Element Access

Accessing a character in a string is very easy and similar to element access in arrays.

Access to the elements of a string `str` is very convenient because the strings support random access iterators. We can access with `str.front()` the first character and with `str.back()` the last character of the string. With `str[n]` and `str.at(n)` we get the n-th element by index.

The following table provides an overview.

Access the elements of the string

Methods	Example
`str.front()`	Returns the first character of `str`.
`str.back()`	Returns the last character of `str`.
`str[n]`	Returns the n-th character of `str`. The string boundaries **will not be checked**.
`str.at(n)`	Returns the n-th character of `str`. The string boundaries **will be checked**. If the boundaries are violated a `std::out_of_range` exception is thrown.

```
#include <iostream>
#include <stdexcept>
#include <string>
#include <vector>

int main(){

  std::cout << std::endl;

  std::string str= {"0123456789"};
  std::cout << "str.front(): " << str.front() << std::endl;
  std::cout << "str.back(): " << str.back() << std::endl;

  std::cout << std::endl;

  for (int i=0; i <= 10; ++i){
    std::cout << "str[" << i  << "]: " << str[i] << std::endl;
  }

  std::cout << std::endl;

  try{
    str.at(10);
  }
  catch (const std::out_of_range& e){
    std::cerr << "Exception: " <<  e.what() << std::endl;
  }

  std::cout << std::endl;

  std::cout << "*(&str[0]+5): " << *(&str[0]+5) << std::endl;
  std::cout << "*(&str[5]): " << *(&str[5]) << std::endl;
  std::cout << "str[5] : " << str[5] << std::endl;

  std::cout << std::endl;

}
```

It is particularly interesting to see in the example that the compiler performs the invocation `str[10]`. The access outside the string boundaries is *undefined behavior*. In contrast, the compiler complains about the call `str.at(10)`.

In the next lesson, we'll discuss how we can use strings for taking input and output the data.