

# IT프로그래밍

한성대학교  
IT융합공학부  
오희석  
(ohhs@hansung.ac.kr)

## Chapter 07-1. while문에 의한 문장의 반복

## Chapter 07. 반복실행을 명령하는 반복문

# 반복문의 이해와 while문

## · 반복문이란

하나 이상의 문장을 두 번 이상 반복 실행하기 위해서 구성하는 문장

## · 반복문의 종류

while, do~while, for

반복의 대상이 한 문장이면 중괄호 생략 가능

```
while(num<5)
    printf("Hello world! %d \n", num++);
```

```
while(num<5)
    printf("Hello world! %d \n", num), num++;
```

```
int main(void)
{
    int num=0;
    while(num<5)
    {
        printf("Hello world! %d \n", num);
        num++;
    }
    return 0;
}
```

*while* 반복문

중괄호 내부 반복영역

반복의 목적이 되는 대상

변수 num은 반복의 횟수를 조절하기 위한 것!

```
Hello world! 0
Hello world! 1
Hello world! 2
Hello world! 3
Hello world! 4
```

실행결과

# 반복문 안에서도 들여쓰기 합니다.

---

들여쓰기를 하지 않은 것

```
int main(void)
{
int num=0;
while(num<5)
{
printf("Hello world! %d \n", num);
num++;
}
return 0;
}
```

들여쓰기를 한 것

```
int main(void)
{
    int num=0;
    while(num<5)
    {
        printf("Hello world! %d \n", num);
        num++;
    }
    return 0;
}
```

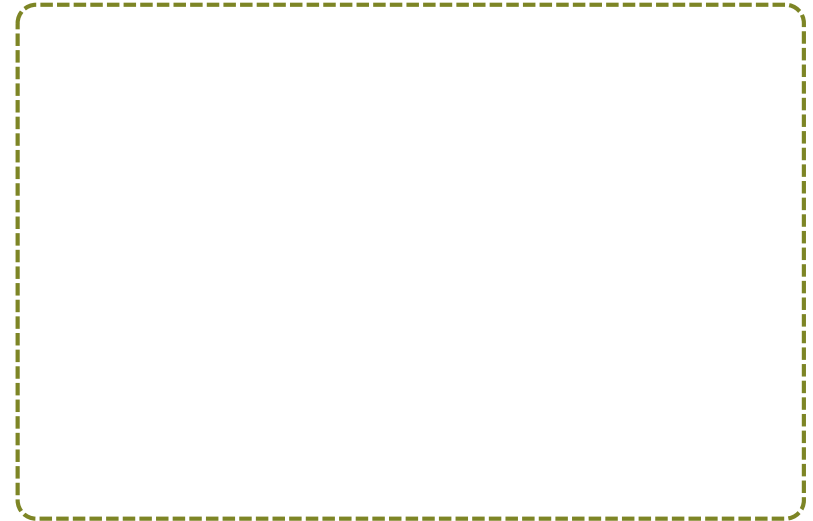
**들여쓰기를 한 것과 하지 않은 것의 차이가 쉽게 눈에 들어온다!**

---

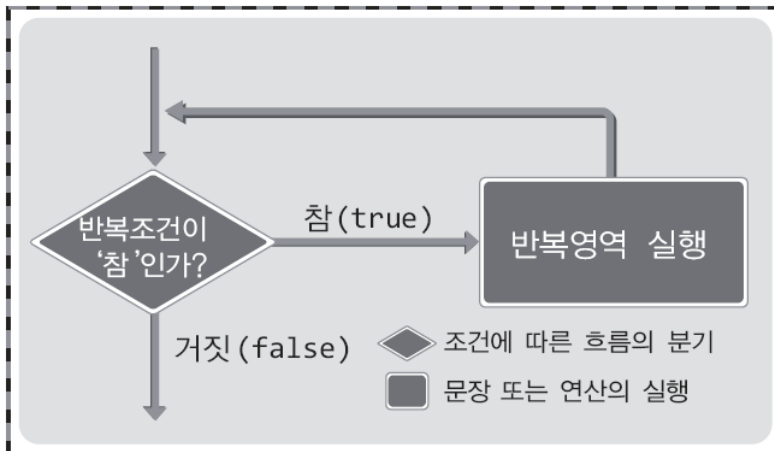


# while문의 구성과 실행흐름의 세세한 관찰

```
int main(void)
{
    int num=0;
    while(num<3)  // 3회 반복
    {
        printf("Hello world! %d \n", num);
        num++;
    }
    . . . .
}
```



반복의 과정은?



flow chart 기준에서의 while문

# 구구단의 출력

---

```
int main(void)
{
    int dan=0, num=1;
    printf("몇 단? ");
    scanf("%d", &dan);

    while(num<10)
    {
        printf("%d×%d=%d \n", dan, num, dan*num);
        num++;
    }
    return 0;
}
```

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

실행결과

구구단은 반복문을 이해하는데 사용되는 대표적인 예제이다.  
이후에 반복문의 중첩에서는 구구단 전체를 출력하는 예제를 접한다.

---



# 무한루프의 구성

---

```
while( 1 )
{
    printf("%d×%d=%d \n", dan, num, dan*num);
    num++;
}
```

숫자 1은 '참'을 의미하므로 반복문의 조건은 계속해서 '참'이 된다.

이렇듯 반복문의 탈출조건이 성립하지 않는 경우 무한루프를 형성한다고 한다.

이러한 무한루프는 실수로 만들어지는 경우도 있지만, break문과 함께 유용하게 사용되기도 한다.



# while문의 중첩

while문 안에 while문이 존재하는 상태를 의미한다. 아래의 예제에서는 while문을 중첩시켜서 구구단 전체를 출력한다. 이 예제를 통해서 중첩된 while문의 코드 흐름을 이해하자.

```
int main(void)
{
    int cur=2;
    int is=0;

    while(cur<10) // 2단부터 9단까지 반복
    {
        is=1; // 새로운 단의 시작을 위해서
        while(is<10) // 각 단의 1부터 9의 곱을 표현
        {
            printf("%d×%d=%d \n", cur, is, cur*is);
            is++;
        }
        cur++; // 다음 단으로 넘어가기 위한 증가
    }

    return 0;
}
```

바깥쪽 while문

안쪽 while문





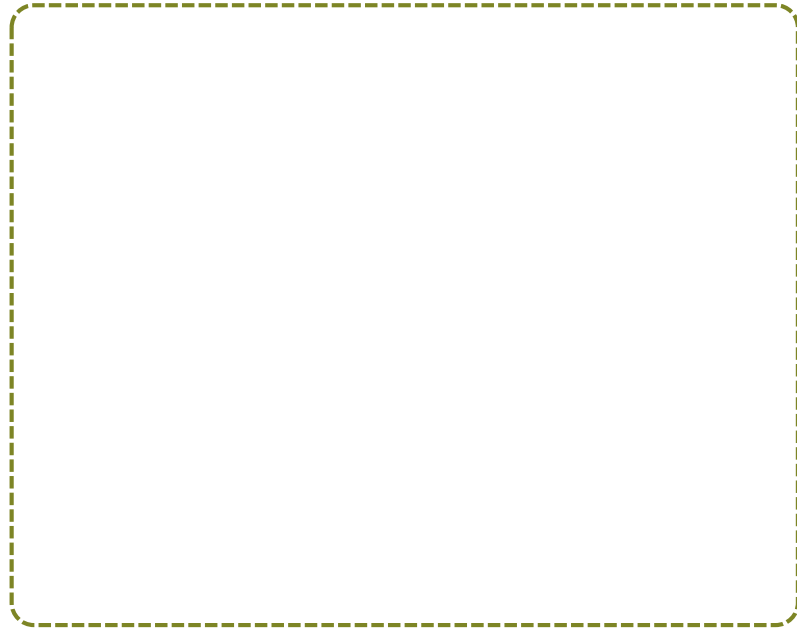
## Chapter 07-2. do~while문에 의한 문장의 반복

### Chapter 07. 반복실행을 명령하는 반복문

# do~while문의 기본구성

---

```
do
{
    printf("Hello world! \n");
    num++;
} while(num<3);
```



반복의 과정은?

반복조건을 반복문의 마지막에 진행하는 형태이기 때문에  
최소한 1회는 반복영역을 실행하게 된다. 이것이 while문과의 가장 큰 차이점이다.

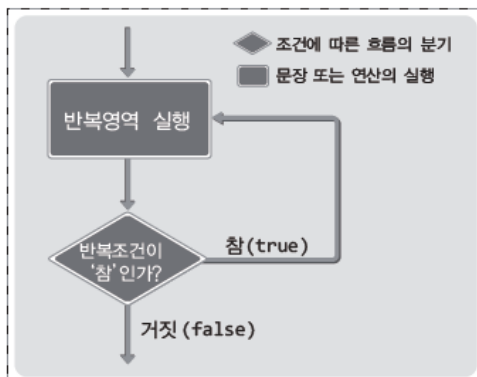


# do~while문이 자연스러운 상황

```
while(num<10)
{
    printf("%d×%d=%d \n", dan, num, dan*num);
    num++;
}
```

↕ 동일한 횟수를 반복하는 반복문들

```
do
{
    printf("%d×%d=%d \n", dan, num, dan*num);
    num++;
} while(num<10);
```




do~while문의 순서도

```
int main(void)
{
    int total=0, num=0;
    do
    {
        printf("정수 입력(0 to quit): ");
        scanf("%d", &num);
        total += num;
    }while(num!=0);
    printf("합계: %d \n", total);
    return 0;
}
```

정수 입력(0 to quit): 1  
정수 입력(0 to quit): 2  
정수 입력(0 to quit): 3  
정수 입력(0 to quit): 4  
정수 입력(0 to quit): 5  
정수 입력(0 to quit): 0  
합계: 15

실행결과

최소한 1회 이상 실행되어야 하는 반복문은  
do~while문으로 구성하는 것이 자연스럽다.



## Chapter 07-3. for문에 의한 문장의 반복



## Chapter 07. 반복실행을 명령하는 반복문

# 반복문의 필수3요소

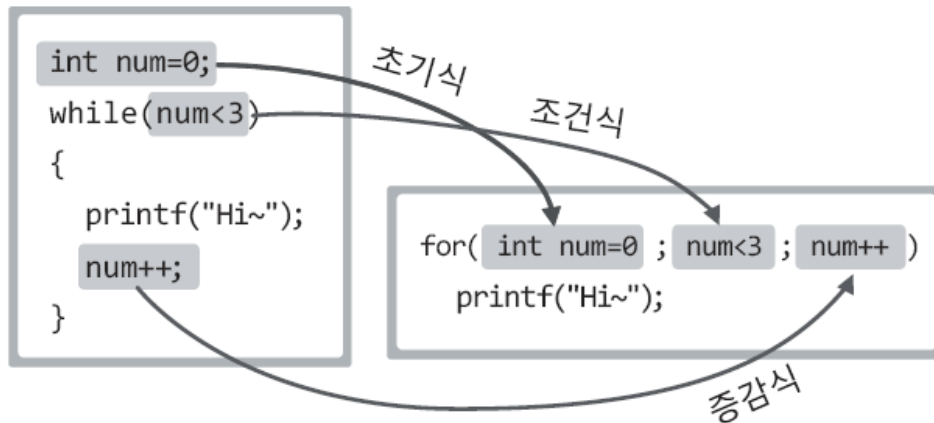
```
int main(void)
{
    int num=0;    // 필수요소 1. 반복을 위한 변수의 선언
    while(num<3)  // 필수요소 2. 반복의 조건검사
    {
        printf("Hi~");
        num++;    // 필수요소 3. 반복의 조건을 '거짓'으로 만들기 위한 연산
    }
    . . . .
}
```

정해진 횟수의 반복을 위해서는 하나의 변수가 필요하다.  
그 변수를 기반으로 하는 조건검사가 필요하다.  
조건검사가 false가 되게 하기 위한 연산이 필요하다.

이 세 가지를 한 줄에 표시하도록  
돕는 것이 for문이다.

위의 while문에서 보이듯이 반복문에 필요한 세 가지 요소가 여러 행에 걸쳐서 분산되어 있다.  
따라서 반복의 횟수가 바로 인식 불가능하다.

# for문의 구조와 이해



```
for( 초기식 ; 조건식 ; 증감식 )
{
    // 반복의 대상이 되는 문장들
}
```

```
int main(void)
{
    int num;
    for(num=0; num<3; num++)
        printf("Hi~");
    . . .
}
```

일부 컴파일러는 여전히 초기식에서의 변수 선언을 허용하지 않는다.  
for문의 반복영역도 한 줄이면 중괄호 생략 가능!

# for문의 흐름 이해

## for문의 구성요소

- ✓ 초기식    본격적으로 반복을 시작하기에 앞서 딱 한번 실행된다.
- ✓ 조건식    매 반복의 시작에 앞서 실행되며, 그 결과를 기반으로 반복유무를 결정!
- ✓ 증감식    매 반복실행 후 마지막에 연산이 이뤄진다.

## for문 흐름의 핵심

int num=0에 해당하는 초기화는 반복문의 시작에 앞서 딱 1회 진행!  
num<3에 해당하는 조건의 검사는 매 반복문의 시작에 앞서 진행!  
num++에 해당하는 증감연산은 반복영역을 실행한 후에 진행!

### ① 첫 번째 반복의 흐름

1 → 2 → 3 → 4 [num=1]

### ② 두 번째 반복의 흐름

2 → 3 → 4 [num=2]

### ③ 세 번째 반복의 흐름

2 → 3 → 4 [num=3]

### ④ 네 번째 반복의 흐름

2 [num=3] 따라서 탈출!

```
1      2      4
for( int num=0 ; num<3 ; num++ )
{ 3
    printf("Hi~");
}
```

# for문 기반의 다양한 예제

```
int main(void)
{
    int total=0;
    int i, num;
    printf("0부터 num까지의 덧셈, num은? ");
    scanf("%d", &num);

    for(i=0; i<num+1; i++)
        total+=i;

    printf("0부터 %d까지 덧셈결과: %d \n", num, total);
    return 0;
}
```

0부터 num까지의 덧셈, num은? 10  
0부터 10까지 덧셈결과: 55

실행결과

다양한 예제를 통해서 for문에 익숙해지자!

오른쪽 예제에서 보이듯이 불필요하다면, 초기식, 조건식, 증감식을 생략할 수 있다.  
단 조건식을 생략하면 참으로 인식이 되어 무한루프를 형성하게 된다.

```
int main(void)
{
    double total=0.0;
    double input=0.0;
    int num=0;

    for( ; input>=0.0 ; )
    {
        total+=input;
        printf("실수 입력(minus to quit) : ");
        scanf("%lf", &input);
        num++;
    }
    printf("평균: %f \n", total/(num-1));
    return 0;
}
```

실수 입력(minus to quit) : 3.2323  
실수 입력(minus to quit) : 5.1891  
실수 입력(minus to quit) : 2.9297  
실수 입력(minus to quit) : -1.0  
평균: 3.783700

실행결과





# for문의 중첩

---

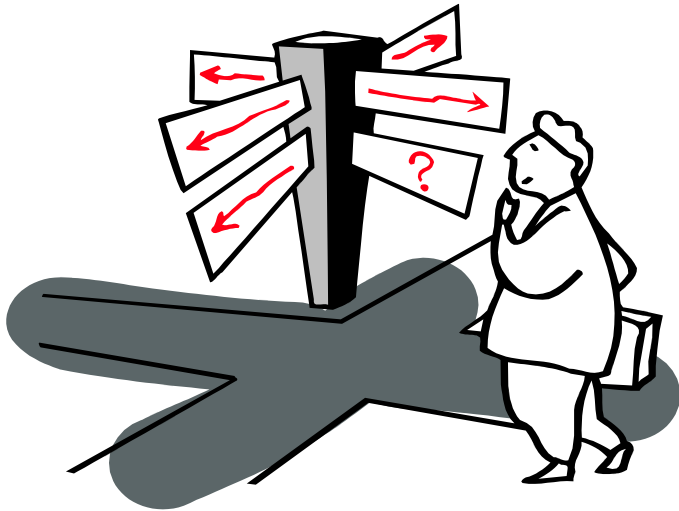
```
int main(void)
{
    int cur, is;

    for(cur=2; cur<10; cur++)
    {
        for(is=1; is<10; is++)
            printf("%d×%d=%d \n", cur, is, cur*is);
        printf("\n");
    }
    return 0;
}
```

**for문의 중첩은 while, do~while문의 중첩과 다르지 않다.**

**구구단 전체를 출력하는 원편의 예제를 통해서 for문의 중첩을 이해하자.**





Chapter 07이 끝났습니다. 질문 있으신지요?