

EC C언어 스터디

-5강-

변수의 종류와 범위



함수



재귀 함수





함수

함수 선언 반환형식 함수이름(인자, 인자...);

함수 정의 반환형식 함수이름(인자, 인자...)
 {
 함수 내용
 }

```
int sum(int, int);  
int sum(int a, int b);  
int sum(int a, int b)  
{  
    return a+b;  
}
```

```
#include<stdio.h>

int sum(int a, int b)
{
    return a+b;
}

int main()
{
    printf("%d", sum(1,3));
}
```

```
#include<stdio.h>

int main()
{
    printf("%d", sum(1,3));
}

int sum(int a, int b)
{
    return a+b;
}
```

```
#include<stdio.h>

int sum(int, int);

int main()
{
    printf("%d", sum(1,3));
}

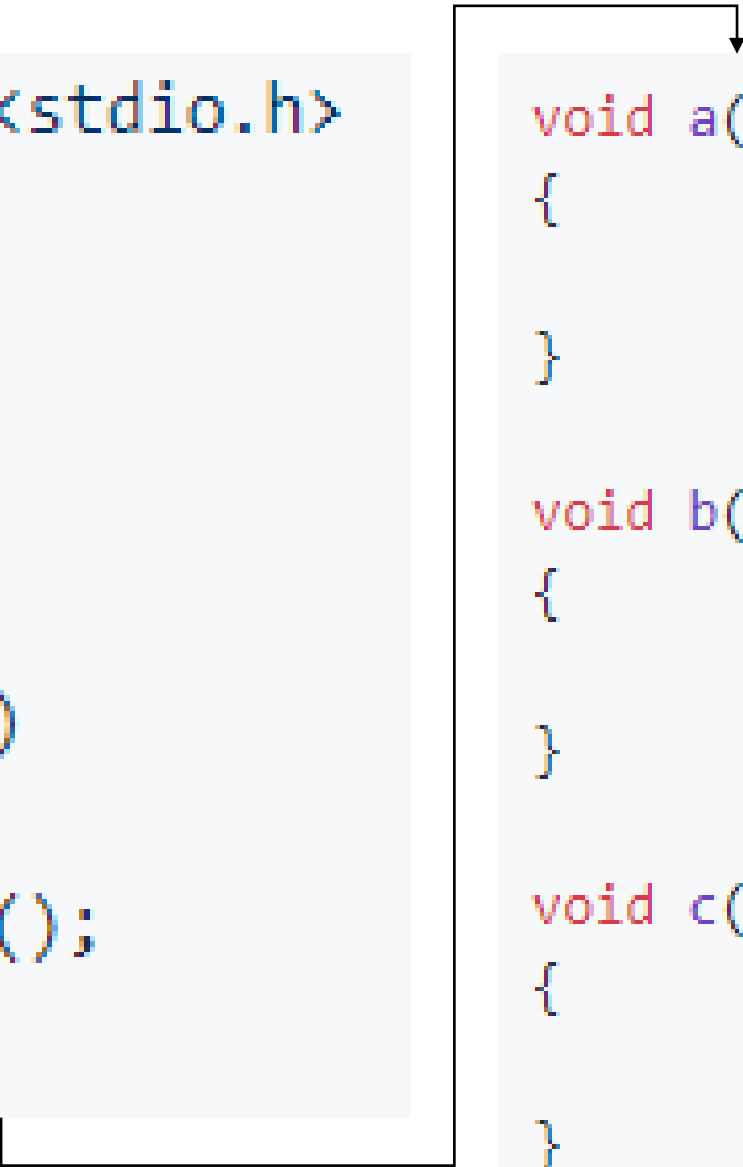
int sum(int a, int b)
{
    return a+b;
}
```



```
#include <stdio.h>

void a();
void b();
void c();

int main()
{
    a();
}
```



```
void a()
{
    b();
}

void b()
{
    c();
}

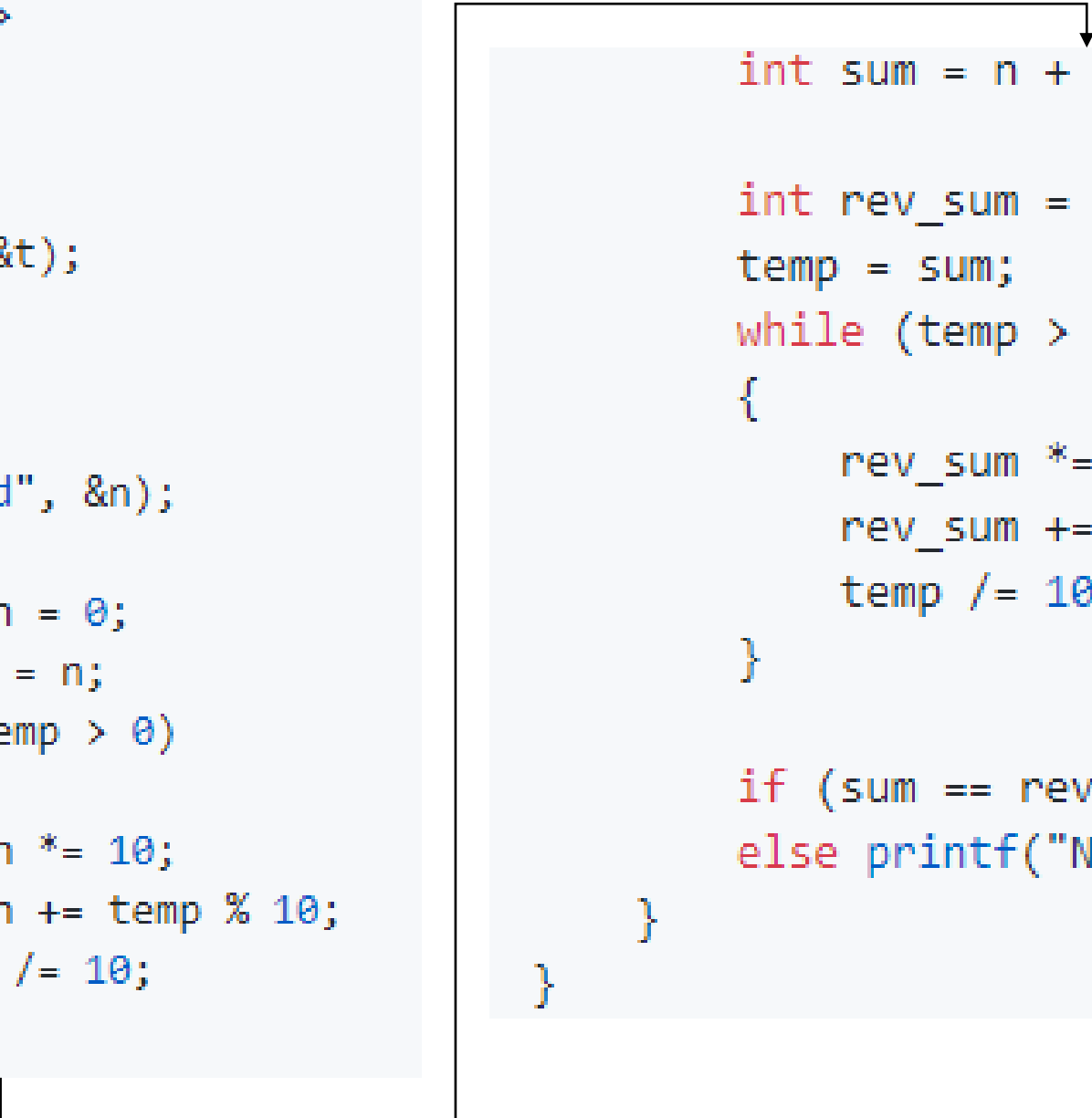
void c()
{
    a();
}
```

The diagram illustrates a recursive call. A line originates from the `a();` statement within the `main` function's block, extends horizontally to the right, and then turns vertically upwards. An arrowhead at the top of this vertical line points to the `void a()` function definition, indicating that `main` is calling `a`.

```
int sum(int a, int b);  
int getValue();  
void setValue(int a);  
void printLogo();
```

```
#include<stdio.h>
int main()
{
    int t;
    scanf("%d", &t);
    while (t--)
    {
        int n;
        scanf("%d", &n);

        int rev_n = 0;
        int temp = n;
        while (temp > 0)
        {
            rev_n *= 10;
            rev_n += temp % 10;
            temp /= 10;
        }
    }
}
```



```
int sum = n + rev_n;

int rev_sum = 0;
temp = sum;
while (temp > 0)
{
    rev_sum *= 10;
    rev_sum += temp % 10;
    temp /= 10;
}

if (sum == rev_sum) printf("YES\n");
else printf("NO\n");
}
```

```
#include <stdio.h>
int rev(int n)
{
    int res = 0;
    while (n > 0)
    {
        res = res * 10 + n % 10;
        n /= 10;
    }
    return res;
}

int main()
{
    int t;
    scanf("%d", &t);
    while (t--)
    {
        int n;
        scanf("%d", &n);
        int sum = n + rev(n);
        printf("%s\n", sum == rev(sum) ? "YES" : "NO");
    }
}
```

```
int sum(int a, int b)
{
    printf("%d + %d = %d\n", a, b, a+b);
    return a+b;
}
```

```
int sum(int a ,int b)
{
    return a+b;
}
```

```
int main()
{
    printf("%d + %d = %d\n", a, b, sum(a, b));
}
```

풀어보세요!

- 뒤집힌 덧셈: rev()함수 만들어 코드 재활용하기
 - 분해 합: 프로그램의 동작 분리시키기



변수의 종류와 범위


```
#include <stdio.h>
int sum(int a, int b)
{
    return a+b;
}

int main()
{
    for(int i=1;i<5;i++)
    {
        printf("%d + %d = %d", i, i+1, sum(i, i+1));
    }
}
```

- 중괄호 내부에서 선언된 변수
- 함수의 매개변수

```
#include <stdio.h>

int main()
{
    {
        int a = 1;
    }
    printf("%d", a); //에러
}
```

```
#include <stdio.h>

int main()
{
    //int sum = 0;

    for (int i = 0; i < 5; i++)
    {
        int sum = 0;
        sum += i;
    }

    printf("%d", sum);
}
```

```
#include <stdio.h>

void f(int a, int b)
{
    a = 3, b = 4;
}

int main()
{
    int a = 1, b = 2;
    f(a, b);
    printf("a=%d, b=%d", a, b); //a=1 b=2
}
```

<f>

a = 3

b = 4

<main>

a = 1

b = 2

<메모리>

```
#include <stdio.h>

int ans;

void solve(int n)
{
    while (n > 0)
    {
        ans += n % 10;
        n /= 10;
    }
}

int main()
{
    solve(12345);
    printf("답 = %d", ans); //15
}
```

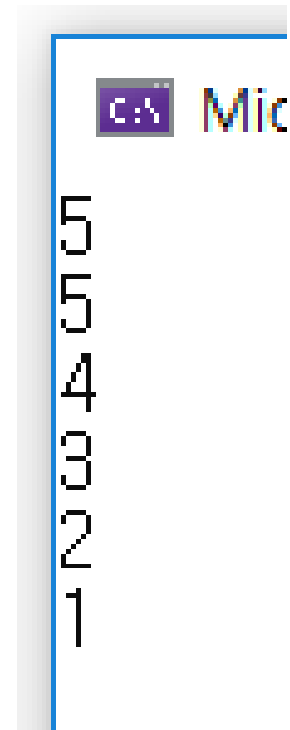


재귀 함수

```
#include <stdio.h>

void f(int n)
{
    if (n == 0) return;
    printf("%d\n", n);
    f(n - 1);
}

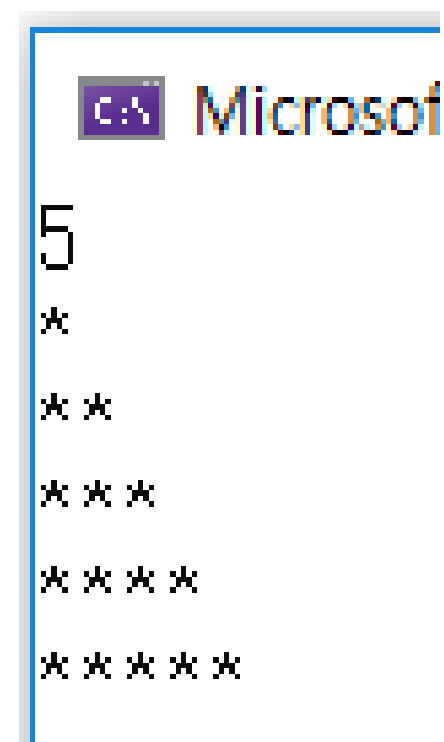
int main()
{
    int n;
    scanf("%d", &n);
    f(n);
}
```



```
#include <stdio.h>

void star(int n, int cnt)
{
    if (n < cnt) return;
    for (int i = 0; i < cnt; i++)
        printf("*");
    printf("\n");
    star(n, cnt + 1);
}

int main()
{
    int n;
    scanf("%d", &n);
    star(n, 1);
}
```



The screenshot shows a Windows command prompt window with the title bar "C:\> Microsoft". The prompt is at "C:\>". The user has entered "5" and pressed Enter. The output of the program is a recursive star pattern consisting of 5 lines. Each line contains a number of asterisks equal to the line number, starting from 1 and increasing by 1 up to 5.

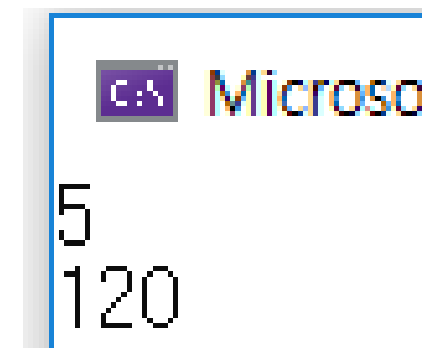
```
C:\> 5
*
* *
* * *
* * * *
* * * * *
```


- $n! = n \times (n-1)!$
- $0! = 1$

```
#include <stdio.h>

int fac(int n)
{
    if (n == 0) return 1;
    return n * fac(n - 1);
}

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

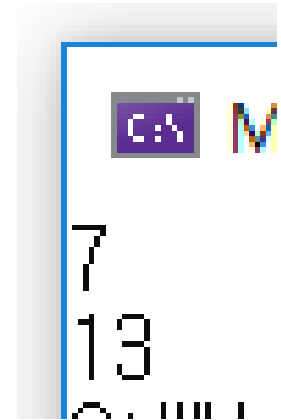


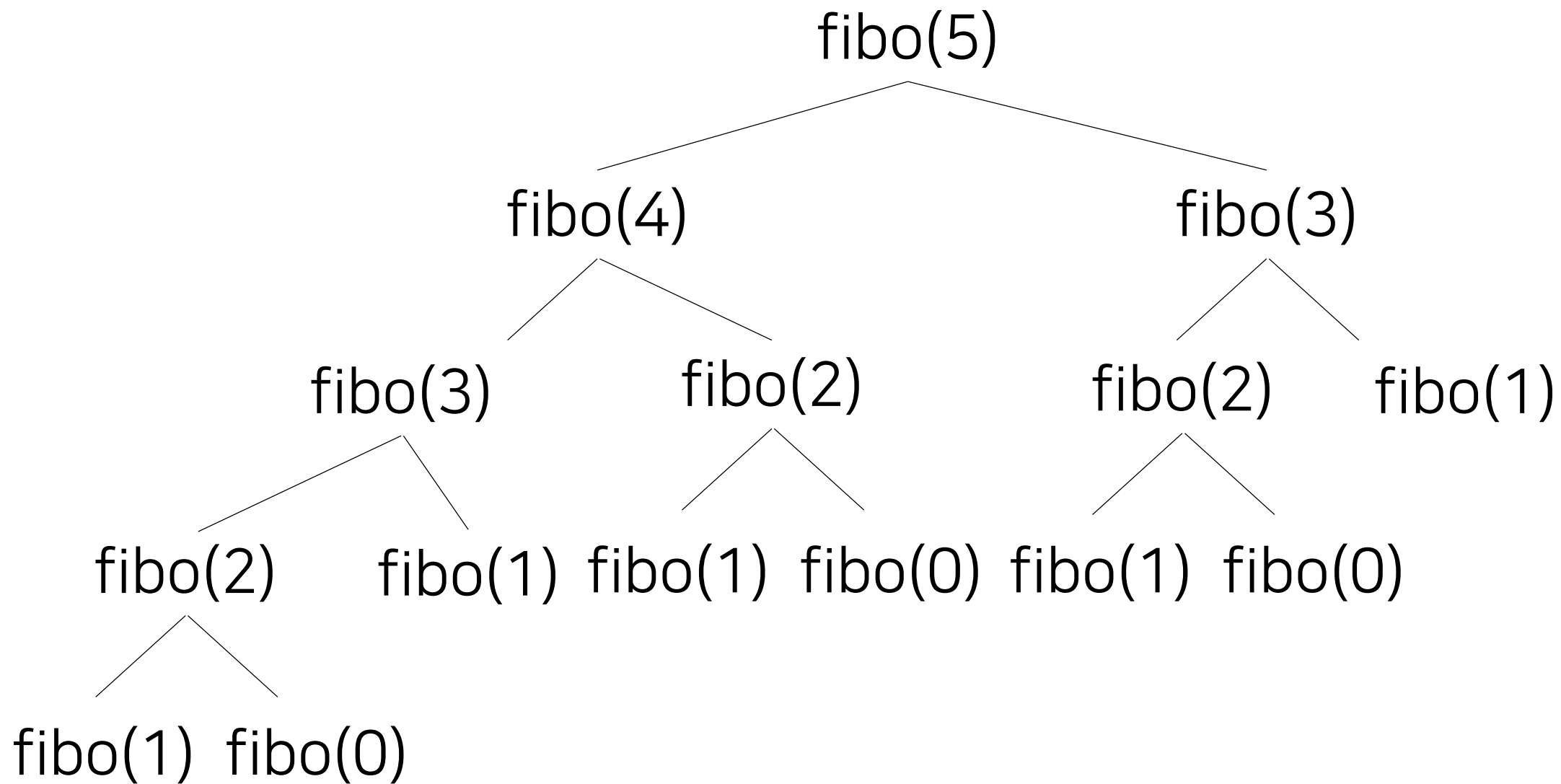
- $f(n) = f(n-1) + f(n-2)$
- $f(0) = 0$
- $f(1) = 1$

```
#include <stdio.h>

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

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





- $nCr = n-1Cr-1 + n-1Cr$
- $nCn = 1$
- $nC0 = 1$

```
#include <stdio.h>

long long int C(int n, int r)
{
    if (n == r || r == 0) return 1;
    return C(n - 1, r - 1) + C(n - 1, r);
}

int main()
{
    int n, r;
    scanf("%d%d", &n, &r);
    printf("%lld", C(n, r));
}
```



```
#include <stdio.h>

void pick(int size, int arr[], int pos, int picked[], int pickedNum, int toPick)
{
    if (pickedNum == toPick)
    {
        for (int i = 0; i < toPick; i++)
            printf("%d ", picked[i]);
        printf("\n");
        return;
    }
    for (int i = pos; i < size; i++)
    {
        picked[pickedNum] = arr[i];
        pick(size, arr, i + 1, picked, pickedNum + 1, toPick);
    }
}

int main()
{
    int arr[8] = { 1,2,3,4,5,6,7,8};
    int picked[3];
    pick(sizeof(arr) / sizeof(int), arr, 0, picked, 0, 3);
}
```

c:\ Mic

```
1 2 3
1 2 4
1 2 5
1 2 6
1 2 7
1 2 8
1 3 4
1 3 5
1 3 6
1 3 7
1 3 8
1 4 5
1 4 6
1 4 7
1 4 8
1 5 6
1 5 7
1 5 8
1 6 7
1 6 8
1 7 8
2 3 4
2 3 5
2 3 6
2 3 7
2 3 8
2 4 5
2 4 6
2 4 7
2 4 8
```

```
#include <stdio.h>

int fac(int n)
{
    if (n == 0) return 1;
    return n * fac(n - 1);
}

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

$O(n)$

```
#include <stdio.h>

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

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

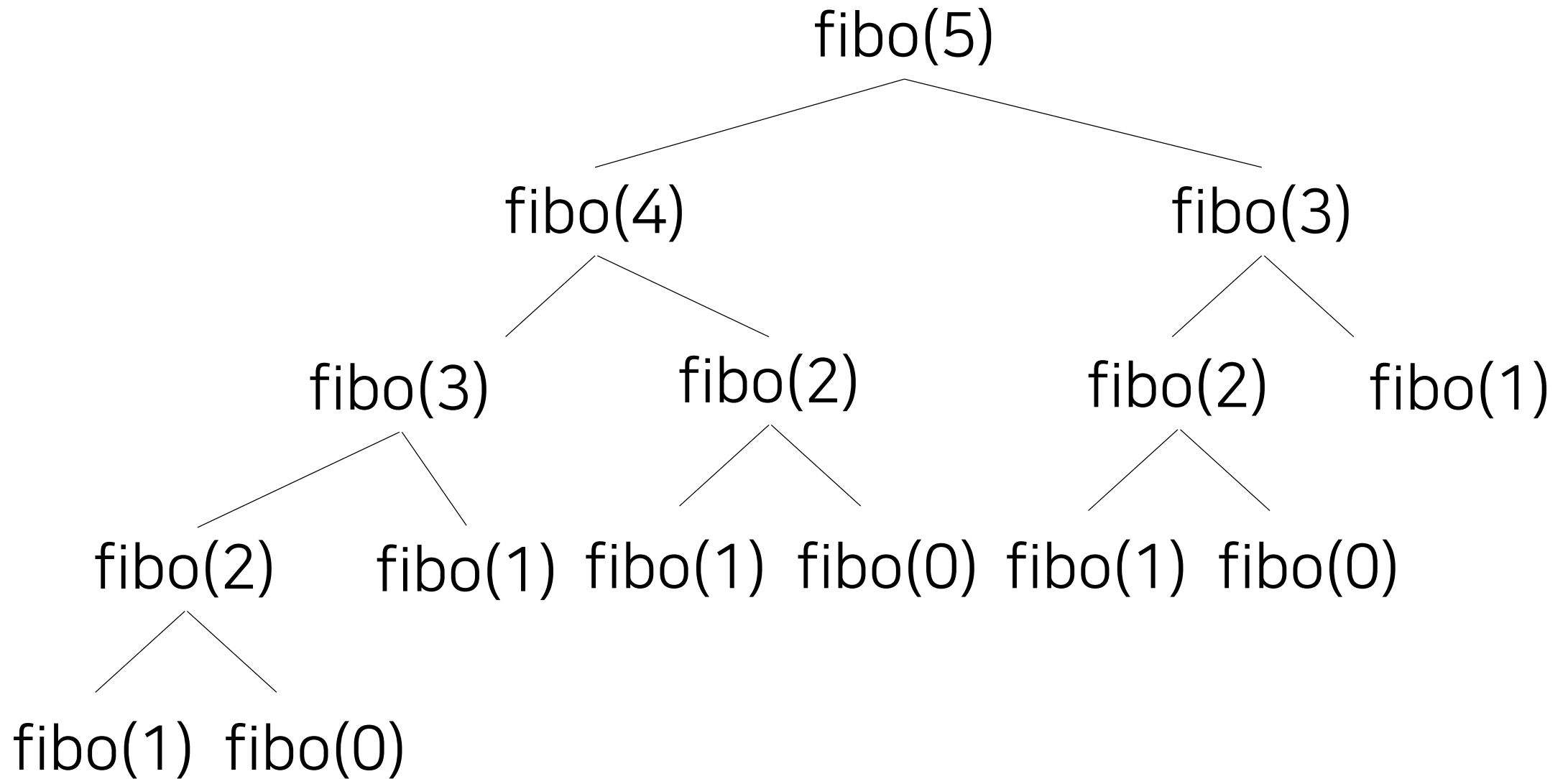
$O(2^n)$

```
#include <stdio.h>

int main()
{
    int arr[100] = { 0,1 };
    int n;
    scanf("%d", &n);

    for (int i = 2; i <= n; i++) arr[i] = arr[i - 1] + arr[i - 2];
    printf("%d", arr[n]);
}
```

$O(n)$



풀어보세요!

- 팩토리얼 : 재귀함수로 풀어보세요
- 피보나치 수 5 : 재귀함수로 풀어보세요
- 이항 계수 1 : 재귀함수로 풀어보세요

```
#include <stdio.h>

int gcd(int a, int b)
{
    if (b == 0) return a;
    return gcd(b, a % b);
}

int main()
{
    int a, b;
    scanf("%d%d", &a, &b);
    printf("%d", gcd(a, b));
}
```

풀어보세요!

- GCD 합: 유클리드 호제법을 이용해 간단하고 빠르게 최대공약수 구하기 + 배열에서 고르기 알고리즘



수고하셨습니다!