Because the function shouldn't modify the structure, use the **const** modifier.

- Because the formal parameter is a pointer instead of a structure, use the indirect membership operator (\texttt{->}) rather than the membership operator (dot).

After the changes, the function looks like this:

```cpp
// show polar coordinates, converting angle to degrees
void show_polar (const polar * pda)
{
    const double Rad_to_deg = 57.29577951;

    cout << "distance = " << pda->distance;
    cout <<", angle = " << pda->angle * Rad_to_deg;
    cout <<" degrees\n";
}
```

Next, let's alter \texttt{rect_to_polar}. This is more involved because the original \texttt{rect_to_polar} function returns a structure. To take full advantage of pointer efficiency, you should use a pointer instead of a return value. The way to do this is to pass two pointers to the function. The first points to the structure to be converted, and the second points to the structure that's to hold the conversion. Instead of \textit{returning} a new structure, the function \textit{modifies} an existing structure in the calling function. Hence, although the first argument is \texttt{const} pointer, the second is not \texttt{const}. Otherwise, apply the same principles used to convert \texttt{show_polar}() to pointer arguments. Listing 7.13 shows the reworked program.

**Listing 7.13 strctptr.cpp**

```cpp
// strctptr.cpp -- functions with pointer to structure arguments
#include <iostream>
#include <cmath>
using namespace std;
// structure templates
struct polar
{
    double distance; // distance from origin
    double angle;   // direction from origin
```