C++: switch Statement — Complete Guide

1. Basic switch

Multi-way branching based on an integral or enum value.

```
switch (code) {
    case 0:
        handleZero();
        break;
    case 1:
        handleOne();
        break;
    default:
        handleDefault();
}
```

2. switch with enum / enum class

Use enums for clear case labels; with enum class you need to qualify labels.

```
enum class Color { Red, Green, Blue };
Color c = Color::Red;
switch (c) {
    case Color::Red:
        // ...
        break;
    case Color::Green:
        // ...
        break;
    default:
        break;
}
```

3. Fall-through and [[fallthrough]] (C++17)

Cases fall through by default. Use break to prevent it. Use [[fallthrough]] to document intentional fall-through.

```
switch(n) {
    case 1:
        doA();
        [[fallthrough]]; // intentionally continue
    case 2:
        doB();
        break;
}
```

4. Multiple case labels & ranges

Group cases that share logic by stacking labels. C++ has no native range case; use if inside case or computed tricks.

```
switch(ch) {
    case 'a':
    case 'e':
    case 'i':
    case 'o':
    case 'u':
        vowel();
        break;
    default:
        consonant();
}
```

5. switch with initializer (C++17)

You can initialize a value inside switch for scoping.

```
switch (int x = compute(); x) {
    case 0: break;
    default: break;
}
```

6. Limitations and best practices

- switch only works with integral/enums/char; not with std::string. - Prefer switch for dense integral cases; compilers may use jump tables. - For string-like branching, use unordered_map. - Always handle default case unless exhaustive enum handled.

7. Example — token handling via switch

A realistic example handling simple byte-code tokens.

```
enum Token { ADD=1, SUB=2, MUL=3 };
void exec(int token) {
    switch(token) {
        case ADD: doAdd(); break;
        case SUB: doSub(); break;
        case MUL: doMul(); break;
        default: throw std::runtime_error("Unknown token");
    }
}
```