



# 第四章 程序设计基本结构

## 模块4.2：分支语句和逻辑运算符

主讲教师：同济大学电子与信息工程学院 陈宇飞  
同济大学电子与信息工程学院 龚晓亮



# 目录

- if语句
- 逻辑表达式
- 字符函数库cctype
- ?:运算符
- switch语句
- break和continue语句



# 目录

- if语句

1. if 语句

2. if else语句

3. 格式化if else语句

4. if else if else结构



# 1.1 if语句

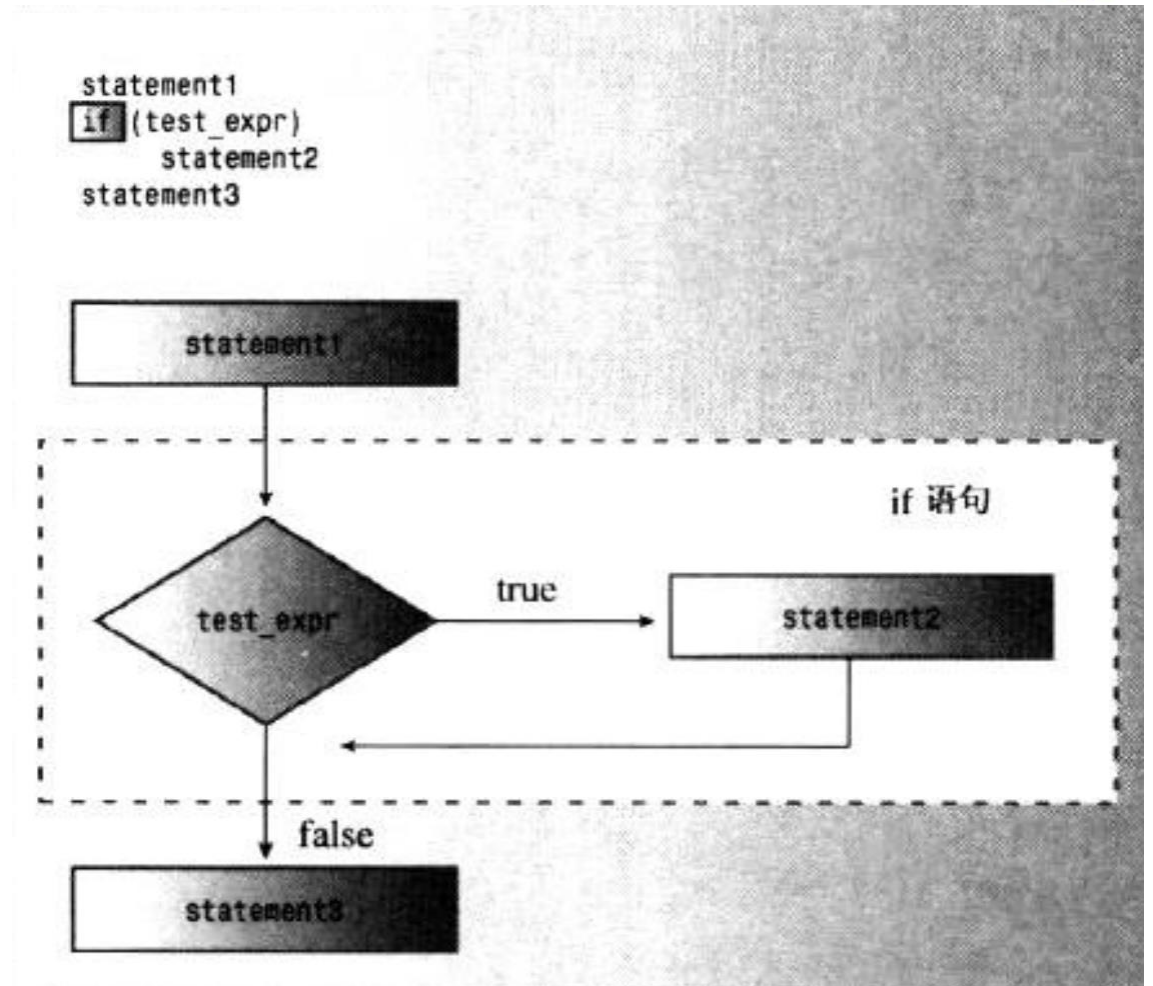
- ✓当C++程序必须决定是否执行某个操作时，通常使用if语句来实现选择
- ✓if有两种格式：if 和 if else

# 1.1 if语句

## ✓单支语句

```
if (test-condition)  
    statement
```

- 当表达式为真时执行语句序列，  
为假则不执行
- 表达式可以是任意类型，但按逻辑值求解 (非0为真0为假)





# 1.1 if语句

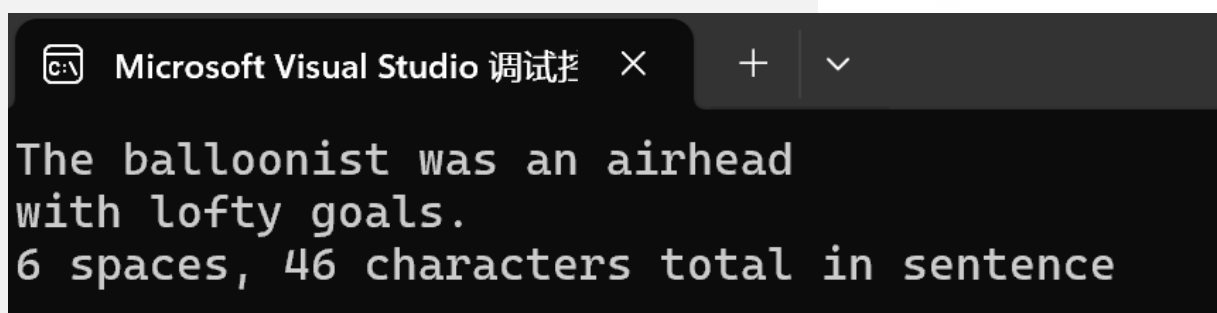
## ✓单支语句

□表达式后**无**; (表达式和表达式语句的区别)

□整个单支语句可以作为一个语句来看待(**复合语句**)

```
if (表达式;)    //编译错  
    语句序列;
```

```
// if.cpp -- using the if statement
#include <iostream>
int main()
{
    using std::cin;        // using declarations
    using std::cout;
    char ch;
    int spaces = 0;
    int total = 0;
    cin.get(ch);
    while (ch != '.')      // quit at end of sentence
    {
        if (ch == ' ')    // check if ch is a space
            ++spaces;
        ++total;          // done every time
        cin.get(ch);
    }
    cout << spaces << " spaces, " << total;
    cout << " characters total in sentence\n";
    return 0;
}
```



Microsoft Visual Studio 调试控制台

```
The balloonist was an airhead
with lofty goals.
6 spaces, 46 characters total in sentence
```

- ❖ 仅当ch为空格时，语句++spaces；才被执行
- ❖ 语句++total：位于if语句的外面，因此每轮循环中都将被执行
- ❖ 字符总数中包括按回车键生成的换行符



## 1.2 if else语句

- ✓if语句让程序决定是否执行特定的语句或语句块，而if else语句则让程序决定执行两条语句或语句块中的哪一条，这种语句对于选择其中一种操作很有作用
- ✓双支语句 if else语句的通用格式

```
if (test-condition)
    statement1
else
    statement2
```



## 1.2 if else语句

- ✓ 当表达式为真时执行语句序列1，为假则执行语句序列2
- ✓ 表达式可以是任意类型，但按逻辑值求解（非0为真0为假）
- ✓ 表达式后无;

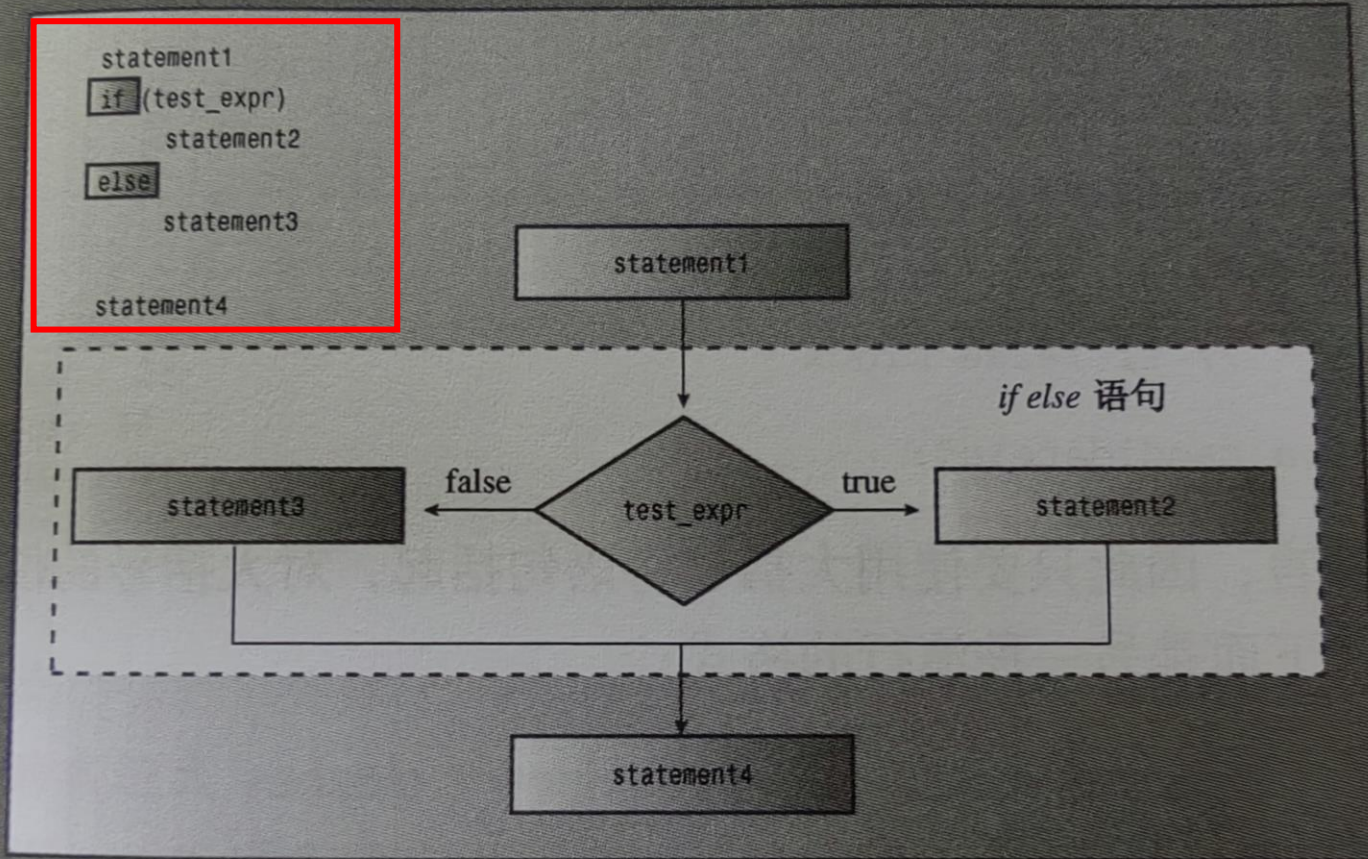
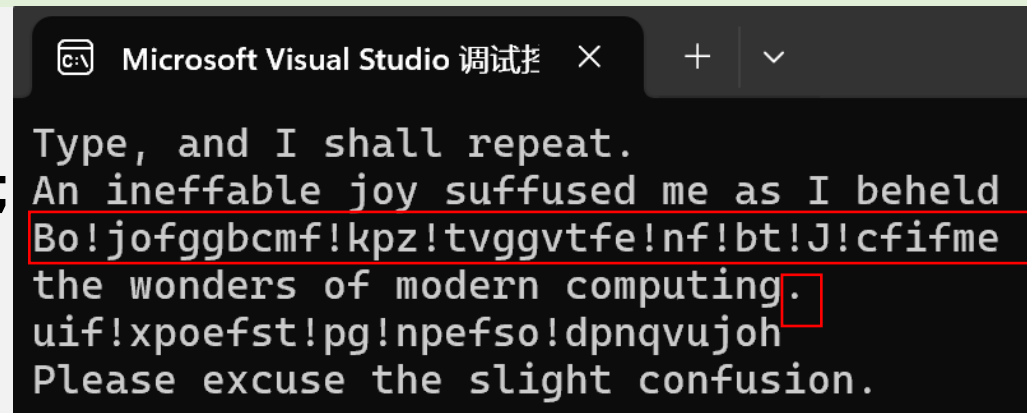


图 6.2 if else 语句的结构

```
// ifelse.cpp -- using the if else statement
#include <iostream>
int main()
{
    char ch;
    std::cout << "Type, and I shall repeat.\n";
    std::cin.get(ch);
    while (ch != '.')
    {
        if (ch == '\n')
            std::cout << ch;    // done if newline
        else
            std::cout << ++ch;  // done otherwise
        std::cin.get(ch);
    }
    std::cout << "\nPlease excuse the slight confusion.\n";
    return 0;
}
```

需求:

假设要通过对字母进行加密编码来修改输入的文本（换行符不变）



```
Microsoft Visual Studio 调试器
Type, and I shall repeat.
An ineffable joy suffused me as I beheld
Bo!jofggbcmf!kpz!tvggvtfe!nf!bt!J!cfifme
the wonders of modern computing.
uif!xpofst!pg!npefso!dpnqvujoh
Please excuse the slight confusion.
```



## 1.3 格式化if else语句

- ✓if else 中的两种操作都必须是一条语句
- ✓如果需要多条语句，需要用大括号括{}括起来，组成一个块语句

```
if (ch == 'z')  
{  
    // if true block  
    zorro++;  
    cout << "Another Zorro candidate\n";  
}  
else  
{  
    // if false block  
    dull++;  
    cout << "Not a Zorro candidate\n";  
}
```

强调语句的块结构

```
if (ch == 'z') {  
    zorro++;  
    cout << "Another Zorro candidate\n";  
}  
else {  
    dull++;  
    cout << "Not a Zorro candidate\n";  
}
```

将语句块与关键字if和else紧密结合在一起



# 1.4 if else if else语句

✓计算机也可提供两个以上的选择

✓多支语句

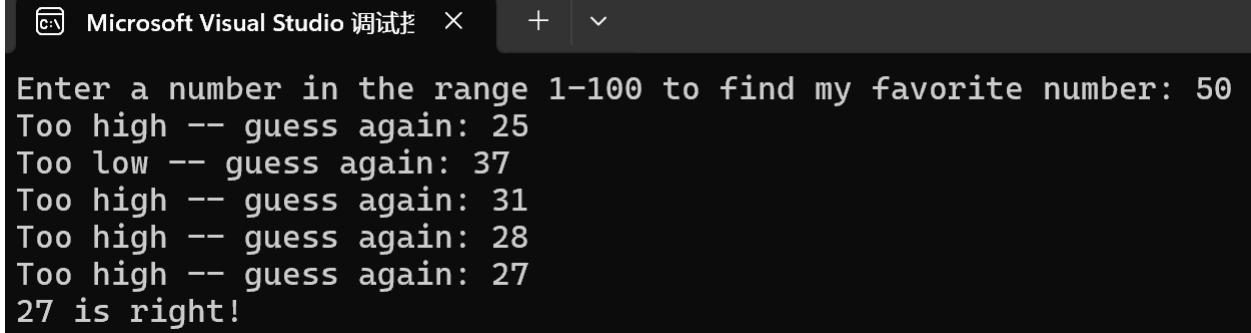
```
if (ch == 'A')
    a_grade++;    // alternative # 1
else
    if (ch == 'B') // alternative # 2
        B_grade++; // subalternative # 2a
    else
        soso++;    // subalternative # 2b
```

```
if (ch == 'A')
    a_grade++;    // alternative # 1
else if (ch == 'B')
    B_grade++;    // alternative # 2
else
    soso++;        // alternative # 3
```

实际上，它只是一个if else 被包含在另一个if else中。整个构造仍被视为一条语句

```
// ifelseif.cpp -- using if else if else
#include <iostream>
const int Fave = 27;
int main()
{
    using namespace std;
    int n;

    cout << "Enter a number in the range 1-100 to find ";
    cout << "my favorite number: ";
    do
    {
        cin >> n;
        if (n < Fave)
            cout << "Too low -- guess again: ";
        else if (n > Fave)
            cout << "Too high -- guess again: ";
        else
            cout << Fave << " is right!\n";
    } while (n != Fave);
    return 0;
}
```



Microsoft Visual Studio 调试 × + ▾

```
Enter a number in the range 1-100 to find my favorite number: 50
Too high -- guess again: 25
Too low -- guess again: 37
Too high -- guess again: 31
Too high -- guess again: 28
Too high -- guess again: 27
27 is right!
```



# 1.4 if else if else语句

注意=和==的区别

```
int a;
```

```
a == 10; 关系表达式, 值为0/1
```

```
a = 10; 赋值表达式, 值为10
```

✓条件运算符和错误防范

variable == value

例: myNumber == 3



myNumber = 3

value == variable

例: 3 == myNumber



3 = myNumber

后果: 编译器只是将3赋给myNmuber  
(难以发现错误, 导致错误结果)

后果: 编译器将会发出警告 (可及时发现错误, 并修正)

✓一般来说, 编写让编译器能够发现错误的代码, 比找出导致难以理解的错误的原因要容易得多。





# 目录

- 逻辑表达式

1. 逻辑OR运算符: `||`
2. 逻辑AND运算符: `&&`
3. 用`&&`来设置取值范围
4. 逻辑NOT运算符: `!`
5. 逻辑运算符细节
6. 其他表示方式



## 2.1 逻辑OR运算符: ||

✓为满足测试多种条件需要，C++提供3中逻辑运算符，来组合或修改已有的表达式

逻辑OR或(||)

逻辑AND与(&&)

逻辑NOT非(!)





## 2.1 逻辑OR运算符： ||

✓ 由于 || 的优先级比关系运算符低，因此不需要在表达式中使用括号

表 6.1

||运算符

	expr1    expr2 的值	
	expr1 == true	expr1 == false
expr2 == true	true	true
expr2 == false	true	false



## 2.1 逻辑OR运算符： ||

✓C++规定， || 运算符是个**顺序点**（sequence point）。运算符左边的子表达式先于右边的子表达式。即，先修改左侧的值，再对右侧的进行判断

```
i = 10;  
j = 10;  
i++ < 6 || i == j;
```

✓如果左侧的表达式为true，则C++将不会去判定右侧的表达式

（即**短路运算**：当有多个表达式(值)时，左边的表达式值可以确定结果的话，就不再继续运算右边的表达式的值）

```
// or.cpp -- using the logical OR operator
```

```
#include <iostream>
```

```
int main()
```

```
{
```

```
    using namespace std;
```

```
    cout << "This program may reformat your hard disk\n"
```

```
        "and destroy all your data. \n"
```

```
        "Do you wish to continue? <y/n> ";
```

```
    char ch;
```

```
    cin >> ch;
```

```
    if (ch == 'y' || ch == 'Y')                // y or Y
```

```
        cout << "You were warned!\a\a\n";
```

```
    else if (ch == 'n' || ch == 'N')           // n or N
```

```
        cout << "A wise choice ... bye\n";
```

```
    else
```

```
        cout << "That wasn't a y or n! Apparently you "
```

```
        "can't follow\ninstructions, so "
```

```
        "I'll trash your disk anyway. \a\a\a\n";
```

```
    return 0;
```

```
}
```

Microsoft Visual Studio 调试

This program may reformat your hard disk  
and destroy all your data.

Do you wish to continue? <y/n> d

That wasn't a y or n! Apparently you can't follow  
instructions, so I'll trash your disk anyway.



## 2.2 逻辑AND运算符: &&

- ✓C++可以采用逻辑AND运算符 (&&), 将两个表达式组合在一起
- ✓仅当原来两个表达式中都为true (或非零), 得到表达式的值为true; 否则, 表达式的值为false

```
5 == 5 && 4 == 4 // true because both expression are true
5 == 3 && 4 == 4 // false because first expression is false
5 > 3 && 5 > 10 // false because second expression is false
5 > 8 && 5 < 10 // false because first expression is false
5 < 8 && 5 > 2 // true because both expression are true
5 > 8 && 5 < 2 // false because both expression are false
```



## 2.2 逻辑AND运算符: &&

- ✓ 由于&&的优先级比关系运算符低, 因此不需要在表达式中使用括号
- ✓ 和||运算符一样, &&运算符也是**顺序点**, 因此将**首先判定左侧**, 并且**在右侧被判定前产生所有副作用**。如果左侧为false, 则整个逻辑表达式必定为false, 这种情况下, C++将**不会**再对右侧进行判定

表 6.2		&&运算符	
		expr1 && expr2 的值	
		expr1 == true	expr1 == false
expr2 == true		true	false
expr2 == false		false	false

```
// and.cpp -- using the logical AND operator
```

```
#include <iostream>
```

```
const int ArSize = 6;
```

```
int main()
```

```
{
```

```
    using namespace std;
```

```
    float naaq[ArSize];
```

```
    cout << "Enter the NAAQs (New Age Awareness Quotients) "
```

```
         << "of\nyour neighbors. Program terminates "
```

```
         << "when you make\n" << ArSize << " entries "
```

```
         << "or enter a negative value.\n";
```

```
    int i = 0;
```

```
    float temp;
```

```
    cout << "First value: ";
```

```
    cin >> temp;
```

```
    while (i < ArSize && temp >= 0) // 2 quitting criteria
```

```
    {
```

```
        naaq[i] = temp;
```

```
        ++i;
```

```
        if (i < ArSize)           // room left in the array
```

```
        {
```

```
            cout << "Next value: ";
```

```
            cin >> temp;           // so get next value
```

```
        }
```

```
    }
```

Microsoft Visual Studio 调试

+

```
Enter the NAAQs (New Age Awareness Quotients) of
your neighbors. Program terminates when you make
6 entries or enter a negative value.
```

```
First value: 28
```

```
Next value: 72
```

```
Next value: 15
```

```
Next value: 6
```

```
Next value: 130
```

```
Next value: 145
```

```
Enter your NAAQ: 50
```

```
3 of your neighbors have greater awareness of
the New Age than you do.
```

```
if (i == 0)
```

```
    cout << "No data--bye\n";
```

```
else
```

```
{
```

```
    cout << "Enter your NAAQ: ";
```

```
    float you;
```

```
    cin >> you;
```

```
    int count = 0;
```

```
    for (int j = 0; j < i; j++)
```

```
        if (naaq[j] > you)
```

```
            ++count;
```

```
    cout << count;
```

```
    cout << " of your neighbors have greater awareness of\n"
```

```
        << "the New Age than you do.\n";
```

```
}
```

```
return 0;
```

```
}
```



## 2.3 用&&来设置取值范围

- ✓&&运算符还允许建立一系列if else if else语句，其中每种选择都对应一个特定的取值范围
- ✓在使用取值范围测试时，应确保范围之间既没有缝隙，又没有重叠

```
if (age > 17 && age < 35)
    index = 0;
else if (age >= 35 && age < 50)
    index = 1;
else if (age >= 50 && age < 65)
    index = 2;
else
    index = 3;
```



## 2.3 用&&来设置取值范围

✓另外，应确保**正确设置**每个取值范围

```
if (age > 17 && age < 35) // ok
```

```
if ( 17 < age < 35) // Don' t do this!
```

**编译器不会捕获这种错误，因为它是有有效的C++语句**



＜运算符从左向右结合，因此上述表达式的含义是：

```
if ( (17 < age) < 35)
```

但：17<age的值要么为true (1)，要么为false (0)。0/1 均<35

**整个测试结果都是true！（这是错误的）**



```
// more_and.cpp -- using the logical AND operator
#include <iostream>
const char* qualify[4] = // an array of pointers*
                          // to strings
    "10,000-meter race.\n",
    "mud tug-of-war.\n",
    "masters canoe jousting.\n",
    "pie-throwing festival.\n"
};
int main()
{
    using namespace std;
    int age;
    cout << "Enter your age in years: ";
    cin >> age;
    int index;
```

Microsoft Visual Studio 调试器

Enter your age in years: 87  
You qualify for the pie-throwing festival.

```
    if (age > 17 && age < 35)
        index = 0;
    else if (age >= 35 && age < 50)
        index = 1;
    else if (age >= 50 && age < 65)
        index = 2;
    else
        index = 3;

    cout << "You qualify for the " << qualify[index];
    return 0;
}
```

指针数组为后续内容，  
此处了解即可

- ❖ 该程序演示了一种用于处理一系列消息的技术
- ❖ char指针变量可以通过指向一个字符串的开始位置来标识该字符串
- ❖ char指针数组也可以标识一系列字符串，将每个字符串的地址赋给各个数组元素即可



## 2.4 逻辑NOT运算符：！

✓ ！运算符将它后面的表达式的真值取反

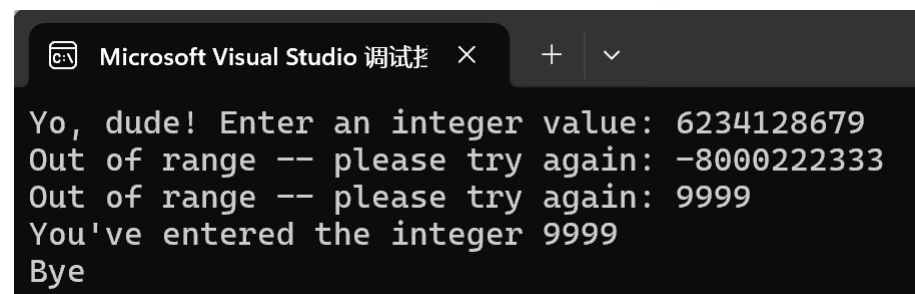
expression	true (非0)	false (0)
!expression	false (0)	true (非0)

✓ 通常，不使用！运算符表示关系

```
if (!(x>5))    // if (x <= 5) is clearer
```

✓ 然而，！运算符对于返回值为true-false或可以被解释为true-false的函数来说作用很大

```
// not.cpp -- using the not operator
#include <iostream>
#include <climits>
bool is_int(double);
int main()
{
    using namespace std;
    double num;
    cout << "Yo, dude! Enter an integer value: ";
    cin >> num;
    while (!is_int(num))    // continue while num is not int-able
    {
        cout << "Out of range -- please try again: ";
        cin >> num;
    }
    int val = int(num);    // type cast
    cout << "You've entered the integer " << val << "\nBye\n";
    return 0;
}
bool is_int(double x)
{
    if (x <= INT_MAX && x >= INT_MIN)    // use climits values
        return true;
    else
        return false;
}
```



```
Microsoft Visual Studio 调试
Yo, dude! Enter an integer value: 6234128679
Out of range -- please try again: -8000222333
Out of range -- please try again: 9999
You've entered the integer 9999
Bye
```

- ❖ 当读取int值的程序输入一个过大的值，一般C++实现程序只是将这个值截短为合适的大小，并不会通知丢失了数据
- ❖ 将可能的int值作为double值来读取，double类型的精度足以存储典型的int值，且取值范围更大
- ❖ 布尔函数is\_int()使用了climits文件中定义的两个符号常量（INT\_MAX和INT\_MIN）来确定参数是否位于适当的范围内
- ❖ main()程序使用了while循环来拒绝无效输入，直到用户输入有效的值为止



## 2.5 逻辑运算符细节

!      !A      取反(单目运算符) (3级), 高于关系运算符

&&      A&&B      均为真, 取值为真(12级), 低于关系运算符

||      A||B      有一个为真, 值为真(13级), 低于关系运算符

$x > 5 \ \&\& \ x < 10$        $\longleftrightarrow$       等价于       $(x > 5) \ \&\& \ (x < 10)$

$age > 30 \ \&\& \ age < 45 \ || \ weight > 300$        $\searrow$       等价于       $(age > 30 \ \&\& \ age < 45) \ || \ weight > 300$



## 2.5 逻辑运算符细节

!      !A      取反(单目运算符) (3级), 高于关系运算符

&&      A&&B      均为真, 取值为真(12级), 低于关系运算符

||      A||B      有一个为真, 值为真(13级), 低于关系运算符

(age > 50 || weight > 300) && donation > 1000

人为去掉括号

等价于 age > 50 || weight > 300 && donation > 1000

age > 50 || (weight > 300 && donation > 1000)



## 2.5 逻辑运算符细节

!      !A      取反(单目运算符) (3级), 高于关系运算符

&&      A&&B      均为真, 取值为真(12级), 低于关系运算符

||      A||B      有一个为真, 值为真(13级), 低于关系运算符

`!(x > 5) // is it false that x is greater than 5`

人为去掉括号

`!x > 5 // is !x greater than 5`

`// !x的值只能是true (0) 或false (1), 所以该表达式总为false`



## 2.5 逻辑运算符细节

✓ 虽然C++运算符的优先级规则常可能不使用括号便可以编写复合比较的语句，但最简单的方法还是用括号将测试进行分组，而不管是否需要括号（使得代码易读，并且减少由于没有准确记住所使用的规则而出错的可能性）

✓ C++确保程序从左向右进行计算逻辑表达式，并在知道答案后立刻停止

（短路运算）

$a \&\& b \&\& c$  若 $a=0$ ，值必为0， $b, c$ 不求解

$a || b || c$  若 $a=1$ ，值必为1， $b, c$ 不求解

$(m = a > b) \&\& (n = c > d);$  //  $m=0$ ,  $n$ 不再求解, 保持原值



# 2.6 其他表示方式

- ✓当键盘不能提供用作逻辑运算符的符号时，C++标准提供了标识符and、or、和not，这些C++保留字，不能被用作变量名
- ✓C语言程序需要包含头文件iso646.h，可以将它们用作运算符
- ✓C++不要求使用头文件

表 6.3

逻辑运算符：另一种表示方式

运算符	另一种表示方式
&&	and
	or
!	not





# 目录

- 字符函数库cctype



## 3.1 字符函数库cctype

✓C++从C语言继承了一个与字符相关的、非常方便的函数软件包，它可以简化诸如确定字符是否为大写字母、数学、标点符号等工作，这些函数的原型在头文件cctype（cctype.h）中定义

```
if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'))
```

↕ 等价于

```
if (isalpha(ch))
```

表 6.4

ctype 中的字符函数



函 数 名 称	返 回 值
isalnum( )	如果参数是字母数字，即字母或数字，该函数返回 true
isalpha( )	如果参数是字母，该函数返回 true
isctrl( )	如果参数是控制字符，该函数返回 true
isdigit( )	如果参数是数字（0~9），该函数返回 true
isgraph( )	如果参数是除空格之外的打印字符，该函数返回 true
islower( )	如果参数是小写字母，该函数返回 true
isprint( )	如果参数是打印字符（包括空格），该函数返回 true
ispunct( )	如果参数是标点符号，该函数返回 true
isspace( )	如果参数是标准空白字符，如空格、进纸、换行符、回车、水平制表符或者垂直制表符，该函数返回 true
isupper( )	如果参数是大写字母，该函数返回 true
isxdigit( )	如果参数是十六进制数字，即 0~9、a~f 或 A~F，该函数返回 true
tolower( )	如果参数是大写字符，则返回其小写，否则返回该参数
toupper( )	如果参数是小写字符，则返回其大写，否则返回该参数

```
// cctypes.cpp -- using the ctype.h library
#include <iostream>
#include <ctype>           // prototypes for character functions
int main()
{
    using namespace std;
    cout << "Enter text for analysis, and type @"
         << " to terminate input.\n";
    char ch;
    int whitespace = 0, digits = 0, chars = 0, punct = 0, others = 0;

    cin.get(ch);           // get first character
    while (ch != '@')       // test for sentinel
    {
        if (isalpha(ch))    // is it an alphabetic character?
            chars++;
        else if (isspace(ch)) // is it a whitespace character?
            whitespace++;
        else if (isdigit(ch)) // is it a digit?
            digits++;
        else if (ispunct(ch)) // is it punctuation?
            punct++;
        else
            others++;
        cin.get(ch);        // get next character
    }
}
```

Microsoft Visual Studio 调试 × + ▾

```
Enter text for analysis, and type @ to terminate input.
khdfflksadjklf djafjdiol jiohoho
hiohpihi hiohjoho
!@
46 letters, 5 whitespace, 0 digits, 1 punctuations, 0 others
```

```
cout << chars << " letters, "
     << whitespace << " whitespace, "
     << digits << " digits, "
     << punct << " punctuations, "
     << others << " others.\n";
return 0;
}
```



# 目录

- ?:运算符



## 4.1 ?:运算符

✓C++有一个常被用来替代if else语句的运算符，被称为条件运算符(?:)，它是C++中唯一一个需要3个操作数的运算符

✓该运算符的通用格式为: `expression1 ? expression2 : expression3`

如果expression1为true，则整个条件表达式的值为expression2的值；否则，整个表达式的值为expression3的值

```
5 > 3 ? 10 : 12 // 5 > 3 is true, so expression value is 10
3 == 9 ? 25 : 18 // 3 == 9 is false, so expression value is 18
int c = a > b ? a : b; // c = a if a > b, else c = b
```



## 4.1 ?:运算符

```
// condit.cpp -- using the conditional operator
```

```
#include <iostream>
```

```
int main()
```

```
{
```

```
    using namespace std;
```

```
    int a, b;
```

```
    cout << "Enter two integers: ";
```

```
    cin >> a >> b;
```

```
    cout << "The larger of " << a << " and " << b;
```

```
    int c = a > b ? a : b;    // c = a if a > b, else c = b
```

```
    cout << " is " << c << endl;
```

```
    return 0;
```

```
}
```

Microsoft Visual Studio 调试 × + ▾

Enter two integers: 25 28  
The larger of 25 and 28 is 28



## 4.1 ?:运算符

与if else 序列相比,?:运算符的区别:

- ✓ 条件运算符更简洁
- ✓ 条件运算符生成一个表达式,是一个值,可以将其赋给变量或将其放到一个更大的表达式中
- ✓ 可以将条件表达式嵌套在另一个条件表达式中 (隐藏代码)

```
cout << ((i < 2) ? !i ? x [i] : y : x [1]));
```





## 4.1 ?:运算符

与if else 序列相比,?:运算符的区别:

- ✓ 从可读性来说,条件运算符最适用于简单关系和简单表达式的值

```
x = (x > y) ? x : y;
```

- ✓ 当代码变得更复杂时,使用if else语句来表达可能更清晰



# 目录

- switch语句

1. switch语句
2. 将枚举量用作标签
3. switch和if else



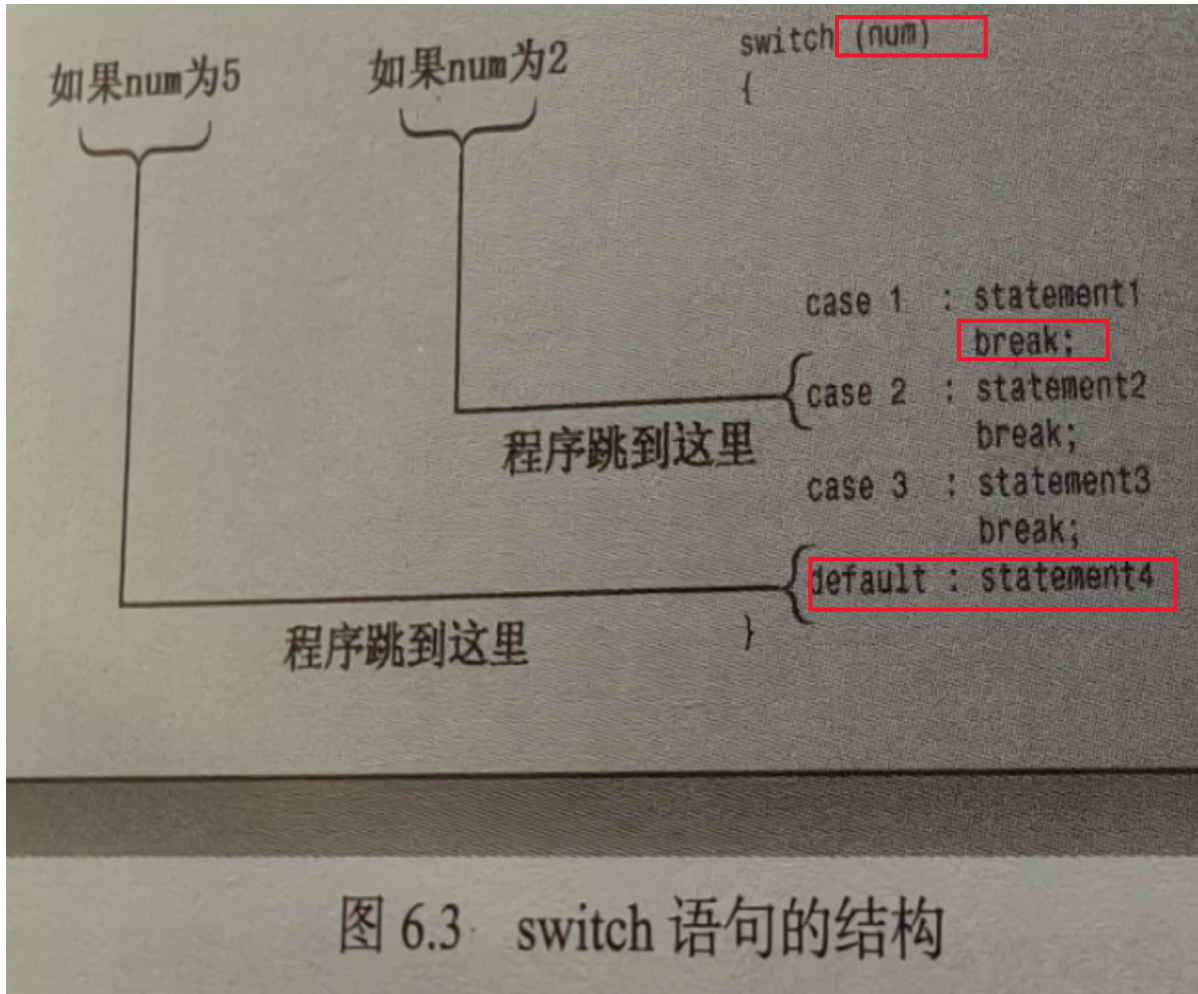
## 5.1 switch语句

✓当要从多个选项中选择一个时，可以扩展if else if else 序列处理多种情况，但C++的switch语句能够更容易地从大型列表中进行选择

✓switch语句的通用格式：

```
switch (integer-expression)
{
    case label1 : statement(s)
    case label2 : statement(s)
    ...
    default : statement(s)
}
```

## 5.1 switch语句



- ❖ 每个标签都**必须是整数常量表达式**
- ❖ 常见标签是int或char常量（如'1'或'q'），也可以枚举量
- ❖ 如果integer-expression不与任何标签匹配，则程序跳到标签为**default**行
- ❖ 如果default标签被省略，而又没有匹配的标签，则程序将跳到switch后面的执行语句
- ❖ 程序不会在执行到下一个case处自动停止，**必须使用break语句**

```
// switch.cpp -- using the switch statement
#include <iostream>
using namespace std;
void showmenu(); // function prototypes
void report();
void comfort();
int main()
{
    showmenu();
    int choice;
    cin >> choice;
    while (choice != 5)
    {
        switch (choice)
        {
            case 1:  cout << "\a\n";
                     break;
            case 2:  report();
                     break;
            case 3:  cout << "The boss was in all day.\n";
                     break;
            case 4:  comfort();
                     break;
            default: cout << "That's not a choice.\n";
        }
        showmenu();
        cin >> choice;
    }
    cout << "Bye!\n";
    return 0;
}
```

```
Microsoft Visual Studio 调试
Please enter 1, 2, 3, 4, or 5:
1) alarm          2) report
3) alibi          4) comfort
5) quit
1
Please enter 1, 2, 3, 4, or 5:
1) alarm          2) report
3) alibi          4) comfort
5) quit
2
It's been an excellent week for business.
Sales are up 120%. Expenses are down 35%.
Please enter 1, 2, 3, 4, or 5:
1) alarm          2) report
3) alibi          4) comfort
5) quit
```

```
3
The boss was in all day.
Please enter 1, 2, 3, 4, or 5:
1) alarm          2) report
3) alibi          4) comfort
5) quit
4
Your employees think you are the finest CEO
in the industry. The board of directors think
you are the finest CEO in the industry.
Please enter 1, 2, 3, 4, or 5:
1) alarm          2) report
3) alibi          4) comfort
5) quit
5
Bye!
```

```
void showmenu()
{
    cout << "Please enter 1, 2, 3, 4, or 5:\n"
          "1) alarm          2) report\n"
          "3) alibi          4) comfort\n"
          "5) quit\n";
}
void report()
{
    cout << "It's been an excellent week for business.\n"
          "Sales are up 120%. Expenses are down 35%.\n";
}
void comfort()
{
    cout << "Your employees think you are the finest CEO\n"
          "in the industry. The board of directors think\n"
          "you are the finest CEO in the industry.\n";
}
```



# 5.1 switch语句

- ✓如果删除break语句，程序将执行case标签后的所有语句
- ✓这样设计可以使得使用多个标签很简单
- ✓如例所示，可以为大写标签和小写标签提供相同的语句（case 'a' 后面没有break，因此将执行下一行case 'A'）

```
char choice;  
cin >> choice;  
while (choice != 'Q' && choice != 'q')  
{  
    switch(choice)  
    {  
        case 'a' :  
        case 'A' : cout << "\a\n";  
                    break;  
  
        case 'r' :  
        case 'R' : cout << report();  
                    break;  
        default : cout << "That's not a choice.\n"  
    }  
}
```



## 5.2 将枚举量用作标签

- ✓程序enum定义一组相关的常量，然后再switch语句中使用这些常量
- ✓cin无法识别枚举常量，因此该程序要求用户选择选项时输入一个整数
- ✓switch语句将int值和枚举量标签进行比较时，将枚举量提升为int
- ✓在while循环测试条件中，也会将枚举量提升为int类型

//枚举的细节为后续知识，此处了解用法，看懂程序即可

```

// enum.cpp -- using enum
#include <iostream>
// create named constants for 0 - 6
enum { red, orange, yellow, green, blue, violet, indigo };
int main()
{
    using namespace std;
    cout << "Enter color code (0-6): ";
    int code;
    cin >> code;
    while (code >= red && code <= indigo)
    {
        switch (code)
        {
            case red: cout << "Her lips were red.\n"; break;
            case orange: cout << "Her hair was orange.\n"; break;
            case yellow: cout << "Her shoes were yellow.\n"; break;
            case green: cout << "Her nails were green.\n"; break;
            case blue: cout << "Her sweatsuit was blue.\n"; break;
            case violet: cout << "Her eyes were violet.\n"; break;
            case indigo: cout << "Her mood was indigo.\n"; break;
        }
        cout << "Enter color code (0-6): ";
        cin >> code;
    }
    cout << "Bye\n";
    return 0;
}

```


Microsoft Visual Studio 调试

```

Enter color code (0-6): 3
Her nails were green.
Enter color code (0-6): 5
Her eyes were violet.
Enter color code (0-6): 2
Her shoes were yellow.
Enter color code (0-6): 8
Bye

```





## 5.3 switch和if else

✓switch语句和if else语句都允许程序从选项中进行选择

区别:

✓相比之下, if else更通用, 它可以处理取值范围

✓switch语句中每个case标签都必须是一个单独的值 (常量),  
且必须是整数 (包括char), 因此switch无法处理浮点测试



## 5.3 switch和if else

- ✓如果选项涉及取值范围、浮点测试或两个变量的比较，则应使用if else语句
- ✓如果所有的选项都可以使用整数常量来标识，则可以使用switch语句或if else语句。如果选项超过两个，则就代码长度和执行速度而言，switch语句效率更高

提示：如果既可以使用if else语句，也可以使用switch语句，则当选项不小于3个时，应使用switch语句



# 目录

- break和continue语句

# 6.1 break和continue语句

- ✓break和continue语句都使程序能够跳过部分代码
- ✓switch语句或任何循环中使用break语句，使程序跳到switch或循环后面的语句处执行
- ✓continue语言用于循环中，让程序跳过循环体中余下的代码，并开始新一轮循环

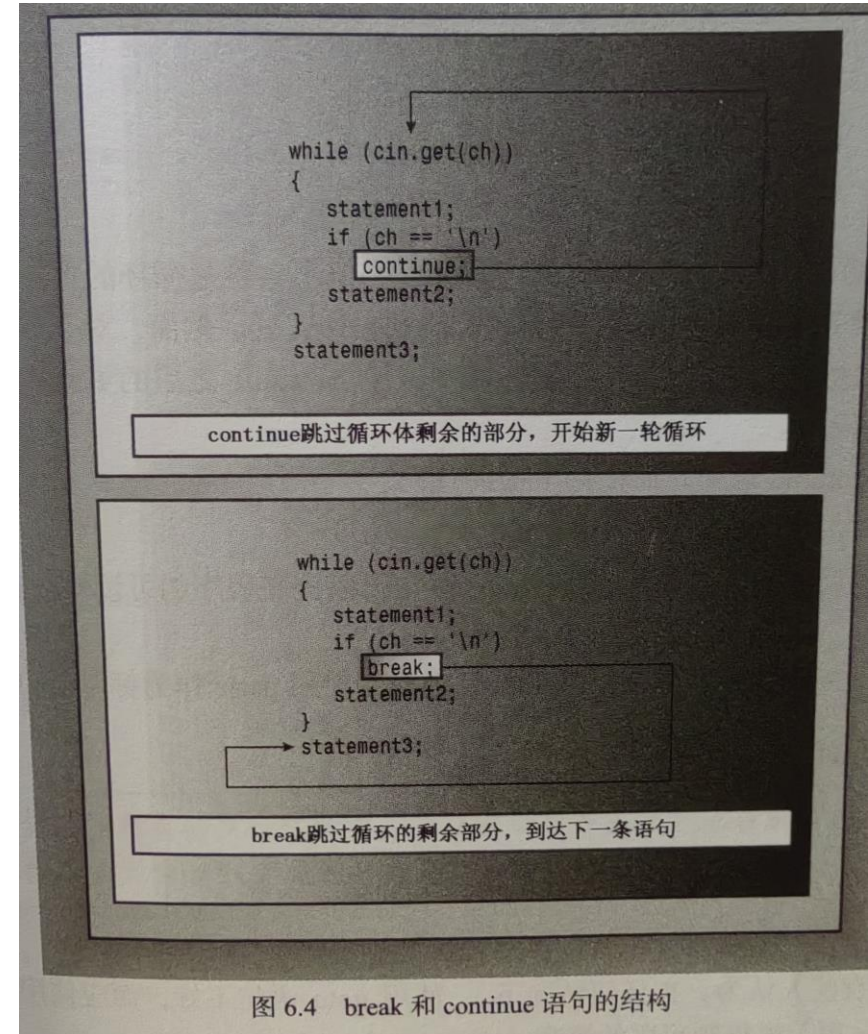
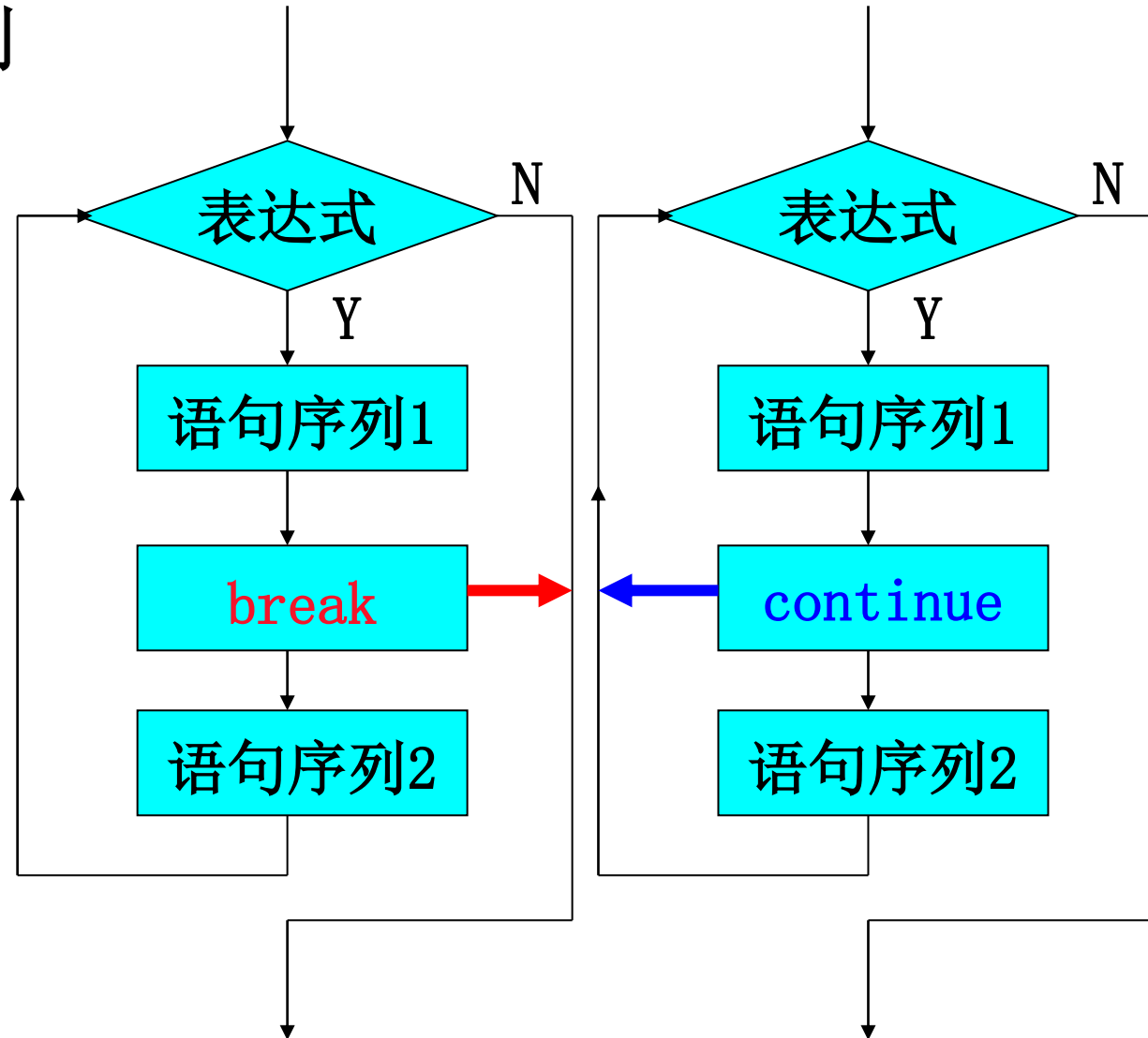


图 6.4 break 和 continue 语句的结构

# 6.1 break和continue语句

## ✓ break和continue的区别

```
while(表达式1) {  
    语句序列1;  
    break/continue;  
    语句序列2;  
}
```





例：给出下列程序的运行结果

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

int main()      i=1 sum=0
{
    int i=0, sum=0;
    while(i<1000) {
        i++;
        break;
        sum=sum+i;
    }
    cout << "i=" << i
         << " sum=" << sum
         << endl;
    return 0;
}
```

Microsoft Visual Studio 调试控制台  
i=1 sum=0

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

int main()      i=1000 sum=0
{
    int i=0, sum=0;
    while(i<1000) {
        i++;
        continue;
        sum=sum+i;
    }
    cout << "i=" << i
         << " sum=" << sum
         << endl;
    return 0;
}
```

Microsoft Visual Studio 调试控制台  
i=1000 sum=0



# 总结

## • if语句

1. if 语句
2. if else语句
3. 格式化if else  
语句
4. if else if  
else结构

## • 逻辑表达式

1. 逻辑OR运算符: ||
2. 逻辑AND运算符: &&
3. 用&&来设置取值范围
4. 逻辑NOT运算符: !
5. 逻辑运算符细节
6. 其他表示方式

## • 字符函数库ctype

## • ?:运算符

## • switch语句

1. switch语句
2. 将枚举量用作标签
3. switch和if else

## • break和continue语句