Switch Cases

By: **Annamalai A**

The switch statement in C is a multi-way selection statement that allows a variable to be tested against a list of values, called cases. These list of values are constant, meaning you cannot compare it with another variable inside.

Switch cases somewhat mimic `if...else` because the variable in the condition only matches one of the values in the list and the statements are executed accordingly. However, you can only compare a variable to a constant or literal and it only does equal to comparison, unlike `if...else` which can do other comparison operations as well.

The structure for switch case is as follows:

```
switch (/* expression */) {
    case const1:
        // Set of statements
        break;
    case const2:
        // Set of statements
        break;
    ...
    case constn:
        // Set of statements
        break;
    default: // Optional
        // Set of statements
}
```

The `case const1` contains the value that the expression is compared with. If the comparison holds true, the set of statements under it gets executed.

The `break` statement is important here. It comes out of the switch after the statements are executed. Not including this statement causes the compiler to behave wierdly.

The `default` consists of the set of statements that are executed if none of the cases match. i.e. All comparisons hold false for the given expression. It is optional and is not required for every switch case, however it is useful.

So why is the `break` statement important here? What `switch` usually does is it compares the expression with each of the constants for equality. When the expression matches with a constant, the set of statements are executed, and the statements of the following cases are also executed, including any statement in default. Hence, the `break` statement is required to *break* out of the switch statement.

Common Errors

Switch cases are way more efficient that `if...else` when there are multiple conditions, but it must be used carefully. Here are some common errors made using this:

- **Wrong data type**: `switch` compares with only integer types such as `int` and `char` datatypes.
- **Fall-through**: As discussed above, missing a `break` statement will cause the compiler to execute statements from subsequent cases as well.
- **Placement of jump statements**: Although `break` is required, placing it in between other expressions causes it to break out of the switch before executing every statement.