Strings in Constant Expressions

Now, we'll learn why string_view works with constant expressions.

The interesting property of `string_view` is that all of the methods are marked as `constexpr` (except for `copy`, `operator <<` and `std::hash` functions specialised for string views). With this capability, you can work on strings at compile time.

For example:

```
#include <iostream>
#include <string_view>
using namespace std;
using namespace std::literals;

int main() {
  constexpr auto strv = "Hello Programming World"sv;
  constexpr auto strvCut = strv.substr("Hello "sv.size());

  static_assert(strvCut == "Programming World"sv);
  cout << strvCut.size();
}
```

A similar version of such code, but with `std::string` would generate much more code. Since the example uses long strings, then Small String Optimisation is not possible, and then the compiler must generate code for `new/delete` to manage the memory of the strings.

We're almost at the end of our discussion on `string_view`. We will now point out the similarities between this class and the boost library functions.