If m ≠ n, any subsequent use of p will cause undefined behavior.

Variably modified types are subject to certain restrictions, just as variable-length arrays are. The most important restriction is that the declaration of a variably modified type must be inside the body of a function or in a function prototype.

Pointer arithmetic works with VLAs just as it does for ordinary arrays. Returning to the example of Section 12.4 that clears a single column of a two-dimensional array a, let's declare a as a VLA this time:

```
int a[m][n];
```

A pointer capable of pointing to a row of a would be declared as follows:

```
int (*p)[n];
```

The loop that clears column i is almost identical to the one we used in Section 12.4:

```
for (p = a; p < a + m; p++)
  (*p)[i] = 0;
```

Q & A

Q: I don't understand pointer arithmetic. If a pointer is an address, does that mean that an expression like p + j adds j to the address stored in p? [p. 258]

A: No. Integers used in pointer arithmetic are scaled depending on the type of the pointer. If p is of type `int *`, for example, then p + j typically adds 4 × j to p, assuming that `int` values are stored using 4 bytes. But if p has type `double *`, then p + j will probably add 8 × j to p, since `double` values are usually 8 bytes long.

Q: When writing a loop to process an array, is it better to use array subscripting or pointer arithmetic? [p. 261]

A: There's no easy answer to this question, since it depends on the machine you're using and the compiler itself. In the early days of C on the PDP-11, pointer arithmetic yielded a faster program. On today's machines, using today's compilers, array subscripting is often just as good, and sometimes even better. The bottom line: Learn both ways and then use whichever is more natural for the kind of program you're writing.

***Q: I read somewhere that i[a] is the same as a[i]. Is this true?**

A: Yes, it is, oddly enough. The compiler treats i[a] as *(i + a), which is the same as *(a + i). (Pointer addition, like ordinary addition, is commutative.) But *(a + i) is equivalent to a[i]. Q.E.D. But please don't use i[a] in programs unless you're planning to enter the next Obfuscated C contest.