std::pair

C++ allows us to pair values, which often comes in handy in programming.

WE'LL COVER THE FOLLOWING

With `std::pair`, we can build pairs of arbitrary types. The class template `std::pair` needs the header `<utility>`. `std::pair` has a default, copy, and move constructor. Pair objects can be swapped: `std::swap(pair1, pair2)`.

Pairs are used in the C++ library. For example, the function `std::minmax` returns its result as a pair and the associative containers `std::map`, `std::unordered_map`, `std::multimap`, and `std::unordered_multimap` manage their key/value association in pairs.

To get the elements of a pair `p`, we can either access them directly or via an index. So, with `p.first` or `std::get<0>(p)` we get the first element of the pair, and with `p.second` or `std::get<1>(p)`, we get the second element of the pair.

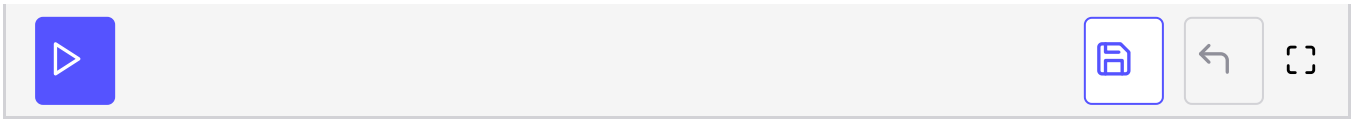
Pairs support the comparison operators `==`, `!=`, `<`, `>`, `<=`, and `>=`. If we want to know if two pairs are identical, first the members `pair1.first` and `pair2.first` will be compared, followed by `pair1.second` and `pair2.second`. The same strategy holds true for the other comparison operators.

std::make_pair

C++ has the practical help function `std::make_pair` to generate pairs, without specifying their types, since the function will automatically deduce their types.

Let's take a look at an example:

```cpp
#include <iostream>
#include <utility>

int main(){

  std::cout << std::endl;

  std::pair<const char*, double> charDoub("string", 3.14);
  std::pair<const char*, double> charDoub2 = std::make_pair("string", 3.14);
  auto charDoub3 = std::make_pair("string", 3.14);

  std::cout << "charDoub: (" << charDoub.first << ", " << charDoub.second << ")" << std::endl
  charDoub.first = "String";
  std::get<1>(charDoub) = 4.14;
  std::cout << "charDoub: (" << charDoub.first << ", " << charDoub.second << ")" << std::endl

  std::cout << std::endl;

  if (charDoub2 == charDoub3) std::cout << "charDoub2 == charDoub3" << std::endl;

  if (charDoub > charDoub3) std::cout << "charDoub > charDoub3" << std::endl;

  std::cout << std::endl;

}
```



The helper function `std::make_pair

Further information

- associative container

In the next lesson, we will talk about tuples in C++.