中国人民大学
RENMIN UNIVERSITY OF CHINA

第6讲 函数

余力

buaayuli@ruc.edu.cn

内容提要

1 函数的基本用法

2 数组名作为函数参数

3 全局变量与局部变量

4 函数嵌套调用

5 函数应用实例

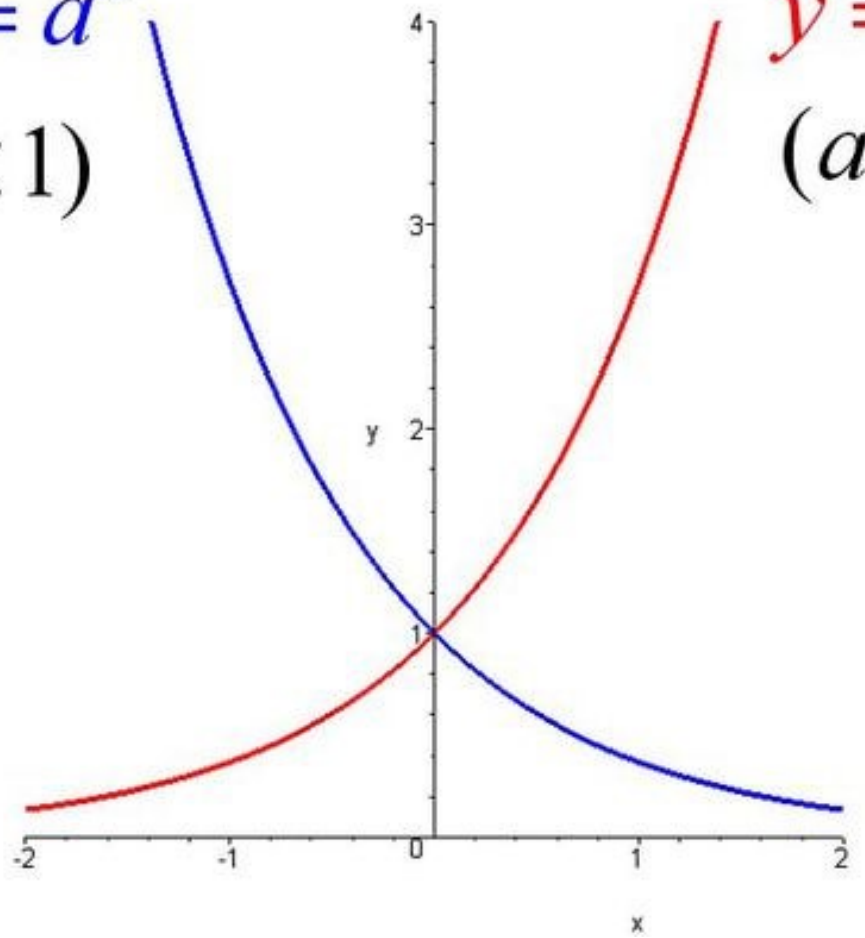
中国人民大学
RENMIN UNIVERSITY OF CHINA

01. 函数的基本使用

你熟悉的函数

$y = a^x$

$(0 < a < 1)$



$y = a^x$

$(a > 1)$

为什么要函数？回文素数和

```
int su(int x){//判断素数 1-yes 0-no
    for(int i=2;i*i<=x;i++)
        if(x%i==0)
            return 0;
    return 1;
}
```

```
int digitsum(int x){
    int sum=0;
    while(x!=0){
        sum+=x%10;
        x/=10;}
    return sum;
}
```

```
int hui(int x){//判断回文 1-yes 0-no
    char a[20];
    int num=0;
    while(x!=0){
        a[num++]=x%10;
        x/=10;
        num++;}
    for(int i=1;2*i<=num;i++)
        if(a[i]!=a[num--i+1])
            return 0;
    return 1;
}
```

```
int main(){
    cin>>m>>n;
    for(int i=m;i<=n;i++)
        if(hui(i)&&su(i)){
            int t=digitsum(i);
            if(t>summax){
                summax=t;
                end=i;
            }
        }
    cout<<end<<"-"<<summax;
    return 0;
}
```

定义函数

```
类型名  函数名()
 {
     函数体
 }
```

```
类型名  函数名(void)
 {
     函数体
 }
```

包括声明部分和
语句部分

形式参数和实际参数

- 形式参数和实际参数
  - 在调用有参函数时，主调函数和被调用函数之间有数据传递关系
  - 定义函数时函数名后面的变量名称为“形式参数”（简称“形参”）
  - 主调函数中调用一个函数时，函数名后面参数称为“实际参数”（简称“实参”）
  - 实际参数可以是常量、变量或表达式

形式参数与实在参数

```
#include <stdio.h>
#include <math.h>
int checkPrime(int af);

int main()
{
    int a;
    printf("请输入一个正整数 a\n");
    scanf( "%d", &a);
    if (checkPrime(a)) checkPrime函数
        printf("%d是质数\n", a);
    else
        printf("%d不是质数\n", a);
}
```

函数调用时的数据传递

例 输入两个整数，要求输出其中值较大者。要求用函数来找到大数。

- 解题思路：

  (1)函数名应是见名知意，今定名为max

  (2) 由于给定的两个数是整数，返回主调函数的值（即较大数）应该是整型

  (3)max函数应当有两个参数，以便从主函数接收两个整数，因此参数的类型应当是整型

函数调用时的数据传递

在max函数上面，再编写主函数

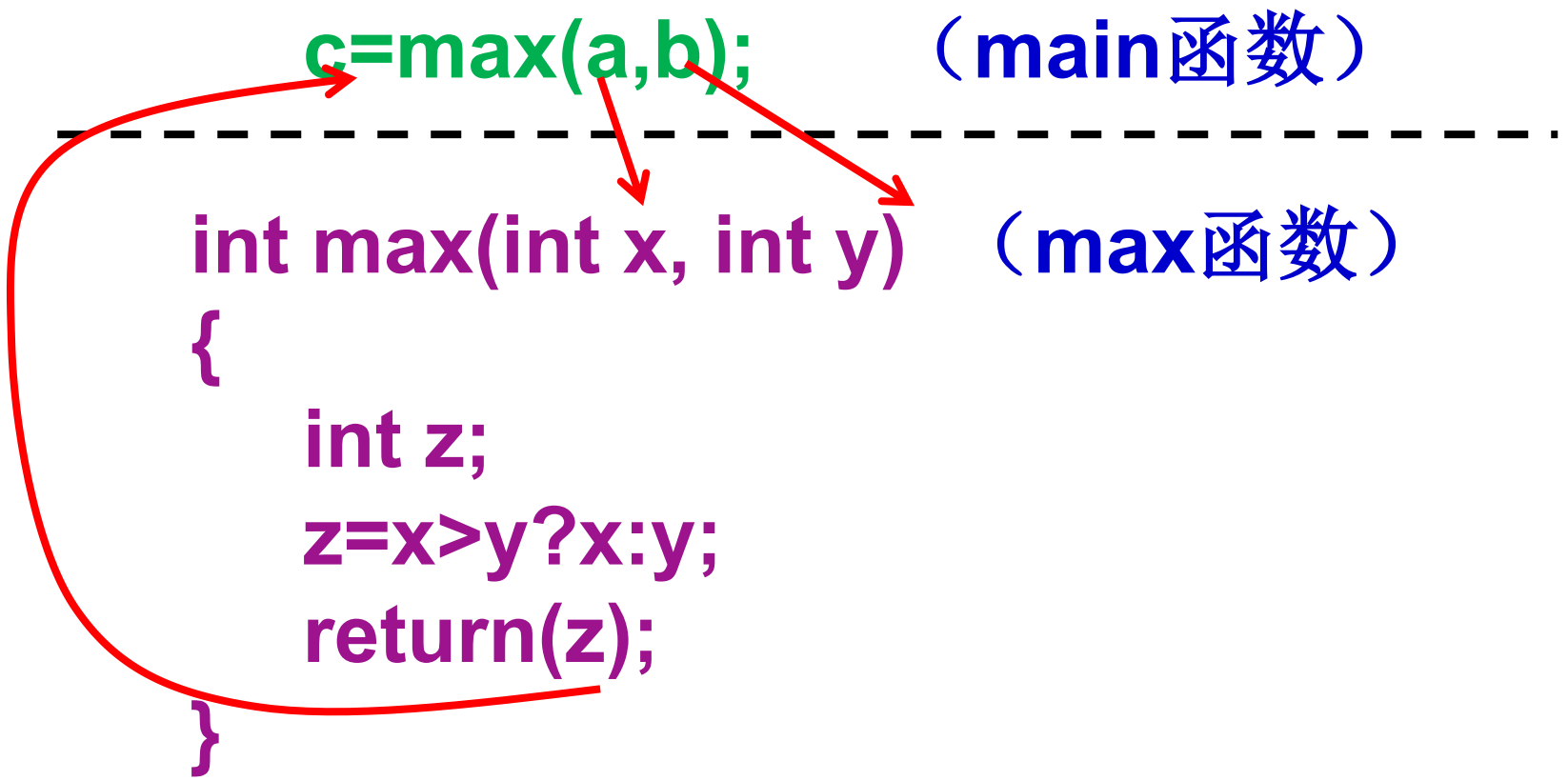
```
#include <stdio.h>
int main()
{ int max(int x,int y);   int a,b,c;
   printf("two integer numbers: ");
   scanf("%d,%d",&a,&b);
   c=max(a,b);
   printf("max is %d\n",c);
}
```

```
int max(int x,int y)
{
x++;
     return(x);
}
```

实参可以是常量、变量或表达式

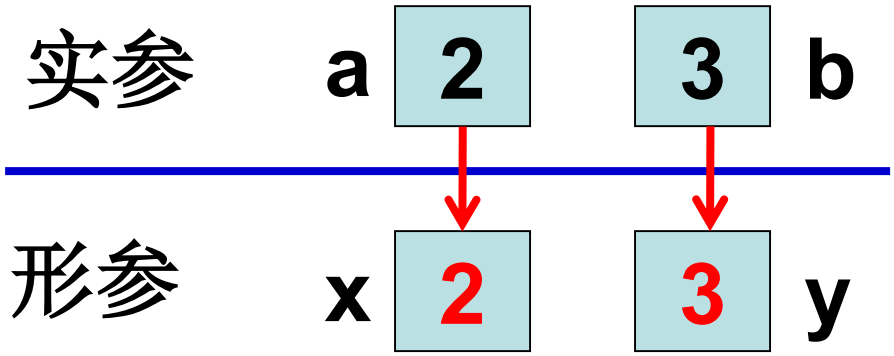
```
two integer numbers:12,-34
max is 12
```

函数调用时的数据传递



函数调用的过程

- 调用结束，形参单元被释放
- 实参单元仍保留并维持原值，没有改变
- 如果在执行一个被调用函数时，形参的值发生改变，不会改变主调函数的实参的值



形式参数与实在参数

- 形式参数特点：
  - 定义函数时放在函数名后括号中的参数；
  - 未被调用不占内存单元；
  - 被调用后系统为其分配内存单元；
  - 调用结束释放内存单元；
  - 作用域限定在子函数内，属于局部变量
- 实在参数：
  - 具有确定值的表达式
  - 函数调用时将实在参数赋值给形参变量

考虑下面函数调用例子

```
#include<stdio.h>
void  swap (int x1, int x2);
int main () {
  int a[2];
    a[0] = 10;
    a[1] = 100;
    printf ("%d, %d\n", a[0], a[1]);
    swap (a[0],a[1]);
    printf ("%d, %d\n", a[0], a[1]);
    return 0;
}
      void swap (int x1, int x2) {
        int tmp = x1;
        x1 = x2;
        x2 = tmp;
      }
```

什么运行结果？

函数的返回值

➢通常，希望通过函数调用使主调函数能得到一个确定的值，这就是函数值(函数的返回值)

(1) 函数的返回值是通过函数中的return语句获得的。

◆一个函数中可以有一个以上的return语句，执行到哪一个return语句，哪一个就起作用

◆return语句后面的括号可以不要

➢通常，希望通过函数调用使主调函数能得到一个确定的值，这就是函数值(函数的返回值)

(2) 函数值的类型。应当在定义函数时指定函数值的类型

max函数.cpp

```
#include <stdio.h>
int main()
 { int max(float x,float y);
    float a,b;  int c;
    scanf("%f,%f,",&a,&b);
    c=max(a,b);
    printf("max is %d\n",c);
    return 0;
 }
int max(float x,float y)
 {  float z;
     z=x>y?x:y;
     return( z ) ;
}
```

变为2

1.5 2.6

2.6 → 2

```
1.5,2.6
max is 2
```

将在max函数中定义的变量z改为float型

函数的声明

- 在一个函数中调用另一个函数需要具备如下条件：

  (1) 被调用函数必须是已经定义的函数（是库函数或用户自己定义的函数）

  (2) 如果使用库函数，应该在本文件开头加相应的#include指令

  (3) 如果使用自己定义的函数，而该函数的位置在调用它的函数后面，应该声明

例 输入两个实数，用一个函数求出它们之和。

- 解题思路：用add函数实现。首先要定义add函数，它为float型，它应有两个参数，也应为float型。特别要注意的是：要对add函数进行声明。

```
#include <stdio.h>
int main()
{ float add(float x, float y);
   float a,b,c;
   printf("Please enter a and b:");
   scanf("%f,%f",&a,&b);
   c=add(a,b);
   printf("sum is %f\n",c);
   return 0;
}
```

对add函数声明

求两个实数之和，
函数值也是实型

```
float add(float x,float y)
{ float z;
   z=x+y;
   return(z);
}
```

Add函数.cpp

2. 数组名作参数

数组名作为参数

```
#include<stdio.h>
void swap1 (int a[], int i, int j) ;
int main () {
  int a[2];
  a[0] = 10;
  a[1] = 100;
  printf  ("%d, %d\n", a[0], a[1]);
  swap1 (a, 0, 1);
  printf  ("%d, %d\n", a[0], a[1]);
  return  0;
}

   void swap1 (int x[], int i, int j) {
     int tmp = x[i];
     x[i] = x[j];
     x[j] = tmp;
   }
```

什么运行结果？

数组名作函数参数

例有一个一维数组score，内放10个学生成绩，求平均成绩。

- 解题思路：
  - 用函数average求平均成绩，用数组名作为函数实参，形参也用数组名
  - 在average函数中引用各数组元素，求平均成绩并返回main函数

```
#include <stdio.h>
Int averge;
int main()
{ float average(float array[10]);
   float score[10],aver;  int i;
   printf("input 10 scores:\n");
   for(i=0;i<10;i++)
      scanf("%f",&score[i]);
   printf("\n");
aver=average(score);
   printf("%5.2f\n",aver);
   return 0;
}
```

定义实参数组

定义形参数组

```
float average(float array[])
{  int i;
   float aver,sum=array[0];
   for(i=1;i<10;i++)
      sum=sum+array[i];
   aver=sum/10;
   return(aver);
}
```

相当于score[0]

相当于score[i]

```
input 10 scores:
100 56 78 98 67.5 99 54 88.5 76 58

77.50
```

- 例 有两个班级，分别有35名和30名学生，调用一个average函数，分别求这两个班的学生的平均成绩。
- 解题思路：
  - 需要解决怎样用同一个函数求两个不同长度的数组的平均值的问题
  - 定义average函数时不指定数组的长度，在形参表中增加一个整型变量i
  - 从主函数把数组实际长度从实参传递给形参i
  - 这个i用来在average函数中控制循环的次数
  - 为简化，设两个班的学生数分别为5和10

```
#include <stdio.h>
int main()
{ float average(float array[ ],int n);
   float score1[5]={98.5,97,91.5,60,55};
   float score2[10]={67.5,89.5,99,69.5,
                     77,89.5,76.5,54,60,99.5};
   printf( "%6.2f\n" ,average(score1,5));
   printf( "%6.2f\n" ,average(score2+3,5));
   return 0;
}
```

调用形式为**average(score1,5)**时

```
float average(float array[ ],int n)
 { int i;
    float aver,sum=array[0];
    for(i=1;i<n;i++)
       sum=sum+array[i];
    aver=sum/n;
    return(aver);
}
```

相当于**score1[0]**

相当于**score1[i]**

调用形式为**average(score2,10)**时

```
float average(float array[ ],int n)
 { int i;
    float aver,sum=array[0];
    for(i=1;i<n;i++)
       sum=sum+array[i];
    aver=sum/n;
    return(aver);
}
```

相当于10

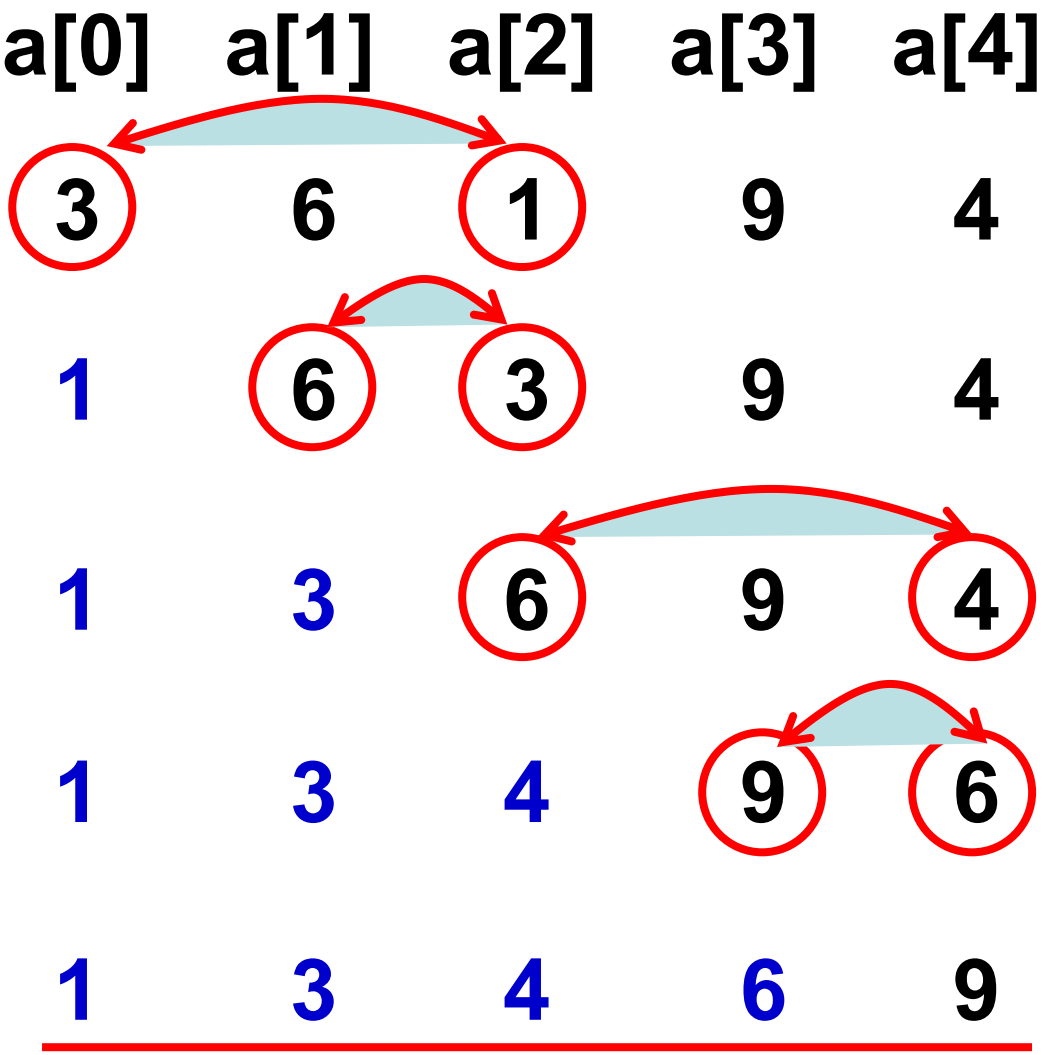
相当于score2[0]

相当于score2[i]

```
80.40
78.20
```

例7.12用选择法对数组中10个整数按由小到大排序。

- 解题思路：
  - 所谓选择法就是先将10个数中最小的数与a[0]对换；再将a[1]到a[9]中最小的数与a[1]对换……每比较一轮，找出一个未经排序的数中最小的一个
  - 共比较9轮



小到大排序

```
#include <stdio.h>
int main()
{ void sort(int array[],int n);
   int a[10],i;
   printf("enter array:\n");
   for(i=0;i<10;i++)  scanf("%d",&a[i]);
   sort(a+6,5);
   printf("The sorted array:\n");
   for(i=0;i<10;i++)   printf("%d ",a[i]);
   printf("\n");
   return 0;
}
```

```
void sort(int array[],int n)
 { int i,j,k,t;
    for(i=0;i<n-1;i++)
    { k=i;
      for(j=i+1;j<n;j++)
        if(array[j]<array[k])    k=j;
      t=array[k];
      array[k]=array[i];
      array[i]=t;
    }
 }
```

在sort[i]~sort[9]中，
最小数与sort[i]对换

```
enter array:
45 2 9 0 -3 54 12 5 66 33
The sorted array:
-3 0 2 5 9 12 33 45 54 66
```

中国人民大学
RENMIN UNIVERSITY OF CHINA

3. 全局与局部变量

局部变量

- 定义变量可能有三种情况：
  - 在函数的开头定义
  - 在函数内的复合语句内定义
  - 在函数的外部定义

- 在一个函数内部定义的变量只在本函数范围内有效
- 在复合语句内定义的变量只在本复合语句范围内有效
- 在函数内部或复合语句内部定义的变量称为“局部变量”

```
float f1( int a)
{ int b,c;
    ......
}
```

a、b、c仅在此函数内有效

```
char f2(int x,int y)
{ int i,j;
    ......
}
```

x、y、i、j仅在此函数内有效

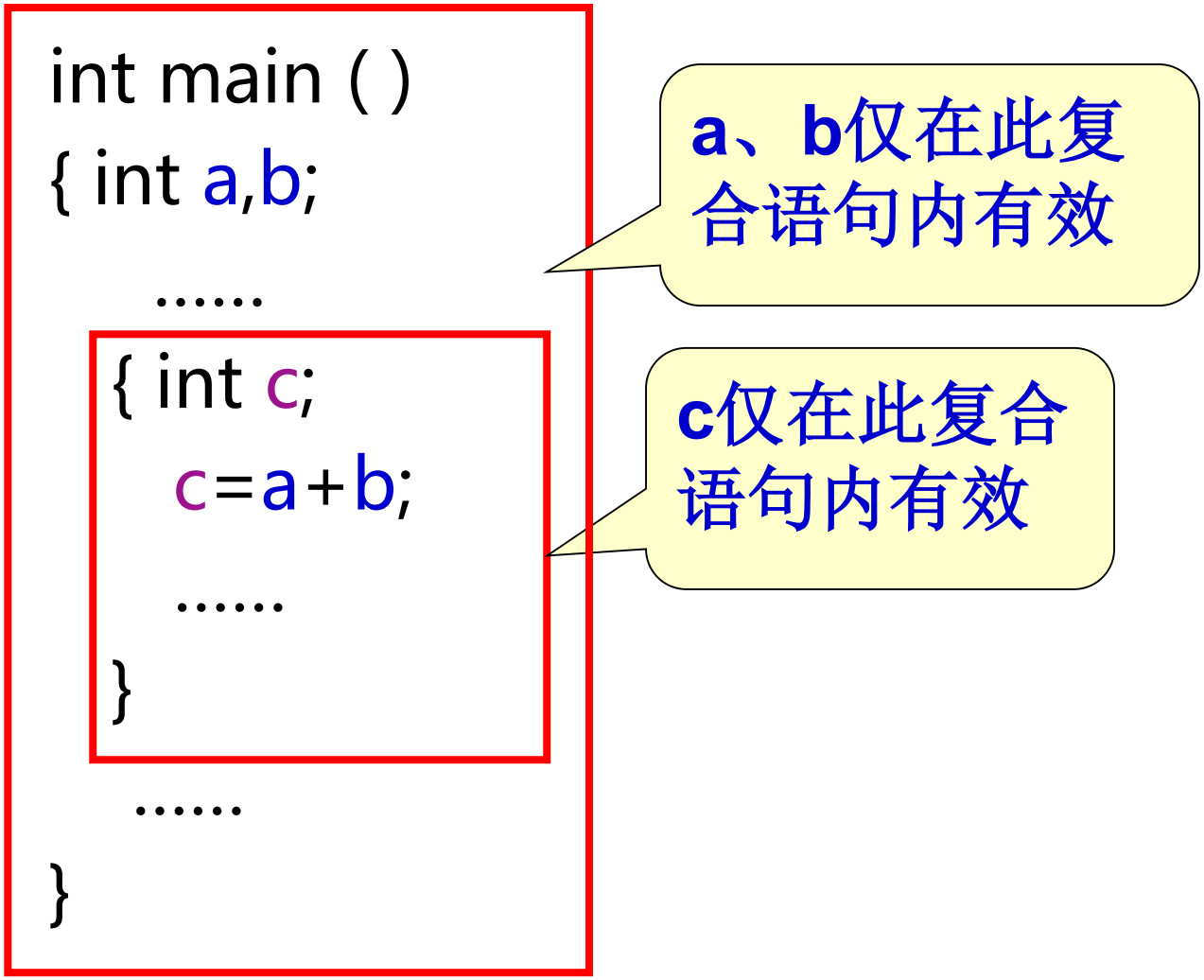
```
int main( )
{ int m,n;
    ......
    return 0;
}
```

m、n仅在此函数内有效

```
float f1( int a)
{ int b,c;
   ......
}
char f2(int x,int y)
{ int i,j;
   ......
}
int main( )
{ int a,b;
   ......
   return 0;
}
```

类似于不同班同名学生

a、b也仅在此函数内有效


int main ( )
{ int a,b;
.....
{ int c;
c=a+b;
.....
}
.....
}
a、b仅在此复合语句内有效
c仅在此复合语句内有效

全局变量

- 在函数内定义的变量是局部变量，而在函数之外定义的变量称为外部变量
- 外部变量是全局变量(也称全程变量)
- 全局变量可以为本文件中其他函数所共用
- 有效范围为从定义变量的位置开始到本源文件结束

```
int p=1,q=5
float f1(int a)
{  int b,c;   ……  }
char c1,c2;
char f2 (int x, int y)
{  int i,j;   ……  }
int main ( )
{  int m,n;
    ……
    return 0;
}
```

p、q、c1、c2为全局变量

```
int p=1,q=5
float f1(int a)
{  int b,c;   ……  }
char c1,c2;
char f2 (int x, int y)
{  int i,j;   ……  }
int main ( )
{  int m,n;
    ……
    return 0;
}
```

p、q的有效范围

c1、c2的有效范围

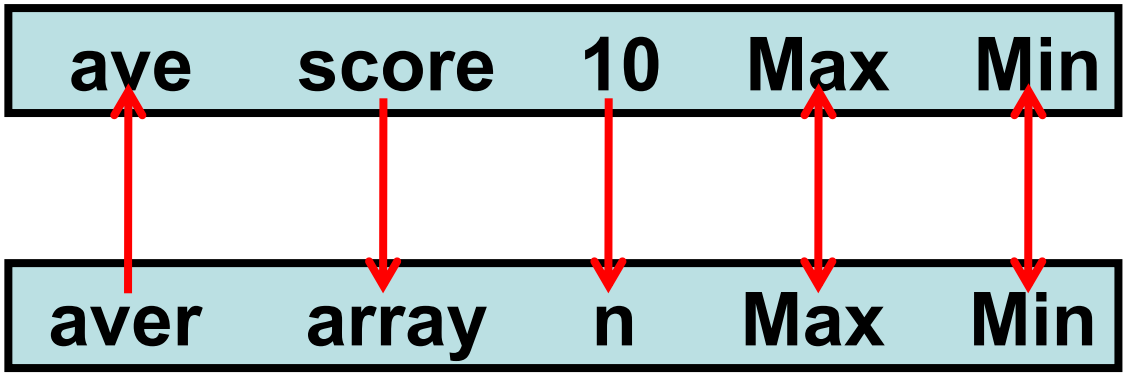
例 有一个一维数组，内放10个学生成绩，写一个函数，当主函数调用此函数后，能求出平均分、最高分和最低分。

- 解题思路：调用一个函数可以得到一个函数返回值，现在希望通过函数调用能得到3个结果。可以利用全局变量来达到此目的。

```
#include <stdio.h>
float Max=0,Min=0;
int main()
{ float average(float array[ ],int n);
   float ave,score[10];  int i;
   printf("Please enter 10 scores:\n");
   for(i=0;i<10;i++)
       scanf("%f",&score[i]);
   ave=average(score,10);
   printf("max=%6.2f\nmin=%6.2f\n
         average=%6.2f\n",Max,Min,ave);
   return 0;
 }
```

```
float average(float array[ ],int n)
{ int i;  float aver,sum=array[0];
   Max=Min=array[0];
   for(i=1;i<n;i++)
   { if(array[i]>Max) Max=array[i];
      else if(array[i]<Min) Min=array[i];
      sum=sum+array[i];
    }
    aver=sum/n;
    return(aver);
 }
```

```
Please enter 10 scores:
89 95 87.5 100 67.5 97 59 84 73 90
max=100.00
min= 59.00
average= 84.20
```



main
函数

average
函数

建议不在必要时不要使用全局变量

若外部变量与局部变量同名

```
#include <stdio.h>
int a=3,b=5;
int main()
{ int max(int a,int b);
   int a=8;
   printf( “max=%d\n” ,max(a,b));
   return 0;
}
int max(int a,int b)
{ int c;
   c=a>b?a:b;
   return(c);
}
```

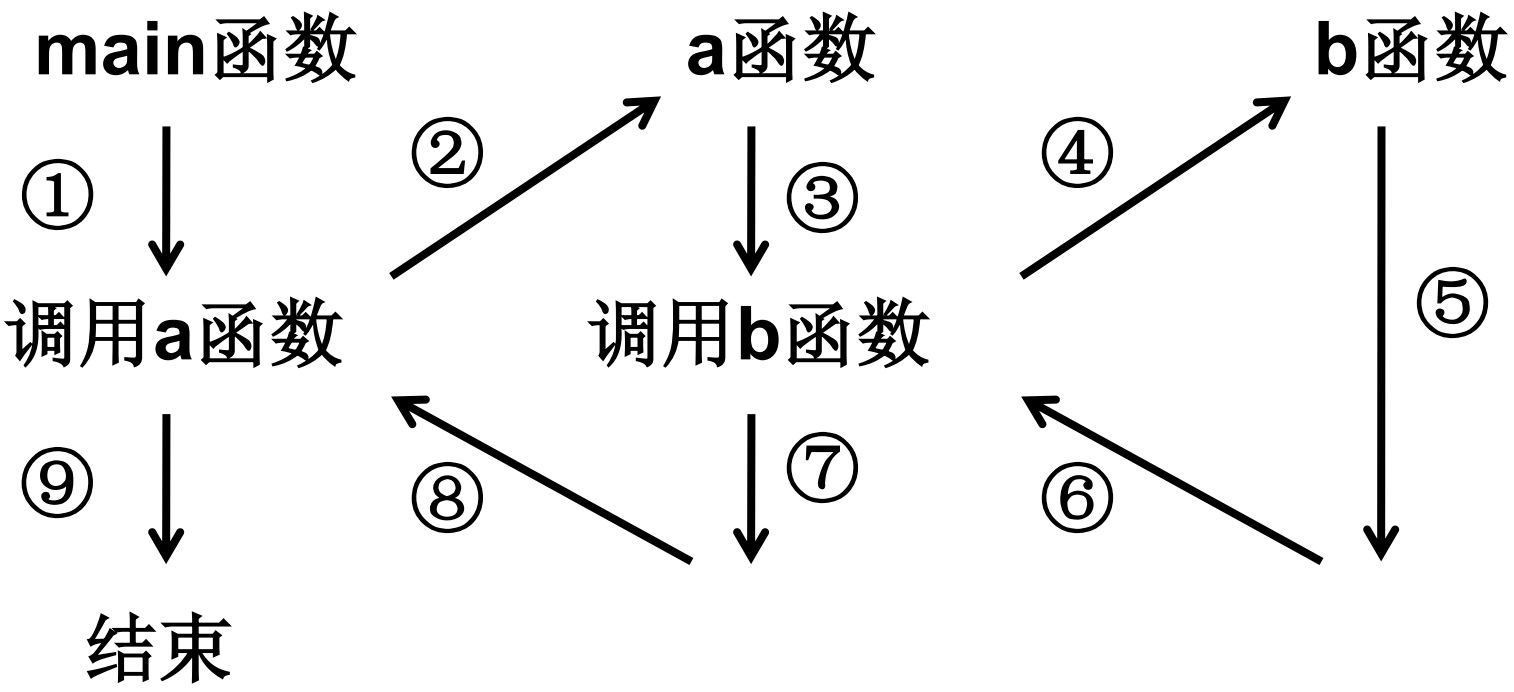
b为全部变量

a为局部变量，仅在此函数内有效

中国人民大学
RENMIN UNIVERSITY OF CHINA

4. 嵌套调用

函数的嵌套调用



函数的嵌套调用

例7.5 输入4个整数，找出其中最大的数。用函数的嵌套调用来处理

- 解题思路：
  - main中调用max4函数，找4个数中最大者
  - max4中再调用max2，找两个数中的大者
  - max4中多次调用max2，可找4个数中的大者，然后把它作为函数值返回main函数
  - main函数中输出结果

主函数

对max4 函数声明

```
#include <stdio.h>
int main()
{ int max4(int a,int b,int c,int d);
   int a,b,c,d,max;
   printf(  “4 interger numbers:");
   scanf("%d%d%d%d",&a,&b,&c,&d);
   max=max4(a,b,c,d);
   printf("max=%d \n",max);
   return 0;
}
```

主函数

```
#include <stdio.h>
int main()
{ int max4(int a,int b,int c,int d);
   int a,b,c,d,max;
   printf(  “4 interger numbers:");
   scanf("%d%d%d%d",&a,&b,&c,&d);
   max= max4(a,b,c,d);
   printf("max=%d \n",max);
   return 0;
}
```

输入4个整数

max4函数

```
int max4(int a,int b,int c,int d)
{  int max2(int a,int b);
    int m;
    m=max2(a,b);
    m=max2(m,c);
    m=max2(m,d);
    return(m);
}
```

对max2 函数声明

max4函数

```
int max4(int a,int b,int c,int d)
{  int max2(int a,int b);
    int m;
    m=max2(a,b);
    m=max2(m,c);
    m=max2(m,d);
    return(m);
}
```

return(a>b?a:b);

max2函数

```
int max2(int a,int b)
{  if(a>=b)
       return a;
   else
       return b;
}
```

max4函数

```
int max4(int a,int b,int c,int d)
{  int max2(int a,int b);
    int m;
    m=max2(a,b);
    m=max2(m,c);
    m=max2(m,d);
    return(m);
}
```

```
int max2(int a,int b) {
return(a>b?a:b);   }
```

max4函数

```
int max4(int a,int b,int c,int d)
{  int max2(int a,int b);
   int m;
   m=max2(a,b);
   m=max2(m,c);
   m=max2(m,d);
   return(m);
}
```

m=max2(max2(a,b),c);

```
int max2(int a,int b) {
return(a>b?a:b);   }
```

max4函数

```
int max4(int a,int b,int c,int d)
{  int max2(int a,
    int m;
    m=max2(a,b);
    m=max2(m,c);
    m=max2(m,d);
    return(m);
}
```

m=max2(max2(max2(a,b),c),d);

```
int max2(int a,int b) {
return(a>b?a:b);   }
```

max4函数

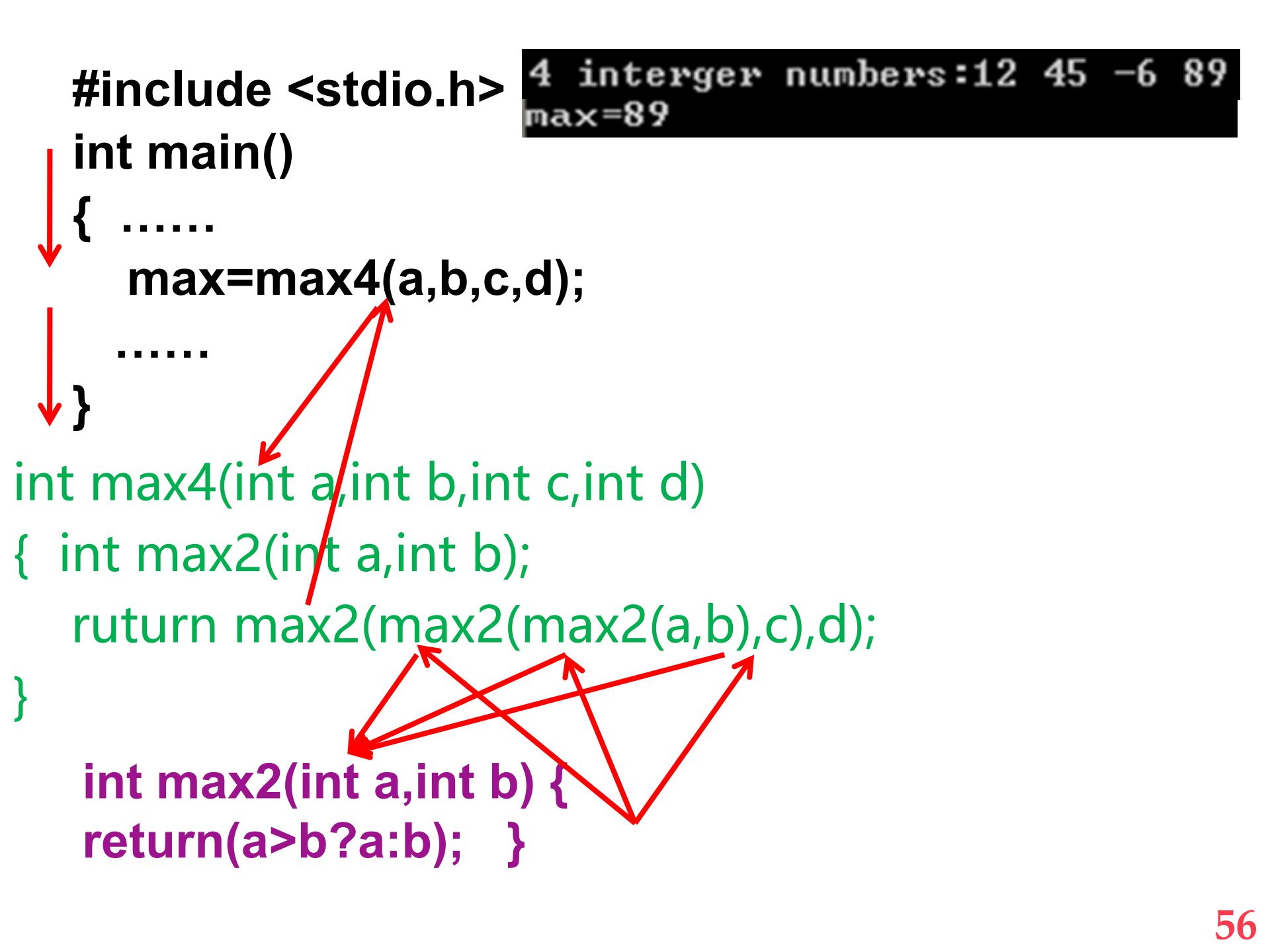
```
int max4(in
{  int max2(int a,in
    int m;
    m=max2(a,b);
    m=max2(m,c);
    m=max2(m,d);
    return(m);
}
```

ruturn max2(max2(max2(a,b),c),d);

```
int max2(int a,int b) {
return(a>b?a:b);   }
```

```
4 interger numbers:12 45 -6 89
max=89
```

```
#include <stdio.h>
int main()
{  ……
    max=max4(a,b,c,d);
   ……
}
int max4(int a,int b,int c,int d)
{  int max2(int a,int b);
   ruturn max2(max2(max2(a,b),c),d);
}
    int max2(int a,int b) {
    return(a>b?a:b);   }
```



4. 模块化编程举例

编程实例

- 问题：编程求解

$$\sum_{x=1}^{n} x^k$$

- 思路：假定考虑n=6, k=4

1、该式可分解为 $1^4+2^4+3^4+4^4+5^4+6^4$

2、定义一个函数

$$\text{power}(i,l)=i^l$$

让 $l=4,\quad i=1,2,\cdots,6$

这个函数可以表示

$$1^4,2^4,\cdots,6^4$$

```
#include <stdio.h>   // 预编译命令
const int n = 6;                    // 定义符号常量 n 为 6
const int k = 4;                    // 定义符号常量 k 为 4

int SOP(int m, int l ); // 声明函数SOP
int power(int p, int q );           // 声明函数power

int main()                                          // 主函数
{   printf("Sum of %d the powers of integers from 1 to %d is %d\n",
              k, n, SOP(n, k) );// 调用SOP函数
    return 0;
}
```

```
int SOP(int m, int l ) // 整型自定义函数, m, l 为形参
{
        int i, sum;                 // 定义整型变量i, sum
        sum=0;                      // 初始化累加器
        for(i=1; i<=m; i=i+1 )      // 计数循环 i
                sum = sum + power( i, l ); // 累加
        return sum;                 // 返回sum
}
```

```
// 以下函数是被函数SOP(n, k)调用的函数
// 功能：计算p的q次幂

int power(int p, int q)          // 整型自定义函数
{                                // 自定义函数体开始
    int i, product;              // 整型变量
    product=1;                   // 初始化累乘器
    for(i=1; i<=q; i=i+1)        // 计数循环 i
        product=product*p;       // 累乘
    return product;              // 累乘值返回给power
}                                // 自定义函数体结束
```

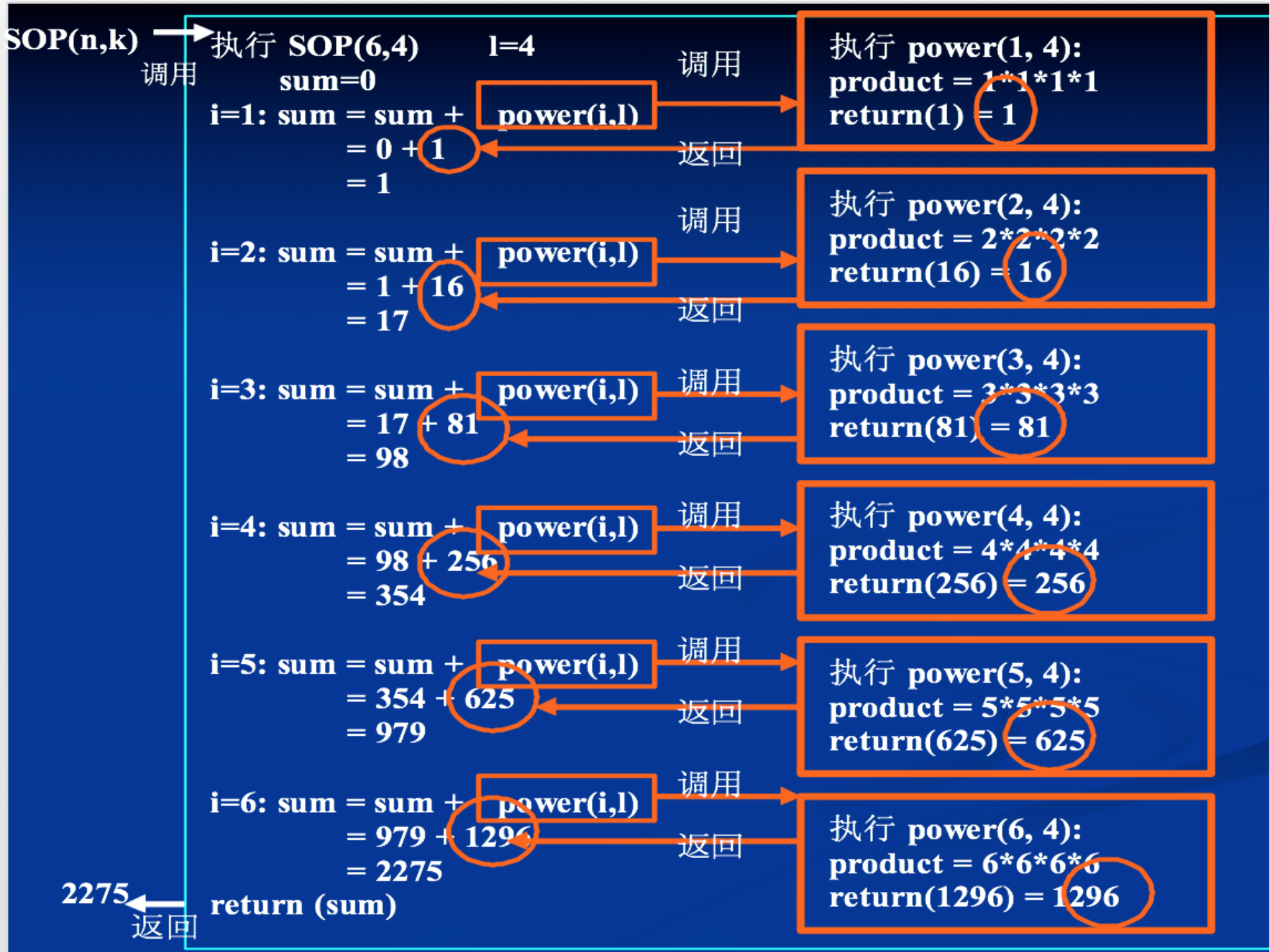
在定义函数的时候，必须指定形参变量的类型，

如下所示:

```
int power(int p, int
    n)
{
    ..... // 函数体
}
```

实参是一个具有确定值的表达式。函数在调用时，将实在参数赋给形式参数。

主函数调用SOP(n, k)，这时，n, k为实参，n的值为6，k的值为4。在被调用函数定义中，int SOP(m, *l*) 中的 m, *l* 为形参，在SOP被调用时，系统给 m, *l* 这两个形式参数分配了内存单元。之后，n 的值 6 赋给 m; k 的值 4 赋给 *l*。


SOP(n,k)
调用
执行 SOP(6,4)
l=4
sum=0
i=1: sum = sum + power(i,l)
= 0 + 1
= 1
i=2: sum = sum + power(i,l)
= 1 + 16
= 17
i=3: sum = sum + power(i,l)
= 17 + 81
= 98
i=4: sum = sum + power(i,l)
= 98 + 256
= 354
i=5: sum = sum + power(i,l)
= 354 + 625
= 979
i=6: sum = sum + power(i,l)
= 979 + 1296
= 2275
return (sum)
2275
返回
调用
返回
执行 power(1, 4):
product = 1*1*1*1
return(1) = 1
执行 power(2, 4):
product = 2*2*2*2
return(16) = 16
执行 power(3, 4):
product = 3*3*3*3
return(81) = 81
执行 power(4, 4):
product = 4*4*4*4
return(256) = 256
执行 power(5, 4):
product = 5*5*5*5
return(625) = 625
执行 power(6, 4):
product = 6*6*6*6
return(1296) = 1296

#304 整数计数

编写一个程序计算整数区间[a, b]内，其个位数是 n，且能被 k 整除的 m 位正整数共有多少个。

【输入格式】

输入只有一行，输入 5 个整数 a、b、n、k、m，空格分隔，其中：1≤a≤b≤1,000,000，且 0≤n, k, m≤9。

【输出格式】

输出一行，为符合要求的整数个数。

【样例输入 1】

1019 1 2 2

【样例输出 1】

0

【样例输入 2】

1050 4 2 2

【样例输出 2】

4

```
int main()
{
    int a,b,c,d,e,s=0,i,f;
    scanf("%d%d%d%d%d",&a,&b,&c,&d,&e);
    f=(pow(10,e-1));
    for(i=a;i<=b;i++)
        if(i%10==c)
            if(i%d==0)
                if(i/(10*f)<1&&i/f>=1)  s++;
    printf("%d",s);
    return 0;
}
```

#同构数

```
int main() {
    int from, to, n, m, weishu, sum=0, i;
    scanf("%d%d", &from, &to);
    for (n = from; n <= to; ++n) {
        i = n;
        weishu = 0;
        do {i = i / 10;
            weishu++;
        } while (i != 0);
        m = n * n;
        for (i = 0; i < weishu; i++)
            m = m / 10;
        for (i = 0; i < weishu; i++)
            m = m * 10;
        if (n == n * n - m)  sum += n;
    }
    printf("%d", sum);
    return 0;
}
```

```
int ismp(int x){
    int sq = x*x;  int t= x;  int base= 1;
    while(t!=0){
        t /= 10;  base *= 10; }
    if(sq%base == x) return 1;
    else return 0;
}

int main(){
    int a,b;
    scanf("%d%d",&a,&b);
    int sum=0;
    for(int i=a; i<=b; i++)
        if(ismp(i)) sum += i;
    printf("%d",sum);
    return 0;
}
```

#291 大整数加减法

```
void parseStrToIntArr(char str[],int a[],int size)
{  for(int i=0;i<size;i++)
      a[i]=str[size-1-i]-'0';
}
int bigIntMinus(int m[],int a[],int b[],int len)
{for(int i=0;i<len;i++)
      {  m[i]=a[i]-b[i];
         if(m[i]<0)  {a[i+1]--;  m[i]+=10;}
      }
      while(len>1&&m[len-1]==0) len--;
      return len;
}
int add(int s[],int a[],int b[],int len)
{  for(int i=0;i<len;i++)
      {s[i]+=a[i]+b[i];
         if(s[i]>9){s[i+1]+=s[i]/10; s[i]%=10;}
      }
      if(s[len])len++;
      return len;
}
```

```
int main()
{
    char ac[2000],bc[2000];
    int an[2000]={0},bn[2000]={0};
    char op, s1=1,s2=1;
    cin>>op;
    cin>>ac;
    cin>>bc;
    if(ac[0]=='-')
    {   s1=-1;
        strcpy(ac,&ac[1]);//将 ac 去除第一位,
    }
    if(bc[0]=='-')
    {   s2=-1;
        strcpy(bc,&bc[1]);
    }
```

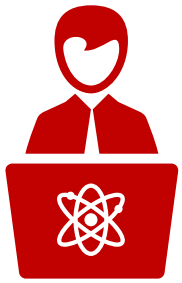
#291 大整数加减法

```
int lena=strlen(ac);
int lenb=strlen(bc);

parseStrToIntArr(ac,an,lena);
parseStrToIntArr(bc,bn,lenb);

int len=(lena>lenb)?lena:lenb;

if(op=='-')s2*=-1;
int sum[2018]={0};
if(s1==s2)
{   len=add(sum,bn,an,len);
    if(s1==-1)cout<<"-";
}
else
    if((lena>lenb)||(lena==lenb&&strcmp(ac,bc)>0))
        {len=bigIntMinus(sum,an,bn,len);
        if(s1==-1)cout<<"-";}
    else
        {len=bigIntMinus(sum,bn,an,len);
        if(s2==-1)cout<<"-";}
for(int i=len-1;i>=0;i--)
    cout<<sum[i];
```

#288 字符串之差

```
int fun ( char *s, char *t) {
    while (*s == *t) {
        if (*s == 0)
            return (0);
        ++s;
        ++t;
    }
    return (*s - *t);
}

int main() {
    char s1[100], s2[100];
    gets(s1);
    gets(s2);
    printf("%d\n", fun(s1, s2));
    return  0;
}
```

中国人民大学
RENMIN UNIVERSITY OF CHINA



谢谢大家！