Learning Objectives: String Comparison

- **Compare strings with `==` and `!=`**
- **Compare strings with `compare()`**

== & !=

Comparing with ==

The `==` operator can be used with strings just like it is with numbers or boolean values. **Note** that without the `boolalpha` flag, the system will return `1` if `true` and `0` if `false`. `1` represents string equality and `0` represents inequality.

```
string string1 = "It's Friday!";
string string2 = "It's Friday!";

cout << (string1 == string2);
```

→ 1

challenge

What happens if you:

- Change the value of `string1` to `"it's friday!"`? → 0
- Change the value of `string2` to `"it\'s friday!"`? → 1
- Change the cout statement to `cout << boolalpha << (string1 == string2);`? → true

Comparing with !=

You can also test for string inequality with the `!=` operator.

```
string string1 = "It's Friday!";
string string2 = "It's Monday.";

cout << (string1 != string2);
```

→ 1

challenge

What happens if you:

- Change the value of `string2` to `"It's Friday"`? → 1
- Change the value of `string2` to `"It's Friday!"`? → 0
- Change the cout statement to `cout << boolalpha << (string1 != string2);`? → False

Compare

Lexicographical Order

In C++, strings can be compared lexicographically, meaning they can be compared according to how they will appear in the dictionary. You can use the `compare()` method to determine which of two strings comes first. A return value of a **negative** integer means the first string comes first, a return value of a **positive** integer means the second string comes first, and a return value of `0` means the strings are equal and neither comes first.

```
string string1 = "apple";
string string2 = "cat";

if (string1.compare(string2) < 0) {
  cout << "string1 comes first" << endl;
}
else if (string1.compare(string2) > 0) {
  cout << "string2 comes first" << endl;
}
else {
  cout << "the strings are equal" << endl;
}
```

→ string1 comes first

challenge

What happens if you:

- Change `string2` to `"apple"`? → the strings are equal
- Change `string2` to `"10"`? string2 comes first
- Change `string1` to `"2"` in your current code? string2 comes first

Why Does “10” Come Before “2”?

When C++ compares strings lexicographically, it compares each character of the strings one by one from left to right. Since the first character in `10` is `1`, and `1` comes before `2`, `10` is considered to come before `2` even though numerically `2` is supposed to come first.

```
string string1 = "123";
string string2 = "9";

if (string1.compare(string2) < 0) {
  cout << "string1 comes first" << endl;
}
else if (string1.compare(string2) > 0) {
  cout << "string2 comes first" << endl;
}
else {
  cout << "the strings are equal" << endl;
}
```

→ string1 comes first

challenge

What happens if you:

- Change `string1` to `"apple"`? → string2 comes first
- Change `string2` to `"Apple"` in your current code? → string2 comes first
- Change `string1` to an empty string `""` in your current code? → string1 comes first

Letters vs. Numbers vs. Empty Strings

Lexicographically speaking, empty strings always come first, followed by numbers, then uppercase letters, and finally lowercase letters.