C99 and `short` arguments to be converted to `int`. (In C99, the integer promotions are performed.)

⚠ Relying on the default argument promotions is dangerous. Consider the following program:

```
#include <stdio.h>

int main(void)
{
  double x = 3.0;
  printf("Square: %d\n", square(x));

  return 0;
}

int square(int n)
{
  return n * n;
}
```

At the time `square` is called, the compiler hasn't seen a prototype yet, so it doesn't know that `square` expects an argument of type `int`. Instead, the compiler performs the default argument promotions on `x`, with no effect. Since it's expecting an argument of type `int` but has been given a `double` value instead, the effect of calling `square` is undefined. The problem can be fixed by casting `square`'s argument to the proper type:

```
printf("Square: %d\n", square((int) x));
```

C99 Of course, a much better solution is to provide a prototype for `square` before calling it. In C99, calling `square` without first providing a declaration or definition of the function is an error.

Array Arguments

Q&A Arrays are often used as arguments. When a function parameter is a one-dimensional array, the length of the array can be (and is normally) left unspecified:

```
int f(int a[])   /* no length specified */
{
  ...
}
```

The argument can be any one-dimensional array whose elements are of the proper type. There's just one problem: how will `f` know how long the array is? Unfortunately, C doesn't provide any easy way for a function to determine the length of an array passed to it. Instead, we'll have to supply the length—if the function needs it—as an additional argument.