4일 차

사물 인터넷(IoT/ICT)환경에서의 임베디드 응용SW 개발자 양성과정

은태영

복습_for

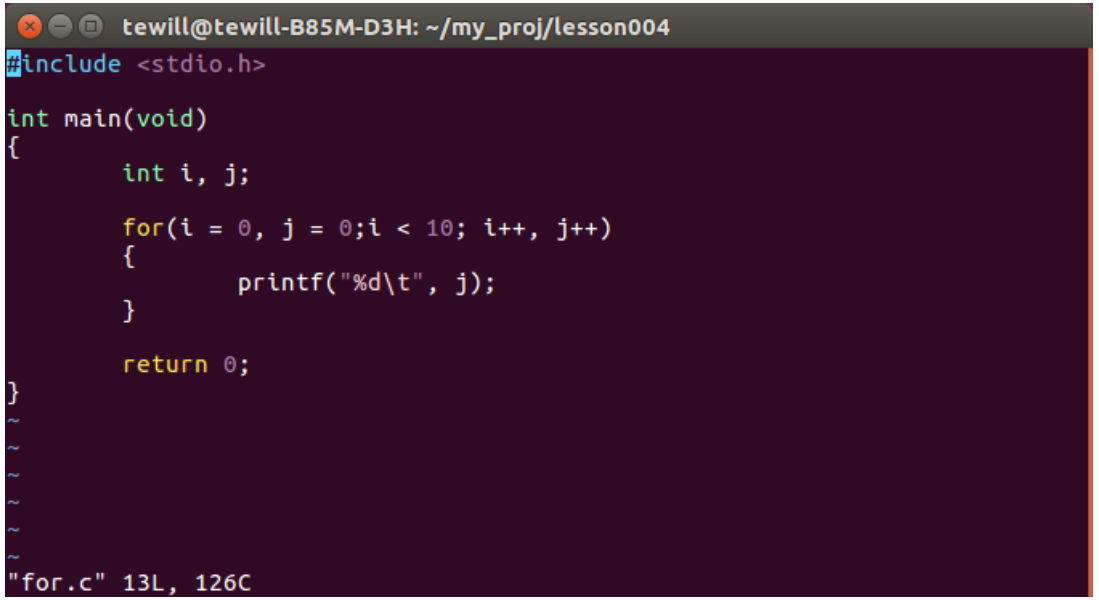
for 문 이란?

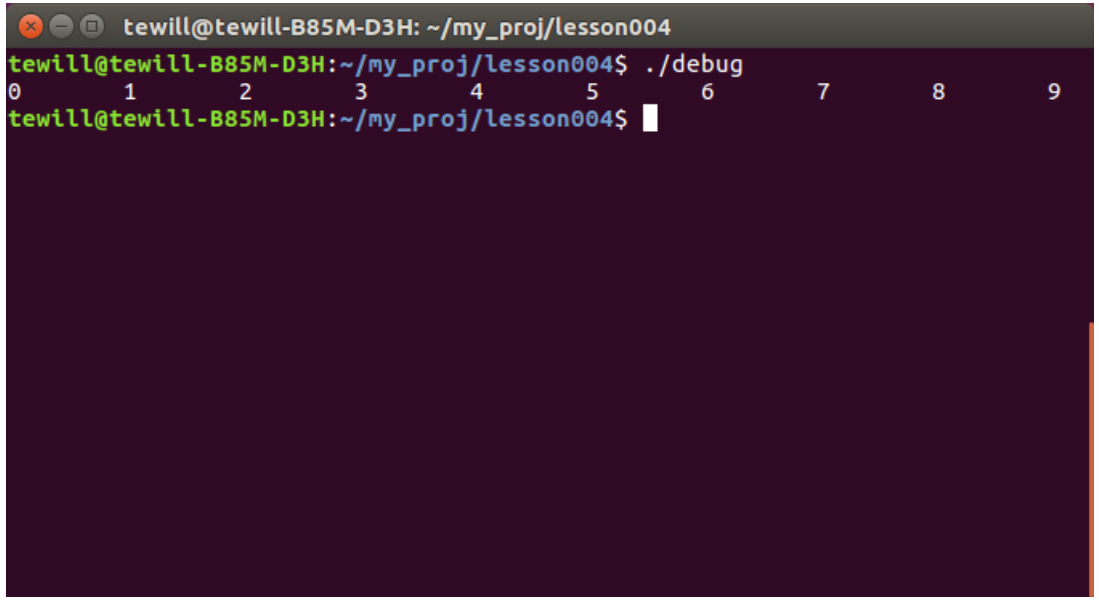
- 정해진 횟수만큼, 지정된 동작을 반복 행동을 한다.

for 문의 형식?

- Ex) for(i = 0 [초기값] ; i < 10 [조건식] ; i++ [증감식])
- 이때, 초기값과 증감식은 복수의 값이 들어갈 수 있다.
- Ex) for(i = 0, j = 0 ; i < 10 ; i++)
- 주의 사항 : 리눅스의 경우 초기값에서 변수 선언이 불가능 하다.

복습_for


tewill@tewill-B85M-D3H: ~/my_proj/lesson004
#include <stdio.h>
int main(void)
{
int i, j;
for(i = 0, j = 0;i < 10; i++, j++)
{
printf("%d\t", j);
}
return 0;
}
"for.c" 13L, 126C


tewill@tewill-B85M-D3H: ~/my_proj/lesson004
tewill@tewill-B85M-D3H:~/my_proj/lesson004$ ./debug
0 1 2 3 4 5 6 7 8 9
tewill@tewill-B85M-D3H:~/my_proj/lesson004$

복습_goto

goto 문 이란?

- 지정된 레이블로 강제 이동시킨다.
- 반복적인 jmp로 인한 파이프 라인 손실을 줄일 수 있다.
- Ex) break 로 인한 연속된 jmp는 한 번 호출당 3clock 이상의 손실이 발생한다.

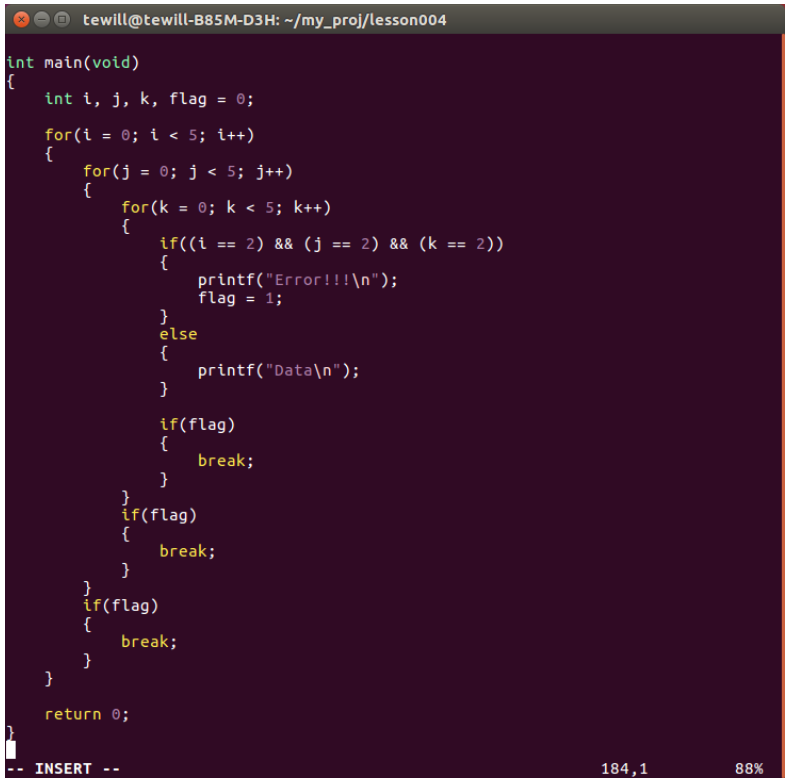
Goto 문의 형식?

- Ex) goto err_name;
- … ;
- err_name:
- …;

복습_goto

goto 를 사용할 경우

Break 을 통해 탈출할 경우



복습_파이프 라인

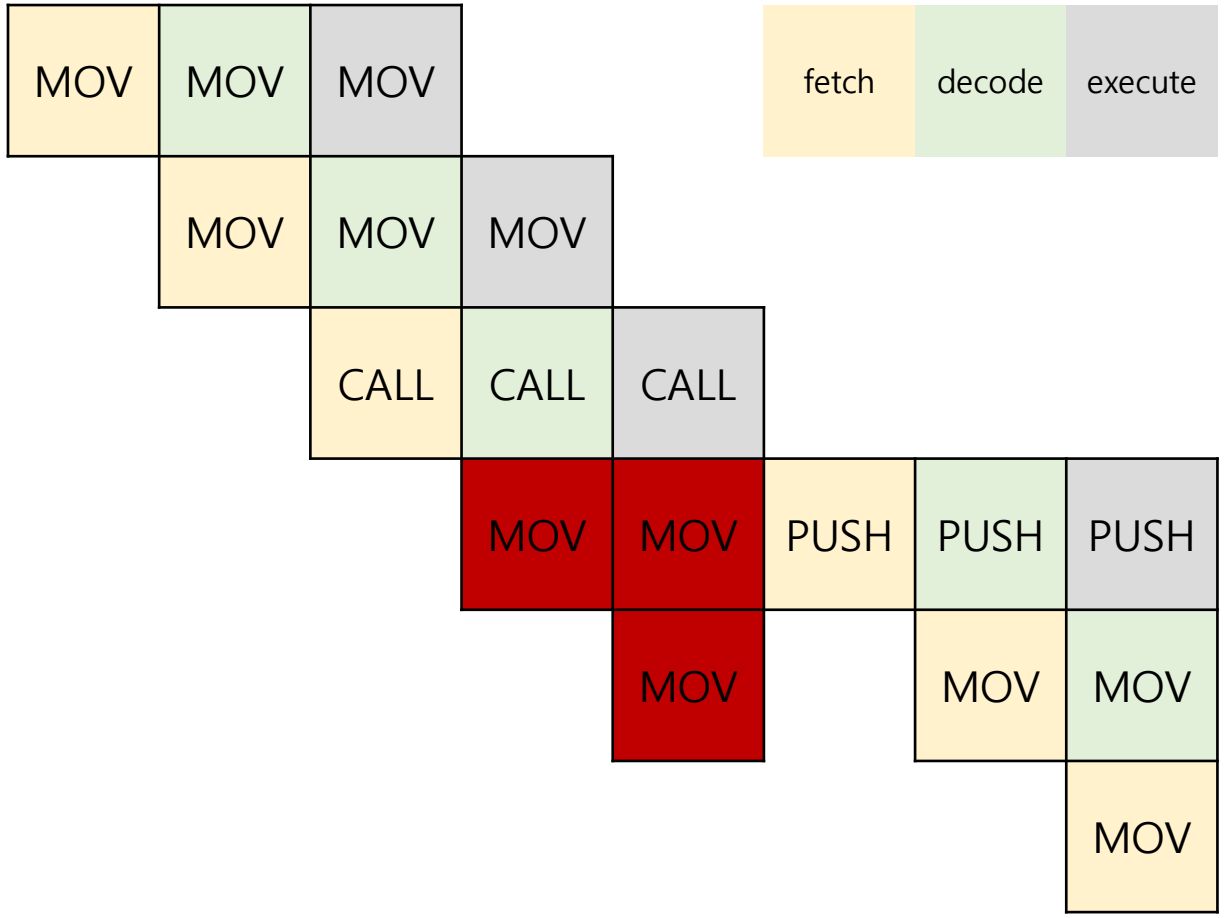
파이프 라인 이란?

- 컴퓨터가 명령어를 읽고 실행하는 과정이다.
- CPU에 구성된 회로이기 때문에, 모든 모듈이 동시에 동작되는 과정이다.

파이프 라인의 형식?

- 보드마다 다르며, 가장 단순한 구조가 fetch, decode, execute 3 단계 로 이루어져 있다.
- Fetch는 실행할 명령어를 받아온다.
- Decode는 받아온 명령어가 어떤 명령어인지 해석한다.
- Execute는 해석한 명령어를 실행한다.

복습_파이프 라인



jmp 로 인한 파이프 라인 손실

예를 들어 call 이 실행 될 경우,
아래에 진행되고 있던 fetch mov 2개와
decode 1개 가 손실되는 것을 볼 수 있다.

문제은행_1번.

스키장에서 스키 장비를 임대하는데 37500원이 든다.
또 3일 이상 이용할 경우 20%를 할인 해준다.
일주일간 이용할 경우 임대 요금은 얼마일까 ?
(연산 과정은 모두 함수로 돌린다)

문제은행_1번.

tewill@tewill-B85M-D3H: ~/my_proj/lesson004

```c
#include <stdio.h>

int discount(int day, int money)
{
        int result = 0, i;

        for(i = 0; i < day; i++)
                result += money;

        if(day >= 3)
                result *= 0.8;

        return result;
}

int main(void)
{
        printf("7일 임대 요금 : %d\n", discount(7, 37500));
        return 0;
}
~
"que01.c" 20L, 263C
```

tewill@tewill-B85M-D3H: ~/my_proj/lesson004

```
tewill@tewill-B85M-D3H:~/my_proj/lesson004$ ./debug
7일 임대 요금 : 210000
tewill@tewill-B85M-D3H:~/my_proj/lesson004$
```

문제은행_3번.

1 ~ 1000사이에 3의 배수의 합을 구하시오.

문제은행_3번.

tewill@tewill-B85M-D3H: ~/my_proj/lesson004

```
#include <stdio.h>

int addmul(int min, int max)
{
        int result = 0;

        for( ;min <= max; min++)
        {
                if(!(min%3))
                        result += min;
        }
        return result;
}

int main(void)
{
        printf("1~1000 사이 3의 배수 합 : %d\n", addmul(1, 1000));
        return 0;
}
~
~
"que03.c" 19L, 251C                                   1,1           All
```

tewill@tewill-B85M-D3H: ~/my_proj/lesson004

```
tewill@tewill-B85M-D3H:~/my_proj/lesson004$ ./debug
1~1000 사이 3의 배수 합 : 166833
tewill@tewill-B85M-D3H:~/my_proj/lesson004$
```

문제은행_4번.

1 ~ 1000사이에 4나 6으로 나눠도 나머지가 1인 수의 합을 출력하라.

문제은행_4번.

tewill@tewill-B85M-D3H: ~/my_proj/lesson004

```
#include <stdio.h>

int cal(int min, int max, int num1, int num2)
{
        int result = 0;

        for( ; min < max ; min++)
        {
                if(1 == min%num1 || 1 == min%num2)
                        result += min;
        }

        return result;
}

int main(void)
{
        printf("1~1000 사이에  4나 6으로 나눠도\n 나머지가 1인 수의 합 : %
d\n", cal(1,1000,4,6));
        return 0;
}
"que04.c" 20L, 334C                                   1,1           All
```

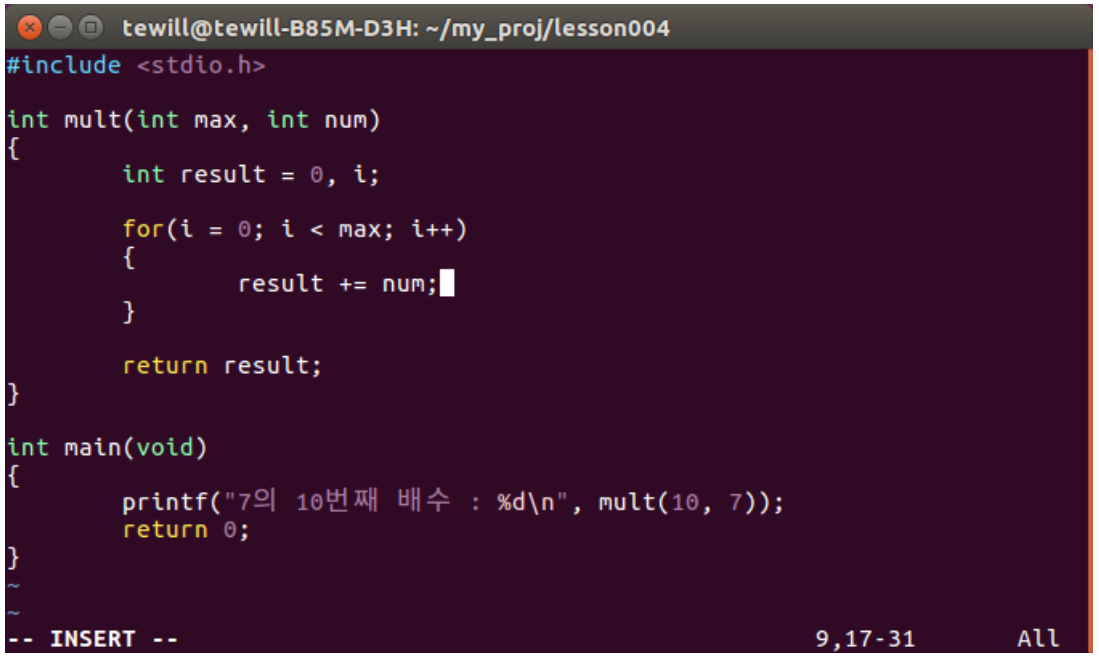
tewill@tewill-B85M-D3H: ~/my_proj/lesson004

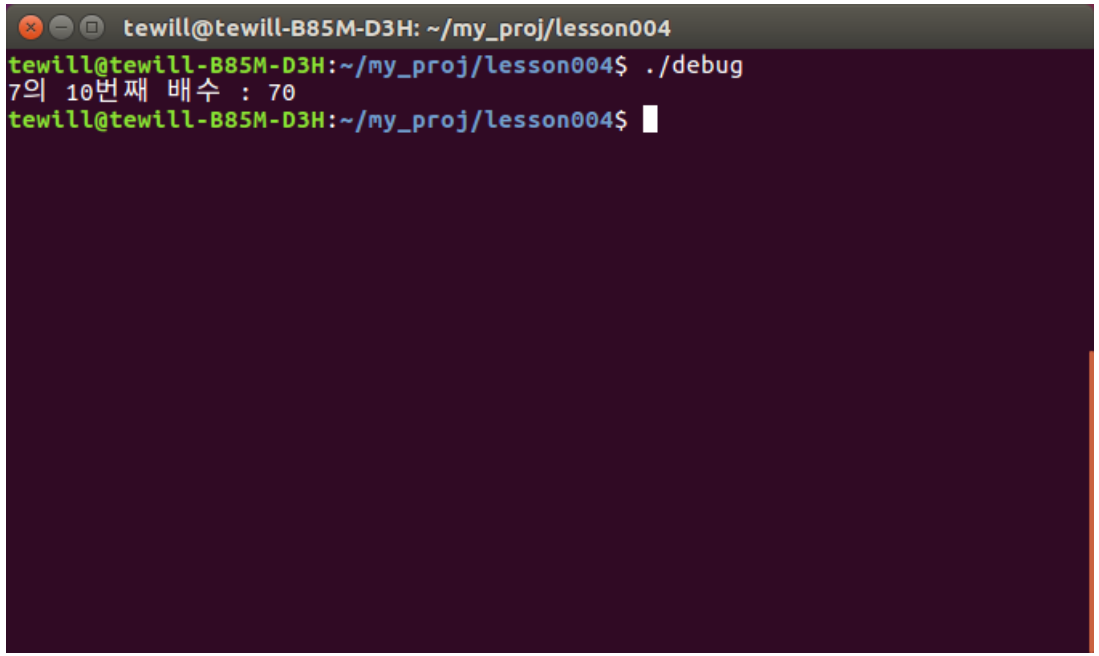
```
tewill@tewill-B85M-D3H:~/my_proj/lesson004$ ./debug
1~1000 사이에  4나 6으로 나눠도
 나머지가 1인 수의 합 : 166167
tewill@tewill-B85M-D3H:~/my_proj/lesson004$
```

문제은행_5번.

7의 배수로 이루어진 값들이 나열되어 있다고 가정한다.
함수의 인자(input)로 항의 갯수를 받아서 마지막 항의 값을 구하는 프로그램을 작성하라.

문제은행_5번.


tewill@tewill-B85M-D3H: ~/my_proj/lesson004
#include <stdio.h>
int mult(int max, int num)
{
int result = 0, i;
for(i = 0; i < max; i++)
{
result += num;
}
return result;
}
int main(void)
{
printf("7의 10번째 배수 : %d\n", mult(10, 7));
return 0;
}
-- INSERT -- 9,17-31 All


tewill@tewill-B85M-D3H: ~/my_proj/lesson004
tewill@tewill-B85M-D3H:~/my_proj/lesson004$ ./debug
7의 10번째 배수 : 70
tewill@tewill-B85M-D3H:~/my_proj/lesson004$

문제은행_10번.

구구단을 만들어보시오.

문제은행_10번.

tewill@tewill-B85M-D3H: ~/my_proj/lesson004

```c
#include <stdio.h>

void mult(int num1, int num2)
{
        int i, j;
        for(i = 2; i <= num1; i++)
        {
                for(j = 1; j <= num2; j++)
                {
                        printf("%d * %d = %d\n", i, j, i * j);
                }
                printf("\n");
                j = 1;
        }
}

int main()
{
        mult(9, 9);
        return 0;
}
```

"que10.c" 21L, 244C 1,1 All

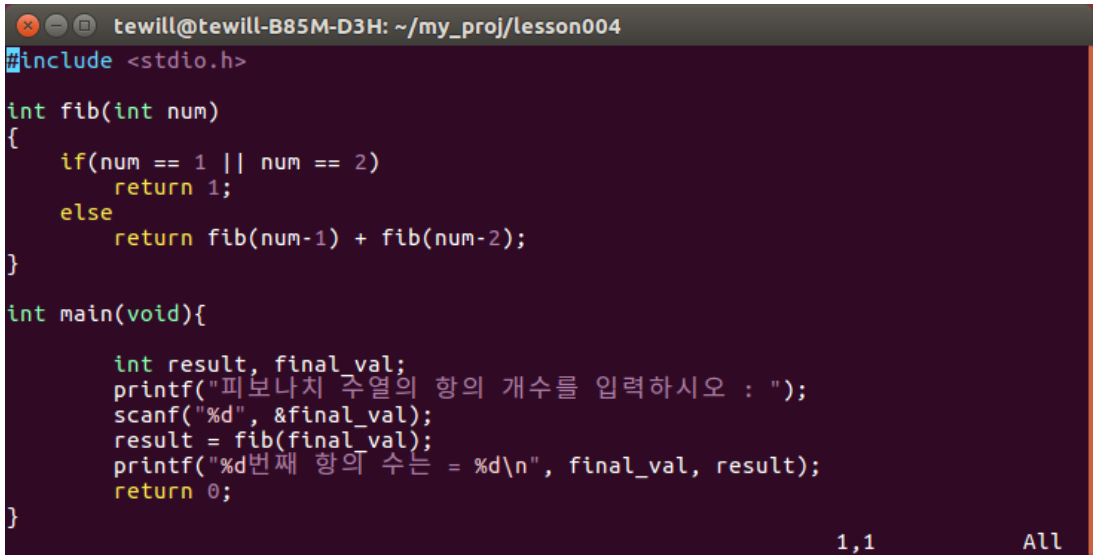
tewill@tewill-B85M-D3H: ~/my_proj/lesson004

```
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72

9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81

tewill@tewill-B85M-D3H:~/my_proj/lesson004$
```

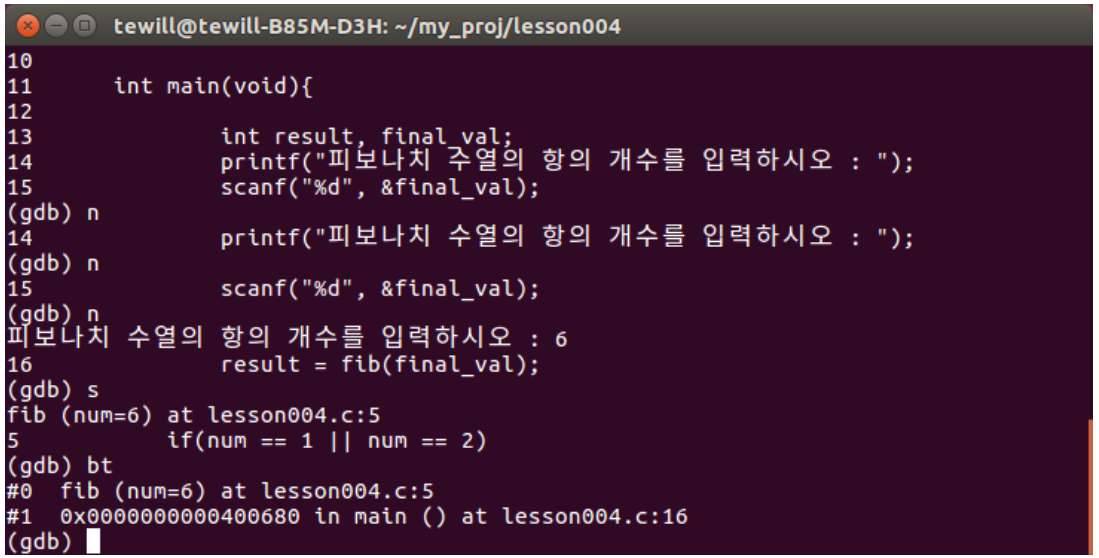
재귀 함수



재귀 함수 란?

자기 자신을 부르는 함수를 말한다.

재귀 함수

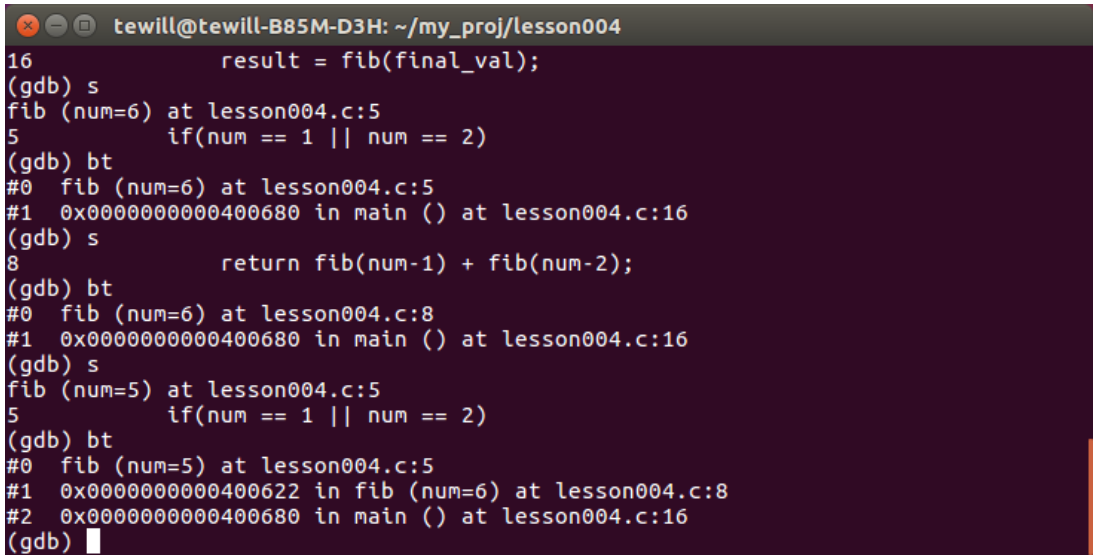


1번째 함수 진입.

Num = 6 의 값을 갖고 있다.

자신을 2개 호출한다.

재귀 함수

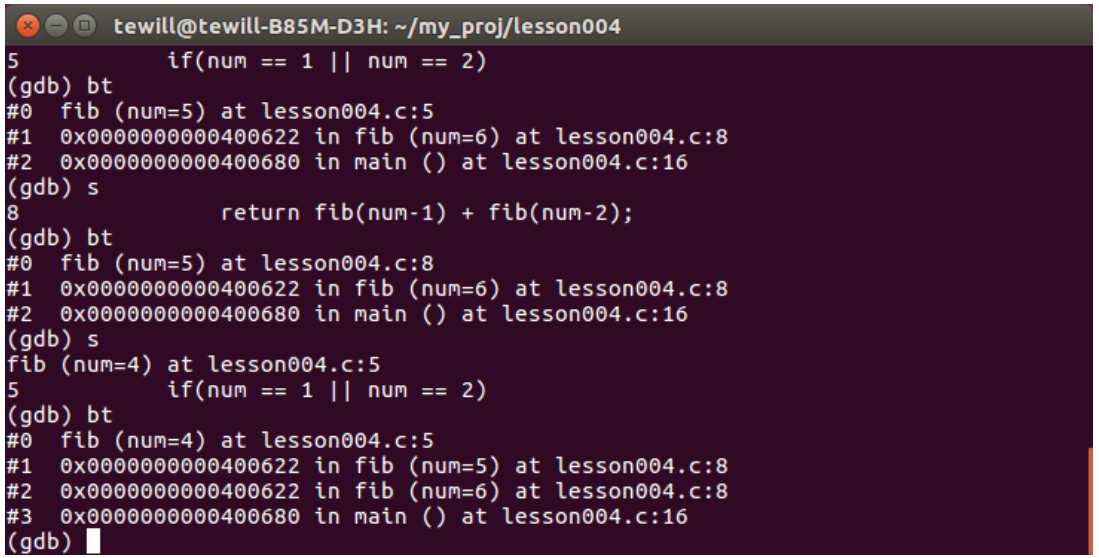


2번째 함수 진입.

Num = 5 의 값을 갖고 있다.

자신을 2개 호출한다.

재귀 함수

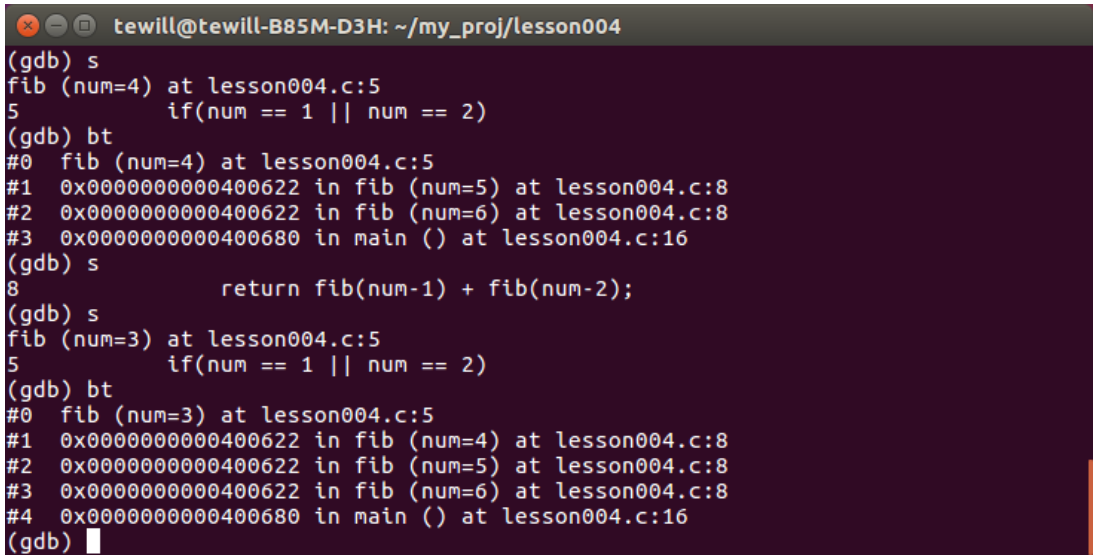


3번째 함수 진입.

Num = 4 의 값을 갖고 있다.

자신을 2개 호출한다.

재귀 함수

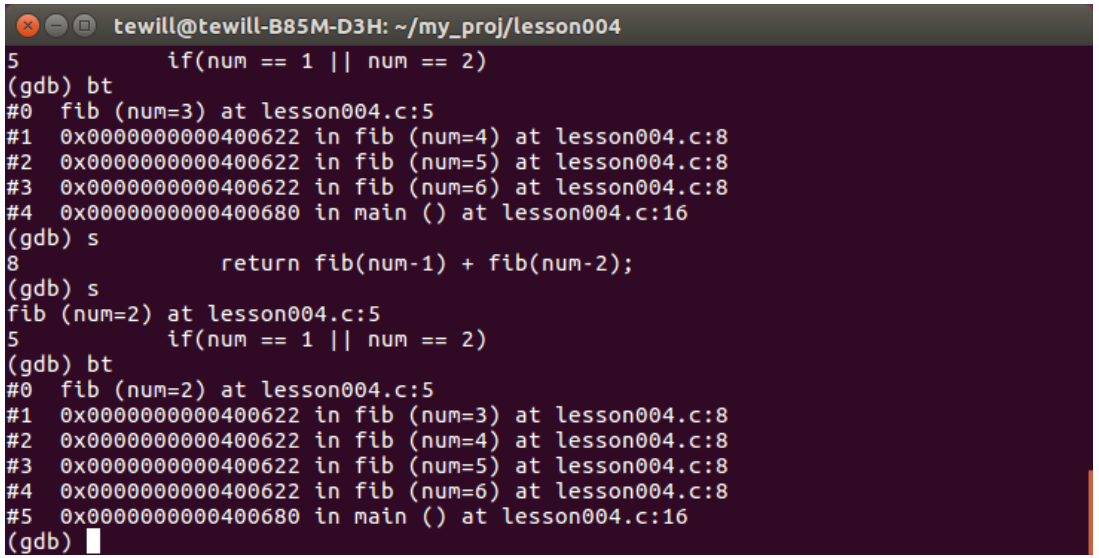


4번째 함수 진입.

Num = 3 의 값을 갖고 있다.

자신을 2개 호출한다.

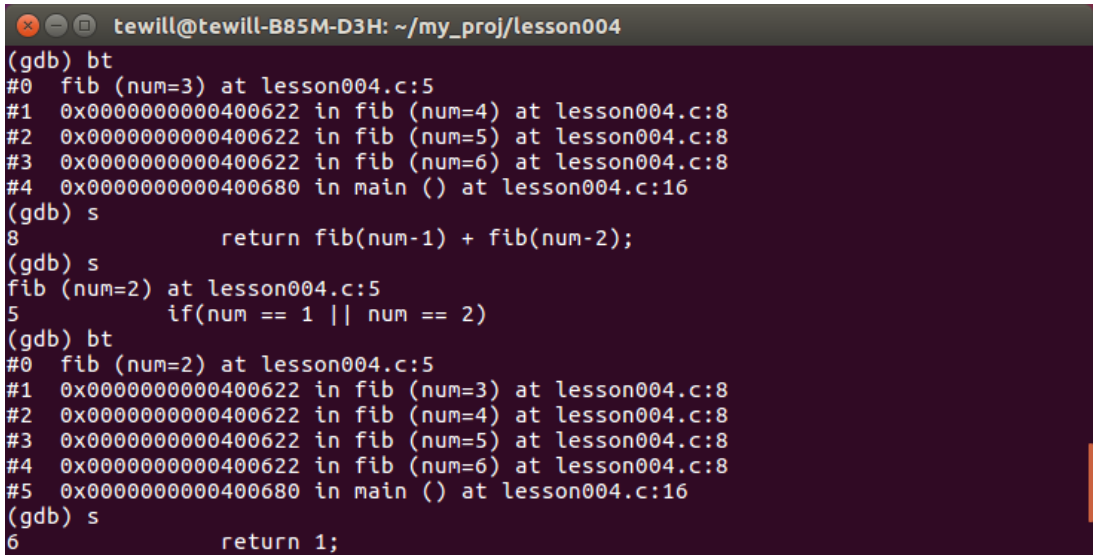
재귀 함수



5번째 함수 진입.

Num = 2로 조건이 만족한다.

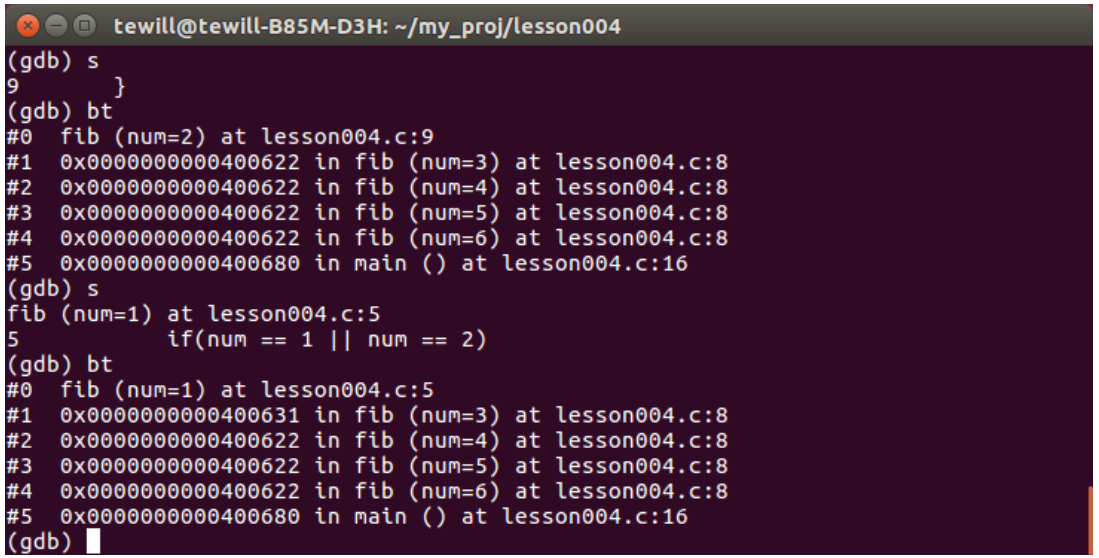
재귀 함수



Return 1을 실행함.

Return 값을 4번째 함수에 올려준다.

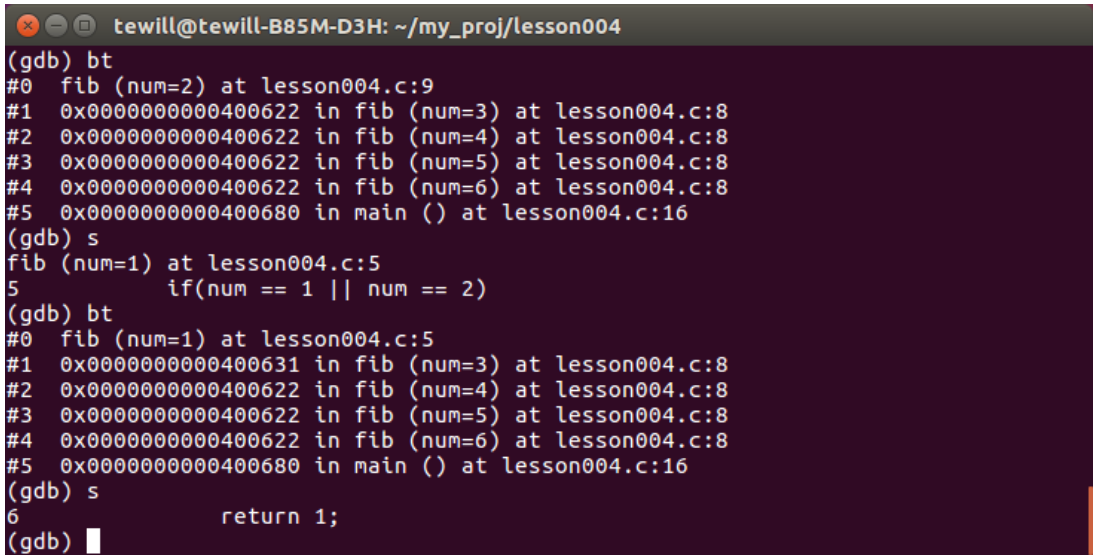
재귀 함수



4번째 함수에서 미실행한 함수 호출을 실시.

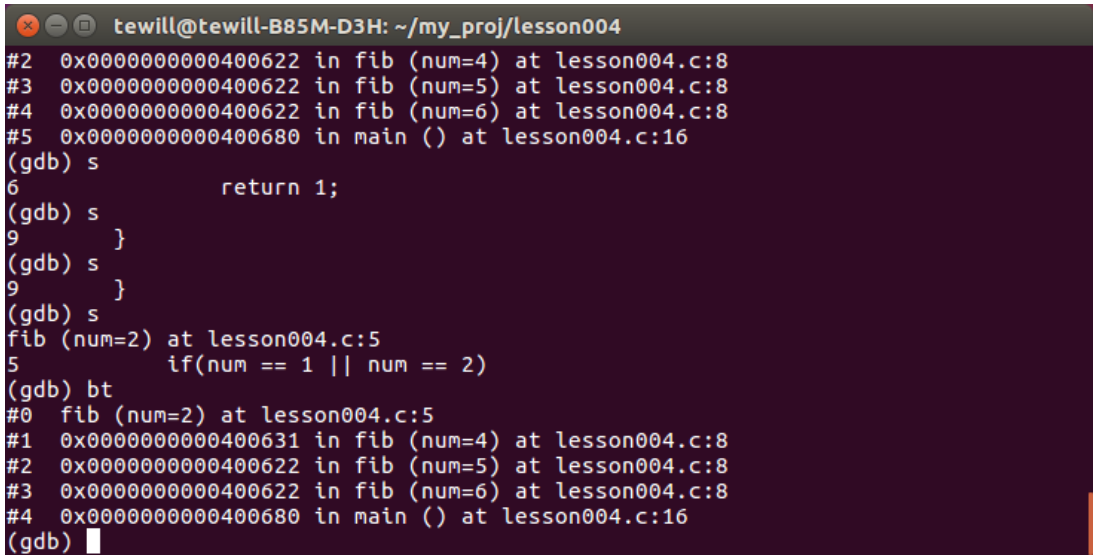
Num = 1 이다.

재귀 함수



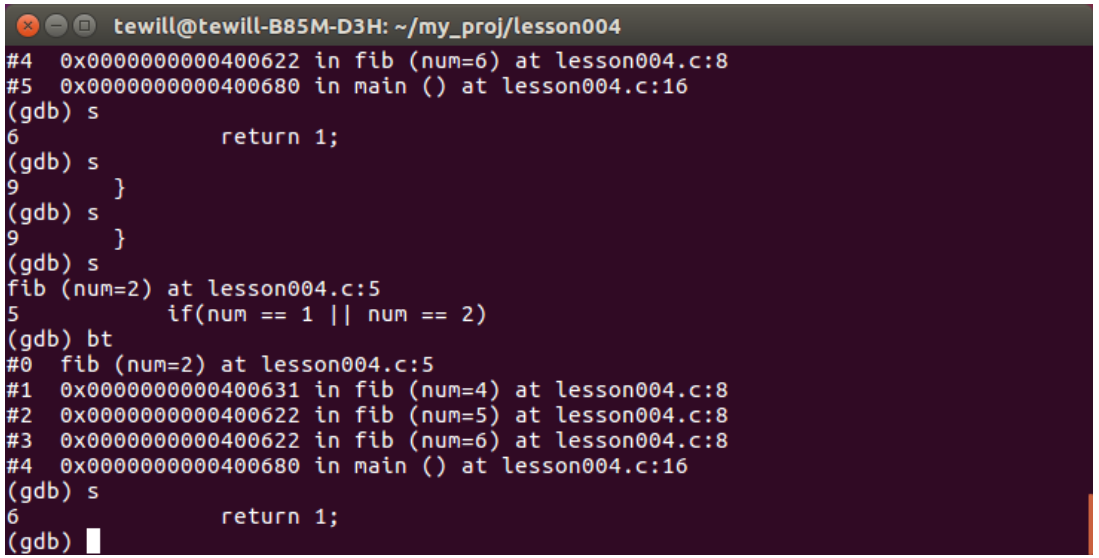
4번째 함수에 Return 1을 한다.

재귀 함수



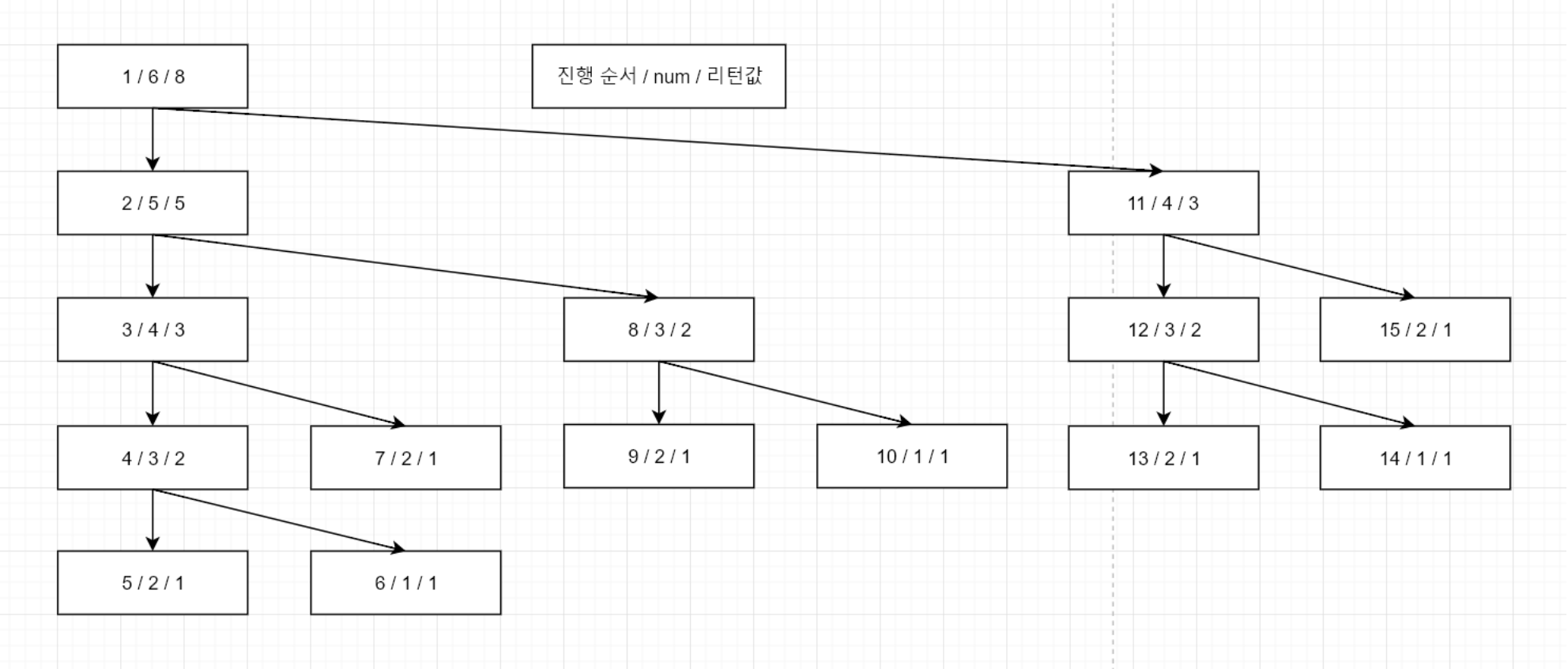
4번째 함수에서 리턴한 값을 더해 2를 3번째 함수에 리턴한다.

재귀 함수



3번째 함수에서 미실시한 자신을 호출한다.
해당 과정을 반복하여, 최종적으로 8의 값을 리턴한다.

재귀 함수


진행 순서 / num / 리턴값
1 / 6 / 8
2 / 5 / 5
3 / 4 / 3
4 / 3 / 2
5 / 2 / 1
6 / 1 / 1
7 / 2 / 1
8 / 3 / 2
9 / 2 / 1
10 / 1 / 1
11 / 4 / 3
12 / 3 / 2
13 / 2 / 1
14 / 1 / 1
15 / 2 / 1