



인하공업전문대학
INHA TECHNICAL COLLEGE

C 프로그래밍

6주차

인하공업전문대학 컴퓨터 정보과
김한결 강사

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

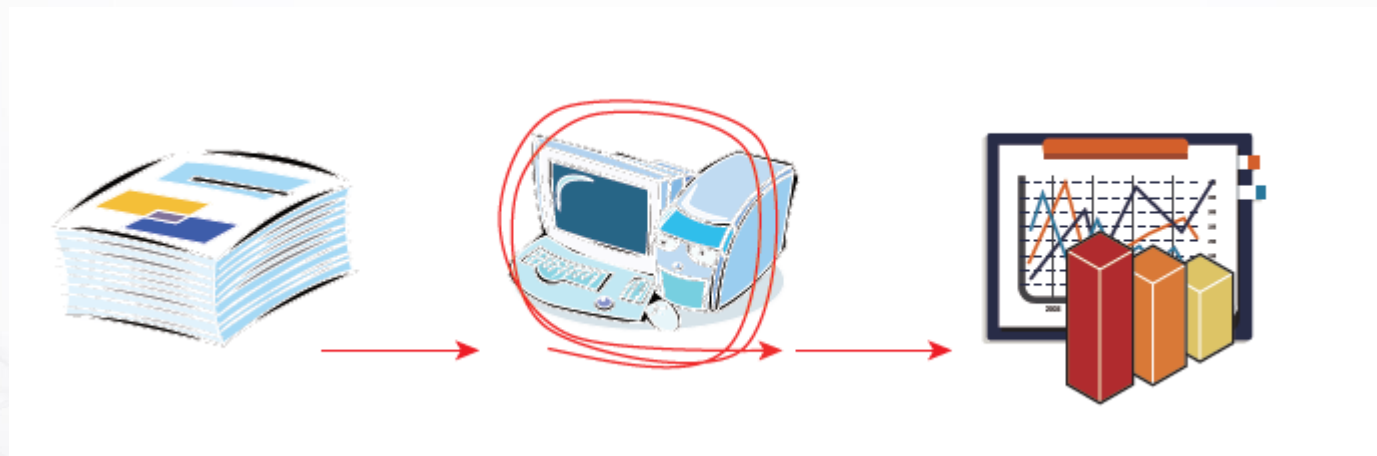
- 인간은 반복을 싫어하지만 프로그램에서는 반복적인 작업들이 반드시 필요하다.
- 반복(iteration)은 같은 처리 과정을 여러 번 되풀이하는 것이다



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?

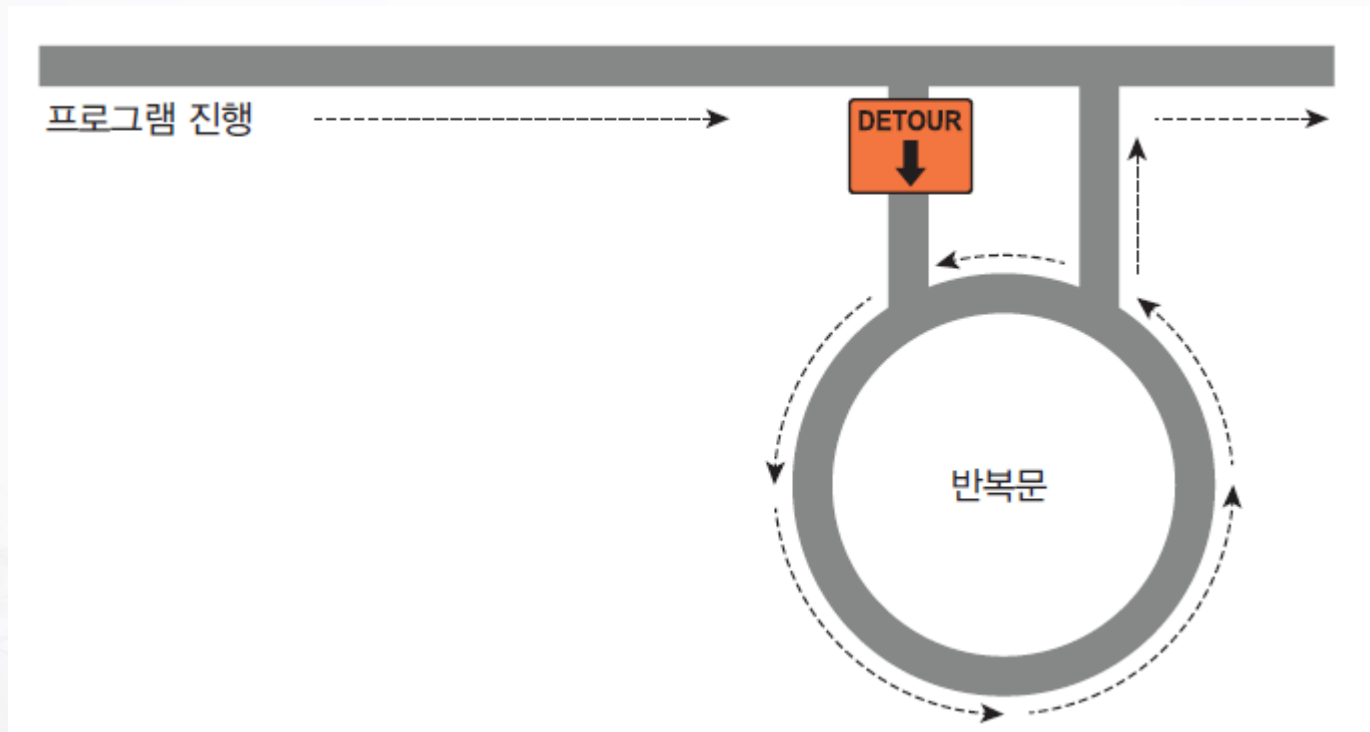
```
printf("Hello World! \n");  
printf("Hello World! \n");  
printf("Hello World! \n");  
printf("Hello World! \n");  
printf("Hello World! \n");
```



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

반복 구조

- 어떤 조건이 만족될 때까지 루프를 도는 구조





while 루프



for 루프

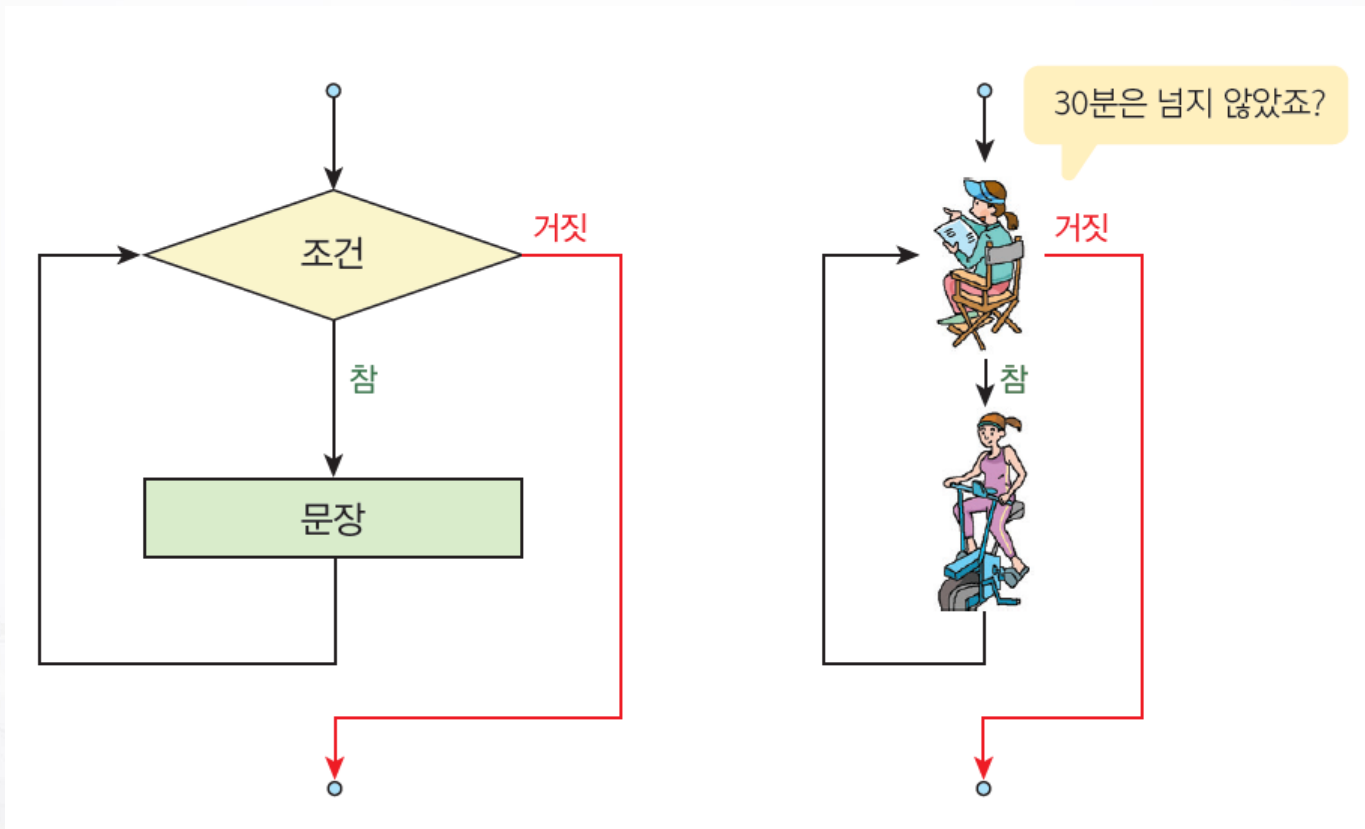
중간 점검

1. 프로그램에 반복 구조가 필요한 이유는 무엇인가?
2. 반복문에는 _____, _____문이 있다.



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;
```

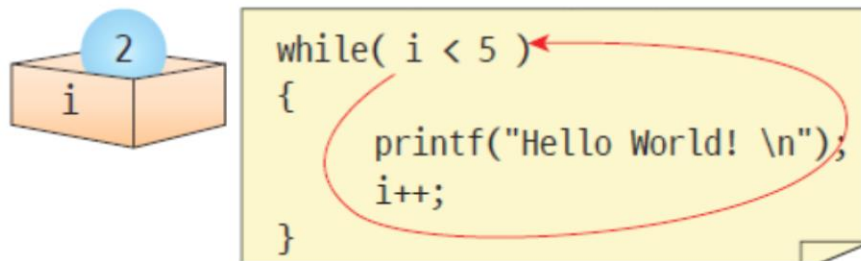
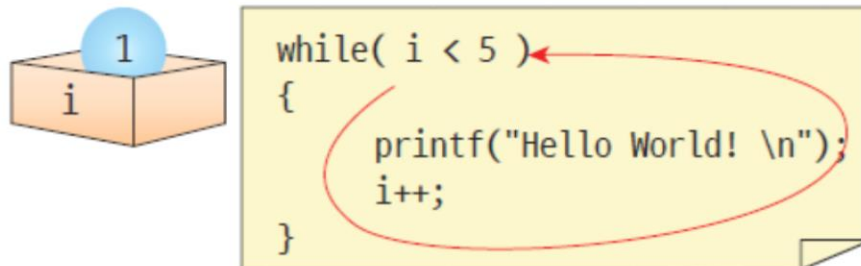
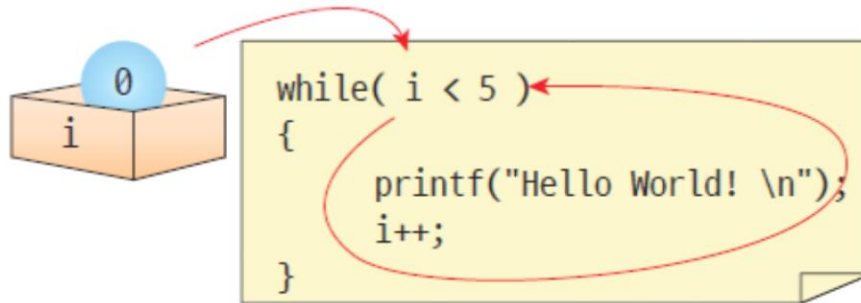
```
}
```

반복 조건

반복 내용

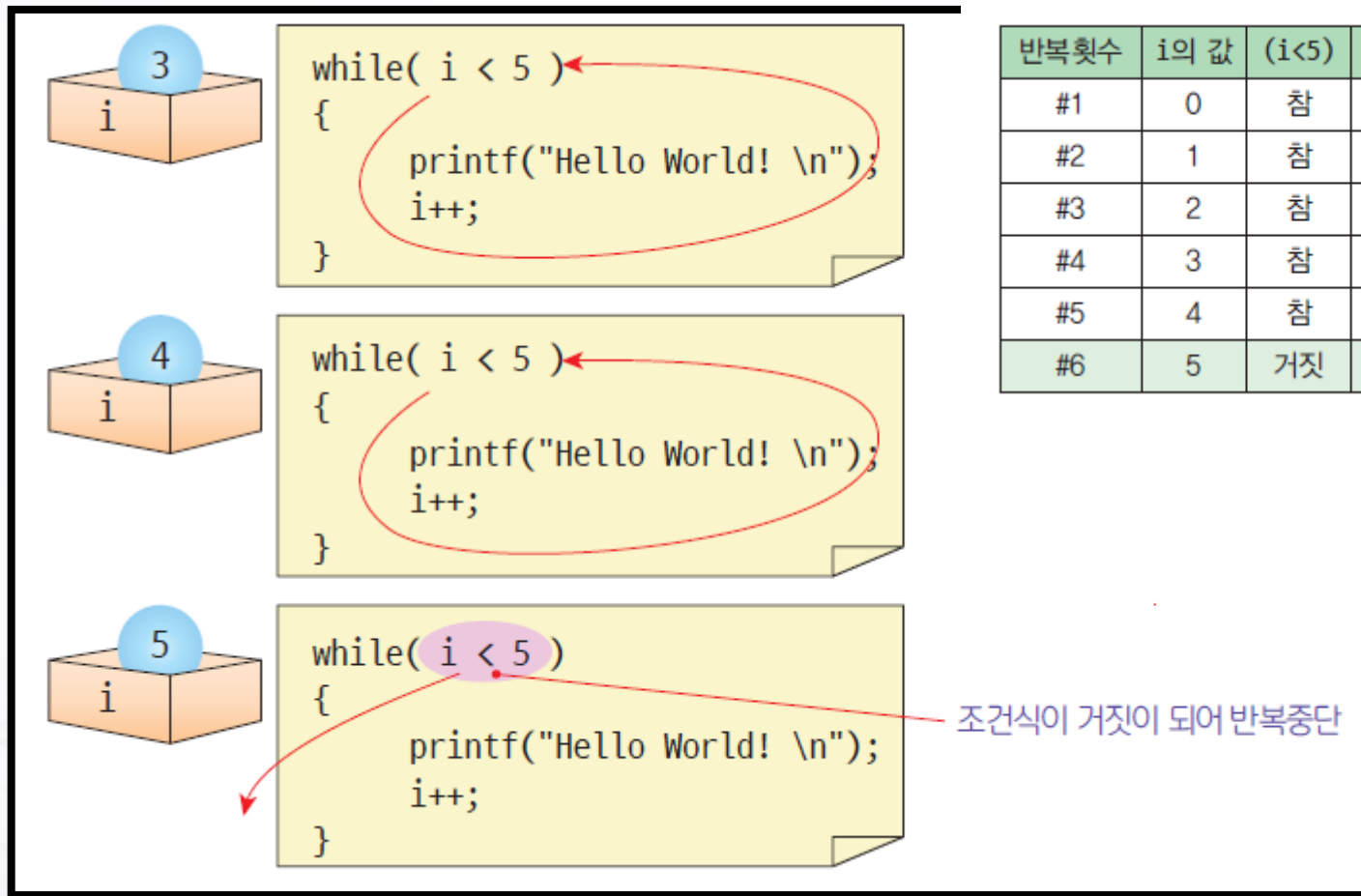
```
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!
```

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



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)
```

```
    {
```

```
    }
```

```
    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

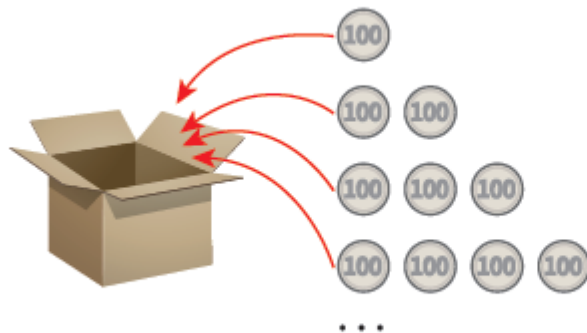
- 1부터 n까지의 합 계산하는 프로그램

정수를 입력하시오: 3
1부터 3까지의 합은 6입니다

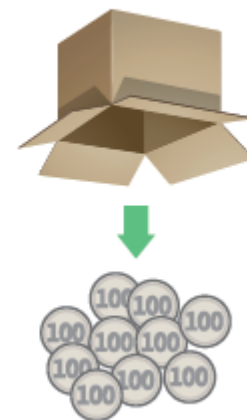
① 빈통을 준비한다.



② 통에 1부터 n까지를 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
```



```
// 변수 초기화
```

```
// sum = sum + i;와 같다.
```

```
// i = i + 1과 같다.
```

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

```
    return 0;
```

```
}
```

정수를 입력하시오: 3
1부터 3까지의 합은 6입니다

예제 #4

- n 이하의 모든 짝수의 합만을 구하려면 어떻게 변경하여야 할까? 짝수의 합을 출력하려면 짝수들만을 `sum`에 더해야 한다.



정수를 입력하시오: 10
1부터 10까지의 짝수합은 30입니다.

예제 #4

```
#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;
```



```
    // sum = sum + i;와 같다.
```

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

```
    return 0;
```


```
}
```

정수를 입력하시오: 10

1부터 10까지의 짝수합은 30입니다.

예제 #5

- 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

중간점검

1. `if` 문과 `while` 문을 비교하여 보라. 조건식이 같다면 어떻게 동작하는가?
2. `while` 루프를 이용하여 무한 루프를 만들어 보라.
3. 다음 코드의 출력을 쓰시오.

```
int n = 10;
while (n > 0) {
    printf("%d\n", n);
    n = n - 3;
}
```

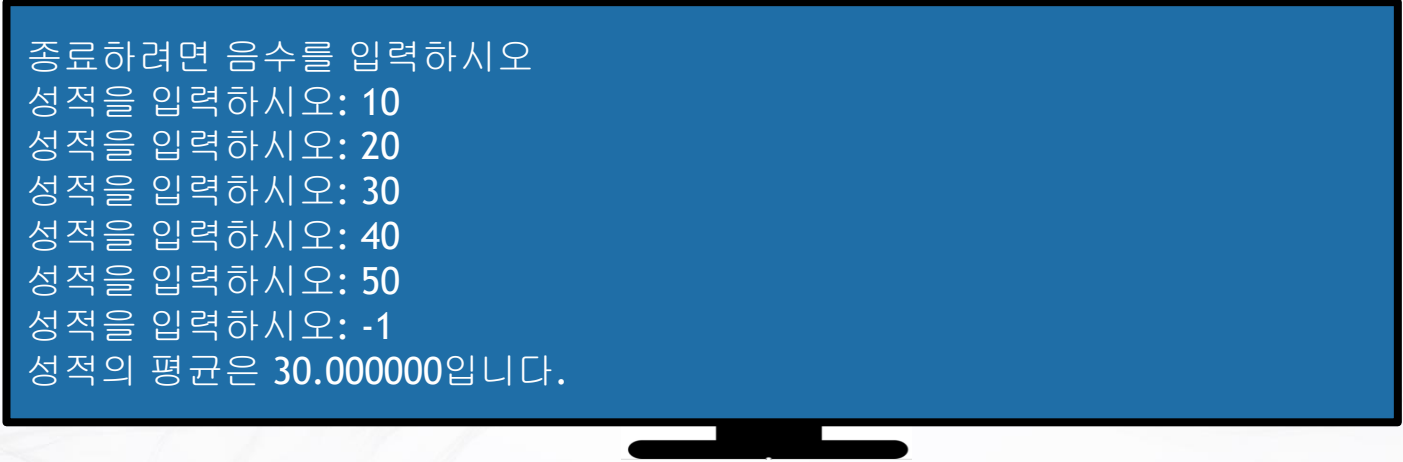
센티널(보초값의 이용)

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



성적의 평균을 계산하는 문제

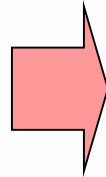
- 사용자로부터 임의의 개수의 성적을 받아서 평균을 계산한 후에 출력하는 프로그램을 작성하여 보자.
- -1을 보초값으로 사용한다.



```
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.
```

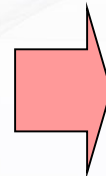
성적들의 평균을 구하는 문제

성적의 평균을 구한다.



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

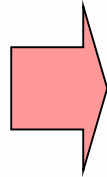
1. 필요한 변수들을 초기화한다.



- (1) sum을 0으로 초기화한다.
- (2) n을 0으로 초기화한다.
- (3) grade를 0으로 초기화한다.

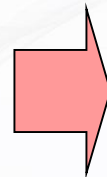
성적들의 평균을 구하는 문제

2. 성적을 입력받아서 합계를 구하고 성적의 개수를 센다.



while 성적이 0보다 작지 않으면
(1) 사용자로부터 성적을 읽어서 grade에 저장한다.
(2) sum에 이 점수를 누적한다.
(3) n을 하나 증가한다.

3. 평균을 계산하고 화면에 출력한다.



(1) sum을 n으로 나누어서 average에 저장한다.
(2) average를 화면에 출력한다.

센티넬 예제 1/2

```
#define _CRT_SECURE_NO_WARNINGS
// 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");
```

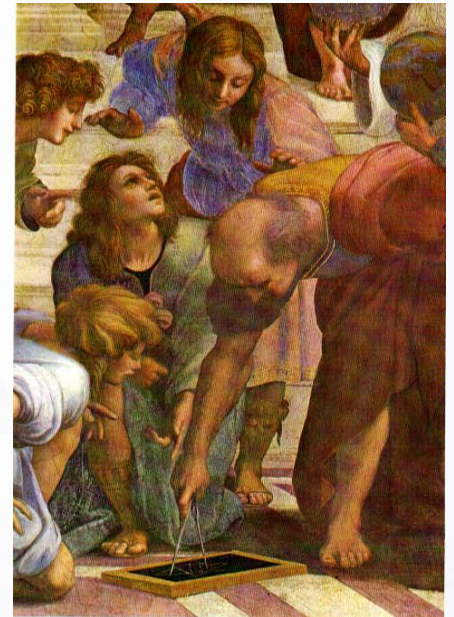

센티넬 예제 2/2

```
// 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.
```

Lab: 최대 공약수 찾기

두 개의 정수를 입력하시오(큰수, 작은수): 25 10
최대공약수는 5입니다.



lab: 최대 공약수 찾기

- 유클리드 알고리즘

- ① 두 수 가운데 큰 수를 x , 작은 수를 y 라 한다.
- ② y 가 0이면 공약수는 x 와 같다.
- ③ $r \leftarrow x \% y$
- ④ $x \leftarrow y$
- ⑤ $y \leftarrow r$
- ⑥ 단계 ②로 되돌아간다.

최대 공약수 찾기

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

```
int main(void)  
{
```

```
    int x, y, r;
```

```
    printf("두개의 정수를 입력하시오(큰수, 작은수): ");  
    scanf("%d%d", &x, &y);
```

```
    while (y != 0)
```

```
    {
```

```
        r = x % y;
```

```
        x = y;
```

```
        y = r;
```

```
    }
```

```
    printf("최대 공약수는 %d입니다.\n", x);
```

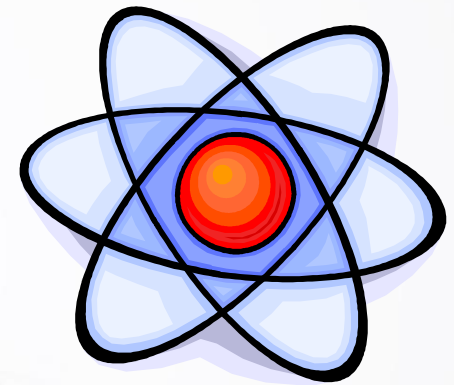
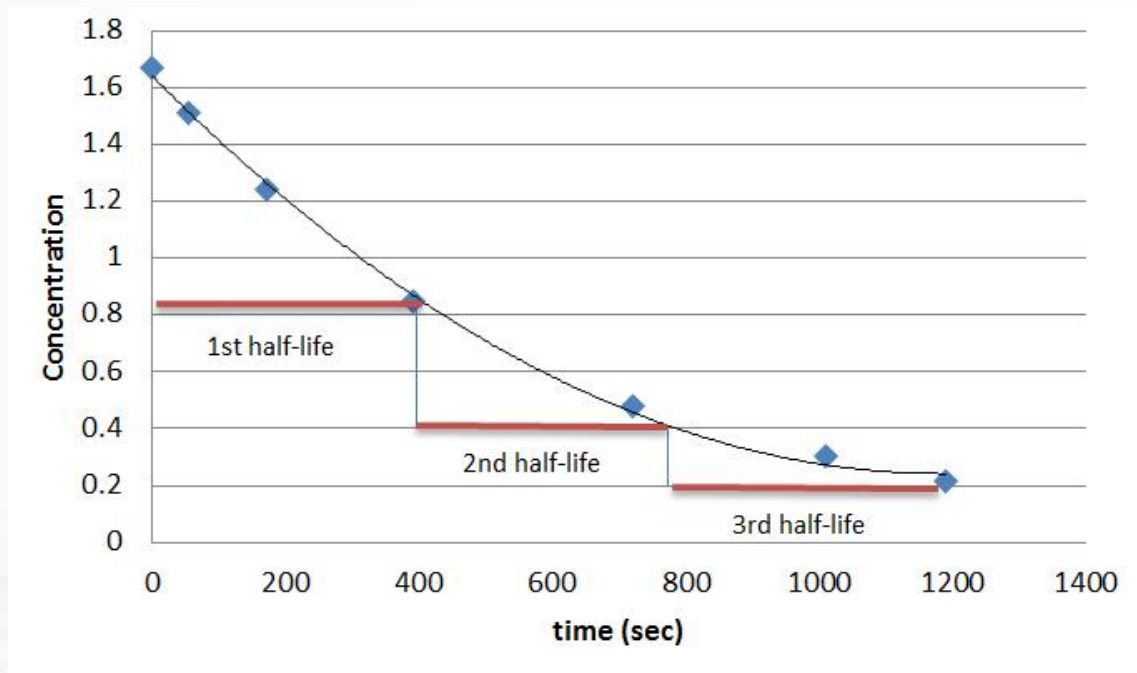
```
    return 0;
```

```
}
```

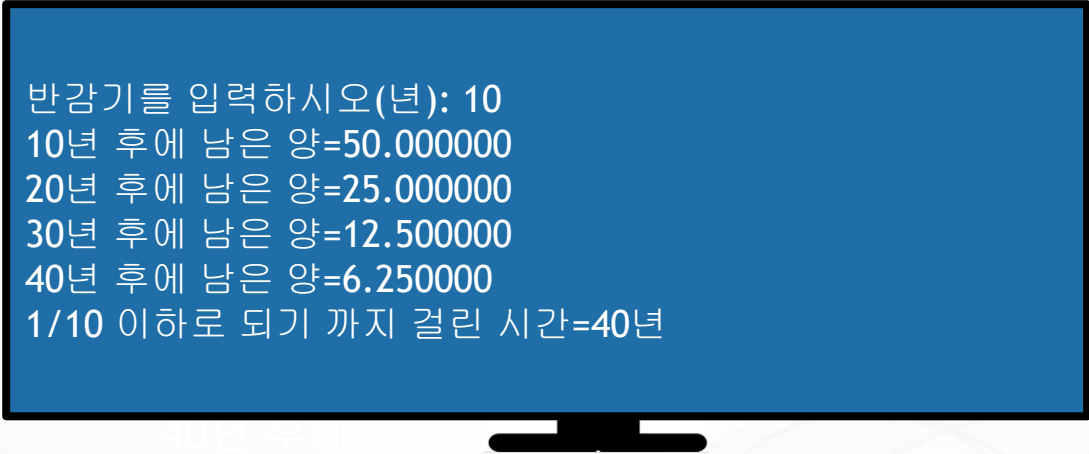
두개의 정수를 입력하시오(큰수, 작은수): 25 10
최대 공약수는 5입니다.

lab: 반감기

- 반감기: 방사능 물질의 양이 $\frac{1}{2}$ 로 되는 시간



실행 결과



반감기를 입력하시오(년): 10
10년 후에 남은 양=50.000000
20년 후에 남은 양=25.000000
30년 후에 남은 양=12.500000
40년 후에 남은 양=6.250000
1/10 이하로 되기 까지 걸린 시간=40년

- 로그 함수는 사용하지 않는다!
- 반복문을 사용한다.

알고리즘

사용자로부터 반감기를 입력받는다.

while(물질의 양 > 초기 물질의 양*0.1)

반감기만큼 시간을 더한다.

물질의 양은 1/2로 줄어든다.

현재 물질의 양을 출력한다.

10% 이하로 되기까지 걸린 시간을 출력한다.

소스

```
#include <stdio.h>
int main(void)
{
    int halflife;
    double initial;
    double current;
    int years=0;

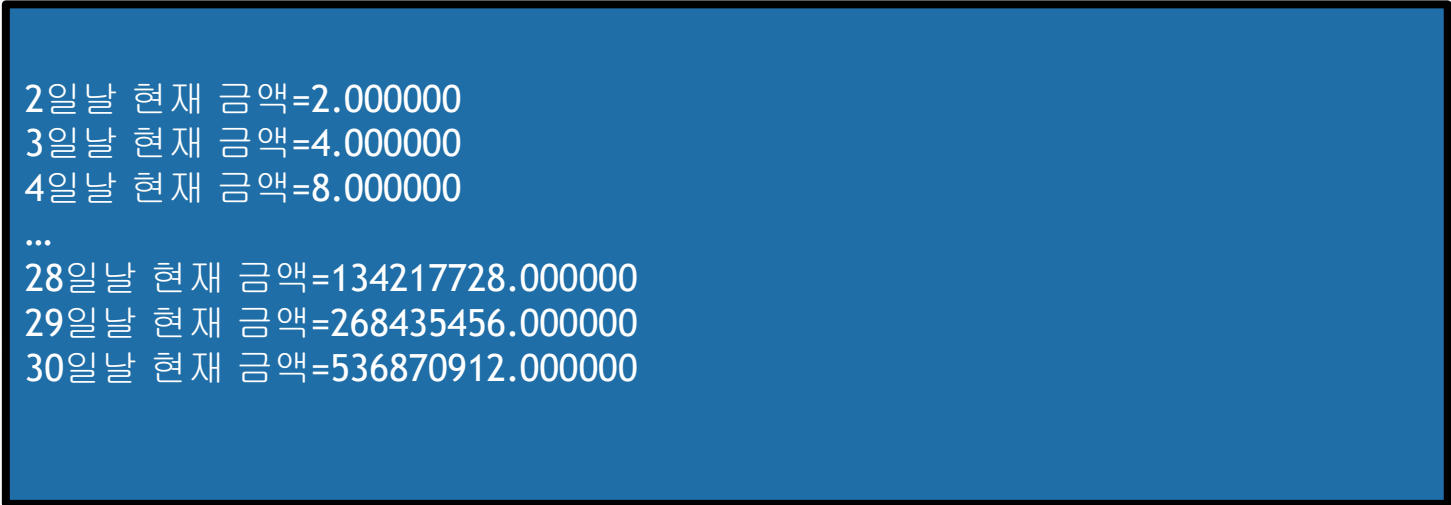
    printf("반감기를 입력하시오(년): ");
    scanf("%d", &halflife);

    initial = 100.0;
    current = initial;
    while( current > initial/10.0 ){
        years += halflife;
        current = current / 2.0;
        printf("%d년 후에 남은 양=%f", years, current);
    }

    printf("1/10 이하로 되기까지 걸린 시간=%d년", years);
    return 0;
}
```


추가 Lab: 복리의 무서움

- 여러분이라면 1억원을 일시불로 받을 것인가? 아니면 첫날 1원을 받지만, 이후 30일 동안 전날보다 두 배씩 받는 것을 선택할 것인가?



2일날 현재 금액=2.000000
3일날 현재 금액=4.000000
4일날 현재 금액=8.000000
...
28일날 현재 금액=134217728.000000
29일날 현재 금액=268435456.000000
30일날 현재 금액=536870912.000000

Sol:

```
#include <stdio.h>

int main(void) {
    double money = 1.0;

    for (int i = 2; i <= 30; i++) {
        money *= 2.0;
        printf("%d일날 현재 금액=%lf\n", i, money);
    }
    return 0;
}
```

도전문제

- 위와 비슷한 문제를 하나 더 작성해보자. 세균이 1시간마다 4배씩 증가한다고 가정하자. 이 세균 10마리를 배양하면 7시간 후의 세균의 수는 얼마나 될까? 역시 지수 함수나 로그 함수를 이용하지 말고 반복 구조만을 사용하여서 해결하여 보자.
- 종이를 한번 접으면 면적이 $1/2$ 로 줄어든다. 종이를 몇 번 접어야 원래 면적의 $1/100$ 로 줄어드는가? 역시 로그 함수나 지수 함수를 사용하지 말고 반복 구조를 이용하여서 해결하여 보자.



do...while문

Syntax

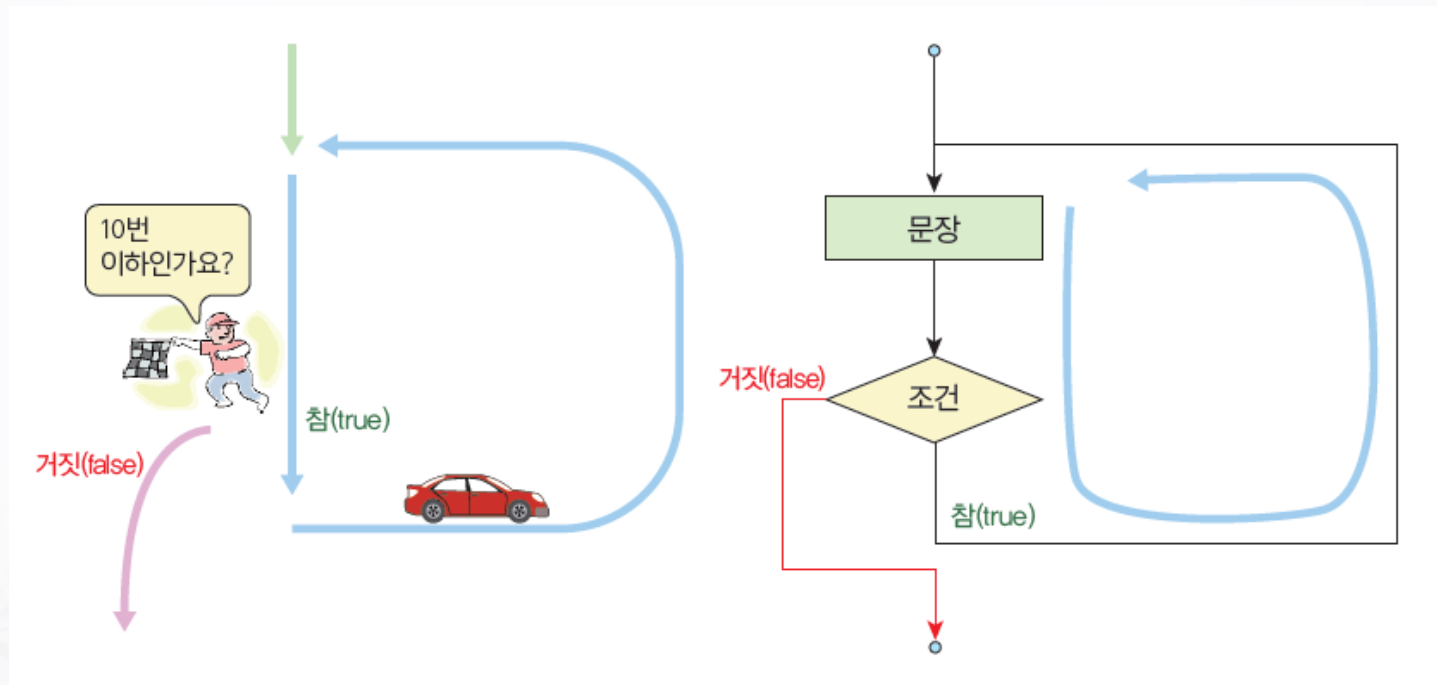
do...while 문

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 선택하시요.\n");
        scanf("%d", &i);
    } while(i < 1 || i > 3);

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

중간점검

1. 다음 코드의 출력을 쓰시오.

```
int n = 0;  
do {  
    printf("%d\\n", n);  
    n = n + 1;  
} while( n < 3 );
```

Lab: 숫자 추측 게임

- 프로그램이 가지고 있는 정수를 사용자가 알아맞히는 게임

정답을 추측하시오: 50
LOW
정답을 추측하시오: 75
HIGH
정답을 추측하시오: 60
축하합니다. 시도횟수=3



알고리즘

do

사용자로부터 숫자를 `guess`로 입력받는다.

시도횟수를 증가한다.

if(`guess < answer`)

숫자가 낮다고 출력한다.

if(`guess > answer`)

숫자가 높다고 출력한다.

while(`guess != answer`);

“축하합니다”와 시도횟수를 출력한다.

소스

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

int main(void)
{
    srand((unsigned)time(NULL));    // 난수 발생기 시드 설정
    int answer = rand()%100;    // 정답을 난수로 발생한다.
    int guess;
    int tries = 0;
```

소스

```
// 반복 구조
do {
    printf("정답을 추측하여 보시오: ");
    scanf("%d", &guess);
    tries++;
    if (guess > answer) // 사용자가 입력한 정수가 정답보다 높으면
        printf("HIGH \n");
    if (guess < answer) // 사용자가 입력한 정수가 정답보다 낮으면
        printf("LOW \n");
} while (guess != answer);

printf("축하합니다. 시도횟수=%d\n", tries);
return 0;
}
```

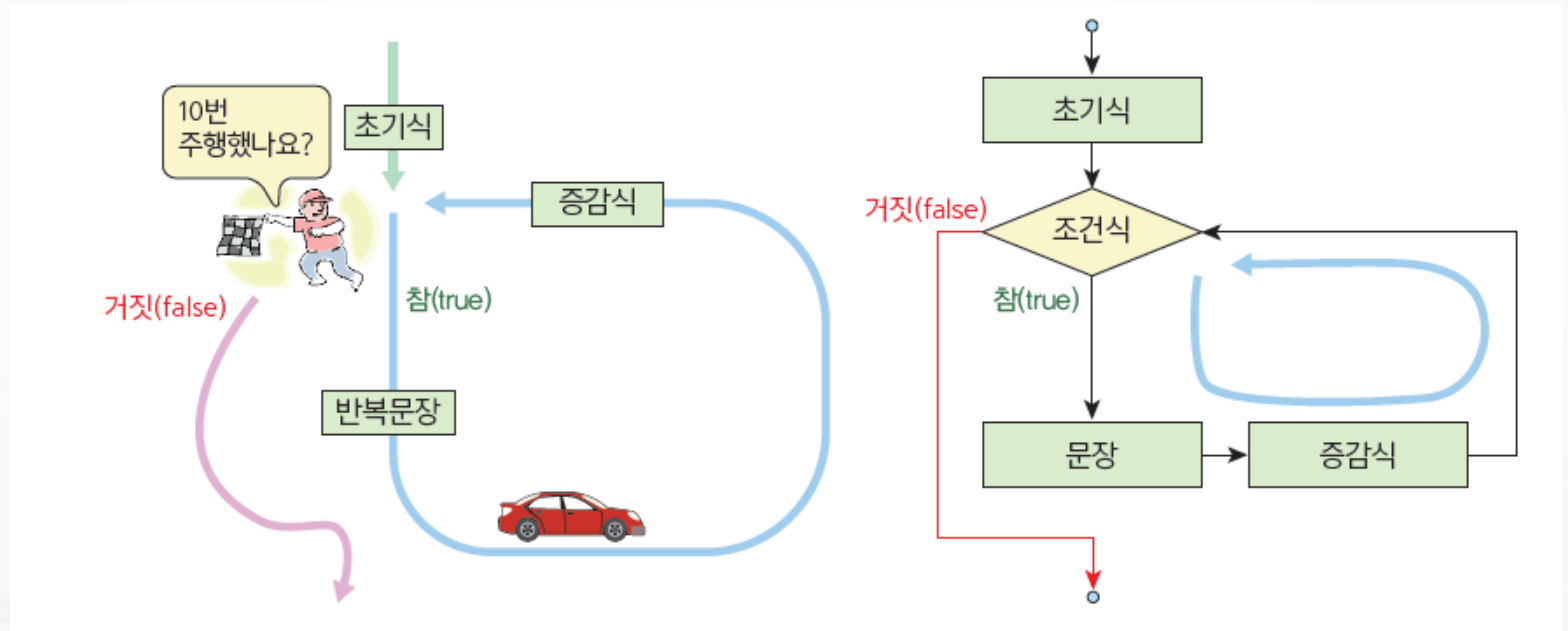
도전문제

- 위의 프로그램이 게임이 되려면 난수를 발생시키는 것이 좋다. 난수는 $(\text{rand()} \% 100)$ 으로 발생이 가능하다. `stdlib.h` 헤더 파일도 포함시켜야 한다.



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.

The diagram illustrates the components of a C-style for loop. A yellow rectangular box contains the code: `for (i = 0; i < 5; i++) printf("Hello World!\n");`. Above the box, three blue rectangular labels are positioned: '초기식' (Initialization) above 'i = 0;', '조건식' (Condition) above 'i < 5;', and '증감식' (Increment) above 'i++'. Each label is connected to its corresponding part of the loop by a thin blue line. Additionally, each of the three expressions 'i = 0;', 'i < 5;', and 'i++' is individually circled with a red line.

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

예제

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

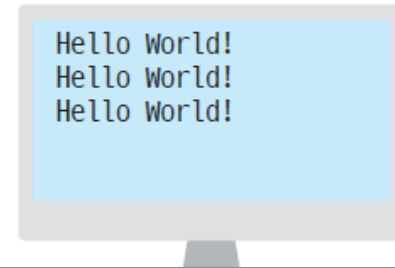
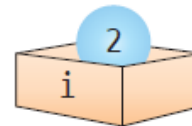
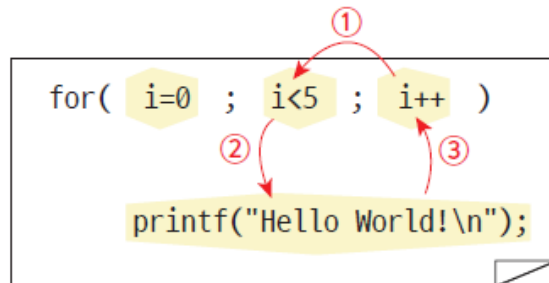
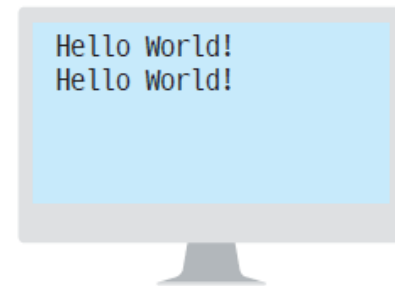
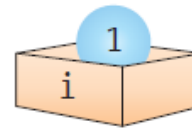
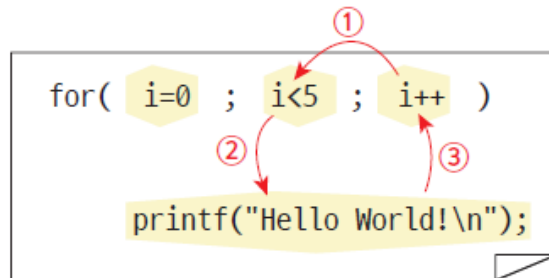
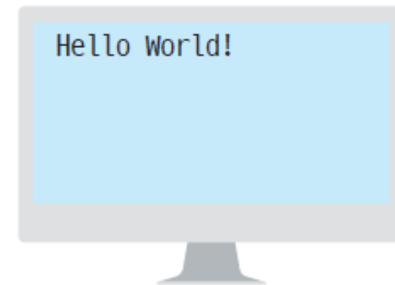
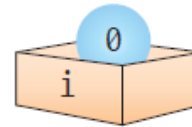
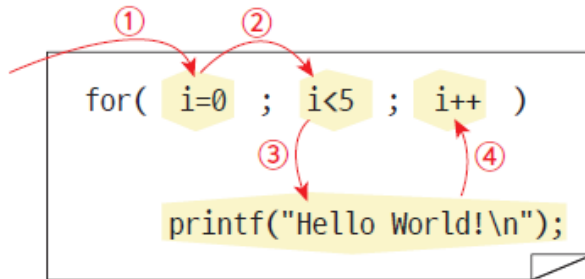
int main(void)
{
    int i;

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

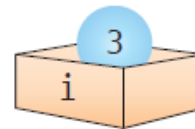
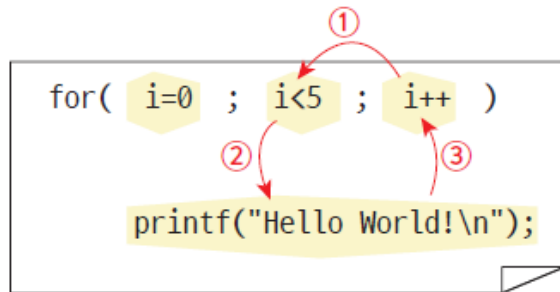
    return 0;
}
```

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

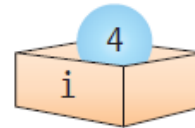
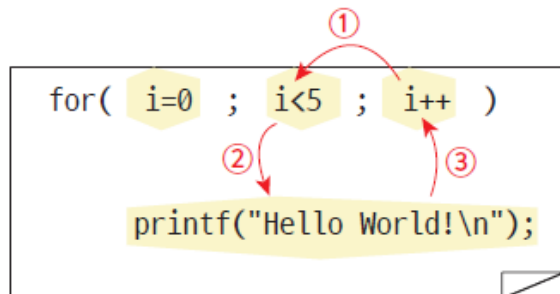
for문의 실행과정



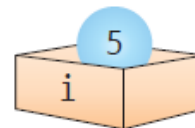
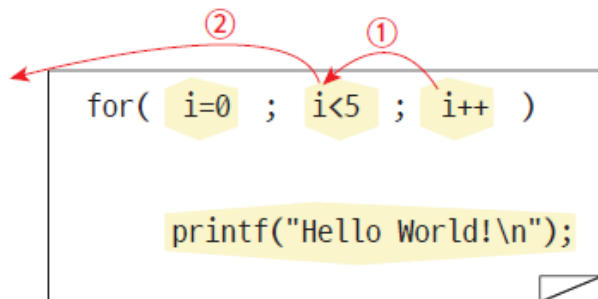
for문의 실행과정



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



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



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

예제 #2

- 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

- 이번 예제에서는 **for** 루프를 이용하여 일정 범위의 정수에 대하여 세제곱값을 구하여 보자. 즉 1의 세제곱부터 시작해서 사용자가 입력하는 수의 세제곱까지를 나열하는 프로그램을 작성해보자.

정수를 입력하시요:5

=====

i	i의 세제곱
---	--------

=====

1	1
2	8
3	27
4	64
5	125

예제 #3

```
// 반복을 이용한 세제곱값구하기
#include <stdio.h>

int main(void)
{
    int i, n;

    printf("정수를 입력하시요:");
    scanf("%d", &n);

    printf("=====\n");
    printf(" i      i의 세제곱\n");
    printf("=====\n");
    for(i = 1; i <= n; i++)
        printf("%5d    %5d\n", i, i*i*i);

    return 0;
}
```

정수를 입력하시요:5

=====

i i의 세제곱

=====

1	1
2	8
3	27
4	64
5	125

예제 #4

- 화면에 * 글자를 이용하여 다음과 같은 네모를 그려보자.



예제 #4

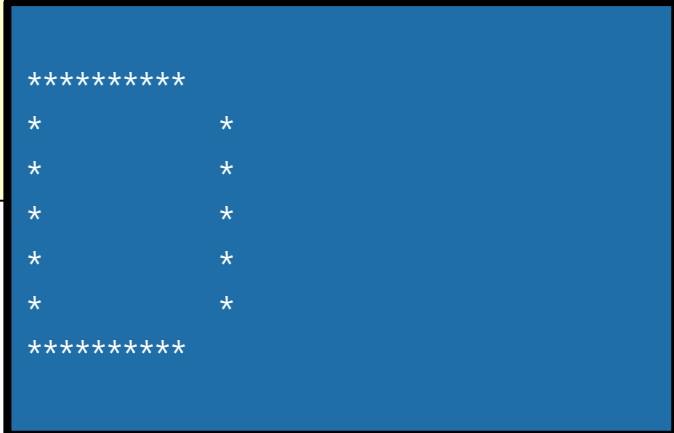
```
// 반복을 이용한 네모 그리기
#include <stdio.h>

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

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

    printf("*****");

    return 0;
}
```



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

예제 #5

- 이번 예제에서는 팩토리얼 값을 계산하여 보자. 팩토리얼이란 다음과 같이 정의된다.

$$n! = 1 \times 2 \times 3 \times \dots \times n$$

정수를 입력하세요: 10
10!은 3628800입니다.

예제 #5

```
// 반복을 이용한 팩토리얼 구하기
#include <stdio.h>

int main(void)
{
    long fact=1;
    int i, n;

    printf("정수를 입력하시요:");
    scanf("%d", &n);

    for(i = 1; i <= n; i++)
        fact = fact * i;

    printf("%d!은 %d입니다.\n",n,fact);

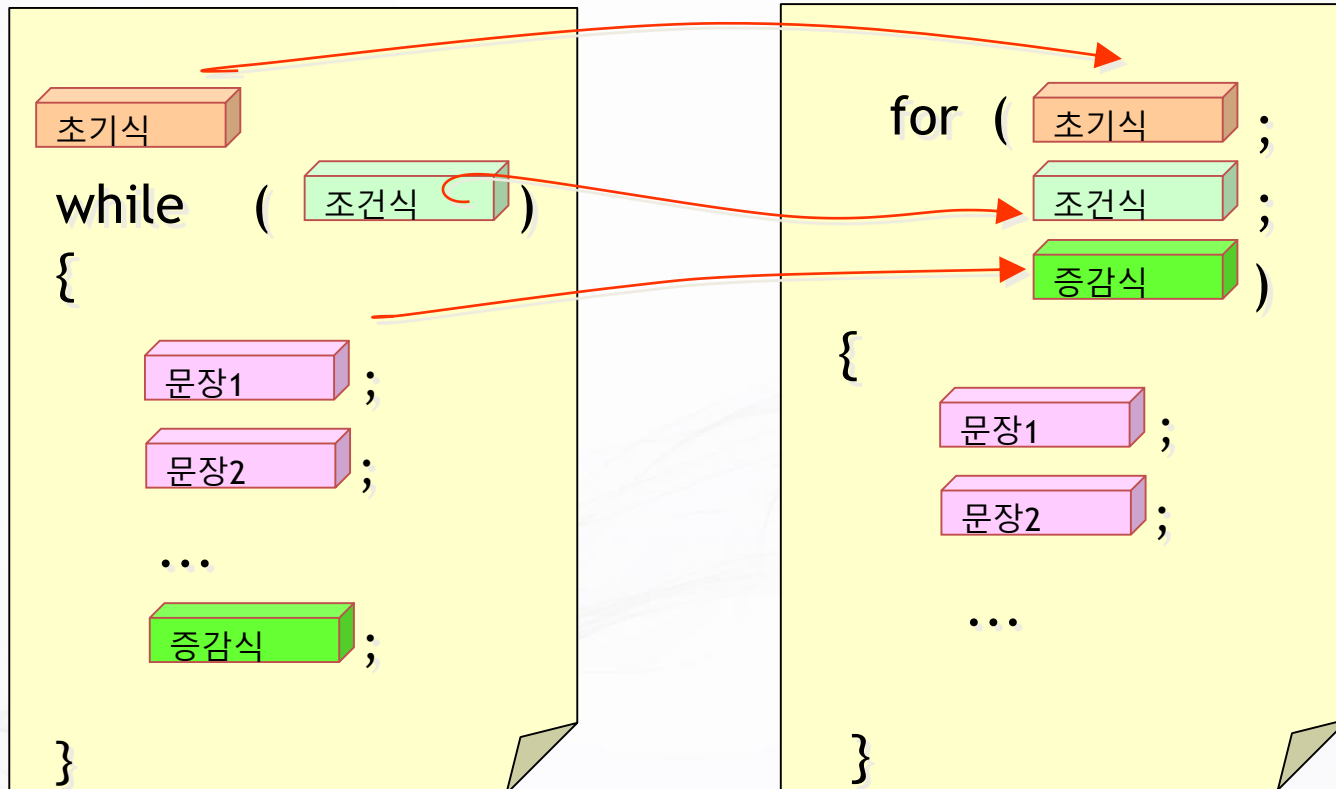
    return 0;
}
```

정수를 입력하시요: 10
10!은 3628800입니다.

예제 #5에서의 반복 진행 상황

	i의 값	$i \leq 5$	반복여부	fact의 값
1번째 반복	1	$1 \leq 5$ (참)	반복	$1 * 1$
2번째 반복	2	$2 \leq 5$ (참)	반복	$1 * 1 * 2$
3번째 반복	3	$3 \leq 5$ (참)	반복	$1 * 1 * 2 * 3$
4번째 반복	4	$4 \leq 5$ (참)	반복	$1 * 1 * 2 * 3 * 4$
5번째 반복	5	$5 \leq 5$ (참)	반복	$1 * 1 * 2 * 3 * 4 * 5$
6번째 반복	6	$6 \leq 5$ (거짓)	중단	

while 루프와 for 루프와의 관계



팩토리얼 계산 예제(while 버전)

```
// 반복을 이용한 팩토리얼 구하기
#include <stdio.h>
int main(void)
{
    long fact = 1;
    int i = 1, n;
    printf("정수를 입력하세요: ");
    scanf("%d", &n);
    while (i <= n)
    {
        fact = fact * i;
        i++;
    }
    printf("%d!은 %d입니다.", n, fact);
    return 0;
}
```

정수를 입력하세요: 10
10!은 3628800입니다.

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.

중간 점검

1. 다음 코드의 출력을 쓰시오.

```
for(i = 1; i < 5; i++)  
    printf("%d ", 2 * i);
```

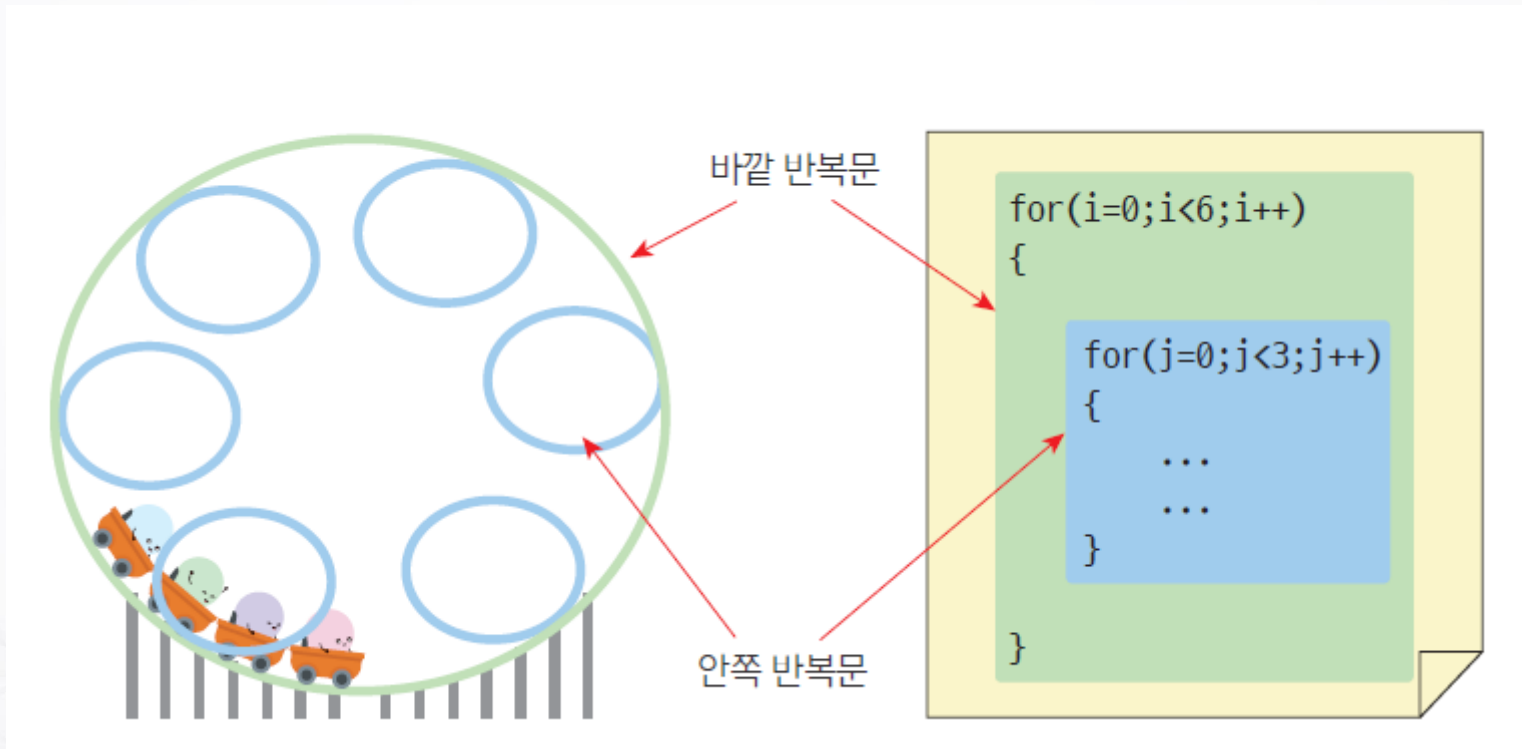
2. 다음 코드의 출력을 쓰시오.

```
for(i = 10; i > 0; i = i - 2)  
    printf("Student%d\n", i);
```



중첩 반복문

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



예제 #1

- 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;
}
```



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

중간 점검

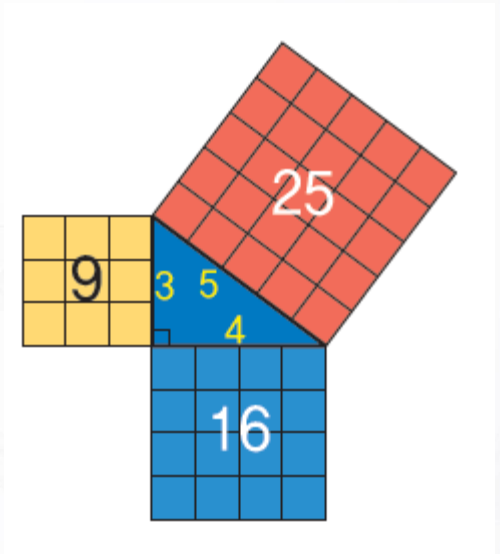
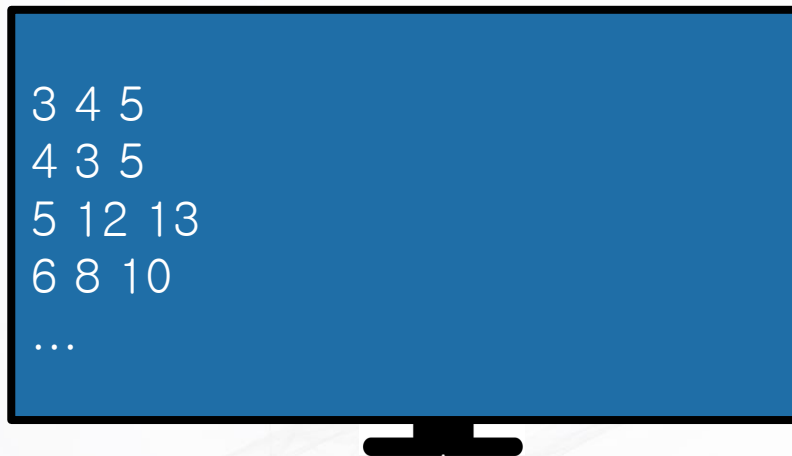
1. 다음 코드의 출력을 쓰시오.

```
for(i = 0; i < 3; i++)  
    for(j = 0; j < 3; j++)  
        printf("%d 곱하기 %d은 %d\n", i, j, i*j);
```



Lab: 직각 삼각형 찾기

- 각 변의 길이가 100보다 작은 삼각형 중에서 피타고라스의 정리가 성립하는 직각 삼각형은 몇 개나 있을까?



알고리즘


```
for(a=1;a<=100;a++)  
    for(b=1;b<=100;b++)  
        for(c=1;c<=100;c++)  
            if( a*a + b*b == c*c )  
                a와 b와 c를 화면에 출력한다.
```

소스

```
#include <stdio.h>
int main(void)
{
    int a, b, c;
    for(a=1; a<=100; a++) {
        for(b=1; b<=100; b++) {
            for(c=1; c<=100; c++) {
                if( (a*a+b*b)==c*c )
                    printf("%d %d %d\n", a, b, c);
            }
        }
    }
    return 0;
}
```

도전문제

- 위의 문제의 실행 결과를 자세히 보면 (3, 4, 5), (4, 3, 5), (5, 3, 4)와 같이 동일한 삼각형이 되풀이하여 출력되는 것을 알 수 있다. (3, 4, 5)와 같은 삼각형이 한번만 출력되게 하려면 소스의 어떤 부분을 수정하여야 할까?



```
3 4 5  
5 12 13  
6 8 10  
7 24 25  
8 15 17  
9 12 15  
...
```

Solution

```
#include <stdio.h>
int main(void)
{
    for(int a=1; a<=100; a++)
        for(int b=a; b<=100; b++)
            for(int c=b; c<=100; c++)
                if( (a*a+b*b)==c*c )
                    printf("%d %d %d\n", a, b, c);
    return 0;
}
```


도전문제

- 위와 비슷한 문제를 하나 더 작성해보자. 라스베가스와 같은 도박장에 가면 주사위 게임이 있다. 주사위 2개를 던졌을 때, 합이 6이 되는 경우를 전부 출력하여 보자. 예를 들어서 (1, 5), (2, 4),...와 같이 출력되면 된다. 또 주사위 3개를 사용하여서 합이 10이 되는 경우를 전부 출력하여 보자.



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.

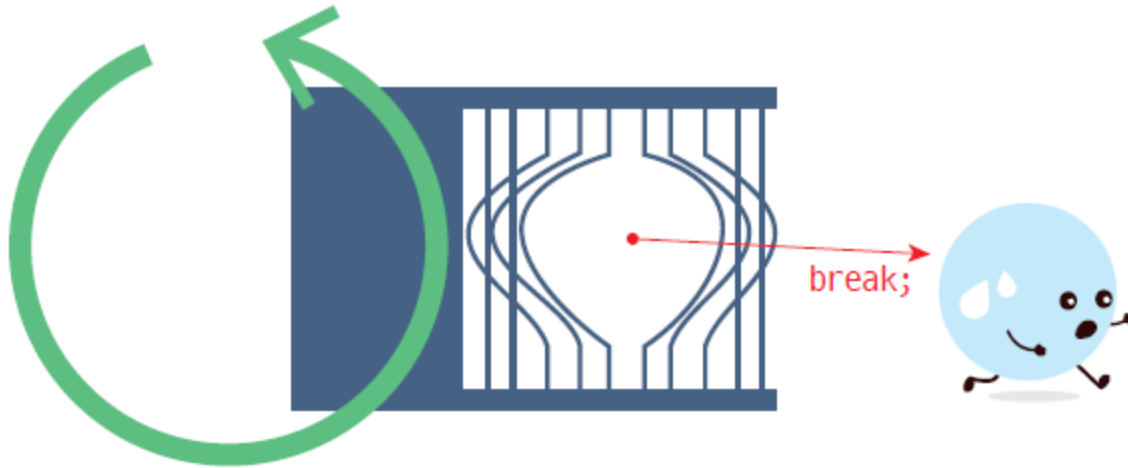
```
while((x % 3 != 0) && (x >= 0)) {  
    ...  
    ...  
    ...  
}
```



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

break 문

- break 문은 반복 루프를 빠져 나오는데 사용된다.



예제

- 예를 들어서 100만원으로 재테크를 시작한 사람이 1년에 30%의 수익을 얻는다면 몇 년 만에 원금의 10배가 되는지를 계산하여 보자.
- 이런 경우에는 무한 반복 구조를 사용하고 조건이 만족되었을 때 break문이 실행되도록 하면 좋다.



예제

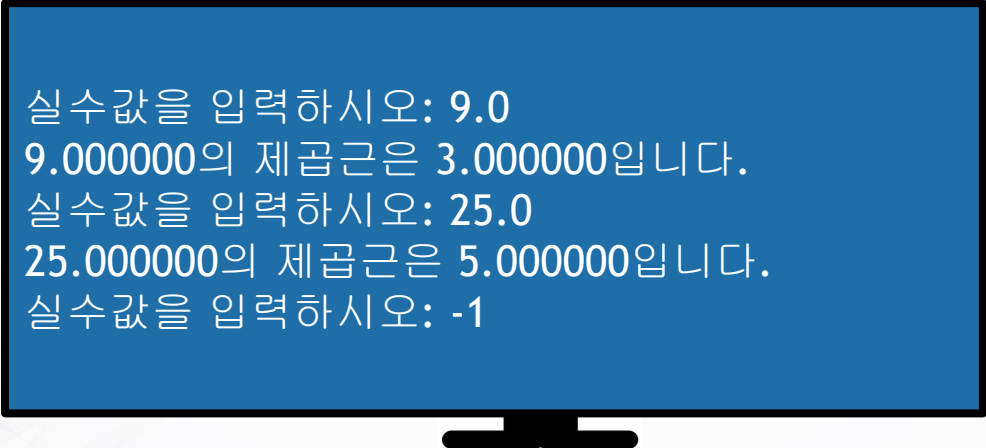
```
#include <stdio.h>
#define SEED_MONEY 1000000

int main(void)
{
    int year=0, money=SEED_MONEY;
    while(1)
    {
        year++;
        money += money*0.30;
        if( money > 10*SEED_MONEY )
            break;
    }
    printf("%d", year);
    return 0;
}
```

원금의 10배가 되면

예제

- 여기서는 무한 루프를 만들어서 사용자로 부터 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.

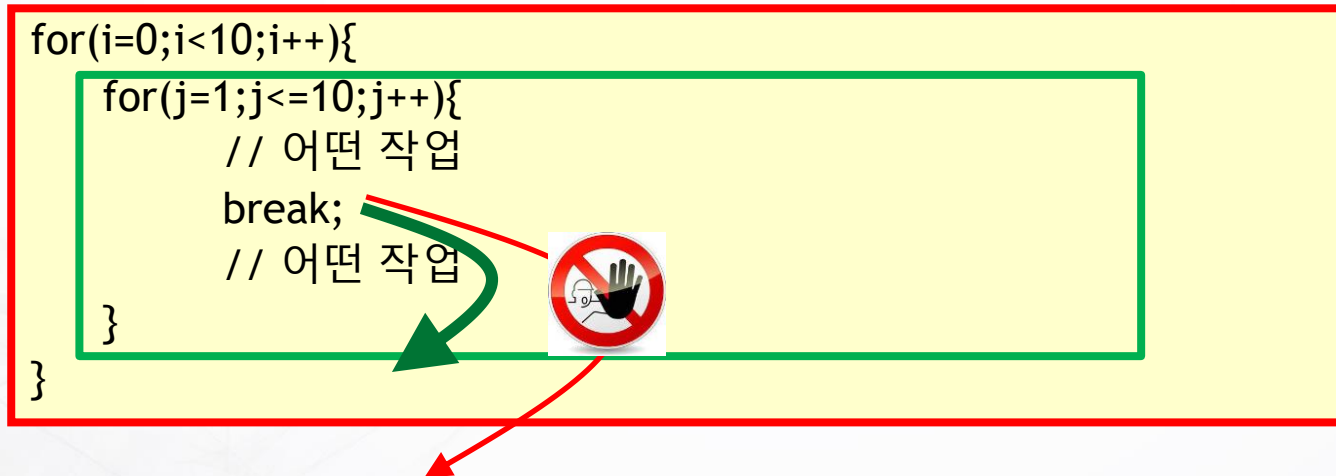
실수값을 입력하시오: 25.0

25.000000의 제곱근은 5.000000입니다.

실수값을 입력하시오: -1

goto문이 필요한 유일한 경우

- 중첩 루프 안에서 어떤 문제가 발생했을 경우, **goto**를 이용하면 단번에 외부로 빠져 나올 수 있다.
- **break**를 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.

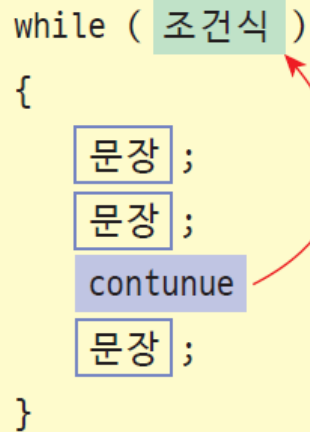
```
#include <stdio.h>

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

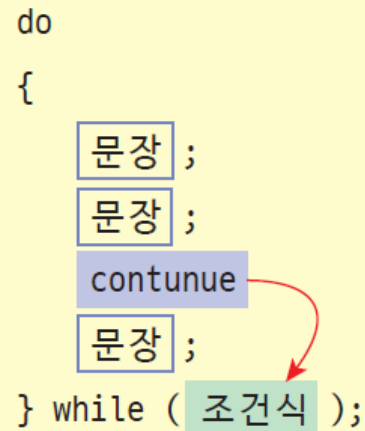
1 2 4 5 7 8

continue 문

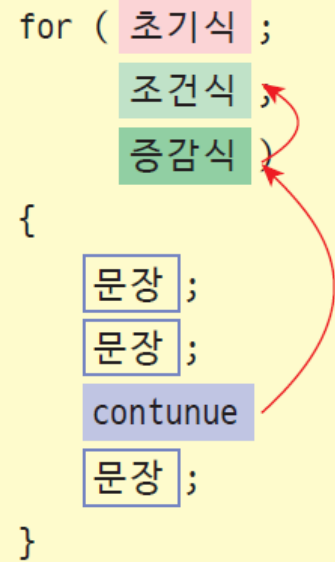
```
while ( 조건식 )  
{  
    문장 ;  
    문장 ;  
    contunue  
    문장 ;  
}
```



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

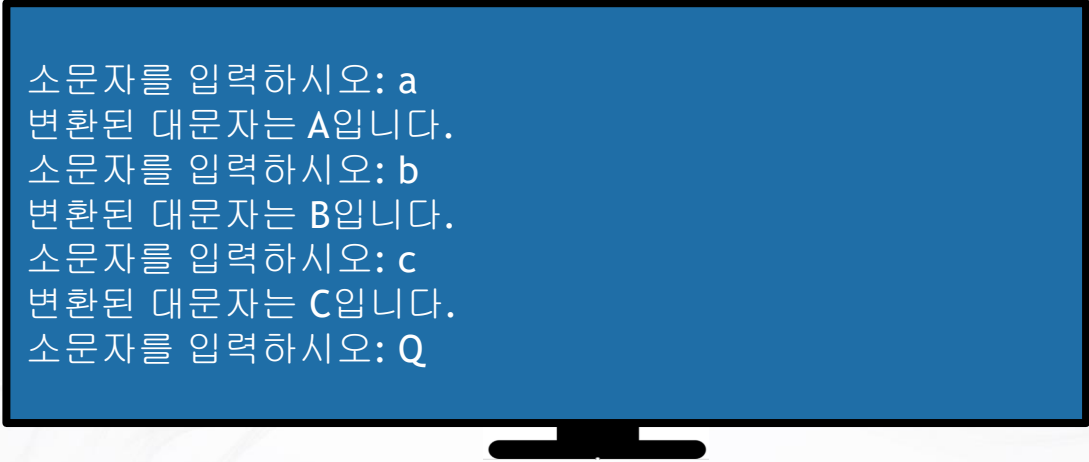


```
for ( 초기식 ;  
      조건식 ;  
      증감식 )  
{  
    문장 ;  
    문장 ;  
    contunue  
    문장 ;  
}
```



예제

- 사용자로부터 알파벳 소문자를 받아서 대문자로 바꾸는 다음의 프로그램을 살펴보자. 만약 사용자로부터 받은 문자가 소문자가 아니면 사용자로부터 다시 문자를 입력받는다.



소문자를 입력하시오: a
변환된 대문자는 A입니다.
소문자를 입력하시오: b
변환된 대문자는 B입니다.
소문자를 입력하시오: c
변환된 대문자는 C입니다.
소문자를 입력하시오: Q

예제 #2

```
// 소문자를 대문자로 변경한다.
#include <stdio.h>

int main(void)
{
    char letter;

    while(1)
    {
        printf("소문자를 입력하시오: ");
        scanf(" %c", &letter);

        if( letter == 'Q' )
            break ;
        if( letter < 'a' || letter > 'z' )
            continue ;

        letter -= 32;
        printf("변환된 대문자는 %c입니다.\n", letter);
    }

    return 0;
}
```

중간 점검

1. _____ 문이 반복문에서 실행되면 현재의 반복을 중단하고 다음번 반복 처리가 시작된다.
2. _____ 문이 반복문에서 실행되면 반복문을 빠져 나온다.
3. 다음 코드의 출력을 쓰시오.

```
int i;  
for(i = 1; i < 10; i++) {  
    if( i % 3 == 0 ) break;  
    printf("%d\n", i);  
}
```

4. 3번 문제에서 **break**를 **continue**로 변경하면 어떻게 되는가?



Lab: 파이 구하기

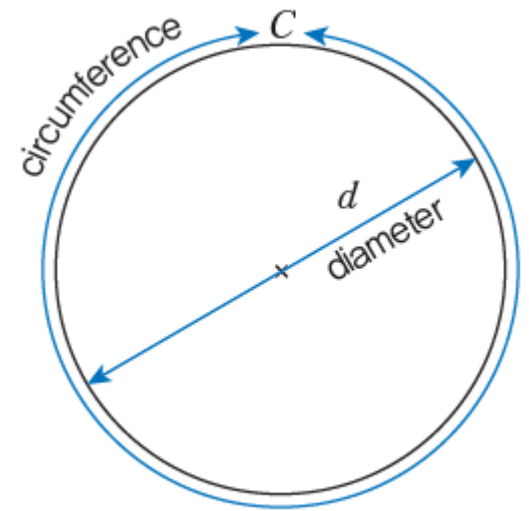
- 파이를 계산하는 가장 고전적인 방법은 Gregory-Leibniz 무한 수열을 이용하는 것

$$\pi = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \dots$$

반복횟수:100000

Pi = 3.141583

계속하려면 아무 키나 누르십시오 . . .



알고리즘

사용자로부터 반복횟수 `loop_count`를 입력받는다.

```
분자 = 4.0;
```

```
분모 = 1.0;
```

```
sum = 0.0;
```

```
while(loop_count > 0)
```

```
    sum = sum + 분자 / 분모;
```

```
    분자 = -1.0* 분자;
```

```
    분모 = 분모 + 2.0;
```

```
    --loop_count;
```

`sum`을 출력한다.

Solution

```
#include <stdio.h>

int main(void)
{
    double divisor, dividend, sum;
    int loop_count;

    divisor = 1.0;
    dividend = 4.0;
    sum = 0.0;
    printf("반복횟수:");
    scanf("%d", &loop_count);

    while(loop_count > 0) {
        sum = sum + dividend / divisor;
        dividend = -1.0 * dividend;
        divisor = divisor + 2;
        loop_count--;
    }
    printf("Pi = %f", sum);
    return 0;
}
```

Lab: 복리 이자 계산

- 복리는 우리가 알다시피 이자 계산 방법의 하나로서 일정 기간마다 이자를 원금에 더해 이것을 새로운 원금으로 계산하는 방법이다.

원금: 1000000

이율(%): 5

기간(년): 10

=====

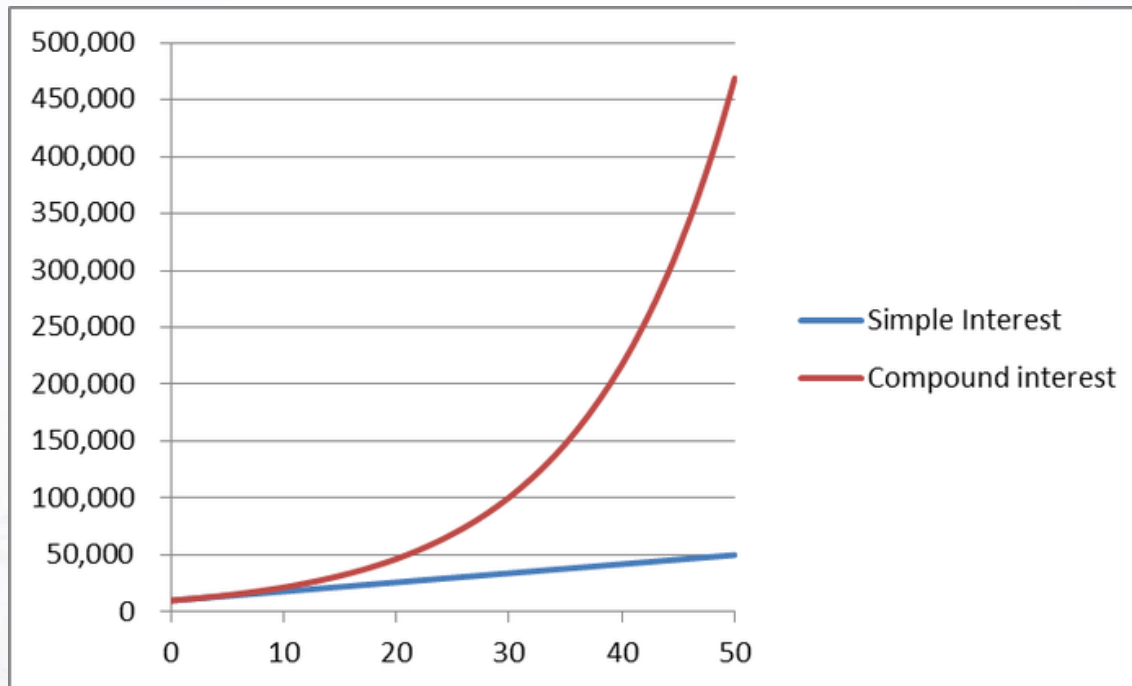
연도 원리금

=====

1	1050000.0
2	1102500.0
3	1157625.0
4	1215506.2
5	1276281.6
6	1340095.6
7	1407100.4
8	1477455.4
9	1551328.2
10	1628894.6



복리에서 원리금 합계



복리에서 원리금 합계


```
#include <stdio.h>
int main(void)
{
    int i, years;
    double total, rate, investment;

    printf("원금: ");
    scanf("%lf", &investment);
    printf("이율(%): ");
    scanf("%lf", &rate);
    printf("기간(년): ");
    scanf("%d", &years);

    printf("=====\n");
    printf("연도 원리금\n");
    printf("=====\n");
    total = investment;
    rate /= 100.0;
    for(i = 0; i < years; i++)
    {
        total = total * ( 1 + rate );    // 새로운 원리금 계산
        printf("%2d      %10.1f\n", i+1, total);
    }
    return 0;
}
```

Lab: 자동으로 수학문제 생성하기

- 초등학생용 수학 문제 10개를 자동으로 출제하는 프로그램을 작성해보자.



3 + 7 = 10
맞았습니다.
9 + 3 = 12
맞았습니다.
8 + 3 = _

난수 발생

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

int main(void)
{
    srand(time(NULL));
    for (int i = 0; i < 10; i++)
        printf(" %d \n", rand());
}
```



```
32173
20715
3647
23562
8327
19429
6136
24360
1419
25594
```

Solution

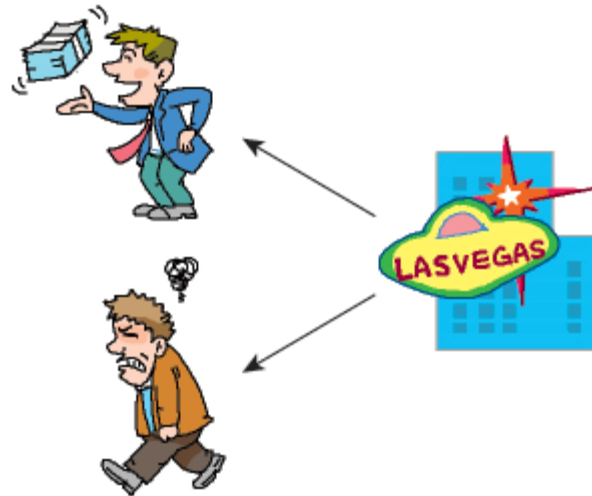
```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int x, y, answer, i;
    srand(time(NULL));

    for (i = 0; i < 10; i++) {
        x = rand() % 10;
        y = rand() % 10;
        printf("%d + %d = ", x, y);
        scanf("%d", &answer);
        if (x + y == answer)
            printf("맞았습니다.\n");
        else
            printf("틀렸습니다.\n");
    }
    return 0;
}
```


Lab: 도박사의 확률

- 어떤 사람이 50달러를 가지고 라스베가스에서 슬롯 머신 게임을 한다고 하자. 한 번의 게임에 1달러를 건다고 가정하자. 돈을 딸 확률은 0.5이라고 가정하자(현실과는 많이 다르다). 라스베가스에 가면, 가진 돈을 다 잃거나 목표 금액인 250달러에 도달할 때까지 게임을 계속한다 (while 루프가 생각나지 않은가?). 어떤 사람이 라스베가스에 100번을 갔다면 몇 번이나 250달러를 따서 돌아올수 있을까? 게임의 확률은 절반이지만 게임을 진행하다 보면 확률 분포의 극단까지도 갈 수 있다.



Lab:도박사의 확률

초기 금액 \$50
목표 금액 \$250
100번 중에서 20번 성공

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

```
int main(void)
{
    int initial_money = 50;
    int goal = 250;
    int i;
    int wins = 0;

    srand((unsigned)time(NULL));
    for (i = 0; i < 100; i++) {
        int cash = initial_money;
        while (cash > 0 && cash < goal) {
            if ((double)rand()/RAND_MAX < 0.5) cash++;
            else cash--;
        }
        if (cash == goal) wins++;
    }

    printf("초기 금액 %d \n", initial_money);
    printf("목표 금액 %d \n", goal);
    printf("100번 중에서 %d번 성공\n", wins);
    return 0;
}
```

Q & A

