Object Oriented Programming

Shaobai Kan

chapter 5

Outline

Outline

- Recursion

Outline

- Recursion
- Fibonacci series

Outline

- Recursion
- Fibonacci series
- Exercise

Recursion

Recursive function

Recursive function. A recursive function is a function that call itself, either directly, or indirectly (through another function).

Fact. C++ standard document indicate that main function should not be called within a program or recursively.

Concept of recursion

Recursive problem-solving approaches:

1. A recursive function is called to solve a problem.

2. The called function divides the complex problem into two conceptual pieces:
   - base case(s) that function knows how to do;
   - the other piece that function does not know how to do.

Concept of recursion

3. The latter piece must resemble the original problem but be a slightly simpler or slightly smaller version.

4. The function launches a fresh copy of itself to work on the smaller problem until the recursive call converges on the base case.

Example: recursive function

Example. Write a program that computes "5 factorial" and prints the result.

Example: recursive function

Example. Write a program that computes "5 factorial" and prints the result.

Recall.

$$\begin{aligned} 5! &= 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \\ &= 120 \end{aligned}$$

Iteration: application

using **iteration**

```
# include <iostream>
using namespace std;


int main( )
{
      int factorial = 1;

      for (int i = 5; i >= 1; i --)
                  factorial *= i ;

      cout << "5! = " << factorial << endl ;

      return 0;
 }
```

Recursion: application

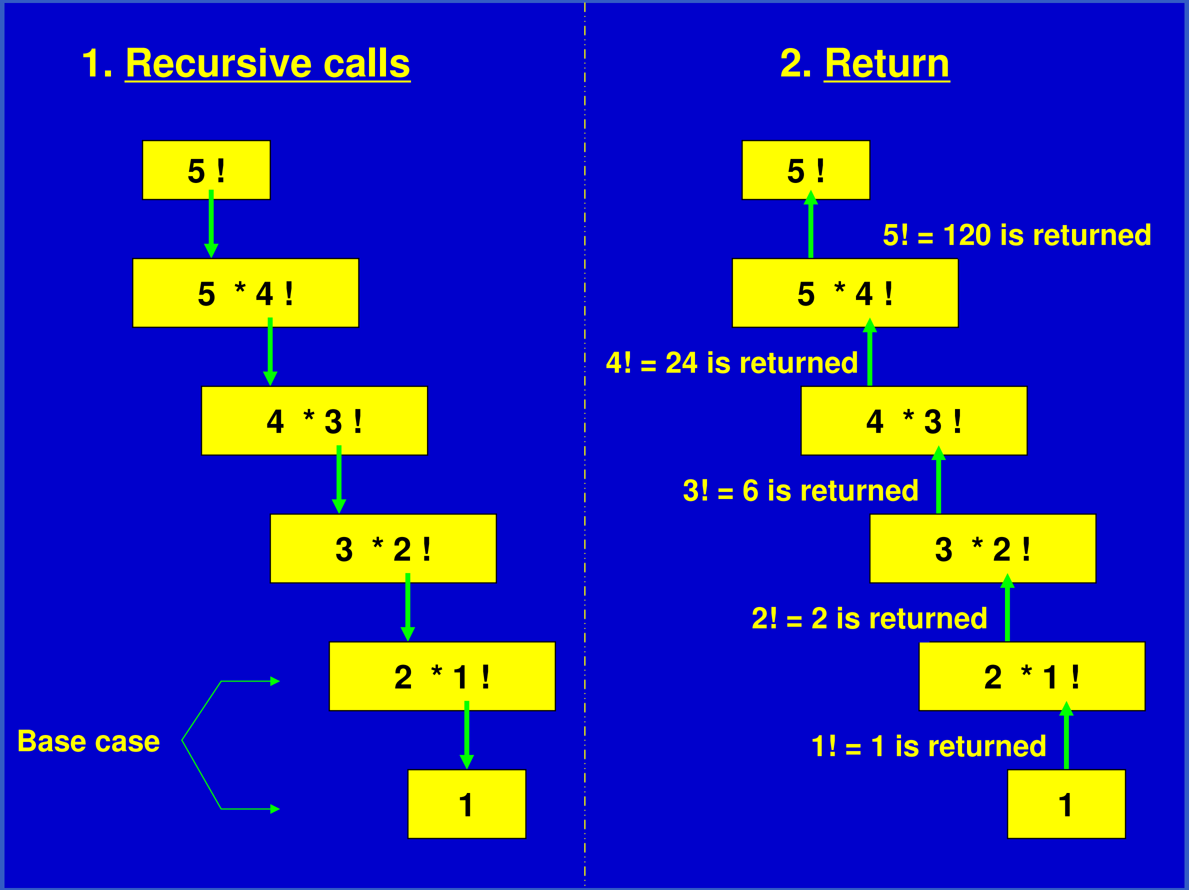
Recursive problem-solving approach:

1. Base cases: $0! = 1$ and $1! = 1$.

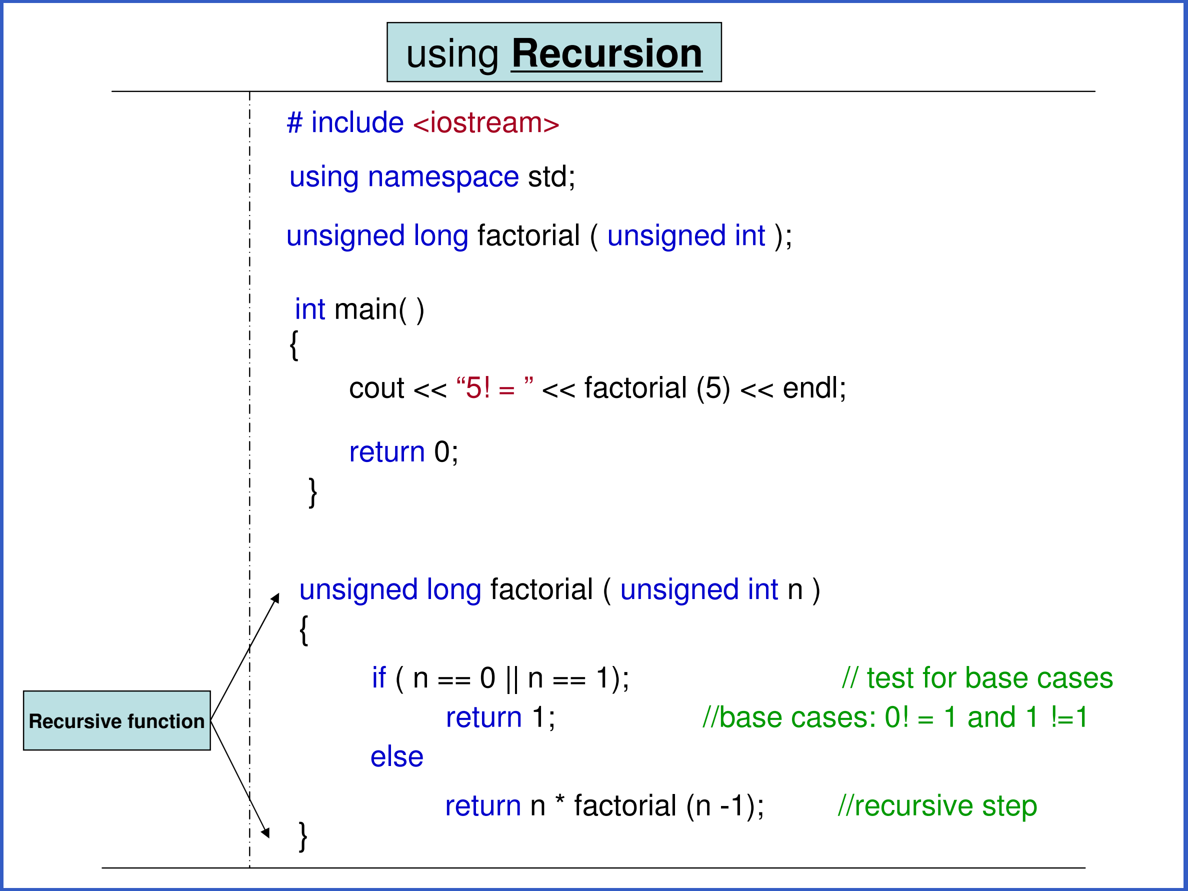
2. Fact. for any fixed integer $n > 1$,

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

Recursion: analysis

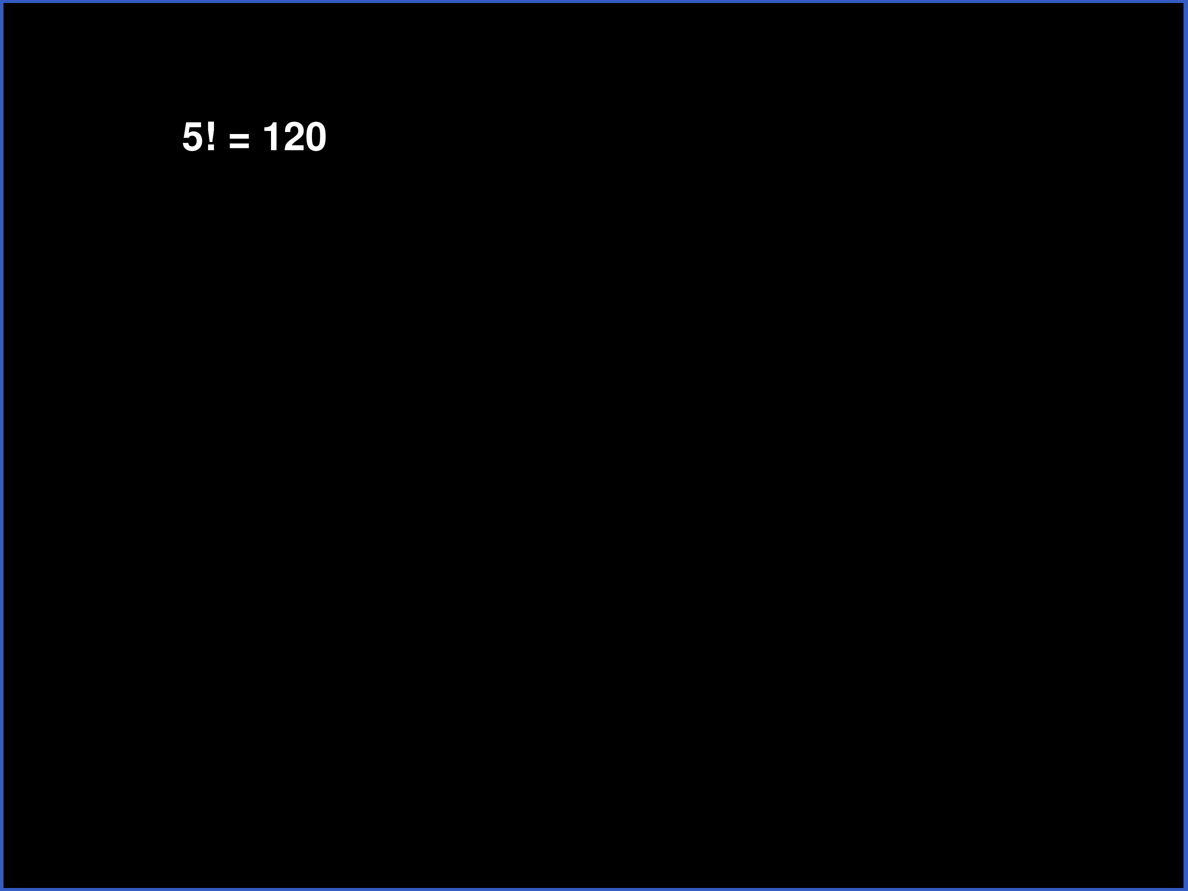


Recursion: Source code


using Recursion
Recursive function

```
# include <iostream>
using namespace std;
unsigned long factorial ( unsigned int );

int main( )
{
    cout << "5! = " << factorial (5) << endl;

    return 0;
}

unsigned long factorial ( unsigned int n )
{
    if ( n == 0 || n == 1);                 // test for base cases
        return 1;                //base cases: 0! = 1 and 1 !=1
    else
        return n * factorial (n -1);        //recursive step
}
```

Output



```
5! = 120
```

Variable types: integer

Variable type for integers

Type	bytes	Range
int	4	-2147483648 ∼ 2147483647
long (int)	4	-2147483648 ∼ 2147483647
short (int)	2	-32768 ∼ 32767
unsigned (int)	4	0 ∼ 4294967295
unsigned long (int)	4	0 ∼ 4294967295
unsigned short (int)	2	0 ∼ 65535

Fibonacci series

Fibonacci series

Definition. 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.

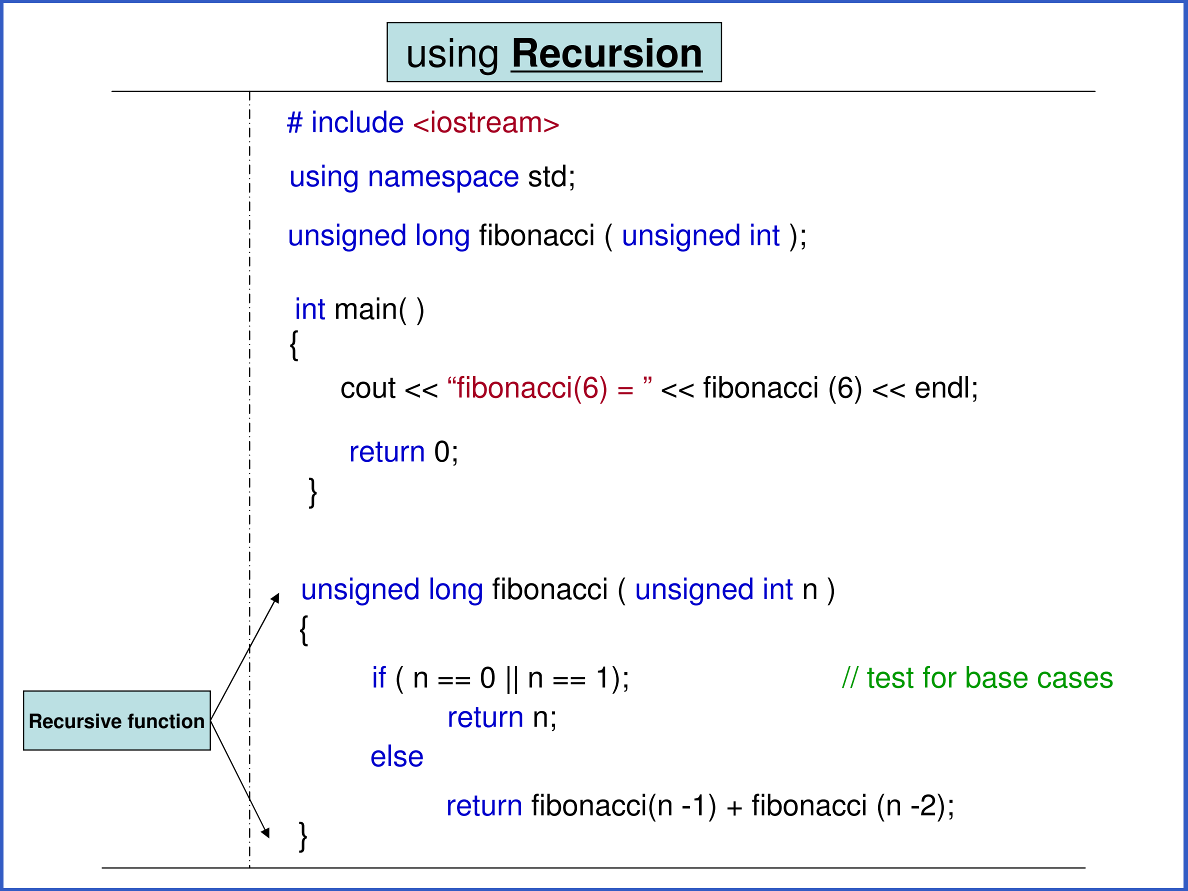
Fibonacci series

Fact. The Fibonacci series can be defined recursively as follows:

1. Base cases: $fibonacci(0) = 0$ and $fibonacci(1) = 1$.
2. Fact. for any fixed integer $n > 1$,

$$fibonacci(n) = fibonacci(n-1) + fibonacci(n-2).$$

Recursion: Source code

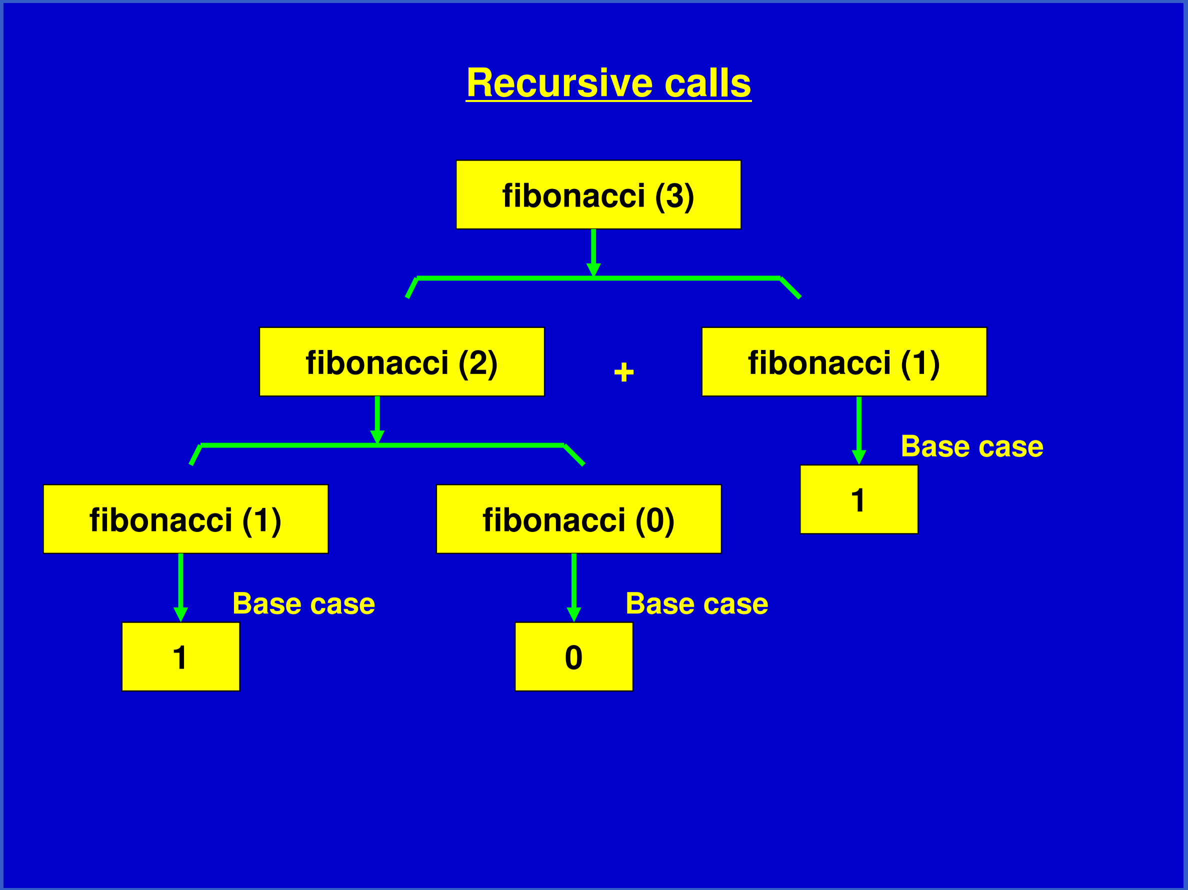


using **Recursion**

```
# include <iostream>
using namespace std;
unsigned long fibonacci ( unsigned int );

int main( )
{
    cout << "fibonacci(6) = " << fibonacci (6) << endl;

    return 0;
}

unsigned long fibonacci ( unsigned int n )
{
    if ( n == 0 || n == 1);                    // test for base cases
        return n;
    else
        return fibonacci(n -1) + fibonacci (n -2);
}
```

Recursive function

Variable types: integer



Exercise

Exercise

Exercise. Write a program that reads a nonnegative integer and computes and prints its factorial (using recursion).

Hint: Modify previous example!!!

Homework:

- Read Sec. 5.19, 5.20, 5.21, 5.22.
- Exercise 5.36, 5 .37.