Comparison and Concatenation

Can we merge and compare strings like we did with ranges? This lesson shows us how.

WE'LL COVER THE FOLLOWING

Comparison

Strings support the well-known comparison operators `==, !=, <, >, >=`. The comparison of two strings takes place on their elements.

```
#include <iostream>
#include <string>

int main(){

  std::cout << std::boolalpha << std::endl;

  std::string first{"aaa"};
  std::string second{"aaaa"};

  std::cout << "first <  first :" << (first < first) << std::endl;
  std::cout << "first <=  first :" << (first <= first) << std::endl;
  std::cout << "first <  second :" << (first < second) << std::endl;

  std::cout << std::endl;

  std::string one{"1"};
  std::string oneOneOne= one+ std::string("1") +"1";

  std::cout << "1 + 1 + 1: " << oneOneOne << std::endl;

  std::cout << std::endl;

}
```

String comparison

String concatenation

The + operator is overloaded for strings, so we can *add* strings.

> ⚠ **The + operator is only overloaded for C++ strings**
> The C++ type system permits concatenation of C++ and C strings into C++ strings, but not concatenation of C++ and C strings into C strings. The reason is that the + operator is overloaded for C++ strings. Therefore only the second line is valid C++, because the C string is implicitly converted to a C++ string:

```
//...
#include <string>
//...
std::string wrong= "1" + "1"; // ERROR
std::string right= std::string("1") + "1"; // 11
```

In the next lesson, we'll learn how we can access the elements of a string.