華中科技大學

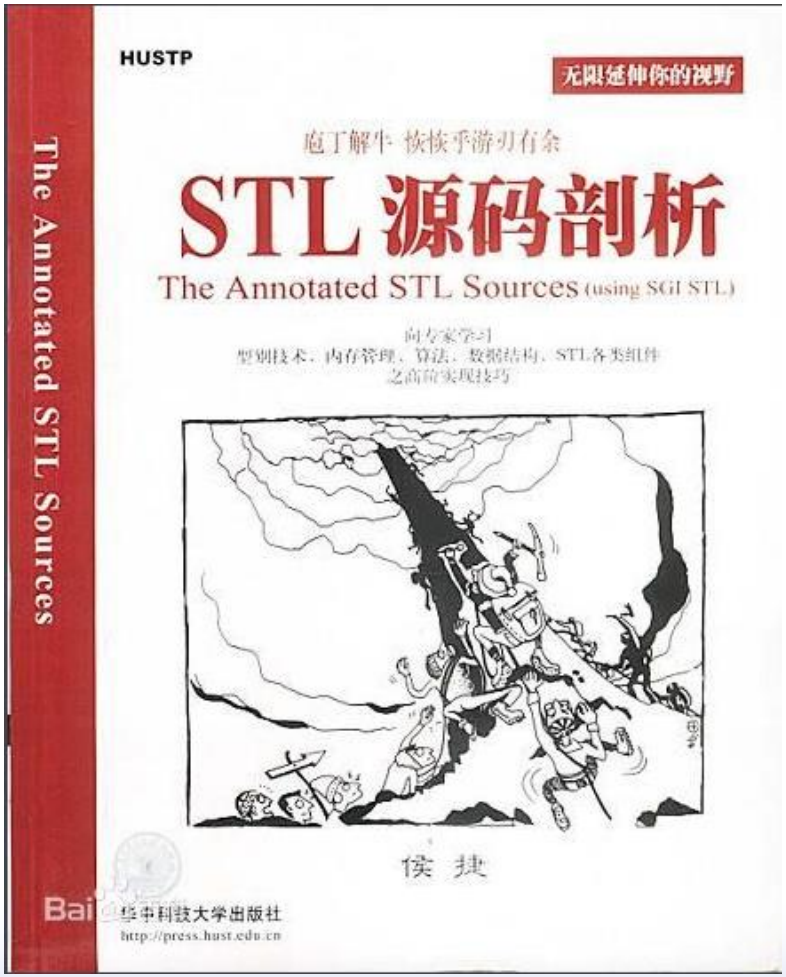
C++的标准模版库 STL

Standard Template Library

许向阳

xuxy@hust.edu.cn

参考书



肖波. 数据结构与STL. 北京邮电大学出版社. 2010年

提纲

華中科技大學

1. 概论
2. STL中的基本概念
3. 容器
4. 迭代器
5. 算法

软件重用

- 面向对象的思想

  继承和多态

  标准类库 Microsoft Foundation Classes

- 泛型程序设计的思想

  generic programming

  模板机制，以及标准模板库 STL

  Standard Template Library

1. 概论

泛型程序设计——使用模板的程序设计法

- 常用的数据结构（如链表，数组，二叉树）
- 常用算法（比如排序，查找）
- 常用的数据结构和算法写成模板
- 不论数据结构里放的是什么对象，算法针对什么对象，都不必重新实现数据结构，重新编写算法。
- STL主要由 Alex Stepanov 开发，于1998年被添加进C++标准

2. STL中的基本概念

容器：可容纳各种数据类型的**数据结构**。

Container ,Sequence /Associative Containers

Container Adapters

迭代器：可依次存取容器中元素

Iterator

算法：用来操作容器中的元素的**函数模板**。

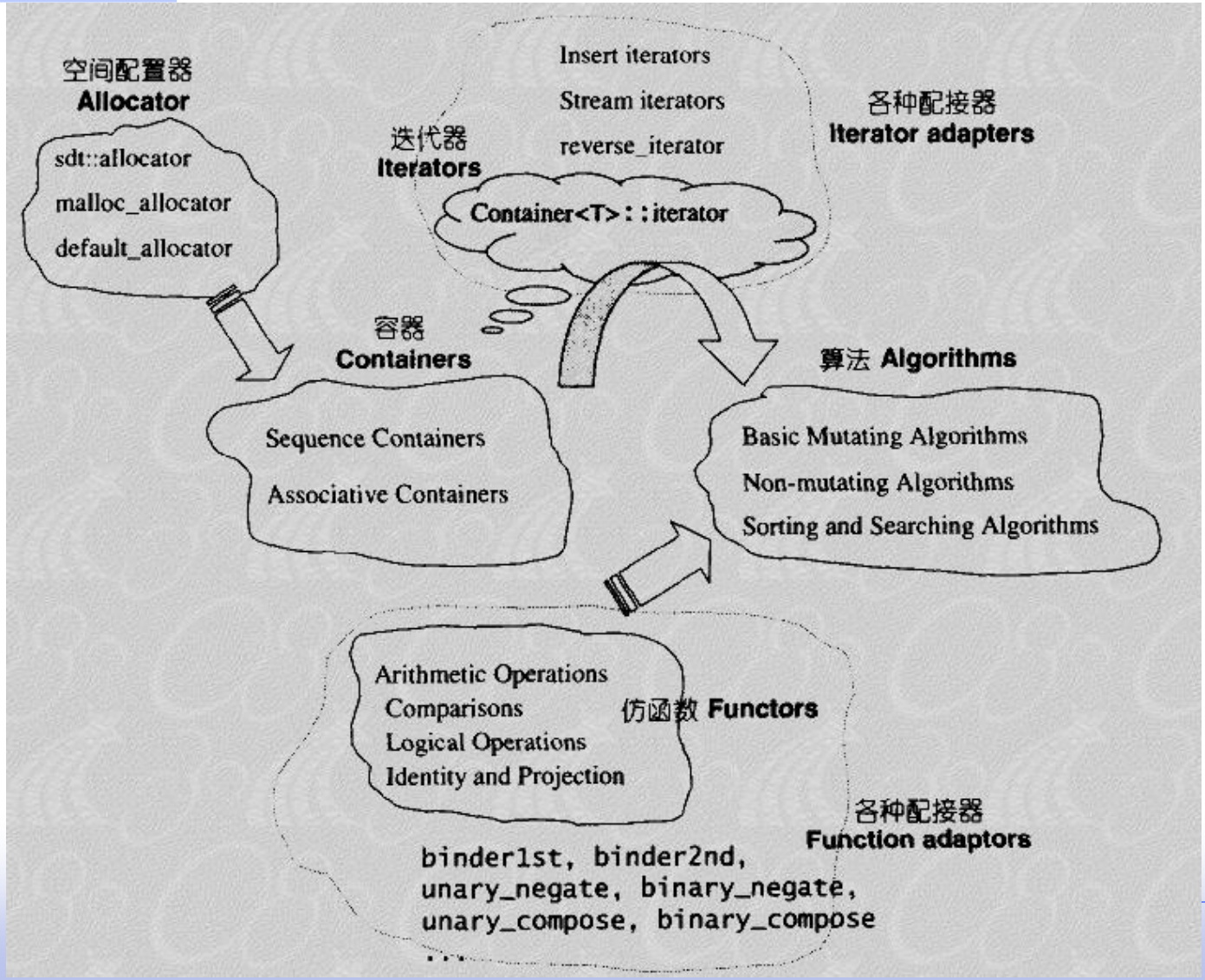
Algorithm

空间配置器： 负责空间配置和管理。

Allocators

仿函数：类似于函数，重载operator (), functors

2. STL中的基本概念


空间配置器
Allocator
sdt::allocator
malloc_allocator
default_allocator
迭代器
Iterators
Insert iterators
Stream iterators
reverse_iterator
Container<T>::iterator
各种配接器
Iterator adapters
容器
Containers
Sequence Containers
Associative Containers
算法 Algorithms
Basic Mutating Algorithms
Non-mutating Algorithms
Sorting and Searching Algorithms
Arithmetic Operations
Comparisons
Logical Operations
Identity and Projection
仿函数 Functors
各种配接器
Function adaptors
binder1st, binder2nd,
unary_negate, binary_negate,
unary_compose, binary_compose
...

2. STL中的基本概念

- Container 通过Allocator 取得数据储存空间
- Algorithm 通过Iterator 存取Container 中的内容
- Functor 协助Algorithm 完成不同的策略变化
- Adapter 可以修饰或套接Functor

3、容器

顺序容器

vector： 数组，后部插入/删除，直接访问

list： 双向链表，任意位置插入/删除

deque： 双端数组，前部插入/删除，
后部插入/删除，直接访问

3、容器

关联容器

set：　快速查找，无重复元素

multiset：快速查找，可有重复元素

map：　一对一映射，无重复元素，基于关键字查找

multimap：一对一映射，可有重复元素，基于关键字查找

hash_map

hash_multimap

hash_set

hash_multiset

3、容器

容器适配器

stack：LIFO

queue：FIFO

priority_queue：优先级高的元素先出

3、容器

list

- **Element access**:
  front Access first element (public member function)
- back Access last element (public member function)

Modifiers:

assign Assign new content to container (public member function)

- push_front Insert element at beginning (public member function)
- pop_front Delete first element (public member function)
- push_back Add element at the end (public member function)
- pop_back Delete last element (public member function)
- insert Insert elements (public member function)
- erase Erase elements (public member function)
- swap Swap content (public member function)
- clear Clear content (public member function)

3、容器

- **Iterators**:
  begin Return iterator to beginning (public member function)
- **end** Return iterator to end (public member function)
- **rbegin** Return reverse iterator to reverse beginning (public member function)
- **rend** Return reverse iterator to reverse end (public member function)

Capacity:

empty Test whether container is empty (public member function)

- **size** Return size (public member function)
- **max_size** Return maximum size (public member function)
- **resize** Change size (public member function)

3、容器

Operations:

splice Move elements from list to list (public member function)

- **remove** Remove elements with specific value (public member function)
- **remove_if** Remove elements fulfilling condition (public member function template)
- **unique** Remove duplicate values (member function)
- **merge** Merge sorted lists (public member function)
- **sort** Sort elements in container (public member function)
- **reverse** Reverse the order of elements (public member function)

4、迭代器

Iterator

类似于指针，亦称 广义指针，指向某个对象。

- 为算法提供输入数据
- 遍历容器或者流中的对象
- 不同容器上支持的迭代器功能强弱有所不同。
- 容器的迭代器的功能强弱，决定了该容器是否支持STL中的某种算法。

4、迭代器

容器所支持的迭代器类别

容器	迭代器类别
vector	随机，
deque	随机
list	双向
set/multiset	双向
map/multimap	双向
stack	不支持迭代器
queue	不支持迭代器
priority_queue	不支持迭代器

5、算法

STL中提供能在各种容器中通用的算法

- 插入
- 删除
- 查找
- 排序
- 大约有70种标准算法。

算法可以处理容器，也可以处理C语言的数组

5、算法

- 变化序列算法

copy ,remove,fill,replace,random_shuffle,swap, …..

会改变容器

- 非变化序列算法

adjacent-find, equal, mismatch,find ,count, search, count_if, for_each, search_n

- 以上函数模板都在<algorithm> 中定义
- 此外还有其他算法

5、算法

- ostream_iterator<int> output(cout , “*”);

  定义了一个 ostream_iterator 对象，可以通过cout输出以 * 分隔的一个个整数

- copy (v.begin(),v.end(),output);

  导致v的内容在 cout上输出

5、算法

- copy 函数模板(算法)

  template<class InIt, class OutIt>

  OutIt copy(InIt first, InIt last, OutIt x);

  本函数对每个在区间[0, last - first)中的N执行一次 *(x+N) = * (first + N) , 返回 x + N

- copy (v.begin(),v.end(),output)

  first 和 last 的类型是

  vector<int>::const_iterator

  output 的类型是 ostream_iterator<int>

5、算法

排序和查找算法

- **Sort**

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

- **find**

```
template<class InIt, class T>
InIt find(InIt first, InIt last, const T& val);
```

5、算法


```
int main()  {
      const int SIZE = 10;
      int a1[] = { 2,8,1,50,3,100,8,9,10,2 };
      vector<int> v(a1,a1+SIZE);
      ostream_iterator<int> output(cout," ");
      vector<int>::iterator location;
      location = find(v.begin(),v.end(),10);
      if( location != v.end()) {
            cout << endl << "1) " << location - v.begin();
      }
       sort(v.begin(),v.end());
       if( binary_search(v.begin(),v.end(),9))
           cout << endl << "3) " << "9 found";
      else    cout << endl << " 3) " << " 9 not found";
      return 0;
}
```

6、空间管理器

一般用户使用　new、malloc、delete、free

高级用户：改变内存分配策略，

采用自己定义的策略来实现内存管理

allocator

每种容器中，都隐藏了 allocator，默默完成内存的配置与释放，对象构造与析构的工作。

6、空间管理器

vector 头文件：

```
template <class _Ty, class _Alloc = allocator<_Ty>>
class vector { // varying size array of values
    ……
};
```

例子


```
typedef vector<int, allocator<int>> INTVECT;

void f()
{
        INTVECT  myVector(10);
        int  i;
        for (i=0;i<10;i++)
                myVector[i]=i;
        for (i=0;i<10;i++)
                cout<<myVector[i]<<endl;
}

void main()
{
    f();
}
```

例子

華中科技大學

vector::assign
vector::at
vector::back
vector::begin
vector::capacity
vector::cbegin
vector::cend
vector::clear
vector::crbegin
vector::crend
vector::data

vector::emplace
vector::emplace_back
vector::empty
vector::end
vector::erase
vector::front
vector::get_allocator
vector::insert
vector::max_size
vector::pop_back
vector::push_back

vector::rbegin
vector::rend
vector::reserve
vector::resize
vector::shrink_to_fit
vector::size
vector::swap
vector::vector

例子


```
typedef list<int, allocator<int>> INTLIST;

void f()
{
    INTLIST  myList;
    INTLIST::iterator myListPtr;
    int  i;
    for (i=9;i>=0;i--)
            myList.push_front(i);

    for (myListPtr=myList.begin(); myListPtr!=myList.end(); myListPtr++)
            cout<<*myListPtr<<endl;
}

void main()
{
      f();
}
```

头文件

C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.22.27905\include

<algorithm>、<deque>、<functional>、

<iterator>、<vector>、<list>、

<map>、<memory>、<numeric>、

<queue>、<set>、<stack>、<utility>