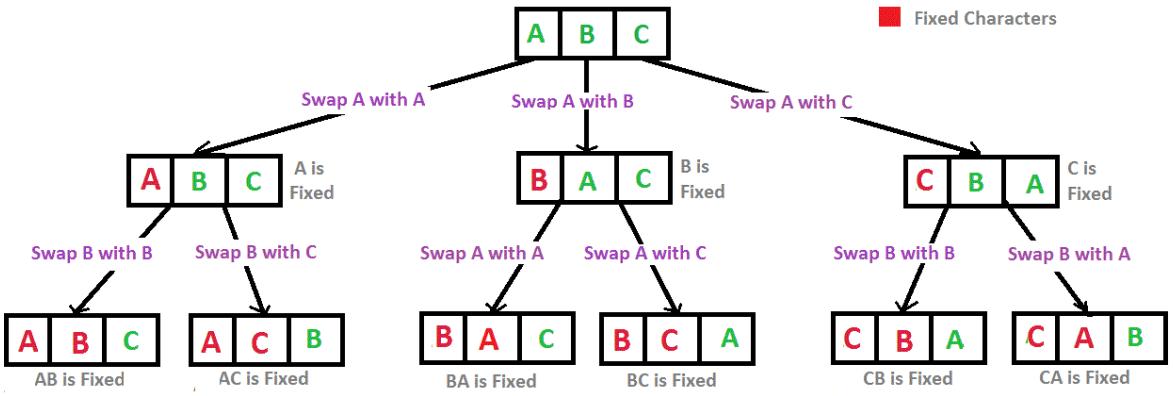
字符串

27 字符串的排列【1】【递归思想】

- **题目**
  输入一个字符串,按字典序打印出该字符串中字符的所有排列。例如输入字符串abc,则打印出由字符a,b,c所能排列出来的所有字符串abc,acb,bac,bca,cab和cba。
- **思路**
  列举多个结果的，递归求解
  1、对于无重复的字符串，固定第一个字符，求剩下n-1个字符的排序方法，接着第一个字符与第二个字符交换位置，继续求剩下n-1字符的排序方法（n-1个字符的排序方法采用递归来求）
  2、对于有重复的值时，比如n个数，剩余n-1个数中有和第一个数相同时，则不进行交换



Recursion Tree for Permutations of String "ABC"

第二排中，依次固定第一个字符不变
第三排中，固定第二三个字符其中一个不变

- 代码实现

```
class Solution {
public:
    vector<string> Permutation(string str) {
        int len = str.length();
        if(!len)
            return {};
        vector<string> result;
        Permutations(result, str, 0, len);
        return result;
    }
    void Permutations(vector<string>&result, string str,int index,
int len){
     //当索引指向字符串尾部时，将str压入数组
     if (index == len){
            result.push_back(str);
            return;
        }
        for (int i = index; i < len; ++i) {
            if (i!=index && str[i]== str[index]) continue;// 保证当输
入多个重复字符时，不会重复计算
            swap(str[i],str[index]);//每一次，交换从第index位以后的一个
元素
            Permutations(result, str, index+1, len);
        }
    }
};
```

34 第一个只出现一次的字符

- **题目**
  在一个字符串(0<=字符串长度<=10000，全部由字母组成)中找到第一个只出现一次的字符,并返回它的位置, 如果没有则返回 -1（需要区分大小写）.
- **思路**
  第一次遍历统计每个字符出现的次数。

第二次遍历统计出最早出现一次的字符

- **基础知识**
  map temp ;
  char c= 'a';
  temp[c]++;
- 代码实现

```
class Solution {
public:
    int FirstNotRepeatingChar(string str) {
        map<char,int> temp;
        int len = str.size();
        for(int i = 0; i < len ;i++)
            temp[str[i]]++;
        for(int i = 0; i < len ;i++){
            if(temp[str[i]]== 1 )
              return i;
        }
         return -1;
    }
};
```

43 左转字符串

- **题目**
  汇编语言中有一种移位指令叫做循环左移（ROL），现在有个简单的任务，就是用字符串模拟这个指令的运算结果。对于一个给定的字符序列S，请你把其循环左移K位后的序列输出。例如，字符序列S=”abcXYZdef”,要求输出循环左移3位后的结果，即“XYZdefabc”。是不是很简单？OK，搞定它！
- **思路**
  第一种，数组a复制出前面n个值，把原数组从n位置之后的值都往前推n个，最后把数组a加到最后。需要加n的存储空间
  第二种，不外加空间。分别进行0到n-1的翻转，n到len-1的翻转，最后0到len-1的翻转，实现前面n个数移到最后。（推荐）
- 代码实现

```
class Solution {
    void swap(string &str, int low, int high){
        while(low < high){
            char temp = str[low];
            str[low++] = str[high];
            str[high--] = temp;
        }
    }
public:
    string LeftRotateString(string str, int n) {
        int len = str.length();
        if(!len)
            return "";
        n = n % len;
        swap(str, 0, n - 1);  //reverse(str.begin(), str.begin() + n); 可以用内置交换函数
        swap(str, n, len - 1);
        swap(str, 0, len - 1);
        return str;
    }
};
```

44 翻转单词序列【1】【不使用额外空间】

- **题目**
  例如，“student. a am I”。正确的句子应该是“I am a student.”。Cat对一一的翻转这些单词顺序可不在行，你能帮助他么？
- **思路**
  此题应该增加限制条件，不能额外增加空间。
  实现思路：1、翻转整个字符串序列。2、从头开始遍历字符数组，读出一个字符串，进行翻转，直到结束
- 代码实现

```
class Solution {
public:
    string ReverseSentence(string str) {
        int len = str.length();
```

```
        if(!len)
            return "";
        reverse(str.begin(), str.end()); //翻转全部字符
        int front = 0, back = 0;
        while(front < len){
            while(str[front] == ' ' && front < len)  //跳过多个空格
                ++front;
            back = front;
            while(str[back] != ' ' && back < len)  //记录一个单词的开始位置和起始位置（目前该单词是倒着的）
                ++back;
            reverse(str.begin() + front, str.begin() + back); //翻转单个单词
            front = back;
        }
        return str;
    }
};
```

49 把字符串转换成整数【1】【开始的符号处理】

- **题目**

  题目描述

  将一个字符串转换成一个整数(实现Integer.valueOf(string)的功能，但是string不符合数字要求时返回0)，要求不能使用字符串转换整数的库函数。 数值为0或者字符串不是一个合法的数值则返回0。

  输入描述:

  输入一个字符串,包括数字字母符号,可以为空

  输出描述:

  如果是合法的数值表达则返回该数字，否则返回0

  示例1

  输入

  +2147483647

  1a33

  输出

  2147483647

  0

- **思路**

编程的细节，实现左移乘以10，以及多个左移运算需要带括号第一个为正负号的处理。
左移乘以10的实现：res = (res << 1) + (res<<3) ;

- 代码实现

```
class Solution {
public:
    int StrToInt(string str) {
        int len = str.length();
        if(!len)
            return 0;
        int res = 0;
        bool flag = true; //保存正负号
        if(str[0] == '-')
             flag = false;
        for(int i = (str[0]=='-' || str[0]=='+') ? 1:0 ; i < len; ++i){ //如果第一个为正负号，i则从1开始
            if(str[i] >= '0' && str[i] <= '9')
                res = (res << 3) + (res << 1) + str[i] - '0';
            else
                return 0;
        }
        return flag ? res:-res;
    }
};
```

52 正则表达式匹配【1】

- **题目**
请实现一个函数用来匹配包括'.'和'*'的正则表达式。模式中的字符 '.' 表示 任意一个字符，而 * 表示 它前面的字符可以出现任意次（包含0次）。 在本题中，匹配是指字符串的所有字符匹配整个模式。例如，字符串"aaa"与模式"a.a"和"ab*ac*a"匹配，但是与"aa.a"和"ab*a"均不匹配

- **思路**
首先，对题目进行简单解释

'*' 表示 它前面的字符可以出现任意次的意思为：比如 ba*，那么字符a的次数为任意次。

分类讨论：

模式中的第二个字符是 * 时：
如果字符串和模式的第一个字符不匹配时，模式后移2个字符（*前面次数为0），继续匹配；
如果两者的第一个字符匹配时，接下来有三种匹配方式：
1、模型后移2字符（相当忽略 x*）
2、字符串后移1字符，模式后移2字符（x*当做一个字符）
3、字符串后移1字符，模式不变（x*当做多个相同字符）

模式中的第二个字符不是 *时：
1、字符串第一个字符和模式中第一个字符相匹配，两个都后移一位继续匹配
2、如果字符串第一个字符和模式的第一个字符不匹配，则失败，返回false

- 代码实现

```
class Solution {
public:
    bool match(char* str, char* pattern){
        if(!str || !pattern)
            return false;
        if(*str == '\0') //字符串结束的时候，模式为\0 或 x*\0 时字符串和模式匹配；否则不匹配
            return *pattern=='\0' || (*(pattern+1)=='*' && *(pattern+2)=='\0') ? true:false;
        if(*pattern == '\0') //模式比字符串提前结束，不匹配
            return false;
        if(*(pattern+1) == '*'){ //m模式第二个字符为*时
            if(*pattern==*str || *pattern=='.') //匹配时，模式把该字符当做多个（str后移一位，pattern不动） 或 跳两个字符忽略*（str不动，pattern后移两位）
                return match(str+1,pattern) || match(str,pattern+2);
            else //不匹配时，str不动，pattern跳两个字符
                return match(str,pattern+2);
```

```
        }
        return (*pattern==*str || *pattern=='.') ?
match(str+1,pattern+1) : false;
    }
};
```

53 表示数值的字符串

- **题目**
  题目描述
  请实现一个函数用来判断字符串是否表示数值（包括整数和小数）。例如，字符串"+100","5e2","-123","3.1416"和"-1E-16"都表示数值。 但是"12e","1a3.14","1.2.3","+-5"和"12e+4.3"都不是
- **思路**
  主要考察一些符号的规则，写判断条件
  字符 'e' 和 'E'：只能出现一次，且不能在最后一位
  字符 '+' 和 '-'：只要不在首位出现时，它必须紧跟在e后面（判断前一个字符是否为e），且不能在最后一位
  字符 '.' ：最多只能出现一次 且 只能出现在 e符号之前 且 不能在最后一位
  除以上字符外得是 0到9之间的字符
- 代码实现

```
class Solution {
public:
    bool isNumeric(char* str){
        if(!str)
            return false;
        int len = strlen(str);
        bool hasE = false,hasdem = false,sign = false;//dem是小数点，
sigh正负标记
        for (int i = 0 ; i < len ; i++){
            if(str[i] == 'e' || str[i] == 'E'){ //出现 e的时候
                if(i == len-1 || hasE) //&&优先级大于||优先级
                    return false;   //没出现过，但在最后一位或出现过，返
回错误
                hasE = true;
            }
```

```
            else if(str[i]== '+' || str[i] == '-'){ //加减号不在首位就会与e绑定在一起
                if((i!=0 && str[i-1]!='e' && str[i-1]!='E')|| i == len-1)//出现正负号且不在首位时，如果后面不是e的话返回错误；正负号在最后一位返回错误
                    return false;
                sign = true;
            }
            else if(str[i] == '.'){
                if(hasdem || hasE || i == len-1) //出现点 或 出现过e或在最后一位，返回错误
                    return false;
                hasdem = true;
            }
            else if(str[i] < '0' || str[i] > '9')
                return false;
        }
        return true;
    }
};
```

54 字符流中第一个不重复的字符

- **题目**
  题目描述
  请实现一个函数用来找出字符流中第一个只出现一次的字符。例如，当从字符流中只读出前两个字符"go"时，第一个只出现一次的字符是"g"。当从该字符流中读出前六个字符“google"时，第一个只出现一次的字符是"l"。
  输出描述:
  如果当前字符流没有存在出现一次的字符，返回#字符。
- **思路**
  首先，相当于程序使用insert函数不断的输入字符，然后会调用FirstAppearingOnce()寻找第一个只出现一次的字符。
  inert函数：
  字符的ASCII值范围为0~127,用一个数组存储每个字符出现的次数，用一个队列记录每个字符最早出现的顺序（重复字符不重复入队）

1. 定义一个128长度的整形数组，用于记录每个字符出现的次数。

2. 用队列，把第一次出现的字符压入队列。
   FirstAppearingOnce函数：
   找第一个只出现一次的字符过程
3. 若队列不为空且队头字符在count数组里的次数大于1，一直出队；退出循环后根据队是否为空返回 #（无只出现一次的字符） 或队头元素（第一个只出现一次的字符）。

- 代码实现

```
class Solution{
    int count[128];
    queue <char> que;
public:
    void Insert(char ch){
        if(count[ch - ' '] == 0) //' '，对应ASCII码的0
            que.push(ch); //第一次出现的字符入队
         ++count[ch-' ']; //统计出现的次数
    }
    char FirstAppearingOnce(){
        while(!que.empty()&&count[que.front() - ' '] != 1){ //队不为空且多次出现的字符出队
            que.pop();
        }
        return que.empty()? '#':que.front();
    }
};
};
```