제07장

반복

Perfect C

C언어로 배우는 프로그래밍 기초

단원 목표

학습목표

- **반복문에 대하여 이해하고 구현할 수 있다.**
  - while문의 구조를 이해하고 필요한 반복의 구현이 가능
  - do while문의 구조를 이해하고 필요한 반복의 구현이 가능
  - for문의 구조를 이해하고 필요한 반복의 구현이 가능
  - 반복문 내부에서의 break와 continue의 가능
  - 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

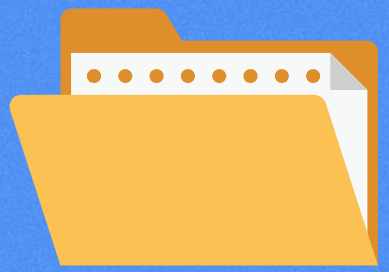
학습목차

7.1 반복 개요와 while 문

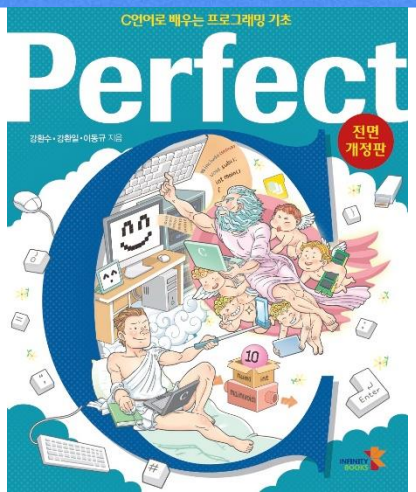
7.2 do while 문과 for 문

7.3 분기문

7.4 중첩된 반복문



01. 반복 개요와 while 문



반복 개요

- **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)

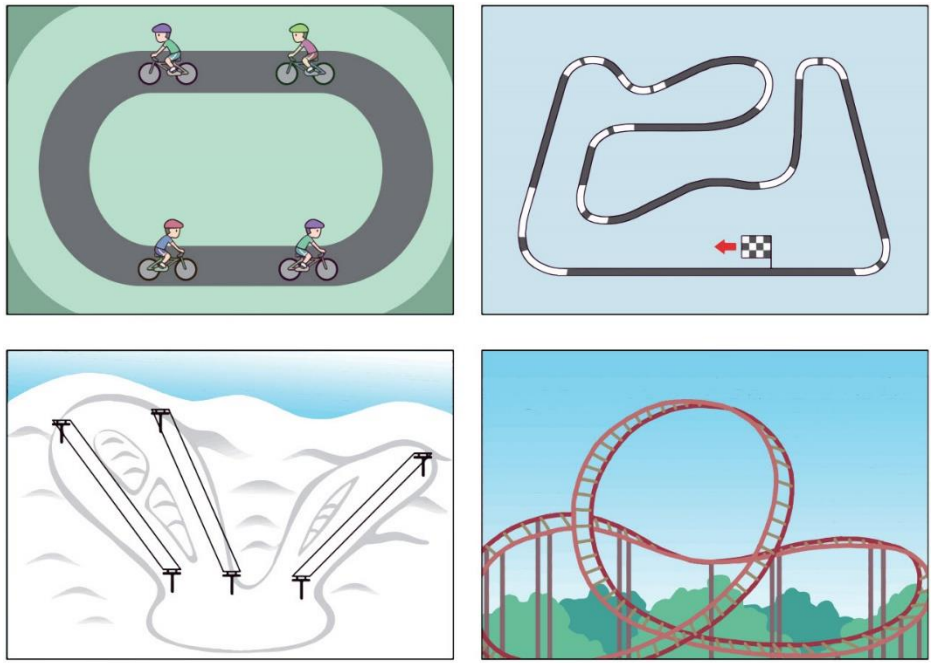


그림 7-1 스포츠와 일상에서의 반복과 순환

반복문 종류

- **while**
  - 반복문 while은 단순한 숫자의 반복이 아니라 반복할 때마다 조건을 따지는 반복문
  - 조건식이 반복몸체 앞에 위치
- **do while**
  - 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
- **for**
  - 반복문 for는 숫자로 반복하는 횟수를 제어하는 반복문
  - 명시적으로 반복 횟수를 결정할 때 주로 사용

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

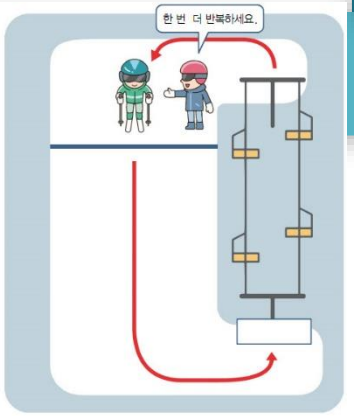


그림 7-2 while 반복

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

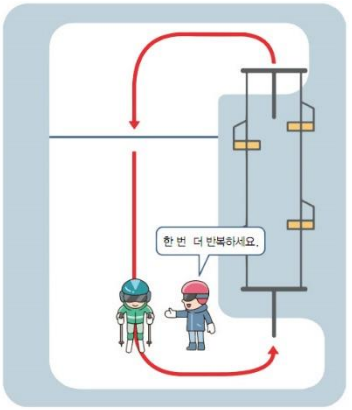


그림 7-3 do while 반복

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

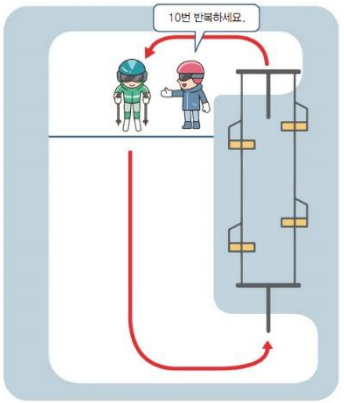


그림 7-4 for 반복

반복 구문의 필요성

- **동일하거나 또는 약간 다른 출력을 반복**
  - 함수 printf() 호출을 여러 번 반복
- **섭씨온도는 12.46도에서 10씩 2번 증가하면 각각의 화씨온도를 출력**
  - 이러한 소스도 뭔가 반복을 지원하는 간편한 구문이 있으면 훨씬 손쉽게 해결
- **정수의 각 비트 값 출력**
  - 비트 연산식 ((정수 >> n-1) & 1)의 결과값: 정수의 오른쪽 n 번째 비트값

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

```
1
2
3
4
5
```

그림 7-5 단순 반복의 출력

$$F(\text{화씨온도}) = \frac{9}{5}C(\text{섭씨온도}) + 32$$

```
double celcius = 12.46;

printf("%8.2lf %8.2lf\n", celcius, 9.0 / 5 * celcius + 32);
celcius += 10;
printf("%8.2lf %8.2lf\n", celcius, 9.0 / 5 * celcius + 32);
celcius += 10;
printf("%8.2lf %8.2lf\n", celcius, 9.0 / 5 * celcius + 32);
```

9/5로 쓰면 이 결과가 10이므로 부정확한 결과가 나온다.

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

그림 7-6 여러 섭씨온도를 화씨온도로 바꾸는 반복

```
printf("%d", 13 & 1);     //오른쪽 첫 비트값 출력
```

그림 7-7 정수 13의 오른쪽 첫 비트(LSB) 출력

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

그림 7-8 정수 13의 오른쪽 8개의 비트 모두 출력

while 문 구조와 제어흐름

- **문장 while (cond) stmt;**
  - cond를 평가하여 0이 아니면(참) 반복몸체인 stmt1를 실행
    - 다시 반복조건 cond를 평가하여 while 문 종료 시까지 반복
    - 반복은 cond가 0(거짓)이 될 때 계속
  - 반복이 실행되는 stmt를 반복몸체(repetition body)라 부름
    - 필요하면 블록으로 구성
  - while 문은 for나 do while 반복문보다 간단

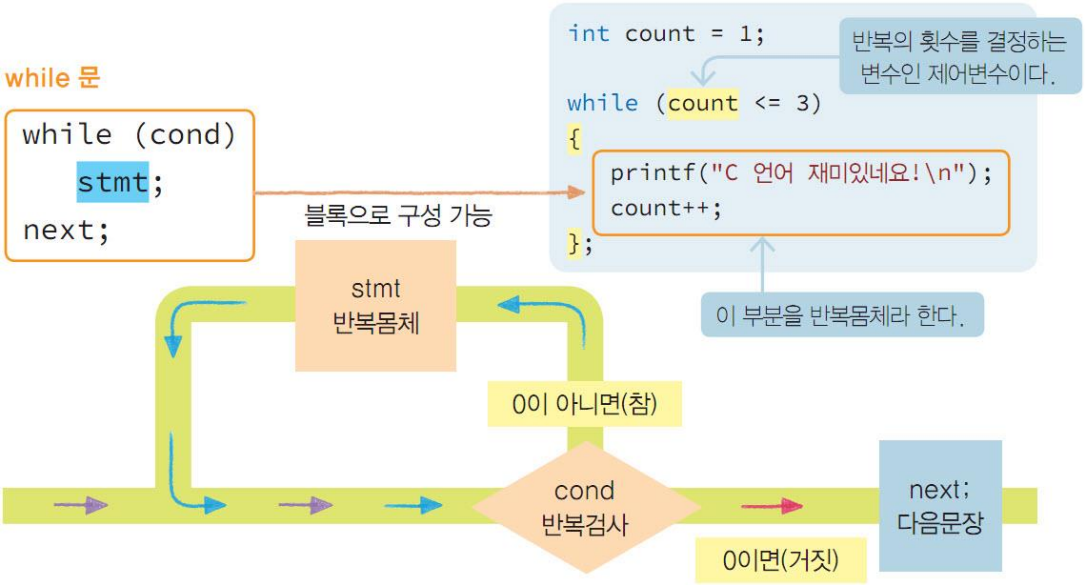


그림 7-9 반복 while문과 제어흐름

반복 while 문

예제 whilebasic.c

- 동일한 printf() 출력이 3번 반복되는 작업을 while 문으로 구현

제어변수

- 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++;
};
```

그림 7-10 동일 출력문 3개를 위한 while

실습예제 7-4 whilebasic.c

```
01  // file: whilebasic.c
02
03  #include <stdio.h>
04
05  int main(void)
06  {
07     int count = 1;
08
09     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  }
```

설명

09 조건식 (count <= 3)은 count 값이 3보다 작거나 같으면 참, 3보다 크면 즉 4 이상이면 거짓이 되어 while 문이 종료

10~13 while 반복몸체가 두 문장이므로 반드시 블록이 필요한데, 실수로 블록이 빠지면 논리오류가 발생하여 무한반복이 발생

11 반복몸체인 출력문

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

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

실행결과

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

제어변수 count => 4
```

반복 while 문

예제 whilenumber.c

- while문을 이용하여 1에서부터 5까지 1씩 증가되는 값을 출력하는 프로그램

반복과정

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

실습예제 7-5 whilenumber.c

```c
// file: whilenumber.c

#include <stdio.h>
#define MAX 5

int main(void)
{
   int n = 1;

   while (n <= MAX)
      printf("%d\n", n++);

   printf("\n제어변수 count => %d\n", n);

   return 0;
}
```

설명

04 매크로 상수 5를 정의, MAX는 반복횟수값으로 지정
08 제어변수 n 선언하면서 초기값으로 1 저장
10 전처리 수행 후, MAX가 5로 대체되어 while (n <= 5)가 됨
11 반복몸체인 출력문으로, n++의 연산값이 출력되는데, n++는 후위 증가연산자로 증가되기 이전값이 연산값이므로 1, 2, 3, 4, 5 출력
11 만일 ++n이라면, 전위 증가연산자로 증가된 이후 값이 연산값이므로 2, 3, 4, 5, 6 출력
14 while 문이 종료된 이후의 n 값은 5가 아니라 6이라는 사실에 주의, 그러므로 출력값은 6

실행결과

```
1
2
3
4
5

제어변수 n => 6
```

```c
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++);
```

```c
#define MAX 5
…

int n = 1;

while (n <= MAX)
   printf("%d\n", n++);
```

출력하고 싶은 최대 정수를 수정할 수 있다.

그림 7-13 1에서 5까지 출력하는 while

LAB 0부터 20까지 3의 배수 출력

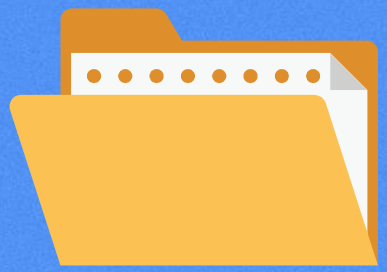
- **반복문 while 문을 사용**
  - 0부터 20까지의 3의 배수를 출력하는 프로그램
    - 정수는 모두 한 줄에 출력
- **결과**
  - 0 3 6 9 12 15 18

Lab 7-1 whilelab.c

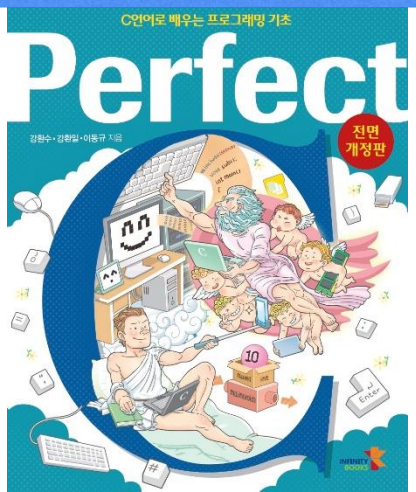
```
01  // file: whilelab.c
02
03  #include <stdio.h>
04  #define MAX 20
05
06  int main(void)
07  {
08     int n = 0;
09
10     while (__________________) {
11        printf("%4d", n);
12        __________________;
13     }
14     puts("");
15
16     return 0;
17  }
```

정답

```
10  while ((n <= MAX)) {
12  n += 3;
```



02. do while 문과 for 문



do while문 구조와 제어흐름

- **문장 do stmt; while (cond)**
  - 가장 먼저 stmt를 실행한 이후 반복조건인 cond를 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

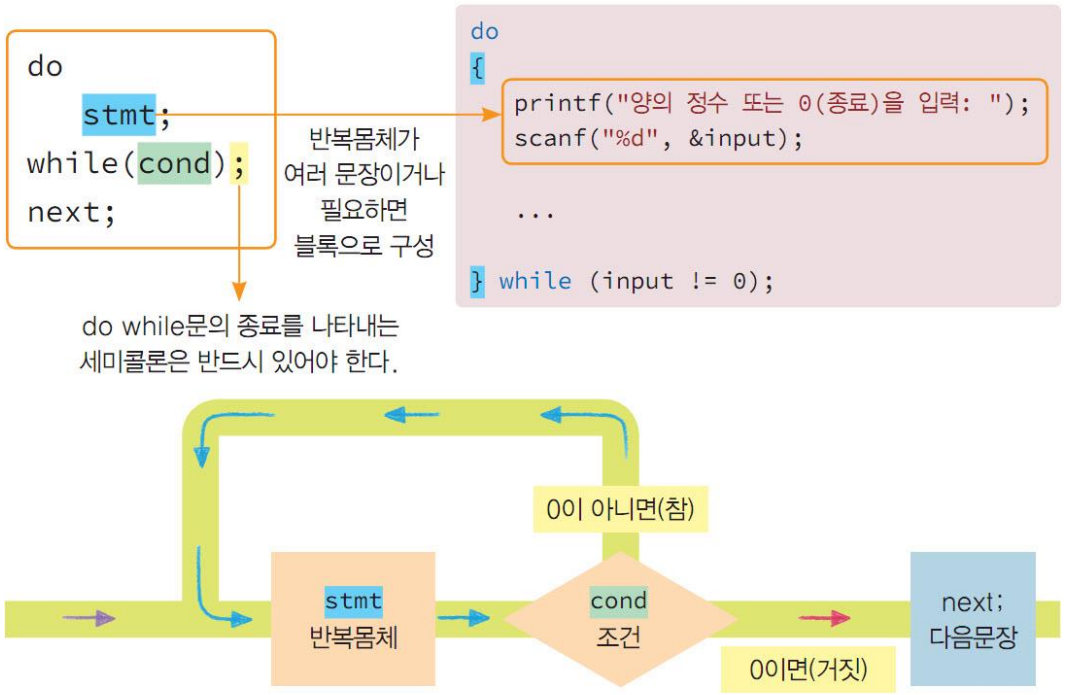


그림 7-14 반복 do while문의 제어흐름

센티널 값 검사에 유용

예제 dowhile.c

- 표준입력으로 받은 정수가 양수 또는 음수이면 계속 입력을 반복
- 입력한 수가 0이면 프로그램이 종료되는 프로그램

반복과정

- 입력 후에 반복 검사를 진행하는 처리 과정으로 do while 문으로 구현이 적합
- 센티널 값(sentinel value)
  - 반복의 종료를 알리는 특정한 자료값을
  - do while 반복문은 이러한 센티널 값 검사에 유용하게 사용

```
do
{
   printf("정수 또는 0(종료)을 입력: ");
   scanf("%d", &input);
} while (input != 0); //while (input);
```

그림 7-15 프롬프트와 표준입력으로 구성된 do while 몸체

실습예제 7-6 dowhile.c

```
01  // file: dowhile.c
02  #define _CRT_SECURE_NO_WARNINGS //scanf() 오류를 방지하기 위한 상수 정의
03
04  #include <stdio.h>
05
06  int main(void)
07  {
08     int input;
09
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  }
```

설명

```
08  표준입력값이 저장되는 변수 input
12  사용자에게 입력을 알리는 프롬프트인 메시지 "정수 또는 0(종료)을 입력:"를 출력
13  표준입력으로 입력한 값을 변수 input에 저장하는데, 반드시 &input으로 기술
11  조건식 (input != 0)을 사용하므로 0이 아니어야 12행으로 이동하여 반복하며, 0이면 반복을 종료
    하고 16행을 실행, 조건식 (input != 0)는 (input)과 같음
```

실행결과

```
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.
```

LAB 백단위 정수의 각 자릿수 출력

- **백단위의 양의 정수를 입력 받아 각각 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 / 100은 6
  - 정수의 나머지 연산자 %의 결과는 나머지 값으로, 673 % 100은 73
- **결과**
  - 양의 정수[100~999] 입력 : 853
  - 100단위 출력: 8
  - 10단위 출력: 5
  - 1단위 출력: 3

Lab 7-2 dowhilelab.c

```c
// file: forlab.c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

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

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

   return 0;
}
```

정답

```
result / devider
devider /= 10;
```

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'

- **for (i=1; i<=10; i++)**
  printf("%3d",i);
  - 1부터 10까지 출력
    - ❶ 초기화를 위한 init를 실행한다. 이 init는 단 한번만 수행된다.
    - ❷ 반복조건 검사 cond를 평가
      - 0이 아닌 결과값(참)이면 반복문의 몸체에 해당하는 문장 stmt를 실행
      - 그러나 결과값이 0(거짓)이면 for 문을 종료하고 다음 문장 next를 실행
    - ❸ 반복몸체인 stmt를 실행한 후 증감연산 inc를 실행한다.
    - ❹ 다시 반복조건인 cond를 검사하여 반복한다.

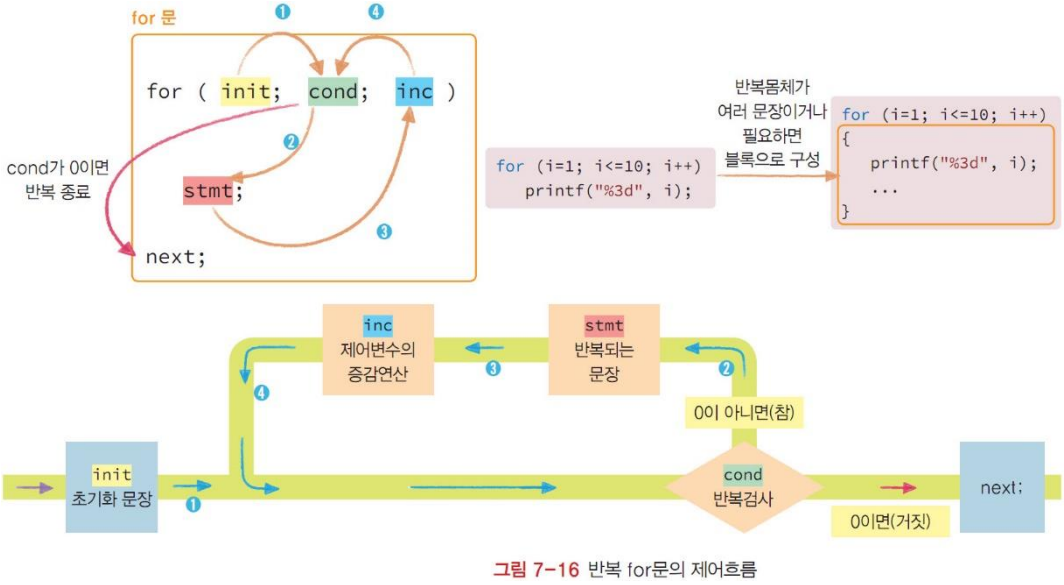


그림 7-16 반복 for문의 제어흐름

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

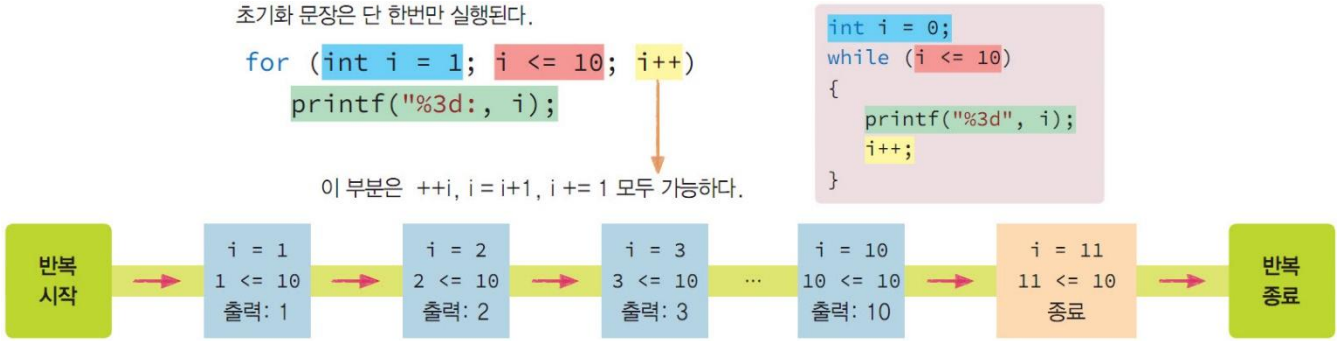


그림 7-17 1부터 10까지 출력하는 for문

표 7-2 for 문의 실행 과정

반복횟수	⓪ 초기화	❶ 조건식			❷ 반복몸체		❸ 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		

for 예제

예제 forbasic.c

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

실습예제 7-8 forbasic.c

```
01  // file: forbasic.c
02
03  #include <stdio.h>
04  #define MAX 5
05
06  int main(void)
07  {
08     int i;
09
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  }
```

설명

```
04    매크로 상수 5를 정의, MAX는 반복횟수값으로 지정
08    제어변수 i 선언
10    초기화에서 i = 1로 초기값 1저장, 이 문장은 시작할 때 한 번만 실행
10    조건식 i <= MAX은 전처리 수행 후, MAX가 5로 대체되어 i <= 5가 되며, i가 5보다 크면
      조건식이 거짓이 되어 반복을 종료
10    증감의 i++는 반복몸체인 12 행의 문장이 실행된 이후 실행
11~13 반복몸체가 하나이므로 블록을 위한 {}은 없어도 무방
12    반복몸체인 출력문으로, 문자열 "C 언어 재미있네요!"가 출력되고, i값이 출력되므로 첫 출력값은
      초기값인 1
10    12 행이 실행되고 이어 증가의 i++가 실행으로 이전 값을 1증가시키므로, ++i, i += 1,
      i = i + 1도 가능
10    다시 조건식 i <= 5가 실행되며, 만족하면 반복몸체를 다시 실행
15    for 문이 종료된 이후의 i값은 5가 아니라 6라는 사실에 주의, 그러므로 출력값은 6
```

실행결과

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

제어변수 i => 6
```

다양한 for 문

예제 dowhile.c

- 10도씩 증가하는 3 개의 섭씨온도 celcius를 화씨온도로 출력

반복과정

- 섭씨와 화씨온도를 출력할 반복횟수는 매크로 상수 MAX로 정의
- 섭씨 온도의 증가 값은 매크로 상수 INCREMENT로 정의
- 섭씨온도 celcius를 12.46으로 시작으로 3개의 화씨온도를 각각 출력하는 for문

10도씩 증가하는 3개의 섭씨온도를 화씨온도 변환하여 출력하는 소스

```
printf("%8.2lf %8.2lf\n", celcius, 9.0 / 5 * celcius + 32);
celcius += 10;
printf("%8.2lf %8.2lf\n", celcius, 9.0 / 5 * celcius + 32);
celcius += 10;
printf("%8.2lf %8.2lf\n", celcius, 9.0 / 5 * celcius + 32);
celcius += 10;
```

```
for (int i = 1; i <= 3; i++, celcius += 10)
{
   printf("%8.2lf %8.2lf\n", celcius, 9.0 / 5 * celcius + 32);
}
```

그림 7-18 여러 개의 섭씨온도를 화씨온도로 변환

실습예제 7-9 forcel2far3.c

```
01  // file: forcel2far3.c
02
03  #include <stdio.h>
04  #define MAX 3
05  #define INCREMENT 10
06
07  int main(void)
08  {
09     double celcius = 12.46;
10
11     printf("  섭씨(C)  화씨(F)\n");
12     printf("--------------------\n");
13
14     for (int i = 1; i <= MAX; i++, celcius += INCREMENT)
15     {
16        printf("%8.2lf %8.2lf\n", celcius, 9.0 / 5 * celcius + 32);
17     }
18
19
20     return 0;
21  }
```

설명

```
04   매크로 상수 3을 정의, MAX는 반복횟수값으로 지정
05   섭씨온도의 간격을 10도 증가시키면서 변환하기 위한 증분값 10도 매크로 상수로 정의
09   섭씨온도가 저장되는 변수 celcius 선언과 초기값 12.46 저장
11~12 출력 양식을 위한 제목 등의 헤드라인 출력
14   for 문 초기화는 int i = 1과 같이 변수선언과 초기화도 가능, 변수 i는 for 문 내부에서만
     사용 가능한 지역변수라고 함
14   조건식인 i <= MAX 은 전처리 수행 후, MAX가 3으로 대체되어 i <= 3 이 되며, 3보다 크면
     조건식이 거짓이 되어 반복을 종료
14   증감은 i++, celcius += INCREMENT와 같이 여러 문장을 콤마로 나열이 가능하며, 제어변수도
     1 증가시키고, 섭씨온도도 증분인 INCREMENT(10) 만큼 증가시킴
15~17 for 반복몸체가 한 문장이므로 블록은 없어도 상관 없음
16   반복몸체인 printf()에서 섭씨온도와 연산식에 의해 화씨온도 출력
```

실행결과

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

비트 출력

예제 forbit.c

- int 형 정수의 32 비트 모두를 출력하는 프로그램

반복과정

- 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);
```

그림 7-21 정수의 8비트를 출력하는 for 문

실습예제 7-10 forbit.c

```
01  // file: forbit.c
02  #include <stdio.h>
03
04  #define TOTAL_BIT 32
05
06  int main(void)
07  {
08     int num = 13;
09     printf("정수 %d의 %d비트 내부값:\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  }
```

설명

04 매크로 상수 TOTAL_BIT를 32로 정의, 출력하려는 비트 수로 32 이하로 수정 가능

08 출력하려는 정수 13을 저장, 만일 표준입력으로 받으려면 이 부분을 수정

09 정보 출력

11 for 문 초기화는 int i = TOTAL_BIT-1과 같이 변수선언과 초기값 31(상수 TOTAL_BIT는 전처리 수행 후 32로 대체되어) 저장, 변수 i는 for 문 내부에서만 사용 가능한 지역변수라고 함

11 조건식인 i >= 0 에서 0까지 반복이 수행되고 음수이면 조건식이 거짓이 되어 반복을 종료

14 증감은 i-- 에서 i 값을 1 감소시키므로, --i, i -= 1, i = i - 1 도 가능

16 반복몸체인 printf()에서 각 비트를 출력, 첫 출력은 num >> 31 & 1로 최상위 비트 출력하며, 다음은 num >> 30 & 1 으로 진행되고, 마지막은 num >> 0 & 1 으로 최하위 비트를 출력

실행결과

```
정수 13의 32비트 내부값:
00000000000000000000000000001101
```

for문의 합 구하기

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

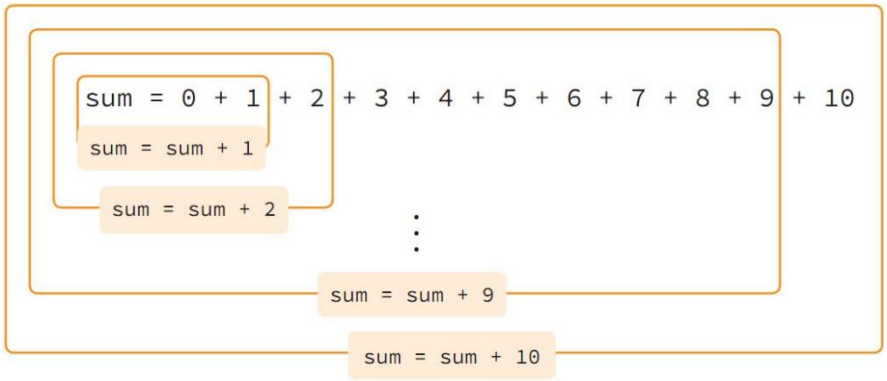


그림 7-22 1부터 10까지 합을 구하는 for문(계속)

```
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.

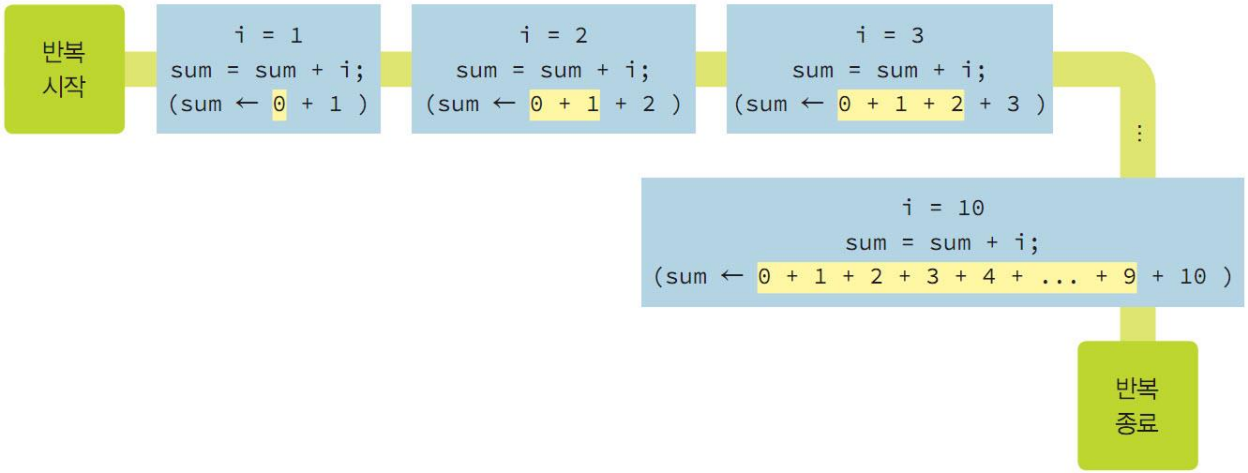


그림 7-23 1부터 10까지 합을 구하는 for문

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**

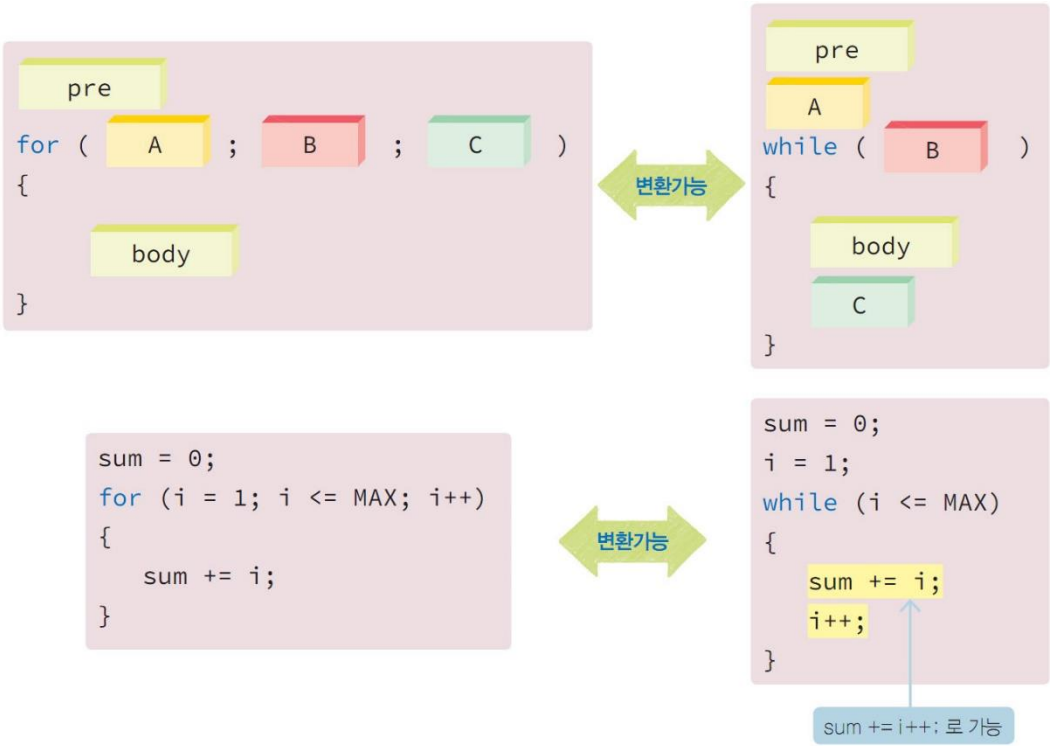


그림 7-25 반복문 for와 while간의 변환

LAB 구구단을 위한 준비 출력

- **반복문 for 문을 사용**
  - 2단부터 9단까지의 구구단의 제목을 출력하는 프로그램
- **결과**
  - === 구구단 출력 ===
  - 2단 출력
  - 3단 출력
  - 4단 출력
  - 5단 출력
  - 6단 출력
  - 7단 출력
  - 8단 출력
  - 9단 출력

Lab 7-3 forlab.c

```
01  // file: forlab.c
02
03  #include <stdio.h>
04  #define MAX 9
05
06  int main(void)
07  {
08     printf("=== 구구단 출력 ===\n");
09     for (int _____________; i <= __________; i++)
10     {
11        printf("%6d단 출력\n", i);
12     }
13
14     return 0;
15  }
```

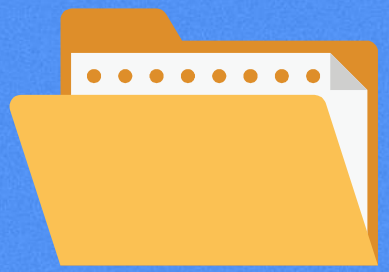
정답

```
09  for (int i = 2; i <= MAX; i++)
```

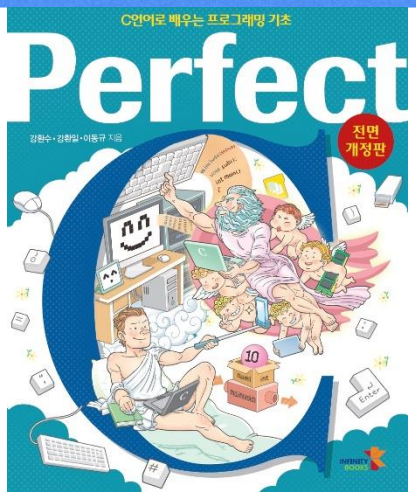
중간고사

중간고사

- **대면 필기 시험**
  - 10월 23일(금) ~ 27일(화) 중의 하루
- **대면 필기 시험 내용**
  - 1번
    - o, x 문제 5개
  - 2번
    - 빈 부분 채우기 5개
  - 3번
    - 객관식 10개
  - 4번
    - 하나의 문장이나 표현식을 쓰는 5개 문제
  - 5번
    - 프로그램 문제의 결과 쓰기 2개
      - 반복문과 조건문
      - % / 연산자
  - 6번
    - 프로그램 결과 10개
      - 연산자 ^ && (int) ++ -- ==
      - switch



03. 분기문



분기문

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

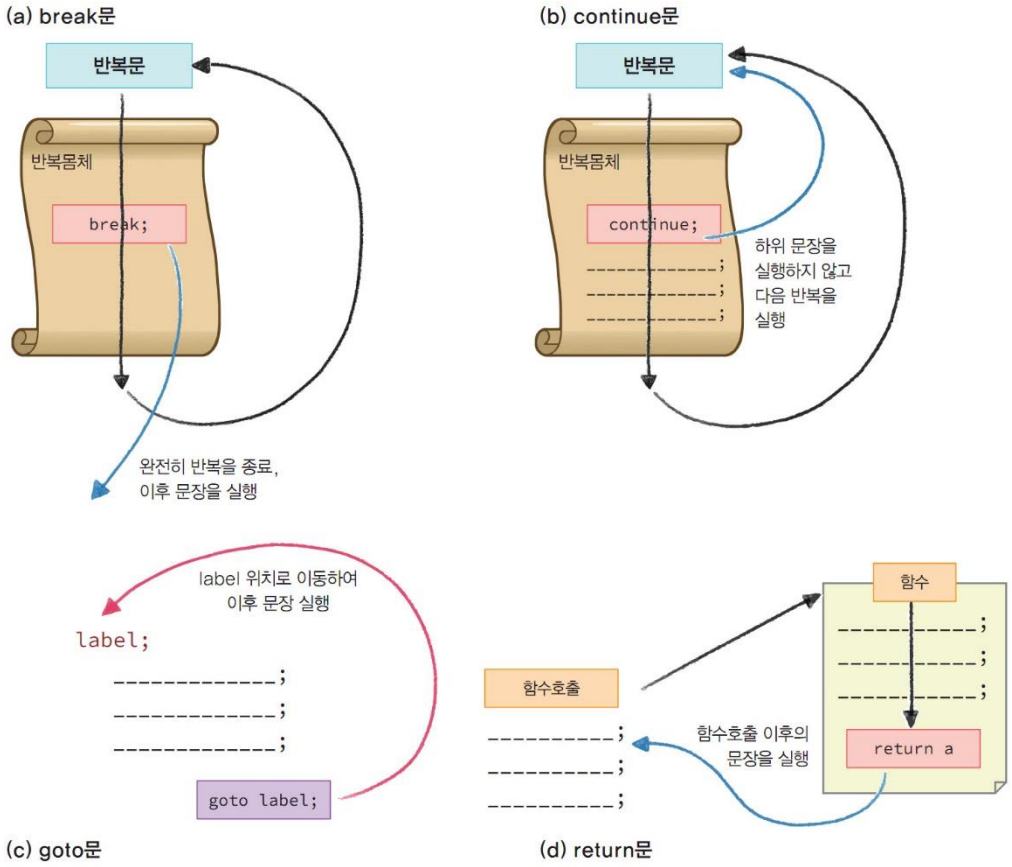


그림 7-26 여러 분기문의 개념

반복의 중단 break

- **반복내부에서 반복을 종료하려면 break 문장을 사용**
  - 만일 반복문이 중첩되어 있다면 break를 포함하는 가장 근접한 내부반복을 종료

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

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

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

그림 7-27 반복문의 break

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

}
next;
```

그림 7-28 중첩된 반복문의 break

실습예제 7-14 continue.c

```
01  // file: continue.c
02
03  #include <stdio.h>
04
05  int main(void)
06  {
07     const int MAX = 15;
08
09     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  }
```

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

continue를 만나지 않으면 이 출력문이 실행

설명

07 최대값 15를 const 상수로 정의
09 출력문
12 조건식 (I % 3 == 0)은 3으로 나누어 떨어지면 참, 떨어지지 않으면 거짓으로, (!(i % 3))으로도 가능
13 3으로 나누어 떨어지면 continue 문에 의해 for의 증감부분인 i++로 이동하여 14행의 출력이 안됨
14 들여쓰기에 주의하고, 3으로 나누어 떨어지면 않으면 14행이 실행되어 값이 출력

실행결과

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

반복의 계속 continue

- **continue 문**
  - continue 문이 위치한 이후의 반목몸체의 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

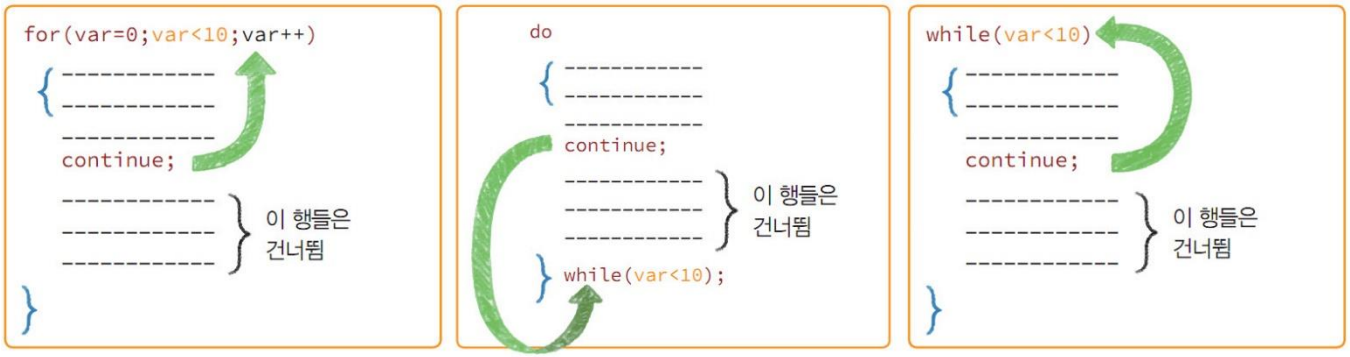


그림 7-29 다양한 continue문 이후의 실행 순서

continue 문 예제

예제 continue.c

- 1에서 15까지 정수 중에서 3으로 나누어 떨어지지 않는 수를 출력

실습예제 7-14 continue.c

```
01  // file: continue.c
02
03  #include <stdio.h>
04
05  int main(void)
06  {
07      const int MAX = 15;
08
09      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  }
```

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

continue를 만나지 않으면 이 출력문이 실행

설명

```
07  최대값 15를 const 상수로 정의
09  출력문
12  조건식 (I % 3 == 0)은 3으로 나누어 떨어지면 참, 떨어지지 않으면 거짓으로,
    (!(i % 3))으로도 가능
13  3으로 나누어 떨어지면 continue 문에 의해 for의 증감부분인 i++로 이동하여 14행의 출력이
    안됨
14  들여쓰기에 주의하고, 3으로 나누어 떨어지면 않으면 14행이 실행되어 값이 출력
```

실행결과

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

주의사항

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

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

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

        stmt1;
        stmt2;

    }

}
```

바로 위 ❶ continue를 만나면 실행되지 않는 부분

바로 위 ❷ continue를 만나면 실행되지 않는 부분

그림 7-30 중첩된 반복에서의 continue

무한반복

예제 menu.c

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

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 (  )
```

그림 7-31 반복문의 무한반복

실습예제 7-16 menu.c

```
01  // file: menu.c
02  #define _CRT_SECURE_NO_WARNINGS //scanf() 오류를 방지하기 위한 상수 정의
03
04  #include <stdio.h>
05
06  int main(void)
07  {
08     int input;
09
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  }
```

설명

```
11~14 사용자가 선택할 메뉴를 출력
15    메뉴 선택을 알리는 프롬프트 출력
16    정수 입력을 위한 scanf에서 주소연산자 빠지지 않도록 &input
17    입력 받은 정수 출력
19    반복을 빠져나가기 위한 조건문으로 입력한 값이 메뉴인 1에서 4이면 반복문을 종료하며, 메뉴에
      해당하는 번호를 입력하지 않으면 다시 반복을 실행하여 메뉴를 선택하도록 계속
20    break는 while 문을 종료
21    while (1)로 무한반복
```

실행결과

```
          [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
```

LAB 1부터 15까지 5의 배수가 아닌 정수 출력

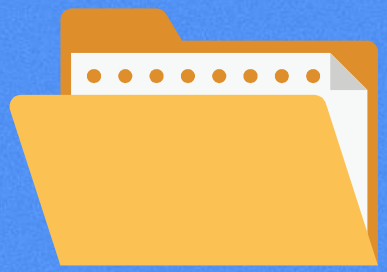
- **분기문 continue 문을 사용하여 1부터 15까지의 정수 중에서 5의 배수가 아닌 수를 출력**
  - 정수는 모두 한 줄에 계속 출력
- **결과**
  - 1에서 15까지 정수 중에서 5로 나누어 떨어지지 않는 수
  - 1 2 3 4 6 7 8 9 11 12 13 14

Lab 7-4 continuelab.c

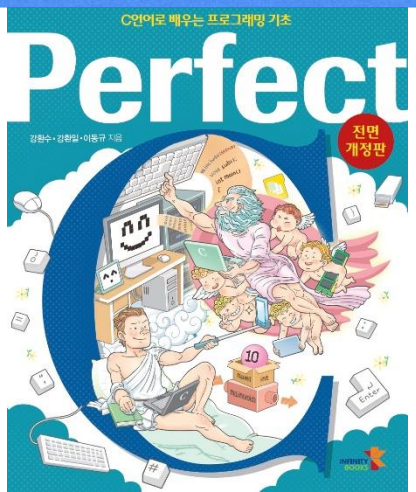
```
01  // file: continuelab.c
02
03  #include <stdio.h>
04
05  int main(void)
06  {
07     const int MAX = 15;
08
09     printf("1에서 %d까지 정수 중에서 5로 나누어 떨어지지 않는 수\n", MAX);
10     for (int i = 1; i <= MAX; i++)
11     {
12        if (__________________)
13           continue;
14        printf("%3d", i);
15     }
16     puts("");
17
18     return 0;
19  }
```

정답

```
12  if (i % 5)
```



04. 중첩된 반복문



중첩된 for

예제 nestedloop.c

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

```c
// file: nestedloop.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;
}
```

무한반복

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

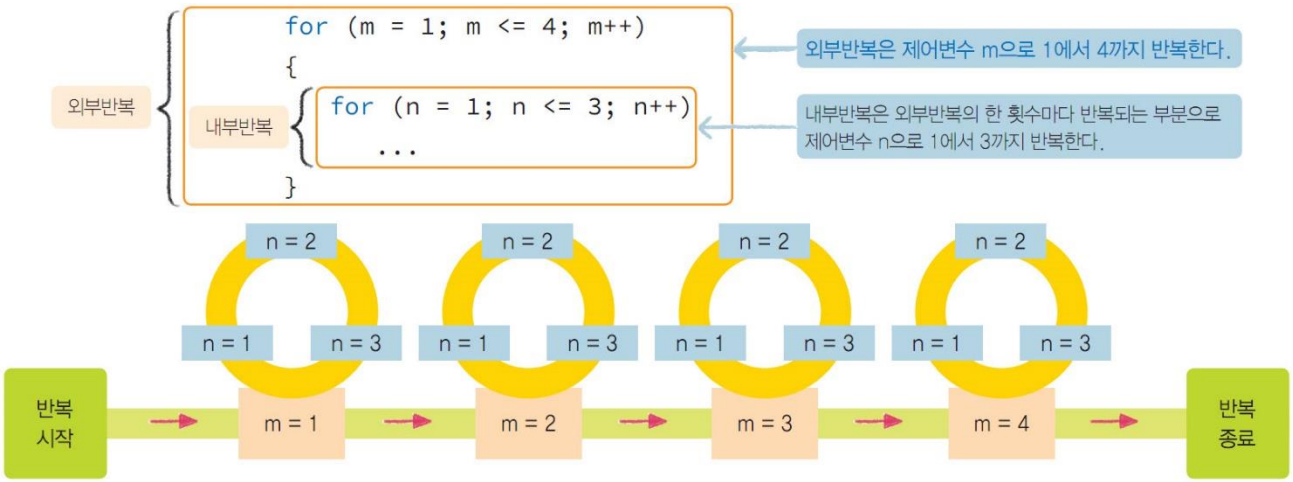


그림 7-33 중첩된 for문에서 제어변수의 변화(외부반복 제어변수: m, 내부반복 제어변수: n)

삼중 중첩반복

- **외부반복은 1에서 입력값 input까지 제어변수 i로 반복**
  - 내부반복은 제어변수 j를 사용하여 1에서 i까지 반복
  - 변수 sum은 1에서 i까지 합을 저장
  - 출력되는 콘솔 한 행에 1에서 j까지 합을 1+2+...+j = sum으로 출력
    - 조건연산자 j == i ? printf(" = ") : printf(" + ")를 이용
    - 즉 출력값이 중간이며 +를 출력하고 마지막이면 =를 출력하고 내부 for 반복을 종료한 후, 바로 그 때까지의 합인 sum을 출력

```
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
```

그림 7-35 1에서 n까지 합에서 모든 과정이 보이도록 하는 모듈

중첩된 do for

예제 loop.c

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

loops.c

```
01  // file: loops.c
02  #define _CRT_SECURE_NO_WARNINGS
03
04  #include <stdio.h>
05
06  int main(void)
07  {
08     int input, sum, i, j;
09
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  }
```

설명

```
08   외부반복의 제어변수 i, 내부반복의 제어변수 j 선언
13   입력 정수를 변수 input에 저장, &input과 같이 &가 빠지지 않도록 주의
15   외부반복의 for문으로 1에서 입력된 정수까지 반복
16~24 외부반복 for문의 반복몸체는 for(), printf() 2개의 문장으로 구성
17   내부반복의 for문으로 1에서 i(내부반복의 제어변수)까지 반복, i는 반복에 따라 1에서
     input까지 변화
17   초기화에서 j = 1 로, sum = 0가 반드시 필요
19~21 내부반복 for문의 반복몸체는 printf(), 조건연산자, 축약대입연산문 3개의 문장으로 구성
19   j 값을 한 줄에 출력
20   연산식 (j == i)는 j가 마지막이면 참으로, 반복의 중간이며 거짓, 중간이면j 값을 한 줄에
     출력한 이후에 + 연산자 출력, 마지막이면 = 를 출력
21   sum에는 1에서부터 j까지의 합이 저장
23   합 sum을 출력
25   입력 정수가 양수이면 반복 계속 실행하고, 0이나 음수이면 종료
```

실행결과

```
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

- 반복문 for 문을 사용하여 2단부터 9단까지의 구구단을 출력

Lab 7-5 mtable.c

```
01  // file: mtable.c
02
03  #include <stdio.h>
04  #define MAX 9
05
06  int main(void)
07  {
08     printf("=== 구구단 출력 ===\n");
09     for (int i = 2; i <= MAX; i++)
10     {
11        printf("%6d단 출력\n", i);
12        for (int j = 2; j <= MAX; j++)
13           _______________________________;
14                 _______________________________;
15     }
16
17     return 0;
18  }
```

정답

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

Thank you