Chapter 5

Selection--Making Decision

Brooks/Cole
Thomson Learning™

OBJECTIVES

After studying this chapter you will be able to:

- ❑ Understand and use logical data in a program.
- ❑ Understand and use the: *not*, *and*, and *or* logical operators.
- ❑ Understand and use the six relational operators.
- ❑ Write selection statements using two-way selection and multiway selection.
- ❑ Understand and avoid the dangling-else problem.
- ❑ Implement multiway selection using the *switch* statement or the *else-if* format.
- ❑ Use the standard character functions to test or reformat character data.
- ❑ Understand why controlled statements should be indented and use indentation for program readability.
- ❑ Create structure charts that show logic flow in selection paths.

Brooks/Cole
Thomson Learning™

5.1

LOGICAL DATA AND OPERATORS

Figure 5-1 Design for print report

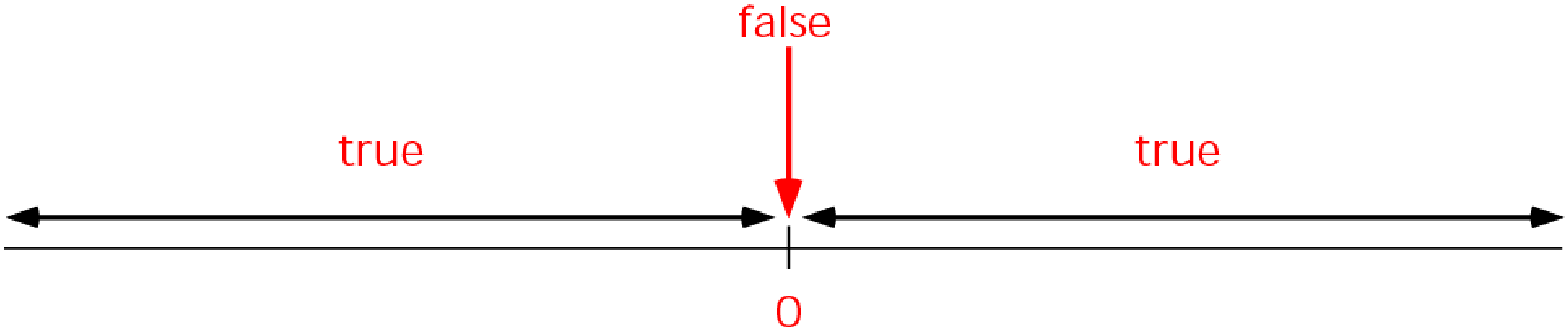


Figure 5-2 Logical operators truth table

not (!)

x	!x
false	true
true	false

logical

!

x	!x
zero	1
nonzero	0

C++ Language

and (&&)

x	y	x && y
false	false	false
false	true	false
true	false	false
true	true	true

logical

&&

x	y	x && y
zero	zero	0
zero	nonzero	0
nonzero	zero	0
nonzero	nonzero	1

C++ Language

or (||)

x	y	x || y
false	false	false
false	true	true
true	false	true
true	true	true

logical

||

x	y	x || y
zero	zero	0
zero	nonzero	1
nonzero	zero	1
nonzero	nonzero	1

C++ Language

Brooks/Cole
Thomson Learning™

Note:

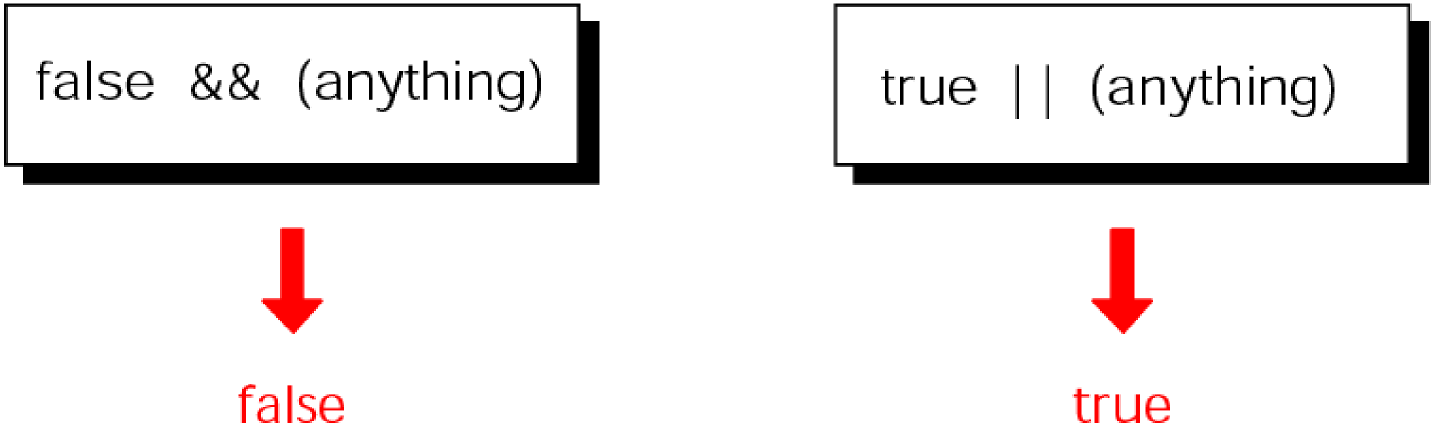
In C++

If a value is zero, it can be used as the logical value false.

If a value is not zero, it can be used as the logical value true.

Zero	<===>	***False***
Nonzero	<===>	***True***

Figure 5-3 Short-circuit methods for and and or



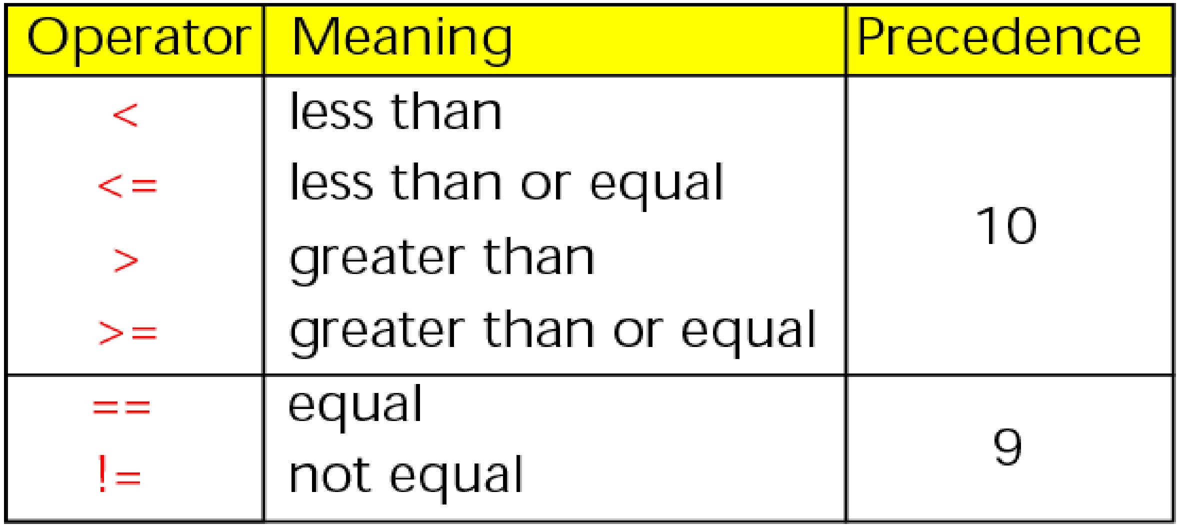
false && (anything)

false

true || (anything)

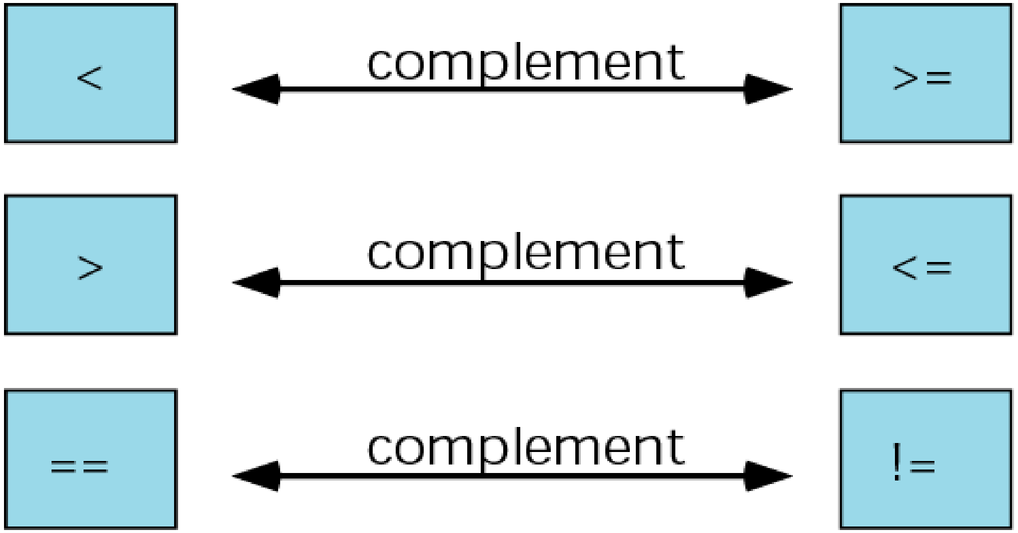
true

Figure 5-4 Relational operators



Operator	Meaning	Precedence
<	less than	10
<=	less than or equal	
>	greater than	
>=	greater than or equal	
==	equal	9
!=	not equal	

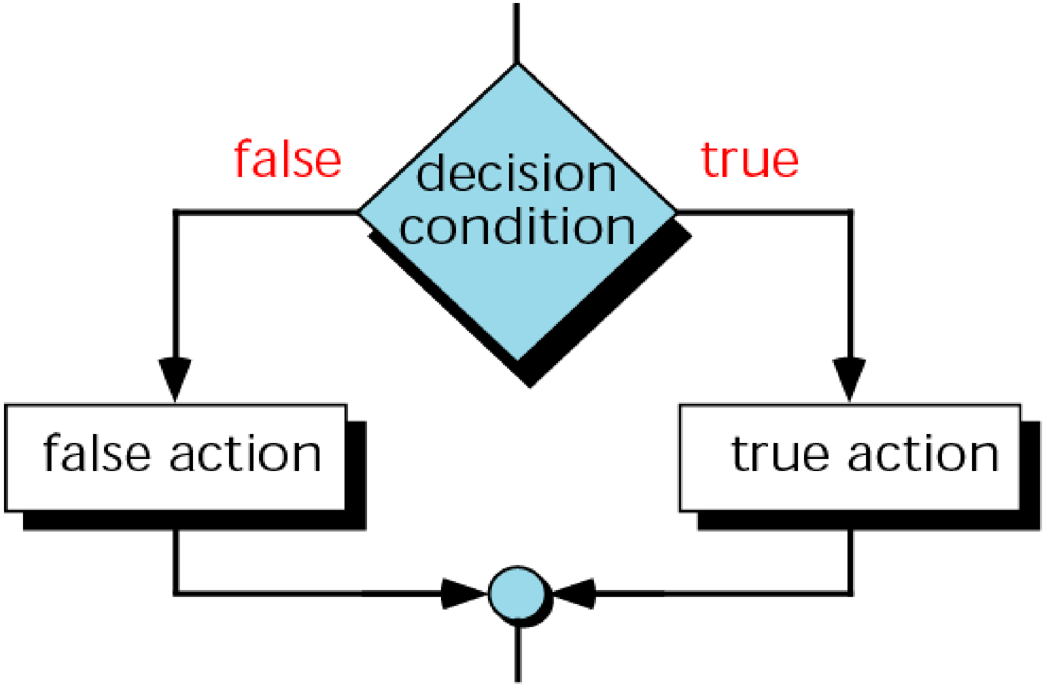
Figure 5-5 Logical operator complements



5.2

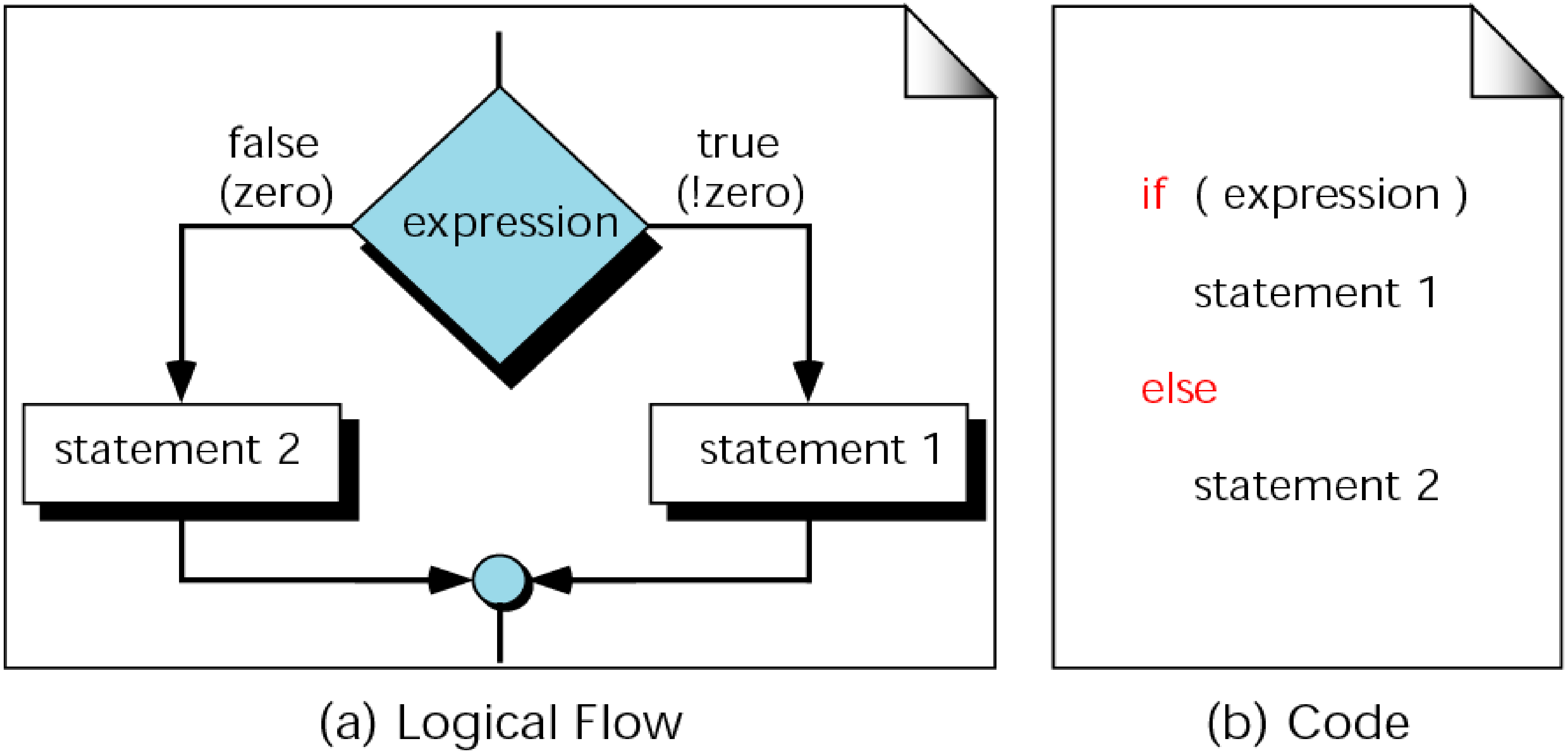
TWO-WAY SELECTION

Figure 5-6 Two-way decision logic



Brooks/Cole
Thomson Learning™

Figure 5-7 if...else logic flow



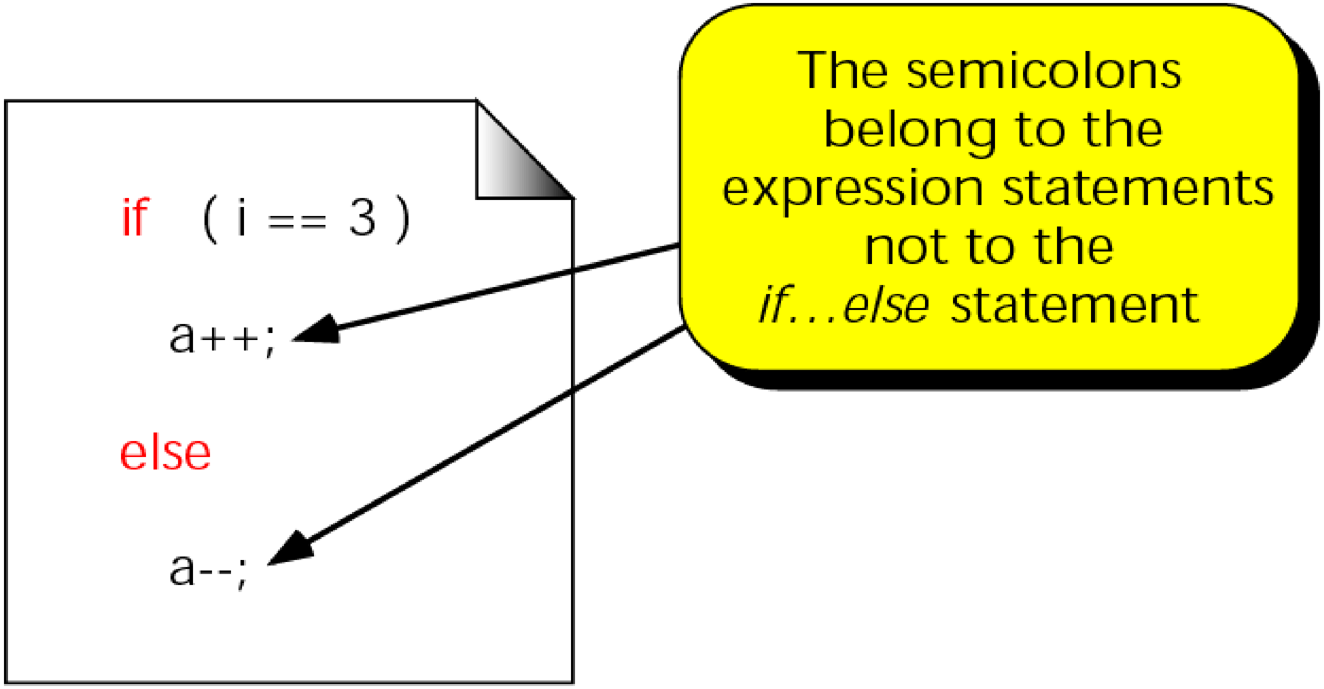
(a) Logical Flow

```
if ( expression )
    statement 1
else
    statement 2
```

(b) Code

Brooks/Cole
Thomson Learning

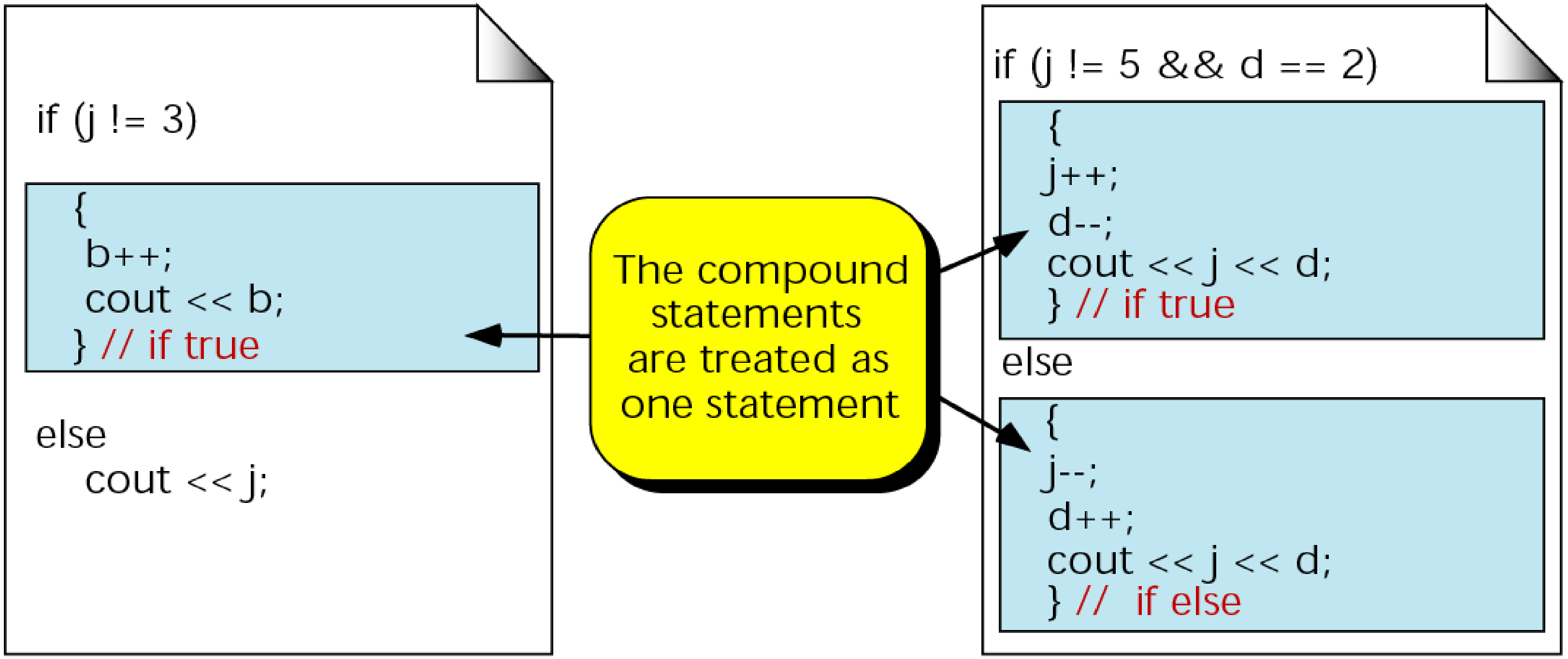
Figure 5-8 A simple if...else statement



```
if   ( i == 3 )
    a++;
else
    a--;
```

The semicolons belong to the expression statements not to the *if...else* statement

Figure 5-9 Compound statements in an if...else



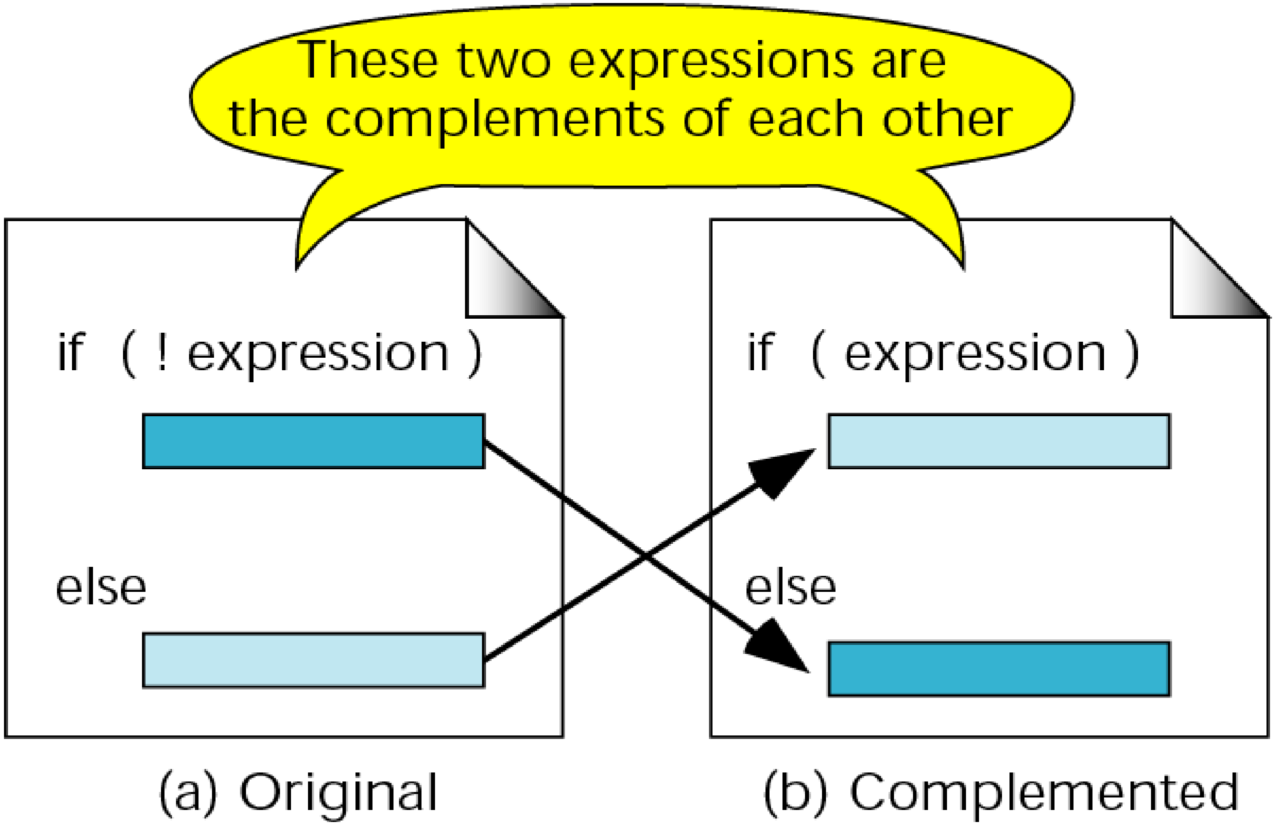
```
if (j != 3)
  {
   b++;
   cout << b;
  } // if true

else
   cout << j;
```

The compound statements are treated as one statement

```
if (j != 5 && d == 2)
  {
  j++;
  d--;
  cout << j << d;
  } // if true
else
  {
  j--;
  d++;
  cout << j << d;
  } //  if else
```

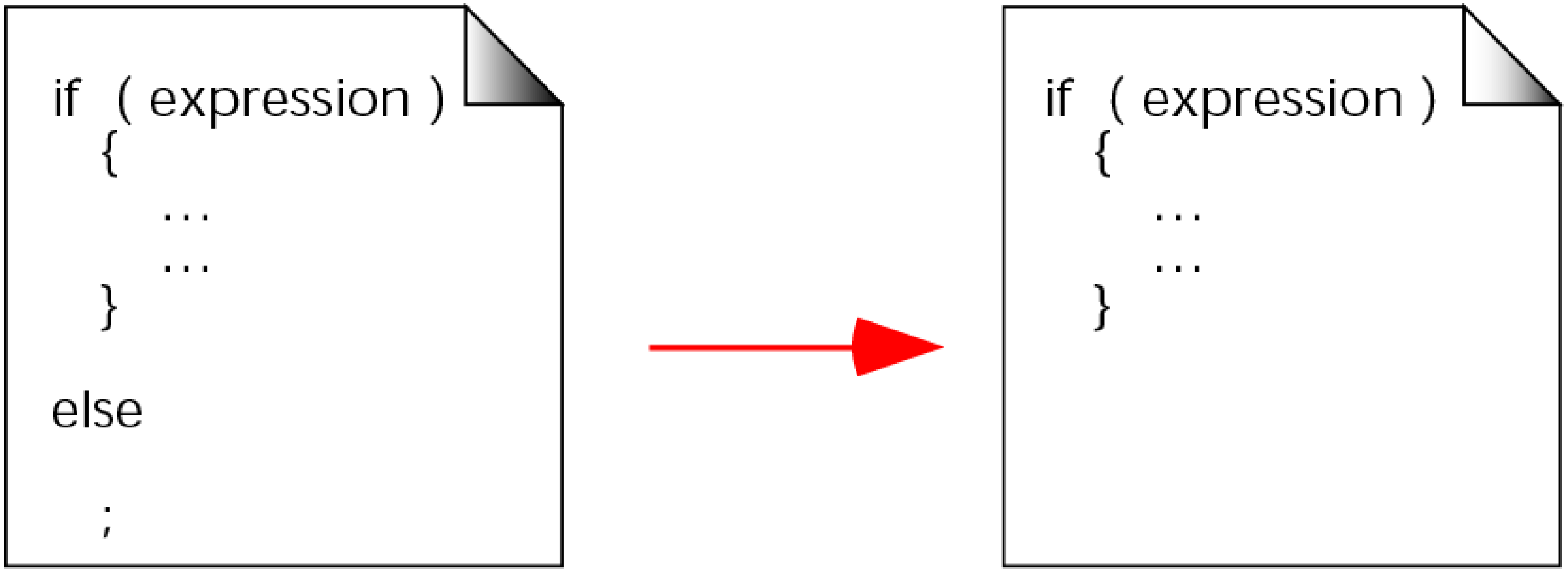
Brooks/Cole
Thomson Learning™

Figure 5-10 Complemented if...then statements



Brooks/Cole
Thomson Learning™

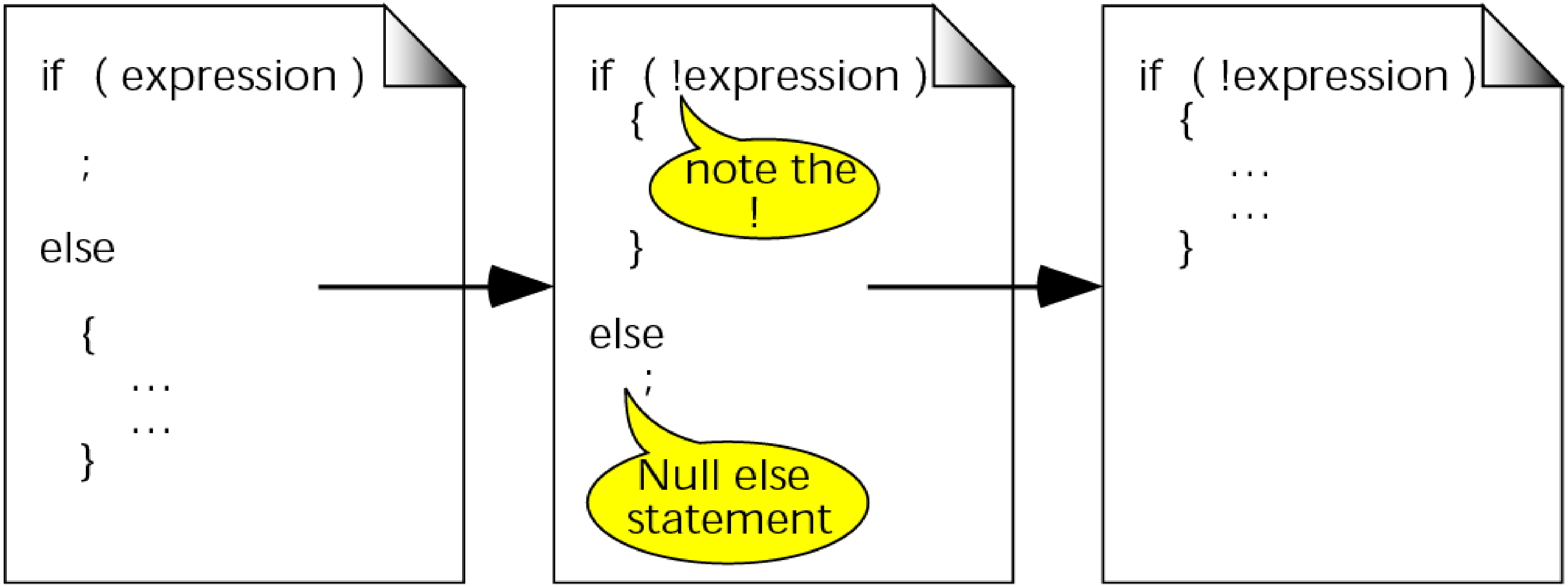
Figure 5-11 A null else statement



```
if ( expression )
  {
     ...
     ...
  }

else

  ;
```

```
if ( expression )
  {
     ...
     ...
  }
```

Figure 5-12 A null if statement

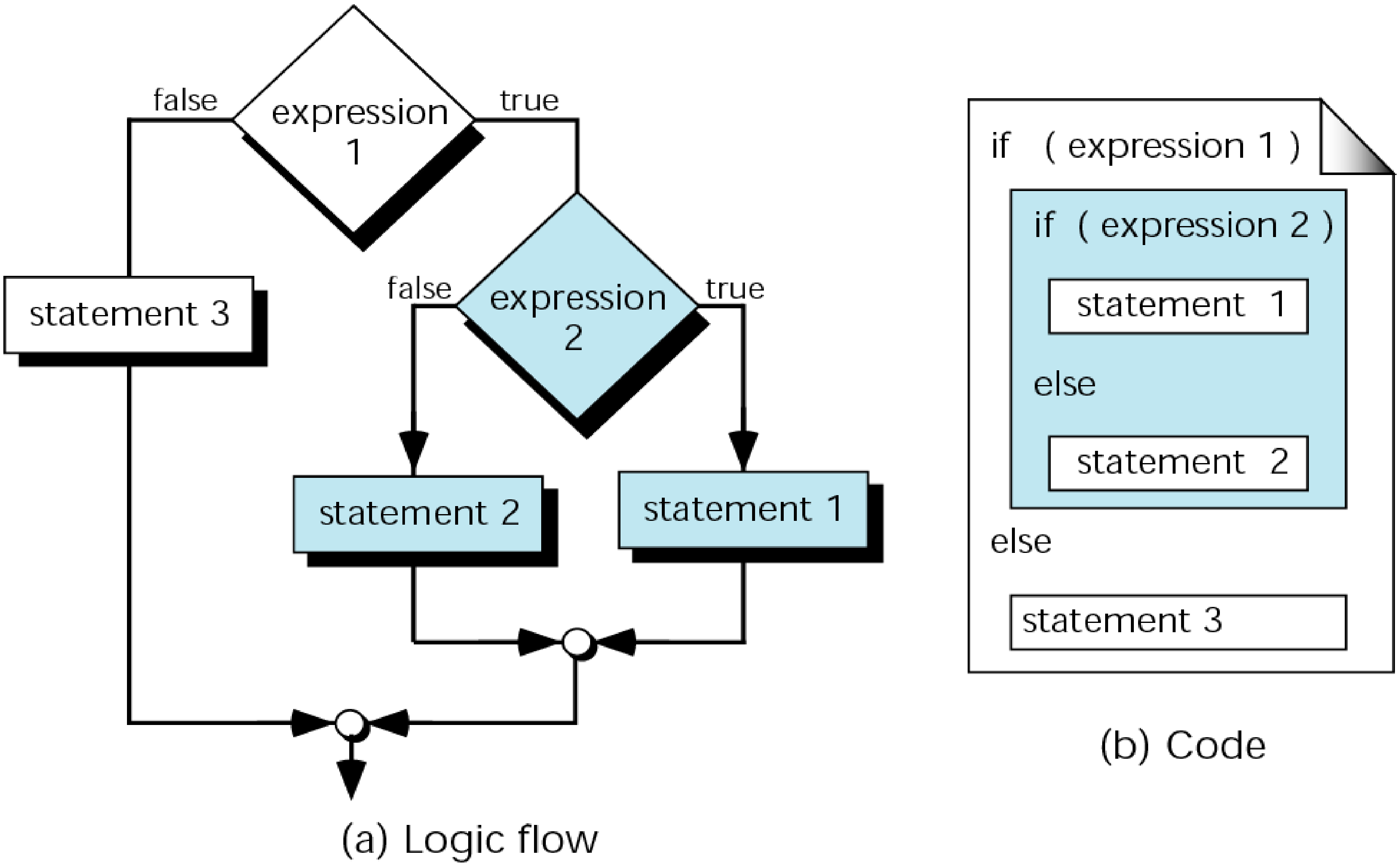


```
if ( expression )
   ;
else
   {
      ...
      ...
   }
```

```
if ( !expression )
   {
   }
else
   ;
```

```
if ( !expression )
   {
      ...
      ...
   }
```

Figure 5-13 Nested if statements



(a) Logic flow

```
if  ( expression 1 )
    if  ( expression 2 )
        statement  1
    else
        statement  2
else
    statement 3
```

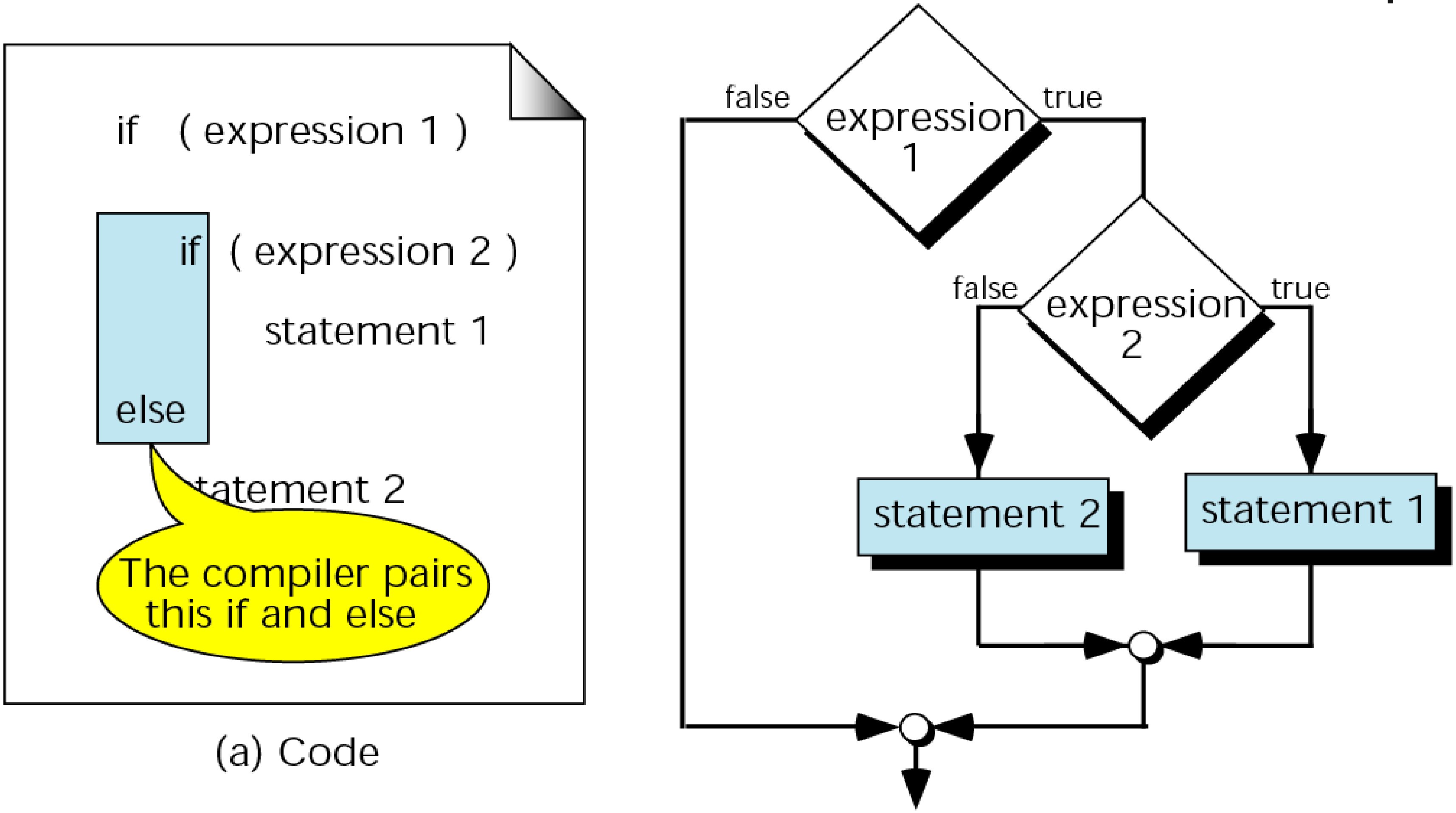
(b) Code

Brooks/Cole
Thomson Learning

Note:

else* is always paired with the most recent, unpaired *if

Figure 5-14 Dangling else



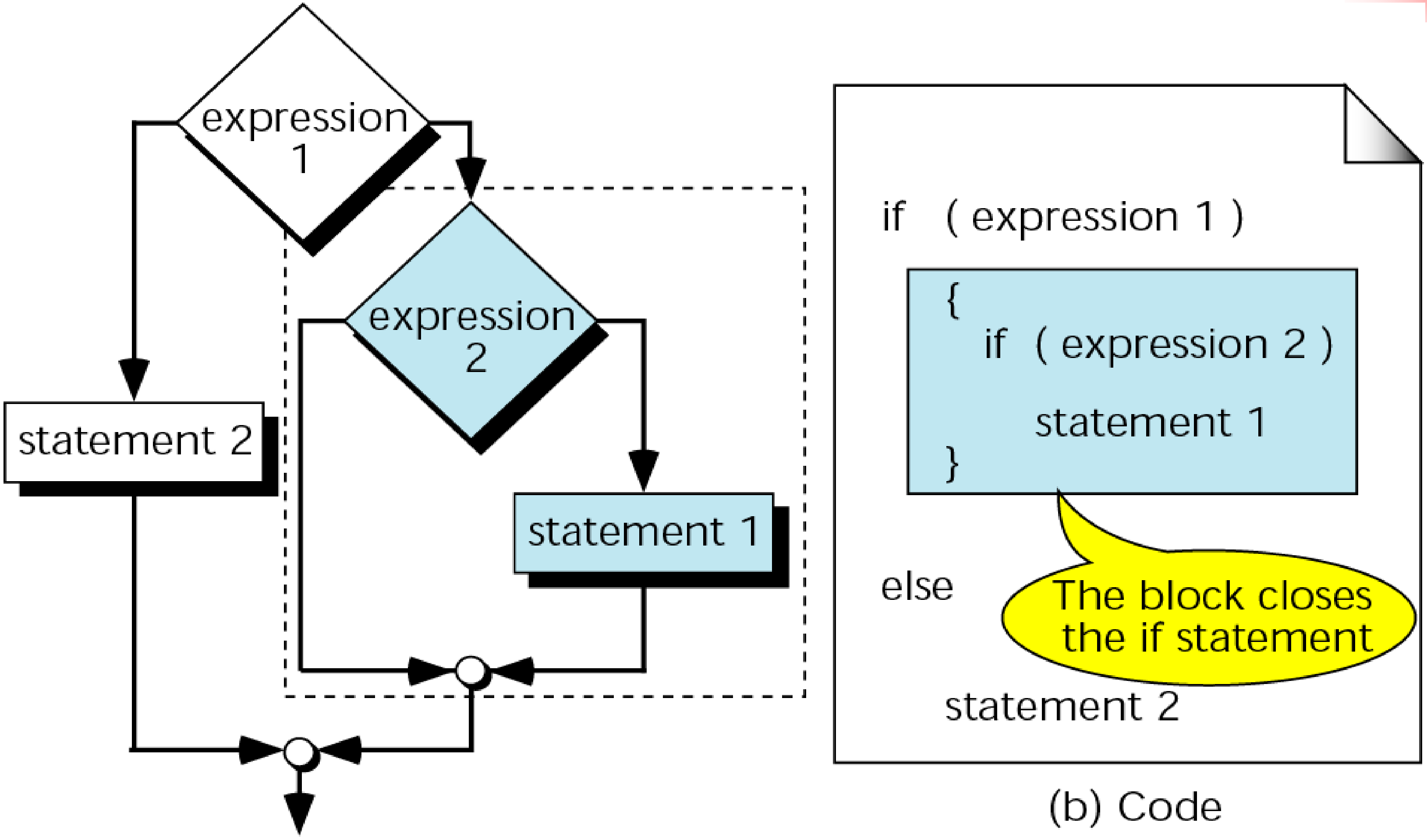
```
if   ( expression 1 )

     if  ( expression 2 )

            statement 1

     else

            statement 2
```

The compiler pairs this if and else

(a) Code

(b) Logic Flow

Figure 5-15 Dangling else solution



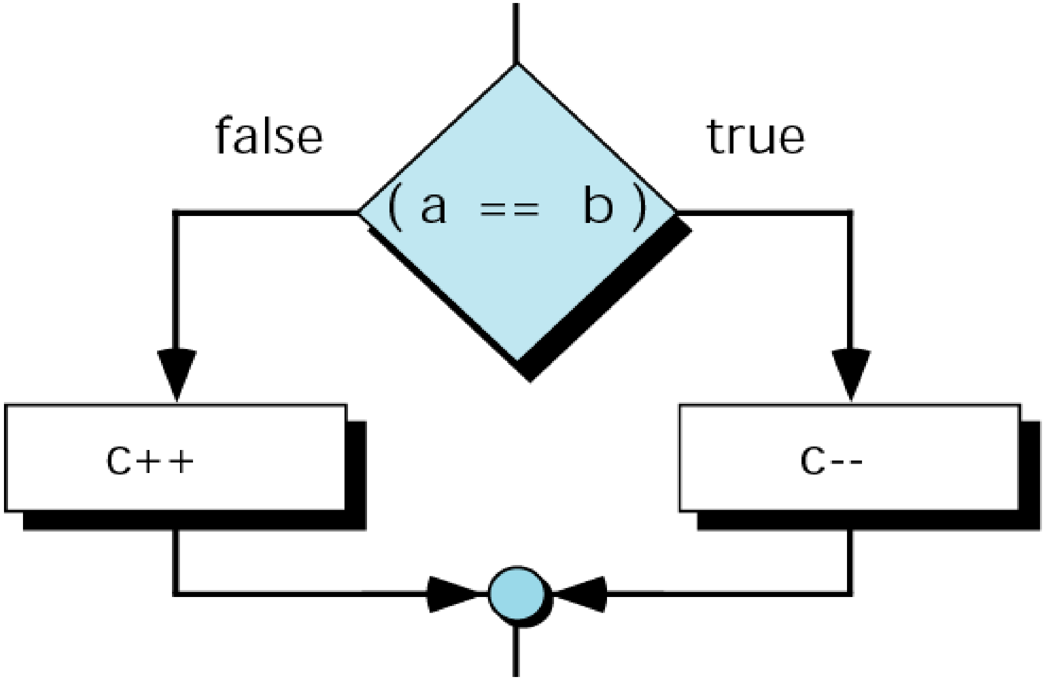
```
if   ( expression 1 )
    {
      if  ( expression 2 )
            statement 1
    }
else
      statement 2
```

(a) Logic Flow

(b) Code

Brooks/Cole
Thomson Learning

Figure 5-16 Conditional expression

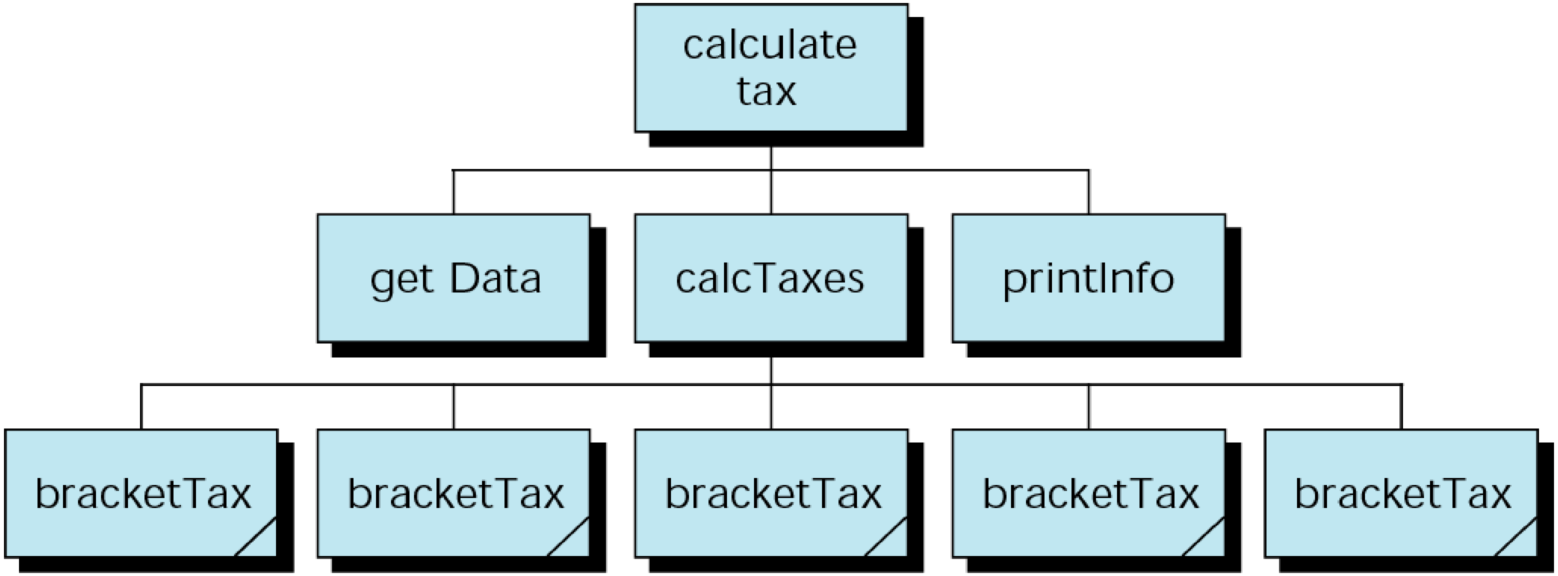


(a) Logic Flow

```
a == b ? c-- : c++
```

(b) Code

Figure 5-17 Design for calculate taxes



5.3

Multiway Selection

Figure 5-18 switch decision logic

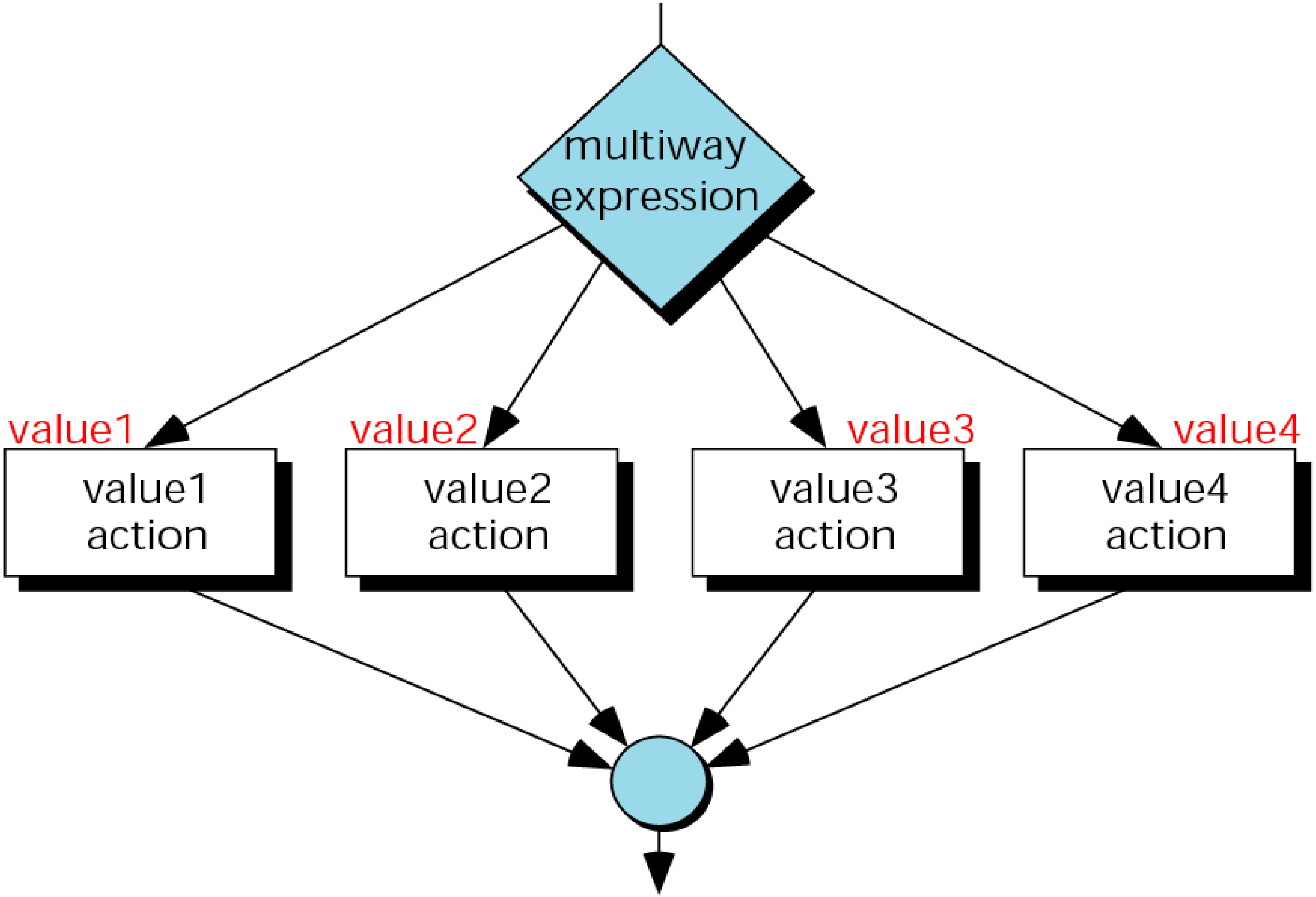
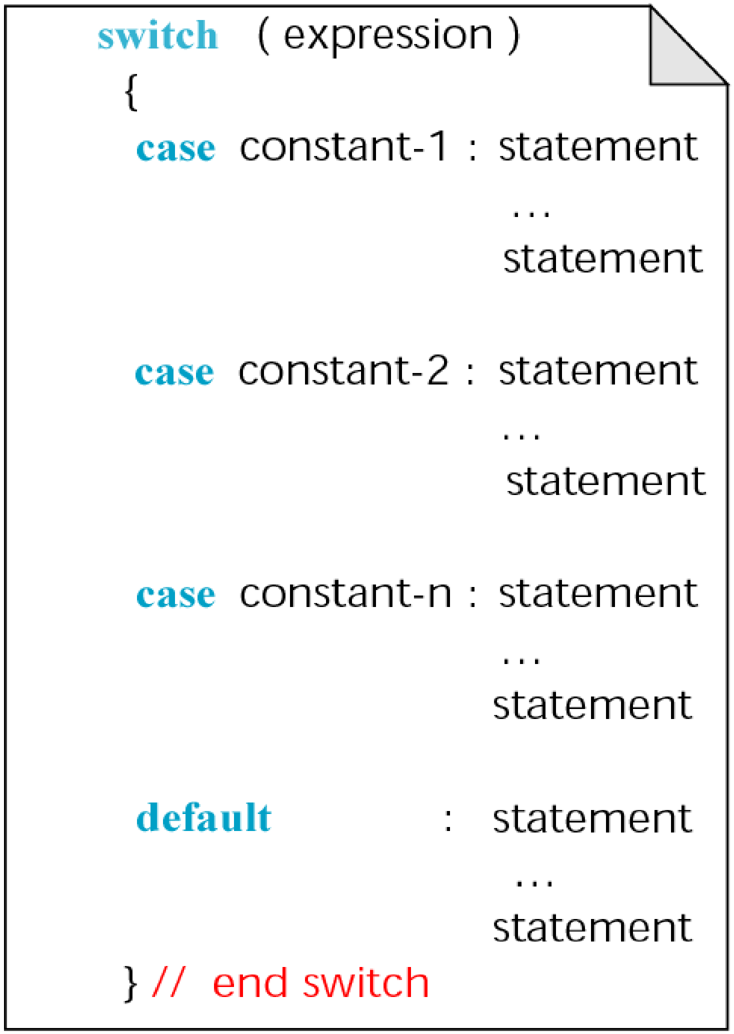


Figure 5-19 switch statement



```
switch  ( expression )
 {
  case  constant-1 :  statement
                      ...
                      statement

  case  constant-2 :  statement
                      ...
                      statement

  case  constant-n :  statement
                      ...
                      statement

  default          :  statement
                      ...
                      statement
 } //  end switch
```

Figure 5-20 switch flow

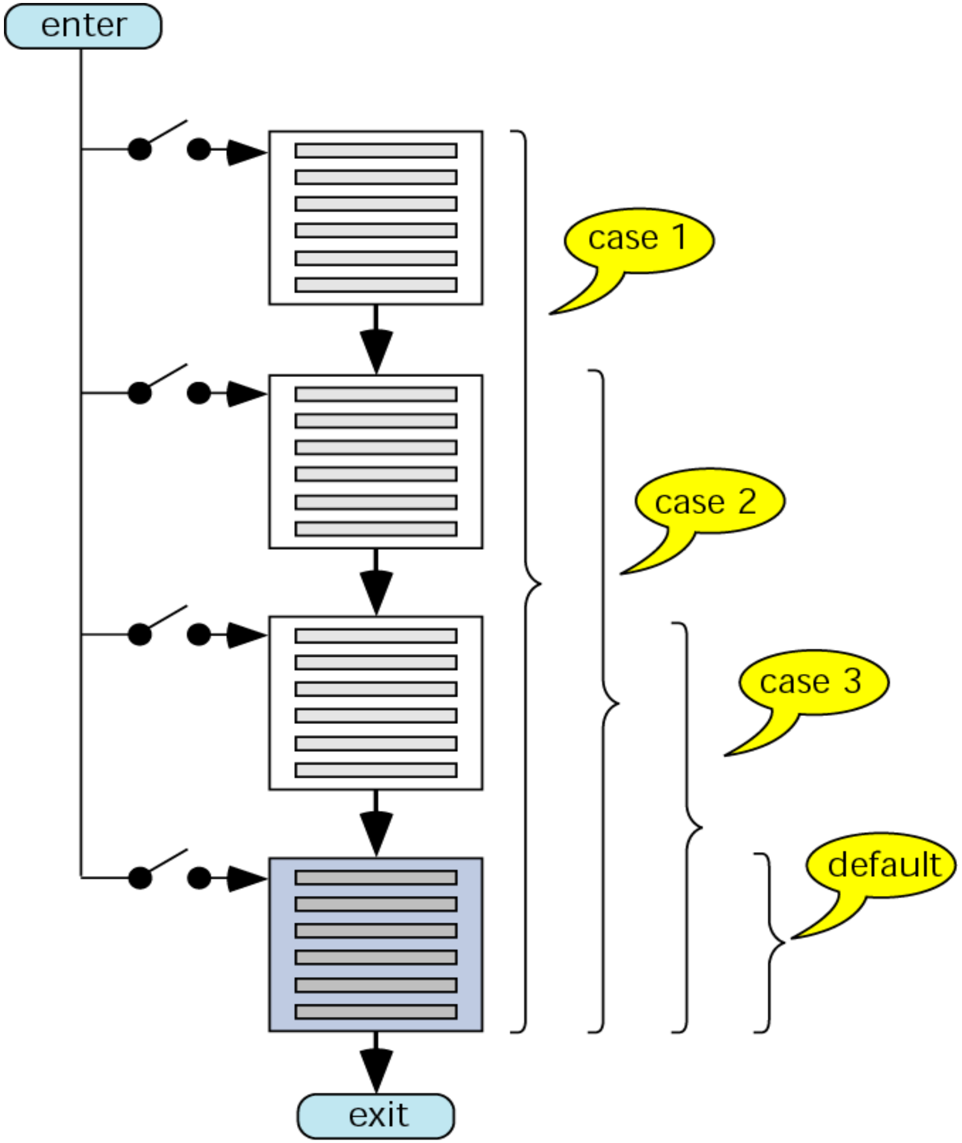


Figure 5-21 switch results

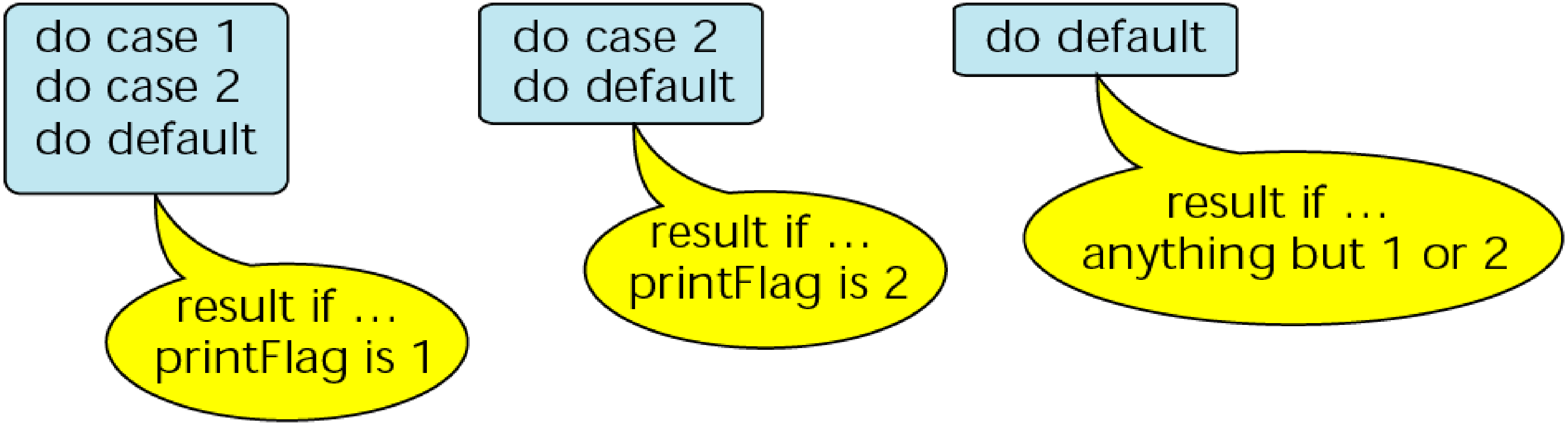
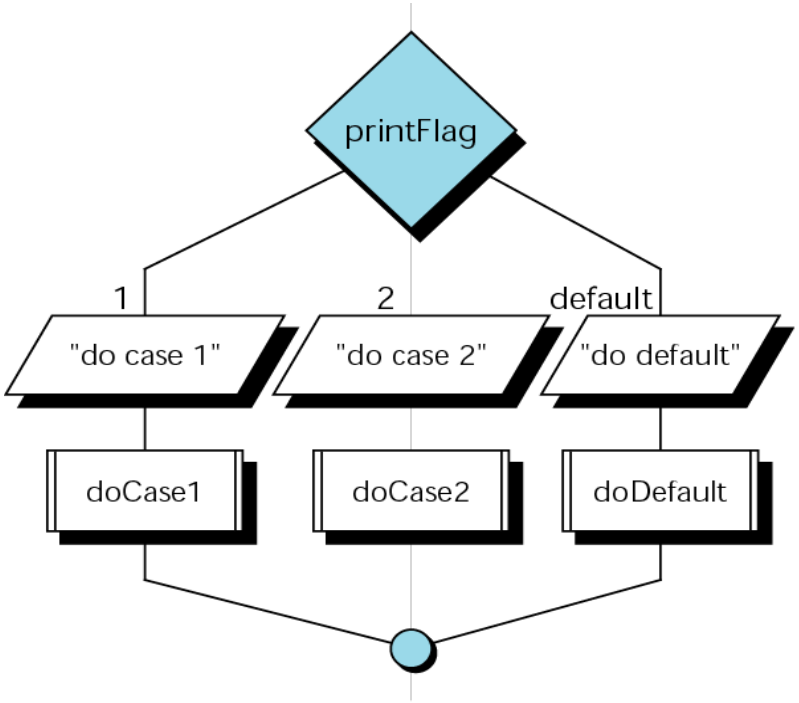


Figure 5-22 A switch with break statements



(a) Logic Flow

```
switch (printFlag)
  {
   case 1:      cout << "do case 1\n";
                doCase1 ();
                break ;
   case 2:      cout << "do case 2\n";
                doCase2 ();
                break ;
  default:      cout << "do default\n";
                doDefault ();
                break ;

  }  // switch
```

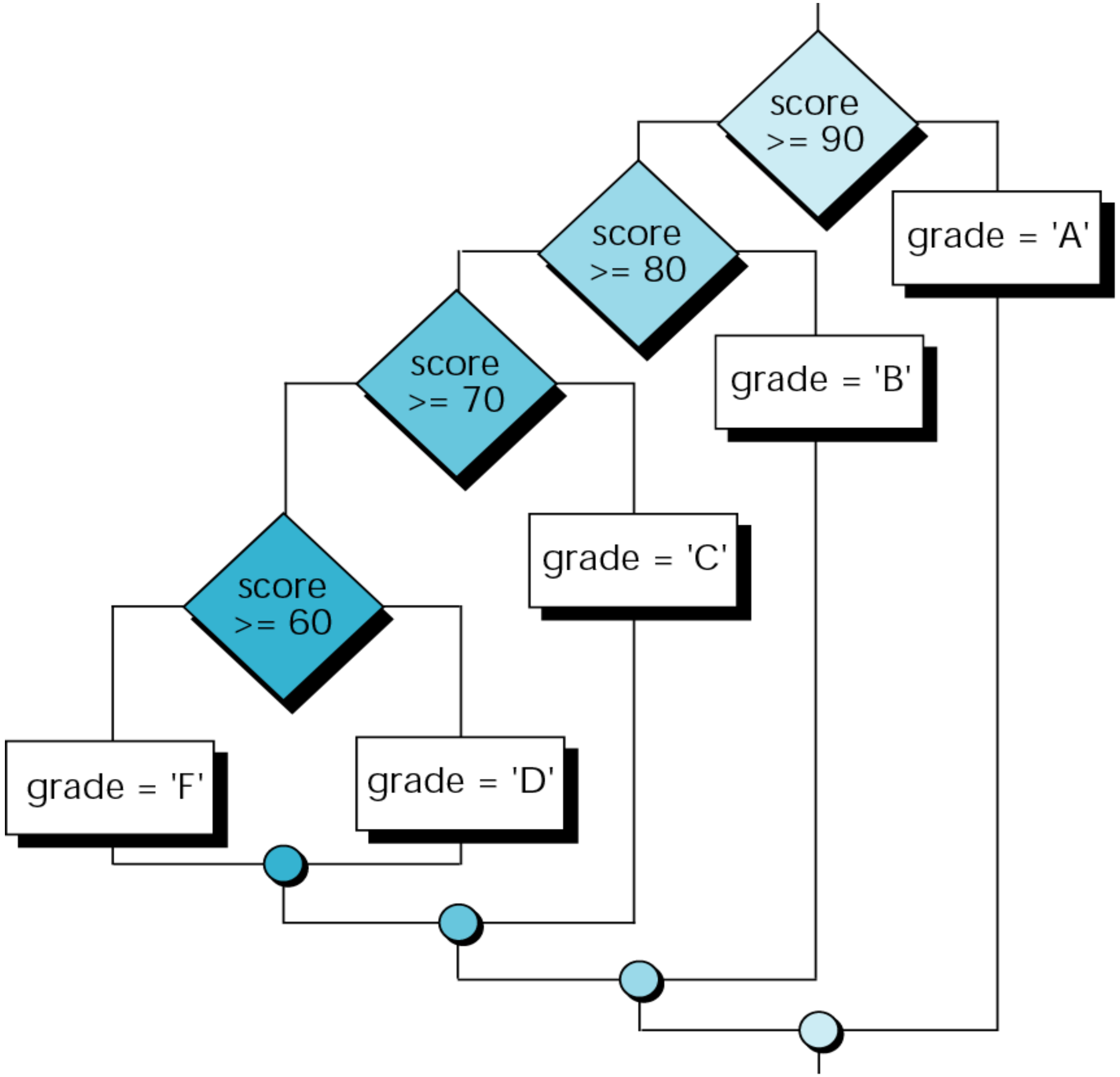
(b) Code

do case 1

do case 2

do default

Figure 5-23 The else…if for Program 5-9



Note:

The else-if is an artificial C++ construct that is only used when

1. ***The selection variable is not an integral, and***

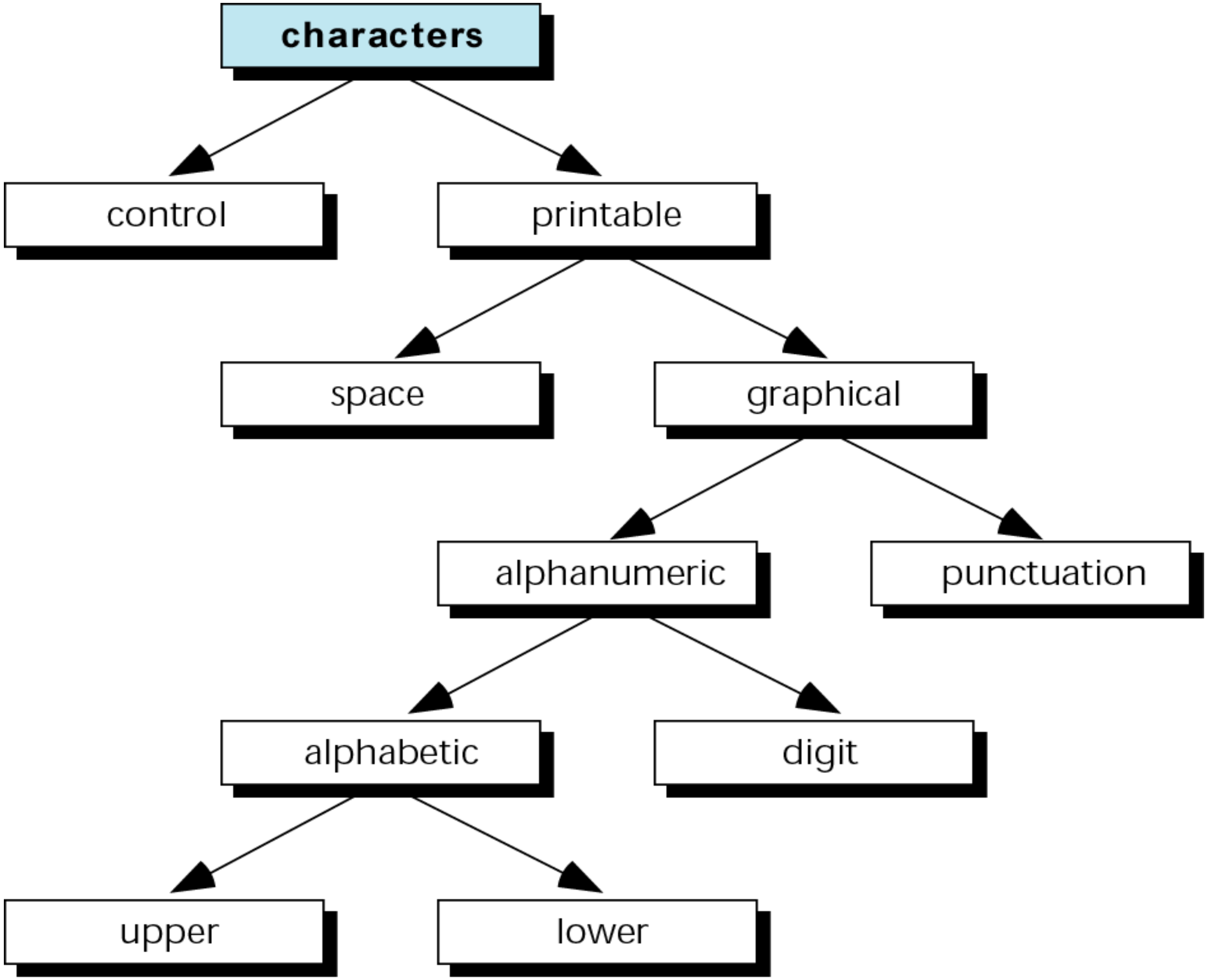
2. ***The same variable is being tested in the expressions.***

5.4

More Standard Library Functions

Brooks/Cole
Thomson Learning

Figure 5-24 Classifications of the character type

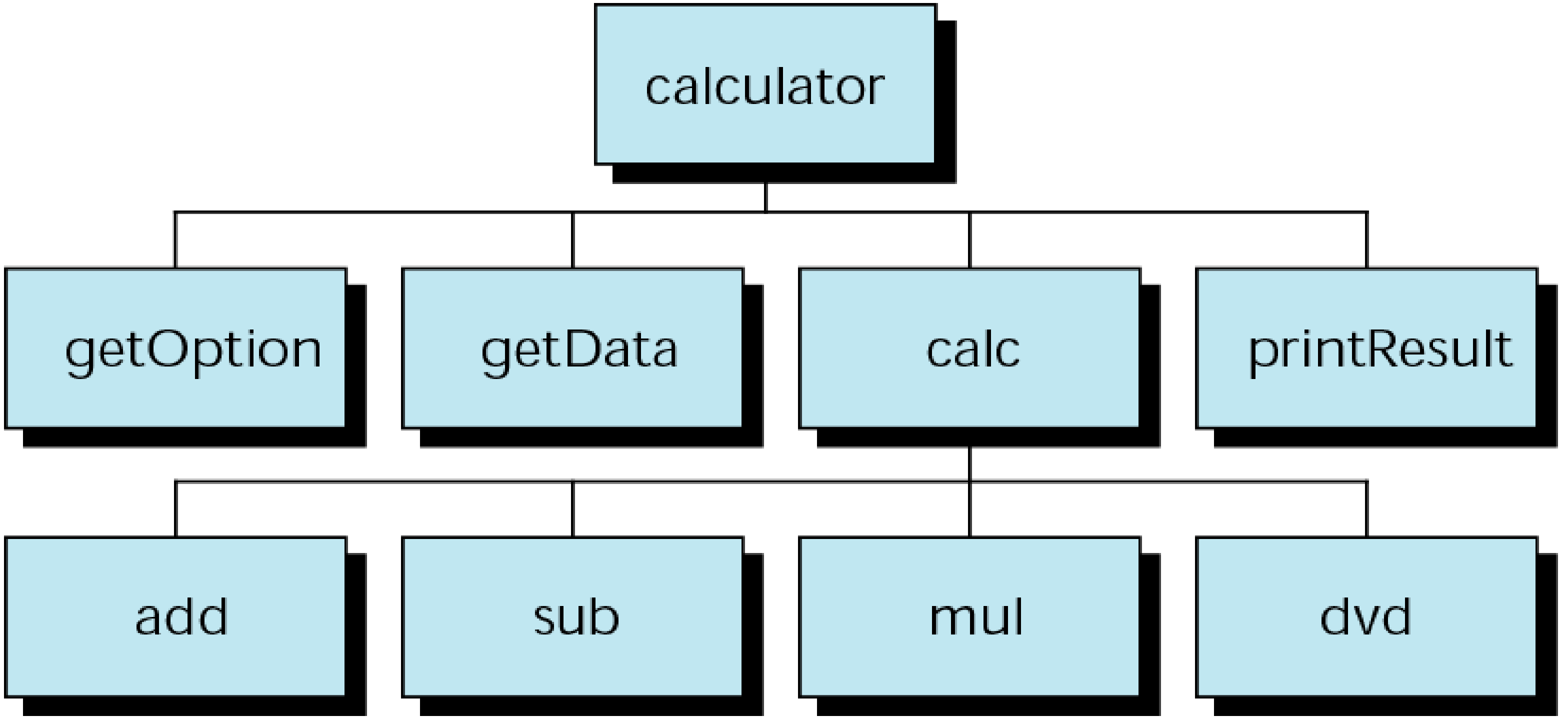


Brooks/Cole
Thomson Learning™

5.5

A Menu Program

Figure 5-25 Design for menu-driven calculator

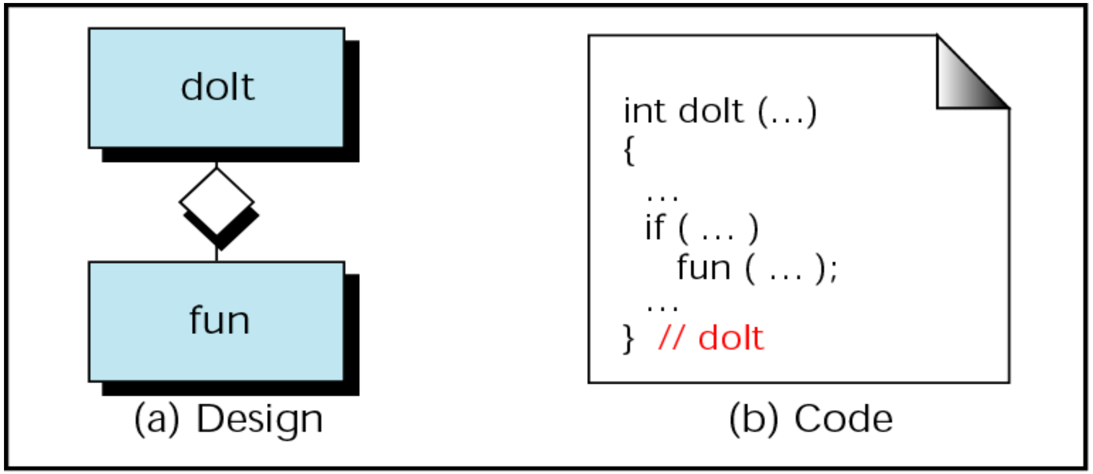


Software Engineering and Programming Style

Note:

Avoid compound negative statements!

Figure 5-26 Structure chart symbols for selection

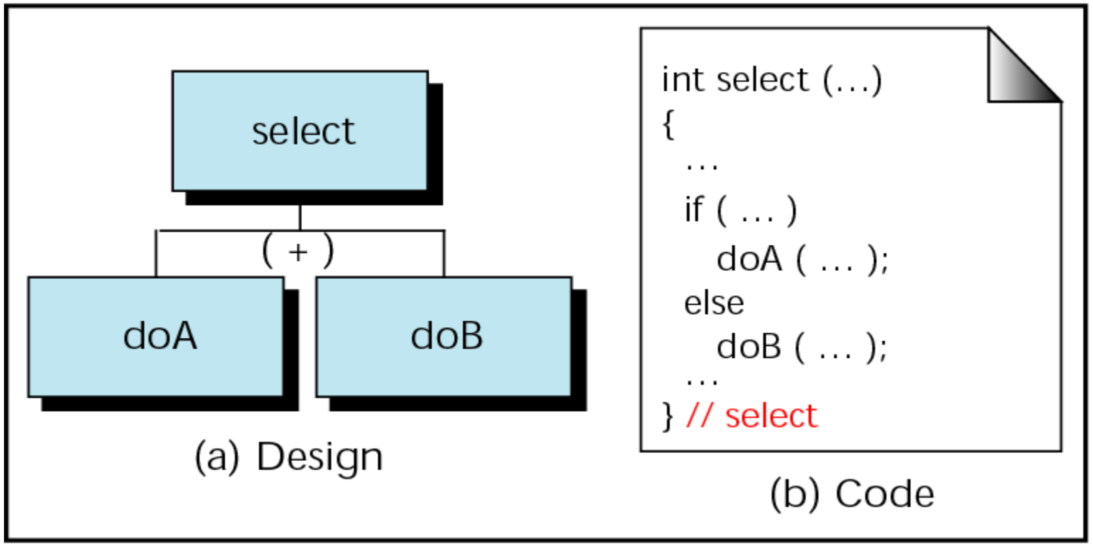


(a) Design

```
int doIt (...)
{
  ...
  if ( ... )
     fun ( ... );
  ...
}  // doIt
```

(b) Code

conditional



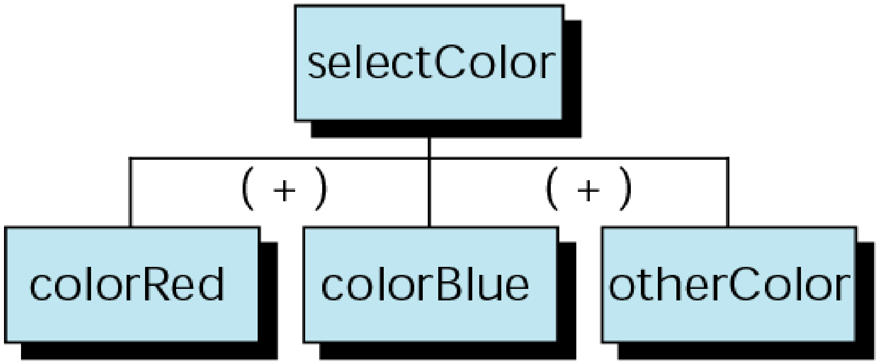
(a) Design

```
int select (...)
{
  ...
  if ( ... )
     doA ( ... );
  else
     doB ( ... );
  ...
} // select
```

(b) Code

exclusive or

Figure 5-27 Multiway selection in a structure chart



(a) Design

```
switch (color)
{
     case 'R'     : colorRed   (…);
                    break ;
     case 'B'     : colorBlue  (…);
                    break ;
     default      : otherColor (…);
} // switch
```

(b) Code