Сибирь I

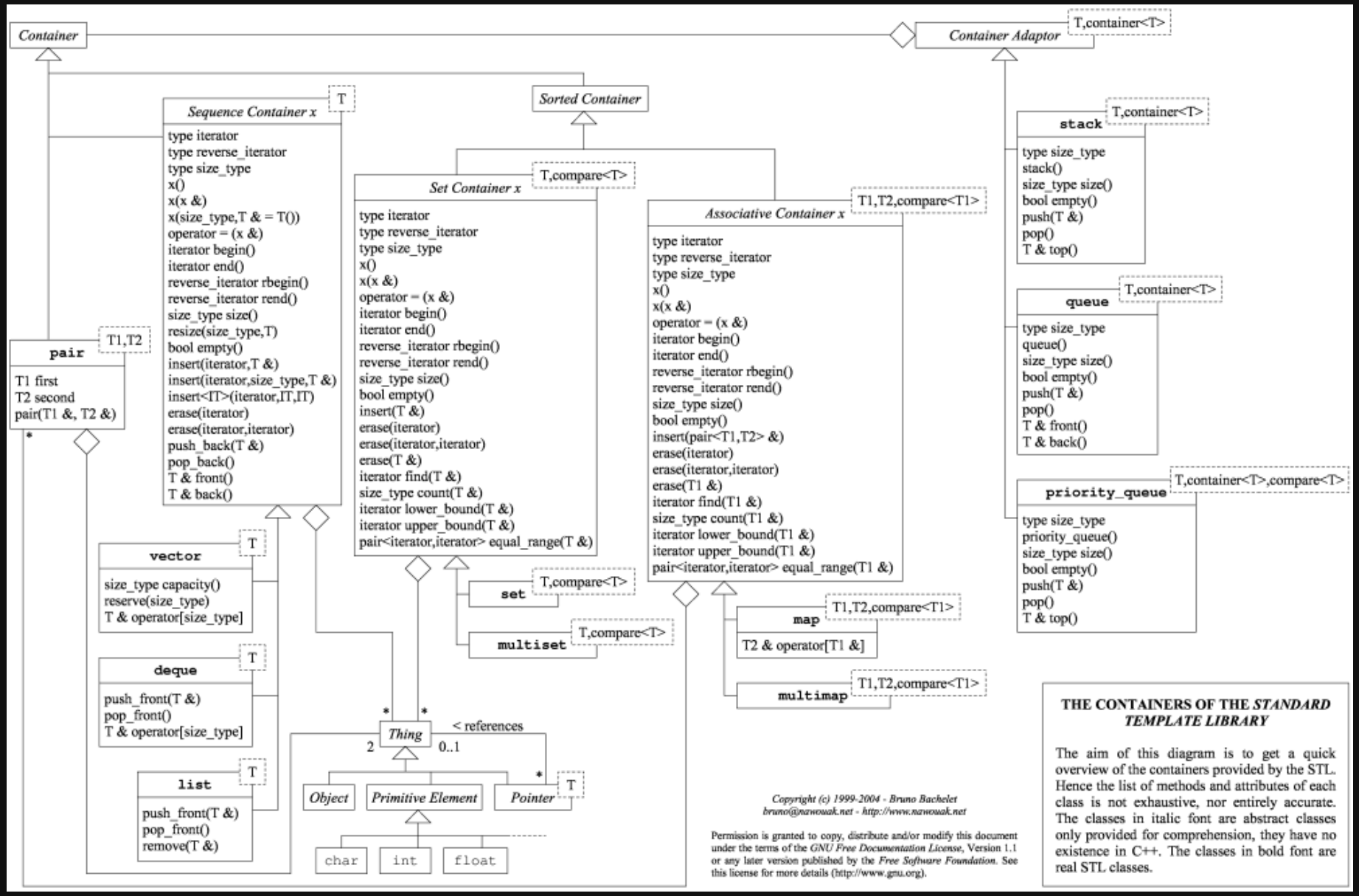
STL Алгоритмы

STL – Standard Template Library

Библиотека STL содержит набор шаблонов, представляющих контейнеры, итераторы, объекты функций и алгоритмов

Зачем оно вообще нужно?

Вот они (контейнеры) слева направо


Container
Container Adaptor
T,container<T>
Sequence Container x
T
type iterator
type reverse_iterator
type size_type
x()
x(x &)
x(size_type,T & = T())
operator = (x &)
iterator begin()
iterator end()
reverse_iterator rbegin()
reverse_iterator rend()
size_type size()
resize(size_type,T)
bool empty()
insert(iterator,T &)
insert(iterator,size_type,T &)
insert<IT>(iterator,IT,IT)
erase(iterator)
erase(iterator,iterator)
push_back(T &)
pop_back()
T & front()
T & back()
Sorted Container
Set Container x
T,compare<T>
type iterator
type reverse_iterator
type size_type
x()
x(x &)
operator = (x &)
iterator begin()
iterator end()
reverse_iterator rbegin()
reverse_iterator rend()
size_type size()
bool empty()
insert(T &)
erase(iterator)
erase(iterator,iterator)
erase(T &)
iterator find(T &)
size_type count(T &)
iterator lower_bound(T &)
iterator upper_bound(T &)
pair<iterator,iterator> equal_range(T &)
Associative Container x
T1,T2,compare<T1>
type iterator
type reverse_iterator
type size_type
x()
x(x &)
operator = (x &)
iterator begin()
iterator end()
reverse_iterator rbegin()
reverse_iterator rend()
size_type size()
bool empty()
insert(pair<T1,T2> &)
erase(iterator)
erase(iterator,iterator)
erase(T1 &)
iterator find(T1 &)
size_type count(T1 &)
iterator lower_bound(T1 &)
iterator upper_bound(T1 &)
pair<iterator,iterator> equal_range(T1 &)
stack
T,container<T>
type size_type
stack()
size_type size()
bool empty()
push(T &)
pop()
T & top()
queue
T,container<T>
type size_type
queue()
size_type size()
bool empty()
push(T &)
pop()
T & front()
T & back()
priority_queue
T,container<T>,compare<T>
type size_type
priority_queue()
size_type size()
bool empty()
push(T &)
pop()
T & top()
pair
T1,T2
T1 first
T2 second
pair(T1 &, T2 &)
vector
T
size_type capacity()
reserve(size_type)
T & operator[size_type]
deque
T
push_front(T &)
pop_front()
T & operator[size_type]
list
T
push_front(T &)
pop_front()
remove(T &)
set
T,compare<T>
multiset
T,compare<T>
map
T1,T2,compare<T1>
T2 & operator[T1 &]
multimap
T1,T2,compare<T1>
Thing
< references
2
0..1
Object
Primitive Element
Pointer
T
char
int
float
THE CONTAINERS OF THE STANDARD TEMPLATE LIBRARY
The aim of this diagram is to get a quick overview of the containers provided by the STL. Hence the list of methods and attributes of each class is not exhaustive, nor entirely accurate. The classes in italic font are abstract classes only provided for comprehension, they have no existence in C++. The classes in bold font are real STL classes.
Copyright (c) 1999-2004 - Bruno Bachelet
bruno@nawouak.net - http://www.nawouak.net
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation. See this license for more details (http://www.gnu.org).

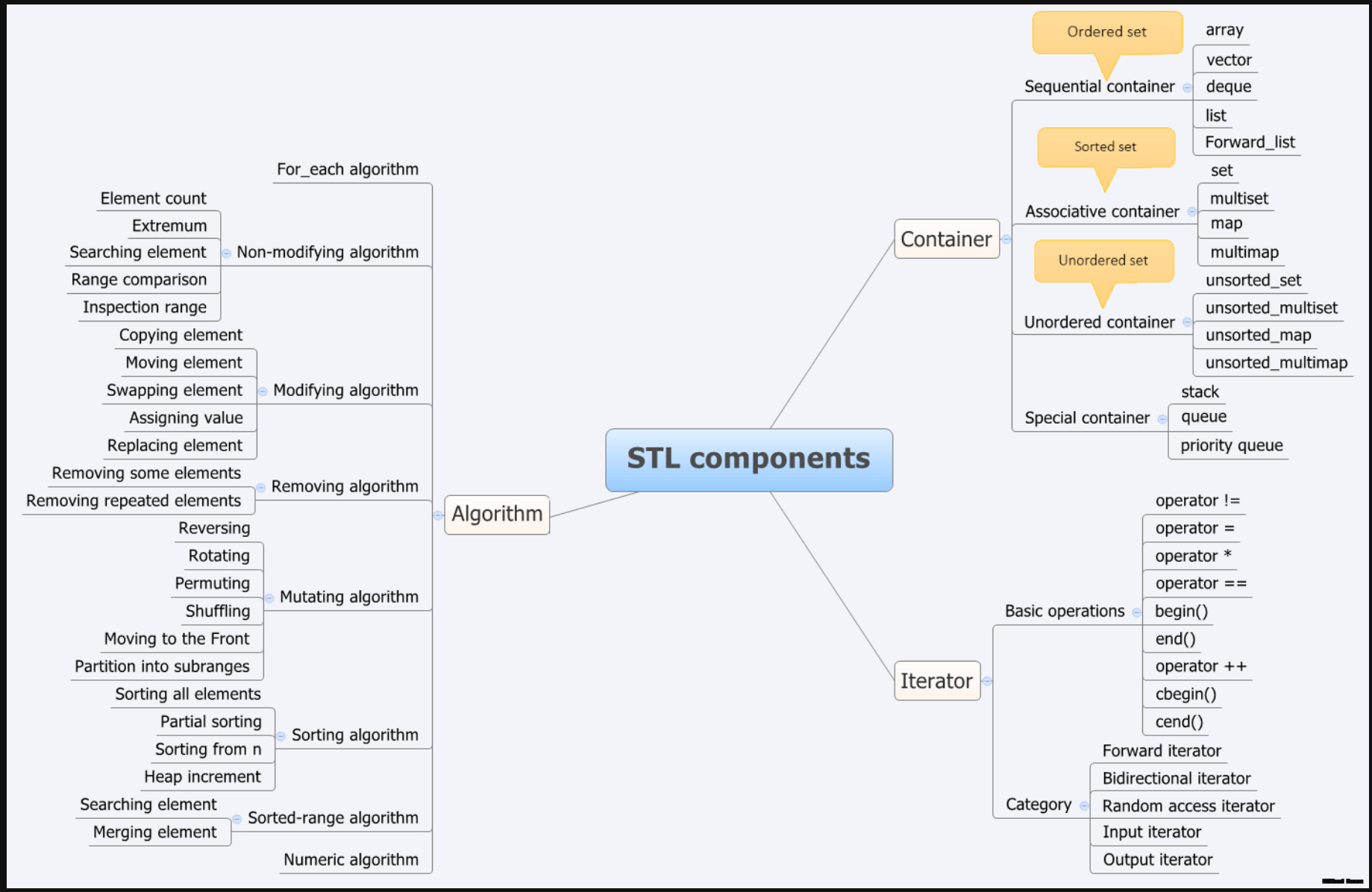
Возьмём std::vector

Часто ли приходится его сортировать, искать в нем элементы?

A в std::list?

Идея STL – обобщенное программирование – создание кода, ***не зависящего от типа данных***

Вот они (алгоритмы) сверху вниз


STL components
Container
Ordered set
array
vector
deque
list
Forward_list
Sequential container
Sorted set
set
multiset
map
multimap
Associative container
Unordered set
unsorted_set
unsorted_multiset
unsorted_map
unsorted_multimap
Unordered container
stack
queue
priority queue
Special container
Iterator
Basic operations
operator !=
operator =
operator *
operator ==
begin()
end()
operator ++
cbegin()
cend()
Category
Forward iterator
Bidirectional iterator
Random access iterator
Input iterator
Output iterator
Algorithm
For_each algorithm
Non-modifying algorithm
Element count
Extremum
Searching element
Range comparison
Inspection range
Modifying algorithm
Copying element
Moving element
Swapping element
Assigning value
Replacing element
Removing algorithm
Removing some elements
Removing repeated elements
Mutating algorithm
Reversing
Rotating
Permuting
Shuffling
Moving to the Front
Partition into subranges
Sorting algorithm
Sorting all elements
Partial sorting
Sorting from n
Heap increment
Sorted-range algorithm
Searching element
Merging element
Numeric algorithm

Небольшой list

- std::find
- std::find_if
- std::count_if
- std::transform
- std::sort
- std::any_of
- std::for_each

std::find

Возвращает итератор на первый элемент, равный value

```
template< class InputIt, class T >
InputIt find( InputIt first, InputIt last, const T& value);
```

std::find_if

Возвращает итератор на первый элемент, удовлетворяющий условию pred

```
template<class InputIterator, class Predicate>
InputIterator find_if(InputIterator first, InputIterator last, Pr
```

std::any_of

Проверяет, выполняется ли pred хоть для одного элемента последовательности

```
template< class InputIt, class UnaryPredicate >
bool any_of(InputIt first, InputIt last, UnaryPredicate pred);
```

std::count_if

Возвращает количество элементов, удовлетворяющих условию pred

```
template< class InputIt, class UnaryPredicate >
typename iterator_traits<InputIt>::difference_type
count_if(InputIt first, InputIt last, UnaryPredicate pred);
```

std::for_each

Выполняет функцию для каждого элемента

```
template< class InputIt, class UnaryFunction >
UnaryFunction for_each( InputIt first, InputIt last, UnaryFunctio
```

std::transform

Возвращает результат применения функции к каждому элементу последовательности

```
template <class InIter, class OutIter, class Func>
OutIter transform(InIter start, InIter end,
      OutIter result, Func unaryfunc);
```

std::sort

Сортирует

```
template<class RandomAccessIterator>
void sort(RandomAccessIterator first,
          RandomAccessIterator last);

template<class RandomAccessIterator, class Predicate>
void sort(RandomAccessIterator first,
          RandomAccessIterator last,
          Predicate comp);
```

lambda functions

aka. анонимные функции


LAMBDA
ME
A NORMAL FUNCTION
imgflip.com