Given that the parameter \texttt{ar2} is a pointer to an array, how do we use it in the function definition? The simplest way is to use \texttt{ar2} as if it were the name of a two-dimensional array. Here's a possible function definition:

\begin{verbatim}
int sum(int ar2[][4], int size)
{
    int total = 0;
    for (int row = 0; row < size; row++)
        for (col = 0; col < 4; col++)
            total += ar2[row][col];
    return total;
}
\end{verbatim}

Again, note that the number of rows is whatever is passed to the \texttt{size} parameter, but the number of columns is fixed at 4, both in the parameter declaration for \texttt{ar2} and in the inner \texttt{for} loop.

Here's why you can use array notation. Because \texttt{ar2} points to the first element (element 0) of an array whose elements are array-of-4-int, the expression \texttt{ar2 + row} points to element number \texttt{row}. Therefore \texttt{ar2[row]} is element number \texttt{row}. That element is itself an array-of-4-int, so \texttt{ar2[row]} is the name of that array-of-4-int. Applying a subscript to an array name gives an array element, so \texttt{ar2[row][col]} is an element of the array-of-4-int, hence is a single \texttt{int} value. The pointer \texttt{ar2} has to be dereferenced twice to get to the data. The simplest way is to use brackets twice.

Incidentally, the code for \texttt{sum()} doesn't use \texttt{const} in declaring the parameter \texttt{ar2} because that technique is for pointers to fundamental types, whereas \texttt{ar2} is a pointer to a pointer.

\section*{Functions and C-Style Strings}

A C-style string, you recall, consists of a series of characters terminated by the null character. Much of what you've learned about designing array functions applies to string functions, too. But there are a few special twists to strings that we unravel now.

Suppose you want to pass a string as an argument to a function. You have three choices for representing a string: