between 0 and 100. The following call asks `fread` to read one block of 100 bytes:

```
fread(a, 100, 1, fp)
```

`fread`'s return value in this case will be either 0 or 1.

`fwrite` is convenient for a program that needs to store data in a file before terminating. Later, the program (or another program, for that matter) can use `fread` to read the data back into memory. Despite appearances, the data doesn't need to be in array form; `fread` and `fwrite` work just as well with variables of all kinds. Structures, in particular, can be read by `fread` or written by `fwrite`. To write a structure variable `s` to a file, for instance, we could use the following call of `fwrite`:

```
fwrite(&s, sizeof(s), 1, fp);
```

Be careful when using `fwrite` to write out structures that contain pointer values; these values aren't guaranteed to be valid when read back in.

22.7 File Positioning

```
int fgetpos(FILE * restrict stream,
            fpos_t * restrict pos);
int fseek(FILE *stream, long int offset, int whence);
int fsetpos(FILE *stream, const fpos_t *pos);
long int ftell(FILE *stream);
void rewind(FILE *stream);
```

Every stream has an associated *file position*. When a file is opened, the file position is set at the beginning of the file. (If the file is opened in "append" mode, however, the initial file position may be at the beginning or end of the file, depending on the implementation.) Then, when a read or write operation is performed, the file position advances automatically, allowing us to move through the file in a sequential manner.

Although sequential access is fine for many applications, some programs need the ability to jump around within a file, accessing some data here and other data there. If a file contains a series of records, for example, we might want to jump directly to a particular record and read it or update it. `<stdio.h>` supports this form of access by providing five functions that allow a program to determine the current file position or to change it.

fseek

The `fseek` function changes the file position associated with the first argument (a file pointer). The third argument specifies whether the new position is to