# C 프로그래밍 1
# Lecture Note #07

---

백윤철
ybaek@smu.ac.kr

# 내용

- while
- do while
- for

# 반복(Loop)문의 이해와 while 문

- 반복문이란
  - 한 개 이상의 문장을 두 번 이상 반복 실행하기 위해서 구성하는 문장
- 반복문의 종류
  - while, do-while, for

실행결과

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

```
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!

# 반복문

- 모든 반복문에서 반복의 대상이 한 문장이면 중괄호 생략 가능

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

- 반복문 안에서도 들여쓰기 함

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;
}
```

# 구구단의 출력

- 구구단은 반복문을 이해하는데 사용되는 대표적 예제
- 이후에 구구단 전체를 출력하는 예제를 볼 예정 (중첩)

```
int main(void) {
    int dan = 0, num = 1;
    printf("몇 단? ");
    scanf("%d", &dan);
    while (num < 10) {
        printf("%d x %d = %d\n", dan, num,
                dan * num);
        num++;
    }
    return 0;
}
```

실행결과

```
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
```

# 무한 반복의 구성

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

- 숫자 1은 '참'을 의미하므로 반복문의 조건은 계속 해서 '참'이 됨
- 이렇듯 반복문의 탈출 조건이 성립하지 않는 경우 무한 반복을 형성한다고 함
- 이러한 무한 반복은 실수로 만드는 경우도 있지만, break문과 함께 유용하게 사용되기도 함

# while문의 중첩

- while문 안에 while문이 존재하는 상태를 의미함

```
int main(void) {
    int dan = 2, num = 0;
    while (dan < 10) { /* 2단부터 9단까지 반복*/
        num = 1; /* 새로운 단의 시작을 위해 */
        while (num < 10) { /* 1부터 9의 곱을 표현 */
            printf("%d x %d = %d\n", dan, num,
                    dan * num);
            num++;
        }
        dan++;
    }
    return 0;
}
```

안쪽 while문

바깥쪽 while문

# do~while문의 기본 구성

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

- 반복 조건 검사를 반복문의 마지막에 진행하는 형태이므로 최소한 1회는 반복영역을 실행 → while 문과의 가장 큰 차이점

# do~while문이 자연스러운 상황

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

실행결과

```
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 반복은 do~while문으로 구성

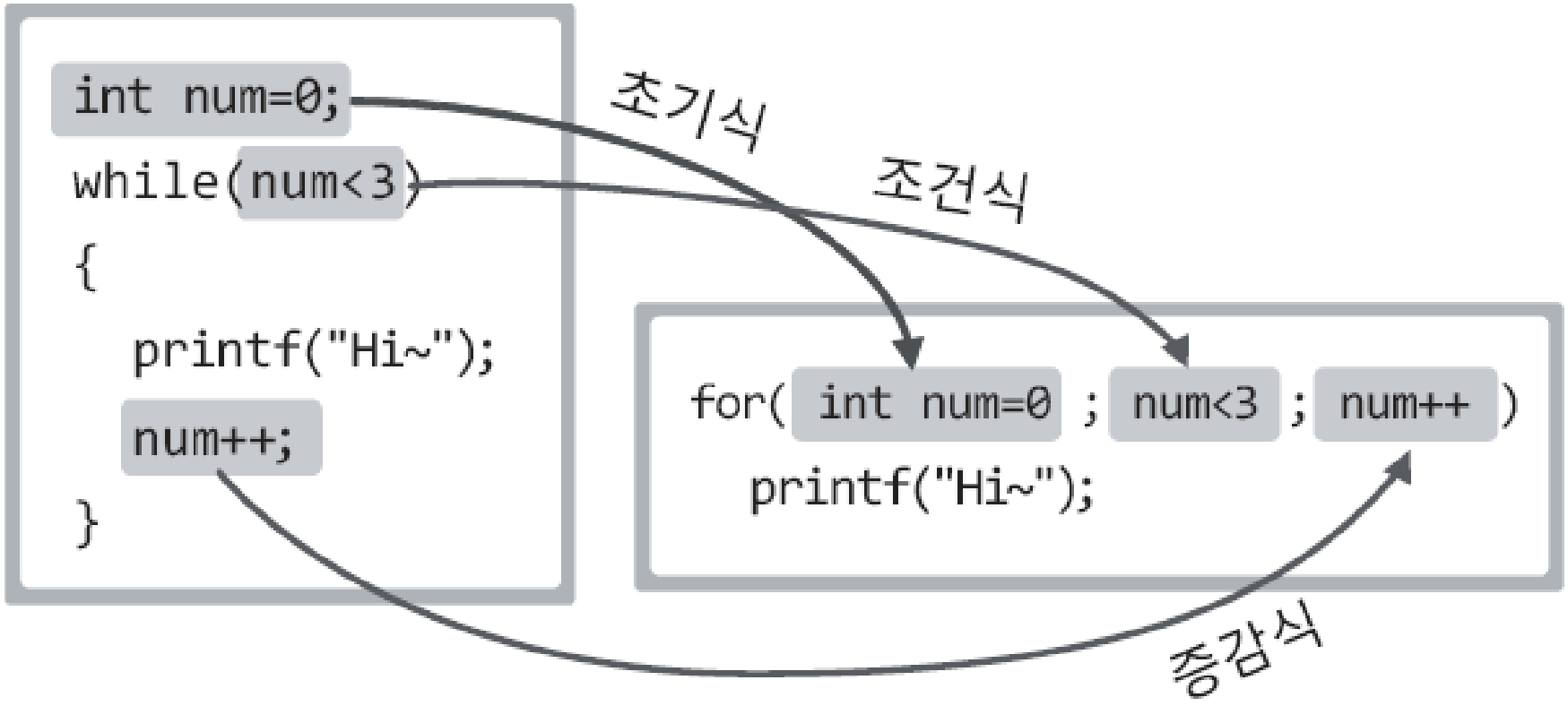
# 반복문의 요소

```
int main(void) {
    int num = 0; /*필수요소 1.반복을 위한 변수 정의*/
    while (num < 3) {/*필수요소 2.반복의 조건 검사*/
        printf("Hi~");
        /* 필수요소 3.반복의 조건을 '거짓'으로 만들기 위한 연산*/
        num++;
    }
    …
    return 0;
}
```

# 반복문의 요소

- 정해진 횟수의 반복을 위해서는 **한 개의 변수**가 필요함
- 그 변수를 기반으로 하는 **조건 검사**가 필요
- 조건검사가 false가 되게 하기 위한 **연산**이 필요
- while문에서는 반복문에 필요한 세 가지 요소가 여러 행에 걸쳐 분산되어 있어 반복의 횟수를 바로 인식하는 것이 어려움
- 이 세 가지를 한 줄에 표시하도록 만든 것이 for 반복문

# for문의 구조와 이해




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

# for문의 구조와 이해

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

- 일부 컴파일러는 초기식에서의 변수 선언을 허용하지 않음
- for문의 반복 영역이 한 줄이면 중괄호 생략 가능

# for 문의 흐름 이해

- for문의 구성요소
  - 초기식
    - 본격적으로 반복을 시작하기 전에 한 번만 실행됨
  - 조건식
    - 매 반복의 시작에 앞서 실행됨
    - 결과를 바탕으로 반복 또는 정지를 결정
  - 증감식
    - 매 반복 실행 후에 연산이 이루어짐

# for 문의 흐름 이해

- 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~");
}
```

# for문 기반의 다양한 예제

```
int main(void) {
    int sum = 0;
    int i, num;
    printf("0부터 num까지의 덧셈, num은? ");
    scanf("%d", &num);
    for (i = 0; i < num+1; i++)
        sum += i;
    printf("0부터 %d까지 덧셈 결과: %d\n", num, sum);
    return 0;
}
```

실행결과

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

# for문 기반의 다양한 예제

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

실행결과

```
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
```

# 구구단

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

- for 문의 중첩은 while, do-while과 비슷함

# 정리

- while
- do while
- 반목문의 요소
- for