

Today

1. 지난 시간 실습 복습
2. 흐름제어 – 반복문1 (while)
3. 흐름제어 – 반복문3 (do-while)
4. 흐름제어 – 반복문2 (for)
5. break 와 continue 키워드

반복문 for

■ while문과 동일한 기능 / 다른 문법

- for 옆의 괄호안의 초기값을 이용해서 <조건식>을 판단함
- 조건식이 '참'이면 블록 내 수행문을 실행하고, 증감식을 적용함
- 조건식이 '거짓'이면 블록은 수행되지 않음

■ for문의 구조

```
for ( 초기값; <조건식>; 증감식 )  
    수행문;
```

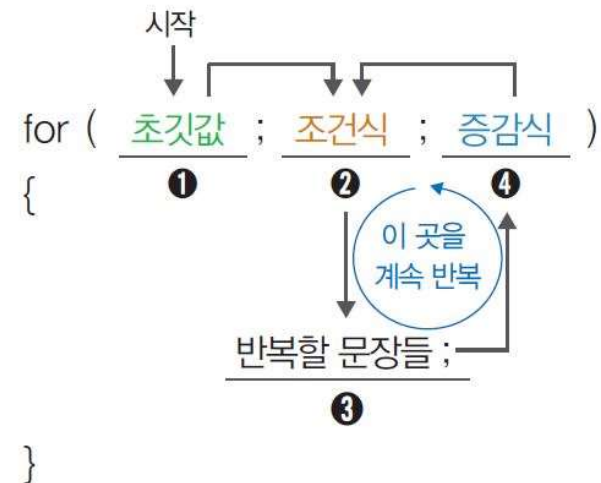


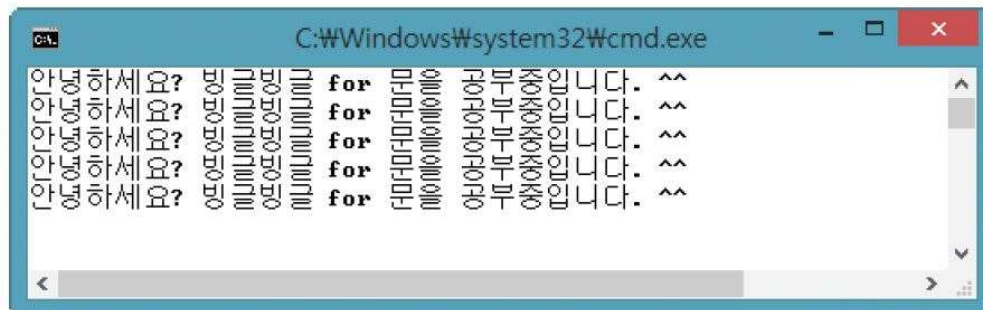
그림 6-1 for문의 기본 형식

- 괄호 안에 초기값, <조건식>, 증감식이 세미콜론(;)으로 구분됨
- 반복되는 순서 : ①, ②를 수행한 뒤 <조건식> 결과가 '참'이면, ③, ④, ②를 통해 반복할 문장들을 계속 수행함
- 중괄호({ }) 로 for 문으로 반복할 문장들을 블록으로 만듦

[6-2] 기본 for문 사용 예

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int i;
06
07     for(i=0; i<5 ; i++)
08     {
09         printf("안녕하세요? 빙글빙글 for 문을 공부중입니다.^^\n");
10     }
11 }
```

실행 결과 ▼



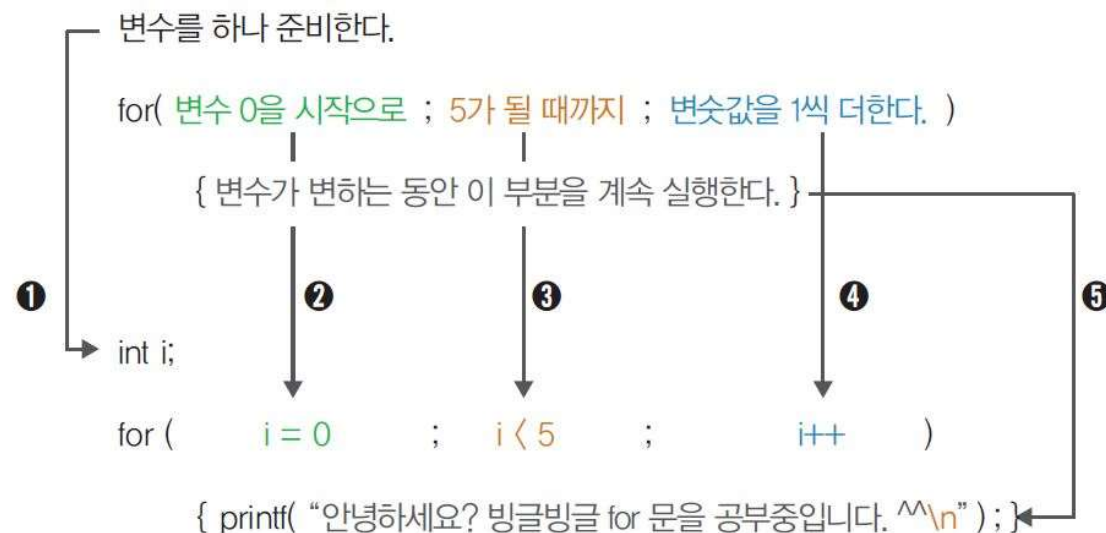
```
C:\Windows\system32\cmd.exe
안녕하세요? 빙글빙글 for 문을 공부중입니다. ^^
안녕하세요? 빙글빙글 for 문을 공부중입니다. ^^
안녕하세요? 빙글빙글 for 문을 공부중입니다. ^^
안녕하세요? 빙글빙글 for 문을 공부중입니다. ^^
안녕하세요? 빙글빙글 for 문을 공부중입니다. ^^
```

[6-2] 예제 설명

■ [기본 6-2]에서 사용한 for문(7~10행)의 기본 구조

- ❶ for문을 사용하려면 **무조건 변수를 하나** 준비함.
- ❷ 사용할 변수의 초기값이 꼭 필요함.
- ❸ 다섯 번 실행해야 하므로 ' $i \leq 5$ '라고 생각하기 쉽지만, ❷에서 초기값이 0부터 시작하므로 ' $i < 5$ '로 설정해야 0, 1, 2, 3, 4로 총 다섯 번 실행됨. 만약 ' $i \leq 5$ '라고 쓰면 여섯 번 실행됨
- ❹ ' $i++$ '는 ' $i=i+1$ '과 동일한 역할. 즉 i 값을 1 증가시킴
이 부분은 for문을 한 번 돌 때마다 수행함
- ❺ 실제로 반복되는 내용. printf문 한 줄만 들어있지만, 복잡한 프로그램일수록 많은 내용이 들어감

➔ 즉, for 문은 초기값을 한 번만 실행하고 나머지 부분이 반복되는 구조



for문 정리

❶ 초기식 실행
(= 초깃값 대입)

```
int i;  
for ( i = 0 ; i < 3 ; i++ )  
{  
    printf("안녕하세요? C입니다.\n");  
}
```

❷ 조건식 확인

```
int i;  
for ( i = 0 ; i < 3 ; i++ )  
{  
    printf("안녕하세요? C입니다.\n");  
}
```

조건이
거짓이면

❸ 반복문 탈출

❸ 반복할 문장 실행

```
int i;  
for ( i = 0 ; i < 3 ; i++ )  
{  
    printf("안녕하세요? C입니다.\n");  
}
```

조건이 참이면

❹ 증감식 실행

```
int i;  
for ( i = 0 ; i < 3 ; i++ )  
{  
    printf("안녕하세요? C입니다.\n");  
}
```

그림 6-3 for문이 반복되는 순서

단순 for문

- 반복 실행할 문장이 두 개 이상이면 반드시 블록 사용해야 함

```
int i;  
for ( i=0; i < 3; i++ )  
    printf("빙글빙글 for문입니다. ^^\\n");
```

==

```
int i;  
for ( i=0; i < 3; i++ )  
{  
    printf("빙글빙글 for문입니다. ^^\\n");  
}
```

- 다음 for문들의 실행 결과는?

```
int main()  
{  
    int i;  
    for (i = 1; i <= 4; i++)  
        printf("[%d] Hello.\\n", i);  
}
```

```
int main()  
{  
    int i;  
    for (i = 1; i < 10; i*=2)  
        printf("[%d] Hello.\\n", i);  
}
```

```
int main()  
{  
    int i;  
    for (i = 4; i > 0; i--)  
        printf("[%d] Hello.\\n", i);  
}
```

```
int main()  
{  
    int i;  
    for (i = 0; i < 10; i*=2)  
        printf("[%d] Hello.\\n", i);  
}
```

```
int main()  
{  
    int i;  
    for (i = 0; i < 7; i+=2)  
        printf("[%d] Hello.\\n", i);  
}
```

```
int main()  
{  
    int i;  
    for (i = 0; i%4 != 3; i++)  
        printf("[%d] Hello.\\n", i);  
}
```

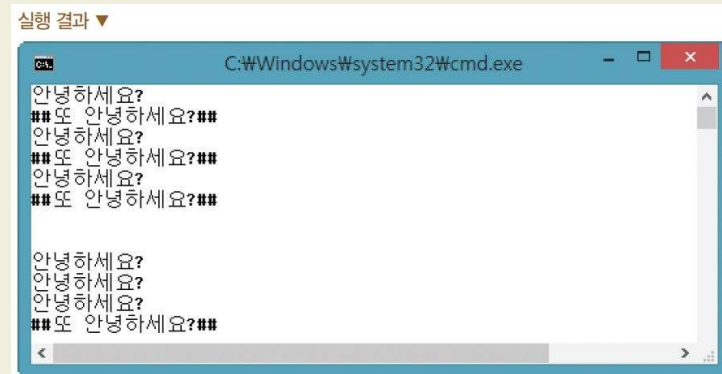
[6-3] for문과 블록 사용 예

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int i;
06     for ( i=0; i < 3; i++ )
07     {
08         printf("안녕하세요? \n");
09         printf("##또 안녕하세요?## \n");
10     }
11
12     printf("\n\n");
13
14     for ( i=0; i < 3; i++ )
15         printf("안녕하세요? \n");
16         printf("##또 안녕하세요?## \n");
17 }
```

----블록을 사용한 for문이다.

----블록을 사용하지 않은 for문이다.

실행 결과 ▼



```
C:\Windows\system32\cmd.exe
안녕하세요?
##또 안녕하세요?##
안녕하세요?
##또 안녕하세요?##
안녕하세요?
##또 안녕하세요?##

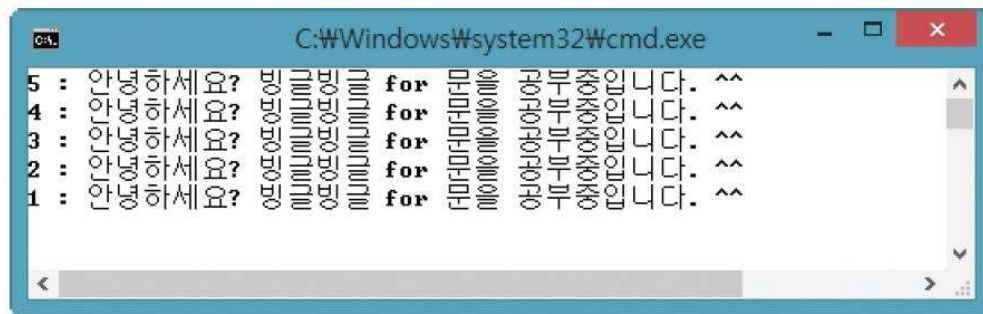
안녕하세요?
안녕하세요?
안녕하세요?
##또 안녕하세요?##
```

[6-4] for문 사용 예 ①

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int i;
06
07     for ( i=5; i> 0; i-- )  
08     {  
09         printf("%d : 안녕하세요? 빙글빙글 for 문을 공부중입니다. ^^\\n", i);  
10     }  
11 }
```

---초깃값, 조건식, 증감식을 수정한다.

실행 결과 ▼

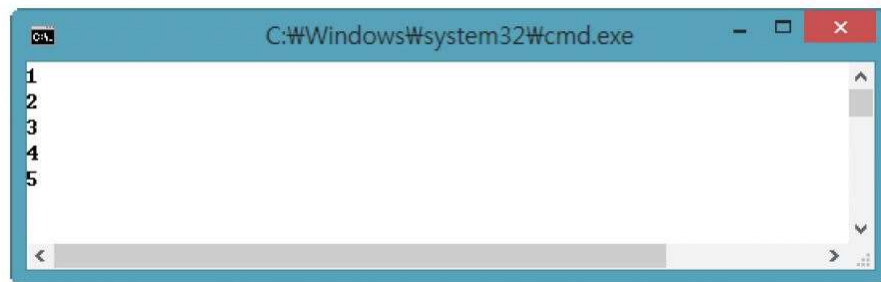


```
C:\Windows\system32\cmd.exe
5 : 안녕하세요? 빙글빙글 for 문을 공부중입니다. ^^
4 : 안녕하세요? 빙글빙글 for 문을 공부중입니다. ^^
3 : 안녕하세요? 빙글빙글 for 문을 공부중입니다. ^^
2 : 안녕하세요? 빙글빙글 for 문을 공부중입니다. ^^
1 : 안녕하세요? 빙글빙글 for 문을 공부중입니다. ^^
```


[6-5] for문 사용 예 ②

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int i;
06
07     for (i=1; i<=5; i++) ---i 값이 1부터 5까지 변경된다.
08     {
09         printf("%d \n", i);
10     }
11 }
```

실행 결과 ▼

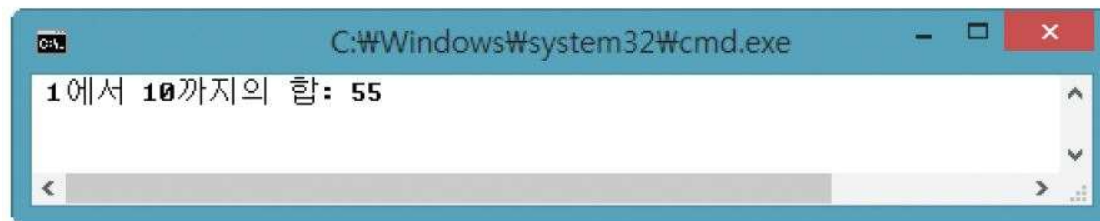


```
C:\Windows\system32\cmd.exe
1
2
3
4
5
```

[6-6] for문을 활용하지 않고 합계 구하기

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int hap;
06
07     hap = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10; ---hap에 1부터 10까지 더해서 입력한다.
08
09     printf("1에서 10까지의 합: %d \n", hap);
10 }
```

실행 결과 ▼



The screenshot shows a Windows Command Prompt window titled "C:\Windows\system32\cmd.exe". The command prompt displays the output of the program: "1에서 10까지의 합: 55". The window has a blue title bar and standard Windows window controls (minimize, maximize, close).

[실습] for문을 활용한 합계 구하기

- 소스가 아주 간단하고 결과도 잘 나왔지만, 만약 1000까지 더한다면?
“hap = 1 + 2 + 3 + ... + 1000”
- 이와 같이 반복적인 덧셈을 수행할 때는 for문을 활용해야 한다.

합계가 들어갈 변수 준비(hap)

1부터 10까지 변할 변수 준비(i)

```
for ( i가 1부터 시작했으므로; i가 10보다 작거나 같을 때까지; i가 1씩 증가 )  
{  
    hap 값에 i 값을 더함  
}
```

[기본 6-7] for문을 활용하여 합계 구하기 ①

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int hap;
06     int i;
07
08     for ( i=1; i<=10; i++ ) {
09         hap = hap + i;
10     }
11
12     printf(" 1에서 10까지의 합: %d \n", hap);
13 }
```

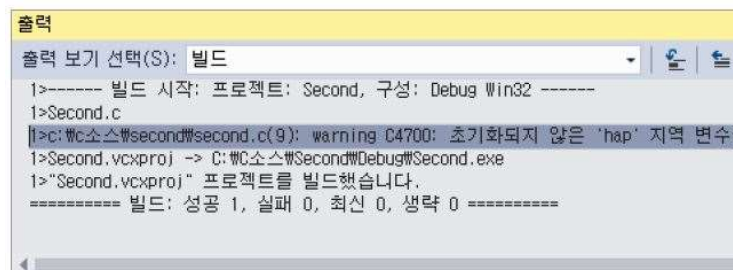
----합계를 누적할 변수를 선언한다

----1부터 10까지 변하는 변수를 선언한다.

----for문에 의해 1부터 10까지 열 번 반복된다.

-----hap 변수에 1부터 10까지 반복해서 누적된다.

실행 결과 ▼



```
출력
출력 보기 선택(S): 빌드
1>----- 빌드 시작: 프로젝트: Second, 구성: Debug Win32 -----
1>Second.c
1>c:\c소스\Second\Second.c(9): warning C4700: 초기화되지 않은 'hap' 지역 변수
1>Second.vcxproj -> C:\c소스\Second\Debug\Second.exe
1>"Second.vcxproj" 프로젝트를 빌드했습니다.
===== 빌드: 성공 1, 실패 0, 최신 0, 생략 0 =====
```

- 실행 결과 오류 발생함.
 - 변수 hap이 초기화되지 않았기 때문
- 쓰레기값의 이해
 - 변수 안에 이미 다른 값이 있는 상태로 연산을 수행함으로써 비정상적인 결과 값 출력
 - 프로그램 내에서 변수가 초기화되지 않은 것이 원인
 - 누적값을 표현하는 변수의 초기화를 수행함으로써 문제 해결

`int hap = 0;`

[기본 6-8] for문을 활용하여 합계 구하기 ②

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int hap = 0;      ---합계를 누적할 변수를 선언하고 0으로 초기화한다.
06     int i;
07
08     for ( i=1; i<=10; i++ ) {
09         hap += i;      ---hap 변수에 1부터 10까지 반복해서 누적한다.
10     }                hap = hap + i와 동일하다.
11
12     printf(" 1에서 10까지의 합: %d \n", hap);
13 }
```

실행 결과 ▼

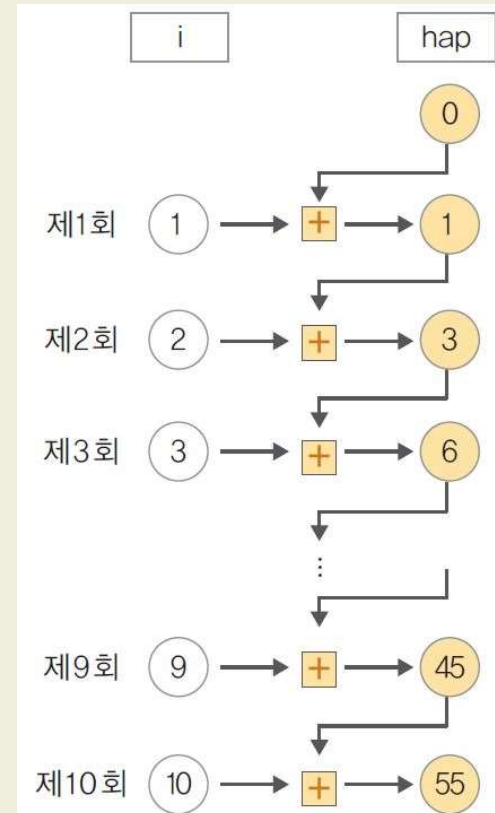
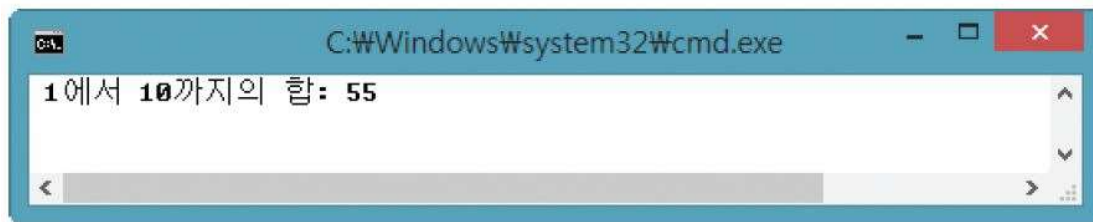
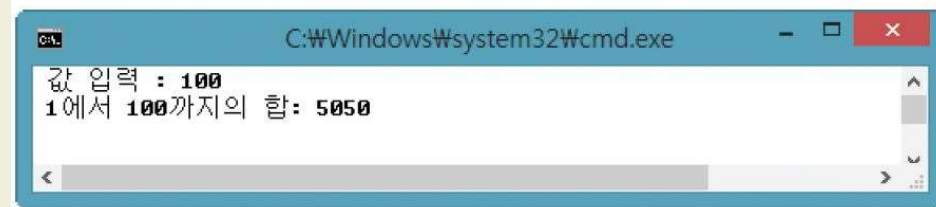


그림 6-4 변수 *i*와 *hap*의 변화

[6-10] 1에서 100 까지 합계 구하기

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int hap=0;      ---합계를 누적할 변수를 선언하고 0으로 초기화한다.
06     int i;          ---1씩 증가시킬 변수이다.
07     int num;        ---입력받을 최종값이다.
08
09     printf(" 값 입력 : ");
10     scanf("%d", &num); ---최종값을 입력한다.
11
12     for ( i=1; i <= num; i++ ) { ---1부터 최종값까지 1씩 증가시키며 반복한다.
13         hap = hap + i;
14     }
15
16     printf(" 1에서 %d까지의 합: %d \n", num, hap);
17 }
```

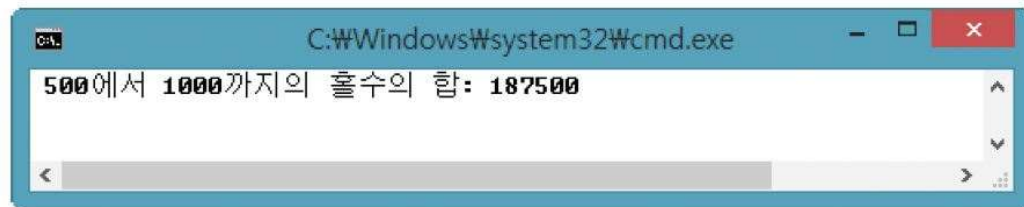
실행 결과 ▼



[6-9] 500에서 1000 까지 홀수 합계 구하기

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int hap = 0;
06     int i;
07
08     for ( ____①____ ; i<=1000 ; ____②____ ) { ---i를 501부터 2씩 증가시킨다.
09         hap = hap + i;
10     }
11
12     printf(" 500에서 1000까지의 홀수의 합: %d \n", hap);
13 }
```

실행 결과 ▼



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output displayed is "500에서 1000까지의 홀수의 합: 187500". The window has a blue title bar and standard Windows window controls (minimize, maximize, close).

[6-11] for문을 활용하여 합계 구하기 (응용)

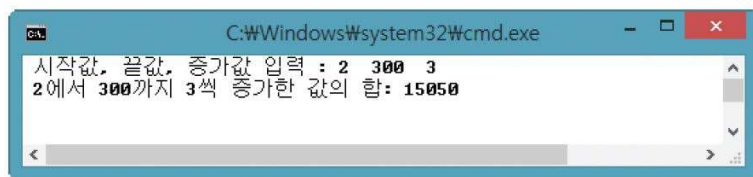
```
01 #include <stdio.h>
02
03 int main()
04 {
05     int hap=0;
06     int i;
07     int num1, num2, num3;
08
09     printf(" 시작값, 끝값, 증가값 입력 : ");
10     scanf("%d %d %d", &num1, &num2, &num3);
11
12     for ( ____①____ ) {
13         hap = hap + i;
14     }
15
16     printf(" %d에서 %d까지 %d씩 증가한 값의 합: %d \n", num1, num2, num3, hap);
17 }
```

---입력받을 변수 세 개를 선언한다.

---공백 문자로 구분해서 세 개의 수를 입력받는다.

---시작값은 num1, 최종값은 num2, 증가값은 num3이다.

실행 결과 ▼



```
C:\Windows\system32\cmd.exe
시작값, 끝값, 증가값 입력 : 2 300 3
2에서 300까지 3씩 증가한 값의 합: 15050
```


[6-12] for문을 사용한 구구단 프로그램

```
01 #include <stdio.h>
```

```
02
```

```
03 int main()
```

```
04 {
```

```
05     int i;
```

```
06     int dan;
```

----계산한 단을 입력받을 변수를 선언한다.

```
07
```

```
08     printf(" 몇 단 ? ");
```

```
09     scanf("%d", &dan);
```

----계산할 단을 입력받는다.

```
10
```

```
11     for ( i=1; i <=9; i++ ) {
```

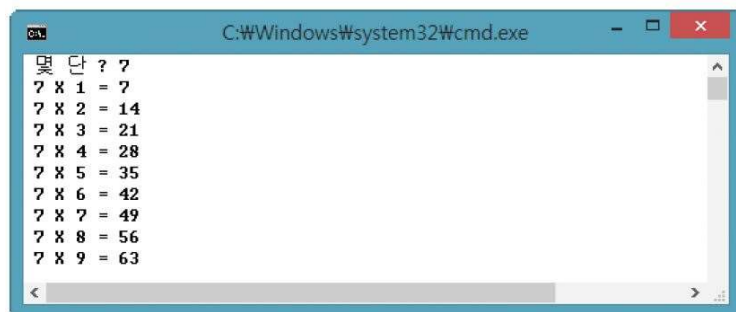
----1부터 9까지 반복하며 입력한 단을 출력한다.

```
12         printf(" %d X %d = %d \n", dan, i, dan*i);
```

```
13     }
```

```
14 }
```

실행 결과 ▼



```
C:\Windows\system32\cmd.exe

몇 단 ? 7
7 X 1 = 7
7 X 2 = 14
7 X 3 = 21
7 X 4 = 28
7 X 5 = 35
7 X 6 = 42
7 X 7 = 49
7 X 8 = 56
7 X 9 = 63
```

중첩 for 문

■ 중첩 for문의 개념

- for문 내부에 또 다른 for문이 들어 있는 형태
- 총 반복 횟수 = 바깥 for문 횟수 x 안쪽 for문 횟수

```
int main()
{
    int i, j;

    for (i = 0; i < 3; i++)
        for (j = 0; j < 2; j++)
            printf("[i=%d][j=%d]Hello.\n", i, j);
}
```

```
[i=0][j=0]Hello.
[i=0][j=1]Hello.
[i=1][j=0]Hello.
[i=1][j=1]Hello.
[i=2][j=0]Hello.
[i=2][j=1]Hello.
```



그림 6-5 중첩 for문의 동작 개념

```
for (i = 0; i < 3; i++)
{
    for (j = 0; j < 2; j++)
    {
        printf("[i=%d][j=%d]Hello.\n", i, j);
    }
}
```

[block을 활용한 명시적 기술]

중첩 for 문의 작동 방식

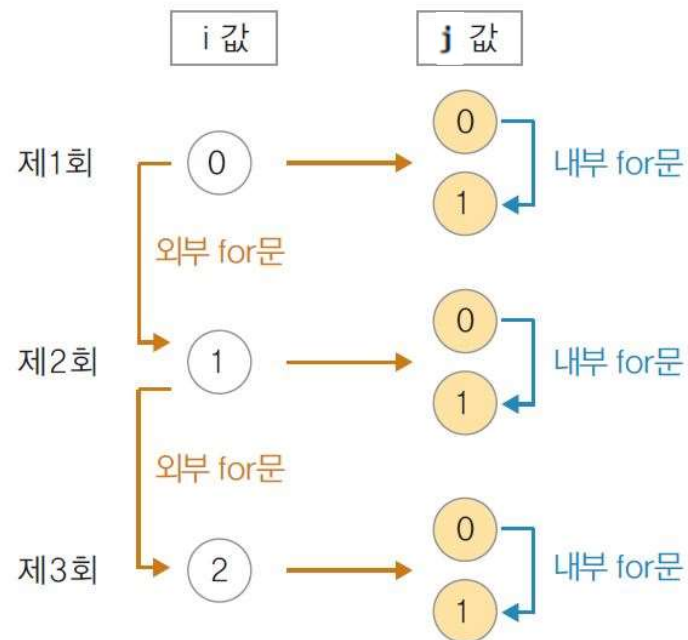


그림 6-7 중첩 for문에서 i 값과 j 값의 변화

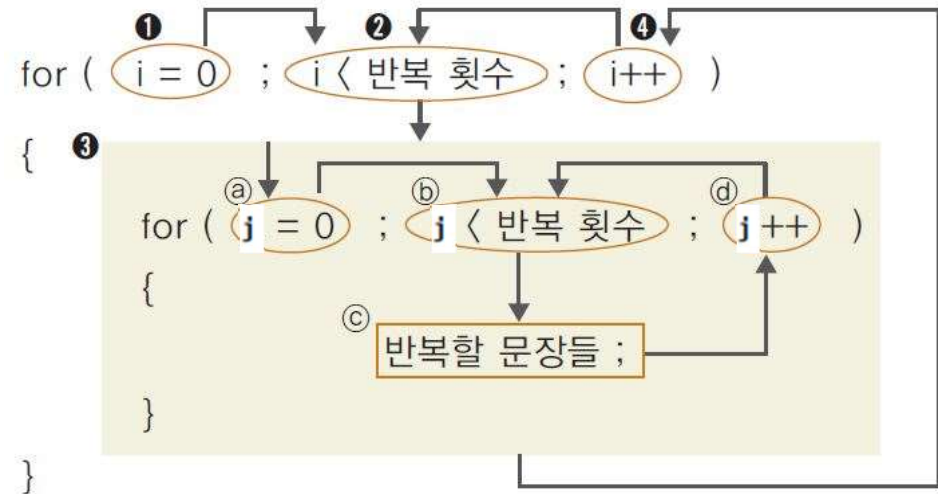


그림 6-6 중첩 for문의 작동 방식

Q) 다음 for문들의 실행결과는?

```
int main()
{
    int i, j;

    for (i = 0; i < 3; i++)
        for (j = i; j < 2; j++)
            printf("[i=%d][j=%d]Hello.\n", i, j);
}
```

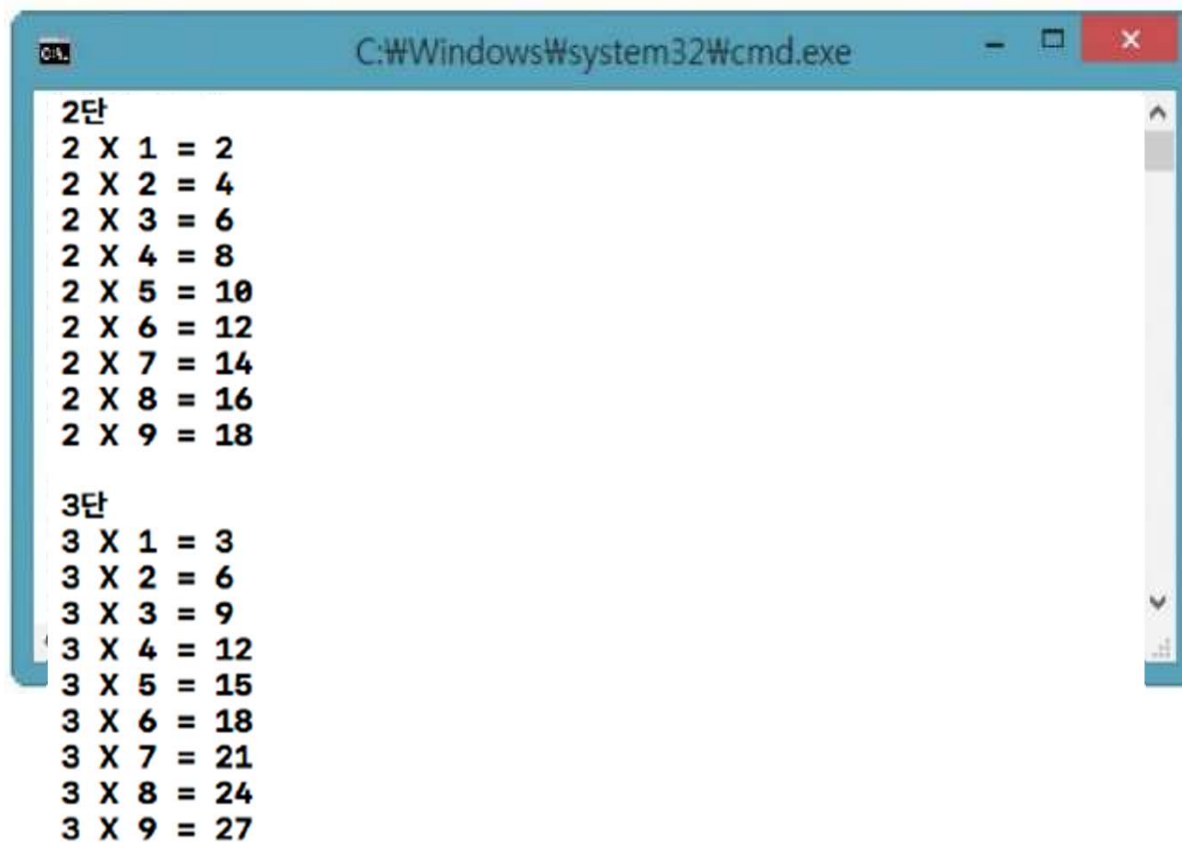
```
int main()
{
    int i, j;

    for (i = 0; i < 3; i++)
        for (j = 0; j < i; j++)
            printf("[i=%d][j=%d]Hello.\n", i, j);
}
```

[실습] 중첩 for 문 활용한 구구단 출력

- 다음 결과와 같이 2단~9단까지 구구단을 출력하기

실행 결과 ▼



```
C:\Windows\system32\cmd.exe

2단
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
2 X 7 = 14
2 X 8 = 16
2 X 9 = 18

3단
3 X 1 = 3
3 X 2 = 6
3 X 3 = 9
3 X 4 = 12
3 X 5 = 15
3 X 6 = 18
3 X 7 = 21
3 X 8 = 24
3 X 9 = 27
```

다양한 for 문의 형태

- 여러 개의 초깃값과 증감식을 사용하는 for문
 - 초깃값이 하나일 필요 없음
 - 초깃값이 여러 개일 때는 콤마(,)로 구분
 - 증감식도 하나 이상 사용 가능

```
for ( 초깃값 1, 초깃값 2; 조건식; 증감식 1, 증감식 2 )
```

```
for ( i = 1, k = 1; i <= 9; i++, k++ )  
    printf ( " %d X %d = %d \n", i, k, i*k );
```

---초깃값과 증감식이 두 개이다.

다양한 for 문의 형태

- 초기값과 증감식이 없는 for문
 - 세 가지 형식 모두 결과가 같음

❶ 기본 형식

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

❷ 초기값 빼기

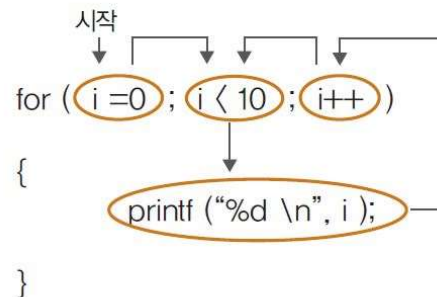
```
int i;
i = 0;
for ( _____ ; i < 10 ; i ++ )
{
    printf ("%d \n", i) ;
}
```

❸ 초기값과 증감식 빼기

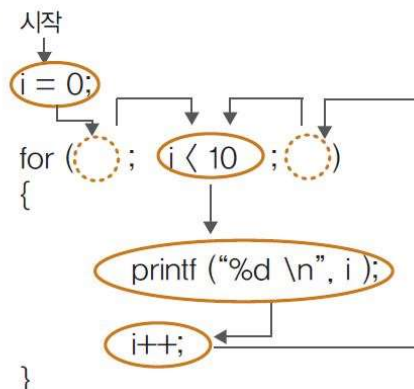
```
int i;
i = 0;
for ( _____ ; i < 10 ; _____ )
{
    printf ("%d \n", i) ;
    i ++ ;
}
```

- 초기식이 없더라도 그 자리는 반드시 세미콜론(;)으로 구분해야 함

❶



❷



무한 반복 for 문

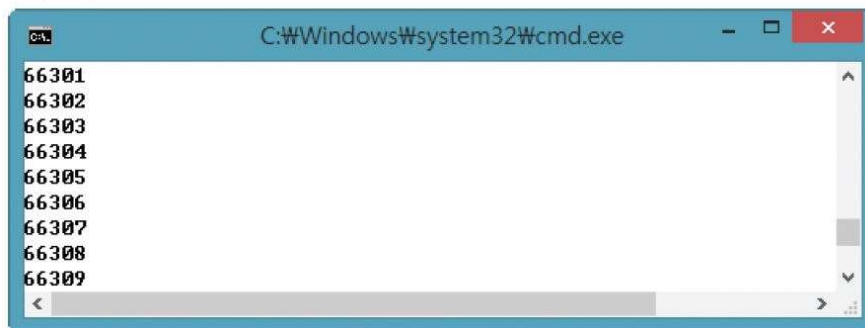
- for문에서 초깃값, 조건식, 증감식을 모두 지운 형태
- 사용자가 [Ctrl]+[C]를 눌러야 중단됨

[6-17] 다양한 for 문의 활용 예 ②

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int i;
06     i = 0;
07     for (;;)
08     {
09         printf ("%d \n", i);
10         i ++;
11     }
12 }
```

---초깃값, 조건식, 증감식이 아무것도 없다.

실행 결과 ▼

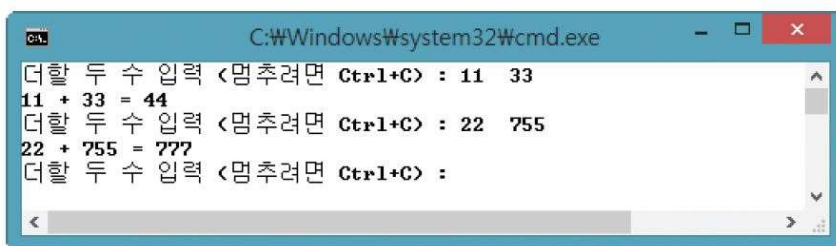


The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window displays the output of the program, which consists of the numbers 66301 through 66309, each on a new line. The window has a blue title bar and a scroll bar on the right side.

[6-18] 다양한 for 문의 활용 예 ③

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a, b;
06
07     __①__ ( ; ; ) ----무한 루프가 발생한다.
08     {
09         printf ("더할 두 수 입력 (멈추려면 Ctrl+C) : ");
10         scanf("%d %d", &a, &b); ----두 값을 입력받는다.
11
12         printf("%d + %d = %d \n", a, b, a+b); ----더하기 결과를 출력한다.
13     }
14 }
```

실행 결과 ▼

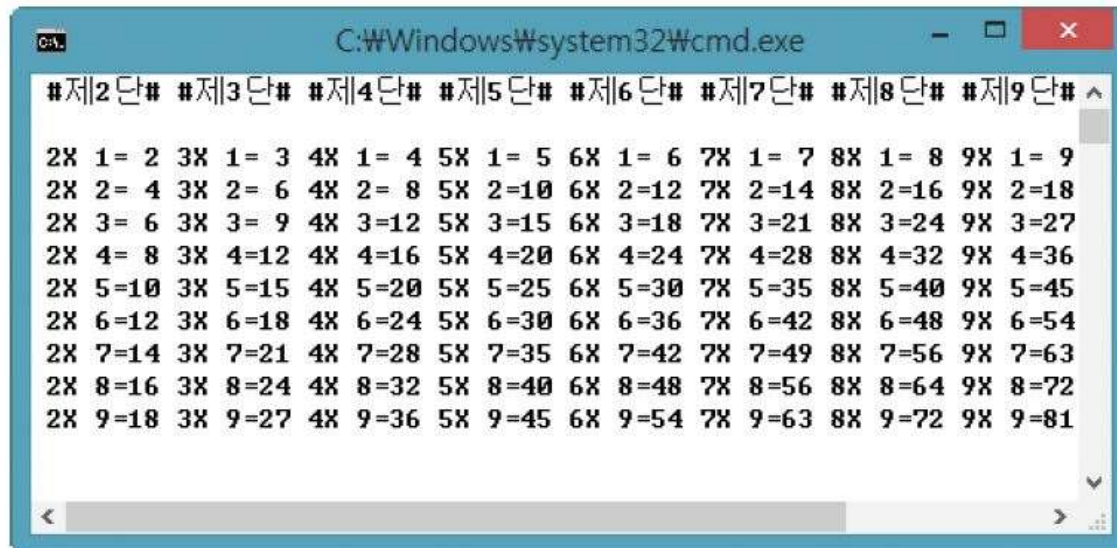


```
C:\Windows\system32\cmd.exe
더할 두 수 입력 <멈추려면 Ctrl+C> : 11 33
11 + 33 = 44
더할 두 수 입력 <멈추려면 Ctrl+C> : 22 755
22 + 755 = 777
더할 두 수 입력 <멈추려면 Ctrl+C> :
```

[실습1] 구구단 출력 프로그램

예제 설명 중첩 for문을 사용하여 제목과 구구단을 출력하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe

#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
```

[실습2] 아스키코드표 출력 프로그램

예제 설명 for문과 if문을 사용하여 아스키코드의 0~127을 10진수, 16진수, 문자로 출력하는 프로그램이다.

실행 결과

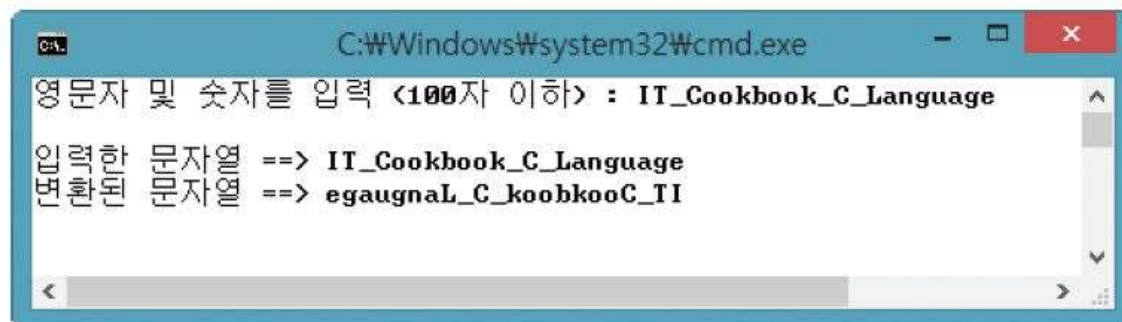
10진수	16진수	문자
0	0	
1	1	r
2	2	l
3	3	.
4	4	.
5	5	.
6	6	.
7	7	.
8	8	.
9	9	.
10	a	.
11	b	.
12	c	.
13	d	.
14	e	.
15	f	.

10진수	16진수	문자
16	10	†
17	11	‡

[실습3] 입력한 문자를 반대 순서로 출력

예제 설명 입력된 영문자나 숫자를 for문을 사용하여 반대 순서로 출력하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe
영문자 및 숫자를 입력 <100자 이하> : IT_Cookbook_C_Language
입력한 문자열 ==> IT_Cookbook_C_Language
변환된 문자열 ==> egaugnaL_C_koobkooC_TI
```

Summary

1. for문

- ❶ for문은 반복할 문장을 원하는 만큼 반복함
- ❷ for문의 형식

```
for ( 초깃값 ; 조건식 ; 증감식 )  
{  
    반복할 문장들;  
}
```

2. 중첩 for문

- ❶ for문 안에 또 다른 for문이 있는 형태
- ❷ 중첩 for문의 형식

```
for ( i = 0 ; i < 반복 횟수 ; i ++ )  
{  
    for ( k = 0 ; k < 반복 횟수 ; k ++ )  
    {  
        반복할 문장들;  
    }  
}
```

3. for문의 다른 형태

- ❶ for문의 초깃값, 조건식, 증감식은 하나 이상 생략 가능함
- ❷ 초깃값, 조건식, 증감식을 모두 지운 `for(;;)` 문장은 문법적으로 아무 문제가 없으며 무한 루프 역할을 하지만 사용은 지양함

Today

1. 지난 시간 실습 복습
2. 흐름제어 – 반복문1 (while)
3. 흐름제어 – 반복문3 (do-while)
4. 흐름제어 – 반복문2 (for)
5. break 와 continue 키워드

break 키워드

- 반복문 또는 조건문을 미리 종료하고 싶을 때가 있음
 - if, while, do-while, for
- **break;** 가장 가까운 **반복문을 빠져나옴**
- switch 문에서는 해당 위치에서 종료시킴

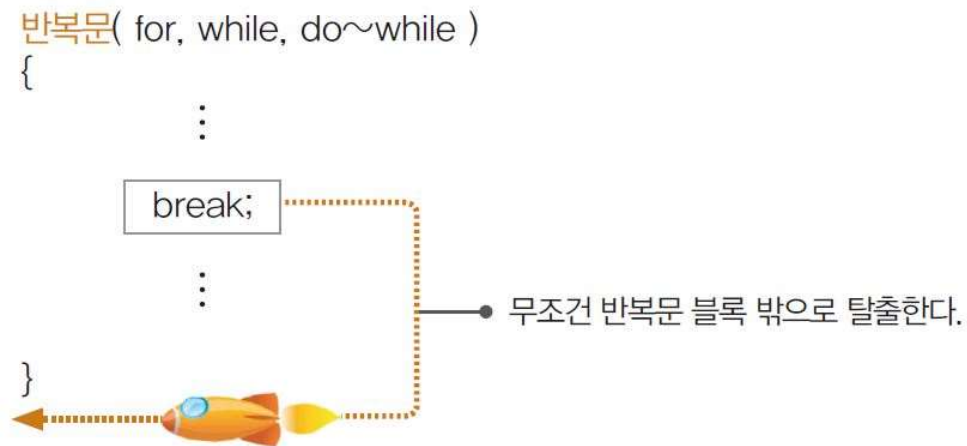


그림 7-5 break문의 작동

continue 키워드

- 반복문을 건너뛰고 싶을 때
- **continue;** 가까운 반복문의 나머지 부분을 건너뛰고, <조건문>으로 가서 **실행** 하도록 함

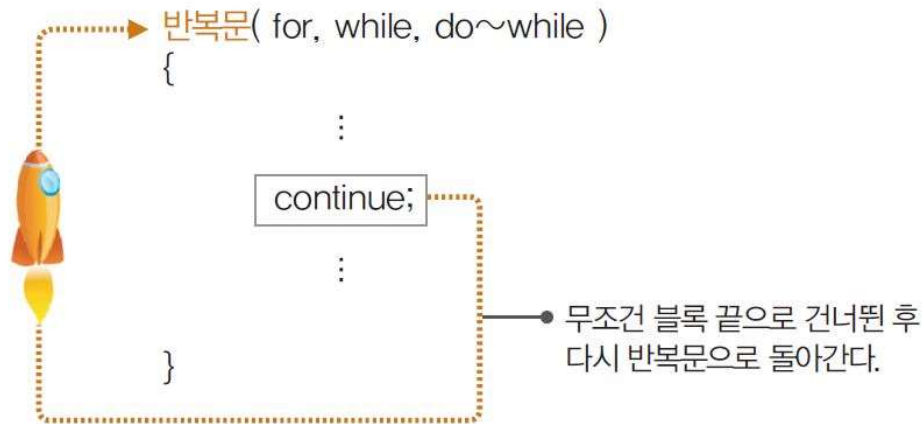
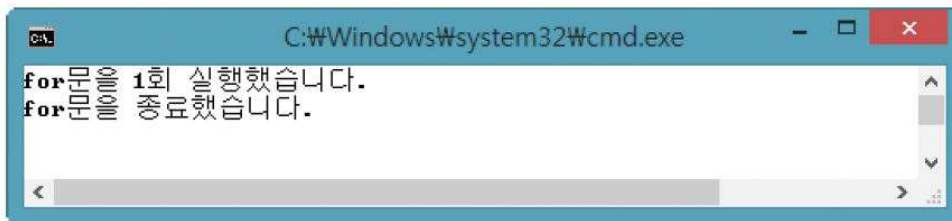


그림 7-6 continue문의 작동

[7-7] break 문 사용 예 ①

```
01 #include <stdio.h>
02
03 int 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 ▼



```
C:\Windows\system32\cmd.exe
for문을 1회 실행했습니다.
for문을 종료했습니다.
```

[7-8] break 문 사용 예 ②

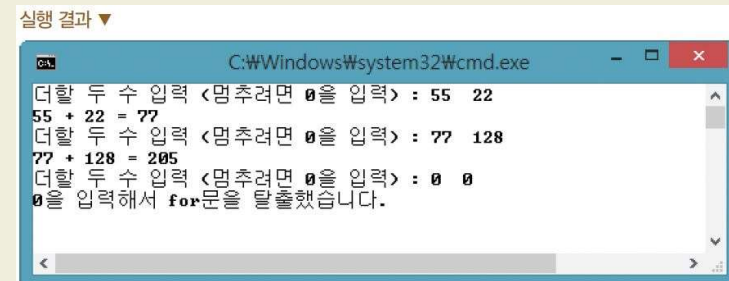
```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a, b;
06
07     while ( 1 )
08     {
09         printf ("더할 두 수 입력 (멈추려면 0을 입력) : ");
10         scanf("%d %d", &a, &b);
11
12         if (a == 0)
13             break;
14
15         printf("%d + %d = %d \n", a, b, a+b);
16     }
17
18     printf("0을 입력해서 for문을 탈출했습니다.\n");
19 }
```

----무한 루프를 만드는 코드이다.

-----2개의 수를 입력받는다.

-----첫 번째 입력값이 0이면 무조건 while문을 빠져 나간다.

실행 결과 ▼

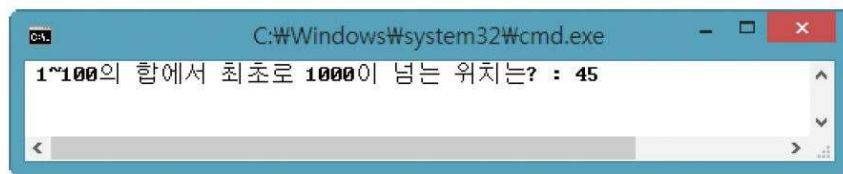


```
C:\Windows\system32\cmd.exe
더할 두 수 입력 <멈추려면 0을 입력> : 55 22
55 + 22 = 77
더할 두 수 입력 <멈추려면 0을 입력> : 77 128
77 + 128 = 205
더할 두 수 입력 <멈추려면 0을 입력> : 0 0
0을 입력해서 for문을 탈출했습니다.
```

[7-9] break 문 사용 예 ③

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int hap = 0;
06     int i;
07
08     for ( i=1; i<=100; i++ )      ---i값이 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 ▼



```
C:\Windows\system32\cmd.exe
1~100의 합에서 최초로 1000이 넘는 위치는? : 45
```

0001 => hap 1 1000

[7-10] continue 문 사용 예

```
01 #include <stdio.h>
02
03 int 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 }
```

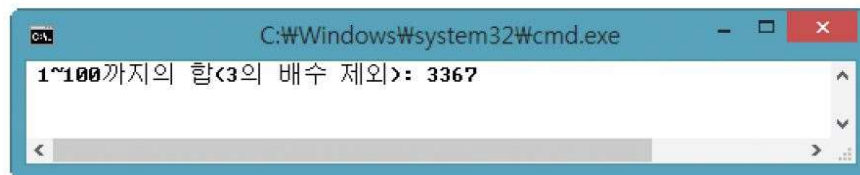
---i 값이 1부터 100까지 100회 반복한다.

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

-----3의 배수가 아닌 i 값이 누적된다.

---누적된 값을 출력한다.

실행 결과 ▼



기타 제어문 - goto

- 지정한 위치로 이동하는 goto문
 - 지정된 레이블로 건너뛰게 하는 명령문
 - 프로그램의 흐름을 복잡하게 만드는 단점이 있음



그림 7-7 goto문의 작동

[7-11] goto 문 사용 예

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int hap = 0;
06     int i;
07
08     for( i=1; i<=100; i++ )
09     {
10         hap += i;
11
12         if (hap > 2000)
13             goto mygoto;
14     }
15
16 mygoto:
17     printf ("1부터 %d까지 합하면 2000이 넘어요.\n", i);
18 }
```

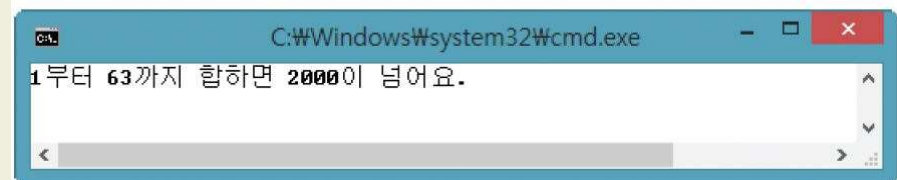
--- i 값이 1부터 100까지 100회 반복한다.

-----합계를 누적한다.

-----누적된 값이 200을 넘으면 mygoto:로
무조건 이동한다.

---goto문이 이동할 mygoto 레이블이다.

실행 결과 ▼



기타 제어문 - return

- 현재 함수를 불렀던 곳으로 돌아가는 return문
 - 현재 실행하는 함수를 끝내고 해당 함수를 호출한 곳으로 돌아가게 함
 - return문을 만나면 프로그램이 종료되는 효과

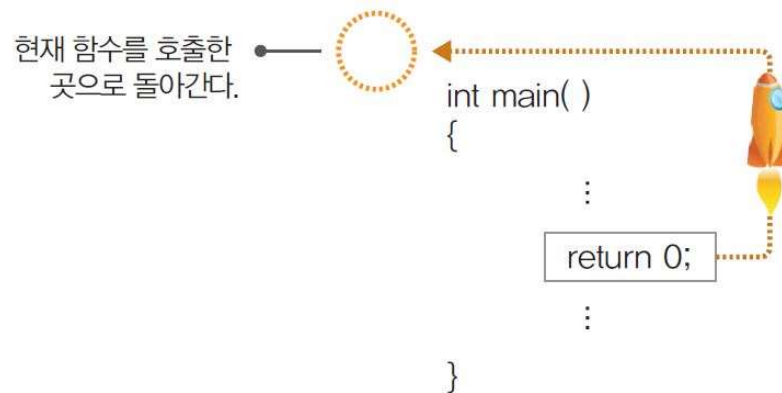


그림 7-8 return문의 작동

[7-12] return 문 사용 예

```
01 #include <stdio.h>
```

```
02
```

```
03 int 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 0;
```

----현재 함수를 호출한 곳으로 되돌린다.

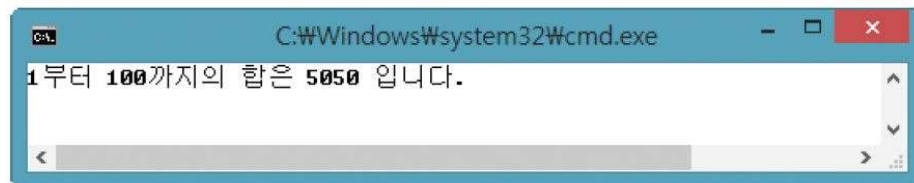
```
13
```

```
14     printf("프로그램의 끝입니다.");
```

----한 번도 실행되지 않는다

```
15 }
```

실행 결과 ▼

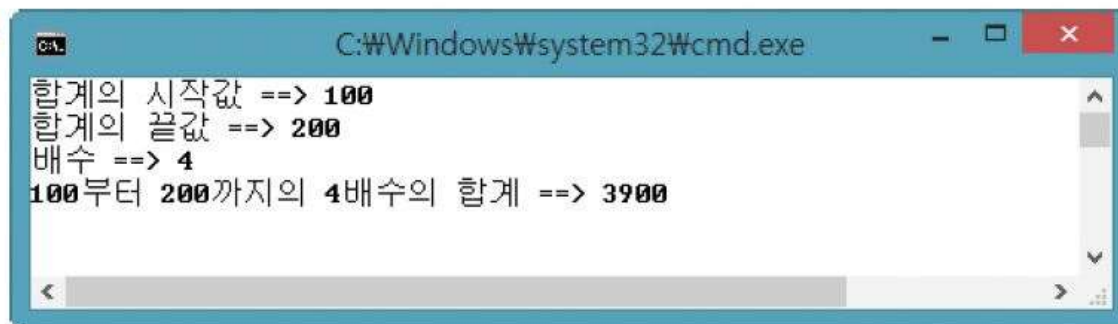


The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed as "1부터 100까지의 합은 5050 입니다." (The sum from 1 to 100 is 5050). The window has a blue title bar and a white background with a scrollbar on the right.

[실습1] 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

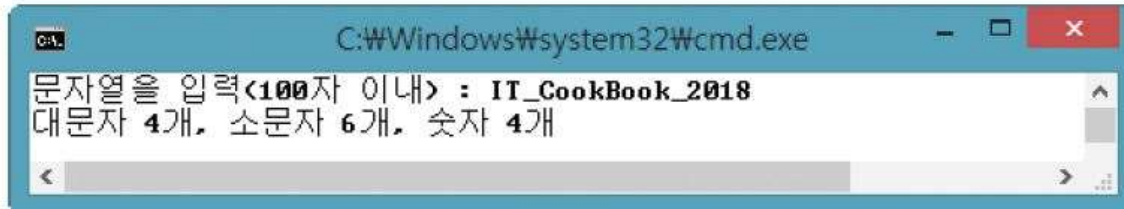


```
C:\Windows\system32\cmd.exe
합계의 시작값 ==> 100
합계의 끝값 ==> 200
배수 ==> 4
100부터 200까지의 4배수의 합계 ==> 3900
```

[실습2] 입력한 문자열의 종류 구분

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

실행 결과

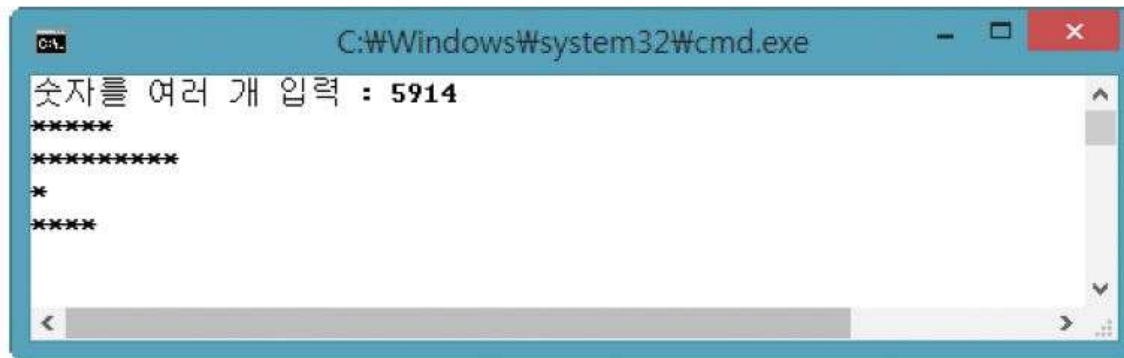


```
C:\Windows\system32\cmd.exe
문자열을 입력<100자 이내> : IT_CookBook_2018
대문자 4개, 소문자 6개, 숫자 4개
```

[실습3] 입력된 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



```
C:\Windows\system32\cmd.exe
숫자를 여러 개 입력 : 5914
*****
*****
*
****
```

Summary

1. `break`문을 만나면 현재의 반복문을 무조건 탈출함
2. `continue`문을 만나면 무조건 블록의 끝으로 이동한 후 다시 반복문의 처음으로 돌아감
3. `goto` 문은 사용자가 원하는 곳의 문장을 실행하도록 함
4. `return`문을 만나면 현재 함수를 호출한 곳으로 돌아감
 `main()` 함수에서 `return`문을 만나면 프로그램을 종료하는 효과