

栈与队列

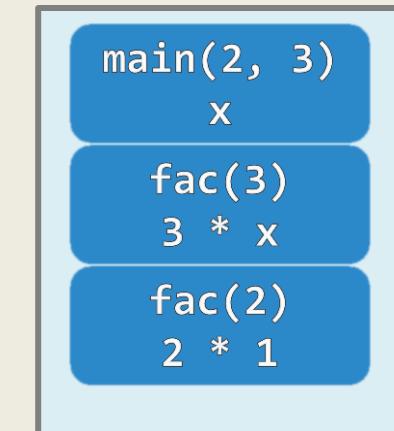
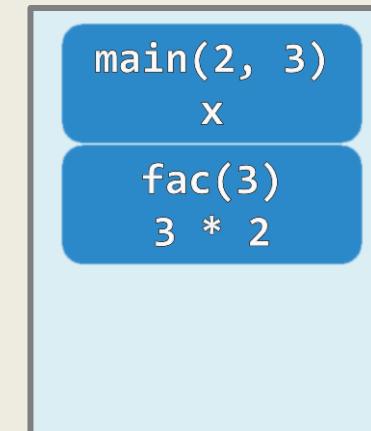
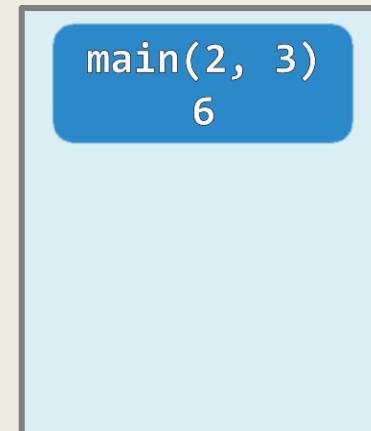
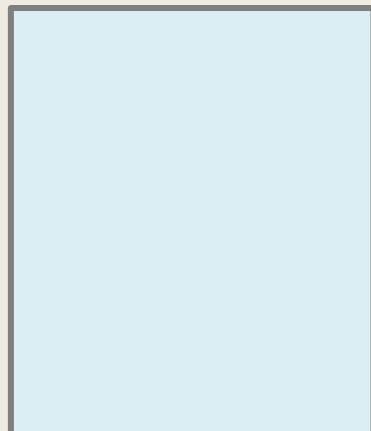
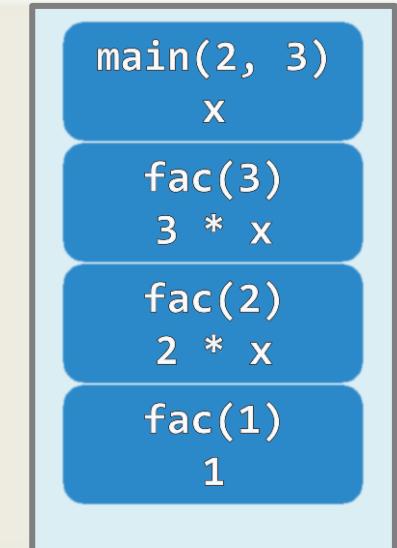
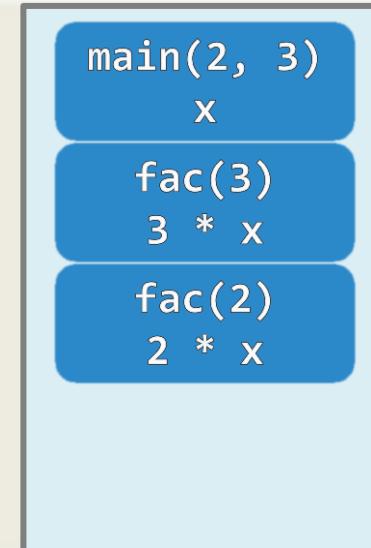
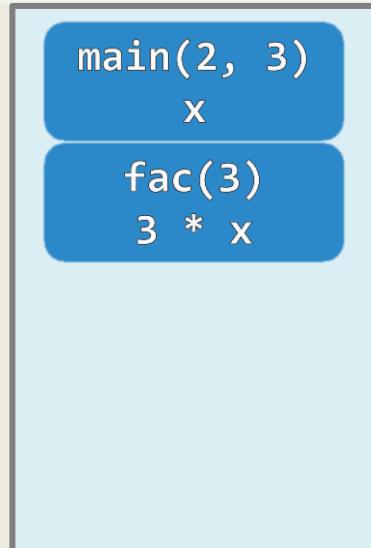
调用栈：实例

04-B2

邓俊辉

deng@tsinghua.edu.cn

```
int fac(int n) { return (n < 2) ? 1 : n * fac(n - 1); }
```



```
int fib( int n ) { return (n < 2) ? n - fib(n - 1) + fib(n - 2); }
```

main(2,3)
x

main(2,3)
x
fib(3)
x + x

main(2,3)
x
fib(3)
x + x
fib(2)
x + x

main(2,3)
x
fib(3)
x + x
fib(2)
x + x
fib(1)
1

main(2,3)
x
fib(3)
x + x
fib(2)
1 + x

main(2,3)
x
fib(3)
x + x
fib(2)
1 + x
fib(0)
0



main(2,3)
2

main(2,3)
x
fib(3)
1 + 1

main(2,3)
x
fib(3)
1 + x
fib(1)
1

main(2,3)
x
fib(3)
1 + x

main(2,3)
x
fib(3)
x + x
fib(2)
1 + 0

空间复杂度

```
❖ hailstone(int n) {  
    if ( 1 < n )  
        n % 2 ? odd( n ) : even( n );  
}  
  
❖ even( int n ) { hailstone( n / 2 ); }  
odd( int n ) { hailstone( 3*n + 1 ); }  
  
❖ main( int argc, char* argv[] )  
{ hailstone( atoi( argv[1] ) ); }  
  
❖ 可见，递归算法所需的空间  
主要取决于递归深度，而非递归实例总数
```

call stack

```
main(2, 10)  
hailstone(10)  
even(10)  
hailstone(5)  
odd(5)  
hailstone(16)  
even(16)  
hailstone(8)  
even(8)  
hailstone(4)  
even(4)  
hailstone(2)  
even(2)  
hailstone(1)
```

call stack

```
main(2, 27)  
hailstone(27)  
odd(27)  
hailstone(82)  
even(82)  
hailstone(41)  
odd(41)  
hailstone(124)  
even(124)  
hailstone(62)  
even(62)  
hailstone(31)  
odd(31)  
hailstone(94)  
... ...
```