Кирилл Волков @ MERA
github.com/vulko/Cpp_Basics_Lectures

C++

Стандартная библиотека (STL): контейнеры данных, вспомогательные классы и функции.

Шаблоны функций, шаблоны классов и их применение.

STL компоненты

- Algorithms
- Containers
- Functions
- Iterators

Sequence Containers:

data structures which can be accessed in a sequential manner

- vector
- list
- deque
- arrays
- forward_list

Container Adaptors:

provide a different interface for sequential containers

- queue
- priority_queue
- stack

- Array based
- Capacity can be increased

	Add front	Add random	Add back	Remove front	Remove random	Remove back	Access random	Find
Complexity	O(N)	O(N)	O(1) *O(N)	O(N)	O(N)	O(1)	O(1)	O(N) *O(log N)

- Double-linked list
- Capacity is unlimited. Items can be fragmented

	Add front	Add random	Add back	Remove front	Remove random	Remove back	Access random	Find
Complexity	O(1)	O(N)	O(1)	O(1)	O(N)	O(1)	O(N)	O(N)

- Based on array
- Can consist of multiple arrays, fragmented in different parts of memory

	Add front	Add random	Add back	Remove front	Remove random	Remove back	Access random	Find
Complexity	O(1)	O(N)	O(1)	O(1)	O(N)	O(1)	O(1)	O(N) *O(log N)

Associative Containers:

sorted data structures that can be quickly searched

Key -> Value

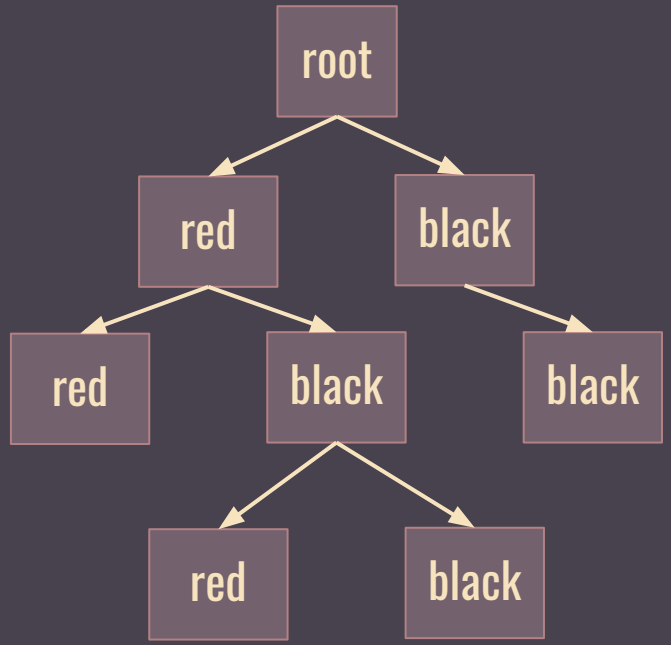
- set
- multiset
- map
- multimap

Unordered Associative Containers: *C++11*

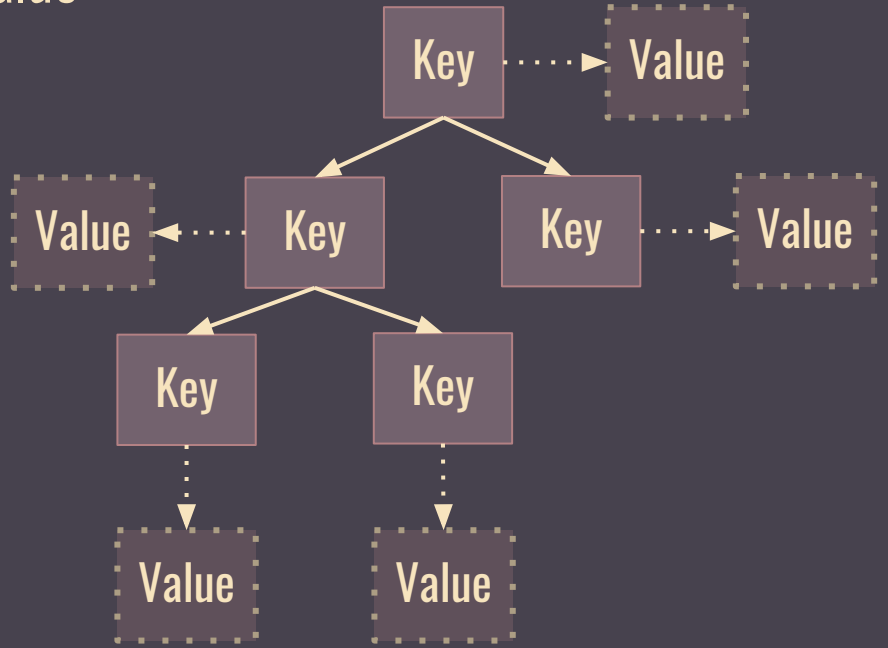
- unordered_set
- unordered_multiset
- unordered_map
- unordered_multimap

- Based on Red-Black Tree
- Sorted through comparing when adding elements
- Can’t have equal elements. Multiset can.

	Add	Remove	Find
Complexity	O(log N)	O(log N)	O(log N)

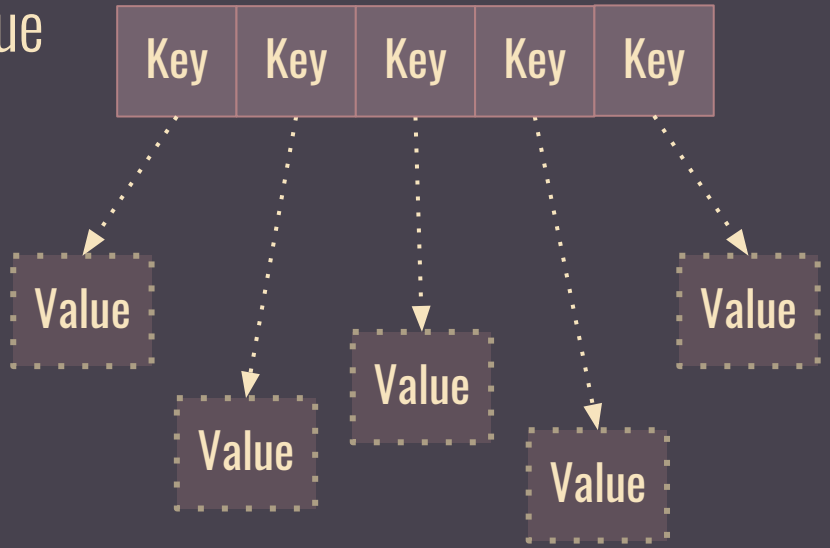


- Each item represents a Key -> Value
- Based on Red-Black Tree
- Sorted through comparing when adding element
- Can’t have multiple items with same key. Multiset can.



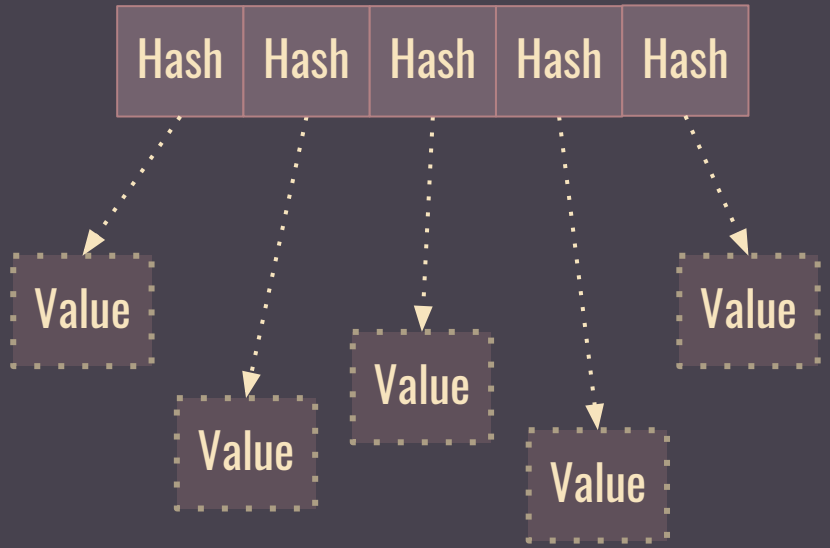
	Add	Remove	Find
Complexity	O(log N)	O(log N)	O(log N)

- Each item represents a Key -> Value
- Based on Hash Table
- Unsorted
- Can have worser complexity in case map is unbalanced



	Add	Remove	Find
Complexity	O(1) *O(N)	O(1) *O(N)	O(1) *O(N)

- Each item represents a Value
- Based on Hash Table
- Unsorted
- Can have worser complexity in case map is unbalanced



	Add	Remove	Find
Complexity	O(1) *O(N)	O(1) *O(N)	O(1) *O(N)

std::sort()

sorts arrays and iterable elements

Based on Quick Sort. O(n log[n]) complexity..

```
int a[10]= {1, 5, 8, 9, 6, 7, 3, 4, 2, 0};
sort(a, a+10);

// now a is:
// {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

vector<int> v(a);
sort(v.begin(), v.end());
```

```
bool compare(const int& lhs, const int& rhs){
    return lhs > rhs;
}

...

    sort(v.begin(), v.end(), compare);
```

std::binary_search() *Search a sorted collection for an item*

std::find() *Returns iterator to a first occurence of an element in range*

```
int a[10]= {1, 5, 8, 9, 6, 7, 3, 4, 2, 0};
sort(a, a+10);

// now a is:
// {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

vector<int> v(a);
sort(v.begin(), v.end());
```

```
bool found = false;

found = binary_search(a, a + 10, 2);

found = binary_search(v.begin(), v.end(), 2);
```

*max_element(begin, end)	*Maximum element in range*
*min_element(begin, end)	*Minimum element in range*
accumulate(begin, end, initialValue)	*Does the summation of vector elements*

```
int maxVal = *max_element(vec.begin(), vec.end());

int minVal = *min_element(vec.begin(), vec.end());

int sumVal = accumulate(vec.begin(), vec.end(), 0);
```

Templates allow to generalises functions and classes to be type independent.

```
template <class T>
T getMax(T a, T b) {
    return (a > b ? a : b);
}
```

```
    int a = 4, b = 3, c;
    c = getMax(a, b);

    float max = getMax(0.5f, 1.2f);
```

Templates support multiple types. Can lead to issues in case used in a wrong way!

```
template <class T, class U>
T getMin(T a, U b) {
    return (a < b ? a : b);
}
```

```
    int a = 4;
    float b = 4.2f;

    cout << getMin(a, b);
```

```
template <class T>
class Pair {

public:
    Pair (T first, T second) {
        mValues[0] = first;
        mValues[1] = second;
    }

private;
    T mValues[2];
};
```

```
Pair<int, int> mCoords;

Pair<Player, Monster> mBattle;
```