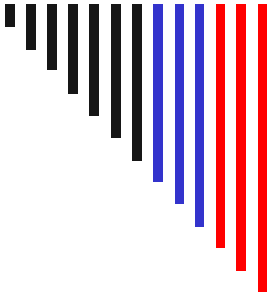


Chapter 3
Structure of a C Program

Objectives

- To be able to list and describe the six expression categories
- To understand the rules of precedence and associativity in evaluating expressions
- To understand the result of side effects in expression evaluation
- To be able to predict the results when an expression is evaluated
- To understand implicit and explicit type conversion
- To understand and use the first four statement types: null, expression, return, and compound

3-1 Expressions

An expression is a sequence of operands and operators that reduces to a single value. Expressions can be simple or complex. An operator is a syntactical token that requires an action be taken. An operand is an object on which an operation is performed; it receives an operator's action.

Topics discussed in this section:

Primary Expressions
Postfix Expressions
Prefix Expressions
Unary Expressions
Binary Expressions

Note

An expression always reduces to a single value.

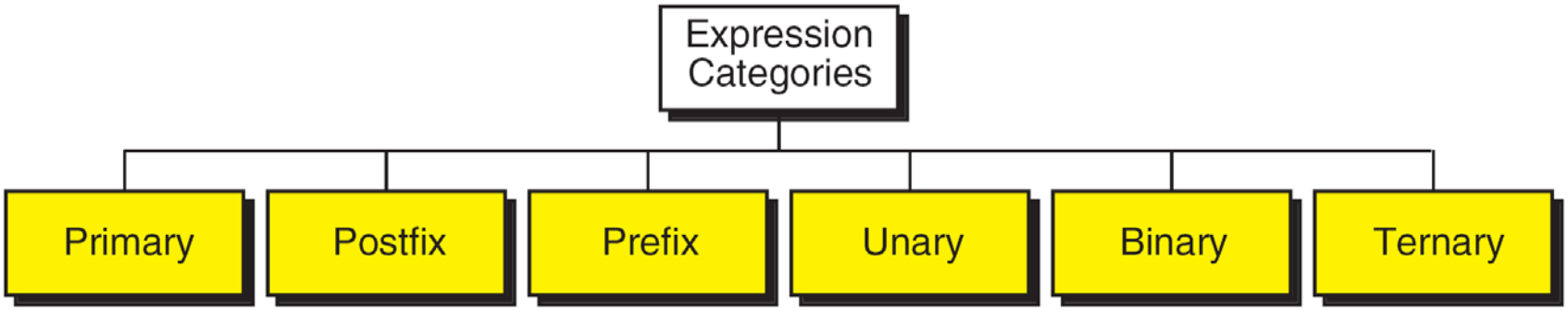


FIGURE 3-1 Expression Categories

Primary Expressions

- Names
  - a, b12, price, INT_MAX, SIZE
- Literal constants
  - 5, 123.98, ‘A’, “Welcome”
- Parenthetical expressions
  - (2 * 3 + 4), (a = 23 + b * 6)

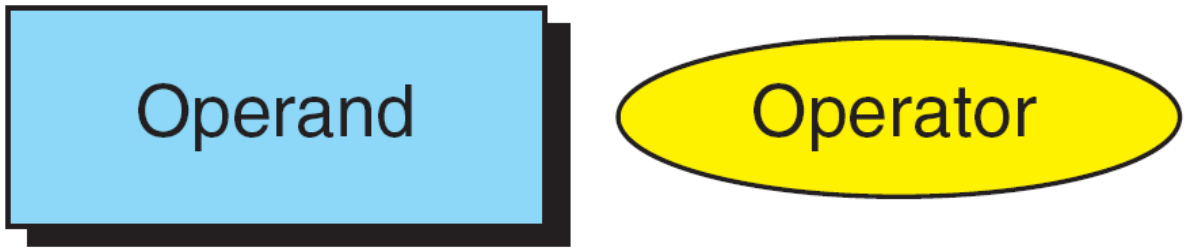


FIGURE 3-2 Postfix Expressions

Note

(a++) has the same effect as (a = a + 1)

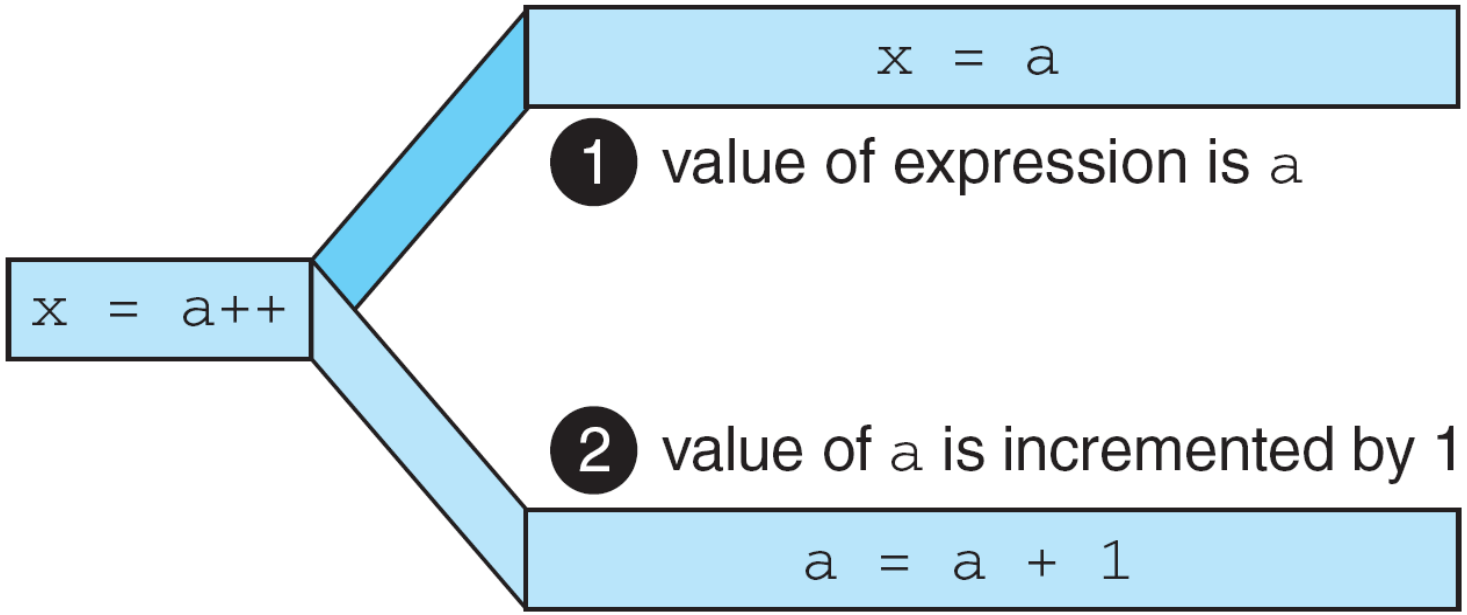


FIGURE 3-3 Result of Postfix a++

Note

The operand in a postfix expression must be a variable.

PROGRAM 3-1 Demonstrate Postfix Increment

```
 1  /* Example of postfix increment.
 2        Written by:
 3        Date:
 4  */
 5  #include <stdio.h>
 6  int main (void)
 7  {
 8  // Local Declarations
 9     int a;
10
11  // Statements
12     a = 4;
13     printf("value of a     : %2d\n", a);
14     printf("value of a++   : %2d\n",   a++);
15     printf("new value of a: %2d\n\n", a);
16     return 0;
17  }  // main
```

PROGRAM 3-1 Demonstrate Postfix Increment (continued)

```
Results:
value of a     :  4
value of a++   :  4
new value of a:  5
```

Function Call

- A postfix expression
- Function call operator ()
- printf(“hello world\ n”)

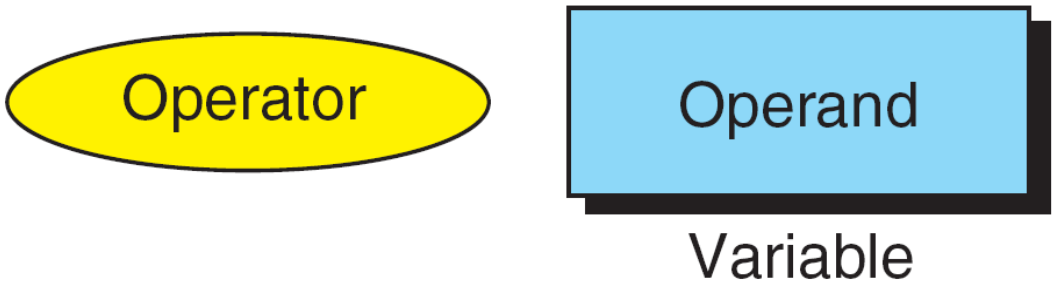


FIGURE 3-4 Prefix Expression

Note

(++a) has the same effect as (a = a + 1)

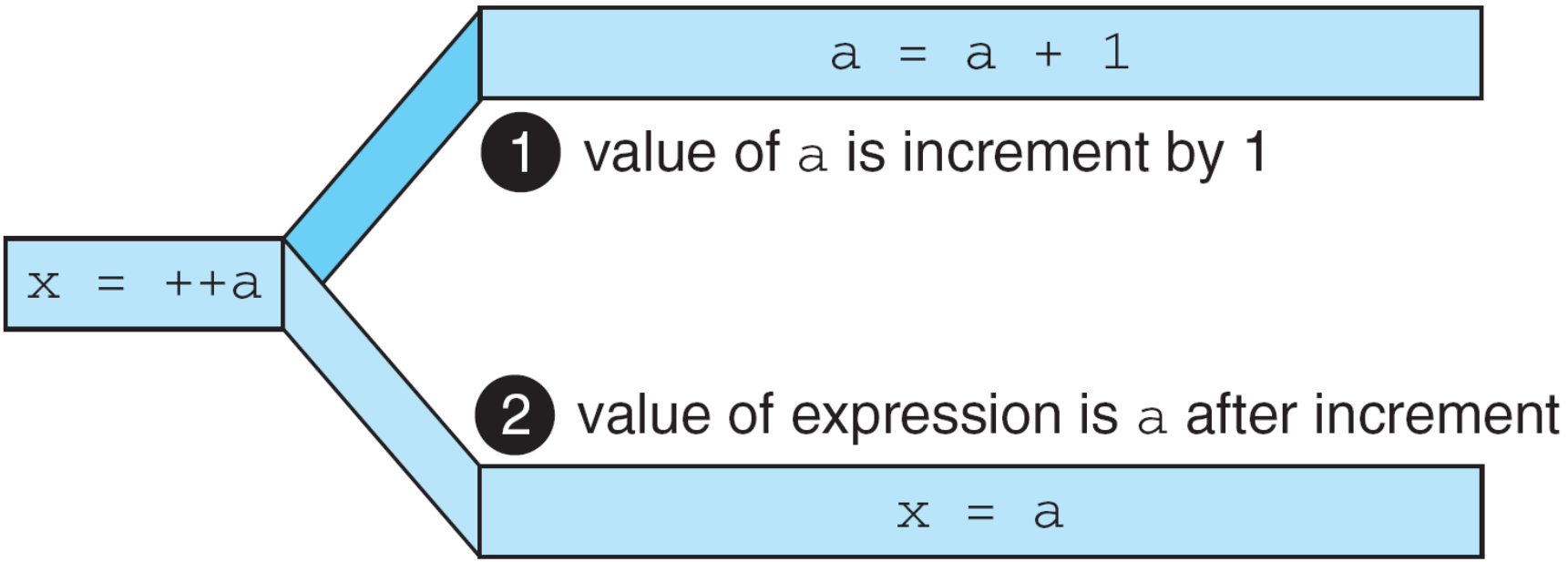


FIGURE 3-5 Result of Prefix ++a

Note

The operand of a prefix expression must be a variable.

PROGRAM 3-2 Demonstrate Prefix Increment

```
1  /* Example of prefix increment.
2        Written by:
3        Date:
4  */
5  #include <stdio.h>
6  int main (void)
7  {
8  // Local Declarations
9     int a;
10
11 // Statements
12    a = 4;
13    printf("value of a     : %2d\n", a);
14    printf("value of ++a   : %2d\n", ++a);
15    printf("new value of a: %2d\n", a);
16    return 0;
17 }  // main
```

PROGRAM 3-2 Demonstrate Prefix Increment (continued)

```
Results:
value of a     :  4
value of ++a   :  5
new value of a:  5
```

Note

If ++ is after the operand, as in a++, the increment takes place after the expression is evaluated.

If ++ is before the operand, as in ++a, the increment takes place before the expression is evaluated.

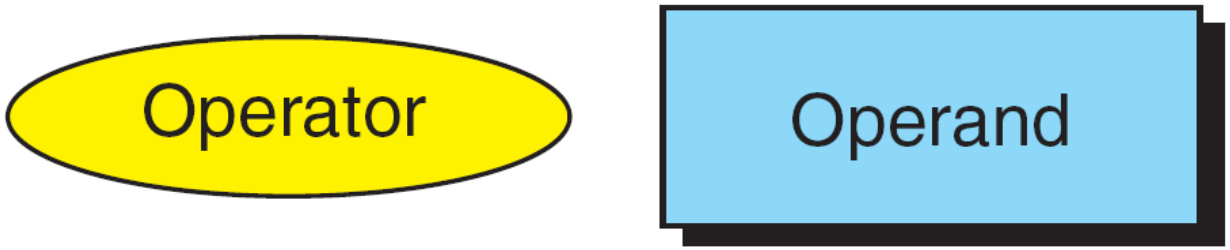


FIGURE 3-6 Unary Expressions

Expression	Contents of a Before *and* After Expression	Expression Value
+a	3	+3
-a	3	-3
+a	-5	-5
-a	-5	+5

Table 3-1 Examples of Unary Plus And Minus Expressions

FIGURE 3-7 Binary Expressions

Binary Expressions

- 10 * 3, true * 4, ‘A’ * 2, 22.3 * 2
- 10 / 3, true / 4, ‘A’ / 2, 22.3 / 2
- 10 % 3, true % 4, ‘A’ % 2, 22.3 % 2
- 3 / 5, 3 % 5
- 3 + 7, 3 – 7
- a = 5, b = x + 1, i = i + 1

Note

Both operands of the modulo operator (%) must be integral types.

PROGRAM 3-3 Binary Expressions

```
 1  /* This program demonstrates binary expressions.
 2        Written by:
 3        Date:
 4  */
 5  #include <stdio.h>
 6  int main (void)
 7  {
 8  // Local Declarations
 9     int   a = 17;
10     int   b = 5;
11     float x = 17.67;
12     float y = 5.1;
13
14  // Statements
15     printf("Integral calculations\n");
16     printf("%d + %d  = %d\n", a, b, a + b);
```

PROGRAM 3-3 Binary Expressions (continued)

```
17      printf("%d - %d  = %d\n", a, b, a - b);
18      printf("%d * %d  = %d\n", a, b, a * b);
19      printf("%d / %d  = %d\n", a, b, a / b);
20      printf("%d %% %d  = %d\n", a, b, a % b);
21      printf("\n");
25      printf("%f - %f  = %f\n", x, y, x - y);
26      printf("%f * %f  = %f\n", x, y, x * y);
27      printf("%f / %f  = %f\n", x, y, x / y);
28      return 0;
29  }   // main
```

PROGRAM 3-3 Binary Expressions (continued)

```
Results:
Integral calculations
17 + 5 = 22
17 - 5 = 12
17 * 5 = 85
17 / 5 = 3
17 % 5 = 2

Floating-point calculations
17.670000 + 5.100000  = 22.770000
17.670000 - 5.100000  = 12.570000
17.670000 * 5.100000  = 90.116997
17.670000 / 5.100000  = 3.464706
```

Note

The left operand in an assignment expression must be a single variable.

Compound Expression	Equivalent Simple Expression
`x *= expression`	`x = x * expression`
`x /= expression`	`x = x / expression`
`x %= expression`	`x = x % expression`
`x += expression`	`x = x + expression`
`x -= expression`	`x = x - expression`

Table 3-2 Expansion of Compound Expressions

PROGRAM 3-4 Demonstration of Compound Assignments

```
 1  /* Demonstrate examples of compound assignments.
 2        Written by:
 3        Date:
 4  */
 5  #include <stdio.h>
 6
 7  int main (void)
 8  {
 9  // Local Declarations
10     int x;
11     int y;
12
13  // Statements
14     x = 10;
15     y = 5;
16
```

PROGRAM 3-4 Demonstration of Compound Assignments

```
17      printf("x: %2d  |  y: %2d ",     x, y);
18      printf("  |  x *= y + 2: %2d ", x *= y + 2);
19      printf("  |  x is now: %2d\n",  x);
20
21      x = 10;
22      printf("x: %2d  |  y: %2d ",     x, y);
23      printf("  |  x /= y + 1: %2d ", x /= y + 1);
24      printf("  |  x is now: %2d\n",  x);
25
26      x = 10;
27      printf("x: %2d  |  y: %2d ",      x, y);
28      printf("  |  x %%= y - 3: %2d ", x %= y - 3);
29      printf("  |  x is now: %2d\n", x);
30
31      return 0;
32  }   // main
```

PROGRAM 3-4 Demonstration of Compound Assignments

```
Results:
x: 10  |  y:  5  |  x *= y + 2: 70  |  x is now: 70
x: 10  |  y:  5  |  x /= y + 1:  1  |  x is now:  1
x: 10  |  y:  5  |  x %= y - 3:  0  |  x is now:  0
```

3-2 Precedence and Associativity

Precedence is used to determine the order in which different operators in a complex expression are evaluated. Associativity is used to determine the order in which operators with the same precedence are evaluated in a complex expression.

Topics discussed in this section:

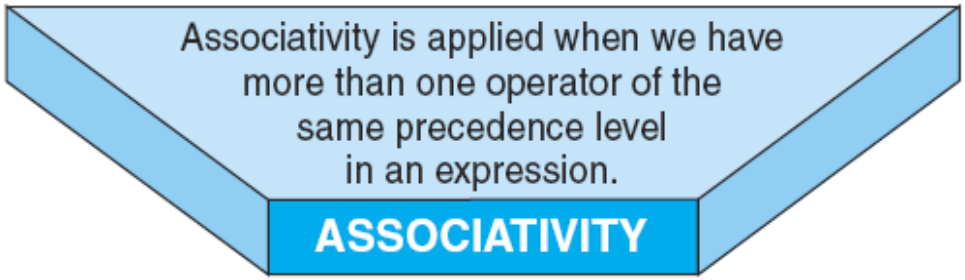
Precedence
Associativity

PROGRAM 3-5 Precedence

```
 1  /* Examine the effect of precedence on an expression.
 2        Written by:
 3        Date:
 4  */
 5  #include <stdio.h>
 6
 7  int main (void)
 8  {
 9  // Local Declarations
10     int a = 10;
11     int b = 20;
12     int c = 30;
13
14  // Statements
15     printf ("a *  b + c  is: %d\n", a *  b + c);
16     printf ("a * (b + c) is: %d\n", a * (b + c));
17     return 0;
18  }  // main
```

PROGRAM 3-5 Precedence

```
Results:
a *  b + c  is: 230
a * (b + c) is: 500
```


Associativity is applied when we have more than one operator of the same precedence level in an expression.
ASSOCIATIVITY

- a + b + c + d
- a * b * c * d

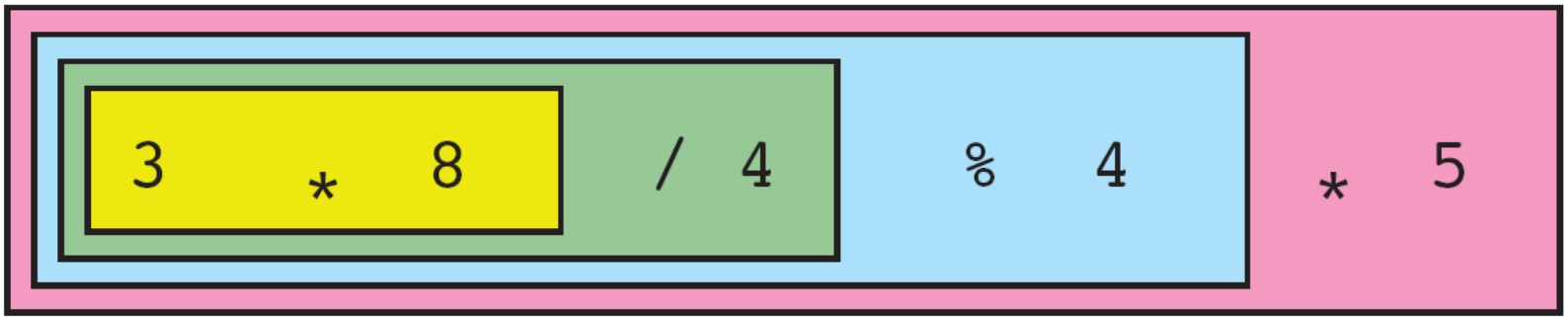


FIGURE 3-8 Left-to-Right Associativity

- a = b = c = d = 10

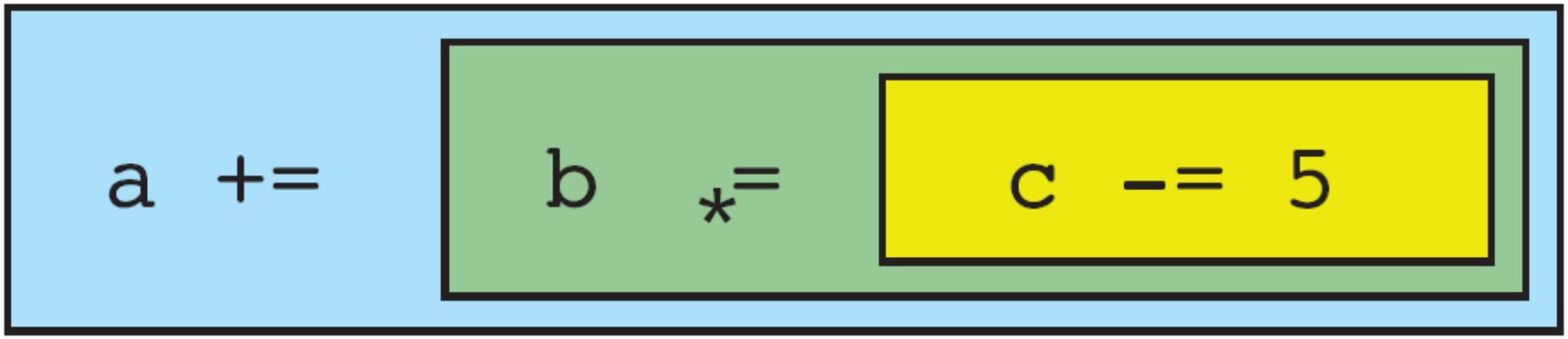


FIGURE 3-9 Right-to-Left Associativity

3-3 Side Effects

A side effect is an action that results from the evaluation of an expression. For example, in an assignment, C first evaluates the expression on the right of the assignment operator and then places the value in the left variable. Changing the value of the left variable is a side effect.

Side Effects

- x = 4;
- x = x + 4;
- a++
- printf(“x = %d\ n”, x);
- printf(“x = %d\ n”, x = x + 4);
- scanf(“%d”, &x);

Side Effects

- A side effect also means an unexpected change due to the use of a global variable
- You are advised not use a global variable!
- A clear example of this kind of side effect could be made when we have multiple functions

3-4 Evaluating Expressions

Now that we have introduced the concepts of precedence, associativity, and side effects, let's work through some examples.

Topics discussed in this section:

Expressions without Side Effects
Expressions with Side Effects

Expressions and Side Effects

- a * 4 + b / 2 – c * b
- d = a * 4 + b / 2 – c * b
- f = a * 4 + b++ / 2 – c * --b

PROGRAM 3-6 Evaluating Expressions

```
 1  /* Evaluate two complex expressions.
 2        Written by:
 3        Date:
 4  */
 5  #include <stdio.h>
 6  int main (void)
 7  {
 8  // Local Declarations
 9     int a = 3;
10     int b = 4;
11     int c = 5;
12     int x;
13     int y;
14
15  // Statements
16     printf("Initial values of the variables: \n");
17     printf("a = %d\tb = %d\tc = %d\n\n", a, b, c);
18
```

PROGRAM 3-6 Evaluating Expressions

```
19      x = a * 4 + b / 2 - c * b;
20      printf
21        ("Value of  a *  4 + b  / 2 - c  * b:  %d\n", x);
22
23      y = --a * (3 + b) / 2 - c++ * b;
24      printf
25        ("Value of --a * (3 + b) / 2 - c++ * b: %d\n", y);
26      printf("\nValues of the variables are now: \n");
27      printf("a = %d\tb = %d\tc = %d\n\n", a, b, c);
28
29      return 0;
30  }   // main
```

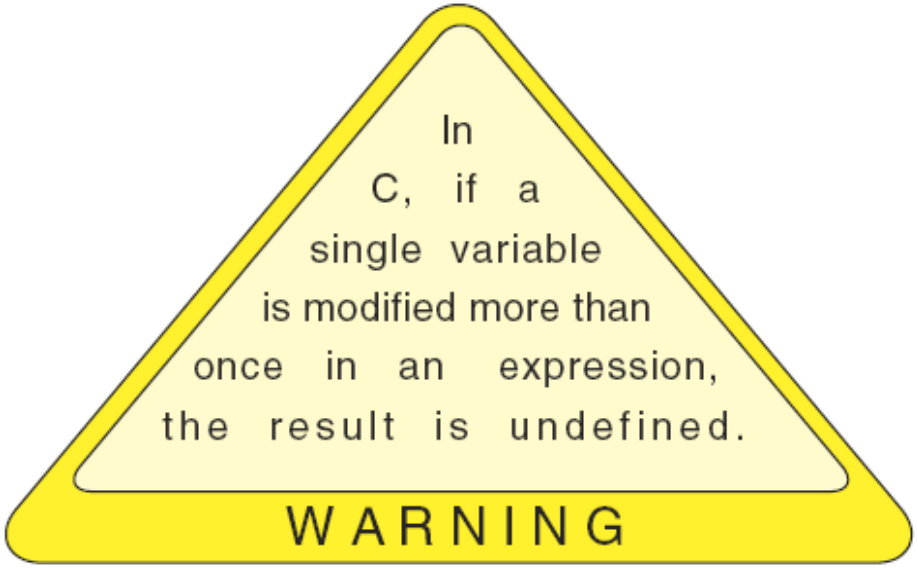
PROGRAM 3-6 Evaluating Expressions

```
Results:
Initial values of the variables:
a = 3   b = 4   c = 5

Value of   a *  4 + b  / 2 - c   * b:  -6
Value of --a * (3 + b) / 2 - c++ * b: -13

Values of the variables are now:
a = 2   b = 4   c = 6
```

Warning


In
C, if a
single variable
is modified more than
once in an expression,
the result is undefined.
WARNING

3-5 Type Conversion

Up to this point, we have assumed that all of our expressions involved data of the same type. But, what happens when we write an expression that involves two different data types, such as multiplying an integer and a floating-point number? To perform these evaluations, one of the types must be converted.

Topics discussed in this section:

Implicit Type Conversion
Explicit Type Conversion (Cast)

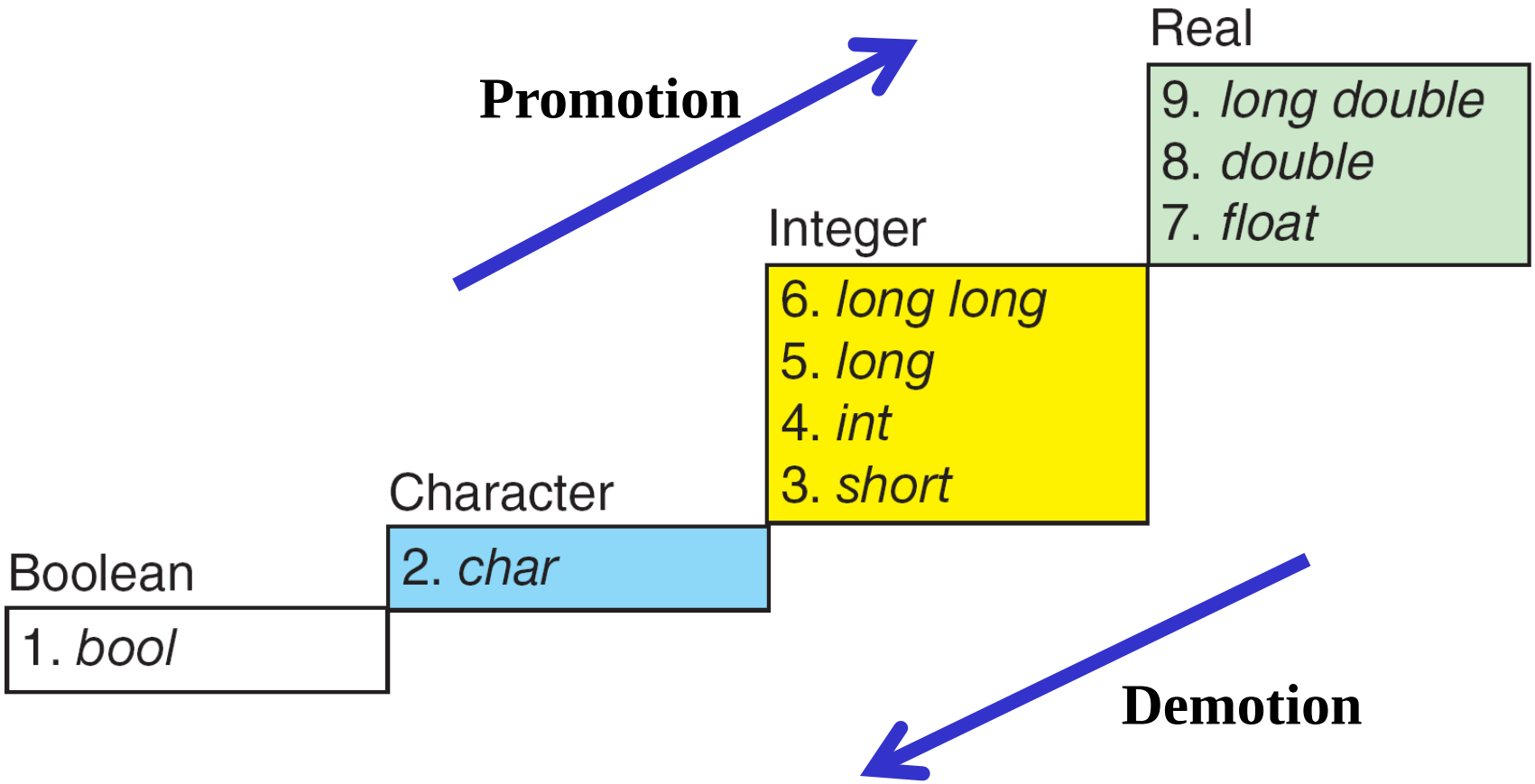


FIGURE 3-10 Conversion Rank

PROGRAM 3-7 Implicit Type Conversion

```
 1  /* Demonstrate automatic promotion of numeric types.
 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  b  = true;
12     char  c  = 'A';
13     float d  = 245.3;
14     int   i  = 3650;
15     short s  = 78;
16
```

PROGRAM 3-7 Implicit Type Conversion

```
17  // Statements
18     printf("bool + char is char:   %c\n", b + c);
19     printf("int * short is int:    %d\n", i * s);
20     printf("float * char is float: %f\n", d * c);
21
22     c = c + b;                    // bool promoted to char
23     d = d + c;                    // char promoted to float
24     b = false;
25     b = -d;                       // float demoted to bool
26
27     printf("\nAfter execution...\n");
28     printf("char + true:    %c\n", c);
29     printf("float + char:   %f\n", d);
30     printf("bool = -float:  %f\n", b);
31
32     return 0;
33  }  // main
```

PROGRAM 3-7 Implicit Type Conversion

```
Results:
bool + char is char:    B
int * short is int:     284700
float * char is float: 15944.500000

After execution...
char + true:     B
float + char:    311.299988
bool = -float:   1
```

Explicit Type Conversion: Cast

- Cast Operator: (type-to-cast) operand

- Example

```
int a=3;
float b=4.0;

b = (float)a;
```

PROGRAM 3-8 Explicit Casts

```
 1  /* Demonstrate casting of numeric types.
 2        Written by:
 3        Date:
 4  */
 5  #include <stdio.h>
 6
 7  int main (void)
 8  {
 9  // Local Declarations
10     char   aChar    = '\0';
11     int    intNum1  = 100;
12     int    intNum2  =  45;
13     double fltNum1  = 100.0;
14     double fltNum2  =  45.0;
15     double fltNum3;
16
17  // Statements
18     printf("aChar numeric   :  %3d\n",   aChar);
```

PROGRAM 3-8 Explicit Casts

```
19      printf("intNum1 contains:  %3d\n",    intNum1);
20      printf("intNum2 contains:  %3d\n",    intNum2);
21      printf("fltNum1 contains:  %6.2f\n", fltNum1);
22      printf("fltNum2 contains:  %6.2f\n", fltNum2);
23
24      fltNum3 = (double)(intNum1 / intNum2);
25      printf
26         ("\n(double)(intNum1 / intNum2): %6.2f\n",
27          fltNum3);
28
29      fltNum3 = (double)intNum1 / intNum2;
30      printf("(double) intNum1 / intNum2 : %6.2f\n",
31             fltNum3);
32
33      aChar = (char)(fltNum1 / fltNum2);
34      printf("  (char)(fltNum1 / fltNum2): %3d\n",   aChar);
35
```

PROGRAM 3-8 Explicit Casts

```
36      return 0;
37  }   // main
```

```
Results:
aChar numeric   :    0
intNum1 contains:  100
intNum2 contains:   45
fltNum1 contains:  100.00
fltNum2 contains:   45.00

(double)(intNum1 / intNum2):   2.00
(double) intNum1 / intNum2 :   2.22
  (char)(fltNum1 / fltNum2):   2
```

3-6 Statements

A statement causes an action to be performed by the program. It translates directly into one or more executable computer instructions.

You may have noticed that we have used a semicolon at the end of the statements in our programs. Most statements need a semicolon at the end; some do not.

Topics discussed in this section:

Statement Type
The Role of the Semicolon
Statements and Defined Constants

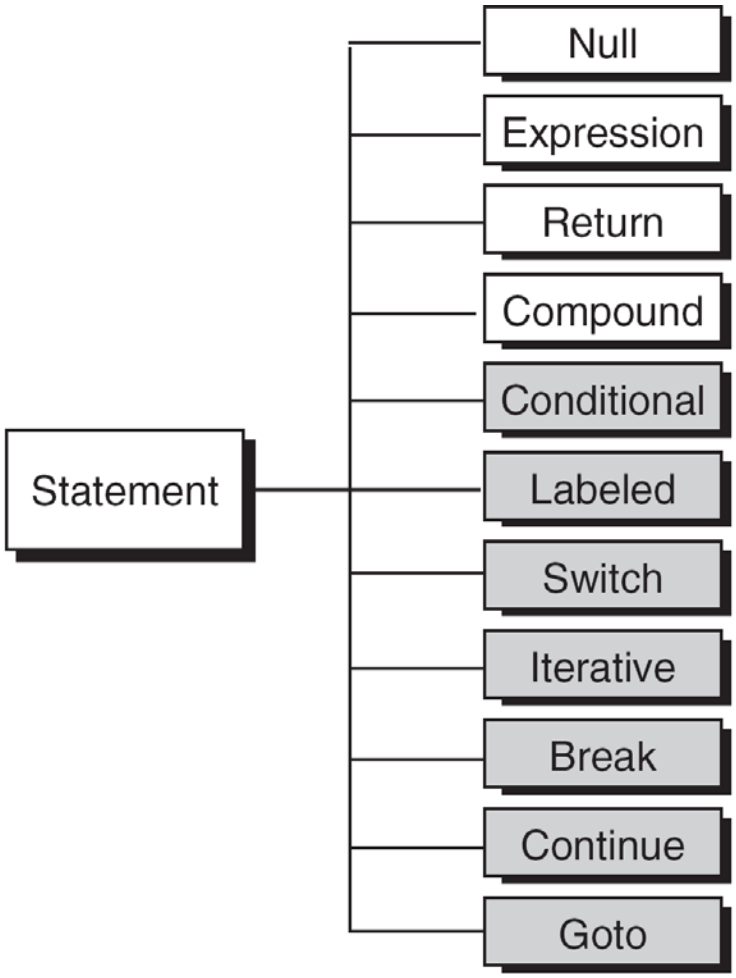


FIGURE 3-11 **Types of Statements**

Null Statement

- Just a semicolon
- Example

```
;                    // null statement
```

- They do nothing, but are still valid syntactical objects.

Expression Statement

- A statement consisting of an expression
- Example
  expression;

  a = 2;

  b = c = 3;

Return Statement

- A return statement terminates a function.

- Example

  return;

  return expression;

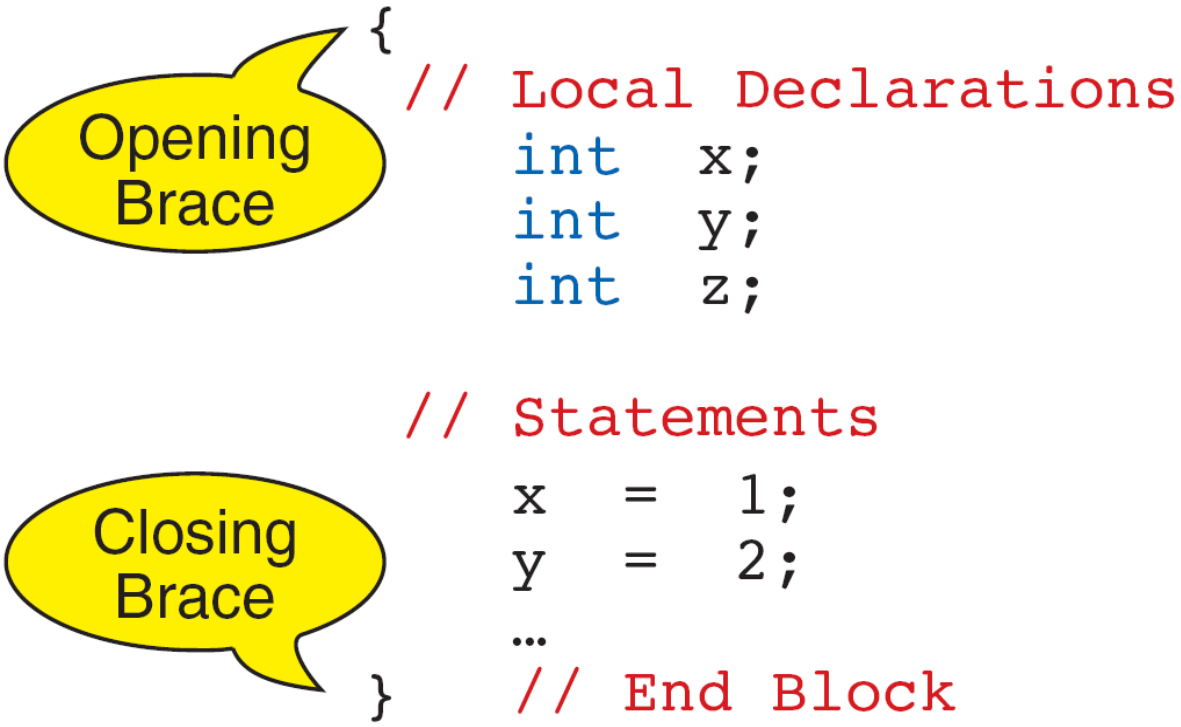


FIGURE 3-12 **Compound Statement**

Note

The compound statement does not need a semicolon.

The Role of the Semicolon

- Every declaration in C is terminated by a semicolon
- Most statements in C are terminated by a semicolon

Statements and Defined Constants

```
#define TAX_RATE 0.825;

tax = TAX_RATE * amount;
```

3-7 Sample Programs

This section contains several programs that you should study for programming technique and style.

PROGRAM 3-9 Calculate Quotient and Remainder

```
 1  /* Calculate and print quotient and remainder of two
 2     numbers.
 3        Written by:
 4        Date:
 5  */
 6  #include <stdio.h>
 7
 8  int main (void)
 9  {
10  // Local Declarations
11     int  intNum1;
12     int  intNum2;
13     int  intCalc;
14
15  // Statements
16     printf("Enter two integral numbers: ");
17     scanf ("%d %d", &intNum1, &intNum2);
18
```

PROGRAM 3-9 Calculate Quotient and Remainder

```
19      intCalc = intNum1 / intNum2;
20      printf("%d / %d is %d", intNum1, intNum2, intCalc);
21
22      intCalc = intNum1 % intNum2;
23      printf(" with a remainder of: %d\n", intCalc);
24
25      return 0;
26  }   // main
```

```
Results:
Enter two integral numbers: 13 2
13 / 2 is 6 with a remainder of: 1
```

PROGRAM 3-10 Print Right Digit of Integer

```
 1  /* Print rightmost digit of an integer.
 2        Written by:
 3        Date:
 4  */
 5  #include <stdio.h>
 6
 7  int main (void)
 8  {
 9  // Local Declarations
10     int  intNum;
11     int  oneDigit;
12
13  // Statements
14     printf("Enter an integral number: ");
15     scanf ("%d", &intNum);
16
17     oneDigit = intNum % 10;
18     printf("\nThe right digit is: %d", oneDigit);
```

PROGRAM 3-10 Print Right Digit of Integer

```
19
20      return 0;
21  }  // main
```

```
Results:

Enter an integral number: 185

The right digit is: 5
```

PROGRAM 3-11 Calculate Average of Four Numbers

```
 1  /* Calculate the average of four integers and print
 2     the numbers and their deviation from the average.
 3        Written by:
 4        Date:
 5  */
 6  #include <stdio.h>
 7  int main (void)
 8  {
 9  // Local Declarations
10     int    num1;
11     int    num2;
12     int    num3;
13     int    num4;
14     int    sum;
15     float  average;
16
```

PROGRAM 3-11 Calculate Average of Four Numbers

```
17  // Statements
18     printf("\nEnter the first number  : ");
19     scanf("%d", &num1);
20     printf("Enter the second number : ");
21     scanf("%d", &num2);
22     printf("Enter the third  number : ");
23     scanf("%d", &num3);
24     printf("Enter the fourth number : ");
25     scanf("%d", &num4);
26
27     sum     = num1 + num2 + num3 + num4;
28     average = sum / 4.0;
29
30     printf("\n ******** average is %6.2f ******** ",
31               average);
32     printf("\n");
```

PROGRAM 3-11 Calculate Average of Four Numbers

```
33
34      printf("\nfirst number:  %6d -- deviation: %8.2f",
35              num1, num1 - average);
36      printf("\nsecond number: %6d -- deviation: %8.2f",
37              num2, num2 - average);
38      printf("\nthird number:  %6d -- deviation: %8.2f",
39              num3, num3 - average);
40      printf("\nfourth number: %6d -- deviation: %8.2f",
41              num4, num4 - average);
42
43      return 0;
44  }   // main
```

PROGRAM 3-11 Calculate Average of Four Numbers

```
Results:
Enter the first number:  23
Enter the second number: 12
Enter the third  number: 45
Enter the fourth number: 23

******** average is 25.75  ********

first number:      23 -- deviation:    -2.75
second number:     12 -- deviation:   -13.75
third number:      45 -- deviation:    19.25
fourth number:     23 -- deviation:    -2.75
```

PROGRAM 3-12 Convert Radians to Degrees

```
1  /* This program prompts the user to enter an angle
2     measured in radians and converts it into degrees.
3        Written by:
4        Date:
5  */
6  #include <stdio.h>
7
8  #define DEGREE_FACTOR 57.295779
9
10 int main (void)
11 {
12 // Local Declarations
13    double  radians;
14    double  degrees;
15
16 // Statements
17    printf("Enter the angle in radians: ");
18    scanf("%lf", &radians);
```

PROGRAM 3-12 Convert Radians to Degrees

```
19
20      degrees = radians * DEGREE_FACTOR;
21
22      printf("%6.3f radians is %6.3f degrees\n",
23                radians, degrees);
24      return 0;
25  }   // main
```

```
Results:
Enter the angle in radians: 1.57080
 1.571 radians is 90.000 degrees
```

PROGRAM 3-13 Calculate Sales Total

```
1  /* Calculates the total sale given the unit price,
2     quantity, discount, and tax rate.
3        Written by:
4        Date:
5  */
6  #include <stdio.h>
7
8  #define TAX_RATE 8.50
9
10 int main (void)
11 {
12 // Local Declarations
13    int    quantity;
14
15    float  discountRate;
16    float  discountAm;
17    float  unitPrice;
18    float  subTotal;
19    float  subTaxable;
```

PROGRAM 3-13 Calculate Sales Total

```
20      float  taxAm;
21      float  total;
22
23  // Statements
24      printf("\nEnter number of items sold:          ");
25      scanf("%d", &quantity);
26
27      printf("Enter the unit price:                 ");
28      scanf("%f", &unitPrice);
29
30      printf("Enter the discount rate (per cent): ");
31      scanf("%f", &discountRate);
32
33      subTotal    = quantity * unitPrice;
34      discountAm  = subTotal * discountRate / 100.0;
35      subTaxable  = subTotal - discountAm;
36      taxAm = subTaxable * TAX_RATE/ 100.00;
37      total = subTaxable + taxAm;
38
39      printf("\nQuantity sold:       %6d\n", quantity);
```

PROGRAM 3-13 Calculate Sales Total

```
40      printf("Unit Price of items: %9.2f\n", unitPrice);
41      printf("                     ------------\n");
42
43      printf("Subtotal :            %9.2f\n", subTotal);
44      printf("Discount:            -%9.2f\n", discountAm);
45      printf("Discounted total:     %9.2f\n", subTaxable);
46      printf("Sales tax:           +%9.2f\n", taxAm);
47      printf("Total sale:           %9.2f\n", total);
48
49      return 0;
50  }   // main
```

PROGRAM 3-13 Calculate Sales Total

```
Results:
Enter number of items sold:          34
Enter the unit price:                12.89
Enter the discount rate (per cent): 7

Quantity sold:            34
Unit Price of items:      12.89
                      ------------
Subtotal :               438.26
Discount:            -    30.68
Discounted total:        407.58
Sales tax:           +    34.64
Total sale:              442.23
```

PROGRAM 3-14 Calculate Student Score

```
1  /* Calculate a student's average score for a course
2     with 4 quizzes, 2 midterms, and a final. The quizzes
3     are weighted 30%, the midterms 40%, & the final 30%.
4        Written by:
5        Date:
6  */
7  #include <stdio.h>
8
9  #define QUIZ_WEIGHT     30
10 #define MIDTERM_WEIGHT  40
11 #define FINAL_WEIGHT    30
12 #define QUIZ_MAX       400.00
13 #define MIDTERM_MAX    200.00
14 #define FINAL_MAX      100.00
15
16 int main (void)
17 {
```

PROGRAM 3-14 Calculate Student Score

```
18  // Local Declarations
19      int     quiz1;
20      int     quiz2;
21      int     quiz3;
22      int     quiz4;
23      int     totalQuiz;
24      int     midterm1;
25      int     midterm2;
26      int     totalMidterm;
27      int     final;
28
29      float   quizPercent;
30      float   midtermPercent;
31      float   finalPercent;
32      float   totalPercent;
33
```

PROGRAM 3-14 Calculate Student Score

```
34  // Statements
35     printf("=========== QUIZZES ==================\n");
36     printf("Enter the score for the first quiz:  ");
37     scanf("%d", &quiz1);
38     printf("Enter the score for the second quiz: ");
39     scanf("%d", &quiz2);
40     printf("Enter the score for the third quiz:  ");
41     scanf("%d", &quiz3);
42     printf("Enter the score for the fourth quiz: ");
43     scanf("%d", &quiz4);
44
45     printf("============= MIDTERM  =============\n");
46     printf("Enter the score for the first midterm:  ");
47     scanf("%d", &midterm1);
48     printf("Enter the score for the second midterm: ");
49     scanf("%d", &midterm2);
50
```

PROGRAM 3-14 Calculate Student Score

```
51      printf("=============== FINAL ==============\n");
52      printf("Enter the score for the final: ");
53      scanf("%d", &final);
54      printf("\n");
55
56      totalQuiz = quiz1 + quiz2 + quiz3 + quiz4;
57      totalMidterm = midterm1 + midterm2;
58
59      quizPercent =
60         (float)totalQuiz * QUIZ_WEIGHT / QUIZ_MAX;
61      midtermPercent =
62         (float)totalMidterm * MIDTERM_WEIGHT / MIDTERM_MAX;
63      finalPercent =
64         (float)final * FINAL_WEIGHT / FINAL_MAX;
65
66      totalPercent =
67            quizPercent + midtermPercent + finalPercent;
68
```

PROGRAM 3-14 Calculate Student Score

```
69      printf("First Quiz  %4d\n",   quiz1);
70      printf("Second Quiz %4d\n",   quiz2);
71      printf("Third Quiz  %4d\n",   quiz3);
72      printf("Fourth Quiz %4d\n",   quiz4);
73      printf("Quiz Total  %4d\n\n", totalQuiz);
74
75      printf("First Midterm  %4d\n",   midterm1);
76      printf("Second Midterm %4d\n",   midterm2);
77      printf("Total Midterms %4d\n\n", totalMidterm);
78
79      printf("Final          %4d\n\n", final);
80
81      printf("Quiz    %6.1f%%\n" , quizPercent);
82      printf("Midterm %6.1f%%\n" , midtermPercent);
83      printf("Final   %6.1f%%\n" , finalPercent);
84      printf("--------------\n");
85      printf("Total   %6.1f%%\n" , totalPercent);
86
87      return 0;
88  }   // main
```

PROGRAM 3-14 Calculate Student Score

```
Results:
========== QUIZZES ================
Enter the score for the first quiz:  98
Enter the score for the second quiz: 89
Enter the score for the third quiz:  78
Enter the score for the fourth quiz: 79
============ MIDTERM  ============
Enter the score for the first midterm:  90
Enter the score for the second midterm: 100
============== FINAL ==============
Enter the score for the final: 92
```

PROGRAM 3-14 Calculate Student Score

```
First Quiz     98
Second Quiz    89
Third Quiz     78
Fourth Quiz    79
Quiz Total    344

First Midterm     90
Second Midterm   100
Total Midterms   190

Final             92

Quiz       25.8%
Midterm    38.0%
Final      27.6%
---------------
Total      91.4%
```

3-8 Software Engineering

In this section we discuss three concepts that, although technically not engineering principles, are important to writing clear and understandable programs.

Topics discussed in this section:

KISS
Parentheses
User Communication

KISS (Keep It Simple and Short)

Note

Blocks of code should be no longer than one screen.

Use parentheses whenever you are not sure of precedence.

Note

Computers do what you tell them to do,
not what you intended to tell them to do.

Make sure your code is as clear and simple as possible.

PROGRAM 3-15 Program That Will Confuse the User

```
 1  #include <stdio.h>
 2  int main (void)
 3  {
 4     int  i;
 5     int  j;
 6     int  sum;
 7
 8     scanf("%d%d", &i, &j);
 9     sum = i + j;
10     printf("The sum of %d & %d is %d\n", i, j, sum);
11     return 0;
12  }  //  main
```

Always provide the user with clear information

```
printf("Enter two integers and key <return>\n");
```