## **Arrays**

The array type is perhaps the most popular sequential container. This lesson will cover its properties in detail.

This is what an array looks like:

1 2 3 4 5 6 7 8 9 10		1	2	3	4	5	6	7	8	9	10
----------------------	--	---	---	---	---	---	---	---	---	---	----

std::array is a homogeneous container of fixed length. It needs the header
<array>. The std::array combines the memory and runtime characteristic of
a C array with the interface of std::vector. This means in particular, the
std::array knows its size. You can use std::array in the algorithms of the
STL.

You have to keep a few special rules in your mind to initialise a std::array.

- std::array<int, 10> arr: The 10 elements are not initialised.
- std::array<int, 10> arr{}: The 10 elements are default initialised.
- std::array<int, 10> arr{1, 2, 3, 4, 5}: The remaining elements are default initialised.

std::array supports three types of index access.

```
- arr[n];
- arr.at(n);
- std::get<n>(arr);
```

The most often used first type form with angle brackets does not check the boundaries of the arr. This is in opposition to arr.at(n). You will get eventually a std::range-error exception. The last type shows the relationship of the std::array with the std::tuple, because both are containers of fixed length.

Here is a little bit of arithmetic with std::array.

```
// array.cpp
                                                                                          6
#include <iostream>
#include <array>
#include <numeric>
using namespace std;
int main(){
 std::array<int, 10> arr{1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
 for (auto a: arr) std::cout << a << " " ; // 1 2 3 4 5 6 7 8 9 10
  cout << "\n";</pre>
 double sum= accumulate(arr.begin(), arr.end(), 0);
  std::cout << sum << std::endl;</pre>
                                               // 55
 double mean= sum / arr.size();
  std::cout << mean << std::endl;</pre>
                                               // 5.5
  std::cout << (arr[0] == std::get<0>(arr)); // 1 (1 represents true)
  return 0;
}
```

**>** 





רה

std::array