Homework One

2017141493004 常家奇

- **1.3**
  - 设我们定义的这个字符串 ADT 名字为 My_String ,其中包含:
  1. 迭代器 begin() 和 end()
  2. 字符串长度 length()
  3. 字符串是否为空 empty()
  4. 清空字符串 clear()
  5. 重置字符串空间 resize(int n)
  6. 追加字符 push_back(char c)
  7. 字符串插入 insert(My_string str,int x)
  8. 删除字符串中特定字符 erase(char c)
  9. 字符串拼接 append(My_string str)
  10. 字符串复制 My_String copy()
  11. 寻找特定字符串在当前字符串中首次出现的位置 int find(My_String str)
  12. 返回特定位置子串 My_String substr(int pos)
  13. 字符串比较 compare(My_String str)
- **1.7**
  - 使用泛型 T ,要求 T 类型含有 compare() 函数.Ex:

```
template <typename T>
void bubble_sort(T const *data,int n){
    for (int i=0;i<n;i++)
        for (int j=1;j<n-i;i++){
            if (data[j-1].compare(data[j]) > 0){
                T t = data[j-1];
                data[j-1] = data[j];
                data[j] = t;
            }
        }
}
```

- **1.8**
  - 创建不同种类的key K 对value V 的映射,即将搜索的 key 为泛型,在此种条件: V 种类相同, K 种类不同下,我们需要定义 K 的 V find(K k) 函数.Ex:

在例题类中:

```
template <typename K>
int find_record(K k){
    if (std::is_same<K, int>::value){
        // Find from salary map.
        return value;
    }
    else if (std::is_same<K, string>::value){
        if (k.size() < 10){
            // Find from zip code map.
            return value;
        }
        else{
            // Find from name map.
            return value;
        }
    }
    else{
        // Find from GPA map.
        return value;
    }
}
```

为了提高搜索效率,我们可以使用高效存储数据结构(HashMap,TreeMap),提高搜索速度,同时,如果搜索的元素的重复次数很多,我们还可以建立搜索缓存,进一步提高搜索速度.

- **1.11**
  - `string find_correct_word(string s)` 这个函数必须完成.
    这个函数即为从字典中搜索出和当前输入 `s` 差异最小的单词.
    假设字典由Trie树构建,从输入的单词开始进行逐字符匹配,如果某个字符匹配不上,那么去其上一级其他字母匹配.

也可以将字典建为一个表,顺序查找和输入的字符串差异最小的单词.遍历需要$O(N)$.
好像现在可以使用nlp处理这个问题了,不过我还没学.

Extra:

- Choice:
  A,A,B
- Program:

```
int swap(int& x,int& y){
    int tmp =x;
    x = y;
    y = tmp;
}
int sort_example(int& x,int& y,int& z){
    if (x>y)  swap(x,y);

    if (x>z)  swap(x,z);

    if (y>z)  swap(y,z);
}
```