```
  case 25: printf("Quarter");
           break;
}
```

If each case consists of a single action (a call of `printf`, in this example), the `break` statement could even go on the same line as the action:

```
switch (coin) {
  case 1:  printf("Cent"); break;
  case 5:  printf("Nickel"); break;
  case 10: printf("Dime"); break;
  case 25: printf("Quarter"); break;
}
```

The other method is to put the statements *under* the case label, indenting the statements to make the case label stand out:

```
switch (coin) {
  case 1:
    printf("Cent");
    break;
  case 5:
    printf("Nickel");
    break;
  case 10:
    printf("Dime");
    break;
  case 25:
    printf("Quarter");
    break;
}
```

In one variation of this scheme, each case label is aligned under the word `switch`.

The first method is fine when the statements in each case are short and there are relatively few of them. The second method is better for large `switch` statements in which the statements in each case are complex and/or numerous.

Exercises

Section 5.1

1. The following program fragments illustrate the relational and equality operators. Show the output produced by each, assuming that `i`, `j`, and `k` are `int` variables.

(a)
```
i = 2; j = 3;
k = i * j == 6;
printf("%d", k);
```
(b)
```
i = 5; j = 10; k = 1;
printf("%d", k > i < j);
```
(c)
```
i = 3; j = 2; k = 1;
printf("%d", i < j == j < k);
```
(d)
```
i = 3; j = 4; k = 5;
printf("%d", i % j + i < k);
```