第四章 字符串

我保证没有抄袭别人作业

1. 回文字符串

判断一个字符串 s 是否为回文字符串（即正读和倒读相同）。请简要说明实现算法，并指出时间复杂度和空间复杂度。

解:

```
bool is_huiwen(char* s){
    if(s == NULL) return false;
    int length = strlen(s);
    for(int i = 0; i < length - i - 1; i += 1){
        if(s[i] != s[length - i - 1]){
            return false;
        }
    }
    return true;
}
```

即从两边向中间比对,在对称位置上的字符是否相等

(如果题目认为长度为零的字符串不是回文串的话,还需要再加上一句判断)

时间复杂度:O(n)　　空间复杂度:O(1)

2. 求出下列模式串的 next 值

模式串 t='abcdaabcddaba';

模式串 s='XYYXYZXZXYYXZ'

```
abcdaabcddaba    -1 0 0 0 -1 1 0 0 0 4 -1 0 2
XYYXYZXZXYYXZ    -1 0 0 -1 0 2 -1 1 -1 0 0 -1 4
```

求解代码(优化后的 next 数组):

```
char a[N] = {};
while(cin >> a){
    int n[N] = {-1}, k = -1, idx = 0, length = strlen(a);
    while(idx < length){
        while(k >= 0 && a[idx] != a[k])
            k = n[k];
        k += 1;
        idx += 1;
        if(idx == length)
            break;
        if(a[idx] == a[k])
            n[idx] = n[k];//优化
        else
            n[idx] = k;
    }
}
```

3. 反转字符串

已知一个字符串 A，由字母数字和“.”组成，存储在定长的数组中。我们需要将该字符串以“.”为分界位置，反转该字符串。例如：portal.pku.edu.cn，转化成 cn.edu.pku.portal。设计一个算法，并分析算法的时间复杂度和空间复杂度。

```
char a[N] = {};
char b[N] = {};
int b_idx = 0;

void convert(int a_idx){//一开始传入 strlen(a)-1
   int length = a_idx + 1;
   if(length <= 0){
      return ;
   }
   while(a_idx >= 0 && a[a_idx] != '.'){
      a_idx -= 1;
   }
   for(int i = a_idx + 1; i < length; i += 1){
      b[b_idx++] = a[i];
   }
   if(a_idx < 0){//a[0] != '.',不用补'.'
      b[b_idx] = 0;//字符串结尾
      return ;//转换结束
   }else{//b 补上一个'.',再加上结尾的'\0'
      a[a_idx] = 0;//a 之前的部分已经转换完成,舍弃掉
      b[b_idx++] = '.';
      b[b_idx] = 0;
   }
   convert(a_idx - 1);
}
```

测试用例:

```
portal.pku.edu.cn    ==>   cn.edu.pku.portal
.pku.pkueecs.        ==>   .pkueecs.pku.
..pku.               ==>   .pku..
.pku...              ==>   ...pku.
.pku.course...       ==>   ...course.pku.
....thu.dean.....    ==>   .....dean.thu....
.pku..pkupku...      ==>   ...pkupku..pku.
```

每次将 a 中最末尾的子串拼接到 b 的尾巴上，直到 a 串长度为零.

空间复杂度:开辟了与原串同样长度的字符串(O(n)),并且每一次调用函数时开辟 O(1)的空间,最坏情况下共调用 n 次,所以空间复杂度为 O(n) + O(n) = O(n).

时间复杂度:整个函数调用过程中 a_idx 从 a 的尾部移动到 a 的头部,同时 b_idx 从 0 增加到 n,这是 O(n)的时间开销；函数执行过程中,其他步骤产生 O(1)的开销,整个调用过程最多产生 O(n)的开销(函数调用最多 n 次),故时间复杂度为 O(n).

4. 找出下列程序的错误，并解释它为什么是错的。

1)
```
void test1(){
    char str[10];
    char* str1 = “0123456789”;
    strcpy(str, str1);
    std::cout<<str<<’\n’;
}
```

2)
```
void test2(){
    char str[10], str1[10];
    for (int i=0; i<10; i++){
       str1[i] = ‘a’;
    }
    strcpy(str, str1);
    std::cout<<str<<’\n’;
}
```

3)
```
void test3(char* str1){
    char str[10];
    if (strlen(str1)<=10)
       strcpy(str, str1);
    std::cout<<str<<’\n’;
}
```

解:

1) str 只能容纳 9 个字符而 str1 长度超过了 9,进行 strcpy 时会出错

2) str1 应以’\0’结尾,并且最多容纳 9 个字符,test2()中通过循环让 str1 容纳了 10 个字符,且没有在结尾加上’\0’,进行 strcpy 到 str 时也会出错

3) str 最多容纳九个字符,而 test3()中长度为 10 的情况下仍可以拷贝到 str 中(此外在进入 strlen 之前最好检查 str1 是否为 NULL)

5. 删除字符

删除字符串中的“b”和“ac”，需要满足如下的条件：

1) 字符串只能遍历一次

2) 不能够使用额外的空间，即 O(1)的空间。

例如：acbac ==> ""；ababac ==> aa；bbbbd ==> d；aaccac ==> “ ”

请设计算法，得到输出的字符串。

```
char a[N] = {};
while(cin >> a){
   int length = strlen(a);
   int idx = 0;
   while(idx < length){//遍历字符串
      int b_cnt = 0;
      while(a[idx + b_cnt++] == 'b');
      b_cnt -= 1;//后面(包括 a[idx])有 b_cnt 个连续的 b
      if(b_cnt){//删除后面的 b
         for(int i = idx; i + b_cnt <= length; i += 1){
            a[i] = a[i + b_cnt];//把结尾的 0 也拷贝过来
         }
         length -= b_cnt;
      }
```

```
        if(idx >= 1 && a[idx] == 'c' && a[idx - 1] == 'a'){
           //删除前面的 ac
           for(int i = idx - 1; i + 2 <= length; i += 1){
              a[i] = a[i + 2];
           }
           length -= 2;
           idx -= 2;
        }
        idx += 1;//检查后面一个字符
    }
    cout << "a: "  << "\""<< a  << "\""<< endl;//测试用
}
```

测试用例:

aaaccc	a: ""
abc	a: ""
aaccac	a: ""
ababac	a: "aa"
aaabbbcccc	a: "c"
bbbbd	a: "d"

每次删去后面的 b,之后将后面的字符提到前面,再检查是否有连续的 ac,有也删掉,直到遍历到字符串结尾