番外篇：C++ 标准模板库

C++ Standard Template Libarary

计算机系 江志英 jiangzy@mail.buct.edu.cn

主要内容

STL概述：组件、容器、迭代器（iterator）、算法

STL容器：

- 常用容器：**vector、deque、 list、 map/multimap 、set**
- 特殊容器：**stack、queue，priority_queue**
- 其他容器：**hashtable**

STL算法：搜寻、排序、拷贝、数值运算

STL概述

- STL是C++标准程序库的核心，深刻影响了标准程序库的整体结构
- STL是泛型（generic）程序库，利用先进、高效的算法来管理数据
- STL由一些可适应不同需求的集合类（collection class），以及在这些数据集合上操作的算法（algorithm）构成
- STL内的所有组件都由模板（template）构成，其元素可以是任意类型
- STL是所有C++编译器和所有操作系统平台都支持的一种库

STL概述

```
//普通C++代码
#include <iostream>
int main(void){
    double a[] = {1, 2, 3, 4, 5};
    std::cout<<mean(a, 5);
    std::cout<<std::endl;
    return 0;
}
```

```
//使用了STL的代码
#include <vector>
#include <iostream>
int main(){
    std::vector<double> a;
    a.push_back(1);
    a.push_back(2);
    a.push_back(3);
    a.push_back(4);
    a.push_back(5);
    for(int i = 0; i < a.size(); ++i){
        std::cout<<a[i]<<std::endl;
    }
    return 0;
}
```


信息学院 江志英 jiangzy@mail.buct.edu.cn

STL概述

```
//使用了STL的代码
#include <vector>
#include <iostream>
int main()
{
    std::vector< int > q;
    q.push_back(10);
    q.push_back(11);
    q.push_back(12);
    std::vector< int > v;
    for(int i=0; i<5; ++i){
        v.push_back(i);
    }
    …
}
```

```
std::vector<int>::iterator it = v.begin() + 1;
it = v.insert(it, 33);
v.insert(it, q.begin(), q.end());
it = v.begin() + 3;
v.insert(it, 3, -1);
it = v.begin() + 4;
v.erase(it);
it = v.begin() + 1;
v.erase(it, it + 4);
v.clear();
return 0;
```


信息学院 江志英 jiangzy@mail.buct.edu.cn

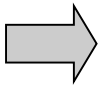
STL概述

针对一个或多个尚未明确的类型所撰写的函数或类

模板（template）

- 函数模板

```
#include<iostream>
#include<string>
using namespace std;
//定义函数模板
template<typename T>
T MAX(T a, T b) {
  return (a>b)?a:b;
}
```



```
int main(){
   int x=2,y=6;
    double x1=9.123,y1=12.6543;
    cout<<MAX(x,y)<<endl;
    cout<<MAX(x1,y1)<<endl;
}
```

STL概述

针对一个或多个尚未明确的类型所撰写的函数或类

模板（template）

- 类模板

```
#include<iostream>
using namespace std;
//定义名为ex_class的类模板
template < typename T>
class ex_class{
    T value;
public:
    ex_class(T v) { value=v; }
    void set_value(T v) { value=v; }
    T get_value(void) {return value;}
};
```

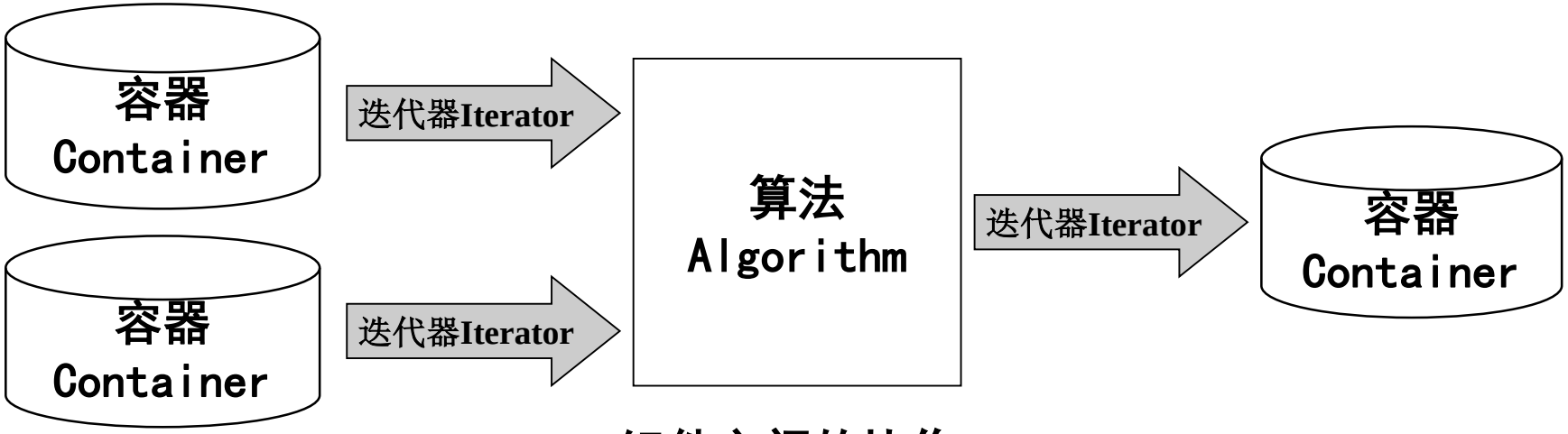
```
int main(){
    //测试char类型数据
    ex_class <char> ch('A');
    cout<<"ch.value:"<<ch.get_value()<<endl;
    ch.set_value('a');
    cout<<"ch.value:"<<ch.get_value()<<endl;
    //测试double类型数据
    ex_class <double> d(5.5);
    cout<<“d.value:"<<d.get_value()<<endl;
    x.set_value(7.5);
    cout<<“d.value:"<<x.get_value()<<endl;
}
```


信息学院 江志英 jiangzy@mail.buct.edu.cn

STL概述

STL组件

- 容器（Container） — 管理某类对象的集合
- 迭代器（Iterator） — 在对象集合上进行遍历
- 算法（Algorithm） — 处理集合内的元素
- 容器适配器（container adaptor）
- 函数对象(functor)

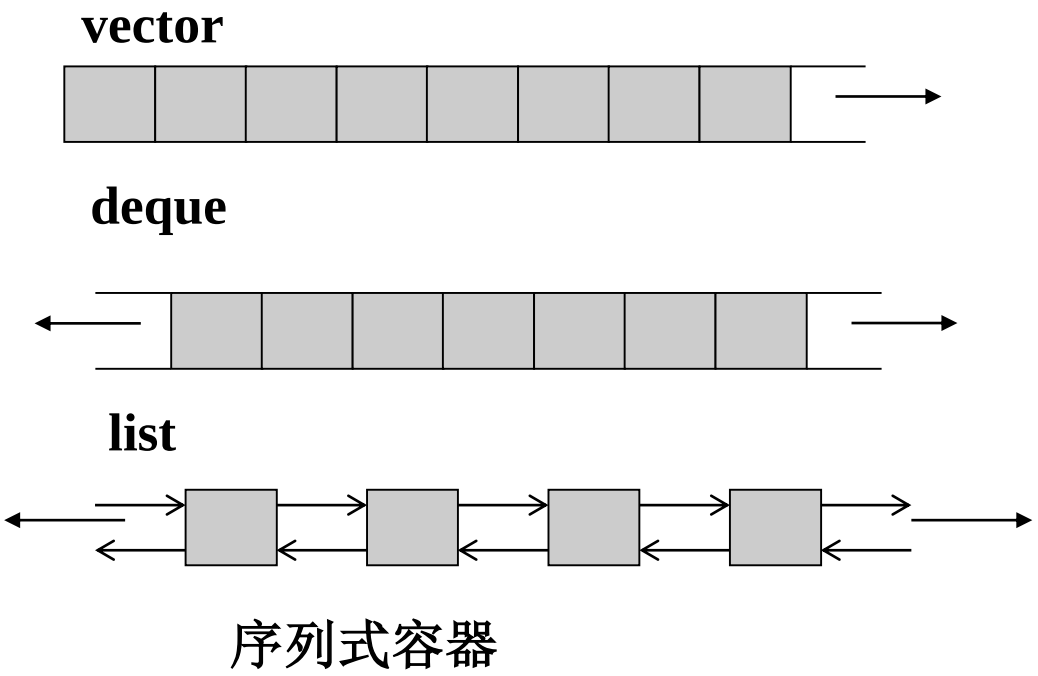


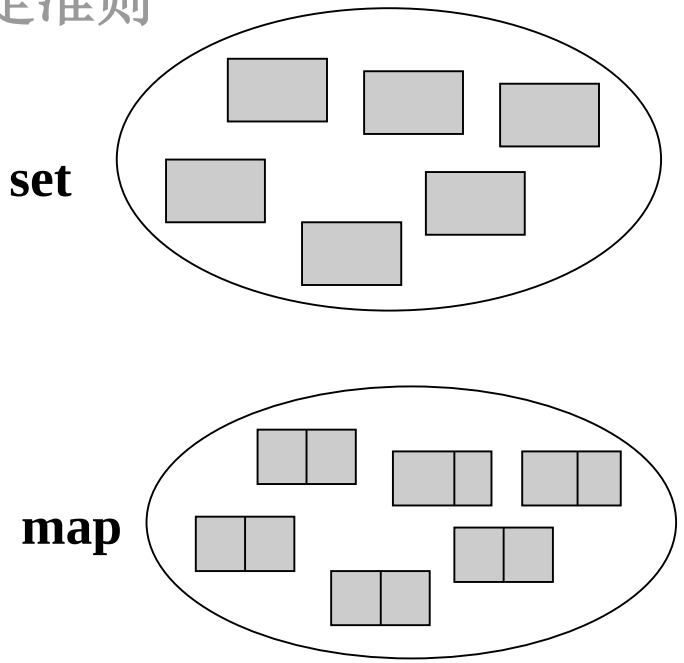
STL组件之间的协作

STL概述

STL容器类别

- 序列式容器－排列次序取决于插入时机和位置
- 关联式容器－排列顺序取决于特定准则





STL概述

STL容器的共通能力

- 所有容器中存放的都是值而非引用，即容器进行安插操作时内部实施的是拷贝操作。因此容器的每个元素必须能够被拷贝。如果希望存放的不是副本，容器元素只能是指针。
- 所有元素都形成一个次序（**order**），可以按相同的次序一次或多次遍历每个元素
- 各项操作并非绝对安全，调用者必须确保传给操作函数的参数符合需求，否则会导致未定义的行为

信息学院 江志英 **jiangzy**@mail.buct.edu.cn

STL概述

STL容器元素的条件

- 必须能够通过拷贝构造函数进行复制
- 必须可以通过赋值运算符完成赋值操作
- 必须可以通过析构函数完称销毁动作
- 序列式容器元素的默认构造函数必须可用
- 某些动作必须定义operator ==，例如搜寻操作
- 关联式容器必须定义出排序准则，默认情况是重载operator <

对于基本数据类型（int, long, char, double, ...）而言，以上条件总是满足

STL概述

STL容器的共通操作

- 初始化（initialization）
  - 产生一个空容器

```
std::list<int> l;
```

  - 以另一个容器元素为初值完成初始化

```
std::list<int> l;
…
std::vector<float> c(l.begin(), l.end());
```

  - 以数组元素为初值完成初始化

```
int array[]={2,4,6,1345};
…
std::set<int> c(array, array+sizeof(array)/sizeof(array[0]));
```

STL概述

STL容器的共通操作

- 与大小相关的操作（**size operator**）
  - size()—返回当前容器的元素数量
  - empty()—判断容器是否为空
  - max_size()—返回容器能容纳的最大元素数量
- 比较（**comparison**）
  - ==,!=,<,<=,>,>=
  - 比较操作两端的容器必须属于同一类型
  - 如果两个容器内的所有元素按序相等，那么这两个容器相等
  - 采用字典式顺序判断某个容器是否小于另一个容器

STL概述

STL容器的共通操作

- 赋值（**assignment**）和交换（**swap**）
  - swap用于提高赋值操作效率
- 与迭代器（iterator）相关的操作
  - begin()—返回一个迭代器，指向第一个元素
  - end()—返回一个迭代器，指向最后一个元素之后
  - rbegin()—返回一个逆向迭代器，指向逆向遍历的第一个元素
  - rend()—返回一个逆向迭代器，指向逆向遍历的最后一个元素之后

STL概述

容器的共通操作

- 元素操作
  - insert(pos,e)—将元素e的拷贝安插于迭代器pos所指的位置
  - erase(beg,end)—移除[beg，end]区间内的所有元素
  - clear()—移除所有元素

STL概述

迭代器（iterator）（示例:iterator）

- 可遍历STL容器内全部或部分元素的对象
- 指出容器中的一个特定位置
- 迭代器的基本操作

操作	效果
*	返回当前位置上的元素值。如果该元素有成员，可以通过迭代器以operator ->取用
++	将迭代器前进至下一元素
==和!=	判断两个迭代器是否指向同一位置
=	为迭代器赋值（将所指元素的位置赋值过去）

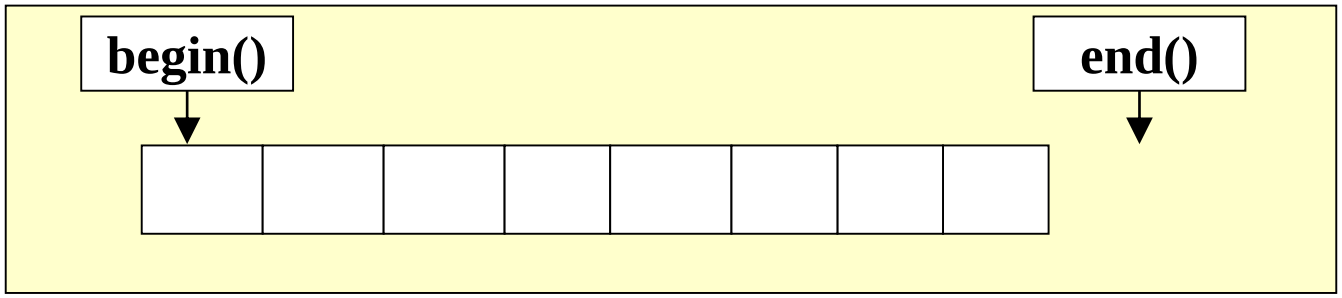
信息学院 江志英 jiangzy@mail.buct.edu.cn

STL概述

迭代器（iterator）

- 所有容器都提供获得迭代器的函数

操作	效果
begin()	返回一个迭代器，指向第一个元素
end()	返回一个迭代器，指向最后一个元素之后



半开区间[beg, end)的好处：

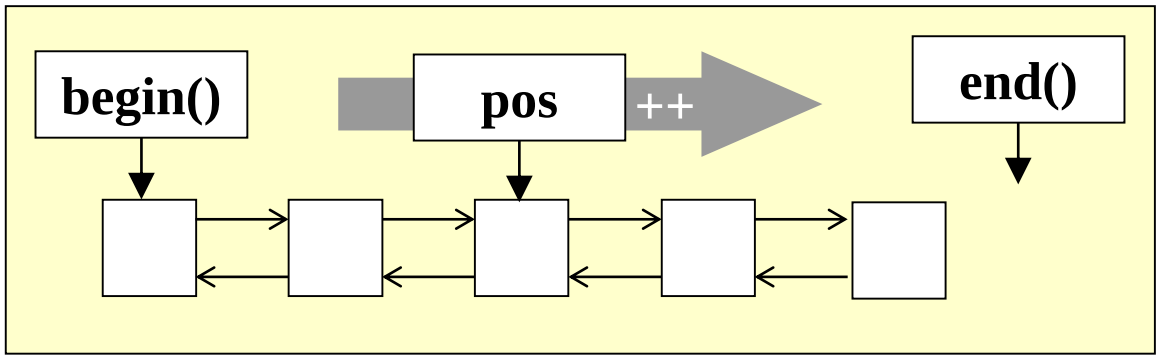
1.为遍历元素时循环的结束时机提供了简单的判断依据（只要未到达end()，循环就可以继续）

2.不必对空区间采取特殊处理（空区间的begin()就等于end()）

STL概述

迭代器（iterator）

- 所有容器都提供两种迭代器
  - container::iterator以“读/写”模式遍历元素
  - container::const_iterator以“只读”模式遍历元素
- 迭代器示例：iterator



STL概述

迭代器（iterator）

- 迭代器分类

```
list<int> l;
for(pos=l.begin();pos!=l.end();++pos{
        …
}
```

– 双向迭代器

可以双向行进,以递增运算前进或以递减运算后退、可以用==和!=比较。

list、set和map提供双向迭代器

– 随机存取迭代器

除了具备双向迭代 所有属性，还具备随机访问能力。

```
vector<int> v;
for(pos=v.begin();pos<v.end();++pos{
        …
}
```

处理迭代器之间的
较两个迭代器。
迭代器

STL容器

vector

- vector模拟动态数组
- vector的元素可以是任意类型T，但必须具备赋值和拷贝能力（具有public拷贝构造函数和重载的赋值操作符）
- 必须包含的头文件#include <vector>
- vector支持随机存取
- vector的大小（size）和容量（capacity）
  - size返回实际元素个数，
  - capacity返回vector能容纳的元素最大数量。如果插入元素时，元素个数超过capacity，需要重新配置内部存储器。

STL容器

vector

- 构造、拷贝和析构

操作	效果
vector<T> c	产生空的**vector**
vector<T> c1(c2)	产生同类型的**c1**，并将复制**c2**的所有元素
vector<T> c(n)	利用类型**T**的默认构造函数和拷贝构造函数生成一个大小为**n**的**vector**
vector<T> c(n,e)	产生一个大小为**n**的**vector**，每个元素都是**e**
vector<T> c(beg,end)	产生一个**vector**，以区间[beg,end]为元素初值
~vector<T>()	销毁所有元素并释放内存。

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL容器

vector

- 非变动操作

操作	效果
c.size()	返回元素个数
c.empty()	判断容器是否为空
c.max_size()	返回元素最大可能数量（固定值）
c.capacity()	返回重新分配空间前可容纳的最大元素数量
c.reserve(n)	扩大容量为n
c1==c2	判断c1是否等于c2
c1!=c2	判断c1是否不等于c2
c1<c2	判断c1是否小于c2
c1>c2	判断c1是否大于c2
c1<=c2	判断c1是否大于等于c2
c1>=c2	判断c1是否小于等于c2

STL容器

vector

- 赋值操作

操作	效果
c1 = c2	将c2的全部元素赋值给c1
c.assign(n,e)	将元素e的n个拷贝赋值给c
c.assign(beg,end)	将区间[beg;end]的元素赋值给c
c1.swap(c2)	将c1和c2元素互换
swap(c1,c2)	同上，全局函数

```
std::list<T> l;
std::vector<T> v;
…
v.assign(l.begin(),l.end());
```

所有的赋值操作都有可能调用元素类型的默认构造函数，拷贝构造函数，赋值操作符和析构函数

STL容器

vector

- 元素存取

操作	效果
at(idx)	返回索引idx所标识的元素的引用，进行越界检查
operator [](idx)	返回索引idx所标识的元素的引用，不进行越界检查
front()	返回第一个元素的引用，不检查元素是否存在
back()	返回最后一个元素的引用，不检查元素是否存在

```
std::vector<T> v;//empty

v[5]= t;                        //runtime error
std::cout << v.front();         //runtime error
```


信息学院 江志英 jiangzy@mail.buct.edu.cn

STL容器

vector

- 迭代器相关函数

操作	效果
begin()	返回一个迭代器，指向第一个元素
end()	返回一个迭代器，指向最后一个元素之后
rbegin()	返回一个逆向迭代器，指向逆向遍历的第一个元素
rend()	返回一个逆向迭代器，指向逆向遍历的最后一个元素

迭代器持续有效，除非发生以下两种情况：
（1）删除或插入元素
（2）容量变化而引起内存重新分配

STL容器

vector

- 安插（insert）元素

操作	效果
c.insert(pos,e)	在**pos**位置插入元素**e**的副本，并返回新元素位置
c.insert(pos,n,e)	在**pos**位置插入**n**个元素**e**的副本
c.insert(pos,beg,end)	在**pos**位置插入区间**[beg;end]**内所有元素的副本
c.push_back(e)	在尾部添加一个元素**e**的副本

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL容器

vector

- 移除（remove）元素

操作	效果
c.pop_back()	移除最后一个元素但不返回最后一个元素
c.erase(pos)	删除**pos**位置的元素，返回下一个元素的位置
c.erase(beg,end)	删除区间**[beg;end]**内所有元素，返回下一个元素的位置
c.clear()	移除所有元素，清空容器
c.resize(num)	将元素数量改为**num**（增加的元素用**defalut**构造函数产生，多余的元素被删除）
c.resize(num,e)	将元素数量改为**num**（增加的元素是**e**的副本）

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL容器

vector

- vector应用实例：vector

STL容器

deque

- deque模拟动态数组
- deque的元素可以是任意类型T，但必须具备赋值和拷贝能力（具有public拷贝构造函数和重载的赋值操作符）
- 必须包含的头文件#include <deque>
- deque支持随机存取
- deque支持在头部和尾部存储数据
- deque不支持capacity和reserve操作

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL容器

deque

- 构造、拷贝和析构

操作	效果
decque<T> c	产生空的**deque**
decque<T> c1(c2)	产生同类型的**c1**，并将复制**c2**的所有元素
decque<T> c(n)	利用类型**T**的默认构造函数和拷贝构造函数生成一个大小为**n**的**deque**
decque<T> c(n,e)	产生一个大小为**n**的**deque**，每个元素都是**e**
decque<T> c(beg,end)	产生一个**deque**，以区间[beg,end]为元素初值
~decque<T>()	销毁所有元素并释放内存。

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL容器

deque

- 非变动操作

操作	效果
c.size()	返回元素个数
c.empty()	判断容器是否为空
c.max_size()	返回元素最大可能数量（固定值）
c1==c2	判断**c1**是否等于**c2**
c1!=c2	判断**c1**是否不等于**c2**
c1<c2	判断**c1**是否小于**c2**
c1>c2	判断**c1**是否大于**c2**
c1<=c2	判断**c1**是否大于等于**c2**
c1>=c2	判断**c1**是否小于等于**c2**

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL容器

deque

- 赋值操作

操作	效果
c1 = c2	将c2的全部元素赋值给c1
c.assign(n,e)	将元素e的n个拷贝赋值给c
c.assign(beg,end)	将区间[beg;end]的元素赋值给c
c1.swap(c2)	将c1和c2元素互换
swap(c1,c2)	同上，全局函数

```
std::list<T> l;
std::deque<T> v;
…
v.assign(l.begin(),l.end());
```

所有的赋值操作都有可能调用元素类型的默认构造函数，拷贝构造函数，赋值操作符和析构函数

STL容器

deque

- 元素存取

操作	效果
at(idx)	返回索引idx所标识的元素的引用，进行越界检查
operator [](idx)	返回索引idx所标识的元素的引用，不进行越界检查
front()	返回第一个元素的引用，不检查元素是否存在
back()	返回最后一个元素的引用，不检查元素是否存在

```
std::deque<T> dq;//empty

dq[5]= t;                          //runtime error
std::cout << dq.front();   //runtime error
```


信息学院 江志英 jiangzy@mail.buct.edu.cn

STL容器

deque

- 迭代器相关函数

操作	效果
begin()	返回一个迭代器，指向第一个元素
end()	返回一个迭代器，指向最后一个元素之后
rbegin()	返回一个逆向迭代器，指向逆向遍历的第一个元素
rend()	返回一个逆向迭代器，指向逆向遍历的最后一个元素

迭代器持续有效，除非发生以下两种情况：
（1）删除或插入元素
（2）容量变化而引起内存重新分配

STL容器

deque

- 安插（insert）元素

操作	效果
c.insert(pos,e)	在**pos**位置插入元素**e**的副本，并返回新元素位置
c.insert(pos,n,e)	在**pos**位置插入**n**个元素**e**的副本
c.insert(pos,beg,end)	在**pos**位置插入区间**[beg;end]**内所有元素的副本
c.push_back(e)	在尾部添加一个元素**e**的副本
c.push_front(e)	在头部添加一个元素**e**的副本

STL容器

deque

- 移除（remove）元素

操作	效果
c.pop_back()	移除最后一个元素但不返回最后一个元素
c.pop_front()	移除第一个元素但不返回第一个元素
c.erase(pos)	删除**pos**位置的元素，返回下一个元素的位置
c.erase(beg,end)	删除区间**[beg;end]**内所有元素，返回下一个元素的位置
c.clear()	移除所有元素，清空容器
c.resize(num)	将元素数量改为**num**（增加的元素用**defalut**构造函数产生，多余的元素被删除）
c.resize(num,e)	将元素数量改为**num**（增加的元素是**e**的副本）

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL容器

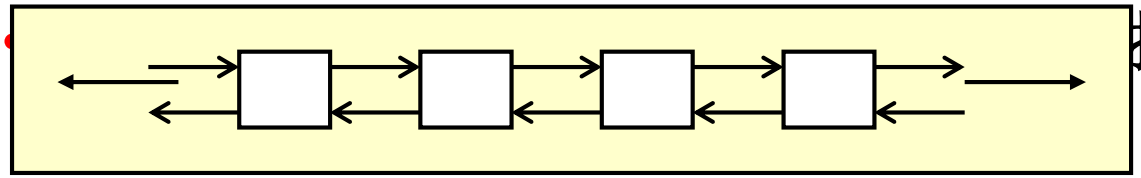
deque

- **deque应用实例：** **deque**

STL容器

list

- 使用双向链表管理元素
- **list**的元素可以是任意类型**T**，但必须具备赋值和拷贝能力
- 必须包含的头文件#include <list>
- **list**不支持随机存取，因此不提供下标操作符
- 在任何位置上执行元素的安插和移除都非常快。
- 指针、引用、



信息学院 江志英 jiangzy@mail.buct.edu.cn

STL容器

list

- 构造、拷贝和析构

操作	效果
list<T> c·	产生空的**list**
list<T> c1(c2)	产生同类型的**c1**，并将复制**c2**的所有元素
list<T> c(n)	利用类型**T**的默认构造函数和拷贝构造函数生成一个大小为**n**的**list**
list<T> c(n,e)	产生一个大小为**n**的**list**，每个元素都是**e**
list<T> c(beg,end)	产生一个**list**，以区间**[beg,end]**为元素初值
~list<T>()	销毁所有元素并释放内存。

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL容器

list

- 非变动性操作

操作	效果
c.size()	返回元素个数
c.empty()	判断容器是否为空
c.max_size()	返回元素最大可能数量
c1==c2	判断c1是否等于c2
c1!=c2	判断c1是否不等于c2
c1<c2	判断c1是否小于c2
c1>c2	判断c1是否大于c2
c1<=c2	判断c1是否大于等于c2
c1>=c2	判断c1是否小于等于c2

STL容器

list

- 赋值

操作	效果
c1 = c2	将**c2**的全部元素赋值给**c1**
c.assign(n,e)	将**e**的**n**个拷贝赋值给**c**
c.assign(beg,end)	将区间**[beg;end]**的元素赋值给**c**
c1.swap(c2)	将**c1**和**c2**的元素互换
swap(c1,c2)	同上，全局函数

STL容器

list

- 元素存取

操作	效果
front()	返回第一个元素的引用，不检查元素是否存在
back()	返回最后一个元素的引用，不检查元素是否存在

```
std::list<T> l;//empty

std::cout << l.front();      //runtime error
if(!l.empty()){
        std::cout<<l.back();        //ok
}
```

STL容器

list

- 迭代器相关函数

操作	效果
begin()	返回一个双向迭代器，指向第一个元素
end()	返回一个双向迭代器，指向最后一个元素之后
rbegin()	返回一个逆向迭代器，指向逆向遍历的第一个元素
rend()	返回一个逆向迭代器，指向逆向遍历的最后一个元素

STL容器

list

- 安插（insert）元素

操作	效果
c.insert(pos,e)	在**pos**位置插入**e**的副本，并返回新元素位置
c.insert(pos,n,e)	在**pos**位置插入**n**个**e**的副本
c.insert(pos,beg,end)	在**pos**位置插入区间**[beg;end]**内所有元素的副本
c.push_back(e)	在尾部添加一个**e**的副本
c.push_front(e)	在头部添加一个**e**的副本

STL容器

list

- 移除（remove）元素

操作	效果
c.pop_back()	移除最后一个元素但不返回
c.pop_front()	移除第一个元素但不返回
c.erase(pos)	删除**pos**位置的元素，返回下一个元素的位置
c.remove(val)	移除所有值为**val**的元素
c.remove_if(op)	移除所有“**op(val)==true**”的元素
c.erase(beg,end)	删除区间**[beg;end]**内所有元素，返回下一个元素的位置
c.clear()	移除所有元素，清空容器
c.resize(num)	将元素数量改为**num**（多出的元素用**defalut**构造函数产生）
c.resize(num,e)	将元素数量改为**num**（多出的元素是**e**的副本）

STL容器

list

- 特殊变动性操作

操作	效果
c.unique	移除重复元素，只留下一个
c.unique(op)	移除使op()结果为true的重复元素
c1.splice(pos,c2)	将c2内的所有元素转移到c1的迭代器pos之前
c1.splice(pos,c2,c2pos)	将c2内c2pos所指元素转移到c1内的pos之前
c1.splice(pos,c2,c2beg,c2end)	将c2内[c2beg;c2end)区间内所有元素转移到c2的pos之前

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL容器

list

- 特殊变动性操作（续）

操作	效果
c.sort()	以**operator <**为准则对所有元素排序
c.sort(op)	以**op**为准则对所有元素排序
c1.merge(c2)	假设**c1**和**c2**都已排序，将**c2**全部元素转移到**c1**并保证合并后**list**仍为已排序
c.reverse()	将所有元素反序

STL容器

list

- list应用实例：list

STL容器

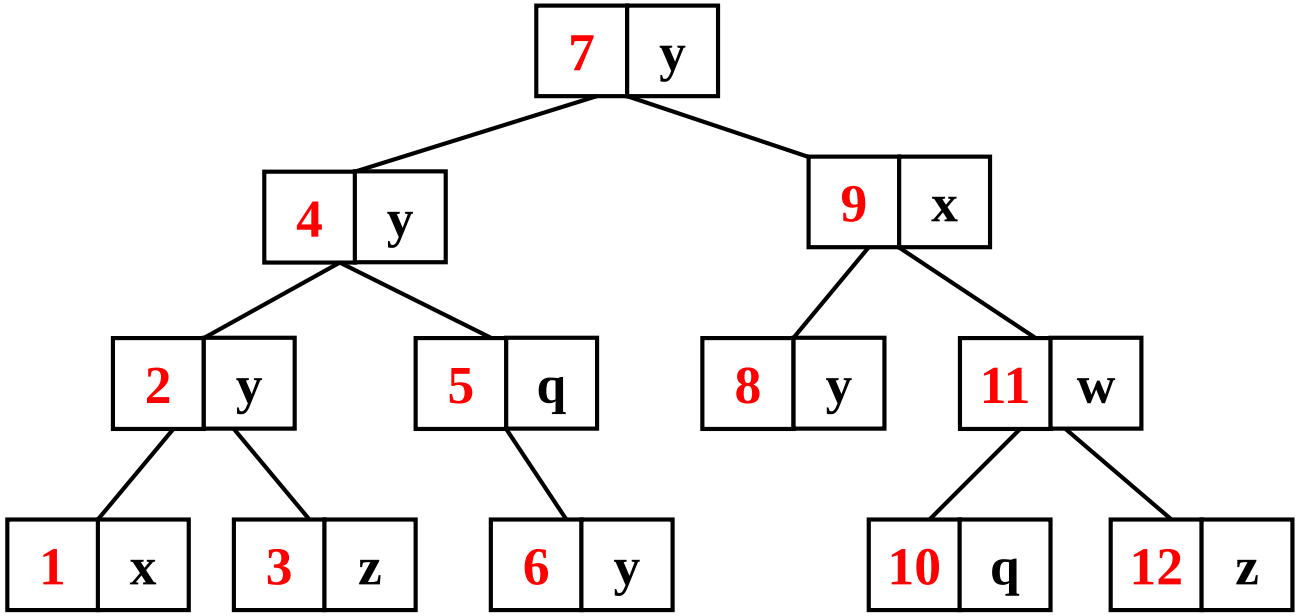
map/multimap

- 使用平衡二叉树管理元素
- 元素包含两部分(key,value)，key和value可以是任意类型
- 必须包含的头文件#include <map>
- 根据元素的key自动对元素排序，因此根据元素的key进行定位很快，但根据元素的value定位很慢
- 不能直接改变元素的key，可以通过operator []直接存取元素值
- map中不允许key相同的元素，multimap允许key相同的元素

STL容器

map/multimap

- 内部存储结构



STL容器

map/multimap

- 构造、拷贝和析构

操作	效果
map c	产生空的**map**
map c1(c2)	产生同类型的**c1**，并复制**c2**的所有元素
map c(op)	以**op**为排序准则产生一个空的**map**
map c(beg,end)	以区间**[beg,end]**内的元素产生一个**map**
map c(beg,end,op)	以**op**为排序准则，以区间**[beg,end]**内的元素产生一个**map**
~ *map*()	销毁所有元素并释放内存。

其中*map*可以是下列形式

map<key,value>	一个以**less**（**<**）为排序准则的**map**，
map<key,value,op>	一个以**op**为排序准则的**map**

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL容器

map/multimap

- 非变动性操作

操作	效果
c.size()	返回元素个数
c.empty()	判断容器是否为空
c.max_size()	返回元素最大可能数量
c1==c2	判断**c1**是否等于**c2**
c1!=c2	判断**c1**是否不等于**c2**
c1<c2	判断**c1**是否小于**c2**
c1>c2	判断**c1**是否大于**c2**
c1<=c2	判断**c1**是否大于等于**c2**
c1>=c2	判断**c1**是否小于等于**c2**

STL容器

map/multimap

- 赋值

操作	效果
c1 = c2	将c2的全部元素赋值给c1
c1.swap(c2)	将c1和c2的元素互换
swap(c1,c2)	同上，全局函数

STL容器

map/multimap

- 特殊搜寻操作

操作	效果
count(key)	返回”键值等于**key**”的元素个数
find(key)	返回”键值等于**key**”的第一个元素，找不到返回**end**
lower_bound(key)	返回”键值大于等于**key**”的第一个元素
upper_bound(key)	返回”键值大于**key**”的第一个元素
equal_range(key)	返回”键值等于**key**”的元素区间

STL容器

map/multimap

- 迭代器相关函数

操作	效果
begin()	返回一个双向迭代器，指向第一个元素
end()	返回一个双向迭代器，指向最后一个元素之后
rbegin()	返回一个逆向迭代器，指向逆向遍历的第一个元素
rend()	返回一个逆向迭代器，指向逆向遍历的最后一个元素

STL容器

map/multimap

- 安插（insert）元素

操作	效果
c.insert(pos,e)	在pos位置为起点插入e的副本，并返回新元素位置（插入速度取决于pos）
c.insert(e)	插入e的副本，并返回新元素位置
c.insert(beg,end)	将区间[beg;end]内所有元素的副本插入到c中

STL容器

map/multimap

- 移除（remove）元素

操作	效果
c.erase(pos)	删除迭代器**pos**所指位置的元素，无返回值
c.erase(val)	移除所有值为**val**的元素，返回移除元素个数
c.erase(beg,end)	删除区间**[beg;end]**内所有元素，无返回值
c.clear()	移除所有元素，清空容器

STL容器

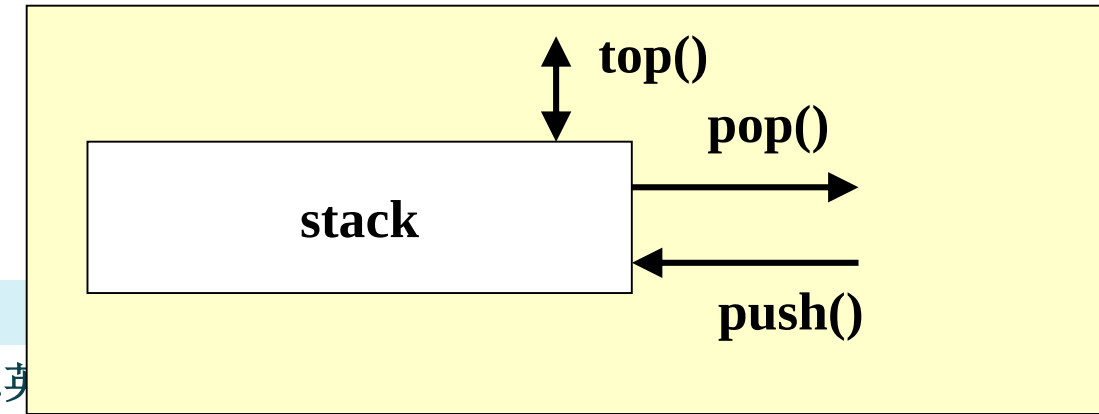
map/multimap

- map应用实例：map

STL容器

stack（实例：stack ）

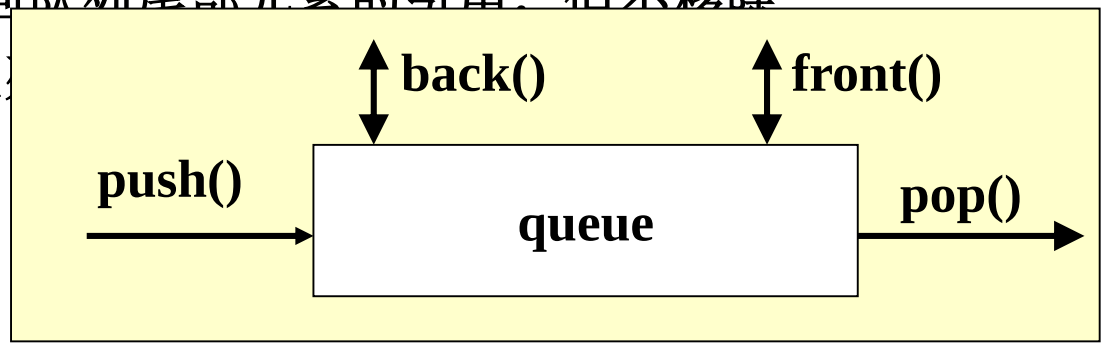
- 后进先出（**LIFO**）
- #include <stack>
- 核心接口
  - push(value)—将元素压栈
  - top()—返回栈顶元素的引用，但不移除
  - pop()—从栈中移除栈顶元素，但不返回
- 实例：**stack**



STL容器

queue（实例：queue ）

- 先进先出（FIFO）
- #include <queue>
- 核心接口
  - push(e)—将元素置入队列
  - front()—返回队列头部元素的引用，但不移除
  - back()—返回队列尾部元素的引用，但不移除
  - pop()—从队
- 实例：queue



STL容器

priority_queue （实例：priority_queue ）

- 以某种排序准则（默认为less）管理队列中的元素
- #include <queue>
- 核心接口
  - push(e)—根据元素的优先级将元素置入队列
  - top()—返回优先队列头部最大的元素的引用，但不移除
  - pop()—从栈中移除最大元素，但不返回
  - empty() —队列是否为空

STL算法

STL提供了一些标准算法来处理容器内的元素

- 搜寻、排序、拷贝、数值运算

STL的算法是全局函数

- 明确划分数据和操作
- 泛型函数式编程模式
- 所有算法可以对所有容器适用，甚至可以操作不同类型容器的元素

算法头文件

- #include <algorithm>
- #include <numeric>

STL算法实例：**algorithm**

STL算法

区间（range）

- 所有算法都用来处理一个或多个区间内的元素。
- 区间可以但不一定涵盖容器内所有元素
- 为了操作元素的某个子集必须将区间的首尾（iterator）当作两个参数传递给算法
- 调用时必须确保区间有效性
  - 从起点出发，逐一前进，能够到达终点。
  - 区间首尾两个迭代器必须属于同一容器，且前后放置正确
  - 无效区间可能会引起无限循环或者内存非法访问
- 所有算法处理的都是半开区间[begin, end)

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL算法

STL算法分类

- 非变动性算法（**nonmodifying algorithms**）
- 变动性算法（**modifying algorithms**）
- 移除性算法（**removing algorithms**）
- 变序性算法（**mutating algorithms**）
- 排序性算法（**sorting algorithms**）
- 已序区间算法（**sorted range algorithms**）
- 数值算法（**numeric algorithms**）

STL算法

for_each()算法

- **for_each(InputIterator beg, InputIterator end, UnaryProc op)**
- **对区间[beg, end)中的每一个元素调用op(elem)**
- **返回op之后的容器副本**
- **op可以改变元素**
- **op的返回值被忽略**
- **复杂度：O(n)**
- **示例:foreach**

STL算法

非变动性算法

名称	作用
count()	返回元素个数
count_if()	返回满足某一条件的元素个数
min_element()	返回最小元素（以迭代器表示）
max_element()	返回最大元素（以迭代器表示）
find()	搜寻等于某值的第一个元素
find_if()	搜寻满足某个准则的第一个元素
search_n()	搜寻具有某种特性的第一段“n个连续元素”
search()	搜寻某个区间第一次出现的位置
…	…

既不改变元素次序也不改变元素值

STL算法

非变动性算法

- 元素计数
- **count(InputIterator beg,InputIterator end, const T& value)**
  - 计算区间中值等于**value**的元素个数
- **count(InputIterator beg,InputIterator end, Predicate op)**
  - 计算区间中使判断式**op**结果为**true**的元素个数
  - op接受单个参数，返回值为bool型
- 复杂度：**O(n)**
- 示例:**count**

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL算法

非变动性算法

- 最小值和最大值
- **min_element(InputIterator beg,InputIterator end)**
- **min_element(InputIterator beg,InputIterator end, CompFunc op)**
- **max_element(InputIterator beg,InputIterator end)**
- **max_element(InputIterator beg,InputIterator end, CompFunc op)**
  - 返回区间中最大或最小元素的位置（迭代器）
  - 无op参数的版本以<（“小于”运算符）进行比较
  - op用来比较两个元素：**bool op(elem1,elem2)**，如果**elem1**“小于”**elem2**返回**true**否则返回**false**
- 复杂度： **O(n)**
- 示例:**minmax**

STL算法

非变动性算法

- 搜寻元素
- **find (InputIterator beg,InputIterator end, const T& value)**
  - 返回区间中第一个“元素值等于**value**”的元素位置
- **find_if (InputIterator beg,InputIterator end, Predicate op)**
  - 返回区间中第一个“使**op**结果为**true**”的元素位置
- 如果没有找到匹配元素，返回**end**
- 复杂度： **O(n)**
- 示例:**find**

STL算法

变动性算法

- 直接改变元素值或者在复制到另一区间过程中改变元素值

名称	作用
copy()	从第一个元素开始正向复制某段区间
copy_backward()	从最后一个元素开始反向复制某段区间
transform()	变动（并复制）元素，将两个区间的元素合并
merge()	合并两个区间
swap_ranges()	交换两个区间的元素
fill()	以给定值替换每个元素
fill_n()	以给定值替换**n**个元素
generate()	以某项操作的结果替换每个元素
…	…

STL算法

变动性算法

- **copy(s_beg, s_end, d_beg) －将[s_beg,s_end)区间内的元素复制到d_beg位置之后**
- **copy_backword(s_beg, s_end, d_end)－将[s_beg,s_end)区间内的元素复制到dend位置之前**
- 复杂度：**O(n)**
- 示例:**copy**

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL算法

移除性算法

- 移除某区间内的元素或者在复制过程中移除元素值

名称	作用
remove()	将等于某个特定值的元素全部移除
remove_if()	将满足某准则的元素全部移除
remove_copy()	将不等于某特定值的元素全部复制到其他地方
remove_copy_if()	将不满足某准则的元素全部复制到其他地方
unique()	移除相邻的重复元素
unique_copy()	移除相邻的重复元素，并复制到其他地方

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL算法

移除性算法

- **remove(beg,end,value)**—移除区间**[beg,end)**内和**value**相等的元素
- **remove_if(beg,end,op)**—移除区间**[beg,end)**内使操作**op**为**true**的元素
- **remove**和**remove_if**只是将未移除元素向前移动，覆盖移除元素，并返回新的终点，并没有真正删除元素，真正删除元素需要使用**erase**
- 复杂度：**O(n)**
- 示例:**remove**

STL算法

变序性算法

- 通过元素的赋值和交换改变元素顺序

名称	作用
reverse()	将元素的次序逆转
reverse_copy()	复制的同时逆转元素次序
rotate()	旋转元素次序
rotate_copy()	复制的同时旋转元素次序
random_shufle()	将元素次序随机打乱
partion()	改变元素次序使“符合某准则”的元素移到前面
stable_partion()	与**partion**类似，但保持符合准则与不符合准则元素的相对位置
…	…

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL算法

变序性算法

- 逆转元素次序
  - **reverse(beg, end)**—将**[beg, end)**区间内的元素逆序
  - **reverse_copy(s_beg, s_end, d_beg)**—将**[s_beg, s_end)**区间内的元素逆序后拷贝到从**d_beg**开始的区间
  - 示例：**reverse**
- 旋转元素次序
  - **rotate(first, middle, last)**—将**[first, last)**区间内的元素，从**middle**位置分为 **[first,middle)**和**[middle,last)**两部分，将两部分交换位置
  - 示例：**rotate**
- 复杂度：**O(n)**

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL算法

排序算法

- 需要动用随机存取迭代器

名称	作用
sort()	对所有元素排序
stable_sort()	对所有元素排序，并保持相等元素间的相对次序
partial_sort()	排序，直到前**n**个元素就位
nth_element()	根据第**n**个位置排序
make_heap()	将区间转换为**heap**
push_heap()	将一个元素加入**heap**
pop_heap()	从**heap**中移除一个元素
sort_heap()	对**heap**进行排序（排序后不再是**heap**）
…	…

信息学院 江志英 **jiangzy**@mail.buct.edu.cn

STL算法

排序算法

- 对所有元素排序
  - **sort(beg, end)**
  - **sort(beg, end, op)**

  } **经过优化的快速排序算法**
  - **stable_sort(beg, end)**
  - **stable_sort(beg, end, op)**

  } **归并排序算法**
  - **不带op参数的版本使用 < (“小于”运算符)对区间[beg, end)内的所有元素排序**
  - **带op参数的版本使用op(elem1,elem2)为准则对区间[beg, end)内的所有元素排序**
  - **sort和stable_sort的区别是，后者保持相等元素原来的相对次序**
  - **不能对list调用这些算法，因为list不支持随机存取迭代器**
  - **复杂度：O(nlogn)**
- 示例：**sort**

STL算法

排序算法

- 局部排序
  - **partial_sort(beg, sortEnd,end)**
  - **partial_sort(beg, sortEnd,end,op)**

  } **堆排序算法**
  - 不带**op**参数的版本使用 < (“小于”运算符)对区间**[beg, end)**内的元素排序，使区间**[beg, sortEnd)**内的元素有序
  - 带**op**参数的版本使用op(elem1,elem2)对区间**[beg, end)**内的元素排序，使区间**[beg, sortEnd)**内的元素有序
  - 复杂度： **O(n)**和**O(nlogn)**之间
- 示例：**psort**

信息学院 江志英 **jiangzy**@mail.buct.edu.cn

STL算法

排序算法

- 根据第n个位置排序
  - nth_element (beg, nth, end)
  - nth_element(beg, nth, end, op)

  } 快速排序算法
  - 对区间[beg, end)内的元素排序，使所有在位置n之前的元素都小于等于它，所有在位置n之后的元素都大于等于它，从而得到两个分隔开的子序列，但子序列并不一定有序。
  - 不带op参数的版本使用<运算符作为排序准则
  - **带op参数的版本使用op(elem1,elem2)作为排序准则**
  - 复杂度：O(n)
- 示例：nsort

信息学院 江志英 jiangzy@mail.buct.edu.cn

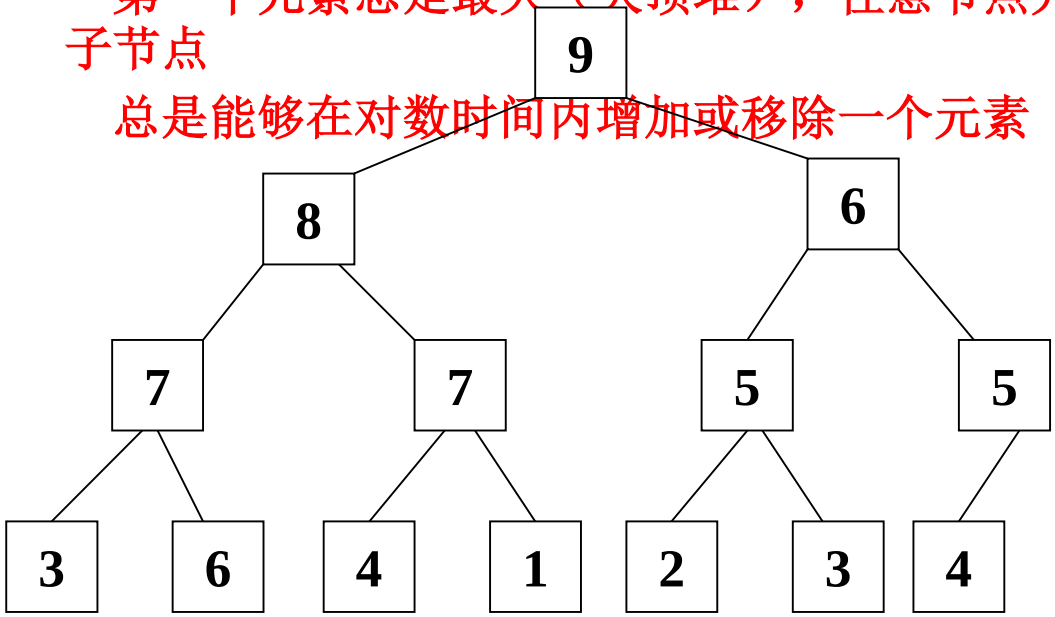
STL算法

排序算法

- 堆排序算法（ heap sort）
  - heap是一种特殊的元素组织方式，可以被视为完全二叉树

第一个元素总是最大（大顶堆），任意节点元素大于其孩子节点

总是能够在对数时间内增加或移除一个元素



STL算法

排序算法

- **heap算法**
  - **make_heap()－将某区间内的元素转化为heap，复杂度O(n)**
  - **push_heap(beg, end)－[beg, end-1)原本就是heap，将end之前的那个元素加入使区间[beg, end)]重新成为heap，复杂度O(logn)**
  - **pop_heap(beg, end)－从区间[beg, end)取出第一个元素，放到最后位置，区间[beg, end-1) 重新组成heap，复杂度O(logn)**
  - **sort_heap()－将heap转换为一个有序集合，复杂度O(nlogn)**
  - **可以用<运算符或op(elem1,elem2)作为排序准则**
- 示例：**heap**

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL算法

已序区间算法

- 所作用的区间以按某种准则排序

名称	作用
binary_serach()	判断区间内是否包含某个元素
includes()	判断区间内的每个元素是否都包含在另一个区间中
lower_bound()	搜寻第一个“大于等于给定值”的元素
upper_bound()	搜寻第一个“大于给定值”的元素
equal_range()	返回“等于给定值”的元素构成的区间
merge()	将两个区间的元素合并
set_union()	求两个区间的并集
set_intersection()	
set_difference()	求两个区间的差集
…	…

STL算法

已序区间算法

- **bool binary_serach (beg, end, value)**
- **bool binary_serach (beg, end, value, op)**
- 判断已序区间**[beg, end)**内是否包含“和**value**相等”的元素
- **op**可以作为排序准则
- 返回值只说明搜寻的值是否存在，不指明位置
- 调用者必须确保区间已序
- 复杂度：使用随机存取迭代器为**O(logn)**，否则是**O(n)**
- 示例：**bserach**

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL算法

已序区间算法

- lower_bound (beg, end, value)
- upper_bound (beg, end, value)
- lower_bound (beg, end, value,op)
- upper_bound (beg, end, value,op)
- lower_bound()返回第一个大于等于value的元素位置，既可以插入value且不破坏区间已序性的第一个位置
- upper_bound()返回第一个大于value的元素位置，既可以插入value且不破坏区间已序性的最后一个位置
- op可以作为排序准则
- 调用者必须确保区间已序
- 复杂度：使用随机存取迭代器为O(logn)，否则是O(n)
- 示例：bound

信息学院 江志英 jiangzy@mail.buct.edu.cn

STL算法

数值算法

- 以不同的方式组合数值元素
- #include <numeric>

名称	作用
accumulate()	组合所有元素（求和，求积，...）
inner_product()	组合两区间内的所有元素的内积
adjacent_difference()	将每个元素和其前一元素组合
partial_sum()	将每个元素和其先前所有元素组合

STL算法

数值算法

- **accumulate(beg, end, initValue)**
- **accumulate(beg, end, initValue,op)**
- 对于序列：**a1, a2, a3, …**
  - 第一种形式计算**initValue + a1 + a2 + a3 + …**
  - 第二种形式计算**initValue op a1 op a2 op a3 op …**
- 示例：**accu**

STL算法

数值算法

- **inner_product(beg1, end1, beg2, initValue)**
- **inner_product(beg1, end2, beg2,initValue,op1,op2)**
- 对于序列：**a1, a2, a3, …;b1, b2, b3, …**
  - 第一种形式计算**initValue + (a1*b1) + (a2*b2) + (a3*b3) + …**
  - 第二种形式计算**initValue op1 (a1 op2 b1) op1 (a2 op2 a2) op …**
- 示例：**inner**

参考书籍

数据结构

C++标准程序库