函数的返回值

- 通过函数调用使主调函数能得到一个确定的值，该值就是函数的返回值。
- 函数的返回值通过函数中return语句获得。
- return语句的一般形式：
  - return 表达式；　或　return;
  - return （表达式);
- 函数定义时，如果该函数不需返回任何值，那么函数返回值类型应为void，否则就不能为void。
- 其后不跟表达式的return语句，只适用于返回值类型为void的函数。

函数返回值的说明

- 有返回值的函数必须有return语句。
- return的作用:
  - 使流程返回主调函数；
  - 将函数值返回到主调函数中
- 一个函数允许有一个或多个return语句，但每次调用只能有一个return被执行，因此函数的返回值只能有一个
- 一般函数返回值的类型应和return语句中表达式的类型一致，如果二者不一致，则以函数返回值的类型为准

- 例：输入两个整数，求两者中较大的并输出

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

例：判断闰年

```
#include <stdio.h>
int leap(int year)
{   int  s;
    if ((year%4==0 && year%100!=0) || (year%400==0))
            s = 1;
    else
            s = 0;
    return s;  /* 或 return (s);*/
}
int main(void)
{   int  y;
    scanf("%d", &y);
    t = leap(y);
    if (t == 1)
            printf("%d 年是闰年\n", y);
    else
            printf("%d 年不是闰年\n", y);
    return 0;
}
```

如果使用多个return语句怎么写？

- 例：输入一个正整数，判断该数是否为素数。

```
#include <stdio.h>
#include <math.h>
int  isprime ( int  m)
{    int  i, k;
     k = sqrt(m);
     for(i=2; i<=k; i++)
     {    if(m%i == 0)
               return  (0);
     }
     return  (1);
}
int main(void )
{
     int num, t;
     scanf("%d", &num);
     t = isprime(num);
     if (t==1)
               printf("%d is a prime\n", num);
     else
               printf("%d is not a prime\n", num);
     return 0;
}
```

- 例：求两个数的最大公约数和最小公倍数。
  - 求最大公约数的方法: 辗转相除法
    - 给出两个数，如果两数相除的余数不是0，则除数作为新的被除数，余数作为新的除数，继续相除，当两数相除的余数是0时结束, 此时除数就是最大公约数。
  - 最小公倍数的方法：用两数的乘积除以最大公约数

例: a=25 ;　　b=15;　　s=a%b=10;

a=b; (a=15)　b=s; (b=10)　s=a%b=5;

a=b; (a=10)　b=s; (b=5)　s=a%b=0;

b就是最大公约数;

最小公倍数:　25*15/5=75

- 求最大公约数的函数为：

```
int  gys(int  a,  int  b)  /*a为被除数，b为除数*/
{
    int  s;    /* s存放余数*/
    s = a % b;
    while(s != 0)
    {
          a = b;
          b = s;
          s = a % b;
    }
    return (b);
}
```

```
while((s=a%b) != 0)
{
      a = b;
      b = s;
}
```

- 完整程序的实现如下：

```
#include  <stdio.h>
int  main(void)
{
    int  p, q, r, z ;
    printf("please input 2 numbers: ");
    scanf("%d%d", &p, &q);
    r = gys(p, q);
    printf("Greatest common divisor is %d\n", r);
    z = p * q / r;
    printf("least common multiple is %d\n", z );
    return 0;
}
```

函数的原型与声明

- 就像变量需要先定义后使用一样，函数也必须先定义后使用
- 当一个函数去调用另一个函数时，需具备如下条件：
  - 不管是库函数还是自定义函数，该被调函数都必须是已经定义好的函数
  - 如果该被调函数是库函数，需要在文件开头加#include指令把相关头文件包含进来
  - 如果该被调函数是自定义函数，且其位置在主调函数后面，就应该在主调函数中对该被调函数进行声明。

- 声明的作用：
  - 把函数的返回值类型、函数名、函数参数的个数和参数类型等信息通知编译器，以便在遇到函数调用时，编译器能正确识别函数并检查调用是否合法。
- 函数声明的一般形式：

  函数类型　函数名 (形参列表) ;
- 函数声明的注意事项:
  - 函数声明末尾要加分号;
  - 形参列表中可以只写数据类型；如 int　f(int , float);
  - 形参的先后次序和函数定义时的次序应一致
  - 当被调函数的定义写在主调函数之前时, 允许省略函数声明
  - 建议将程序中用到的函数都在程序的前面加以声明
  - 函数的声明其实就是函数定义时的函数首部再加一个分号，函数首部称为函数原型

函数声明的位置及说明

- 声明的位置
  - 在主调函数中对被调函数进行声明，这种声明只能在本函数内起作用
  - 在所有函数的外部对被调函数进行声明，这种声明可以在全局起作用
- 说明
  - 函数的定义和声明不是一回事。
  - 定义是指对函数功能的确立，它是一个完整、独立的函数单位。
  - 声明是指函数的名字、函数类型，以及形参类型、个数和顺序通知编译器，以便在调用该函数时系统按此进行对照检查。

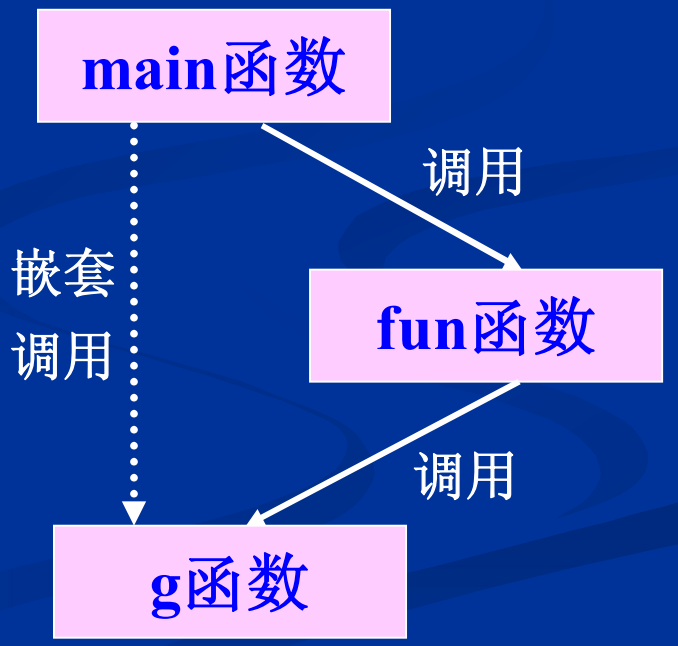
函数声明示例

```
#include <stdio.h>
double  average(int  x,  int  y);   //函数声明，出现在函数体外，对下面函数都可见
int  main(void)
{
     int  a, b;
     double  ave;
     double  average(int  x,  int  y);   //函数声明，出现在函数体内，只对该函数可见
     scanf("%d%d", &a, &b);
     ave = average(a, b);
     printf("Average of %d and %d is %.1f\n", a, b, ave);
     return  0;
}
double average(int x, int y)     //函数定义出现在主调函数后面
{
     double result;
     result = (x +y) / 2.0;
     return  result;
}
```

函数的嵌套

- 函数的嵌套调用是指在被调用函数的执行过程中又调用另一个函数。

```
void  main( )      void  fun( void )
{       :          {       :
     fun( ) ;              g( ) ;
        :                     :
}                  }
```



函数的嵌套调用示例

- 例：求1～n之间所有素数的和。
- 分析:
  - 该问题的主体结构还是求和运算，而参与求和的数是1～n之间的素数。
  - 可以分3步求解:

① 输入n的值

② 求和运算

③ 输出结果

  - 求和计算时先要判断当前的数是否为素数, 把判断素数编成一个函数在求和计算中调用这个函数

- 程序代码如下：

```
#include   <stdio.h>
#include   <math.h>
long  sum (int  n);
int  isprime (int  m);

int   main(void)
{
    int  n;
    long  s;
    scanf("%d", &n);
    s = sum(n);
    printf("%ld", s);
    return  0;
}
```

函数声明
(外部)

```
long  sum(int n)
{
    int  i;
    long s = 0;
    for(i=1; i<=n; i++)
        if ( isprime(i) == 1 )
            s = s + i;
    return  s;
}

int   isprime (int  m)
{
    int  i, k;
    k = sqrt(m);
    for(i=2; i<=k; i++)
        if(m%i == 0)
            return  0;
    return  1;
}
```

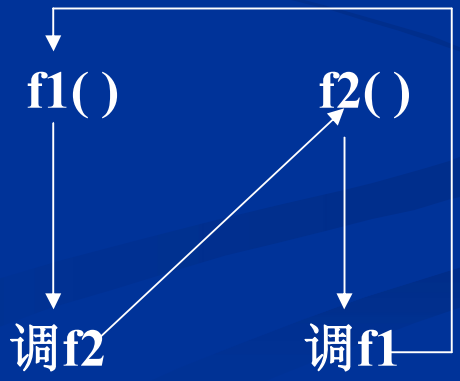
函数的递归

- 函数直接或间接的调用自身叫函数的递归调用

```
int  f(int x)
{    int y,z;
      ……
     z=f(y);
     ……
    return(2*z);
}
```

```
int  f1(int x)
{    int y,z;
      ……
     z=f2(y);
     ……
    return(2*z);
}
```

```
int  f2(int t)
{    int a,c;
      ……
     c=f1(a);
     ……
    return(3+c);
}
```



递归求解问题

- 用递归求解问题的特点
  - 存在递归的终止条件
  - 存在导致问题求解的递归方式
- 使用递归求解问题的优缺点
  - 优点：程序简洁，代码紧凑
  - 缺点：每调用函数一次，在内存堆栈区分配空间，用于存放函数变量、返回值等信息，所以递归次数过多，可能引起堆栈溢出，且时间效率较差

递归求解问题示例

- 例：求n的阶乘

递归的终止条件

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

递归方式

```
#include  <stdio.h>   /*递归方式*/
#inlcude  <stdlib.h>
float fac(int n)
{   float  f;
    if(n<0)  exit(0);
    else if(n==0||n==1)  f = 1;
    else f = fac(n - 1) * n;
    return  f;
}
int  main(void)
{   float  y;
     int  n;
     printf("Input a integer number:");
     scanf("%d", &n);
     y = fac(n);
     printf("%d! =%.0f", n, y);
     return 0;
}
```

库函数的使用

- 库函数就是系统提供的可以实现某种功能的函数的集合
- 每一类库函数都一个相应的头文件，该头文件包含了该库中所有函数的函数原型，一般扩展名为.h
- 使用库函数应注意：
  - 函数功能
  - 函数参数的数目和顺序，及各参数意义和类型
  - 函数返回值意义和类型
  - 需要使用函数所在的包含文件，若包含的头文件多于一个，则每个#include需单独占一行