creating and disposing of objects belonging to a class. Finally, you meet the this pointer, an important component of some class programming. The following chapters extend the discussion to operator overloading (another variety of polymorphism) and inheritance, the basis for reusing code.

**Procedural and Object-Oriented Programming**

Although we have occasionally explored the OOP perspective on programming, we’ve usually stuck pretty close to the standard procedural approach of languages such as C, Pascal, and BASIC. Let’s look at an example that clarifies how the OOP outlook differs from that of procedural programming.

As the newest member of the Genre Giants softball team, you’ve been asked to keep the team statistics. Naturally, you turn to your computer for help. If you were a procedural programmer, you might think along these lines:

Let’s see, I want to enter the name, times at bat, number of hits, batting averages (for those who don’t follow baseball or softball, the batting average is the number of hits divided by the player’s official number of times at bat; an at bat terminates when a player gets on base or makes an out, but certain events, such as getting a walk, don’t count as official times at bat), and all those other great basic statistics for each player. Wait, the computer is supposed to make life easier for me, so let’s have it figure out some of that stuff, such as the batting average. Also, I want the program to report the results. How should I organize this? I guess I should do things right and use functions. Yeah, I’ll make main() call a function to get the input, call a function to make the calculations, and then call a function to report the results. Hmmm, what happens when I get data from the next game? I don’t want to start from scratch again. Okay, I can add a function to update the statistics. Golly, maybe I’ll need a menu in main() to select between entering, calculating, updating, and showing the data. Hmmm—how am I going to represent the data? I could use an array of strings to hold the players’ names, another array to hold the at bats for each player, yet another array to hold the hits, and so on. No, that’s dumb. I can design a structure to hold all the information for a single player and then use an array of those structures to represent the whole team.

In short, you first concentrate on the procedures you will follow and then think about how to