

# 计算机程序设计基础

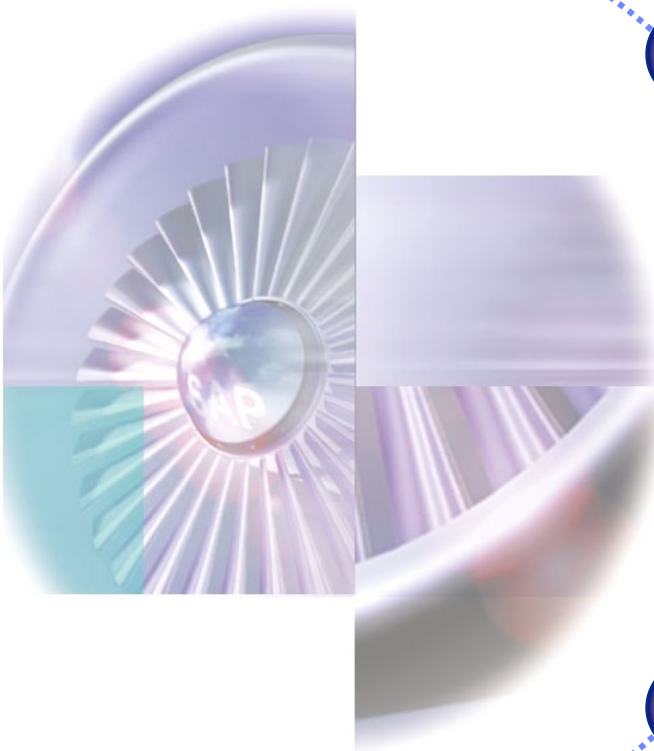
Programming Fundamentals

韩文弢

清华大学计算机系



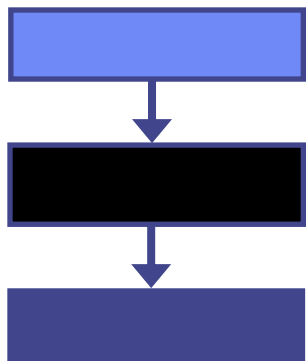
# 第五章 函数

- 
- ① 函数概述
  - ② 函数的使用
  - ③ 函数调用的实现过程
  - ④ 函数与设计

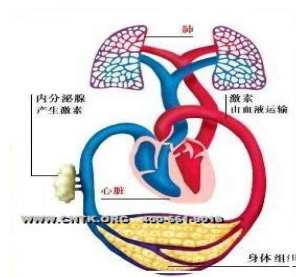
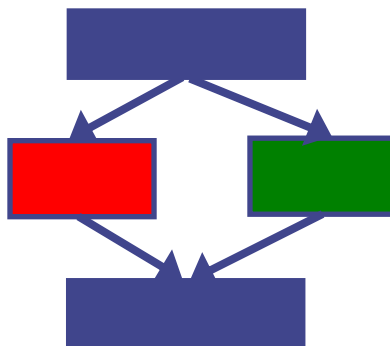
# 5.1 函数概述



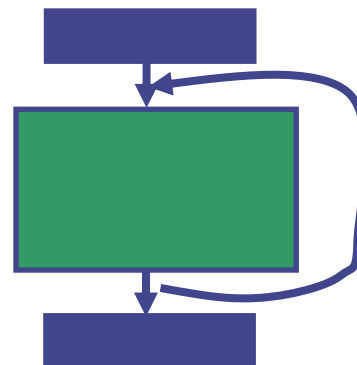
顺序



选择



循环



两个特点？

1. 重复执行
2. 连续执行

# 大学生小明的一周

一周的学习/生活总结：

1. 睡觉7次；
2. 去食堂吃饭21次（或14次）；
3. 上课11次；
4. 去教室上自习6次；
5. 参加拔河比赛1次；
6. 去大礼堂看电影1次；
7. ....

重复不连续！

1. 睡觉;
2. 吃饭;
3. 上课;
4. 上自习;
5. 睡觉;
6. 吃饭;
7. 上课;
8. ....

```
for (i=1; i<=7; i++)  
{  
    睡觉;  
}  
for (i=1; i<=21; i++)  
{  
    吃饭;  
}  
for (i=1; i<=11; i++)  
{  
    上课;  
}
```

如何处理“重复但不连续”的操作？

——函数

# 什么是函数？

- 数学里面的函数：正弦函数、余弦函数、指数函数等等；
- C++语言中的函数：也称为子程序（**subroutine**），它是一组程序代码（包括数据和指令），用来完成某个特定的功能。
- C++语言的函数既可以有返回值，也可以无返回值，不像Pascal等语言，严格地区分为函数（**function**）和过程（**procedure**）。

## C++语言的函数类型

- 主函数**main**，每个程序有且仅有一个；
- 库函数，也叫标准函数，由系统提供，用户可以直接使用；
- 自定义函数，用来完成用户特制的功能。



## 在编程时...

我想生成一个随机数

用rand函数！

我想计算一个字符串的长度

用strlen函数！

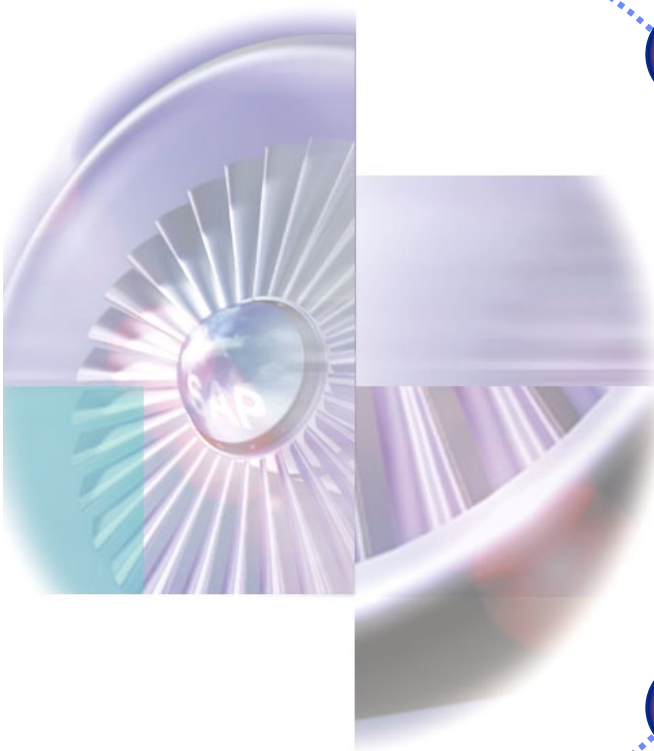
我想知道一个人的星座和属相

自己编写函数！

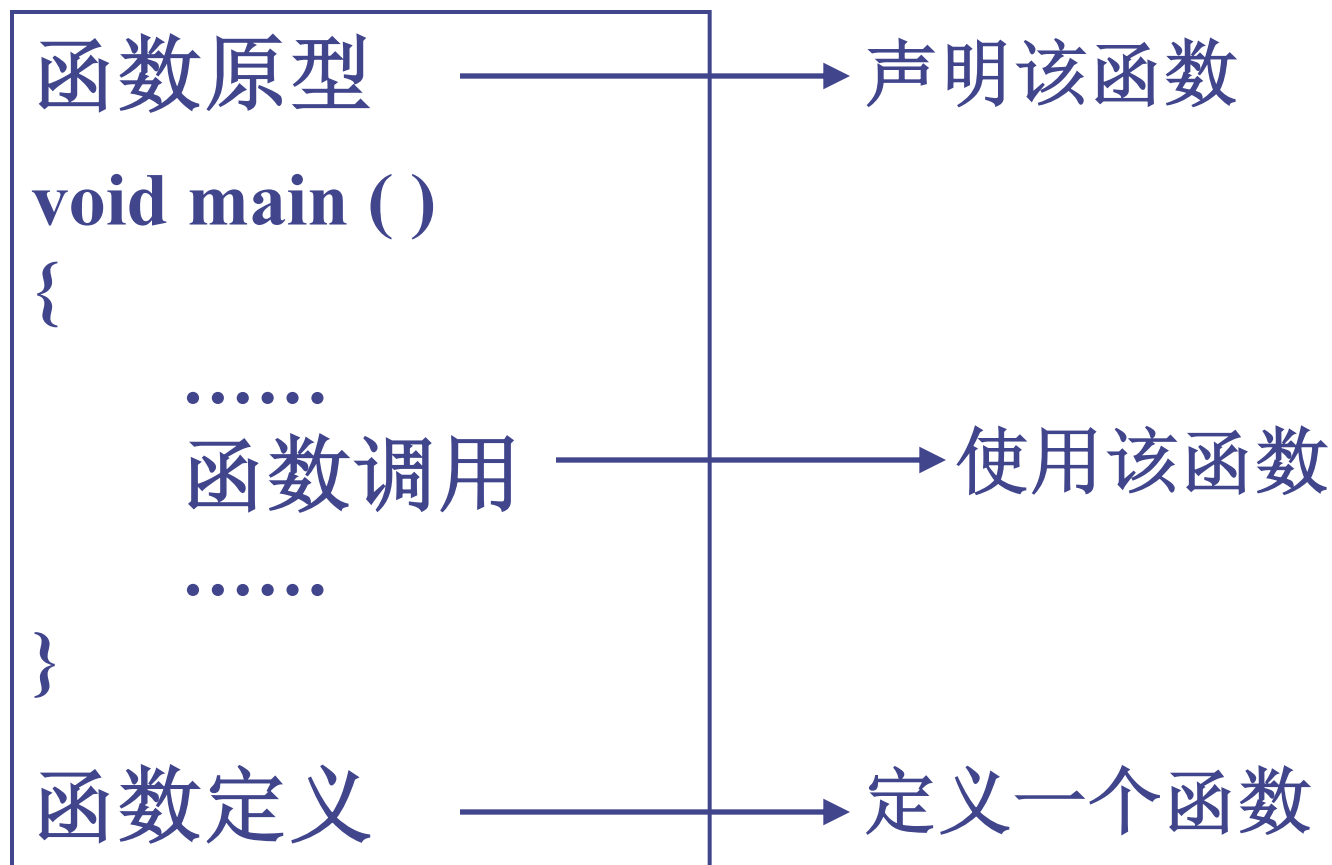
# 程序员如何摧毁世界？

一名合格的程序员是不会直接写出诸如“摧毁地球”这样的代码的，他们会先定义一个函数叫做“摧毁行星”，然后调用该函数，并将“地球”作为一个参数传进去。

# 第五章 函数

- 
- 1 函数概述
  - 2 函数的使用
  - 3 函数调用的实现过程
  - 4 函数与设计

## 5.2 函数的使用



函数的使用模式

## 5.2.1 函数的定义

函数定义的一般形式：

```
<数据类型> <函数名> (<参数列表>)  
{  
    <数据声明部分>  
    <执行语句>  
}
```

```

int WhoWin(char child1, char child2)
{
    int result;

    if(child1 == child2)    result = 0;
    else if(child1 == 'S' && child2 == 'J')
        result = 1;
    else if(child1 == 'S' && child2 == 'B')
        result = -1;
    else if(child1 == 'J' && child2 == 'B')
        result = 1;
    ....
    return result;
}

```

函数无返回值时能否用return语句?  
 return语句是否必须在函数末尾?  
 在函数中能否出现多个return语句?

## 5.2.2 函数的声明

在使用一个函数的时候，除了要对它进行定义以外，还要对它进行声明，即给出这个函数的函数原型（**prototype**）。

- 一些函数原型的例子:

- **void** Useless( **void** );
- **void** PrintInteger(**int** value);
- **double** CalculateTax(**double** amount,  
**double** rate);



## 5.2.3 函数的使用

### 1. 用户自定义的函数

```
void hello( ); // 先声明后使用
void main( )
{
    hello( );    // 函数调用
}
void hello( )    // 函数定义
{
    cout << "hello";
}
```

## 2. 使用其他文件中的函数

问题描述：

在一个程序中，有**多个**源文件，如何在一个文件中调用其他文件中定义的函数？

- 假设有源文件`tools.cpp`，存放一些工具函数；
- 另定义一个头文件`tools.h`，存放函数原型；
- 在`main.cpp`中`#include "tools.h"`；
- 调用相应的函数。

### 3. 使用系统提供的库函数

问题描述：

如何在源程序中，调用系统提供的库函数，如`sin`，`cos`等，编译器需要它们的函数原型和代码吗？

- 编译器需要，但程序员不需要；
- 将相应头文件包含进来，如`#include <cmath>`；
- 链接程序知道这些库函数的代码所在的位置，会自动地链接到目标程序当中。

## 5.2.4 函数的调用

### 1. 函数调用的一般形式

函数名 (参数1, 参数2, ..., 参数n);

例如:

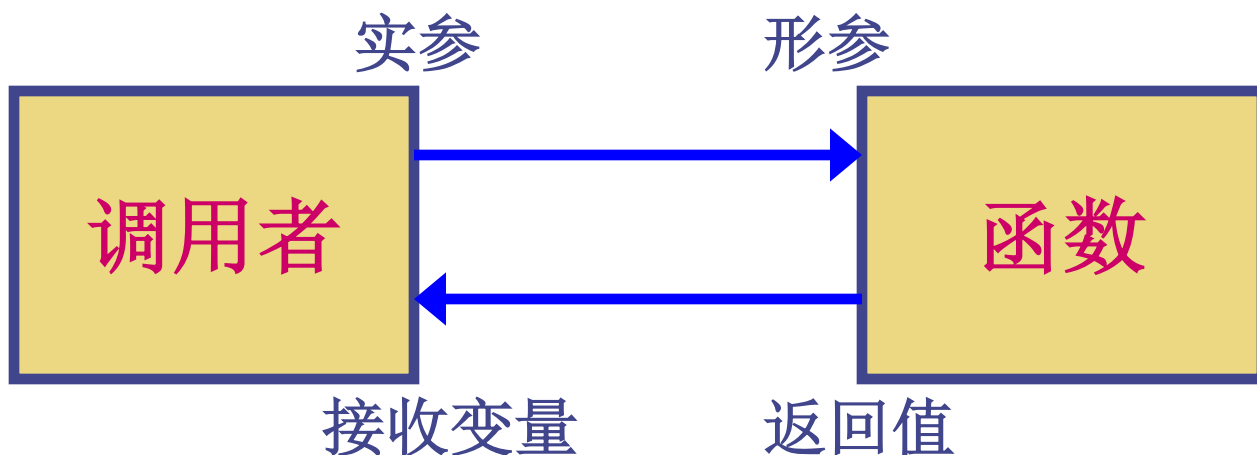
```
y = sqrt(x);
```

```
cout << "hello world!";
```

## 2. 实参与形参

- ☺ 实参：在调用一个函数时，所指定的参数称为“实际参数”，可以是常量、变量或表达式；
- ☺ 形参：在定义一个函数时，所指定的参数称为“形式参数”，必须是一个变量。

1. 个数？
2. 方向？



```
void main( )
{
    int salary, nCars, nHouses;

    salary = 6000;
    nCars = 0;
    nHouses = 0;
    DayDreaming(salary, nCars, nHouses);
    cout << salary <<" "<< nCars <<" "<< nHouses;
}
void DayDreaming(int salary, int cars, int houses)
{
    salary = salary * 3;
    cars += 2;
    houses ++;
}
```

## 5.2.5 变量的作用范围

变量的作用范围：也称为变量的作用域，指程序中的一段代码范围，在此范围内，这个变量是有效的，可以被访问。而在此范围之外，该变量是无效的，不能被访问。

一个变量在它的作用域以内是“可见”的，在它的作用域以外是“不可见”的。

变量的作用范围可以分为三类：

- 函数一级的作用范围，即局部变量；
- 文件一级的作用范围，即全局变量。
- 复合语句一级的作用范围



# 1. 局部变量

**局部变量：** 在一个函数内部定义的变量

- ✦ **局部变量只在本函数范围内有效；**
- ✦ **在不同函数中可使用相同名字的局部变量；**
- ✦ **形参也是局部变量，也只能在本函数中使用；**
- ✦ **局部变量的生存期：当函数被调用时，其局部变量才被创建，并分配相应内存空间；当函数调用结束后，局部变量即消亡，其空间被释放。**

```
float f1(int a)
```

```
{
```

```
    int b, c;
```

```
    ...
```

```
}
```

} a、b、c有效

```
char f2(int x, int y)
```

```
{
```

```
    int i, j;
```

```
    ...
```

```
}
```

} x、y、i、j有效

```
void main()
```

```
{
```

```
    int m, n;
```

```
    ...
```

```
}
```

} m、n有效

## 2. 全局变量

**全局变量：** 在所有函数之外定义的变量

- ✦ 全局变量可以为本文件中的其他函数所共用；
- ✦ 其有效范围为从定义该变量的位置开始，到本源文件结束为止。
- ✦ 生存期：程序运行过程中始终存在。

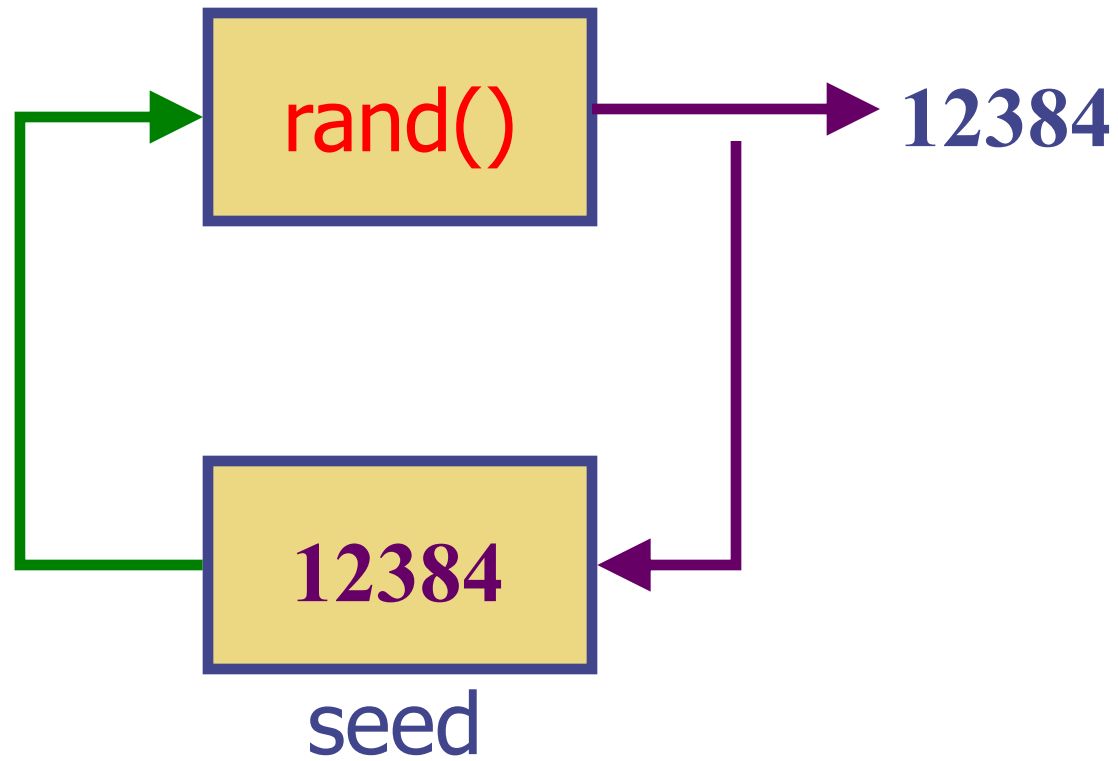
```
int p, q;  
float f1(int a)  
{  
    int b, c;  
    ...  
}
```

```
char c1, c2;  
char f2(int x, int y)  
{  
    int i, j;  
    ...  
}
```

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

全局变量  
c1、c2的  
作用范围

全局变量  
p、q的  
作用范围



rand.c

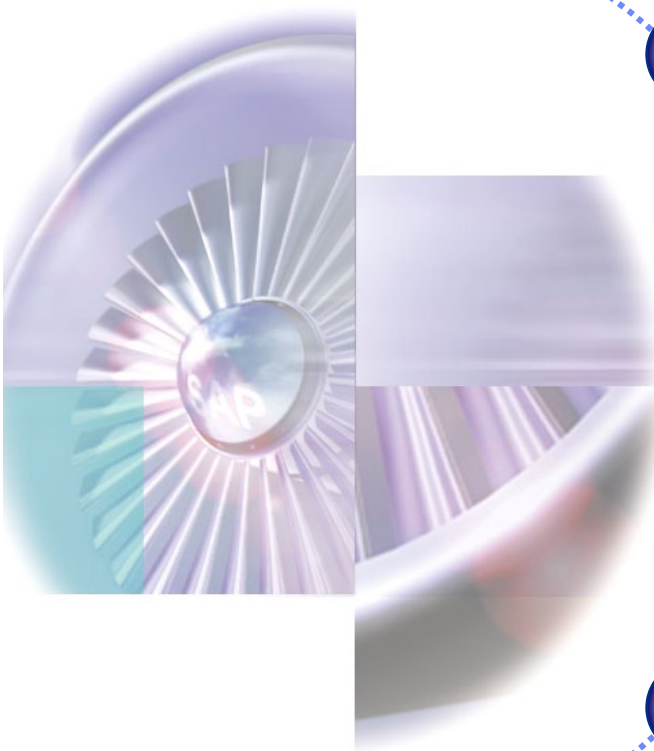
```
int seed;  
int rand( int seed )  
{  
    int seed;  
    . . . . .  
}
```

### 3. 复合语句

```
int main()  
{  
    int sum = 0;  
    for( int i = 1; i <= 10; i++ )  
    {  
        sum += i;  
    }  
    cout << i << endl; // 编译错误  
}
```

} sum、i有效

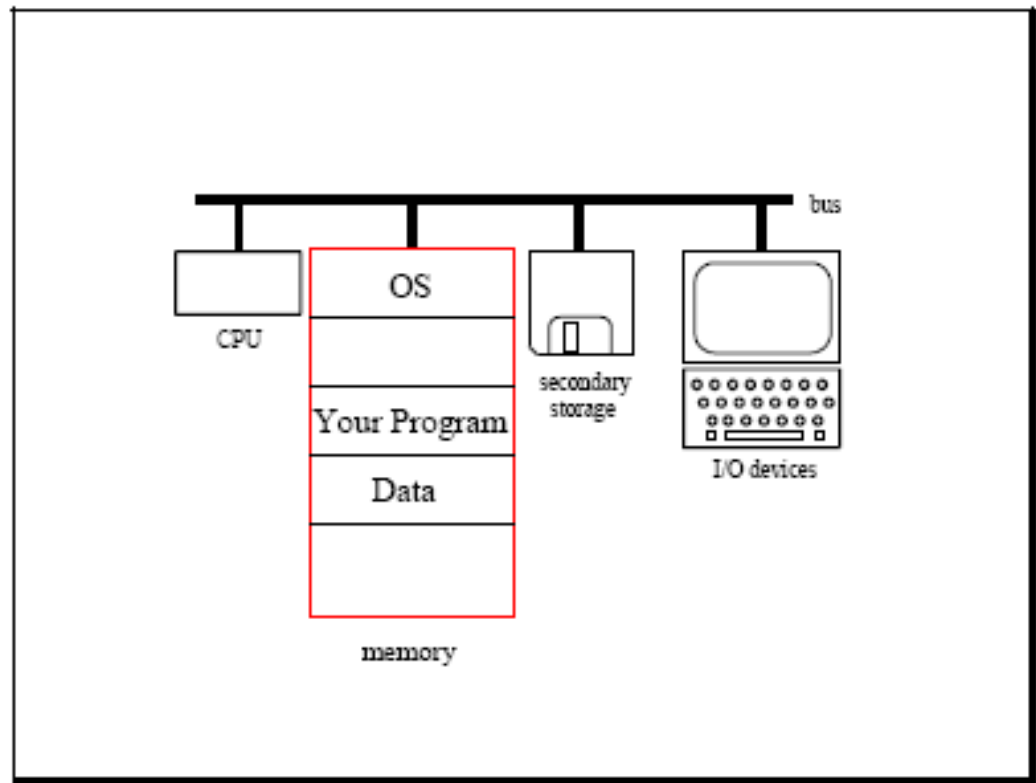
# 第五章 函数

- 
- 1 函数概述
  - 2 函数的使用
  - 3 函数调用的实现过程
  - 4 函数与设计

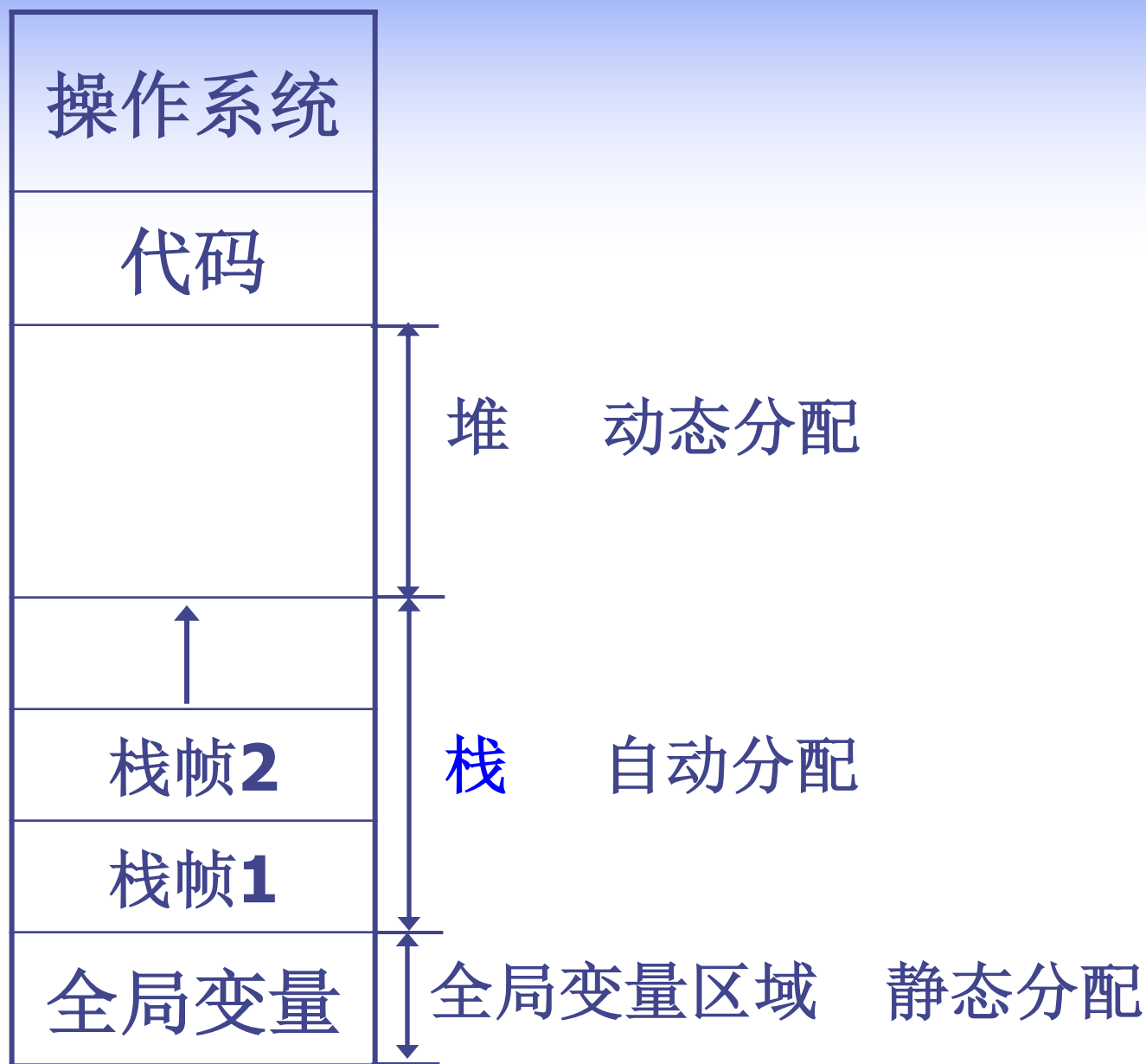


## 5.3.1 内存布局

- 存储程序原理，  
1945，John von Neumann
- 把代码和数据都存放在内存中



# 内存分布状况



```
#include <stdio.h>
int sum;
int Add(int a, int b);
void main()
{
    int x, y;
    cin >> x >> y;
    sum = Add(x, y);
    cout << sum << endl;
}
int Add(int a, int b)
{
    int result;
    result = a + b;
    return result;
}
```

**sum:        0x004257B0**

**scanf: 0x00401160 ~ 0x004011BA**

**printf: 0x004010E0 ~ 0x0040115B**

**Add:       0x004010A0 ~ 0x004010CA**

**main:       0x00401020 ~ 0x00401086**

**x:           0x0012FF7C**

**y:           0x0012FF78**

**a:           0x0012FF24**

**b:           0x0012FF28**

**result: 0x0012FF18**

## 5.3.2 函数调用的实现过程

**控制流：** 程序当前执行位置的流向；

**数据流：** 函数调用发生及结束时，数据在  
函数之间流转的过程。

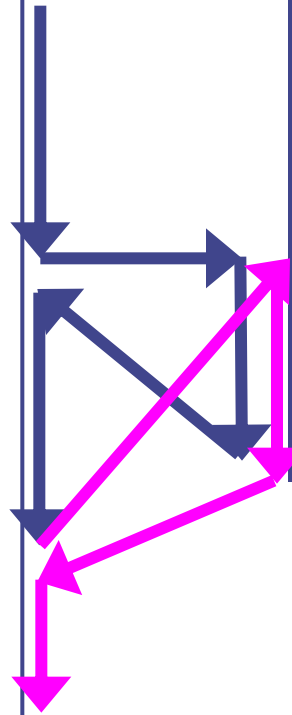
## 当一个函数被调用时：

1. 在内存的栈空间当中为其分配一个栈帧，用来存放该函数的形参和局部变量；
2. 把实参的值复制到相应的形参变量；
3. 控制转移到该函数的起始位置；
4. 该函数开始执行；
5. 控制流和返回值返回到函数调用点，栈帧释放。

# 控制流的变化

```
void main()  
{  
    double x, y, z;  
  
    y = 6.0;  
    x = Area( y / 3.0 );  
    ...  
  
    z = 3.4 * Area(7.88);  
    ...  
}
```

```
/* 给定半径，计算一个圆的面积 */  
double Area(double r)  
{  
    return(3.14 * r * r);  
}
```



# 一个简单的例子

```
int Times2(int value);  
void main ( )  
{  
    int number;  
    cout << "请输入一个整数: ";  
    cin >> number;  
    cout << "该数的两倍是: " << Times2(number);  
}  
int Times2(int value)  
{  
    return(2 * value);  
}
```

**main**

**number**

**3**



```

int Times2(int value);
void main ( )
{
    int number;
    cout << "请输入一个整数: ";
    cin >> number;
    cout << "该数的两倍是: " << Times2(number);
}

int Times2(int value)
{
    return(2 * value);
}

```

**main**

**number**

3

**Times2**

**value**

Times2也得到一个栈帧，  
它的参数看成局部变量

```
int Times2(int value);  
void main ( )  
{  
    int number;  
    cout << "请输入一个整数: ";  
    cin >> number;  
    cout << "该数的两倍是: " << Times2(number);  
}  
→ int Times2(int value)  
{  
    return(2 * value);  
}
```

**main**

**number**

3

**Times2**

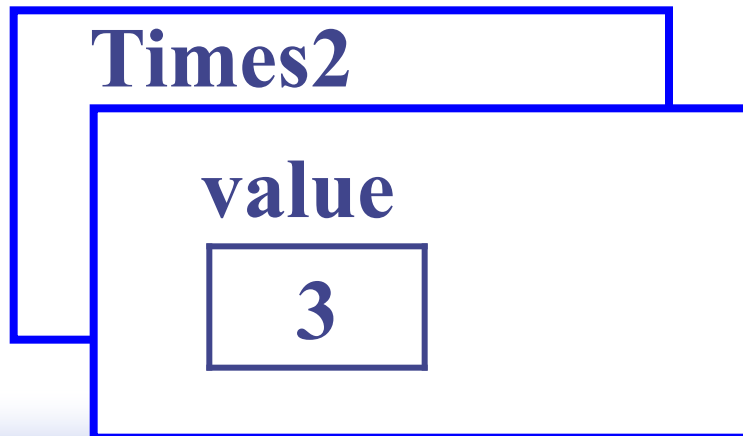
**value**

3

“值传递”，把实参的值传给形参。

```
int Times2(int value);  
void main ( )  
{  
    int number;  
    cout << "请输入一个整数: ";  
    cin >> number;  
    cout << "该数的两倍是: " << Times2(number);  
}  
→ int Times2(int value)  
{  
    return(2 * value);  
}
```

**main**



把**Times2**的栈帧叠在主函数的栈帧之上，说明在执行**Times2**函数时，主函数中的变量是不可见的。

```
int Times2(int value);  
void main ( )  
{  
    int number;  
    cout << "请输入一个整数: ";  
    cin >> number;  
    cout << "该数的两倍是: " << Times2(number));  
}  
  
int Times2(int value)  
{  
    return(2 * value);  
}
```

6

**main**

**number**

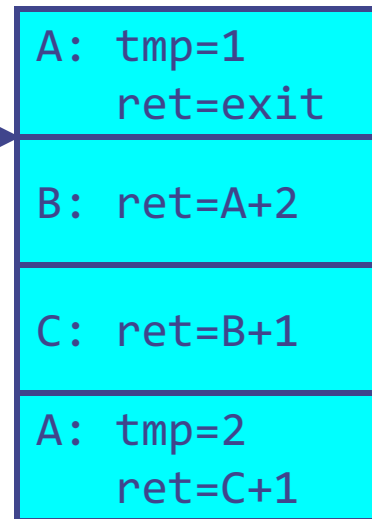
3

Times2函数的返回值被放在函数的调用位置上, 然后, 分配给Times2函数的堆栈区域被释放。

# 另一个例子

```
A(int tmp) {  
    if (tmp<2)  
        B();  
    printf(tmp);  
}  
B() {  
    C();  
}  
C() {  
    A(2);  
}  
A(1);
```

**Stack  
Pointer**

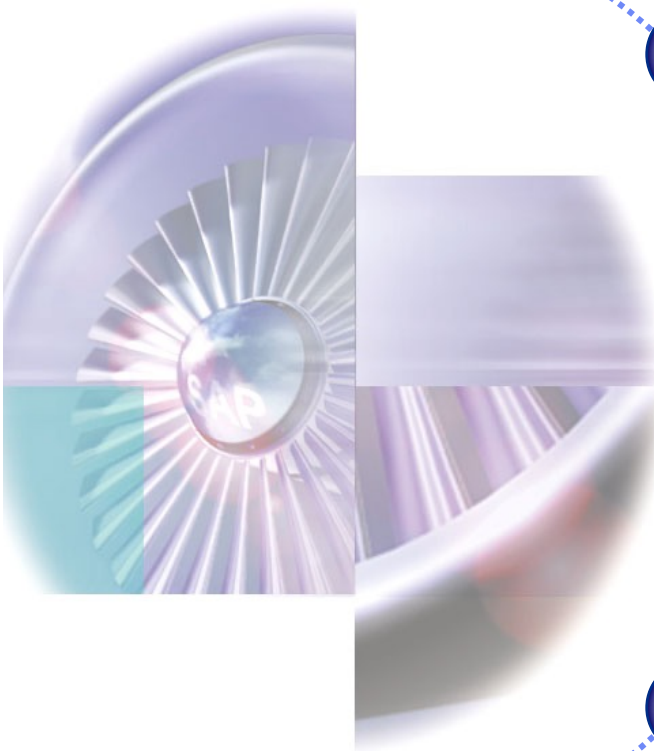


Stack Growth

## 函数与栈帧：

1. 若一个程序中定义了10个函数，是否栈中一定会有10个栈帧？
2. 若一个程序中定义了1个函数，是否栈中最多只会有1个栈帧？若不只1个，最多多少个？
3. 栈帧的个数到底如何确定？

# 第五章 函数

- 
- 1 函数概述
  - 2 函数的使用
  - 3 函数调用的实现过程
  - 4 函数与设计

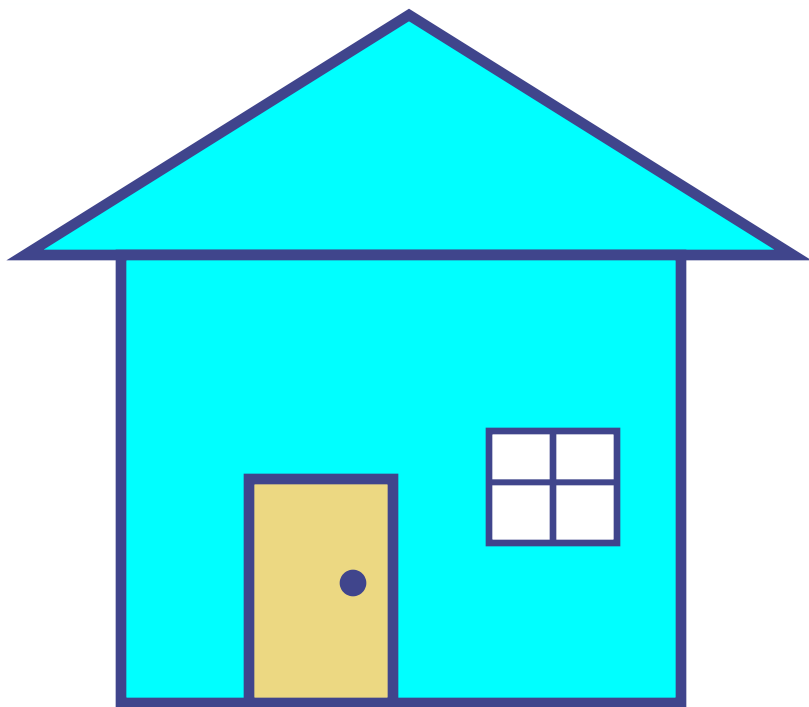
## 5.4.1 设计过程

如何设计一栋简易的房子？

输入：一张白纸

输出：一张图纸





```
draw_house(color, num_windows)
```

```
    draw body as a colored rectangle
```

```
    draw roof as a colored triangle
```

```
    if num_windows is one
```

```
        draw door
```

```
        draw window
```

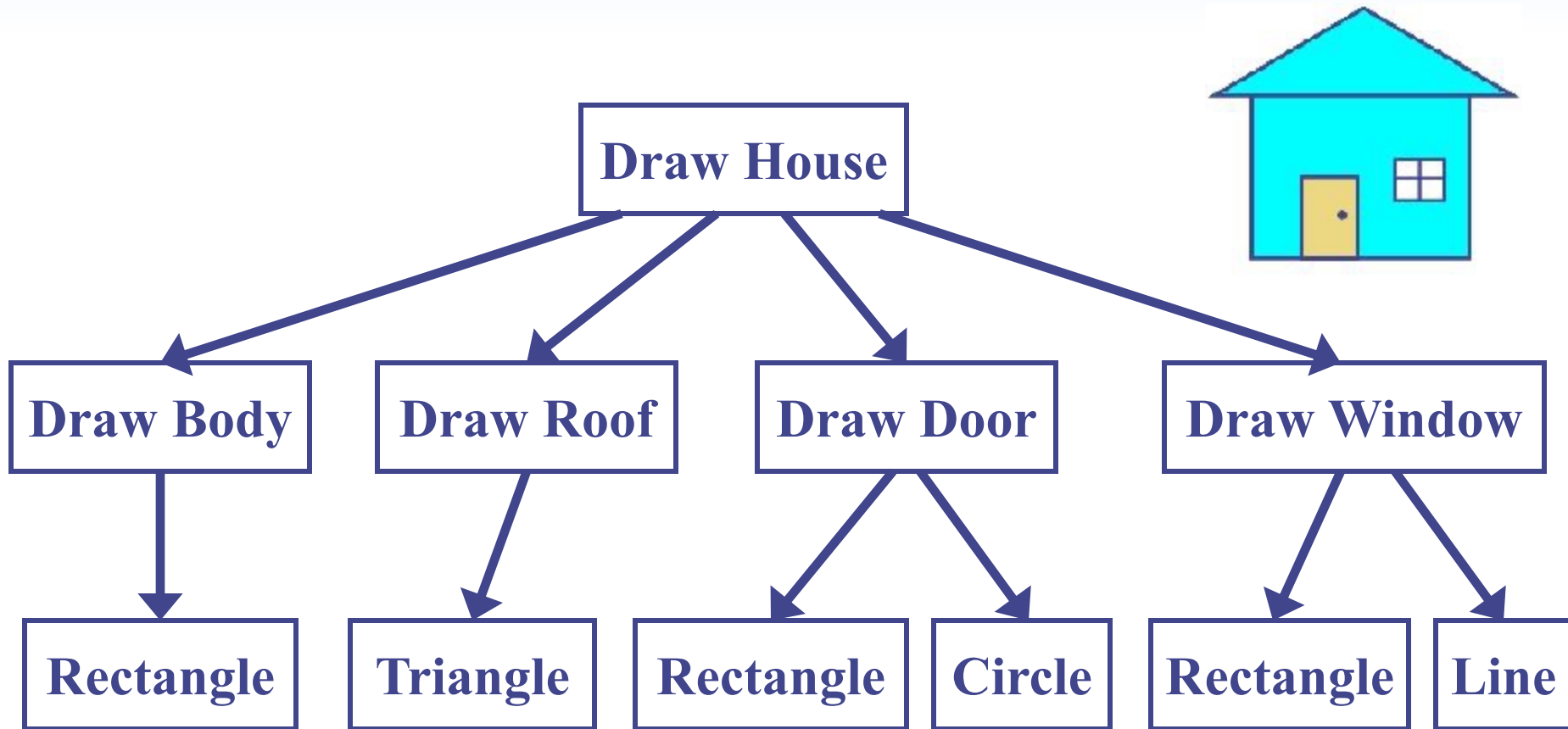
```
    if num_windows is two
```

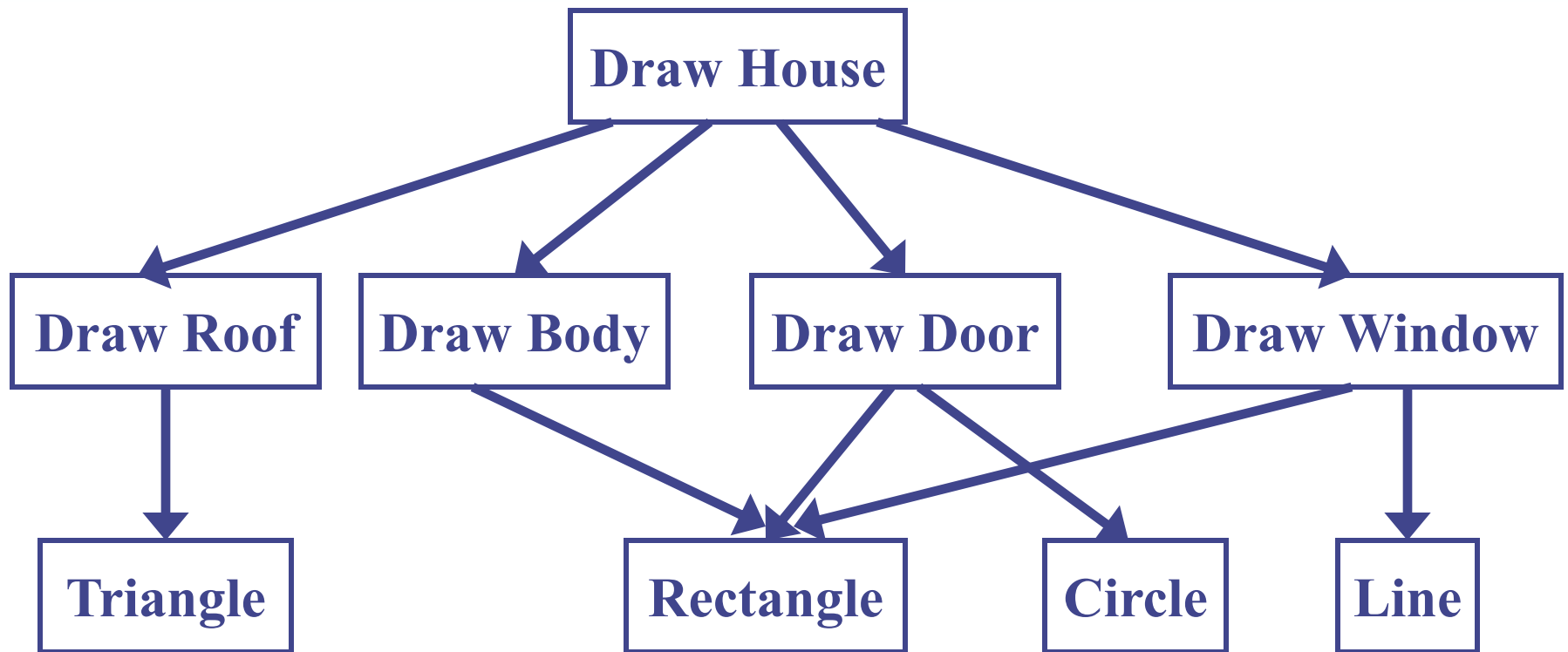
```
        draw door
```

```
        draw window
```

```
        draw window
```

## 5.4.2 函数分解





# 分析、设计与编程

- ◆ 分析问题
- ◆ 设计一个“big-picture”解决方案
  - ☺ 功能分解、相互关系。
- ◆ 设计各个函数
  - ☺ 基于现有的函数
- ◆ 编程实现

## 5.4.3 质数对

### 问题描述:

一个加密算法需要用到一对质数，符合要求的质数对必须满足：

- 两个数不相同，且都是质数；
- 在这两个质数之间没有其他质数。

编写一个程序，输入任意两个正整数，判断它们是否是有效的质数对。

整数1	整数2	是否有效	原因
<b>15</b>	<b>17</b>	无效	<b>15</b> 不是质数
<b>13</b>	<b>19</b>	无效	<b>17</b> 是质数
<b>17</b>	<b>19</b>	有效	
<b>31</b>	<b>29</b>	有效	
<b>31</b>	<b>31</b>	无效	两个数相同

# 问题分析

1. 需要定义几个函数？
2. 函数原型是什么？
3. 质数函数如何实现？
4. 如何判断一对整数是否有效？需要做哪些事情？



```
#include <iostream>
#include <cmath>
using namespace std;
void main()
{
    int num1, num2, i, temp;
    int result = 1;

    cin >> num1 >> num2;
    if((num1 == num2) || !IsPrime(num1) ||
        !IsPrime(num2))
        result = 0;
    else
    {
        if(num1 > num2)
        {
            temp = num1;
            num1 = num2;
            num2 = temp;
        }
    }
}
```

```
    for(i = num1+1; i < num2; i++)
    {
        if(IsPrime(i))
        {
            result = 0;
            break;
        }
    }
}
if(result == 1)
    cout <<num1<<"和"<<num2<<"是有效质数对";
else
    cout <<num1<<"和"<<num2<<"是无效质数对";
}
```

判断整数 $m$ 是否为质数的方法：让  $m$  被 2 到  $\sqrt{m}$  除，若  $m$  能被其中任何一个数整除，则说明它不是一个质数；否则的话，说明它是一个质数。

```
bool IsPrime(int m)
{
    int j, sq;

    sq = (int)sqrt((double)m);
    for(j = 2; j <= sq; j++)
    {
        if(m % j == 0) break;
    }
    if(j > sq)
        return true;
    else
        return false;
}
```

## 5.4.4 循环右移

### 问题描述:

编写一个程序，读入一组整数（不超过20个），当用户输入0时，表示输入结束。接下来再输入一个正整数M，然后程序将把这组整数循环右移M次，然后把循环右移的结果打印出来。

所谓循环右移，就是把每个数组元素往右移动一格，然后把最右边的那个元素移回到最左边。例如，对于一组整数“100 400 200 300”，把它循环右移一次的结果是“300 100 400 200”；把它循环右移两次的结果是“200 300 100 400”。

# 问题分析

1. 如何循环右移**M**位？
2. 能否进行问题**分解**？
3. 如何循环右移**1**位？

# 如何将每个数组元素循环右移1位？

0	2
1	1
...	...
N-2	4
N-1	3

```
temp = a[N-1];  
for( k = N-1; k > 0; k-- )  
{  
    a[k] = a[k-1];  
}  
a[0] = temp;
```

```

#include <stdio.h>
void shift(int a[], int N);
void main()
{
    int N=0, b[20], i, M;

    while(1)
    {
        cin >> b[N];
        if(b[N] == 0) break;
        else N++;
    }
    cin >> M;
    for(i = 1; i <= M; i++) shift(b, N);
    for(i = 0; i < N; i++) cout << b[i]<<' ';
}

```



```
void shift(int a[], int N)
{
    int temp, k;

    temp = a[N-1];
    for(k = N-1; k > 0; k--)
    {
        a[k] = a[k-1];
    }
    a[0] = temp;
}
```

```

#include <stdio.h>
void shift(int a[], int N);
void main()
{
    int N=0, b[20], i, M;
    while(1)
    {
        cin << b[N];
        if(b[N] == 0) break;
        else N++;
    }
    cin << M;
    for(i = 1; i <= M; i++) shift(b, N);
    for(i = 0; i < N; i++) cout << b[i]<<' ';
}

```

```

void shift(int a[], int N)
{
    int temp, k;
    temp = a[N-1];
    for(k = N-1; k > 0; k--)
    {
        a[k] = a[k-1];
    }
    a[0] = temp;
}

```

在函数调用时，传地址而不传值

Question?

**a    a[0]    a[1]    a[2]    a[3]    a[4]**

<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>
----------	----------	----------	----------	----------

**b    b[0]    b[1]    b[2]    b[3]    b[4]**

main的栈帧

# 本讲小结

- ◆ 函数的概念
- ◆ 多文件程序
- ◆ 变量的作用域与内存布局
- ◆ 设计思想