

# Chapter 13

## 함수 3



01

Question

반복문을 사용하지 않고 자신의 이름을 10번 출력하는 프로그램을 작성하시오

출력 예      홍길동  
                홍길동  
                :  
                홍길동

## 소스1

```
#include <stdio.h>

void name(int n)
{
    printf("홍길동\n");
    if (n < 10) name(n + 1);
}

int main()
{
    name(1);
    return 0;
}
```

#### 설명

- 위의 문제를 반복문을 사용하여 출력한다면 다음과 같이 작성하면 된다.

```
for (i = 1; i <= 10; i++) {  
    printf("홍길동\n");  
}
```

- void name(int n)  
n번째 이름을 출력하라는 함수로 정의한 것이다. 이름을 출력한 후 지금 출력한 것이 10번째가 아니면 같은 함수를 다시 호출하여 n+1번째를 출력하도록 설계한 것이다. 따라서 반복문은 없지만 자신을 계속 호출하면서 반복이 되므로 for 문을 대신 할 수 있다.

- 이렇게 어떤 함수가 자기 자신을 다시 호출할 수 있게 만든 함수를 재귀함수라 하고 이러한 함수를 호출하는 것을 재귀호출이라 한다.
- if (n < 10) name(n + 1);  
함수를 호출한 횟수(n)가 10보다 작다면 자기 자신(name함수)을 다시 호출한다.  
이 때 현재의 횟수에 1을 더한 값(n+1)을 인수로 전달하여 몇 번째 호출인지를 표시해주는 것이다.

## 소스2

```
#include <stdio.h>

void name(int n)
{
    if (n < 1) return;
    name(n - 1);
    printf("홍길동\n");
}


int main()
{
    name(10);
    return 0;
}
```

#### 설명

- `void name(int n)`  
이름을  $n$ 번 출력하라는 함수로 정의한 것이다.  
즉 `main`함수에서 `name(10);`이라고 호출한 것은 이름을 10번 출력하라는 의미가 된다.
- `name(n - 1);`  
`printf("홍길동\n");`  
함수를 호출하여 이름을  $n-1$ 번 출력한 후 한 번을 더 출력한다.  
 $n-1$ 번을 출력한 후 한 번을 더 출력하면 결국  $n$ 번을 출력하게 되는 것이다.
- `if (n < 1) return;`  
만약  $n$ 이 0이라면 더 이상 출력할 것이 없다. 그러므로 그냥 리턴을 해주는 것이다. 만약 이 문장이 생략되면 `name` 함수는 무한히 호출되다가 스택 오버플로우 에러가 발생된다.

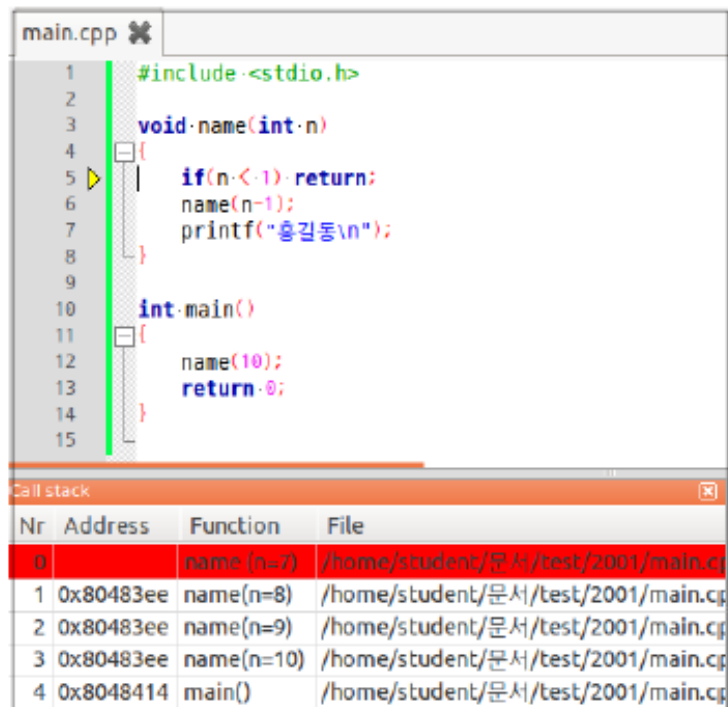
- 위와 같이 **모든 재귀함수에는 종료조건이 반드시 포함되어야 한다**. 종료조건은 조건에 맞을 때만 호출하도록 하는 방법과 일단 호출을 한 후 조건을 벗어나면 리턴하는 두 가지 방법이 있다.
  - 소스1에서는 조건에 맞을 때에만 재귀함수를 호출하도록 하였고  
`if (n < 10) name(n + 1);`
  - 소스2에서는 일단 호출을 한 후 조건을 벗어나면 리턴하도록 하였다.  
`if (n < 1) return;`



- 재귀함수를 호출하면 호출한 위치를 스택이라는 공간에 저장해 놓았다가 함수실행이 끝나고 돌아오면 그 위치부터 계속 실행이 된다. 따라서 재귀함수를 디버깅 할 때에는 스택창을 보면 함수가 호출된 상황을 확인할 수 있다. code::blocks에서 스택창을 보려면 메뉴에서 "Debug" - "Debugging windows" 또는 Debugger Toolbars의  단추에서 "Call stack"을 체크하면 되고, Visual Studio에서는 "디버그" - "창" - "호출스택" 또는 "ALT"키와 "7"을 함께 누르면 된다.

## Chapter 13

### 함수 3



The screenshot shows the code::blocks IDE with a file named `main.cpp`. The code is as follows:

```

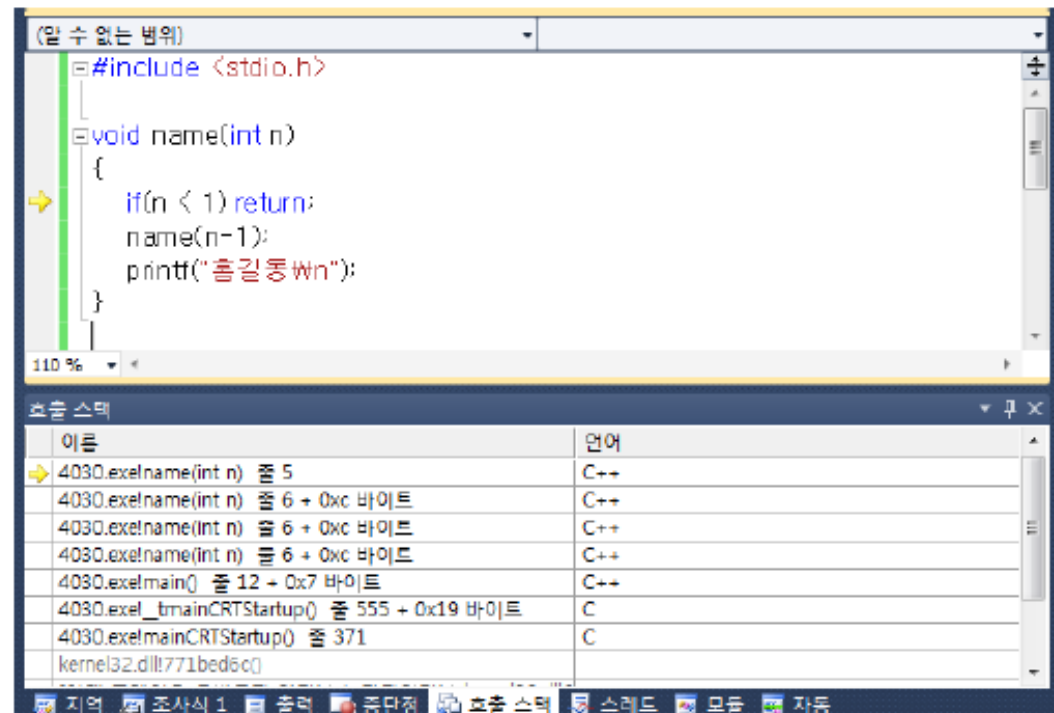
1  #include <stdio.h>
2
3  void name(int n)
4  {
5      if(n < 1) return;
6      name(n-1);
7      printf("홀길통\n");
8  }
9
10 int main()
11 {
12     name(10);
13     return 0;
14 }
15

```

Below the code editor, the 'Call stack' window is open, displaying the following table:

Nr	Address	Function	File
0		name (n=7)	/home/student/문서/test/2001/main.cj
1	0x80483ee	name(n=8)	/home/student/문서/test/2001/main.cj
2	0x80483ee	name(n=9)	/home/student/문서/test/2001/main.cj
3	0x80483ee	name(n=10)	/home/student/문서/test/2001/main.cj
4	0x8048414	main()	/home/student/문서/test/2001/main.cj

code::blocks



The screenshot shows the Visual Studio IDE with a file named `main.cpp`. The code is as follows:

```

(알 수 없는 범위)
#include <stdio.h>
void name(int n)
{
    if(n < 1) return;
    name(n-1);
    printf("홀길통\n");
}

```

Below the code editor, the 'Call stack' window is open, displaying the following table:

이름	언어
4030.exe!name(int n) 줄 5	C++
4030.exe!name(int n) 줄 6 + 0xc 바이트	C++
4030.exe!name(int n) 줄 6 + 0xc 바이트	C++
4030.exe!name(int n) 줄 6 + 0xc 바이트	C++
4030.exe!main() 줄 12 + 0x7 바이트	C++
4030.exe!_tmainCRTStartup() 줄 555 + 0x19 바이트	C
4030.exe!mainCRTStartup() 줄 371	C
kernel32.dll!771bed6c()	

Visual Studio

## 자가진단 1

20 이하의 자연수 N을 입력받아 재귀함수를 이용해서 “recursive”를 N번 출력하는 프로그램을 작성하시오

입력 예

3

출력 예

recursive  
recursive  
recursive

보충

재귀함수는 같은 일을 여러번 처리해야하는 경우 함수 내에서 단 한 번만 처리하고 그 한 번을 제외한 나머지 일을 다시 같은 함수를 호출하여 처리하도록 하는 원리로 작성되는 것이다.

즉 위의 예제를 보면 num(10)이라는 함수를 호출하게 되면 이것은 1부터 10까지를 출력하라는 것인데 10을 호출받은 함수 내에서는 마지막 10만 출력하도록 하고 1부터 9까지는 같은 함수를 호출하여 처리하도록 하는 것이다.

결국 10부터 1까지 차례대로 호출이 되므로 각각 호출된 함수 내에서 호출된 값을 출력하게 되는 것이다. (0도 호출되지만 이 때 바로 리턴을 하여 호출을 종료하는 역할을 수행한다.)

따라서 재귀함수를 이용한 프로그래밍을 할 때에는 여러 가지 해야 할 작업 중 현재 호출된 함수 내에서 처리할 단 한 가지만 작성하고 그 작업을 제외한 나머지를 다시 호출하도록 작성하면 된다.

03

Question

12 이하의 자연수  $N$ 을 입력받아 재귀함수를 이용하여  $N!$ (팩토리얼)을 구하여 출력하는 프로그램을 작성하시오. ( $N$ 팩토리얼이란 1부터  $N$ 까지의 곱을 말한다.  $1 * 2 * 3 \dots * N$ )

입력 예      10

출력 예      3628800

소스

```
#include <stdio.h>

int fac(int n)
{
    if (n <= 1) return 1;
    return n * fac(n - 1);
}

int main()
{
    int N;
    scanf("%d", &N);
    printf("%d\n", fac(N));
    return 0;
}
```

## 설명

- `int fac(int n)`  
`n`팩토리얼( $n!$ )을 리턴하도록 정의된 함수이다.
- `if (n <= 1) return 1;`  
`n`이 1이면  $1!$ 은 1이므로 1을 리턴한다.
- `return n * fac(n - 1)`  
 $n! = n * (n-1)!$ 과 같다. 그러므로 `fac(n - 1)`을 호출하여  $(n - 1)!$ 이 구해지면 그 값에 `n`을 곱하여  $n!$ 을 리턴하게 된다. (예를 들어  $10! = 10 * 9!$ 이다.)
- 참고로 `N`이 5가 입력되었을 때 실행되는 과정을 식으로 나타내면 다음과 같다.  
$$\begin{aligned} 5! &= 5 * 4! \\ &= 5 * (4 * 3!) \\ &= 5 * (4 * (3 * 2!)) \\ &= 5 * (4 * (3 * (2 * 1!))) \\ &= 5 * 4 * 3 * 2 * 1 \end{aligned}$$

### 자가진단 3

100 이하의 자연수  $N$ 을 입력받아 재귀함수를 이용하여 1부터  $N$ 까지의 합을 구하는 프로그램을 작성하시오

입력 예

100

출력 예

5050



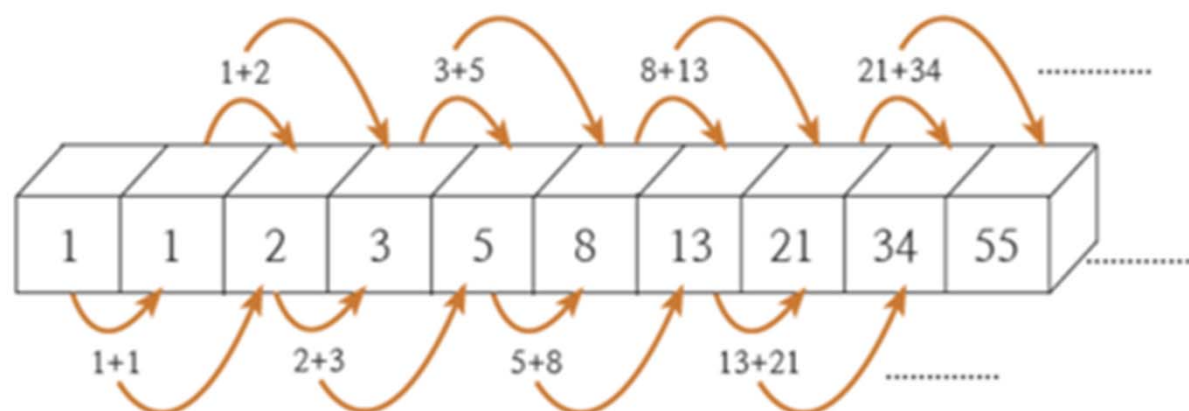
05

Question

50 이하의 자연수  $N$ 을 입력받아 재귀호출을 이용하여 피보나치 수열에서  $N$ 번째 수를 출력하는 프로그램을 작성하시오. 피보나치 수열이란 첫 번째와 두 번째 수는 1이고 세 번째 수부터는 바로 앞 두 수의 합으로 구성된 수열이다. (1 1 2 3 5 8 13 21 ...)

입력 예 7

출력 예 13



## 소스1

```
#include <stdio.h>

int fibo(int n)
{
    if (n < 3) return 1;
    return fibo(n - 1) + fibo(n - 2);
}

int main()
{
    int N;
    scanf("%d", &N);
    printf("%d\n", fibo(N));
    return 0;
}
```

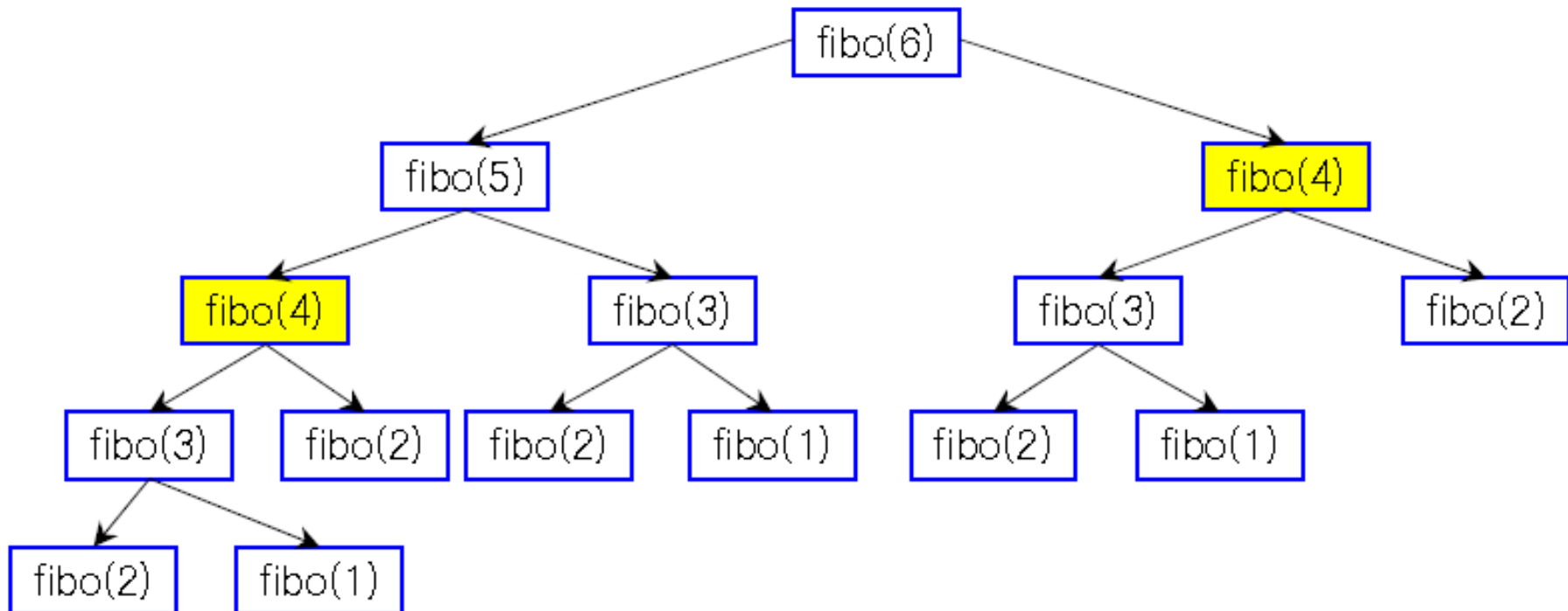
## 설명

```
• int fibo(int n)
{
    if (n < 3) return 1;
    return fibo(n - 1) + fibo(n - 2);
}
```

피보나치 수열의  $n$ 번째 값을 리턴하는 함수이다.  $n$ 이 3보다 작다면(즉, 1이나 2이면) 1을 리턴하고, 3 이상이면 피보나치 수열에서 앞의 두 수를 구하여(재귀함수를 호출하여) 그 합을 리턴한다.

$(\text{fibo}(n - 1) + \text{fibo}(n - 2))$

- 그런데 이 재귀함수는 매우 비효율적으로 작동한다. 만약 `fibonacci(6)`을 호출한다고 가정을 하면 다음과 같이 모두 15회의 호출이 발생된다.



오른쪽 `fibonacci(4)`의 경우 왼쪽에서 이미 `fibonacci(3)`과 `fibonacci(2)`를 호출하여 값을 구한 적이 있음에도 다시 재귀함수를 호출하여 같은 작업을 반복하게 된다.

**인수가 커질수록 이러한 불필요한 반복 작업이 기하급수적으로 늘어나게 되어 시간을 허비하게 된다.** 이러한 불필요한 작업을 줄이기 위해 값이 한 번 정해진 것은 재귀호출을 하지 않고 그 값을 바로 리턴해 주도록 다음과 같이 소스를 수정하면 훨씬 효율적인 프로그램이 된다.

## 소스2

```
#include <stdio.h>

int arr[100] = {0, 1, 1};

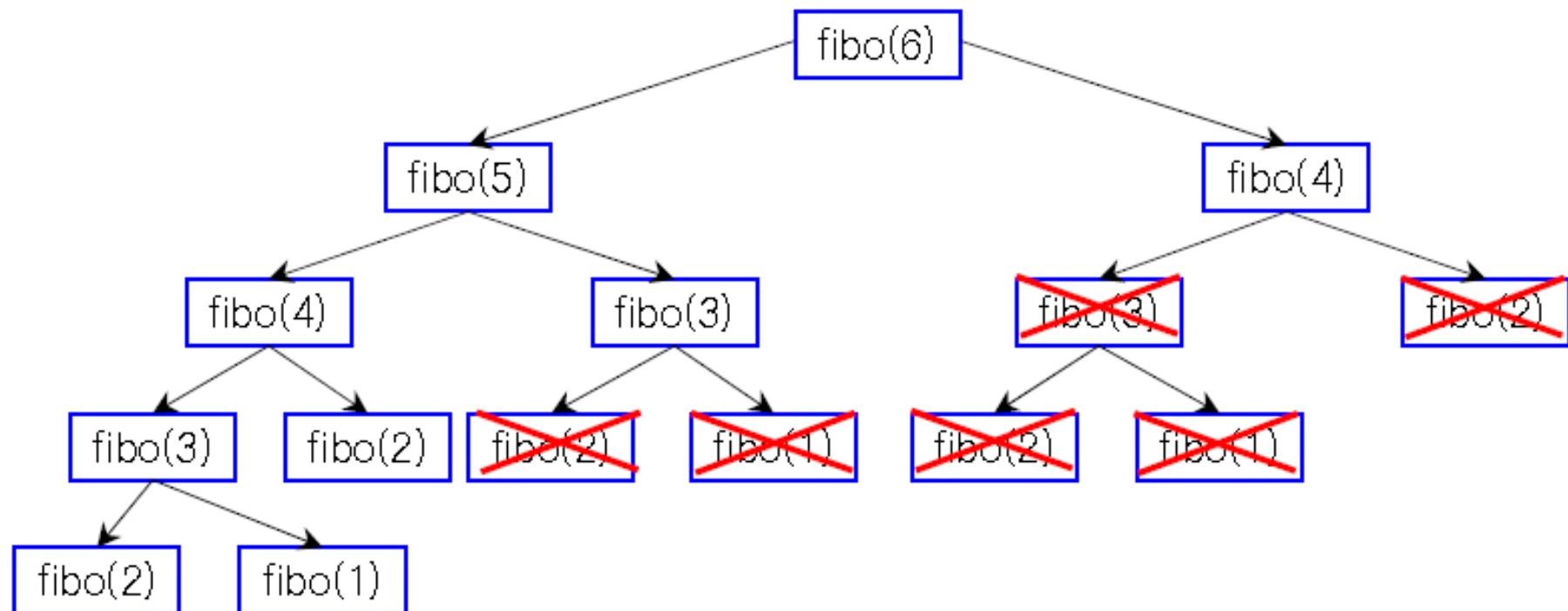
int fibo(int n)
{
    if (arr[n] == 0) arr[n] = fibo(n - 1) + fibo(n - 2);
    return arr[n];
}

int main()
{
    int N;
    scanf("%d", &N);
    printf("%d\n", fibo(N));
    return 0;
}
```

## 설명

- `int arr[100] = {0, 1, 1};`  
피보나치 수열을 저장하기 위한 배열을 선언하고 `arr[1]`과 `arr[2]`를 1로 초기화한다.
- `int fibo(int n)`  
{  
    if (`arr[n] == 0`) `arr[n] = fibo(n - 1) + fibo(n - 2);`  
    return `arr[n];`  
}

`n`번째 피보나치 수열이 아직 구해지지 않은 경우에만 재귀함수를 호출하여 값을 구한 후 `arr[n]`에 저장한다. 그리고 그 저장된 값을 리턴한다. 만약 이미 값이 구해져 있다면 재귀함수를 호출하는 작업을 생략하고 바로 그 값을 리턴하면 된다. `fibo(6)`을 호출할 때 아래와 같이 X표한 호출은 생략이 된다.





## 자가진단 5

첫 번째 수는 1이고  $N$ 번째 수는  $(N / 2)$ 번째 수와  $(N - 1)$ 번째 수의 합으로 구성된 수열이 있다. 50 이하의 자연수  $N$ 을 입력받아 재귀호출을 이용하여 이 수열에서  $N$ 번째 수를 출력하는 프로그램을 작성하시오.  
(1 2 3 5 7 10 13 18 ...)

입력 예

8

출력 예

18

- *hint*

8번째 수는 4번째( $8/2$ ) 수인 5와 7번째( $8-1$ ) 수인 13의 합이다.

# Thank You!!!

