Using `switch` involves creating a complex structure that includes the `case`, `break`, and `default` keywords. Here's how it may look:

```
switch(choice)
{
      case item1:
                        statement(s);
                        break;
      case item2:
      case item3:
                        statement(s);
                        break;
      default:
                        statement(s);
}
```

`choice` must be a variable. It can be a key typed at the keyboard, a value returned from the mouse or joystick, or some other interesting number or character the program has to evaluate.

After the `case` keyword come various *items;* `item1`, `item2`, `item3`, and so on are the various items that *`choice`* can be. Each one is a constant, either a character or a value; *they cannot be variables*. The `case` line ends in a colon, not in a semicolon.

Belonging to each `case item` are one or more *statements*. The program executes these *statements* when `item` matches the *choice* that `switch` is making — like an `if` statement match. The *statements* are *not* enclosed in curly braces. The *statements* are also optional. (More on that in a second.)

The last statement in a group of `case` statements is typically a `break` command. Without the `break` there, the program keeps working its way through the next `case` statement.

The last item in the `switch` structure is `default`. It contains the statements to be executed when no match occurs — like the final `else` in an `if-else` structure. The `default` statements are executed no matter what (unless you `break` out of the structure earlier).

The most important thing to remember about `switch-case` is that the program always walks through the entire thing unless you put a `break` in there when you want it to stop. For example, consider this program snippet:

```
switch(key)
{
    case 'A':
        printf("The A key.\n");
        break;
```