

## Project1: Recursion vs. Iteration

수업 시간에 배운, factorial 프로그램을 recursion version과 iteration version으로 각각 구현하고, 입력 값을 1~10까지 증가시키면서 화면에 출력되게 실행하시오. n값은 scanf으로 입력받지 말고, main 함수에서 parameter로 전달해서 수행하시오. (의도: 구현의 정확하게 되었는지를 보는 문제입니다.)

Windows 환경에서 작성하였고, IDE는 비주얼 스튜디오를 이용했습니다.

### 문제 1)

#### Factorial function with Iteration

##### 1.1 코드 작성 순서

- 1) 입력 정수 받아오기
- 2) 정수를 1씩 감소시키며 1이 될 때까지 해당 값에 곱하기

로 단계를 나눴고, 소스 코드입니다.

```
void IterationVersion(unsigned long long int inputNum)
{
    for (int i = inputNum - 1; i > 0; i--)
    {
        inputNum *= i; // iteration을 이용한 순차적 반복
    }
    printf("%lld\n", inputNum);
}
```

##### 1.2 출력 예시

```
1. Iteration_Result
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
iteration 수행 완료
```

## Factorial function with Recursion

### 2.1 코드 작성 순서

1. 입력 정수 받아오기
2. 재귀함수로 전달
3. 2번 과정을 전달 파라미터를 1씩 감소시키며 반복 (Basecase 1이 될 때까지)

```
unsigned long long int RecursionVersion(unsigned long long int inputNum)
{
    if (inputNum == 1)
    {
        return 1;
    }
    return inputNum * RecursionVersion(inputNum - 1);
}
```

### 2.2 출력 예시

```
2. Recursion_Result
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
recursion 수행 완료
```

## 문제 2)

### 2.1 코드 작성 순서

1. 입력 정수 받아오기
2. 재귀, 반복함수로 전달
3. 2번 과정을 전달 파라미터를 1씩 감소시키며 반복 (Basecase 1이 될 때까지)

```
unsigned long long int iterfib(unsigned long long int num)
{
    int one = 1, two = 1, result = -1; // one은 n-2, two는n-1 값
    if (num == 1 || num == 2) // 1 또는 2면 1을 돌려줌
        return 1;
    else {
        for (int i = 3; i <= num; i++) {
            result = one + two; // 직전(n-1)과 직직전(n-2) 값을 더해서 돌려줌
            one = two; // 직직전(n-2) = 직전(n-1)
            two = result; // 직전(n-1) = 현재값(now)
        }
    }
    return result;
}

unsigned long long int recurfib(unsigned long long int num)
{
    if (num <= 1)
    {
        return num;
    }
    else
    {
        return recurfib(num - 1) + recurfib(num - 2);
    }
}
```

```
3. Fibonacci_Result
Fibonacci(1) 재귀 함수를 통한 출력 : 1, 반복 함수를 통한 출력 : 1
Fibonacci(2) 재귀 함수를 통한 출력 : 1, 반복 함수를 통한 출력 : 1
Fibonacci(3) 재귀 함수를 통한 출력 : 2, 반복 함수를 통한 출력 : 2
Fibonacci(4) 재귀 함수를 통한 출력 : 3, 반복 함수를 통한 출력 : 3
Fibonacci(5) 재귀 함수를 통한 출력 : 5, 반복 함수를 통한 출력 : 5
Fibonacci(6) 재귀 함수를 통한 출력 : 8, 반복 함수를 통한 출력 : 8
Fibonacci(7) 재귀 함수를 통한 출력 : 13, 반복 함수를 통한 출력 : 13
Fibonacci(8) 재귀 함수를 통한 출력 : 21, 반복 함수를 통한 출력 : 21
Fibonacci(9) 재귀 함수를 통한 출력 : 34, 반복 함수를 통한 출력 : 34
Fibonacci(10) 재귀 함수를 통한 출력 : 55, 반복 함수를 통한 출력 : 55
```

문제 3,4)

Factorial recursion version의 경우, n이 증가함에 따라 어느 정도의 n에서까지는 수행(예, 1000, 2000, 3000, 4000. 수행 컴퓨터마다 다를 수 있음)되나, 어느 정도 이후(약 5000)에는 “stack overflow”로 수행이 안됨을 보이시오. 단, factorial(1000)등의 결과값은 너무 커서, 결과값을 unsigned long long 타입으로 설정하여도 overflow됨으로 결과값은 무시해도 됩니다. (의도: stack overflow되 인해 run time error로 수행이 멈춘다는 것을 확인하는 것입니다.) 하지만, 이 경우(n=5000)도 iteration version은 수행됨도 보이시오. 이 경우도 결과 값은 정확히 나오지 않아도 됩니다. (의도: loop이 끝까지 동작해서, 만약 결과값만 담을 수 있는 방법만 있다면 수행이 가능함을 확인하는 것입니다.)

맨 마지막 줄 return 0; 위에 코드 정상 종료 구문을 추가하여 수행이 정상 종료 되었는지 확인하고자 했고, 이에 대한 결과값을 첨부함으로 3,4번에 대한 답변을 적겠습니다.

### 1. Iteration Result

overflow로 인해 부정확한 값, 수행은 멈추지 않은 실행 결과.

```
[input_Num : 4000]
1. Iteration_Result
Factorial_Calculation 4000!= 0
1 increased, 4001!= 0
2 increased, 4002!= 0
3 increased, 4003!= 0
4 increased, 4004!= 0
5 increased, 4005!= 0
6 increased, 4006!= 0
7 increased, 4007!= 0
8 increased, 4008!= 0
9 increased, 4009!= 0
10 increased, 4010!= 0
코드 정상 종료
```

```
[input_Num : 5000]
1. Iteration_Result
Factorial_Calculation 5000!= 0
1 increased, 5001!= 0
2 increased, 5002!= 0
3 increased, 5003!= 0
4 increased, 5004!= 0
5 increased, 5005!= 0
6 increased, 5006!= 0
7 increased, 5007!= 0
8 increased, 5008!= 0
9 increased, 5009!= 0
10 increased, 5010!= 0
코드 정상 종료
```

```
[input_Num : 110000]
1. Iteration_Result
Factorial_Calculation 110000!= 0
1 increased, 110001!= 0
2 increased, 110002!= 0
3 increased, 110003!= 0
4 increased, 110004!= 0
5 increased, 110005!= 0
6 increased, 110006!= 0
7 increased, 110007!= 0
8 increased, 110008!= 0
9 increased, 110009!= 0
10 increased, 110010!= 0
```

## 2. Recursion Result

5000 이후 수행이 되지 않은 모습

[illegible]

```
[input_Num : 5000]
2. Recursion_Result
```

<- 처음 정수를 메인함수의 파라미터로 전달하는 방식으로 입력하여 stack overflow를 확인했던 것이고, 과제 문제 그대로 해석하여 코딩을 다시 해보았습니다.

밑에 첨부.

```

printf("#값이 4000일때 팩토리얼의 iteration연산과 Recursive연산#\n");
printf("%lld", RecursionVersion(4000));
IterationVersion(4000);
printf("결과값은 중요하지 않으므로 정상 종료 유무 판별, 연산 정상 종료#\n");

printf("값을 5000을 넣었을 때, iteration#\n");
IterationVersion(5000);
printf("연산 정상 종료#\n");
printf("값을 5000을 넣었을 때, Recursion#\n");
printf("%lld", RecursionVersion(5000));
//말의 구문이 실행되지 않음으로 Recursion Version은 정상 종료가 되지 않았습니다.
printf("연산 정상 종료#\n");
return 0;

```

각 함수함에 파라미터로 정수를 직접 전달 후, 그 다음 줄의 print함수를 통해 정상적으로 함수가 동작하고 있는지 확인,

결과값 :

```

값이 4000일때 팩토리얼의 iteration연산과 Recursive연산
00
결과값은 중요하지 않으므로 정상 종료 유무 판별, 연산 정상 종료
값을 5000을 넣었을 때, iteration
0
연산 정상 종료
값을 5000을 넣었을 때, Recursion

```

값이 4000일때, 두 연산 다 부정확하지만 함수가 수행이 완료됨이 보이지만, 5000이 넘었을 때, iteration은 정상 종료가 되지만, Recursion은 overflow가 생긴 실행화면이다.