学号	10212819108

武汉華夏理工学院

课　　程　　设　　计

课程名称 数据结构与算法

题　　目 判断一个字符串是否为中心对称的字符串

专　　业 软件工程

班　　级 软件1191Z

姓　　名 胡文卓

成　　绩 _____

指 导 老 师 汤　琳

2020 年 6 月 16 日至 2020 年 6 月 24 日

武汉华夏理工学院信息工程学院

课 程 设 计 任 务 书

课程名称：数据结构与算法课程设计　　指导教师：汤 琳

班级名称：计算机 1191z，软件 1191z　　开课院、系：计算机与网络工程系

一、课程设计目的与任务

《数据结构》课程设计是为训练学生的数据组织能力和提高程序设计能力而设置的增强实践能力的课程。目的：学习数据结构课程，旨在使学生学会分析研究数据对象的特性，学会数据的组织方法，以便选择合适的数据的逻辑结构和存储结构以及相应操作，把现实世界中的问题转换为计算机内部的表示和处理，这就是一个良好的程序设计技能训练的过程。提高学生的程序设计能力、掌握基本知识、基本技能，提高算法设计质量与程序设计素质的培养就是本门课程的课程设计的目的。

任务：根据题目要求，完成算法设计与程序实现，并按规定写出课程设计报告。

二、课程设计的内容与基本要求

设计题目：判断一个字符串是否为中心对称的字符串

〔问题描述〕设单链表中存放着 n 个字符组成的字符串，试设计程序判断其是否为中心对称的字符串。

〔基本要求〕以单链表作为存储结构，任意输入一个字符串，输出其是否为中心对称。

具体要完成的任务是：

A. 编制完成上述问题的 C 语言程序、进行程序调试并能得出正确的运行结果。

B. 写出规范的课程设计报告书；

三、学时分配进度安排

序号	设计内容	所用时间
1	选题及调研	1 天
2	功能及算法分析，编写程序	1 天
3	调试程序，运行系统	1 天
4	程序完成及撰写报告	1 天
5	答辩	1 天
合　　计		1 周

四、课程设计考核及评分标准

1.设计报告要求

课程设计报告要求逻辑清晰、层次分明、书写整洁。报告书应包括设计题目、设计思想、系统结构、数据结构说明及模块算法说明、使用说明书、测试结果分析，附录（程序清单）。

设计报告须每人一份，独立完成。

2.评分标准

课程设计考核将综合考虑学生的系统设计方案、运行结果、课程设计报告书的质量、态度、考勤、答辩情况等各因素。具体评分标准如下：

评分标准	分值
（1）设计方案正确，具有可行性、创新性；	20 分
（2）系统开发效果较好；	20 分
（3）课程设计报告书写规范、质量高；	30 分
（4）课程设计答辩时，回答问题流利且正确；	20 分
（5）态度认真、刻苦钻研、遵守纪律；	10 分
总分	100 分

注：成绩等级：优（90 分—100 分）、良（80 分—89 分）、中（70 分—79 分）、及格（60 分—69 分）、60 分以下为不及格。

五、指导时间

周次	星期二	星期三	星期四	星期四
第 19 周	第 7-8 节	第 7-8 节	第 1-2 节	第 5-8 节

执笔：　汤 琳　　日期：2020-6-09

审阅：　李小艳　　日期：2020-6-11

目　录

1. 设计题目

判断一个字符串是否为中心对称的字符串

2. 设计思想

看到我的课设题目“判断一个字符串是否为中心对称的字符串”我首先想到的是如何存储一个字符串。通过数据结构课程的学习，我了解到计算机中有顺序存储方式、链接存储方式、索引存储方式和散列存储方式。我首先考虑的顺序存储方式，但这样做并没有用到任何数据结构的知识，而且如果用字符数组存放字符串还要提前知道有几个字符串，不易扩展，故顺序存储方式不考虑。接下来我想到了链表这种数据结构，它可以自由的进行增加删除，而且不需要提前指定空间大小，只需要通过指针记录上一个节点的位置就行，易于扩展，并且还很好的用到了数据结构课程的知识。

那么接下来考虑下如何判断一个字符串是否是中心对称的，对此我有三种思路。

方法一：对于任意一个字符串而言，它的长度永远只有三种情况：为零（空），为奇数，为偶数。假设使用一个变量 sLen 来存放字符串的长度，可以分以下几种情况讨论，如果 sLen == 0，则该字符串为空，那么它必然是一个中心对称的字符串；如果 sLen % 2 == 0，则该字符串长度为偶数，那么只需要对比区间[0, sLen / 2 - 1]与区间[sLen / 2, sLen - 1]的两部分字符串是否相同即可；如果 sLen % 2 != 0，则该字符串长度为奇数，相对于字符串长度为偶数的情况，可以直接将奇数长度的字符串认为是偶数长度的字符串在 sLen / 2 处插入了一个字符，结合字符串长度时偶数的情况，可以发现插入的字符串不影响判断。三种情况的讨论参考图 1.1 字符串长度的三种情况。那么该方法的实现就是记录字符串长度，然后通过长度对比两个区间的字符是否完全相同即可。

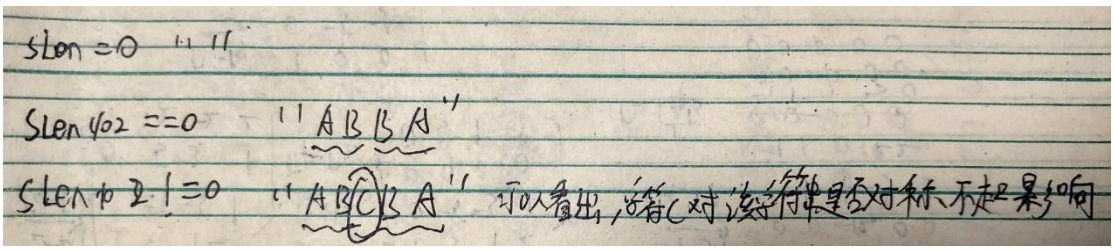

图 1.1 字符串长度的三种情况

方法二：如果一个字符串是中心对称的，那么该字符串正序与倒序应该都是相同的。比如字符串”ABBA”，它从左或从右看都是相同的。那么该方法的实现就是对原字符串进行一

次倒序遍历并且倒序后的字符串，再与原字符串进行对比。

方法三：该方法是基于方法一、方法二的改进，也是我的程序用的方法。首先考虑下方法一的缺点，因为我的程序是用链表存放字符串的，故而如果要知道字符串长度或者要知道当前是第几位字符，都需要在链表结构体里利用一个变量记录，而且还要分字符串奇偶长度，较为麻烦。而方法二虽然易于实现，但是却有个问题，如何存放倒序后的字符串，可以想到再利用一条链表记录，但这样显然会照成空间的浪费，当字符串很长的时候，这种方法无疑不能适用。而如果考虑下方法二的本质，即可发现完全不需要使用一个新的链表存放倒序后的字符串。同样的，将字符串长度设为 sLen，那么将字符串倒序，再进行对比，实际上就是用第 0 位的字符与第 sLen 位的字符进行对比，依次类推，可以得到第 1 位与第 sLen-2 位对比，第 2 位与第 sLen-3 为对比，这里同样要考虑字符串长度的问题，但实际上可以用两个指针来头尾字符的对比，用指针 lf 指向链表头，指针 rt 指向链表尾，然后对比 lf 与 rt 指向的节点存放的字符是否相同即可。方法三实质上是双指针算法的简单应用，运行流程参考图 1.2 双指针判断字符串是否中心对称。

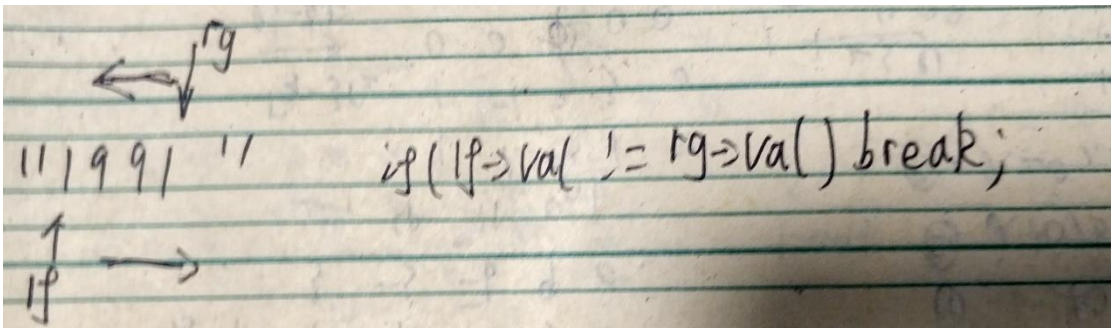

图 1.2 双指针判断字符串是否中心对称

选定方法三作为判断字符串是否中心对称还需要思考下使用链表如何实现。首先可以考虑普通单链表，但单链表只有一个指针，用来记录当前节点的后续节点。这显然是无法实现双指针的功能的，因为使用双指针不仅要前驱节点，也要后续节点。所以这里我选用了双链表来存储字符串，这样的话头结点的左指针可以向右移动，尾结点的右指针可以向左移动，即可实现双指针判断字符串是否中心对称的功能。双链表的预览参考图 1.3 双链表存放字符串。

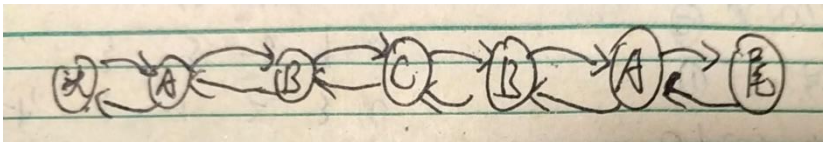

图 1.3 双链表存放字符串

3. 系统结构

程序运行流程图参考图 2.1 流程图。

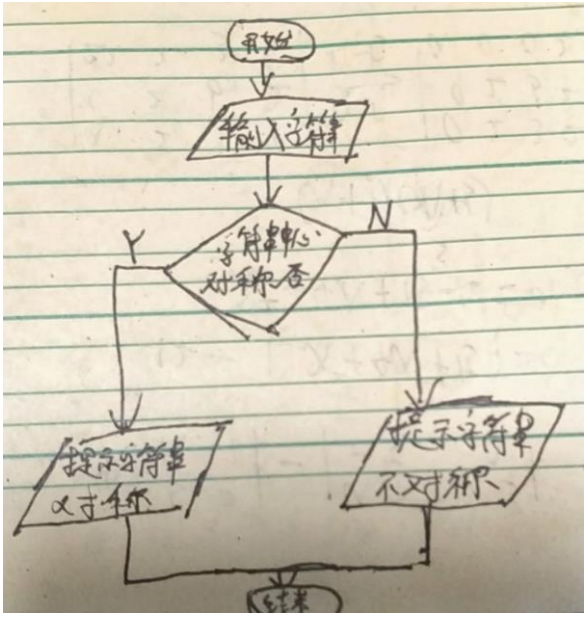

图 2.1 流程图

4. 数据结构说明和模块算法说明

4.1 链表结构体

该结构体用来存放节点数据，它提供了一个自定义的 DATA 变量，两个结构体指针。其中 val 变量存放的是当前节点的字符，left，right 指针则存放的上一个节点的地址和下一个节点的地址。结构体代码参考图 3.1 结构体代码。

```
typedef char DATA;
DATA input;
typedef struct OwO {
    DATA val;
    struct OwO *left, *right;
} LinkedList;
```

图 3.1 结构体代码

4.2 main 函数

该函数主要提供了接受用户输入数据，输出结果的功能。代码参考图 3.2 main 函数代码。其中 init_node()函数用来初始化一个结构体指针，add_node()用来进行连接链表的功能。

```
int main() {
    head = init_node(head), end = head;
    puts("Please input your string!(Enter ' ' or '\\n' to check result)");
    while(scanf("%c", &input) != EOF && input != '\n' && input != ' ') add_node();
    puts(solve());
    clear_node();
    return 0;
}
```

图 3.2 main 函数代码

4.3 init_node 函数

该函数主要用来初始化节点，它通过 malloc 函数给 LinkedList 类似的结构体指针分配内存，并且将新创建的节点的 val 属性设置为了字符’\0’，并且将当前节点的左右节点设置为空，用来防止指针地址未定义的错误。init_node 函数代码参考图 3.3 init_node 函数代码。

```
LinkedList* init_node() {
    LinkedList *list;
    list = (LinkedList*)(malloc(sizeof(LinkedList)));
    list->val = '\0', list->left = list->right = NULL;
    return list;
}
```

图 3.3 init_node 函数代码

4.4 add_node 函数

该函数用来连接新节点，它先创建了一个新节点 next，并且将当前输入的字符存入到当前节点的 val 变量中，再将尾结点的右指针指向 next 节点，将 next 的左指针指向 end 节点，再将 end 节点替换为 next 节点，这样就实现了双链表的尾插法。add_node 函数代码参考图 3.4 add_node 函数代码。

```
void add_node() {
    LinkedList *next = init_node();
    next->val = input, end->right = next, next->left = end, end = next;
}
```

图 3.4 add_node 函数代码

4.4 solve 函数

这个函数实现了判断当前字符串是否中心对称，并且返回一个字符串提示用户输入的字符串是否是对称字符串。它首先给 end 指针又加了个右指针，这主要是为了和 head 指针相对应，head 指针是一个空指针，它只是用来方便找到链表头，它的右指针指向的节点才是真正的有数据的头结点。这里用到的 print_str()函数就是输出遍用户输入的数据，用来方便的对比结果。而 lf 和 rt 指针就是两个指向链表头尾的指针，他们首先对比头尾节点的字符是否相同，如果不同就直接返回一个字符串提示，而如果相同，则将两个指针都向后移动一位，并重复进行上述过程，如果整个字符串都对比完了，则说明该字符串是中心对称的。solve 函数代码参考图 3.5 solve 函数代码。

```
char* solve() {
    end->right = init_node();
    LinkedList *lf = head->right, *rt = end;
    print_str(lf);
    while (lf != rt) {
        if (lf->val != rt->val) return "is not a huiwen string!";
        lf = lf->right, rt = rt->left;
    }
    return "is a huiwen string!";
}
```

图 3.5 solve 函数代码

5. 使用说明书

运行程序，首先会给出一个提示：“Please input your string!(Enter ' ' or '\n' to check result)”，它提示用户输入一个字符串，并且该字符串以空格或者换号结束。用户输入完字符串后程序会判断该字符串是否是中心对称字符串并给出一个提示：“用户输入的字符串 is (not) a huiwen string!”用来提示用户其输入的字符串是否为中心对称字符串。程序运行图参考图 4.1 程序运行参考。

```
Please input your string!(Enter ' ' or '\n' to check result)
191919
191919 is not a huiwen string!
```

图 4.1 程序运行参考

6. 运行结果

用户输入：191919

程序输出：191919 is not a huiwen string!

用户输入：ABCCBA

程序输出：ABCCBA is a huiwen string!

程序运行结果参考图 4.1 程序运行参考

7. 测试过程及结果分析

用户输入：191919

程序输出：191919 is not a huiwen string!

用户输入：ABCCBA

程序输出：ABCCBA is a huiwen string!

用户输入：114514

程序输出：114514 is not a huiwen string!

用户输入：thisisastring

程序输出：thisisastring is not a huiwen string!

用户输入：backkcab

程序输出：backkcab is a huiwen string!

8. 附录：源程序清单

```
#include <stdio.h>
#include <stdlib.h>
typedef char DATA;
DATA input;
typedef struct OwO {
    DATA val;
    struct OwO *left, *right;
} LinkedList;
LinkedList *head = NULL, *end = NULL;
LinkedList* init_node() {
    LinkedList *list;
    list = (LinkedList*)(malloc(sizeof(LinkedList)));
    list->val = '\0', list->left = list->right = NULL;
    return list;
}
void add_node() {
```

```
    LinkedList *next = init_node();
    next->val = input, end->right = next, next->left = end, end = next;
}
void clear_node() {
    while (head && end) {
        LinkedList *ht = head->right, *et = end->right;
        free(head), free(end);
        head = ht, end = et;
    }
}
void print_str(LinkedList *str) {
    while(str) printf("%c", str->val), str = str->right;
}
char* solve() {
    end->right = init_node();
    LinkedList *lf = head->right, *rt = end;
    print_str(lf);
    while (lf != rt) {
        if (lf->val != rt->val) return "is not a huiwen string!";
        lf = lf->right, rt = rt->left;
    }
    return "is a huiwen string!";
}
int main() {
    head = init_node(head), end = head;
    puts("Please input your string!(Enter ' ' or '\\n' to check result)");
    while(scanf("%c", &input) != EOF && input != '\n' && input != ' ') add_node();
    puts(solve());
    clear_node();
    return 0;
}
```

课程设计成绩评定表

课程设计题目	

课程设计学生答辩或质疑记录：

问：两个指针判断字符串是否对称的思路是什么？

答：对称字符串翻转后一定和原字符串相同，故而可以用第一个位置的字符与最后一个位置的字符进行对比，以此类推，直到两个指针走到了相同的位置。

问：几个函数的作用是什么？

答：add_node()函数，用尾插法新增字符；clear_node()函数，清空动态分配的链表内存；solve()函数，利用头尾两个指针判断字符串是否对称并返回提示。

问：链表结构体存放了那些信息？

答：链表结构体存放了当前节点的字符，前驱节点的地址和后续节点的地址。

评　分　依　据	分 值	评分成绩
（1）设计方案正确，具有可行性、创新性；	20 分	
（2）系统开发效果较好；	20 分	
（3）课程设计报告书写规范、质量高；	30 分	
（4）课程设计答辩时，回答问题正确；	20 分	
（5）态度认真、刻苦钻研、遵守纪律；	10 分	
总 分	100 分	

最终评定等级为：

指导老师签字：

2020 年 6 月 29 日

注明：1 答辩记录至少应为 3 个；2.最终评定成绩（以优、良、中、及格、不及格评定）