11. 재귀함수(recursive function)

재귀함수는 함수 내에서 **자신**을 다시 호출하는 것이다.**(되부름, 재귀호출, 자기호출, 순환호출)**
재귀함수를 설명하기 전에 **반복함수**와 **재귀함수(순환함수)**를 비교해 본다.
다음은 **계승함수** n!을 구하는 과정을 반복함수와 재귀함수로 나타낸 것이다.

```
int factorial(int n)        //반복함수
{
    int i, f;

    f = 1;
    for(i=2; i<=n; i++)
      f*=i;

    return f;
}
void main()
{
    int y;
    y = factorial(4);
    printf("%d\n", y);
}
```

```
int factorial(int n)        //재귀함수
{
    int i, f;

    if(n<=1)                //완료조건(중요!)
      f = 1;
    else
      f = n * factorial(n-1);
    return f;
}
void main()
{
    int y;
    y = factorial(4);
    printf("%d\n", y);
}
```

◇ f = n * factorial(n-1) //재귀함수

↓ n = 4이면

f = 4 * factorial(3)
= 4 * 3 * factorial(2)
= 4 * 3 * 2 * factorial(1) //완료조건 : factorial(1) = 1
= 4 * 3 * 2 * 1 //factorial(1)이면 f = 1이므로 factorial(1) = 1이다.
= 24

◈ **반복함수와 순환함수 정리**

① 순환함수로 구현된 **모든 알고리즘**은 순환을 사용하지 않는 방법으로 표현할 수 있다.
② 순환 알고리즘은 반복에 비해 명확하고 간결하게 표현된다.(세상 원리)
③ 순환 알고리즘은 반복 알고리즘에 비해 일반적으로 수행시간이 느리다.
④ 순환 알고리즘은 반복 알고리즘에 비해 기억공간이 더 많이 필요하다.(별도 Stack을 사용하므로)
⑤ 문제 자체가 순환적으로 정의되어 있으면, 순환함수로 처리하는 것이 명확하고 간단해 진다.
예를 들면, 2개의 이진트리에 대한 동일성 비교 같은 문제이다.
⑥ 순환함수에는 **완료조건**이 필요하다.(그렇지 않으면 Stack overflow가 발생하여 시스템은 정지)

예제 1	순환(recursive; 되부름, 재귀호출)으로 처리된 함수

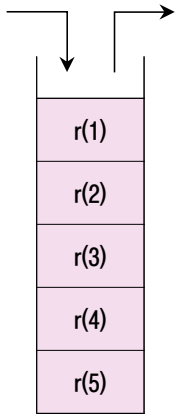
```
int r(int   n){
    int h;

    if ( n == 1 )              //완료조건
        h = 1;
    else
        h = n + r(n - 1);      //되부름

    return   h;
}

void main(void){
    int d, k = 5;

    d = r(k);
    printf("실행결과 : %d\n", d);
}
실행결과 : 15
```



- 함수가 되부름되면 함수의 상태는 순차적으로 스택에 적재되고
- 함수 종료시 return문이 실행되면서 스택의 내용은 하나씩 제거된다.
- 이때 되부름 순간에 적용된 **연산**이 **수행**된다.(여기서는 **+ 연산**이 수행된다)

```
h = n + r(n - 1);
        ↓n = 5일 때,
h = 5 + r(4)
  = 5 + 4 + r(3)
  = 5 + 4 + 3 + r(2)
  = 5 + 4 + 3 + 2 + r(1) 처럼 내부적으로 진행된다.
                    └→ r(1)의 값은 1이다.
                       마지막, r(1)인 경우에 h = 1;로 함수값을 반환하므로
h = 5 + 4 + 3 + 2 + 1 = 15
```

예제 2	다음 C 프로그램 출력 결과는? – 되부름 수식에 음수가 포함된 경우

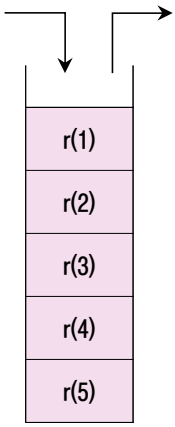
```
int r(int n)
{
    int h;

    if ( n == 1 )              //완료조건
        h = 1;
    else
        h = n - r(n - 1);      //되부름

    return  h;
}

void main(void)
{
    int d, k = 5;

    d = r(k);
    printf("실행결과 : %d\n", d);
}
실행결과 : 3
```



- 함수가 되부름되면 함수의 상태는 순차적으로 스택에 적재되고
- 함수 종료시 return문이 실행되면서 스택의 내용은 하나씩 제거된다.
- 이때 되부름 순간에 적용된 **연산**이 **수행**된다.(여기서는 **–** **연산**이 수행된다)

// 되부름 수식에 음수가 포함된 경우 – 완료조건을 구한 후에 수식의 값을 찾는다.

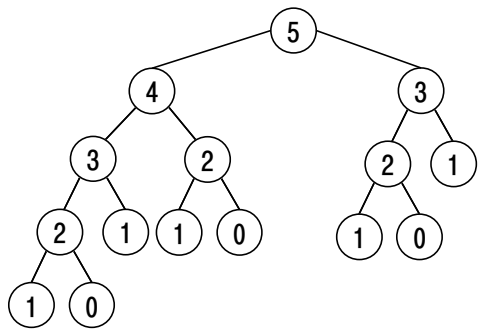
- r(1) = 1 //완료조건
- r(2) = 2 - r(1) = 2 - 1 = 1
- r(3) = 3 - r(2) = 3 - 1 = 2
- r(4) = 4 - r(3) = 4 - 2 = 2
- r(5) = 5 - r(4) = 5 - 2 = 3 //출력결과 : 3

예제 3	다음 함수 fib()를 사용하여 fib(5)를 실행했을 때, fib(5)를 포함한 fib() 함수의 총 재귀호출 횟수와 최종 리턴 값은?

```
int fib(int n)
{
  if(n<=0) return 0;    //완료조건
  if(n==1) return 1;    //완료조건
  else return fib(n-1) + fib(n-2);
}
```

◈ 함수 fib()가 호출되는 횟수 : 15번

함수 fib()가 호출되는 횟수는 다음처럼 재귀트리를 그려서 풀면 된다.



• 동그라미 수가 **재귀호출 횟수**이다.(15번)

◈ 함수 fib()의 리턴 값 : 5

• fib(0) = 0 **//완료조건**
• fib(1) = 1 **//완료조건**
• fib(2) = fib(1) + fib(0) = 1 + 0 = 1
• fib(3) = fib(2) + fib(1) = 1 + 1 = 2
• fib(4) = fib(3) + fib(2) = 2 + 1 = 3
• fib(5) = fib(4) + fib(3) = 3 + 2 = **5** → 리턴 값

[Tip] 전산 시험에서 재귀호출 문제 유형(출제빈도가 매우 높음)

• 재귀함수가 호출되는 횟수 구하기 : 재귀트리로 해결
• 재귀함수의 리턴 값 구하기 : 완료조건으로 해결
• 재귀함수의 출력 결과 값 구하기 : 가장 어려운 부분(뒤에서 다룸)

예제 4	다음 C 프로그램 출력 결과는?

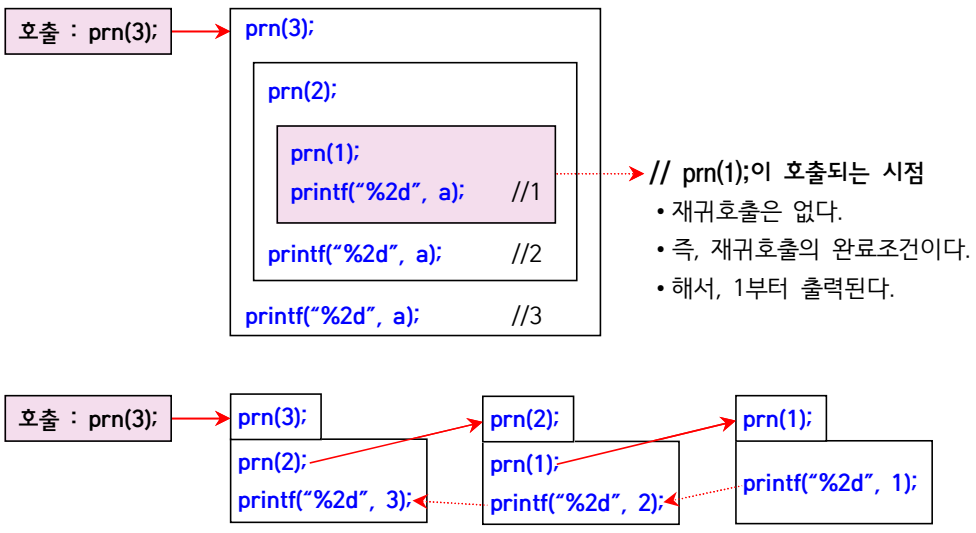
```
-----------------------
#include <stdio.h>
void prn(int a){
   if(a>1)        //완료조건
      prn(a-1);
   printf("%2d", a);
}
void main(){ prn(3); }
-----------------------
```

① 1 2 3　　② 2 3

③ 3 2 1　　④ 3 2

♤ **재귀호출**

- 재귀호출은 자신을 그대로 호출하지만, 내부적으로 변수 값은 변할 수 있다.(당연!)
- 재귀호출은 변수 값이 어떻게 변하느냐? 에 따라서 실행 결과는 다르게 된다.(당연!)
- 다음은 재귀호출 수행 과정을 2종류의 그림으로 나타낸 것이다.**(이해하기 쉬운 것 택일!)**



- 주어진 그림을 보면, 출력 결과가 **1 2 3**라는 것을 알 수 있을 것이다.
- 재귀호출문과 출력문 위치에 따라서 실행 결과는 다르게 된다.

정답 : ①

예제 5	다음 C 프로그램 출력 결과는?

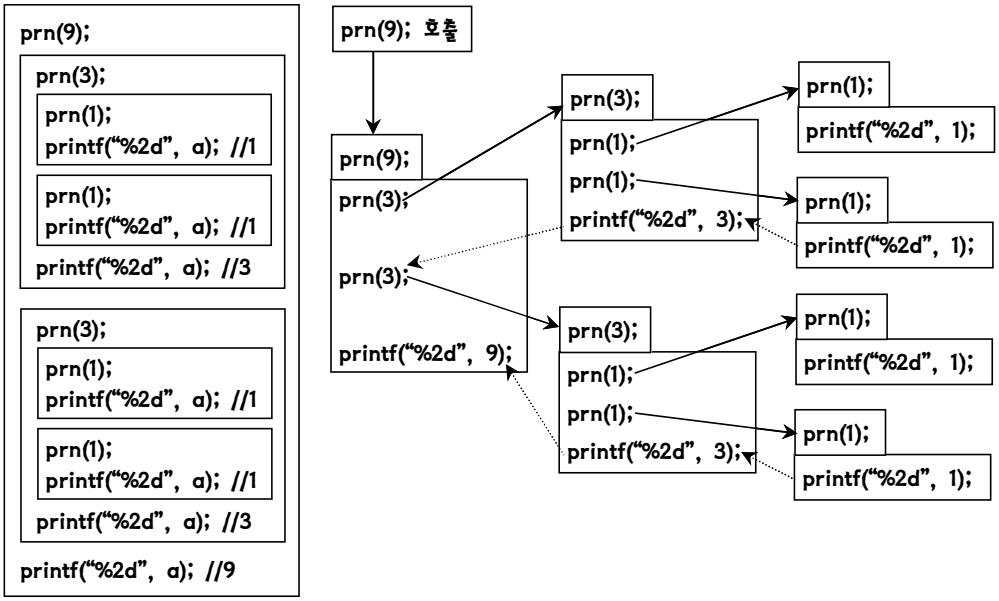
```
-------------------------
#include <stdio.h>
void prn(int a) {
  if(a>1){
    prn(a/3);
    prn(a/3);
  }
  printf("%2d", a);
}
void main(){ prn(9); }
-------------------------
```

① 1　　② 1 1 3 1 1 3 9

③ 9　　④ 9 3 1 1 3 1 1

♤ 재귀호출

• 주어진 재귀호출 수행 과정을 2종류의 그림으로 나타낸 것이다.(이해하기 쉬운 것 택일!)



• 주어진 그림을 보면, 출력 결과가 1 1 3 1 1 3 9라는 것을 알 수 있을 것이다.

• 재귀호출은 자신을 그대로 호출하지만, 내부적으로 변수 값은 변할 수 있다.(당연!)

정답 : ②

예제 6	다음 C 프로그램 출력 결과는?

```
-------------------------
#include <stdio.h>
void prn(int a) {
   printf("%2d", a);
   if(a>1){
      prn(a/3);
      prn(a/3);
   }
}
void main(){ prn(9); }
-------------------------
```

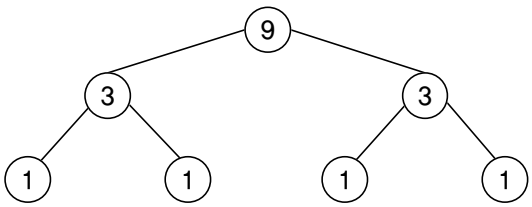
① 1　　② 1 1 3 1 1 3 9

③ 9　　④ 9 3 1 1 3 1 1

♤ **재귀호출 - 재귀트리 이용**

• 먼저, 주어진 재귀호출 문제를 재귀트리로 나타내면 다음과 같다.

• prn(9);를 수행



• 전위운행(중,좌,우) : 9 3 1 1 3 1 1

• 후위운행(좌,우,중) : 1 1 3 1 1 3 9

void prn(int a)
{
  printf("%2d", a); //출력문이 앞에 있음
  if(a>1){
  prn(a/3);
  prn(a/3);
  }
}
	void prn(int a)
	{
	  if(a>1){
	  prn(a/3);
	  prn(a/3);
	  }
	  printf("%2d", a); //출력문이 뒤에 있음
	}

출력 결과 : 9 3 1 1 3 1 1	출력 결과 : 1 1 3 1 1 3 9

• 출력문이 **앞**에 있는 경우 : 트리를 **전위**운행한 결과와 같다.

• 출력문이 **뒤**에 있는 경우 : 트리를 **후위**운행한 결과와 같다.

정답 : ④

기출문제 분석

1. 다음 C 프로그램의 실행 결과로 옳은 것은? [2014년 계리직]

```
---------------------------------
#include <stdio.h>
int sub(int n)
{
  if(n==0) return 0;
  if(n==1) return 1;
  return (sub(n-1) + sub(n-2));
}
void main()
{
  int a = 0;
  a = sub(4);
  printf("%d", a);
}
---------------------------------
```

① 0 ② 1
③ 2 ④ 3

♤ 재귀호출

- sub(0) = 0 //완료조건
- sub(1) = 1 //완료조건

•sub(n) = sub(n-1) + sub(n-2)

↓n = 4일 때,

- sub(4) = sub(3) + sub(2)
  = sub(2) + sub(1) + sub(2)
  = sub(1) + sub(0) + sub(1) + sub(2)
  = sub(1) + sub(0) + sub(1) + sub(1) + sub(0)
  = 1 + 0 + 1 + 1 + 0
  = **3**

정답 : ④

2. 다음 C 언어로 작성된 프로그램의 실행 결과에서 세 번째 줄에 출력되는 것은? [2015년 국가 9급]

```
#include <stdio.h>
int func(int num)
{
  if(num == 1)
    return 1;
  else
    return num * func(num - 1);
}
int main()
{
  int i;
  for(i = 5; i >= 0; i--)
  {
    if(i % 2 == 1)
      printf("func(%d) : %d\n", i, func(i));
  }
  return 0;
}
```

① func(3) : 6　　② func(2) : 2
③ func(1) : 1　　④ func(0) : 0

♤ **재귀호출**

• 먼저, 세 번째 줄에는 1이 출력된다.

• **if(i % 2 == 1) printf("func(%d) : %d\n", i, func(i));**
　　↓
• i 값이 홀수일 때 조건이 참이 되어 func(i)이 호출된다.
　　↓
• **첫 번째 줄 : func(5) : 120**　　//5 × 4 × 3 × 2 × 1 = 120
• **두 번째 줄 : func(3) : 6**　　//3 × 2 × 1 = 6
• **세 번째 줄 : func(1) : 1**　　//func(1)이면 return 1

정답 : ③

3. 다음 C 프로그램의 출력 값은? [2015년 지방 9급]

```
#include <stdio.h>
int recur(int a, int b)
{
  if (a<=1)
    return a * b;
  else
    return a * recur(a-1, b+1) + recur(a-1, b);
}
int main()
{
  int a = 3, b = 2;
  printf("%d\n", recur(a, b));
}
```

① 24 ② 30
③ 41 ④ 52

♤ **재귀호출**

• 재귀호출에서 연산우선순위가 다른 연산자가 섞여 있을 때는 조심해야 한다.

• recur(a, b) = a * recur(a-1, b+1) + recur(a-1, b)
• recur(3, 2)
↓
= 3 * recur(2, 3) + recur(2, 2)
= 3 * recur(2, 3) + [2 * recur(1, 3) + recur(1, 2)] → 대괄호부터 연산 실시
= 3 * recur(2, 3) + [2 * 3 + 2] → a<=1이면 return a * b;
= 3 * recur(2, 3) + 8
= 3 * [2 * recur(1, 4) + recur(1, 3)] + 8
= 3 * [2 * 4 + 3] + 8
= 3 * 11 + 8
= **41**

정답 : ③

4. 다음 C 언어로 작성된 함수는 정수 배열의 원소들의 총합을 구하여 리턴하는 함수이다. ㉠의 위치에 들어가야 할 코드는? (단, 함수의 첫 번째 인자는 배열의 시작 위치를 나타내는 포인터, 두 번째 인자는 배열의 크기임) [2016년 국회 9급]

```
-----------------------------------------
int sum(int *data, const int dsize)
{
    if(dsize > 0)
        return ㉠;
    return 0;
}
-----------------------------------------
```

① *data + sum(data − 1, dsize + 1)
② data + sum(data − 1, dsize + 1)
③ *data + sum(data − 1, dsize − 1)
④ *data + sum(data + 1, dsize − 1)
⑤ data + sum(data + 1, dsize − 1)

♤ **재귀호출을 이용한 배열원소 누적**

```
// 먼저, 프로그램을 완성시키면 다음과 같다.
int sum(int *data, const int dsize)
{
    if(dsize > 0)
        return *data + sum(data + 1, dsize - 1);
    return 0;
}
void main(){
  int a[] = {1, 2, 3, 4, 5};      //정수 배열
  int h;
  h = sum(a, 5);
  printf("%d\n", h);              //15 출력
}
```

*data + sum(data + 1, dsize − 1);
↓ ↳ 정수 배열원소의 주소를 차례로 증가시킨다.
정수 배열원소를 차례로 누적시킨다

정답 : ④

5. 다음 C 프로그램에서 main() 함수를 실행할 때 fib() 함수가 호출되는 횟수로 옳은 것은? [2017년 국회 9급]

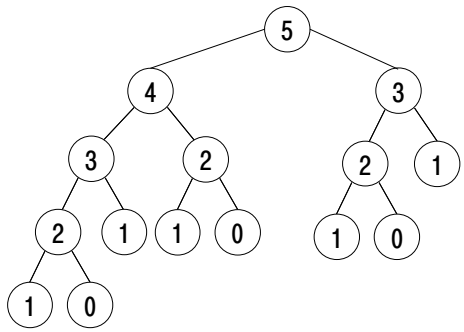
```
#include <stdio.h>
int fib(int n)
{
  if(n == 0) return 0;
  if(n == 1) return 1;
  return(fib(n-1) + fib(n-2));
}
int main(){
  fib(5);
}
```

① 3번 ② 5번 ③ 10번
④ 12번 ⑤ 15번

♤ 피보나치수열과 재귀트리

// 다음은 피보나치수열에서 f(5)을 구하는 재귀 알고리즘에 대한 재귀트리이다.

- f(5) = f(4) + f(3) → f(5)를 구하기 위해서는 f(4)와 f(3)을 필요로 한다.
- f(4) = f(3) + f(2) → f(4)를 구하기 위해서는 f(3)과 f(2)를 필요로 한다.
- f(3) = f(2) + f(1)
- f(2) = f(1) + f(0)
- f(1) = 1 → 완료조건
- f(0) = 0 → 완료조건



- 동그라미 수가 재귀호출되는 수이다.**(15번)**

정답 : ⑤

6. 다음 C 프로그램의 실행 결과는? [2017년 법무 9급]

```
#include <stdio.h>
int func(int n)
{
  if(n <= 1) return(n);
  else       return(func(n - 1) + func(n - 2));
}
int main(void)
{
  int n = 7;
  int i;
  int result = 0;
  for(i = 0; i < n; i++) result += func(i);
  printf("%d", result);
  return(0);
}
```

① 0　　② 12
③ 19　　④ 20

♤ **재귀호출**

• func(n) = func(n − 1) + func(n − 2)

• i = 0일 때, func(0) = 0
• i = 1일 때, func(1) = 1
• i = 2일 때, func(2) = func(1) + func(0) = 1 + 0 = 1
• i = 3일 때, func(3) = func(2) + func(1) = 1 + 1 = 2
• i = 4일 때, func(4) = func(3) + func(2) = 2 + 1 = 3
• i = 5일 때, func(5) = func(4) + func(3) = 3 + 2 = 5
• i = 6일 때, func(6) = func(5) + func(4) = 5 + 3 = 8

• result = 0 + 1 + 1 + 2 + 3 + 5 + 8 = **20**

정답 : ④

7. 다음 C 프로그램의 실행 결과로 옳은 것은? [2018년 국회 9급]

```
#include <stdio.h>
int recursion(int n)
{
  if (n < 5) return 1;
  else if (n % 5 == 1) return n + recursion(n - 1);
  else recursion(n - 1);
}
int main() {
  int n = recursion(16);
  printf("%d", n);
  return 0;
}
```

① 34 ② 33 ③ 31
④ 29 ⑤ 28

♤ **재귀호출**

• n = recursion(16)
= 16 + recursion(15)
= 16 + recursion(14)
= 16 + recursion(13)
= 16 + recursion(12)
= 16 + recursion(11)
= 16 + 11 + recursion(10)
= 16 + 11 + recursion(9)
= 16 + 11 + recursion(8)
= 16 + 11 + recursion(7)
= 16 + 11 + recursion(6)
= 16 + 11 + 6 + recursion(5)
= 16 + 11 + 6 + recursion(4)
= 16 + 11 + 6 + 1
= **34**

정답 : ①