1. 栈容器 Stack
   1. 符合先进后出的数据结构
   2. 对外接口
      1. 入栈 push
      2. 出栈 pop
      3. 栈顶 top
      4. 是否为空 empty
      5. 栈大小 size
2. 队列容器 Queue
   1. 符合先进先出的数据结构
   2. 对外接口
      1. 入队 push
      2. 出队 pop
      3. 队头元素 front
      4. 队尾元素 back
      5. 是否为空 empty
      6. 队列大小 size
3. list容器
   1. 双向循环链表
   2. 对外接口
      1. 构造、赋值、大小、重置大小、是否为空
      2. 反转 reverse
      3. 排序 sort
         1. //如果容器的迭代器支持随机访问，可以使用系统提供的标志算法
         2. //不支持随机访问的迭代器的容器，内部会提供对应的算法接口
         3. 对于自定义数据类型，必须要指定排序规则
      4. 对自定义数据类型做了高级排序
      5. 如果利用remove删除自定义数据类型，需要重载 ==
4. set 容器
   1. 关联式容器 key就是value
   2. 默认排好序 从小到大
   3. 插入 insert 大小 size 是否为空 empty
   4. 查找 find 返回值 迭代器
   5. 统计 count 对于set的结果 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。
   9. pair对组
      1. 创建方式
      2. *pair*<*string*, int> p("Tom", 10);
      3. *pair*<*string*, int> p2 = *make\_pair*("Jerry", 18);
   10. set.insert的返回值是个对组 pair<iterator, bool> bool代表插入是否成功
   11. multiset可以插入重复的key值
   12. 可以指定set容器的排序规则，但是必须在插入前指定，利用仿函数的技术
   13. 对于自定义数据类型，set通常都会指定出排序规则
5. map容器
   1. 关联式容器
   2. 默认按照key从小到大排序
   3. 插入
      1. m.insert(pair<int, int>(1, 10));
      2. m.insert(make\_pair(2, 20));
      3. m.insert(map<int, int>::value\_type(3, 30));
      4. m[4] = 40;
   4. 查找 find 返回值 是迭代器
   5. 统计 count
   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。
   9. 利用仿函数 实现指定排序规则
6. STL容器使用时机
7. 作业布置