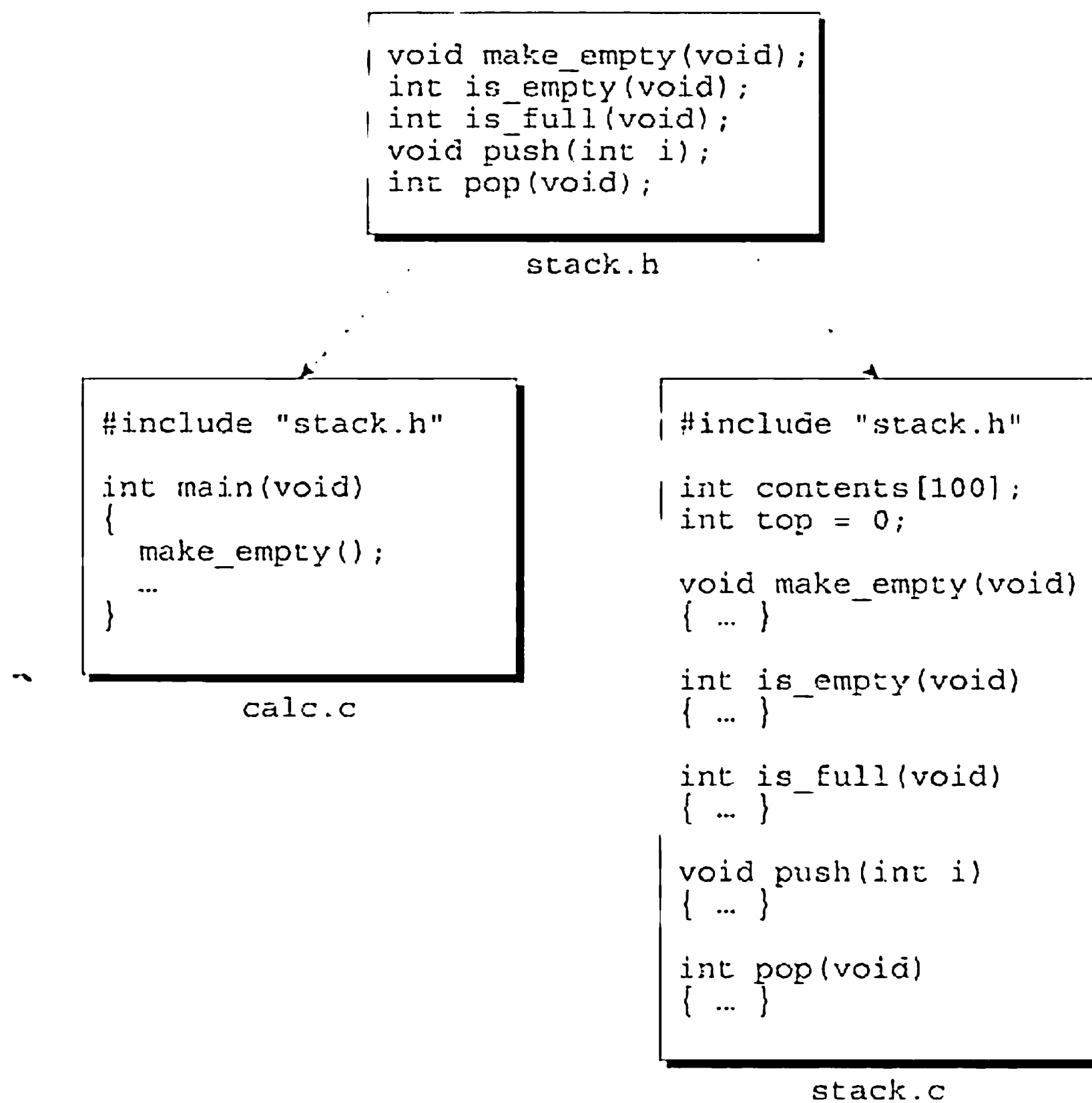


```
int is_full(void);
void push(int i);
int pop(void);
```

(To avoid complicating the example, `is_empty` and `is_full` will return `int` values instead of Boolean values.) We'll include `stack.h` in `calc.c` to allow the compiler to check any calls of stack functions that appear in the latter file. We'll also include `stack.h` in `stack.c` so the compiler can verify that the prototypes in `stack.h` match the definitions in `stack.c`. The following figure shows `stack.h`, `stack.c`, and `calc.c`:



Sharing Variable Declarations

external variables ► 10.2

External variables can be shared among files in much the same way functions are. To share a function, we put its *definition* in one source file, then put *declarations* in other files that need to call the function. Sharing an external variable is done in much the same way.

Up to this point, we haven't needed to distinguish between a variable's declaration and its definition. To declare a variable `i`, we've written

```
int i;          /* declares i and defines it as well */
```