01.06 字符串的全排列

题目描述

输入一个字符串，打印出该字符串中字符的所有排列。

例如输入字符串abc，则输出由字符a、b、c 所能排列出来的所有字符串

abc、acb、bac、bca、cab 和 cba。

分析与解法

解法一、递归实现

从集合中依次选出每一个元素，作为排列的第一个元素，然后对剩余的元素进行全排列，如此递归处理，从而得到所有元素的全排列。以对字符串abc进行全排列为例，我们可以这么做：以abc为例

- 固定a，求后面bc的排列：abc，acb，求好后，a和b交换，得到bac
- 固定b，求后面ac的排列：bac，bca，求好后，c放到第一位置，得到cba
- 固定c，求后面ba的排列：cba，cab。

代码可如下编写所示：

```
void CalcAllPermutation(char* perm, int from, int to)
{
        if (to <= 1)
        {
                return;
        }

        if (from == to)
        {
                for (int i = 0; i <= to; i++)
                        cout << perm[i];
                cout << endl;
        }
        else
        {
                for (int j = from; j <= to; j++)
                {
                        swap(perm[j], perm[from]);
                        CalcAllPermutation(perm, from + 1, to);
                        swap(perm[j], perm[from]);
                }
        }
}
```

解法二、字典序排列

首先，咱们得清楚什么是字典序。根据维基百科的定义：给定两个偏序集A和B,(a,b)和(a′,b′)属于笛卡尔集 A × B，则字典序定义为

(a,b) ≤ (a′,b′) 当且仅当 a < a′ 或 (a = a′ 且 b ≤ b′)。

所以给定两个字符串，逐个字符比较，那么先出现较小字符的那个串字典顺序小，如果字符一直相等，较短的串字典顺序小。例如：abc < abcd < abde < afab。

那有没有这样的算法，使得

- 起点： 字典序最小的排列, 1-n , 例如12345
- 终点： 字典序最大的排列，n-1, 例如54321
- 过程： 从当前排列生成字典序刚好比它大的下一个排列

答案是肯定的：有，即是STL中的next_permutation算法。

在了解next_permutation算法是怎么一个过程之前，咱们得先来分析下“下一个排列”的性质。

- 假定现有字符串(A)x(B)，它的下一个排列是：(A)y(B’)，其中A、B和B’是“字符串”(可能为空），x和y是“字符”，前缀相同，都是A，且一定有y > x。
- 那么，为使下一个排列字典顺序尽可能小，必有：
- A尽可能长
- y尽可能小
- B’里的字符按由小到大递增排列

现在的问题是：找到x和y。怎么找到呢？咱们来看一个例子。

比如说，现在我们要找21543的下一个排列，我们可以从左至右逐个扫描每个数，看哪个能增大（至于如何判定能增大，是根据如果一个数右面有比它大的数存在，那么这个数就能增大），我们可以看到最后一个能增大的数是：x = 1。

而1应该增大到多少？1能增大到它右面比它大的那一系列数中最小的那个数，即：y = 3，故此时21543的下一个排列应该变为23xxx，显然 xxx(对应之前的B’）应由小到大排，于是我们最终找到比“21543”大，但字典顺序尽量小的23145，找到的23145刚好比21543大。

由这个例子可以得出next_permutation算法流程为：

next_permutation算法

- 定义
- 升序：相邻两个位置ai < ai+1，ai 称作该升序的首位
- 步骤（二找、一交换、一翻转）
- 找到排列中最后（最右）一个升序的首位位置i，x = ai
- 找到排列中第i位右边最后一个比ai 大的位置j，y = aj
- 交换x，y
- 把第(i+ 1)位到最后的部分翻转

还是拿上面的21543举例，那么，应用next_permutation算法的过程如下：

- x = 1；
- y = 3
- 1和3交换
- 得23541
- 翻转541
- 得23145

23145即为所求的21543的下一个排列。参考实现代码如下：

```
bool CalcAllPermutation(char* perm, int num){
        int i;

        //①找到排列中最后（最右）一个升序的首位位置i，x = ai
        for (i = num - 2; (i >= 0) && (perm[i] >= perm[i + 1]); --i){
                ;
        }
        // 已经找到所有排列
        if (i < 0){
                return false;
        }

        int k;
        //②找到排列中第i位右边最后一个比ai 大的位置j，y = aj
        for (k = num - 1; (k > i) && (perm[k] <= perm[i]); --k){
                ;
        }

        //③交换x，y
        swap(perm[i], perm[k]);
        //④把第(i+ 1)位到最后的部分翻转
        reverse(perm + i + 1, perm + num);
        return true;
}
```

然后在主函数里循环判断和调用calcAllPermutation函数输出全排列即可。

解法总结

由于全排列总共有n!种排列情况，所以不论解法一中的递归方法，还是上述解法二的字典序排列方法，这两种方法的时间复杂度都为O(n!)。

类似问题

1、已知字符串里的字符是互不相同的，现在任意组合，比如ab，则输出aa，ab，ba，bb，编程按照字典序输出所有的组合。

分析：非简单的全排列问题（跟全排列的形式不同,abc全排列的话，只有6个不同的输出）。 本题可用递归的思想，设置一个变量表示已输出的个数，然后当个数达到字符串长度时，就输出。

```
//copyright@ 一直很安静 && World Gao
//假设str已经有序
void perm(char* result, char *str, int size, int resPos)
{
  if(resPos == size)
    printf("%s\n", result);
  else
    {
      for(int i = 0; i < size; ++i)
        {
          result[resPos] = str[i];
          perm(result, str, size, resPos + 1);
        }
    }
}
```

2、如果不是求字符的所有排列，而是求字符的所有组合，应该怎么办呢？当输入的字符串中含有相同的字符串时，相同的字符交换位置是不同的排列，但是同一个组合。举个例子，如果输入abc，它的组合有a、b、c、ab、ac、bc、abc。

3、写一个程序，打印出以下的序列。

(a),(b),(c),(d),(e).....(z)

(a,b),(a,c),(a,d),(a,e).....(a,z),(b,c),(b,d).....(b,z),(c,d).....(y,z)

(a,b,c),(a,b,d)....(a,b,z),(a,c,d)....(x,y,z)

....

(a,b,c,d,.....x,y,z)