Operator,express,flow of control

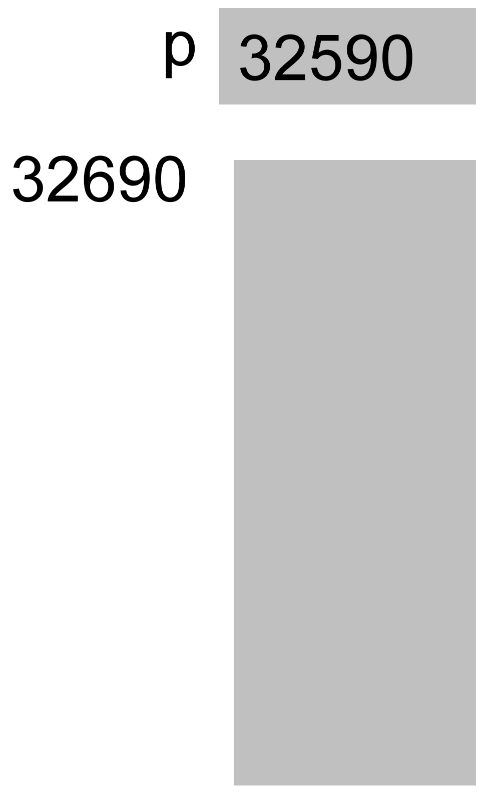
董洪伟

http://hwdong.com

常量和变量

- 数据有常量和变量。变量是一个命名的内存块

```
const int a = 10;
const char *str = “hello world”;
int ints[ ] = {10,20,30,40};
Int *p = new int[10];
//指针变量存储一个10个整型数
//的内存块地址
```



运算符

- 运算符（操作符）对数据（操作数）进行各种运算操作。运算符有算术运算符、逻辑运算符、赋值运算符、位操作符等
- 表达式是由操作数、运算符或括号所组成的运算式。根据表达式结果的类型可以将它们分为算术表达式、关系/逻辑表达式、地址表达式等。如果表达式中的操作数在编译时就能确定为常量值，该表达式成为常量表达式。

```
    “hello”        35
const int a = 2,b=4;    a+b
```

运算符

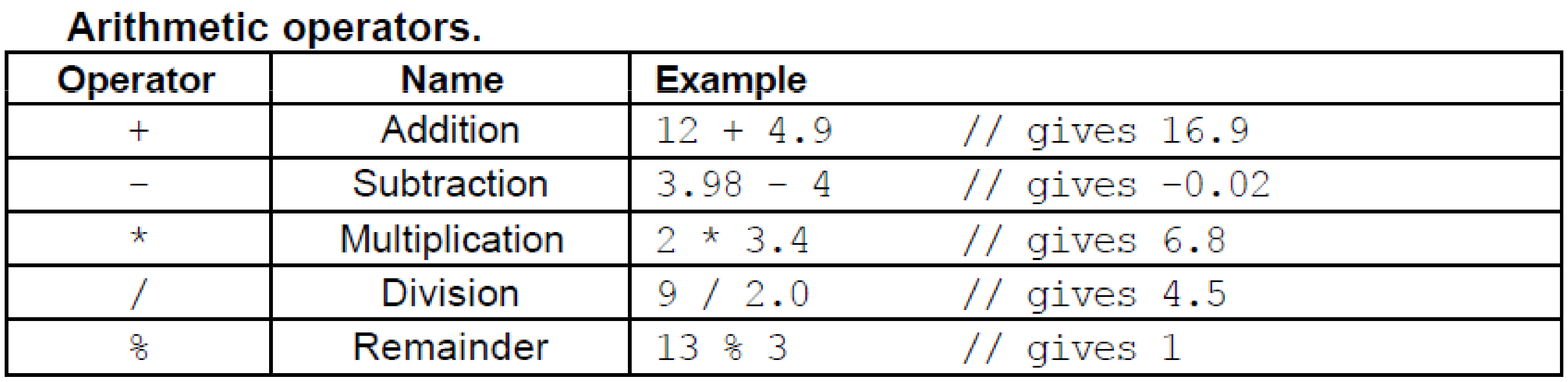
- 根据运算符所作用对象的个数，分为一元、二元、三元运算符。如

  ```
  a + b;   a++;   c = a>b ? a: b;
  ```

- 根据运算符的功能分为：算术运算符、关系运算符、逻辑运算符、位运算符、赋值运算符等

运算符-算术运算符

- C++提供5个基本的算术运算符



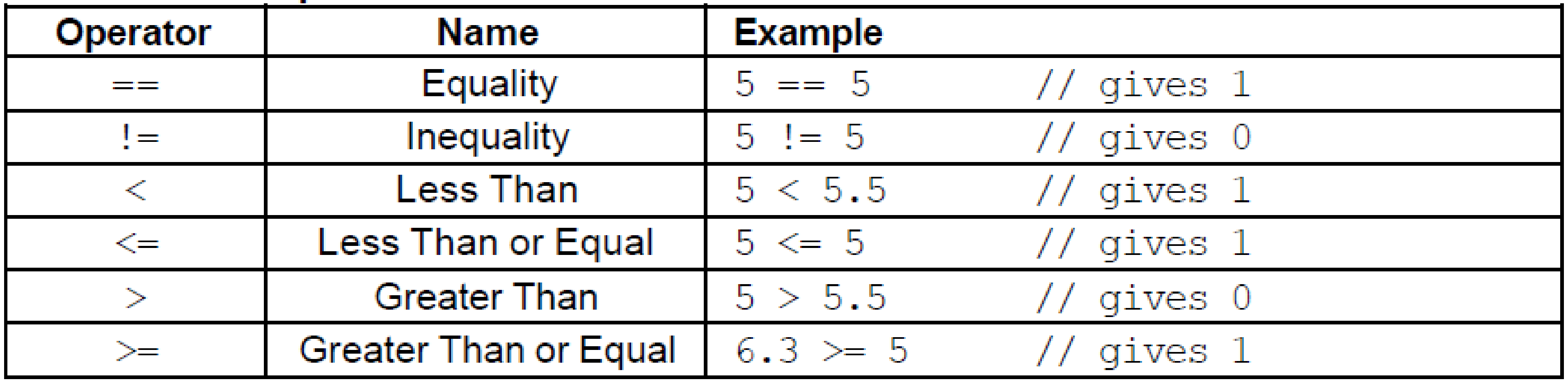
Arithmetic operators.

Operator	Name	Example
+	Addition	`12 + 4.9        // gives 16.9`
-	Subtraction	`3.98 - 4        // gives -0.02`
*	Multiplication	`2 * 3.4         // gives 6.8`
/	Division	`9 / 2.0         // gives 4.5`
%	Remainder	`13 % 3          // gives 1`

运算符-关系运算符

- C++提供6个关系运算符用于比较数值的。返回值1或0.因此关系运算符不能用于比较字符串
  “Hello” >”world”

Relational operators.



Operator	Name	Example
`==`	Equality	`5 == 5        // gives 1`
`!=`	Inequality	`5 != 5        // gives 0`
`<`	Less Than	`5 < 5.5       // gives 1`
`<=`	Less Than or Equal	`5 <= 5        // gives 1`
`>`	Greater Than	`5 > 5.5       // gives 0`
`>=`	Greater Than or Equal	`6.3 >= 5      // gives 1`

运算符-逻辑运算符

- 3个逻辑运算符用于比较逻辑表达式

Logical operators.

Operator	Name	Example	
`!`	Logical Negation	`!(5 == 5)`	`// gives 0`
`&&`	Logical And	`5 < 6 && 6 < 6`	`// gives 1`
`||`	Logical Or	`5 < 6 || 6 < 5`	`// gives 1`

- 非0数表示逻辑值true，而0表示逻辑值false

```
!20                 // gives 0
10 && 5             // gives 1
10 || 5.5           // gives 1
10 && 0             // gives 0
```

运算符-位运算符

- 6个位运算符用于处理整数中的单独的位。

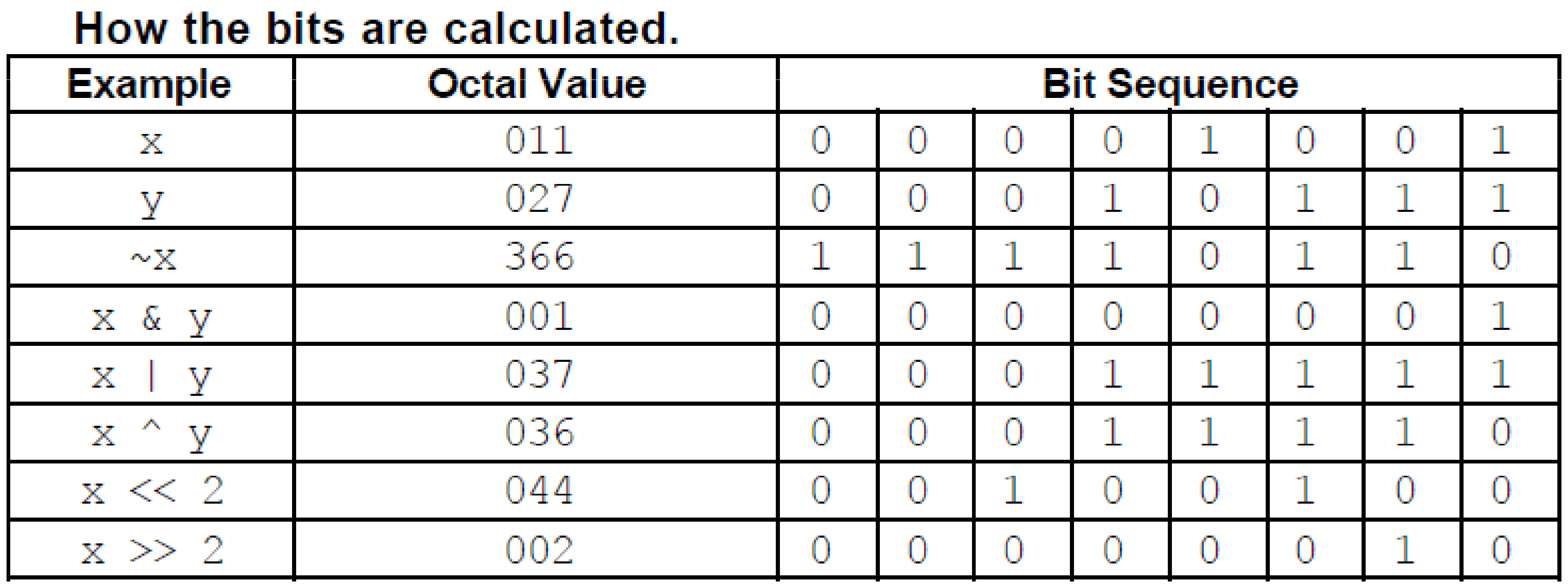
Bitwise operators.

Operator	Name	Example
~	Bitwise Negation	`~'\011'            // gives '\366'`
&	Bitwise And	`'\011' & '\027'     // gives '\001'`
|	Bitwise Or	`'\011' | '\027'     // gives '\037'`
^	Bitwise Exclusive Or	`'\011' ^ '\027'     // gives '\036'`
<<	Bitwise Left Shift	`'\011' << 2         // gives '\044'`
>>	Bitwise Right Shift	`'\011' >> 2         // gives '\002'`

运算符-位运算符

unsigned char x = ‘011’;

unsigned char y = ‘027’;



How the bits are calculated.

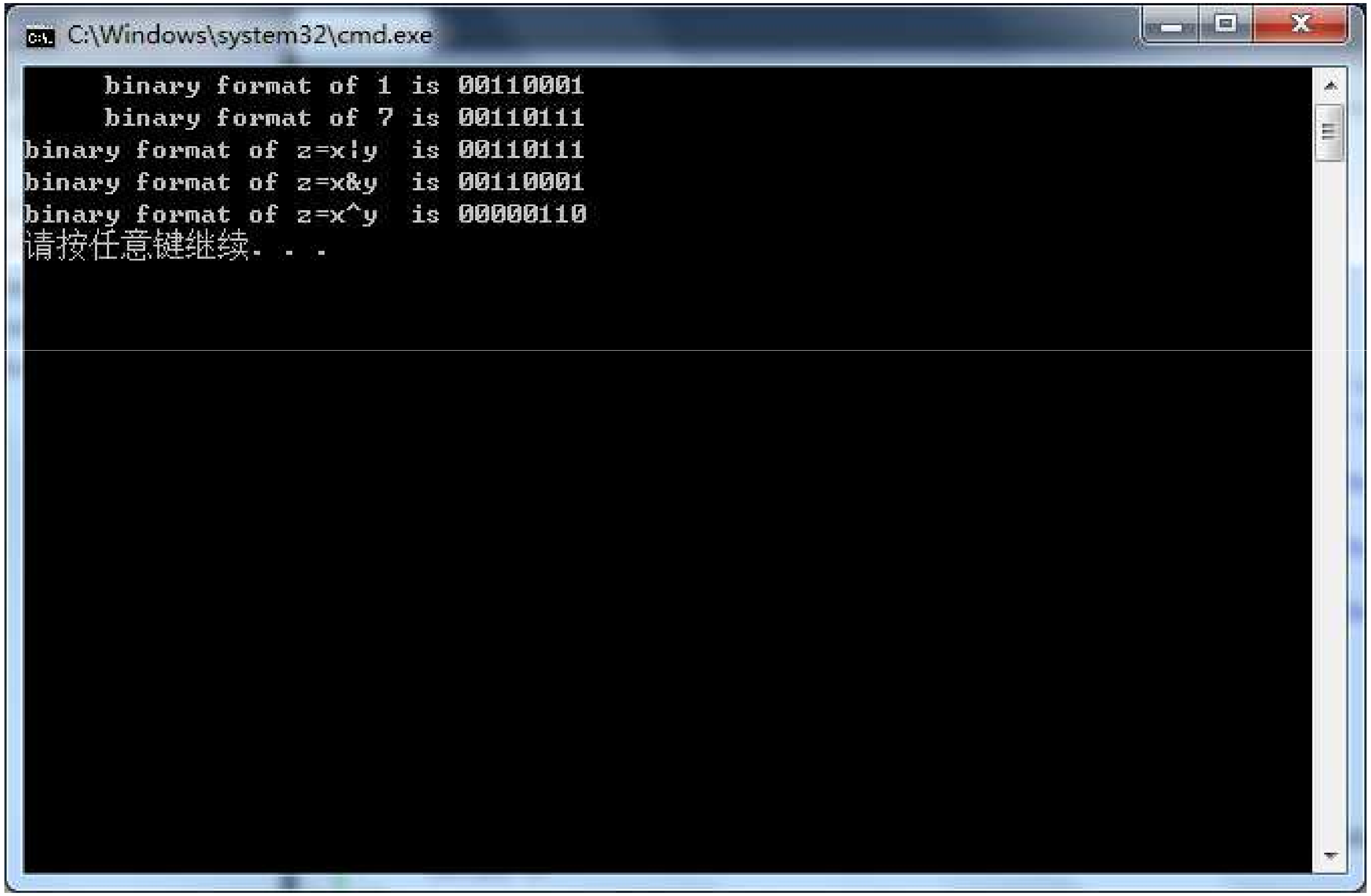
Example	Octal Value	Bit Sequence							
x	011	0	0	0	0	1	0	0	1
y	027	0	0	0	1	0	1	1	1
~x	366	1	1	1	1	0	1	1	0
x & y	001	0	0	0	0	0	0	0	1
x | y	037	0	0	0	1	1	1	1	1
x ^ y	036	0	0	0	1	1	1	1	0
x << 2	044	0	0	1	0	0	1	0	0
x >> 2	002	0	0	0	0	0	0	1	0

运算符-位运算符

```
#include <iostream>
#include <iomanip>
#include <bitset>
using namespace std;

int main(){
unsigned char x = '011';
unsigned char y = '027';
cout<<"     binary format of "<<x<<" is "<<bitset<sizeof(unsigned char)*8>(x)<<endl;
cout<<"     binary format of "<<y<<" is "<<bitset<sizeof(unsigned char)*8>(y)<<endl;
unsigned char z = x|y;
cout<<"binary format of z=x|y "<<" is "<<bitset<sizeof(unsigned char)*8>(z)<<endl;
z = x&y;
cout<<"binary format of z=x&y "<<" is "<<bitset<sizeof(unsigned char)*8>(z)<<endl;
z = x^y;
cout<<"binary format of z=x^y "<<" is "<<bitset<sizeof(unsigned char)*8>(z)<<endl;
return 0;
}
```

运算符-位运算符



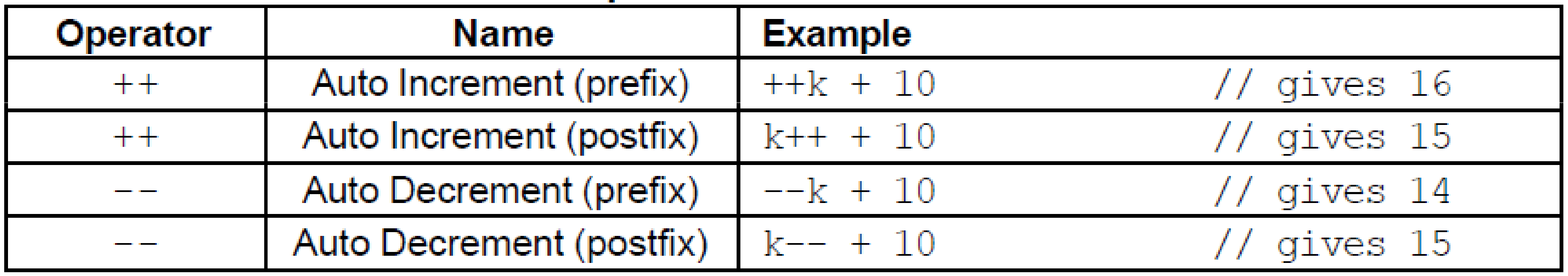
C:\Windows\system32\cmd.exe

```
     binary format of 1 is 00110001
     binary format of 7 is 00110111
binary format of z=x|y  is 00110111
binary format of z=x&y  is 00110001
binary format of z=x^y  is 00000110
请按任意键继续. . .
```

运算符-增量/减量运算符

int k=5;   k++; --k;

Increment and decrement operators.



Operator	Name	Example	
++	Auto Increment (prefix)	`++k + 10`	`// gives 16`
++	Auto Increment (postfix)	`k++ + 10`	`// gives 15`
--	Auto Decrement (prefix)	`--k + 10`	`// gives 14`
--	Auto Decrement (postfix)	`k-- + 10`	`// gives 15`

运算符-赋值运算符

- 用于将一个值存储在内存的某个位置（变量）

- 左边的操作数必须是一个左值(left Value)，右操作数可以是任意一个表达式。因此左值通常是一个变量，但也可能是引用或指针指向的内存。如

```
int sum = (a+b)*(b-a)/2;
bool f  = f1&&f2;
double d +=3;
n <<= 4;
```

运算符-赋值运算符

Assignment operators.

Operator	Example	Equivalent To
=	n = 25	
+=	n += 25	n = n + 25
-=	n -= 25	n = n - 25
*=	n *= 25	n = n * 25
/=	n /= 25	n = n / 25
%=	n %= 25	n = n % 25
&=	n &= 0xF2F2	n = n & 0xF2F2
|=	n |= 0xF2F2	n = n | 0xF2F2
^=	n ^= 0xF2F2	n = n ^ 0xF2F2
<<=	n <<= 4	n = n << 4
>>=	n >>= 4	n = n >> 4

运算符-赋值运算符

- 赋值运算符本身也是表达式，其值就是左值的值，因此可以继续用于其他表达式中

```
int m, n, p;
m = n = p = 100;          // means: n = (m = (p = 100));
m = (n = p = 100) + 2;    // means: m = (n = (p = 100)) + 2;


m = 100;
m += n = p = 10;          // means: m = m + (n = p = 10);
```

运算符-条件运算符 ?:

- 根据operand1的值是true或false，其值取operand2或operand3

$$operand1\ ?\ operand2\ :\ operand3$$

```
int m = 1, n = 2;
int min = (m < n ? m : n);        // min receives 1
```

运算符-逗号运算符,

- 其值取右操作数

  opdn1,opnd2

```
int m, n, min;
int mCount = 0, nCount = 0;
//...
min = (m < n ? mCount++, m : nCount++, n);
```

运算符-sizeof运算符

- 返回任何数据项或类型占用的空间，以字节为单位。

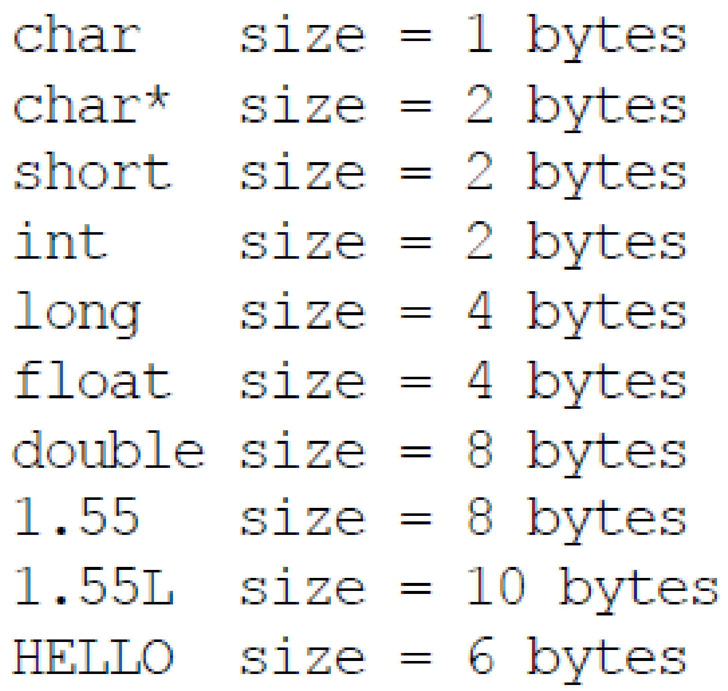
```
#include <iostream.h>

int main (void)
{
    cout << "char   size = " << sizeof(char) << " bytes\n";
    cout << "char*  size = " << sizeof(char*) << " bytes\n";
    cout << "short  size = " << sizeof(short) << " bytes\n";
    cout << "int    size = " << sizeof(int) << " bytes\n";
    cout << "long   size = " << sizeof(long) << " bytes\n";
    cout << "float  size = " << sizeof(float) << " bytes\n";
    cout << "double size = " << sizeof(double) << " bytes\n";

    cout << "1.55   size = " << sizeof(1.55) << " bytes\n";
    cout << "1.55L  size = " << sizeof(1.55L) << " bytes\n";
    cout << "HELLO  size = " << sizeof("HELLO") << " bytes\n";
}
```

运算符-sizeof运算符

- 返回任何数据项或类型占用的空间，以字节为单位。



```
char   size = 1 bytes
char*  size = 2 bytes
short  size = 2 bytes
int    size = 2 bytes
long   size = 4 bytes
float  size = 4 bytes
double size = 8 bytes
1.55   size = 8 bytes
1.55L  size = 10 bytes
HELLO  size = 6 bytes
```

运算符的优先级别

- 表达式中运算符执行的次序很重要，由运算符优先规则决定。
- 如 a == b + c*d 中运算符执行的次序是：

  *   +   ==

- 而 a== (b+c)*d 中运算符执行的次序是：

  +   *   ==

运算符的优先级别

Operator precedence levels.

Level	Operator						Kind	Order
Highest	::						Unary	Both
	()	[]	->	.			Binary	Left to Right
	+ -	++ --	! ~	* &	new delete	sizeof ()	Unary	Right to Left
	->*	.*					Binary	Left to Right
	*	/	%				Binary	Left to Right
	+	-					Binary	Left to Right
	<<	>>					Binary	Left to Right
	<	<=	>	>=			Binary	Left to Right
	==	!=					Binary	Left to Right
	&						Binary	Left to Right
	^						Binary	Left to Right
	|						Binary	Left to Right
	&&						Binary	Left to Right
	||						Binary	Left to Right
	? :						Ternary	Left to Right
	=	+= -=	*= /=	^= %=	&= |=	<<= >>=	Binary	Right to Left
Lowest	,						Binary	Left to Right

练习

- 写程序解决下列问题：

  判断一个数是否是奇数；

  判断一个字符是否是数字或字母；

  将long int 的第n位设为1；

  确定一个数的绝对值；

表达式和语句

- 表达式是运算符、运算数和( )构成的，有一个值
- 表达式后跟一个分号';'就构成一个最简单的语句。如

```
int a;
double d = 2.5;
```

- 控制语句不是表达式(因为没有值).如if、for、while、switch、go to等语句

复合语句

- 多个语句可以组合成一个符合语句，用花括号{ }包围。如

```
{  int min, i = 10, j = 20;
   min = (i < j ? i : j);
   cout << min << '\n';
}
```

复合语句

- 复合语句称为程序块，其中可以定义相应的局部变量，只存在于该程序块的作用域内
- 函数是一个命名的程序块

  ```
  int add (int a, int b) { return a+b; }
  ```
- 嵌套程序块：内部程序块的变量覆盖外部程序块的同名变量

复合语句

- 内部程序块的变量覆盖外部程序块的同名变量

```
int i = 10;
void main(){
    int i = 20;
    for(int i = 0; i<3; i++)
        std::cout << I << “ “;
    std::cout << “\n” << i << “ “::i << “\n”;
}
```

if语句

```
if (expression)
        statement;
```

如

```
if (count != 0)
    average = sum / count;
```

```
if (expression)
        statement1;
else
        statement2;
```

如

```
if (balance > 0) {
    interest = balance * creditRate;
    balance += interest;
} else {
    interest = balance * debitRate;
    balance += interest;
}
```

if语句-嵌套

```
if (ch >= '0' && ch <= '9')
    kind = digit;
else {
    if (ch >= 'A' && ch <= 'Z')
        kind = upperLetter;
    else {
        if (ch >= 'a' && ch <= 'z')
            kind = lowerLetter;
        else
            kind = special;
    }
}
```

if语句-嵌套

```
if (ch >= '0' && ch <= '9')
    kind = digit;
else if (cha >= 'A' && ch <= 'Z')
    kind = capitalLetter;
else if (ch >= 'a' && ch <= 'z')
    kind = smallLetter;
else
    kind = special;
```

if语句-嵌套

```
#include <iostream>
using namespace std;

int main() {
    int x = 6;
    int y = 0;

    if(x > y) {
        cout << "x is greater than y\n";
        if(x == 6)
            cout << "x is equal to 6\n";
        else
            cout << "x is not equalt to 6\n";
    } else
        cout << "x is not greater than y\n";

    return 0;
}
```

switch语句

```
switch (expression) {
    case constant_1:
        statements;
    ...
    case constant_n:
        statements;
    default:
        statements;
}
```

switch语句

- 假如要编写求解一个字符串表示的表达式的程序

```
switch (operator) {
    case '+':   result = operand1 + operand2;
                break;
    case '-':   result = operand1 - operand2;
                break;
    case '*':   result = operand1 * operand2;
                break;
    case '/':   result = operand1 / operand2;
                break;
    default:    cout << "unknown operator: " << ch << '\n';
                break;
}
```

switch语句

```
#include <iostream>
using namespace std;

int main() {
    int x = 6;
    cin>>x;
    switch(x) {
            case 1:
                    cout << "x is 1\n";
                    break;
            case 2:
            case 3:
                    cout << "x is 2 or 3";
                    break;
            default:
                    cout << "x is not 1, 2, or 3";
    }

    return 0;
}
```

while语句

- 根据表达式的值是否为true,执行*statement*

```
while (expression)
      statement;
```

如

```
i = 1;
sum = 0;
while (i <= n)
      sum += i++;
```

While loop trace.

Iteration	**i**	**n**	**i <= n**	**sum += i++**
First	1	5	1	1
Second	2	5	1	3
Third	3	5	1	6
Fourth	4	5	1	10
Fifth	5	5	1	15
Sixth	6	5	0	

do语句

```
do
    statement;
while (expression);
```

如

```
do {
    cin >> n;
    cout << n * n << '\n';
} while (n != 0);
```

for语句

```
for (expression1; expression2; expression3)
    statement;
```

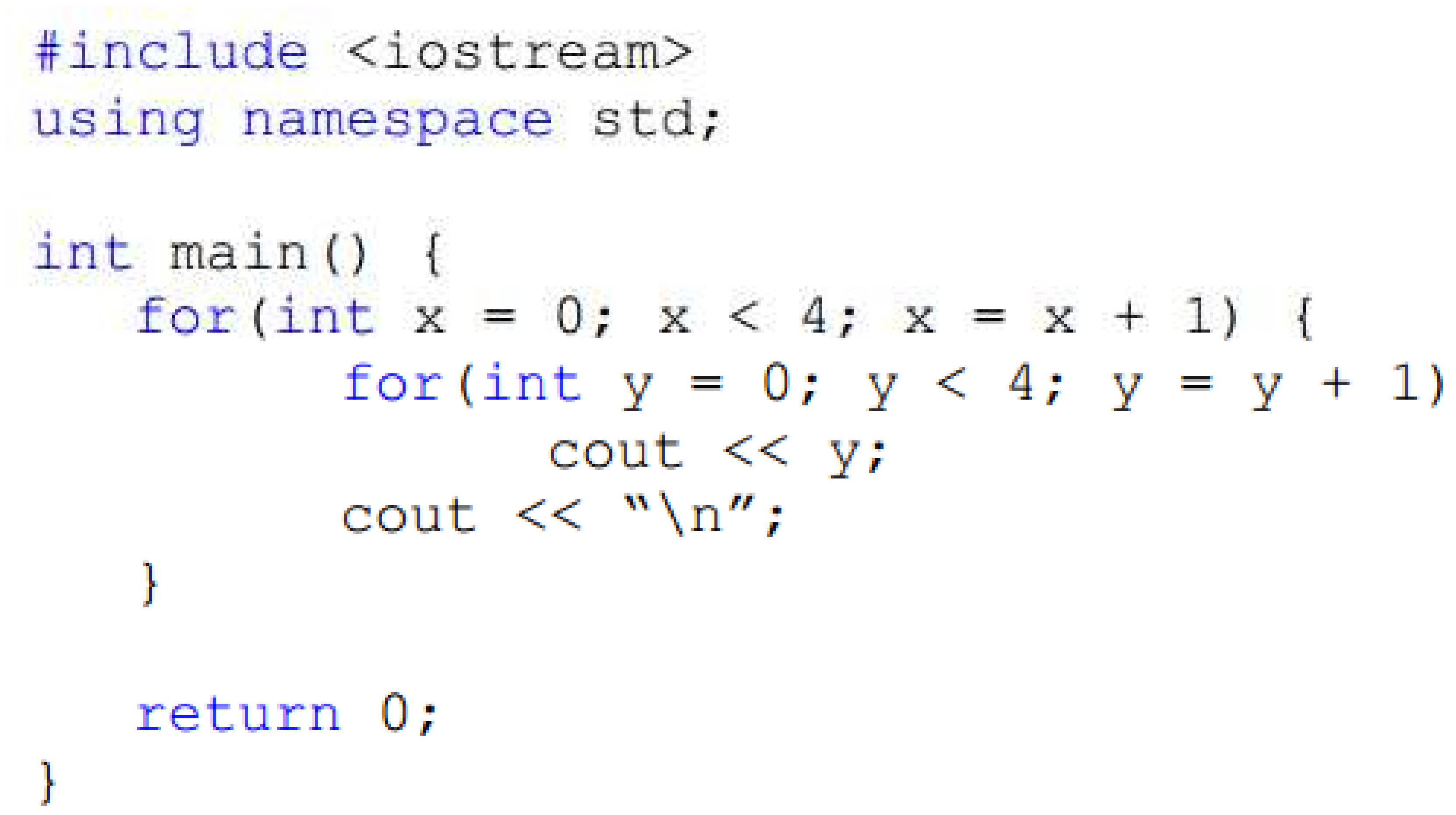
等价于

```
expression1;
while (expression2) {
    statement;
    expression3;
}
```

如

```
sum = 0;
for (i = 1; i <= n; ++i)
    sum += i;
```

for语句



```
#include <iostream>
using namespace std;

int main() {
    for(int x = 0; x < 4; x = x + 1) {
          for(int y = 0; y < 4; y = y + 1)
                cout << y;
          cout << "\n";
    }

    return 0;
}
```

for语句

```
for (;;) // infinite loop
   something;
```

continue语句

```
for (int i = 0; i < n; ++i) {
    cin >> num;
    if (num < 0) continue; // causes a jump to: ++i
    // process num here...
}
```

break语句

```
for (int i = 0; i < attempts; ++i) {
    cout << "Please enter your password: ";
    cin >> password;
    if (Verify(password)) // check password
        break; // drop out of the loop
     cout << "Incorrect!\n";
}
```

goto语句

```
for (int i = 0; i < attempts; ++i) {
    cout << "Please enter your password: ";
    cin >> password;
    if (Verify(password)) // check password
        goto out; // drop out of the loop
    cout << "Incorrect!\n";
}
out:
//etc...
```

return语句

- 使一个函数返回值给它的调用者。格式

  return expression;
- 如：

  return; //用空表达式返回空值

  return a; //返回一个变量的值给调用者

练习

- 用牛顿法求解$\sqrt[3]{a}$.
- 解：牛顿迭代公式为：

$$x_{n+1} = \frac{1}{3}(2x_n + \frac{a}{x_n{}^2})$$

当$|x_{n+1} - x_n| < \varepsilon$ 时，收敛到解。

练习

1.输入是“dd/mm/yy”格式的日期，输出是“month dd, year”格式的日期。

2. 输入一个整数，输出如下的函数值：

   factorial(0) = 1

   factorial(n) = n × factorial(n-1)

3.写一个程序，输出整数范围1-9之间整数相乘的乘法表

   1 x 1 = 1

   1 x 2 = 2

   …

   9 x 9 = 81