

中南林业科技大学

课程设计报告

设计名称： 简单行编辑器

姓 名： 段志超 学 号： 20192902

专业班级： 2019 级计算机科学与技术 3 班

系（院）： 计算机与信息工程学院

设计时间： 2020~2021 学年第一学期

设计地点： 电子信息楼 机房

指导教师评语：

成绩：

项目	内容	得分
答辩情况	把握问题、分析问题以及解决问题等诸多方面的表达能力（20%）	
程序正确性	包括程序整体结构是否合理、编程风格是否规范等（30%）	
工作态度	学生的工作态度、独立工作能力（30%）	
课程设计报告	课程设计报告（含课程设计心得）（20%）	
总分		

签名： _____

年 月 日

1. 课程设计目的

- 1、训练学生灵活应用所学数据结构知识，独立完成问题分析，结合数据结构理论知识，编写程序求解指定问题。
2. 初步掌握软件开发过程的问题分析、系统设计、程序编码、测试等基本方法和技能；
3. 提高综合运用所学的理论知识和方法独立分析和解决问题的能力；
4. 训练用系统的观点和软件开发一般规范进行软件开发，巩固、深化学生的理论知识，提高编程水平，并在此过程中培养他们严谨的科学态度和良好的工作作风。

2. 课程设计任务与要求：

任务

简单行编辑器

[问题描述]

设计一个简单的行编辑程序，输入一页文字，程序可以统计出文字、数字、空格的个数。静态存储一页文章，每行最多不超过 80 个字符，共 N 行。存储结构使用线性表，文字中可以是 大写、小写的英文字母、任何数字及标点符号。

[基本要求]

- (1) 分别统计出其中英文字母数和空格数及整篇文章总字数；
- (2) 统计某一字符串在文章中出现的次数，并输出该次数；
- (3) 删除某一子串，并将后面的字符前移。
- (4) 输出形式：
 - 1) 分行输出用户输入的各行字符；
 - 2) 分 4 行输出“全部字母数”、“数字个数”、“空格个数”、“文章总字数”
 - 3) 输出删除某一字符串后的文章；

要求：

- 1、在处理每个题目时，要求从分析题目的需求入手，按设计抽象数据类型、构思算法、通过设计实现抽象数据类型、编制上机程序和上机调试等若干步骤完成题目，最终写出完整的分析报告。前期准备工作完备与否直接影响到后序上机调试工作的效率。在程序设计阶段应尽量利用已有的标准函数，加大代码的重用率。
- 2、设计的题目要求达到一定工作量（300 行以上代码），并具有一定的深度和难度。
- 3、程序设计语言推荐使用 C/C++，程序书写规范，源程序需加必要的注释；
- 4、每位同学需提交可独立运行的程序；
- 5、每位同学需独立提交设计报告书（每人一份），要求编排格式统一、规范、内容充实，不少于 10 页（代码不算）；
- 6、课程设计实践作为培养学生动手能力的一种手段，单独考核。

3. 课程设计说明书

一 需求分析

要求静态存储一页文字, 由于文字的多少不定, 因此采用链表存储较为合适。需要熟悉链表的插入、遍历等操作。

要求对文章中文字、数字、空格个数、字母数进行统计。这便需要非常熟悉链表的遍历, 除此之外对数字、空格、字母、单词的判断也尤为重要, 尤其是单词的统计。对于空格、数字、字母标准库已有函数可以直接判断, 而单词的判断算法则需要自己实现。

要求统计某一字符串出现的次数。由于整篇文章内容可能较多, 如果对整篇文章直接进行子串暴力匹配, 效率必然十分低, 也是不切实际的。KMP 算法刚好能解燃眉之急, 对文章的每一行 (即链表的每一个节点) 使用 KMP 算法模式匹配要查找的字符串, 并统计次数。

要求删除某一子串。删除子串无非是查找子串后多一步删除操作, 不过这个删除是对节点而言, 即删除每一个节点 (每一行) 中该子串的出现。

要求输出文章, 即为链表的遍历。

虽然题目没有要求文件读写, 但为了扩展其实用性, 我期望在标准输入输出文章的基础上增加文件输入输出, 即从文件读取文章, 做相应修改后存回原文件或新文件。因此需要对 IO 流知识较为熟悉。

考虑了很多方面, 我决定充分利用 STL 以及 c++ 实现好的数据结构, 在此基础上实现编辑器。这需要对 STL 的各容器、适配器的使用及源码比较熟悉, 才能综合课题的需求设计出效率高的数据结构。

利用 string 存储每一行文本, 利用元素类型为 string 的 list 双向链表来抽象每一页, 对于单词的查找还可以使用红黑树 map 来存储单词以提高查找效率。因此需要对这些容器的用法和内部结构较为熟悉。

除此之外还要设计完好的用户交互界面, 这需要熟练掌握语言方面的格式化输入输出。

综上, 做这个课题, 要具备的知识就是链表的基本操作算法, 串的基本使用和操作, 串的 KMP 模式匹配算法, 熟悉相关容器, 必要的 C 或者 C++ 知识, 以及丰富的程序调适经验。

二 概要设计

我将数据结构的定义以及相关算法函数的声明放在头文件 editor.h 中, 并将函数的定义以及算法的实现放在 cpp 文件 editor.cpp 中, 而将主函数以及用户交互界面相关控制函数放在 main.cpp 中。

对于整个课题的实现, 整体逻辑并不复杂, 但如果完全使用 c 语言基础语法来实现, 将所有数据结构全部实现一遍 (这包括 string 串类型、list 链表类型等等), 然后再在此基础上实现编辑器, 想必工作量是非常巨大的。另一方面, 考虑到我的能力, 这不仅会耗费很多时间, 而且最后的成果代码方面必然会非常复杂。而 STL 中已经实现好了很多经典的数据结构, 在本次课题中也能用得上, 再是在前一个课题中已经手写了链表数据结构, 有了一定基础, 这个课题不妨直接使用 STL 来提高产出。

对于编辑器的实现, 我使用单链表存储文章, 并增加对单链表 (文章) 的操作 (成员函数), 将它们封装为一个编辑器 struct。并将数据 (存储的文章) 和对数据的操作分离开来, 避免对数据的直接访问。

程序利用了 c++ 的面向对象属性, 提高代码的重用率。大量使用了容器或数据类型对应的元素类型, 而不是基本的数据类型, 便于以后数据结构元素发生变化时重定义。

编辑器数据结构定义如下:

```
struct lineList {  
public:  
    lineList() {}  
  
    using listType = list<string>;  
  
    /* 成员函数-函数声明 */
```

```
// 清空编辑器
void clear() { L.clear(); line = 0; }

// 编辑器是否为空
bool empty() { return L.empty(); }

// 存储文章
istream& read(istream& in = cin);
// 输出文章
ostream& print(ostream& out = cout);

// 统计英文字母数、数字数、空格数及总字数
void word_count();
// 统计某一字符串出现次数
unsigned string_count(const listType::value_type t,
    unordered_map<listType::size_type, string>& smatch);

// 删除某一子串的所有出现
void remove_substr(listType::value_type remove_str);
// 删除某一子串在指定行中的出现
void remove_substr(listType::value_type remove_str,
    listType::size_type n);
// 按行删除指定区间
void remove_line(listType::size_type begin_line,
    listType::size_type end_line = 0);

unsigned get_alpha() { return alpha; }
unsigned get_digit() { return digit; }
unsigned get_space() { return space; }
unsigned get_word() { return word; }

private:
    listType L;
    listType::size_type line = 0;

    unsigned alpha = 0;
    unsigned digit = 0;
    unsigned space = 0;
    unsigned word = 0;

};
```

三 详细设计

1) 宏定义, 定义全局变量、结构体:

```
#ifndef EDITOR_H_
#define EDITOR_H_

#include <iostream>
#include <string>
#include <list>
#include <unordered_map>
#include <iomanip>
using namespace std;

/* 非成员函数-函数声明 */
// KMP算法
void getNext(const string& t, string::size_type next[]);
string::size_type KMP(const string& s, const string& t);

/* 数据结构 */
struct lineList {
public:
    lineList() {}

    using listType = list<string>;

    /* 成员函数-函数声明 */

    // 清空编辑器
    void clear() { L.clear(); line = 0; }
    // 编辑器是否为空
    bool empty() { return L.empty(); }

    // 存储文章
    istream& read(istream& in = cin);
    // 输出文章
    ostream& print(ostream& out = cout);

    // 统计英文字母数、数字数、空格数及总字数
    void word_count();
    // 统计某一字符串出现次数
    unsigned string_count(const listType::value_type t,
        unordered_map<listType::size_type, string>& smatch);

    // 删除某一子串的所有出现
    void remove_substr(listType::value_type remove_str);
```

```
// 删除某一子串在指定行中的出现
void remove_substr(listType::value_type remove_str,
    listType::size_type n);
// 按行删除指定区间
void remove_line(listType::size_type begin_line,
    listType::size_type end_line = 0);

unsigned get_alpha() { return alpha; }
unsigned get_digit() { return digit; }
unsigned get_space() { return space; }
unsigned get_word() { return word; }

private:
    listType L;
    listType::size_type line = 0;

    unsigned alpha = 0;
    unsigned digit = 0;
    unsigned space = 0;
    unsigned word = 0;
};

/* 用户交互界面处理函数-函数声明 */
void select();
// 输入文章
bool Input();
// 显示文章内容
bool Display();
// 统计文章内容
bool Statistic();
// 查找字符串, 并统计出现次数
bool Lookup();
// 删除字符串
bool Deletestr();
// 删除行
bool Deletelin();
// 清空文章
bool Clearup();
// 保存文章
bool Save();
// 重新输入
bool Reenter();
// 清空输入区
```

```
void clear();
```

2) 主函数处理用户选择的算法如下:

```
int choose = 1;
while (choose) {
    select();
    cin >> choose;
    //clear();
    if (choose >= 2 && choose <= 8 && line_editor.empty()) {
        cout << "\n请先输入文章! \n";
        continue;
    }

    switch (choose)
    {
    case 1:
        if (Input()) {
            cout << "\n文章输入完成, 请继续操作: \n";
            //select();
            //cin >> choose; clear();
        }
        else{
            cout << "\n输入失败, 请重新操作:\n";
            line_editor.clear();
            //select();
            //cin >> choose; clear();
        }
        break;

    case 2:
        if (!line_editor.empty() && Display()) {
            cout << "\n输出完成, 请继续操作: \n";
            //select();
        }
        break;

    case 3:
        if (!line_editor.empty() && Statistic()) {
            cout << "\n统计完成, 请继续操作: \n";
            //select();
        }
        break;

    case 4:
```

```
    if (Lookup()) {
        cout << "\n查找完成, 请继续操作:\n";
    }
    else
        cout << "\n查找失败, 请重新操作\n";
    break;

case 5:
    if (Deletestr()) {
        cout << "\n删除完成, 请继续操作\n";
    }
    else
        cout << "\n删除失败, 请重新操作\n";
    break;

case 6:
    if (Deletelin()) {
        cout << "\n删除完成, 请继续操作\n";
    }
    else
        cerr << "\n删除失败! 请重新操作\n";
    break;

case 7:
    if (Clearup()) {
        cout << "\n清空完成, 请继续操作\n";
    }
    break;

case 8:
    if (Save()) {
        cout << "\n保存完成, 请继续操作\n";
    }
    else
        cout << "\n保存失败, 请重新操作\n";
    break;

case 9:
    if (Reenter()) {
        cout << "\n已清空编辑器, 请重新输入文章: \n";
        if (Input()) {
            cout << "\n文章输入完成, 请继续操作: \n";
        }
        else {
            cout << "\n输入失败, 请重新操作:\n";
        }
    }
}
```



```
    }  
    }  
    break;  
case 0:  
    break;  
default:  
    cerr << "\n错误指令, 请输入 0~9 的整数! \n";  
    break;  
}  
  
}
```

3) 静态存储文章的算法如下:

// 静态存储文章

```
istream& lineList::read(istream& in)  
{  
    listType::value_type temp;  
    // 存入当前行  
    while (getline(in, temp)) {  
        L.push_back(temp);  
        ++line;  
    }  
  
    return in;  
}
```

4) 输出文章的算法如下:

// 输出文章

```
ostream& lineList::print(ostream& out)  
{  
    listType::size_type lcount = 0;  
    listType::const_iterator it = L.cbegin();  
    while (it != L.cend()) {  
        //fprintf(out, "%3d. ", ++lcount);  
        //if(out==cout)  
        out << setw(2) << ++lcount << ". "; //输出显示行号以便操作  
        out << *it++ << endl;  
    }  
    return out;  
}
```

5) 统计字符串出现次数的算法如下:

// 统计字符串出现次数

```
unsigned lineList::string_count(const listType::value_type t,
```

```
unordered_map<listType::size_type, string> &smatch) {

    unsigned scount = 0; // 字符串出现次数
    listType::size_type lcount = 0; // 当前行
    // unordered_map<int, string> smatch; // 匹配子串所在行的行号和行内容构成的哈希map

    listType::value_type s;
    string::size_type kmp_index;
    listType::const_iterator it = L.cbegin();
    while (it != L.cend()) {
        ++lcount;
        //s = *it; // 当前行字符串
        //kmp_index = string::npos; // 匹配子串在主串中的位置
        //
        //// 当前行中可能有多个子串匹配,
        //// 每次匹配成功后更改主串, 继续匹配
        //while ((kmp_index = KMP(s, t)) != string::npos) {
        //    // 匹配到该子串, 计数加一
        //    ++scount;
        //    // 并将该行添加到哈希map中
        //    smatch.insert(make_pair(lcount, *it));

        //    // 更新主串
        //    if (kmp_index + t.length() < s.length())
        //        s = s.substr(kmp_index + t.length(), s.length() - (kmp_index + 1));
        //}

        kmp_index = 0; // 匹配子串在主串中的位置
        while ((kmp_index = (*it).find(t, kmp_index)) != string::npos) {
            // 匹配到该子串, 计数加一
            ++scount;
            // 并将该行添加到哈希map中
            smatch.insert(make_pair(lcount, *it));

            // 更新下次匹配位置
            if (kmp_index + t.length() < s.length())
                kmp_index += t.length();
            else
                break;
        }
        ++it;
    }

    return scount;
}
```

//求两集合并集

```
bool Union(List L1, List L2, List& L3) {
    PNode node_l2 = L2.head;
    Copy(L1, L3);

    for (int i = 1; i <= L2.length; i++) {
        node_l2 = node_l2->next;
        if (!LocateElem(L3, node_l2->data)) {
            Append(L3, node_l2->data);
        }
    }
    sort(L3);
    return OK;
}
```

5) 删除子串的算法如下:

// 删除某一子串

// 删除子串在全文的一切出现

```
void lineList::remove_substr(listType::value_type remove_str)
{
    listType::value_type::size_type kmp_index;
    listType::iterator it = L.begin();
    while (it != L.end()) {

        if (*it == remove_str) {
            // 如果要删除的字符串与当前行相同, 直接删除整行
            L.erase(it);
        }
        else {
            // 否则从当前行匹配子串并删除
            //kmp_index = string::npos; // 匹配子串在主串中的位置
            //while ((kmp_index = KMP(*it, remove_str)) != string::npos) {
            //    (*it).erase(kmp_index, remove_str.length());
            //}

            kmp_index = 0; // 匹配子串在主串中的位置
            while ((kmp_index = (*it).find(remove_str, kmp_index)) != string::npos) {
                (*it).erase(kmp_index, remove_str.length());
                kmp_index = 0;
            }
        }

        ++it;
    }
}
```

```
}  
// 重载版本: 删除子串在指定行的出现  
void lineList::remove_substr(listType::value_type remove_str,  
    listType::size_type n)  
{  
    listType::size_type lcount = 0; // 当前行  
  
    listType::value_type::size_type kmp_index;  
    listType::iterator it = L.begin();  
    while (it != L.end()) {  
        // 如果当前行不是指定行, 不做处理  
        if (++lcount != n)  
            continue;  
  
        if (*it == remove_str) {  
            // 如果要删除的字符串与当前行相同, 直接删除整行  
            L.erase(it);  
        }  
        else {  
            // 否则从当前行匹配子串并删除  
            //kmp_index = string::npos; // 匹配子串在主串中的位置  
            //while ((kmp_index = KMP(*it, remove_str)) != string::npos) {  
            //    (*it).erase(kmp_index, remove_str.length());  
            //}  
  
            kmp_index = 0; // 匹配子串在主串中的位置  
            while ((kmp_index = (*it).find(remove_str, kmp_index)) != string::npos) {  
                (*it).erase(kmp_index, remove_str.length());  
            }  
        }  
        // 处理完毕, 跳出循环  
        if (lcount == n)  
            break;  
  
        ++it;  
    }  
}
```

5) 删除行的算法如下:

```
// 删除某一行  
// 按行号删除  
void lineList::remove_line(listType::size_type begin_line,  
    listType::size_type end_line) {  
  
    // 如果只有一个行号, 即只删除一行
```

```
if (end_line == 0)
    end_line = begin_line;
listType::size_type lcount;
listType::iterator it;
for (it = L.begin(), lcount = 1;
     it != L.end() && lcount <= end_line; ++lcount) {
    if (lcount >= begin_line)
        it = L.erase(it);
    else
        ++it;
}
```

5) KMP 模式匹配子串的算法如下:

/* 非成员函数-函数定义 */

// 利用修正的KMP算法查找子串

```
void getNext(const string& t, string::size_type next[]) {
    string::size_type j = 0, k = string::npos;
    next[0] = string::npos;
    while (j < t.length()) {
        if (k == string::npos || t.at(j) == t.at(k)) {
            ++j;
            ++k;
            if (t.at(j) != t.at(k))
                next[j] = k;
            else
                next[j] = next[k];
        }
        else
            k = next[k];
    }
}

/*
 * KMP 模式匹配子串
 * 若匹配成功, 返回第一次匹配主串位置
 * 匹配失败, 返回 string::npos
 */
string::size_type KMP(const string& s, const string& t) {
    // /*string::size_type*/ int const max_size = max(s.length(), t.length());
    string::size_type next[80];
    string::size_type i=0, j=0;
    getNext(t, next);
    while (i < s.length() && j < t.length()) {
        if (j == string::npos || s.at(i) == t.at(j)) {
            ++i;

```

```
        ++j;
    }
    else
        j = next[j];
}
if (j >= t.length())
    return i - t.length();
else
    return string::npos;
}
```

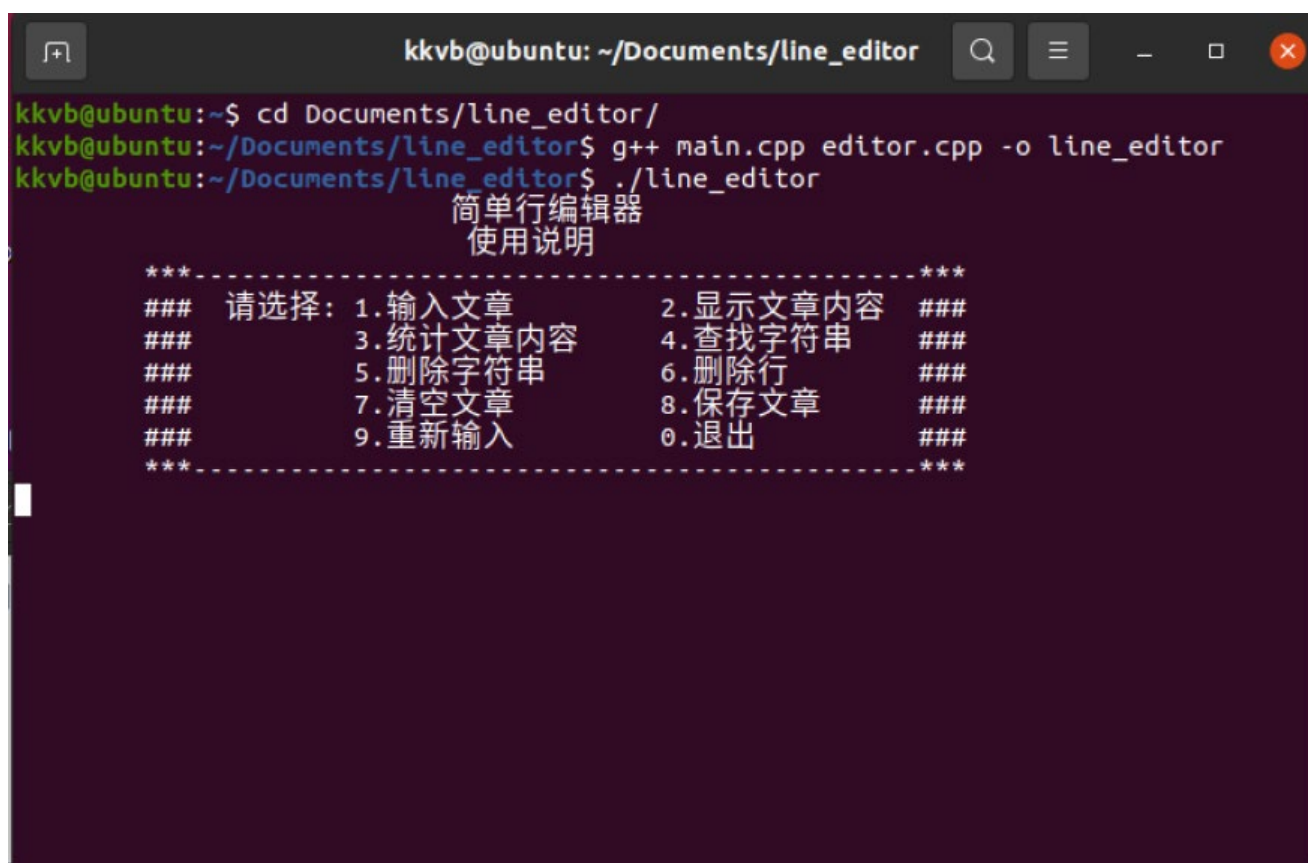
四 设计与调试分析

从上面的算法和调用关系可以看出, 这个程序的基本样子已经非常的清楚, 但是真正的程序中还要考虑各种限制条件。例如在查找的过程中, 可能不存在要查找的信息, 就要给出不存在此信息的提示等。

还有就是涉及到返回值得问题和程序中所要用到的变量的问题。在调试的过程中所遇到的问题很多, 结果最后我还是不够仔细, 没有检查出全部问题, 在答辩时被老师发现了问题。当前版本已修复。

五 用户手册

- 1 本程序可以在 Visual Studio2019 环境下运行。
- 2 在 Visual Studio 中打开该项目, 直接运行即可。
- 3 本项目所有源程序完全符合 c11/c++11 标准, 因此也可以使用支持该标准的其他编译器如 g++编译通过并生成可执行文件。
- 4 务必将 main.cpp 和 editor.cpp 联合编译, 以下为 gcc9.3.0 测试示例, 编译运行开始界面如下:



```
kkvb@ubuntu: ~/Documents/line_editor
kkvb@ubuntu:~$ cd Documents/line_editor/
kkvb@ubuntu:~/Documents/line_editor$ g++ main.cpp editor.cpp -o line_editor
kkvb@ubuntu:~/Documents/line_editor$ ./line_editor
简单行编辑器
使用说明
***-----***
### 请选择: 1.输入文章      2.显示文章内容    ###
###          3.统计文章内容  4.查找字符串    ###
###          5.删除字符串    6.删除行          ###
###          7.清空文章      8.保存文章        ###
###          9.重新输入      0.退出            ###
***-----***
```

六 测试成果

1. 文章输入测试:

```
kkvb@ubuntu: ~/Documents/line_editor
kkvb@ubuntu:~$ cd Documents/line_editor/
kkvb@ubuntu:~/Documents/line_editor$ g++ main.cpp editor.cpp -o line_editor
kkvb@ubuntu:~/Documents/line_editor$ ./line_editor
      简单行编辑器
      使用说明
***-----***
### 请选择: 1.输入文章      2.显示文章内容  ###
###          3.统计文章内容  4.查找字符串   ###
###          5.删除字符串    6.删除行       ###
###          7.清空文章      8.保存文章      ###
###          9.重新输入      0.退出          ###
***-----***
1
-->输入文章, 请选择:
---->1. 键盘输入
---->2. 文件输入
2
-->请输入文件名, 如 <readme.txt> : readme.txt
成功打开文件, 将从文件中读取文章.....
文章输入完成, 请继续操作:
***-----***
### 请选择: 1.输入文章      2.显示文章内容  ###
```

2. 显示文章内容测试:

```
kkvb@ubuntu: ~/Documents/line_editor
-->请输入文件名, 如 <readme.txt> : readme.txt
成功打开文件, 将从文件中读取文章.....
文章输入完成, 请继续操作:
***-----***
### 请选择: 1.输入文章      2.显示文章内容  ###
###          3.统计文章内容  4.查找字符串   ###
###          5.删除字符串    6.删除行       ###
###          7.清空文章      8.保存文章      ###
###          9.重新输入      0.退出          ###
***-----***
2
1. Based on the time series AR model, principal component
2. analysis comprehensive evaluation model, time series
3. ARIMA (p, d, q) model, AHP model, and data visualization
4. analysis methods, through the processing and prediction
5. of relevant quantitative data, comprehensively evaluate
6. the directly impacts of the two candidates election,
7. Trump and Biden, on the U.S. and China's economy, and
8. then analyze the indirectly impacts on it caused by the
9. election results. Last, according to the results of election,
10. optimized and suggestions are made on China's economic
11. countermeasures and policies.
```


3. 统计文章内容测试

```

kkvb@ubuntu: ~/Documents/line_editor
34. which China trade to the United States is used as an indicator to
35. judge influence on China's economy, and the ARIMA (p, d, q) model
36. is used to predict the qualities of net exports of different
37. candidates' election in the next four quarters. The forecast result
38. shows that if Biden is elected, the net export forecasts for the
39. four quarters of the next year are 806.6279, 623.7142, 811.5738, and
40. 896.0007 billion U.S. dollars.

输出完成, 请继续操作:
***-----***
### 请选择: 1.输入文章      2.显示文章内容  ###
###          3.统计文章内容  4.查找字符串    ###
###          5.删除字符串    6.删除行        ###
###          7.清空文章      8.保存文章      ###
###          9.重新输入      0.退出          ###
***-----***

3
-->统计文章内容, 请选择:
---->1. 统计全部字母数
---->2. 统计数字个数
---->3. 统计空格个数
---->4. 统计文章总字数
---->5. 统计以上全部
5

本篇文章中, 全部字母数为: 1758
          数字个数为: 72
          空格个数为: 311
          总字数为: 351

统计完成, 请继续操作:

```

4. 查找字符串测试

```

统计完成, 请继续操作:
***-----***
### 请选择: 1.输入文章      2.显示文章内容  ###
###          3.统计文章内容  4.查找字符串    ###
###          5.删除字符串    6.删除行        ###
###          7.清空文章      8.保存文章      ###
###          9.重新输入      0.退出          ###
***-----***

4
-->请输入要查找的字符串:
time

字符串 time 在整篇文章中出现了3次!

您想显示它们的具体位置吗? (y/n): y

字符串 time 出现了3次, 如下:
第16行: indicators to reflect the U.S. economy. We use a time
第2行: analysis comprehensive evaluation model, time series
第1行: Based on the time series AR model, principal component

查找完成, 请继续操作:

```

5. 删除字符串测试

删除前:

```

查找完成, 请继续操作:
***-----***
### 请选择: 1.输入文章      2.显示文章内容  ###
###          3.统计文章内容  4.查找字符串   ###
###          5.删除字符串    6.删除行        ###
###          7.清空文章      8.保存文章      ###
###          9.重新输入      0.退出          ###
***-----***

2
1. Based on the time series AR model, principal component
2. analysis comprehensive evaluation model, time series
3. ARIMA (p, d, q) model, AHP model, and data visualization
4. analysis methods, through the processing and prediction
5. of relevant quantitative data, comprehensively evaluate
6. the directly impacts of the two candidates election,
7. Trump and Biden, on the U.S. and China's economy, and
8. then analyze the indirectly impacts on it caused by the
9. election results. Last, according to the results of election,
10. optimized and suggestions are made on China's economic
11. countermeasures and policies.
12. For part 1, quantitatively analyze the possible impact of
13. different candidates on the U.S. economy. We use total tax
14. revenue, total net exports, employment rate, the growth
15. rate of GDP, DJIA, and average hourly earnings as six
16. indicators to reflect the U.S. economy. We use a time
17. series AR model to predict them in the next four quarters.
18. Established the principal component analysis comprehensive
19. evaluation system model to precise judge the impact, the
20. comprehensive evaluation scores of Trump in the next four

```

单行删除后:

```

5
-->删除字符串, 请选择:
---->1. 删除文章中所有出现
---->2. 删除指定行中的出现
2

请输入要删除的字符串: time
-->请输入行号: 1

删除完成, 请继续操作
***-----***
### 请选择: 1.输入文章      2.显示文章内容  ###
###          3.统计文章内容  4.查找字符串   ###
###          5.删除字符串    6.删除行        ###
###          7.清空文章      8.保存文章      ###
###          9.重新输入      0.退出          ###
***-----***

2
1. Based on the  series AR model, principal component
2. analysis comprehensive evaluation model, time series
3. ARIMA (p, d, q) model, AHP model, and data visualization
4. analysis methods, through the processing and prediction
5. of relevant quantitative data, comprehensively evaluate
6. the directly impacts of the two candidates election,
7. Trump and Biden, on the U.S. and China's economy, and
8. then analyze the indirectly impacts on it caused by the
9. election results. Last, according to the results of election,
10. optimized and suggestions are made on China's economic
11. countermeasures and policies.
12. For part 1, quantitatively analyze the possible impact of
13. different candidates on the U.S. economy. We use total tax
14. revenue, total net exports, employment rate, the growth
15. rate of GDP, DJIA, and average hourly earnings as six
16. indicators to reflect the U.S. economy. We use a time

```

全部删除后:

```

5
-->删除字符串, 请选择:
---->1. 删除文章中所有出现
---->2. 删除指定行中的出现
1

请输入要删除的字符串: time

删除完成, 请继续操作

***-----***
###  请选择: 1.输入文章      2.显示文章内容  ###
###          3.统计文章内容  4.查找字符串   ###
###          5.删除字符串    6.删除行        ###
###          7.清空文章      8.保存文章      ###
###          9.重新输入      0.退出          ###
***-----***

2
1. Based on the ☐ series AR model, principal component
2. analysis comprehensive evaluation model, ☐ series
3. ARIMA (p, d, q) model, AHP model, and data visualization
4. analysis methods, through the processing and prediction
5. of relevant quantitative data, comprehensively evaluate
6. the directly impacts of the two candidates election,
7. Trump and Biden, on the U.S. and China's economy, and
8. then analyze the indirectly impacts on it caused by the
9. election results. Last, according to the results of election,
10. optimized and suggestions are made on China's economic
11. countermeasures and policies.
12. For part 1, quantitatively analyze the possible impact of
13. different candidates on the U.S. economy. We use total tax
14. revenue, total net exports, employment rate, the growth
15. rate of GDP, DJIA, and average hourly earnings as six
16. indicators to reflect the U.S. economy. We use a ☐

```

6. 删除行测试

删除 1~27 行后:

```

kkvb@ubuntu: ~/Documents/line_editor
输出完成, 请继续操作:

***-----***
###  请选择: 1.输入文章      2.显示文章内容  ###
###          3.统计文章内容  4.查找字符串   ###
###          5.删除字符串    6.删除行        ###
###          7.清空文章      8.保存文章      ###
###          9.重新输入      0.退出          ###
***-----***

6
-->请输入要删除的行区间[m,n], m 必须小于等于 n, 例如 <4,5> <6,6> : 1,27

删除完成, 请继续操作

***-----***
###  请选择: 1.输入文章      2.显示文章内容  ###
###          3.统计文章内容  4.查找字符串   ###
###          5.删除字符串    6.删除行        ###
###          7.清空文章      8.保存文章      ###
###          9.重新输入      0.退出          ###
***-----***

2
1. In addition, based on their different political positions and
2. policies, making supplementary analysis of COVID-19 response
3. measures, immigration policies, economic and employment policies,
4. and tax policies through data visualization.
5. For part 2, quantitatively analyze the possible impact of
6. different candidates on China's economy. The total net exports
7. which China trade to the United States is used as an indicator to
8. judge influence on China's economy, and the ARIMA (p, d, q) model
9. is used to predict the qualities of net exports of different
10. candidates' election in the next four quarters. The forecast result
11. shows that if Biden is elected, the net export forecasts for the
12. four quarters of the next year are 806.6279, 623.7142, 811.5738, and
13. 896.0007 billion U.S. dollars.

输出完成, 请继续操作:

```


7. 保存文章测试

将删除文章保存到 writeme.txt:

```

输出完成, 请继续操作:
***-----***
#### 请选择: 1.输入文章          2.显示文章内容  ####
####          3.统计文章内容      4.查找字符串   ####
####          5.删除字符串        6.删除行        ####
####          7.清空文章          8.保存文章       ####
####          9.重新输入          0.退出           ####
***-----***

8
-->请输入保存文件名 如<readme.txt> : writeme.txt

成功打开文件, 正在将文章保存到文件.....

保存完成, 请继续操作

```

8. 清空文章测试

```

保存完成, 请继续操作
***-----***
#### 请选择: 1.输入文章          2.显示文章内容  ####
####          3.统计文章内容      4.查找字符串   ####
####          5.删除字符串        6.删除行        ####
####          7.清空文章          8.保存文章       ####
####          9.重新输入          0.退出           ####
***-----***

7

清空完成, 请继续操作
***-----***
#### 请选择: 1.输入文章          2.显示文章内容  ####
####          3.统计文章内容      4.查找字符串   ####
####          5.删除字符串        6.删除行        ####
####          7.清空文章          8.保存文章       ####
####          9.重新输入          0.退出           ####
***-----***

2

请先输入文章!

```

9. 重新输入测试

从 writeme.txt 输入:

```

kkvb@ubuntu: ~/Documents/line_editor
9
已清空编辑器, 请重新输入文章:
-->输入文章, 请选择:
---->1. 键盘输入
---->2. 文件输入
2
-->请输入文件名, 如 <readme.txt> : writeme.txt

成功打开文件, 将从文件中读取文章.....

文章输入完成, 请继续操作:

***-----***
### 请选择: 1.输入文章      2.显示文章内容  ###
###          3.统计文章内容  4.查找字符串   ###
###          5.删除字符串    6.删除行        ###
###          7.清空文章      8.保存文章      ###
###          9.重新输入      0.退出          ###
***-----***

2
1. 1. In addition, based on their different political positions and
2. 2. policies, making supplementary analysis of COVID-19 response
3. 3. measures, immigration policies, economic and employment policies,
4. 4. and tax policies through data visualization.
5. 5. For part 2, quantitatively analyze the possible impact of
6. 6. different candidates on China's economy. The total net exports
7. 7. which China trade to the United States is used as an indicator to
8. 8. judge influence on China's economy, and the ARIMA (p, d, q) model
9. 9. is used to predict the qualities of net exports of different
10. 10. candidates' election in the next four quarters. The forecast result
11. 11. shows that if Biden is elected, the net export forecasts for the
12. 12. four quarters of the next year are 806.6279, 623.7142, 811.5738, and
13. 13. 896.0007 billion U.S. dollars.

输出完成, 请继续操作:

```

10. 退出测试:

```

2
1. 1. In addition, based on their different political positions and
2. 2. policies, making supplementary analysis of COVID-19 response
3. 3. measures, immigration policies, economic and employment policies,
4. 4. and tax policies through data visualization.
5. 5. For part 2, quantitatively analyze the possible impact of
6. 6. different candidates on China's economy. The total net exports
7. 7. which China trade to the United States is used as an indicator to
8. 8. judge influence on China's economy, and the ARIMA (p, d, q) model
9. 9. is used to predict the qualities of net exports of different
10. 10. candidates' election in the next four quarters. The forecast result
11. 11. shows that if Biden is elected, the net export forecasts for the
12. 12. four quarters of the next year are 806.6279, 623.7142, 811.5738, and
13. 13. 896.0007 billion U.S. dollars.

输出完成, 请继续操作:

***-----***
### 请选择: 1.输入文章      2.显示文章内容  ###
###          3.统计文章内容  4.查找字符串   ###
###          5.删除字符串    6.删除行        ###
###          7.清空文章      8.保存文章      ###
###          9.重新输入      0.退出          ###
***-----***

0

谢谢使用!
kkvb@ubuntu:~/Documents/line_editor$

```

七 附录（源程序清单）

/*计算机科学与技术3班 段志超的课程设计题目，选题是：简单行编辑器*/

editor.h

```
1. #ifndef EDITOR_H_
2. #define EDITOR_H_
3.
4. #include <iostream>
5. #include <string>
6. #include <list>
7. #include <unordered_map>
8. #include <iomanip>
9. using namespace std;
10.
11. /* 非成员函数-函数声明 */
12. // KMP 算法
13. void getNext(const string& t, string::size_type next[]);
14. string::size_type KMP(const string& s, const string& t);
15.
16. /* 数据结构 */
17. struct lineList {
18. public:
19.     lineList() {}
20.
21.     using listType = list<string>;
22.
23.     /* 成员函数-函数声明 */
24.
25.     // 清空编辑器
26.     void clear() { L.clear(); line = 0; }
27.     // 编辑器是否为空
28.     bool empty() { return L.empty(); }
29.
30.     // 存储文章
31.     istream& read(istream& in = cin);
32.     // 输出文章
33.     ostream& print(ostream& out = cout);
34.
35.     // 统计英文字母数、数字数、空格数及总字数
36.     void word_count();
37.     // 统计某一字符串出现次数
38.     unsigned string_count(const listType::value_type t,
39.         unordered_map<listType::size_type, string>& smatch);
```

```
40.
41. // 删除某一子串的所有出现
42. void remove_substr(listType::value_type remove_str);
43. // 删除某一子串在指定行中的出现
44. void remove_substr(listType::value_type remove_str,
45.     listType::size_type n);
46. // 按行删除指定区间
47. void remove_line(listType::size_type begin_line,
48.     listType::size_type end_line = 0);
49.
50.
51. unsigned get_alpha() { return alpha; }
52. unsigned get_digit() { return digit; }
53. unsigned get_space() { return space; }
54. unsigned get_word() { return word; }
55.
56. private:
57.     listType L;
58.     listType::size_type line = 0;
59.
60.     unsigned alpha = 0;
61.     unsigned digit = 0;
62.     unsigned space = 0;
63.     unsigned word = 0;
64.
65. };
66.
67. /* 用户交互界面处理函数-函数声明 */
68. void select();
69. // 输入文章
70. bool Input();
71. // 显示文章内容
72. bool Display();
73. // 统计文章内容
74. bool Statistic();
75. // 查找字符串, 并统计出现次数
76. bool Lookup();
77. // 删除字符串
78. bool Deletestr();
79. // 删除行
80. bool Deletelin();
81. // 清空文章
82. bool Clearup();
83. // 保存文章
84. bool Save();
```

```
85. // 重新输入
86. bool Reenter();
87. // 清空输入区
88. void clear();
89.
90.
91. #endif // !EDITOR_H_
```

editor.cpp

```
1. #include"editor.h"
2.
3.
4. /* 成员函数-函数定义 */
5. // 静态存储文章
6. istream& lineList::read(istream& in)
7. {
8.     listType::value_type temp;
9.     // 存入当前行
10.    while (getline(in, temp)) {
11.        L.push_back(temp);
12.        ++line;
13.    }
14.
15.    return in;
16. }
17.
18. // 输出文章
19. ostream& lineList::print(ostream& out)
20. {
21.     listType::size_type lcount = 0;
22.     listType::const_iterator it = L.cbegin();
23.     while (it != L.cend()) {
24.         //fprintf(out, "%3d. ", ++lcount);
25.         //if(out==cout)
26.         out << setw(2) << ++lcount << ". "; //输出显示行号以便操作
27.         out << *it++ << endl;
28.     }
29.     return out;
30. }
31.
32. void lineList::word_count() {
33.
```



```
34.     listType::const_iterator it = L.cbegin();
35.     while (it != L.cend()) {
36.         // 统计英文字母、数字和空格
37.         for (signed char i : *it) {
38.             if (isalpha(i))
39.                 alpha++;
40.             else if (isdigit(i))
41.                 digit++;
42.             else if (isspace(i))
43.                 space++;
44.         }
45.
46.         // 统计字数
47.         bool flag = false;
48.         for (signed char i : *it) {
49.             if (i == ' ')
50.                 flag = false; //如果当前字符是空格, 则使 flag 为 false
51.             else if (!flag)
52.             {
53.                 flag = true;
54.                 word++;
55.             }
56.         }
57.
58.         ++it;
59.     }
60. }
61.
62.
63.
64. // 统计字符串出现次数
65. unsigned lineList::string_count(const listType::value_type t,
66.     unordered_map<listType::size_type, string> &smatch) {
67.
68.     unsigned scount = 0; // 字符串出现次数
69.     listType::size_type lcount = 0; // 当前行
70.     // unordered_map<int, string> smatch; // 匹配子串所在行的行号和行内容构成的哈希 map
71.
72.     listType::value_type s;
73.     string::size_type kmp_index;
74.     listType::const_iterator it = L.cbegin();
75.     while (it != L.cend()) {
76.         ++lcount;
77.         //s = *it; // 当前行字符串
78.         //kmp_index = string::npos; // 匹配子串在主串中的位置
```

```
79.      //
80.      //// 当前行中可能有多个子串匹配,
81.      //// 每次匹配成功后更改主串, 继续匹配
82.      //while ((kmp_index = KMP(s, t)) != string::npos) {
83.      //    // 匹配到该子串, 计数加一
84.      //    ++scount;
85.      //    // 并将该行添加到哈希 map 中
86.      //    smatch.insert(make_pair(lcount, *it));
87.
88.      //    // 更新主串
89.      //    if (kmp_index + t.length() < s.length())
90.      //        s = s.substr(kmp_index + t.length(), s.length() - (kmp_index + 1));
91.      //}
92.
93.      kmp_index = 0; // 匹配子串在主串中的位置
94.      while ((kmp_index = (*it).find(t, kmp_index)) != string::npos) {
95.          // 匹配到该子串, 计数加一
96.          ++scount;
97.          // 并将该行添加到哈希 map 中
98.          smatch.insert(make_pair(lcount, *it));
99.
100.         // 更新下次匹配位置
101.         if (kmp_index + t.length() < s.length())
102.             kmp_index += t.length();
103.         else
104.             break;
105.     }
106.     ++it;
107. }
108.
109. return scount;
110. }
111.
112.
113.
114. // 删除某一子串
115. // 删除子串在全文的一切出现
116. void lineList::remove_substr(listType::value_type remove_str)
117. {
118.
119.     listType::value_type::size_type kmp_index;
120.     listType::iterator it = L.begin();
121.     while (it != L.end()) {
122.
123.         if (*it == remove_str) {
```

```
124.         // 如果要删除的字符串与当前行相同, 直接删除整行
125.         L.erase(it);
126.     }
127.     else {
128.         // 否则从当前行匹配子串并删除
129.         //kmp_index = string::npos; // 匹配子串在主串中的位置
130.         //while ((kmp_index = KMP(*it, remove_str)) != string::npos) {
131.         //    (*it).erase(kmp_index, remove_str.length());
132.         //}
133.
134.         kmp_index = 0; // 匹配子串在主串中的位置
135.         while ((kmp_index = (*it).find(remove_str, kmp_index)) != string::npos) {
136.             (*it).erase(kmp_index, remove_str.length());
137.             kmp_index = 0;
138.         }
139.
140.     }
141.     ++it;
142. }
143. }
144. // 重载版本: 删除子串在指定行的出现
145. void lineList::remove_substr(listType::value_type remove_str,
146.     listType::size_type n)
147. {
148.     listType::size_type lcount = 0; // 当前行
149.
150.     listType::value_type::size_type kmp_index;
151.     listType::iterator it = L.begin();
152.     while (it != L.end()) {
153.         // 如果当前行不是指定行, 不做处理
154.         if (++lcount != n)
155.             continue;
156.
157.         if (*it == remove_str) {
158.             // 如果要删除的字符串与当前行相同, 直接删除整行
159.             L.erase(it);
160.         }
161.         else {
162.             // 否则从当前行匹配子串并删除
163.             //kmp_index = string::npos; // 匹配子串在主串中的位置
164.             //while ((kmp_index = KMP(*it, remove_str)) != string::npos) {
165.             //    (*it).erase(kmp_index, remove_str.length());
166.             //}
167.
168.             kmp_index = 0; // 匹配子串在主串中的位置
```

```
169.         while ((kmp_index = (*it).find(remove_str, kmp_index)) != string::npos) {
170.             (*it).erase(kmp_index, remove_str.length());
171.         }
172.     }
173.     // 处理完毕, 跳出循环
174.     if (lcount == n)
175.         break;
176.
177.     ++it;
178. }
179. }
180.
181. // 删除某一行
182. // 按行号删除
183. void lineList::remove_line(listType::size_type begin_line,
184.    listType::size_type end_line) {
185.
186.     // 如果只有一个行号, 即只删除一行
187.     if (end_line == 0)
188.         end_line = begin_line;
189.     listType::size_type lcount;
190.     listType::iterator it;
191.     for (it = L.begin(), lcount = 1;
192.         it != L.end() && lcount <= end_line; ++lcount) {
193.         if (lcount >= begin_line)
194.             it = L.erase(it);
195.         else
196.             ++it;
197.     }
198. }
199.
200.
201. /* 非成员函数-函数定义 */
202. // 利用修正的 KMP 算法查找子串
203. void getNext(const string& t, string::size_type next[]) {
204.     string::size_type j = 0, k = string::npos;
205.     next[0] = string::npos;
206.     while (j < t.length()) {
207.         if (k == string::npos || t.at(j) == t.at(k)) {
208.             ++j;
209.             ++k;
210.             if (t.at(j) != t.at(k))
211.                 next[j] = k;
212.             else
213.                 next[j] = next[k];
```

```
214.     }
215.     else
216.         k = next[k];
217.     }
218. }
219. /*
220.  * KMP 模式匹配子串
221.  * 若匹配成功, 返回第一次匹配主串位置
222.  * 匹配失败, 返回 string::npos
223.  */
224. string::size_type KMP(const string& s, const string& t) {
225. // /*string::size_type*/ int const max_size = max(s.length(), t.length());
226.     string::size_type next[80];
227.     string::size_type i=0, j=0;
228.     getNext(t, next);
229.     while (i < s.length() && j < t.length()) {
230.         if (j == string::npos || s.at(i) == t.at(j)) {
231.             ++i;
232.             ++j;
233.         }
234.         else
235.             j = next[j];
236.     }
237.     if (j >= t.length())
238.         return i - t.length();
239.     else
240.         return string::npos;
241. }
242.
243.
244.
245. //void strcount(lineList*& lList, const string& str) {
246. // // 统计英文字母、数字和空格
247. // for (const char& i : str) {
248. //     if (isalpha(i))
249. //         lList.alpha++;
250. //     else if (isdigit(i))
251. //         lList.digit++;
252. //     else if (isspace(i))
253. //         lList.space++;
254. // }
255. //
256. // // 统计字数
257. // bool flag = false;
258. // for (const char& i : str) {
```

```
259. //      if (i == ' ')
260. //          flag = false; //如果当前字符是空格，则使 flag 为 false
261. //      else if (!flag)
262. //      {
263. //          flag = true;
264. //          llist.word++;
265. //      }
266. //  }
267. //}
```

main.cpp

```

1. #define _CRT_SECURE_NO_WARNINGS
2. #include "editor.h"
3. #include <fstream>
4. #include <cstdio>
5.
6. using namespace std;
7.
8. lineList line_editor;
9.
10. int main() {
11.
12.     printf("\t\t\t\t\t简单行编辑器\t\t\t\t\t\n");
13.     printf("\t\t\t\t\t使用说明\t\t\t\t\t\n");
14.
15.     //select();
16.     int choose = 1;
17.     //cin >> choose;
18.     while (choose) {
19.         select();
20.         cin >> choose;
21.         //clear();
22.         if (choose >= 2 && choose <= 8 && line_editor.empty()) {
23.             cout << "\n 请先输入文章! \n";
24.             continue;
25.         }
26.
27.         switch (choose)
28.         {
29.             case 1:
30.                 if (Input()) {

```

```
31.         cout << "\n 文章输入完成, 请继续操作: \n";
32.         //select();
33.         //cin >> choose; clear();
34.     }
35.     else{
36.         cout << "\n 输入失败, 请重新操作:\n";
37.         line_editor.clear();
38.         //select();
39.         //cin >> choose; clear();
40.     }
41.     break;
42.
43.     case 2:
44.         if (!line_editor.empty() && Display()) {
45.             cout << "\n 输出完成, 请继续操作: \n";
46.             //select();
47.         }
48.         break;
49.
50.     case 3:
51.         if (!line_editor.empty() && Statistic()) {
52.             cout << "\n 统计完成, 请继续操作: \n";
53.             //select();
54.         }
55.         break;
56.
57.     case 4:
58.         if (Lookup()) {
59.             cout << "\n 查找完成, 请继续操作:\n";
60.         }
61.         else
62.             cout << "\n 查找失败, 请重新操作\n";
63.         break;
64.
65.     case 5:
66.         if (Deletestr()) {
67.             cout << "\n 删除完成, 请继续操作\n";
68.         }
69.         else
70.             cout << "\n 删除失败, 请重新操作\n";
71.         break;
72.
73.     case 6:
74.         if (Deletelin()) {
75.             cout << "\n 删除完成, 请继续操作\n";
```

```
76.         }
77.         else
78.             cerr << "\n 删除失败! 请重新操作\n";
79.         break;
80.
81.     case 7:
82.         if (Cleanup()) {
83.             cout << "\n 清空完成, 请继续操作\n";
84.         }
85.         break;
86.
87.     case 8:
88.         if (Save()) {
89.             cout << "\n 保存完成, 请继续操作\n";
90.         }
91.         else
92.             cout << "\n 保存失败, 请重新操作\n";
93.         break;
94.
95.     case 9:
96.         if (Reenter()) {
97.             cout << "\n 已清空编辑器, 请重新输入文章: \n";
98.             if (Input()) {
99.                 cout << "\n 文章输入完成, 请继续操作: \n";
100.            }
101.            else {
102.                cout << "\n 输入失败, 请重新操作:\n";
103.            }
104.        }
105.        break;
106.    case 0:
107.        break;
108.    default:
109.        cerr << "\n 错误指令, 请输入 0~9 的整数! \n";
110.        break;
111.    }
112.
113. }
114.
115. cout << "\n 谢谢使用!" << endl;
116. return 0;
117. }
118.
119. /* 用户交互界面处理界面-函数定义 */
120.
```



```
121. void select() {
122.     printf("\t***-----***\n");
123.     printf("\t###  请选择: 1.输入文章      2.显示文章内容  ###\n");
124.     printf("\t###      3.统计文章内容    4.查找字符串    ###\n");
125.     printf("\t###      5.删除字符串      6.删除行        ###\n");
126.     printf("\t###      7.清空文章        8.保存文章      ###\n");
127.     printf("\t###      9.重新输入        0.退出          ###\n");
128.     printf("\t***-----***\n");
129. }
130.
131.
132. // 输入文章
133. bool Input() {
134.     int choose;
135.     string file_way;
136.     ifstream infile;
137.     cout
138.         << "-->输入文章, 请选择: " << endl
139.         << "---->1. 键盘输入" << endl
140.         << "---->2. 文件输入" << endl;
141.     cin >> choose;
142.     clear();
143.     switch (choose)
144.     {
145.     case 1:
146.         cout << "-->请输入文章 输入 <Ctrl+Z> 以结束输入: " << endl;
147.         line_editor.read();
148.         cin.clear();
149.         cin.eof();
150.         break;
151.     case 2:
152.         cout << "-->请输入文件名, 如 <readme.txt> : ";
153.         cin >> file_way;
154.         infile.open(file_way);
155.         if (!infile) {
156.             cerr << "\n糟糕! 没有成功打开文件。请检查文件名和路径是否正确! \n";
157.             return false;
158.         }
159.         else {
160.             cout << "\n成功打开文件, 将从文件中读取文章....." << endl;
161.             line_editor.read(infile);
162.         }
163.         break;
164.     default:
165.         cerr << "\n错误输入! 请输入数字 1 或 2 !" << endl;
```

```
166.     return false;
167.     break;
168. }
169. //cout << "文章输入完成, 请继续操作: \n";
170. return true;
171. }
172.
173. // 显示文章内容
174. bool Display() {
175.     line_editor.print();
176.     return true;
177. }
178.
179. // 统计文章内容
180. bool Statistic() {
181.     int choose;
182.     line_editor.word_count();
183.
184.     cout
185.         << "--->统计文章内容, 请选择: " << endl
186.         << "---->1. 统计全部字母数" << endl
187.         << "---->2. 统计数字个数" << endl
188.         << "---->3. 统计空格个数" << endl
189.         << "---->4. 统计文章总字数" << endl
190.         << "---->5. 统计以上全部" << endl;
191.     cin >> choose;
192.     switch (choose)
193.     {
194.     case 1:
195.         cout << "\n 本篇文章中, 全部字母数为: " << line_editor.get_alpha() << endl;
196.         break;
197.     case 2:
198.         cout << "\n 本篇文章中, 数字个数为: " << line_editor.get_digit() << endl;
199.         break;
200.     case 3:
201.         cout << "\n 本篇文章中, 空格个数为: " << line_editor.get_space() << endl;
202.         break;
203.     case 4:
204.         cout << "\n 本篇文章中, 总字数为: " << line_editor.get_word() << endl;
205.         break;
206.     case 5:
207.         cout
208.             << "\n 本篇文章中, 全部字母数为: " << line_editor.get_alpha() << endl
209.             << "                数字个数为: " << line_editor.get_digit() << endl
210.             << "                空格个数为: " << line_editor.get_space() << endl
```

```
211.         << "                总字数为: " << line_editor.get_word() << endl;
212.         break;
213.     default:
214.         cerr << "\n 错误输入! 请输入数字 1 ~ 5 !" << endl;
215.         return false;
216.         break;
217.     }
218.     return true;
219. }
220.
221. // 查找字符串, 并统计出现次数
222. bool Lookup() {
223.     string str;
224.     unsigned scount = 0;
225.     char choose;
226.     unordered_map<list<string>::size_type,
227.         string> smatch; // 匹配子串所在行的行号和行内容构成的哈希 map
228.
229.     cout << "-->请输入要查找的字符串: " << endl;
230.     cin >> str;
231.     if (str.length() > 80) {
232.         cerr << "\n 糟糕! 查找的字符串太长啦! ";
233.         return false;
234.     }
235.
236.     scount = line_editor.string_count(str, smatch);
237.     cout << "\n 字符串 " + str + " 在整篇文章中出现了" << scount << "次!" << endl;
238.
239.     cout << "\n 您想显示它们的具体位置吗? (y/n): ";
240.     cin >> choose; clear();
241.
242.     while (choose != 'y' && choose != 'n' && choose != 'Y' && choose != 'N') {
243.         cerr << "-->请输入 y/Y 或 n/N : ";
244.     }
245.
246.     if (choose == 'y' || choose == 'Y') {
247.         cout << "\n 字符串 " + str + " 出现了" << scount << "次, 如下: " << endl;
248.         unordered_map<list<string>::size_type, string>::const_iterator it
249.             = smatch.cbegin();
250.         while (it != smatch.cend()) {
251.             cout << "    第" << it->first << "行: " + it->second << endl;
252.             ++it;
253.         }
254.     }
255.     return true;
```

```
256. }
257.
258. // 删除字符串
259. bool Deletestr() {
260.     int choose;
261.     string delestr;
262.     list<string>::size_type lin;
263.
264.     cout
265.         << "--->删除字符串, 请选择: " << endl
266.         << "---->1. 删除文章中所有出现" << endl
267.         << "---->2. 删除指定行中的出现" << endl;
268.     cin >> choose; clear();
269.     cout << "\n 请输入要删除的字符串: ";
270.     cin >> delestr;
271.
272.     switch (choose)
273.     {
274.     case 1:
275.         line_editor.remove_substr(delestr);
276.         break;
277.     case 2:
278.         cout << "--->请输入行号: ";
279.         cin >> lin;
280.         line_editor.remove_substr(delestr, lin);
281.         break;
282.     default:
283.         cerr << "\n 错误输入, 请输入数字 1 或 2 !" << endl;
284.         return false;
285.         break;
286.     }
287.     return true;
288. }
289.
290. // 删除行
291. bool Deletelin() {
292.     list<string>::size_type beg, end=0;
293.
294.     cout << "--->请输入要删除的行区间[m,n], m 必须小于等于 n, 例如 <4,5> <6,6> : ";
295.     if (scanf("%lu,%lu", &beg, &end) != 2) {
296.         cerr << "\n 错误输入, 请输入区间[m,n], m 小于等于 n !" << endl;
297.         clear();
298.         return false;
299.     }
300.
```

```
301.     line_editor.remove_line(beg, end);
302.     return true;
303. }
304.
305. // 清空文章
306. bool Clearup() {
307.     line_editor.clear();
308.     return true;
309. }
310.
311. // 保存文章
312. bool Save() {
313.     string file_way;
314.
315.     cout << "-->请输入保存文件名 如<writeme.txt> : ";
316.     cin >> file_way;
317.     ofstream outfile(file_way);
318.     if (!outfile) {
319.         cerr << "\n 糟糕! 没有成功打开文件。请检查文件名和路径是否正确! \n";
320.         return false;
321.     }
322.     else {
323.         cout << "\n 成功打开文件, 正在将文章保存到文件....." << endl;
324.         line_editor.print(outfile);
325.     }
326.     return true;
327. }
328.
329. // 重新输入
330. bool Reenter() {
331.     line_editor.clear();
332.     return true;
333. }
334.
335.
336. // 冲洗缓冲区
337. void clear() {
338.     while (getchar() != '\n')
339.         continue;
340. }
```

八. 课程设计心得

这次数据结构课设, 过程很艰难, 答辩的前一个晚上还在不断的调试, 不断修改, 但结果还算挺不错吧。虽然因为时间关系, 答辩没能完整地讲述我实现的功能, 有点小遗憾, 但是这个过程中还算有所收获。实在让我耿耿于怀的, 就是第二个课题删除字符串的 bug 了 (当前版本已修复啦), 我急于答辩, 居然都没有认真看测试结果! 功能全试一遍, “好像似乎” 没问题, 就草草过去答辩.. 果然是被 “0 warning 0 error” 给冲昏了头脑..

第一个课题的手写链表, 着实让人心烦, 很多繁琐的东西, 一遍又一遍地试, 还有 “遍地” 的指针.. 有时候莫名其妙的 bug 真是让人无从下手! (归根结底还是我太菜吧...)。还好, 最后总算写出了个像样的东西, 也算是重新认识了一番 c 语言 (虽然本来也不熟)。做完这个课题, 除了瞬间轻松很多好像没别的想法, 我只是觉得我一定会更爱 c++ 了吧! (不过我只爱 STL, 庞大复杂的 C++ 可一点都不可爱!)

于是第二个课题我打算投入 c++ 的怀抱了! 串? 不是有 string 嘛! 链表? 不是有 list 嘛! 封装好的数据结构用起来简直不要太爽! (虽然这是数据结构课设, 不过.. 第一个课题应该已经有所锻炼了吧.. 我还是想偷偷懒.. 果然人都是懒的..) 模板库的设计者万岁!

然而人总是这样, 有了好工具, 工作没那么累了 (其实还是很累, c++ 的复杂语法可一点不让人轻松), 就想干更多事儿了——要不查找串添加个输出行的功能? 要不加个删除行区间的功能? 要不删除串加个指定行的功能? (然后这里的 bug 我就没发现..) .. 所以我就是没事找事吧.. 最后, 也是历经磨难, 总算写出了个像样的东西。然后我就感觉 c++ 似乎没我想的那么好用, 但用起来真的还是很爽呀!

总而言之, 言而总之, 这次课设不止让我认识了自己的菜, 也让我实打实的学到了不少呀! 然后, 下学期的算法再见咯!