



程序设计与算法(一)

C语言程序设计

郭 炜

微信公众号



微博: <http://weibo.com/guoweiofpku>

学会程序和算法，走遍天下都不怕!

讲义照片均为郭炜拍摄



北京大学
PEKING UNIVERSITY

信息科学技术学院

指定教材：

《新标准C++程序设计教程》

郭炜 编著

清华大学出版社

重点大学计算机专业系列教材

新标准C++程序设计教程

郭炜 编著



清华大学出版社



北京大学
PEKING UNIVERSITY

信息科学技术学院

函数



壶口瀑布

为什么需要函数

- 写了一段牛顿迭代法求平方根的代码，程序里面无数地方都要求平方根，难道需要的地方都把这段代码拷贝一遍？
- 一个数十万行的程序，都写在 main 里面？数百个程序员如何合写一个 main？

为什么需要函数

- “函数” 可以将实现了某一功能，并需要反复使用的代码包装起来形成一个功能模块（即写成一个“函数”），那么当程序中需要使用该项功能时，只需写一条语句，调用实现该功能的“函数”即可。
- 不同的程序员可以分别写不同的函数，拼起来形成一个大程序

函数的定义

```
返回值类型 函数名(参数1类型 参数1名称, 参数2类型 参数2名称.....)
{
    语句组(即 “函数体” )
}
```

- 如果函数不需要返回值，则 “返回值类型” 可以写 “void”

函数调用和return语句

- 调用函数：

函数名 (参数1,参数2,)

- 对函数的调用，也是一个表达式。函数调用表达式的值，由函数内部的return语句决定。return语句语法如下：

return 返回值;

- return语句的功能是结束函数的执行，并将“返回值”作为结果返回。“返回值”是常量、变量或复杂的表达式均可。如果函数返回值类型为“void”，return语句就直接写：

return ;

函数调用和return语句

- return 语句作为函数的出口，可以在函数中多次出现。多个return语句的“返回值”可以不同。在哪个return语句结束函数的执行，函数的返回值就和哪个return语句里面的“返回值”相等。

函数使用实例1：Max函数

```
#include <iostream>
using namespace std;
int Max(int x,int y) //求两个整型变量中的较大值
```

```
{
    if( x > y )
        return x;
    return y;
}
```

形参

=

形参实参类型需兼容!

```
int main()
{
    int n = Max(4,6);
    cout << n << ", " << Max(20,n) << endl;
    return 0;
} => 6,20
```

实参

函数使用实例2：判断是否是素数的函数

```
#include <iostream>
using namespace std;
bool IsPrime(unsigned int n)
{
    if( n <= 1 )
        return false;
    for( int i = 2; i < n; ++i )    //看看有没有n的因子
        if( n % i == 0 )
            return false;
    return true;
}

int main() {
    cout << IsPrime(2) << "," << IsPrime(4) << "," << IsPrime(5);
    return 0;
} => 1,0,1
```

返回值为void的函数

```
void DrawCircle(double x,double y,double r)
{
    //下面的代码在屏幕上以(x,y)点为圆心, r为半径画圆
    .....
    return;
}
```

调用:

```
DrawCircle(0,0,1);
```

函数使用实例3：已知三角形三个顶点位置, 求边长

给定平面上不共线的三个点，其坐标都是整数，编写程序，求它们构成的三角形的三条边的长度。输入是6个整数: $x_1, y_1, x_2, y_2, x_3, y_3$ 代表三个点的坐标，以任意顺序输出三条边的长度均可。

```
#include <iostream>
using namespace std;
#define EPS 0.001 //用以控制计算精度
double Sqrt(double a)
{
    //求a的平方根
    double x = a/2, lastX = x + 1 + EPS; //确保能够进行至少一次迭代
    while( x - lastX > EPS || lastX - x > EPS) {
        //只要精度没有达到要求, 就继续迭代
        lastX = x;
        x = (x + a/x)/2;
    }
    return x;
}

double Distance(double x1, double y1, double x2, double y2)
{
    //求两点(x1,y1), (x2,y2) 的距离
    return Sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
}
```

```
int main()
{
    int x1,y1,x2,y2,x3,y3;
    cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
    cout << Distance(x1,y1,x2,y2) << endl;
    //输出(x1,y1)到 (x2,y2) 距离
    cout << Distance(x1,y1,x3,y3) << endl;
    cout << Distance(x3,y3,x2,y2) << endl;
    return 0;
}
```

返回值为void的函数

```
void DrawCircle(double x,double y,double r)
{
    //下面的代码在屏幕上以(x,y)点为圆心, r为半径画圆
    .....
    return;
}
```

=

调用:

```
DrawCircle(0,0,1);
```

函数的声明

- 一般来说函数的**定义**必须出现在函数调用语句之前，
否则调用语句编译出错

如果过函数A内部调用了B,B内部调用了A， 哪个写前面？

函数的声明

- 函数的调用语句前面有函数的**声明**即可，不一定要有定义！

返回值类型 函数名(参数1类型 参数1名称, 参数2类型 参数2名称.....);

例如:

```
int Max(int a,int b);  
double Sqrt(double);  
double Distance(double,double,double,double);
```

参数名称可以省略。函数声明也称为“函数的原型”

函数的声明

```
void FunctionB(); //声明
void FunctionA() {
    .....
    FunctionB();
    .....
    return;
}
void FunctionB() {
    .....
    FunctionA();
    .....
    return;
}
```

main函数

C/C++程序从main函数开始执行，执行到main中的return则结束。

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello,world" << endl;
    return 0;
}
```

函数参数的传递

- 函数的形参是实参的一个拷贝，且形参的改变不会影响到实参（除非形参类型是数组、引用 --- 引用的内容本课不涉及）

```
#include <iostream>
using namespace std;
void Swap(int a,int b)
{
    int tmp;
    //以下三行将a,b值互换
    tmp = a ;
    a = b;
    b = tmp;
    cout << "In Swap: a=" << a << " b=" << b << endl;
}
```

函数参数的传递

```
int main()
{
    int a = 4, b = 5;
    Swap(a,b);
    cout << "After swaping: a=" << a << " b=" << b;
    return 0;
}
```

In Swap: a=5 b=4

After swaping: a=4 b=5

一维数组作为函数的参数

- 一维数组作为形参时的写法如下：

类型名 数组名[]

不用写出数组的元素个数。例如：

```
void PrintArray( int a[ ] ) { }
```

- 数组作为函数参数时，是传引用的，即形参数组改变了，实参数组也会改变。

一维数组作为函数的参数

●编写一个求整型数组最大值的函数

```
#include <iostream>
using namespace std;
int a1[4] = {4,15,6,9};
int a2[] = {3,18,56,40,78};
int FindMax( int a[] ,int length)  { //length是数组长度
    int mx = a[0];
    for(int i = 1;i < length; ++i)
        if( mx < a[i])
            mx = a[i];
    return mx;
}
int main() {
    cout << FindMax(a1,sizeof(a1)/sizeof(int)) << endl;
    cout << FindMax(a2,sizeof(a2)/sizeof(int)) << endl;
    return 0;
}
```

一维数组作为函数的参数

●编写一个把int数组所有元素置0的函数

```
#include <iostream>
using namespace std;
int a1[4] = {4,15,6,9};
void SetToZero(int a[] ,int length)
{
    for(int i = 0;i < length; ++i)
        a[i] = 0;
}
int main()
{
    SetToZero(a1,4);
    for(int i = 0;i < 4; ++i)
        cout << a1[i] << "," ;
    return 0;
} => 0,0,0,0
```


二维数组作为函数的参数

- 二维数组作为形参时，必须写明数组有多少列，不用写明有多少行：

```
void PrintArray( int a[][5])  
{  
    cout << a[4][3];  
}
```

- 必须要写明列数，编译器才能根据下标算出元素的地址。

$a[i][j]$ 的地址：

数组首地址 + $i \times N \times \text{sizeof}(a[0][0]) + j \times \text{sizeof}(a[0][0])$ (N是数组列数)

形参数组的首地址就是实参数组的首地址

库函数和头文件

难道真的求平方根都要自己写一个? **No!!**

```
#include <iostream>
```

```
#include <cmath> // 头文件<cmath>中包含许多数学库函数
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    double a;
```

```
    cin >> a;
```

```
    if( a < 0 ) {
```

```
        cout << "Illegal input" << endl;
```

```
        return 0;
```

```
    }
```

```
    cout << sqrt(a); //调用标准库函数求平方根
```

```
    return 0;
```

```
}
```

库函数和头文件

库函数：C/C++标准规定的，编译器自带的函数

头文件：C++编译器提供许多“头文件”，如

```
iostream  
cmath  
string
```

头文件内部包含许多库函数的声明以及其他信息，如cin,cout的定义

```
#include <iostream>
```

即可将头文件包含到程序中，此后即可使用头文件中定义的库函数及其他信息

库函数和头文件示例

●数学函数

数学库函数声明在`cmath`中，主要有：

<code>int abs(int x)</code>	求整型数x的绝对值
<code>double cos(double x)</code>	求x(弧度)的余弦
<code>double fabs(double x)</code>	求浮点数x的绝对值
<code>int ceil(double x)</code>	求不小于x的最小整数
<code>double sin(double x)</code>	求x(弧度)的正弦
<code>double sqrt(double x)</code>	求x的平方根
.....	

库函数和头文件示例

●字符处理函数

这些库函数在ctype中声明，主要有：

int isdigit(int c)	判断 c 是否是数字字符
int isalpha(int c)	判断 c 是否是一个字母
int isalnum(int c)	判断 c 是否是一个数字或字母
int islower(int c)	判断 c 是否是一个小写字母
int islower(int c)	判断 c 是否是一个小写字母
int isupper(int c)	判断 c 是否是一个大写字母
int toupper(int c)	如果 c 是一个小写字母，则返回对应大写字母
int tolower (int c)	如果 c 是一个大写字母，则返回对应小写字母



北京大学
PEKING UNIVERSITY

信息科学技术学院

递归初步



甘肃麦积山石窟

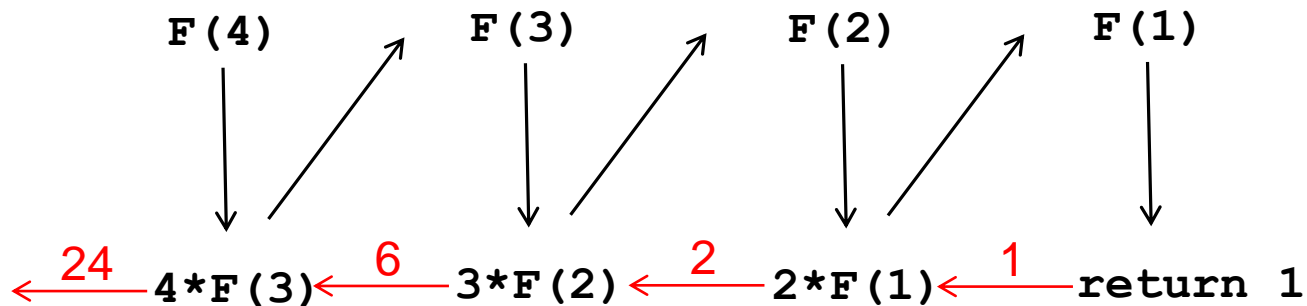
递归

- 一个函数，自己调用自己，就是递归

```
int Factorial(int n)
//函数返回n的阶乘
{
    if( n < 2)
        return 1; //终止条件
    else
        return n * Factorial(n-1);
} // cout << Factorial(5); ==> 120
```

- 递归函数需要有终止条件，否则就会无穷递归导致程序无法终止甚至崩溃

```
int F (int n) //函数返回n的阶乘
{
    if( n < 2)
        return 1; //终止条件
    else
        return n * F (n-1);
}
```



递归

□ 求斐波那契数列第 n 项

```
int Fib(int n)
{
    if( n == 1 || n == 2)
        return 1;
    else
        return Fib(n-1)+Fib(n-2);
}
```



北京大学
PEKING UNIVERSITY

信息科学技术学院

位运算



峨眉山金顶

基本概念

位运算:

用于对整数类型 (int,char, long 等) 变量中的**某一位**(bit), 或者**若干位**进行操作。比如:

- 1) 判断某一位是否为1
- 2) 只改变其中某一位, 而保持其他位都不变。

基本概念

位运算:

用于对整数类型 (int, char, long 等) 变量中的
某一位(bit), 或者**若干位**进行操作。比如:

- 1) 判断某一位是否为1
- 2) 只改变其中某一位, 而保持其他位都不变。

C/C++ 语言提供了**六种位运算符**来进行位运算操作:

&	按位与(双目)
 	按位或(双目)
^	按位异或(双目)
~	按位非(取反)(单目)
<<	左移(双目)
>>	右移(双目)

按位与“&”

将参与运算的两操作数各对应的二进制位进行与操作，只有对应的两个二进制位均为1时，结果的对应二进制位才为1，否则为0。

按位与“&”

例如：表达式 “21 & 18 ” 的计算结果是16
(即二进制数10000)， 因为：

21 用二进制表示就是：

0000 0000 0000 0000 0000 0000 0001 0101

18 用二进制表示就是：

0000 0000 0000 0000 0000 0000 0001 0010

二者按位与所得结果是：

0000 0000 0000 0000 0000 0000 0001 0000

按位与“&”

通常用来将某变量中的某些位清0且同时保留其他位不变。
也可以用来获取某变量中的某一位。

例如，如果需要将int型变量n的低8位全置成0，而其余位不变，则可以执行：

```
n = n & 0xffffffff00;
```

也可以写成：

```
n &= 0xffffffff00;
```

如果n是short类型的，则只需执行：

```
n &= 0xff00;
```

按位与“&”

如何判断一个int型变量n的第7位（从右往左，从0开始数）是否是1？

只需看表达式 “`n & 0x80`” 的值是否等于0x80即可。

0x80: 1000 0000

按位或“|”

将参与运算的两操作数各对应的二进制位进行或操作，只有对应的两个二进位都为0时，结果的对应二进制位才是0，否则为1。

按位或 “|”

例如：表达式 “21 | 18 ” 的值是23， 因为：

```
21:  0000 0000 0000 0000 0000 0000 0001 0101
18:  0000 0000 0000 0000 0000 0000 0001 0010
21|18: 0000 0000 0000 0000 0000 0000 0001 0111
```

按位或 “|”

按位或运算通常用来将某变量中的某些位置1且保留其他位不变。

例如，如果需要将int型变量n的低8位全置成1，而其余位不变，则可以执行：

```
n |= 0xff;
```

0xff: 1111 1111

按位异或 “^”

将参与运算的两操作数各对应的二进制位进行异或操作，即只有对应的两个二进制位不相同，结果的对应二进制位才是1，否则为0。

例如：表达式 “21 ^ 18 ” 的值是7(即二进制数111)。

21: 0000 0000 0000 0000 0000 0000 0001 0101

18: 0000 0000 0000 0000 0000 0000 0001 0010

21^18: 0000 0000 0000 0000 0000 0000 0000 0111

按位异或 “^”

按位异或运算通常用来将某变量中的某些位取反，且保留其他位不变。

例如，如果需要将int型变量n的低8位取反，而其余位不变，则可以执行：

```
n ^= 0xff;
```

0xff: 1111 1111

按位异或 “^”

异或运算的特点是：

如果 $a \oplus b = c$ ，那么就有 $c \oplus b = a$ 以及 $c \oplus a = b$ 。(穷举法可证)

此规律可以用来进行最简单的加密和解密。

按位异或 “^”

另外异或运算还能实现不通过临时变量，就能交换两个变量的值：

```
int a = 5, b = 7;  
a = a ^ b;  
b = b ^ a;  
a = a ^ b;
```

即实现a,b值交换。穷举法可证。

按位非 “~”

按位非运算符 “~” 是单目运算符。
其功能是将操作数中的二进制位0变成1，1变成0。

例如，表达式 “~21” 的值是整型数 0xfffffea

21: 0000 0000 0000 0000 0000 0000 0001 0101

~21: 1111 1111 1111 1111 1111 1111 1110 1010

左移运算符 “<<”

表达式：

$a \ll b$

的值是：将a各二进位全部左移b位后得到的值。左移时，高位丢弃，低位补0。a 的值不因运算而改变。

左移运算符 “<<”

例如:

$9 << 4$

9的二进制形式:

0000 0000 0000 0000 0000 0000 0000 1001

因此, 表达式 “ $9 << 4$ ” 的值, 就是将上面的二进制数左移4位, 得:

0000 0000 0000 0000 0000 0000 1001 0000

即为十进制的144。

左移运算符 “<<”

实际上，左移1位，就等于是乘以2，左移n位，就等于是乘以 2^n 。而左移操作比乘法操作快得多。

右移运算符 “>>”

表达式：

`a >> b`

的值是：将a各二进制位全部右移b位后得到的值。右移时，移出最右边的位就被丢弃。a 的值不因运算而改变。

对于有符号数，如long,int,short,char类型变量，在右移时，符号位（即最高位）将一起移动，并且大多数C/C++编译器规定，**如果原符号位为1，则右移时高位就补充1**，原符号位为0，则右移时高位就补充0。

右移运算符 “>>”

实际上，右移 n 位，就相当于左操作数除以 2^n ，并且将结果往小里取整。

$$-25 \gg 4 = -2$$

$$-2 \gg 4 = -1$$

$$18 \gg 4 = 1$$

```
#include <stdio.h>
int main()
{
    int n1 = 15;
    short n2 = -15;
    unsigned short n3 = 0xffe0;
    char c = 15;
    n1 = n1>>2;
    n2 >>= 3;
    n3 >>= 4;
    c >>= 3;
    printf( "n1=%d,n2=%x,n3=%x,c=%x",n1,n2,n3,c);
} //输出结果是: n1=3,n2=fffffffe,n3=ffe,c=1
// n2被自动转换成int输出
```

n1: 0000 0000 0000 0000 0000 0000 0000 1111

n1 >>= 2: 变成3

0000 0000 0000 0000 0000 0000 0000 0011

n2: 1111 1111 1111 0001

n2 >>= 3: 变成 fffe, 即-2

1111 1111 1111 1110

n3: 1111 1111 1110 0000

n3 >>= 4: 变成 ffe

0000 1111 1111 1110

c: 0000 1111

c >>= 3; 变成1

0000 0001

int n1 = 15;

short n2 = -15;

unsigned short n3 = 0xffe0;

char c = 15;

思考题：

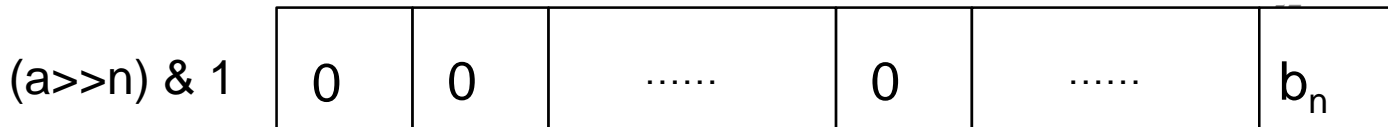
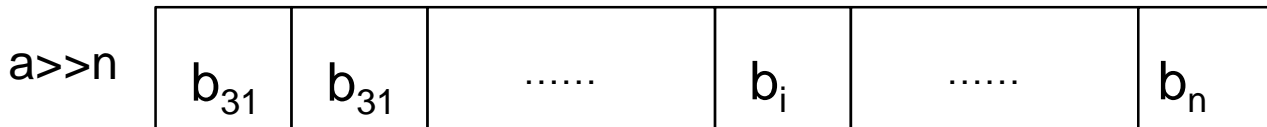
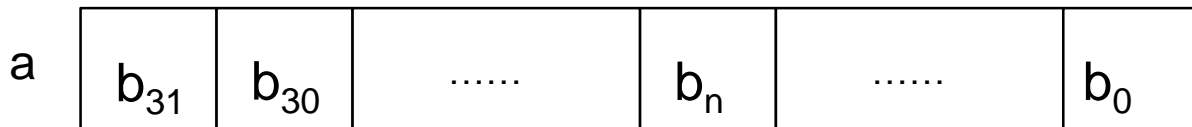
有两个int型的变量a和n($0 \leq n \leq 31$),
要求写一个表达式, 使该表达式的值和a的第n位相同。

思考题:

有两个int型的变量a和n($0 \leq n \leq 31$),
要求写一个表达式, 使该表达式的值和a的第n位相同。

答案:

$(a \gg n) \& 1$



思考题：

有两个int型的变量a和n($0 \leq n < 31$),
要求写一个表达式, 使该表达式的值和a的第n位相同。

另一答案：

$(a \ \& \ (1 \ll n)) \gg n$