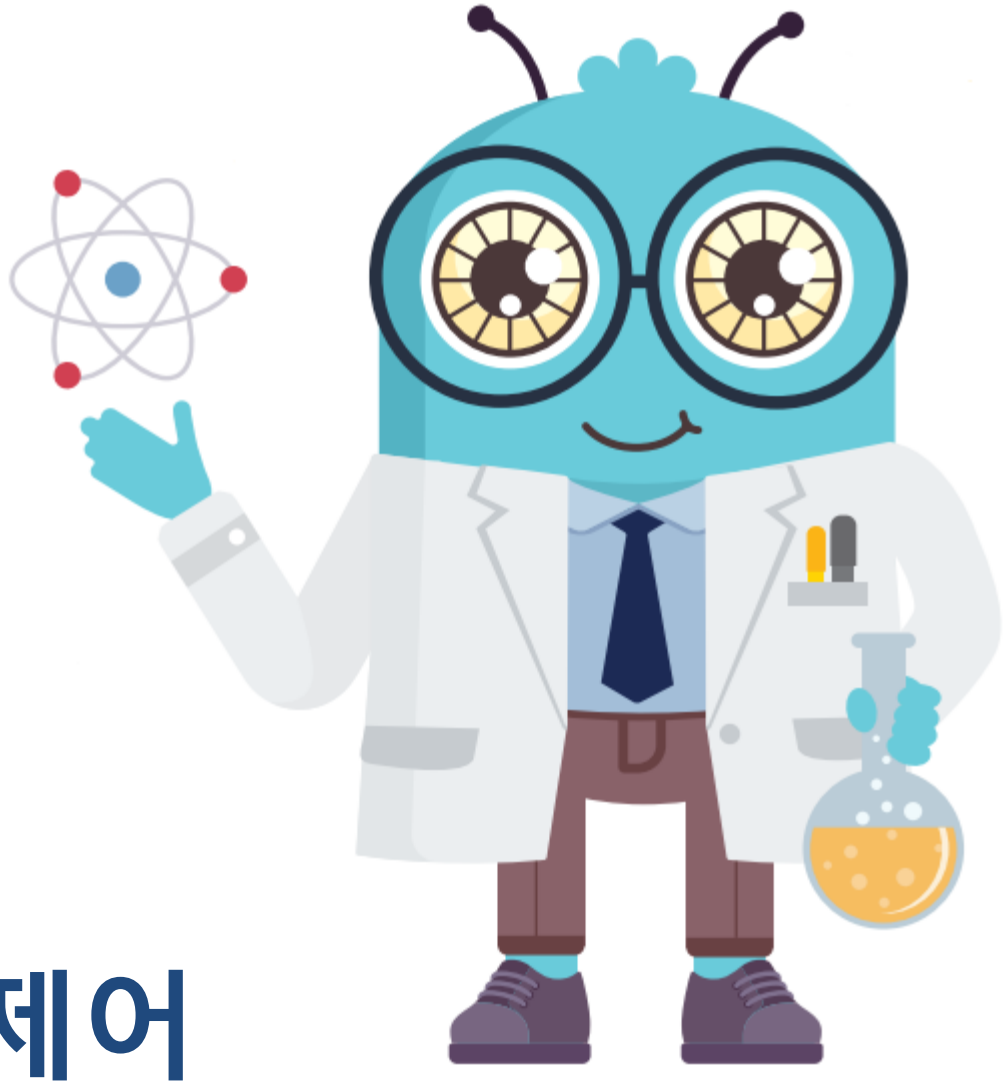


Chapter 07

while문과 흐름제어

목차

01
while문

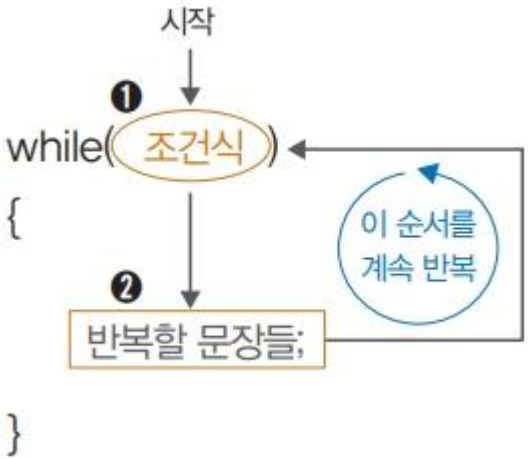
1. while문

1. for문과 while문의 비교

- **For문**

```
for(초깃값; 조건식; 증감식)
```

- **while 문의 실행 순서**
  - 조건식이 참인 동안 반복할 문장 수행
  - 중괄호가 끝나는 곳에서 조건식으로 돌아와 같은 동작 반복



1. while문

1. for문과 while문의 비교

▪ for문과 while문의 사용 코드 비교

- 0~9까지 출력하는 예

❶ 원래 for문

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

❷ 초깃값과 증감식의 위치 이동

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

❸ while문으로 변환

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

- ❶은 가장 기본적인 for문의 형태로 0~9를 출력 하는 프로그램
- ❷는 6장의 후반부에서 살펴본 예제로 ❶에 서 for문의 초깃값 'i=0'을 for문 밖으로 , 증감식 'i++'를 for문 블록의 맨 아랫부분으로 내려놓은 변형식
- while문으로 표현하면 ❸

1. for문과 while문의 비교

- for문과 while문의 사용 코드 비교

기본 7-1 for문을 while문으로 바꾸는 예 1 — 7-1.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int i;
06    i=0;                          ----- 초깃값을 지정한다.
07
08    while( i < 5 ) {              ----- 조건식이다.
09       printf("while문을 공부합니다.\n");
10       i++;                       ----- 증감식이다.
11    }
12 }
```

실행 결과

```
while문을 공부합니다.
while문을 공부합니다.
while문을 공부합니다.
while문을 공부합니다.
while문을 공부합니다.
```

1. while문

1. for문과 while문의 비교

- **for 문을 while 문으로 변환하는 방법**
  - [기본 6-2]의 7행에서는 for(i=0; i < 5; i++)를 사용
  - 이 for문을 while문으로 변환하려면 `초깃값을 while문 위로 빼고 증감식은 while문 블록 안의 맨 아래에 놓음
  - for문 안의 세미콜론(;)을 제거

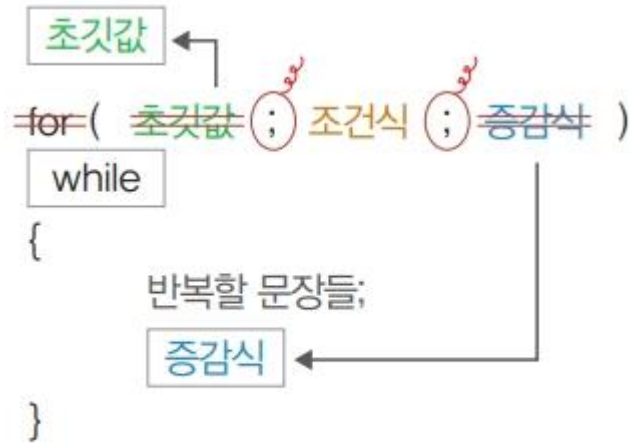


그림 7-2 for문을 while문으로 변환하는 방법

1. while문

- **for 문을 while 문으로 변환하는 방법**

응용 7-2 for문을 while문으로 바꾸는 예 2 — 7-2.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int hap=0;
06    int i;
07
08    __1__                          ---- 초깃값을 지정한다.
09    while( i <= 10 ) {             ---- 조건식이다.
10       hap = hap + i;
11    __2__                          ---- 증감식이다.
12    }
13
14    printf(" 1에서 10까지의 합: %d \n", hap);
15 }
```

실행 결과

```
1에서 10까지의 합: 55
```

1. while문

2. 무한루프를 위한 while문

- 조건식이 무조건 참이어야 함
- for(; ;)와 동일한 역할
- while(1) 로 표현

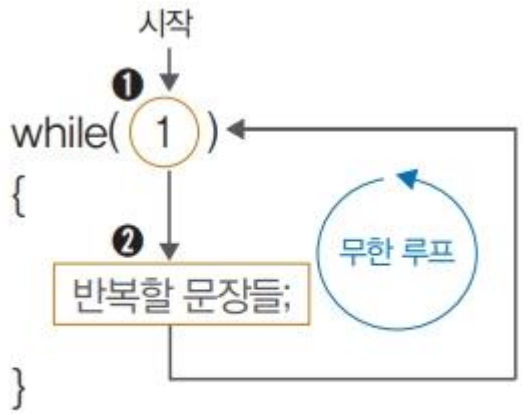


그림 7-3 while문을 이용한 무한 루프

2. 무한루프를 위한 while문

- [응용 6-18]의 for문을 이용한 무한 루프 예제를 while문을 이용한 무한 루프로 변환

기본 7-3 while문으로 무한 루프 만들기 7-3.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    int a, b;
06
07    while( 1 )                          ---- 무한 루프를 만드는 코드이다.
08    {
09       printf("더할 두 수 입력 (멈추려면 Ctrl+C) : ");
10       scanf("%d %d", &a, &b);          ------ 입력값을 공백으로 분리한다.
11
12       printf("%d + %d = %d \n", a, b, a+b);   ------ 결과를 출력한다.
13    }
14 }
```

실행 결과

```
더할 두 수 입력 (멈추려면 Ctrl+C) : 55  22
55 + 22 = 77
더할 두 수 입력 (멈추려면 Ctrl+C) : 77  128
77 + 128 = 205
더할 두 수 입력 (멈추려면 Ctrl+C) :
```

1. while문

2. 무한루프를 위한 while문

- 사용자가 실행을 취소(Ctrl+C)할 때까지 실행되는 계산기 프로그램 작성

응용 7-4 무한 루프를 활용한 계산기 7-4.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    int a, b;
06    char ch;
07
08    __1__                                  ---- 무한 루프를 만드는 코드이다.
09    {
10       printf("계산할 두 수를 입력 (멈추려면 Ctrl+C) : ");
11       scanf("%d %d", &a, &b);             ------ 연산할 2개의 수를 입력받는다.
12
13       printf("계산할 연산자를 입력하세요 : ");
14       scanf(" %c", &ch);                  ------ 연산자를 입력받는다. %c 앞에 공백 문자를 넣는다.
15
16       __2__(ch)                           ------ 입력받은 ch 연산자에 의해 +, -, *, /, %로 분기한다. 그 외는 오류 메시지를 출력한다.
17       {
18       case '+' :
19          printf("%d + %d = %d 입니다. \n", a, b, a+b);
20          break;
21       case '-' :
```

1. while문

2. 무한루프를 위한 while문

- 사용자가 실행을 취소(Ctrl+C)할 때까지 실행되는 계산기 프로그램 작성

```
22          printf("%d - %d = %d 입니다. \n", a, b, a-b);
23          break;
24       case '*' :
25          printf("%d * %d = %d 입니다. \n", a, b, a*b);
26          break;
27       case '/' :
28          printf("%d / %d = %f 입니다. \n", a, b, a/(float)b);
29          break;
30       case '%' :
31          printf("%d %% %d = %d 입니다. \n", a, b, a%b);
32          break;
33       default :
34          printf("연산자를 잘못 입력했습니다. \n");
35       }
36    }
37 }
```

실행 결과

```
계산할 두 수를 입력 (멈추려면 Ctrl+C) : 22  33
계산할 연산자를 입력하세요 : *
22 * 33 = 726 입니다.
계산할 두 수를 입력 (멈추려면 Ctrl+C) : 10  4
계산할 연산자를 입력하세요 : %
10 % 4 = 2 입니다.
계산할 두 수를 입력 (멈추려면 Ctrl+C) :
```

02
do~while문

2. do~while문

1. do~while문과 while문의 차이

- 조건식을 확인하기 전에 '반복할 문장'을 수행됨, 무조건 한 번은 실행됨
- 형식은 while 문과 동일하지만, 조건식이 아래에 위치

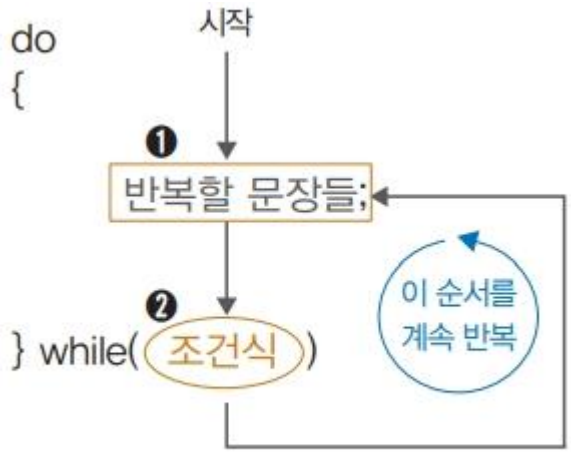


그림 7-4 do~while문의 형식과 실행 순서

- while문의 경우 처음 의 조건식이 거짓이면 '반복할 문장들'을 한 번도 실행하지 않음
- 하지만 do~while문에서는 조건식을 확인하기 전에 일단 '반복할 문장들'을 실행하므로 조건식이 거짓이든 참이든 무조건 한 번은 실행

2. do~while문

1. do~while문과 while문의 차이

기본 7-5 do~while문 사용 예 1 — 7-5.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int a = 100;
06
07    while( a == 200 )
08    {
09       printf("while문 내부에 들어 왔습니다.\n");
10    }
11
12    do {
13       printf("do ~ while문 내부에 들어 왔습니다.\n");
14    } while( a == 200 );
15 }
```

----- 조건식을 먼저 판단하므로 while문 내부가 실행되지 않는다.

----- 먼저 문장을 실행한 후 조건식을 판단하므로 do~while문 내부가 실행된다.

실행 결과

```
do ~ while문 내부에 들어 왔습니다.
```

2. do~while문

1. do~while문과 while문의 차이

응용 7-6 do~while문 사용 예 2 — 7-6.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    int menu;
06
07    __1__ {                          ----- do~while문이므로 한 번은 꼭 실행된다.
08       printf("\n손님 주문하시겠습니까 ? \n");
09       printf("<1>카페라떼 <2>카푸치노 <3>아메리카노 <4>그만 시킬래요 ==> ");
10       scanf("%d", &menu);              ----- 커피를 선택한다.
11
```

2. do~while문

1. do~while문과 while문의 차이

```
12      switch(menu)                                  ----- 선택한 커피에 따라서
13      {                                                   주문을 접수한다.
14      case 1 : printf("#카페라떼 주문하셨습니다.\n"); break;
15      case 2 : printf("#카푸치노 주문하셨습니다.\n"); break;
16      case 3 : printf("#아메리카노 주문하셨습니다.\n"); break;
17      case 4 : printf("주문하신 커피 준비하겠습니다.\n"); break;
18      default : printf("잘못 주문하셨습니다.\n");
19      }
20    } __2__ (menu != 4);          ----- 선택한 메뉴가 4번이 아니면 계속 반복해서 주문을 받는다.
21  }
```

정답_1 do 2 while

실행 결과

```
손님 주문하시겠습니까 ?
<1>카페라떼 <2>카푸치노 <3>아메리카노 <4>그만 시킬래요 ==> 2
#카푸치노 주문하셨습니다.

손님 주문하시겠습니까 ?
<1>카페라떼 <2>카푸치노 <3>아메리카노 <4>그만 시킬래요 ==> 3
#아메리카노 주문하셨습니다.

손님 주문하시겠습니까 ?
<1>카페라떼 <2>카푸치노 <3>아메리카노 <4>그만 시킬래요 ==> 4
주문하신 커피 준비하겠습니다.
```

03
기타 제어문

3. 기타 제어문

1. 반복문을 탈출하는 break문

- for, while, do~while과 같은 반복문을 탈출할 때 사용
- if 문과 결합하여 무한루프 안에 사용
  - 무한루프를 돌다 특정 조건을 만족하면 프로그램을 종료하는 역할

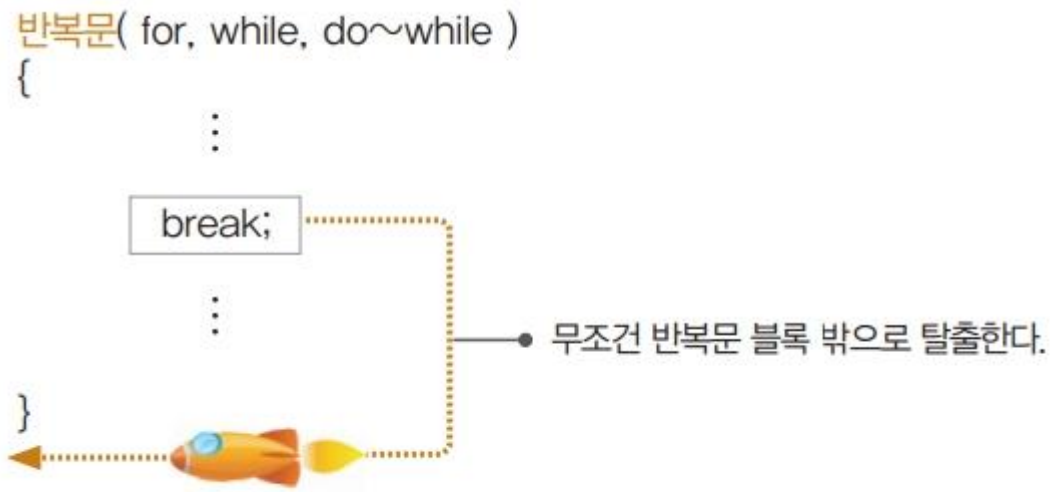


그림 7-5 break문의 작동

3. 기타 제어문

1. 반복문을 탈출하는 break문

기본 7-7 break문 사용 예 1 — 7-7.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int i;
06
07    for( i=1; i <= 100; i++ )          ----- 100번 반복한다.
08    {
09       printf("for문을 %d회 실행했습니다.\n", i);   ----- 변수 i번째를 출력한다.
10       break;                          ----- 무조건 for문을 빠져나간다.
11    }
12
13    printf("for문을 종료했습니다.\n");
14 }
```

실행 결과

```
for문을 1 회 실행했습니다.
for문을 종료했습니다.
```

3. 기타 제어문

1. 반복문을 탈출하는 break문

- [기본 7-7]은 10행의 break문이 없다면 무조건 문장을 100번 출력하는 프로그램
- break문은 무한 루프를 돌다가 특정 조건이 되면 빠져나가도록 할 때 사용

기본 7-8 break문 사용 예 2 — 7-8.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    int a, b;
06
07    while( 1 )                          ---- 무한 루프를 만드는 코드이다.
08    {
09       printf("더할 두 수 입력 (멈추려면 0을 입력) : ");
10       scanf("%d %d", &a, &b);          ---- 2개의 수를 입력받는다.
11
12       if(a == 0)                       ---- 첫 번째 입력값이 0이면 무조건
13          break;                             while문을 빠져나간다.
```

3. 기타 제어문

1. 반복문을 탈출하는 break문

```
14
15        printf("%d + %d = %d \n", a, b, a+b);
16    }
17
18    printf("0을 입력해서 for문을 탈출했습니다.\n");
19 }
```

실행 결과

```
더할 두 수 입력 (멈추려면 0을 입력) : 55  22
55 + 22 = 77
더할 두 수 입력 (멈추려면 0을 입력) : 77  128
77 + 128 = 205
더할 두 수 입력 (멈추려면 0을 입력) : 0  0
0을 입력해서 for문을 탈출했습니다.
```

- [기본 7-8]의 10행에서 입력된 값 중 처음 값(변수 a)에 0을 넣었다면 12행의 a==0이 참이 되 고 13행의 break문을 만나 17행으로 이동함으로써 반복문을 탈출

3. 기타 제어문

1. 반복문을 탈출하는 break문

여기서 잠깐 실행 문장이 하나일 때의 블록 사용

- 실행할 문장이 하나뿐이라면 블록({ })을 사용하거나 사용하지 않아도 됨
- 줄을 띄어도 되고 한 줄에 다 써도 상관없음

❶

```
if(a= = 0)
   break;
```

❷

```
if(a= = 0)
   { break; }
```

❸

```
if(a= = 0) { break; }
```

❹

```
if(a= = 0) break;
```

3. 기타 제어문

1. 반복문을 탈출하는 break문

응용 7-9 break문 사용 예 3 — 7-9.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int hap = 0;
06    int i;
07
08    for( i=1; i <=100; i++ )        ----- i 값이 1부터 100까지 100회 반복된다.
09    {
10       hap = hap + i;               ----- i 값이 hap에 누적된다.
11
12       if( __1__ )                  ----- hap이 1000보다 크거나 같으면 for문을 빠져나간다.
13          break;
14    }
15
16    printf(" 1~100의 합에서 최초로 1000이 넘는 위치는? : %d\n", i);
17 }
```

실행 결과

```
1~100의 합에서 최초로 1000이 넘는 위치는? : 45
```

정답_ 1 : hap >= 1000

3. 반복문으로 다시 돌아가는 continue문

- 블록의 끝으로 이동한 후 반복문을 처음부터 다시 수행

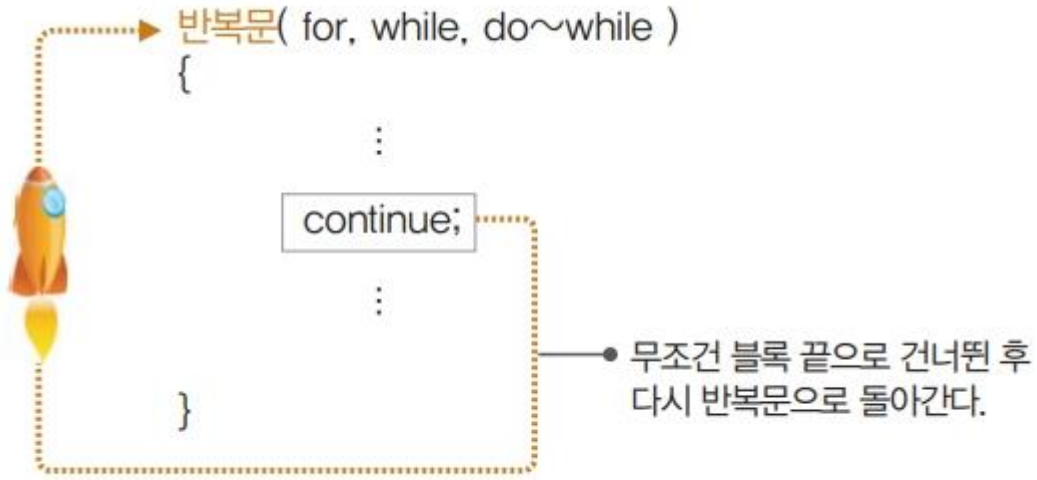


그림 7-6 continue문의 작동

3. 반복문으로 다시 돌아가는 continue문

기본 7-10 continue문 사용 예 7-10.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int hap = 0;
06    int i;
07
08    for( i=1; i <= 100; i++ )          ---- i 값이 1부터 100까지 100회 반복된다.
09    {
10       if( i % 3 == 0 )                ------ i 값을 3으로 나눈 나머지값이 0이면(3의 배수이면)
11          continue;                          블록의 끝으로 건너뛰고 다시 8행으로 돌아간다.
12
13       hap += i;                       ------ 3의 배수가 아닌 i 값이 누적된다.
14    }
15
16    printf(" 1~100까지의 합(3의 배수 제외): %d\n", hap);   ---- 누적된 값을 출력한다.
17 }
```

실행 결과

```
1~100까지의 합(3의 배수 제외): 3367
```

3. 반복문으로 다시 돌아가는 continue문

- [기본 7-10]의 10행의 i % 3 == 0은 i를 3으로 나눈 나머지값이 0일 때 참(즉 3의 배수)이라는 의미

제1회: i 값 1을 3으로 나누면 나머지는 1(거짓)이다. ⇨ hap += 1을 수행한다.
제2회: i 값 2를 3으로 나누면 나머지는 2(거짓)이다. ⇨ hap += 2를 수행한다.
제3회: i 값 3을 3으로 나누면 나머지는 0(참)이다. ⇨ continue문을 수행한다.
끝(14행)으로 건너뛰고 다시 8행으로 올라가서 증감식을 수행한다.

제4회: i 값 4를 3으로 나누면 나머지는 1(거짓)이다. ⇨ hap += 4를 수행한다.
제5회: i 값 5를 3으로 나누면 나머지는 2(거짓)이다. ⇨ hap += 5를 수행한다.
제6회: i 값 6을 3으로 나누면 나머지는 0(참)이다. ⇨ continue문을 수행한다.
끝(14행)으로 건너뛰고 다시 8행으로 올라가서 증감식을 수행한다.

제7회: …

3. 지정한 위치로 이동하는 goto문

- 지정된 레이블(label)로 건너뛰게 하는 명령문
- 프로그램의 흐름을 복잡하게 만드는 단점이 있음

그림 7-7 goto문의 작동

3. 지정한 위치로 이동하는 goto문

기본 7-11 goto문 사용 예 — 7-11.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int hap = 0;
06    int i;
07
08    for( i=1; i <= 100; i++ )          ----- i 값이 1부터 100까지 100회 반복된다.
09    {
10       hap += i;                       ------ 합계를 누적한다.
11
12       if(hap > 2000)                  ------ 누적된 값이 200을 넘으면 mygoto:로
13          goto mygoto;                        무조건 이동한다.
14    }
15
16 mygoto:                               ----- goto문이 이동할 레이블이다.
17     printf("1부터 %d까지 합하면 2000이 넘어요.\n", i);
18 }
```

실행 결과

```
1부터 63까지 합하면 2000이 넘어요.
```

4. 현재 함수를 불렀던 곳으로 돌아가는 return문

- 현재 실행중인 함수를 끝내고, 해당 함수를 호출한 곳으로 돌아가게 함
- return 문을 만나면 프로그램이 종료되는 효과

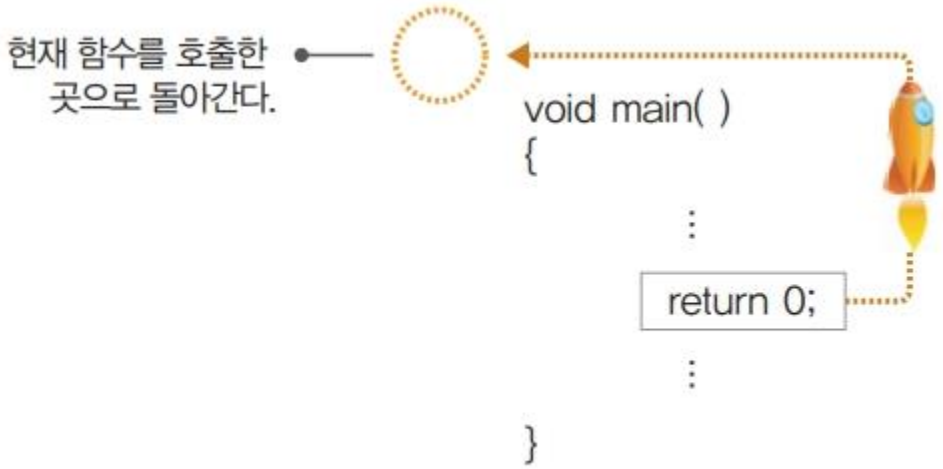


그림 7-8 return문의 작동

4. 현재 함수를 불렀던 곳으로 돌아가는 return문

기본 7-12 return문 사용 예 — 7-12.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int hap = 0;
06    int i;
07
08    for( i=1; i <= 100; i++ )                  ----- 1~100의 합계가 누적된다.
09       hap += i;
10
11    printf("1부터 100까지의 합은 %d 입니다.\n", hap);   ----- 합계를 출력한다.
12    return;                                    ----- 현재 함수를 호출한 곳으로
13                                                     되돌린다.
14    printf("프로그램의 끝입니다.");                 ----- 한 번도 실행되지 않는다.
15 }
```

실행 결과

```
1부터 100까지의 합은 5050 입니다.
```

*

예제 모음

[예제모음 17] 배수의 합계를 구하는 계산기

예제 설명 입력한 두 수 사이의 합계를 구하되 원하는 배수를 선택하는 프로그램이다. 예를 들어 100~200 중에서 4배수의 합계를 구할 수 있다.

실행 결과

```
합계의 시작값 ==> 100
합계의 끝값 ==> 200
배수 ==> 4
100부터 200까지의 4배수의 합계 ==> 3900
```

[예제모음 17] 배수의 합계를 구하는 계산기

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    int start, end;                  ----- 변수 선언과 함께 초기화한다.
06    int basu, i;
07    int hap = 0;
08
09    printf("합계의 시작값 ==> ");
10    scanf("%d", &start);             ----- 시작값을 입력한다.
11    printf("합계의 끝값 ==> ");
12    scanf("%d", &end);               ----- 끝값을 입력한다.
13    printf("배수 ==> ");
14    scanf("%d", &basu);              ----- 배숫값을 입력한다.
15
16    i = start;                       ----- i 값을 시작값으로 초기화한다.
17    while(i <= end)                  ----- i 값이 끝값보다 작거나 같을 동안 반복한다.
18    {
19       if(i % basu == 0)             ------ i 값이 입력한 배수에 해당하면 합계에 누적된다.
20          hap = hap + i;
21
22       i++;
23    }
24
25    printf("%d부터 %d까지의 %d배수의 합계 ==> %d\n", start, end, basu, hap);
26 }
```

[예제모음 18] 입력한 문자열의 종류 구분

예제 설명 입력한 문자열에 대문자와 소문자, 숫자가 각각 몇 개 입력되었는지 세는 프로그램이다. 그 외는 무시한다.

실행 결과

```
문자열을 입력(100자 이내) : IT_CookBook_2024
대문자 4개, 소문자 6개, 숫자 4개
```

[예제모음 18] 입력한 문자열의 종류 구분

```
01 #define  _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    char str[100];                          ----- 문자열 배열과 문자형 변수를 선언한다.
06    char ch;
07
08    int upper_cnt = 0, lower_cnt=0, digit_cnt=0;   ----- 대문자, 소문자, 숫자의 개수를
09    int i;                                          초기화한다.
10
11    printf("문자열을 입력(100자 이내) : ");
12    scanf("%s", str);                       ----- 문자열을 입력받는다.
13
14    i = 0;                                  ----- 문자열의 위치를 나타낼 변수 i이다.
15    do {                                    ----- 입력한 문자열의 끝(\0)까지 반복한다.
16       ch = str[i];                         ------ 문자열에서 한 글자를 추출한다.
17
18       if(ch >= 'A' && ch <= 'Z')           ------ 추출한 글자 하나가 A~Z에 속하면 대문자의 개수가
19          upper_cnt ++;                             하나 증가한다.
20       if(ch >= 'a' && ch <= 'z')           ------ 추출한 글자 하나가 a~z에 속하면 소문자의 개수가
21          lower_cnt ++;                             하나 증가한다.
22       if(ch >= '0' && ch <= '9')           ------ 추출한 글자 하나가 0~9에 속하면 숫자의 개수가
23          digit_cnt++;                              하나 증가한다.
24
25       i++;                                 ------ 다음 글자를 추출하기 위해 i 값을
26    } while(ch != '\0');                             증가시킨다.
27
28    printf("대문자 %d개, 소문자 %d개, 숫자 %d개\n", upper_cnt, lower_cnt, digit_cnt);
29 }
```

[예제모음 19] 입력한 숫자만큼 별표 출력

예제 설명 입력한 숫자만큼의 별 모양을 출력하는 프로그램이다. 예를 들어 5914를 입력하면 각 줄에 별을 5개, 9개, 1개, 4개 출력한다.

실행 결과

```
숫자를 여러 개 입력 : 5914
*****
*********
*
****
```

[예제모음 19] 입력한 숫자만큼 별표 출력

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    char str[100];                  ----- 문자열 배열과 문자형 변수를 선언한다.
06    char ch;
07
08    int i, k;                       ----- 정수형 변수를 선언한다. i, k는 반복문에서
09    int star;                             사용한다. star는 별의 개수를 추출한다.
10
11    printf("숫자를 여러 개 입력 : ");
12    scanf("%s", str);               ----- 문자열(숫자만)을 입력받는다.
13
14    i = 0;                          ----- 문자열의 위치를 나타낼 변수 i이다.
15    ch = str[i];                    ----- 문자열에서 한 글자(숫자)를 추출한다.
16    while(ch != '\0') {             ----- 문자가 있는 동안 반복한다(4회 반복).
17       star = (int)ch - 48;         ------ 아스키코드 값으로 계산해서 문자를 숫자로
18                                           변환한다.
19       for(k=0; k < star; k++)      ------ 별의 개수만큼 *를 화면에 출력한다.
20          printf("*");
21
22       printf("\n");                ------ 한 줄을 띄운다.
23       i = i + 1;                   ------ 다음 문자를 추출하기 위해 i 값을 증가시킨다.
24       ch = str[i];
25    }
26 }
```

감사합니다!