数据结构和算法

作者：小甲鱼

让编程改变世界

Change the world by program

递归定义

- 在高级语言中，函数自己调用和调用其他函数并没有本质的不同。我们把一个直接调用自己或通过一系列的调用语句间接地调用自己的函数，称作递归函数。
- 不过，写递归程序最怕的就是陷入永不结束的无穷递归中。切记，每个递归定义必须至少有一个条件，当满足这个条件时递归不再进行，即函数不再调用自身而是返回值。
- 比如之前我们的Fbi函数结束条件是：i <2。

递归定义

- 对比了两种实现斐波那契的代码，迭代和递归的区别是：迭代使用的是循环结构，递归使用的是选择结构。
- 使用递归能使程序的结构更清晰、更简洁、更容易让人理解，从而减少读懂代码的时间。
- 但大量的递归调用会建立函数的副本，会消耗大量的时间和内存，而迭代则不需要此种付出。
- 递归函数分为调用和回退阶段，递归的回退顺序是它调用顺序的逆序。

递归定义

- 举个例子，计算n的阶乘n!

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

- 这样我们就不难设计出递归算法：

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

递归定义

- 假设我们n的值传入是5，那么：

factorial(5) = 5 * factorial(4)

factorial(4) = 4 * factorial(3)

factorial(3) = 3 * factorial(2)

factorial(2) = 2 * factorial(1)

factorial(1) = 1 * factorial(0)

实例分析

- 题目要求：编写一个递归函数，实现将输入的任意长度的字符串反向输出的功能。例如输入字符串FishC，则输出字符串ChsiF。
- 应用递归的思想有时可以很轻松地实现一些看似不太容易实现的功能，例如这道题。
- 要将一个字符串反向地输出，童鞋们一般采用的方法是将该字符串存放到一个数组中，然后将数组元素反向的输出即可。这道题要求输入是任意长度，所以不用递归的话，实现起来会比较麻烦（当然你可以用之前我们讲过的动态申请内存那招）。

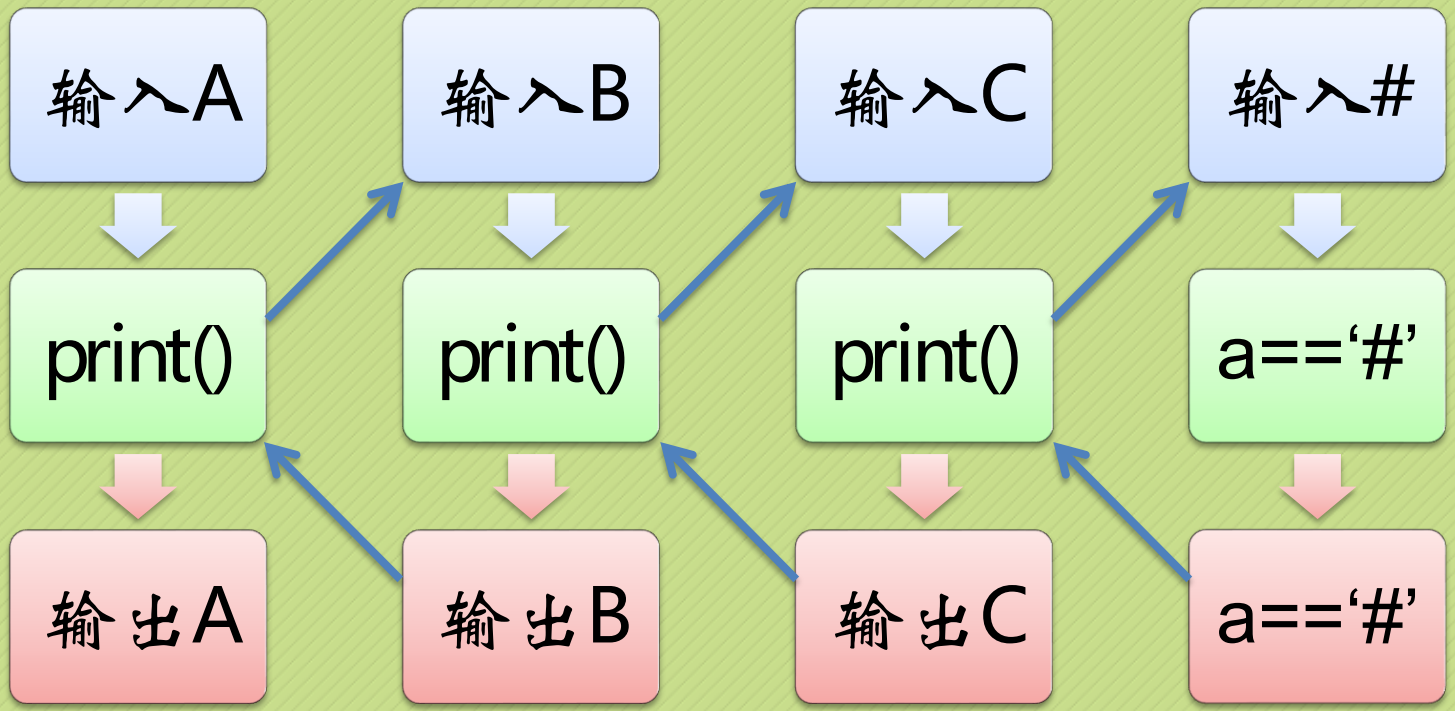
实例分析

- 我们说过，递归它需要有一个结束的条件，那么我们可以将“#”作为一个输入结束的条件。

```
void print()
{
    char a;
    scanf( %c", &a);
    if( a != '#' ) print();
    if( a != '#' ) printf( %c", a);
}
```

实例分析

- 假设我们从屏幕上输入字符串：ABC#



分治思想

- 分而治之的思想古已有之，秦灭六国，统一天下正是采取各个击破、分而治之的原则。
- 而分治思想在算法设计中也是非常常见的，当一个问题规模较大且不易求解的时候，就可以考虑将问题分成几个小的模块，逐一解决。
- 分治思想和递归算是有亲兄弟的关系了，因为采用分治思想处理问题，其各个小模块通常具有与大问题相同的结构，这种特性也使递归技术有了用武之地。我们接下来通过实例来讲解。

折半查找算法的递归实现

- 折半查找法是一种常用的查找方法，该方法通过不断缩小一半查找的范围，直到达到目的，所以效率比较高。
- 因为这个在《零基础入门学习C语言》等基础教程中已经详细讲解过，小甲鱼这里就通过文字教程简单给大家回顾下算法的主要思路。
- 从算法的折半查找的过程我们不难看出，这实际上也是一个递归的过程：因为每次都将问题的规模减小至原来的一半，而缩小后的子问题和原问题类型保持一致。

折半查找算法的递归实现

- 作为课后题大家实现并体验下分治思想的妙处~
- 下节课我们将讨论递归算法和分治思想最有名的典型例题：汉诺塔问题