# 定义

一个函数直接或间接调用自己。

# 原理

## 函数调用

当在一个函数的运行期间调用另一个函数时，在运行被调函数之前，系统需要完成三件事：

1. 将所有的实际参数，返回地址（下一个函数地址）等信息传递给被调函数保存；
2. 为被调函数的局部变量（也包括形参）分配存储空间；
3. 将控制转移到被调函数的入口。

从被调函数返回主函数之前，系统也要完成三件事：

1. 保存被调函数的返回结果；
2. 释放被调函数所占的存储空间；
3. 依照被调函数保存的返回地址将控制转移到调用函数。

当有多个函数相互调用时，按照“后调用先返回”的原则，上述函数之间信息传递和控制转移必须借助“栈”来实现，即系统将整个程序运行时所需的数据空间安排在一个栈中，每当调用一个函数时，就在栈顶分配一个存储区，进行压栈操作，每当一个函数退出时，就释放它的存储区，进行出栈操作，当前运行的函数永远在栈顶位置。

A函数调用A函数和A函数调用B函数在计算机看来是没有任何区别的，只不过用我们思维理解比较奇怪。

## 满足条件

递归需要满足的三个条件：

1. 递归必须得有一个明确的中止条件；
2. 该函数所处理的数据规模必须在递减；
3. 这个转换必须是可解的。

## 循环和递归

所有的循环都可以用递归实现，所有的递归不一定可以用循环实现。

递归特点：

1. 易于理解；
2. 速度慢；
3. 存储空间大。

循环特点：

1. 不易理解；
2. 速度快；
3. 存储空间小。

# 应用

树和森林都是以递归的方式定义的

树和图的很多算法都是以递归来实现的

很多数学公式就是以递归方式实现的，比如斐波拉切序列

## 求阶乘

long f(long n)

{

if(1==n)

{

return 1;

}

else

{

return f(n-1)\*n;

}

}

## 1+2+3+…+100的和

long sum(int n)

{

if(1==n)

{

return 1;

}

else

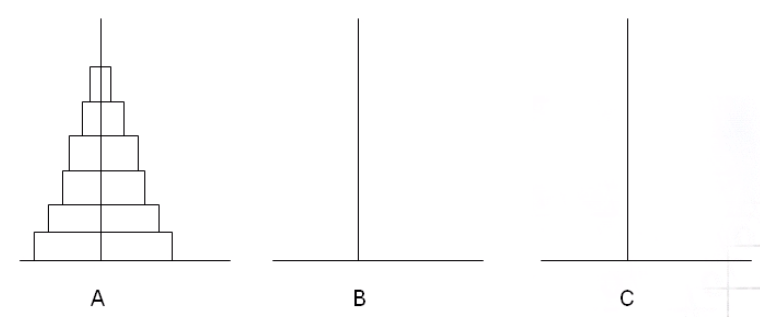
{

return n+sum(n-1);

}

}

## 汉诺塔



如何把A上面的N个盘子借助B移动到C上，要求：

1. 一次只能移动一个盘子；
2. 移动过程当中大盘子永远不能放在小盘子上面

## 走迷宫