1. 函数对象
   1. 本质是一个类的对象，因此称为函数对象，也叫仿函数
   2. 函数对象 超出了普通函数的概念，可以拥有自己状态
   3. 函数对象可以作为函数参数
2. 谓词
   1. 普通函数或者仿函数的返回值是bool类型，称为谓词
   2. 一元谓词
      1. 查找容器中大于20的数字 find\_if
   3. 二元谓词
      1. 对容器进行排序 sort
   4. lambda表达式 [](){}
3. 内建函数对象
   1. 引入头文件 #include< functional>
   2. 取反 negate<int>
   3. 加法 plus<int>
   4. 大于 greater<int>
4. 适配器
   1. 函数对象适配器
      1. //1、利用bind2nd 进行绑定
      2. //2、继承 public binary\_function<参数1 类型,参数2类型,返回值类型>
      3. //3、加const
   2. 取反适配器
      1. 一元取反 not1
         1. //1、利用not1进行取反
         2. //2、继承 public unary\_function<int,bool>
         3. //3、加const
      2. 二元取反 not2
   3. 函数指针适配器
      1. ptr\_fun将普通函数指针 适配成函数对象
   4. 成员函数适配器
      1. 如果存放的是对象实体 mem\_fun\_ref
      2. 如果存放的是对象指针 mem\_fun
5. 常用遍历算法
   1. **for\_each**
      1. 用于遍历
      2. 有返回值
      3. 可以绑定参数进行输出
   2. transform
      1. 搬运
      2. 注意：目标容器要有容量
6. 常用查找算法
   1. **find 查找**
   2. **find\_if 按条件查找**
   3. adjacent\_find算法 查找相邻重复元素
   4. binary\_search算法 二分查找法
      1. 注意: 在无序序列中不可用
   5. **count算法 统计元素出现次数**
   6. **count\_if 按条件进行统计**
7. 常用排序算法
   1. merge 合并
      1. 将两个容器合并到 目标容器中
      2. 注意： 两个容器必须是有序序列
      3. 目标容器必须有容量
   2. **sort 排序**
   3. **random\_shuffle 洗牌**
   4. reverse 反转
8. 常用的拷贝和替换算法
   1. copy 拷贝
      1. 实现打印 copy(v.begin(),v.end() , ostream\_iterator<int>(cout , “ ”));
   2. replace 替换
   3. replace\_if 按条件替换
   4. swap 交换
9. 常用的算数生成算法
   1. 头文件 #include <numeric>
   2. **accumulate**算法 计算容器元素累计总和
   3. fill算法 向容器中添加元素
10. 常用集合算法
    1. set\_intersection算法 求两个set集合的交集
    2. set\_union算法 求两个set集合的并集
    3. set\_difference算法 求两个set集合的差集
    4. 注意:两个集合必须是有序序列