## **Element Access**

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

WE'LL COVER THE FOLLOWING

Accessing the elements of a string

The access to the elements of a string str is very convenient, because the string supports random access iterators. You 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) you get the n-th element by index.

The following table provides an overview.

## Accessing the elements of a 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;</pre>
  std::string str= {"0123456789"};
  std::cout << "str.front(): " << str.front() << std::endl;</pre>
  std::cout << "str.back(): " << str.back() << std::endl;</pre>
  std::cout << std::endl;</pre>
  for (int i=0; i <= 10; ++i){
    std::cout << "str[" << i << "]: " << str[i] << std::endl;</pre>
  }
  std::cout << std::endl;</pre>
  try{
    str.at(10);
  }
  catch (const std::out_of_range& e){
    std::cerr << "Exception: " << e.what() << std::endl;</pre>
  std::cout << std::endl;</pre>
  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;</pre>
  std::cout << std::endl;</pre>
}
```

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
behaviour. In contrary, the compiler complains the call str.at(10).