4주차 2차시 반복문

【학습목표】

1. while 문을 설명할 수 있다.
2. for 문을 설명할 수 있다.

학습내용1 : while 문

1. 반복문의 이해와 while문

√반복문이란?
•하나 이상의 문장을 두 번 이상 반복 실행하기 위해서 구성하는 문장

√반복문의 종류
•while, do~while, for

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

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

√네모칸 안의 내용이 반복의 목적이 되는 대상이다.
변수 num은 반복의 횟수를 조절하기 위한 것!

√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++;
```

2. 반복문 안에서도 들여쓰기 합니다.

들여쓰기를 하지 않는 것

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

들여쓰기를 한 것

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

들여쓰기를 한 것과 하지 않은 것의 차이가 쉽게 눈에 들어온다!

3. while문의 구성과 실해흐름의 세세한 관찰

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

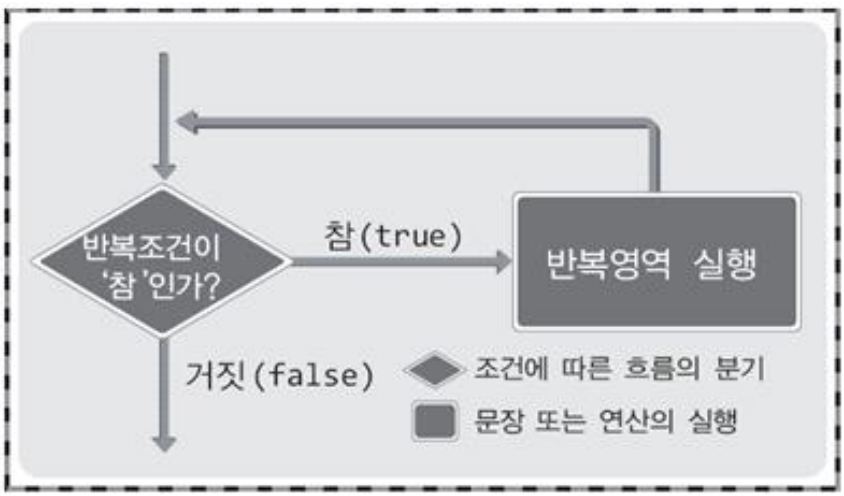
1단계: num = 0 (참)
Hello world
num = 1

2단계: num=1 (참)
Hello world
num = 2

3단계: num=2 (참)
Hello world
num = 3

4단계: num =3 (거짓)
반복구간 벗어남

flow chart 기준에서의 while문



4. 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
```

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

5. 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.

6. 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

학습내용2 : do~while문

1. do~while 문의 기본 구성

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

1단계: Hello world
Num = 1

2단계: Hello world
Num = 2

3단계: Hello world
Num = 3

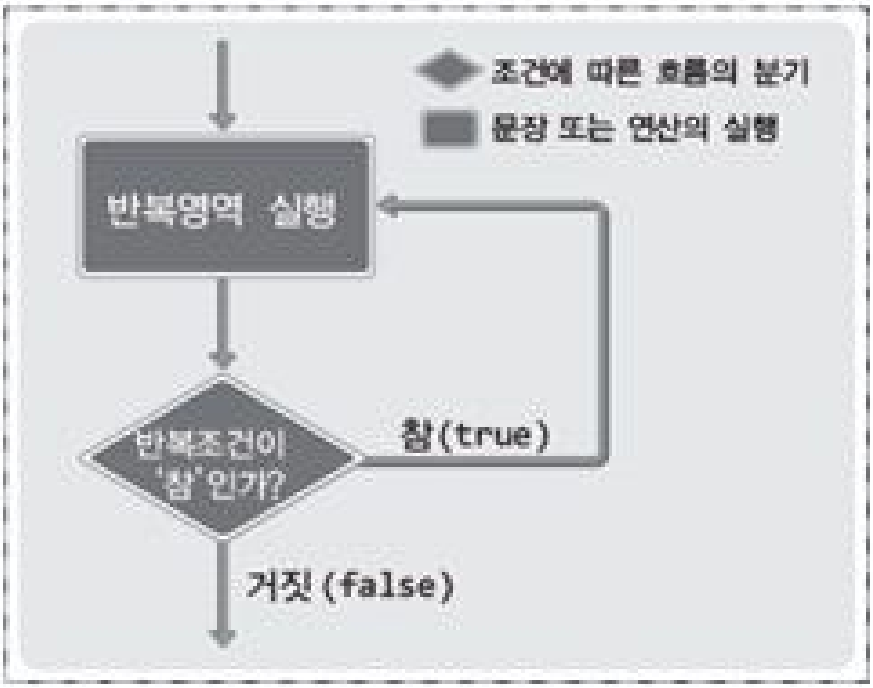
반복조건을 반복문의 마지막에 진행하는 형태이기 때문에
최소한 1회는 반복영역을 실행하게 된다. 이것이 while문과의 가장 큰 차이점이다.

2. do~while문이 자연스러운 상황

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

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

do~while문의 순서도



```c
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
```

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

학습내용3 : for 문

1. 반복문의 필수 3요소

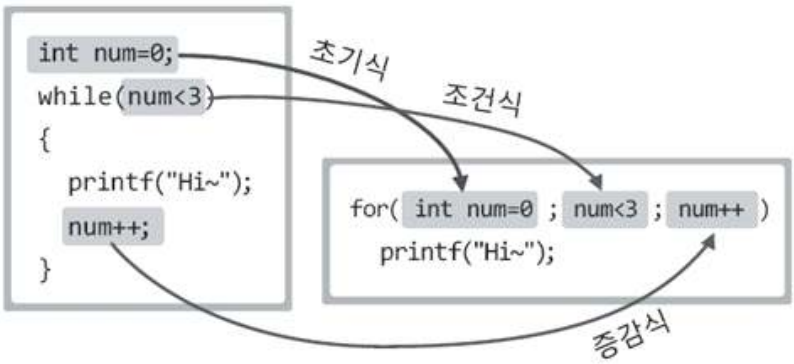
```c
int main(void)
{
    int num=0;   // 필수요소 1. 반복을 위한 변수의 선언
    while(num<3)    // 필수요소 2. 반복의 조건검사
    {
        printf("Hi~");
        num++;   // 필수요소 3. 반복의 조건을 '거짓'으로 만들기 위한 연산
    }
    . . . .
}
```

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

위의 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.

2. for 문의 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!

3. for문 흐름의 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.

- 첫 번째 반복의 흐름
  1 → 2 → 3 → 4 [num=1]
- 두 번째 반복의 흐름
  2 → 3 → 4 [num=2]
- 세 번째 반복의 흐름
  2 → 3 → 4 [num=3]
- 네 번째 반복의 흐름
  2 [num=3] 따라서 탈출!

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

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!

4. 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
```

```
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
```

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

5. 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 중첩 과 다르지 않다.
구구단 전체를 출력하는 위의 예제를 통해서 for문의 중첩을 이해하자.

【학습정리】

1. 조건식이 참인 동안 문장을 반복해서 실행한다. while 문을 만나면 먼저 조건식을 검사하는데, 만약 조건식이 거짓이면 문장을 실행하지 않고 while 문을 빠져 나온다

2. do~while문은 while 문과 유사하지만, 시작 부분에서 종료 조건을 검사하는 while 문과는 달리 do-while 문은 문장을 먼저 실행하고 마지막 부분에서 종료 조건을 검사한다.

3. for 문을 만나면 초기식을 먼저 계산한다. 조건식이 참이면 문장을 실행하여 변환식을 계산하고 다시 조건식을 검사하는 반면, 조건식이 거짓이면 문장을 실행하지 않고 for 문을 종료한다.