第7章

函数

```c
#include <stdio.h>

void two();
void three();

int main()
{
    printf("I'm in main.\n");
    two();

    return 0;
}

void two()
{
    printf("I'm in two.\n");
    three();
}

void three()
{
    printf("I'm in three.\n");
}
```

关于函数

- 程序由多个函数组成
- 程序的执行总是从main函数开始
- 所有函数都是平行、独立的，一个函数不属于另一个

函数定义

void, int…

标识符

逗号分隔，可以为空

```
<返回值类型> <函数名>(<参数列表>)
{
        <函数体>
}
```

可以为空

```
int min(int a, int b)
{
    return a < b ? a : b;
}
```

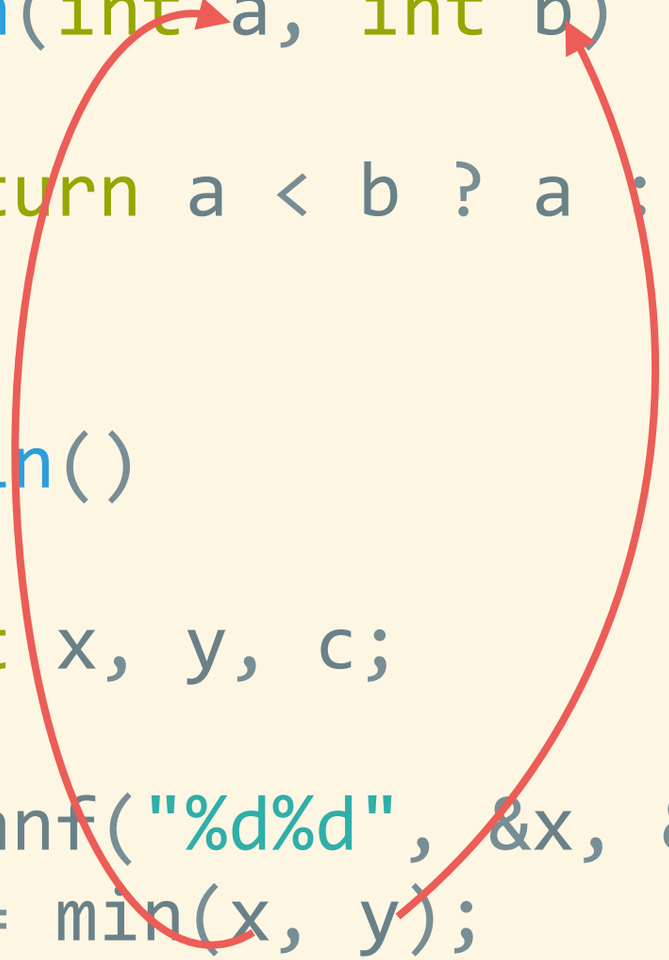
调用过程

```c
#include <stdio.h>

int min(int a, int b)
{
    return a < b ? a : b;
}

int main()
{
    int x, y, c;

    scanf("%d%d", &x, &y);
    c = min(x, y);

    printf("%d\n", c);

    return 0;
}
```

被调函数

主调函数

```c
#include <stdio.h>

int min(int a, int b)
{
    return a < b ? a : b;
}

int main()
{
    int x, y, c;

    scanf("%d%d", &x, &y);
    c = min(x, y);

    printf("%d\n", c);

    return 0;
}
```



值传递

求最大公约数

```c
int gcd(int u, int v)
{
    int tmp;

    while (v != 0) {
        tmp = u % v;
        u = v;
        v = tmp;
    }

    return u;
}
```

```c
#include <stdio.h>

int main()
{
    int x, y, z;

    x = 145;
    y = 25;
    z = gcd(x, y); /* 1 */
    printf("GCD(%d, %d) = %d\n", x, y, z);

    x = 16;
    y = 24;
    printf("GCD(%d, %d) = %d\n", x, y, gcd(x, y)); /* 2 */

    z = gcd(x, x + y); /* 3 */
    printf("GCD(%d, %d) = %d\n", x, x + y, z);

    return 0;
}
```

函数可以多次调用

函数返回值

返回值的形式

可以为空

```
return <表达式>;
```

返回值作用

- 从被调函数退出，返回到调用前位置
- 可以返回1个值（也可以不返回值）

多个return

- 执行到哪一个哪个起作用

返回值类型

- 由函数定义决定

```
int min(int x, int y);

float max(float x, float y);

double sin(double x);
```

函数的调用

函数调用形式

函数名(参数列表);

```
y = sin(x);

z = min(x, y);

n = printf("Hello, %d", x);
```

说明

- 函数调用语句也有“值”和“类型”
- 调用完成后，返回主调函数执行下一条语句

调用之前……

- 函数必须存在
- 必须声明

返回值类型 函数名(参数列表类型);

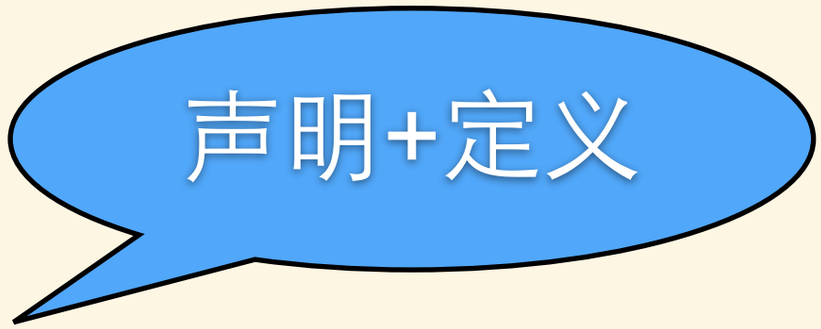
声明在前，定义在后

```c
#include <stdio.h>

int min(int a, int b);

int main()
{
    int x, y, c;

    scanf("%d%d", &x, &y);
    c = min(x, y);

    printf("%d\n", c);

    return 0;
}

int min(int a, int b)
{
    return a < b ? a : b;
}
```

```c
#include <stdio.h>

int min(int, int);

int main()
{
    int x, y, c;

    scanf("%d%d", &x, &y);
    c = min(x, y);

    printf("%d\n", c);

    return 0;
}

int min(int a, int b)
{
    return a < b ? a : b;
}
```

声明定义在一起

```c
#include <stdio.h>

int min(int a, int b)
{
    return a < b ? a : b;
}

int main()
{
    int x, y, c;

    scanf("%d%d", &x, &y);
    c = min(x, y);

    printf("%d\n", c);

    return 0;
}
```

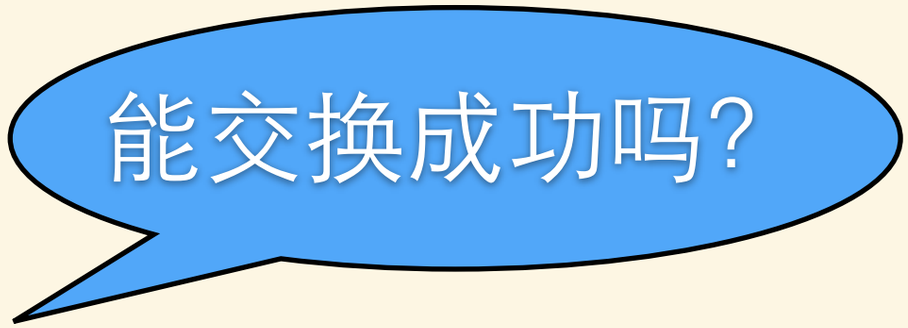


printf是如何声明的?

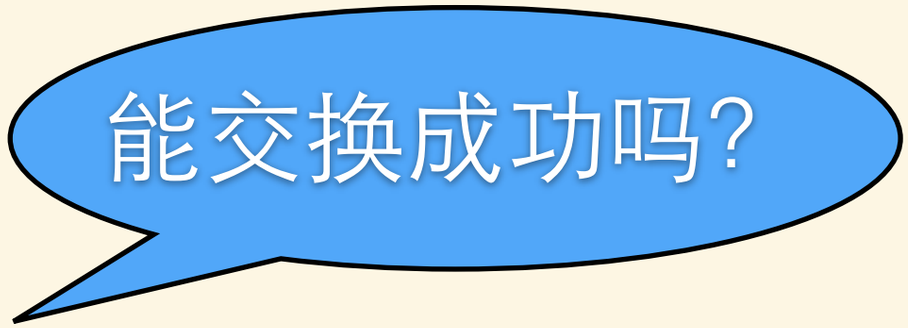
include 背后做了什么……

参数的值传递

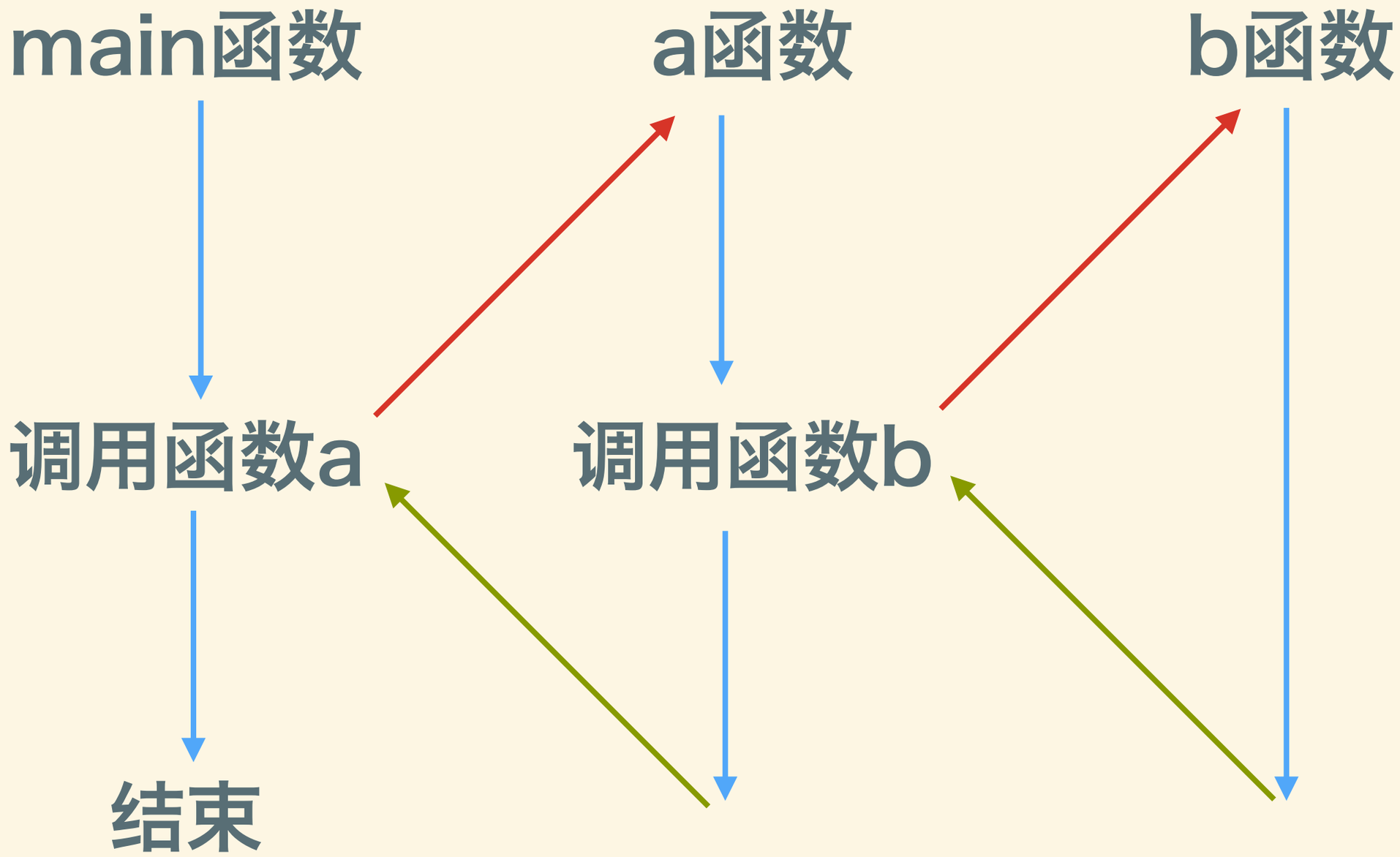
```c
#include <stdio.h>

void swap(int x, int y)
{
    int tmp;

    tmp = x;
    x = y;
    y = x;
}

int main()
{
    int a = 10;
    int b = 20;

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

    swap(a, b);

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

    return 0;
}
```



```c
#include <stdio.h>

void swap(int x, int y)
{
    int tmp;

    tmp = x;
    x = y;
    y = x;
}

int main()
{
    int a = 10;
    int b = 20;

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

    swap(a, b);

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

    return 0;
}
```



嵌套调用


main函数
a函数
b函数
调用函数a
调用函数b
结束

递归调用

一个函数直接或间接地调用自身……

计算n!

循环

```c
#include <stdio.h>

int fact(int n)
{
    int i, result;

    result = 1;
    for (i = 1; i <= n; i++)
        result *= i;

    return result;
}

int main()
{
    int n;

    scanf("%d", &n);
    printf("%d\n", fact(n));

    return 0;
}
```

递归

```c
#include <stdio.h>

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

int main()
{
    int n;

    scanf("%d", &n);
    printf("%d\n", fact(n));

    return 0;
}
```