《计算概论A》课程 程序设计部分

函数的递归调用（1）

李　戈

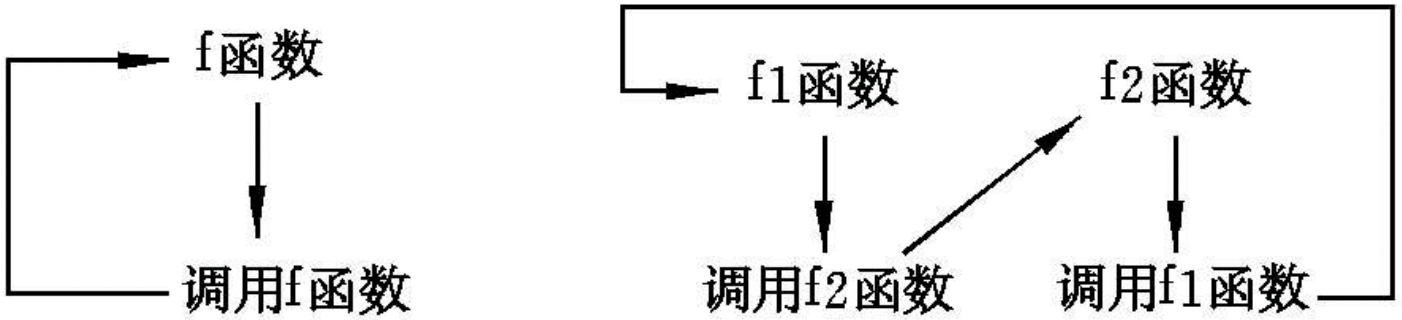
北京大学 信息科学技术学院 软件研究所

lige@sei.pku.edu.cn

北京大学

函数的递归调用

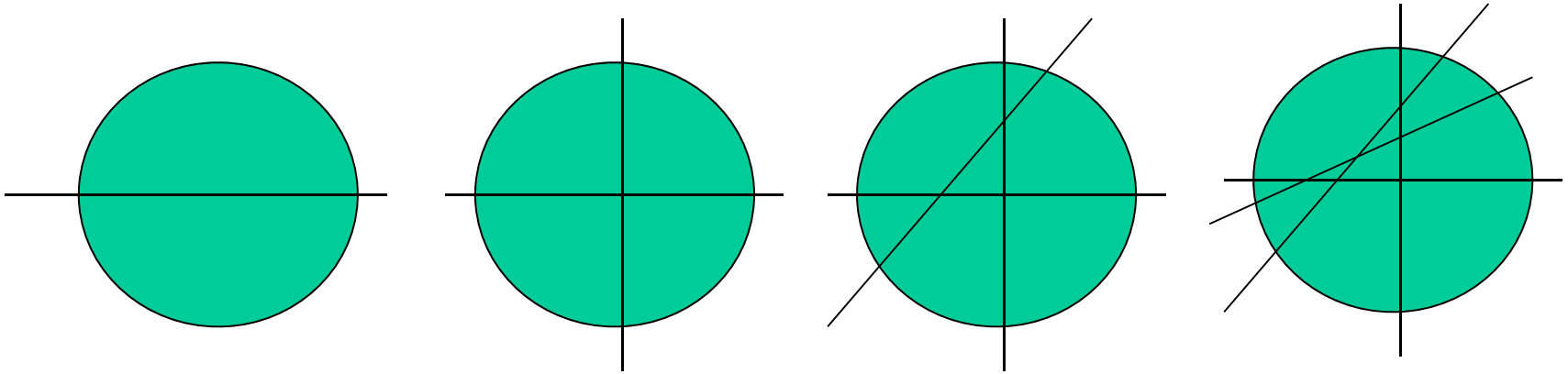
- C++语言允许递归调用
  - 在调用一个函数的过程中又出现直接或间接地调用该函数本身。



  - 程序中不应出现无终止的递归调用，必须控制只有在某一条件成立时才继续执行递归调用，否则就不再继续。

切饼

- 切饼，100刀最多能切多少块？

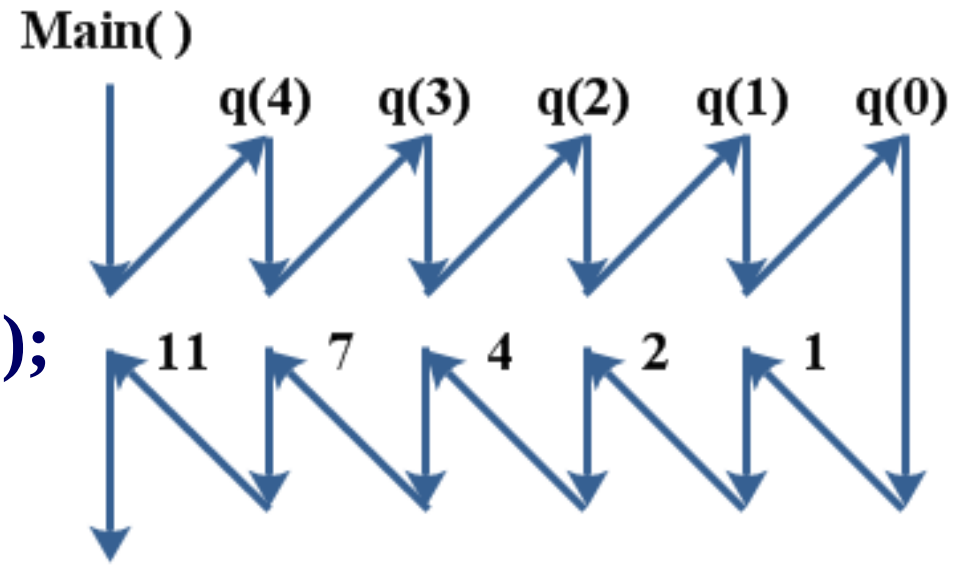


- q(1)=1+1=2
- q(2)=1+1+2=4;
- q(3)=1+1+2+3=7;
- q(4)=1+1+2+3+4=11;
- q(n)=q(n−1)+n;q(0)=1;

切饼

```
#include<iostream>
using namespace std;
int q(int n){
    if (n == 0)
        return 1;
    else
        return( n + q(n-1));
}

int main( ){
        cout<<q(4)<<endl;
        return 0;
}
```



用循环的解决方案

```
#include <iostream>
using namespace std;
int main()
{  int q[101];
   q[0] = 1;
   for (int i = 1; i <= 100; i++)
   {
        q[i] = q[i-1] + i;
   }
   cout<<"100刀最多可切"<<q[100]<<"块"<<endl;
   return 0;
}
```

递归与循环的不同

- 通常意义上
  - 循环的关注点放在起始点条件

而

  - 递归的关注点放在求解目标上

方法总结之一

- **如何利用递归解决问题？**
  - **把关注点放在要求解的目标上**

  进而

  - **找到第n次做与第n-1次做之间的关系，进而找到第 i 次做与第 i-1 次做之间的关系；**
  - **确认哪一次做时不需要“再重复”（通常是n=1的时候），而且能够给前一次返回结果；**

递推数列

- 一个数列从某一项起，它的任何一项都可以用它前面的若干项来确定，这样的数列称为递推数列：
  - **1!, 2!, 3!, … n!**
    - **fact(n) = n * fact(n-1)（通项公式）；**
    - **fact(1) =1 （边界条件）**
  - **1, 1, 2, 3, 5, 8, 13, 21, …**
    - **fab(n)= fab(n-1) + fab(n-2)（通项公式）；**
    - **fab(1)=1, fab(2)=1;（边界条件）**

递推与递归

- 递推问题
  - 后续的运算依赖于已知的条件；当前的运算是下一步运算的基础；
- 解法1：从已知的初始条件出发，逐次去求所需要的值。

  如求n!

  初始条件 **fact(1) = 1**

  fact(2) = 2*fact(1) = 2

  fact(3) = 3*fact(2) = 6

  …… ……

递推与递归

- 递推问题
  - 后续的运算依赖于已知的条件；当前的运算是下一步运算的基础；
- 解法2：递归算法
  - 出发点不放在初始条件上，而放在求解的目标上，从所求的未知项出发逐次调用本身的求解过程，直到递归的边界（即初始条件）。

求 n!

- 问题：已知n，求 n！

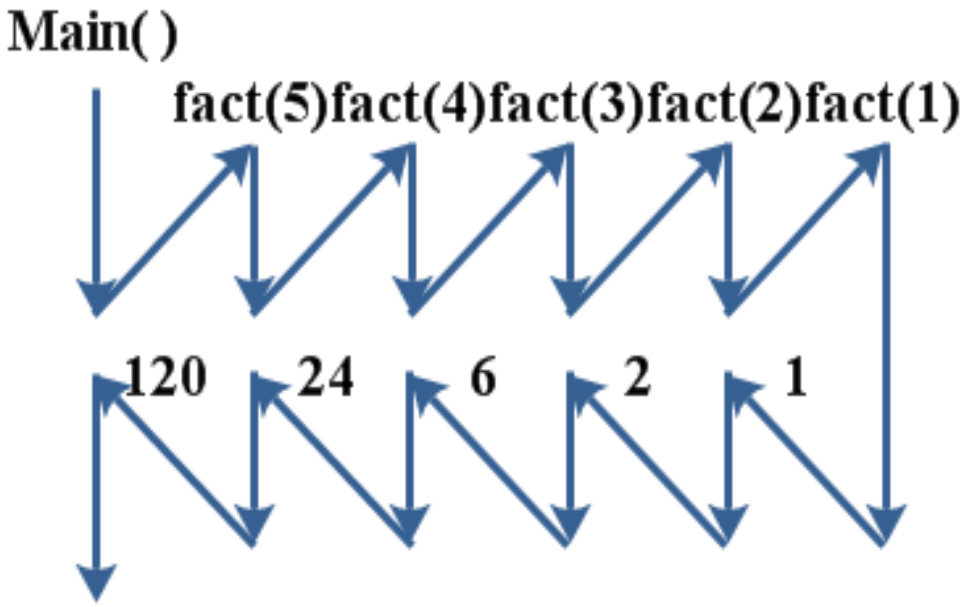
  假设计算阶乘的任务由一个函数fact来做

  fact(n)　　等于　　fact(n-1)*n
  fact(n-1)　等于　　fact(n-2)*(n-1)
  fact(3)　　等于　　fact(2)*3;
  fact(2)　　等于　　fact(1)*2
  fact(1)　　等于　　1;

- 可知
  - fact(n)的值等于fact(n-1)*n;
  - fact(1) = 1;

求 n!

```
#include<iostream>
using namespace std;
int Factorial(int n){
    int temp;
    if(n == 1)
        return 1;
    else{
        temp = Factorial( n-1 );
        temp = n*temp;
        return temp;
    }
}
int main(){
        cout<<Factorial(5)<<endl;
        return 0;
}
```



北京大学

求 n!

栈与递归

```
(1)  int Factorial(int n);
(2)  {  int temp;
(3)     if(n==0)
(4)         return 1;
(5)     else{
(6)         temp=Factorial(n-1);
(7)         temp=n*temp;
(8)         return temp;
(9)         }
(10)  }
```

本演示只给出函数Factorial(n)执行过程中，系统栈的变化。为方便说明，本演示只对参数n和返回地址进行入栈和出栈操作，局部变量 temp不做入栈和出栈操作。

知道了

方法总结之二

- 如何利用递归解决问题？
  - 关注每一次重复/迭代的执行历程

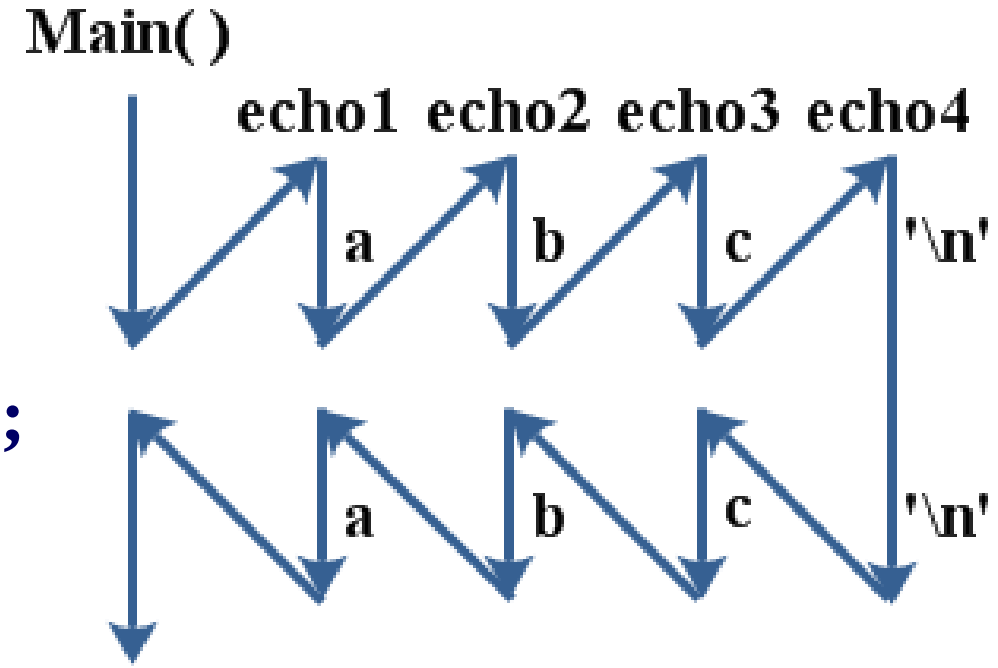
程序分析

```
#include<iostream>
using namespace std;
int echo()
{  char c;
   c = cin.get();
   if (c != '\n')   echo( );
   cout<<c;
   return 0;
}
void main()
{  echo( ); }
```

试分析这个程序执行了什么功能。

程序分析

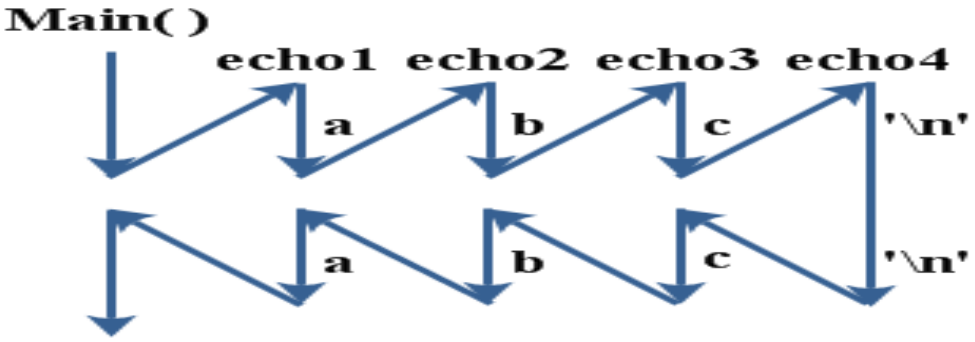
```
#include<iostream>
using namespace std;
void echo()
{  char c;
   c = cin.get();
   if (c != '\n')   echo( );
   cout<<c;
}
void main()
{  echo( ); }
```



方法总结之二

- 如何利用递归解决问题？
  - 关注每一次重复/迭代的执行历程
    - 在递归调用之前执行的动作 **顺序输出**
    - 递归调用之后的动作 **逆序输出**

注：这里的“序”指**调用**顺序



分析一下这个程序作了什么?

```
#include<iostream>
using namespace std;
void function(int num)
{
   cout<<num%10<<" ";
   if ((num/10)!= 0)
      function (num/10);
}
```

```
void main()
{
   int num;
   cin>>num;
   function (num);
}
```

分析一下这个程序作了什么?

```
#include<iostream>
using namespace std;
void function(int num)
{
    if ((num/10)!= 0)
      function (num/10);
    cout<<num%10<<" ";
}
```

```
void main()
{
   int num;
   cin>>num;
   function (num);
}
```

进制转换

- 问题
  - 编写一个程序，对任意输入的十进制正整数给出其二进制表示，并打印输出。

- 例如：
  - 输入：**97**
  - 输出：**1100001**

进制转换

- 将123转换成等值的二进制数：

除以2的商（取整）	余数
123/2 = 61	1
61/2 = 30	1
30/2 = 15	0
15/2 = 7	1
7/2 = 3	1
3/2 = 1	1
1/2 = 0	1

- 自下而上收集余数：1111011

进制转换

```
#include<iostream>
using namespace std;
void convert (int x)
{
   if((x/2)!=0)
   {
     convert (x/2);
     cout<<x%2;
   }
   else
     cout<<x;
}
```

```
void main()
{
   int x;
   cin>>x;
   convert (x);
}
```

方法总结之三

- 如何利用递归解决问题？
  - 如何 找到 第 i 次做与第 i-1 次做之间的关系？

放苹果

- 题目描述
  - 把M个同样的苹果放在N个同样的盘子里，允许有的盘子空着不放，问共有多少种不同的分法？
  - 注意：5，1，1和1，5，1 是同一种分法
  - 输入：7　3
  - 输出：8

放苹果

- 如果n>m：必定有n-m 个盘子永远空着，去掉它们对摆放苹果方法数目不产生影响；即
  - if(n>m) f(m,n) = f(m,m)
- 当n <= m 时，不同的放法可以分成两类：
  - 至少一个盘子空着：
    - 该情况相当于f(m , n) = f(m , n-1)
  - 所有盘子都有苹果：
    - 若从每个盘子中拿掉一个苹果，不影响放法的数目，即f(m , n) = f(m-n , n)

北京大学

放苹果

- 极限情况：
  - n 会逐渐减少，终会当n= 1 时：
    - 所有苹果都必须放在一个盘子里，返回 1；
  - m 会逐渐减少，因为n>m 时，我们会 return f(m , m) 代替f(n, m)，最终当 m = 0时：
    - 没有苹果可放，返回 1；

方法总结之三

- 如何利用递归解决问题？
  - 如何 找到 第 i 次做与第 i-1 次做之间的关系？
    - 找到 两次之间的逻辑关系
    - 找到 体现两次之间不同的变化因素
    - 将 变化因素 体现为 函数参数

逆波兰表达式

- 题目描述
  - 逆波兰表达式是一种把运算符前置的算术表达式:
    - 如表达式2 + 3的逆波兰表示法为+ 2 3。
    - 如(2 + 3) * 4的逆波兰表示法为* + 2 3 4。
  - 编写程序求解任一仅包含+ - * /四个运算符的逆波兰表达式的值。
- 输入：* + 11.0 12.0 + 24.0 35.0
- 输出：1357.0

好好想想,有没有问题?

谢　谢　！

北京大学