小马加编信息学教案(三十八)

STL（string，algorithm）

一. 课程内容

1. $string$
2. $algorithm$

二. 知识讲解

1. $string$

1. 1 $string$的介绍

> 在之前的学习中，我们一般使用字符数组`char str[100];`来存放字符串，但是操作麻烦，容易出错。而STL中的$string$类型，对字符串常用的需求功能进行了封装，使得操作起来更加方便，且不必担心内存是否足够、字符串的长度等问题。原本实现复杂的问题，可以用一句简单的语句代替。

1. 2 $string$的定义

> 使用$string$，需要添加$string$头文件，即`#include <string>`，同时必须要有`using namespace std`。

> 定义$string$的方法为：

```
string name;
```

> 其中，$name$是字符串变量的名字。

1. 3 $string$的访问

> 若要将一个字符串赋值给一个$string$类型变量，可以直接用双引号括住进行赋值。

```
string str;
str = "abcd"
```

> 当然，在定义一个$string$类型变量时也能为它赋初值。

```
string str = "abcd";
```

> $string$类型单个字符的访问方法和普通字符数组一样，直接按顺序用下标进行访问。

```
string str = "abcd";
for (int i = 0; i < 4; i ++) printf("%c", str[i]);
//输出abcd。
```

> 当然$string$类型变量也可以作为一个整体输入输出。

```
cin >> str;
cout << str << endl;
```

1. 4 $string$的运算

> 对于$string$类型，STL重新定义了加法运算和比较运算，使我们能方便的对字符串进行处理。

> 对于两个字符串的加法，自然就是将他们拼接起来。

```
string str1 = "abc", str2 = "xyz", str3;
str3 = str1 + str2;
// str3 = abcxyz
str1 += str2;
//str1 = abcxyz
```

> 对于两个字符串的比较则是按字典序大小的比较，即从前往后，找到第一个不相同的位置进行比较，空字符视为最小。

```
string str1 = abc, str2 = abd, str3 = ab;
```

> 对于上面三个字符串，从小到大依次是$str3 < str1 < str2$。

1. 5 $string$的常用函数

> 定义$string$类型变量strh和$str2$，设n为str的长度，m为$str2$的长度。

```
string str, str2;
```

- `str.length(), str.size()`
  这两个功能一样，都是返回字符串str的长度，时间复杂度为$O(1)$。
- `str.clear()`
  将str清空，即存储空字符串，时间复杂度为$O(1)$。
- `str.substr(pos, len)`
  返回str中从第pos个位置开始长度为len的$string$类型的字符串，比如`str.substr(2, 5)`即返回str中第2个字符到第6个字符组成的字符串，时间复杂度为$O(n)$。

- `str.insert(pos, str2)`
  在str的pos位置插入$str2$，即将原来pos位置处的字符向后移，时间复杂度为$O(n)$。
- `str.erase(pos, len)`
  删除str从第pos个位置开始，长度为len的字符串，时间复杂度为$O(n)$。
- `str.find(str2)`
  查找$str2$第一次在str中出现的位置，若无则返回常数$str.npos = 4294967295$，
  即`str.find(str2) == str.npos`时为未找到，时间复杂度是$O(nm)$。
- `str.find(str2, pos)`
  与上一个类似，只不过是从str的第pos个位置开始查找，时间复杂度为$O(nm)$。
- `str.replace(pos, len, str2)`
  将str中从第pos个位置开始的长度为len的字符串替换成$str2$。相当于
  先`str.erase(pos, len)`再`str.insert(pos, str2)`，时间复杂度是$O(n)$。

2. $algorithm$

2. 1 $algorithm$的介绍

> $algorithm$翻译为**算法**，它不是一种数据类型，而是提供了大量写程序中需要使用的基础函数。要使用这些函数需要添加$algorithm$头文件，即`#include <algorithm>`，同时必须要又`using namespace std`。

2. 2 $min(), max(), abs(), swap()$

- `min(x, y), max(x, y)`
  分别为返回x, y中较小的数和较大的数，需要x, y是相同类型，可以是浮点数，返回类型和x, y相同。
- `abs(x)`
  返回x的绝对值，x必须要是整数，返回类型和x相同。
- `swap(x, y)`
  交换x, y的值，需要x和y是相同类型，没有返回值。

2. 3 $sort()$

> $sort()$是实现排序功能的函数，可以对一个长度为n的序列，在$O(nlog_2^n)$的时间内进行排序。

> $sort$的基本格式为$sort($首元素地址，尾元素地址的下一个地址$)$

- 对于数组a，$sort(a + x, a + y + 1)$表示将第x个到第y个元素从小到大排序。比如$sort(a, a + n)$就是将第0个到第$n - 1$个元素从小到大排序，
- 对于$vector$类型变量a，$sort(a.begin(), a.end())$就是将a中的元素从小到大排序。

> 更强大的是$sort()$函数可以增加比较函数作为参数，自定义排序的优先级。格式为$sort($首元素地址，尾元素的下一个地址，比较函数$)$，比如要将长度为n的数组a从大到小排序，可以通过如下方法实现。

```
bool cmp(int x, int y) {
    //x，y的类型去排序的元素类型一致。如果排序的是字符数组，那么x，y的数据类型就用char
    return x > y;
    //可以视为x放在y前的条件，即返回值为真时将x放在y前面。
}

int main() {
    scanf("%d", &n);
    for (int i = 0; i < n; i ++) scanf("%d", &a[i]);
    sort(a, a + n, cmp);
    for (int i = 0; i < n; i ++) printf("%d ", a[i]);
}
```

2. 4 $lower\_bound()$ 和 $upper\_bound()$

> $lower\_bound()$ 和 $upper_bound()$都是要对升序序列进行操作，这个序列可以用数组或$vector$存储。他们存在细微的差别。

- $lower_bound()$是求序列中第一个大于等于某个值的位置。
- $upper_bound()$是求序列中第一个大于某个值的位置。

> 它们的用法也是一样的，那下面就只对$lower_bound()$进行描述。

- 对数组进行操作时与$sort()$一样，对于数组a，$lower_bound(a+x,a+y+1,val)$是找$a$数组第$x$个元素到第$y$个元素中，第一个大于等于$val$的位置。具体的实现如下：

```
sort(a, a + n);
int side = lower_bound(a, a + n, val) - a;
//side就是对应的值在a数组中的下标，即a[side]是第一个满足大于等于val的值。
//若所有元素都小于val，则side等于n。
```

- 对$vector$进行操作时，也与$sort()$类似，对于$vector$类型a，，$lower_bound(a.begin(),a.end(),val)$是找第一个大于等于$val$的位置。具体实现如下：

```
sort(a.begin(), a.end());
int side = lower_bound(a.begin(), a.end(), val) - a.begin();
//side就是对应的值在vector中的下标，即a[side]是第一个满足大于等于val的值。
//若所有元素都小于val，则side等于a.size()。
```

> $lower\_bound()$ 和 $upper_bound()$查询实际上是用二分实现的，所以时间复杂度就是$O(log_2^n)$，其中n为查询序列的长度。

三. 经典例题

1. 字符串匹配

> 现定义两个仅由大写字母组成的字符串的匹配程度如下：将某一字符串的首字符与另一字符串的某一字符对齐，然后后面的字符也一一对齐，直至某一字符串的串尾为止。对于每一组对齐的两个字符，若这两个字符相等，则计数。匹配程度为每种对齐方法的计数的最大值。
> 用$string$实现。

> 输入格式：
> 第一行一个正整数t表示比较组数。
> 接下来t行，每行两个字符串，s,t分别表示比较的两个字符串。
> $t \leq 10$

> 输出格式：
> t行，每行一个整数表示对应字符串的比较程度。

样例输入	样例输出
4 CAR CART TURKEY CHICKEN MONEY POVERTY ROUGH PESKY	3 2 2 0

2. 查找位置

> 读入两个字符串s,t，输出t在s中出现的不重叠的位置。如$s=ababa$，$t=aba$时，出现位置只算位置1开头的字符串，位置3处由于与前一个重叠所以不算。

> 输入格式：
> 第一行一个字符串s
> 第二行一个字符串t
> $|s|\leq 10^5$
> $|t|\leq 10^3$

> 输出格式：
> 第一行一个正整数ans表示不重叠的出现次数。
> 接下来一行ans个整数，从小到大输出t出现的位置。

样例输入	样例输出
abababa aba	2 1 5

3. 查字典

> 小明正在复习全国英语四级考试，他手里有一本词典，现在有很多单词要查。请编写程序帮助他快速找到要查的单词所在的页码。

> 输入格式：
> 第一行一个正整数n表示字典中一共有多少个单词。
> 接下来$2n$行，每两行表示一个单词，第一行是一个长度小于100的字符串表示这个单词，第二行是这个单词出现的页码。
> 第$2n+2$行一个正整数m表示要查的单词个数。
> 接下来m行，每行一个字符串表示要查找的单词，保证在字典中存在。
> $n,m\leq 10^5$

> 输出格式：
> m行，每行一个正整数表示查询单词出现的页码。

样例输入	样例输出
2 scan 10 word 15 2 scan word	10 15

四. 提高巩固

1. 最长不下降子序列

> 给定长度为n的正整数序列a。寻找它的一个最长子序列，使得子序列是不下降的，输出此子序列的长度（可以不连续）。

> 输入格式:
> 第一行一个正整数n表示序列长度。
> 第二行n个正整数，表示序列a。
> $n \leq 10^5$
> $a_i \leq 10^9$

> 输出格式:
> 一行一个整数表示最长不下降子序列的长度。

样例输入	样例输出
8 1 3 1 5 9 7 4 8	5

2. 排名

> 一年一度的江苏省小学生程序设计比赛结束后，组委会公布了所有学生的成绩，成绩按分数从高到低排名，成绩相同按年级从低到高排。现在主办单位想知道每一个排名的学生前，有几位学生的年级低于他。
> 用$sort$和$pair$实现。

> 输入格式:
> 第一行一个正整数n，表示参赛学生人数。
> 接下来n行，每行两个正整数分别表示第i个学生的成绩和年级。
> $n \leq 1000$

> 输出格式:
> n行，每行一个正整数表示有多少名学生排名比i前，且年级比排名i的学生低。

样例输入	样例输出
5 300 5 200 6 350 4 400 6 250 5	0 0 1 1 3

3. 字符串乘方

> 给定两个字符串a和b，定义$a * b$为它们的连接。例如，如果 $a = abc$而 $b = “def$，则 $a * b = abcdef$ 。如果将连接考虑成乘法，一个非负整数的乘方将用一种通常的方式定义：$a^0 =$ 空字符串，$a^{n+1} = a * a^n$。
> 现在给定一个字符串s，要求输出一个最大的n，使得存在一个长度为n的字符串a满足$s = a^n$。

> 输入格式:
> 第一行一个正整数t表示测试组数。
> 接下来t行，每行一个字符串s。
> $t \leq 10$
> $|s| \leq 10^5$

> 输出格式:
> t行，每行一个正整数表示答案。

样例输入	样例输出
3 abcd aaaa ababab	1 4 3