Structures can be passed as parameters, which can reduce the number of parameters that must be passed. For example, passing a variable of the line structure type can take the place of passing four coordinates (two x and two y values). In the exercise that follows, you will run a program that passes a structure variable to calculate the slope of a line.

EXERCISE 10-9 SLOPE OF A LINE

1. Open *LNSLOPE.CPP*. A program appears that uses the structures above to calculate the slope of a line.
2. Compile and run the program. Provide the points (1,2) and (2,5) as input.
3. Add the following statements after the declaration of the variable **m.**

```
line horizontal_line;

horizontal_line.p1.x = 1;
horizontal_line.p1.y = 2;
horizontal_line.p2.x = 5;
horizontal_line.p2.y = 2;
m = slope(horizontal_line);
cout << "The slope of every horizontal line is " << m << endl;
```

4. Compile and run the program again to see the result of the new lines.
5. Save the source code and close.

SECTION 10.4 QUESTIONS

1. What is the purpose of a structure?
2. What is the term for the variables in a structure?
3. Write a declaration for a structure named **house** that is to be used to store records in a database of house descriptions for a real estate agent. The fields in the record should include address, square footage of the house, number of bedrooms, number of bathrooms, number of cars that can fit in the garage, and the listed price of the house.
4. Write a statement that declares a structure variable named **featured_home** using the structure declared in question 3.
5. Write a series of statements that initialize the structure variable declared in question 4. Do your best to initialize the structure variable with realistic values.
6. Write a statement that prints the information in the **featured_home** structure variable in the format of the example below, where 3-2-2 is the number of bedrooms, baths, and garage stalls respectively, and 1800 is the square footage.

   3918 Shonle Road 3-2-2 1800 $64,000

PROBLEM 10.4.1

1. Open *STRUCT.CPP* (the program you saved in Exercise 10-7).
2. Add code to the end of the program that asks the user for the anticipated number of sales during the day the item is on special and the anticipated sales for the product if it were not on special.
3. Check the value entered against the quantity on hand to make sure that the anticipated orders can be filled. Warn the user if the quantity on hand is less than the anticipated sales.
4. Calculate the amount of income the product is anticipated to generate if the product is put on special and the amount of income the product is anticipated to generate if the product is not put on special.
5. Save the new source code as *STRUCT2.CPP*. Compile and run the program. After testing the program, close the source code file.

PROBLEM 10.4.2

Write a program that uses the point and line structures from Exercise 10-9 to calculate the midpoint of a given line. Have the program ask the user for the points that define the line, then use the formula below to calculate the midpoint of the line and output the coordinates of the midpoint. Save the program as *MIDPOINT.CPP*.

$$\left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}\right)$$

KEY TERMS

address-of operator	pointer
data structure	pointer constant
dereferencing operator	pointer variable
dot operator	primitive data structure
enum	simple data structure
fields	records
members	structures
nested structure	subscript notation

SUMMARY

- Pointers are variables and constants that hold memory addresses.
- The dereferencing operator (*) is used to declare pointers and to access the value in the variable to which the pointer points.