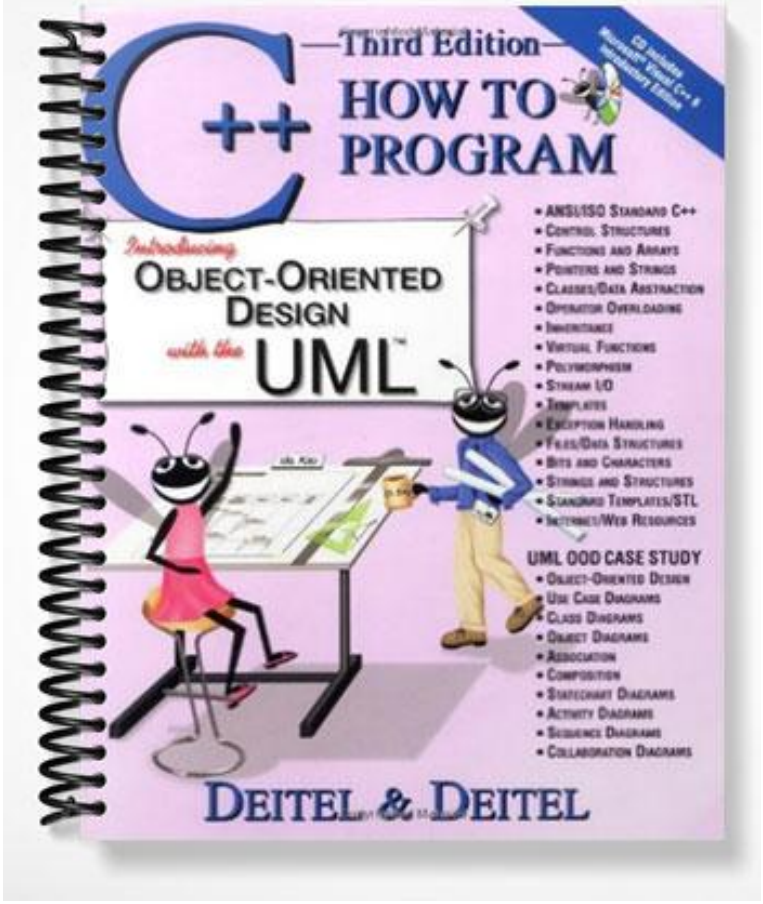
SOLUTIONS MANUAL



Instructor's Manual

for

C++ How to Program, 3/e

Deitel, Deitel & Nieto

C++ How to Program: Third Edition
Instructor's Manual Contents

Preface

Thank you for considering and/or adopting our text *C++ How to Program: Third Edition.* If you have not read the preface to *C++ How to Program: Third Edition*, please do so. The preface contains a careful walkthrough of the book's key features, including our new Unified Modeling Language™ (UML™) case study, which carefully introduces the reader to the UML and object-oriented design (OOD). Students are presented with a detailed problem statement and guided through a simplified, UML-based object-oriented design process. The complete 1000-line C++ program solution for the case study is presented in the book and provided on the CD-ROM in the back of the textbook.

We have worked hard to produce a textbook and ancillaries that we hope you and your students will find valuable. The following ancillary resources are available:

- *C++ How to Program: Third Edition's 250 program examples* are included on the CD-ROM in the back of the textbook. This helps instructors prepare lectures faster and helps students master C++. The examples are also available for download at **www.deitel.com**. When extracting the source code from the ZIP file, you must use a ZIP-file reader such as WinZip (**www.winzip.com**) or PKZIP (**www.pkware.com**) that understands directories. The file should be extracted into a separate directory (e.g., **cpphtp3e_examples**).
- *Microsoft Visual C++ 6 Introductory Edition software* is provided on the textbook's CD-ROM. This software allows students to edit, compile and debug C++ programs. We have provided at no charge a short Visual C++ 6 tutorial (in Adobe PDF format) on our Web site (**www.deitel.com**).
- This *C++ How to Program: Third Edition Instructor's Manual* on CD contains answers to most of the exercises in the textbook. The programs are separated into directories by chapter and exercise number.
- The optional *C++ Multimedia Cyber Classroom: Third Edition* is an interactive multimedia CD version of the book for Windows. Its features include audio walkthroughs of programs, section review questions (which are available only on the *C++ Multimedia Cyber Classroom: Third Edition*), a text-search engine, the ability to execute example programs, and more. The *Cyber Classroom* helps students get more out of their courses. The *Cyber Classroom* is also useful for students who miss a lecture and have to catch up quickly. The *Cyber Classroom* is available as a stand-alone product (see the last few pages of the textbook for the ISBN number) or bundled with the textbook (at a discount) in a product called *The Complete C++ Training Course: Third Edition* (ISBN# 0-13-089563-6).
- *Companion Web site* (**www.prenhall.com/deitel**) provides instructor and student resources. Instructor resources include textbook appendices (e.g., Appendix D, "C++ Internet and Web Resources") and a syllabus manager for lesson planning. Student resources include chapter objectives, true/false questions, chapter highlights, reference materials and a message board.
- Customizable *Powerpoint Instructor Lecture Notes*, with many complete features including source code, and key discussion points for each program and major illustration. These lecture notes are available for instructors and students at no charge at our Web site **www.deitel.com**.
- *Lab Manual* (available Spring 2001)—a for-sale item containing closed-lab sessions.

We would sincerely appreciate your questions, comments, criticisms and corrections addressed to us at:

deitel@deitel.com

We will respond immediately. Please read the latest copy of the *Deitel Buzz* (published every April and November) for information on forthcoming Deitel publications, ancillaries, product options and ordering information. To receive the *Deitel Buzz*, please contact Jennie Burger (**jennie_burger@prenhall.com**).

Watch our Deitel & Associates, Inc. Web site (**www.deitel.com**) and our Prentice Hall Web site (**www.prenhall.com/deitel**) for the latest publication updates.

We would like to thank the extraordinary team of publishing professionals at Prentice Hall who made *C++ How to Program: Third Edition* and its ancillaries possible. Our Computer Science editor, Petra Recter, worked closely with us to ensure the timely availability and professional quality of these ancillaries.

We would also like to thank two of our student interns—Aftab Bukhari (a Computer Science major at Boston University) and Jason Rosenfeld (a Computer Science major at Northwestern University) for their assistance in preparing this Instructor's Manual.

Harvey M. Deitel
Paul J. Deitel

1

Introduction to Computers and C++ Programming Solutions

SOLUTIONS

1.10 Categorize each of the following items as either hardware or software:

a) CPU

ANS: hardware.

b) C++ compiler

ANS: software.

c) ALU

ANS: hardware.

d) C++ preprocessor

ANS: software.

e) input unit

ANS: hardware.

f) an editor program

ANS: software.

1.11 Why might you want to write a program in a machine-independent language instead of a machine-dependent language? Why might a machine-dependent language be more appropriate for writing certain types of programs?

ANS: Machine independent languages are useful for writing programs to be executed on multiple computer platforms. Machine dependent languages are appropriate for writing programs to be executed on a single platform. Machine dependent languages tend to exploit the efficiencies of a particular machine.

1.12 Fill in the blanks in each of the following statements:

a) Which logical unit of the computer receives information from outside the computer for use by the computer? _____.

ANS: input unit.

b) The process of instructing the computer to solve specific problems is called _____.

ANS: computer programming.

c) What type of computer language uses English-like abbreviations for machine language instructions? _____.

ANS: high-level language.

d) Which logical unit of the computer sends information that has already been processed by the computer to various devices so that the information may be used outside the computer? _____.

ANS: output unit.

e) Which logical unit of the computer retains information? _____.

ANS: memory unit and secondary storage unit.

f) Which logical unit of the computer performs calculations? _____.

ANS: arithmetic and logical unit.

g) Which logical unit of the computer makes logical decisions? _____.

ANS: arithmetic and logical unit.

h) The level of computer language most convenient to the programmer for writing programs quickly and easily is _____.

ANS: high-level language.

i) The only language that a computer can directly understand is called that computer's _____.

ANS: machine language.

j) Which logical unit of the computer coordinates the activities of all the other logical units? _____.

ANS: central processing unit.

1.13 Discuss the meaning of each of the following objects:

a) **`cin`**

ANS: This object refers to the standard input device that is normally connected to the keyboard.

b) **`cout`**

ANS: This object refers to the standard output device that is normally connected to the computer screen.

c) **`cerr`**

ANS: This object refers to the standard error device that is normally connected to the computer screen.

1.14 Why is so much attention today focused on object-oriented programming in general and C++ in particular?

ANS: Object-oriented programming enables the programmer to build reusable software components that model items in the real world. Building software quickly, correctly, and economically has been an elusive goal in the software industry. The modular, object-oriented design and implementation approach has been found to increase productivity 10 to 100 times over conventional programming languages while reducing development time, errors, and cost. C++ is extremely popular because it is a superset of the widely used C programming language. Programmers already familiar with C have an easier time learing C++.

1.15 Fill in the blanks in each of the following:

a) _____ are used to document a program and improve its readability.

ANS: comments

b) The object used to print information on the screen is _____.

ANS: **`cout`**

c) A C++ statement that makes a decision is _____.

ANS: **`if`**

d) Calculations are normally performed by _____ statements.

ANS: assignment

e) The _____ object inputs values from the keyboard.

ANS: **`cin`**

1.16 Write a single C++ statement or line that accomplishes each of the following:

a) Print the message **`"Enter two numbers"`**.

ANS:
```
cout << "Enter two numbers";
```

b) Assign the product of variables **`b`** and **`c`** to variable **`a`**.

ANS:
```
a = b * c;
```

c) State that a program performs a sample payroll calculation (i.e., use text that helps to document a program).

ANS:
```
// Sample Payroll Calculation Program
```

d) Input three integer values from the keyboard and into integer variables **`a`**, **`b`** and **`c`**.

ANS:
```
cin >> a >> b >> c;
```

1.17 State which of the following are *true* and which are *false*. If *false*, explain your answers.

a) C++ operators are evaluated from left to right.

ANS: False. Some operators are evaluated from left to right, while other operators are evaluated right to left.

b) The following are all valid variable names: **`_under_bar_`**, **`m928134`**, **`t5`**, **`j7`**, **`her_sales`**, **`his_account_total`**, **`a`**, **`b`**, **`c`**, **`z`**, **`z2`**.

ANS: True. All variables begin with an underscore or letter.

c) The statement **`cout << "a = 5;"`**; is a typical example of an assignment statement.

ANS: False. The statement is an output statement. **`a = 5;`** is output to the screen.

d) A valid C++ arithmetic expression with no parentheses is evaluated from left to right.

ANS: False. Arithmetic operators can appear in any order in an expression. Since multiplication, division, and modulus have higher precendence than addition and subtraction the statement cannot be true.

e) The following are all invalid variable names: **`3g`**, **`87`**, **`67h2`**, **`h22`**, **`2h`**.

ANS: False. **`h22`** is a valid variable name.

1.18 Fill in the blanks in each of the following:

a) What arithmetic operations are on the same level of precedence as multiplication? _____.

ANS: division and modulus.

b) When parentheses are nested, which set of parentheses is evaluated first in an arithmetic expression? _____.

ANS: innermost.

c) A location in the computer's memory that may contain different values at various times throughout the execution of a program is called a _____.

ANS: variable.

1.19 What, if anything, prints when each of the following C++ statements is performed? If nothing prints, then answer "nothing." Assume **`x = 2`** and **`y = 3`**.

a) **`cout << x;`**

ANS: **`2`**

b) **`cout << x + x;`**

ANS: **`4`**

c) **`cout << "x=";`**

ANS: **`x=`**

d) **`cout << "x = " << x;`**

ANS: **`x = 2`**

e) **`cout << x + y << " = " << y + x;`**

ANS: **`5 = 5`**

f) **`z = x + y;`**

ANS: nothing.

g) **`cin >> x >> y;`**

ANS: 23.

h) **`// cout << "x + y = " << x + y;`**

ANS: nothing.

i) **`cout << "\n";`**

ANS: A newline is output which positions the cursor at the beginning of the next line on the screen.

1.20 Which of the following C++ statements contain variables whose values are replaced?

a) **`cin >> b >> c >> d >> e >> f;`**

b) **`p = i + j + k + 7;`**

c) **`cout << "variables whose values are destroyed";`**

d) **`cout << "a = 5";`**

ANS: Parts (a) and (b).

1.21 Given the algebraic equation $y = ax^3 + 7$, which of the following, if any, are correct C++ statements for this equation?

a) **`y = a * x * x * x + 7;`**

b) **`y = a * x * x * ( x + 7 );`**

c) **`y = ( a * x ) * x * ( x + 7 );`**

d) **`y = (a * x) * x * x + 7;`**

e) **`y = a * ( x * x * x ) + 7;`**

f) **`y = a * x * ( x * x + 7 );`**

ANS: Parts (a), (d) and (e).

1.22 State the order of evaluation of the operators in each of the following C++ statements and show the value of **`x`** after each statement is performed.

a) **`x = 7 + 3 * 6 / 2 - 1;`**

ANS: **`*, /, +, -, =, 15`**

b) **`x = 2 % 2 + 2 * 2 - 2 / 2;`**

ANS: **`%, *, /, +, -, =, 3`**

c) **`x = ( 3 * 9 * ( 3 + ( 9 * 3 / ( 3 ) ) ) );`**

ANS: **`*`**, **`/`**, **`+`**, **`*`**, **`*, 324`**

1.23 Write a program that asks the user to enter two numbers, obtains the two numbers from the user and prints the sum, product, difference, and quotient of the two numbers.

```
// Exercise 1.23 Solution
#include <iostream>

using std::cout;
using std::endl;
using std::cin;

int main()
{
   int num1, num2;  // declare variables

   cout << "Enter two integers: ";  // prompt user
   cin >> num1 >> num2;             // read values from keyboard

   // output the results
   cout << "The sum is " << num1 + num2
        << "\nThe product is " << num1 * num2
        << "\nThe difference is " << num1 - num2
        << "\nThe quotient is " << num1 / num2 << endl;

   return 0;  // indicate successful termination
}
```

```
Enter two integers: 8 22
The sum is 30
The product is 176
The difference is -14
The quotient is 0
```

1.24 Write a program that prints the numbers 1 to 4 on the same line with each pair of adjacent numbers separated by one space. Write the program using the following methods:

a) Using one output statement with one stream insertion operator.
b) Using one output statement with four stream insertion operators.
c) Using four output statements.

```
// Exercise 1.24 Solution
#include <iostream>

using std::cout;
using std::endl;

int main ()
{
   // Part A
   cout << "1 2 3 4\n";

   // Part B
   cout << "1 " << "2 " << "3 " << "4\n";

   // Part C
   cout << "1 ";
   cout << "2 ";
   cout << "3 ";
   cout << "4" << endl;

   return 0;
```

```
22  }
```

```
1 2 3 4
1 2 3 4
1 2 3 4
```

1.25 Write a program that asks the user to enter two integers, obtains the numbers from the user, then prints the larger number followed by the words "**is larger**." If the numbers are equal, print the message "**These numbers are equal**."

```
1  // Exercise 1.25 Solution
2  #include <iostream>
3
4  using std::cout;
5  using std::endl;
6  using std::cin;
7
8  int main()
9  {
10    int num1, num2;   // declaration
11
12    cout << "Enter two integers: ";  // prompt
13    cin >> num1 >> num2;             // input to numbers
14
15    if ( num1 == num2 )
16       cout << "These numbers are equal." << endl;
17
18    if ( num1 > num2 )
19       cout << num1 << " is larger." << endl;
20
21    if ( num2 > num1 )
22       cout << num2 << " is larger." << endl;
23
24    return 0;
25 }
```

```
Enter two integers: 22 8
22 is larger.
```

1.26 Write a program that inputs three integers from the keyboard and prints the sum, average, product, smallest and largest of these numbers. The screen dialogue should appear as follows:

```
Input three different integers: 13 27 14
Sum is 54
Average is 18
Product is 4914
Smallest is 13
Largest is 27
```

```
1  // Exercise 1.26 Solution
2  #include <iostream>
3
4  using std::cout;
```

```
using std::endl;
using std::cin;

int main()
{
   int num1, num2, num3, smallest, largest;  // declaration

   cout << "Input three different integers: ";  // prompt
   cin >> num1 >> num2 >> num3;                 // input

   largest = num1;  // assume first number is largest

   if ( num2 > largest )  // is num2 larger?
      largest = num2;

   if ( num3 > largest )  // is num3 larger?
      largest = num3;

   smallest = num1;  // assume first number is smallest

   if ( num2 < smallest )
      smallest = num2;

   if ( num3 < smallest )
      smallest = num3;

   cout << "Sum is " << num1 + num2 + num3
        << "\nAverage is " << (num1 + num2 + num3) / 3
        << "\nProduct is " << num1 * num2 * num3
        << "\nSmallest is " << smallest
        << "\nLargest is " << largest << endl;

   return 0;
}
```

```
Input three different integers: 13 27 14
Sum is 54
Average is 18
Product is 4914
Smallest is 13
Largest is 27
```

1.27 Write a program that reads in the radius of a circle and prints the circle's diameter, circumference and area. Use the constant value 3.14159 for π. Do these calculations in output statements. (Note: In this chapter, we have discussed only integer constants and variables. In Chapter 3 we will discuss floating-point numbers, i.e., values that can have decimal points.)

```
// Exercise 1.27 Solution
#include <iostream>

using std::cout;
using std::endl;
using std::cin;

int main()
{
   int radius;  // declaration

   cout << "Enter the circle radius: ";  // prompt
   cin >> radius;                        // input
```

```
14
15     cout << "Diameter is " << radius * 2.0
16          << "\nCircumference is " << 2 * 3.14159 * radius
17          << "\nArea is " << 3.14159 * radius * radius << endl;
18
19     return 0;
20  }
```

```
Enter the circle radius: 8
Diameter is 16
Circumference is 50.2654
Area is 201.062
```

1.28 Write a program that prints a box, an oval, an arrow and a diamond as follows:

```
*********          ***             *             *
*       *        *     *          ***           * *
*       *       *       *        *****         *   *
*       *       *       *          *          *     *
*       *       *       *          *         *       *
*       *       *       *          *          *     *
*       *       *       *          *           *   *
*       *        *     *           *            * *
*********          ***             *             *
```

```
1  // Exercise 1.28 Solution
2  #include <iostream>
3
4  using std::cout;
5  using std::endl;
6
7  main()
8  {
9     cout << "*********      ***         *         *\n"
10         << "*       *    *   *      *** *       * *\n"
11         << "*       *   *     *    *****     *   *\n"
12         << "*       *   *     *      *      *     *\n"
13         << "*       *   *     *      *     *       *\n"
14         << "*       *   *     *      *      *     *\n"
15         << "*       *    *   *       *       *   *\n"
16         << "*       *     * *        *        * *\n"
17         << "*********      ***       *         *" << endl;
18
19    return 0;
20 }
```

1.29 What does the following code print?

```
cout << "*\n**\n***\n****\n*****\n";
```

ANS:

```
*
**
***
****
*****
```

1.30 Write a program that reads in five integers and determines and prints the largest and the smallest integers in the group. Use only the programming techniques you learned in this chapter.

```
// Exercise 1.30 Solution
#include <iostream>

using std::cout;
using std::endl;
using std::cin;

int main()
{
   int num1, num2, num3, num4, num5, largest, smallest;

   cout << "Enter five integers: ";
   cin >> num1 >> num2 >> num3 >> num4 >> num5;

   largest = num1;
   smallest = num1;

   if ( num1 > largest )
      largest = num1;

   if ( num2 > largest )
      largest = num2;

   if ( num3 > largest )
      largest = num3;

   if ( num4 > largest )
      largest = num4;

   if ( num5 > largest )
      largest = num5;

   if ( num1 < smallest )
      smallest = num1;

   if ( num2 < smallest )
      smallest = num2;

   if ( num3 < smallest )
      smallest = num3;

   if ( num4 < smallest )
      smallest = num4;

   if ( num5 < smallest )
      smallest = num5;

   cout << "Largest is " << largest
```

```
         << "\nSmallest is " << smallest << endl;

   return 0;
}
```

```
Enter five integers: 88 22 8 78 21
Largest is 88
Smallest is 8
```

1.31 Write a program that reads an integer and determines and prints whether it is odd or even. (Hint: Use the modulus operator. An even number is a multiple of two. Any multiple of two leaves a remainder of zero when divided by 2.)

```
// Exercise 1.31 Solution
#include <iostream>

using std::cout;
using std::endl;
using std::cin;

int main()
{
   int num;

   cout << "Enter a number: ";
   cin >> num;

   if ( num % 2 == 0 )
      cout << "The number " << num << " is even." << endl;

   if ( num % 2 != 0 )
      cout << "The number " << num << " is odd." << endl;

   return 0;
}
```

```
Enter a number: 73
The number 73 is odd.
```

1.32 Write a program that reads in two integers and determines and prints if the first is a multiple of the second. (Hint: Use the modulus operator.)

```
// Exercise 1.32 Solution
#include <iostream>

using std::cout;
using std::endl;
using std::cin;

int main()
{
   int num1, num2;

   cout << "Enter two integers: ";
   cin >> num1 >> num2;

   if ( num1 % num2 == 0 )
      cout << num1 << " is a multiple of " << num2 << endl;
```

```
17
18    if ( num1 % num2 != 0 )
19       cout << num1 << " is not a multiple of " << num2 << endl;
20
21    return 0;
22 }
```

```
Enter two integers: 22 8
22 is not a multiple of 8
```

1.33 Display a checkerboard pattern with eight output statements, then display the same pattern with as few output statements as possible.

```
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
```

```
1  // Exercise 1.33 Solution
2  #include <iostream>
3
4  using std::cout;
5  using std::endl;
6
7  int main()
8  {
9     // Eight output statements
10    cout << "* * * * * * * *\n";
11    cout << " * * * * * * * *\n";
12    cout << "* * * * * * * *\n";
13    cout << " * * * * * * * *\n";
14    cout << "* * * * * * * *\n";
15    cout << " * * * * * * * *\n";
16    cout << "* * * * * * * *\n";
17    cout << " * * * * * * * *\n\n";
18
19    // One output statement; 3 parts
20    cout << "* * * * * * * *\n * * * * * * * *\n* * * * * * * *\n"
21         << " * * * * * * * *\n* * * * * * * *\n  * * * * * * * *\n"
22         << "* * * * * * * *\n * * * * * * * *\n";
23
24    cout << endl;   // ensure everything is displayed
25
26    return 0;
27 }
```

```
 * * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *

* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
* * * * * * * *
  * * * * * * * *
* * * * * * * *
 * * * * * * * *
```

1.34 Distinguish between the terms fatal error and non–fatal error. Why might you prefer to experience a fatal error rather than a non–fatal error?

ANS: A fatal error causes a program to terminate prematurely. A nonfatal error occurs when the logic of the program is incorrect, and the program does not work properly. A fatal error is preferred for debugging purposes. A fatal error immediately lets you know there is a problem with the program, whereas a nonfatal error can be subtle and possibly go undetected.

1.35 Here is a peek ahead. In this chapter you learned about integers and the type **int**. C++ can also represent uppercase letters, lowercase letters and a considerable variety of special symbols. C++ uses small integers internally to represent each different character. The set of characters a computer uses and the corresponding integer representations for those characters is called that computer's *character set.* You can print a character by simply enclosing that character in single quotes as with

```
cout << 'A';
```

You can print the integer equivalent of a character using **static_cast** as follows:

```
cout << static_cast< int >( 'A' );
```

This is called a *cast* operation (we formally introduce casts in Chapter 2). When the preceding statement executes, it prints the value 65 (on systems that use the *ASCII character set*). Write a program that prints the integer equivalents of some uppercase letters, lowercase letters, digits and special symbols. At a minimum, determine the integer equivalents of the following: **A B C a b c 0 1 2 $ * + /** and the blank character.

```
1  // Exercise 1.35 Solution
2  #include <iostream>
3
4  using std::cout;
5  using std::endl;
6  using std::cin;
7
8  int main()
9  {
10    char symbol;
11
12    cout << "Enter a character: ";
13    cin >> symbol;
14
15    cout << symbol << "'s integer equivalent is "
16         << static_cast< int > ( symbol ) << endl;
17
18    return 0;
```

```
}
```

```
Enter a character: A
A's integer equivalent is 65
```

1.36 Write a program that inputs a five-digit number, separates the number into its individual digits and prints the digits separated from one another by three spaces each. (Hint: Use the integer division and modulus operators.) For example, if the user types in **42339** the program should print

```
4   2   3   3   9
```

```
// Exercise 1.36 Solution
#include <iostream>

using std::cout;
using std::endl;
using std::cin;

int main()
{
   int num;

   cout << "Enter a five-digit number: ";
   cin >> num;

   cout << num / 10000 << "   ";
   num = num % 10000;
   cout << num / 1000 << "   ";
   num = num % 1000;
   cout << num / 100 << "   ";
   num = num % 100;
   cout << num / 10 << "   ";
   num = num % 10;
   cout << num << endl;

   return 0;
}
```

```
Enter a five-digit number: 42339
4   2   3   3   9
```

1.37 Using only the techniques you learned in this chapter, write a program that calculates the squares and cubes of the numbers from 0 to 10 and uses tabs to print the following table of values:

```
number  square  cube
0       0       0
1       1       1
2       4       8
3       9       27
4       16      64
5       25      125
6       36      216
7       49      343
8       64      512
9       81      729
10      100     1000
```

```
1  // Exercise 1.37 Solution
2  #include <iostream>
3
4  using std::cout;
5  using std::endl;
6
7  int main()
8  {
9     int num;
10
11    num = 0;
12    cout << "\nnumber\tsquare\tcube\n"
13         << num << '\t' << num * num << '\t' << num * num * num << "\n";
14
15    num = num + 1;
16    cout << num << '\t' << num * num << '\t' << num * num * num << "\n";
17
18    num = num + 1;
19    cout << num << '\t' << num * num << '\t' << num * num * num << "\n";
20
21    num = num + 1;
22    cout << num << '\t' << num * num << '\t' << num * num * num << "\n";
23
24    num = num + 1;
25    cout << num << '\t' << num * num << '\t' << num * num * num << "\n";
26
27    num = num + 1;
28    cout << num << '\t' << num * num << '\t' << num * num * num << "\n";
29
30    num = num + 1;
31    cout << num << '\t' << num * num << '\t' << num * num * num << "\n";
32
33    num = num + 1;
34    cout << num << '\t' << num * num << '\t' << num * num * num << "\n";
35
36    num = num + 1;
37    cout << num << '\t' << num * num << '\t' << num * num * num << "\n";
38
39    num = num + 1;
40    cout << num << '\t' << num * num << '\t' << num * num * num << "\n";
41
42    num = num + 1;
43    cout << num << '\t' << num * num << '\t' << num * num * num << endl;
```

```
44
45    return 0;
46 }
```

1.38 Give a brief answer to each of the following "object think" questions:

a) Why does this text choose to discuss structured programming in detail before proceeding with an in-depth treatment of object-oriented programming?

ANS: Objects are composed in part by structured program pieces.

b) What are the typical steps (mentioned in the text) of an object-oriented design process?

ANS: (1) Determine which objects are needed to implement the system. (2) Determine's each object's attributes. (3) Determine each object's behaviors. (4) Determine the interaction between the objects.

c) How is multiple inheritance exhibited by human beings?

ANS: Children. A child receives genes from both parents.

d) What kinds of messages do people send to one another?

ANS: People send messages through body language, speech, writings, email, telephones, etc.

e) Objects send messages to one another across well-defined interfaces. What interfaces does a car radio (object) present to its user (a person object)?

ANS: Dials and buttons that allow the user to select a station, adjust the volume, adjust bass and treble, play a CD or tape, etc.

1.39 You are probably wearing on your wrist one of the world's most common types of objects—a watch. Discuss how each of the following terms and concepts applies to the notion of a watch: object, attributes, behaviors, class, inheritance (consider, for example, an alarm clock), abstraction, modeling, messages, encapsulation, interface, information hiding, data members and member functions.

ANS: The entire watch is an object that is composed of many other objects (such as the moving parts, the band, the face, etc.) Watch attributes are time, color, band, style (digital or analog), etc. The behaviors of the watch include setting the time and getting the time. A watch can be considered a specific type of clock (as can an alarm clock). With that in mind, it is possible that a class called **`Clock`** could exist from which other classes such as watch and alarm clock can inherit the basic features in the clock. The watch is an abstraction of the mechanics needed to keep track of the time. The user of the watch does not need to know the mechanics of the watch in order to use it; the user only needs to know that the watch keeps the proper time. In this sense, the mechanics of the watch are encapsulated (hidden) inside the watch. The interface to the watch (its face and controls for setting the time) allows the user to set and get the time. The user is not allowed to directly touch the internal mechanics of the watch. All interaction with the internal mechanics is controlled by the interface to the watch. The data members stored in the watch are hidden inside the watch and the member functions (looking at the face to get the time and setting the time) provide the interface to the data.

Control Structures Solutions

Solutions

Exercises 2.14 through 2.38 correspond to Sections 2.1 through 2.12.
Exercises 2.39 through 2.63 correspond to Sections 2.13 through 2.21.

2.14 Identify and correct the error(s) in each of the following:

a)
```
if ( age >= 65 );
   cout << "Age is greater than or equal to 65" << endl;
else
   cout << "Age is less than 65 << endl";
```
ANS: The semicolon at the end of the `if` should be removed. The closing double quote after the second `endl` should be placed after `65`.

b)
```
if ( age >= 65 )
   cout << "Age is greater than or equal to 65" << endl;
else;
   cout << "Age is less than 65 << endl";
```
ANS: The semicolon after the `else` should be removed. The closing double quote after the second `endl` should be placed after `65`.

c)
```
int x = 1, total;
while ( x <= 10 ) {
   total += x;
   ++x;
}
```
ANS: Variable `total` should be initialized to `0`.

d)
```
While ( x <= 100 )
   total += x;
   ++x;
```
ANS: The `W` in `while` should be lowercase. The `while`'s body should be enclosed in braces `{}`.

e)
```
while ( y > 0 ) {
   cout << y << endl;
   ++y;
}
```
ANS: The variable `y` should be decremented (i.e., `--y;`) not incremented (`++y;`).

2.15 What does the following program print?

```
#include <iostream>

using std::cout;
using std::endl;

int main()
{
   int y, x = 1, total = 0;

   while ( x <= 10 ) {
      y = x * x;
      cout << y << endl;
      total += y;
      ++x;
   }

   cout << "Total is " << total << endl;
   return 0;
}
```

```
1
4
9
16
25
36
49
64
81
100
Total is 385
```

For Exercises 2.16 to 2.19, perform each of these steps:

a) Read the problem statement.
b) Formulate the algorithm using pseudocode and top-down, stepwise refinement.
c) Write a C++ program.
d) Test, debug and execute the C++ program.