STL 笔记:

What is STL?

STL(标准模板库)是一套强大的C++模板类，提供了通用的模板类和函数，这些模板类和函数可以实现多种流行和常用的算法和数据结构。

组成部件:

容器: list,vector,map

算法:对容器类容的初始化，排序，搜索和转换

迭代器:用于便利操作

1. Vector

Vector基于模板实现，需要包含头文件<vector>

    //1.定义和初始化

    vector<int> vec1;    //默认初始化，vec1为空

    vector<int> vec2(vec1);  //使用vec1初始化vec2

    vector<int> vec3(vec1.begin(),vec1.end());//使用vec1初始化vec2

    vector<int> vec4(10);    //10个值为的元素

    vector<int> vec5(10,4);  //10个值为的元素

    //2.常用操作方法

    vec1.push\_back(100);            //添加元素

    int size = vec1.size();         //元素个数

    bool isEmpty = vec1.empty();    //判断是否为空

    cout<<vec1[0]<<endl;        //取得第一个元素

    vec1.insert(vec1.end(),5,3);    //从vec1.back位置插入5个值3为的元素

    //vec1.pop\_back();              //删除末尾元素

    //vec1.erase(vec1.begin(),vec1.end());//删除之间的元素，其他元素前移

    cout<<(vec1==vec2)?true:false;  //判断是否相等==、！=、>=、<=...

    vector<int>::iterator iter = vec1.begin();    //获取迭代器首地址

    vector<int>::const\_iterator c\_iter = vec1.begin();   //获取const类型迭代器

    //vec1.clear();                 //清空元素

    //3.遍历

    //下标法

    int length = vec1.size();

    for(int i=0;i<length;i++)

    {

       cout<<vec1[i];

    }

    cout<<endl<<endl;

    //迭代器法

    vector<int>::const\_iterator iterator = vec1.begin();

    for(;iterator != vec1.end();iterator++)

    {

       cout<<\*iterator;

    }

1. List

List是stl实现的双向链表，与 向量(vectors)相比, 它允许快速的插入和删除，但是随机访问却比较慢。需要添加头文件list

 //1.定义和初始化

    list<int> lst1;          //创建空list

    list<int> lst2(3);       //创建含有三个元素的list

    list<int> lst3(3,2); //创建含有三个元素为2的list

    list<int> lst4(lst2);    //使用lst2初始化lst4

    list<int> lst5(lst2.begin(),lst2.end());  //同lst4

    //2.常用操作方法

    lst1.assign(lst2.begin(),lst2.end());  //分配值

    lst1.push\_back(10);                    //添加值

    lst1.pop\_back();                   //删除末尾值

    lst1.begin();                      //返回首值的迭代器

    lst1.end();                            //返回尾值的迭代器

    lst1.clear();                      //清空值

    bool isEmpty1 = lst1.empty();          //判断为空

    lst1.erase(lst1.begin(),lst1.end());                        //删除元素

    lst1.front();                      //返回第一个元素的引用

    lst1.back();                       //返回最后一个元素的引用

    lst1.insert(lst1.begin(),3,2);         //从指定位置插入个

    lst1.rbegin();                         //返回第一个元素的前向指针

    lst1.remove(2);                        //相同的元素全部删除

    lst1.reverse();                        //反转

    lst1.size();                       //含有元素个数

    lst1.sort();                       //排序

    lst1.unique();                         //删除相邻重复元素

    //3.遍历

    //迭代器法

    for(list<int>::const\_iterator iter = lst1.begin();iter != lst1.end();iter++)

    {

       cout<<\*iter;

    }

    cout<<endl;