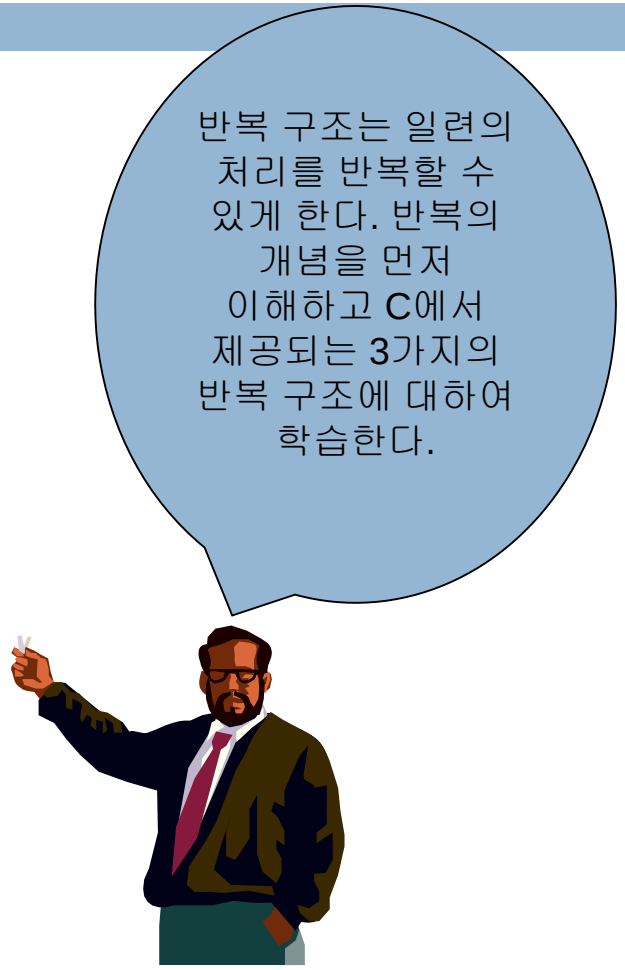
C 언어 EXPRESS(개정3판)

제 5장 반복문

이번 장에서 학습할 내용

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



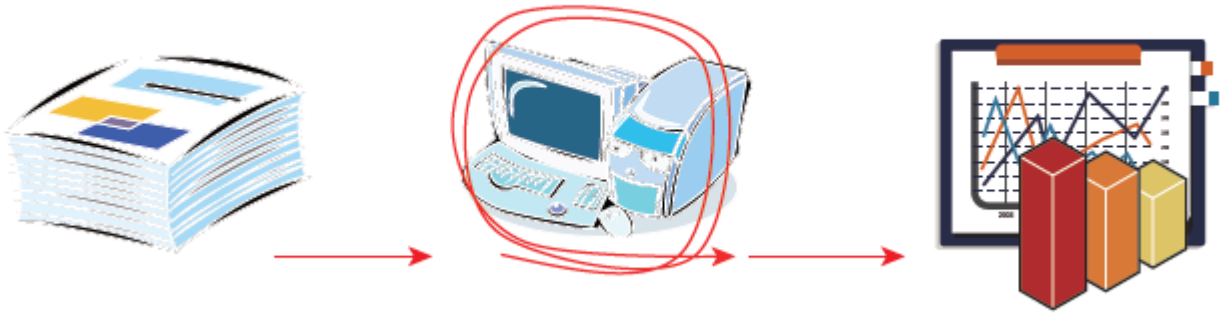
반복

- 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.



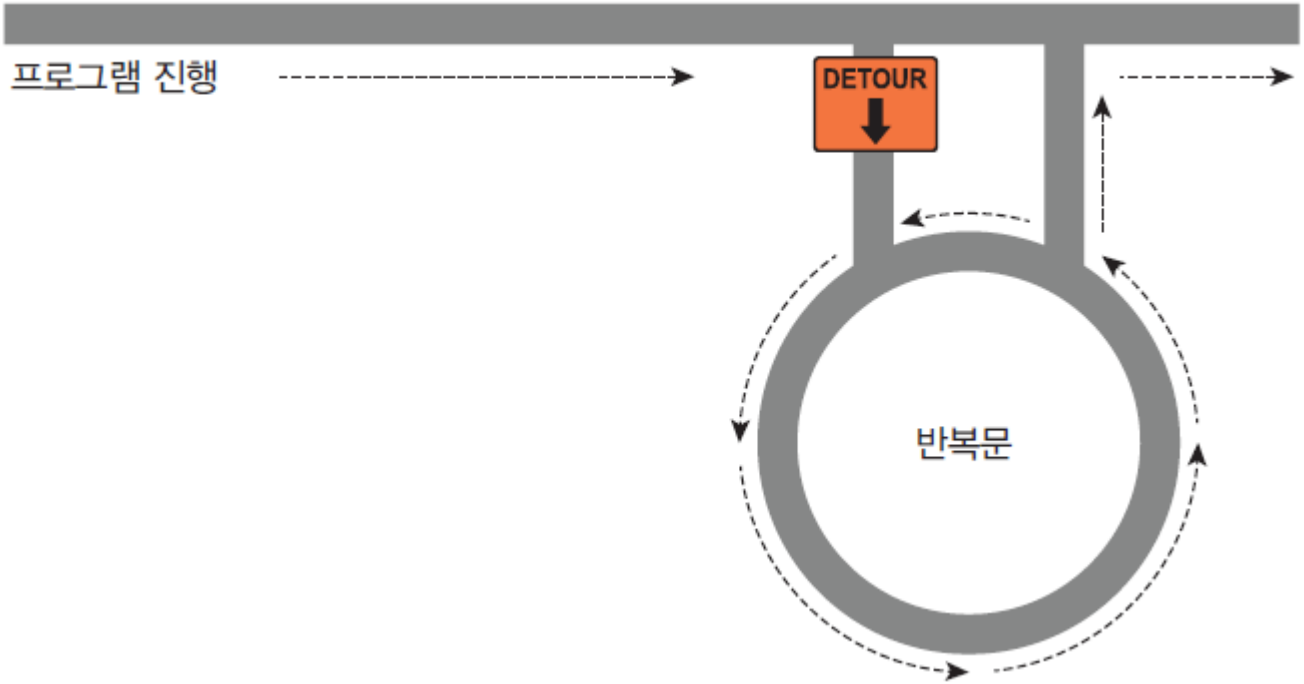
왜 반복이 중요한가?

```
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



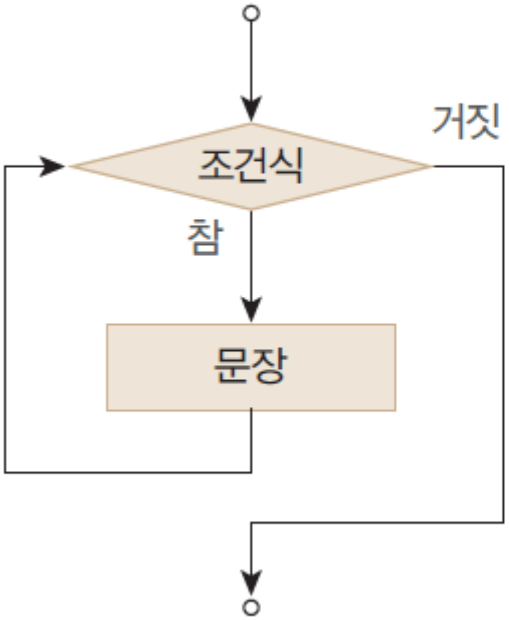
반복문의 종류

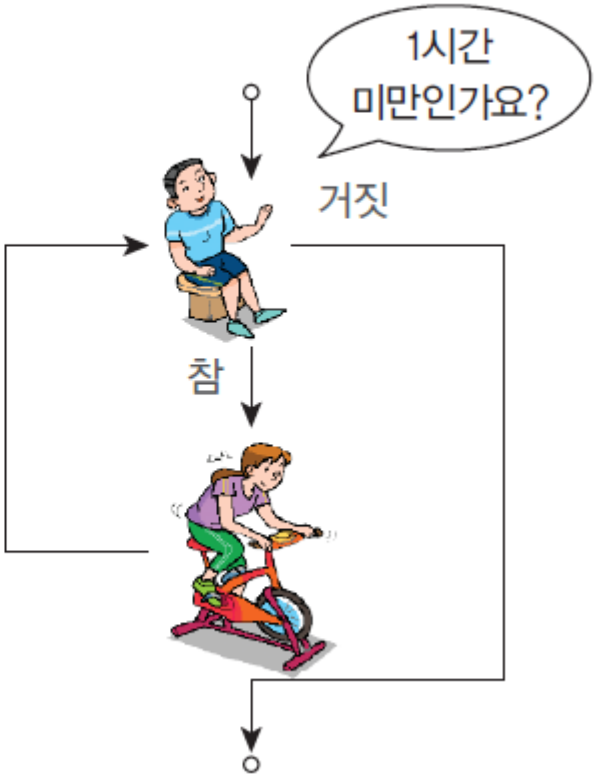
while 루프

for 루프

while 문

- 주어진 조건이 만족되는 동안 문장들을 반복 실행한다.





while 문

형식

```
while (조건식)
{
        실행문;
        .
        .
        실행문;
}
```

코드 블록

Syntax: while 문

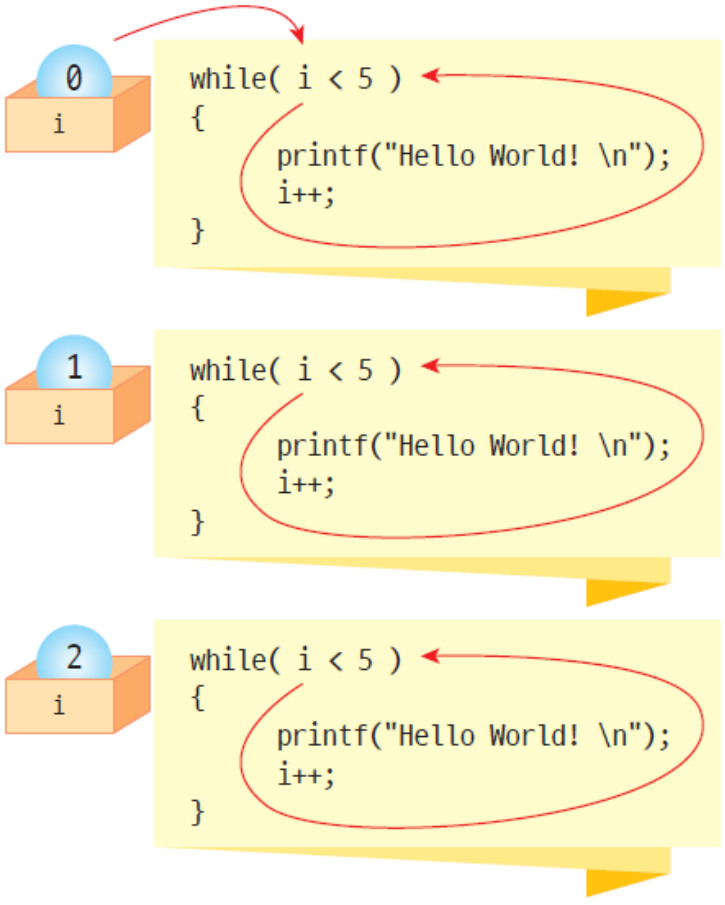
예

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

조건식

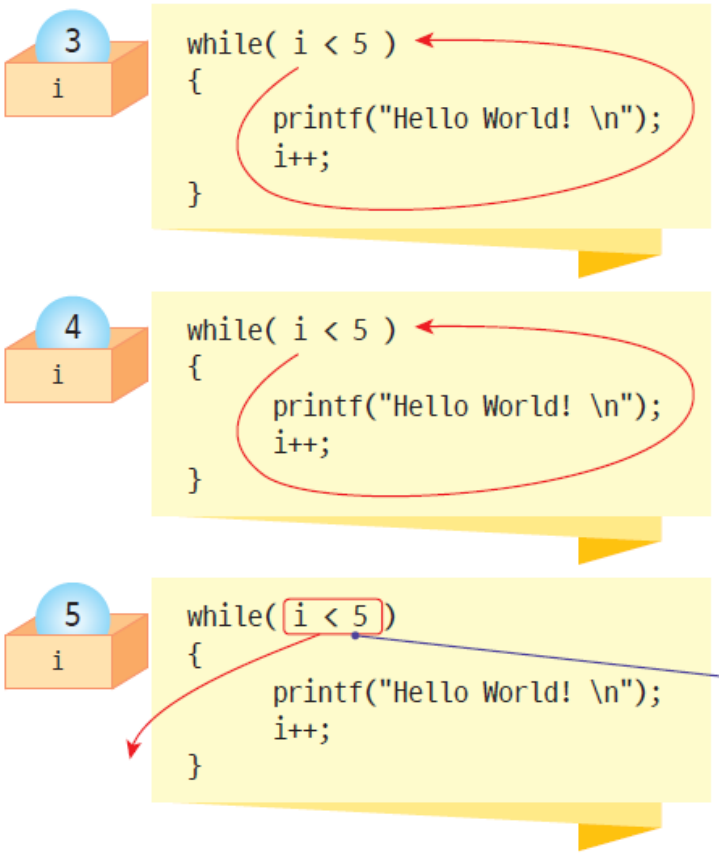
조건식이 참이면 문장을 반복 실행한다.

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



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

ch05_ex2.c

```c
// 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
```

if 문과 while 문의 비교

```
if( 조건 )
{
 …
 …
}
```

조건이 만족되면 한번만 실행된다.

```
while( 조건 )
{
 …
 …
}
```

조건이 만족되면 여러 번 반복 실행된다.

while 문에서 주의할 점

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

반복 루프에 포함되어 있지 않다.

주의

오류 주의

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

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

하나의 문장으로 취급되어서 이것만 반복된다.

반복되지 않는다.

```
while(i = 2)
{
    ...
}
```

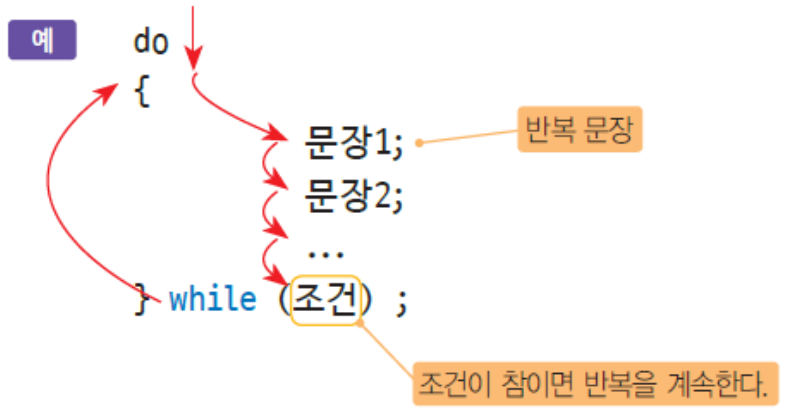
수식의 값이 2이므로 항상 참이 되어서 무한 루프

do...while문

코드 블록

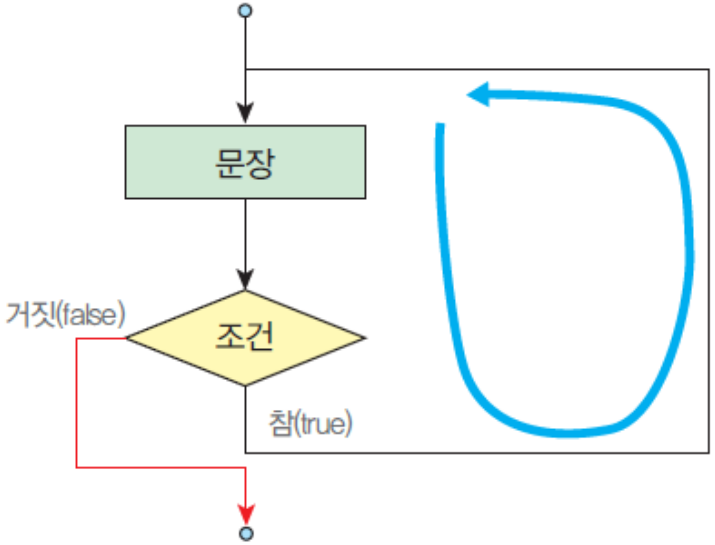
```
do {
실행문;
.
.
실행문;

} while (조건식);
```

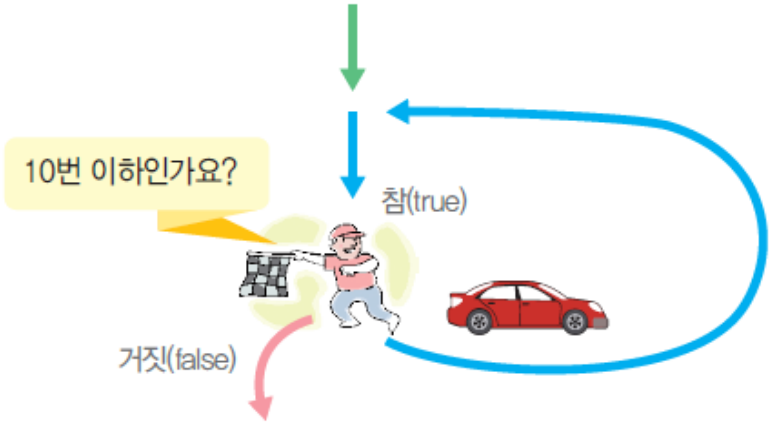
Syntax: do...while문



do-while 문

- 적어도 한번은 반복문장을 실행한다.





예제

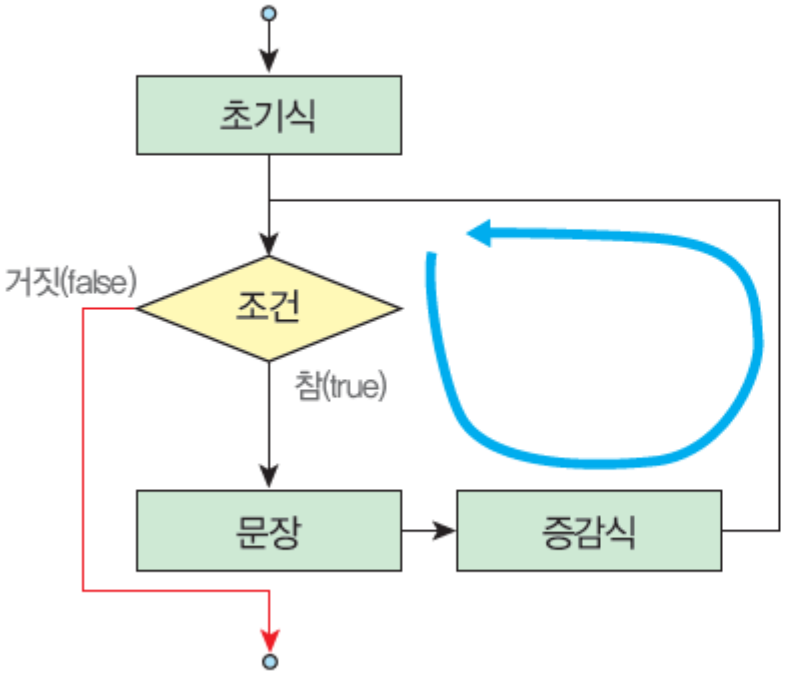
ch05_ex3.c

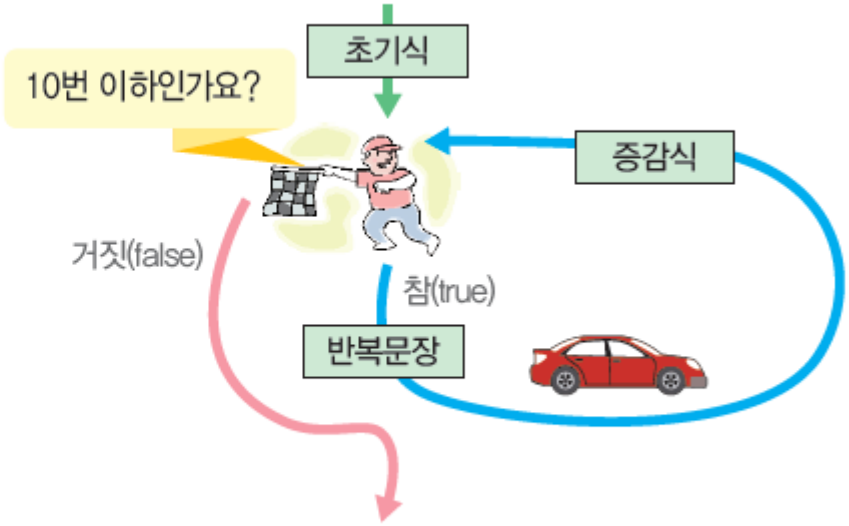
```
#include <stdio.h>

int main(void)
{
	int i = 10;

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

i=10

for 루프

- 정해진 횟수만큼 반복하는 구조





for 문의 구조

형식

for (초기식; 조건식; 증감식)

```
{
        실행문;
        .
            .
        실행문;
}
```

코드블럭

Syntax: for문

초기식 조건식 증감식

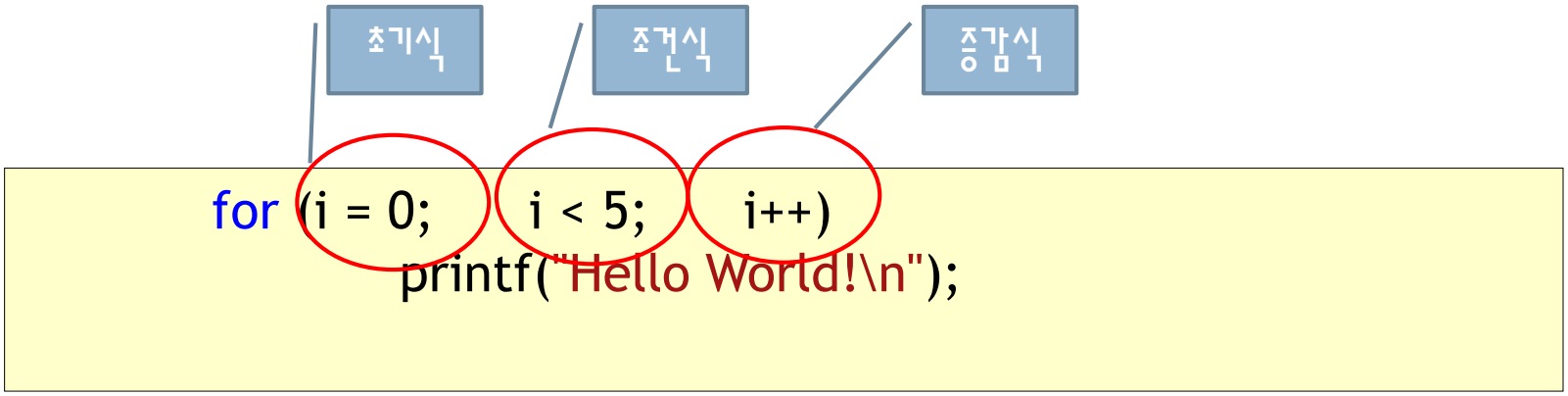
예

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

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.



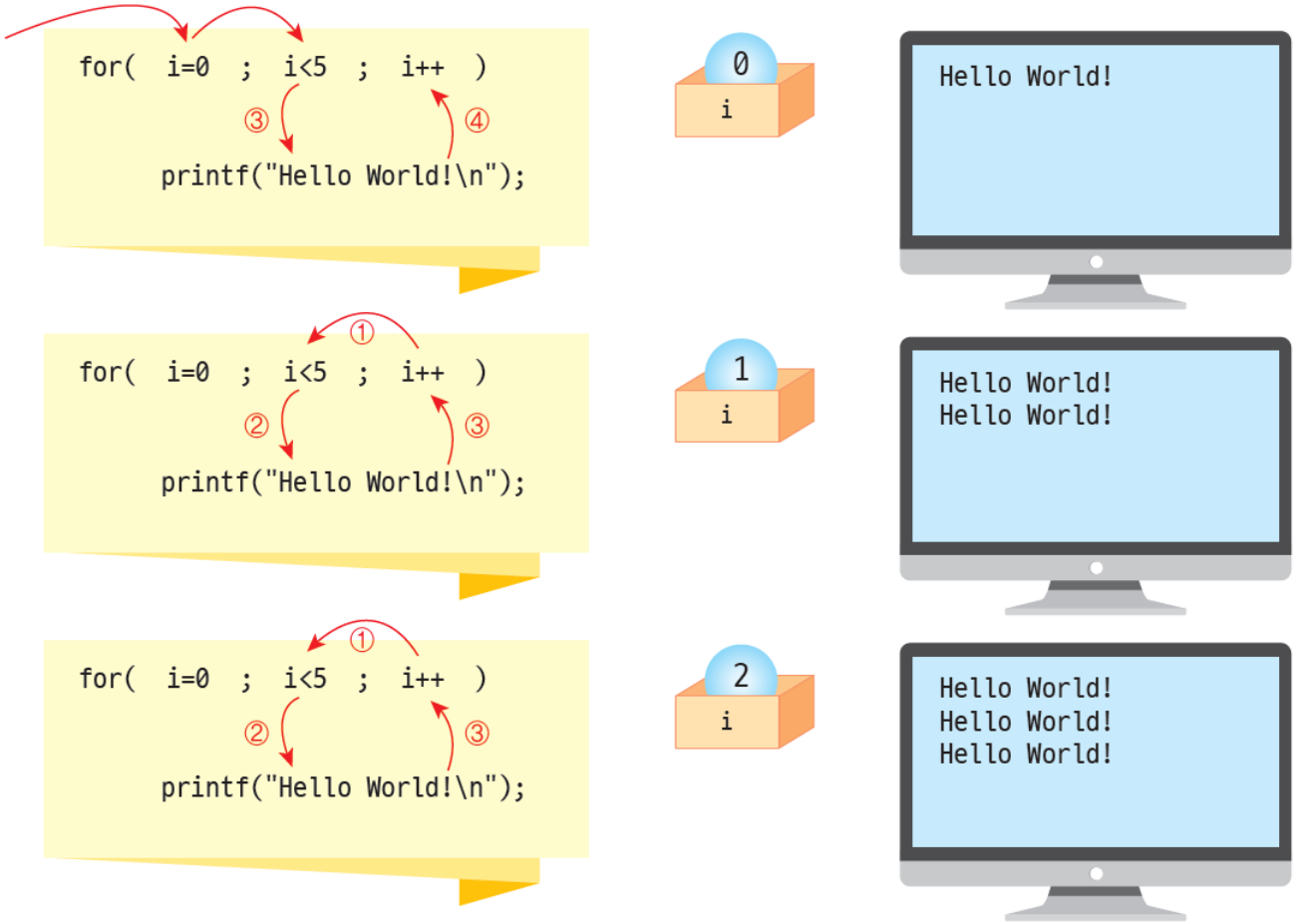
예제

ch05_ex4.c

```
// “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!
```

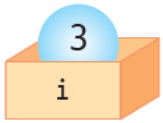
for문의 실행과정



for문의 실행과정

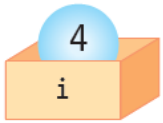
```
for(  i=0  ;  i<5  ;  i++  )
      printf("Hello World!\n");
```

① ② ③



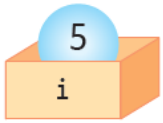

```
for(  i=0  ;  i<5  ;  i++  )
      printf("Hello World!\n");
```

① ② ③




```
for(  i=0  ;  i<5  ;  i++  )

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

② ①



예제

ch05_ex5.c

```c
// 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
```

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 회수는 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");
```

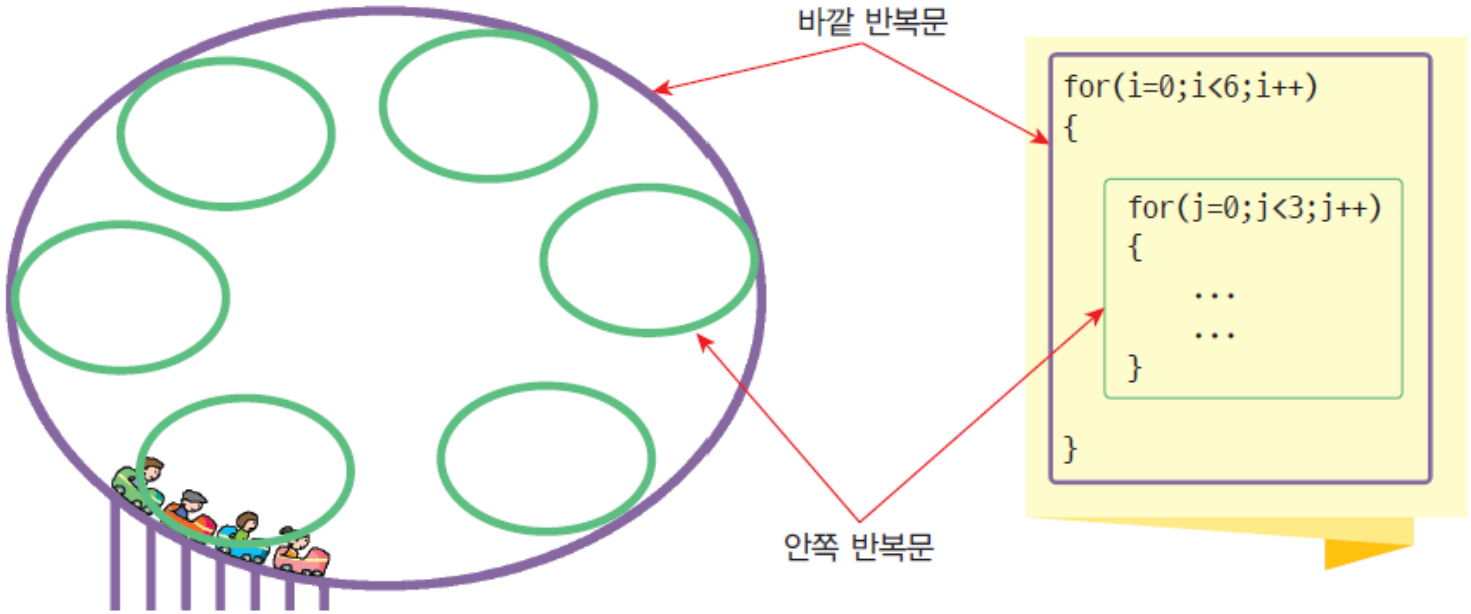
어떤 수식도 가능

```
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

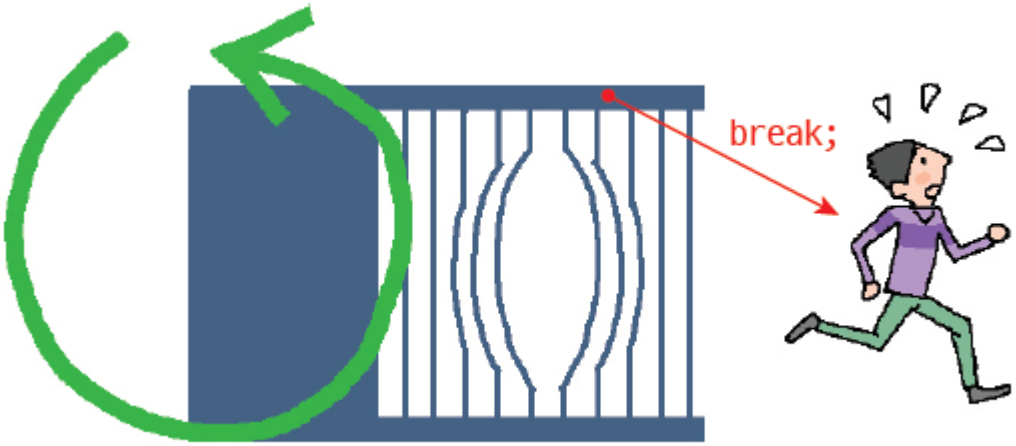
// 구구단 출력 프로그램

ch05_ex6.c

```
#include <stdio.h>
int main(void)
{
    int i = 0, j = 0, res=0;

    for (i = 2; i < 9; i++)
    {
        for (j = 1; j < 9; j++)
        {
            res = i * j;
            printf("%d x %d = %d \t",i, j, res);
        }
        printf("\n");
    }
    return 0;
}
```

break 문

- break 문은 반복 루프를 빠져 나오는데 사용된다.



예제

- 1부터 10까지의 수들의 총합을 구할 경우, 총합의 값이 30이상이 되면 총합을 구하는 것을 멈추고 30이상의 총합의 값과 마지막에 더해진 수를 출력
- 이런 경우에는 무한 반복 구조를 사용하고 조건이 만족되었을 때 break문이 실행되도록 하면 좋다.

예제

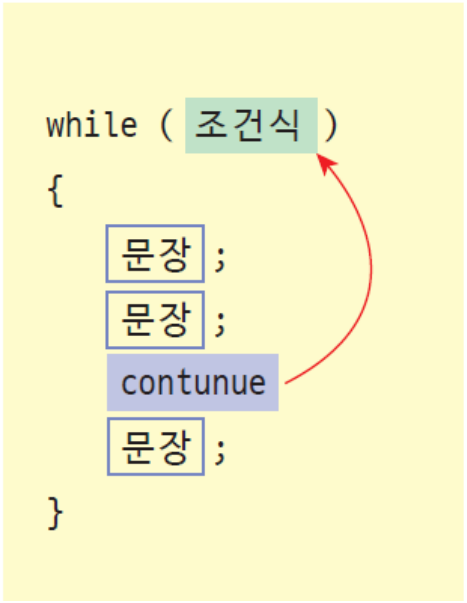
ch05_ex8.c

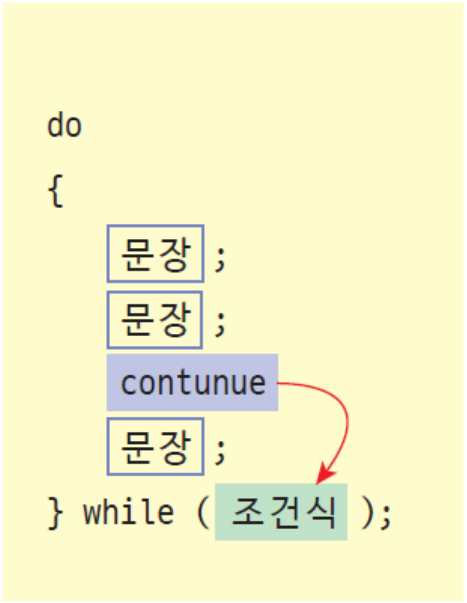
```
#include <stdio.h>

int main(void)
{
        int i = 0, n=0, n_sum = 0;
        int n_limit = 30;

        for (i = 0; i <= 10; i++)
        {
                n_sum = n_sum + i;
                if (n_sum > n_limit)
                {
                        printf("%d\n", i);
                        printf("%d\n", n_sum);
                        break;
                }
        }
}
```

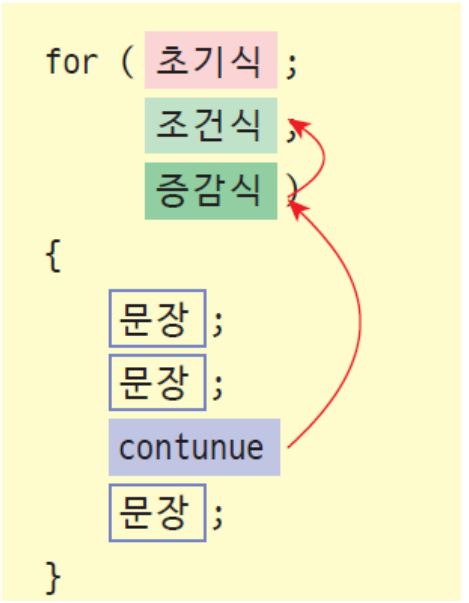
```
8
36
```

continue 문

- 반복문의 코드블록에서 continue 를 만나면 조건식이나 증감식으로 점프한다.







continue 문

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

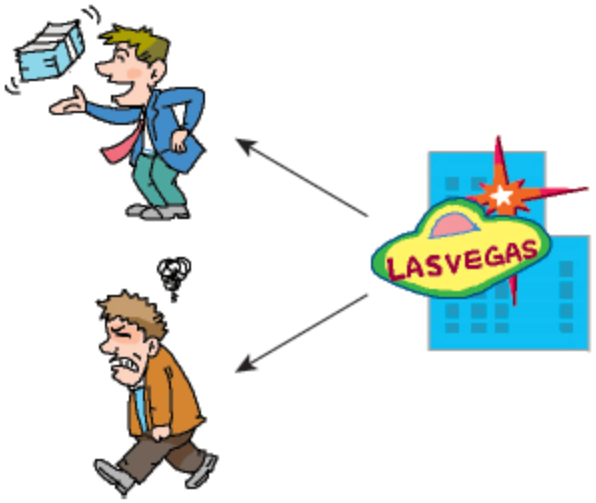
ch05_ex7.c

```c
#include <stdio.h>
int main(void)
{
    int i;

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

continue 문을 만나면 다음 반복을 즉시 시작한다.

Lab1: 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?



Lab1:도박사의 확률

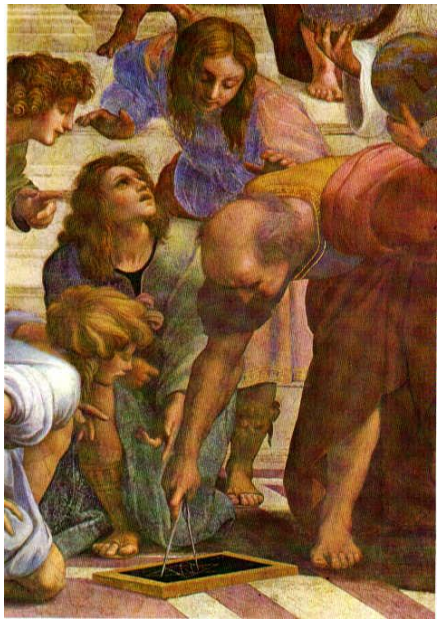
초기 금액 $50
목표 금액 $250
100번 중에서 20번 성공

Lab1-sol

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

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

	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;
}
```

lab1: 최대 공약수 찾기

두개의 정수를 입력하시오(큰수, 작은수): 12 8
최대 공약수는 4입니다.



lab: 최대 공약수 찾기

- 유클리드 알고리즘

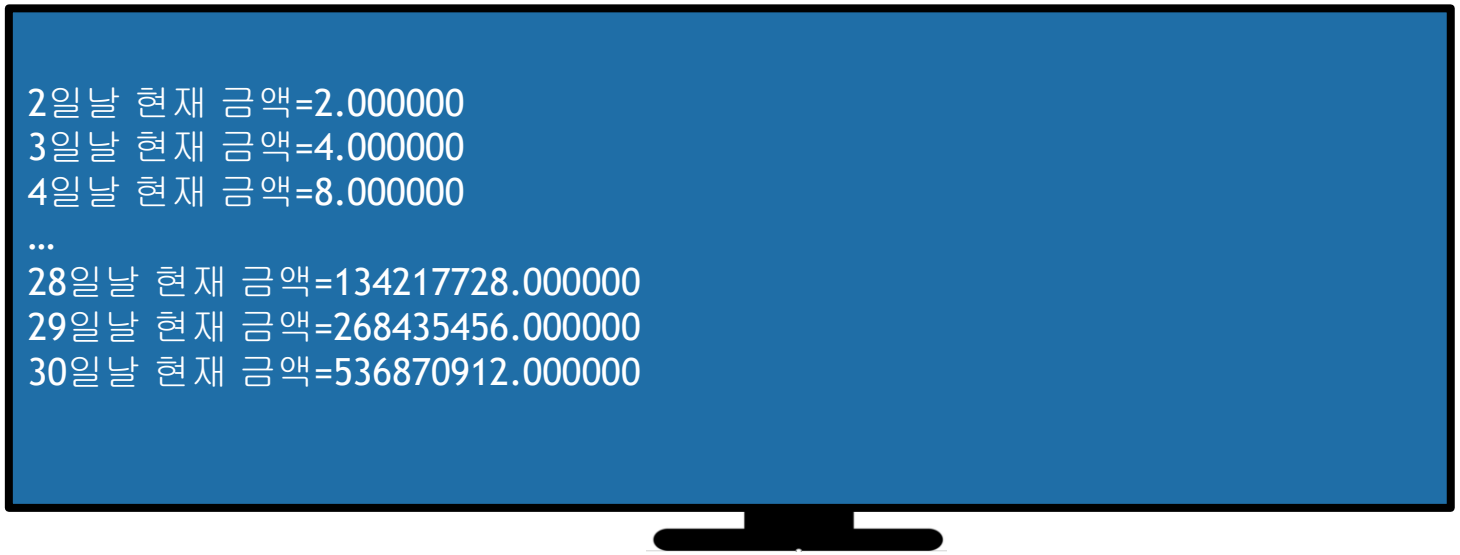
① 두 수 가운데 큰 수를 x, 작은 수를 y라 한다.
② y가 0이면 공약수는 x와 같다.
③ r ← x % y
④ x ← y
⑤ y ← r
⑥ 단계 ②로 되돌아간다.

```c
#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): 12 8
최대 공약수는 4입니다.

Lab2: 복리의 무서움

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



Lab2-Sol:

```c
#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;
}
```

lab3: 숫자 추측 게임

- 프로그램이 가지고 있는 정수를 사용자가 알아맞히는 게임

```
정답을 추측하여 보시오: 10
제시한 정수가 낮습니다.
정답을 추측하여 보시오: 30
제시한 정수가 낮습니다.
정답을 추측하여 보시오: 60
제시한 정수가 높습니다.
정답을 추측하여 보시오: 59
축하합니다. 시도횟수=4
```

알고리즘

```
do
        사용자로부터 숫자를 guess로 입력받는다.
        시도횟수를 증가한다.
        if( guess < answer )
                숫자가 낮다고 출력한다.
        if( guess > answer )
                숫자가 높다고 출력한다.
while(guess != answer);
"축하합니다"와 시도횟수를 출력한다.
```

Lab3-sol

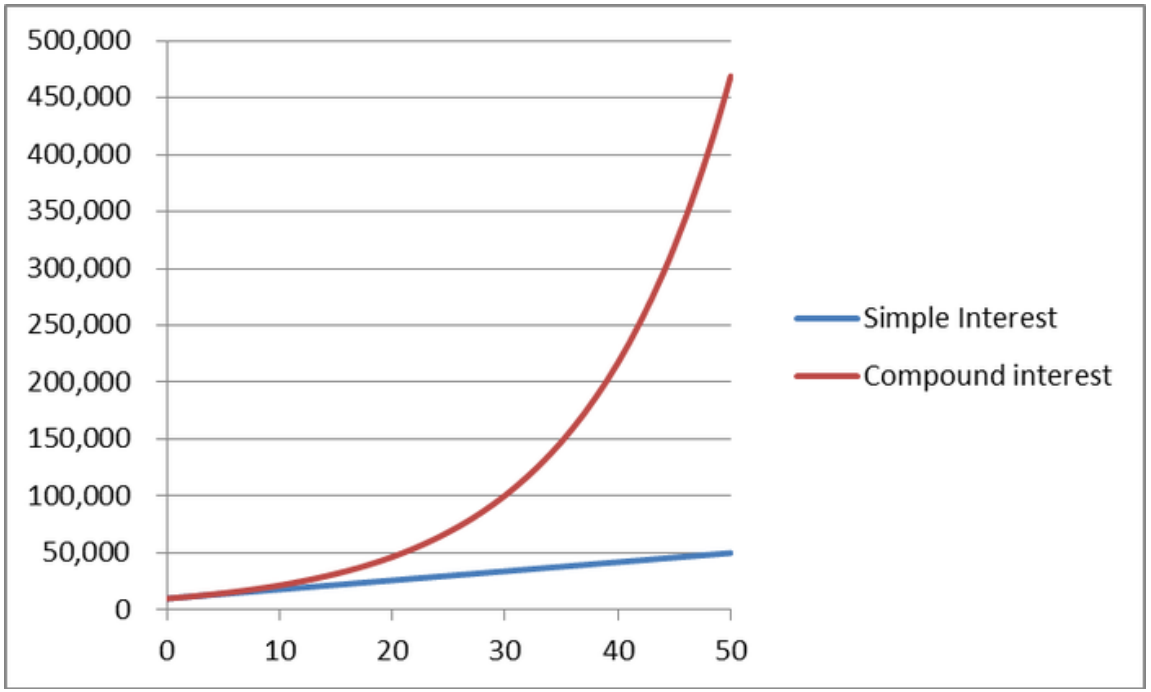
```
#include <stdio.h>
int main(void)
{
    int answer =59;    // 정답
    int guess;
    int tries = 0;
    do {
        printf("정답을 추측하여 보시오: ");
        scanf("%d", &guess);
        tries++;
        if (guess >answer)  // 사용자가 입력한 정수가 정답보다 높으면
            printf("제시한 정수가 높습니다.");
        if (guess <answer)  // 사용자가 입력한 정수가 정답보다 낮으면
            printf("제시한 정수가 낮습니다.");
     } while (guess !=answer);
        printf("축하합니다. 시도횟수=%d", tries);
    return 0;
```

lab4: 복리 이자 계산

```
=================
연도 원리금
=================
1 1050000.0
2 1102500.0
3 1157625.0
4 1215506.3
5 1276281.6
6 1340095.6
7 1407100.4
8 1477455.4
9 1551328.2
10 1628894.6
```

복리에서 원리금 합계

$$원리합계 = 원금 \times (1+이율)^{기간}$$



Lab4-sol

```c
#include <stdio.h>

#define RATE   0.07// 이율
#define INVESTMENT  10000000 // 초기 투자금
#define YEARS  10 // 투자 기간

int main(void)
{
	int i;
	double total = INVESTMENT;// 원리금 합계

	printf("==============\n");
	printf("연도  원리금\n");
	printf("==============\n");

	for (i = 1; i <= YEARS; i++)
	{
		total = total * (1 + RATE);// 새로운 원리금 계산
		printf("%2d%10.1f\n", i, total);
	}

	return 0;
}
```

Lab5: 자동으로 수학문제 생성하기

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
```

Lab5-Sol

```
#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;
}
```

Q & A