



# 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

# 条件分支结构 之 if 语句



冰岛公路边冰川

# 条件分支结构

有时，并非所有的程序语句都要被顺序执行到，会希望满足某种条件就执行这部分语句，满足另一条件就执行另一部分语句。这就需要“条件分支结构”

# if 语句

```
if (表达式1) {  
    语句组1  
}  
else if(表达式2) {  
    语句组2  
}  
..... //可以有多个 else if  
else if( 表达式n-1) {  
    语句组n-1  
}  
else {  
    语句组n  
}
```

依次计算表达式1、表达式2...只要碰到一个表达式i为真，则执行语句组i（前面为假的表达式对应的语句组不会被执行），后面的表达式不再计算，后面的语句组也都不会被执行。

若所有表达式都为假，则执行语句组n

# if 语句

可以没有 else if, 也可以没有 else, 也可以都没有

```
if (表达式1) {  
    语句组1  
}  
else {  
    语句组2  
}
```

```
if (表达式1) {  
    语句组1  
}  
else if(表达式2) {  
    语句组2  
}
```

```
if (表达式1) {  
    语句组1  
}
```

# if 语句

如果“语句组”只有一条语句，可以不用{ }

```
if (n > 4)
    printf("%d",n) ;
```

# if 语句

**例题:** 写一个判断整数奇偶性的程序, 要求输入一个整数, 如果是奇数, 就输出 "It's odd.", 如果是偶数, 就输出 "It's even." 。

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    scanf("%d", &n);
    if( n % 2 == 1)
        printf("It's odd.\n") ;
    else
        printf("It's even.\n") ;
    return 0;
}
```



# if 语句

**例题:** 写一个判断整数奇偶性的程序, 要求输入一个整数, 如果是奇数, 就输出 "It's odd.", 如果是偶数, 就输出 "It's even."。

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    scanf("%d", &n);
    if( n % 2 )
        printf("It's odd.\n") ;
    else
        printf("It's even.\n") ;
    return 0;
}
```

# if 语句嵌套

在一条if语句的某个分支(语句组) 里, 还可以再写if语句。

```
#include <iostream>
using namespace std;
int main()
{
    int a;
    scanf("%d",&a);
    if( a > 0)
        if ( a % 2 )
            cout << "good";
    else //这个else到底和哪个if配对?
        cout << "bad";
    return 0;
} //输入-1, 输出?
```

# if 语句嵌套

else总是和离它最近的if配对

```
#include <iostream>
using namespace std;
int main()
{
    int a;
    scanf("%d",&a);
    if( a > 0)
        if ( a % 2 )
            cout << "good";
        else
            cout << "bad";
    return 0;
} 输入-1, 程序无输出!!!
```

# if 语句嵌套

else总是和离它最近的if配对

```
#include <iostream>
using namespace std;
int main()
{
    int a;
    scanf("%d",&a);
    if( a > 0)
        if ( a % 2 )
            cout << "good";
        else
            cout << "bad";
    return 0;
} 输入-1, 程序无输出!!!
```

```
int main()
{
    int a;
    scanf("%d",&a);
    if( a > 0) {
        if ( a % 2 )
            cout << "good";
    }
    else
        cout << "bad";
    return 0;
} 输入-1,输出 bad
```

## if 语句嵌套

**例题:**请写一个程序，该程序输入一个年份，根据该年份是否是建国整十周年、建党整十周年以及是否是闰年给出不同的输出。

```
#include <iostream>
using namespace std;
int main()
{
    int year;
    scanf("%d",& year);
    if( year <= 0)
        printf("Illegal year.\n") ;
    else {
        printf("Legal year.\n");
        if( year > 1949 && (year - 1949) % 10 == 0 )           //建国整十
            printf("Luky year.\n");
        else if( year > 1921 && !((year - 1921) % 10))        //建党整十
            printf("Good year.\n");
        else if( year % 4 == 0 && year % 100 || year % 400 == 0 )
            printf("Leap year\n"); //闰年
        else printf("Common year.\n");
    }
    return 0;
}
```

-2✓

Illegal year.

1959✓

Legal year.

Luky year.

1931✓

Legal year.

Good year.

2008✓

Legal year.

Leap year.

2011✓

Legal year.

Common

year.

```
#include <iostream>
using namespace std;
int main()
{
    int year;
    scanf("%d",& year);
    if( year <= 0)
        printf("Illegal year.\n") ;
    else {
        printf("Legal year.\n");
        if( year > 1949 && (year - 1949) % 10 == 0 )
            printf("Luky year.\n");
        else if( year > 1921 && !((year - 1921) % 10))
            printf("Good year.\n");
        else if(year%4 == 0 && year % 100 || year % 400 == 0 )
            printf("Leap year\n"); //闰年
        else printf("Common year.\n");
    }
    return 0;
}
```

## if 语句常见错误

```
int a = 0;  
if( a = 0)  
    printf("hello");  
if( a = 5 )  
    printf("Hi");
```



## if 语句常见错误

```
int a = 0;  
if( a = 0 ) //a = 0的值是0  
    printf("hello");  
if( a = 5 ) // a = 5的值是5  
    printf("Hi");
```

=> Hi

# if 语句常见错误

- 互相矛盾的多个条件，如果确实只希望执行其中一个分支，应该用if和多个else if，而不要写多个if

```
int a = 0;
if( a >=0 && a < 5 )
    a = 8;
else if( a >= 5 && a < 10 )
    cout << "hello";
else if( a > 10 && a < 20)
    .....
else
    .....
```

不会输出 hello

## 错误写法:

```
int a = 0;
if( a >=0 && a < 5 )
    a = 8;
if( a >= 5 && a < 10 )
    cout << "hello";
if( a > 10 && a < 20)
    .....
if( a >= 20)
    .....
会输出 hello
```



北京大学  
PEKING UNIVERSITY

信息科学技术学院《程序设计与算法》

# 条件分支结构 之 switch语句



俯瞰冰岛雷克雅未克

# switch语句

```
if( n % 5 == 0 ) {  
    .....  
}  
else if( n % 5 == 1 ) {  
    .....  
}  
else if( n % 5 == 2 ) {  
    .....  
}  
else if( n % 5 == 3 ) {  
    .....  
}  
else {  
    .....  
}
```

- 太多的else if 不方便
- $n \% 5$ 多次计算，浪费

# switch语句

```
switch (表达式) { //表达式的值 必须是整数类型 (int,char .....)  
    case 常量表达式1: //常量表达式必须是整数类型的常量 (int,char...)  
        语句组1  
        break;  
    case 常量表达式2:  
        语句组2  
        break;  
    .....  
    case 常量表达式n:  
        语句组n  
        break;  
    default:  
        语句组n+1  
}
```

“表达式” 的值等于哪个 “常量表达式” ,  
就执行相应的语句组。都不相等, 则执行  
default的语句组。也可以没有default分支  
“常量表达式” 里面不能包含变量!

# switch语句

**例题:** 请写一个程序，接受一个整数作为输入，如果输入1，则输出 “Monday”，输入2，则输出 “Tuesday” .....输入7,则输出 “Sunday”，输入其他数，则输出 “Illegal”。

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    scanf("%d",& n);
    switch(n) {
        case 1:
            printf("Monday");
            break;
        case 2:
            printf("Tuesday");
            break;
        case 3:
            printf("Wednesday");
            break;
        case 4:
            printf("Thursday");
            break;
```

```
    case 5:
        printf("Friday");
        break;
    case 6:
        printf("Saturday");
        break;
    case 7:
        printf("Sunday");
        break;
    default:
        printf("Illegal");
}
return 0;
}
```



## switch语句

switch语句在进入某个case分支后，会一直执行到第一个碰到的“break;”，即使这个“break;”是在后面的case分支里面。如果没有碰到“break;”，则会向下一直执行到switch语句末尾的“}”，包括“default:”部分的语句组也会被执行。

```
#include <iostream>
using namespace std;
int main()    {
    int n;
    scanf("%d",&n);
    switch(n%6) {
        case 0:
            printf( "case 0" );
            break;
        case 1:
            printf( "case 1" );
        case 2:
        case 3:
            printf( "case 2 or 3" );
            break;
        case 4:
            printf( "case 4" );
            break;
    }
    return 0;
}
```

1✓  
case 1  
case 2 or 3

2✓  
case 2 or 3

3✓  
case 2 or 3



北京大学  
PEKING UNIVERSITY

循环结构  
之  
for循环



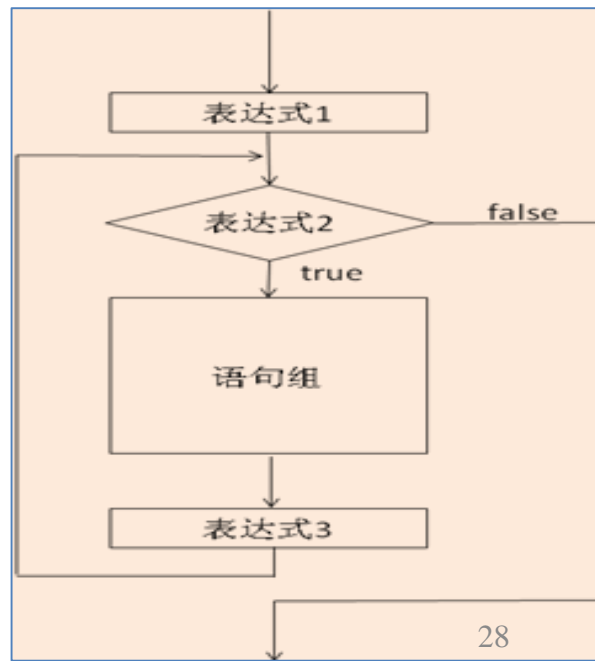
冰岛黛提瀑布

# for循环语句

```
for( 表达式1 ;表达式2;表达式3) {  
    语句组  
}
```

- 1) 计算“表达式1”。
- 2) 计算“表达式2”，若其值为true，则执行“{ }”中的语句组，然后转到3)；若为false,则不再执行“{ }”中的语句组，for语句结束，转到5)。
- 3) 计算“表达式3”。
- 4) 转到2)。
- 5) 从for语句后面继续往下执行程序。

**一般用于将某段代码(语句组)重复执行若干次!!!**



# for循环语句

## 例：连续输出26个字母

```
int i;
for( i = 0; i < 26; ++i ) {
    cout << char('a'+i); // 'a'+i强制转换成char类型
}
```

循环控制变量

或

```
for( int i = 0; i < 26; ++i ) //语句组里只有一条一句就可以不用写" { } "
    printf("%c", 'a'+i);
```

循环控制变量

=> abcdefghijklmnopqrstuvwxyz

# for循环语句

循环控制变量定义在"表达式1"中, 则其只在for语句内部起作用, 可以不用担心循环控制变量重名

```
int i = 5;
for( int i = 0; i < 26; ++i )
    cout << char('a'+i );
cout << endl;
for( int i = 0; i < 26; i+=2 ){ //循环控制变量并非每次只能加1
    cout << char('A'+i ) ;
}
cout << endl;
cout << i; //此处的i和for里面的i无关
```

```
abcdefghijklmnopqrstuvwxyz
ACEGIKMOQSUY
5
```

# for循环语句

for循环结构里的“表达式1”和“表达式3”都可以是用逗号连接的若干个表达式。

```
for( int i= 15, j = 0; i > j; i-=2, j+= 3)  
    cout << i << ", " << j << endl;
```

15,0

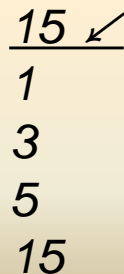
13,3

11,6

# for循环语句

**例题:** 写一个程序, 输入一个正整数n, 从小到大输出它的所有因子

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    cin >> n;
    for( int i = 1; i <= n; ++i)
        if( n % i == 0 )
            cout << i << endl;
    return 0;
}
```



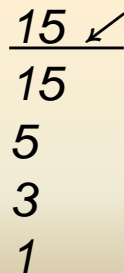
15 ✓  
1  
3  
5  
15



# for循环语句

**例题:** 写一个程序, 输入一个正整数n, 从大到小输出它的所有因子

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    cin >> n;
    for( int i = n; i >= 1; --i)
        if( n % i == 0 )
            cout << i << endl;
    return 0;
}
```



15 ✓  
15  
5  
3  
1

# for循环语句

for循环可以嵌套，形成多重for循环：

```
for(int i = 0; i < n; ++ i) {  
    .....  
    for(int j = 0; j < m; ++j ) {  
        ..... //内重循环的执行次数一共是 $n \times m$ 次  
    }  
    .....  
}
```

## for循环语句

**例题：**给定正整数 $n$ 和 $m$ ,在1至 $n$ 这 $n$ 个数中，取出两个不同的数，使得其和是 $m$ 的因子，问有多少种不同的取法。

**思路：**穷举1- $n$ 这 $n$ 个数中取两个数的所有取法，对每一种取法，判断其和是不是 $m$ 的因子

## for循环语句

**例题：**给定正整数 $n$ 和 $m$ ,在1至 $n$ 这 $n$ 个数中，取出两个不同的数，使得其和是 $m$ 的因子，问有多少种不同的取法。

穷举的办法：

第一个数取1，第二个数分别取2,3,... $n$

第一个数取2，第二个数分别取3,4,... $n$

....

第一个数取 $n-2$ ,第二个数分别取 $n-1,n$

第一个数取 $n-1$ ,第二个数取 $n$

## for循环语句

```
#include <iostream>
using namespace std;
int main()
{
    int n,m;
    int total = 0;    //取法总数
    cin >> n >> m;
    for( int i = 1; i < n; ++i )    { //取第一个数, 共n-1种取法
        for( int j = i + 1; j <= n; ++j ) //第二个数要比第
            //一个数大, 以免取法重复
            if( m % (i + j) == 0 )
                ++ total ;
    }
    cout << total;
    return 0;
}
```

## for循环语句

for 语句括号里面的“表达式1”，“表达式2”，“表达式3”  
任何一个都可以不写，甚至可以全都不写，但是“;”必须保留。

```
for(    ; i < 100; ++ i )    //假设i在for前已经有合理值
    cout << i ;
```

```
for( ; ; )
    cout << "hello" <<endl;    //永远不停输出 hello
```

可以用 break 语句从 for(;;)死循环中跳出



北京大学  
PEKING UNIVERSITY

循环结构  
之  
while循环  
do...while循环



冰岛天然玄武岩长城

# while循环

并非到达指定次数，而是满足某条件时即停止循环，则适合用while语句来实现循环

```
while(表达式) {  
    语句组  
}
```

- 1) 判断“表达式”是否为真，如果不为真，则转4)
- 2) 执行“语句组”
- 3) 转1)
- 4) while语句结束，继续执行while语句后面的语句。



# while循环

并非到达指定次数，而是满足某条件时即停止循环，则适合用while语句来实现循环

```
while(表达式) {  
    语句组  
}
```

```
while(true) {  
    语句组  
} //死循环，可以用break跳出
```

- 1) 判断“表达式”是否为真，如果不为真，则转4)
- 2) 执行“语句组”
- 3) 转1)
- 4) while语句结束，从while语句后面的语句继续执行。

# while循环

**例题:**输入若干个(至少1个) 不超过100的正整数, 输出其中的最大值、最小值以及所有数的和。输入的最后一个数是0, 标志着输入结束。

```
#include <iostream>
using namespace std;
int main()
{
    int sum = 0, maxN = 0, minN = 200, n;
    cin >> n;
    while( n ) {
        if( n > maxN)        maxN = n;
        if( n < minN)        minN = n;
        sum += n;
        cin >> n;
    }
    cout << maxN << " " << minN << " " << sum;
    return 0;
}
```

5 8 2 7 0 ✓

8 2 22

12 43 11 98 47 34 0 ✓

98 11 245

1 0 ✓

1 1 1

# while循环

**例题:**用牛顿迭代法求输入的数的平方根。

欲求a的平方根，首先猜测一个值 $x_1 = a/2$ （也可以是随便什么其他值）作为其平方根，然后根据下面的迭代公式算出 $x_2$ ，再将 $x_2$ 代入公式右边算出 $x_3$ .....直到连续两次算出的 $x_n$ 和 $x_{n+1}$ 的差的绝对值小于某个值 $\epsilon$ ，即认为找到了足够精确的平方根。这个 $\epsilon$ 值取得越小，计算出来的平方根就越精确。

$$\text{迭代公式: } x_{n+1} = (x_n + a / x_n) / 2$$

```
#include <iostream>
using namespace std;
double EPS = 0.001; //用以控制计算精度
int main()
{
    double a;
    cin >> a ; //输入a,要求a的平方根
    if( a >= 0) {
        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){ //只要精度未达要求,
                                                    //就继续迭代

            lastX = x;
            x = (x + a/x)/2;
        }
        cout << x;
    }
    else
        cout << "It can't be nagitive.";
    return 0;
}
```

```

#include <iostream>
using namespace std;
double EPS = 0.001; //用以控制计算精度
int main()
{
    double a;
    cin >> a ; //输入a,要求a的平方根
    if( a >= 0) {
        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){ //只要精度未达要求,
                                                    //就继续迭代

            lastX = x;
            x = (x + a/x)/2;
        }
        cout << x;
    }
    else
        cout << "It can't be nagitive.";
    return 0;
}

```

输入 2时, EPS取值	输出结果
1	1.5
0.1	1.41667
0.01	1.41422
0.001	1.41421
0.0001	1.41421

# do...while循环

如果希望循环至少要执行一次，  
就可以使用do...while语句

```
do {  
    语句组  
} while (表达式);
```

每执行一次循环后，都要判断“表达式”的值是否为真，如果真就继续循环，如果为假，就停止循环。

## do...while循环

输出1到10000以内所有2的整数次幂:

```
int n = 1;
do {
    cout << n << endl;
    n *= 2;
} while( n < 10000);
```