C 언어

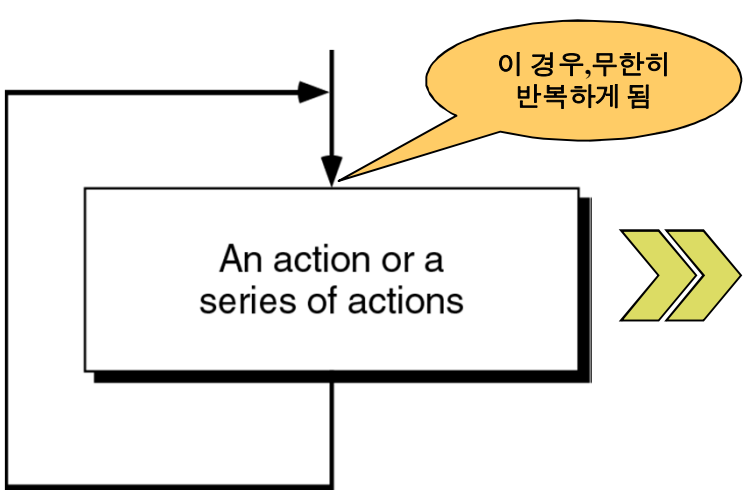
반복문

- **for**
- **while**

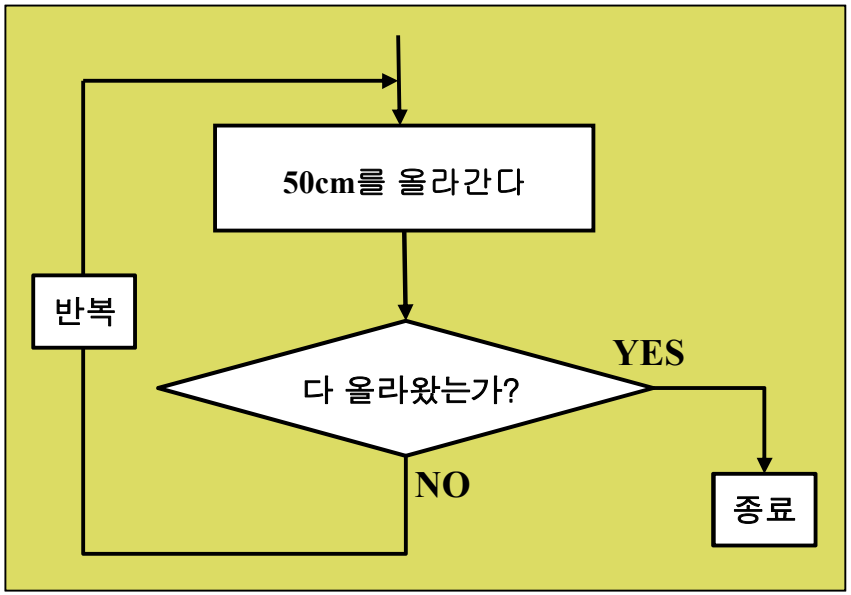
Concept of a loop

반복문은 어떤 조건에 도달할 때까지 특정 상태(condition)를
만족하는 동안 동일한 작업을 반복하여 수행할 수 있도록 해준다.

우물의 깊이는 **3m**이고 달팽이는 하루에 **50cm**를 올라갑니다.
만약 달팽이가 미끄러지지 않는다면 달팽이는 몇 일만에 우물을 벗어날 수 있을까요?



<Loop의 개념>

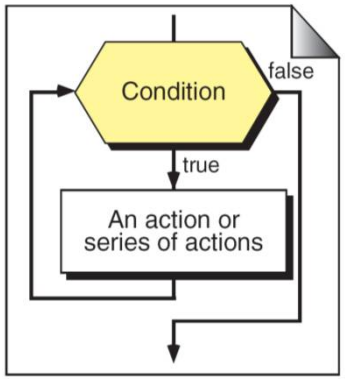


Pretest and Post-test Loops

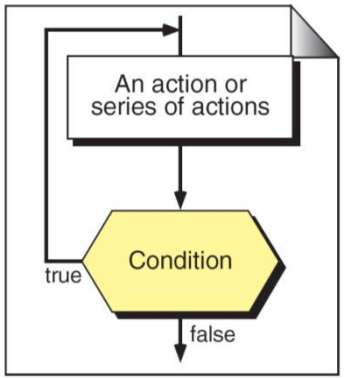
Pretest loop

- loop control expression 체크
- true ➔ loop안의 statements를 실행
  & loop의 첫 단계로 돌아감
- false ➔ loop를 중단

Condition 에 따라서 statement가 실행되지 않을 가능성이 있다.



(a) Pretest Loop

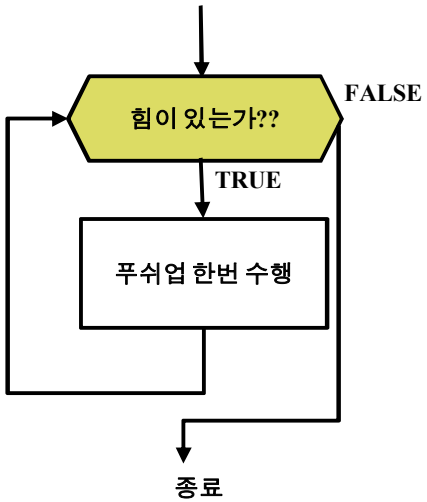


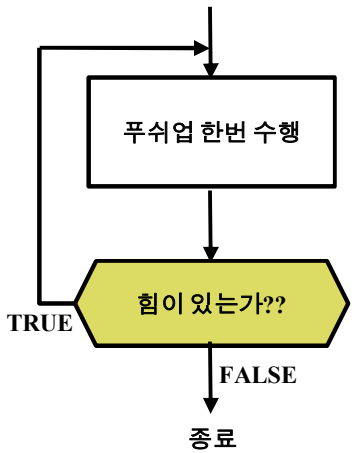
(b) Post-test Loop

Post-test loop

- loop 안의 statements를 먼저 실행
- loop control expression 체크
- true ➔ loop의 첫 단계로 돌아감
- false ➔ loop를 중단

Condition 에 상관없이 statement는 최소한 1회 이상 실행된다.





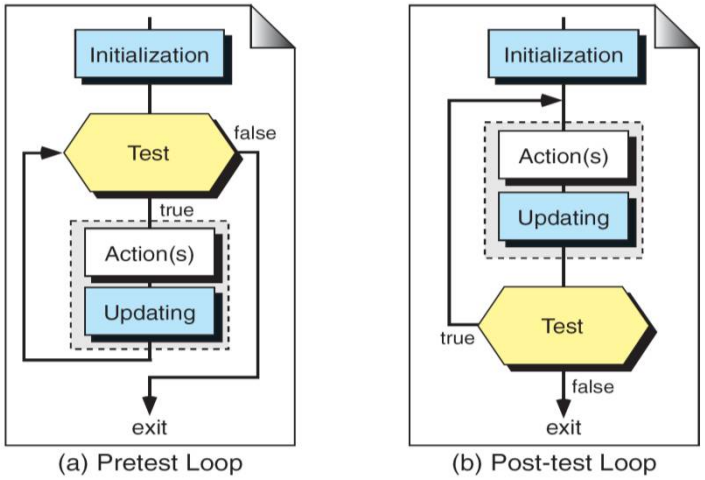
Initialization & Updating

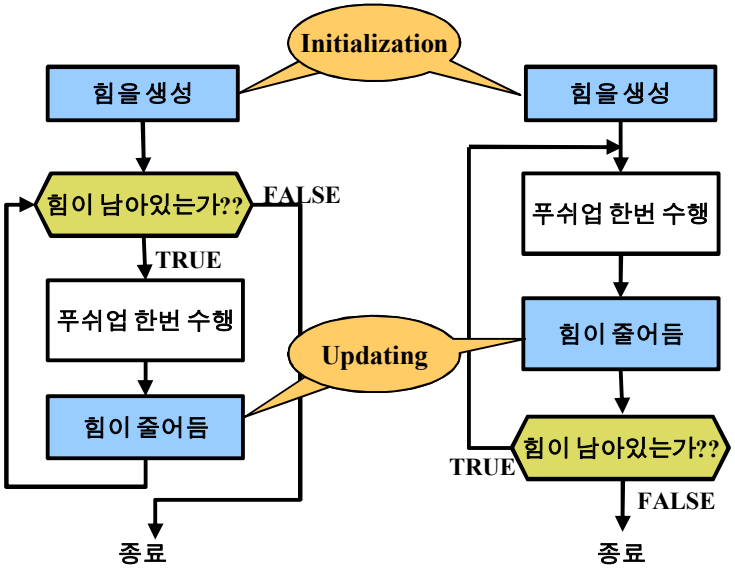
Initialization

- Loop가 시작되기 전에,
  loop initialization(초기화)가 필요
- 일반적으로 제어변수(control variable)값을 할당

Updating

- loop body에서 condition의 변화가 필요
  ➔ 무한 루프를 피하기 위해서
- Loop Condition에 변화를 줌으로서,
  제어조건을 FALSE로 바꿈
  ➔ 루프 종료



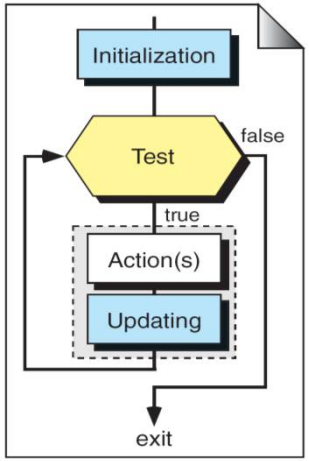


Event- & Count-controlled Loops

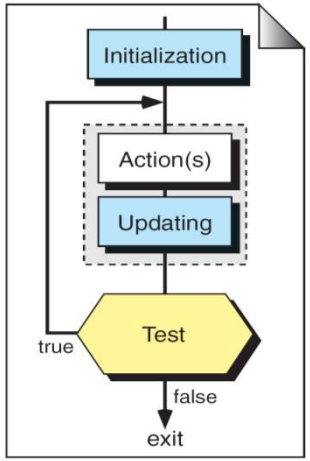
Event-Controlled Loops

- loop control expression 의 값을 true 에서 false로 바꾸는 어떤 **event**가 발생
- 반복횟수를 알 수 없음.

  ➔ 별도의 count변수 필요



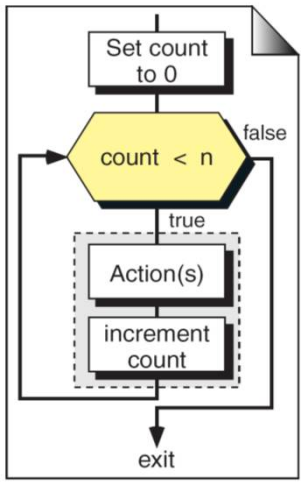
(a) Pretest Loop



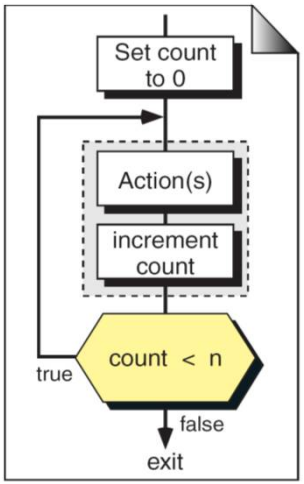
(b) Post-test Loop

Count-Controlled Loops

- 반복횟수를 condition으로 사용
- count 변수의 값을 증가 또는 감소시키면서 원하는 횟수만큼 반복했을 때 중지



(a) Pretest Loop



(b) Post-test Loop

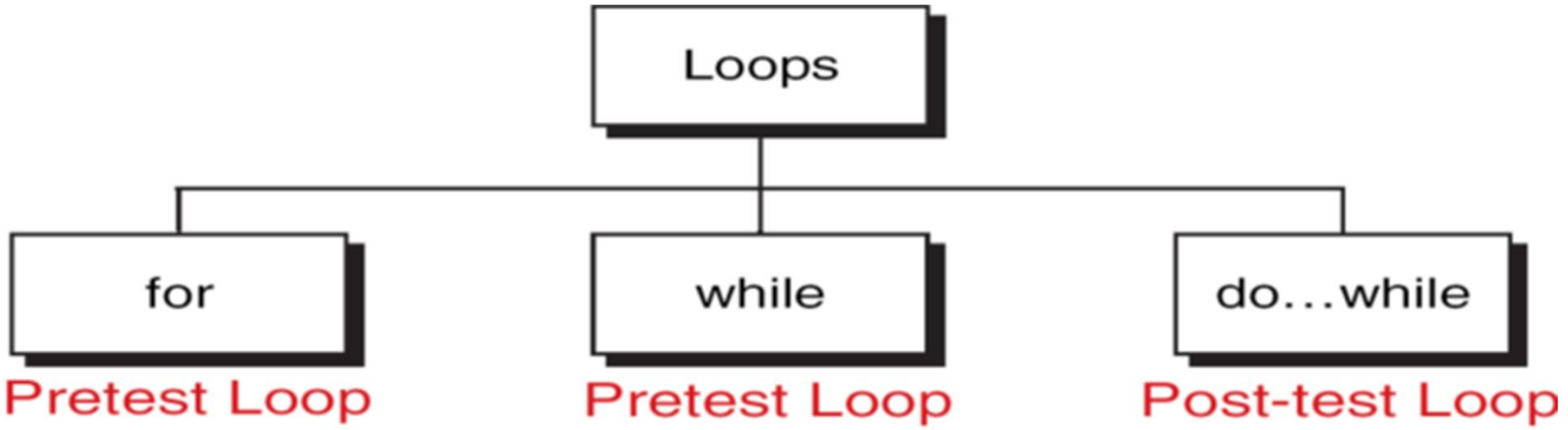
Event- & Count-controlled Loops

Loop Comparison

- 반복을 n번 수행한다고 할 때, Pretest Loop과 Post-test 반복횟수 비교

	Pretest Loop	Post-test Loop
Initialization	1	1
Number of tests	n+1	n
Action execute	n	n
Updating executed	n	n
Minimum iteration	0	1

Loops in C



<for문>

```
for( 초기문 ; 조건문 ; 증감문 )
{
    반복할 내용
}
```

최초 "초기문"을 실행하고, "조건문"을 만족하면 "반복할 내용"과 "증감문"을 차례로 실행한다.

<while문>

```
while( 반복 조건 )
{
    반복할 내용
}
```

"반복 조건"이 만족되는 동안 "반복할 내용"을 반복 실행한다.

<do while문>

```
do
{
    반복할 내용
} while ( 반복 조건 );
```

일단 루프를 한번 실행 후, "반복조건"이 만족되는 동안 "반복할 내용"을 반복실행한다.

실습예제

0부터 10까지의 총합을 구하는 프로그램을 작성하시오

- 실행결과예시

```
0부터 10까지의 총합은 55
```

The *for* Loop

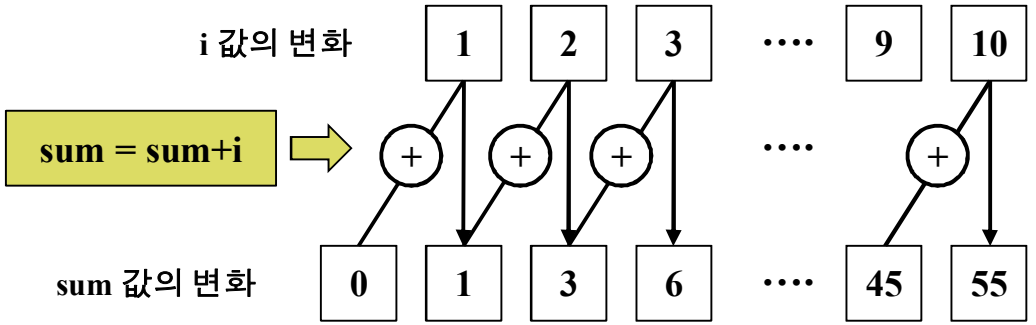
예제 프로그램 - for loop

```
 1 #include <stdio.h>
 2
 3 int main(void)
 4 {
 5         int i, sum=0;
 6
 7         for(i=0; i<=10; i++)
 8                 sum+=i;
 9
10         printf("sum=%d\n", sum);
11         return 0;
12 }
```

```
[root@mclab c-lang]# vi chap6-2.c
[root@mclab c-lang]# gcc -o chap6-2 chap6-2.c
[root@mclab c-lang]# ./chap6-2
sum=55
[root@mclab c-lang]#
```

1부터 10까지 더하는 프로그램

➔ i 를 1부터 10까지 증가시키면서 sum에 더해준다.



실습예제 1

1이상의 정수 n1과 n2를 입력 받는다. 그리고 n1부터 입력 받은 정수 n2 까지의 합을 계산해서 그 결과를 출력하는 프로그램을 작성하시오.

- INPUT

  정수를 입력하세요. 1 5

- OUTPUT

  1+2+3+4+5 = 15

▪ 실행결과예시

정수를 입력하세요: 2 6

2+3+4+5+6 = 20

실습예제 2

1이상의 정수 n1과 n2를 입력 받는다. 그리고 n1부터 입력 받은 정수 n2 까지의 합을 계산해서 그 결과를 출력하되, 3의 배수이거나 5의 배수이면 총 합에서 제외 시키는 는 프로그램을 작성하시오.

- INPUT

  정수를 입력하세요. 1 7

- OUTPUT

  1+2+4+7 = 14

▪ 실행결과예시

정수를 입력하세요: 1 7
3는 합에서 제외
5는 합에서 제외
6는 합에서 제외
1+2+4+7 = 14

The *for* Loop

예제 프로그램 - nested for loop

```c
#include <stdio.h>

int main(void)
{
        int i, j;
        for(i=2; i<3; i++)
        {
                for(j=1; j<10; j++)
                        printf("%d * %d = %d\n", i, j, i*j);
        }
        return 0;
}
```

구구단 2 – 3단 출력하는 프로그램

➔ for 문이 중첩되어 사용되고 있다.

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
```

실습예제 3

출력하고 싶은 구구단의 단수를 입력 받아서 해당 되는 단의 구구단을 출력하시오

- INPUT
  - 단수를 입력하세요. 9
- OUTPUT
  - 9 * 1 = 9
  - 9 * 2 = 18

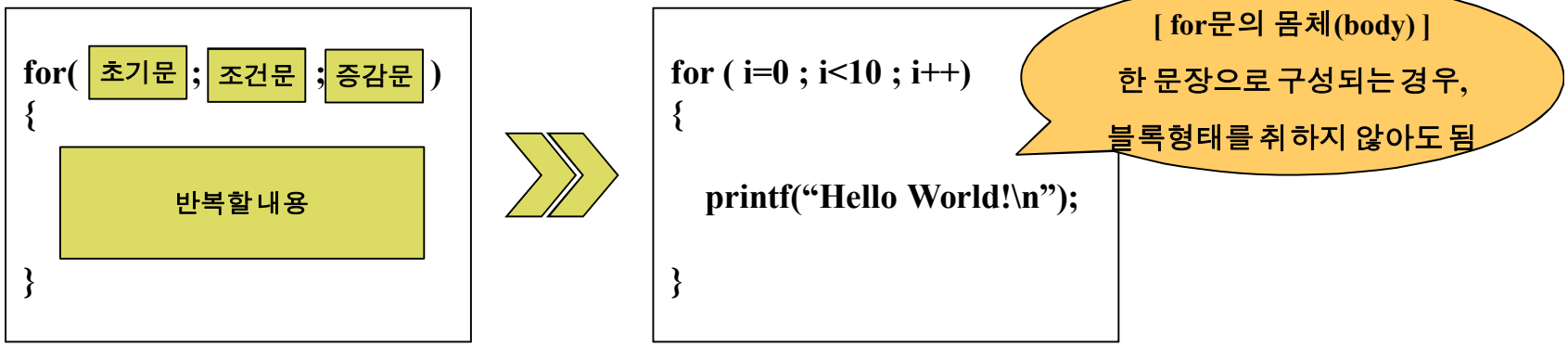
▪ 실행결과예시

```
구구단 중에서 출력하고 싶은 단을 입력하시오: 9
9*1 = 9
9*2 = 18
9*3 = 27
...
```

The *for* Loop

for문 구조

- **초기문** (Initilization), **조건문** (Limit-test expression), **증감문** (Updating expression)
- 각 구문별로 세미콜론(;) 으로 구분
  - 괄호( ) 안에 세미콜론은 반드시 필요하다.
  - 괄호( ) 안에 각 세 부분은 생략되어도 문법 오류는 발생하지 않음
    - ☞ for (; ;) { …. };



The *for* Loop

for 문의 실행순서

① 초기화(expr1)를 실행한다.

② 조건검사(expr2)을 실행한다.

③ 조건이 참이면

loop body (statement)를 수행한다.

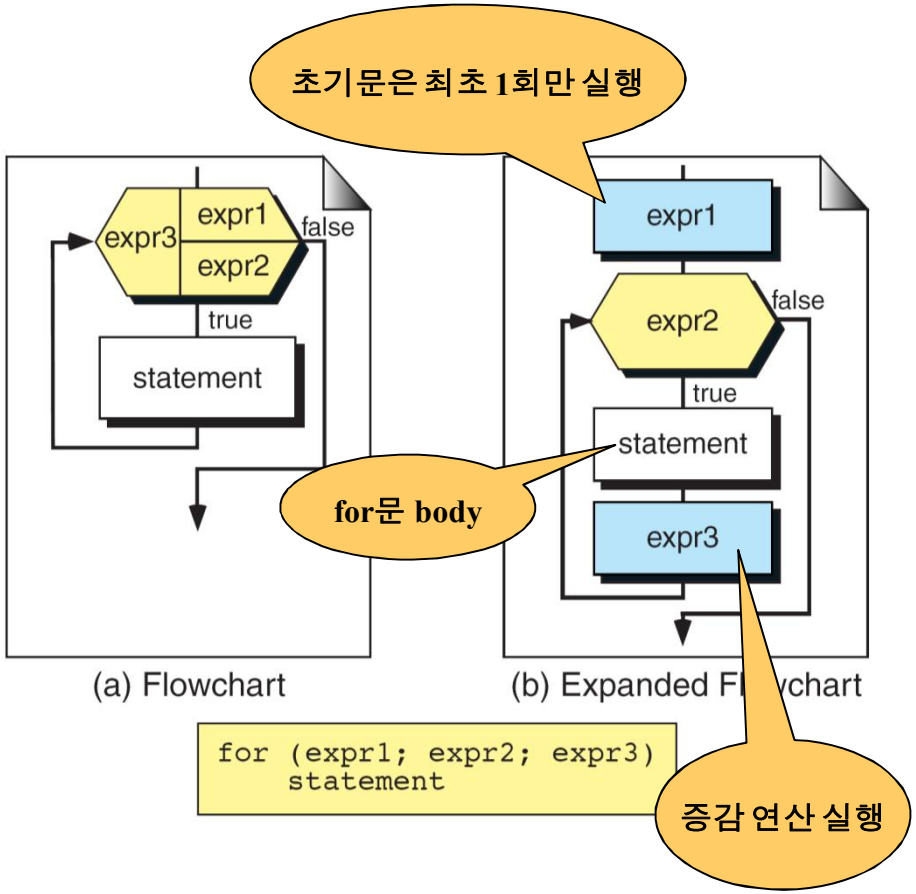
expr3을 수행하고 ②로 돌아간다.

④ 조건이 거짓이면 for문을 종료한다.

```
for ( expr1 ; expr2 ; expr3)
{

   statement

}
```



실습예제 4

하나의 정수를 입력 받고, 입력 받은 수에 해당하는 구구단을 출력하는 프로그램을 작성하시오.

- 입력 받는 정수는 0, 2~9 사이의 범위를 갖고, 그 이외의 수가 들어오면 에러 메시지를 출력, 0을 입력 받으면 2~9단 전부 출력한다.

▪ 실행결과예시

```
Number : 3
Result :
3 * 1 = 3
3 * 2 = 6
 …
```

The *while* Loop

예제 프로그램 – while loop

```c
#include <stdio.h>

int main(void)
{
        int height = 0;
        int days = 0;
        int depth;

        printf("Input the depth of well(cm) : ");
        scanf("%d", &depth);

        while(height < depth)
        {
                height = height + 50;
                days = days + 1;
        }

        printf("Total days: %d\n", days);
}
```

달팽이 우물 탈출 일수 구하는 프로그램

1) 우물의 높이에 미치지 못하면

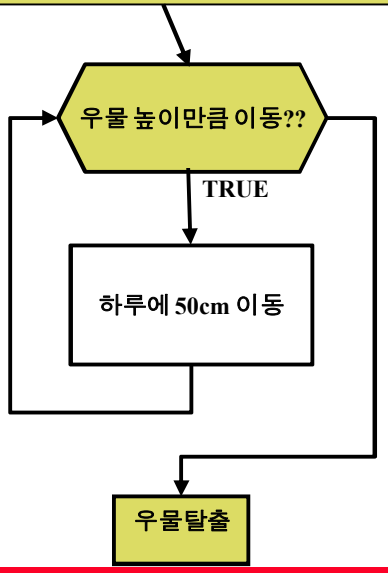
(height < depth)

➔ while 문의 내용을 반복 실행

2) 우물의 높이에 도달하면

(height >= depth)

➔ while문 종료 & 총 소요일수 출력



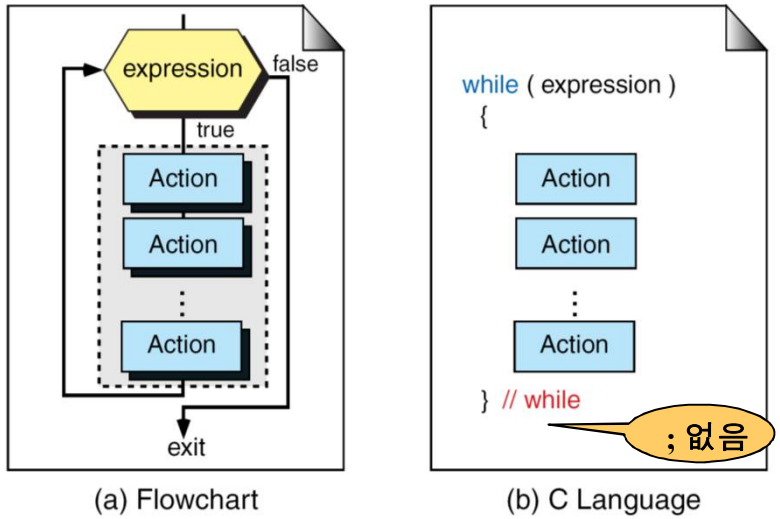

```
[root@mclab c-lang]# vi chap6-1.c
[root@mclab c-lang]# gcc -o chap6-1 chap6-1.c
[root@mclab c-lang]# ./chap6-1
Input the depth of well(cm) : 300
Total days: 6
[root@mclab c-lang]#
```

The *while* Loop

- while 문 구조

기본원리

특정 조건을 주고 그 조건이 만족될 때까지
계속해서 반복을 시킨다



(a) Flowchart (b) C Language

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

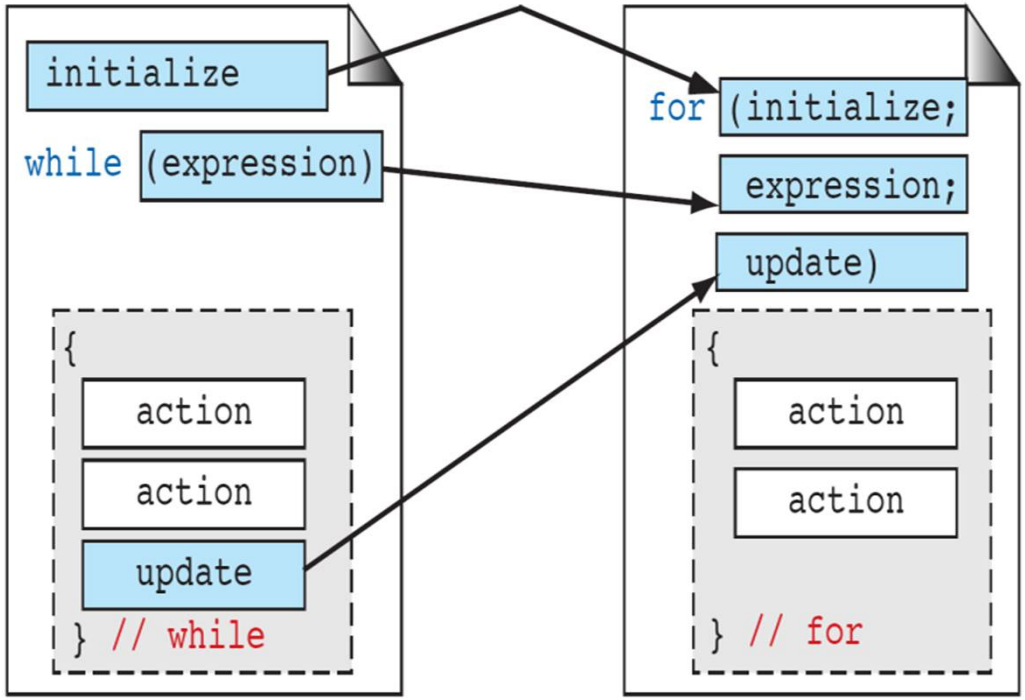
[while문의 몸체(body)]
한 문장으로 구성되는 경우,
블록형태를 취하지 않아도 됨

- **while 문 실행순서**
  - 조건문인 i < 10 을 검사하여 i 가 조건을 만족하면(true) 반복문의 몸체(body)에 당하는 문장이나 블록을 실행한다.
  - 모든 몸체를 실행하면 다시 조건문을 검사하는 첫 번째 과정을 반복한다.
  - 첫 번째 과정에서 조건을 만족하지 않으면(false) while문을 종료한다.

The *for* Loop

for와 **while** loop의 비교

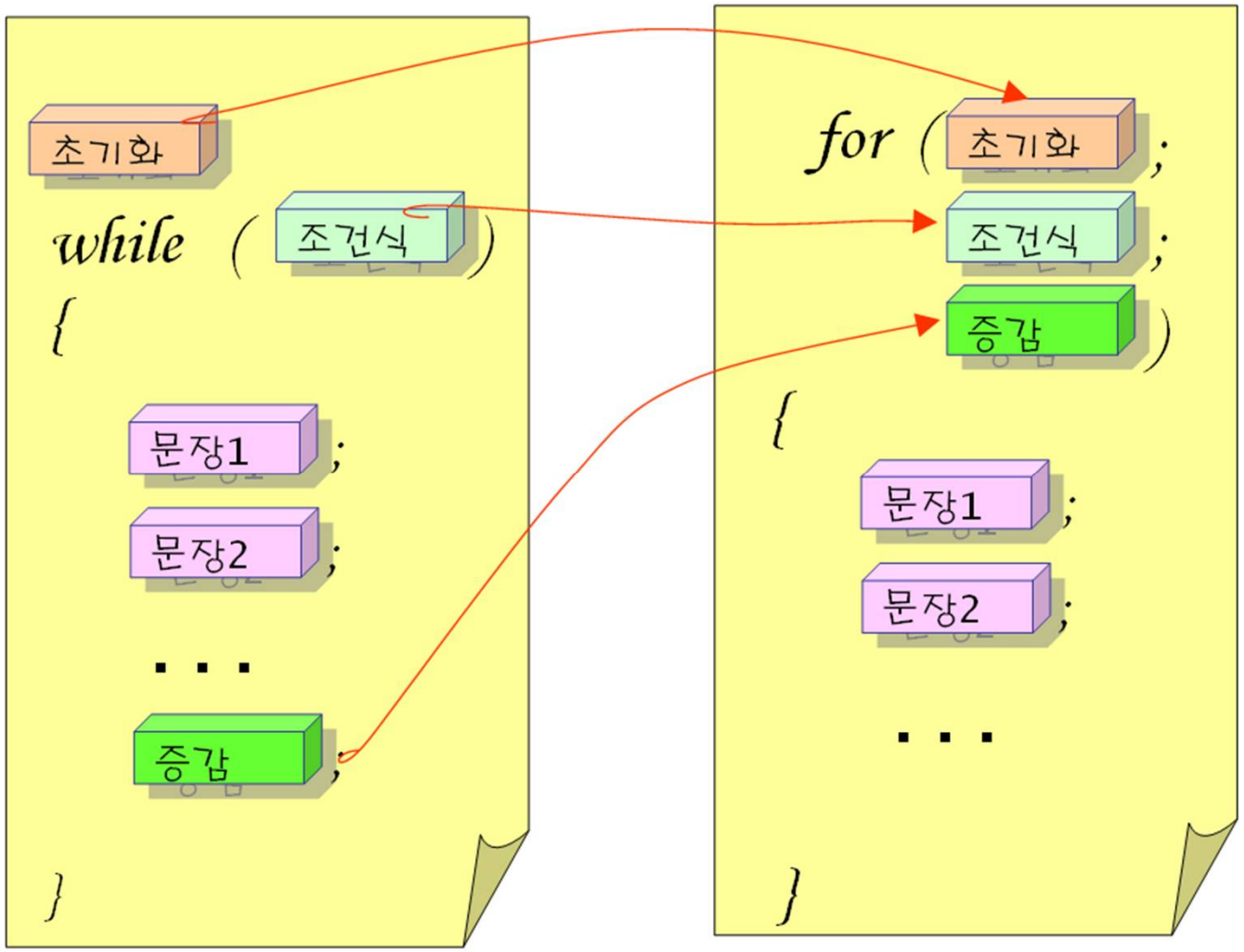
- for loop은 while loop과 동일한 역할을 수행하지만 readability 가 좋으며 counting loop에 자연스럽다.




```
i=1;
sum=0;
while(i<=20)
{
    scanf("%d",&a);
    sum += a;
    i++;
} //while
```

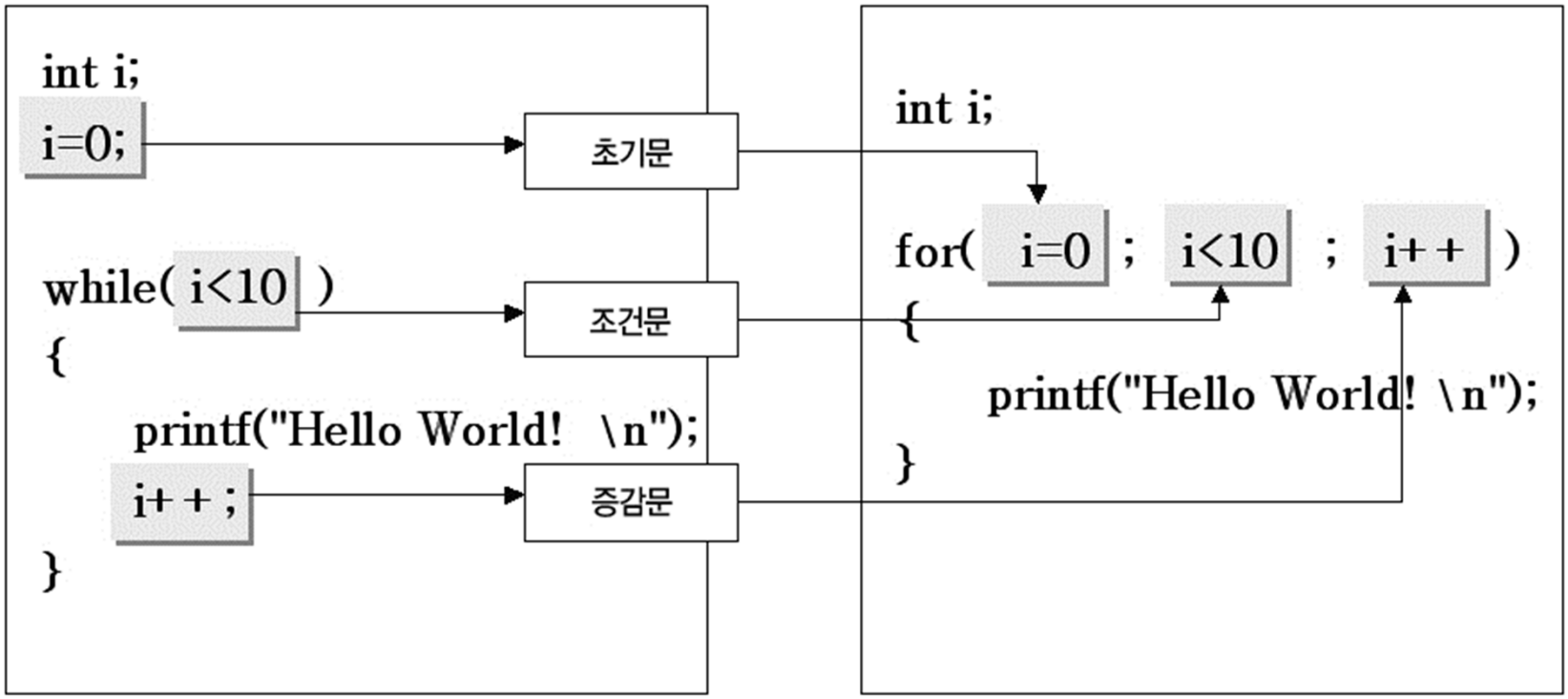
```
sum = 0;
for (i = 1; i <=20 ; i++)
{
    scanf("%d",&a);
    sum += a;
} // for
```

while 과 for 의 비교



The *for* Loop

for와 while 비교



The *do..while* Loop

예제 프로그램 – do..while loop

```
 1 #include <stdio.h>
 2
 3 int main(void)
 4 {
 5         int total = 0;
 6         int val = 0;
 7
 8         do {
 9                 printf("Input number(Quit : 0) : ");
10                 scanf("%d", &val);
11                 total+=val;
12         }while(val != 0);
13
14         printf("Total : %d\n",total);
15         return 0;
16 }
```

사용자가 입력하는 수를 계속해서 더하는 프로그램

1) 먼저 사용자로부터 숫자를 입력받음
2) 입력받은 값으로 진행할 것인지 말것인지 결정

➔ 최소한 한 번의 실행이 필요한 상황에서는 **do ~ while** 문이 **while** 문 보다 자연스럽다.

```
Input number(Quit : 0) : 2
Input number(Quit : 0) : 4
Input number(Quit : 0) : -5
Input number(Quit : 0) : 3
Input number(Quit : 0) : 1
Input number(Quit : 0) : 0
Total : 5
```

The *do..while* Loop

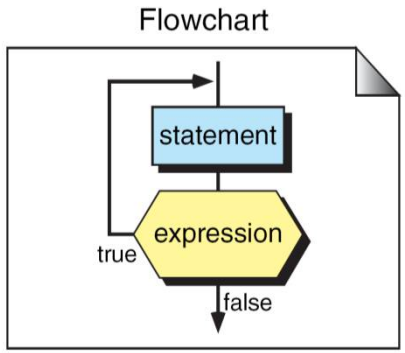
while 문은 조건이 만족되지 않으면 루프를 한번도 실행하지 않는다.

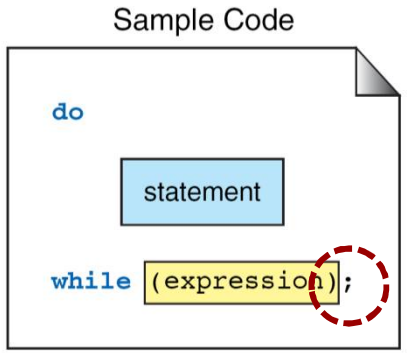
do..while문은 반드시 한번은 루프를 실행하도록 되어있다.

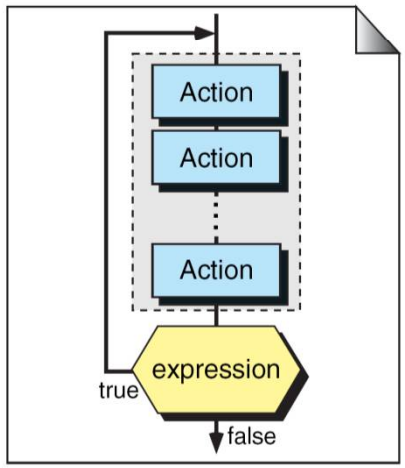
➔ **Post-test Loop**

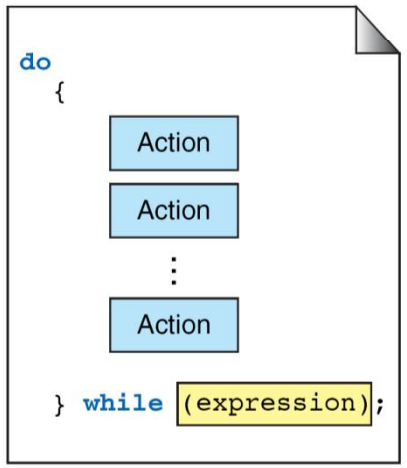
경우에 따라서는 **while**문보다 **do..while**문을 사용하는 것이 훨씬 자연스러운 경우가 있다

while(expression) 끝에 세미콜론(;)을 반드시 붙여야 한다!!!









Other Statements Related to Looping

예제 프로그램 - break와 continue

```
 1 #include <stdio.h>
 2
 3 int main(void)
 4 {
 5         int i, sum = 0;
 6
 7         for(i=0; ; i++) {
 8                 if(i>=100)
 9                         break;
10                 else if(i%4 == 0)
11                         continue;
12                 sum+=i;
13         }
14         printf("sum=%d\n", sum);
15 }
```

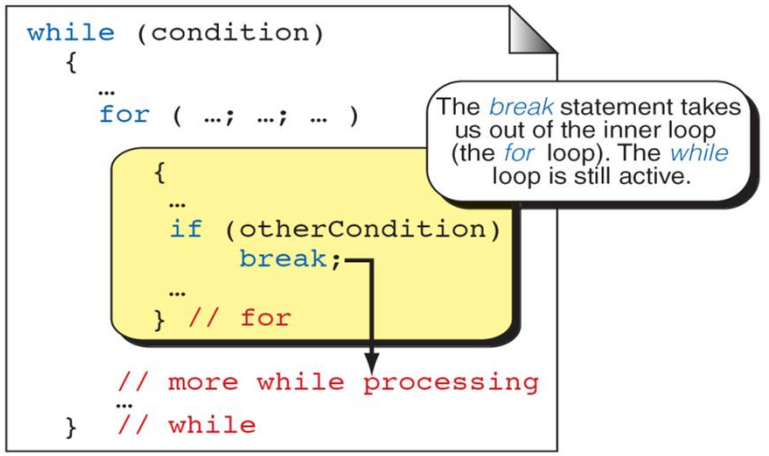
```
sum=3750
```

1과 99사이의 4의 배수가 아닌 숫자들의 합을 구하는 프로그램

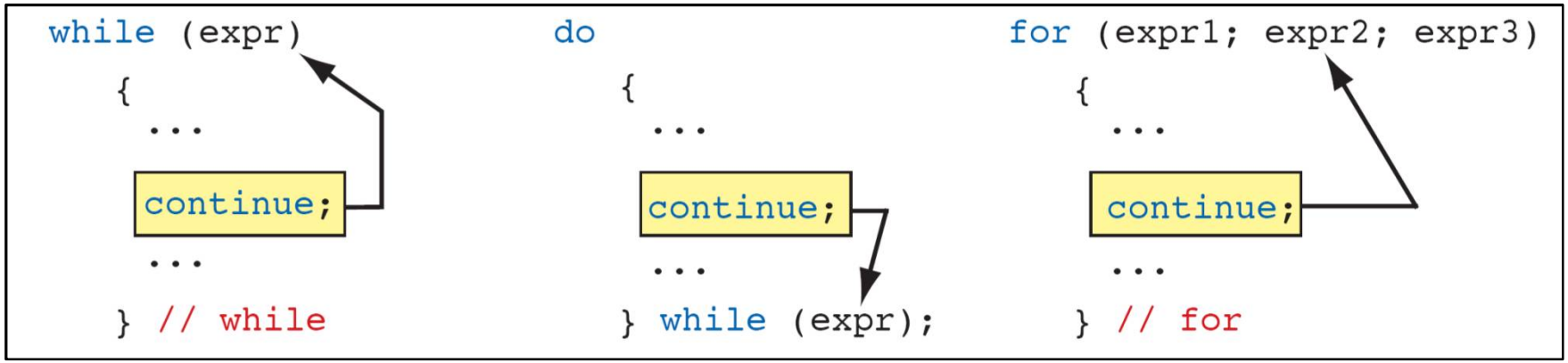
1) for 문에서 조건식이 없으므로 기본적으로는 무한루프를 돌게 되어 있다.
2) i가 100보다 커지는 경우 break를 만나 루프를 빠져나옴.
3) 4의 배수인 경우에는 continue를 만나서 sum에 더해지지 않는다.

Other Statements Related to Looping

break : 가장 가까이서 감싸고 있는
반복문 하나를 빠져 나오게 한다.
주로 제한조건이 반복문, 조건문의
중간에 만족하였을 때 사용



continue : 돌던 루프의 남아있는 부분을 그냥 건너뛰는 것.
- while이나 do-while문은 검사식(conditional expression)으로 제어 이동
- for문은 변경식(update expression) 으로 제어 이동



실습예제 5

다음의 수학식을 계산하는 코드를 작성하시오.

$$\sum_{i=10}^{100}\sum_{j=0}^{50}(i^2 + j^2)$$

- 실행결과예시

```
int partial=0, sum=0;

for(int i=1; i<=100; i++){
   for(int j=1;j<=50;j++){
      partial = i*i + j*j;
      sum += partial;
   }
}
```

실습예제 6-팩토리얼 구하기

팩토리얼 값을 계산하여 보자. 팩토리얼이란 다음과 같이 정의된다

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

- INPUT
  - 정수를 입력하세요. 20
- OUTPUT
  - 20!은 2432902008176640000

▪ 실행결과예시

```
정수를 입력하시오:20
20!은 2432902008176640000입니다.
```

Recursion

예제 프로그램 - recursion

```c
#include <stdio.h>

int factorial(int n);

int main(void) {
        int a;
        printf("Input a number : ");
        scanf("%d", &a);

        printf("%d! = %d\n", a, factorial(a));
}

int factorial(int n) {
        if(n==0)
                return 1;
        else
                return n * factorial(n-1);
}
```

```
Input a number : 5
5! = 120
```

함수 factorial() 안에서 factorial() 를 호출하는 프로그램

1) 호출할 때 인자는 매번 1씩 줄어들기 때문에 언젠가는 factorial(0)을 호출하게 된다.
2) factorial(0)은 1을 반환한다.
3) 호출된 순서를 거꾸로 올라가며 factorial값을 구한다.

실습예제 7

Fibonacci 수열은 앞의 두개의 원소를 합하여 뒤의 원소를 만드는 수열이며, 다음과 같이 정의한다. 정의: $f_0 = 0$, $f_1 = 1$, $f_{i+1} = f_i + f_{i-1}$ (for i=1, 2, …), 예: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, …

반복문을 이용하여 **Fibonacci** 수열을 구하는 프로그램을 작성하시오.

- 실행결과예시

실습예제 8

사용자에게서 받은 문자열에서 자음과 모음의 개수를 출력하는 프로그램을 작성하시오. (Hint: 문자열 객체에 포함된 length(), charAt() 메소드를 활용, 소문자만 고려하며 모음은 'a', 'e', 'i', 'o', 'u')

- 실행결과예시

실습예제 9

키보드에서 영문자 하나를 읽어서 모음과 자음을 구분하고 개수도 세는 프로그램을 작성하여 보자.

- 실행결과예시