FUNCTIONS AND PROGRAM FLOW

In the example above, the main function, called the **`print_title`** function is executed, then the remainder of the program is executed. When a function is called, the computer executes the statements in the function beginning with the first statement. When the end of the function is reached, program execution resumes with the statement that follows the call to the function. Figure 9-3 shows the sequence of execution in a simple three-function example.

The program begins with the first statement of the main function (1), which is a call to the **`print_title`** function. The flow of logic goes to the **`print_title`** function which includes two statements (2 and 3) and they are executed next. When the last statement (3) in the **`print_title`** function is executed, the flow of logic returns to the statement (4) that follows the previous function call in **`main()`**. The statement in the **`print_goodbye`** function (5) is then executed. The flow of logic then returns to **`main()`** where the program ends (6).

FUNCTION PROTOTYPES

There is one more thing you have to do to make your own functions work. At the top of your program, you must tell the compiler that your function exists. You do this by creating a *prototype.* Basically, a prototype defines the function for the compiler. Figure 9-4 shows the functions from Figure 9-3 assembled into a working program.

```
                    int main()
Program Begins ---->  {
               (1)    print_title();         // call to print_title

                      // insert the rest of the program here

               (4)    print_goodbye();

                      return 0;
Program Ends ----> (6) } // end of main function

                    // Function to print program title to screen.
                    void print_title()
                      {
               (2)     cout << "Tennis Tournament Scheduler Program\n";
               (3)     cout << "By Jennifer Baker\n";
                      }

                    // Function to print closing message to screen.
                    void print_goodbye()
                      {
               (5)     cout << "Thank you for using the Tennis Tournament Scheduler.\n";
                      }
```

FIGURE 9-3
The numbers next to the statements show the order of execution.

```
#include<iostream.h>

void print_title();    // prototype for print_title function
void print_goodbye();  // prototype for print_goodbye function


int main()
 {
  print_title();         // call to print_title

  // insert the rest of the program here

  print_goodbye();
  return 0;

 } // end of main function


// Function to print program title to screen.
void print_title()
 {
  cout << "Tennis Tournament Scheduler Program\n";
  cout << "By Jennifer Baker\n";
 }

// Function to print closing message to screen.
void print_goodbye()
 {
  cout << "Thank you for using the Tennis Tournament Scheduler.\n";
 }
```

FIGURE 9-4
Functions you create must be prototyped at the beginning of the program.

PITFALLS

Notice that a function's prototype is just like the first line of the function, except for the semicolon. An error will result if you forget to include the semicolon at the end of a prototype.

On the Net

C++ programs begin with a main function. The order that other functions are placed in the program is not important because the function prototype has defined the function for the compiler. Therefore, when a call is made to one of your functions, the compiler knows that the function being called exists, even if it exists below the place where it is called.

Some languages (such as Pascal) end with the main part of the program and place the other functions above the main function (called a procedure in Pascal). In Pascal, you must include a special declaration called a forward declaration if you plan to call a function from code which is above the function. The concept is similar to the function prototypes used in C++.

For more information about the order of functions and the effects of the order of functions, see http://www.ProgramCPP.com. See topic 9.1.2.

EXERCISE 9-2 MAKING A FUNCTION

1. Open *1STFUNCT.CPP*. The program from Figure 9-4 appears.
2. Compile, link, and run the program.
3. When you have run the program successfully, close the source code file.