Create and Delete

Below, we examine the different methods available for constructing and destroying containers with particular parameters.

We can construct each container using a multitude of constructors. To delete all elements of a container <code>cont</code>, use <code>cont.clear()</code>. It makes no difference if you create and delete a container or if we add and remove elements. Each time the container takes care of memory management.

The table shows the constructors and destructors of a container. All these functions can be used with std:vector.

Туре	Example
Default	<pre>std::vector<int> vec1</int></pre>
Range	<pre>std::vector<int> vec2(vec1.begin(), vec1.end())</int></pre>
Сору	<pre>std::vector<int> vec3(vec2)</int></pre>
Сору	<pre>std::vector<int> vec3= vec2</int></pre>
Move	<pre>std::vector<int> vec4(std::move(vec3))</int></pre>
Move	<pre>std::vector<int> vec4= std::move(vec3)</int></pre>
Sequence (Initializer list)	<pre>std::vector<int> vec5 {1, 2, 3, 4, 5}</int></pre>

Sequence (Initializer list)	std::vector <int> vec5= {1, 2, 3,</int>
	4, 5}
Destructor	vec5.~vector()
Delete elements	<pre>vec5.clear()</pre>

Creation and deletion of a container

Because std::array is generated at compile-time, there are a few things that
are special. std::array has no move constructor and can't be created with a
range or with an initializer list. However, an std::array can be initialized
with an aggregate initialization. Also, std::array has no method for removing
its elements.

Now we can use the different constructors on the different containers.

```
// containerConstructor.cpp
#include <iostream>
#include <map>
#include <unordered map>
#include <vector>
using namespace std;
int main(){
  vector<int> vec= {1, 2, 3, 4, 5, 6, 7, 8, 9};
  map<string, int> m= {{"bart", 12345}, {"jenne", 34929}, {"huber", 840284} };
  unordered_map<string, int> um{m.begin(), m.end()};
  for (auto v: vec) cout << v << " "; // 1 2 3 4 5 6 7 8 9
  cout << "\n";
  for (auto p: m) cout << p.first << "," << p.second << " "; //bart,12345 huber,840284 jenne,
  cout << "\n";</pre>
  for (auto p: um) cout << p.first << "," << p.second << " "; //bart,12345 jenne,34929 huber,
  cout << "\n";</pre>
  vector<int> vec2= vec;
  cout << vec.size() << endl; // 9</pre>
  cout << vec2.size() << endl; // 9</pre>
  vector<int> vec3= move(vec);
  cout << vec.size() << endl; // 0</pre>
  cout << vec3.size() << endl; // 9</pre>
  vec3.clear();
  cout << vec3.size() << endl; // 0</pre>
  return 0;
```





Various Constructors

In the next lesson, we'll discuss how to determine the size and capacity of a container.