C프로그래밍

6 While문과 기타 제어문

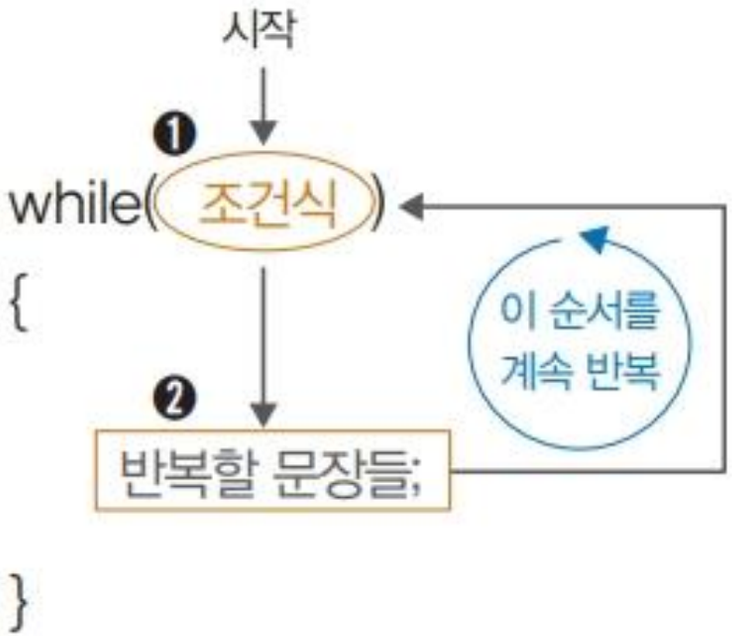
01
while문

for문과 while문의 비교 (1/5)

- **for문**

```
for(초깃값; 조건식; 증감식)
```

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



초깃값과 증감식을 생략한

for문을 생각해보라!

for문과 while문의 비교 (2/5)

- **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 ❶에서 for문의 초깃값 'i=0'을 for문 밖으로,

증감식 'i++'를 for문 블록의 맨 아랫부분으로 내려놓은 변형식

✓ while문으로 표현하면 ❸

for문과 while문의 비교 (3/5)

- **for문과 while문의 사용 코드 비교 (vs. 기본 6-2)**

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.
```

for문과 while문의 비교 (4/5)

- **기본 7-1 복기**
  - ✓ [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

for문과 while문의 비교 (5/5)

- **1에서 10까지의 합을 구하는 예제**

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
```

정답_ 1 i=1; 2 i++;

무한루프를 위한 while문 (1/4)

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

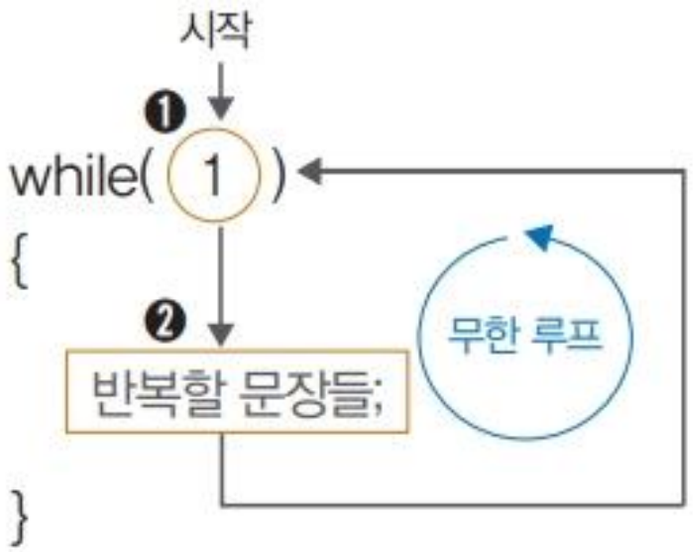


그림 7-3 while문을 이용한 무한 루프

무한루프를 위한 while문 (2/4)

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

기본 7-3 while문으로 무한 루프 만들기

```
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) :
```

무한루프를 위한 while문 (3/4)

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

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
12                                                 입력받는다.
13       printf("계산할 연산자를 입력하세요 : ");
14       scanf(" %c", &ch);                  ------ 연산자를 입력받는다.
15                                                 %c 앞에 공백 문자를 넣는다.
16       __2__(ch)                           ------ 입력받은 ch 연산자에 의해
17       {                                          +, -, *, /, %로 분기한다.
18       case '+' :                                 그 외는 오류 메시지를
19          printf("%d + %d = %d 입니다. \n", a, b, a+b);   출력한다.
20          break;
21       case '-' :
```

무한루프를 위한 while문 (4/4)

- **사용자가 실행을 취소(Ctrl+C)할 때까지 실행되는 계산기 프로그램 (cont'd)**

```
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) :
```

정답_1 while (1) 2 switch

02
do~ while문

do~while문과 while문의 차이 (1/4)

- **조건식을 확인하기 전에 먼저 '반복할 문장'을 수행**
  - ✓ 반복할 문장이 무조건 한 번은 실행됨
  - ✓ while문의 경우 처음의 조건식이 거짓이면 '반복할 문장들'을 한 번도 실행하지 않음
- **형식은 while 문과 동일하지만, 조건식이 아래에 위치**

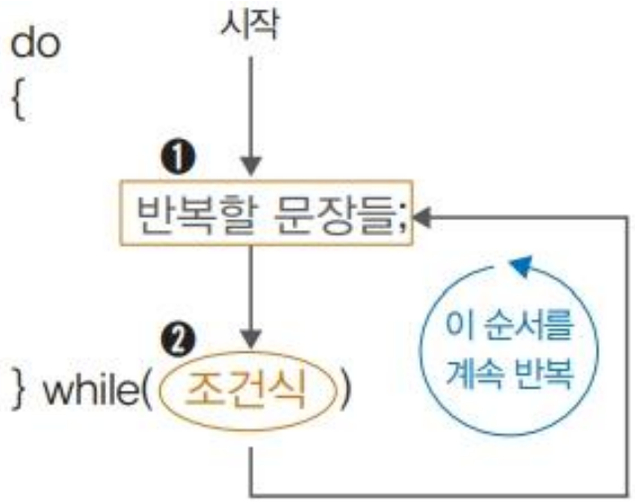


그림 7-4 do~while문의 형식과 실행 순서

do~while문과 while문의 차이 (2/4)

- **do~while문과 while문의 비교**

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.
```

do~while문과 while문의 차이 (3/4)

- **do~while문을 이용한 키오스크 예제**

응용 7-6 do~while문 사용 예 2

```
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
```

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.
```

do~while문과 while문의 차이 (4/4)

- **do~while문을 이용한 키오스크 예제 (cont'd)**

```
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

03
기타 제어문

반복문을 탈출하는 break문 (1/5)

- **for, while, do~while과 같은 반복문을 탈출할 때 사용**
  - ✓ 추가적인 반복의 조건을 사용하는 형태로 활용 가능
- **if 문과 결합하여 무한루프 안에 사용**
  - ✓ 무한루프를 돌다 특정 조건을 만족하면 프로그램을 종료하는 역할

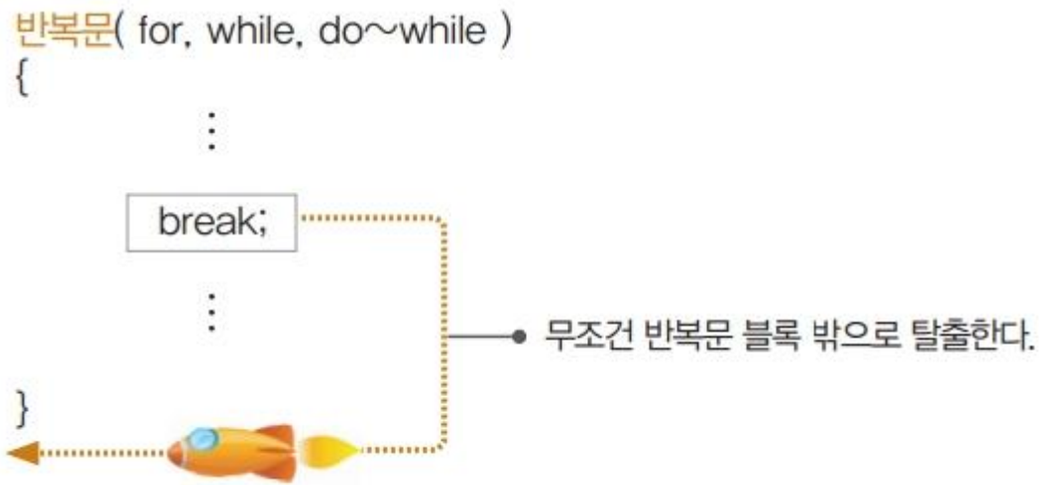


그림 7-5 break문의 작동

반복문을 탈출하는 break문 (2/5)

- **break를 사용하는 간단한 예제**

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.
```

반복문을 탈출하는 break문 (3/5)

- **무한반복문에서 break를 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.
```

반복문을 탈출하는 break문 (4/5)

- **무한반복문에서 break를 사용 (cont'd)**
  - ✓ 10행에서 입력된 값 중 처음 값(변수 a)에 0을 넣었다면 12행의 a==0이 참이 되고 13행의 break문을 만나 17행으로 이동함으로써 반복문을 탈출

```
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.
```

반복문을 탈출하는 break문 (5/5)

- **조건을 추가해서 break를 사용**

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

반복문의 처음으로 돌아가는 continue문 (1/3)

- **블록의 끝으로 이동한 후 반복문을 처음부터 다시 수행**

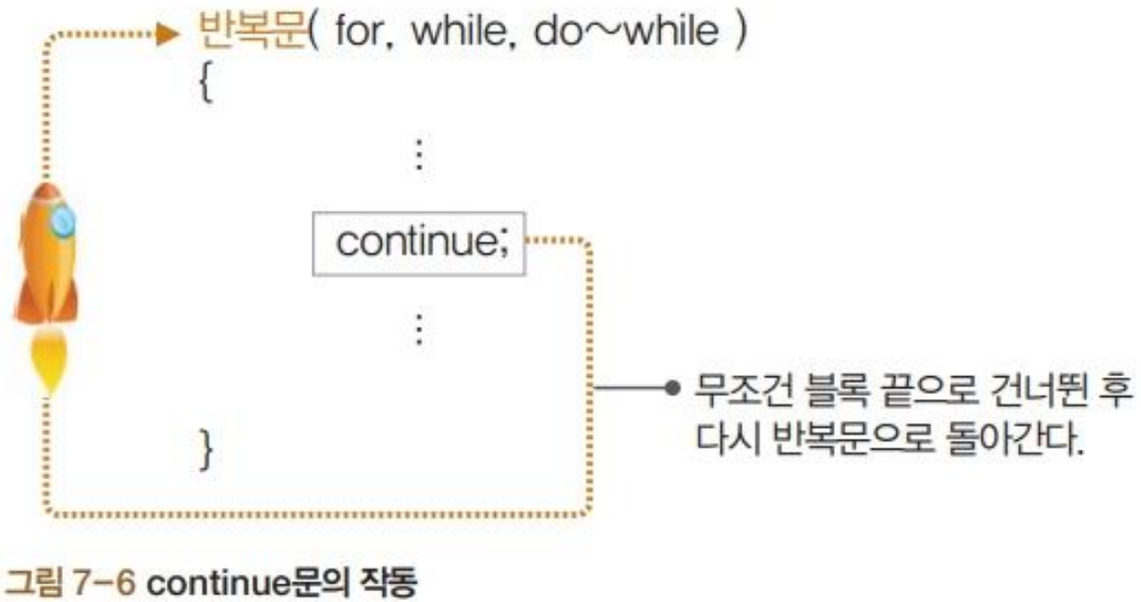


그림 7-6 continue문의 작동

반복문의 처음으로 돌아가는 continue문 (2/3)

- **1~100의 숫자 중 3의 배수가 아닌 숫자들의 합을 계산**

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++ )
09    {
10       if( i % 3 == 0 )
11          continue;
12
13       hap += i;
14    }
15
16    printf(" 1~100까지의 합(3의 배수 제외): %d\n", hap);
17 }
```

실행 결과

```
1~100까지의 합(3의 배수 제외): 3367
```

08행 ---- i 값이 1부터 100까지 100회 반복된다.

10~11행 ---- i 값을 3으로 나눈 나머지값이 0이면(3의 배수이면) 블록의 끝으로 건너뛰고 다시 8행으로 돌아간다.

13행 ---- 3의 배수가 아닌 i 값이 누적된다.

16행 ---- 누적된 값을 출력한다.

반복문의 처음으로 돌아가는 continue문 (3/3)

- **기본 7-10 복기**
  - ✓ 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: …

지정한 위치로 이동하는 goto문 (1/2)

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

그림 7-7 goto문의 작동

지정한 위치로 이동하는 goto문 (2/2)

- **1부터 합을 누적했을 때 합이 2000이 넘는 수 구하기**

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.
```

현재 함수를 불렀던 곳으로 돌아가는 return문 (1/2)

- **현재 실행중인 함수를 끝내고, 해당 함수를 호출한 곳으로 돌아가게 함**
- **main( )에서 return 문을 만나면 프로그램이 종료되는 효과**

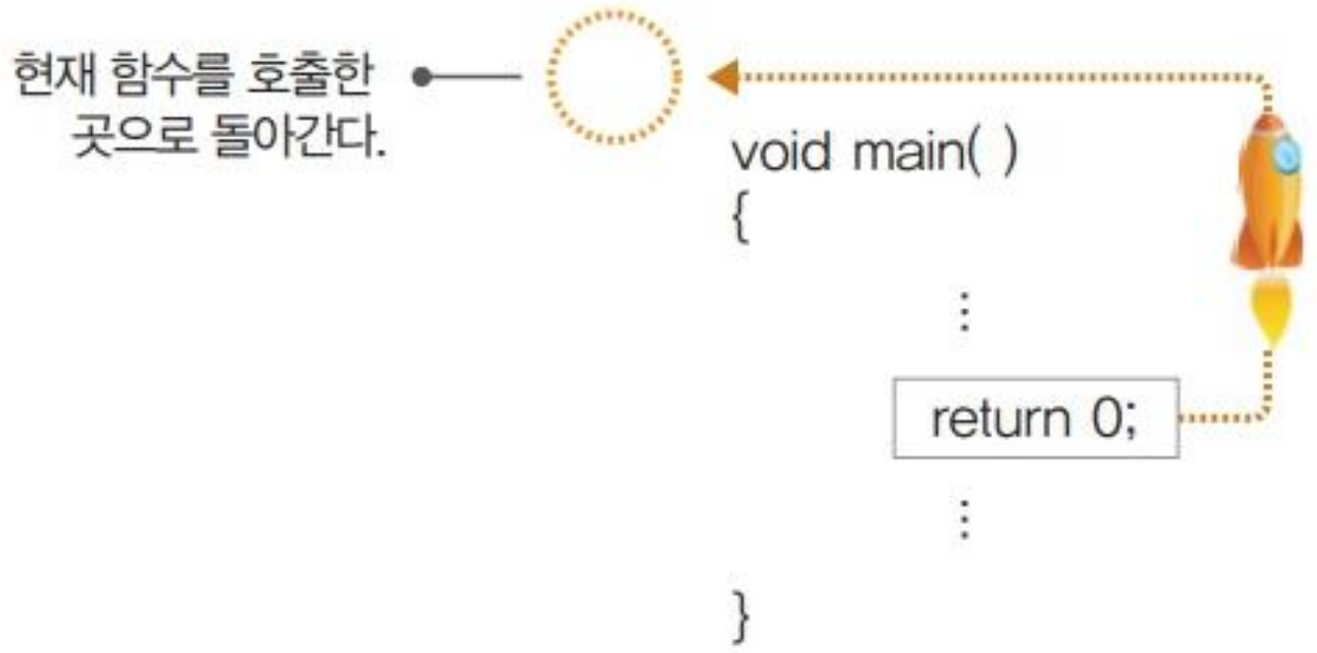


그림 7-8 return문의 작동

현재 함수를 불렀던 곳으로 돌아가는 return문 (2/2)

- **return의 간단한 사용 예**

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
                                                     되돌린다.
13
14    printf("프로그램의 끝입니다.");              ----- 한 번도 실행되지 않는다.
15 }
```

예 제

[예제 01] 배수의 합계를 구하는 계산기

- **while문을 이용하여 배수의 합계를 계산**

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
```

[예제 02] 구구단 출력 프로그램 v2.0

- **중첩 while문을 사용하고, 제목도 출력하는 구구단 프로그램**

실행 결과

```
#제2단# #제3단# #제4단# #제5단# #제6단# #제7단# #제8단# #제9단#

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

[예제 03] 제한된 구구단 작성

- **짝수 단(2단, 4단, 6단, 8단)만 출력하는 구구단 프로그램**
  - ✓ 2단은 2 x 2, 4단은 4 x 4, 6단은 6 x 6, 8단은 8 x 8까지 출력

<프로그램 구현 힌트>

- **중첩 for문 사용**
  - ✓ 바깥쪽 for문은 1~9단까지 반복, 안쪽 for문은 x 1 ~ x 9까지 반복
- **continue와 break 활용**
  - ✓ continue는 짝수단만 출력하는데 활용
  - ✓ break는 짝수단에서 x 2, x 4, x 6, x 8 까지만 출력하기 위해 사용

[예제 04] 별표(*) 출력

- **사용자로부터 입력 받은 숫자만큼 별표(*)를 출력**
  - ✓ 사용자가 0을 입력 시 다시 입력 받음
  - ✓ 사용자가 11 이상 입력 시 “그렇게 많은 별표를 출력할 수 없습니다.”라는 에러 메시지와 함께 프로그램을 종료

<프로그램 구현 힌트>

- **무한 반복문/중첩 반복문 사용**
- **continue와 break 활용**

Q & A