**STL笔记**

**string**

string是动态数组，头文件#include <string>

空间扩大的时候会申请新的空间，这时迭代器会失效

构造函数

string str(); 无参构造

string str(5, 'a'); 用五个a初始化

string str("abcdef"); 用字符串初始化

string str("abcdef", 3); 用前三个字符初始化

string str(const string& str1); 用拷贝构造初始化

string str(string& str1, int index, int lenth); 用下标从[index,lenth-1]初始化

运算符重载

<< 比如cout << str会输出str的字符

>> 比如cin >> str输入的内容会将str原来的内容全部覆盖掉

[ ] 下标运算符[]也重载了

< > >= <= == != 用来比较两个string对象或者一个string对象跟字符串

+ += =

成员函数

c\_str 返回string本体字符串的地址

capacity()获得容量，初始值是15，每超一次最大值加16个字符的空间

length()用来获取字符串中字符的个数，比如"abc\0dfs"中得到的是3

size()用来获取字符串中字符的个数，比如"abc\0dfs"中得到的是3

at()的使用方式和下标运算符一样

empty()判断字符串是否是空的，是返回true，不是返回false

修改

修改指定元素可以用下标运算符和at()

比如str[1] = 's'; str.at(1) = 's';

中间插入用insert

insert(index, num, ‘a’);

insert(index, “hello”);

insert(index, “hello”, num);

insert(index, str);

insert(index, str, StrIndex, num);

insert(ite, 3, ‘n’);在ite前面插入3个n

insert(ite, iterator start, iterator end);在ite前面插入start到end前面一个那一段

尾巴插入

+= 可以拼接对象；也可以拼接字符串，比如str += “hello”; str += str1;

append(num, ‘a’);

append(“hello);

append(“hello”, num);

append(str);

append(str, StrIndex, num);

用迭代器：append(str1.begin(), str1.end());从 begin到end的前一个

赋值（会将原来的覆盖掉，相当于将原来的全部删掉再重新赋值）

用=赋值 可以赋值字符串也可以赋值对象

用>>输入赋值

assign(num , ‘a’);

assign(“hello”);

assign(“hello”, num);

assign(str);

assign(str, StrIndex, num);

删除

erase(index, num); 从index下标位置删除num个字符

erase(ite);删除迭代器指向的那个字符

erase(iterator start, iterator end);删除从start到end前一个这一段

clear();清除所有元素

比较

compare(str)和对象里面的字符串比较

compare(“hello”)和字符串比较

compare(0, 3, str)从0开始的三个字符与str比较

compare(0, 3, str, 0, 3);本对象的一段与str的一段比较

复制到外面

copy(arr, num, index)从index开始复制num个到字符数组arr里

查找

find(str, index); 从下标index的位置往后查找str，找到返回下标，没找到返回-1

find(“hello”, index);

find(‘a’, index);

返回子串

string substr(index, count); 返回从index开始的count个字符，返回值是一个string对象

交换两个对象的内容

swap(str);

迭代器（指针怎么用迭代器就怎么用）

iterator begin();返回一个指向第一个字符的迭代器

iterator end(); 返回指向最后一个元素下一个元素的迭代器，可以理解成是指向’\0’的迭代器。

算法

算法头文件#include <algorithm>

遍历：

for\_each(ite1, ite2, fun)

void fun(char ch) 遍历的动作在fun里执行,fun的参数一定是容器的元素类型

{

}

排序：

sort(ite1, ite2);默认从小到大排序

sort(ite1, ite2, greater<char>());从大到小排序就这样写，尖括号里是容器元素的类型greater是仿函数,头文件是#include <functional>

**vector(向量)**

头文件#include <vector>

vector是一个动态数组，array是一个定死了的固定数组。valarrar是用于数学计算的

空间扩大的时候会申请新的空间，这时迭代器会失效

定义

例如：vector<int> vec;参数列表里是什么这个向量就用来装什么类型。什么类型都可以，比如int char 结构体 类等

构造函数

vector<int> vec()里面没有任何内容，指针应该是指向nullptr

vector<int> vec(num)初始化num个元素,这些元素全初始化成0

vector<int> vec(num, type) 比如用5个6来初始化

vector<int> vec(vec) 用另一个对象来初始化

vector<int> vec(iteStart, iteEnd) 用迭代器来初始化

空间的增加规律

空间不够时，会增加“现有空间/2“的空间

成员函数

capacity() 返回容量

size() 返回元素个数

resize() 重新设置元素个数，容量不变

empty() 这个对象里是否有元素，有返回0，无返回1

back() 返回尾巴元素的引用

迭代器

iterator begin();

iterator end();

算法

算法头文件#include <algorithm>

遍历：

for\_each(ite1, ite2, fun)

void fun(int ch) 遍历的动作在fun里执行,fun的参数一定是容器的元素类型

{

}

排序：

sort(ite1, ite2);默认从小到大排序

sort(ite1, ite2, greater<int>());从大到小排序就这样写，尖括号里是容器元素的类型greater是仿函数,头文件是#include <functional>

运算符重载

[ ]

at()和[ ]用法一样

两个向量比较大小：< > == <= >= != 比较规则跟字符串比较规则一样

修改

尾添加：

push\_back(5) 尾添加了一个5

中间添加：

insert(ite, val); 在ite的位置添加值为val的数

insert(ite, num, val); 在ite的位置添加num个值为val的数

insert(ite, iteStart, iteEnd); 将iteStart到iteEnd前一个这一段插入到ite的位置

删除

pop\_back(); 尾删除一个元素

erase(ite); 删除ite指向的这个元素

erase(iteStart, iteEnd); 删除iteStart到iteEnd前一个这一段

clear();清除所有元素

交换两个向量的内容

swap(vec);

赋值

assign(iteStart, iteEnd);用iteStart到iteEnd前面这一段来赋值

assign(num, val); 用num个val来赋值

乱序算法

srand((unsiged int)time(nullptr)); 用random\_suffle也需要种种子

random\_shuffle(ite1, ite2); 将ite1到ite2的顺序打乱

**双向链表list**

头文件#include <list>

list的迭代器可以ite++这样，但不能ite+2这样

list不支持下标运算符[ ]

定义

例如：list<int> ls;参数列表里是什么这个list就用来装什么类型。什么类型都可以，比如int char 结构体 类等

构造函数

list<node> ls(); 无参构造，里面什么都没有

list<node> ls(count); 初始化count个元素，元素值为0或’\0’或nullptr

list<node> ls(count, val); 用count个val进行初始化

list<node> ls(ls); 用另一个对象给当前对象初始化

list<node> ls(ite1, ite2);用ite1到ite2前面一个这一段来初始化

成员函数

size(); 元素个数

resize(3);重新设置元素个数，容量不变

empty(); 判断元素是否为空

back();返回尾巴的元素

front();返回第一个元素

添加

push\_front(value);头添加

push\_back(value);尾添加

insert(ite, value);在ite的位置插入一个value

insert(ite, num, value); ite的位置插入num个value

insert(ite,iteStart, iteEnd);在ite的位置插入从iteStart到iteEnd前面一个这一段

删除

pop\_back();尾删除

pop\_front();头删除

erase(ite); 删除ite指向的这个元素

erase(iteStart, iteEnd); 删除iteStart到iteEnd前一个这一段

clear();删除全部元素

remove(value);删除与value相等的全部元素，如果是结构体，那就无法比较，因此要自己在结构体里面重载“==”运算符

unique();去重

赋值

可以用迭代器赋值

也可以用assign(num, value);用num个value来赋值 assign(ite1, ite2);用ite1到ite2前面一个这一段来赋值

也可以用法”=”来赋值

交换两个list的内容

swap(ls);

将链表反转过来

reverse();

list自带的sort

sort();如果是结构体就要重载<符号，因为sort里面涉及到比较

拼接

splice(ite, ls);将另一个对象插入到ite的位置

splice(ite, ls, ite1);将另一个对象的ite1指向的元素插入到ite的位置

splice(ite, ls, iteStart, iteEnd);将另一个元素的从iteStart到iteEnd前面一个这一段插入到ite的位置

**双端队列deque**

头文件#include <deque>

deque相较于vector没有capacity()和reserve(),多了push\_front()和push\_back();

**map**

头文件#include <map>

map是关联容器，而string，vector，list都是顺序容器

构造函数

map();构造一个空的map对象

map(mp);用另外一个map来给当前map初始化

map(ite1, ite2);用两个迭代器中间的一段来给当前map初始化

元素类型

元素是pair<key,value>类型pair是一个结构体key一般用整型，value是什么类型都行

pr.first是键 pr.second是值

插入元素

insert(value);用value来插入value是pair类型

insert(ite, value);在ite的位置插入一个value，但是无论在哪个位置插最后都会排序到它相应的位置去

insert(ite1, ite2);用两个迭代器中间的一段来插入