



```
/*  
 * convolve.c  
 */
```

```
/* Standard includes */  
#include <assert.h>  
#include <math.h>  
#include <stdlib.h> /* malloc(), realloc() */
```

```
/* Our includes */  
#include "base.h"  
#include "error.h"  
#include "convolve.h"  
#include "klt_util.h" /* printing */
```

```
#define MAX_KERNEL_WIDTH 71
```

```
typedef struct {  
    int width;  
    float data[MAX_KERNEL_WIDTH];  
} ConvolutionKernel;
```

```
/* Kernels */
```

Fundamentals of Programming

lecture 11

Recursion, Call stack

Recursion

factorial function:

- $n! = 1 * 2 * \dots * n$

Recursion

factorial function:

- $n! = 1 * 2 * \dots * n$

```
#include <stdio.h>

int fact(int);

int main() {
    int n;

    scanf("%d", &n);

    printf("%d\n", fact(n));
}

int fact(int n) {
    int p = 1;

    for (int i = 2; i <= n; i++)
        p *= i;

    return p;
}
```

Recursion

recursive definition:

- $0! = 1$
- $n! = n * (n-1)!$

Recursion

recursive definition:

- $0! = 1$
- $n! = n * (n-1)!$

```
#include <stdio.h>

int fact(int);

int main() {
    int n;

    scanf("%d", &n);

    printf("%d\n", fact(n));
}

int fact(int n) {
    double f;

    if (n == 0)
        return 1;

    // ???
}
```

Recursion

recursive definition:

- $0! = 1$
- $n! = n * (n-1)!$

```
#include <stdio.h>

int fact(int);

int main() {
    int n;

    scanf("%d", &n);

    printf("%d\n", fact(n));
}

int fact(int n) {
    double f;

    if (n == 0)
        return 1;

    f = fact(n-1);

    return n*f;
}
```

Recursion

recursive definition:

- $0! = 1$
- $n! = n * (n-1)!$

```
#include <stdio.h>

int fact(int);

int main() {
    int n;

    scanf("%d", &n);

    printf("%d\n", fact(n));
}

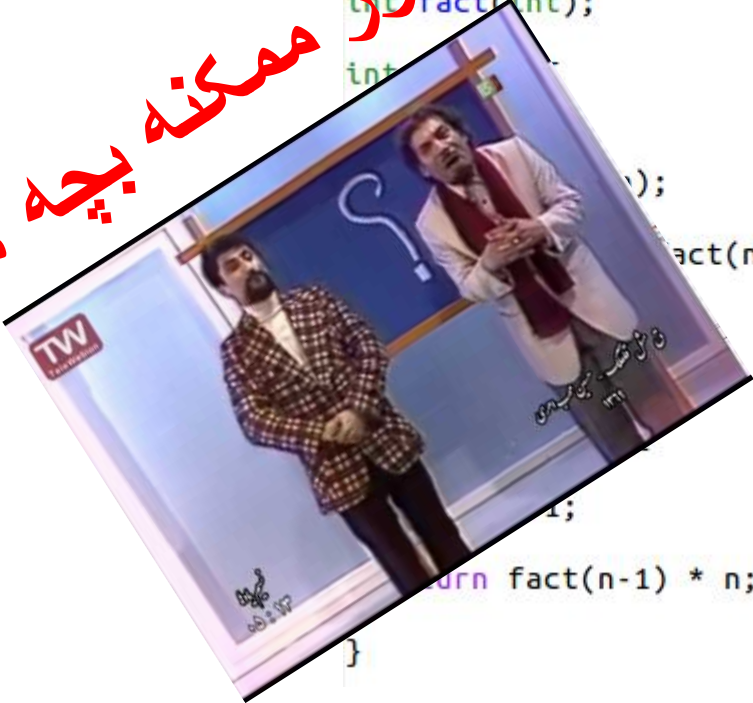
int fact(int n) {
    if (n == 0)
        return 1;

    return fact(n-1) * n;
}
```

Recursion

recursive definition:

- $0! = 1$
- $n! = n * (n-1)!$

چطور ممکنه بچه ها؟! 

```
#include <stdio.h>
```

```
int fact(int);
```

```
int
```

```
);
```

```
fact(n));
```

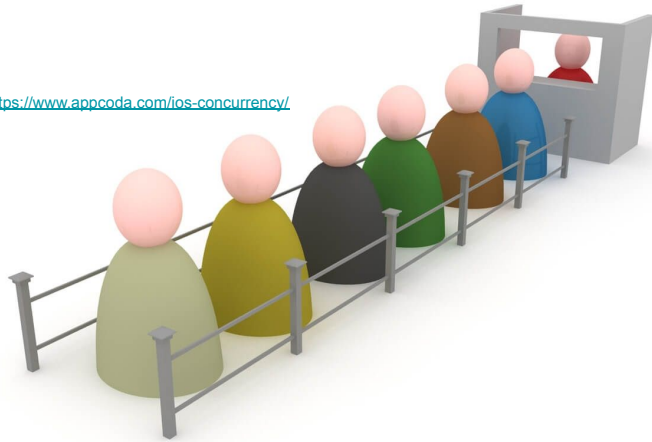
```
1;
```

```
return fact(n-1) * n;
```

```
}
```


Queue vs Stack

<https://www.appcoda.com/ios-concurrency/>



queue

http://wikiclipart.com/stack-of-books-clipart_22617/



stack

Call stack

```
#include <stdio.h>

int fact(int);

int main() {
    int m = 4;

    printf("%d\n", fact(m));
}

int fact(int n) {
    int f;

    if (n == 0)
        return 1;

    f = fact(n-1);

    return n*f;
}
```

stack frame:

f = ?
return address
n=4

Call stack

```
#include <stdio.h>

int fact(int);

int main() {
    int m = 4;

    printf("%d\n", fact(m));
}

int fact(int n) {
    int f;

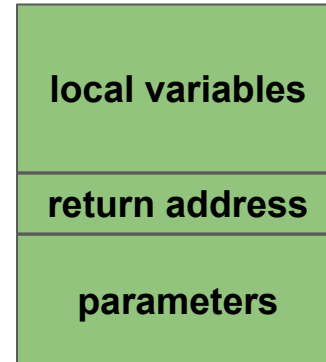
    if (n == 0)
        return 1;

    f = fact(n-1);

    return n*f;
}
```

stack frame:

- parameters (arguments)
- return address
- local variables



```
#include <stdio.h>

int fact(int);

int main() {
    int m = 4;
    int a = fact(m);

    printf("%d\n", a);
}

int fact(int n) {
    int f;

    if (n == 0)
        return 1;

    f = fact(n-1);

    return n*f;
}
```

stack frame (main):

a = ?
m = 4

```

#include <stdio.h>

int fact(int);

int main() {
    int m = 4;
    int a = fact(m);

    printf("%d\n", a);
}

int fact(int n) {
    int f;

    if (n == 0)
        return 1;

    f = fact(n-1);

    return n*f;
}

```

int f;

stack frame (fact):

stack frame (main):

f = ?
return address
n=4
a = ?
m = 4

```

#include <stdio.h>

int fact(int);

int main() {
    int m = 4;
    int a = fact(m);

    printf("%d\n", a);
}

int fact(int n) {
    int f;

    if (n == 0)
        return 1;

    f = fact(n-1);

    return n*f;
}

```

stack frame (fact):

f = ?
return address
n=3

stack frame (fact):

f = ?
return address
n=4

stack frame (main):

a = ?
m = 4

```

#include <stdio.h>

int fact(int);

int main() {
    int m = 4;
    int a = fact(m);

    printf("%d\n", a);
}

int fact(int n) {
    int f;

    if (n == 0)
        return 1;

    f = fact(n-1);

    return n*f;
}

```

stack frame (fact):

f = ?
return address
n=2

stack frame (fact):

f = ?
return address
n=3

stack frame (fact):

f = ?
return address
n=4

stack frame (main):

a = ?
m = 4

```

#include <stdio.h>

int fact(int);

int main() {
    int m = 4;
    int a = fact(m);

    printf("%d\n", a);
}

int fact(int n) {
    int f;

    if (n == 0)
        return 1;

    f = fact(n-1);

    return n*f;
}

```

stack frame (fact):

f = ?
return address
n=1

stack frame (fact):

f = ?
return address
n=2

stack frame (fact):

f = ?
return address
n=3

stack frame (fact):

f = ?
return address
n=4

stack frame (main):

a = ?
m = 4


```

#include <stdio.h>

int fact(int);

int main() {
    int m = 4;
    int a = fact(m);

    printf("%d\n", a);
}

int fact(int n) {
    int f;

    if (n == 0)
        return 1;

    f = fact(n-1);

    return n*f;
}

```

stack frame (fact):

f = ?
return address
n=0

stack frame (fact):

f = ?
return address
n=1

stack frame (fact):

f = ?
return address
n=2

stack frame (fact):

f = ?
return address
n=3

stack frame (fact):

f = ?
return address
n=4

stack frame (main):

a = ?
m = 4

```

#include <stdio.h>

int fact(int);

int main() {
    int m = 4;
    int a = fact(m);

    printf("%d\n", a);
}

int fact(int n) {
    int f;

    if (n == 0)
        return 1;

    f = fact(n-1);

    return n*f;
}

```

stack frame (fact):

f = 1
return address
n=1

stack frame (fact):

f = ?
return address
n=2

stack frame (fact):

f = ?
return address
n=3

stack frame (fact):

f = ?
return address
n=4

stack frame (main):

a = ?
m = 4

```

#include <stdio.h>

int fact(int);

int main() {
    int m = 4;
    int a = fact(m);

    printf("%d\n", a);
}

int fact(int n) {
    int f;

    if (n == 0)
        return 1;

    f = fact(n-1);

    return n*f;
}

```

stack frame (fact):

f = 1
return address
n=2

stack frame (fact):

f = ?
return address
n=3

stack frame (fact):

f = ?
return address
n=4

stack frame (main):

a = ?
m = 4

```

#include <stdio.h>

int fact(int);

int main() {
    int m = 4;
    int a = fact(m);

    printf("%d\n", a);
}

int fact(int n) {
    int f;

    if (n == 0)
        return 1;

    f = fact(n-1);

    return n*f;
}

```

stack frame (fact):

f = 2
return address
n=3

stack frame (fact):

f = ?
return address
n=4

stack frame (main):

a = ?
m = 4

```
#include <stdio.h>

int fact(int);

int main() {
    int m = 4;
    int a = fact(m);

    printf("%d\n", a);
}

int fact(int n) {
    int f;

    if (n == 0)
        return 1;

    f = fact(n-1);

    return n*f;
}
```

stack frame (fact):

f = 6
return address
n=4

stack frame (main):

a = ?
m = 4

```
#include <stdio.h>

int fact(int);

int main() {
    int m = 4;
    int a = fact(m);

    printf("%d\n", a);
}

int fact(int n) {
    int f;

    if (n == 0)
        return 1;

    f = fact(n-1);

    return n*f;
}
```

stack frame (main):

a = 24
m = 4

```
#include <stdio.h>

int fact(int);

int main() {
    int m = 4;
    int a = fact(m);

    printf("%d\n", a);
}

int fact(int n) {
    int f;

    if (n == 0)
        return 1;

    f = fact(n-1);

    return n*f;
}
```

what if $n < 0$?

The ? : conditional operator

- `a = (i > 10 ? 1 : 0)`
- ternary operator

Category	Operator	Associativity
Postfix	() [] -> . ++ --	Left to right
Unary	+ - ! ~ ++ -- (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR		Left to right
Logical AND	&&	Left to right
Logical OR		Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %= >>= <<= &= ^= =	Right to left
Comma	,	Left to right

https://www.tutorialspoint.com/cprogramming/c_operators_precedence.htm

The ? : operator

- `a = (i > 10 ? 1 : 0)`
- `a = i > 10 ? 1 : 0`
-

The ? : operator

- `a = (i > 10 ? 1 : 0)`
- `a = i > 10 ? 1 : 0`
- `a = b >= 0 ? b : -b`

The ? : operator

- `a = (i > 10 ? 1 : 0)`
- `a = i > 10 ? 1 : 0`
- `a = b >= 0 ? b : -b`

`// a = abs(b)`

The ? : operator

- `a = (i > 10 ? 1 : 0)`
- `a = i > 10 ? 1 : 0`
- `a = b >= 0 ? b : -b`
- `z = x >= y ? x : y`

`// a = abs(b)`

The ? : operator

- `a = (i > 10 ? 1 : 0)`
- `a = i > 10 ? 1 : 0`
- `a = b >= 0 ? b : -b` `// a = abs(b)`
- `z = x >= y ? x : y` `// z = max(x, y)`
- `z = x >= y ? 2*x+y : y*x;`

The ? : operator

```
int fact(int n) {  
    return n > 0 ? n * fact(n-1) : 1;  
}
```

Recursion

Fibonacci series

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

Recursion

Fibonacci series

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

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

fibonacci.c

Recursion

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

fibonacci.c

```
int fibo(int n) {  
    int a=1, b=1;  
  
    for (int i=2; i < n; i++) {  
        b = a+b;  
        a = b-a;  
    }  
  
    return b;  
}
```

fibonacci2.c

automatic vs static local variables

```
#include <stdio.h>

void testfunc(void);

int main() {
    for (int i = 1; i <= 10; i++)
        testfunc();
}

void testfunc() {
    int a = 1;
    static int b = 1;

    printf("a=%d, b=%d\n",a,b);

    a++;
    b++;
}
```

automatic vs static local variables

```
#include <stdio.h>

void testfunc(void);

int main() {
    for (int i = 1; i <= 10; i++)
        testfunc();
}

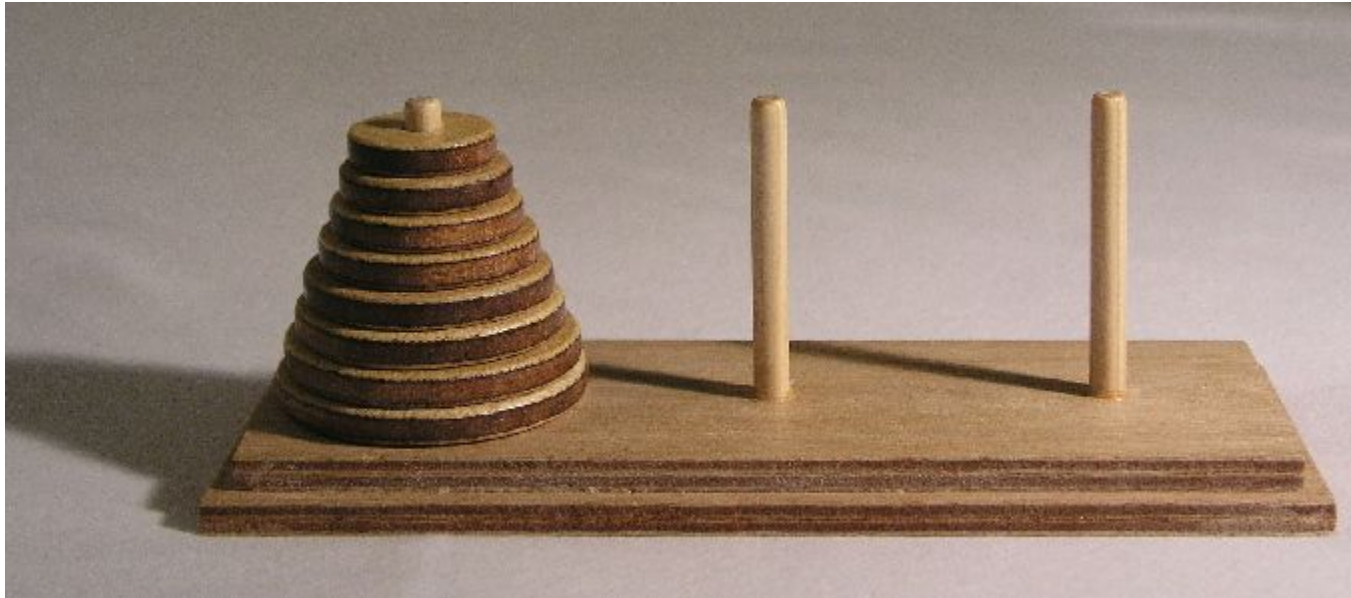
void testfunc() {
    int a = 1;
    static int b = 1;

    printf("a=%d, b=%d\n",a,b);

    a++;
    b++;
}
```

```
a=1, b=1
a=1, b=2
a=1, b=3
a=1, b=4
a=1, b=5
a=1, b=6
a=1, b=7
a=1, b=8
a=1, b=9
a=1, b=10
```

Tower of Hanoi



https://en.wikipedia.org/wiki/Tower_of_Hanoi

Tower of Hanoi



<http://larc.unt.edu/ian/TowersOfHanoi/activelearning.html>

Tower of Hanoi

```
#include <stdio.h>

void hanoi(int, int, int, int);

int main() {
    int n;

    scanf("%d", &n);

    hanoi(n, 1, 2, 3);
}

void hanoi(int n, int from, int to, int aux) {
    if (n == 1) {
        printf("%d -> %d\n", from, to);
        return;
    }

    hanoi(n-1, from, aux, to);
    printf("%d -> %d\n", from, to);
    hanoi(n-1, aux, to, from);
}
```

Tower of Hanoi

```
#include <stdio.h>

void hanoi(int, int, int, int);

int main() {
    int n;

    scanf("%d", &n);

    hanoi(n, 1, 2, 3);
}

void hanoi(int n, int from, int to, int aux) {
    if (n == 0)
        return;

    hanoi(n-1, from, aux, to);
    printf("%d -> %d\n", from, to);
    hanoi(n-1, aux, to, from);
}
```