



MODULE 7

C Recursion

Assc. Prof. Dr. Natasha Dejdumrong

- ★ Recursion = Recursive Function
- ★ Factorial
- ★ Fibonacci

Recursion

- A **recursive function** is a function that **calls itself** either directly or indirectly.

```
1 // Recursion
2 #include <stdio.h>
3
4 int recursion(int n) {
5     if (n == 1) return 5;
6     else return 5 * recursion( n - 1 );
7 }
8
9 int main()
10 {
11     int i, n = 5;
12     i = recursion(n)
13     return 0;
14 }
15
```

Function Declaration

Base Case

Recursive Case

Function Call

Factorial

- The factorial of a nonnegative integer n , written $n!$ (pronounced “ n factorial”), is the product

$$n \cdot (n - 1) \cdot (n - 2) \cdot \dots \cdot 1$$

- with $1!$ equal to 1, and $0!$ defined to be 1.
- Example, $5!$ is the product $5 * 4 * 3 * 2 * 1$, which is equal to 120.
- The factorial of an integer, **number**, greater than or equal to 0 can be calculated iteratively (nonrecursively) using a for statement as follows:

```
factorial = 1;
for ( counter = number; counter >= 1; --counter )
    factorial *= counter;
```

Factorial

- ★ A **recursive definition** of the **factorial function** is arrived at by observing the following relationship:

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

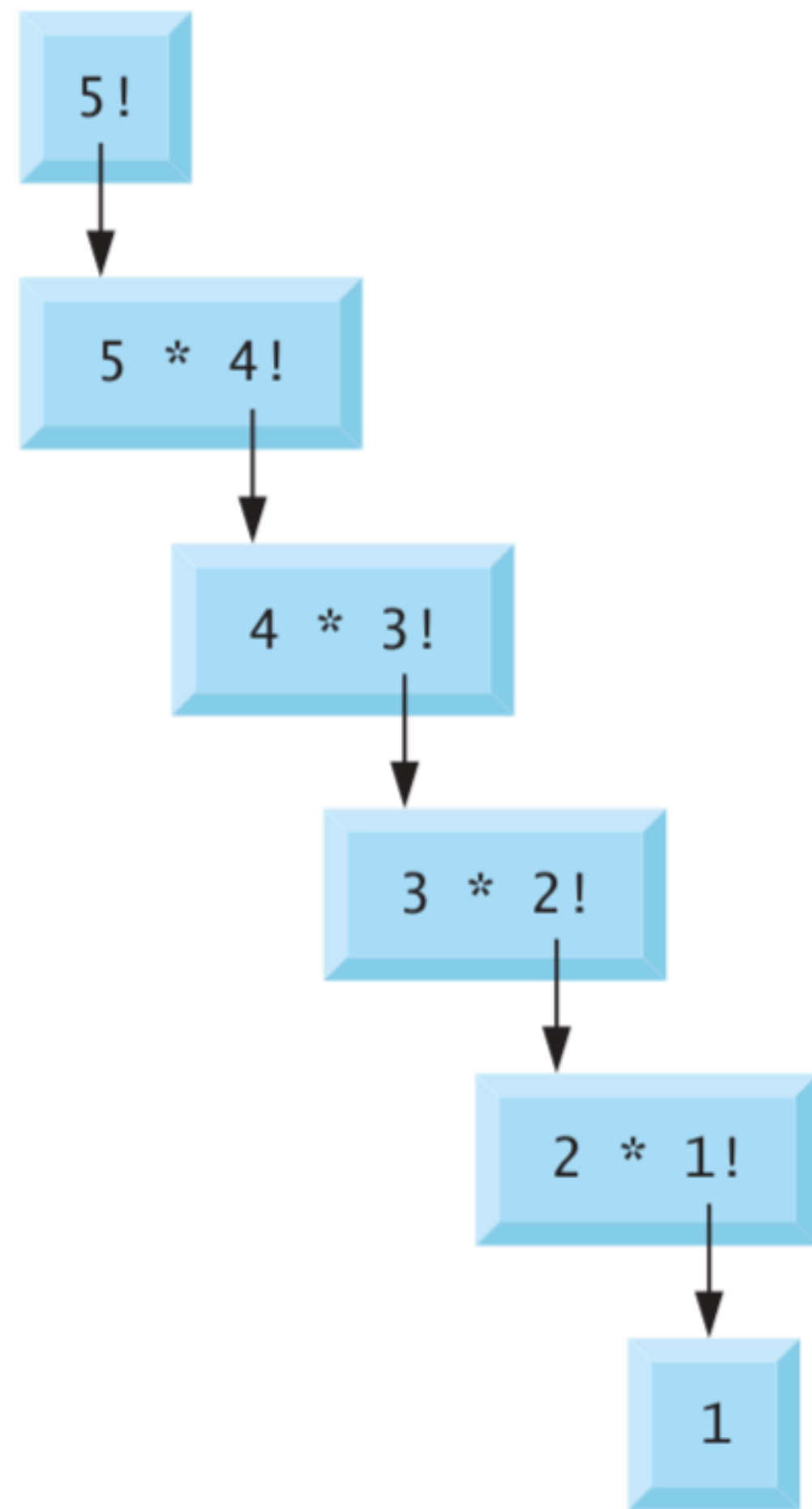
- ★ For example, 5! is clearly equal to 5 * 4! as is shown by the following:

$$5 ! = 5 \times 4 \times 3 \times 2 \times 1$$

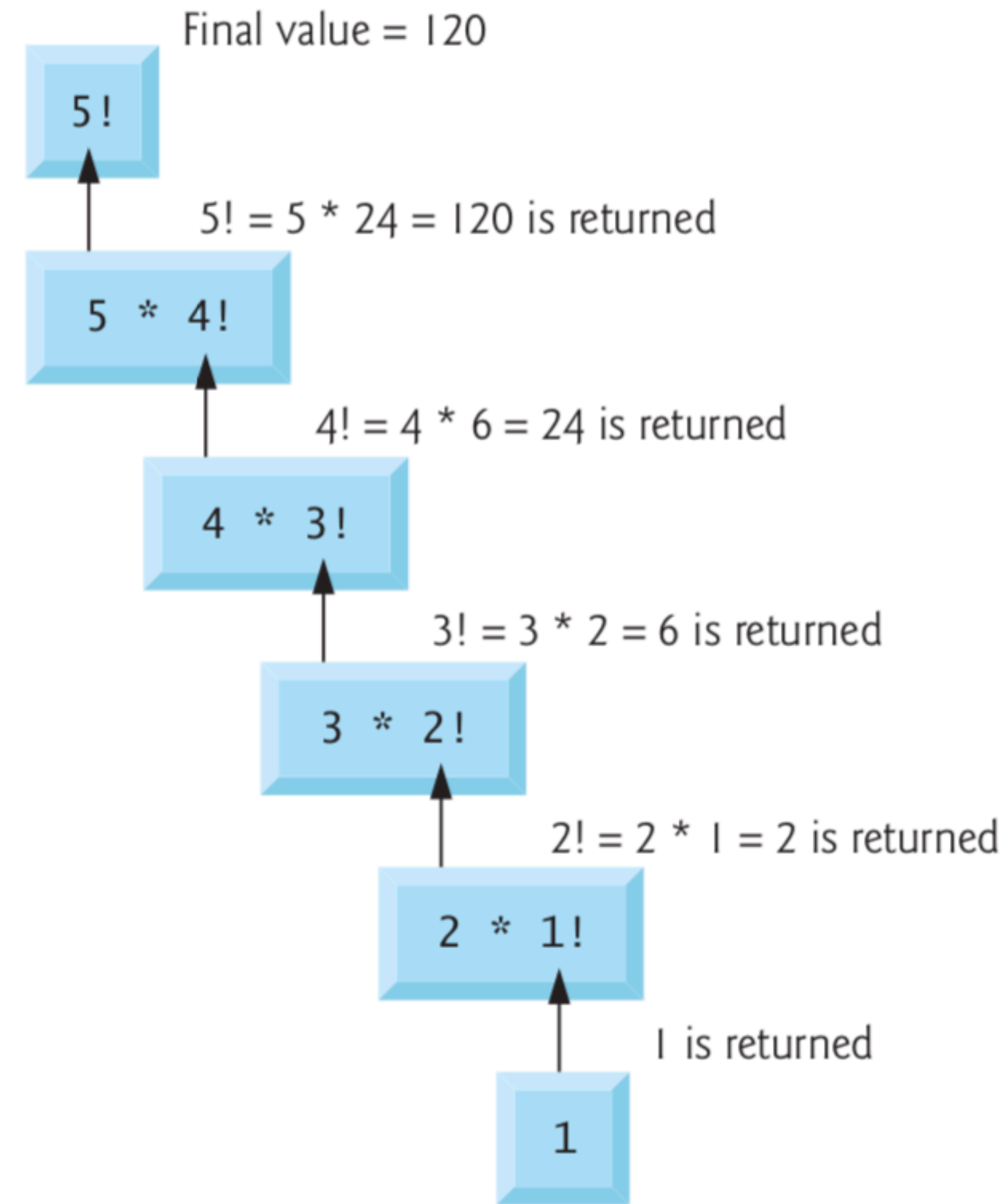
$$5 ! = 5 \times (4 \times 3 \times 2 \times 1)$$

$$5 ! = 5 \times 4 !$$

Factorial



(a) Sequence of recursive calls



(b) Values returned from each recursive call

Factorial function

```
1 // Fig. 5.18: fig05_18.c
2 // Recursive factorial function.
3 #include <stdio.h>
4
5 unsigned long long int factorial( unsigned int number );
6
7 // function main begins program execution
8 int main( void )
9 {
10     unsigned int i; // counter
11
12     // during each iteration, calculate
13     // factorial( i ) and display result
14     for ( i = 0; i <= 21; ++i ) {
15         printf( "%u! = %llu\n", i, factorial( i ) );
16     } // end for
17 } // end main
18
19 // recursive definition of function factorial
20 unsigned long long int factorial( unsigned int number )
21 {
22     // base case
23     if ( number <= 1 ) {
24         return 1;
25     } // end if
26     else { // recursive step
27         return ( number * factorial( number - 1 ) );
28     } // end else
29 } // end function factorial
```

← Function Prototype

← Function Call

← Function Declaration

← Base Case

← Recursive Case

Factorial function

```
1 // Fig. 5.18: fig05_18.c
2 // Recursive factorial function.
3 #include <stdio.h>
4
5 unsigned long long int factorial( unsigned int number );
6
7 // function main begins program execution
8 int main( void )
9 {
10     unsigned int i; // counter
11
12     // during each iteration, calculate
13     // factorial( i ) and display result
14     for ( i = 0; i <= 21; ++i ) {
15         printf( "%u! = %llu\n", i, factorial( i ) );
16     } // end for
17 } // end main
18
19 // recursive definition of function factorial
20 unsigned long long int factorial( unsigned int number )
21 {
22     // base case
23     if ( number <= 1 ) {
24         return 1;
25     } // end if
26     else { // recursive step
27         return ( number * factorial( number - 1 ) );
28     } // end else
29 }
```

```
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600
13! = 6227020800
14! = 87178291200
15! = 1307674368000
16! = 20922789888000
17! = 355687428096000
18! = 6402373705728000
19! = 121645100408832000
20! = 2432902008176640000
21! = 14197454024290336768
```


Fibonacci

- Fibonacci (1170 – c. 1240–50), also known as Leonardo Bonacci, Leonardo of Pisa, or Leonardo Bigollo Pisano ('Leonardo the Traveller from Pisa'[7]), was an Italian mathematician from the Republic of Pisa, considered to be "the most talented Western mathematician of the Middle Ages".

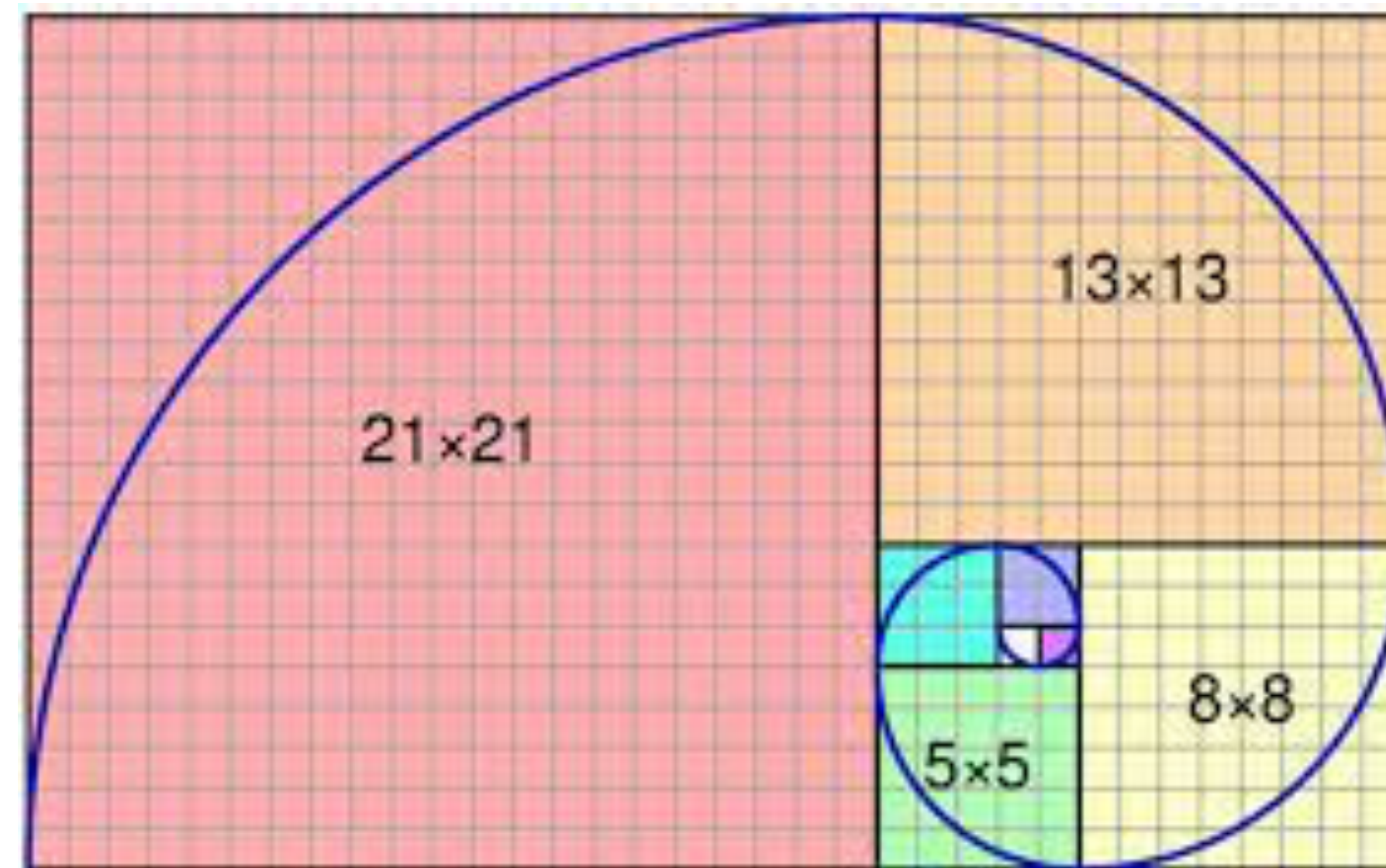


Fibonacci

- The Fibonacci series

0, 1, 1, 2, 3, 5, 8, 13, 21, ...

- begins with 0 and 1 and has the property that each subsequent Fibonacci number is the sum of the previous two Fibonacci numbers.
- Golden Ratio



Fibonacci

- ★ $\text{fibonacci}(0) = 0$
 - ★ $\text{fibonacci}(1) = 1$
 - ★ $\text{fibonacci}(n) = \text{fibonacci}(n - 1) + \text{fibonacci}(n - 2)$
- ✧ calculates the n th Fibonacci number recursively using function `fibonacci`.
Notice that Fibonacci numbers tend to become large quickly.

Fibonacci

```
1 // Fig. 5.19: fig05_19.c
2 // Recursive fibonacci function
3 #include <stdio.h>
4
5 unsigned long long int fibonacci( unsigned int n ); // function prototype
6
7 // function main begins program execution
8 int main( void )
9 {
10     unsigned long long int result; // fibonacci value
11     unsigned int number; // number input by user
12
13     // obtain integer from user
14     printf( "%s", "Enter an integer: " );
15     scanf( "%u", &number );
16
17     // calculate fibonacci value for number input by user
18     result = fibonacci( number );
19
20     // display result
21     printf( "Fibonacci( %u ) = %llu\n", number, result );
22 } // end main
23
24 // Recursive definition of function fibonacci
25 unsigned long long int fibonacci( unsigned int n )
26 {
27     // base case
28     if ( 0 == n || 1 == n ) {
29         return n;
30     } // end if
31     else { // recursive step
32         return fibonacci( n - 1 ) + fibonacci( n - 2 );
33     } // end else
34 } // end function fibonacci
```

Function Prototype

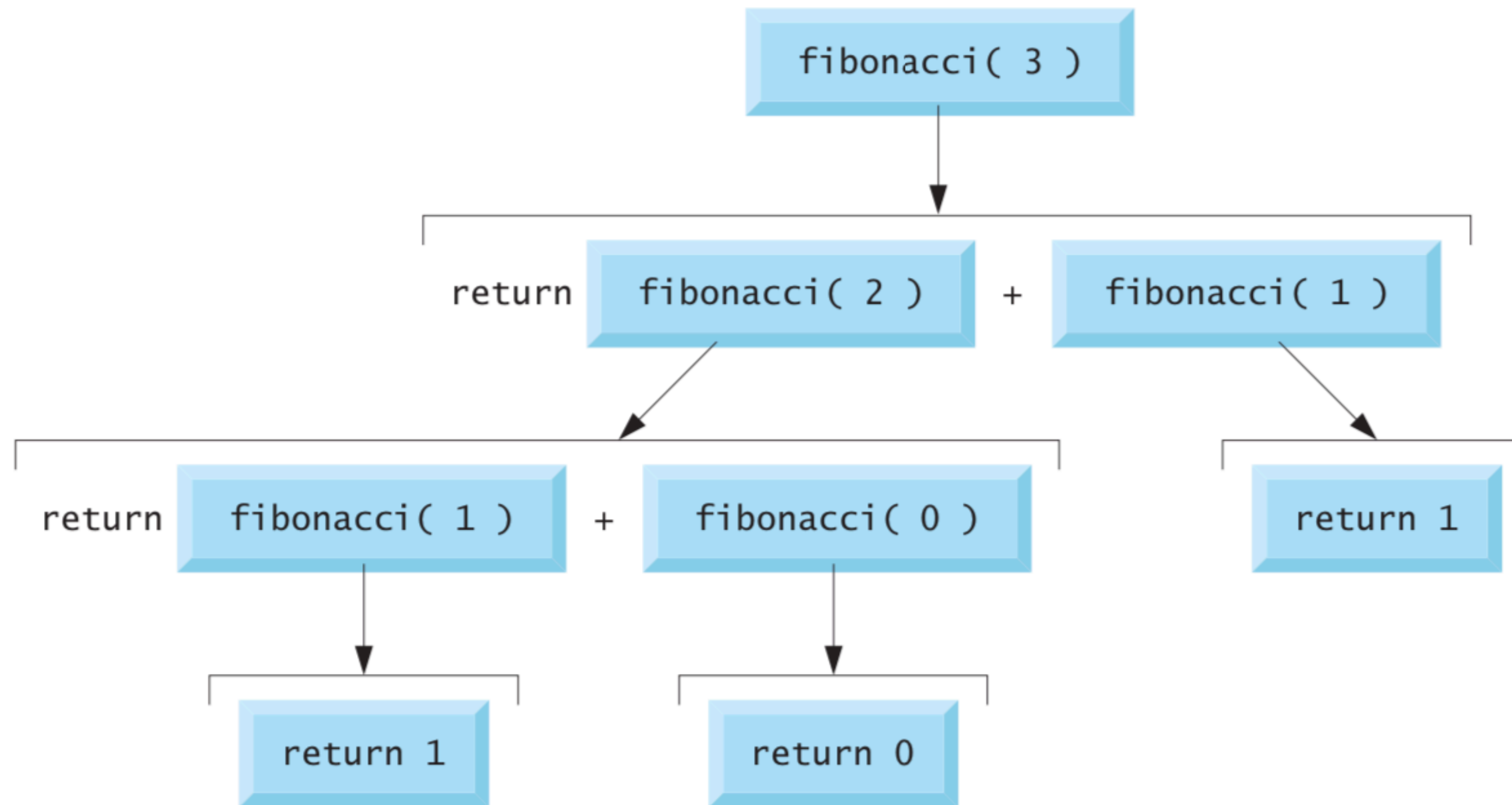
Function Call

Function Declaration

Base Case

Recursive Case

Fibonacci



Question and Answer