



**程序设计课程设计报告**

题 目 校园新闻发布管理系统

学 院

专 业

年级班别

学 号\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

学生姓名

指导教师

成 绩

|  |  |
| --- | --- |
| 程序功能完成情况 |  |
| 测试用例全面情况 |  |
| 报告格式是否与要求相符 |  |
| 报告内容是否准确全面 |  |

2022年1月

## 一、课程设计题目及要求

校园新闻发布管理系统

**一、功能需求说明(必须采用结构体和动态链表、数据用文件存储实现)**

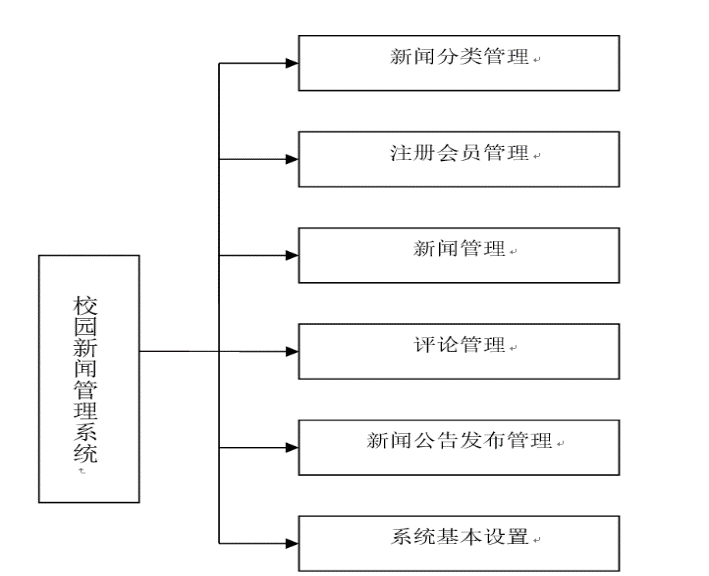
**1、三种类型的用户：新闻系统管理员、普通用户、访客，只有被授权的用户才能访问本新闻管理系统的资源。**

**（1）普通用户：能在本系统中进行新闻浏览，阅读，新闻检索。用户点击每条新闻的标题就能进行新闻阅读；新闻阅读页面，每条新闻的详细信息将被取出，包括内容、标题等；用户能根据自己的需要搜索新闻，如可以通过新闻标题或新闻内容对新闻进行搜索，这样可以快速地找到符合条件的新闻，并输出搜索结果；用户能对新闻进行评论。**

**（2）新闻系统管理员：可以进行新闻分类管理、添加新闻、修改新闻、新闻审核和删除新闻，同时系统管理员能完成用户管理如包括系统用户管理、添加用户和更改帐号。**

**（3）访客：只能浏览新闻。**

**2、功能结构图如下**



**3. 新闻记录存在文件news.txt中,格式如下：**

**每一条记录至少包括一个新闻的标题、内容、发布者、新闻分类、发布日期等内容**

**其中新闻分类：分为通知、公告、简讯三类**

**4．查询搜索功能**

**（1）授权用户能够查询新闻：可以按年、月、日查询**

**（2） 用户可以自定义一个时间段进行查询自己发布新闻的情况**



**（3）用户可以按标题关键字、分类、发布时间进行组合条件查询新闻**

**（4）管理员可查询任一新闻记录，包括已删除记录查询**

**5.排序功能：****按发布日期、发布人中一个进行（升序或降序）排序。**

**6.新闻录入：授权用户能录入某条新闻信息。**

**7.显示功能：可进行全部新闻显示，也可按查询结果进行显示，显示每条新闻的评论信息**

**8.数据变更：管理员可以更改或删除用户的某条新闻记录，但删除的信息要保存在另外一个文件news\_del.txt**

**9.统计功能：（1）用户可统计自己在一个自定义时间段里的新闻发布次数情况**

**（2）管理员可统计各用户在自定义时间段里的新闻发布情况**

**10.用户和管理员的权限要分开。**

**11.其他未描述的功能自行设计。**

**二、加分项**

**1.使用图形界面； 2.使用线、图表示查询记录和统计**

**3.可参考实际情况对软件进行功能扩充 ； 4.可以显示新闻图片信息**

**三、 设计要求（\*注意红色字体部分）**

**1.要一个主界面进行功能实现上述功能使用（文本界面和图形界面均可）。要求编程序实现新闻信息查询、排序、统计、录入、更新、评论等功能。**

**2．不同的模块都要有出错处理，并能给出出错提示。如输入数据错误，文件操作错误**

**3．以上各个功能均编写成子函数，有良好的注释说明，由主函数调用实现。**

**4. 所设计函数须存放到多个文件，不允许所有的函数存放到一个文件里；**

**5．必须采用结构体和动态链表实现对新闻记录数据的存储和访问；**

**6. 必须使用文件保存数据；**

**7.应提供一个界面来调用各个功能,调用界面和各个功能的操作界面应尽可能清晰美观**

**8.测试数据要求**

**(1)不少于8个用户；(2)每个用户不少于15条新闻；**

**(3)每个新闻至少不少于3个评论； (4)删除的记录不少于10条；**

## 二、需求分析

### （1）功能需求

1.信息管理：发布、删除、修改、查看、排序、搜索、评论新闻，添加、删除、修改、查看、搜索用户账号和密码，统计用户发布新闻情况。

2.数据处理：保存新闻数据、用户数据、被删除新闻的数据。

3.权限分离：管理员和用户的权限要分开，不同的权限对应着不同的功能，访客仅能查看新闻。

### （2）交互需求

1.管理员、用户、访客的菜单界面要分开。

2.每一级的界面可进可退，比如在登陆界面登录管理员后即可跳转管理员菜单界面，在管理员菜单界面退出登陆可以返回登陆界面。

3.出错处理，出错时可以出现错误提示并且有出错后的反应。

4.功能执行成功反馈。

5.每一种功能的执行过程中都有着相应的操作提示，可以协助使用者更好地进行操作。

## 概要设计

### （一）系统总体设计



### （二）模块设计

#### 1.主函数模块

文件名：main.c

函数：void main()

作用：启动程序，提供登录选项。

#### 2.菜单函数模块

文件名：menu.c

函数：void login\_interface()

void administrator\_menu()

void user\_menu

void search\_menu

功能：输出各个界面的功能信息。

#### 3.管理员功能函数模块

文件名：admin.c

函数： void admin\_login(struct user\_node\* user\_list, struct news\_node\* news\_list)

void admid\_function(struct user\_node\* user\_list, struct news\_node\* news\_list)

struct user\_node\* create\_user\_list()

struct user\_node\* create\_user\_node(User data)

void insert\_user\_node(struct user\_node\* head, User data)

struct news\_node\* create\_news\_list()

struct news\_node\* create\_news\_node(News data)

void insert\_news\_node(struct news\_node\* head, News data)

void print\_user\_list(struct user\_node\* head)

void print\_news\_list(struct news\_node\* head)

void print\_news\_node(struct news\_node\* cur\_news\_node)

void print\_user\_node(struct user\_node\* cur\_user\_node)

void delete\_user\_node\_by\_account(struct user\_node\* head, char\* account)

void delete\_news\_node\_by\_title(struct news\_node\* head, char\* title,struct news\_node\* deleted\_news\_list)

struct user\_node\* search\_user\_by\_account(struct user\_node\* head, char\* account)

struct news\_node\* search\_news\_by\_title(struct news\_node\* head, char\* title)

void save\_user\_file(char\* user\_file\_name, struct user\_node\* user\_list)

void read\_user\_file(char\* user\_file\_name, struct user\_node\* user\_list)

void save\_news\_file(char\* news\_file\_name, struct news\_node\* news\_list)

void read\_news\_file(char\* news\_file\_name, struct news\_node\* news\_list)

作用：实现管理员所拥有的各项功能。

#### 4.用户功能函数模块

文件：uesr.c

函数：void user\_login(struct user\_node\* user\_list, struct news\_node\* news\_list)

void user\_function(struct user\_node\* user\_list, struct news\_node\* news\_list)

void search\_function(struct news\_node\* news\_list)

int check\_user\_account(char account[N], char password[N], struct user\_node\* user\_list)

struct news\_node\* search\_news\_by\_date(struct news\_node\* head, News temp\_news)

void search\_news\_by\_time(struct news\_node\* head, Date date\_1,Date date\_2)

void search\_news\_by\_category(struct news\_node\* head, char\* category)

void combined\_search(struct news\_node\* head, Date date\_1, Date date\_2,char\* keyword, char\* category)

int Count(struct news\_node\* head, Date date\_1, Date date\_2, char\* account,int\* A,int\* B,int\* C)

struct news\_node\* sorting\_from\_ago\_to\_latest(struct news\_node\* news\_list)

struct news\_node\* sorting\_from\_latest\_to\_ago(struct news\_node\* news\_list)

作用：实现用户所拥有的各项功能。

#### 5.公用数据结构和函数声明

文件：struct..c

结构体：定义发布时间结构体，包含年、月、日

typedef struct \_date {

int year;

int month;

int day;

}Date;

定义用户结构体，包含账号、密码

typedef struct \_user {

char account[N];

char password[N];

}User;

定义用户数据结点

struct user\_node {

User data;

struct user\_node\* next;

};

定义新闻结构体，包含标题、发布者、发布时间、正文、评论、分类

typedef struct \_news {

char title[500];

User;

Date;

char content[10000];

char comment[1000];

char category[N];//1.通知Notice，2.公告Announcement，3.简讯Newsletter

}News;

定义新闻数据结点

struct news\_node {

News data;

struct news\_node\* next;

};

作用：提供公用数据结构和进行函数声明。

## 详细设计

### 菜单函数

1. 函数原型：void login\_interface()

函数作用：打印登录选项信息

函数代码：void login\_interface() {

printf("-------------Login---------------\n");

printf("|\t0.Exit the system \t|\n");

printf("|\t1.administrator login\t|\n");

printf("|\t2.user login \t|\n");

printf("|\t3.visitor \t|\n");

printf("---------------------------------\n\n");

printf("Please enter a number 0~3：");

}

1. 函数原型：void administrator\_menu()

函数作用：打印管理员功能信息

函数代码：void administrator\_menu() {

printf("-------------------Administrator--------------------------\n");

printf("|\t0.Sign out |\n");

printf("|\t1.add news |\n");

printf("|\t2.delete news |\n");

printf("|\t3.modify news |\n");

printf("|\t4.read news |\n");

printf("|\t5.sort news by time from ago to latese |\n");

printf("|\t6.sort news by time from latest to ago |\n");

printf("|\t7.modify user |\n");

printf("|\t8.add user |\n");

printf("|\t9.delete user |\n");

printf("|\t10.read user |\n");

printf("|\t11.search news |\n");

printf("|\t12.Counts news published in a certain time period.|\n");

printf("----------------------------------------------------------\n\n");

printf("Please enter a number 0~12：");

}

1. 函数原型：void user\_menu()

函数作用：打印用户功能选项

函数代码：void user\_menu() {

printf("-------------------User-------------------------\n");

printf("|\t0.Sign out |\n");

printf("|\t1.read news |\n");

printf("|\t2.Count your own news releases |\n");

printf("|\t3.search news |\n");

printf("|\t4.comment news |\n");

printf("|\t5.add news |\n");

printf("|\t6.sort news by time from ago to latese |\n");

printf("|\t7.sort news by time from latest to ago |\n");

printf("------------------------------------------------\n\n");

printf("Please enter a number 0~7：");

}

1. 函数原型：void search\_menu()

函数作用：打印搜索功能信息

函数代码：void search\_menu() {

printf("---------------------Search----------------------------------\n");

printf("\t|0.return |\n");

printf("\t|t1.search news by title. |\n");

printf("\t|t2.search news by date. |\n");

printf("\t|t3.search news by time. |\n");

printf("\t|t4.search news by category. |\n");

printf("\t|t5.search news by keyword of title ,time and category. |\n");

printf("-------------------------------------------------------------\n\n");

printf("Please enter a number 0~5：");

}

### 管理员功能函数

1. 函数原型：void admin\_login(struct user\_node\* user\_list, struct news\_node\* news\_list)

函数作用：验证是否为管理员权限

函数流程图：



函数代码：void admin\_login(struct user\_node\* user\_list, struct news\_node\* news\_list) {

int key = 0;

char admin\_account[10] = { "admin" };

char admin\_password[10] = { "123456" };

char account[10];

char password[10];

while (1) {

printf("Please enter an account number:\n");

rewind(stdin);

scanf("%s", account);

printf("Please enter a password:\n");

rewind(stdin);

scanf("%s", password);

if (strcmp(admin\_account, account) == 0 && strcmp(admin\_password, password) == 0) {

printf("Login successful.\n");

Sleep(1000);

system("cls");

admid\_function(user\_list, news\_list);

break;

}

else {

printf("The account number or password was entered incorrectly!");

Sleep(1000);

system("cls");

printf("Please enter 1 to re-enter it or enter 2 to Return to the previous level:");

rewind(stdin);

scanf("%d", &key);

if (key == 2) { break; }

}

}

}

1. 函数原型：void admid\_function(struct user\_node\* user\_list, struct news\_node\* news\_list)

函数作用：为各管理员功能提供入口

函数流程图：



函数代码：void admid\_function(struct user\_node\* user\_list, struct news\_node\* news\_list) {

int key = 0;

int a = 0;

int b = 0;

int c = 0;

int\* A = &a;

int\* B = &b;

int\* C = &c;

Date date\_1 = { 0 };

Date date\_2 = { 0 };

User temp\_user = { 0 };

News temp\_news = { 0 };

struct user\_node\* cur\_user\_node = { 0 };

struct news\_node\* cur\_news\_node = { 0 };

struct news\_node\* deleted\_news\_list = create\_news\_list();

read\_news\_file("news\_del.txt", deleted\_news\_list);

while (1) {

temp\_news.category[0] = 0;//初始化temp\_news（准确地来说是赋值为0保证数据不会杂糅）

temp\_news.title[0] = 0;

temp\_news.user.account[0] = 0;

temp\_news.user.password[0] = 0;

temp\_news.content[0] = 0;

temp\_news.comment[0] = 0;

temp\_news.date.year = 0;

temp\_news.date.month = 0;

temp\_news.date.day = 0;

temp\_user.account[0] = '0';//初始化temp\_user（准确地来说是赋值为0保证数据不会杂糅）

temp\_user.password[0] = '0';

administrator\_menu();

rewind(stdin);

scanf("%d", &key);

switch (key)

{

case 0:

printf("Sign out\n");

return;

break;

case 1:

printf("Add news\n\n");

printf("Please enter a title:\n");

rewind(stdin);

gets(temp\_news.title);

printf("\nPlease enter the publisher name:\n");

rewind(stdin);

gets(temp\_news.user.account);

printf("Please enter a news category(Notice,Announcement or Newsletter):\n");

rewind(stdin);

gets(temp\_news.category);

printf("Please enter a release date(Separated by commas in the middle->2022,1,7):\n");

rewind(stdin);

scanf("%d,%d,%d", &temp\_news.date.year, &temp\_news.date.month, &temp\_news.date.day);//int类型记得加取地址符

printf("Please enter the body text:\n");

rewind(stdin);

gets(temp\_news.content);

insert\_news\_node(news\_list, temp\_news);

save\_news\_file("news.txt", news\_list);

printf("Added successfully!\n");

break;

case 2:

printf("Delete news\n");

printf("\nPlease enter the title of the news you want to delete:\n");

rewind(stdin);

gets(temp\_news.title);

delete\_news\_node\_by\_title(news\_list, temp\_news.title,deleted\_news\_list);

save\_news\_file("news.txt", news\_list);

break;

case 3:

printf("Modify news\n");

printf("\nPlease enter the title of the news you want to modify:\n");

rewind(stdin);

gets(temp\_news.title);

cur\_news\_node = search\_news\_by\_title(news\_list, temp\_news.title);

if (cur\_news\_node == NULL) {

printf("The specified content could not be found!\n");

}

else {

print\_news\_node(cur\_news\_node);

printf("Please enter a revised title:\n");

rewind(stdin);

gets(cur\_news\_node->data.title);

printf("Please enter the revised publisher name:\n");

rewind(stdin);

gets(cur\_news\_node->data.user.account);

printf("Please enter a revised news category(Notice,Announcement or Newsletter):\n");

rewind(stdin);

gets(cur\_news\_node->data.category);

printf("Please enter a revised release date(Separated by commas in the middle):\n");

rewind(stdin);

scanf("%d,%d,%d", &cur\_news\_node->data.date.year, &cur\_news\_node->data.date.month, &cur\_news\_node->data.date.day);//int类型记得加取地址符

printf("Please enter the revised body text:\n");

rewind(stdin);

gets(cur\_news\_node->data.content);

printf("Please enter the comment:\n");

rewind(stdin);

gets(cur\_news\_node->data.comment);

save\_news\_file("news.txt", news\_list);

printf("Modification successful!\n");

}

break;

case 4:

printf("Read news\n");

print\_news\_list(news\_list);

printf("The end.\n");

break;

case 5:

printf("Sort news by time from ago to latese\n\n");

sorting\_from\_ago\_to\_latest(news\_list);

print\_news\_list(news\_list);

break;

case 6:

printf("Sort news by time from latest to ago\n\n");

sorting\_from\_latest\_to\_ago(news\_list);

print\_news\_list(news\_list);

break;

case 7:

printf("Modify user\n\n");

printf("Please enter the user account number you want to modify:\n");

scanf("%s", &temp\_user.account);

cur\_user\_node = search\_user\_by\_account(user\_list, temp\_user.account);

if (cur\_user\_node == NULL) {

printf("The specified content could not be found!\n");

}

else {

print\_user\_node(cur\_user\_node);

printf("Please enter an account number:\n");

scanf("%s", cur\_user\_node->data.account);

printf("Please enter password:\n");

scanf("%s", cur\_user\_node->data.password);

save\_user\_file("uesr", user\_list);

printf("Modification successful!\n");

}

break;

case 8:

printf("\tAdd user\n");

printf("Please enter an account number:\n\n");

scanf("%s", temp\_user.account);

rewind(stdin);

printf("Please enter password:\n");

scanf("%s", temp\_user.password);

insert\_user\_node(user\_list, temp\_user);

save\_user\_file("uesr.txt", user\_list);

printf("Added successfully!\n");

break;

case 9:

printf("Delete user\n\n");

printf("Please enter the account number of the user you want to delete:\n");

scanf("%s", temp\_user.account);

delete\_user\_node\_by\_account(user\_list, temp\_user.account);

save\_user\_file("uesr.txt", user\_list);

break;

case 10:

printf("Read user\n\n");

print\_user\_list(user\_list);

printf("The end.\n");

break;

case 11:

printf("\t11.search news\n");

search\_function(news\_list);

break;

case 12:

printf("\t12.Counts the number of times news is published in a certain time period.\n");

printf("Please enter the account you want to view:\n");

rewind(stdin);

gets(temp\_news.user.account);

printf("Check out the news between date\_1 and date\_2.\n");

printf("Please enter date\_1:\n");

rewind(stdin);

scanf("%d,%d,%d", &date\_1.year, &date\_1.month, &date\_1.day);

printf("Please enter date\_2:\n");

rewind(stdin);

scanf("%d,%d,%d", &date\_2.year, &date\_2.month, &date\_2.day);

key = Count(news\_list, date\_1, date\_2, temp\_news.user.account, A, B, C);

printf("-Between %d,%d,%d and %d,%d,%d\n", date\_1.year, date\_1.month, date\_1.day, date\_2.year, date\_2.month, date\_2.day);

printf("-User %s posted a total of %d news.\n", temp\_news.user.account, key);

printf("-Notice:%d\n-Announcement:%d\n-Newsletter:%d\n", a, b, c);

break;

default:

printf("Input error, please re-enter：");

break;

}

key = 0;

Sleep(2000);

system("pause");

system("cls");

}

}

1. 函数原型：struct user\_node\* create\_user\_list()

函数作用：创建用户数据链表表头

函数代码：struct user\_node\* create\_user\_list() {

struct user\_node\* head = (struct user\_node\*)malloc(sizeof(struct user\_node));

head->next = NULL;

return head;

}

1. 函数原型：struct user\_node\* create\_user\_node(User data)

函数作用：创建用户数据结点

函数代码：struct user\_node\* create\_user\_node(User data) {// 增加用户数据类型结点

struct user\_node\* new\_node = (struct user\_node\*)malloc(sizeof(struct user\_node));

new\_node->next = NULL;

new\_node->data = data;

return new\_node;

}

1. 函数原型：void insert\_user\_node(struct user\_node\* head, User data)

函数作用：将新的用户数据结点插入到链表当中

函数代码：void insert\_user\_node(struct user\_node\* head, User data) {//将新增的用户数据类型结点链接在表头与第二个结点之间

struct user\_node\* new\_node = create\_user\_node(data);

new\_node->next = head->next;

head->next = new\_node;

}

1. 函数原型：struct news\_node\* create\_news\_list()

函数作用：创建新闻数据链表

函数代码：struct news\_node\* create\_news\_list() {//创造新闻数据类型链表表头

struct news\_node\* head = (struct news\_node\*)malloc(sizeof(struct news\_node));

head->next = NULL;

return head;

}

1. 函数原型：struct news\_node\* create\_news\_node(News data)

函数作用：创建新闻数据结点

函数代码：struct news\_node\* create\_news\_node(News data) {//增加新闻数据类型结点

struct news\_node\* new\_news\_node = (struct news\_node\*)malloc(sizeof(struct news\_node));

new\_news\_node->next = NULL;

new\_news\_node->data = data;

return new\_news\_node;

}

1. 函数原型：void insert\_news\_node(struct news\_node\* head, News data)

函数作用：将新的新闻数据结点插入到链表中

函数代码：void insert\_news\_node(struct news\_node\* head, News data) {//将新增的新闻数据类型结点链接在表头与第二个结点之间

struct news\_node\* new\_news\_node = create\_news\_node(data);

new\_news\_node->next = head->next;

head->next = new\_news\_node;

}

1. 函数原型：void print\_user\_list(struct user\_node\* head)

函数作用：打印用户数据链表

函数流程图：



函数代码：void print\_user\_list(struct user\_node\* head) {//打印全部用户数据

struct user\_node\* pMove = head->next;

printf(" account\t\t\t password\n\n");

while (pMove) {

printf("%12s\t\t%12s\n", pMove->data.account, pMove->data.password);

pMove = pMove->next;

}

printf("\n");

}

1. 函数原型：void print\_news\_list(struct news\_node\* head)

函数作用：打印新闻数据链表

函数流程图：



函数代码：void print\_news\_list(struct news\_node\* head) {

if (head == NULL || head->next == NULL) {

printf("There is no data！");

return 0;

}

struct news\_node\* pMove = head->next;

while (pMove) {

printf("\nTitle:%s\n\nPublisher:%s\n%s\nTime:%d,%d,%d\n", pMove->data.title, pMove->data.user.account, pMove->data.category, pMove->data.date.year, pMove->data.date.month, pMove->data.date.day);

puts(pMove->data.content);

printf("\n");

puts(pMove->data.comment);

pMove = pMove->next;

printf("\n------------------------\n");

}

}

1. 函数原型：void print\_news\_node(struct news\_node\* cur\_news\_node)

函数作用：打印新闻数据结点

函数代码：void print\_news\_node(struct news\_node\* cur\_news\_node) {//打印特定新闻数据

if (cur\_news\_node == NULL) {

printf("There is no data！");

return;

}

printf("%s\n%s\n%s\n%d,%d,%d\n", cur\_news\_node->data.title, cur\_news\_node->data.user.account, cur\_news\_node->data.category, cur\_news\_node->data.date.year, cur\_news\_node->data.date.month, cur\_news\_node->data.date.day);

puts(cur\_news\_node->data.content);

printf("\n");

puts(cur\_news\_node->data.comment);

printf("\n");

}

1. 函数原型：void print\_user\_node(struct user\_node\* cur\_user\_node)

函数作用：打印用户数据结点

函数代码：void print\_user\_node(struct user\_node\* cur\_user\_node) {//打印特定用户数据

printf("account\t\tpassword\n");

printf("%s\t\t%s\n", cur\_user\_node->data.account, cur\_user\_node->data.password);

}

1. 函数原型：void delete\_user\_node\_by\_account(struct user\_node\* head, char\* account)

函数作用：从链表中删除特定用户数据

函数流程图：



函数代码：void delete\_user\_node\_by\_account(struct user\_node\* head, char\* account) {//删除特定用户数据

struct user\_node\* posFrontNode = head;

struct user\_node\* posNode = head->next;

if (posNode == NULL) {

printf("The specified content could not be found!\n");

return;

}

else {

while (strcmp(posNode->data.account, account)) {

posFrontNode = posNode;

posNode = posNode->next;

if (posNode == NULL) {

printf("The specified content could not be found!\n");

return;

}

}

posFrontNode->next = posNode->next;

free(posNode);

}

printf("Delete successful!\n");

}

1. 函数原型：void delete\_news\_node\_by\_title(struct news\_node\* head, char\* title,struct news\_node\* deleted\_news\_list)

函数作用：删除特定新闻数据

函数流程图：



函数代码：void delete\_news\_node\_by\_title(struct news\_node\* head, char\* title,struct news\_node\* deleted\_news\_list) {//删除特定新闻数据

News temp\_news;

struct news\_node\* posFrontNode = head;//把删除用户数据的函数搬过来，居然忘记改数据类型！引以为戒！

struct news\_node\* posNode = head->next;

if (posNode == NULL) {

printf("The specified content could not be found!\n");

return;

}

else {

while (strcmp(posNode->data.title, title)) {

posFrontNode = posNode;

posNode = posNode->next;

if (posNode == NULL) {

printf("The specified content could not be found!\n");

return;

}

}

posFrontNode->next = posNode->next;

temp\_news = posNode->data;

insert\_news\_node(deleted\_news\_list,temp\_news);

save\_news\_file("news\_del.txt", deleted\_news\_list);

free(posNode);

}

printf("Delete successful!\n");

}

1. 函数原型：struct user\_node\* search\_user\_by\_account(struct user\_node\* head, char\* account)

函数作用：通过账号搜索用户，并返回指针

函数流程图：



函数代码：struct user\_node\* search\_user\_by\_account(struct user\_node\* head, char\* account) {//查找特定用户

struct user\_node\* pMove = head->next;

if (pMove == NULL) {

return pMove;

}

else {

while (strcmp(pMove->data.account, account)) {

pMove = pMove->next;

if (pMove == NULL) {

printf("The specified content could not be found!\n");

break;

}

}

return pMove;

}

}

1. 函数原型：struct news\_node\* search\_news\_by\_title(struct news\_node\* head, char\* title)

函数作用：通过标题搜索特定新闻

函数流程图：



函数代码：struct news\_node\* search\_news\_by\_title(struct news\_node\* head, char\* title) {//查找特定新闻

struct news\_node\* pMove = head->next;

if (pMove == NULL) {

return pMove;

}

else {

while (strcmp(pMove->data.title, title)) {

pMove = pMove->next;

if (pMove == NULL) {

printf("The specified content could not be found!\n");

break;

}

}

return pMove;

}

}

1. 函数原型：void save\_user\_file(char\* user\_file\_name, struct user\_node\* user\_list)

函数作用：保存用户数据链表到文件中

函数代码：void save\_user\_file(char\* user\_file\_name, struct user\_node\* user\_list) {

struct user\_node\* pMove = user\_list->next;

User\* p;

FILE\* fp = fopen(user\_file\_name, "wb");

while (pMove) {

p = &pMove->data;

if (fwrite(p, sizeof(User), 1, fp) != 1) {

printf("File write error!\n");

fclose(fp);

}

pMove = pMove->next;

}

rewind(fp);

fclose(fp);

fp = NULL;

}

1. 函数原型：void read\_user\_file(char\* user\_file\_name, struct user\_node\* user\_list)

函数作用：从文件中把用户数据读出来，放进链表中去

函数代码：void read\_user\_file(char\* user\_file\_name, struct user\_node\* user\_list) {

User temp\_user\_data;

FILE\* fp = fopen(user\_file\_name, "rb");

if (fp == NULL) {

printf("Failed to read data.\n");

return;

}

while (fread(&temp\_user\_data, sizeof(User), 1, fp) == 1) {

insert\_user\_node(user\_list, temp\_user\_data);

memset(&temp\_user\_data, 0, sizeof(temp\_user\_data));

}

rewind(fp);

fclose(fp);

fp = NULL;

}

1. 函数原型：void save\_news\_file(char\* news\_file\_name, struct news\_node\* news\_list)

函数作用：保存新闻数据链表到文件中

函数代码：void save\_news\_file(char\* news\_file\_name, struct news\_node\* news\_list) {

struct news\_node\* pMove = news\_list->next;

News\* p;

FILE\* fp = fopen(news\_file\_name, "wb");

while (pMove) {

p = &pMove->data;

if (fwrite(p, sizeof(News), 1, fp) != 1) {

printf("File write error!\n");

fclose(fp);

}

pMove = pMove->next;

}

rewind(fp);

fclose(fp);

fp = NULL;

}

1. 函数原型：void read\_news\_file(char\* news\_file\_name, struct news\_node\* news\_list)

函数作用：从文件中把新闻数据读取到链表中去

函数代码：void read\_news\_file(char\* news\_file\_name, struct news\_node\* news\_list) {

News temp\_news\_data;

FILE\* fp = fopen(news\_file\_name, "rb");

if (fp == NULL) {

printf("Failed to read data.\n");

return;

}

while (fread(&temp\_news\_data, sizeof(News), 1, fp) == 1) {

insert\_news\_node(news\_list, temp\_news\_data);

memset(&temp\_news\_data, 0, sizeof(temp\_news\_data));

}

rewind(fp);

fclose(fp);

fp = NULL;

}

### 用户功能函数

1. 函数原型：void user\_login(struct user\_node\* user\_list, struct news\_node\* news\_list)

函数作用：验证是否为用户权限

函数流程图：



函数代码：void user\_login(struct user\_node\* user\_list, struct news\_node\* news\_list) {

char account[N];

char password[N];

int key = 0;

while (1) {

printf("Please enter an account number:\n");

rewind(stdin);

scanf("%s", account);

printf("Please enter a password:\n");

rewind(stdin);

scanf("%s", password);

key = check\_user\_account(account, password, user\_list);

if (key == 1) {

printf("Login successful.\n");

Sleep(1000);

system("cls");

user\_function(user\_list, news\_list);

break;

}

else {

printf("The account number or password was entered incorrectly!");

Sleep(2000);

system("cls");

printf("Please enter 1 to re-enter it or enter 2 to Return to the previous level:");

rewind(stdin);

scanf("%d", &key);

if (key == 2) { break; }

}

}

}

1. 函数原型：void user\_function(struct user\_node\* user\_list, struct news\_node\* news\_list)

函数作用：提供用户功能选项，是其他功能函数的入口

函数流程图：



函数代码：void user\_function(struct user\_node\* user\_list, struct news\_node\* news\_list) {

int key = 0;

int a = 0;

int b = 0;

int c = 0;

int\* A = &a;

int\* B = &b;

int\* C = &c;

Date date\_1 = { 0 };

Date date\_2 = { 0 };

User temp\_user = { 0 };

News temp\_news = { 0 };

struct user\_node\* cur\_user\_node = { 0 };

struct news\_node\* cur\_news\_node = { 0 };

while (1) {

temp\_news.category[0] = 0;//初始化temp\_news（准确地来说是赋值为0保证数据不会杂糅）

temp\_news.title[0] = 0;

temp\_news.user.account[0] = 0;

temp\_news.user.password[0] = 0;

temp\_news.content[0] = 0;

temp\_news.comment[0] = 0;

temp\_news.date.year = 0;

temp\_news.date.month = 0;

temp\_news.date.day = 0;

temp\_user.account[0] = '\0';//初始化temp\_user（准确地来说是赋值为0保证数据不会杂糅）

temp\_user.password[0] = '\0';

a = 0;

b = 0;

c = 0;

user\_menu();

rewind(stdin);

scanf("%d", &key);

switch (key)

{

case 0:

printf("Sign out\n\n");

return;

break;

case 1:

printf("Read news\n\n");

print\_news\_list(news\_list);

break;

case 2:

printf("Counts the number of times news is published in a certain time period.\n\n");

printf("Please enter the account you want to view:\n");

rewind(stdin);

gets(temp\_news.user.account);

printf("Check out the news between date\_1 and date\_2.\n");

printf("Please enter date\_1:\n");

rewind(stdin);

scanf("%d,%d,%d", &date\_1.year, &date\_1.month, &date\_1.day);

printf("Please enter date\_2:\n");

rewind(stdin);

scanf("%d,%d,%d", &date\_2.year, &date\_2.month, &date\_2.day);

key = Count(news\_list, date\_1, date\_2, temp\_news.user.account,A,B,C);

printf("-Between %d,%d,%d and %d,%d,%d\n", date\_1.year, date\_1.month, date\_1.day, date\_2.year, date\_2.month, date\_2.day);

printf("-User %s posted a total of %d news.\n", temp\_news.user.account, key);

printf("-Notice:%d\n-Announcement:%d\n-Newsletter:%d\n", a, b, c);

break;

case 3:

printf("Search news\n\n");

search\_function(news\_list);

break;

case 4:

printf("Comment news\n\n");

printf("Please enter the title of the news you want to comment on:\n");

rewind(stdin);

gets(temp\_news.title);

cur\_news\_node = search\_news\_by\_title(news\_list, temp\_news.title);

if (cur\_news\_node == NULL) {

printf("The specified content could not be found!\n");

}

else {

printf("Please enter the comment,for example(|\*mary:it is a good news!\*|)\n");

printf("Be careful not to exceed 50 characters!\n");

rewind(stdin);

gets(temp\_news.comment);

strncat(cur\_news\_node->data.comment, temp\_news.comment, 50);

save\_news\_file("news.txt", news\_list);

printf("Comment successful!\n");

}

break;

case 5:

printf("Add news\n\n");

printf("Please enter a title:\n");

rewind(stdin);

gets(temp\_news.title);

printf("Please enter your account:\n");

rewind(stdin);

gets(temp\_news.user.account);

printf("Please enter a news category(Notice,Announcement or Newsletter):\n");

rewind(stdin);

gets(temp\_news.category);

printf("Please enter a release date(Separated by commas in the middle->2022,1,2):\n");

rewind(stdin);

scanf("%d,%d,%d", &temp\_news.date.year, &temp\_news.date.month, &temp\_news.date.day);//int类型记得加取地址符

printf("Please enter the body text:\n");

rewind(stdin);

gets(temp\_news.content);

insert\_news\_node(news\_list, temp\_news);

save\_news\_file("news.txt", news\_list);

printf("Added successfully!\n");

break;

case 6:

printf("Sort news by time from ago to latese\n\n");

sorting\_from\_ago\_to\_latest(news\_list);

print\_news\_list(news\_list);

break;

case 7:

printf("Sort news by time from latest to ago\n\n");

sorting\_from\_latest\_to\_ago(news\_list);

print\_news\_list(news\_list);

break;

default:

printf("Input error, please re-enter：");

break;

}

key = 0;

Sleep(1000);

system("pause");

system("cls");

}

}

1. 函数原型：void search\_function(struct news\_node\* news\_list)

函数作用：提供搜索功能选项，为各搜索函数入口

函数流程图：



函数代码：void search\_function(struct news\_node\* news\_list) {

int key = 0;

Date date\_1={0}, date\_2={0};

User temp\_user = { 0 };

News temp\_news = { 0 };

struct user\_node\* cur\_user\_node = { 0 };

struct news\_node\* cur\_news\_node = { 0 };

while (1) {

system("cls");

temp\_news.category[0] = 0;//初始化temp\_news（准确地来说是赋值为0保证数据不会杂糅）

temp\_news.title[0] = 0;

temp\_news.user.account[0] = 0;

temp\_news.content[0] = 0;

temp\_news.comment[0] = 0;

temp\_news.date.year = 0;

temp\_news.date.month = 0;

temp\_news.date.day = 0;

temp\_user.account[0] = '\0';//初始化temp\_user（准确地来说是赋值为0保证数据不会杂糅）

temp\_user.password[0] = '\0';

search\_menu();

rewind(stdin);

scanf("%d", &key);

switch (key)

{

case 0:

printf("Return\n\n");

return;

break;

case 1:

printf("Search\_news\_by\_title\n\n");

printf("Please enter the title of the news you want to read:\n");

rewind(stdin);

gets(temp\_news.title);

cur\_news\_node = search\_news\_by\_title(news\_list, temp\_news.title);

if (cur\_news\_node == NULL) {

printf("The specified content could not be found!\n");

}

else {

print\_news\_node(cur\_news\_node);

}

break;

case 2:

printf("Search\_news\_by\_date\n\n");

printf("Please enter the release date of the news you want to read(Separated by commas in the middle->2022,1,2):\n");

rewind(stdin);

scanf("%d,%d,%d", &temp\_news.date.year, &temp\_news.date.month, &temp\_news.date.day);

cur\_news\_node = search\_news\_by\_date(news\_list, temp\_news);

if (cur\_news\_node == NULL) {

printf("The specified content could not be found!\n");

}

else {

print\_news\_node(cur\_news\_node);

}

break;

case 3:

printf("Search\_news\_by\_time\n\n");

printf("Check out the news between date\_2 and date\_2.\n");

printf("Please enter date\_1:\n");

rewind(stdin);

scanf("%d,%d,%d", &date\_1.year, &date\_1.month, &date\_1.day);

printf("Please enter date\_2:\n");

rewind(stdin);

scanf("%d,%d,%d", &date\_2.year, &date\_2.month, &date\_2.day);

search\_news\_by\_time(news\_list, date\_1, date\_2);

break;

case 4:

printf("Search\_news\_by\_category\n\n");

printf("Please enter the category of the news you want to read:\n");

rewind(stdin);

gets(temp\_news.category);

search\_news\_by\_category(news\_list, temp\_news.category);

break;

case 5:

printf("Search news by keyword of title ,time and category.\n\n");

printf("Check out the news between date\_1 and date\_2.\n");

printf("Please enter date\_1:\n");

rewind(stdin);

scanf("%d,%d,%d", &date\_1.year, &date\_1.month, &date\_1.day);

printf("Please enter date\_2:\n");

rewind(stdin);

scanf("%d,%d,%d", &date\_2.year, &date\_2.month, &date\_2.day);

printf("Please enter the category of the news you want to read:\n");

rewind(stdin);

gets(temp\_news.category);

printf("Please enter the keyword of title:\n");

rewind(stdin);

gets(temp\_news.title);

combined\_search(news\_list, date\_1, date\_2, temp\_news.title, temp\_news.category);

break;

default:

printf("Input error, please re-enter：");

Sleep(2000);

break;

}

key = 0;

system("pause");

system("cls");

}

}

1. 函数原型：int check\_user\_account(char account[N], char password[N], struct user\_node\* user\_list)

函数作用：检查用户账号密码是否正确

函数代码：int check\_user\_account(char account[N], char password[N], struct user\_node\* user\_list) {//数组作为参数，定义式需要带方括号，使用时则不需要

int key = 0;

struct user\_node\* pMove = user\_list;

if (pMove == NULL || pMove->next == NULL) {

printf("The account number or password was entered incorrectly!");

return key;

}

while (pMove = pMove->next) {

if (strcmp(pMove->data.account, account) == 0 && strcmp(pMove->data.password, password) == 0) {

key = 1;

return key;

}

}

}

1. 函数原型：struct news\_node\* search\_news\_by\_date(struct news\_node\* head, News temp\_news)

函数作用：通过日期搜索新闻

函数流程图：



函数代码：struct news\_node\* search\_news\_by\_date(struct news\_node\* head, News temp\_news) {//通过日期查找特定新闻

struct news\_node\* pMove = head;

if (pMove == NULL|| pMove->next==NULL) {

printf("The specified content could not be found!\n");

return pMove;

}

else {

pMove = pMove->next;

while (pMove) {

if (pMove->data.date.year == temp\_news.date.year && pMove->data.date.month == temp\_news.date.month && pMove->data.date.day == temp\_news.date.day) {

print\_news\_node(pMove);

}

pMove = pMove->next;

}

return pMove;

}

}

1. 函数原型：void search\_news\_by\_time(struct news\_node\* head, Date date\_1,Date date\_2)

函数作用：通过时间段搜索新闻

函数流程图：



函数代码：void search\_news\_by\_time(struct news\_node\* head, Date date\_1,Date date\_2) {//通过时间段查找特定新闻

struct news\_node\* pMove = head;

if (pMove == NULL || pMove->next == NULL) {

printf("The specified content could not be found!\n");

return pMove;

}

else {

pMove = pMove->next;

while (pMove) {

if ((pMove->data.date.year > date\_1.year || (pMove->data.date.year == date\_1.year&&pMove->data.date.month > date\_1.month )|| (pMove->data.date.year == date\_1.year && pMove->data.date.month == date\_1.month&&pMove->data.date.day >= date\_1.day)) &&

(pMove->data.date.year < date\_2.year ||(pMove->data.date.year == date\_2.year&& pMove->data.date.month < date\_2.month )||(pMove->data.date.year == date\_2.year && pMove->data.date.month == date\_2.month&& pMove->data.date.day <= date\_2.day))) {

print\_news\_node(pMove);

}

pMove = pMove->next;

}

return;

}

}

1. 函数原型：void search\_news\_by\_category(struct news\_node\* head, char\* category)

函数作用：通过新闻分类进行搜索

函数流程图：



函数代码：void search\_news\_by\_category(struct news\_node\* head, char\* category) {//通过分类查找特定新闻

struct news\_node\* pMove = head;

if (pMove == NULL||pMove->next==NULL) {

printf("The specified content could not be found!\n");

return pMove;

}

else {

pMove = pMove->next;

while (pMove) {

if (strcmp(pMove->data.category, category)==0) {

print\_news\_node(pMove);

}

pMove = pMove->next;

}

return;

}

}

1. 函数原型：void combined\_search(struct news\_node\* head, Date date\_1, Date date\_2,char\* keyword, char\* category)

函数作用：通过标题关键词、时间段、新闻分类组合搜索新闻

函数流程图：



函数代码：void combined\_search(struct news\_node\* head, Date date\_1, Date date\_2,char\* keyword, char\* category) {

struct news\_node\* pMove = head;

if (pMove == NULL || pMove->next == NULL) {

printf("The specified content could not be found!\n");

return pMove;

}

else {

pMove = pMove->next;

while (pMove) {

if (strcmp(pMove->data.category, category)==0) {

if ((pMove->data.date.year > date\_1.year || (pMove->data.date.year == date\_1.year && pMove->data.date.month > date\_1.month) ||

(pMove->data.date.year == date\_1.year && pMove->data.date.month == date\_1.month && pMove->data.date.day >= date\_1.day)) &&

(pMove->data.date.year < date\_2.year || (pMove->data.date.year == date\_2.year && pMove->data.date.month < date\_2.month) ||

(pMove->data.date.year == date\_2.year && pMove->data.date.month == date\_2.month && pMove->data.date.day <= date\_2.day))) {

if (strstr(pMove->data.title, keyword)) {

print\_news\_node(pMove);

}

}

}

pMove = pMove->next;

}

printf("The end.\n");

return;

}

}

1. 函数原型：int Count(struct news\_node\* head, Date date\_1, Date date\_2, char\* account,int\* A,int\* B,int\* C)

函数作用：统计指定用户发布新闻情况

函数流程图：



函数代码：int Count(struct news\_node\* head, Date date\_1, Date date\_2, char\* account,int\* A,int\* B,int\* C) {

int num = 0;

struct news\_node\* pMove = head;

if (pMove == NULL || pMove->next == NULL) {

printf("The specified content could not be found!\n");

return 0;

}

else {

pMove = pMove->next;

while (pMove) {

if (strcmp(pMove->data.user.account, account)==0) {

if ((pMove->data.date.year > date\_1.year || (pMove->data.date.year == date\_1.year&&pMove->data.date.month > date\_1.month )||

(pMove->data.date.year == date\_1.year && pMove->data.date.month == date\_1.month && pMove->data.date.day >= date\_1.day ))&&

(pMove->data.date.year < date\_2.year || (pMove->data.date.year == date\_2.year&&pMove->data.date.month < date\_2.month )||

(pMove->data.date.year == date\_2.year && pMove->data.date.month == date\_2.month && pMove->data.date.day <= date\_2.day))) {

num++;

if (strcmp(pMove->data.category, "Notice") == 0)(\*A)++;

if (strcmp(pMove->data.category, "Announcement") == 0)(\*B)++;

if (strcmp(pMove->data.category, "Newsletter") == 0)(\*C)++;

}

}

pMove = pMove->next;

}

return num;

}

}

1. 函数原型：struct news\_node\* sorting\_from\_ago\_to\_latest(struct news\_node\* news\_list)

函数作用：对新闻按时间前后进行排序

函数原理：冒泡排序法

函数代码：struct news\_node\* sorting\_from\_ago\_to\_latest(struct news\_node\* news\_list)

{

int i = 1;

struct news\_node\* p1= news\_list;

struct news\_node\* p2 = news\_list->next;

struct news\_node\* temp\_p = { 0 };

if (p1 == NULL|| p2 == NULL)return NULL;

if (p2->next == NULL)return news\_list;

temp\_p = p1;

p1 = p2;

p2 = p2->next;

while (i) {

i = 0;

while (p2) {

if (p1->data.date.year > p2->data.date.year || (p1->data.date.year == p2->data.date.year && p1->data.date.month > p2->data.date.month) ||

(p1->data.date.year == p2->data.date.year && p1->data.date.month == p2->data.date.month && p1->data.date.day > p2->data.date.day)) {

temp\_p->next = p2;

p1->next = p2->next;

p2->next = p1;

//不能把指针也给交换了，只交换数据，否则进入死循环

i = 1;

}

temp\_p = p1;

p1 = p2;

p2 = p2->next;

}

temp\_p = news\_list;

p1 = news\_list->next;

p2 = p1->next;

}

return news\_list;

}

1. 函数原型：struct news\_node\* sorting\_from\_latest\_to\_ago(struct news\_node\* news\_list)

函数作用：对新闻按时间后前进行排序

函数原理：冒泡排序法

函数代码：struct news\_node\* sorting\_from\_latest\_to\_ago(struct news\_node\* news\_list)

{

int i = 1;

struct news\_node\* p1 = news\_list;

struct news\_node\* p2 = news\_list->next;

struct news\_node\* temp\_p = { 0 };

if (p1 == NULL || p2 == NULL)return NULL;

if (p2->next == NULL)return p2;

temp\_p=p1;

p1 = p2;

p2 = p2->next;

while (i) {

i = 0;

while (p2) {

if (p1->data.date.year < p2->data.date.year || (p1->data.date.year == p2->data.date.year && p1->data.date.month < p2->data.date.month) ||

(p1->data.date.year == p2->data.date.year && p1->data.date.month == p2->data.date.month && p1->data.date.day < p2->data.date.day)) {

temp\_p->next = p2;

p1->next = p2->next;

p2->next = p1;

//不能把指针也给交换了，只交换数据，否则进入死循环

i = 1;

}

temp\_p = p1;

p1 = p2;

p2 = p2->next;

}

temp\_p = news\_list;

p1 = news\_list->next;

p2 = p1->next;

}

return news\_list;

}

### 主函数

函数原型：void main()

函数作用：启动整个程序，提供登录选项

函数流程图：



函数代码：void main() {

int key = 0;

struct user\_node\* user\_list = create\_user\_list();

struct news\_node\* news\_list = create\_news\_list();

read\_user\_file("uesr.txt", user\_list);

read\_news\_file("news.txt", news\_list);

while (1) {

login\_interface();

scanf("%d", &key);

switch (key)

{

case 0:

printf("Exit the system\n");

exit(0);

break;

case 1:

printf("Administrator login\n");

admin\_login(user\_list, news\_list);//管理员登录验证

break;

case 2:

printf("User login\n");

user\_login(user\_list, news\_list);//用户登录验证

break;

case 3:

printf("Visitor\n");

print\_news\_list(news\_list);

printf("The end.\n");

break;

default:

printf("Input error, please re-enter：");

break;

}

system("pause");

system("cls");

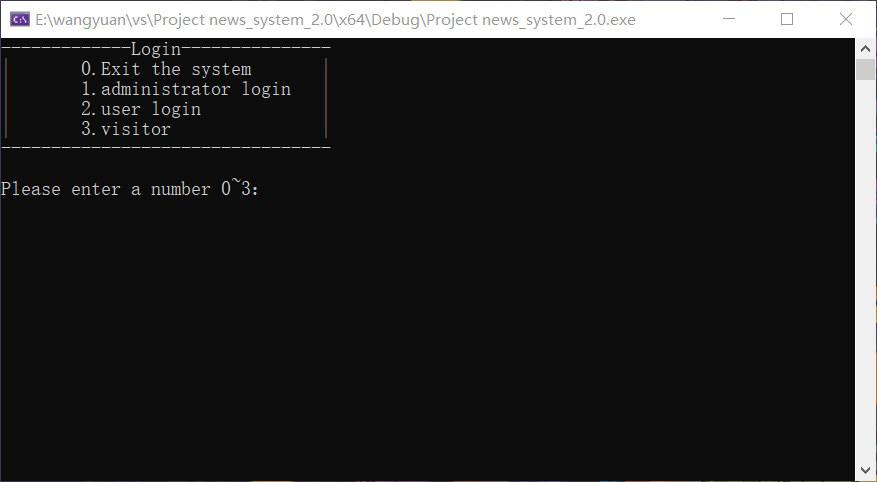
}

}

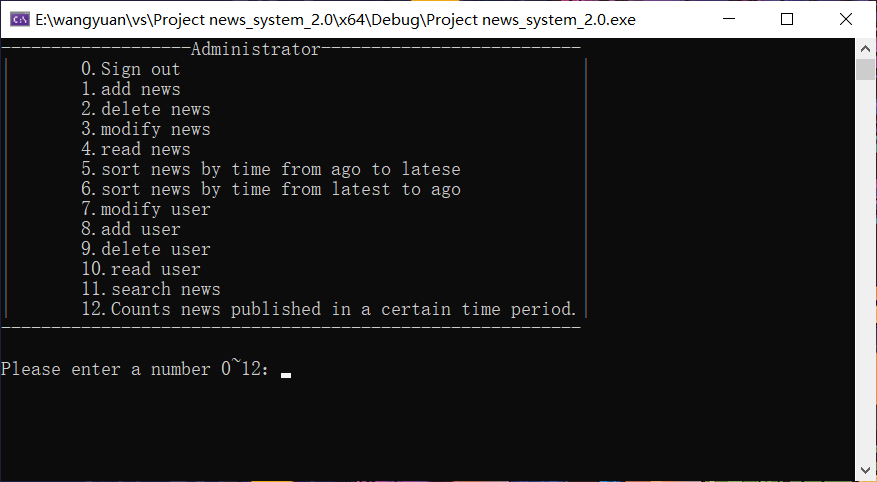
## 实验结果

### 菜单界面

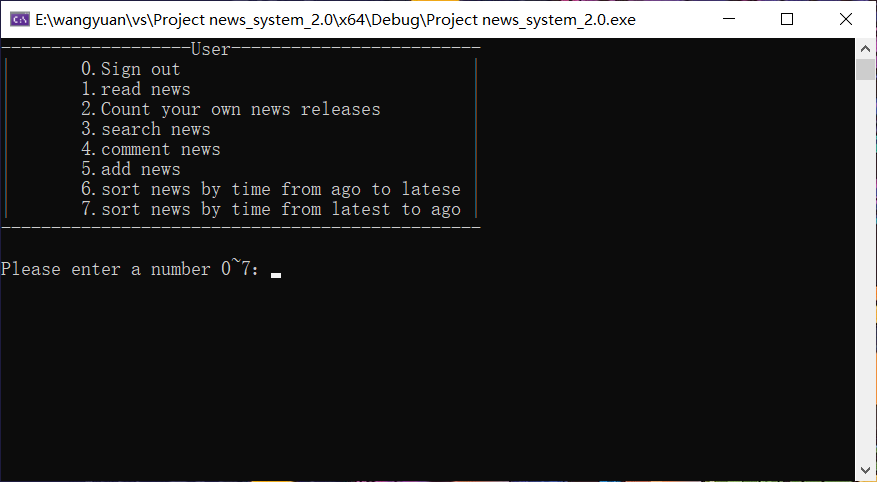
1. 登录菜单



1. 管理员菜单



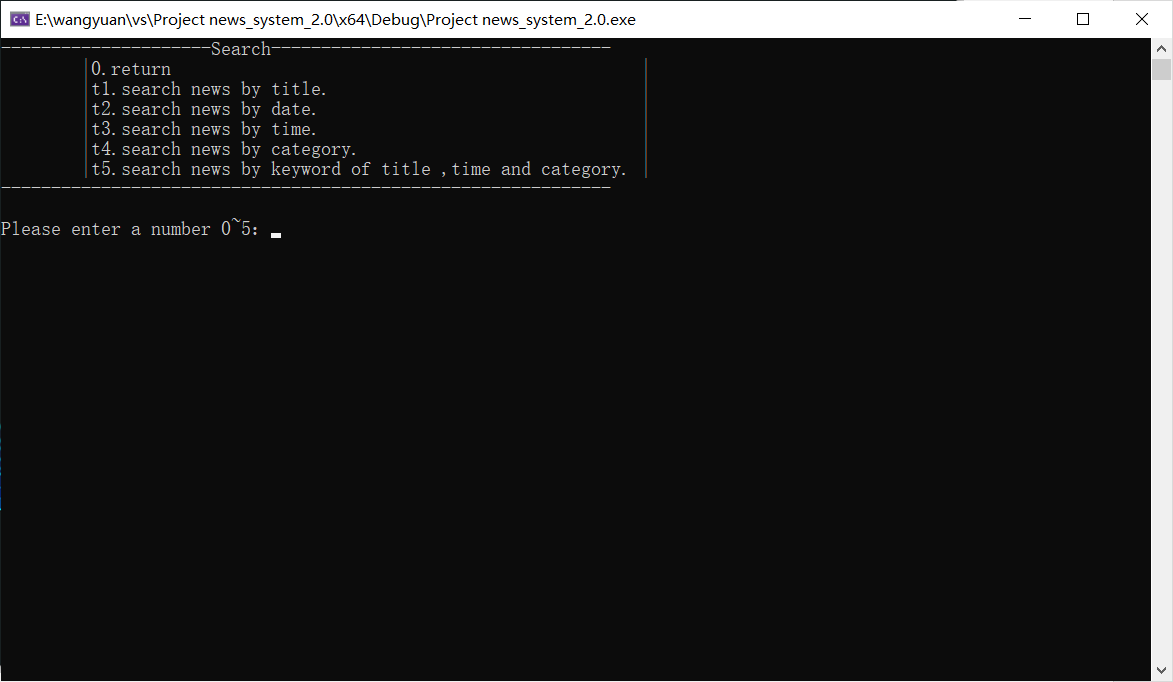
1. 用户菜单



1. 访客界面



1. 搜索功能菜单

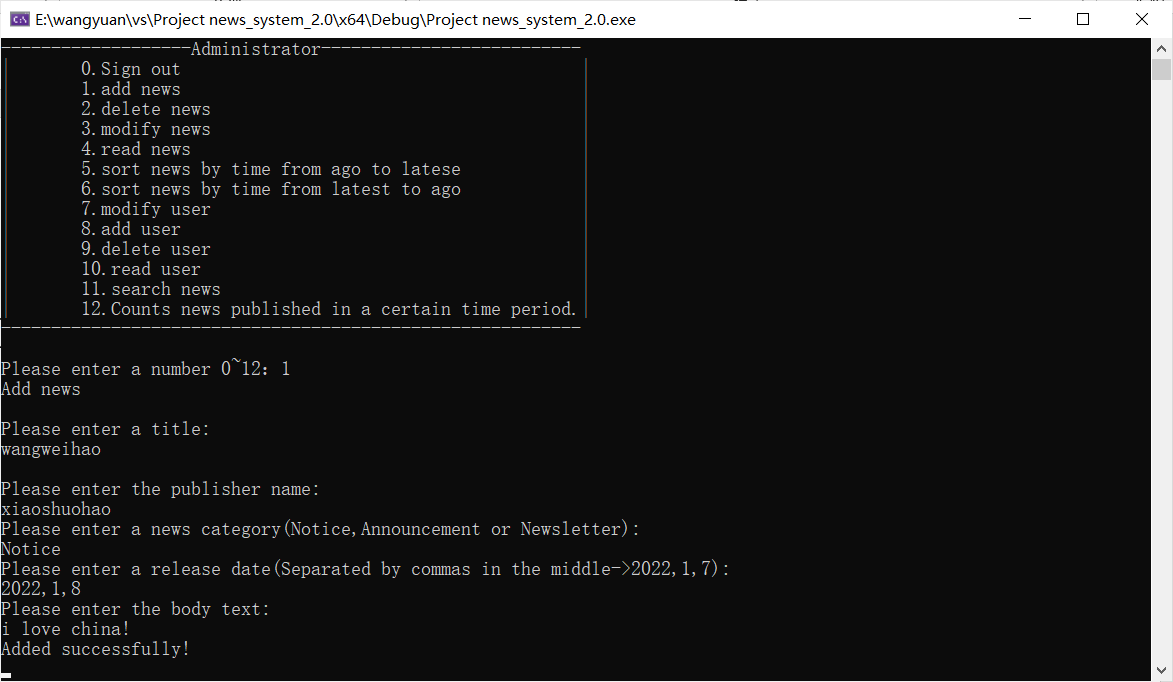


### 功能函数界面

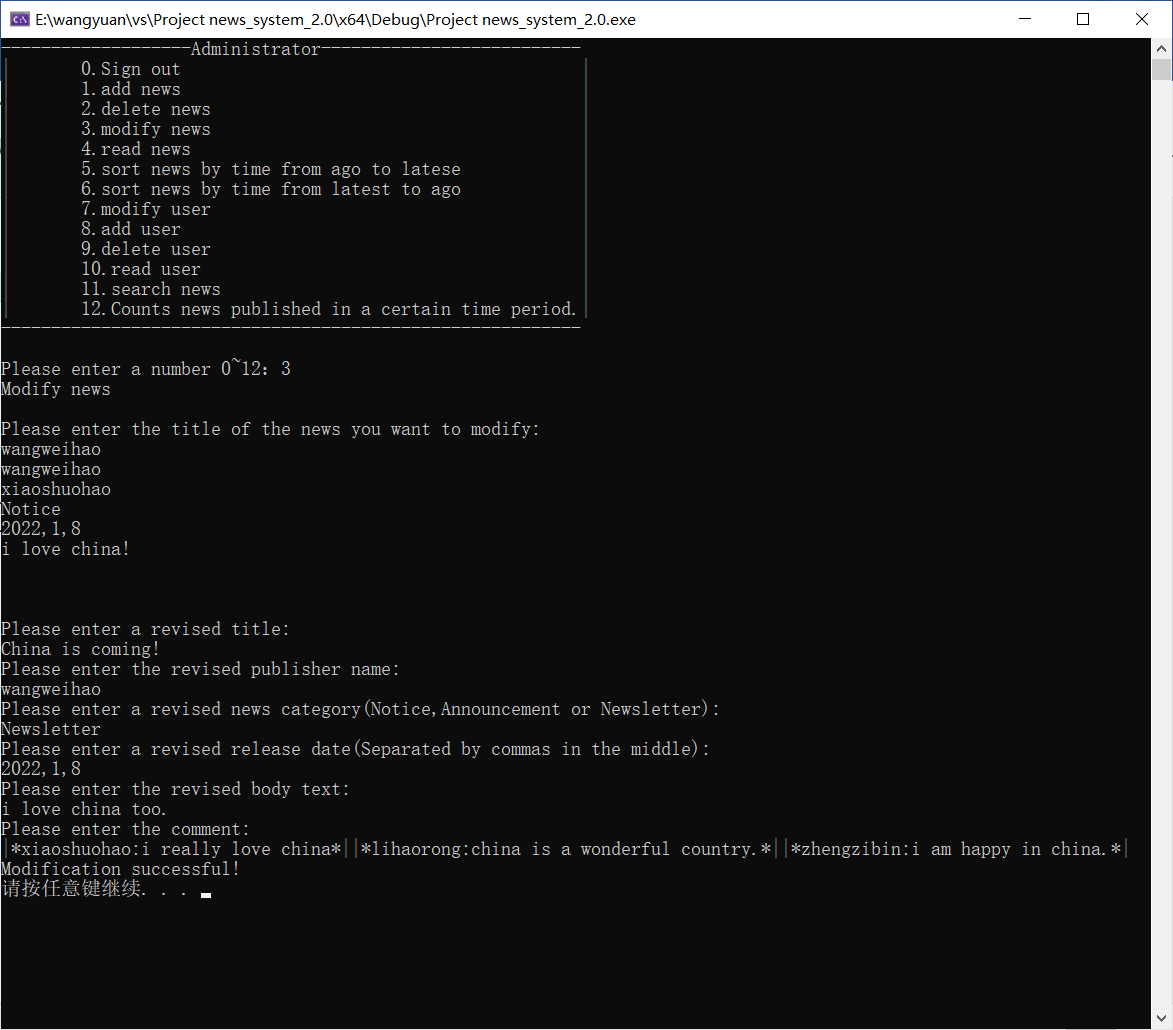
1. 查看新闻



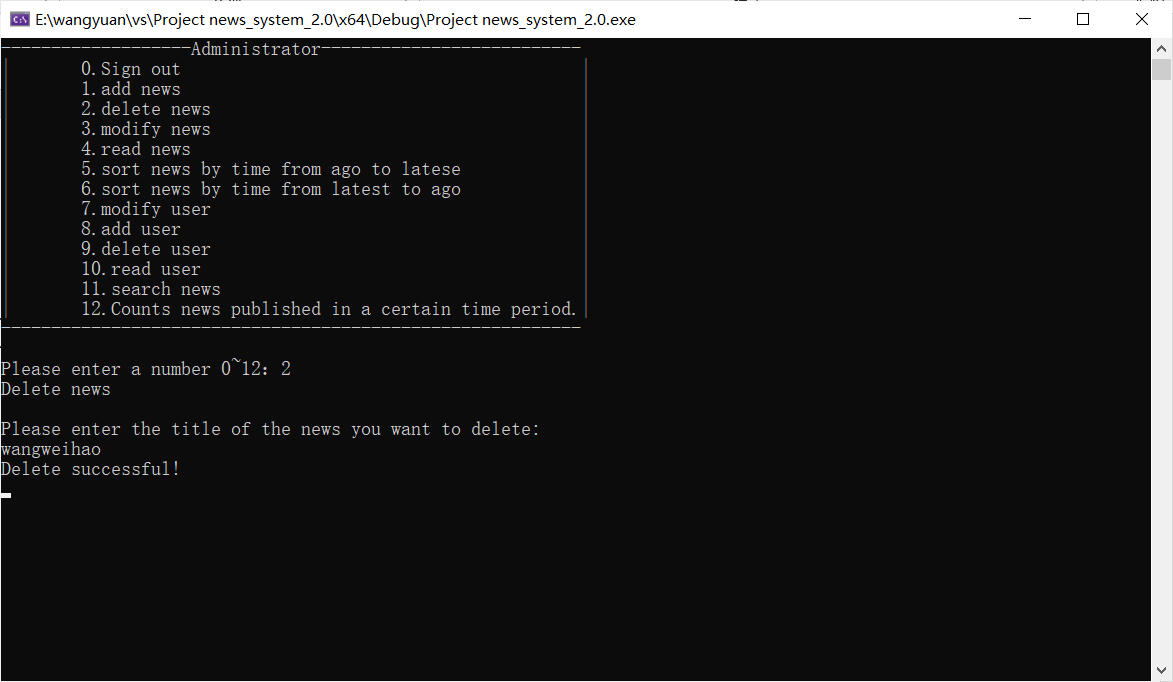
1. 发布新闻



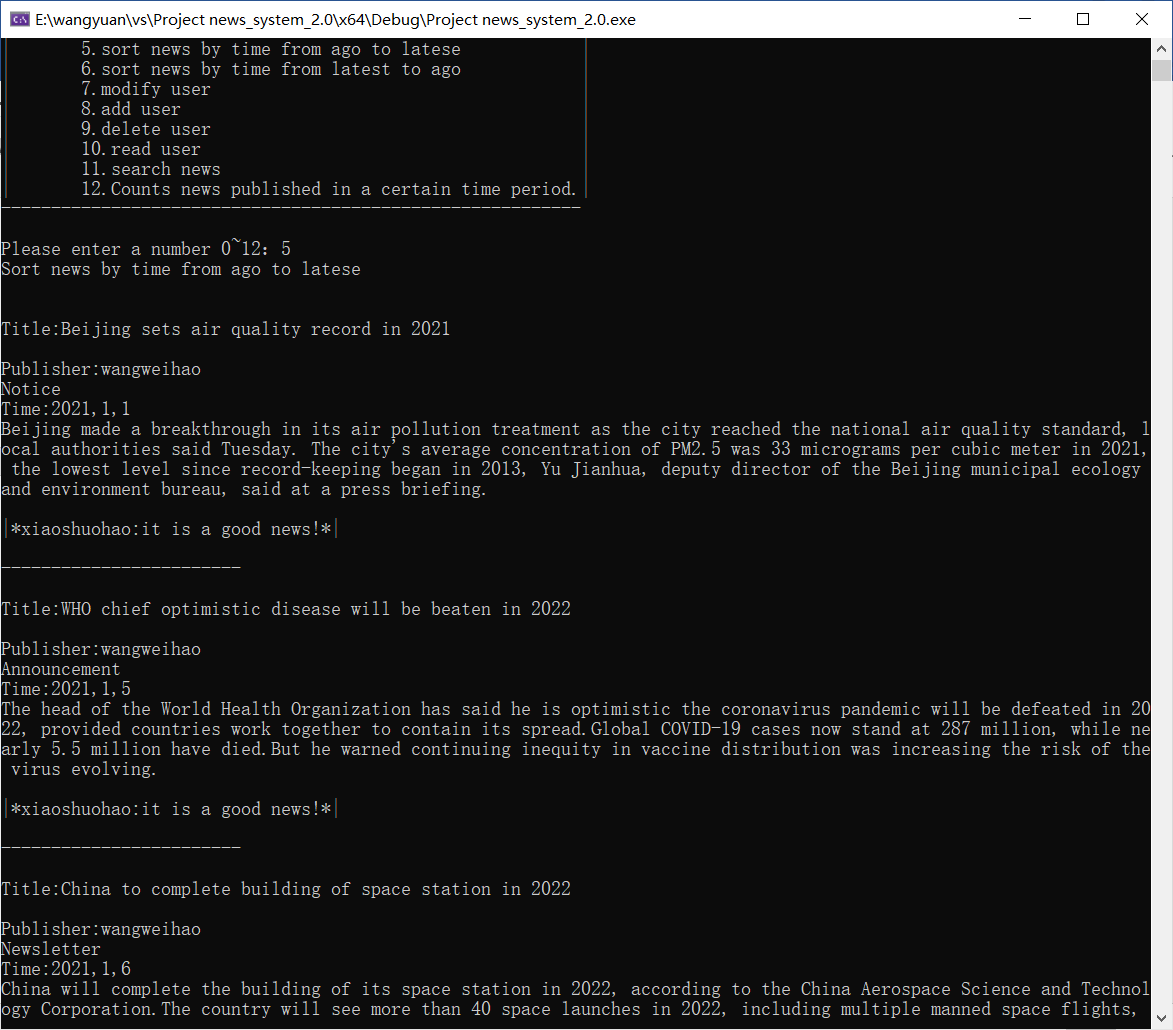
1. 修改新闻



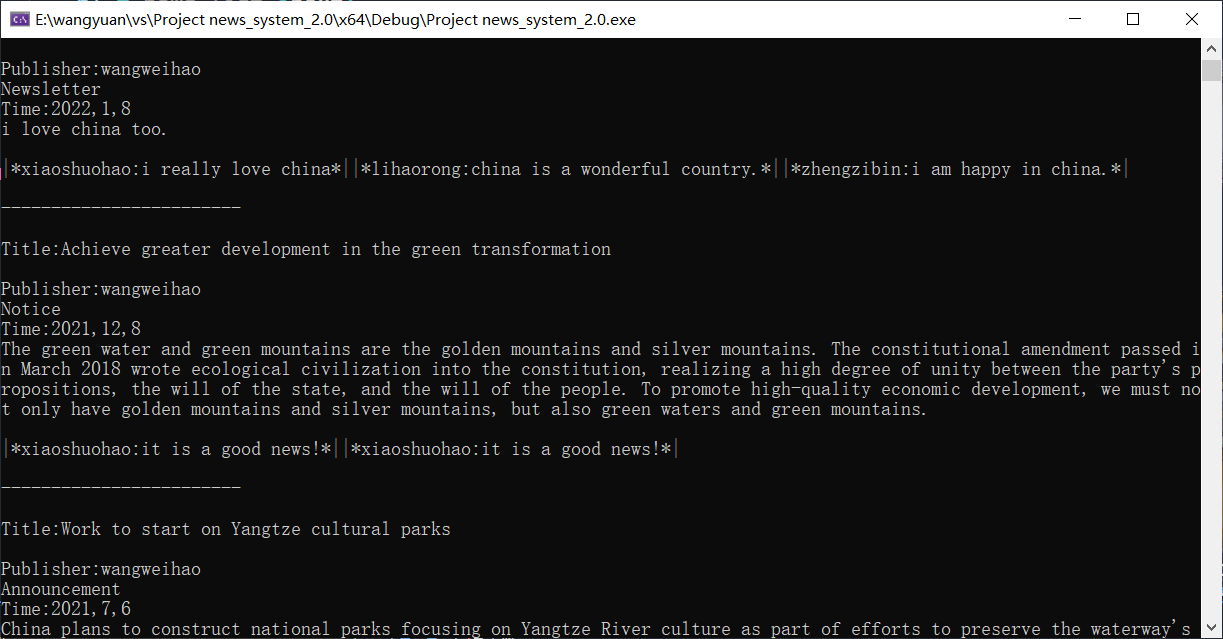
1. 删除新闻



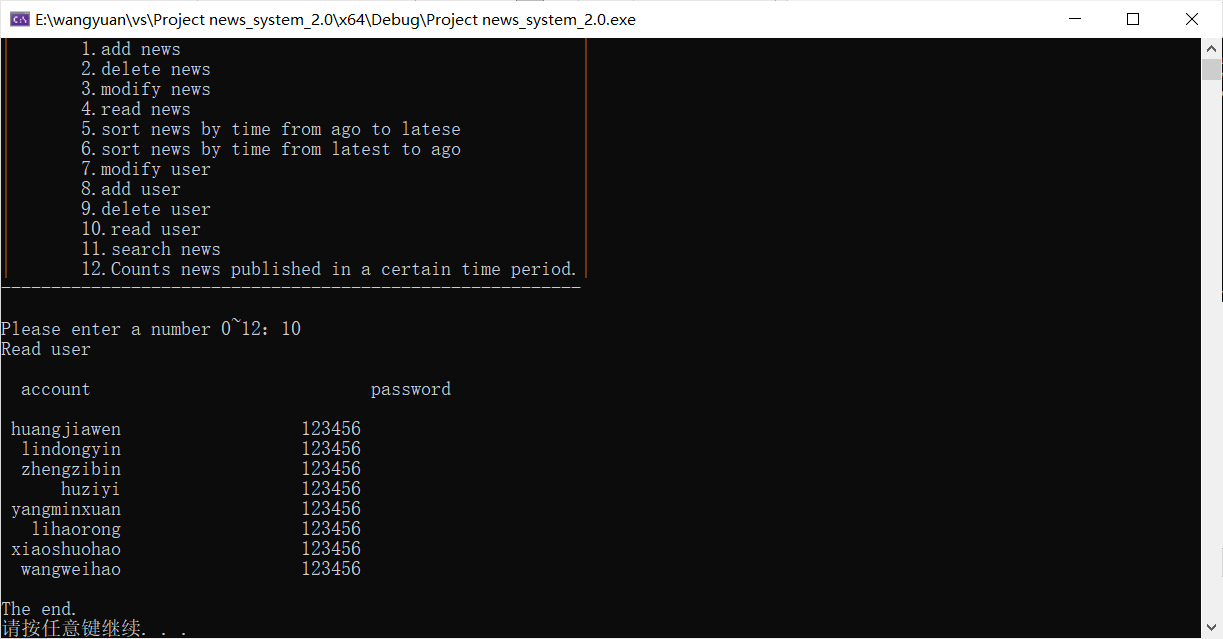
1. 将新闻按时间前后排序



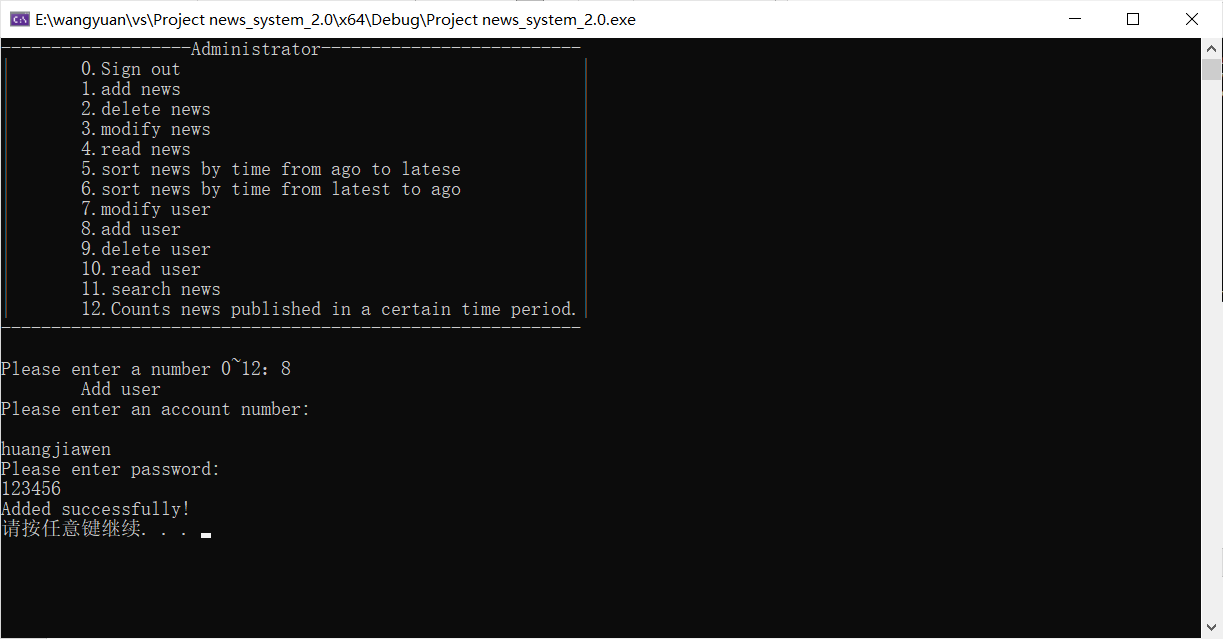
1. 将新闻按时间后前排序



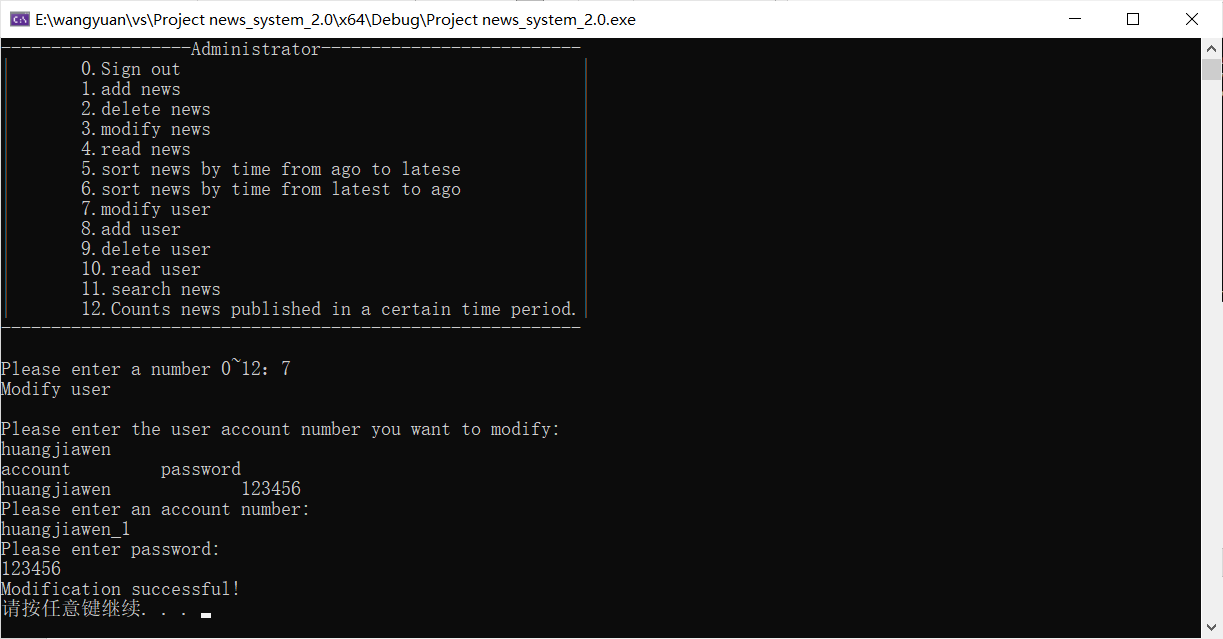
1. 查看用户信息



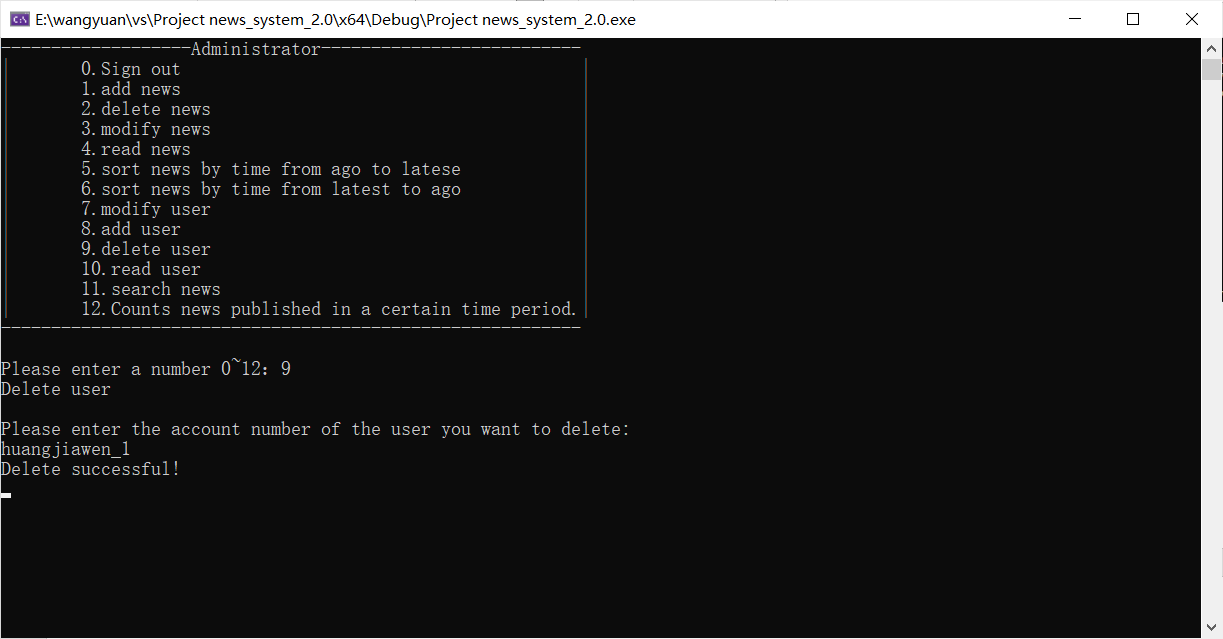
1. 添加用户



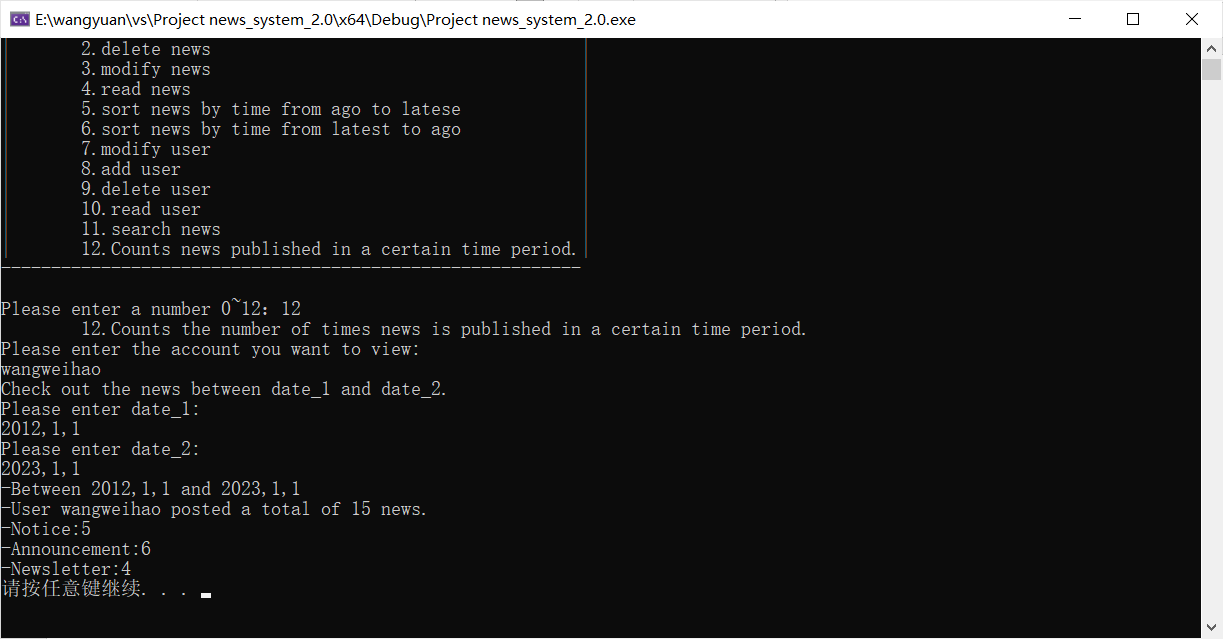
1. 修改用户账号密码



1. 删除用户



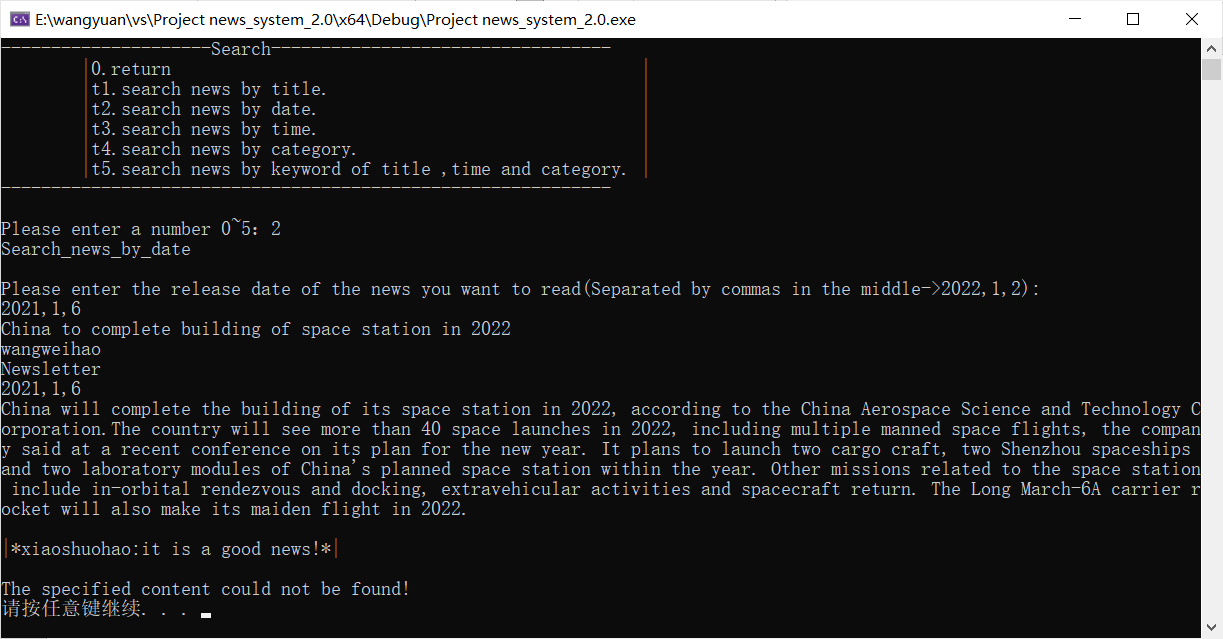
1. 统计指定用户在指定时间内发布新闻的情况



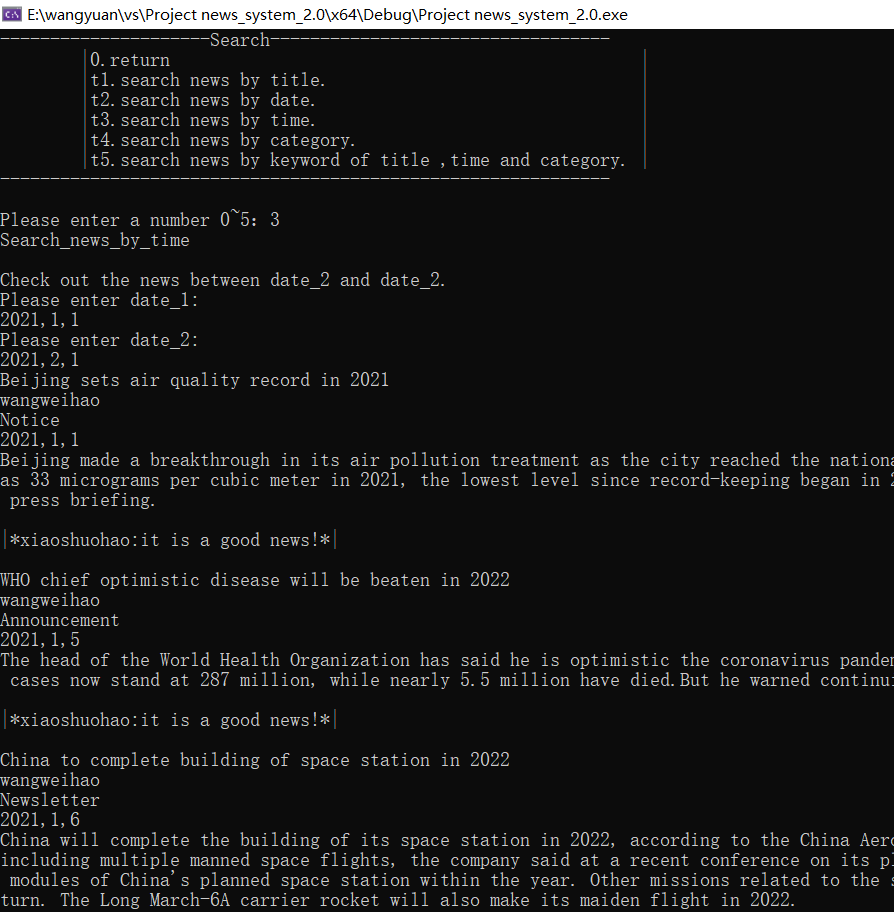
1. 通过新闻标题查找新闻



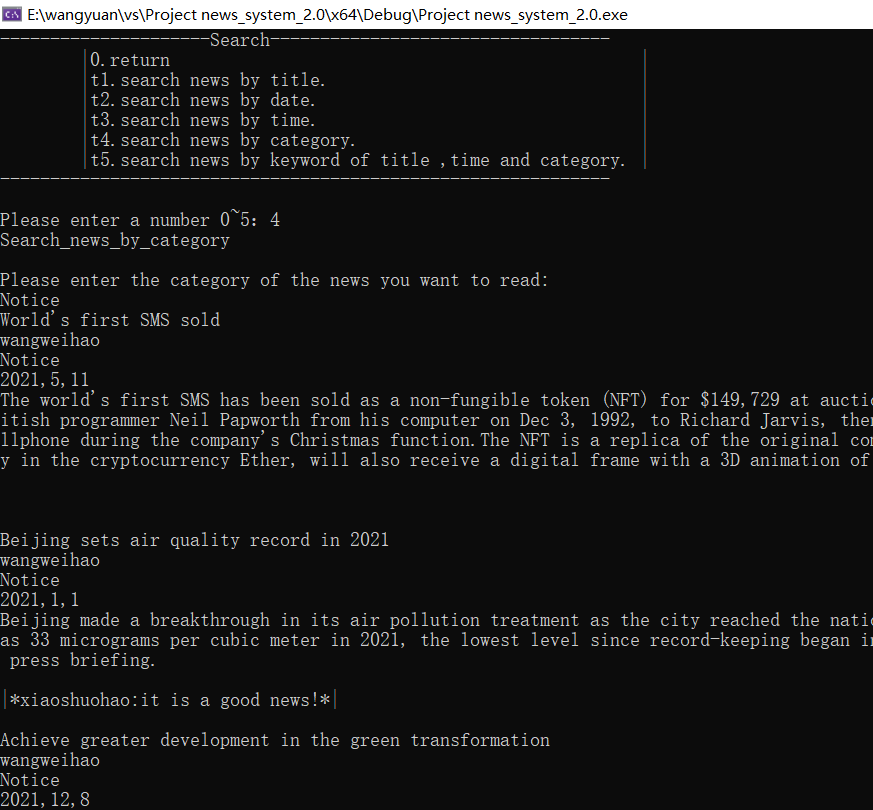
1. 通过发布日期查找新闻



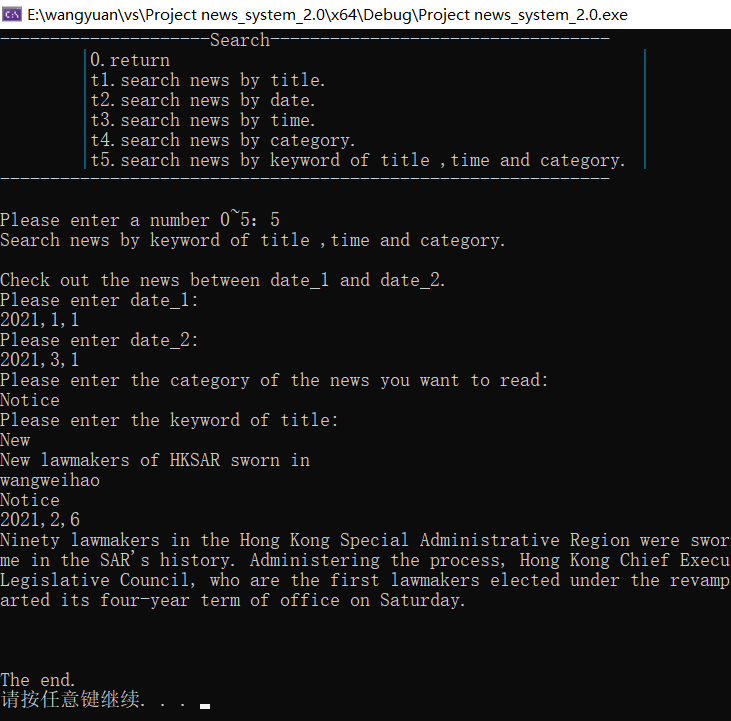
1. 通过时间段查找新闻



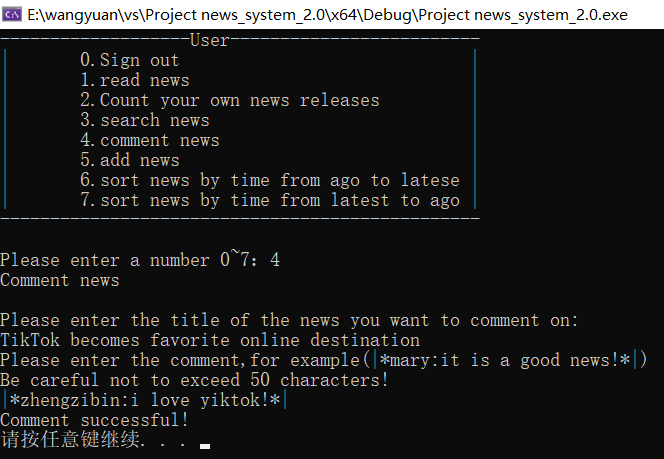
1. 通过分类查找新闻



1. 通过标题关键词、时间段、新闻分类查找新闻

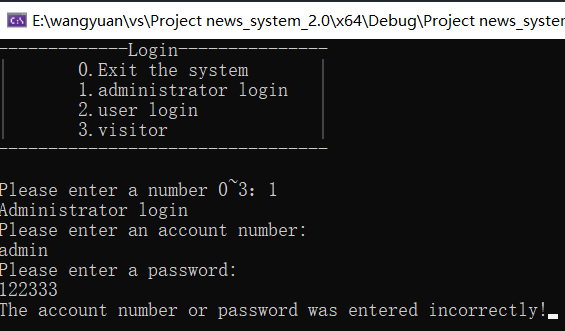


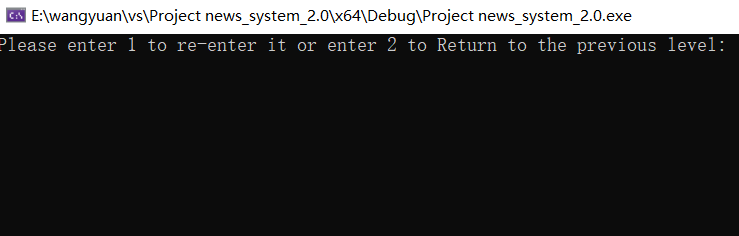
1. 评论新闻



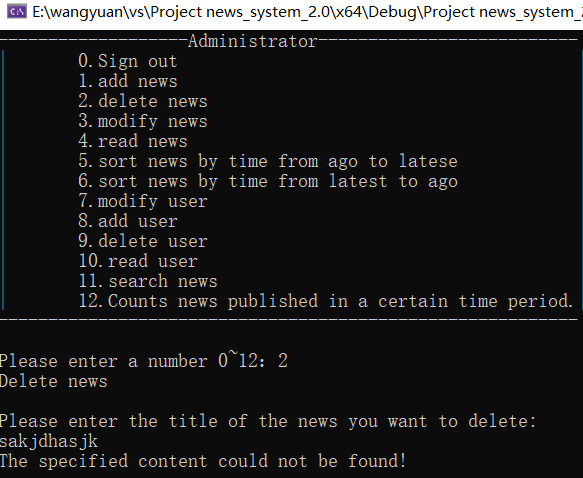
### 出错处理界面

1. 账号密码输入错误



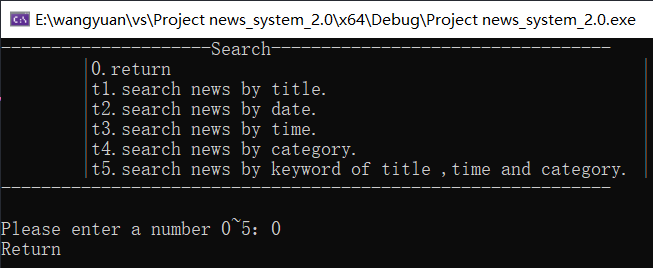


（2）删除新闻时输入错误的标题

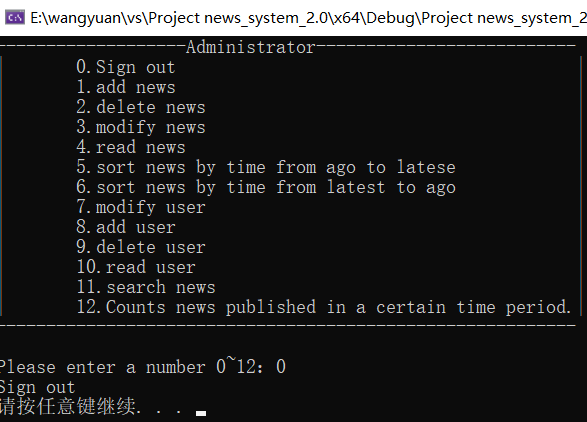
、

### 返回界面

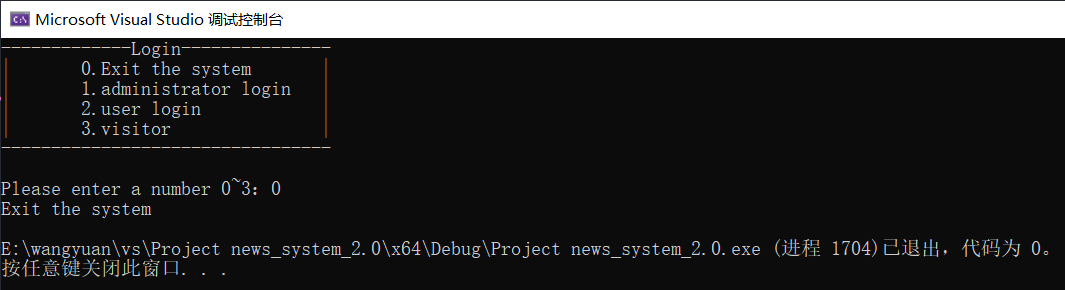
1. 退出搜索界面效果



1. 退出登录效果



1. 退出系统效果



## 总结与心得

从2022年1月1日开始敲下第一行代码，到2022年1月9日完成最后的调试，历时整整九天，《校园新闻发布管理系统》终于诞生了。它具有简洁清晰的菜单，让人一目了然。它能够随时发布、修改、查看、删除新闻，能够随时添加、查看、修改、删除用户账号信息，能够用多种方法搜索各种各样的新闻，也能够统计新闻发布的具体情况。它使用了同步保存的设计，我们不必担心没有正常退出系统而丢失数据；它使用了人性化的操作提示设计，我们用起来更加的舒适和便捷。在完成最低要求的同时，我尽量将它优化得更加人性化，让它更加地友好。

对于我来说，难度一的《校园新闻发布管理系统》无疑是个巨大的挑战。它要求使用结构体链表来储存数据，要求要有文件操作，要求多链表储存新闻数据、用户数据，要求对链表进行增添、删除、排序、搜索等操作，这些对仅仅有C语言基础知识的我们来说无疑是无比巨大的艰难险阻。第一次选择时没有深入学习链表和文件操作，选择了难度一是觉得难度一不是很难，第二次选择时深入了解了链表和文件操作，我依然选择了难度一，是因为我认为在克服困难的同时我能够学到许许多多平时无法学到的东西。设计难度很大，程序十分复杂，而我不懂的东西还有很多。

开始敲下第一行代码后不久，我就遇到了第一个拦路虎。敲着敲着我就迷失了方向，不知道下一步该写哪一部分的代码。对于这个问题，我虚心地请教了师兄。他和我说了一句话，让我至今印象深刻：“如果一艘船不知道该驶去哪个港口，那么任何方向吹来的风都不会是顺风。”他教我，在正式写程序之前，应该先做一个思维导图，把整个系统的框架给勾勒出来，分析他的功能需求，找到它可能需要设计哪一些函数，这样，在写代码的时候，我们就可以根据这个思维导图有目标地去写，而不是像个无头苍蝇一样到处乱撞。即使写着写着卡壳了，我们也可以换一下思路，暂时放下当前的函数，转战另外一个函数，等有思路了再回头来写。

在这九天的时间了，我遇到了不少的大大小小的bug或者编译错误。实话说，编译错误确实很多，不过由于vs会对编译错误进行报错，这类错误到不会使我过于揪心。令人难受的是程序运行中的bug，由于粗心大意造成的如函数声明时漏掉了某个参数，void类型的函数return了一个值，定义指针时没有给其赋值而使用导致程序奔溃，给字符数组赋值时越界导致堆栈损坏，设计函数时逻辑上的错误，数据类型的不统一导致文件操作读写数据成乱码，运行等不到自己的预期结果等等。这一类的错误就只能靠自己不断地debug，不断地查阅资料，不断地和同学讨论，从而找出正确的答案。在这一个过程中，我熟练地掌握了各种debug的方法，比如设置printf查看到底是哪里的语句没有被执行到；巧妙地设置窗口监控变量的值变化，抓住变量脱离正常的那一个瞬间；合理推测哪一部分造成了程序的奔溃，将其注释掉，看看程序下一次还会不会奔溃……我也熟练掌握了各大搜索引擎，可以根据关键词快速地查找出自己想要的答案，我更学会了如何更好地与他人合作，共同解决问题。

没错，在做课设的九天里，我收获最大的就是合作精神。那是一个关于缓冲区的问题。当时我前后混合使用了scanf函数和gets函数，运行时发现我还没有输入，gets语句就被跳过了，而且还真取到了值。一开始我对缓冲区理解不深，上网查了一下，网友介绍了一个发fflush（）函数，我使用了一下发现没有起到作用。思考了很久，我无法解决问题，于是我询问了2班的万博弈同学，他和我详细介绍了缓冲区的知识，引导我用getchar（）函数设计了一个循环来清空缓冲区，随后，我们还认识一个rewind（stdin）函数，更加简洁有效。在此次讨论后，我认识到键盘输入也相当于文件，而缓冲区的清理也有着多种办法。还有就是我的舍友肖烁豪同学，九天里，我每天和他一同前往图书馆编写程序，休息时间和他一起讨论设计中遇到的问题。许多时候，在遇到困难时，每当我们其中一人想要放弃，我们都会互相鼓励着对方，互相打着气，由此，我们也不惧怕任何困难。这让我想起了一句话：一个人可以走得很快，但一群人才可以走得很远。在与同学们的陪伴下，我们一定可以走得越来越远。

这九天的课设时间，我反反复复地翻书，把之前漏掉的许多知识点捡了回来；我也学多了许多书本上学不到的东西，比如说rewind（stdin）函数、strncat（）函数和strstr（）函数，帮我解决了几个难以解决的问题。课设很难，但是也很有趣，只要你积极乐观地面对他，困难总会变成乐趣。课设改了又改，优化了再优化，很累，但是看到自己真正优化到了点上时，自己又很开心，很有成就感。优化是永远也优化不完的，繁杂的代码可以更加简洁，增加的功能可以有更多，系统的运行可以更加有逻辑……我很感激谢老师教给我们的不断优化的思想，精益求精的精神；我很感激谢老师教给我们的不惧困难、耐心应对挑战的品格；我更感激谢老师教给我们的合作的精神，让我们不论合适都有应对挑战的勇气与毅力。知识固然能够开阔人的视野，但是精神的升华更能促进人的成长。在课设之后，还有更大的挑战等着我们，我们当以一往无前的魄力，积极乐观的心态迎接未来的每一个挑战！

## 参考资料

（1）《C程序设计》谭浩强

（2）《大话系列-数据结构》程杰

（3）《C Primer Plus》Stephen Prata

（4）智慧树《程序设计基础》广东工业大学 谢光强

（5）中国大学慕课《C语言程序设计》浙江大学 翁恺

（6）CSDN学习网

（7）互联网搜索引擎

## 八、源代码

在文件夹“代码和头文件”中。