五、容器、迭代器和泛型算法

1.概念

1)容器：利用类模板构建的一组泛型化的数据结构。

2)泛型算法：利用函数模板构建的一组与元素和容器的类型无关的通用算法。

3)迭代器：利用操作符重载为不同的容器提供接口一致的元素访问方式，协助泛型算法做到容器无关。

迭代器又叫游标卡尺，模仿连续内存的访问模式(地址++)。

2.实现案例

双向线性链表

正向顺序可写迭代器

正向：增向尾，减向首

顺序：一次只能迭代一个元素

可写：目标元素可被修改

线性查找

参见：list.cpp

==========================

六、标准模板库(Standard Templates Library, STL)

1.概述

1)十大容器

A.线性容器：向量(vector)、双端队列(deque)、链表(list)

B.适配器容器：堆栈(stack)、队列(queue)、优先队列(priority\_queue)

C.关联容器：映射(map)、多重映射(multimap)、集合(set)、多重集合(multiset)

2)四大迭代器

A.正向可写迭代器

B.正向只读迭代器

C.反向可写迭代器

D.反向只读迭代器

只有向量(vector)和双端队列(deque)这两种连续内存的容器才会提供随机迭代器，其它内存不连续的容器都只提供顺序迭代器。

随机：相对于顺序，能跳跃多个元素访问

3)若干通用算法

常用非数值算法，如：查找、排序、拆分、合并、复制、交换，等等。

-------------------------

2.向量(vector)(动态分配的数组)

1)基本特性

A.连续内存，下标访问。

B.动态内存管理，静态预分配。

2)实例化

#include <vector>

using namespace std;

A.空向量

vector<元素类型> 向量对象;

vector<int> vi;

大小为0，元素数为0，向量对象的字节大小是一个常量——12字节

补充：12个字节分别存放，向量地址，可有元素数和已有元素数。

B.初始大小

vector<元素类型> 向量对象 (元素数);

vector<int> vi(5); // 0 0 0 0 0

基本类型元素，用"0"初始化，类类型元素，用缺省构造函数初始化。

C.初始值

vector<元素类型> 向量对象 (元素数, 初值);

vector<int> vi(5, 10); // 10 10 10 10 10

D.范围初始化

vector<元素类型> 向量对象 (起始迭代器, 终止迭代器);

int ai[5] = {1, 2, 3, 4, 5};

vector<int> vi(ai, ai + 5); // 1 2 3 4 5

vector<int> vi(&ai[0], &ai[5]);

--------------------------

注意在STL中用两个迭代器表示一个范围时，其终止迭代器永远是该范围中最后一个元素下一个位置的迭代器。

1 2 3 [4 5 6 7] 8 9 10

^ ^

| |

起始迭代器 终止迭代器

-------------------------

3)迭代器

A.四个迭代器类型

正向可写迭代器：iterator

正向只读迭代器：const\_iterator

反向可写迭代器：reverse\_iterator

反向只读迭代器：const\_reverse\_iterator

B.八个特征迭代器

begin(void); // 正向可写起始迭代器

end(void); // 正向可写终止迭代器

begin(void)const;

end(void)const;

rbegin(void);

rend(void);

rbegin(void)const;

rend(void)const;

--------------------

begin - 起始

end - 终止

不带const - 可写

带const - 只读

不带r - 正向

带r - 反向

对于正向迭代器而言，起始位置在首元素，终止位置在尾元素之后；对于反向迭代器而言，起始位置在尾元素，终止位置在首元素之前。

-------------------

C.随机迭代器

所有顺序迭代器的功能随机迭代器都支持，同时还提供更多功能：和整数加减运算、同型迭代器之间的大小比较和相减运算

D.任何可能导致容器内存布局发生变化的结构性改变，如增删元素等，都可能使原先获得的迭代器失效，因此稳妥起见，在这样的改变发生以后应该重新获得反应新内存布局的迭代器，保证元素访问正确。

参见：vec2.cpp

4)成员函数

front/back

push\_back/pop\_back(没有push\_front/pop\_front)

insert/erase

size/empty/clear

begin/end

rbegin/rend

operator[]

所有的STL都有支持深拷贝的拷贝构造和拷贝赋值

5)全局函数(只有内存连续的容器，才会有全局的排序查找算法函数)

A.排序

template<typename IT>

void sort(IT begin, IT end);

对[begin,end)区间内的元素做快速排序。

在排序过程中，通过小于运算符比大小。(默认<, 从小到大排序)

template<typename IT, typename LESS>

void sort(IT begin, IT end, LESS less);

在排序过程中，通过比较器(自定义比较规则)比大小。

小于比较器可以是函数指针也可以是仿函数。

参见：vec3.cpp

B.查找

template<typename IT, typename KEY>

IT find(IT begin, IT end, KEY key);

如果查找成功返回在[begin,end)区间内第一个和key匹配的元素的迭代器，否则返回end。容器中元素的类型必须支持"=="运算符。

---------------------------------------------

3.双端队列(deque)

相对与向量，双端队列唯一的区别就是首端和尾端同样都是开放的，因此它同时提供在首尾两端增删元素的接口。

push\_front/pop\_front

push\_back/pop\_back

双端队列的内存开销和时间性能上略逊于向量。