4

Control Statements: Part 1

Let's all move one place on.
—Lewis Carroll

The wheel is come full circle.
—William Shakespeare

How many apples fell on Newton's head before he took the hint!
—Robert Frost

All the evolution we know of proceeds from the vague to the definite.

OBJECTIVES

In this chapter you'll learn:

- Basic problem-solving techniques.
- To develop algorithms through the process of top-down, stepwise refinement.
- To use the `if` and `if...else` selection statements to choose among alternative actions.
- To use the `while` repetition statement to execute statements in a program repeatedly.
- Counter-controlled repetition and sentinel-controlled repetition.
- To use the increment, decrement and assignment operators.

Assignment Checklist

Name: _____ Date: _____

Section: _____

Exercises	Assigned: Circle assignments	Date Due
Prelab Activities		
Matching	YES NO	
Fill in the Blank	13, 14, 15, 16, 17, 18, 19, 20, 21	
Short Answer	22, 23, 24, 25, 26, 27, 28	
Programming Output	29, 30, 31, 32, 33, 34	
Correct the Code	35, 36, 37, 38, 39	
Lab Exercises		
Lab Exercise 1 — Department Store	YES NO	
Follow-Up Questions and Activities	1, 2	
Lab Exercise 2 — Payroll	YES NO	
Follow-Up Questions and Activities	1, 2	
Lab Exercise 3 — Square of Asterisks	YES NO	
Follow-Up Question and Activity	1	
Debugging	YES NO	
Labs Provided by Instructor		
1.		
2.		
3.		
Postlab Activities		
Coding Exercises	1, 2, 3, 4, 5, 6	
Programming Challenges	1, 2, 3, 4	

Prelab Activities

Matching

Name: _____ **Date:** _____

Section: _____

After reading Chapter 4 of *C++ How to Program: Fourth Edition*, answer the given questions. These questions are intended to test and reinforce your understanding of key concepts and may be done either before the lab or during the lab.

For each term in the column on the left, write the corresponding letter for the description that best matches it from the column on the right.

Term	Description
___ 1. `double`	a) Can be used to create a temporary floating-point copy of its operand.
___ 2. Pseudocode	b) Control structure that is used to choose among alternative courses of action.
___ 3. Algorithm	c) Addition assignment operator.
___ 4. Sequence	d) Conditional operator.
___ 5. Selection	e) Control structure that allows programmers to specify an action to be repeated while some condition is true.
___ 6. Sentinel	f) Control structure that causes statements to execute in the order in which they appear.
___ 7. `++`	g) A data type for storing floating-point values.
___ 8. `static_cast< double >`	h) Increment operator.
___ 9. Nested control statement	i) Special value which indicates the end of data entry.
___ 10. `+=`	j) Procedure for solving problems in terms of actions and the order in which they should be performed.
___ 11. `?:`	k) Artificial and informal language for program development.
___ 12. Repetition	l) Appears in the body of another control statement.

Prelab Activities

Name:

Fill in the Blank

Name: _____ **Date:** _____

Section: _____

Fill in the blanks in each of the following statements:

13. A(n) _____ error causes a program to fail and terminate prematurely.

14. The _____ selection structure performs an indicated action only when its condition is `true`; otherwise, the action is skipped.

15. _____-controlled repetition is preferred when the number of repetitions is not known before the loop begins executing.

16. _____ repetition is preferred when the number of repetitions is known before a loop begins executing.

17. The unary cast operator _____ creates a temporary copy of its operand.

18. _____ operators take only one operand; _____ operators take two.

19. A block can be placed anywhere in a program that a _____ can be placed.

20. C++ provides the _____ operators for abbreviating assignment expressions.

21. A null statement, indicating that no action is to be taken, is indicated by a(n) _____.

Prelab Activities

Name:

Short Answer

Name: _____ **Date:** _____

Section: _____

In the space provided, answer each of the given questions. Your answers should be as concise as possible; aim for two or three sentences.

22. When is sentinel-controlled repetition necessary and how is it implemented?

23. List the three selection statements and three repetition statements.

24. What are the two ways in which control structures can be combined?

25. What is the difference between the preincrement and postincrement operators, ++x and x++, respectively?

Prelab Activities

Name:

Short Answer

26. What is the difference between a logic error and a syntax error?

27. Define and discuss the difference between unary, binary and ternary operators. Give an example of each.

28. When must you use curly braces (`{}`) with a selection or repetition statement?

Prelab Activities

Name:

Programming Output

Name: _____ **Date:** _____

Section: _____

For each of the given program segments, read the code and write the output in the space provided below each program. [*Note:* Do not execute these programs on a computer.]

29. What is output by the following program segment?

```
1  int x = 1;
2
3  while ( x <= 5 )
4  {
5     x++;
6     cout << "The value of x is: " << x << endl;
7  }
8
9  cout << "The final value of x is: " << x << endl;
```

Your answer:

30. What is output by the following program segment?

```
1  int x = 2;
2  int y = 4;
3
4  cout << "Hi" << endl;
5
6  if ( x > 4 )
7     if ( y > 3 )
8        cout << "Bye" << endl;
```

Your answer::

Prelab Activities

Name:

Programming Output

31. What is output by the following program segment?

```
int grade1 = 85;
int grade2 = 55;

cout << "The student with a grade of " << grade1 << " "
     << ( grade1 >= 60 ? "Passed\n" : "Failed\n" );

cout << "The student with a grade of " << grade2 << " "
     << ( grade2 >= 60 ? "Passed\n" : "Failed\n" );
```

Your answer:

32. What is output by the following program segment?

```
int x = 1;

while ( x < 10 )
{
   if ( x % 2 == 0 )
      cout << x << " ";

   x++;
}
```

Your answer::

Prelab Activities

Name:

Programming Output

33. What is output by the following program segment?

```
1  int grade = 60;
2
3  if ( grade < 60 )
4     cout << "Failed";
5  else
6     cout << "Passed";
```

Your answer::

34. What is output by the following program segment?

```
1  int a = 6;
2  int b = 2;
3  int c = -4;
4  int d = 1;
5
6  a += c;
7  d -= b;
8  b /= a;
9  c *= a;
10
11 cout << a << " " << b << " " << c << " " << d;
```

Your answer:

Prelab Activities

Name:

Correct the Code

Name: _____ **Date:** _____

Section: _____

For each of the given program segments, determine if there is an error in the code. If there is an error, specify whether it is a logic or compilation error, circle the error in the program and write the corrected code in the space provided after each problem. If the code does not contain an error, write "no error." [*Note:* It is possible that a program segment may contain multiple errors.]

35. The following program segment should calculate if a student has a passing grade. If so, the code should print `"Passed."`. Otherwise, the code should print both `"Failed."` and `"You must take this course again."`.

```
1  if ( grade >= 60 )
2     cout << "Passed.\n"
3  else
4     cout << "Failed.\n"
5     cout << "You must take this course again.\n"
```

Your answer:

36. The following program segment should input 15 integers from the user and calculate their total.

```
1  int total = 0;
2  int counter = 1;
3  int input;
4
5  while ( total <= 15 )
6  {
7     cin >> input;
8     total += input;
9     counter++;
10 }
```

Prelab Activities

Name:

Correct the Code

Your answer:

37. The following program segment should input and sum integers from the user until the sentinel value, -1, is entered.

```
1  int total = 0;
2  int input;
3
4  while ( input != -1 )
5  {
6     cin >> input;
7     total += input;
8  }
```

Your answer:

38. The following program segment should output the number 9.

```
1  int x = 4;
2
3  cout << ( x++ + 4 );
```

Prelab Activities

Name:

Correct the Code

Your answer:

39. The following `while` loop should compute the product of all integers between `1` and `5`, inclusive.

```
1  int i = 1;
2  int product = 1;
3
4  while ( i <= 5 );
5     product *= i;
```

Your answer:

Lab Exercises

Lab Exercise 1—Department Store

Name: _____ **Date:** _____

Section: _____

This problem is intended to be solved in a closed-lab session with a teaching assistant or instructor present. The problem is divided into six parts:

1. Lab Objectives
2. Description of the Problem
3. Sample Output
4. Program Template (Fig. L 4.1)
5. Problem-Solving Tips
6. Follow-Up Questions and Activities

The program template represents a complete working C++ program, with one or more key lines of code replaced with comments. Read the problem description and examine the sample output; then study the template code. Using the problem-solving tips as a guide, replace the `/* */` comments with C++ code. Compile and execute the program. Compare your output with the sample output provided. Then answer the follow-up questions. The source code for the template is available from the Companion Website for *C++ How to Program, Seventh Edition* at `www.pearsonhighered.com/deitel/`.

Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 4 of *C++ How To Program, Seventh Edition*. In this lab, you will practice:

- Using sentinel-controlled repetition.
- Using selection structures.

The follow-up questions and activities also will give you practice:

- Recognizing invalid user input.

Description of the Problem

Develop a C++ program that will determine whether a department-store customer has exceeded the credit limit on a charge account. For each customer, the following information is available:

a) Account number (an integer);

b) Balance at the beginning of the month;

c) Total of all items charged by the customer this month;

d) Total of all credits applied to the customer's account this month;

e) Allowed credit limit.

The program should use a `while` loop to input each of these facts, calculate the new balance (= beginning balance + charges – credits) and determine whether the new balance exceeds the customer's credit limit. For those customers whose credit limit is exceeded, the program should display the customer's account number, credit limit, new balance and the message `"Credit Limit Exceeded."`

Lab Exercises

Name:

Lab Exercise 1—Department Store

Sample Output

```
Enter account number (-1 to end): 100
Enter beginning balance: 5394.78
Enter total charges: 1000.00
Enter total credits: 500.00
Enter credit limit: 5500.00
New balance is 5894.78
Account:      100
Credit limit: 5500.00
Balance:      5894.78
Credit Limit Exceeded.

Enter account number (-1 to end): 200
Enter beginning balance: 1000.00
Enter total charges: 123.45
Enter total credits: 321.00
Enter credit limit: 1500.00
New balance is 802.45

Enter account number (-1 to end): 300
Enter beginning balance: 500.00
Enter total charges: 274.73
Enter total credits: 100.00
Enter credit limit: 800.00
New balance is 674.73

Enter account number (-1 to end): -1
```

Template

```
1  // Lab 1: customerCredit.cpp
2  // Create Credit object and invoke its calculateBalance function.
3  #include <iostream>
4  #include <iomanip> // parameterized stream manipulators
5  using namespace std;
6
7  int main()
8  {
9     int account; // account number
10    /* Write declarations for four double variables:
11       balance, charges, credits and creditLimit */
12
13    cout << "Enter account number (-1 to end): ";
14    /* Write code to read the customer's account number */
15
16    // set floating-point number format
17    /* Write code to use the stream manipulators fixed and
18       setprecision to set the floating-point number format */
19
20    // exit if the input is -1 otherwise, proceed with the program
21    /* Write a while loop with a condition that tests for the sentinel value */
22    {
```

Fig. L 4.1 | Contents of customerCredit.cpp (Part 1 of 2.)

Lab Exercises

Name:

Lab Exercise 1—Department Store

```
23         /* Write code to prompt for and input the customer's balance */
24
25         /* Write code to prompt for and input the customer's charges */
26
27         /* Write code to prompt for and input the customer's credits */
28
29         /* Write code to prompt for and input the customer's credit limit */
30
31         // calculate and display new balance
32         /* Write a statement to calculate the customer's new balance */
33         /* Display the new balance */
34
35         // display a warning if the user has exceed the credit limit
36         /* Write an if statement to determine whether the credit limit is exceeded */
37            /* Display a message if the credit limit was exceeded */
38
39         cout << "\n\nEnter Account Number (or -1 to quit): ";
40         /* Write code to input the next account number */
41      } // end while
42   } // end main
```

Fig. L 4.1 | Contents of `customerCredit.cpp`. (Part 2 of 2.)

Problem-Solving Tips

1. Notice that the number of customers is not specified in advance. The problem statement implies that you should use a sentinel-controlled the loop.
2. The input data consists of one integer and four monetary amounts. The monetary amounts are numbers with decimal points, so you will use type `double` to represent them.
3. You will use a `while` loop to process several sets of customer data.
4. What sentinel value should you use? It needs to be a value that will not be confused with a legitimate data value. A good choice for this program would be to use `-1` for the customer's account number, because account numbers are nonnegative integers.
5. Before the loop, you will prompt for the account. Inside the loop, you will prompt for the monetary amounts. It is a good practice to remind the user of the sentinel value in each prompting message. After processing the data, you obtain the next account number.
6. While you could prompt each time for all five facts needed for a customer, a better strategy would be to prompt just for the account number. If it is negative, terminate the loop. If it is nonnegative, prompt for the remaining four pieces of data for that customer.
7. How do you determine whether the new balance exceeds the credit limit? You have already input the credit limit, but you will need to calculate the new balance which is equal to the starting balance, plus the charges, minus the credits.
8. Be sure to follow the spacing and indentation conventions mentioned in the text. Before and after each control statement, place a line of vertical space to make the control statement stand out. Indent all the body statements of `main`, and indent all the body statements of each control statement.
9. Print your outputs neatly in the indicated format. Use `setprecision( 2 )` to force two positions of precision to the right of the decimal point when printing monetary amounts. You will need to `#include <iomanip>` to use function `setprecision( 2 )`.
10. Also remember to use the `fixed` manipulator. This manipulator specifies that the number output should be output in fixed notation, as opposed to scientific notation. Fixed notation

Lab Exercises

Name:

Lab Exercise 1—Department Store

always keeps a "fixed" decimal point, regardless of the number of digits displayed. For example, 1.50000000000 is expressed in fixed notation. Scientific notation displays numbers in a format such as 1.5E+10.

11. If you have any questions as you proceed, ask your lab instructor for help.

Follow-Up Questions and Activities

1. Execute the program, and enter a floating-point number when you are prompted for an account number. Observe what happens. You are no longer asked to input a `beginning balance`. Your output screen may resemble the output accompanying this question. Instead, the program prompts you to enter only `total charges`, `total credits` and `credit limit`. Why does this situation occur?

```
Enter account number (-1 to end): 123.456
Enter beginning balance: Enter total charges:
```

2. Why is it necessary to ask the user to input the first account number before you begin the `while` loop? What problems could occur if the user were asked for an account number only inside the `while` loop?

Lab Exercises

Name:

Lab Exercise 2 — Payroll

Name: _____ **Date:** _____

Section: _____

This problem is intended to be solved in a closed-lab session with a teaching assistant or instructor present. The problem is divided into six parts:

1. Lab Objectives
2. Description of the Problem
3. Sample Output
4. Program Template (Fig. L 4.2)
5. Problem-Solving Tips
6. Follow-Up Questions and Activities

The program template represents a complete working C++ program, with one or more key lines of code replaced with comments. Read the problem description and examine the sample output; then study the template code. Using the problem-solving tips as a guide, replace the `/* */` comments with C++ code. Compile and execute the program. Compare your output with the sample output provided. Then answer the follow-up questions. The source code for the template is available from the Companion Website for *C++ How to Program, Seventh Edition* at `www.pearsonhighered.com/deitel/`.

Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 4 of *C++ How To Program, Seventh Edition*. In this lab, you will practice:

- Using sentinel-controlled repetition.
- Using `if...else` selection statements.

The follow-up questions and activities also will give you practice:

- Experimenting with `<iomanip>`.
- Using nested `if...else` statements.

Description of the Problem

Develop a C++ program that uses a `while` statement to determine the gross pay for each of several employees. The company pays "straight time" for the first 40 hours worked by each employee and pays "time-and-a-half" for all hours worked in excess of 40 hours. You are given a list of the employees of the company, the number of hours each employee worked last week and the hourly rate of each employee. Your program should input this information for each employee and should determine and display the employee's gross pay.

Lab Exercises

Name:

Lab Exercise 2 — Payroll

Sample Output

```
Enter hours worked (-1 to end): 39
Enter hourly rate of the worker ($00.00): 10.00
Salary is $390.00

Enter hours worked (-1 to end): 40
Enter hourly rate of the worker ($00.00): 10.00
Salary is $400.00

Enter hours worked (-1 to end): 41
Enter hourly rate of the worker ($00.00): 10.00
Salary is $415.00

Enter hours worked (-1 to end): -1
```

Template

```
1  // Lab 2: Payroll.cpp
2  // Calculate wages for a payroll.
3  #include <iostream>
4  #include <iomanip> // parameterized stream manipulators
5  using namespace std;
6
7  int main()
8  {
9     /* Write declarations for hours, rate and salary */
10
11    // processing phase
12    // get first employee's hours worked
13    cout << "Enter hours worked (-1 to end): ";
14    cin >> hours;
15
16    // set floating-point number format
17    /* Write code to use the stream manipulators fixed and
18       setprecision to set the floating-point number format */
19
20    // loop until sentinel value read from user
21    /* Write the while loop */
22    {
23       // get hourly rate
24       /* Write code to prompt for and input hourly rate */
25
26       // if employee worked less than 40 hours
27       /* Write code to determine whether hours worked was
28          less than or equal to 40 and if so, calculate
29          base pay. If not, calculate base + overtime pay */
30
31       /* Write code to display the salary */
32
```

Fig. L 4.2 | Contents of `Payroll.cpp`. (Part 1 of 2.)

Lab Exercises

Name:

Lab Exercise 2 — Payroll

```
33         // prompt for next employee's data
34         cout << "\n\nEnter hours worked (-1 to end): ";
35         cin >> hours;
36      } // end while
37   } // end main
```

Fig. L 4.2 | Contents of `Payroll.cpp`. (Part 2 of 2.)

Problem-Solving Tips

1. Notice that the number of employees is not specified in advance. The problem statement implies that you should use a sentinel value to control the loop.
2. The input data consists of two `doubles` per employee, one for the number of hours worked and one for the hourly rate.
3. You will use a `while` loop to process several sets of employee data.
4. Before the loop, you will prompt for the number of hours worked. Inside the loop, you will prompt for the hourly rate. It is a good programming practice to remind the user of the sentinel value in each prompting message. After processing the data, you obtain the next employee's number of hours worked.
5. What sentinel value should you use? It needs to be a value that will not be confused with one of the legitimate data input values. A good choice for this program would be to use `-1` for the employee's number of hours worked, because the number of hours worked must be a nonnegative value.
6. While you could prompt each time for both pieces of information needed for a customer, a better strategy would be to prompt just for the number of hours worked. If it is `-1`, terminate the loop. If it is not `-1`, prompt for the remaining piece of data for that employee, namely, the hourly rate.
7. How do you calculate the pay for each employee? If the person worked 40 hours or less, multiply the number of hours worked times the hourly salary; otherwise (`else`), calculate the pay by multiplying 40 times the hourly salary rate and adding to this value to the number of overtime hours times the overtime rate. The number of overtime hours is the number of hours in excess of the first 40 hours worked (i.e., *hours – 40*), and the overtime salary is 1.5 times the hourly rate.
8. Be sure to follow the spacing and indentation conventions mentioned in the text. Before and after each control statement, place a line of vertical space to make the control statement stand out. Indent all the body statements of `main` and indent all of the body statements of each control statement.
9. Print your outputs neatly in the indicated format. Remember to use `setprecision( 2 )` to force two positions of precision to the right of the decimal point when printing monetary amounts. You will need to `#include <iomanip>` to use function `setprecision( 2 )`.
10. Also remember to use manipulator `fixed`. This manipulator specifies that the number output should be output in fixed notation, as opposed to scientific notation (`scientific`). Fixed notation always keeps a "fixed" decimal point, regardless of the number of digits displayed. For example, 1.50000000000 is expressed in fixed notation. Scientific notation displays numbers in a format such as 1.5E+10.
11. If you have any questions as you proceed, ask your lab instructor for help.

Lab Exercises

Name:

Lab Exercise 2 — Payroll

Follow-Up Questions and Activities

1. Remove the `setprecision` manipulator on line 23. The line should now read as follows:

```
cout << fixed;
```

Run the program again. Compare the output with the prior output. What are the differences? What is the default precision?

2. Modify the code so that if an employee works exactly 55 hours, that employee receives a $100 bonus, but no longer receives overtime for the number of hours worked over 40. Make an additional modification that if the employee works 75 hours or more, that employee receives a $1000 bonus in addition to overtime. Determining how to nest the `if...else` structures properly might seem complicated. It is recommended that your code resemble the following pseudocode:

```
If worked 40 hours or less
    Calculate salary
Else
    Calculate base salary for 40 hours

    If worked 55 hours
        Add a $100 bonus to base salary
    Else
        Add overtime salary to base salary

    If worked 75 hours or more
        Add a $1000 bonus to base salary
```

Test your code based on an hourly rate of $10. How much does someone who worked exactly 55 hours earn? How much does someone who worked 56 hours earn? How much does someone who worked 75 hours earn?

Lab Exercises

Name:

Lab Exercise 3 — Square of Asterisks

Name: _____ **Date:** _____

Section: _____

This problem is intended to be solved in a closed-lab session with a teaching assistant or instructor present. The problem is divided into six parts:

1. Lab Objectives
2. Description of the Problem
3. Sample Output
4. Program Template (Fig. L 4.5)
5. Problem-Solving Tips
6. Follow-Up Question and Activity

The program template represents a complete working C++ program, with one or more key lines of code replaced with comments. Read the problem description and examine the sample output; then study the template code. Using the problem-solving tips as a guide, replace the `/* */` comments with C++ code. Compile and execute the program. Compare your output with the sample output provided. Then answer the follow-up question. The source code for the template is available from the Companion Website for *C++ How to Program, Seventh Edition* at `www.pearsonhighered.com/deitel/`.

Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 4 of *C++ How To Program, Seventh Edition*. In this lab, you will practice:

- Using counter-controlled repetition.
- Nesting `while` loops.

The follow-up question and activity also will give you practice:

- Modifying nested `while` loops.

Description of the Problem

Write a program that reads in the size of the side of a `square` and then prints a hollow square of that size out of asterisks and blanks. Your program should work for squares of all side sizes between 1 and 20. For example, if your program reads a size of 5, it should print.

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

Lab Exercises

Name:

Lab Exercise 3 — Square of Asterisks

Sample Output

```
Enter length of side: 8
********
*      *
*      *
*      *
*      *
*      *
*      *
********
```

Template

```
1  // Lab 3: Asterisks.cpp
2  // Draw a square of asterisks.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     int stars; // number of stars on a side
9     int column; // the current column of the square being printed
10    int row = 1; // the current row of the square being printed
11
12    // prompt and read the length of the side from the user
13    cout << "Enter length of side: ";
14    cin >> stars;
15
16    // valid input, if invalid, set to default
17    /* Write an if statement to test whether stars is less than 1 */
18    {
19       stars = 1;
20       cout << "Invalid Input\nUsing default value 1\n";
21    } // end if
22    /* Write an else if statement to test whether stars is greater than 20 */
23    {
24       stars = 20;
25       cout << "Invalid Input\nUsing default value 20\n";
26    } // end else if
27
28    // repeat for as many rows as the user entered
29    /* Write a while statement to count rows */
30    {
31       column = 1;
32
33       // and for as many columns as rows
34       /* Write a while statement to count columns */
35       {
36          /* Write a series of if, else if, ... statements to
37             determine whether an asterisk or a space should be displayed
38             at this position in the square */
```

Fig. L 4.3 | `Asterisks.cpp`. (Part 1 of 2.)

Lab Exercises

Name:

Lab Exercise 3 — Square of Asterisks

```
39
40          /* Increment the column counter */
41       } // end inner while
42
43       cout << endl;
44       /* Increment the row counter */
45    } // end outer while
46 } // end main
```

Fig. L 4.3 | `Asterisks.cpp`. (Part 2 of 2.)

Problem-Solving Tips

1. To display your hollow square, you will need two nested `while` loops, one of which will iterate over the rows of the square and the other of which will iterate over the columns of the square.
2. Inside the outer `while` loop, you should initialize the counter variable for the inner `while` loop, perform the inner `while` loop and then output a newline.
3. Inside the inner `while` loop, you should display one character, an asterisk if this position occurs in the first row, last row, first column or last column, or a space otherwise.
4. Be sure to follow the spacing and indentation conventions mentioned in the text. Before and after each control statement, place a line of vertical space to make the control statement stand out. Indent all the body statements of `main`, and indent all the body statements of each control statement.
5. If you have any questions as you proceed, ask your lab instructor for help.

Follow-Up Question and Activity

1. Modify your program so that the user enters two integers, each between 1 and 20, inclusive. The program will then display a rectangle of asterisks, using the first integer as the number of rows and the second integer as the number of columns.

Lab Exercises

Name:

Debugging

Name: _____ **Date:** _____

Section: _____

The program in this section does not run properly. Fix all the syntax errors so that the program will compile successfully. Once the program compiles, compare the output with the sample output, and eliminate any logic errors that may exist. The sample output demonstrates what the program's output should be once the program's code has been corrected.

Sample Output

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

Broken Code

```
1  // Debugging: debugging.cpp
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     int row = 8;
8     int side;
9
10    while ( row-- >= 0 )
11    {
12       side = 8;
13
14       if ( row / 2 == 0 )
15          cout << ' ';
16
17       while ( side++ >= 0 )
18          cout << '* ';
19
20    } // end while
21    cout << endl;
22
23    cout << endl;
24 } // end main
```

Fig. L 4.4 | `debugging.cpp`.

Postlab Activities

Coding Exercises

Name: _____ **Date:** _____

Section: _____

These coding exercises reinforce the lessons learned in the lab and provide additional programming experience outside the classroom and laboratory environment. They serve as a review after you have successfully completed the *Prelab Activities* and *Lab Exercises* successfully.

For each of the following problems, write a program or a program segment that performs the specified action.

1. Write a program that inputs an integer and uses an `if...else` statement to determine whether the integer is odd or even. If it is odd, print `"x is odd"`, and if it is even, print `"x is even"`.

2. Rewrite your solution to *Coding Exercise 1* so that it uses the conditional operator (`?:`).

Postlab Activities

Name:

Coding Exercises

3. Modify the following code to produce the output shown. Use proper indentation techniques. You must not make any changes other than inserting braces. The compiler ignores indentation in a C++ program. We eliminated the indentation from the following code to make the problem more challenging. [*Note:* It is possible that no modification is necessary.]

```
1  if ( y == 8 )
2  if ( x == 5 )
3  cout << "@@@@@" << endl;
4  else
5  cout << "#####" << endl;
6  cout << "$$$$$" << endl;
7  cout << "&&&&&" << endl;
```

Assuming x = 5 and y = 8, the following output is produced.

```
@@@@@
$$$$$
&&&&&
```

4. Insert braces and indentation in the code in *Coding Exercise 3* so that, assuming x = 5 and y = 8, the following output is produced.

```
@@@@@
```

Postlab Activities

Name:

Coding Exercises

5. Insert braces and indentation in the code in *Coding Exercise 3* so that, assuming `x = 5` and `y = 8`, the following output is produced.

```
@@@@@
&&&&&
```

6. Insert braces and indentation in the code in *Coding Exercise 3* so that, assuming `x = 5` and `y = 7`, the following output is produced. [*Hint:* The last three output statements after the `else` are all part of a block.]

```
#####
$$$$$
&&&&&
```

Postlab Activities

Name:

Programming Challenges

Name: _____ **Date:** _____

Section: _____

The *Programming Challenges* are more involved than the *Coding Exercises* and may require a significant amount of time to complete. Write a C++ program for each of the problems in this section. The answers to these problems are available from the Companion Website for *C++ How to Program, Seventh Edition* at `www.pearsonhighered.com/deitel/`. Pseudocode, hints and/or sample outputs are provided to aid you in your programming.

1. One large chemical company pays its salespeople on a commission basis. The salespeople each receive $200 per week plus 9 percent of their gross sales for that week. For example, a salesperson who sells $5000 worth of chemicals in a week receives $200 plus 9 percent of $5000, or a total of $650. Develop a C++ program that uses a `while` statement to input each salesperson's gross sales for last week and calculates and displays that salesperson's earnings. Process one salesperson's figures at a time. The output should resemble that shown below. Pseudocode is provided as a guide.

```
Enter sales in dollars (-1 to end): 5000.00
Salary is: $650.00

Enter sales in dollars (-1 to end): 6000.00
Salary is: $740.00

Enter sales in dollars (-1 to end): 7000.00
Salary is: $830.00

Enter sales in dollars (-1 to end): -1
```

Input the first salesperson's sales in dollars

While the sentinel value (–1) has not been entered for the sales
    Calculate the salesperson's wages for the week
    Print the salesperson's wages for the week
    Input the next salesperson's sales in dollars

2. A palindrome is a number or a text phrase that reads the same backwards as forwards. For example, each of the following five-digit integers is a palindrome: 12321, 55555, 45554 and 11611. Write a program that reads in a five-digit integer and determines whether it is a palindrome.

Hints:

- Use the division and modulus operators to separate the number into its individual digits.
- Store each digit in its own variable.
- Compare the first and fifth digits and the second and fourth digits; if they are equal, the number is a palindrome.

Postlab Activities

Name:

Programming Challenges

- Sample output:

```
Enter a 5-digit number: 12321
12321 is a palindrome!!!
```

```
Enter a 5-digit number: 12345
12345 is not a palindrome.
```

3. A company wants to transmit data over the telephone, but is concerned that its phones could be tapped. All of the data are transmitted as four-digit integers. The company has asked you to write a program that encrypts the data so that it can be transmitted more securely. Your program should read a four-digit integer and encrypt it as follows: Replace each digit by *(the sum of that digit plus 7) modulus 10*. Then, swap the first digit with the third, swap the second digit with the fourth and print the encrypted integer.

Hint:

- Sample output:

```
Enter a four-digit number: 1234
Encrypted number is 189
```

4. Write a program that inputs an encrypted four-digit integer produced by the program from *Programming Challenge 4* and decrypts it to form the original number.

Hints:

- Sample output:

```
Enter an encrypted number: 189
Decrypted number is 1234
```