Learn C++ : Conditionals & Logic

codecademy

`if` Statement

In C++, an `if` statement is used to test an expression for truth.

- If the condition evaluates to `true`, then the code within the block is executed; otherwise, it will be skipped.

```
if (a == 10) {
  // code goes here
}
```

`else` Clause

In C++, an `else` clause can be added to an `if` statement.

- If the condition evaluates to `true`, code in the `if` part is executed.
- If the condition evaluates to `false`, code in the `else` part is executed.

```
if (year == 1991) {
  // executed if it is true
}
else {
  // executed if it is false
}
```

`switch` Statement

In C++, a `switch` statement is an alternative to the `if`/`else if`/`else` statement.

`switch` statement contains an expression and then various cases. The value of the expression is compared with the value of each `case`; if there is a match, the code within starts to execute.

The `break` keyword tells the computer to exit the block and not check any other cases.

`default` is executed when no case matches. It functions as the `else` clause of a `switch` statement.

```
switch (grade) {
  case 9:
    std::cout << "Freshman\n";
    break;
  case 10:
    std::cout << "Sophomore\n";
    break;
  case 11:
    std::cout << "Junior\n";
    break;
  case 12:
    std::cout << "Senior\n";
    break;
  default:
    std::cout << "Invalid\n";
    break;
}
```

Relational Operators

In C++, *relational operators* are used to compare two values:

- `==` equal to
- `!=` not equal to
- `>` greater than
- `<` less than
- `>=` greater than or equal to
- `<=` less than or equal to

```
if (a > 10) {
   // ☝ means greater than
}
```

`else if` Statement

In C++, one or more `else if` statements can be added in between the `if` and `else`.

- If the `if` condition evaluates to `true`, code in the `if` part is executed.
- If the `if` condition evaluates to `false` and the `else if` condition evaluates to `true`, code in the `else if` part is executed.
- If none of the conditions evaluates to true, code in the `else` part is executed.

```
if (apple > 8) {
  // some code here
}
else if (apple > 6) {
  // some code here
}
else {
  // some code here
}
```

Logical Operators

In C++, *logical operators* can be used to combine two different conditions.

- `&&` the logical operator (`and`)
- `||` the logical operator (`or`)
- `!` the logical operator (`not`)

The `&&` requires both conditions to be `true`. The `||` requires either of the condition to be `true`. The `!` negates the result.

```
if (coffee > 0 && donut > 1) {
  // executed if both are true
}

if (coffee > 0 || donut > 1) {
  // executed if either is true
}

if (!tired) {
  // excuted if tired is false
}
```