STL详解

1、Vector - Dynamic arrays(动态数组)

1.1 基本操作

○ 功能

定义一个可变大小的数组,空间的充分利用

I定义

```
//  0、头文件
#include <vector>
//  1、定义一个空的vector
vector<int> v;  //  定义一个一维的动态数组
//  2、定义一个vector并赋初始数据(同数组写法)
vector<int> v = {2,4,2,5,1};
//  3、定义一个大小为10的vector,初始值为0(int类型)
vector<int> v(10);  //  注意是小括号
//  4、重新设定大小
v.resize(6); //  重新设定大小
//  5、定义一个大小为10的vector，初始值为5
vector<int> v(10, 5)
```

II 增加

```
//  1、在v末尾插入一个值
v.push_back(3);
v.push_back(2);
v.push_back(5);
//  2、在指定位置前面插入
v.insert(v.begin(), 5);
//  3、向量中迭代器指向元素前增加10个相同的元素5
v.insert(v.begin(), 10, 5)
```

III 查询

```
//  1、得到数组头的指针
v.begin()
//  2、得到数组的最后一个单元+1的指针
v.end()
//  4、查询vector结构v的大小
cout << v.size() << endl;
//  5、查询vector结构v的末尾元素
```

```
cout << v.back() << endl;
//  6、查询vector结构v的头部元素
cout << v.front() << endl;
//  7.1、遍历v中的每一个元素
for(int i = 0; i < v.size(); i++) {
    cout << v[i] << endl;
}
//  7.2、遍历v中的每一个元素方法二
for(auto x: v) {
    cout << x << endl;
}
//  7.3、遍历v中的每一个元素方法三
vector<int>::iterator it;
for(it=v.begin();it!=v.end();it++) {
    cout<<*it<<" ";
}
```

IV 删除

```
//  1、删除最后一个
v.pop_back()
//  2、清空v中的所有数据
v.clear()
//  3、删除向量中迭代器指向元素
v.erase(v.begin())
//  4、删除向量中[first,last)中元素
v.erase(v.begin(), v.end())
```

IV 二分

```
//  找第一个>=x的位置
auto pos1 = lower_bound(v.begin(), v.end(), x)
//  找第一个>x的位置
auto pos2 = upper_bound(v.begin(), v.end(), x)
//  找有多少个<x的个数
int y = lower_bound(v.begin(), v.end(), x) - v.begin()
int num = v.size() - y;
```

1.1 应用场景

set集合

set是STL中一种标准关联容器。set，顾名思义是“集合”的意思，在set中元素都是唯一的，而且默认情况下会对元素自动进行升序排列。如果需要集合中的元素允许重复那么可以使用multiset。

I定义

```
//  定义
set<int> seta;
//  定义一个按照从大到小排序的set
set<int, greater<int> > setb;

int a[5] = {1,2,3,4,5};
set<int > setc(a,a+5); //数组a初始化一个set;

set<int > setd(setc.begin(),setc.end()); //setc初始化一个set
//上述两例均为区间初始化

set<int > sete(setd); //拷贝构造创建set
```

II添加

```
//  插入一个元素
s.insert(5);

//  插入一个区间
int a[4] = {11,12,13,14};
s.insert(a,a+4); //将区间[a, a+4]里的元素插入容器
```

III删除

```
s.erase(9);          //根据元素删除
s.erase(s.begin()); //删除迭代器指向位置的元素
//  区间删除
ita = s.begin();
itb = s.begin();
itb++;itb++;
s.erase(ita,itb); //删除区间[ita,itb)的元素
s.clear()   //删除set容器中的所有的元素
s.empty()
s.size()
```

IV修改

不支持修改, 只能删除之后再插入

V查找

```
s.find()        //查找一个元素，如果容器中不存在该元素，返回值等于s.end()

if(s.find(2) != s.end())
    cout << "2 is existent" << endl;
```

VI遍历set中的每一个元素

```
for(auto x: s){
    cout << x << " ";
}

set<int>::iterator it;
for(it=v.begin();it!=v.end();it++) {
    cout<<*it<<" ";
}
```

VII 二分

STL提供的二分函数

```
//  找第一个>=x的位置
auto pos1 = lower_bound(s.begin(), s.end(), x)
//  找第一个>x的位置
auto pos2 = upper_bound(s.begin(), s.end(), x)
//  找有多少个<x的个数
int y = distance(s.begin(), lower_bound(s.begin(), s.end(), x)) + 1
```

set自带的二分函数

```
//  返回第一个大于或等于x的元素
s.lower_bound(x)

//  返回第一个大于x的元素
s.upper_bound(x)

//返回一对定位器，分别表示 第一个大于或等于给定关键值的元素 和 第一个大于给定关键值的元素，这个返回值是一个pair类型，如果这一对定位器中哪个返回失败，就会等s.end()
s.equal_range(x)
```

```
//  lower_bound, upper_bound参考代码
#include <iostream>
#include <set>
using namespace std;

int main(){
    set<int> s;
    s.insert(1);
    s.insert(2);
    s.insert(5);
```

```
    cout << "lower_bound & upper_bound test:" << endl;

    cout << "第一个大于或等于3的元素: " << *s.lower_bound(3) << endl;
    cout << "第一个大于或等于2的元素: " <<*s.lower_bound(2) << endl;
    cout << "第一个大于2的元素: " <<*s.upper_bound(2) << endl;

    cout << "equal_range test:" << endl;

    cout << "第一个大于或等于2的元素: " <<  *s.equal_range(2).first << endl;
    cout << "第一个大于2的元素: " << *s.equal_range(2).second << endl;
    return 0;
}
```

multiset

同set, 支持重复元素

```
multiset<int> mset;
//  统计元素的个数
mset.count(x);
//  删除元素, multiset的删除某个值会全部删除
mset.erase(5)
//  通过删除迭代器的位置,可以删除某一个值
if((it=s.find(a))!=s.end()) s.erase(it);
```

pair

```
//  定义
#include <pair>
pair<int, int> p;       //  两个值分别为p.first, p.second
```

map

自动建立Key - value的对应。key 和 value可以是任意你需要的类型。

```
//  头文件
#include <map>
//  定义
map<string, int> mp;    //  string->key(下标类型), int->val(数值类型)
```

对于map中的元素, 会自动拍好序, 所以map的操作均为$log(n)$

```
//  map遍历方式1
for(auto x: mp) {
    cout << x.first << " " << x.second << "\n";
}
//  map遍历方式2
for(auto it = mp.begin(); it != mp.end(); it++) {
    cout << *(it) << " ";
}
//  map遍历方式3
map<string, int>::iterator it;
for(it = mp.begin(); it != mp.end(); it++) {
    cout << *(it) << " ";
}
```

unordered_map

同map, 不自带排序，效率高于map

bitset

```
bitset<10> s;
s[1] = 1;
s[3] = 1;
s[4] = 1;
s[7] = 1;
cout << s[4] << "\n"; // 1
cout << s[5] << "\n"; // 0
```

```
bitset<10> s(string("0010011010")); // from right to left
cout << s[4] << "\n"; // 1
cout << s[5] << "\n"; // 0
```

```
bitset<10> s(string("0010011010"));
cout << s.count() << "\n"; // 4
```

```
bitset<10> a(string("0010110110"));
bitset<10> b(string("1011011000"));
cout << (a&b) << "\n"; // 0010010000
cout << (a|b) << "\n"; // 1011111110
cout << (a^b) << "\n"; // 1001101110
```

迭代器辅助函数

迭代器辅助函数	功能
advance(it, n)	it 表示某个迭代器，n 为整数。该函数的功能是将 it 迭代器前进或后退 n 个位置。
distance(first, last)	first 和 last 都是迭代器，该函数的功能是计算 first 和 last 之间的距离。
begin(cont)	cont 表示某个容器，该函数可以返回一个指向 cont 容器中第一个元素的迭代器。
end(cont)	cont 表示某个容器，该函数可以返回一个指向 cont 容器中最后一个元素之后位置的迭代器。
prev(it)	it 为指定的迭代器，该函数默认可以返回一个指向上一个位置处的迭代器。注意，it 至少为双向迭代器。
next(it)	it 为指定的迭代器，该函数默认可以返回一个指向下一个位置处的迭代器。注意，it 最少为前向迭代器。

C++内置函数

I sort()

```
//  对数组进行排序
sort(a + 1, a + 1 + n);      //  对[a1, an]区间进行排序
//  对vector进行排序     从小到大
sort(a.begin(), a.end());
```

II reverse()

```
// 对vector翻转
reverse(v.begin(), v.end())
// 对数组翻转
reverse(a + 1, a + 1 + n)
```

III next_permutation()

```
next_permutation(a + 1, a + 1 + n);
```

IV prev_permutation()

```
prev_permutation(a + 1, a + 1 + n);
```

V random_shuffle()

```
// 将a数组随机打乱
random_shuffle(a + 1, a + 1 + n)
```