字符串逆序

一：字符颠倒

全部逆置

如:

将“abcdef”逆置为“fedcba”

改变字符数组内容：

```
//改变字符数组
#include <stdio.h>
#include <string.h>
#include <assert.h>
char* reverse(char* s, char* t)
{
    char* p = s;
    assert(s && t);//
    while (s < t)
    {
        char tmp = *s;
        *s++ = *t;
        *t-- = tmp;
    }
    return p;
}
int main()
{
    char s[] = "abcdef";
    int slen = strlen(s);
    char * tmp = reverse(s, s + (slen - 1));
    printf("%s", s);
    return 0;
}
```

左右旋转：

实现一个函数，可以左旋字符串中的k个字符。

例如：

ABCD左旋一个字符得到BCDA

ABCD左旋两个字符得到CDAB

改变字符数组内容：

思路：

先把字符数组整体逆置，再分别逆置左右部分

```
#include <stdio.h>
#include <string.h>
#include <assert.h>
```

```
//实现一个函数，可以左旋字符串中的k个字符。
//例如：
//ABCD左旋一个字符得到BCDA
//ABCD左旋两个字符得到CDAB

void reverse(char *s, char *t) {
    while (s < t)
    {
        char tmp = *t;
        *t-- = *s;
        *s++ = tmp;
    }
}
void Reverse_arr(char * str, int lag, int slen) {
    assert(str && (lag < slen));
    if (0 == lag || lag == slen)
        return;
    //全逆置
    reverse(str, str + slen - 1);
    //逆置左部分
    reverse(str, str + slen - 1 - lag);
    //逆置右部分
    reverse(str + slen - lag, str + slen - 1);
    return;
}
int main()
{
    char s[100] = { 0 };
    int count = 0, slen = 0;

    printf("请输入字符串内容：\t\t");
    gets(s);
    slen = strlen(s);
    printf("\n请输入颠倒的字符个数\t\t");
    scanf("%d", &count);

    Reverse_arr(s, count, slen);

    printf("%s", s);
    return 0;
}
```

二：单词逆置

单个空格为间隔：

> 倒置字符串_牛客题霸_牛客网 (nowcoder.com)
>
> 将一句话的单词进行倒置，标点不倒置。比如 I like beijing. 经过函数后变为：beijing. like I

只改变输出格式：

```
#include<stdio.h>
#include<string.h>
void my_print(char* sH, char* sT, int count)
{
    //从后遍历数组
    while (sT != sH)
```

```
    {
        //输出单词条件
        if (*sT == ' ')
        {
            for (int i = 1; i <= count; i++)
            {
                printf("%c", *(sT + i));
            }
            printf(" ");
            //待输出单词长度count归零
            count = 0;
        }
        //记录单词长度
        else
        {
            count++;
        }
        sT--;
    }
    //从头遍历特殊判断
    //打印原序第一个单词
    for (int i = 0; i <= count; i++)
    {
        printf("%c", *(sT + i));
    }
}
int main()
{
    char str[100] = { 0 };
    gets(str);
    my_print(str, str + strlen(str) - 1, 0);
    return 0;
}
```

改变字符数组内容：

思路：

先把字符串整体逆置，再逆置局部各个单词

```
#include<stdio.h>
#include<string.h>
#include<assert.h>
char* reverse(char* s, char* t)
{
    char* p = s;
    assert(s && t);//1
    while (s < t)
    {
        char tmp = *s;
        *s++ = *t;
        *t-- = tmp;
    }
    return p;
}
int main()
{
    char s[100] = {0};
```

```
    gets(s);
    char* str = s;
    int slen = strlen(s);
    //逆置字符串
    reverse(s, s + slen - 1);
    while (*str != '\0')
    {
        char* tmp = str;
        //遍历单词，' '和'\0'代表单词和字符串结束
        while (*str && *str != ' ')
            str++;
        //逆置单词
        reverse(tmp, str - 1);

        //从尾遍历特殊情况（可用do...while...优化）
        //判断是否结束
        if (*str == '\0')
            break;
        //进入下一个单词
        else
            str++;
    }
    printf("%s", s);
    return 0;
}
```

从输入解决：

若题目给定了合法字符，则可通过只接受保存合法字符的方式解决单词倒序问题，如下题。

不定个非法字符为间隔：

单词倒排*牛客题霸*牛客网 (nowcoder.com)

对字符串中的所有单词进行倒排。

说明：

1、构成单词的字符只有26个大写或小写英文字母；

2、非构成单词的字符均视为单词间隔符；

3、要求倒排后的单词间隔符以一个空格表示；如果原字符串中相邻单词间有多个间隔符时，倒排转换后也只允许出现一个空格间隔符；

4、每个单词最长20个字母；

只改变输出格式：

```
#include<stdio.h>
#include<string.h>
void my_print(char* s, char* p)
{
    char* q = NULL;//单词头部
    int lag = 0;//待输出单词长度

    //从尾遍历
    while (p >= s)
    {
        //如果当前字符属于a~z，A~Z，lag++
```

```
        if ((*p >= 65 && *p <= 90) || (*p >= 97 && *p <= 122))
        {
            lag++;
        }
        //如果当前字符不属于有效字符，且待输出字符长度不为0时，才输出
        else if (lag != 0)
        {
            q = p + 1;
            for (int i = 0; i < lag; i++)
                printf("%c", *q++);
            printf(" ");
            //待输出长度lag归零
            lag = 0;
        }
        p--;
    }
    //开头字符为有效字符时，特殊判断
    q = p + 1;
    for (int i = 0; i < lag; i++)
        printf("%c", *q++);
    printf(" ");
}
int main()
{
    char s[10001] = { 0 };
    gets(s);

    //求字符数组有效长度
    int Len = strlen(s);

    //逆序输出
    my_print(s, s + Len - 1);

    return 0;
}
```

改变字符数组内容：

原数组：

非法字符过多，单词与单词之间非法字符个数也不确定，所以需要标识单词地址，待插入地址，和判断结束地址

```
#include<stdio.h>
#include<string.h>
#include<assert.h>
//逆置函数
char* reverse(char* s, char* t)
{
    char* p = s;
    assert(s && t);//1
    while (s < t)
    {
        char tmp = *s;
        *s++ = *t;
        *t-- = tmp;
    }
```

```
    return p;
}
//保存有效单词函数
void my_reverse(char* s, char* p)
{
    char* q = s;//字符数组应插入位置
    int count = 0;//待输出单词长度
    char* tmp = NULL;//存放单词地址（首元素地址）
    int lag = 0;//标识单词开头是否存在

    //字符数组已逆置，从头遍历，保存单词
    while (s <= p)
    {
        //如果当前字符属于a~z，A~Z，count++
        if ((*s >= 65 && *s <= 90) || (*s >= 97 && *s <= 122))
        {
            count++;
            //保存单词地址
            if (lag == 0)
            {
                lag = 1;
                tmp = s;
            }
        }
        //如果当前字符不属于有效字符，且待输出字符长度不为0时，才输出
        else if (count != 0)
        {
            //逆置单词
            reverse(tmp, s - 1);

            //如果 应插入位置 和 单词地址相同，则只需改变应插入位置，不需要把单词前移
            if (q == tmp)
            {
                //单词间隔统一为' '
                *s = ' ';
                //q 为空格的下一个元素地址
                q = s + 1;
            }
            //如果不相同，则需要前移单词
            else
            {
                for (int i = 0; i < count; i++)
                    *q++ = *tmp++;
                //间隔为' '
                *q++ = ' ';
            }

            //待输出长度count和标识归零
            count = 0;
            lag = 0;
        }

        s++;
    }
    //最后存在单词时，特殊判断
    if (count != 0)
    {
        //逆置单词
```

```
        reverse(tmp, s - 1);
        //不相等
        if (q != tmp)
        {
            for (int i = 0; i < count; i++)
            {
                *q++ = *tmp;
                //结束标志
                *tmp++ = '\0';
            }
        }
        //若相等，则单词结尾后一个就为结束标志
    }
    //不是单词，则变成结束标志
    else
        *q = '\0';
}
int main()
{
    char str[10001] = { 0 };
    gets(str);

    //求字符数组有效长度
    int Len = strlen(str);

    //全部逆序
    reverse(str, str + Len - 1);

    //保存有效单词
    my_reverse(str, str + Len - 1);

    //打印字符串
    printf("%s", str);
    return 0;
}
```

新建数组:

遍历保存即可

```
#include<stdio.h>
#include<string.h>
#include<malloc.h>
char* reverse(char* s, char* p)
{
    //申请空间保存有效数据
    char* tmp = (char*)calloc(p - s + 2, 1);
    //保存首地址
    char* newstr = tmp;

    int lag = 0;//待保存单词长度
    int count = 0;//新数组长度

    //从尾遍历
    while (p >= s)
    {
        //如果当前字符属于a~z，A~Z，lag++
```

```
        if ((*p >= 65 && *p <= 90) || (*p >= 97 && *p <= 122))
        {
            //待保存单词长度
            lag++;
            //数组长度
            count++;
        }
        //如果当前字符不属于有效字符，且待输出字符长度不为0时，才保存
        else if (lag != 0)
        {
            for (int i = 1; i <= lag; i++)
                *tmp++ = *(p + i);
            //间隔
            *tmp++ = ' ';
            //待保存长度lag归零
            //数组长度+1
            count++;
            lag = 0;
        }
        p--;
    }
    //开头字符为有效字符时，特殊判断
    if (lag)
    {
        for (int i = 1; i <= lag; i++)
            *tmp++ = *(p + i);

        *tmp = '\0';
    }
    //无效
    else
        *--tmp = '\0';

    //减小申请空间
    newstr = (char*)realloc(newstr, count + 1);
    return newstr;
}
int main()
{
    //申请堆区，可释放节省空间
    char* s = (char*)calloc(10001,1);
    gets(s);

    //求字符数组有效长度
    int Len = strlen(s);

    //新建数组保存数据
    char* tmp = reverse(s, s + Len - 1);
    free(s);
    s = tmp;

    printf("%s", s);
    return 0;
}
```

从输入解决：

合法字符已知，只接受保存合法字符

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
    char str[100][22];
    int i = 0;
    int x = 0;
    while (1)
    {
        //返回值为输入的个数
        //读取 (a-z|A-Z)，遇到非法字符就结束此次读取
        //每行保存一个有效单词
        x = scanf("%[a-z|A-Z]", str[i]);

        //行结束标志'\n'，代表 输入的一行的所有内容都已遍历
        //最后一个字符
        if (getchar() == '\n')
            break;

        //str[i]有内容才换行
        if (x)
            i++;//记录行数
        x = 0;
    }
    //倒序输出
    for (int j = i; j >= 0; j--)
        printf("%s ", str[j]);
    return 0;
}
```