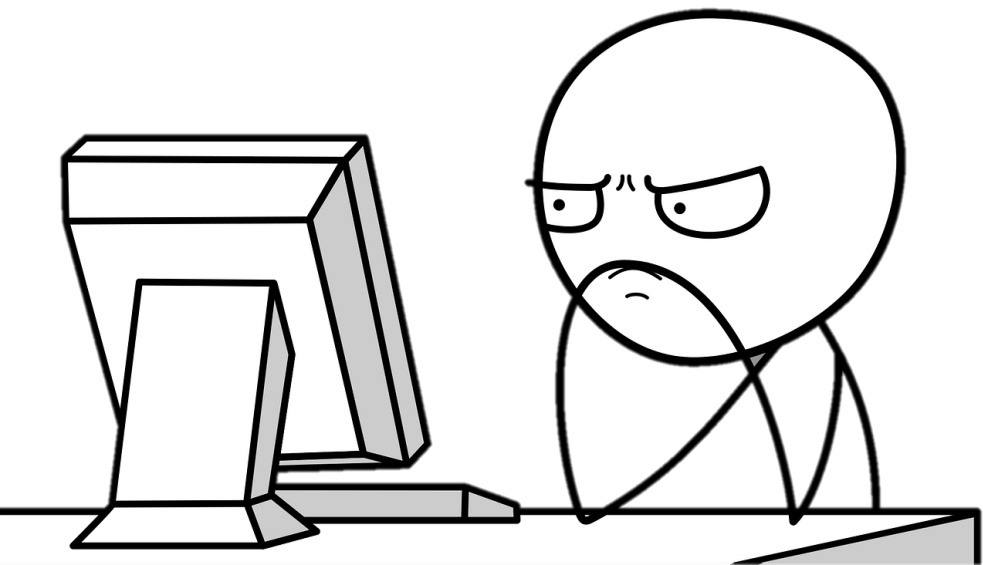
출력하고 싶은 최대 정수를 수정할 수 있다.

Lab #01: 0부터 20까지 3의 배수 출력

- 반복문 while 문을 사용
 - 0부터 20까지의 3의 배수를 출력하는 프로그램
 - 정수는 모두 한 줄에 출력
- 결과
 - 0 3 6 9 12 15 18

```
1 // file: whilelab.c
2
3 #include <stdio.h>
4 #define MAX 20
5
6 int main(void)
7 {
8     int n = 0;
9
10    while (n <= MAX) {
11        printf("%4d", n);
12        n += 3;
13    }
14    puts("");
15
16    return 0;
17 }
```

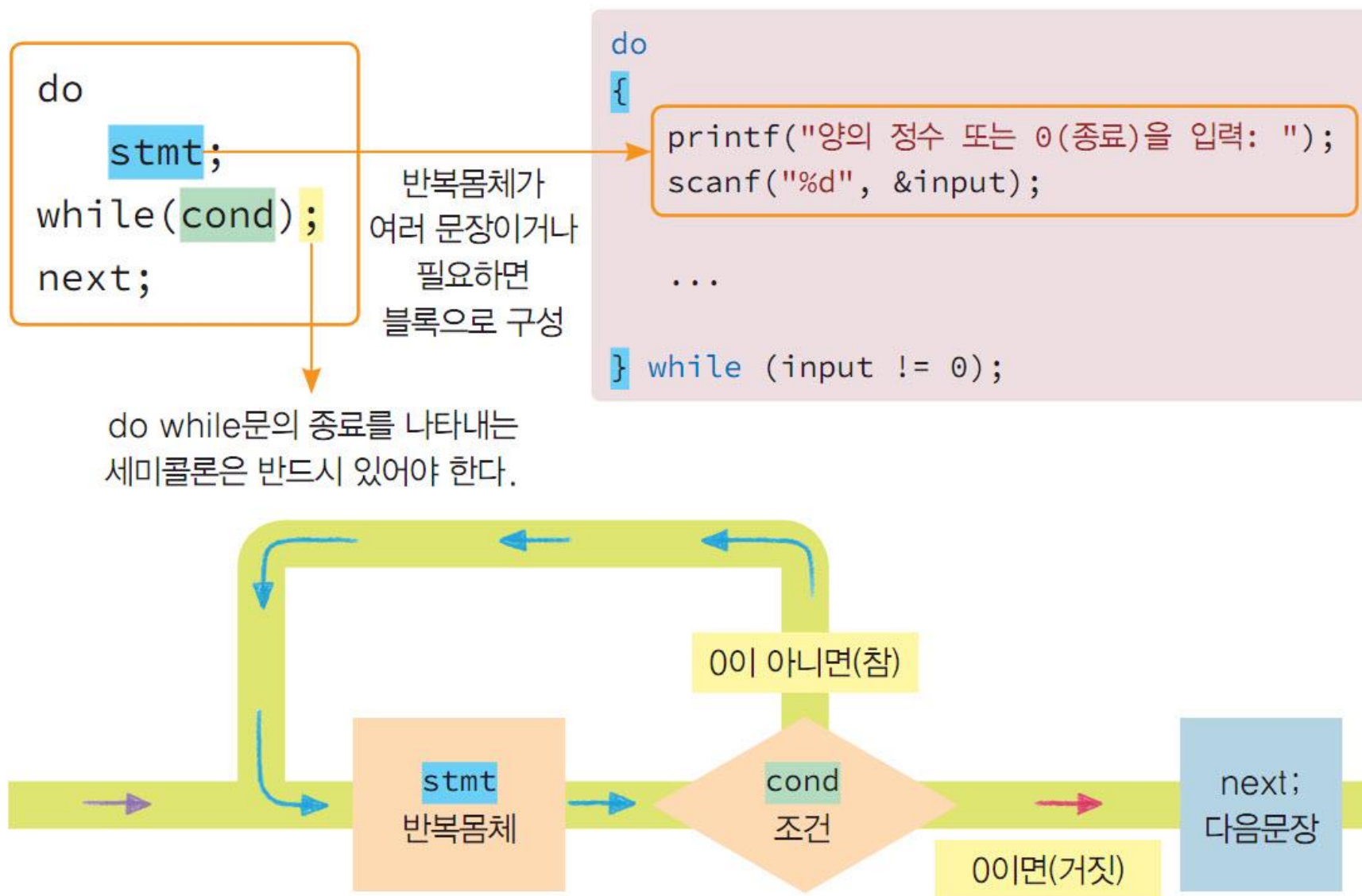
2. do while 문과 for 문



do while문 구조와 제어흐름

- 문장 `do stmt; while (cond)`
- 가장 먼저 `stmt`를 실행한 이후 반복조건인 `cond`를 평가
 - 0이 아니면(참) 다시 반복몸체인 `stmt`를 실행
 - 0이면(거짓) `do while` 문을 종료
- 반복몸체에 특별히 분기 구문이 없는 경우
 - `do while`의 몸체는 적어도 한 번은 실행
- 주의
 - `while` 이후의 세미콜론은 반드시 필요

do while문 구조와 제어흐름



Source Code #03: dowhile.c

```
1 // file: dowhile.c
2 #define _CRT_SECURE_NO_WARNINGS //scanf() 오류를 방지하기 위한 상수 정의
3
4 #include <stdio.h>
5
6 int main(void)
7 {
8     int input;
9
10    do
11    {
12        printf("정수 또는 0(종료)을 입력: ");
13        scanf("%d", &input);
14    } while (input != 0); //while (input);
15
16    puts("종료합니다.");
17
18    return 0;
19 }
```

```
정수 또는 0(종료)을 입력: 7
정수 또는 0(종료)을 입력: -3
정수 또는 0(종료)을 입력: 5
정수 또는 0(종료)을 입력: 0
종료합니다.
```

- 표준입력으로 받은 정수가 양수 또는 음수이면 계속 입력을 반복
- 입력한 수가 0이면 프로그램이 종료되는 프로그램
- 입력 후에 반복 검사를 진행하는 처리 과정으로 do while 문으로 구현이 적합
- 센티널 값^{sentinel value}
 - 반복의 종료를 알리는 특정한 자료값
 - do while 반복문은 이러한 센티널 값 검사에 유용하게 사용

Lab #02: 백단위 정수의 각 자릿수 출력

- 백단위의 양의 정수를 입력 받아 각 각 100단위, 10단위, 1단위 값을 출력하는 프로그램
 - 정수는 100에서 999 사이의 정수를 입력, 나누기 /와 나머지 연산자 %를 잘 활용
 - 정수의 나누기 연산자 /의 결과는 정수 몫으로, $673 / 100$ 은 6
 - 정수의 나머지 연산자 %의 결과는 나머지 값으로, $673 \% 100$ 은 73
- 결과
 - 양의 정수[100~999] 입력 : 853
 - 100단위 출력: 8
 - 10단위 출력: 5
 - 1단위 출력: 3

```
1 // file: forlab.c
2 #define _CRT_SECURE_NO_WARNINGS
3
4 #include <stdio.h>
5
6 int main(void)
7 {
8     int input = 0, result = 0, digit = 0;
9     int devider = 100;
10
11     printf("양의 정수[100~999] 입력 : ");
12     scanf("%d", &input);
13     result = input;
14     do
15     {
16         digit = result % devider;
17         result /= devider;
18         printf("%3d단위 출력: %d\n", devider, digit);
19     } while (devider >= 1);
20
21     return 0;
22 }
23
```

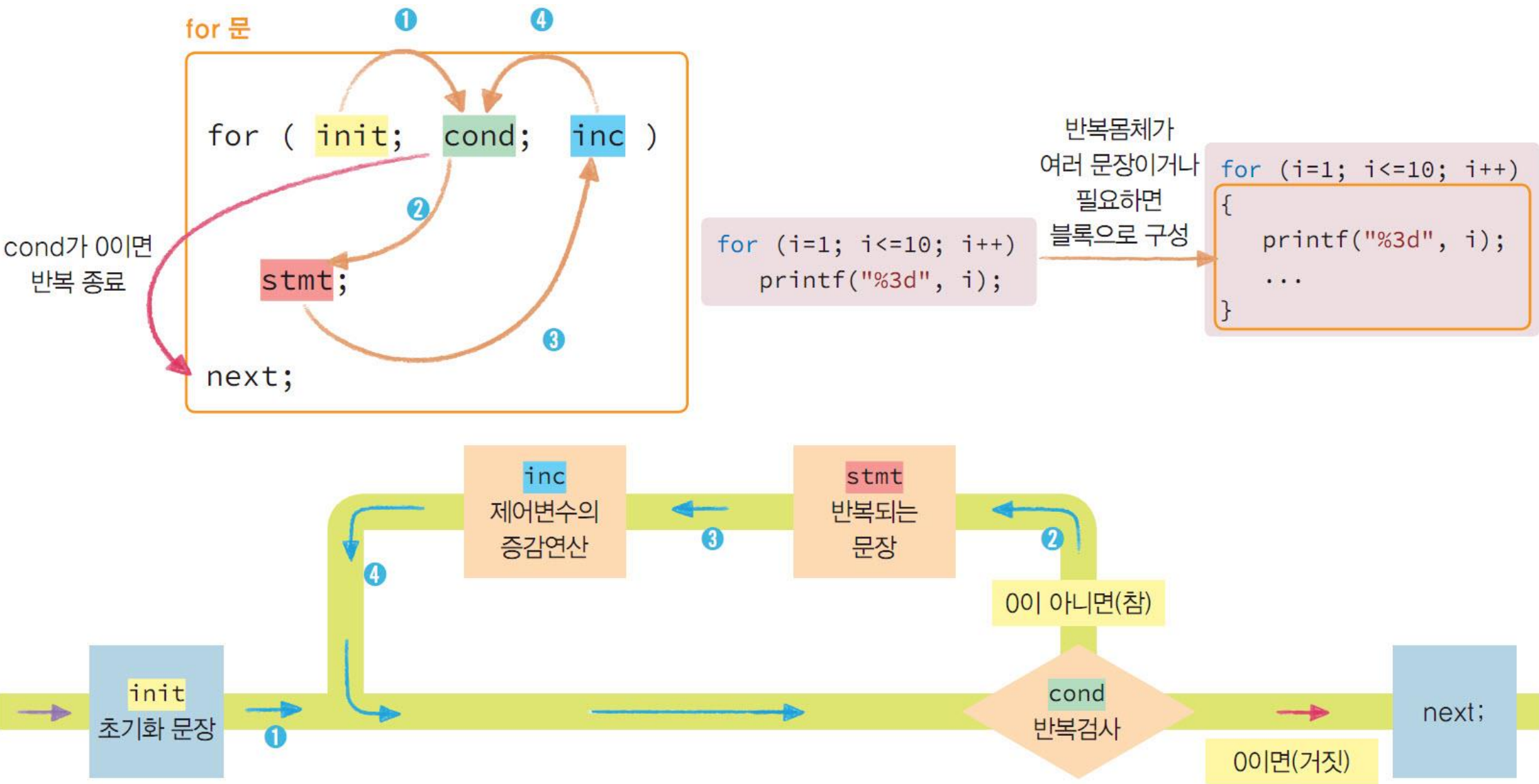
for문 구조와 제어흐름

- 반복문 for (init; cond; inc) stmt;
- 초기화^{initialization}; 반복조건을 검사; 제어변수의 증감^{increment}을 수행
- 주의점
 - for(;;)의 괄호 내부에서 세미콜론으로 구분되는 항목은 모두 생략 가능
 - 2개의 세미콜론은 반드시 필요
 - 반복조건 cond를 아예 제거하면 반복은 무한히 계속
 - 반복할 문장인 반복몸체 stmt가 여러 개라면 반드시 블록으로 구성'

for문 구조와 제어흐름

- `for (i=1; i<=10; i++)`
 `printf("%3d", i);`
- 1부터 10까지 출력
 - ❶ 초기화를 위한 `init`를 실행한다. 이 `init`는 단 한번만 수행된다.
 - ❷ 반복조건 검사 `cond`를 평가
 - 0이 아닌 결과값(참)이면 반복문의 몸체에 해당하는 문장 `stmt`를 실행
 - 그러나 결과값이 0(거짓)이면 `for` 문을 종료하고 다음 문장 `next`를 실행
 - ❸ 반복몸체인 `stmt`를 실행한 후 증감연산 `inc`를 실행한다.
 - ❹ 다시 반복조건인 `cond`를 검사하여 반복한다.

for문 구조와 제어흐름



for 문의 이해: 1에서 10까지 출력하는 프로그램

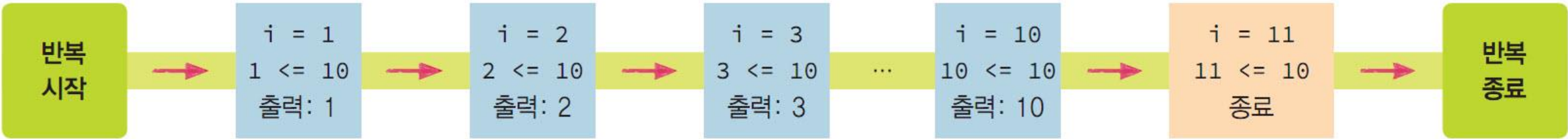
- 반복횟수를 제어하는 제어변수 i 를 1로 초기화
- 조건검사 $i \leq 10$ 를 이용하여 변수 i 를 출력
 - 변수 i 와 같이 반복의 횟수를 제어하는 변수를 제어변수

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++;  
}
```



for 문의 이해: 1에서 10까지 출력하는 프로그램

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

Source Code #04: forbasic.c

```
1 // file: forbasic.c
2
3 #include <stdio.h>
4 #define MAX 5
5
6 int main(void)
7 {
8     int i;
9
10    for (i = 1; i <= MAX; i++)
11    {
12        printf("C 언어 재미있네요! %d\n", i);
13    }
14
15    printf("\n제어변수 i => %d\n", i);
16
17    return 0;
18 }
```

- "C 언어 재미있네요!"라는 문구에 정수를 1에서 5까지 함께 출력하는 프로그램

```
C 언어 재미있네요! 1
C 언어 재미있네요! 2
C 언어 재미있네요! 3
C 언어 재미있네요! 4
C 언어 재미있네요! 5
```

```
제어변수 i => 6
```

Source Code #05: forcel2far3.c

```
1 // file: forcel2far3.c
2
3 #include <stdio.h>
4 #define MAX 3
5 #define INCREMENT 10
6
7 int main(void)
8 {
9     double celsius = 12.46;
10
11     printf(" 섭씨(C) 화씨(F)\n");
12     printf("-----\n");
13
14     for (int i = 1; i <= MAX; i++, celsius += INCREMENT)
15     {
16         printf("%8.2lf %8.2lf\n", celsius, 9.0 / 5 * celsius + 32);
17     }
18
19     return 0;
20 }
```

- 10도씩 증가하는 3 개의 섭씨온도 celsius를 화씨온도로 출력
- 섭씨와 화씨온도를 출력할 반복횟수는 매크로 상수 MAX로 정의
- 섭씨 온도의 증가 값은 매크로 상수 INCREMENT로 정의
- 섭씨온도 Celsius를 12.46으로 시작으로 3 개의 화씨온도를 각각 출력하는 for문

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

Source Code #06: forbit.c

```
1 // file: forbit.c
2 #include <stdio.h>
3
4 #define TOTAL_BIT 32
5
6 int main(void)
7 {
8     int num = 13;
9     printf("정수 %d의 %d비트 내부 값:\n", num, TOTAL_BIT);
10
11     for (int i = TOTAL_BIT - 1; i >= 0; i--)
12         printf("%d", num >> i & 1);
13
14     printf("\n");
15
16     return 0;
17 }
```

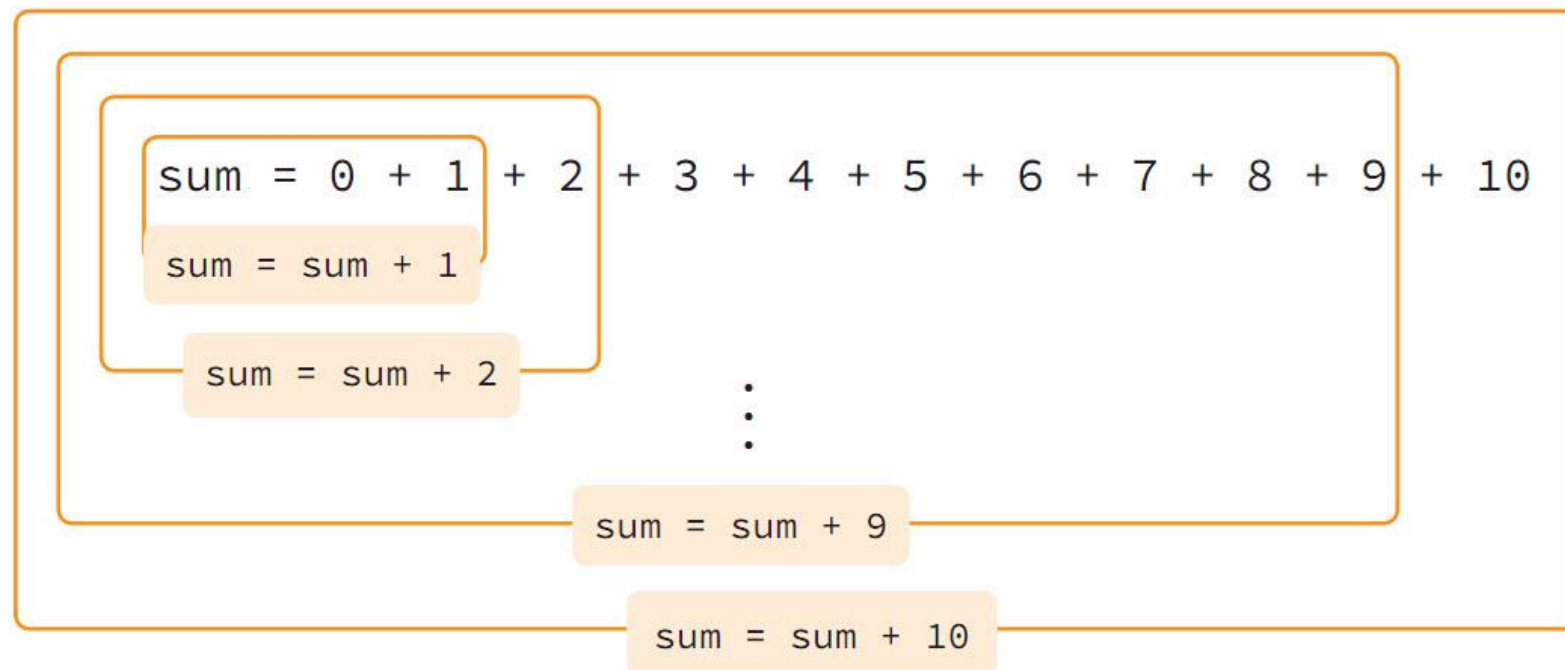
- int 형 정수의 32 비트 모두를 출력하는 프로그램
- 비트 AND 연산자 &를 사용하여 정수의 오른쪽 8비트를 출력하는 for 문

```
printf("%d", 13 >> 7 & 1); //오른쪽 8번째 비트값 출력
printf("%d", 13 >> 6 & 1);
...
printf("%d", 13 >> 2 & 1);
printf("%d", 13 >> 1 & 1);
printf("%d\n", 13 >> 0 & 1); //오른쪽 첫 비트값 출력
```

```
for (int i = 7; i >= 0; i--)
    printf("%d", 13 >> i & 1);
```

for문의 합 구하기

- for문을 이용하여 1에서 10까지 합을 구하는 모듈
 - 순회하는 제어변수 i 값을 계속 합하여 변수 sum 에 누적

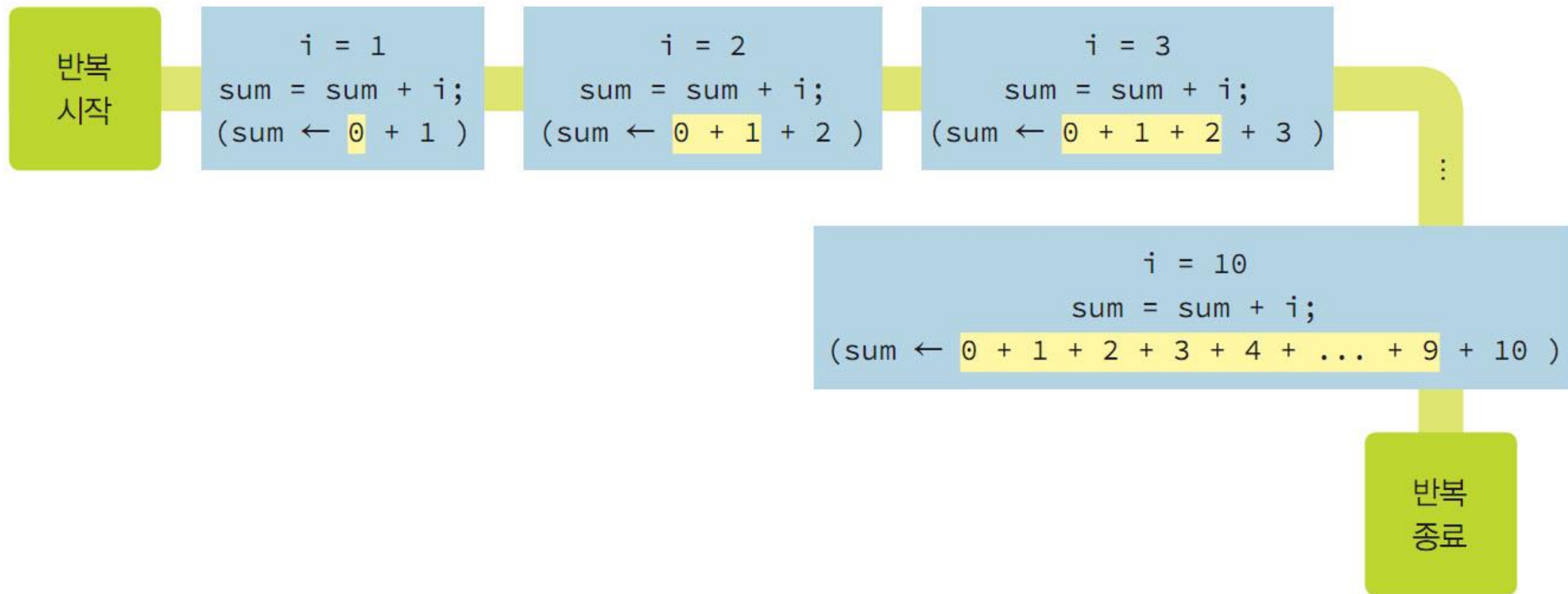


for문의 합 구하기

```
for (i = 1, sum = 0; i <= 10; i++)  
    sum = sum + i;  
printf("1에서 10까지합: %3d\n", sum);
```

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

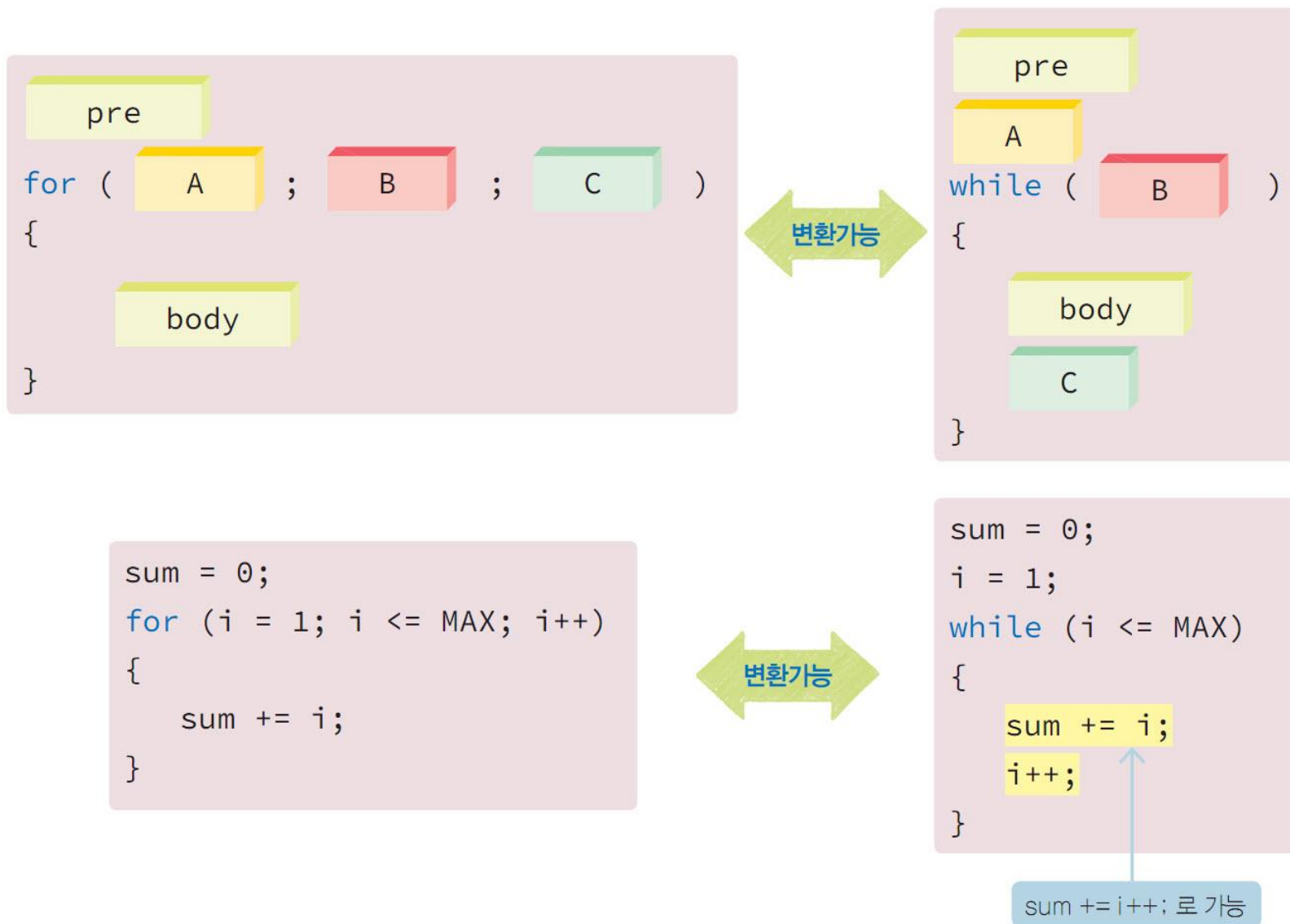
초기화 문장을 콤마연산자로 나열한다.



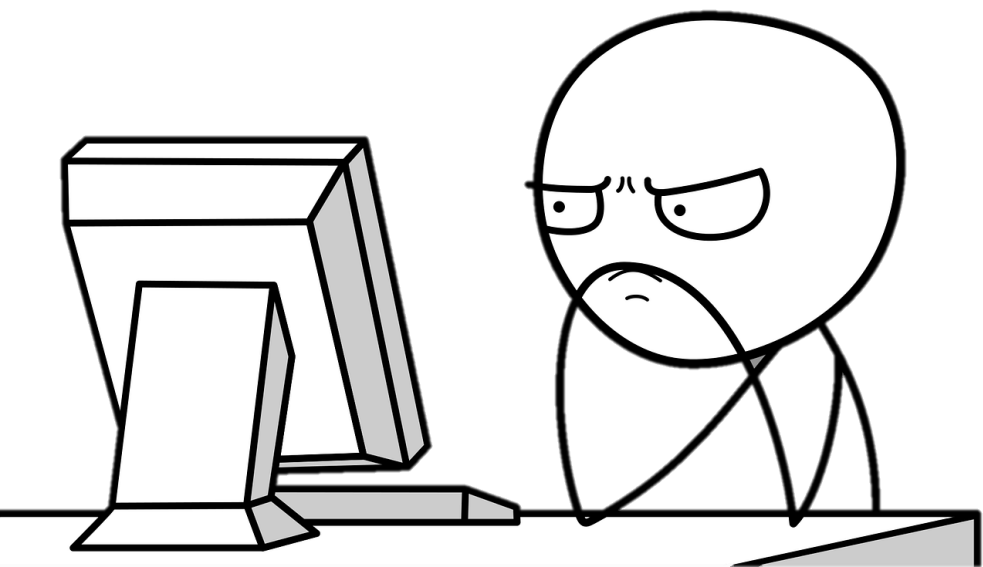
for문과 while문의 비교

- for 문은 주로 반복횟수를 제어하는 제어변수를 사용
 - 초기화와 증감부분이 있는 반복문에 적합
- while문은 문장구조가 간단하므로 다양한 구문에 이용
 - 특히 while문은 반복횟수가 정해지지 않고 특정한 조건에 따라 반복을 결정하는 구문에 적합
- for문과 while문은 서로 변환이 가능

for문과 while문의 비교



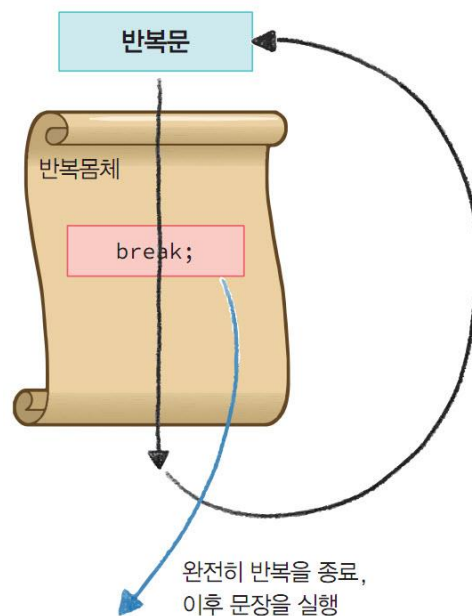
3. 분기문



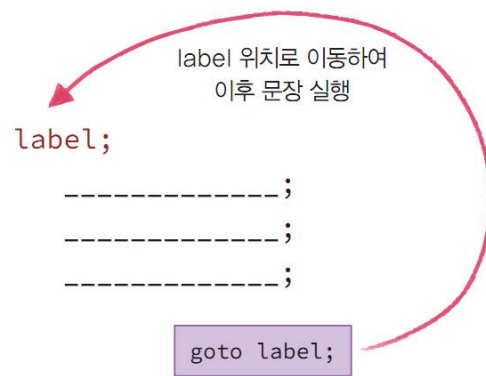
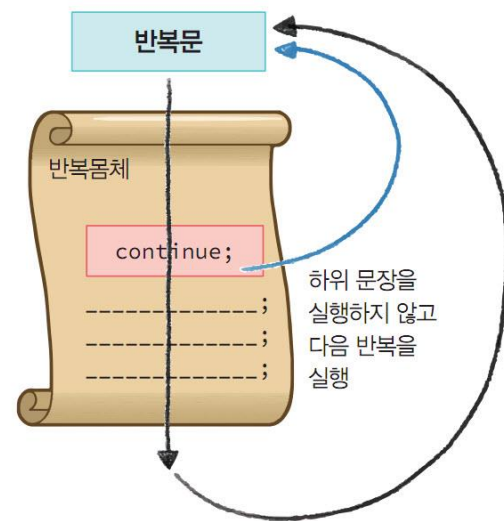
분기문

- 분기문은 정해진 부분으로 바로 실행을 이동(jump)하는 기능을 수행
- C가 지원하는 분기문: break, continue, goto, return 문

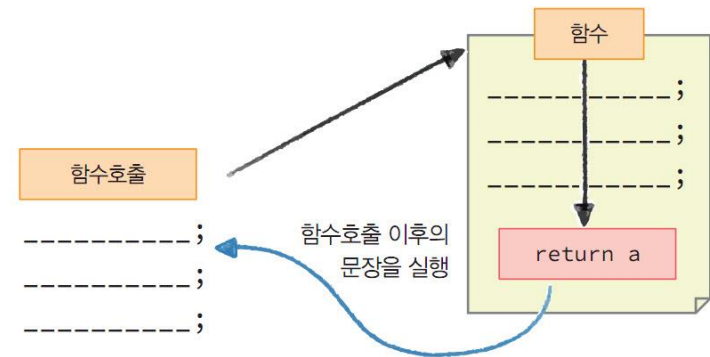
(a) break문



(b) continue문



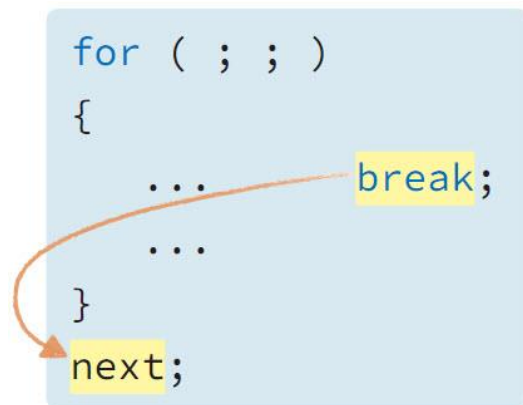
(c) goto문



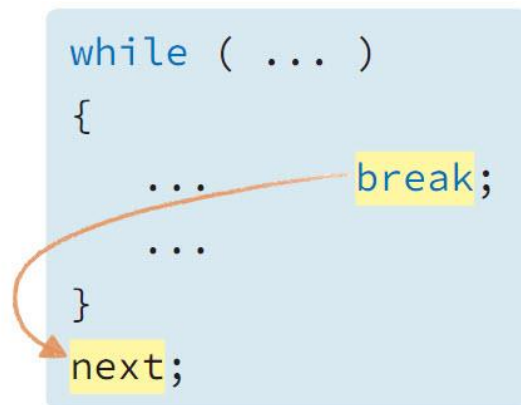
(d) return문

반복의 중단 break

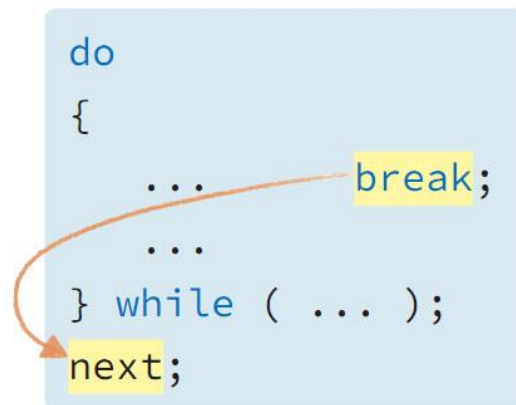
```
for ( ; ; )  
{  
    ...  
    break;  
    ...  
}  
next;
```



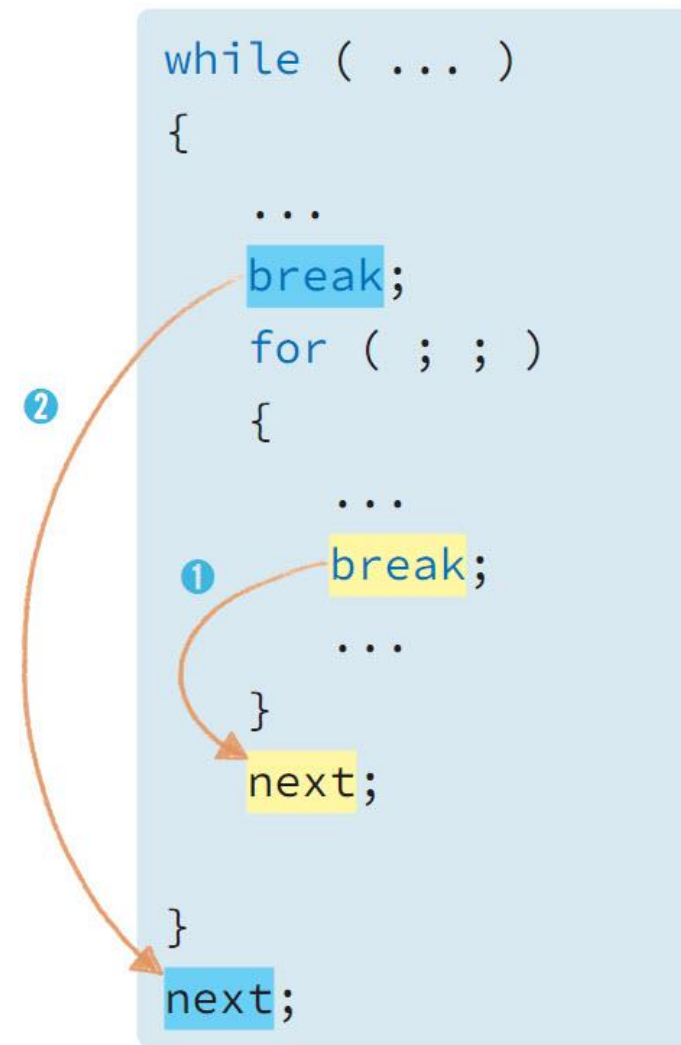
```
while ( ... )  
{  
    ...  
    break;  
    ...  
}  
next;
```



```
do  
{  
    ...  
    break;  
    ...  
} while ( ... );  
next;
```

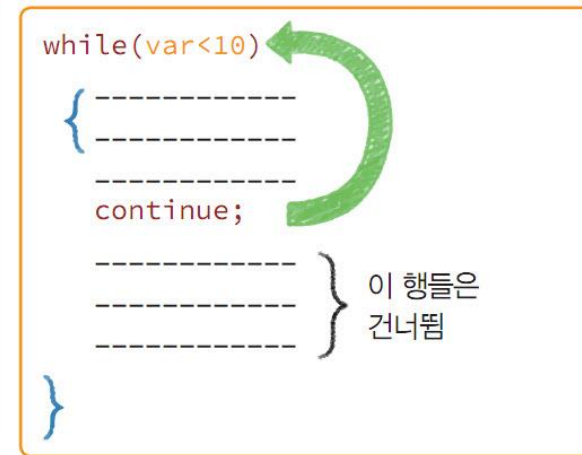
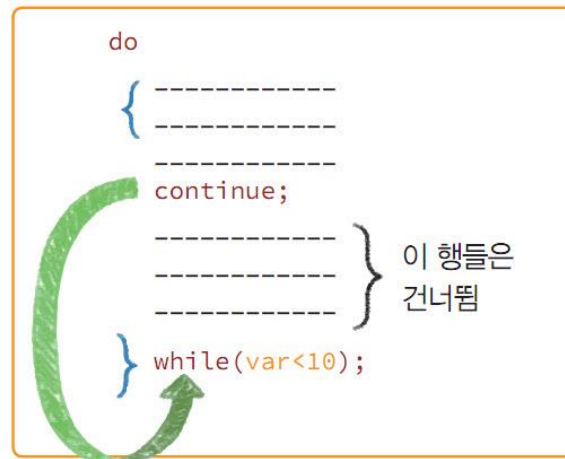
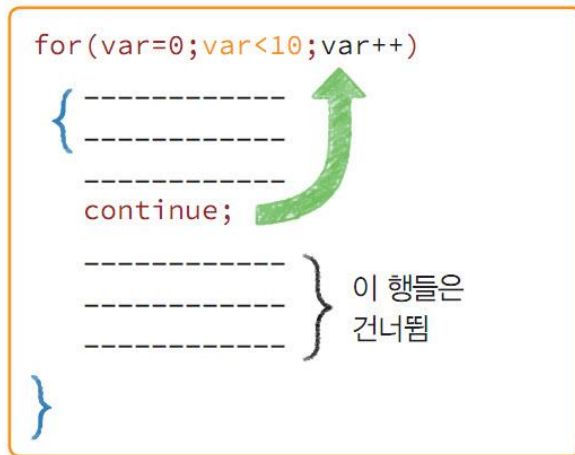


```
while ( ... )  
{  
    ...  
    break;  
    for ( ; ; )  
    {  
        ...  
        break;  
        ...  
    }  
    next;  
}  
next;
```



continue 문

- continue 문이 위치한 이후의 반복문체의 나머지 부분을 실행하지 않고 다음 반복을 계속 유지하는 문장
 - continue 이후의 문장은 실행되지 않고 뛰어 넘어감
- 반복문 while과 do while 반복 내부
 - continue를 만나면 조건검사로 이동하여 실행
- 반복문 for 문
 - continue 문을 만나면 증감 부분으로 이동하여 다음 반복 실행



Source Code #07: continue.c

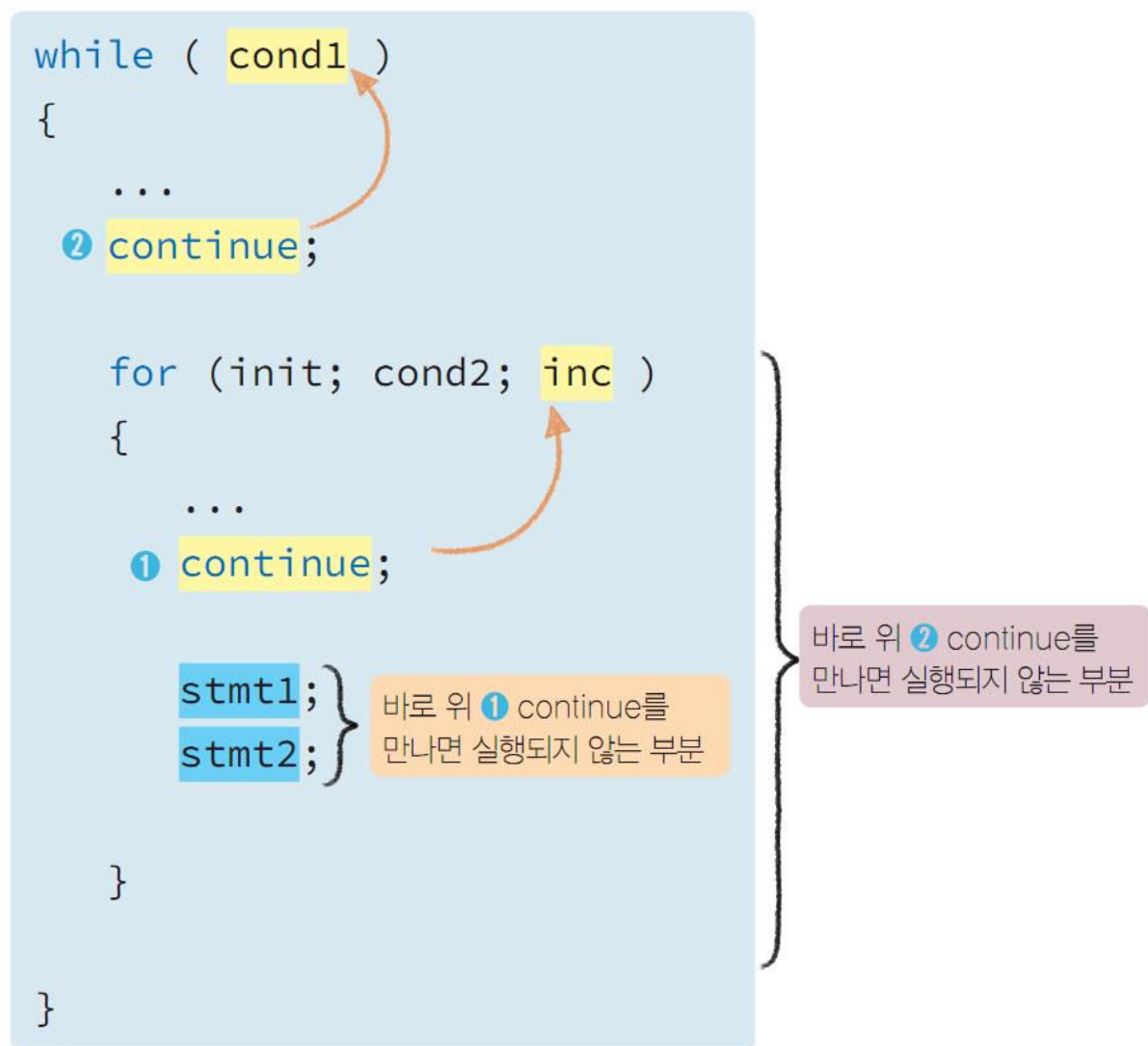
```
1 // file: continue.c
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     const int MAX = 15;
8
9     printf("1에서 %d까지 정수 중에서 3으로 나누어 떨어지지 않는 수\n", MAX);
10    for (int i = 1; i <= MAX; i++)
11    {
12        if (i % 3 == 0) // !(i % 3)
13            continue;
14        printf("%3d", i);
15    }
16    puts("");
17
18    return 0;
19 }
```

- 1에서 15까지 정수 중에서 3으로 나누어 떨어지지 않는 수를 출력
- 반복내부에서 반복을 종료하려면 break문장을 사용
- 만일 반복문이 중첩되어 있다면 break를 포함하는 가장 근접한 내부반복을 종료

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

중첩된 반복의 주의사항

- 중첩된 반복에서의 continue는 자신이 속한 가장 근접한 반복에서 다음 반복을 실행



Source Code #08: break.c

```
1 // file: break.c
2 #define _CRT_SECURE_NO_WARNINGS //scanf() 오류를 방지하기 위한 상수 정의
3
4 #include <stdio.h>
5
6 int main(void)
7 {
8     int input;
9
10    do {
11        printf("\t [1] 한식\n");
12        printf("\t [2] 양식\n");
13        printf("\t [3] 분식\n");
14        printf("\t [4] 기타\n");
15        printf("메뉴 번호 선택 후 [Enter] : ");
16        scanf("%d", &input);
17        printf("선택 메뉴 %d\n", input);
18
19        if (input <= 4 && input >= 1)
20            break;
21    } while (1);
22
23    return 0;
24 }
```

- 간단한 음식 메뉴 구성으로 사용자가 메뉴를 선택하면 프로그램을 종료
- 적당한 메뉴를 선택하지 못하면 선택할 때까지 반복을 실행

```

[1] 한식
[2] 양식
[3] 분식
[4] 기타
메뉴 번호 선택 후 [Enter] : 5
선택 메뉴 5
[1] 한식
[2] 양식
[3] 분식
[4] 기타
메뉴 번호 선택 후 [Enter] : 3
선택 메뉴 3
```

무한반복

- 반복문에서 무한히 반복이 계속되는 것
- while과 do while은 반복조건이 아예 없으면 오류가 발생

무한반복

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

무한반복

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

무한반복

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

무한반복

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

오류

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

오류

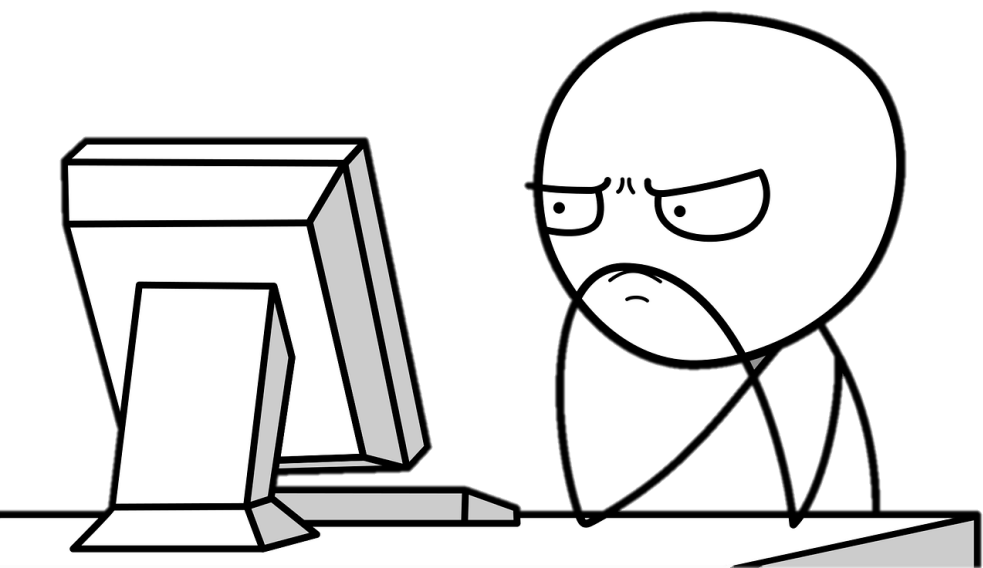
```
do  
{  
  
    ...  
  
} while ( )
```

Lab #03: 1부터 15까지 5의 배수가 아닌 정수 출력

- 분기문 continue 문을 사용하여 1부터 15까지의 정수 중에서 5의 배수가 아닌 수를 출력
- 정수는 모두 한 줄에 계속 출력
- 결과
 - 1에서 15까지 정수 중에서 5로 나누어 떨어지지 않는 수
 - 1 2 3 4 6 7 8 9 11 12 13 14

```
1 // file: continuelab.c
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     const int MAX = 15;
8
9     printf("1에서 %d까지 정수 중에서 5로 나누어 떨어지지 않는 수\n", MAX);
10    for (int i = 1; i <= MAX; i++)
11    {
12        if (i % 5 == 0)
13            continue;
14        printf("%3d", i);
15    }
16    puts("");
17
18    return 0;
19 }
```

4. 중첩된 반복문



Source Code #09: nestedloop.c

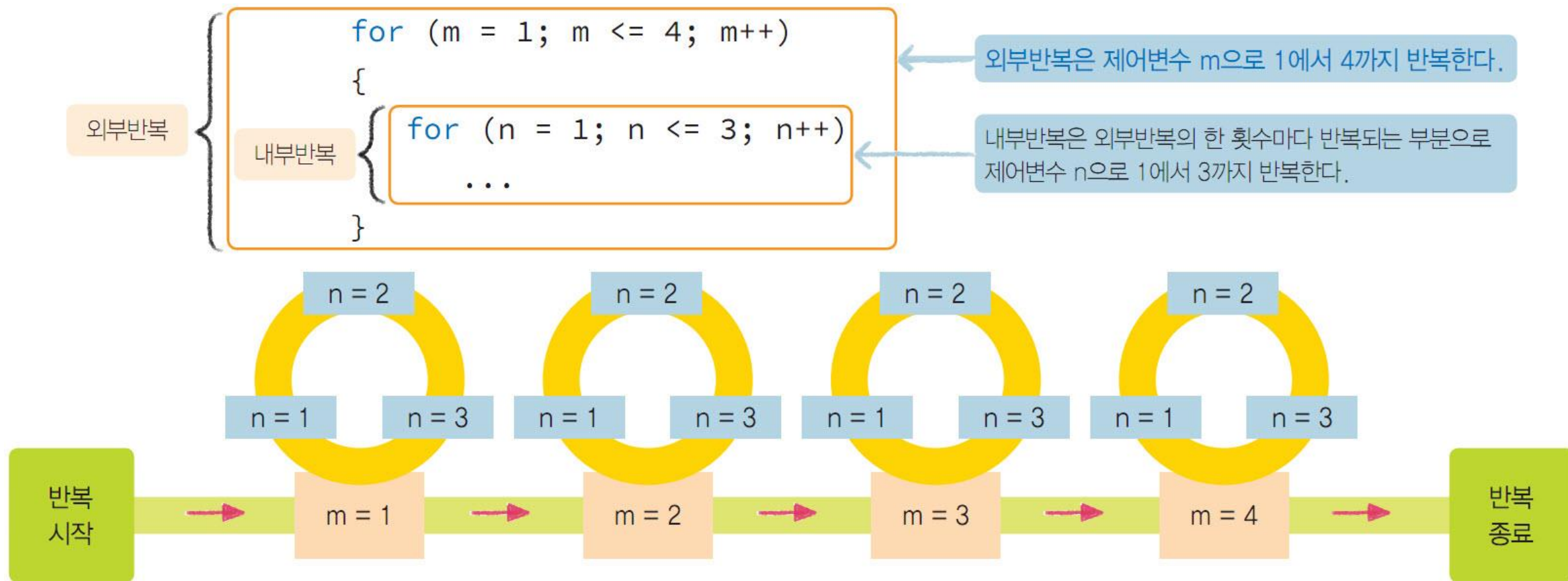
```
1  // file: nestedloop.c
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int m, n;
8
9      for (m = 1; m <= 5; m++)
10     {
11         printf("m = %-2d\n", m);
12         for (n = 1; n <= 7; n++)
13             printf("n = %-3d", n);
14         puts("");
15     }
16
17     return 0;
18 }
```

- 외부반복에서 1에서 5까지
- 내부반복에서 1에서 7까지 반복하면서 각각의 변수값을 출력

```
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
```

중첩된 for

- 반복문 내부에 반복문이 또 있는 구문
- 외부 for 문의 제어변수는 m이며, 내부 for문의 제어변수는 n



Source Code #10: loops.c

```
1 // file: loops.c
2 #define _CRT_SECURE_NO_WARNINGS
3
4 #include <stdio.h>
5
6 int main(void)
7 {
8     int input, sum, i, j;
9
10    do
11    {
12        printf("양의 정수 또는 0(종료)을 입력: ");
13        scanf("%d", &input);
14
15        for (i = 1; i <= input; i++)
16        {
17            for (j = 1, sum = 0; j <= i; j++)
18            {
19                printf("%d", j);
20                j == i ? printf(" = ") : printf(" + ");
21                sum += j;
22            }
23            printf("%d\n", sum);
24        }
25    } while (input > 0);
26
27    puts("종료합니다.");
28
29    return 0;
30 }
```

- 양의 정수를 입력 받아 합을 출력하고
- 0 또는 음수를 입력할 때까지 계속 수행하는 프로그램
- 합의 출력은 그 과정이 모두 보이도록 함
- 센티널 값인 0 또는 음수를 입력하면 프로그램이 종료

```
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: 0
종료합니다.
```

삼중 중첩반복

- 외부반복은 1에서 입력값 input까지 제어변수 i로 반복
- 내부반복은 제어변수 j를 사용하여 1에서 i까지 반복
- 변수sum은 1에서 i까지 합을 저장
- 출력되는 콘솔 한 행에 1에서 j까지 합을 $1+2+\dots+j = \text{sum}$ 으로 출력
- 조건연산자 $j == i ? \text{printf}(" = ") : \text{printf}(" + ")$ 를 이용
- 즉 출력값이 중간이며 +를 출력하고 마지막이면 =를 출력하고 내부 for 반복을 종료한 후, 바로 그 때까지의 합인 sum을 출력

삼중 중첩반복

```
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);  
}
```

양의 정수 또는 0(종료)을 입력: 5

1 = 1

1 + 2 = 3

1 + 2 + 3 = 6

1 + 2 + 3 + 4 = 10

1 + 2 + 3 + 4 + 5 = 15

Lab #04: 구구단 출력

- 반복문 for 문을 사용하여
2단부터 9단까지의 구구단을 출력

```
1 // file: mtable.c
2
3 #include <stdio.h>
4 #define MAX 9
5
6 int main(void)
7 {
8     printf("=== 구구단 출력 ===\n");
9     for (int i = 2; i <= MAX; i++)
10     {
11         printf("%d단 출력\n", i);
12         for (int j = 1; j <= MAX; j++)
13             ;
14         printf("\n");
15     }
16
17     return 0;
18 }
```

