```
int i;                  /* Declaration 1 */

void f(int i)           /* Declaration 2 */
{
  i = 1;
}

void g(void)
{
  int i = 2;            /* Declaration 3 */

  if (i > 0) {
    int i;              /* Declaration 4 */

    i = 3;
  }

  i = 4;
}

void h(void)
{
  i = 5;
}
```

- In Declaration 2, `i` is a parameter with block scope.
- In Declaration 3, `i` is an automatic variable with block scope.
- In Declaration 4, `i` is also automatic and has block scope.

`i` is used five times. C's scope rules allow us to determine the meaning of `i` in each case:

- The `i = 1` assignment refers to the parameter in Declaration 2, not the variable in Declaration 1, since Declaration 2 hides Declaration 1.
- The `i > 0` test refers to the variable in Declaration 3, since Declaration 3 hides Declaration 1 and Declaration 2 is out of scope.
- The `i = 3` assignment refers to the variable in Declaration 4, which hides Declaration 3.
- The `i = 4` assignment refers to the variable in Declaration 3. It can't refer to Declaration 4, which is out of scope.
- The `i = 5` assignment refers to the variable in Declaration 1.

10.5 Organizing a C Program

Now that we've seen the major elements that make up a C program, it's time to develop a strategy for their arrangement. For now, we'll assume that a program