```
#include<iostream.h>

void print_value(int j);  // function prototype

int main()
 {
  int i = 2;
  cout << "The value before the function is " << i << endl;
  print_value(i);
  cout << "The value after the function exits is " << i << endl;
  return 0;
 }

void print_value(int j)
 {
  cout << "The value passed to the function is " << j << endl;
  j = j * 2; // the value in the variable i is doubled
  cout << "The value at the end of the function is " << j << endl;
 }
```

FIGURE 9-6
This program uses passing by value to pass an integer to the `print_value` function. The argument `i` which appears in the parentheses in the function call is passed to a parameter named `j` in the receiving function.

EXERCISE 9-4 PASSING BY VALUE

1. Enter the program shown in Figure 9-6. Save the source code as *PASSVAL.CPP*.
2. Compile and run the program to see that the value passed to the `print_value` function is not passed back to the main function.
3. Leave the source code file open for the next exercise.

PASSING BY REFERENCE

Functions that pass variables by reference will pass any changes you make to the variables back to the calling function. For example, suppose you need a function that gets input from the user. The function below uses ***passing by reference*** to get two values from the user and pass them back through parentheses.

```
void get_values(float &income, float &expense)
 {
  cout << "Enter this month's income amount: $";
  cin >> income;
  cout << "Enter this month's expense amount: $";
  cin >> expense;
 }
```

To pass a variable by reference, simply precede the variable name with an ampersand (&) in the function definition. But even though it is easy to pass by reference, you should do so sparingly. You should write functions that pass variables by value whenever possible. The reason is because passing variables by value is safer. When you pass a variable by value, you know it cannot be changed by the function you call. When you pass a variable by reference, a programming error in the function could cause a problem throughout the program.

As a general rule, you should use passing by reference only when data needs to be passed back to the calling function. In the preceding example, the data entered by the user must be passed back to the calling function.

The program you ran in the last exercise passed a variable by value. Let's modify the program to make it pass the variable by reference.

EXERCISE 9-5 PASSING BY REFERENCE

1. Add an ampersand (&) before the identifier `j` in both the prototype and the function declaration. Save the source code as *PASSREF.CPP*.
2. Compile and run the program again to see the difference passing by reference makes.
3. Close the source code file.

PASSING BY ADDRESS

When you pass an array (such as a character array), the syntax looks just like passing by value. However, what C++ passes to the function is the address of the location in memory where the first element of the array resides. So even though it may seem like you are passing by value, you are actually using a technique called ***passing by address.*** Because of the way arrays are passed, any changes made to an array in a function remain when the function is complete.

EXERCISE 9-6 PASSING ARRAYS

1. Open *PASSARRY.CPP*.
2. Study the source code to see how the array is passed.
3. Compile, link, and run the program.
4. Close the source code file.

RETURNING VALUES USING RETURN

As you learned earlier in this chapter, unless a function is declared with the keyword `void`, the function will return a value. In the case of the main function, it returns a value to the operating system. Other functions, however, return a value to the calling function. The value to be returned is specified using the `return` statement.

The function below is an example of a function that returns a value of type `float`. The temperature in Celsius is passed into the function by value and the temperature in Fahrenheit is returned using the `return` statement.

```
float celsius_to_fahrenheit(float celsius)
 {
  float fahr;  // local variable for calculation
  fahr = celsius * (9.0/5.0) + 32.0;
  return(fahr);
 }
```