1. Stack栈容器
   1. 符合先进后出的数据结构
   2. 对外接口
   3. 构造、赋值
   4. 大小 size
   5. 是否为空 empty
   6. 栈顶 top
   7. 入栈 push
   8. 出栈 pop
   9. 不提供遍历功能，因此没有迭代器
2. queue 队列容器
   1. 符合先进先出数据结构
   2. 对外接口
   3. 构造、赋值
   4. 大小 size
   5. 是否为空 empty
   6. 队头 front
   7. 队尾 back
   8. 入队 push
   9. 出队 pop
   10. 不提供遍历功能，没有迭代器
3. list容器
   1. 双向循环链表
   2. 赋值、构造
   3. 删除 remove（elem） 删除容器中所有与elem匹配的元素
   4. 反转 reverse
   5. 排序 sort
      1. 默认从小到大
      2. 对于自定义数据类型 ，必须要指定排序规则
      3. 案例 给葫芦娃高级排序
4. set容器
   1. 关联式容器，内部插入数据时候自动排序
   2. set容器 不允许插入重复的key值
   3. 删除 erase（key）
   4. find 查找 查找容器中是否有响应的key的数据，如果有返回迭代器，没有返回end的位置
   5. count 统计 统计key的个数，对于set而言，结果要么是0，要么是1
   6. lower\_bound(keyElem);//返回第一个key>=keyElem元素的迭代器。
   7. upper\_bound(keyElem);//返回第一个key>keyElem元素的迭代器。
   8. equal\_range(keyElem);//返回容器中key与keyElem相等的上下限的两个迭代器。
      1. 返回值是一个 对组 pair< 第一个元素，第二个元素 >
   9. 对组声明方式
      1. *pair*<*string*, int> p(*string*("Tom"), 18);
      2. *pair*<*string*, int> p2 = *make\_pair*("Jerry", 20);
      3. 第一个元素 first 第二个元素 second
   10. set容器只能插入不重复key值
       1. insert返回值是 pair< iterator , bool >
       2. bool代表了插入是否成功
   11. 对于set容器的排序
       1. 必须要在插入之前 指定出排序规则
       2. 配合仿函数 实现
       3. 自定义数据类型 ，必须要指定出排序规则
5. map容器
   1. 关联式容器，key 和 value 每个元素都是对组
   2. 插入 4种方式
      1. m.insert( pair<int,int>(1,1) )
      2. m.insert( make\_pair(2,2) )
      3. m.insert(map<int,int>::value\_type(3,3) )
      4. m[4] = 4
   3. find(key);//查找键key是否存在,若存在，返回该键的元素的迭代器；/若不存在，返回map.end();
   4. count(keyElem);//返回容器中key为keyElem的对组个数。对map来说，要么是0，要么是1。对multimap来说，值可能大于1。
   5. lower\_bound(keyElem);//返回第一个key>=keyElem元素的迭代器。
   6. upper\_bound(keyElem);//返回第一个key>keyElem元素的迭代器。
   7. equal\_range(keyElem);//返回容器中key与keyElem相等的上下限的两个迭代器。
   8. 改变map容器的排序规则 ，利用仿函数