universität innsbruck

VU C-C++: std::vector<T>

VU Lecture Slides

Johannes Gerstmayer & Markus Walzthöni

uibk.ac.at

Overview

Content:

- Basics
- std::vector<T>
- STL functions

C++ Basics - References

Key differences between references and pointers:

- References **need** to be initialized
- References **can't** be `NULL`
- References **don't** provide no arithmetic operations
- References **don't** need to be dereferenced

Please use References wherever you can. If you **need** pointers then use `std::shared_ptr<T>` !

C++ Basics - Special Constructors

```
class Reference {
public:
  // Default-, Copy-, Move-Constructor
  Reference();
  Reference(const Reference&);
  Reference(Reference&&);

  // Copy-, Move-Assignment Operator
  Reference& operator=(const Reference&);
  Reference& operator=(Reference&&) noexcept;
};
```

C++ Basics - Lambdas

A lambda is an unnamed function object. It can capture variables within its scope.

```
[ capture ]( params ){ body }
```

Capture options:

- `[=]`
- `[&]`
- `[=,&a]`
- `[&a = b, b = b + 1]`

std::vector<T> - Initialization

```
#include <vector>

std::vector<int> a;
std::vector<int> b = a;
std::vector<int> c {1,2,3,4,5};
```

std::vector<T> - Usage - Insertion

```
std::vector<int> vec;

vec.push_back(1);
vec.emplace_back(3);
vec.insert(vec.begin()+1, 2);
vec.insert(vec.end(), 5);
vec.emplace(vec.end()-1, 4);

...
```

std::vector<T> - Usage - Deletion

```
...

vec.erase(vec.begin());

// will delete all elements BEFORE vec.begin()+2
vec.erase(vec.begin(), vec.begin()+2);
vec.pop_back();
vec.clear();
```

std::vector<T> - Usage - Access

```
#include <vector>
#include <iostream>

std::vector<int> c {1,2,3,4,5};

std::cout << c[1] << std::endl;
std::cout << c.at(2) << std::endl;
std::cout << c.front() << std::endl;
std::cout << c.back() << std::endl;

...
```

std::vector<T> - Usage - Capacity

```
...

std::cout << vec.empty() << std::endl;
std::cout << vec.size() << std::endl;
```

STL utility functions

```
#include <vector>
#include <algorithm>
#include <iostream>

std::vector<int> a {1,2,3,4,5,6,7,8,9};

std::for_each(a.begin(),a.end(), [](int &n){ n++; });

std::count(a.begin(),a.end(), 3);
std::count_if(a.begin(),a.end(), [](const int& n){
  return (n % 2) == 0;
});

...
```

STL utility functions

```
...
#include <random>
auto rng = std::default_random_engine{};
std::shuffle(a.begin(), a.end(), rng);

// using introsort
std::sort(a.begin(), a.end());
std::sort(a.begin(), a.end(),
[](const auto& a, const auto& b){
  return b < a;
});
// std::stable_sort(a.begin(), a.end());
...
```

STL utility functions

```
...

bool r1 = std::all_of(a.begin(), a.end(),
  [](int i){ return i < 10; }
);
bool r2 = std::any_of(a.begin(), a.end(),
  [](int i){ return i == 7; }
);
bool r3 = std::none_of(a.begin(), a.end(),
  [](int i){ return i == 11; }
);

...
```

STL utility functions

```
...

auto r4 = std::remove(a.begin(), a.end(), 7);
a.erase(r4, a.end());

auto r5 = std::remove_if(a.begin(), a.end(),
  [](const auto& elem){ return elem > 9}
);
a.erase(r5, a.end());

...
```

STL utility functions

```
...

auto r6 = std::find(a.begin(), a.end(), 6);
if(r6 != a.end()) {
  std::cout << "Found a 6" << std::endl;
}

auto r7 = std::find_if(a.begin(), a.end(),
  [](const auto& elem){ return elem < 0}
);
if(r7 != a.end()) {
  std::cout << "Negative element found" << std::endl;
}

...
```

STL utility functions

```
...

std::transform(a.begin(), a.end(), a.begin(),
  [&](auto elem){ return elem % 2 ? elem + 1 : elem;
});

auto r8 = std::unique(a.begin(), a.end());
a.erase(r8, a.end());

...
```

STL utility functions

```
...

// return values are iterators
auto min_elem = std::min_element(a.begin(), a.end());
auto max_elem = std::max_element(a.begin(), a.end());
auto [min, max] =
  std::minmax_element(a.begin(), a.end());

...
```

Useful links

Here are some useful links in which you may be interested:

- CppReference - Vector
- CppReference - Algorithm
- CppReference - SharedPtr
- CppReference - CMath
- CppReference - Lambda

Thank you for your attention!

Johannes Gerstmayer & Markus Walzthöni

uibk.ac.at