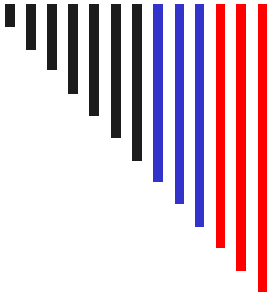


Chapter 5
Selection—Making Decisions

Objectives

- To understand how decisions are made in a computer
- To understand the logical operators: *and*, *or*, and *not*
- To understand how a C program evaluates a logical expression
- To write programs using logical and comparative operators
- To write programs that use two-way selection: *if ... else* statements
- To write programs that use multi-way selection: *switch* and *else ...if*
- To understand C's classification of characters
- To write programs that use C's character functions
- To be able to design a structure chart that specifies function selection

5-1 Logical Data and Operators

A piece of data is called logical if it conveys the idea of true or false. In real life, logical data (true or false) are created in answer to a question that needs a yes–no answer. In computer science, we do not use yes or no, we use true or false.

Topics discussed in this section:

Logical Data in C
Logical Operators
Evaluating Logical Expressions
Comparative Operators

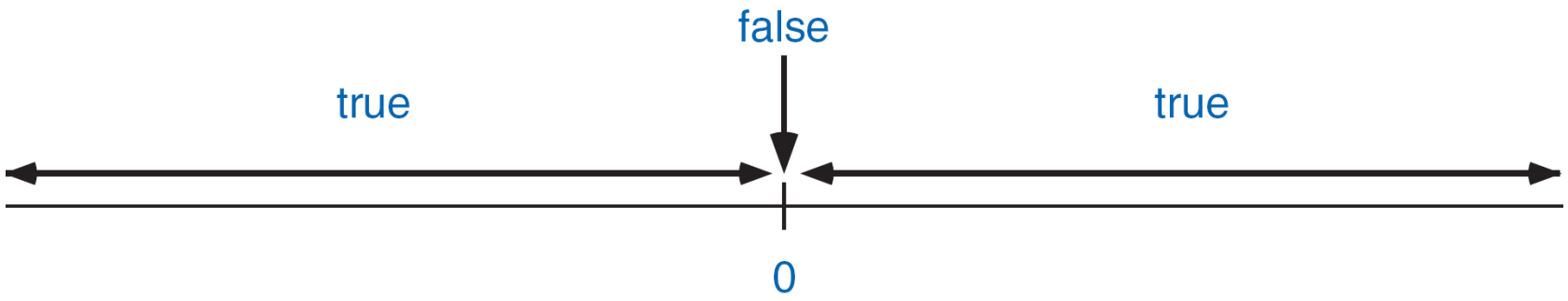


FIGURE 5-1 ***true*** **and** ***false*** **on the Arithmetic Scale**

not

x	!x
false	true
true	false

and

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

or

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

FIGURE 5-2 Logical Operators Truth Table

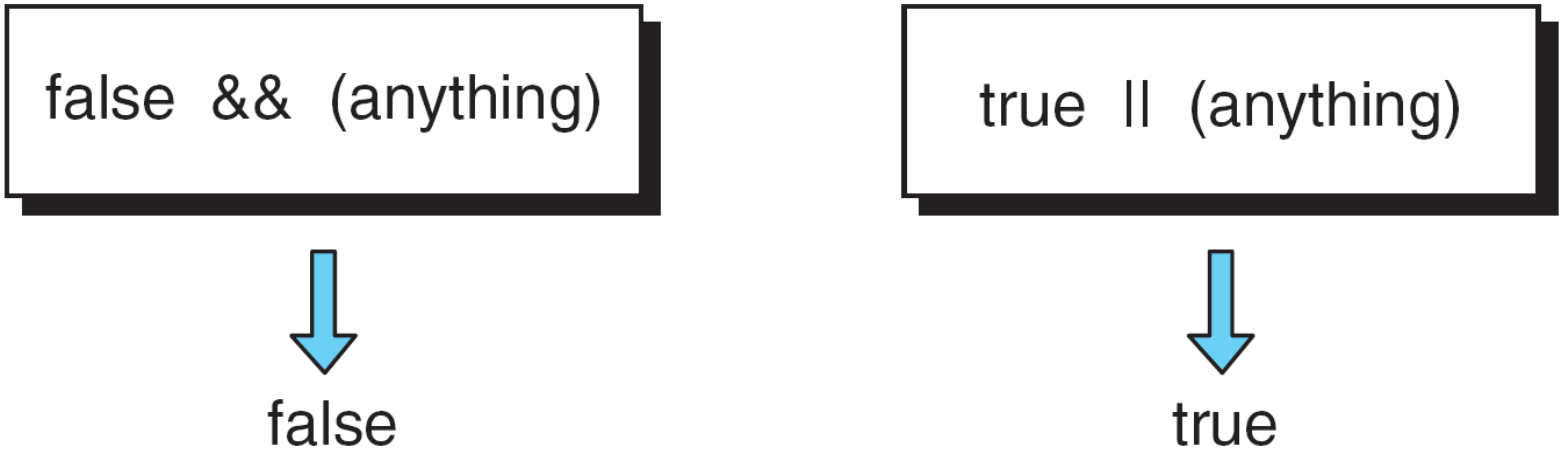


FIGURE 5-3 Short-circuit Methods for *and* /*or*

PROGRAM 5-1 Logical Expressions

```
 1  /* Demonstrate the results of logical operators.
 2        Written by:
 3        Date:
 4  */
 5  #include <stdio.h>
 6  #include <stdbool.h>
 7
 8  int main (void)
 9  {
10  // Local Declarations
11     bool a = true;
12     bool b = true;
13     bool c = false;
14
15  // Statements
16     printf("    %2d AND     %2d: %2d\n", a, b,  a &&  b);
17     printf("    %2d AND     %2d: %2d\n", a, c,  a &&  c);
18     printf("    %2d AND     %2d: %2d\n", c, a,  c &&  a);
```

PROGRAM 5-1 Logical Expressions

```
19      printf("    %2d OR      %2d: %2d\n", a, c,  a ||  c);
20      printf("    %2d OR      %2d: %2d\n", c, a,  c ||  a);
21      printf("    %2d OR      %2d: %2d\n", c, c,  c ||  c);
22      printf("NOT %2d AND NOT %2d: %2d\n", a, c, !a && !c);
23      printf("NOT %2d AND     %2d: %2d\n", a, c, !a &&  c);
24      printf("    %2d AND NOT %2d: %2d\n", a, c,  a && !c);
25      return 0;
26  }   // main
```

```
Results:
      1 AND       1:  1
      1 AND       0:  0
      0 AND       1:  0
      1 OR        0:  1
      0 OR        1:  1
      0 OR        0:  0
NOT   1 AND NOT   0:  0
NOT   1 AND       0:  0
      1 AND NOT   0:  1
```

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

FIGURE 5-4 Relational Operators

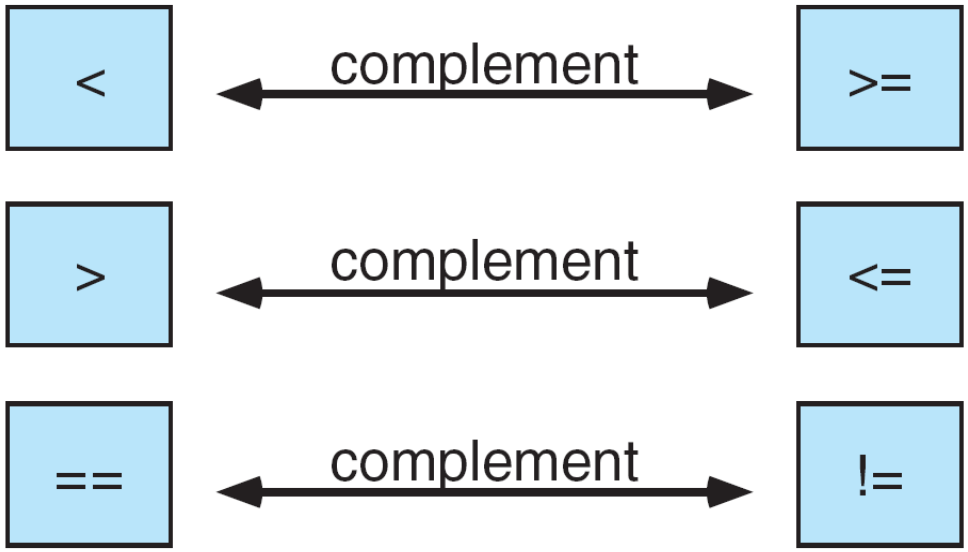


FIGURE 5-5 Comparative Operator Complements

Original Expression	Simplified Expression
`!(x < y)`	`x >= y`
`!(x > y)`	`x <= y`
`!(x != y)`	`x == y`
`!(x <= y)`	`x > y`
`!(x >= y)`	`x < y`
`!(x == y)`	`x != y`

Table 5-1 Examples of Simplifying Operator Complements

PROGRAM 5-2 Comparative Operators

```
 1  /* Demonstrates the results of relational operators.
 2        Written by:
 3        Date:
 4  */
 5  #include <stdio.h>
 6
 7  int main (void)
 8  {
 9  // Local Declarations
10     int a = 5;
11     int b = -3;
12
13  // Statements *
14     printf(" %2d <  %2d is %2d\n", a, b,  a <  b);
15     printf(" %2d == %2d is %2d\n", a, b,  a == b);
16     printf(" %2d != %2d is %2d\n", a, b,  a != b);
17     printf(" %2d >  %2d is %2d\n", a, b,  a >  b);
```

PROGRAM 5-2 Comparative Operators

```
18      printf(" %2d <= %2d is %2d\n", a, b,  a <= b);
19      printf(" %2d >= %2d is %2d\n", a, b,  a >= b);
20      return 0;
21  }   // main
```

```
Results:
5 <  -3 is  0
5 == -3 is  0
5 != -3 is  1
5 >  -3 is  1
5 <= -3 is  0
5 >= -3 is  1
```

5-2 Two-Way Selection

The decision is described to the computer as a conditional statement that can be answered either true or false. If the answer is true, one or more action statements are executed. If the answer is false, then a different action or set of actions is executed.

Topics discussed in this section:

***if*…*else* and Null *else* Statement**
Nested *if* Statements and Dangling *else* Problem
Simplifying *if* Statements
Conditional Expressions

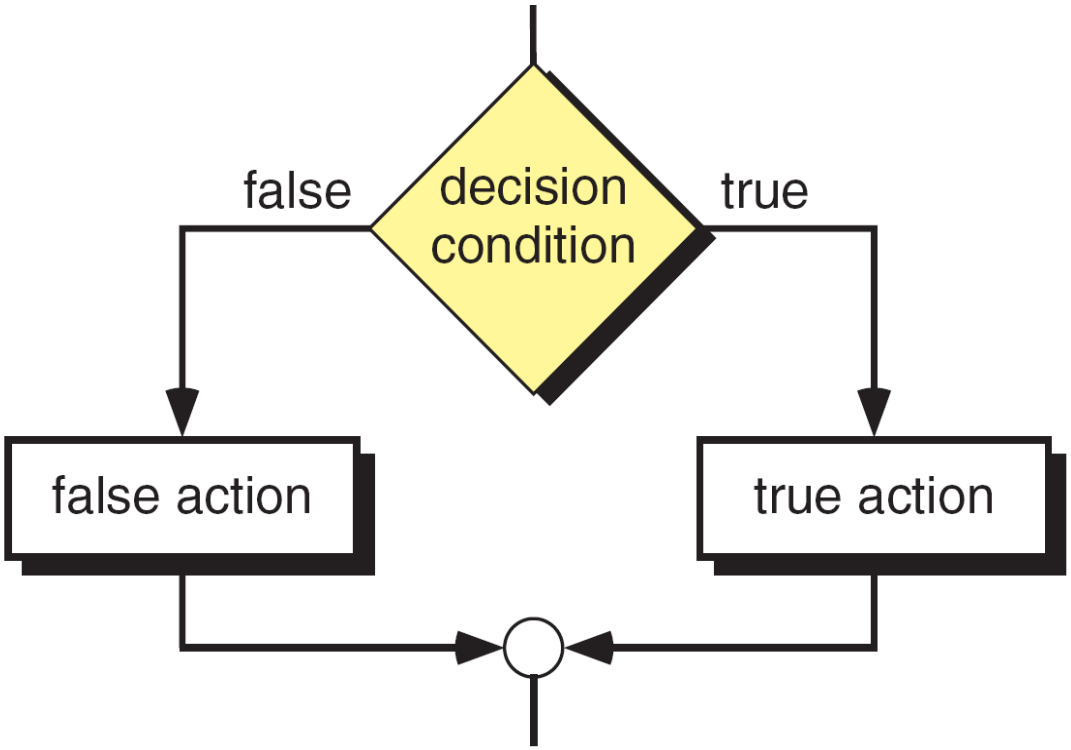
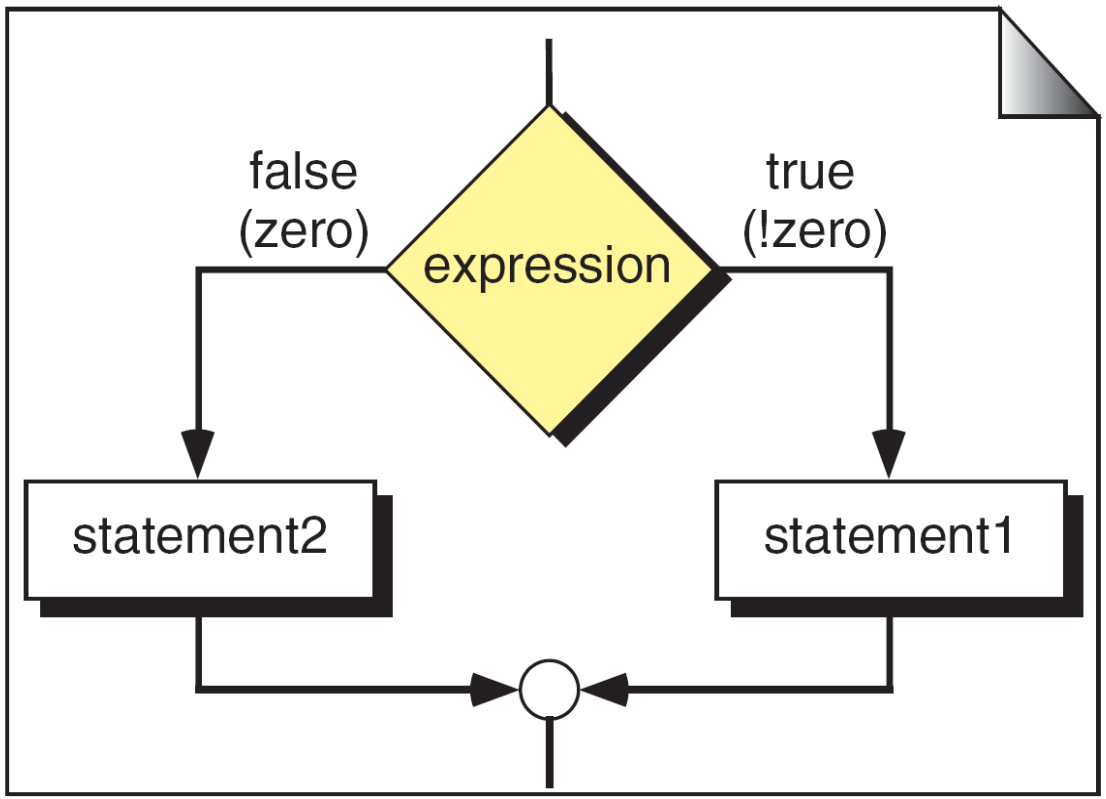


FIGURE 5-6 Two-way Decision Logic



(a) Logical Flow

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

(b) Code

FIGURE 5-7 ***if...else*** **Logic Flow**

1. The expression must be enclosed in parentheses.
2. No semicolon (;) is needed for an *if…else* statement; statement 1 and statement 2 may have a semicolon as required by their types.
3. The expression can have a side effect.
4. Both the true and the false statements can be any statement (even another *if…else* statement) or they can be a null statement.
5. Both statement 1 and statement 2 must be one and only one statement. Remember, however, that multiple statements can be combined into a compound statement through the use of braces.
6. We can swap the position of statement 1 and statement 2 if we use the complement of the original expression.

Table 5-2 Syntactical Rules for *if…else* Statements

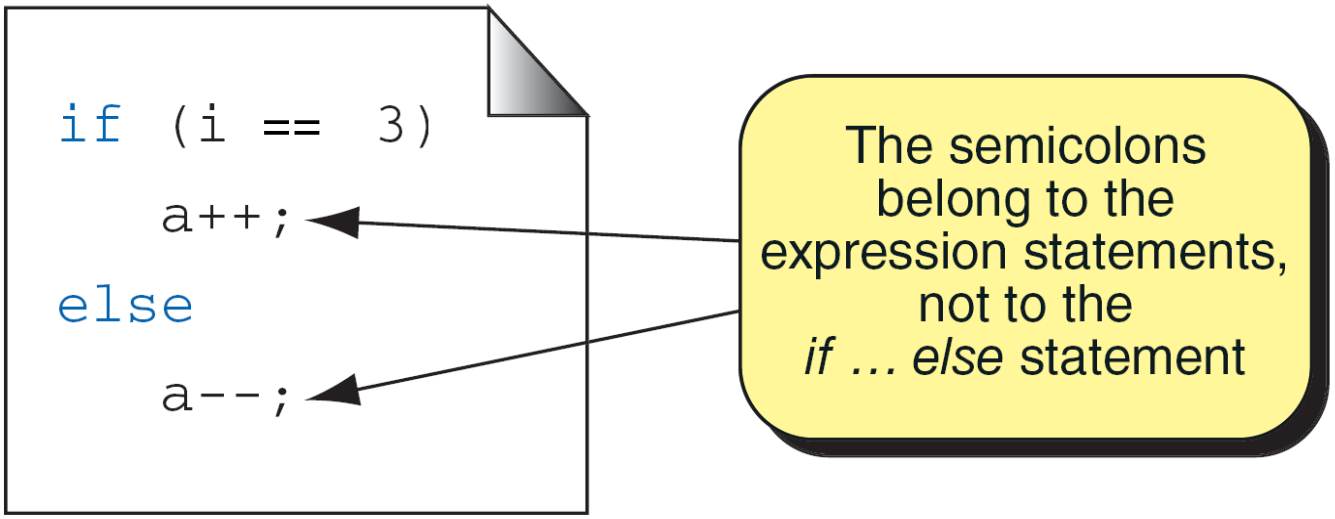


FIGURE 5-8 A Simple *if...else* Statement

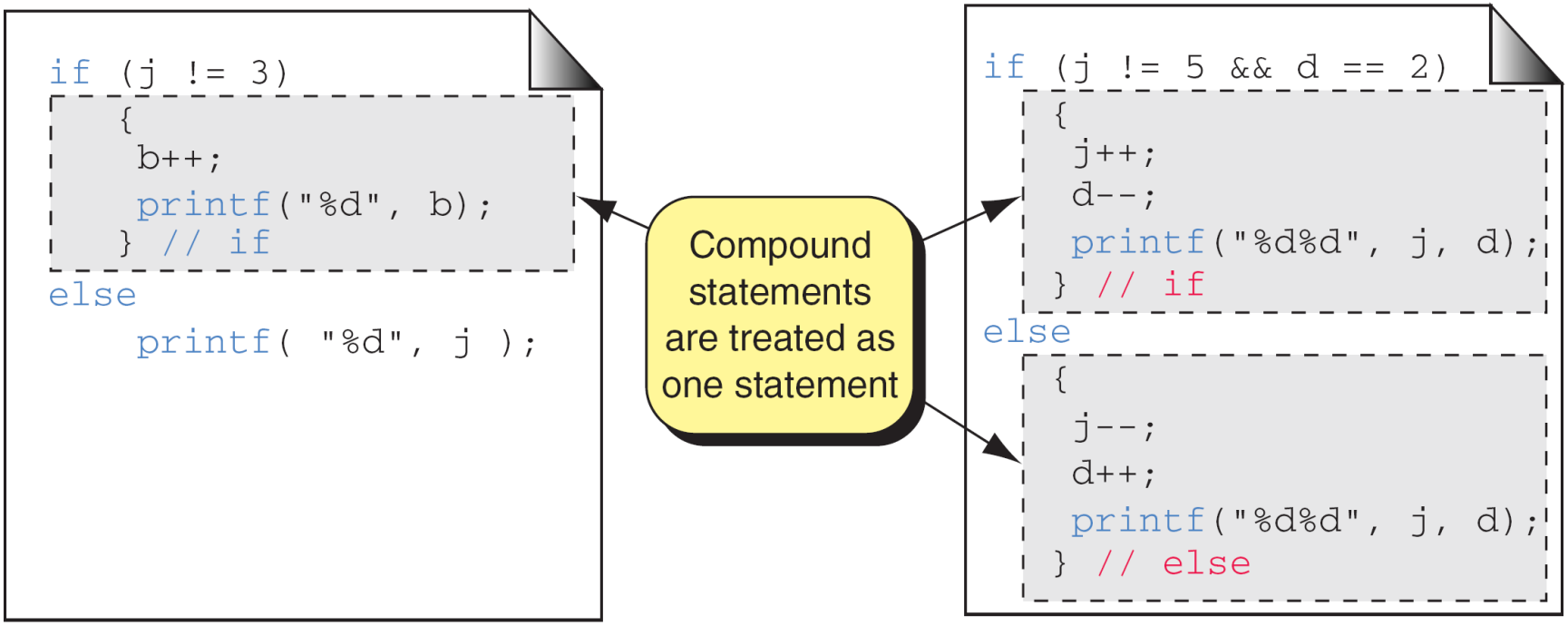


FIGURE 5-9 Compound Statements in an *if...else*

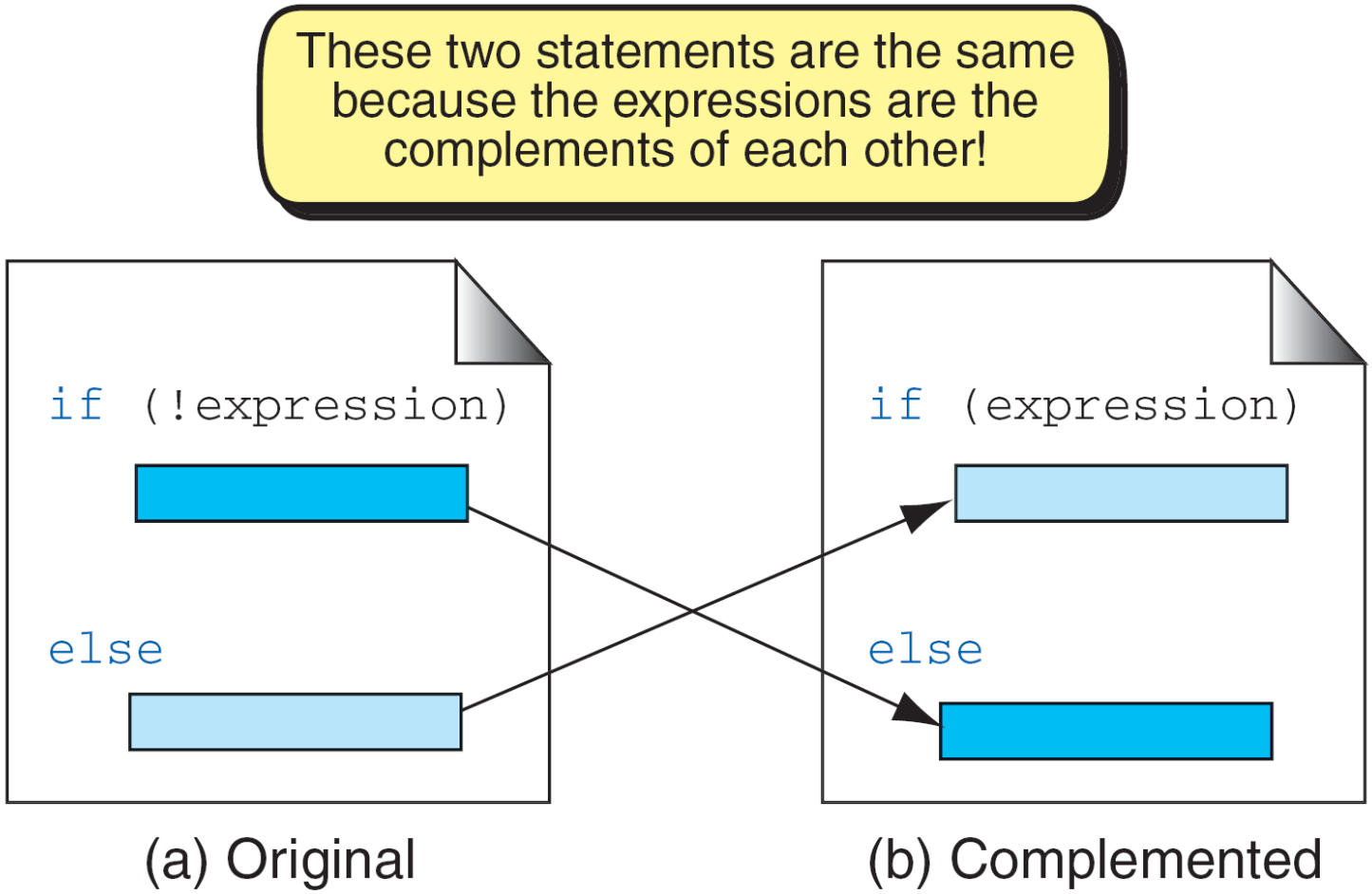


FIGURE 5-10 Complemented *if...else* Statements

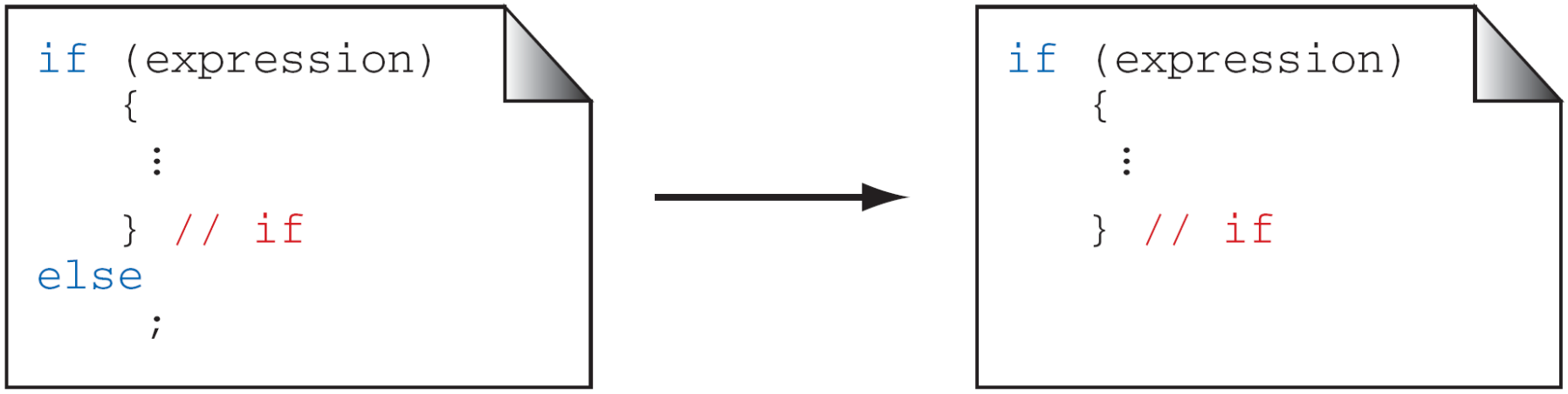


FIGURE 5-11 A Null *else* Statement

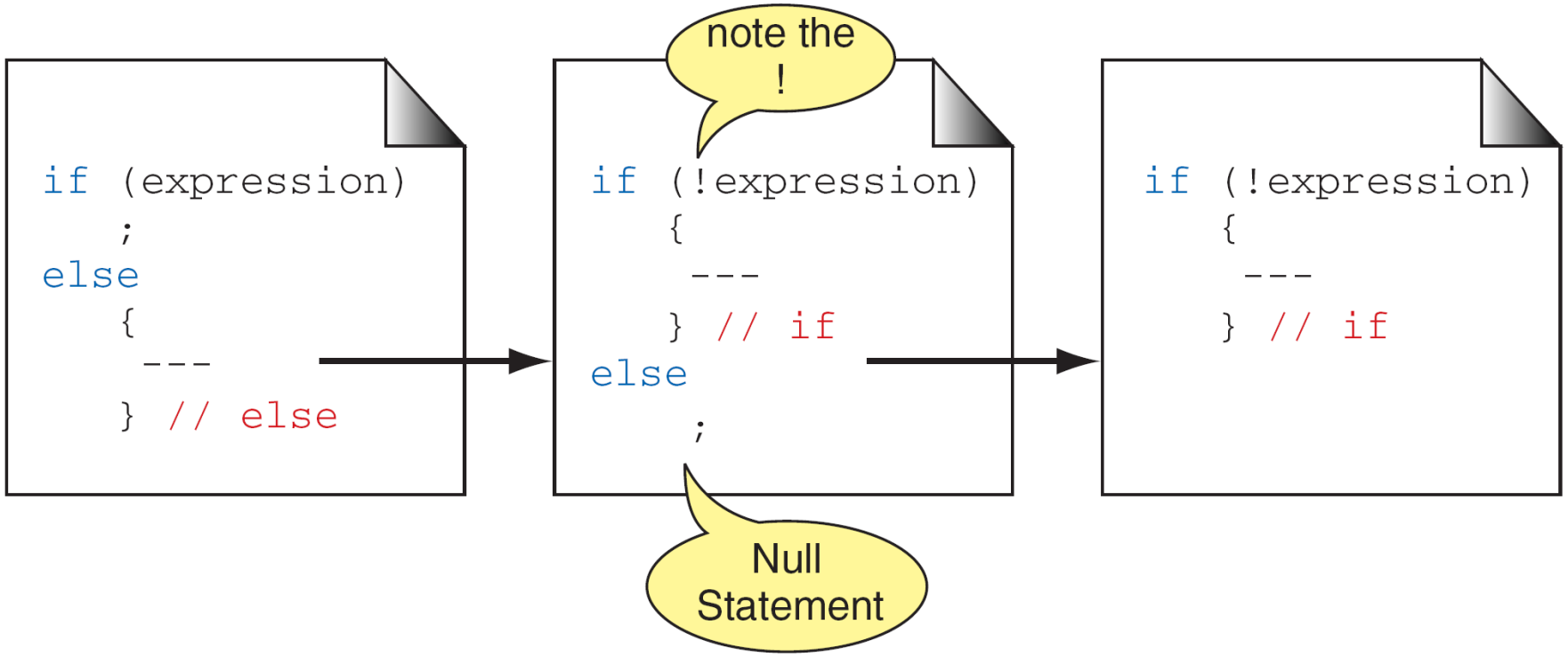


FIGURE 5-12 A Null *if* Statement

De Morgan's Rule

- !(x && y) → !x || !y
- !(x || y) → !x && !y

- Is it a good style?
- Positive logic is easier to read and understand than negative logic.

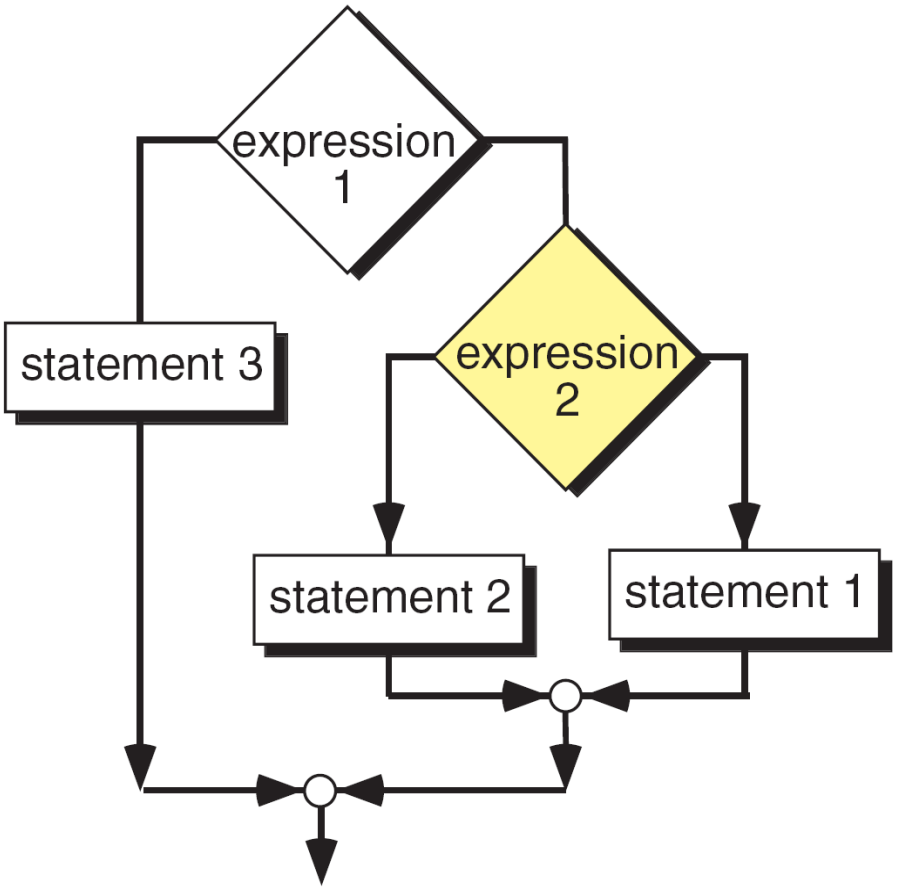
PROGRAM 5-3 Two-way Selection

```
1  /* Two-way selection.
2        Written by:
3        Date:
4  */
5  #include <stdio.h>
6
7  int main (void)
8  {
9     // Local Declarations
10    int a;
11    int b;
12
13    // Statements
14    printf("Please enter two integers: ");
15    scanf ("%d%d", &a, &b);
16
```

PROGRAM 5-3 Two-way Selection

```
17      if (a <= b)
18          printf("%d <= %d\n", a, b);
19      else
20          printf("%d > %d\n", a, b);
21
22      return 0;
23  }  // main
```

```
Results:
Please enter two integers: 10 15
10 <= 15
```



(a) Logic flow

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

(b) Code

FIGURE 5-13 Nested *if* Statements

PROGRAM 5-4 Nested *if* Statements

```
 1  /* Nested if in two-way selection.
 2        Written by:
 3        Date:
 4  */
 5  #include <stdio.h>
 6
 7  int main (void)
 8  {
 9  // Local Declarations
10     int a;
11     int b;
12
13  // Statements
14     printf("Please enter two integers: ");
15     scanf ("%d%d", &a, &b);
16
```

PROGRAM 5-4 Nested *if* Statements

```
17      if (a <= b)
18         if (a < b)
19            printf("%d < %d\n", a, b);
20         else
21            printf("%d == %d\n", a, b);
22      else
23         printf("%d > %d\n", a, b);
24
25      return 0;
26   }  // main
```

```
Results:
Please enter two integers: 10 10
10 == 10
```

Note

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

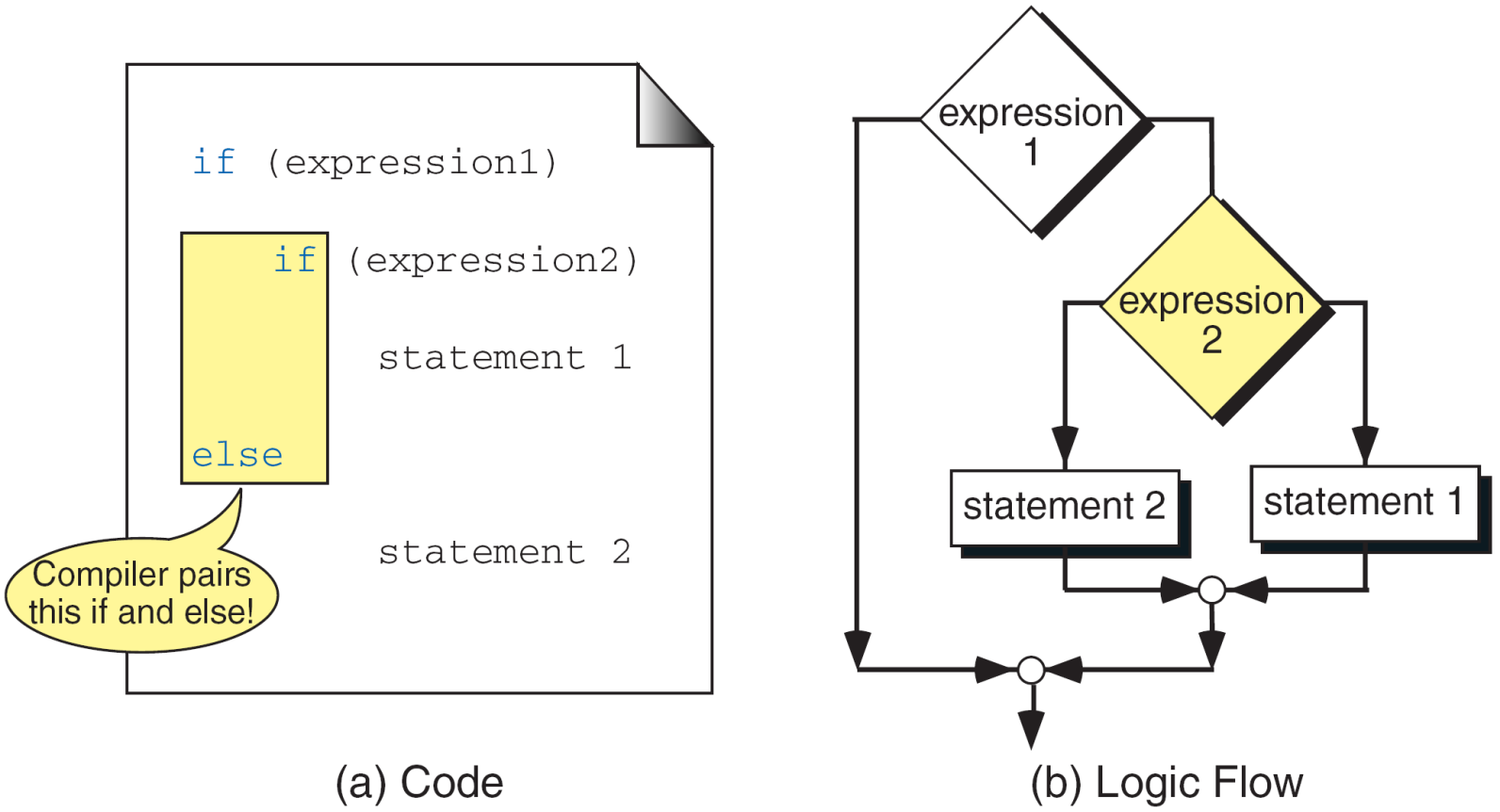
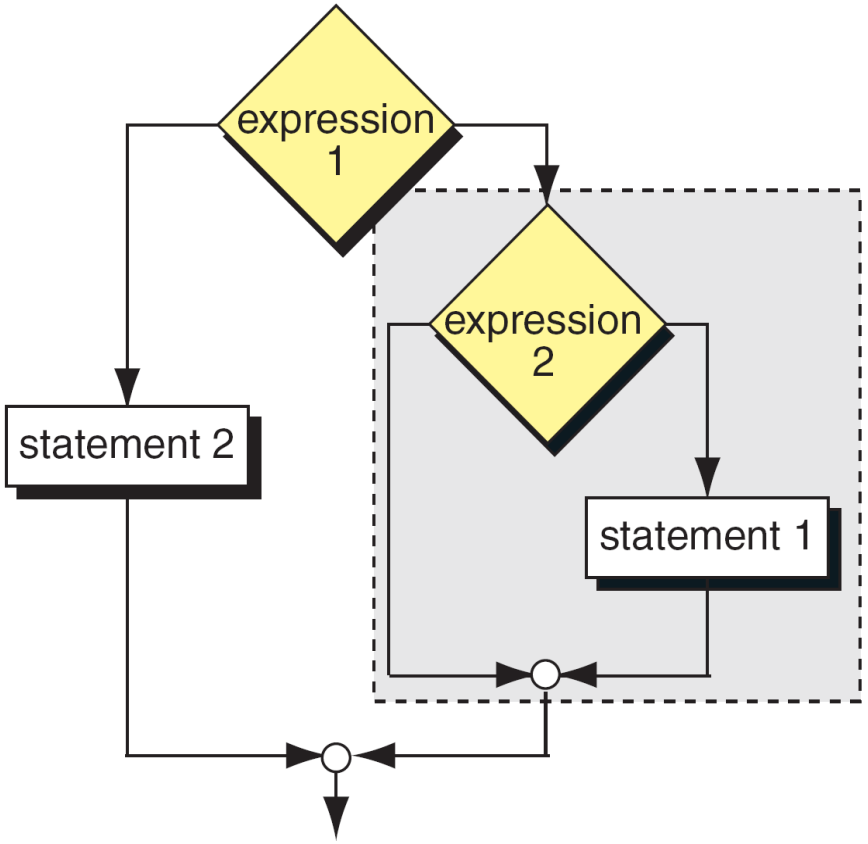
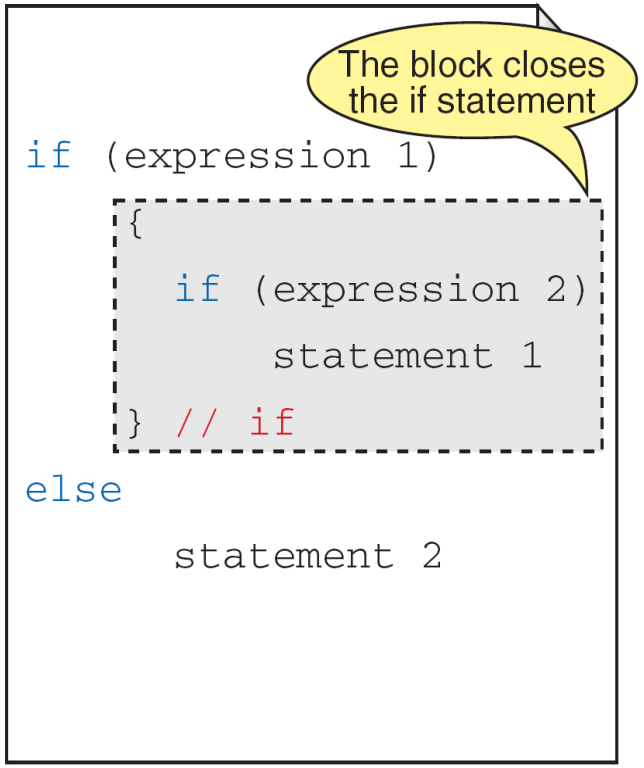


FIGURE 5-14 Dangling *else*



(a) Logic Flow

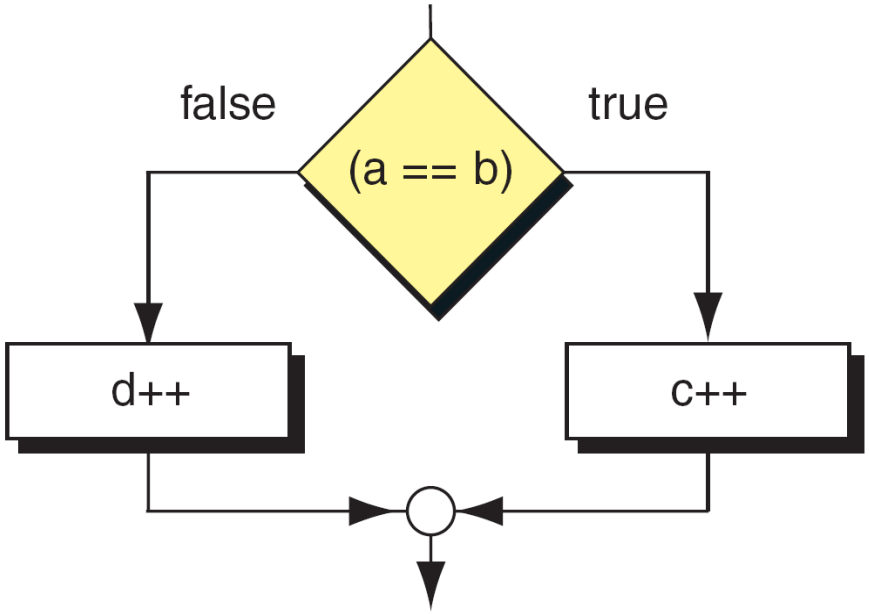


(b) Code

FIGURE 5-15 Dangling *else* Solution

Original Statement	Simplified Statement
`if (a != 0)` `  statement`	`if (a)` `  statement`
`if (a == 0)` `  statement`	`if (!a)` `  statement`

Table 5-3 Simplifying the Condition



(a) Logic Flow

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

(b) Code

FIGURE 5-16 Conditional Expression

Case 1: Total Income 23,000			Case 2: Total Income 18,000		
Income in Bracket	Tax Rate	Tax	Income in Bracket	Tax Rate	Tax
(1) 10,000	2%	200	(1) 10,000	2%	200
(2) 10,000	5%	500	(2) 8,000	5%	400
(3) 3,000	7%	210	(3) none	7%	0
Total Tax		910	Total Tax		600

Table 5-4 **Examples of Marginal Tax Rates**

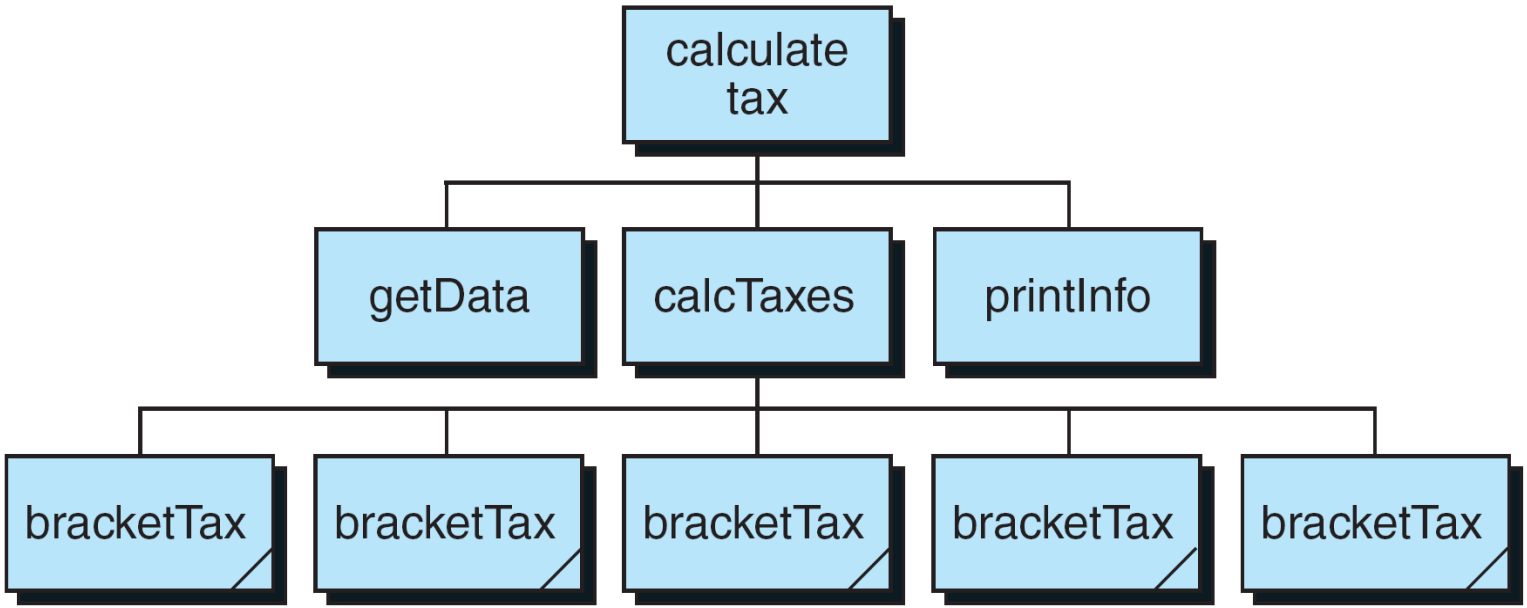


FIGURE 5-17 Design for Calculate Taxes

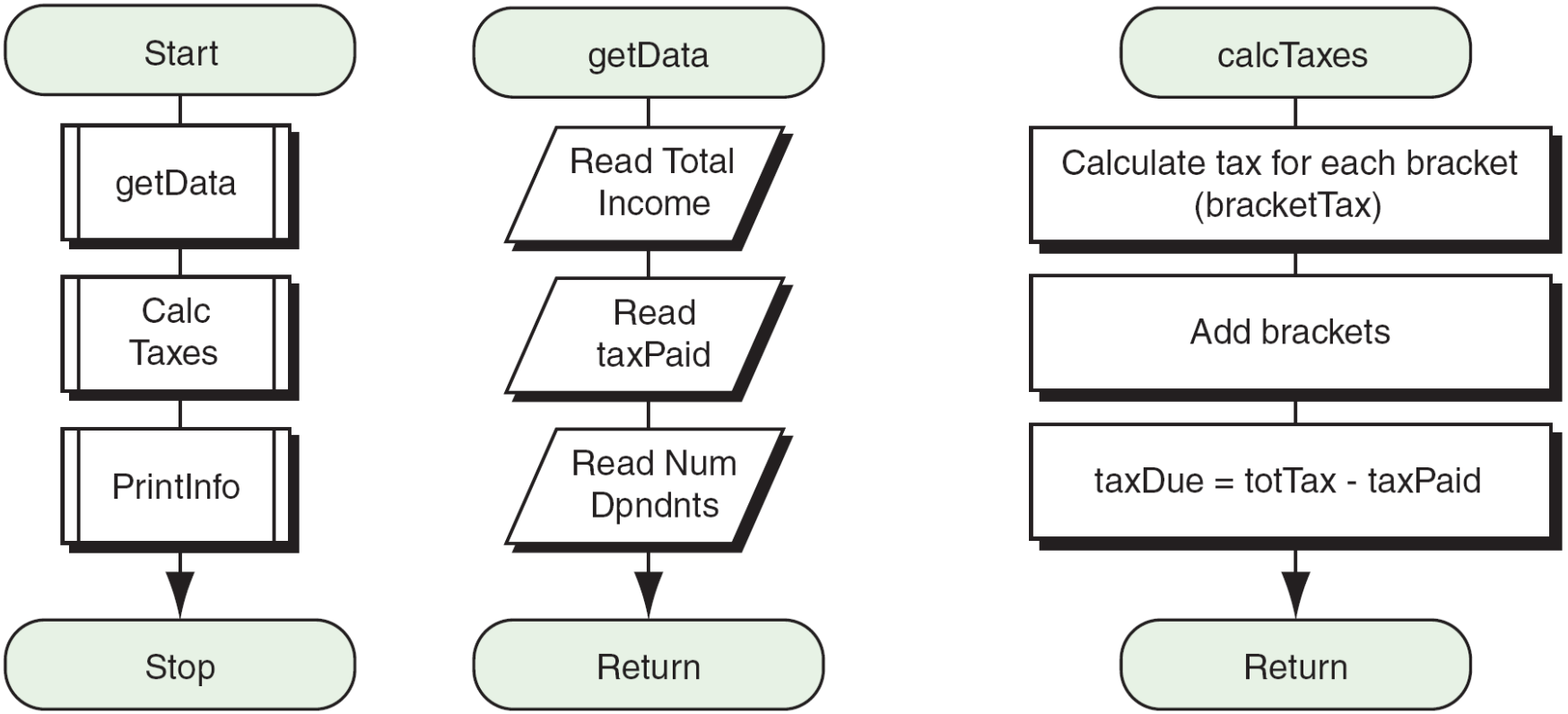


FIGURE 5-18 Design for Program 5-5 (Part I)

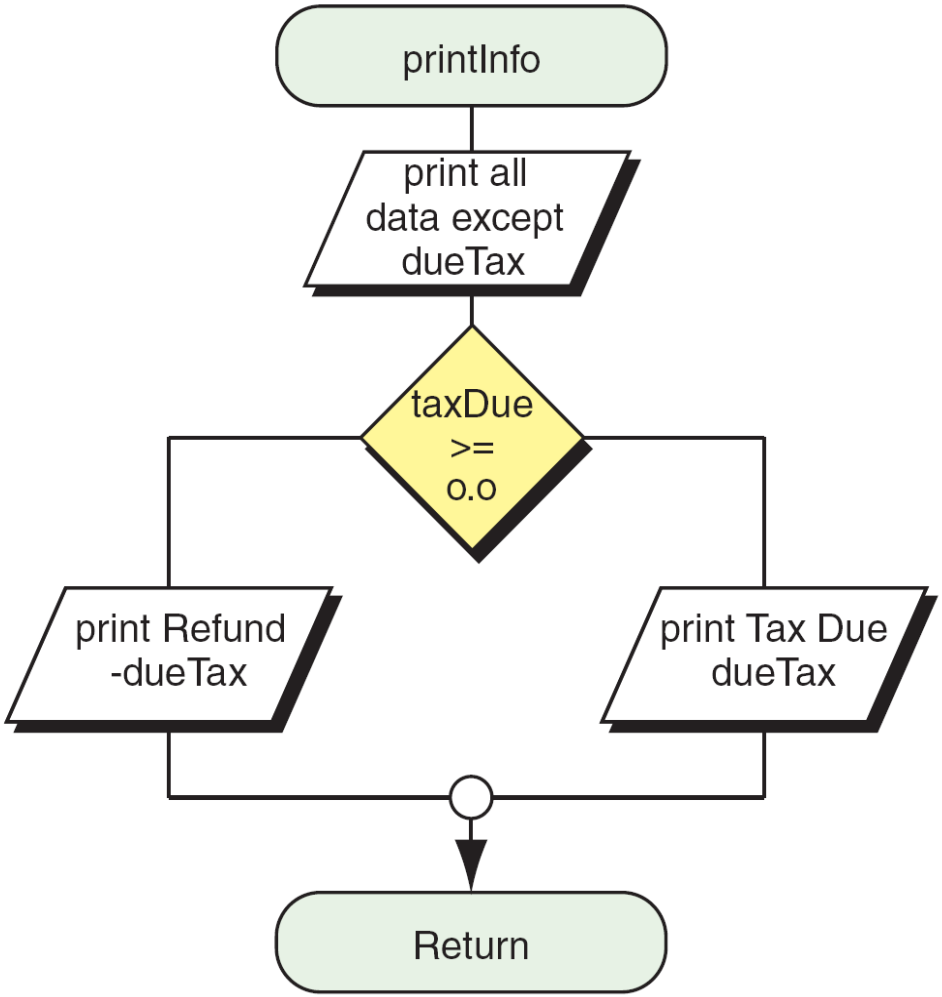


FIGURE 5-18 Design for Program 5-5 (Part II)

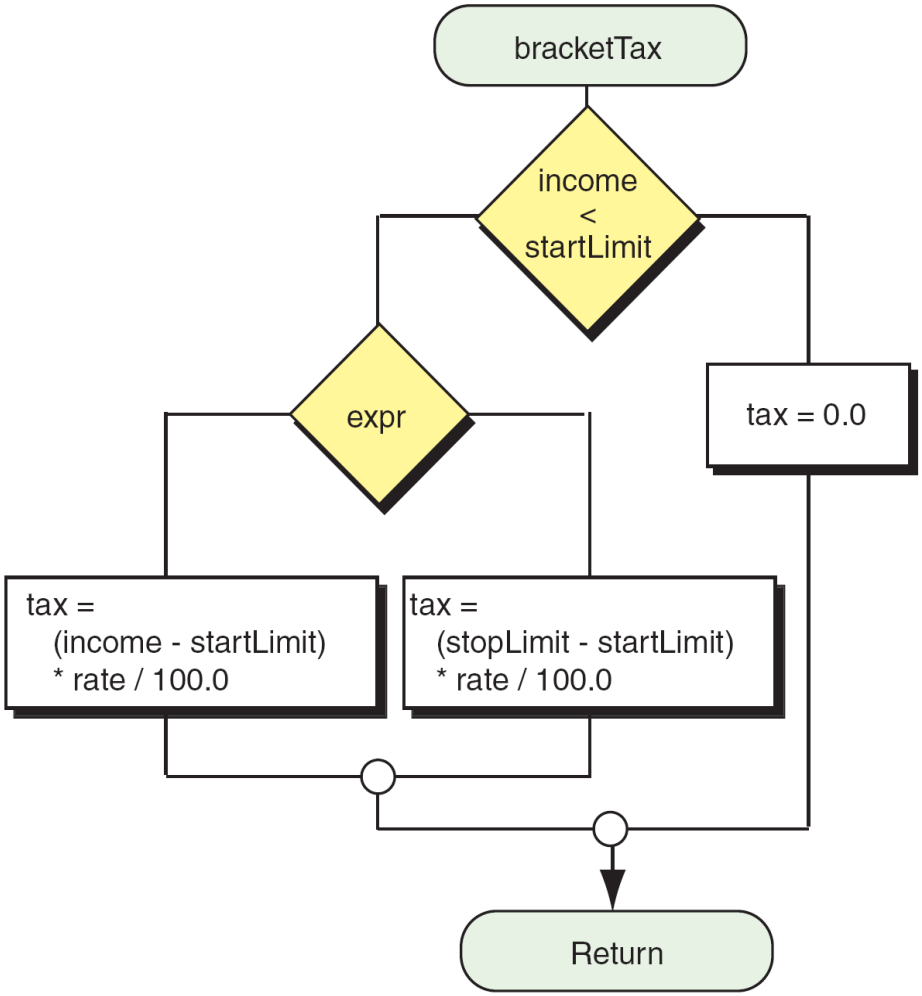


FIGURE 5-18 Design for Program 5-5 (Part III)

PROGRAM 5-5 Calculate Taxes

```
/* Calculate the tax due or the refund for a family based
   on the following imaginary formula.
   1. For each dependent deduct $1,000 from income.
   2. Determine tax rate from the following brackets:
         bracket    taxable income        tax rate
            1              <= 10000           2%
            2         10001 - 20000           5%
            3         20001 - 30000           7%
            4         30001 - 50000          10%
            5         50001 and up           15%
   Then print the amount of tax or the refund.

      Written by:
      Date:
*/
#include <stdio.h>

#define LOWEST  0000000.00
#define HIGHEST 1000000.00
```

PROGRAM 5-5 Calculate Taxes

```
20
21  #define LIMIT1    10000.00
22  #define LIMIT2    20000.00
23  #define LIMIT3    30000.00
24  #define LIMIT4    50000.00
25
26  #define RATE1 02
27  #define RATE2 05
28  #define RATE3 07
29  #define RATE4 10
30  #define RATE5 15
31
32  #define DEDN_PER_DPNDNT 1000
33
34  // Function Declarations
35  void getData   (double* totalIncome, double* taxPaid,
36                  int*    numOfDpndnts);
37
```

PROGRAM 5-5 Calculate Taxes

```
38 void calcTaxes (double  totalIncome,
39                 double  taxPaid,
40                 int     numOfDpndnts,
41                 double* taxableIncome,
42                 double* totalTax,
43                 double* taxDue);
44
45 void printInformation (double totalIncome,
46                         double taxPaid,
47                         int    numOfDpndnts,
48                         double totalTax,
49                         double paidTax,
50                         double taxDue);
51
52 double bracketTax (double income,
53                    double startLimit,
54                    double stopLimit,
55                    int    rate);
56
```

PROGRAM 5-5 Calculate Taxes

```
57  int main (void)
58  {
59  // Local Declarations
60     int    numOfDpndnts;
61     double taxDue;
62     double taxPaid;
63     double totalIncome;
64     double taxableIncome;
65     double totalTax;
66
67  // Statements
68     getData   (&totalIncome,  &taxPaid, &numOfDpndnts);
69     calcTaxes (totalIncome,    taxPaid,  numOfDpndnts,
70                &taxableIncome, &totalTax, &taxDue);
71     printInformation (totalIncome,  taxableIncome,
72                       numOfDpndnts, totalTax,
73                       taxPaid,      taxDue);
74     return 0;
75  }  // main
```

PROGRAM 5-5 Calculate Taxes

```
77  /* ================== getData ==================
78     This function reads tax data from the keyboard.
79        Pre  Nothing
80        Post Reads totalIncome, taxPaid, & numOfDpndnts
81  */
82  void getData ( double* totalIncome, double* taxPaid,
83                 int*    numOfDpndnts)
84  {
85  // Statements
86     printf("Enter your total income for last year: ");
87     scanf ("%lf", totalIncome);
88
89     printf("Enter total of payroll deductions    : ");
90     scanf ("%lf", taxPaid);
91
92     printf("Enter the number of dependents       : ");
93     scanf ("%d", numOfDpndnts);
94     return;
95  }  // getData
96
```

PROGRAM 5-5 Calculate Taxes

```
 97 /* ================= calcTaxes =================
 98    This function calculates the taxes due.
 99       Pre  Given-income, numOfDpndnts, & taxPaid
100       Post Tax income, total tax, and tax due
101            calculated
102 */
103 void calcTaxes (double  totInc,
104                 double  taxPaid,
105                 int     numOfDpndnts,
106                 double* taxableInc,
107                 double* totTax,
108                 double* taxDue)
109 {
110 // Statements
111    *taxableInc = totInc -
112                     (numOfDpndnts* DEDN_PER_DPNDNT);
113    *totTax =
114        bracketTax(*taxableInc, LOWEST, LIMIT1,  RATE1)
115      + bracketTax(*taxableInc, LIMIT1, LIMIT2,  RATE2)
116      + bracketTax(*taxableInc, LIMIT2, LIMIT3,  RATE3)
117      + bracketTax(*taxableInc, LIMIT3, LIMIT4,  RATE4)
118      + bracketTax(*taxableInc, LIMIT4, HIGHEST, RATE5);
```

PROGRAM 5-5 Calculate Taxes

```
119
120     *taxDue = *totTax - taxPaid;
121     return;
122  }  // calcTaxes
123
124  /* ================ printInformation ================
125     This function prints a table showing all information.
126        Pre  The parameter list
127        Post Prints the table
128  */
129  void printInformation (double totalIncome,
130                         double income,
131                         int    numDpndnts,
132                         double totalTax,
133                         double paidTax,
134                         double dueTax)
135  {
136  // Statements
```

PROGRAM 5-5 Calculate Taxes

```
137     printf("\nTotal income           :%10.2f\n",
138             totalIncome);
139     printf("Number of dependents  :%7d\n",   numDpndnts);
140     printf("Taxable income         :%10.2f\n", income);
141     printf("Total tax              :%10.2f\n", totalTax);
142     printf("Tax already paid       :%10.2f\n", paidTax);
143
144     if (dueTax >= 0.0)
145         printf("Tax due                :%10.2f\n", dueTax);
146     else
147         printf("Refund                :%10.2f\n", -dueTax);
148     return;
149 }   // printInformation
150
151 /* ================== bracketTax ==================
152     Calculates the tax for a particular bracket.
153        Pre    The taxableIncome
154        Post   Returns the tax for a particular bracket
155 */
```

PROGRAM 5-5 Calculate Taxes

```
156 double bracketTax (double income,    double startLimit,
157                    double stopLimit, int    rate)
158 {
159 // Local Declarations
160    double tax;
161
162 // Statements
163    if (income <= startLimit)
164        tax = 0.0;
165    else
166        if (income > startLimit && income <= stopLimit)
167            tax = (income - startLimit) * rate / 100.00;
168        else
169           tax = (stopLimit - startLimit) * rate / 100.00;
170
171    return tax;
172 }  // bracketTax
```

PROGRAM 5-5 Calculate Taxes

```
Results:
Enter your total income for last year: 15000
Enter total of payroll deductions: 250
Enter the number of dependents: 2
Total income: 15000.00
Number of dependents: 2
Taxable income: 13000.00
Total Tax: 350.00
Tax already paid: 250.00
Tax due: 100.00
```

5-3 Multiway Selection

In addition to two-way selection, most programming languages provide another selection concept known as multiway selection. Multiway selection chooses among several alternatives. C has two different ways to implement multiway selection: the switch statement and else-if construct.

Topics discussed in this section:

The *switch* Statement
The *else-if*

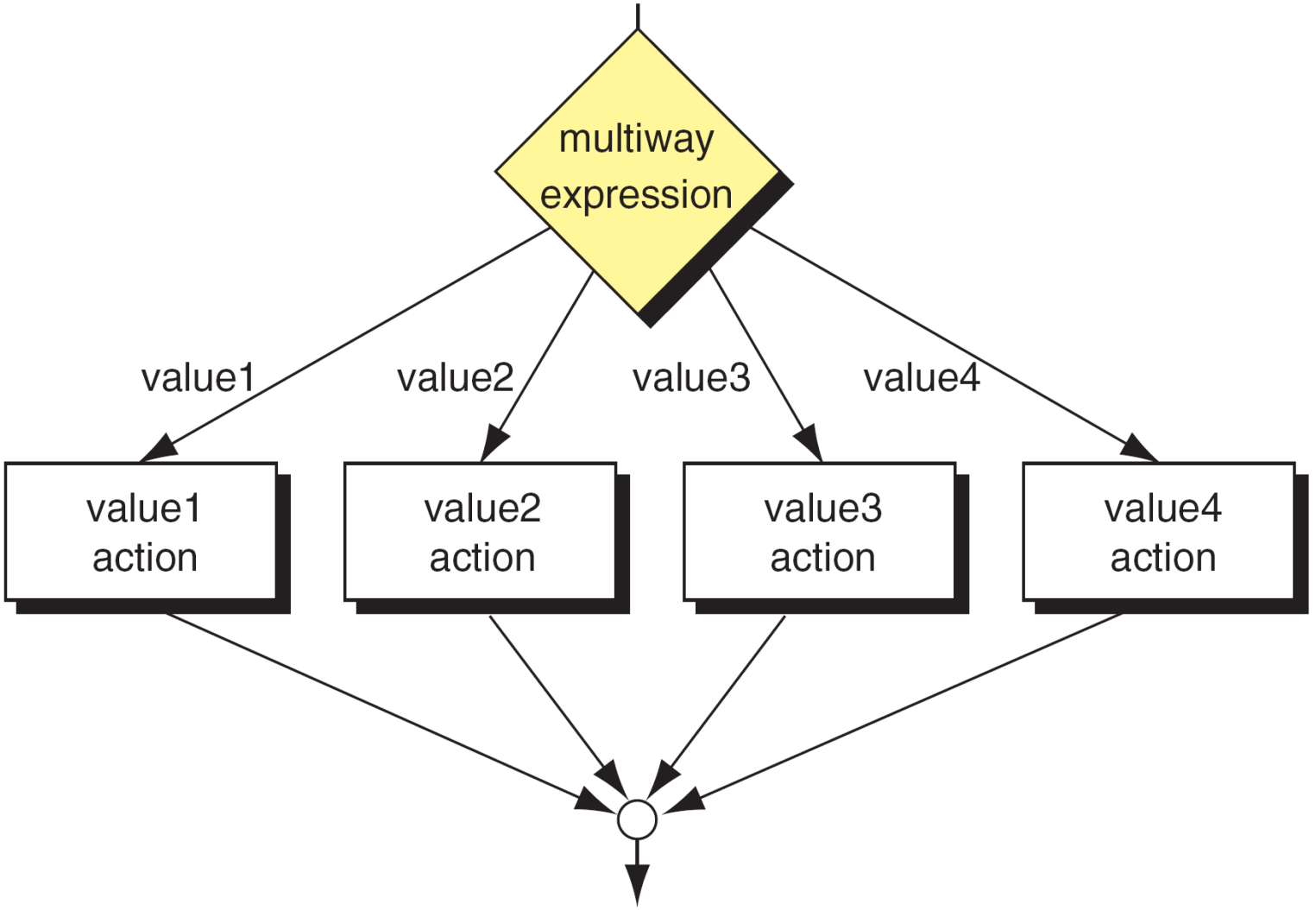


FIGURE 5-19 ***switch*** **Decision Logic**

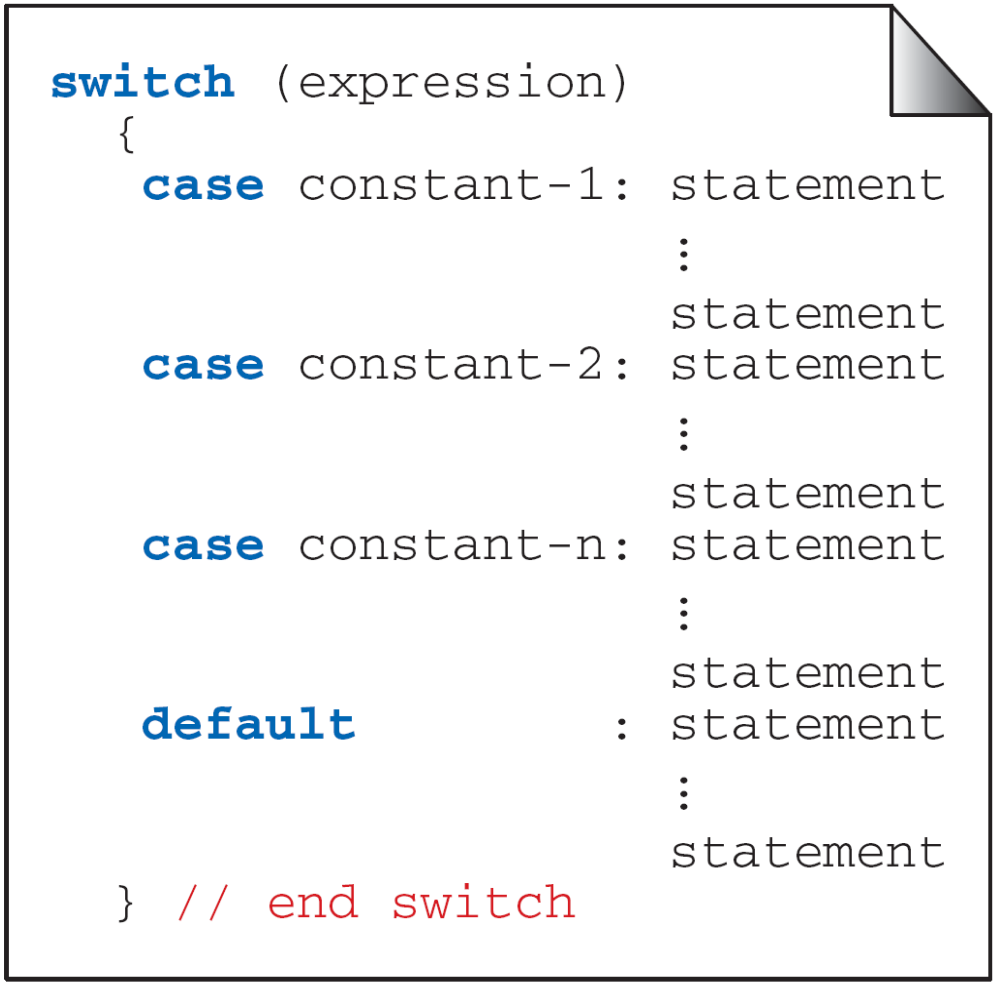


FIGURE 5-20 ***switch*** **Statement Syntax**

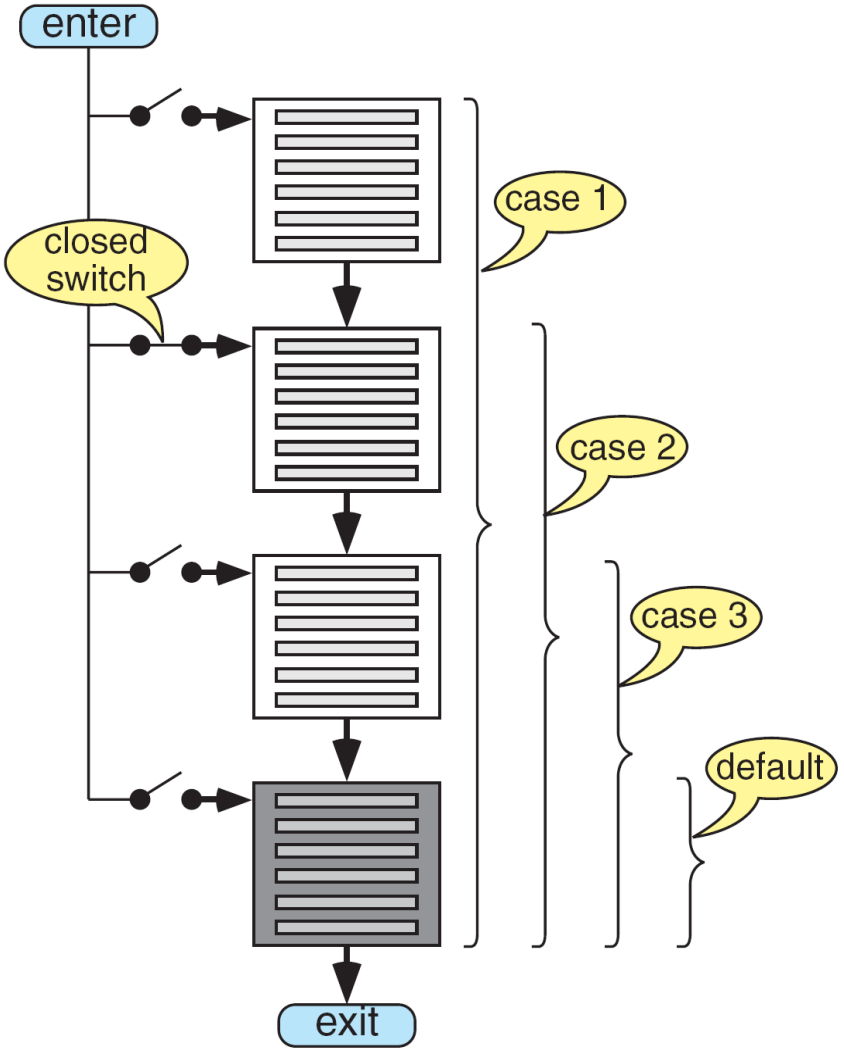


FIGURE 5-21 ***switch*** **Flow**

PROGRAM 5-6 Demonstrate the *switch* Statement

```
1  // Program fragment to demonstrate switch
2     switch (printFlag)
3         {
4          case 1:  printf("This is case 1\n");
5
6          case 2:  printf("This is case 2\n");
7
8          default: printf("This is default\n");
9         } // switch
```

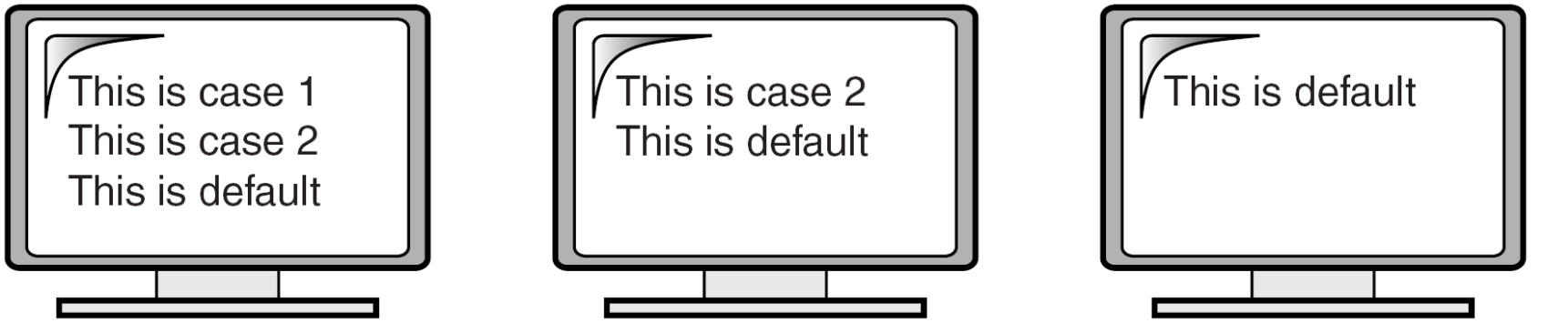
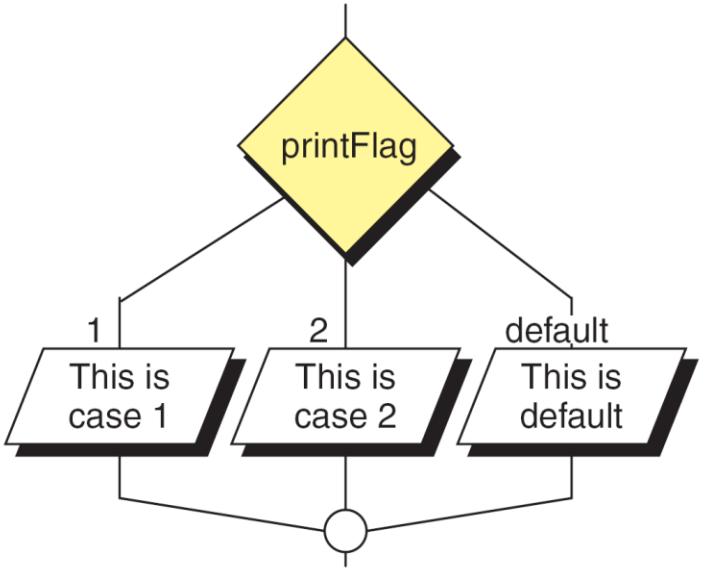


FIGURE 5-22 ***switch*** **Results**



(a) Logic Flow

```
switch (printFlag)
  {
   case 1:
             printf
               ("This is case 1");
             break;
   case 2:
             printf
               ("This is case 2");
             break;
   default:
             printf
               ("This is default");
             break;
  }  // switch
```

(b) Code

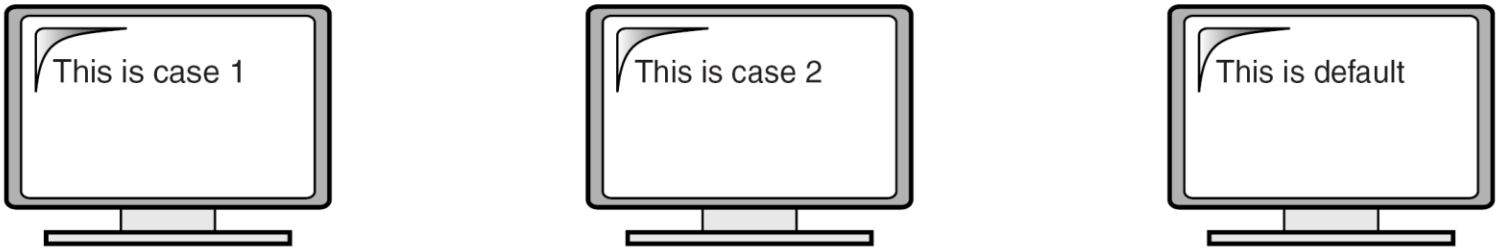


FIGURE 5-23 A *switch* with *break* Statements

PROGRAM 5-7 Multivalued *case* Statements

```
1  /* Program fragment that demonstrates multiple
2     cases for one set of statements
3  */
4  switch (printFlag)
5     {
6      case 1:
7      case 3:  printf("Good Day\n");
8               printf("Odds have it!\n");
9               break;
10     case 2:
11     case 4:  printf("Good Day\n");
12              printf("Evens have it!\n");
13              break;
14     default: printf("Good Day, I'm confused!\n");
15              printf("Bye!\n");
16              break;
17    } // switch
```

1. The control expression that follows the keyword *switch* must be an integral type.
2. Each *case* label is the keyword *case* followed by a constant expression.
3. No two *case* labels can have the same constant expression value.
4. But two *case* labels can be associated with the same set of actions.
5. The *default* label is not required. If the value of the expression does not match with any labeled constant expression, the control transfers outside of the *switch* statement. However, we recommend that all *switch* statements have a *default* label.
6. The *switch* statement can include at most one *default* label. The *default* label may be coded anywhere, but it is traditionally coded last.

Table 5-5 Summary of *switch* Statement Rules

PROGRAM 5-8 Student Grading

```
1  /* This program reads a test score, calculates the letter
2     grade for the score, and prints the grade.
3        Written by:
4        Date:
5  */
6  #include <stdio.h>
7
8  // Function Declarations
9     char scoreToGrade (int score);
10
11 int main (void)
12 {
13 // Local Declarations
14    int  score;
15    char grade;
16
17 // Statements
18    printf("Enter the test score (0-100): ");
19    scanf ("%d", &score);
```

PROGRAM 5-8 Student Grading

```
20
21      grade = scoreToGrade (score);
22      printf("The grade is: %c\n", grade);
23
24      return 0;
25  }   // main
26  /* ================== scoreToGrade ==================
27      This function calculates the letter grade for a score.
28         Pre   the parameter score
29         Post  returns the grade
30  */
31  char scoreToGrade (int score)
32  {
33  // Local Declarations
34      char grade;
35      int  temp;
36
37  // Statements
38      temp = score / 10;
```

PROGRAM 5-8 Student Grading

```
39      switch (temp)
40          {
41           case 10:
42           case 9 : grade = 'A';
43                    break;
44           case 8 : grade = 'B';
45                    break;
46           case 7 : grade = 'C';
47                    break;
48           case 6 : grade = 'D';
49                    break;
50           default: grade = 'F';
51          } // switch
52      return grade;
53  }  // scoreToGrade
```

```
Results:
Enter the test score (0-100): 89
The grade is: B
```

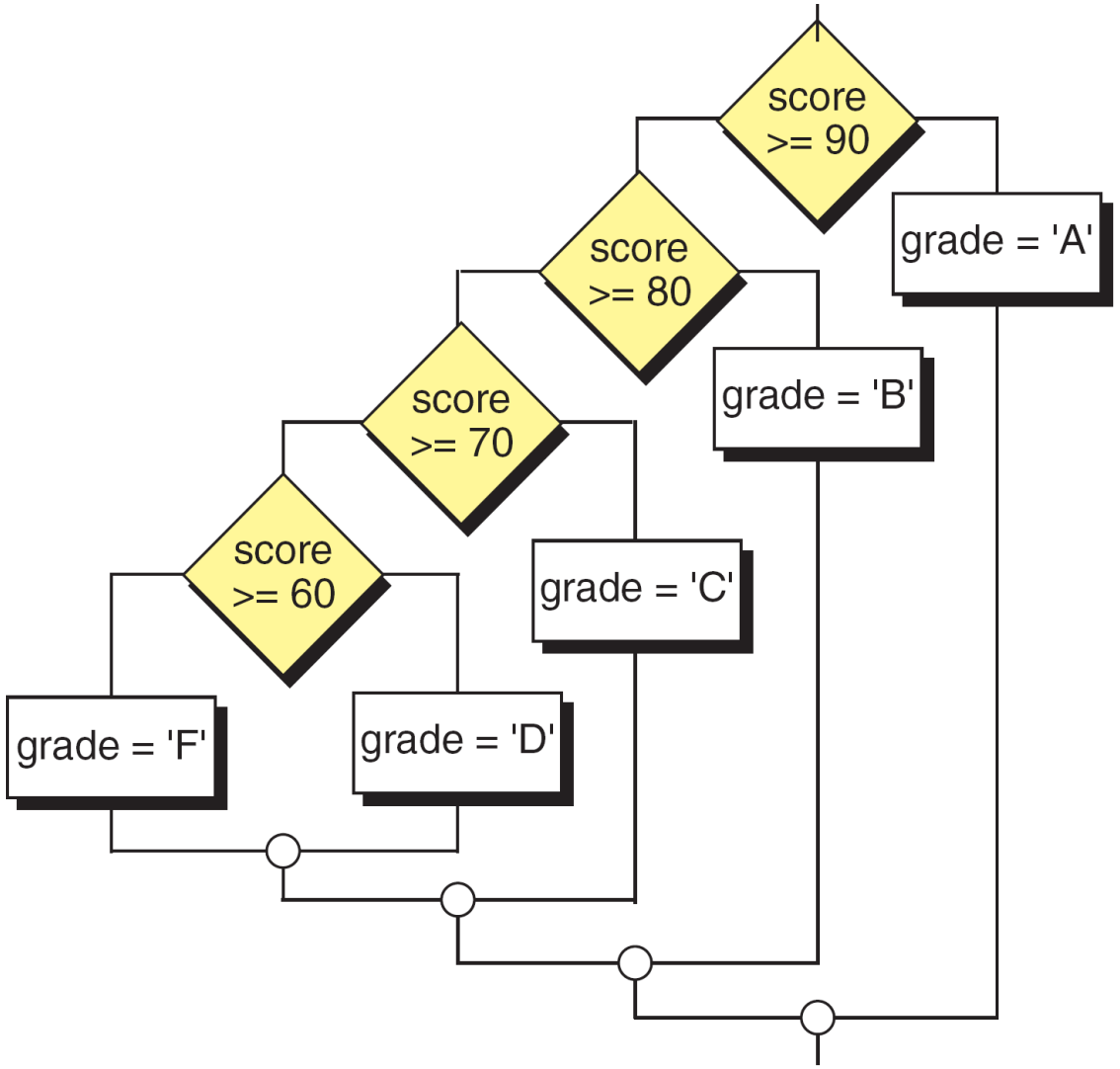


FIGURE 5-24 **The *else-if* Logic Design for Program 5-9**

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.

PROGRAM 5-9 Convert Score to Grade

```
1  /* This program reads a test score, calculates the letter
2     grade based on the absolute scale, and prints it.
3        Written by:
4        Date:
5  */
6  #include <stdio.h>
7
8  // Function Declarations
9  char scoreToGrade (int score);
10
11 int main (void)
12 {
13 // Local Declarations
14    int  score;
15    char grade;
16
17 // Statements
```

PROGRAM 5-9 Convert Score to Grade

```
18      printf("Enter the test score (0-100): ");
19      scanf ("%d", &score);
20
21      grade = scoreToGrade (score);
22      printf("The grade is: %c\n", grade);
23
24      return 0;
25  }   // main
26
27  /* ================== scoreToGrade ==================
28      This function calculates letter grade for a score.
29         Pre   the parameter score
30         Post  returns the grade
31  */
32  char scoreToGrade (int score)
33  {
34  // Local Declarations
35      char grade;
36
```

PROGRAM 5-9 Convert Score to Grade

```
37  // Statements
38     if (score >= 90)
39        grade = 'A';
40     else if (score >= 80)
41        grade = 'B';
42     else if (score >= 70)
43        grade = 'C';
44     else if (score >= 60)
45        grade = 'D';
46     else
47        grade = 'F';
48     return grade;
49  }  // scoreToGrade
```

```
Results:
Enter the test score (0-100): 90
The grade is: A
```

5-4 More Standard Functions

One of the assets of the C language is its rich set of standard functions that make programming much easier. For example, C99 has two parallel but separate header files for manipulating characters: ctype.h and wctype.h.

Topics discussed in this section:

Standard Characters Functions
A Classification Program
Handling Major Errors

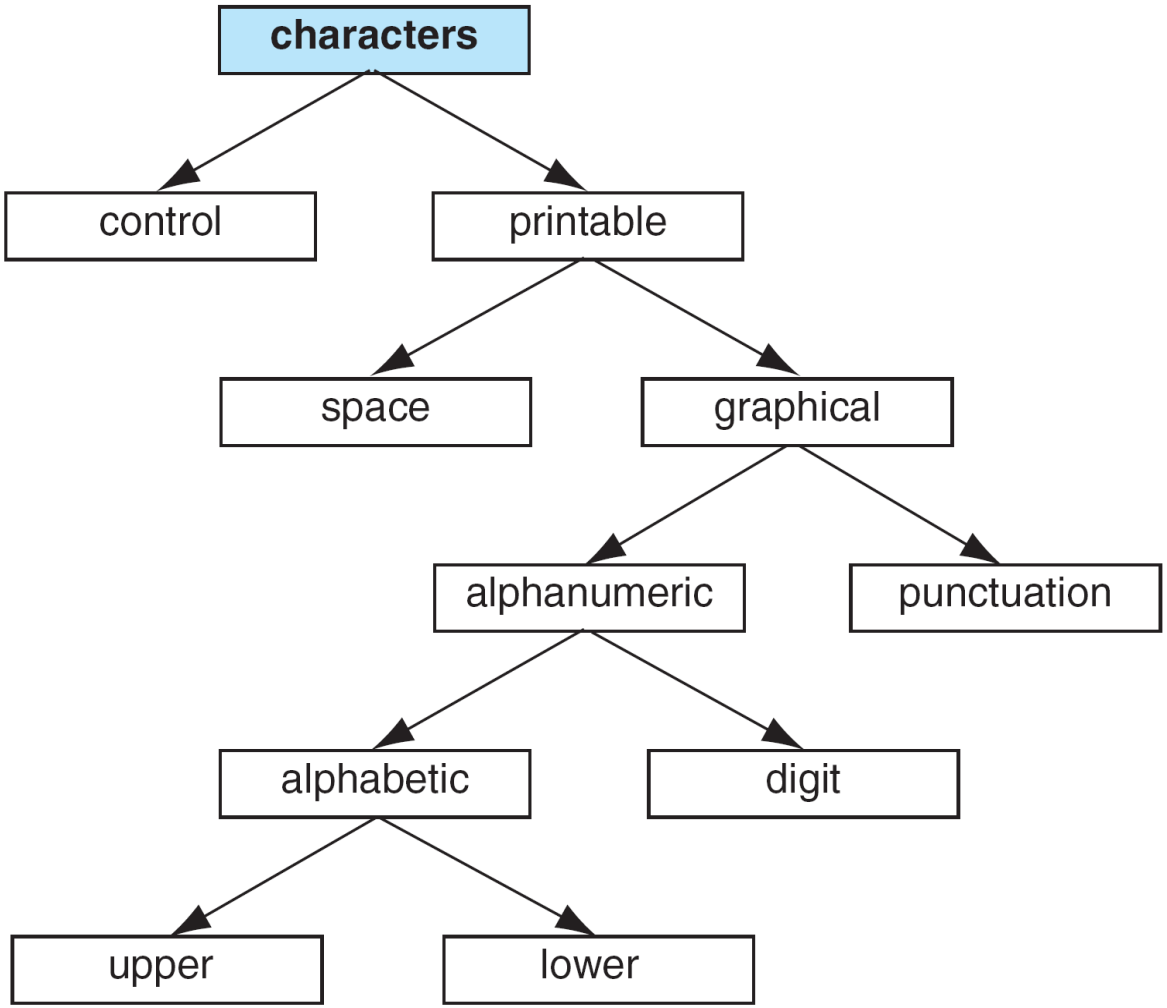


FIGURE 5-25 Classifications of the Character Type

continued

Function	Description
iscntrl	Control characters
isprint	Printable character, that is character with an assigned graphic
isspace	Whitespace character: space character (32), horizontal tab (9), line feed (10), vertical tab (11), form feed (12), and carriage return (13)
isgraph	Character with printable graphic; all printable characters except space
isalnum	Alphanumeric: any alphabetic or numeric character
ispunct	Any graphic character that is not alphanumeric

Table 5-6 Classifying Functions

Function	Description
isalpha	Any alphabetic character, upper- or lowercase
isupper	Only uppercase alphabetic
islower	Only lowercase alphabetic
isdigit	Decimal digits (0...9)
isxdigit	Hexadecimal digits (0...9, a...f, A...F)
isodigit	Octal digits (0...7)

Table 5-6 Classifying Functions *(continued)*

Function	Description
toupper	Converts lower- to uppercase. If not lowercase, returns it unchanged.
tolower	Converts upper- to lowercase. If not uppercase, returns it unchanged.

Table 5-7 Conversion Functions

PROGRAM 5-10 Demonstrate Classification Functions

```
1  /* This program demonstrates the use of the character
2     classification functions found in the c-type library.
3     Given a character, it displays the highest
4     classification for the character.
5        Written by:
6        Date:
7  */
8
9  #include <stdio.h>
10 #include <ctype.h>
11
12 int main (void)
13 {
14 // Local Declarations
15    char charIn;
16
17 // Statements
18    printf("Enter a character to be examined: ");
19    scanf ("%c", &charIn);
20
```

PROGRAM 5-10 Demonstrate Classification Functions

```
21      if (islower(charIn))
22          printf("You entered a lowercase letter.\n");
23      else if (isupper(charIn))
24          printf("You entered an uppercase character.\n");
25      else if (isdigit(charIn))
26          printf("You entered a digit.\n");
27      else if (ispunct(charIn))
28          printf("You entered a punctuation character.\n");
29      else if (isspace(charIn))
30          printf("You entered a whitespace character.\n");
31      else
32          printf("You entered a control character\n");
33      return 0;
34  }   // main
```

```
Results:
Enter a character to be examined: a
You entered a lowercase letter.
```

5-5 Incremental Development Part II

In Chapter 4, we introduced the concept of incremental development with a simple calculator program. We continue the discussion by adding a menu and calculator subfunctions.

Topics discussed in this section:

Calculator Design

Calculator Incremental Design

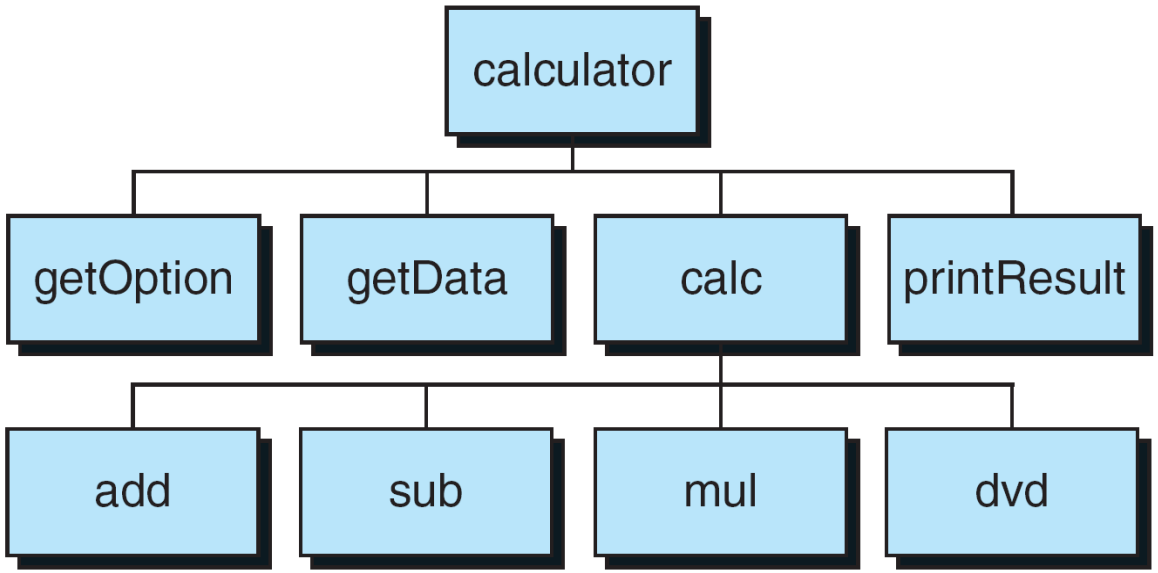


FIGURE 5-26 Design for Menu-driven Calculator

PROGRAM 5-11 Menu-driven Calculator—First Increment

```
 1  /* This program uses a menu to allow the user to add,
 2     multiply, subtract, or divide two integers.
 3        Written by:
 4        Date:
 5  */
 6  #include <stdio.h>
 7  #include <stdlib.h>
 8
 9  // Function Declarations
10  int   getOption   (void);
11
12  int main (void)
13  {
14  // Local Declarations
15     int   option;
16
17  // Statements
18     option = getOption();
19
```

PROGRAM 5-11 Menu-driven Calculator—First Increment

```
20  // Temporary code--to be removed
21  printf("**You selected option %d\n", option);
22
23     return 0;
24  }  // main
25
26  /* ==================== getOption ====================
27     This function shows a menu and reads the user option.
28        Pre    Nothing
29        Post   returns the option
30  */
31  int getOption (void)
32  {
33  // Local Declarations
34     int option;
35
36  // Statements
```

PROGRAM 5-11 Menu-driven Calculator—First Increment

```
37      printf  ("\t**********************************");
38      printf("\n\t*                MENU              *");
39      printf("\n\t*                                  *");
40      printf("\n\t*  1. ADD                          *");
41      printf("\n\t*  2. SUBTRACT                     *");
42      printf("\n\t*  3. MULTIPLY                     *");
43      printf("\n\t*  4. DIVIDE                       *");
44      printf("\n\t*                                  *");
45      printf("\n\t**********************************");
46
47      printf("\n\nPlease type your choice ");
48      printf("and key return: ");
49      scanf ("%d", &option);
50  // Temporary code--to be removed
51  printf("**You selected option %d\n", option);
52      return option;
53  }  // getOption
```

PROGRAM 5-11 Menu-driven Calculator—First Increment

```
Results:
    ************************************

    *                MENU                *
    *                                    *
    *   1. ADD                           *
    *   2. SUBTRACT                      *
    *   3. MULTIPLY                      *
    *   4. DIVIDE                        *
    *                                    *
    ************************************

Please type your choice and key return: 3
**You selected option 3
**You selected option 3
```

PROGRAM 5-12 Menu-driven Calculator—Third Increment

```
 1  /* This program uses a menu to allow the user to add,
 2     multiply, subtract, or divide two integers.
 3        Written by:
 4        Date:
 5  */
 6  #include <stdio.h>
 7  #include <stdlib.h>
 8
 9  // Function Declarations
10  int   getOption (void);
11  void  getData   (int* num1, int* num2);
12  float calc      (int option, int num1, int num2);
13
14  int main (void)
15  {
16  // Local Declarations
17     int   option;
18     int   num1;
```

PROGRAM 5-12 Menu-driven Calculator—Third Increment

```
19      int   num2;
20      float result;
21
22  // Statements
23      option = getOption();
24      getData (&num1, &num2);
25      result = calc (option, num1, num2);
26  printf("**In main result is: %6.2f", result);
27
28      return 0;
29  }  // main
30
31  /* ==================== getOption ====================
32      This function shows a menu and reads the user option.
33         Pre    Nothing
34         Post   returns the option
35  */
36  int getOption (void)
37  {
38  // Local Declarations
```

PROGRAM 5-12 Menu-driven Calculator—Third Increment

```
39      int option;
40
41  // Statements
42      printf  ("\t**********************************");
43      printf("\n\t*                 MENU             *");
44      printf("\n\t*                                  *");
45      printf("\n\t*  1. ADD                          *");
46      printf("\n\t*  2. SUBTRACT                     *");
47      printf("\n\t*  3. MULTIPLY                     *");
48      printf("\n\t*  4. DIVIDE                       *");
49      printf("\n\t*                                  *");
50      printf("\n\t**********************************");
51
52      printf("\n\nPlease type your choice ");
53      printf("and key return: ");
54      scanf ("%d", &option);
55      return option;
56  }   // getOption
57
```

PROGRAM 5-12 Menu-driven Calculator—Third Increment

```
58 /* ====================== getData ====================
59    This function reads two integers from the keyboard.
60       Pre   Parameters a and b are addresses
61       Post  Data read into parameter addresses
62 */
63 void getData (int* a, int* b)
64 {
65    printf("Please enter two integer numbers: ");
66    scanf("%d %d", a, b);
67    return;
68 }  // getData
69
70 /* ==================== calc ====================
71    This function determines the type of operation
72    and calls a function to perform it.
73       Pre   option contains the operation
74             num1 & num2 contains data
75       Post  returns the results
76 */
```

PROGRAM 5-12 Menu-driven Calculator—Third Increment

```
77  float calc (int option, int num1, int num2)
78  {
79  // Local Declarations
80  float result;
81
82  // Statements
83  printf("**In calc input is: %d %d %d\n",
84  option, num1, num2);
85     switch(option)
86          {
87           case 1:  result = 1.0;              // Add
88                    break;
89           case 2:  result = 2.0;              // Subtract
90                    break;
91           case 3:  result = 3.0;              // Multiply
92                    break;
93           case 4:  if (num2 == 0.0)           // Divide
```

PROGRAM 5-12 Menu-driven Calculator—Third Increment

```
 94                          {
 95                           printf("\n\a\aError: ");
 96                           printf("division by zero ***\n");
 97                           exit (100);
 98                          } // if
 99                      else
100                           result = 4.0;
101                      break;
102            /* Better structured programming would validate
103               option in getOption. However, we have not
104               yet learned the technique to code it there.
105            */
106            default: printf("\aOption not available\n");
107                      exit (101);
108         } // switch
109   printf("**In calc result is: %6.2f\n", result);
110      return result;
111      return result;
112   }  // calc
```

PROGRAM 5-12 Menu-driven Calculator—Third Increment

```
Results:
     ************************************
     *                MENU                *
     *                                    *
     *   1. ADD                           *
     *   2. SUBTRACT                      *
     *   3. MULTIPLY                      *
     *   4. DIVIDE                        *
     *                                    *
     ************************************

Please type your choice and key return: 1
Please enter two integer numbers: 13 8
**In calc input is: 1 13 8
**In calc result is:   1.00
**In main result is:   1.00
```

PROGRAM 5-13 Menu-driven Calculator—Fifth Increment

```
 1  /* This program uses a menu to allow the user to add,
 2     multiply, subtract, or divide two integers.
 3        Written by:
 4        Date:
 5  */
 6  #include <stdio.h>
 7  #include <stdlib.h>
 8
 9  // Function Declarations
10  int   getOption (void);
11  void  getData   (int* num1,  int* num2);
12  float calc      (int option, int num1, int num2);
13  float add       (int num1,   int num2);
14  float sub       (int num1,   int num2);
15
16  int main (void)
17  {
18  // Local Declarations
```

PROGRAM 5-13 Menu-driven Calculator—Fifth Increment

```
19      int   option;
20      int   num1;
21      int   num2;
22      float result;
23
24  // Statements
25      option = getOption();
26      getData (&num1, &num2);
27      result = calc (option, num1, num2);
28  printf("**In main result is: %6.2f\n", result);
29
30      return 0;
31  }  // main
32
33  /* ==================== getOption ====================
34      This function shows a menu and reads the user option.
35         Pre    Nothing
36         Post   returns the option
37  */
```

PROGRAM 5-13 Menu-driven Calculator—Fifth Increment

```
38  int getOption (void)
39  {
40  // Local Declarations
41     int option;
42
43  // Statements
44     printf  ("\t*************************************");
45     printf("\n\t*                  MENU                *");
46     printf("\n\t*                                      *");
47     printf("\n\t*  1. ADD                              *");
48     printf("\n\t*  2. SUBTRACT                         *");
49     printf("\n\t*  3. MULTIPLY                         *");
50     printf("\n\t*  4. DIVIDE                           *");
51     printf("\n\t*                                      *");
52     printf("\n\t*************************************");
53
54     printf("\n\nPlease type your choice ");
55     printf("and key return: ");
56     scanf ("%d", &option);
```

PROGRAM 5-13 Menu-driven Calculator—Fifth Increment

```
57      return option;
58  }   // getOption
59
60  /* ======================== getData ====================
61      This function reads two integers from the keyboard.
62         Pre   Parameters a and b are addresses
63         Post  Data read into parameter addresses
64  */
65  void getData (int* a, int* b)
66  {
67      printf("Please enter two integer numbers: ");
68      scanf("%d %d", a, b);
69      return;
70  }   // getData
71
72  /* ====================== calc ====================
73      This function determines the type of operation
74      and calls a function to perform it.
75         Pre   option contains the operation
76               num1 & num2 contains data
```

PROGRAM 5-13 Menu-driven Calculator—Fifth Increment

```
77          Post  returns the results
78  */
79  float calc (int option, int num1, int num2)
80  {
81  // Local Declarations
82  float result;
83
84  // Statements
85     switch(option)
86          {
87           case 1:  result = add (num1, num2);
88                    break;
89           case 2:  result = sub (num1, num2);
90                    break;
91           case 3:  result = 3.0;                  // Multiply
92                    break;
93           case 4:  if (num2 == 0.0)               // Divide
94                       {
95                        printf("\n\a\aError: ");
```

PROGRAM 5-13 Menu-driven Calculator—Fifth Increment

```
 96                         printf("division by zero ***\n");
 97                         exit (100);
 98                       } // if
 99                   else
100                         result = 4.0;
101                   break;
102          /* Better structured programming would validate
103             option in getOption. However, we have not
104             yet learned the technique to code it there.
105          */
106          default: printf("\aOption not available\n");
107                   exit (101);
108       } // switch
109 printf("**In calc result is: %6.2f\n", result);
110    return result;
111 }  // calc
112
113 /* ======================= add =======================
114    This function adds two numbers and returns the sum.
115       Pre   a and b contain values to be added
116       Post  Returns a + b
117 */
```

PROGRAM 5-13 Menu-driven Calculator—Fifth Increment

```
118 float add (int a, int b)
119 {
120 // Local Definitions
121    float sum;
122
123 // Statements
124    sum = a + b;
125    return sum;
126 }  // add
127
128 /* ======================== sub ========================
129    This function subtracts two numbers
130       Pre   a and b contain values to be subtracted
131       Post  Returns a + b
132 */
133 float sub (int a, int b)
134 {
135 // Local Definitions
```

PROGRAM 5-13 Menu-driven Calculator—Fifth Increment

```
136     float dif;
137
138  // Statements
139     dif = a - b;
140  printf("**In sub result is:  %6.2f\n", dif);
141     return dif;
142  }  // sub
```

```
Results:
     ************************************
     *                MENU                *
     *                                    *
     *   1. ADD                           *
     *   2. SUBTRACT                      *
     *   3. MULTIPLY                      *
     *   4. DIVIDE                        *
     *                                    *
     ************************************

Please type your choice and key return: 2
```

PROGRAM 5-13 Menu-driven Calculator—Fifth Increment

```
Please enter two integer numbers: 13 8
**In sub result is:    5.00
**In calc result is:   5.00
**In main result is:   5.00
```

5-6 Software Engineering

In this section, we discuss some software engineering issues related to decisions.

Topics discussed in this section:

Dependent Statements
Negative Logic
Rules for Selection Statements
Selection in Structure Charts

PROGRAM 5-14 Examples of Poor and Good Nesting Styles

Poor Style

```
1  int someFun (int a, int b)
2  {
3  int x;
4
5     if (a < b)
6     x = a;
7     else
8        x = b;
9        x *= .5f;
10    return x;
11 }  // someFun
```

Good Style

```
int someFun (int a, int b)
{
   int x;

   if (a < b)
      x = a;
   else
      x = b;
   x *= .5f;
   return x;
}  // someFun
```

1. Indent statements that are dependent on previous statements. The indentations are at least three spaces from the left end of the controlling statement.
2. Align *else* statements with their corresponding *if* statements. (See Figure 5-13.)
3. Place the opening brace identifying a body of code on a separate line. Indent the statements in the body of the code one space to the right of the opening brace.
4. Align the closing brace identifying a body of code with the opening brace, and place the closing brace on a separate line. Use a comment to identify the block being terminated.
5. Align all code on the same level, which is dependent on the same control statement.
6. Further indent nested statements according to the above rules.
7. Surround operators with whitespace.
8. Code only one definition or statement on a single line.
9. Make your comments meaningful at the block level. Comments should not simply parrot the code.

Table 5-8 Indentation Rules

Note

Avoid compound negative statements!

Original Statement	Complemented Statement
`if (x <= 0)`	`if (x > 0)`
`if (x != 5)`	`if (x == 5)`
`if (!(x <= 0 || !flag))`	`if (x > 0 && flag)`

Table 5-9 Complementing Expressions

1. Code positive statements whenever possible.
2. Code the normal/expected condition first.
3. Code the most probable conditions first.

Table 5-10 Selection Rules

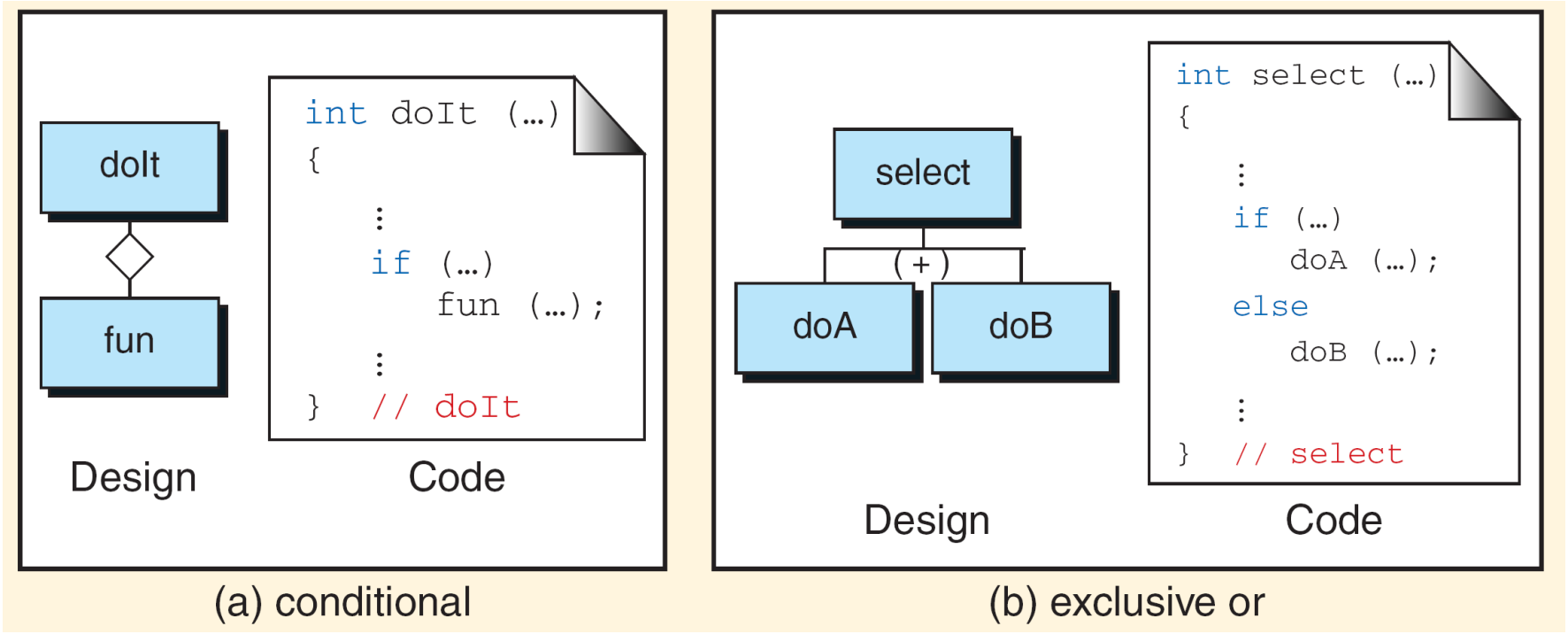
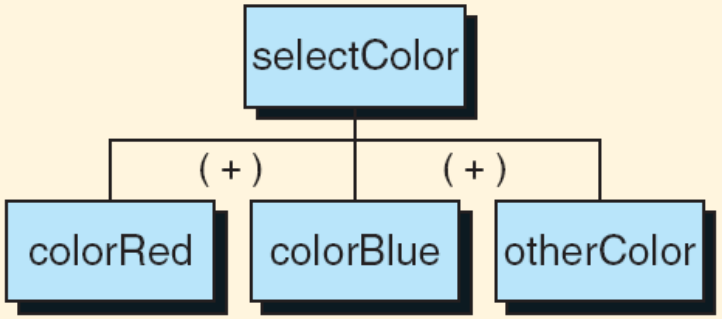


FIGURE 5-27 Structure Chart Symbols for Selection



(a) Design

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

(b) Code

FIGURE 5-28 Multiway Selection in a Structure Chart