CH-231-A

Algorithms and Data Structures

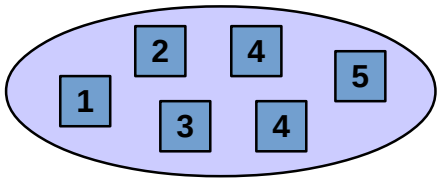
ADS

Lecture 4

Dr. Kinga Lipskoch

Spring 2020

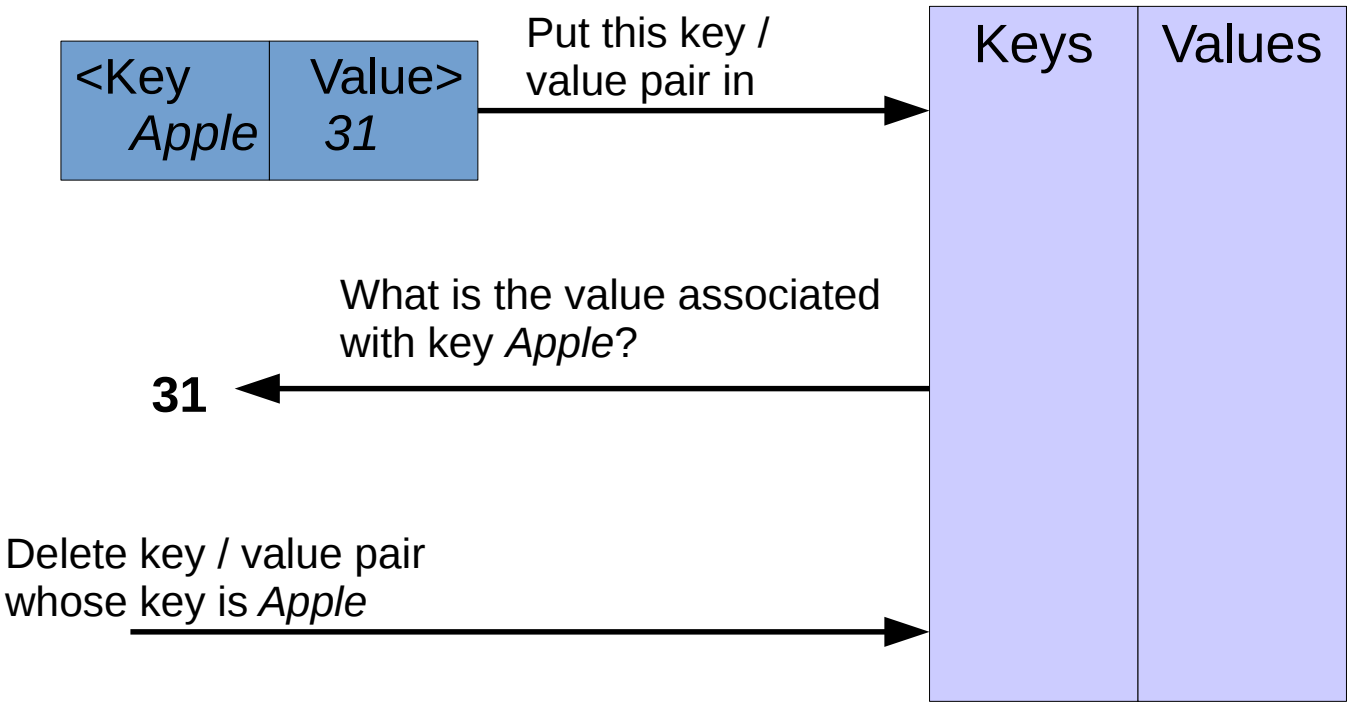
Multisets



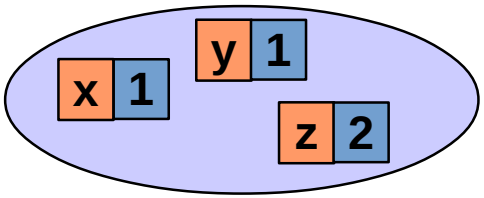
- A multiset is a container with an interface similar to set, but it accepts duplicate elements
- Both for sets and multisets, C++ STL provides algorithms for common (multiset) operations:
  - intersection, union, difference, symmetric difference

Associations

Associations work with pairs of keys and values

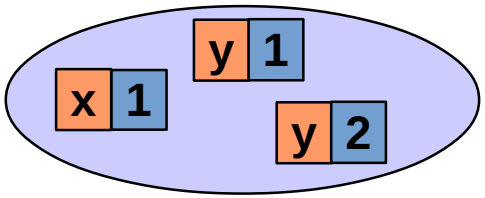


Maps



- Collection of elements, which are key/value pairs; the key is basis for ordering
- Duplicate keys are not allowed
- Called "associative array"

Multimaps



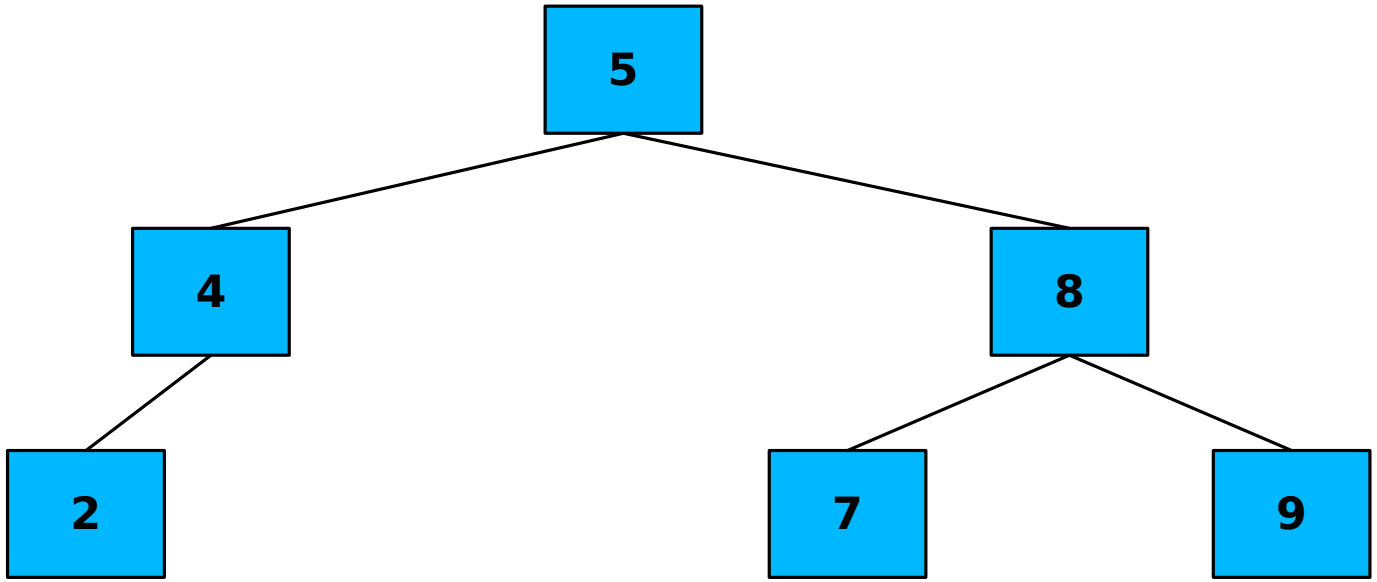
- Collection of elements, which are key/value pairs; the key is basis for ordering
- Duplicate keys are allowed
- Called “dictionary”

Maps and Multimaps

- Basic interface: `find`, `clear`, `erase`, `insert`
- Map iterators return pairs: first element is the key and second element is the value
- `mapsexample.cpp`

Internal Representation of Sets as Binary Tree

How do you iterate over the elements?



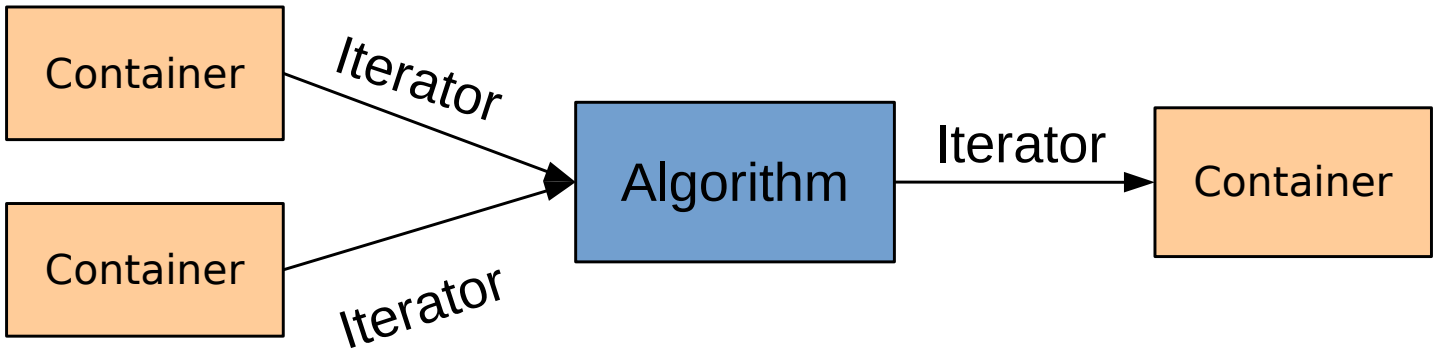
Other Selected Member Functions

- Common to all containers:
  - `begin()`, `end()`, `erase(...)`, `size()`
- Optional member functions:
  - `pop_back()`, `pop_front()`,
    `push_back(const value_type& x)`,
    `push_front(const value_type& x)`
- Specific member functions:
  - sequences, (associative also possible, as hint)
    - `insert(iterator p, const value_type& x)`
  - associative
    - `insert(const value_type& x)`

Time Overhead of Operations Sequence Containers

Operation	Vector	Deque	List
access first element	constant	constant	constant
access last element	constant	constant	constant
access random element	constant	constant	linear
add/delete at beginning	linear	constant	constant
add/delete at end	constant	constant	constant
add/delete at random	linear	linear	linear

Separation of Data and Algorithm



- Data is managed by container classes
- Operations are defined by configurable algorithms
- Iterators are the glue between these components
- Any algorithm may interact with any container

Algorithms (1)

STL provides standard algorithms that may process elements in container

- Non-manipulating algorithms:
  - `find(...)` find value in range
  - `count(...)` count appearances of value in range
  - `for_each(...)` apply function to range
  - `equal(...)` test whether the elements in two ranges are equal
  - `...`
- Manipulating algorithms:
  - `copy(...)` copy range of elements
  - `swap(...)` exchange values of two objects
  - `replace(...)` replace value in range
  - `remove(...)` remove value from range
  - `...`

Algorithms (2)

- Sorting algorithms:
  - `sort(...)` sort elements in range
  - `min(...)` return the smallest
  - `set_union(...)` union of two sorted ranges
  - `...`
- Numerical algorithms:
  - `accumulate(...)` accumulate values in range, use `#include <numeric>`
  - `...`
- They are not member functions of container classes
- Global functions that operate with iterators

Iterator Categories

- Input iterator – can only be used to read a sequence of values
- Output iterator – can only be used to write a sequence of values
- Forward iterator – can be read, written to, and move forward
- Bidirectional iterator – are like forward iterators, but can also move backwards
- Random access iterator – can move freely any number of steps in one operation

set_union() on Different Containers (1)

Used headers in both examples that follow

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
```

```
1 #include <iostream>
2 #include <set>
3 #include <algorithm>
4 using namespace std;
```

```
1 set_union(InpIterator first1, InpIterator last1,
2            InpIterator first2, InpIterator last2,
3            OutIterator result)
4
5 inserter(Container, InpIterator)
```

set_union() on Different Containers (2)

```
1  int main() {
2    typedef vector<int> IntVec;
3    IntVec a, b, c;
4    a.push_back(2);
5    a.push_back(3);
6    b.insert(b.end(), 2);
7    b.insert(b.end(), 4);
8    set_union(a.begin(),a.end(),
9      b.begin(), b.end(),
10     inserter(c, c.begin()));
11   IntVec::const_iterator pos;
12   for (pos=c.begin();pos!=c.
       end(); ++pos) {
13     cout << *pos << ' ';
14   }
15   cout << endl;
16   return 0;
17 }
```

2 3 4

```
1  int main() {
2    typedef set<string> StrSet;
3    StrSet a, b, c;
4    a.insert("BAA");
5    a.insert("CAA");
6    b.insert("BAA");
7    b.insert("DAA");
8    set_union(a.begin(),a.end(),
9      b.begin(), b.end(),
10     inserter(c, c.begin()));
11   StrSet::const_iterator pos;
12   for (pos=c.begin();pos!=c.
       end(); ++pos) {
13     cout << *pos << ' ';
14   }
15   cout << endl;
16   return 0;
17 }
```

BAA CAA DAA

Other Set Operations

- `set_intersection(...)` $A \cap B$
- `set_difference(...)` $A \setminus B$
- `set_symmetric_difference(...)` $(A \setminus B) \cup (B \setminus A)$

Pros and Cons: Algorithms

- Advantages
  - Implemented only once for any container type
  - Might operate on elements of different container types
  - Reduces the code size
- Disadvantages
  - Usage not intuitive (high learning curve)
  - Some combinations of containers and algorithms might not work
  - Or combination is possible but not useful (speed, needed size)

Useful STL Resources

- The C++ Standard Library by Nicolai M. Josuttis, Addison Wesley, 2nd edition, 2012
- C++ Annotations (Version 10.7.2) by Frank B. Brokken http://www.icce.rug.nl/documents/cplusplus/cplusplus.html
- C++ Reference http://www.cppreference.com/
- The C++ Programming Language by Bjarne Stroustrup (3rd edition) Pub. Addison-Wesley, ISBN 0-201-88954-4
- STL Tutorial and Reference Guide C++ Programming with the Standard Template Library by David R. Musser and Atul Saini, Pub. Addison-Wesley, ISBN 0-201-63398-1