12.4-递归函数

C++允许函数自己调用自己(如A函数可以调用A函数本身，称为直接递归)。也允许A函数调用B函数，而后B函数又调用A函数(从而形成间接递归)。

但不论使用哪种递归，程序员都应保障递归函数在执行若干次后能够“退出”递归(不再进行递归调用，也即能够实现递归出口)

递归函数的执行分为“递推”和“回归”两个过程，这两个过程由递归终止条件控制，即逐层递推，直至递归终止条件，然后逐层回归。每次调用发生时都首先判断递归终止条件。

代码示例1：

一只小猴摘了一些桃子回来，第一天就一口气吃了一半，还不过瘾，就又多吃了一个； 第二天，又吃了剩下的桃子的一半，还不过瘾，又多吃了一个；以后每天都吃前一天剩下的一半多一个，到第10天想再吃时，发现只剩下一个桃子了。问小猴一共摘了多少桃子回来？

（1534个）

#include<iostream>

using namespace std;

int peach(int x)

{

//第x天剩余的桃子数

if(x<10)

{

return 2\*(peach(x+1)+1);

}

if(x==10)

{

return 1;

}

}

/\*

分析：

第一次调用：要想知道第一天的桃子数，先需要知道第二天的 peach(1)=(peach(2)+1)\*2

第二次调用：要想知道第二天的桃子数，先需要知道第三天的 peach(2)=(peach(3)+1)\*2

第三次调用：要想知道第三天的桃子数，先需要知道第四天的 peach(3)=(peach(4)+1)\*2

......

第九次调用：要想知道第九天的桃子数，先需要知道第十天的 peach(9)=(peach(10)+1)\*2

第十次调用/第一次回归：第十天剩余一个桃子 peach(10)=1

第二次回归：第九天桃子数为4 peach(9)=(peach(10)+1)\*2=4；

第三次回归：第八天桃子数为10 peach(8)=(peach(9)+1)\*2=10；

......

第十次回归：第一天桃子数为1534 peach(1)=(peach(2)+1)\*2=1534；

类比函数的递推公式

\*/

int main()

{

int d[11]; //第i天剩余的桃子数

d[10]=1;

int i=9;

while(i>=1)

{

d[i]=(d[i+1]+1)\*2;

i--;

}

cout<<d[1]<<endl;

cout<<peach(1)<<endl;

return 0;

}

代码示例2：

利用递归函数求n!

#include<iostream>

using namespace std;

int fac1(int n)

{

if(n==0 || n==1)

{

return 1;

}

else

{

return n\*fac1(n-1);

}

}

/\*

分析：n!=n\*(n-1)!=n\*(n-1)\*(n-2)!=...=n\*(n-1)\*(n-2)\*(n-3)\*.....\*1!

\*/

int fac2(int n)

{

if(n==0 || n==1)

{

return 1;

}

else

{

int res=1;

while(n>1)

{

res\*=n;

n--;

}

return res;

}

}

int main()

{

int n;

cin>>n;

cout<<fac1(n)<<endl;

cout<<fac2(n)<<endl;

return 0;

}

代码示例3：

利用递归函数求斐波那契数列第n项

0、1、1、2、3、5、8、13、21、34、55、89

#include<iostream>

using namespace std;

long long Fibo(int n)

{

if(n==0)

{

return 0;

}

if(n==1)

{

return 1;

}

return Fibo(n-1)+Fibo(n-2);

}

int main()

{

int n;

cin>>n;

cout<<Fibo(n)<<endl;

return 0;

}

代码示例4：

利用递归函数倒序输出数字串

#include<iostream>

using namespace std;

int rev(int n)

{

cout<<n%10;

if(n<10)

return 0;

else

return rev((n-n%10)/10);

}

int main()

{

int a;

cin>>a;

rev(a);

return 0;

}