



01

반복/종료 조건

# 1. 반복/종료 조건

---

- 반복이란 코드가 중복된다는 의미
- C 언어가 제공하는 while 문, for 문, 그리고 do-while 문
- 본질적으로 세 가지 반복문은 동일하지만, 용도가 조금씩 다름
- 반복문을 코딩할 때 반복 또는 종료 조건에 대해서 정확하게 파악해야 함
- 반복이나 종료 조건은 언제 반복해야 할지와 반복을 종료할지 결정하는 것

02

while 문

## 2. while 문

- 조건이 만족되지 않을 때까지 코드를 반복해서 실행

```
while (조건식) {  
    S1;  
}  
S2;
```

- 코드 블록 없이 다음과 같은 형태로도 사용 가능

```
while (조건식) S1;  
while (조건식)  
    S1;
```

- 반복문에서 S1이 빈 명령문인 경우

```
while (func())  
    ;
```

## 2. while 문

### ■ 정수의 약수를 출력하는 프로그램

- 알고리즘 - 1~4까지의 정수를 4로 나누어서 나뉘진 것들만 출력
  - 4를 1로 나누어서 나머지가 0이면 약수로 분류
  - 4를 2로 나누어서 나머지가 0이면 약수로 분류
  - 4를 3으로 나누어서 나머지가 0이면 약수로 분류
  - 4를 4로 나누어서 나머지가 0이면 약수로 분류
  
- 약수를 찾는 과정 - 정수는 사용자가 입력한  $n$ 이라고 가정
  - $n$ 을 1로 나누어서 나머지가 0이면 약수로 분류
  - $n$ 을 2로 나누어서 나머지가 0이면 약수로 분류
  - ...
  - $n$ 을  $n$ 으로 나누어서 나머지가 0이면 약수로 분류

## 2. while 문

### ■ 정수의 약수를 출력하는 프로그램

- 변수 사용
- n은 사용자가 입력한 값을 저장하고 있고 1 이상이라고 가정

```
int divisor = 1;
if (divisor <= n) {
    if (n % divisor == 0) {
        printf("%d ", divisor);
    }
    divisor++;
}
...
if (divisor <= n) {
    if (n % divisor == 0) {
        printf("%d ", divisor);
    }
    divisor++;
}
```

The diagram illustrates the loop structure. A green box highlights the code block inside the `if (divisor <= n)` condition. A bracket to the right of this box is labeled "반복되는 코드" (Repeated code). Another bracket to the right of the entire loop structure is labeled "divisor <= n이면 반복" (Repeat if divisor <= n).

그림 5-1 반복되는 코드 영역과 반복 조건

## 2. while 문

### ■ 정수의 약수를 출력하는 프로그램

- if를 while로 변경

코드 5-1

```
1  int divisor = 1;  ← divisor를 1로 초기화해서 1부터 나눠지는지 확인
2  while (divisor <= n) { ← divisor 값이 n보다 작거나 같을 때 반복
3      if (n % divisor == 0) { ← n을 divisor로 나눠보고 나머지가 0인지 확인
4          printf("%d ", divisor);
5      } ← divisor가 나눠지면 약수이므로 화면에 출력
6      divisor++;
7  } ← divisor를 1 증가시켜서 다음 값 확인
```

※ 정수의 약수를 출력하는 프로그램 완성

→ [코드 5-2](#)  
→ [실행 결과](#)



## 2. while 문

### ■ 알파벳을 입력받을 때까지 지속적으로 반복하는 프로그램

- 올바른 입력까지 반복하는 알고리즘

1. 문자 입력받기
2. 입력 내용이 영문 알파벳이 아니면, 오류 내용 출력 후 다시 입력 받기를 반복
3. 입력한 알파벳 글자를 출력

- 2번 항목 자세히 풀기

```
(ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z')
```

- 반대로 뒤집거나 부정 붙이기

```
(ch < 'A' || ch > 'Z') && (ch < 'a' || ch > 'z')
```

```
!((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z'))
```

## 2. while 문

### ■ 알파벳을 입력받을 때까지 지속적으로 반복하는 프로그램

#### 코드 5-4

```
1 char ch;
2 printf("영문 알파벳 문자 한 개를 입력하세요: ");
3 scanf("%c", &ch);
4 while ((ch < 'A' || ch > 'Z') && (ch < 'a' || ch > 'z')) {
5     printf("알파벳 아닌 문자가 입력되었습니다. 다시 입력하세요: ");
6     scanf("%c", &ch);
7 }
```

## 2. while 문

### ■ 알파벳을 입력받을 때까지 지속적으로 반복하는 프로그램

#### 코드 5-5

```
1 char ch;
2 char newlinchar;
3 printf("영문 알파벳 문자 한 개를 입력하세요: ");
4 scanf("%c", &ch);
5 scanf("%c", &newlinchar); // ch에 다시 저장하면 먼저 읽은 글자가 사라짐
6 while ((ch < 'A' || ch > 'Z') && (ch < 'a' || ch > 'z')) {
7     printf("알파벳 아닌 문자가 입력되었습니다. 다시 입력하세요: ");
8     scanf("%c", &ch);
9     scanf("%c", &newlinchar);
10 }
```

※ 알파벳을 입력받을 때까지 지속적으로 반복하는 프로그램 완성

➔ [코드 5-6](#)  
➔ [실행 결과](#)

03

do-while 문

### 3. do-while 문

- do 명령문을 먼저 실행한 후에 조건식을 확인
- 반복문의 명령문을 최소한 한 번 실행
- do-while 문은 do가 먼저오고 중괄호가 무조건 있어야 함

```
do {  
    S1;  
} while(조건식);  
S2;
```

### 3. do-while 문

#### ■ 정수의 약수를 출력하는 프로그램

- 코드 5-1의 while 문 코드를 do-while 문으로 구현

코드 5-7

```
1  int divisor = 1;
2  do {
3      if (n % divisor == 0) {
4          printf("%d ", divisor);
5      }
6      divisor++;
7  } while (divisor <= n);
```

※ 정수의 약수를 출력하는 프로그램 완성  
(5-2 코드를 do-while 문으로 변경)

➔ [코드 5-8](#)  
➔ [실행 결과](#)

### 3. do-while 문

#### ■ 알파벳을 입력받을 때까지 반복하는 프로그램

- 코드 5-5의 while 문을 do-while 문으로 변경

```
1      printf("영문 알파벳 문자 한 개를 입력하세요: ");
2      scanf("%c", &ch);
3      scanf("%c", &newlinchar);
4      while ((ch < 'A' || ch > 'Z') && (ch < 'a' || ch > 'z')) {
5          printf("알파벳 아닌 문자가 입력되었습니다. 다시 입력하세요: ");
6          scanf("%c", &ch);
7          scanf("%c", &newlinchar);
8      }
```

내용이 다름

동일한 코드

그림 5-2 do-while로 변경하기 전에 중복되는 코드 확인

### 3. do-while 문

- 알파벳을 입력받을 때까지 입력받는 프로그램
  - 코드 5-5의 while 문을 do-while 문으로 변경
    - 1. 출력 문구가 달라짐("알파벳 아닌 문자가 ..." 생략됨)

#### 코드 5-9

```
1  do {  
2      printf("영문 알파벳 문자 한 개를 입력하세요: ");  
3      scanf("%c", &ch);  
4      scanf("%c", &newlinchar);  
5  } while ((ch < 'A' || ch > 'Z') && (ch < 'a' || ch > 'z'));
```



### 3. do-while 문

- 알파벳을 입력받을 때까지 입력받는 프로그램
  - 코드 5-5의 while 문을 do-while 문으로 변경
    - 1. 출력 문구가 달라짐("알파벳 아닌 문자가 ..." 생략됨)

코드 5-10 InputAlpha2.c

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3
4  int main(void)
5  {
6      char ch;
7      char newlinchar;
8
9      코드 5-9 삽입
10
11     printf("ch = %c\n", ch);
12     return 0;
13 }
```

#### 〈실행 결과〉

영문 알파벳 문자 한 개를 입력하세요: 3  
영문 알파벳 문자 한 개를 입력하세요: a  
ch = a

### 3. do-while 문

#### ■ 알파벳을 입력받을 때까지 입력받는 프로그램

- 코드 5-5의 while 문을 do-while 문으로 변경
  - 2. 기존 출력 문구를 유지

코드 5-11

```
1  int first = 1;
2  do {
3      if (first) {
4          printf("영문 알파벳 문자 한 개를 입력하세요: ");
5          first = 0;
6      }
7      else {
8          printf("알파벳 아닌 문자가 입력되었습니다. 다시 입력하세요: ");
9      }
10     scanf("%c", &ch);
11     scanf("%c", &newlinchar);
12 } while ((ch < 'A' || ch > 'Z') && (ch < 'a' || ch > 'z'));
```

반복문이 처음 실행될 때 first는 1로 시작

first가 1일 때, while 문 전에 있던 문자열 출력

반복문을 한 번 실행 후 first를 0으로 변경해 while 문 안에 있던 다른 문자열 출력

while 문 전에 있던 내용이 while 문 안에 있던 내용과 동일 do-while에서는 한 번만 넣으면 됨

do-while의 조건식은 기존 while의 조건식과 같아도 됨 알파벳이 아니라면 과정 반복

※ 사용자로부터 알파벳을 입력받을 때까지  
지속적으로 입력받는 프로그램 완성

➔ [코드 5-12](#)

➔ [실행 결과](#)

04

for 반복문

## 4. for 반복문

- 정해진 횟수만큼 반복하거나 배열처럼 정해진 개수의 자료를 한 개씩 처리할 때 사용하는 반복문
  - while 문보다 짧은 코드 작성 가능

```
for ([초기화_표현식]; [조건식]; [증감_연산_표현식]) {  
    S1;  
}  
S2;
```

- 초기화\_표현식은 반복문에서 횟수 또는 범위를 지정할 변수를 선언하고 초기화시키는 데 사용
- 주로 횟수나 번호를 나타내는 변수
- 조건식은 반복을 지속할지 종료할지 결정
- 증감\_연산\_표현식은 초기화\_표현식에서 선언된 변수값을 변경
- 횟수나 번호를 증감시키는 목적으로 사용

## 4. for 반복문

### ■ for 반복문 실행 순서

```
for ([초기화_표현식]①; [조건식]②; [증감_연산_표현식]④) {  
    S1; ③  
}  
S2; ⑤
```

반복하는 경우: ①②③④②③④...⑤  
반복하지 않고 바로 종료: ①②⑤

그림 5-3 for 반복문 실행 순서

- while 문과 비슷하게 S1이 한 개의 명령문만 있는 경우 중괄호는 생략 가능

```
for (초기화_표현식; 조건식; 증감_연산_표현식) S1; S2; // 두 개의 for 반복문은 동일한 코드  
for (초기화_표현식; 조건식; 증감_연산_표현식)  
    S1;  
S2;
```

## 4. for 반복문

- 1부터 10까지 1씩 증가시키면서 화면에 출력하는 간단한 for 문

코드 5-13

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

- 반복문이 시작되어 반복하는 순서
  - i는 1부터 시작하고, i가 10 이하이므로 화면에 출력
  - i++에서 1만큼 증가되어 i는 2가 됨
  - 다시 조건식을 확인하고 화면에 출력한 다음 1 증가
  - 실행되다 보면 i가 11이 되고, 조건식의 결과값은 거짓이 되어 for 문 종료

## 4. for 반복문

- 변수를 1부터 초기화 하지 않아도 되는 경우

코드 5-14

```
1  for (int i = 0; i < 10; i++) {  
2      printf("%d ", i + 1);  
3  }
```

- 초기화\_표현식에 있던 코드를 for 반복문 밖으로 내보내기

코드 5-15

```
1  int i = 0;  
2  for ( ; i < 10; i++) {  
3      printf("%d ", i + 1);  
4  }
```



## 4. for 반복문

- `i++`를 생략하고 for 문의 코드 블록 안으로 이동

코드 5-16

```
1  for (int i = 0; i < 10; ) {  
2      printf("%d ", i + 1);  
3      i++;  
4  }
```

초기화 코드 실행된 후 조건식이 만족되는지 확인

조건이 만족되면 실행  
3줄까지 실행되고 다시 1줄의 조건식 확인한 다음 반복할지 결정

- 초기화 부분을 반복문 바깥으로 이동(while문과 구조가 같음)

코드 5-17

```
1  int i = 0;  
2  for ( ; i < 10; ) {  
3      printf("%d ", i + 1);  
4      i++;  
5  }
```

## 4. for 반복문

### ■ 조건식이 생략되는 경우(무한 반복)

코드 5-18

```
1  for (int i = 0; ; i++) {  
2      if (i >= 10) { break; } // i가 10보다 크거나 같으면 for 반복문을 빠져나감  
3      print("%d ", i + 1);  
4  }
```

조건식을 생략함

break를 사용하면 1줄의 조건식(지금은 생략됨)을 무시하고, 반복문을 종료. i가 10 이상이 되면 break 문으로 종료. break는 다음 절에서 학습

코드 5-19

```
1  int i = 0;  
2  while (1) { // 무한 반복  
3      if (i >= 10) { break; } // i가 10보다 크거나 같으면 while 반복문을 빠져나감  
4      print("%d ", i + 1);  
5      i++;  
6  }
```

## 4. for 반복문

### ■ for 반복문에서 여러 개 변수 사용 - 콤마 연산자

- for 반복문에서 변수 2개 이상 초기화시키고, 증감 연산을 해서 사용하는 경우

코드 5-20 TwoDigitsSumTo10.c

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int second = 9; // 두 번째 숫자 9부터 1로 변경할 예정
7      for (int first = 1; first < 10; first++) {
8          printf("%d%d\t", first, second);
9          second--;
10     }
11     return 0;
12 }
```

두 번째 자리 숫자를 9로 초기화했다가 한 번씩 반복할 때마다 1을 차감

첫 번째 자리 숫자는 for 반복문에서 1부터 9까지 변경

첫 번째 자리 숫자와 두 번째 자리 숫자를 붙여 출력

숫자 한 개를 출력했으므로 second에서 1을 차감

<실행 결과>

19    28    37    46    55    64    73    82    91

## 4. for 반복문

- for 반복문에서 여러 개 변수 사용 - 콤마 연산자
  - 자료형 변수를 여러 개 선언하고 사용하는 경우

### 코드 5-21

```
1  for (int first = 1, second = 9; first < 10; first++, second--) {  
2      printf("%d%d\t", first, second);  
3  }
```

## 4. for 반복문

### ■ for, while 반복문

```
int i = 0;
while (i < 10) {
    printf("%d ", i + 1);
    i++;
}
```

```
int i = 0;
for ( ; i < 10; ) {
    printf("%d ", i + 1);
    i++;
}
```

```
for (int i = 0; i < 10; i++) {
    printf("%d ", i + 1);
}
```

```
for (int i = 0; i < 10; ) {
    printf("%d ", i + 1);
    i++;
}
```

그림 5-4 while 문과 for 문의 상호 변환

## 4. for 반복문

### ■ 정수의 약수를 출력하는 프로그램

#### ■ 알고리즘

- divisor 변수를 1로 초기화
- divisor  $\leq$  n일 때 n의 약수를 화면에 출력
- divisor를 1만큼 증가

#### ■ 코드 5-1을 for 문으로 작성

##### 코드 5-22

```
1  for (int divisor = 1; divisor <= n; divisor++) {  
2      if (n % divisor == 0) {  
3          printf("%d ", divisor);  
4      }  
5  }
```

※ 정수의 약수를 출력하는 프로그램 완성

➔ [코드 5-23](#)

➔ [실행 결과](#)

## 4. for 반복문

- 알파벳을 입력받을 때까지 지속적으로 반복하는 프로그램
  - 코드 5-5를 for 문으로 수정

코드 5-24 InputAlphaFor.c

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3
4  int main(void)
5  {
6      char ch;
7      char newlinchar;
8      printf("영문 알파벳 문자 한 개를 입력하세요: ");
9      scanf("%c", &ch);
10     scanf("%c", &newlinchar); // ch에 다시 저장하면 먼저 읽은 글자가 사라짐
11     for ( ; (ch < 'A' || ch > 'Z') && (ch < 'a' || ch > 'z') ; ) {
12         printf("알파벳 아닌 문자가 입력되었습니다. 다시 입력하세요: ");
13         scanf("%c", &ch);
14         scanf("%c", &newlinchar);
15     }
16     printf("ch = %c\n", ch);
17     return 0;
18 }
```



## 4. for 반복문

### ■ 반복문 특징

- while 문
  - 조건식의 결과에 따라 반복문의 실행 여부를 결정하는 경우에 적합
  - 그 중에서도 조건에 따라 코드를 실행하지 않을 수도 있는 경우에 사용
- do-while 문
  - while 문과는 달리 최소한 한 번은 실행시켜야 하는 코드가 있을 때 적합한 반복문
- for 문
  - 주로 정해진 횟수 또는 정해진 개수의 요소들을 각각 처리할 때 자주 사용
  - while이나 do-while보다 for 문이 반복 관련 코드를 보다 일목요연하게 보여 줄 수 있음

05

**break와 continue**

## 5. break와 continue

- 프로그램은 보통 순차적으로 실행
- 반복문도 내부 영역의 코드는 순차적으로 실행

<pre>while (조건식) {     S1;     S2;     ...     Sm; }</pre>	<pre>do {     S1;     S2;     ...     Sm; } while (조건식);</pre>	<pre>for (expr1; expr2; expr3) {     S1;     S2;     ...     Sm; }</pre>
--------------------------------------------------------------------------------	------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------

그림 5-5 반복문에서 명령문들이 순차적으로 실행됨

## 5. break와 continue

### ■ break 문

- 반복문의 실행 흐름을 멈추고, 반복문 다음 문장을 실행시키도록 하는 명령문

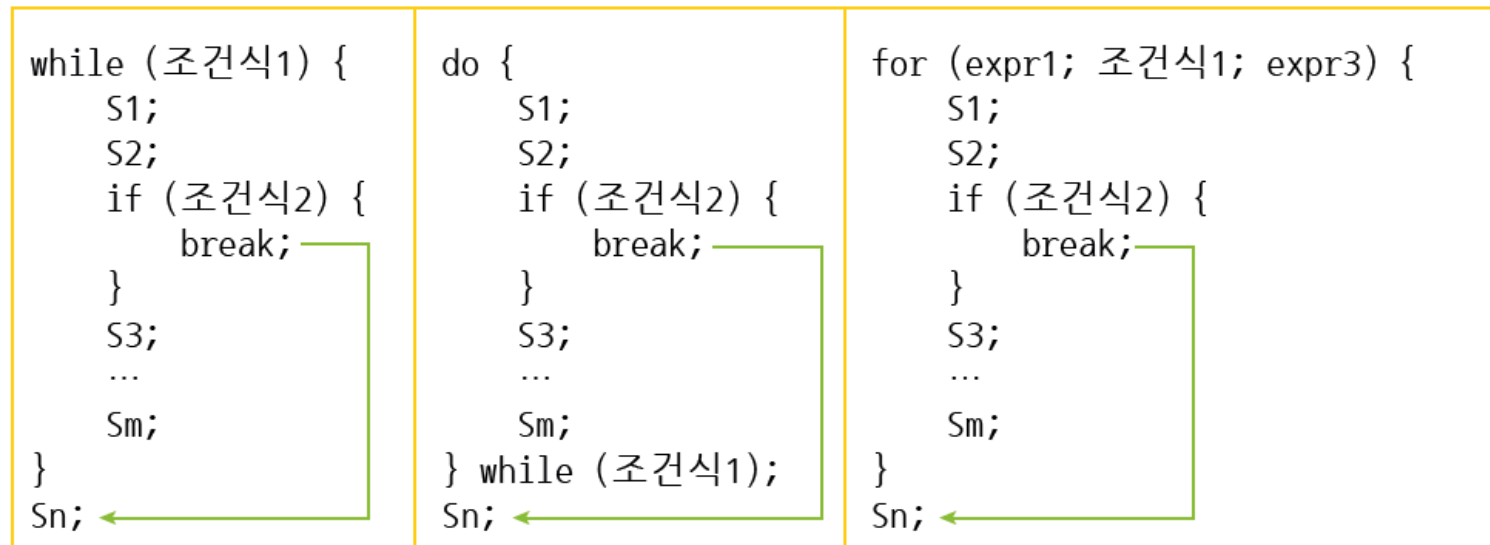
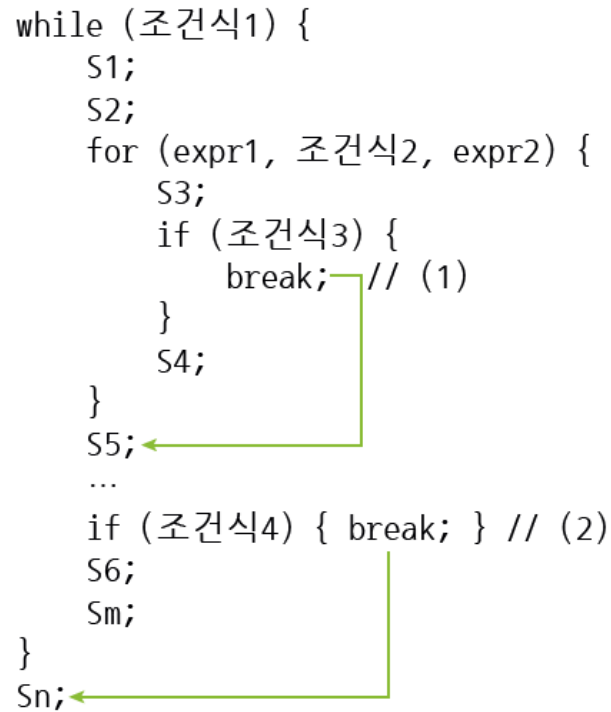


그림 5-6 break 문 동작 방법

## 5. break와 continue

### ■ break 문

- 중첩된 반복문에서 break 문 사용



```
while (조건식1) {  
    S1;  
    S2;  
    for (expr1, 조건식2, expr2) {  
        S3;  
        if (조건식3) {  
            break; // (1)  
        }  
        S4;  
    }  
    S5;  
    ...  
    if (조건식4) { break; } // (2)  
    S6;  
    Sm;  
}  
Sn;
```

그림 5-7 중첩 반복문에서 break 문이 실행되는 방법

## 5. break와 continue

### ■ 약수를 최대 네 개까지 출력하는 프로그램

- 약수 개수가 4개 미만인 경우는 전체 출력
- 1~n까지의 숫자들을 한 번씩 나눠야 한다는 점에서 for 반복문을 사용
- 출력한 약수의 개수를 세고, 4개가 될 때 화면에 출력한 뒤 반복문을 종료
- 약수의 개수를 세기 위해 numOfDivisors 변수를 만들고 0으로 초기화

```
int numOfDivisors = 0;
```

코드 5-25

```
1  for (int divisor = 1; divisor <= n; divisor++) {  
2      if (n % divisor == 0) {  
3          printf("%d ", divisor);  
4          numOfDivisors++;  
5          if (numOfDivisors == 4) { break; } // 약수 개수가 4개이면 종료  
6      }  
7  }
```

divisor를 1~n까지 변경하면서 나눠지는지 확인하기 위해 반복

n을 divisor로 나눌 때 나눠지는지 확인

약수이므로 화면에 출력

약수 개수를 1만큼 증가

약수 개수가 4개이면 반복문 종료하고, 아니면 다시 반복

※ 약수를 최대 네 개까지 출력하는 프로그램 완성

➔ [코드 5-26](#)

➔ [실행 결과](#)

## 5. break와 continue

### ■ continue 문

- 반복문의 순차적인 실행을 중단
- 다시 반복/종료 조건을 확인하면서 반복할지 결정하는 명령문

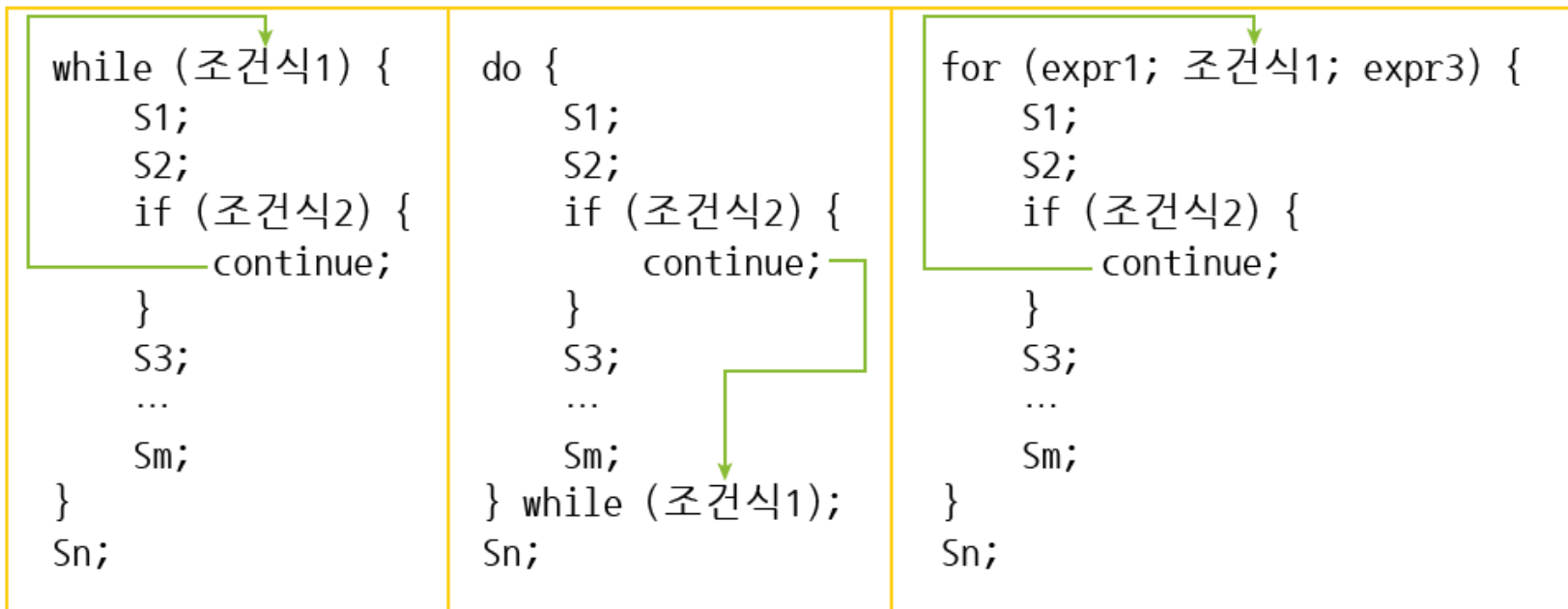


그림 5-8 continue 문 동작 방법



## 5. break와 continue

### ■ continue 문

#### ■ 동작 방법

```
for (초기화_표현식; 조건식; 증감_연산_표현식
    S1;
    if (조건식2) { continue; }
    S3;
```

그림 5-9 for 문에서 continue 문 동작 방법

## 5. break와 continue

### ■ continue 문

#### ■ 동작 방법

```
while (조건식1) {  
    S1;  
    S2;  
    for (expr1, 조건식2, expr2) {  
        S3;  
        if (조건식3) {  
            continue; // (1)  
        }  
        S4;  
    }  
    S5;  
    ...  
    if (조건식4) { continue; } // (2)  
    S6;  
    Sm;  
}  
Sn;
```

그림 5-10 중첩 반복문에서 continue 문 동작 방법

## 5. break와 continue

### ■ 약수 중에서 홀수만 출력하는 프로그램 구현

#### ■ 간단한 방법

코드 5-27

```
1  for (int divisor = 1; divisor <= n; divisor++) {  
2      if (n % divisor == 0) {  
3          if (divisor % 2 != 0) {  
4              printf("%d ", divisor);  
5          }  
6      }  
7  }
```

divisor를 2로 나눠 나머지가 0이 아니면, 홀수이므로 출력

#### ■ 논리 연산자로 묶어 표현

코드 5-28

```
1  for (int divisor = 1; divisor <= n; divisor++) {  
2      if (n % divisor == 0 && divisor % 2 != 0) {  
3          printf("%d ", divisor);  
4      }  
5  }
```

## 5. break와 continue

### ■ 약수 중에서 홀수만 출력하는 프로그램

#### ■ continue 문 사용

코드 5-29

```
1  for (int divisor = 1; divisor <= n; divisor++) {  
2      if (n % divisor == 0) {  
3          if (divisor % 2 == 0) { continue; }  
4          printf("%d ", divisor);  
5      }  
6  }
```

divisor가 약수고 짝수이면 continue 문을 이용해서 divisor++를 호출 후, divisor <= n을 다시 확인하도록 명령

divisor가 홀수이면 출력

※ 약수 중에서 홀수만 출력하는 프로그램 완성

➔ [코드 5-30](#)

➔ [실행 결과](#)