
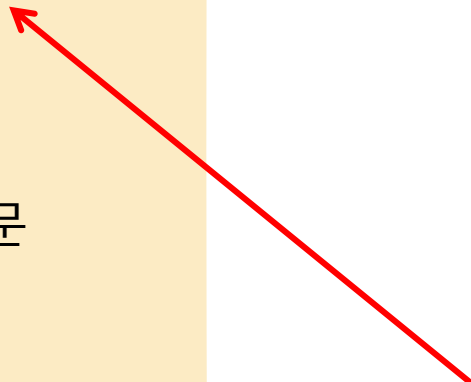


제 7장 반복문

이번 장에서 학습할 내용

- 
- 반복의 개념 이해
 - while 반복문
 - do-while 반복문
 - for 반복문
 - break와 continue문
- 

반복 구조는 일련의 처리를 반복할 수 있게 한다. 반복의 개념을 먼저 이해하고 C에서 제공되는 3가지의 반복 구조에 대하여 학습한다.



반복

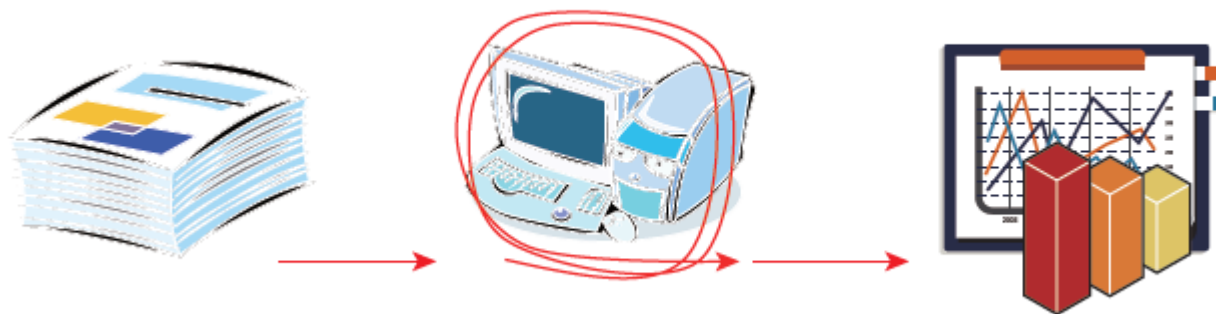
- 인간은 반복을 싫어하지만 프로그램에서는 반복적인 작업들이 반드시 필요하다.
- 반복(iteration)은 같은 처리 과정을 여러 번 되풀이하는 것이다



반복은 왜 필요한가?

Q) 반복 구조는 왜 필요한가?

A) 같은 처리 과정을 되풀이하는 것이 필요하기 때문이다. 학생 30명의 평균 성적을 구하려면 같은 과정을 30번 반복하여야 한다.



왜 반복이 중요한가?

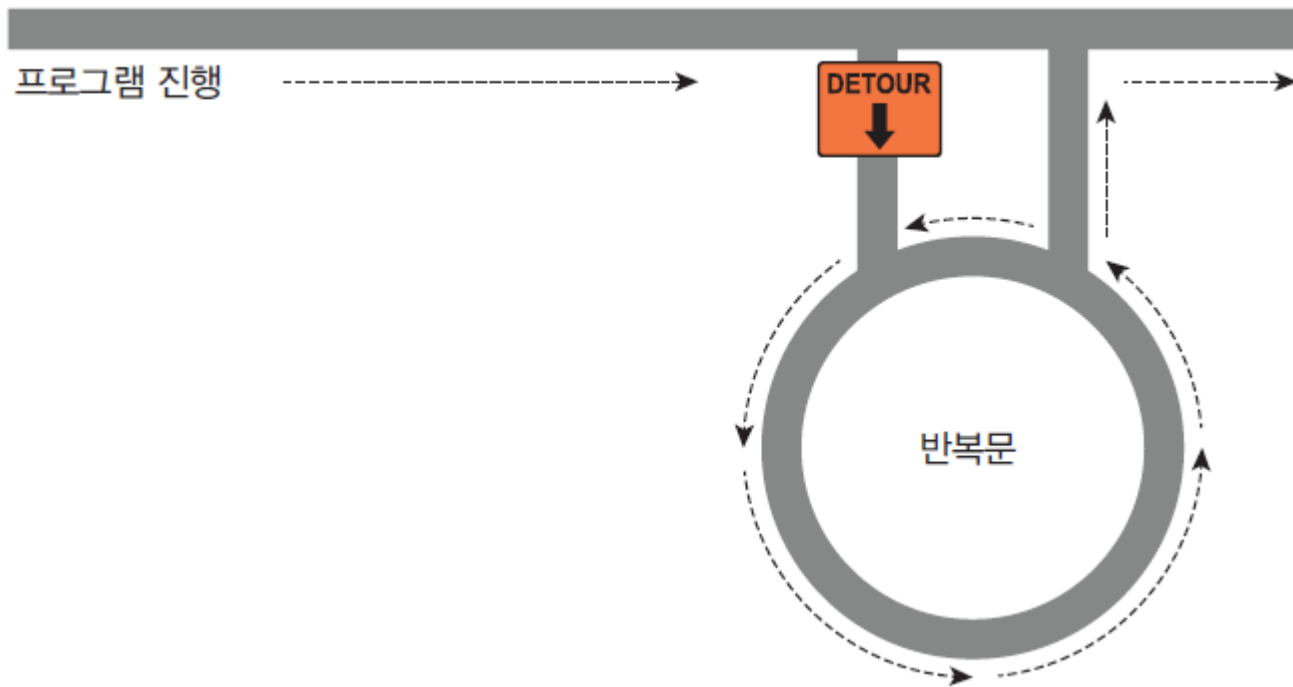
```
printf("Hello World! \n");  
printf("Hello World! \n");  
printf("Hello World! \n");  
printf("Hello World! \n");  
printf("Hello World! \n");
```



```
for (i = 0; i < 5; i++)  
    printf("Hello World! \n");
```

반복 구조

- 어떤 조건이 만족될 때까지 루프를 도는 구조



반복문의 종류



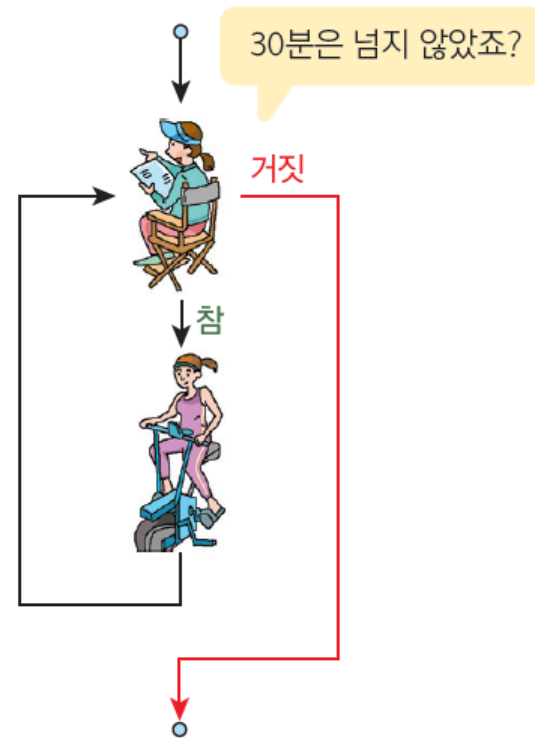
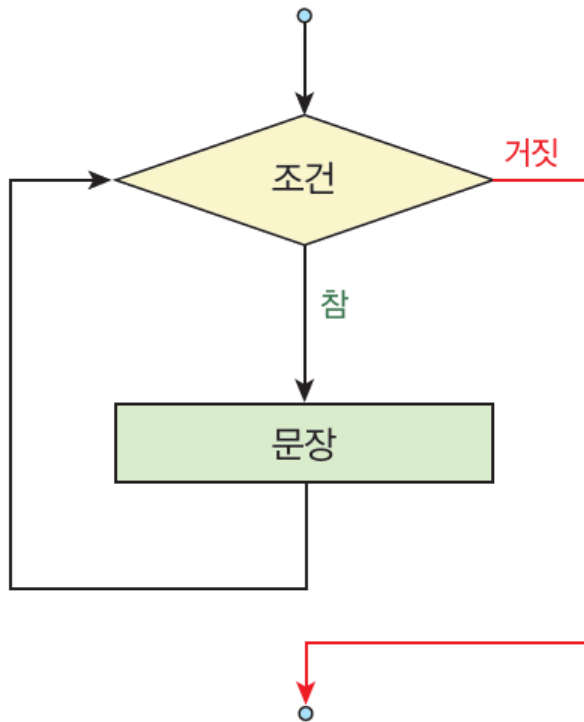
while 루프



for 루프

while 문

- 주어진 조건이 만족되는 동안 문장들을 반복 실행한다.



while 문

Syntax

while 문

예

```
while( i < 10 ) {  
    printf("Hello World!\n");  
    i++;  
}
```

조건식

조건식이 참이면 문장을 반복 실행한다.

예제

```
#include <stdio.h>
int main(void)
{
    int i = 0;
    while( i < 5 )
    {
        printf("Hello World! \n");
        i++;
    }

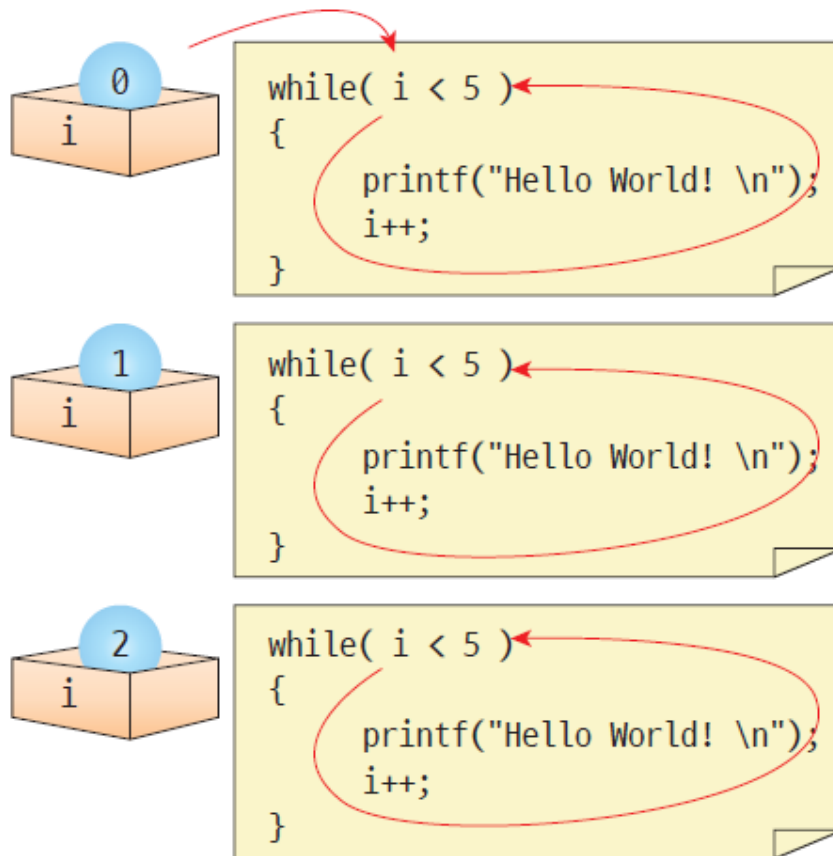
    return 0;
}
```

반복 조건

반복 내용

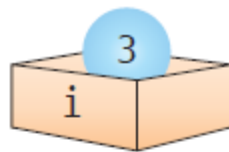
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!

while 문의 실행 과정

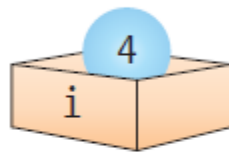


반복횟수	i의 값	(i<5)	반복여부
#1	0	참	반복
#2	1	참	반복
#3	2	참	반복
#4	3	참	반복
#5	4	참	반복
#6	5	거짓	중지

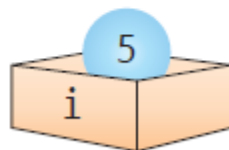
while 문의 실행 과정



```
while( i < 5 )  
{  
    printf("Hello World! \n");  
    i++;  
}
```



```
while( i < 5 )  
{  
    printf("Hello World! \n");  
    i++;  
}
```



```
while( i < 5 )  
{  
    printf("Hello World! \n");  
    i++;  
}
```

조건식이 거짓이 되어 반복종단

반복횟수	i의 값	(i<5)	반복여부
#1	0	참	반복
#2	1	참	반복
#3	2	참	반복
#4	3	참	반복
#5	4	참	반복
#6	5	거짓	중지

예제 #1

```
// while 문을 이용한 구구단 출력 프로그램
#include <stdio.h>

int main(void)
{
    int n;
    int i = 1;

    printf("출력하고 싶은 단: ");
    scanf("%d", &n);

    while (i <= 9)
    {
        printf("%d*%d = %d \n", n, i, n*i);
        i++;
    }

    return 0;
}
```

출력하고 싶은 단을 입력하시오: 9

9*1 = 9

9*2 = 18

9*3 = 27

...

9*9 = 81

예제 #2

```
// while 문을 이용한 제곱값 출력 프로그램
#include <stdio.h>
```

```
int main(void)
{
    int n;

    printf("=====\n");
    printf("  n      n의 제곱 \n");
    printf("=====\n");

    n = 1;
    while (n <= 10)
    {
        printf("%5d   %5d\n", n, n*n);
        n++;
    }

    return 0;
}
```

```
=====
n      n의 제곱
=====
1      1
2      4
3      9
4     16
5     25
6     36
7     49
8     64
9     81
10    100
```

예제 #3



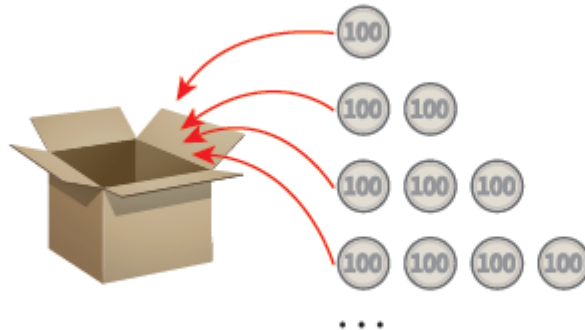
- 1부터 n 까지의 합 계산하는 프로그램

정수를 입력하시오: 3
1부터 3까지의 합은 6입니다

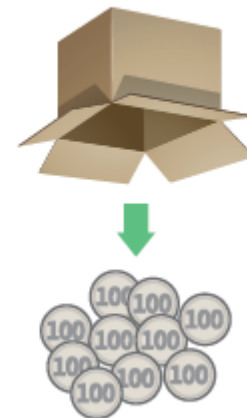
① 빈통을 준비한다.



② 통에 1부터 n 까지를 넣는다.



③ 통에 들어있는 동전의 개수를 출력한다.



예제 #3

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

```
int main(void)
```

```
{
```

```
    int i, n, sum;                // 변수 선언
```

```
    printf("정수를 입력하시오:"); // 입력 안내 메시지 출력
```

```
    scanf("%d", &n);              // 정수값 입력
```

```
    i = 1;                        // 변수 초기화
```

```
    sum = 0;
```

```
    while(i <= n)
```

```
    {
```

```
        sum += i;                // sum = sum + i;와 같다.
```

```
        i++;                     // i = i + 1과 같다.
```

```
    }
```

```
    printf("1부터 %d까지의 합은 %d입니다\n", n, sum);
```

```
    return 0;
```

```
}
```

정수를 입력하시오: 3
1부터 3까지의 합은 6입니다

예제 #5

Practice

- 이번에는 사용자가 입력하는 5개의 값을 합하여 그 결과를 출력하여 보자.

값을 입력하시오: 10
값을 입력하시오: 20
값을 입력하시오: 30
값을 입력하시오: 40
값을 입력하시오: 50
합계는 150입니다.

예제 #5

```
// while 문을 이용한 합계 프로그램
#include <stdio.h>

int main(void)
{
    int i, n, sum;

    i = 0;                // 변수 초기화
    sum = 0;              // 변수 초기화
    while (i < 5)
    {
        printf("값을 입력하시오: ");
        scanf("%d", &n);
        sum = sum + n;    // sum += n;과 같다.
        i++;
    }
    printf("합계는 %d입니다.\n", sum);

    return 0;
}
```

값을 입력하시오: 10
값을 입력하시오: 20
값을 입력하시오: 30
값을 입력하시오: 40
값을 입력하시오: 50
합계는 150입니다.

if 문과 while 문의 비교

```
if( 조건 )  
{  
  ...  
  ...  
}
```

조건이 만족되면
한번만 실행
된다.

```
while( 조건 )  
{  
  ...  
  ...  
}
```

조건이 만족되면
여러 번 반복 실행
된다.

while 문에서 주의할 점

```
int i = 1;
while(i < 10)
{
    printf("반복중입니다\n");
    i--;
}
```

변수가 증가 아니라 감소

```
int i = 0;
while(i < 3)
{
    printf("반복중입니다\n");
    i++;
}
```

반복 루프에 포함되어
있지 않다.

참과 거짓

```
#include <stdio.h>
int main(void)
{
    int i = 3;
    while (i)
    {
        printf("%d은 참입니다.", i);
        i--;
    }
    printf("%d은 거짓입니다.", i);
}
```

3은 참입니다.
2은 참입니다.
1은 참입니다.
0은 거짓입니다.

관습적인 형태

```
while(i != 0)
{
    ...
}
```



```
while( i )
{
    ...
}
```

주의



오류 주의

만약 while의 조건식 끝에 세미콜론(;)을 쓰면 NULL 문장만 반복된다. 세미콜론만 존재하는 문장을 NULL 문장이라고 한다.

```
while (i<10) ;
```

하나의 문장으로 취급되어서 이것만 반복된다.

```
i++;
```

반복되지 않는다.

```
while(i = 2)
```

```
{
```

```
...
```

```
}
```

수식의 값이 2이므로 항상 참이 되어서 무한 루프

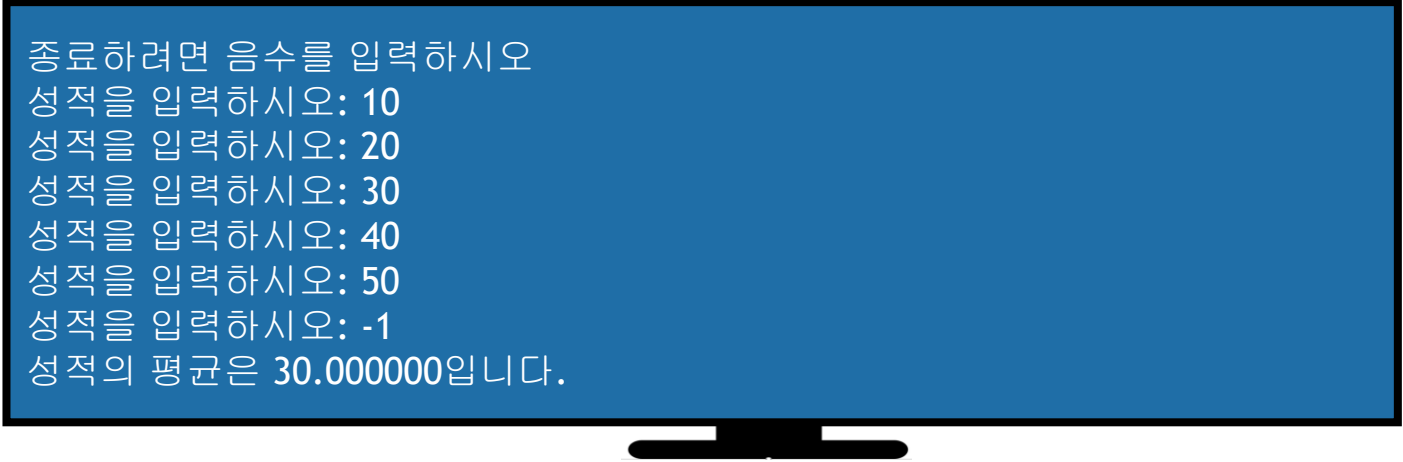
센티널(보초값의 이용)

- 센티널(sentinel): 입력되는 데이터의 끝을 알리는 특수한 값



성적의 평균을 계산하는 문제

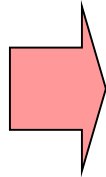
- 사용자로부터 임의의 개수의 성적을 받아서 평균을 계산한 후에 출력하는 프로그램을 작성하여 보자.
- -1을 보조값으로 사용한다.



```
종료하려면 음수를 입력하시오
성적을 입력하시오: 10
성적을 입력하시오: 20
성적을 입력하시오: 30
성적을 입력하시오: 40
성적을 입력하시오: 50
성적을 입력하시오: -1
성적의 평균은 30.000000입니다.
```

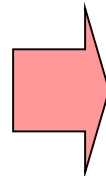
성적들의 평균을 구하는 문제

성적의 평균을 구한다.



1. 필요한 변수들을 초기화한다.
2. 성적을 입력받아서 합계를 구하고 성적의 개수를 센다.
3. 평균을 계산하고 화면에 출력한다.

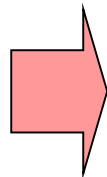
1. 필요한 변수들을 초기화한다.



- (1) `sum`을 0으로 초기화한다.
- (2) `n`을 0으로 초기화한다.
- (3) `grade`를 0으로 초기화한다.

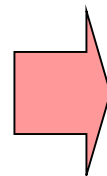
성적들의 평균을 구하는 문제

2. 성적을 입력받아서 합계를 구하고 성적의 개수를 센다.



while 성적이 0보다 작지 않으면
(1) 사용자로부터 성적을 읽어서 **grade**에 저장한다.
(2) **sum**에 이 점수를 누적한다.
(3) **n**을 하나 증가한다.

3. 평균을 계산하고 화면에 출력한다.



(1) **sum**을 **n**으로 나누어서 **average**에 저장한다.
(2) **average**를 화면에 출력한다.

센티넬 예제 1/2

```
// while 문을 이용한 성적의 평균 구하기 프로그램
#include <stdio.h>

int main(void)
{
    int grade, n;
    float sum, average;

    // 필요한 변수들을 초기화한다.
    n = 0;
    sum = 0;
    grade = 0;

    printf("성적 입력을 종료하려면 음수를 입력하시오\n");
```

센티넬 예제 2/2

```
// 성적을 입력받아서 합계를 구하고 학생 수를 센다.  
while (grade >= 0)  
{  
    printf("성적을 입력하시오: ");  
    scanf("%d", &grade);  
  
    sum += grade;  
    n++;  
}  
  
sum = sum - grade; // 마지막 데이터를 제거한다.  
n--; // 마지막 데이터를 제거한다.  
// 평균을 계산하고 화면에 출력한다.  
average = sum / n;  
printf("성적의 평균은 %f입니다.\n", average);  
  
return 0;  
}
```

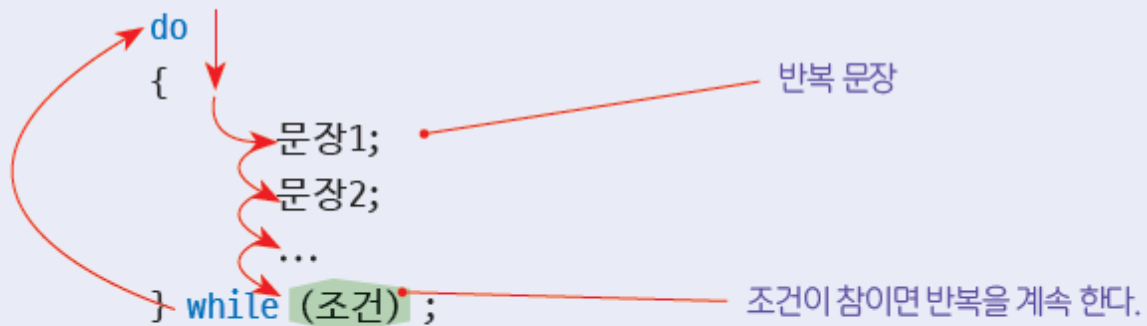
```
종료하려면 음수를 입력하시오  
성적을 입력하시오: 10  
성적을 입력하시오: 20  
성적을 입력하시오: 30  
성적을 입력하시오: 40  
성적을 입력하시오: 50  
성적을 입력하시오: -1  
성적의 평균은 30.000000입니다.
```

do...while문

Syntax

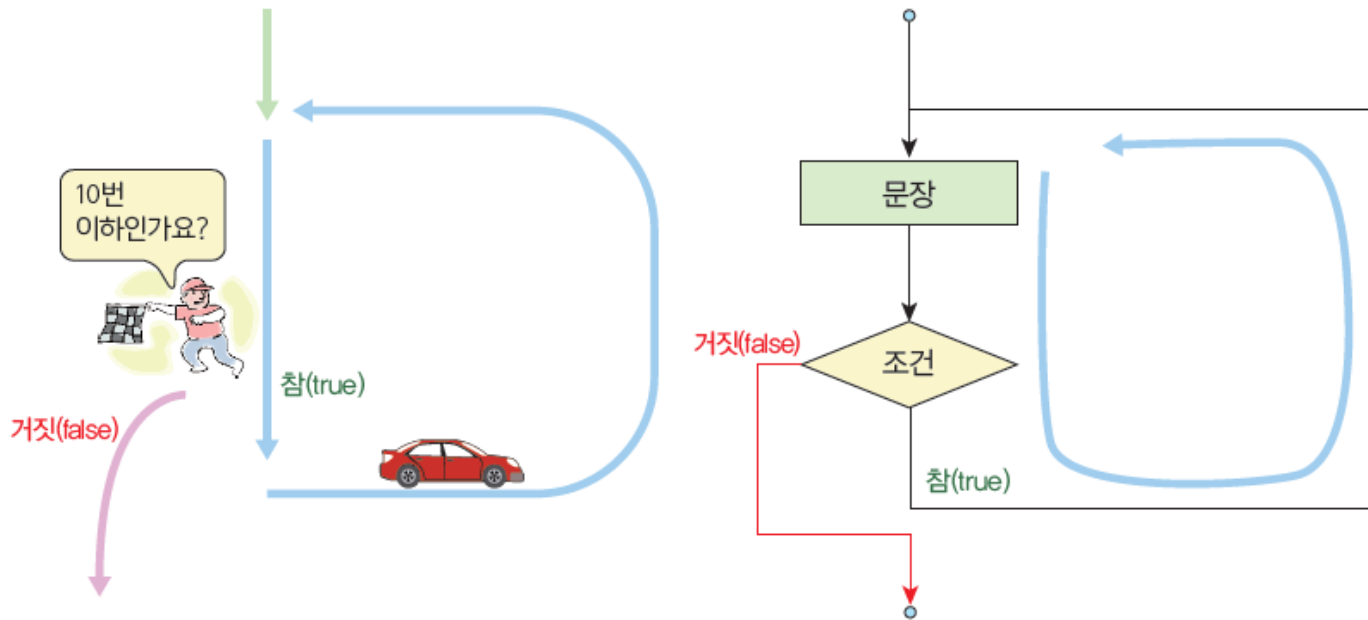
do...while 문

예




do-while 문

- 적어도 한번은 반복문장을 실행한다.



예제 #1

- do...while 문을 이용하여 사용자가 0을 입력할 때까지 입력된 숫자들을 더하는 프로그램을 작성해보자.



```
정수를 입력하시오: 10  
정수를 입력하시오: 20  
정수를 입력하시오: 30  
정수를 입력하시오: 0  
숫자들의 합 = 60
```


예제 #1

```
#define _CRT_SECURE_NO_WARNINGS
// 사용자가 0을 입력할 때까지 숫자를 더한다.
#include <stdio.h>
int main(void)
{
    int number, sum = 0;


    // 루프 몸체가 적어도 한번은 실행된다.
    do
    {
        printf("정수를 입력하시오: ");
        scanf("%d", &number);
        sum += number;
    } while (number != 0);

    printf("숫자들의 합 = %d \n", sum);

    return 0;
}
```

예제 #2

- do..while 문은 입력을 처리하는 부분에서 많이 사용된다.



```
1---새로만들기  
2---파일열기  
3---파일닫기  
하나를 선택하시요: 1  
선택된 메뉴=1
```

예제 #2

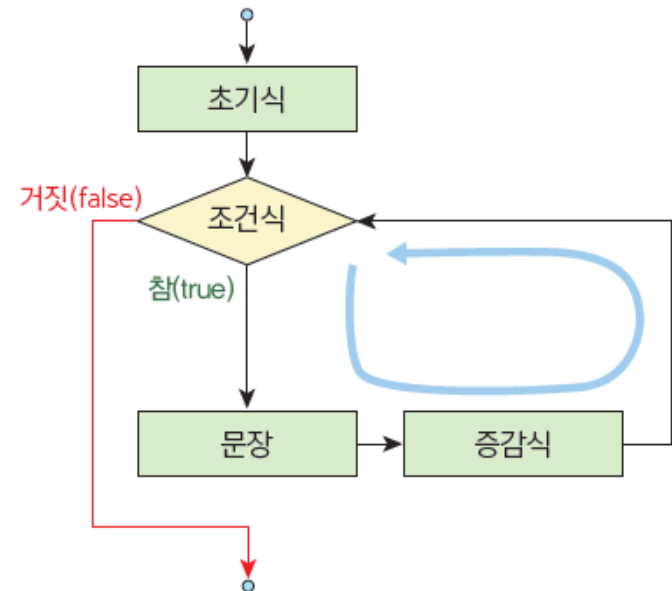
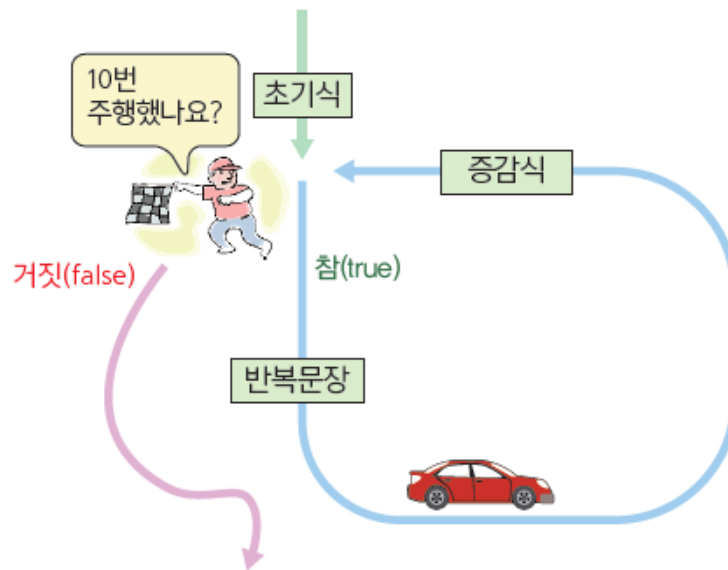
```
// do..while 문을 이용한 메뉴
#include <stdio.h>

int main(void)
{
    int i = 0;
    do
    {
        printf("1---새로만들기\n");
        printf("2---파일열기\n");
        printf("3---파일닫기\n");
        printf("하나를 선택하세요.\n");
        scanf("%d", &i);
    } while(i < 1 || i > 3);

    printf("선택된 메뉴=%d\n",i);
    return 0;
}
```

for 루프

- 정해진 횟수만큼 반복하는 구조



for 문의 구조

Syntax

for 문

예

```
for( 초기식; 조건식; 증감식 ) {  
    printf("Hello World!");  
} ;
```

초기식 조건식 증감식

반복되는 문장

초기식, 조건식, 증감식

- 초기식

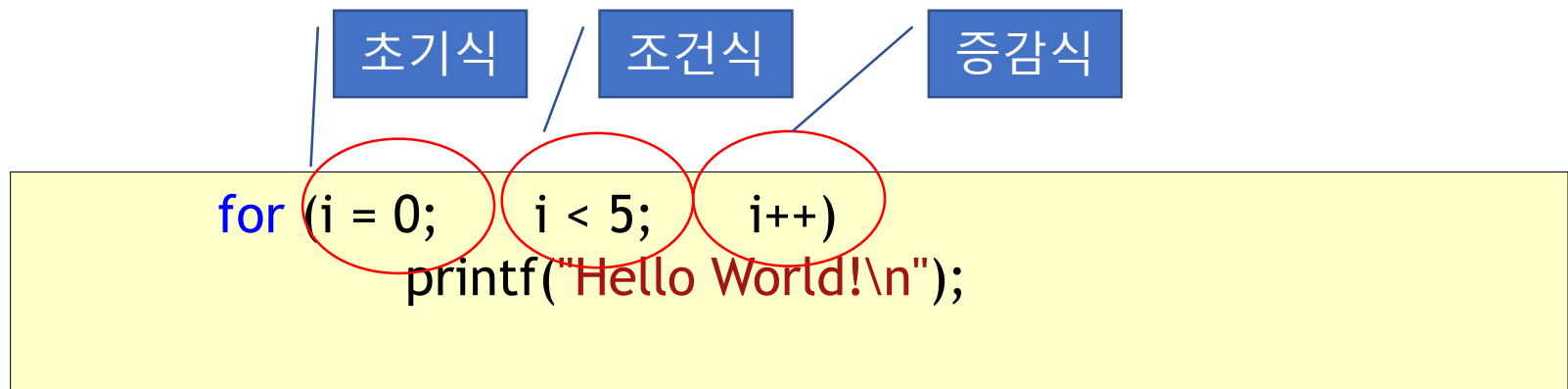
- 초기식은 반복 루프를 시작하기 전에 한번만 실행된다. 주로 변수 값을 초기화하는 용도로 사용된다.

- 조건식

- 반복의 조건을 검사하는 수식이다. 이 수식의 값이 거짓이 되면 반복이 중단된다.

- 증감식

- 한 번의 루프 실행이 끝나면 증감식이 실행된다.




예제

```
// “Hello World!” 5번 출력하기
#include <stdio.h>

int main(void)
{
    int i;

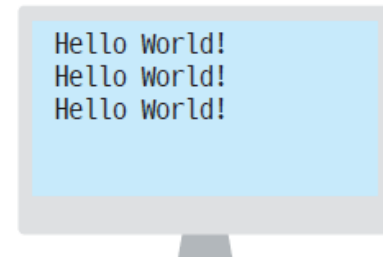
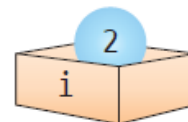
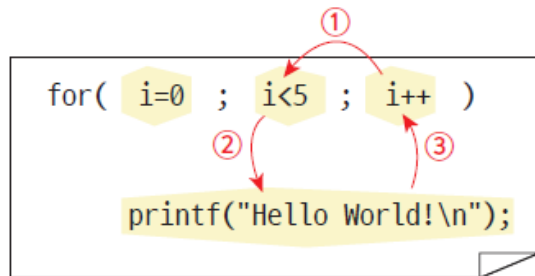
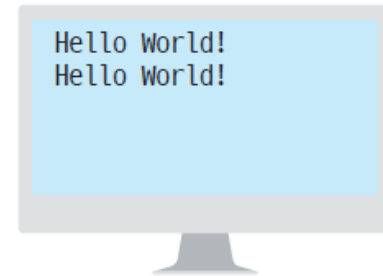
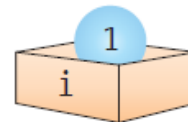
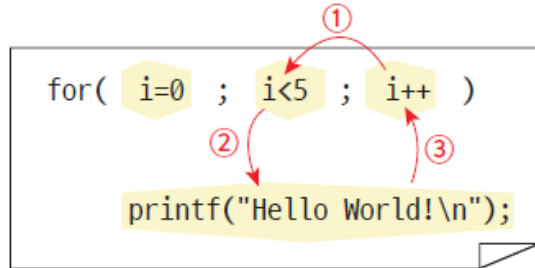
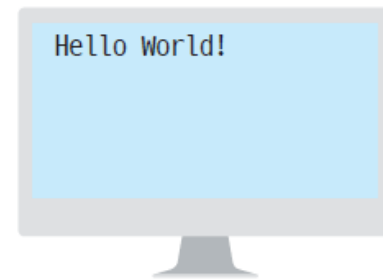
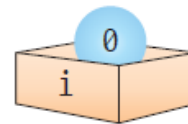
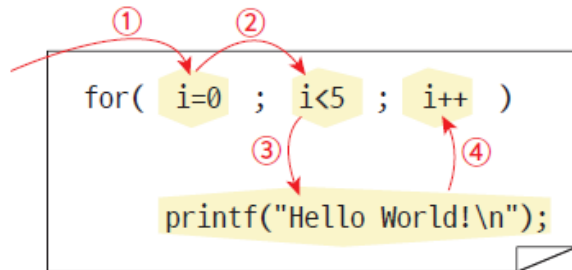
    for (i = 0; i < 5; i++) // i는 0부터 4까지 증가
        printf("Hello World!\n");

    return 0;
}
```

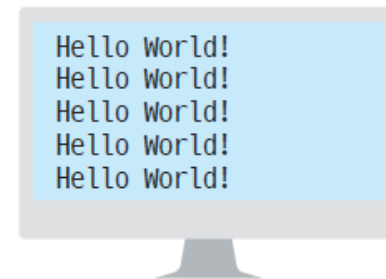
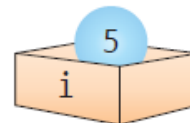
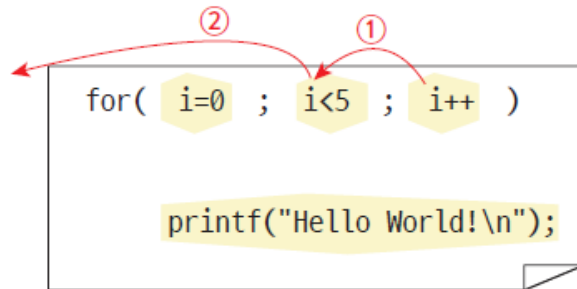
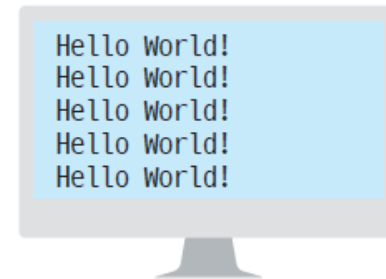
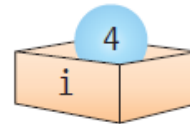
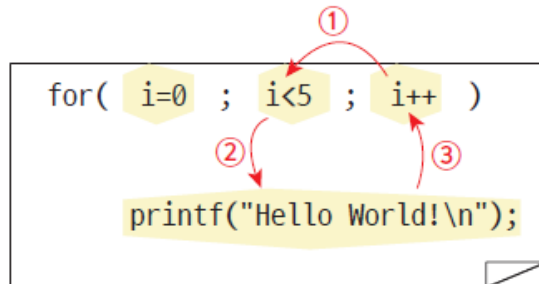
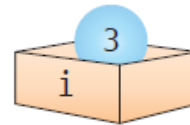
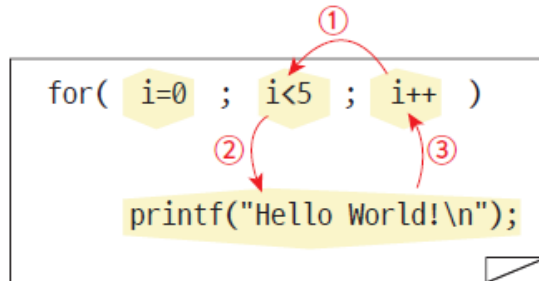


```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

for문의 실행과정



for문의 실행과정



예제 #2



실습

- 1부터 10까지의 정수를 더하여 합계를 구하는 프로그램을 작성해보자.



1부터 10까지의 정수의 합 = 55

예제 #2

```
// 반복을 이용한 정수합 프로그램
#include <stdio.h>

int main(void)
{
    int i, sum;

    sum = 0;
    for(i = 1; i <= 10; i++)
        sum += i;           // sum = sum + i;와 같음

    printf("1부터 10까지의 정수의 합= %d\n",sum);

    return 0;
}
```

1부터 10까지의 정수의 합 = 55

예제 #3

- 화면에 * 글자를 이용하여 다음과 같은 네모를 그려보자.



예제 #3

```
// 반복을 이용한 네모 그리기
#include <stdio.h>

int main(void)
{
    int i;
    printf("*****");

    for(i = 0; i < 5; i++)
        printf("*      *");

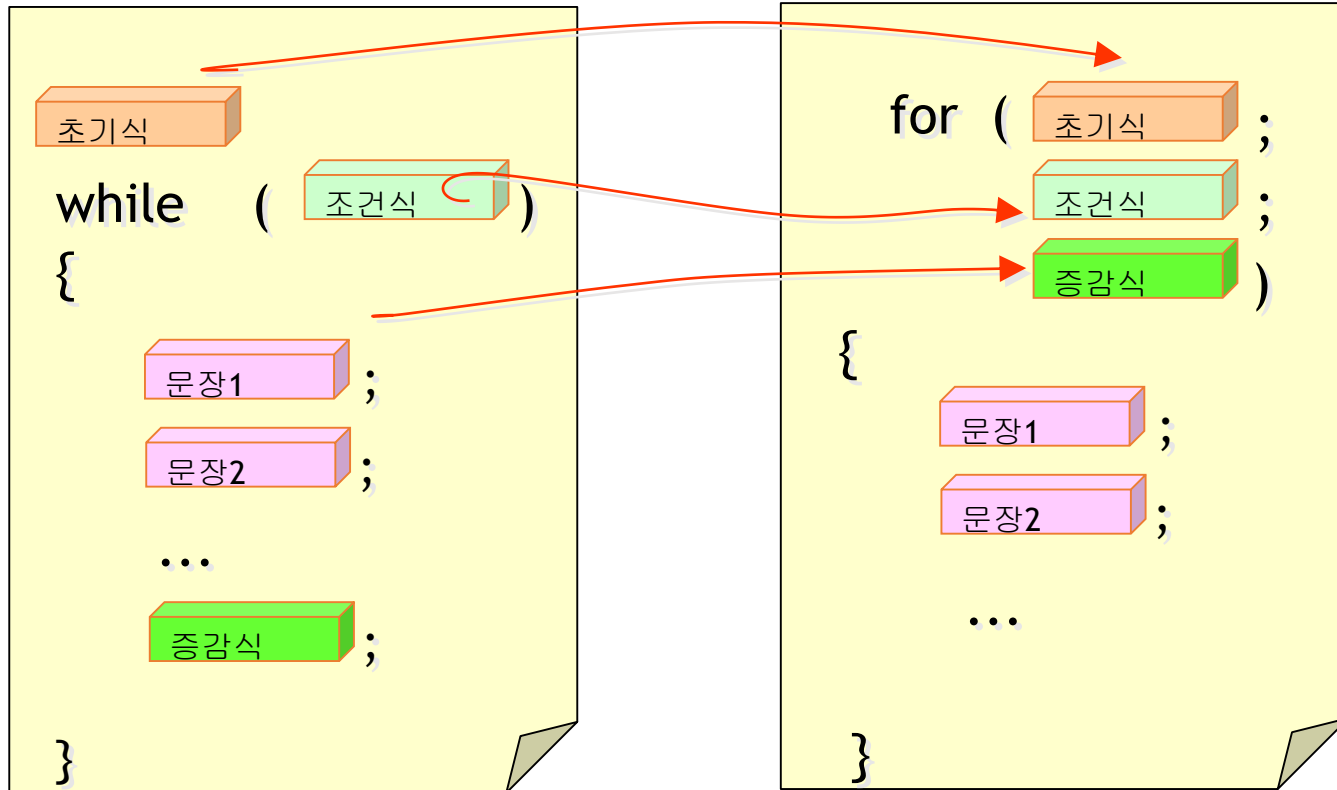
    printf("*****");

    return 0;
}
```



```
*****
*          *
*          *
*          *
*          *
*          *
*****
```

while 루프와 for 루프와의 관계



C11부터는 for 루프 안에서 변수 선언 가능

```
for(int i =0; i< 10; i++) {  
    ...  
}
```

Tip

3가지의 반복문 for, while, do...while 중에서 어떤 것을 사용해야 하는가?

부분적으로는 개인적인 취향의 문제이다. 일반적인 선택 기준은 루프의 반복 횟수를 아는 경우에는 for 루프가 while 루프에 비하여 약간 더 편리하다고 할 수 있다. 즉 루프 제어 변수를 증가하는 것을 잊어버린다거나 하는 일이 while 루프에 비하여 덜 발생한다. 만약 조건만 존재하고 정확한 반복 횟수는 모르는 경우에는 while 구조가 좋다. 만약 반드시 한번은 수행되어야 하는 문장들이 있다면 do...while 구조가 제격이다.

또한 while과 for는 반복하기 전에 조건을 검사하는 구조이고 do...while은 먼저 실행한 후에 반복 조건을 검사한다. 특별한 경우가 아닌 일반적인 경우에는 반복을 하기 전에 조건 검사를 하는 것이 좋다. 뭐든지 실행하기 전에 면밀하게 사전 조사를 하는 것이 좋은 것과 마찬가지이다.



다양한 증감수식의 형태

```
for (int i = 10; i > 0; i-- )  
    printf("Hello World!\n");
```

뺄셈 사용

```
for (int i = 0; i < 10; i += 2 )  
    printf("Hello World!\n");
```

2씩 증가

```
for (int i = 1; i < 10; i *= 2 )  
    printf("Hello World!\n");
```

2를 곱한다.

```
for (int i = 0; i < 100; i = (i * i) + 2 )  
    printf("Hello World!\n");
```

어떤 수식이라도 가능

다양한 증감수식의 형태

```
for ( ; ; )  
    printf("Hello World!\n");
```

무한 반복 루프

```
for ( ; i<100; i++ )  
    printf("Hello World!\n");
```

한 부분이 없을 수도 있다.

```
for (i = 0, k = 0; i < 100; i++ )  
    printf("Hello World!\n");
```

2개 이상의 변수 초기화

```
for (printf("반복시작"), i = 0; i < 100; i++ )  
    printf("Hello World!\n");
```

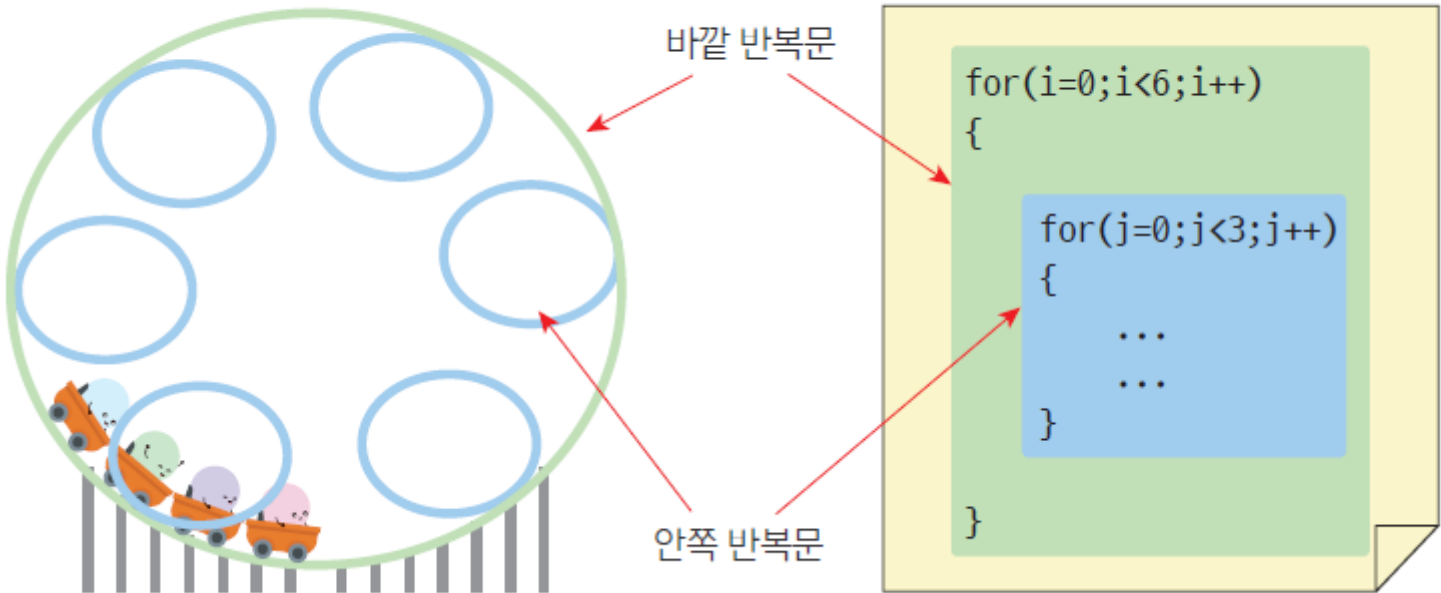
어떤 수식도 가능

```
for (i = 0; i < 100 && sum < 2000; i++ )  
    printf("Hello World!\n");
```

어떤 복잡한 수식도 조건식이 될 수 있다.

중첩 반복문

- 중첩 반복문(nested loop): 반복문 안에 다른 반복문이 위치



예제 #1

실습

- 다음 예제는 *기호를 사각형 모양으로 출력한다.



예제 #1

```
// 중첩 for 문을 이용하여 *기호를 사각형 모양으로 출력하는 프로그램
#include <stdio.h>

int main(void)
{
    int x, y;

    for(y = 0; y < 5; y++)
    {
        for(x = 0; x < 10; x++)
            printf("*");
        printf("\n");
    }

    return 0;
}
```



```
*****
*****
*****
*****
*****
```

예제 #2

- 앞의 예제를 조금 변경시켜서 다음과 같이 출력되도록 하여보자. 실행 결과를 자세히 분석하여 보면 y 번째 줄에서 y 개의 *를 출력하는 것을 알 수 있다



예제 #2

```
#include <stdio.h>
int main(void)
{
    int x, y;
    for(y = 1; y <= 5; y++)
    {
        for(x = 0; x < y; x++)
            printf("*");
        printf("\n"); // 내부 반복문이 종료될 때마다 실행
    }

    return 0;
}
```



```
*
**
***
****
*****
```

무한 루프

- 조건 제어 루프에서 가끔은 프로그램이 무한히 반복하는 일이 발생한다. 이것은 무한 루프(infinite loop)로 알려져 있다. 무한 반복이 발생하면 프로그램은 빠져 나올 수 없기 때문에 문제가 된다.
- 하지만 가끔은 의도적으로 무한 루프가 사용되는데 예를 들면 신호등 제어 프로그램은 무한 반복하여야 하기 때문이다

Syntax

무한 루프 문

```
Syntax while (1) {  
    if (조건)  
        break;    # 반복을 중단한다.  
    if (조건)  
        continue; # 다음 반복을 시작한다.  
}
```


무한루프가 유용한 경우

- 특히 반복을 빠져나가는 조건이 까다로운 경우에 많이 사용된다. 예를 들어서 사용자가 입력한 수가 3의 배수이거나 음수인 경우에 while 루프를 빠져나가야 한다고 하자.

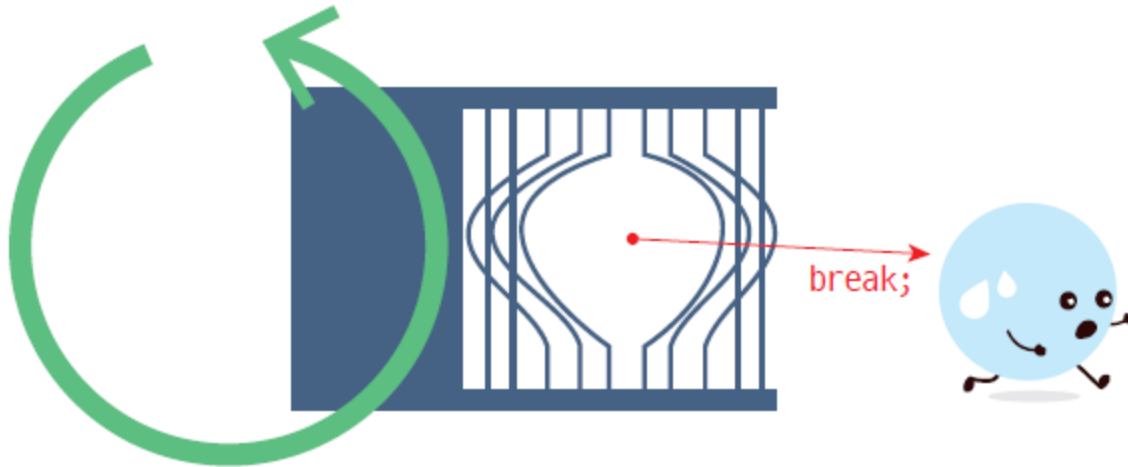
```
while((x % 3 != 0) && (x >= 0)) {  
    ...  
    ...  
    ...  
}
```



```
while (1) {  
    if (x%3 == 0) break;  
    if (x<0) break;  
    ...  
}
```

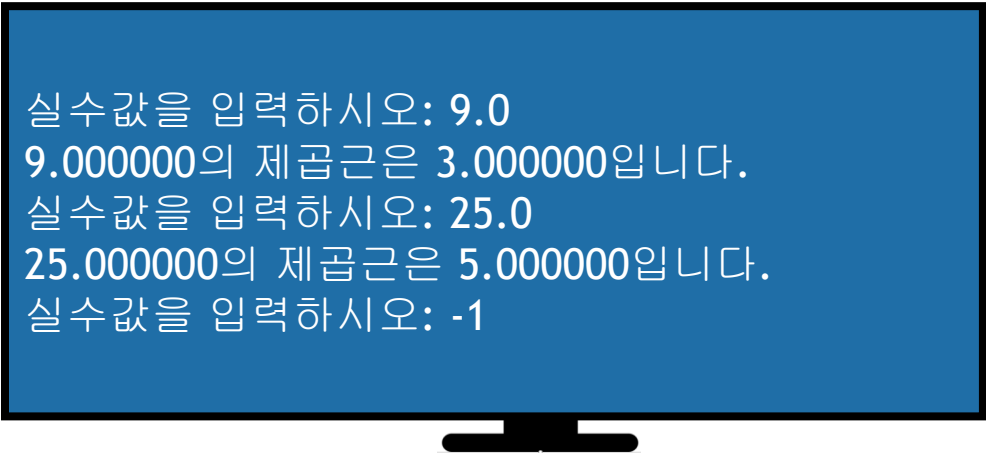
break 문

- break 문은 반복 루프를 빠져 나오는데 사용된다.



예제

- 여기서는 무한 루프를 만들어서 사용자로부터 입력받은 실수의 제곱근을 구하여 출력하는 프로그램을 작성하여 보자.
- 허수는 생각하지 않는다고 하면 제곱근은 양의 실수에 대해서만 계산할 수 있으므로 만약 입력된 값이 음수이면 무한 루프를 종료하도록 하자. 무한 루프를 종료하는데 break 문을 사용한다.



실수값을 입력하시오: 9.0
9.000000의 제곱근은 3.000000입니다.
실수값을 입력하시오: 25.0
25.000000의 제곱근은 5.000000입니다.
실수값을 입력하시오: -1

// break를 이용하여 무한루프를 탈출한다.

#include <stdio.h>

#include <math.h>

int main(void)

{

double v;

while(1)

{

printf("실수값을 입력하시오: ");

scanf("%lf", &v);

if(v < 0.0)

break;

printf("%f의 제곱근은 %f입니다.\n", v, sqrt(v));

}

return 0;

}

실수값을 입력하시오: 9.0

9.000000의 제곱근은 3.000000입니다.

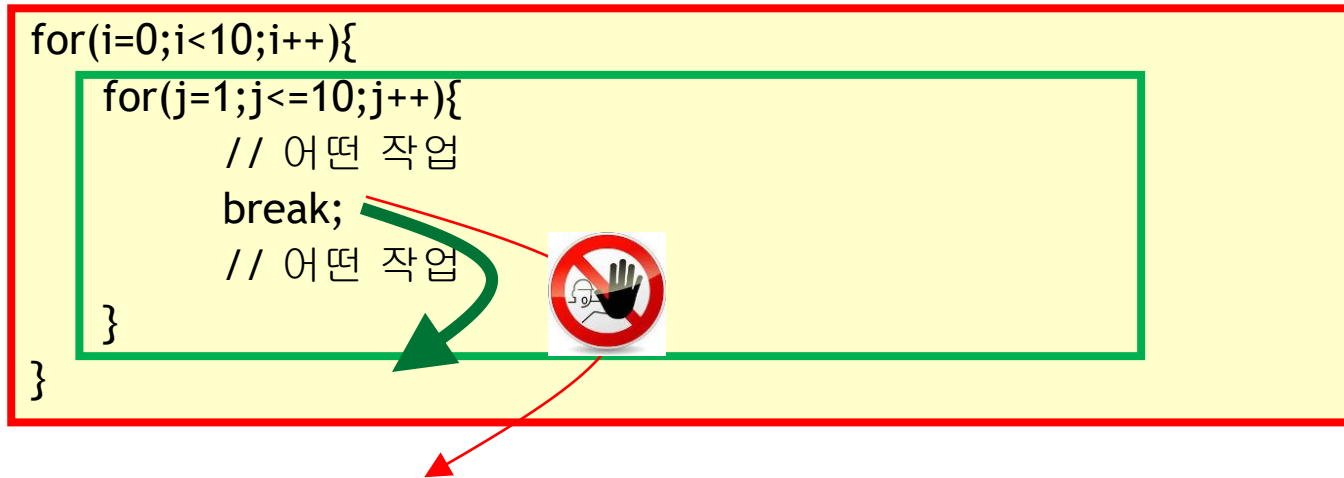
실수값을 입력하시오: 25.0

25.000000의 제곱근은 5.000000입니다.

실수값을 입력하시오: -1

goto문이 필요한 유일한 경우

- 중첩 루프 안에서 어떤 문제가 발생했을 경우, goto를 이용하면 단번에 외부로 빠져 나올 수 있다.
- break를 사용하면, 하나의 루프만을 벗어 날 수 있다.



goto문의 사용

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

```
int main(void)
```

```
{
```

```
    int x, y;
```

```
    for(y = 1; y < 10000; y++)
```

```
    {
```

```
        for(x = 1; x < 50; x++)
```

```
        {
```

```
            if( kbhit() ) goto OUT;
```

```
            printf("*");
```

```
        }
```

```
        printf("\n");
```

```
    }
```

```
OUT:
```

```
    return 0;
```

```
}
```

```
*****  
*****  
*****
```

continue 문

- 0부터 10까지의 정수 중에서 3의 배수만 제외하고 출력하는 예제를 살펴보자.

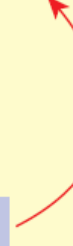
```
#include <stdio.h>

int main(void)
{
    int i;
    for( i=0 ; i<10 ; i++ )
    {
        if( i%3 == 0 )
            continue;
        printf("%d ", i);
    }
    return 0;
}
```


1 2 4 5 7 8

continue 문


```
while ( 조건식 )  
{  
    문장 ;  
    문장 ;  
    contunue  
    문장 ;  
}
```



```
do  
{  
    문장 ;  
    문장 ;  
    contunue  
    문장 ;  
} while ( 조건식 );
```



```
for ( 초기식 ;  
      조건식 ;  
      증감식 )  
{  
    문장 ;  
    문장 ;  
    contunue  
    문장 ;  
}
```



Q & A

