

- Solution

We'll look into the solution of exercise from the previous lesson.

WE'LL COVER THE FOLLOWING ^

- Solution
- Explanation

Solution

```
#include <iostream>
#include <set>
#include <unordered_set>

int main(){

    std::cout << std::endl;

    // constructor
    std::unordered_multiset<int> multiSet{1, 2, 3, 4, 5, 6, 7, 8, 9, 8, 7, 6, 5, 4, 3, 2, 1};
    std::unordered_set<int> uniqSet(multiSet.begin(), multiSet.end());

    // show the difference
    std::cout << "multiSet: ";
    for(auto m : multiSet) std::cout << m << " ";

    std::cout << std::endl;

    std::cout << "uniqSet: ";
    for(auto s : uniqSet) std::cout << s << " ";

    std::cout << std::endl << std::endl;

    // insert elements
    multiSet.insert(-1000);
    uniqSet.insert(-1000);

    std::set<int> mySet{-5, -4, -3, -2, -1};
    multiSet.insert(mySet.begin(), mySet.end());
    uniqSet.insert(mySet.begin(), mySet.end());

    // show the difference
    std::cout << "multiSet: ";
    for(auto m : multiSet) std::cout << m << " ";
```

```

std::cout << std::endl;

std::cout << "uniqSet: ";
for(auto s : uniqSet) std::cout << s << " ";

std::cout << std::endl << std::endl;

// search for elements
auto it = uniqSet.find(5);
if (it != uniqSet.end()){
    std::cout << "uniqSet.find(5): " << *it << std::endl;
}

std::cout << "multiSet.count(5): " << multiSet.count(5) << std::endl;

std::cout << std::endl;

// remove
int numMulti = multiSet.erase(5);
int numUniq = uniqSet.erase(5);

std::cout << "Erased " << numMulti << " times 5 from multiSet." << std::endl;
std::cout << "Erased " << numUniq << " times 5 from uniqSet." << std::endl;

// all
multiSet.clear();
uniqSet.clear();

std::cout << std::endl;

std::cout << "multiSet.size(): " << multiSet.size() << std::endl;
std::cout << "uniqSet.size(): " << uniqSet.size() << std::endl;

std::cout << std::endl;
}

```



Explanation

- In lines 10 and 11, we have initialized an `std::unordered_multiset` with some integer values and also an `std::unordered_set`, which contains unique values which are repeated in `std::unordered_multiset`.
- In lines 25 and 26, we inserted the value -1000 in both sets.
- In lines 54 and 55, we have erased the value 5 from both sets.
- In lines 61 and 62, we have used the `clear` function which deletes all elements from both of the sets.

This concludes our discussion on unordered associative containers. In the next chapter, we'll start off with algorithms.