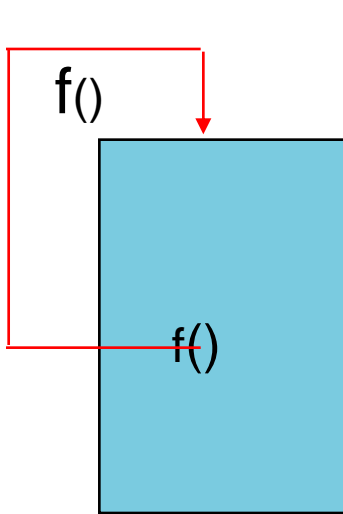


4. 递归函数

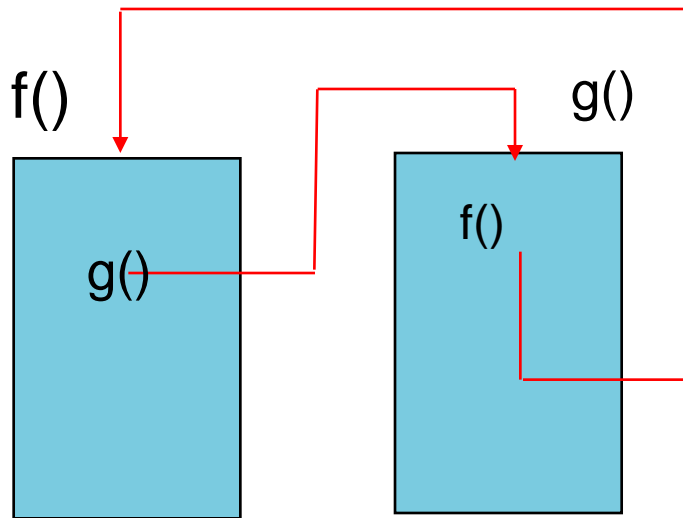
杨振平

递归函数

- 递归函数是直接或间接地调用了自身的函数。



直接递归调用



间接递归调用

利用递归算法可以将一个规模较大的问题转化为规模较小的同类问题来求解。

例如：数学上常见的一个问题-计算一个非负整数 n !

计算 $n!$ ，只要计算出 $(n-1)!$ ，则 $n!=n.(n-1)!$

计算 $(n-1)!$ ，只要计算出 $(n-2)!$ ，则 $(n-1)!=(n-1).(n-2)!$

计算 $(n-2)!$ ，只要计算出 $(n-3)!$ ，则 $(n-2)!=(n-2).(n-3)!$

...

计算 $3!$ ，只要计算出 $2!$ ，则 $3!=3.(2!)$

计算 $2!$ ，只要计算出 $1!$ ，则 $2!=2.(1!)$

而 $1!=1$ （注： $0!$ 定义为1）

递归函数-计算一个非负整数n!

$$f(n) = \begin{cases} 1 & n = 0 \\ n * f(n-1) & n > 0 \end{cases}$$

特征：

1. 定义中包含该函数本身（即递归公式）
2. 必须有终止条件

递归调用过程分为两个阶段-递推和回归

例如：利用递归函数计算5！

是从已知条件出发，按递推的逆过程，逐个求值，最后到达递推的开头，解决原问题。

回归

递推

将原问题不断分解为新的规模更小的问题，逐渐从未知向已知方向推测

f(5)

5*f(4)

4*f(3)

3*f(2)

2*f(1)

1*f(0)

1

5! = 120

= 120

= 24

= 6

= 2

= 1

计算n!的递归函数-f

```
int f(int n) //计算n!  
{
```

```
    if(n==0)
```

```
        return 1;
```

```
    else
```


```
        return n*f(n-1); //直接递归调用
```

```
}
```


$$f(n) = \begin{cases} 1 & n = 0 \\ n * f(n-1) & n > 0 \end{cases}$$

递归的调用过程

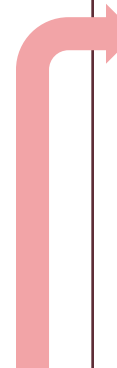
```
int main()
{
    int n;
    int factorial;
    cin >> n;
    factorial = f(n);
    cout << factorial
    << endl;
    return 0;
}  设输入n=3
```



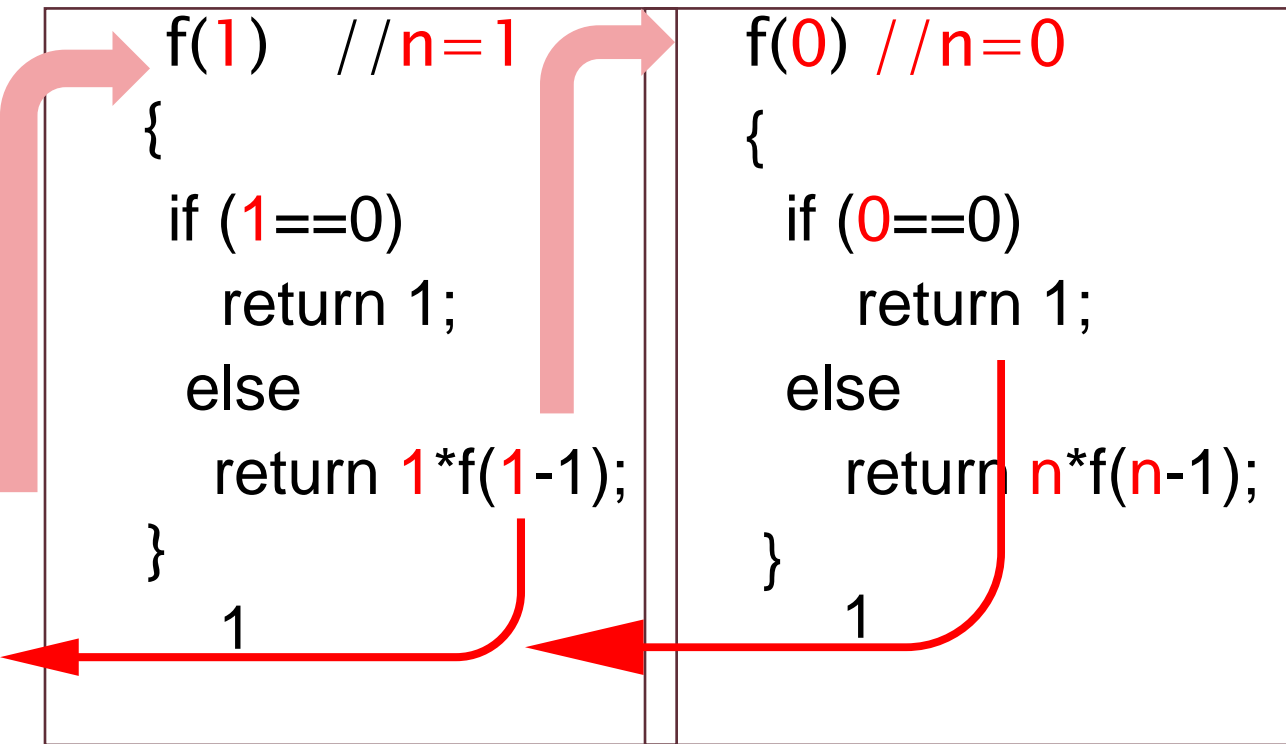
```
f(3) //n=3
{
    if (3==0)
        return 1;
    else
        return 3*f(3-1);
}
```



```
f(2) //n=2
{
    if (2==0)
        return 1;
    else
        return 2*f(2-1);
}
```



递归的调用过程



递归的调用过程

