**Managing System on Address Book**

**Summary**

Our C program is Managing System on Address Book. It is suitable for different users to log in. We expect to implement the following functions: setting up a login system, after different users successfully log in, they can perform a series of operations on the contact information, such as adding contacts, viewing all contact information, deleting contacts, editing contacts and sorting. Contact information is stored in a structure. The contact structure includes information such as name, age, gender, phone number, and email address. All the contact information is stored in the linked list, and then written to the file by the program, which is the core of the whole program. On this basis, the login system can make this address book management system suitable for different users, who have their own address book. Some specific features will be described in the following parts. This report has five parts: Problem Statement, Analysis, Design, Implementation and Innovation.

**Group division**

Song yupeng: read the file, add contact, view all contacts, and a few fault tolerances. Main function, Integration of the code, Coordinate division of labor within the group.

Nie peyang: login system (register, modify the password), sort, edit, Integration of the code.

Dai jialin: delete contact, Integration of the code, Innovation(check e-mail format, Password encryption).

**Problem statement:**

On the purpose of storing and managing our daily contacts efficiently and easily. We need set up a system which can help us to manage information. With this system we can add a contact, delete the contact, search a contact and edit it. So, a system came into being.

**Analyze:**

Design and implement an address book management system. Try to save address book information for multiple users in different text files. A series of operations such as adding, deleting, browsing, searching, sorting, and editing and editing related information can be performed on contacts of different users. Management of address books can be more efficient and easy.

First, we need to make the program be able to read and write the file which is the most important thing. Because we operate the information in the program and we need the information we operated before, which was stored in the file. And reading the file successfully, we can carry the operation.

Second, the information we read is stored in the program as a linked list. We're actually manipulating the list by modifying the information.

Third, we need some algorithms to do the function.

In order to manage the address book, the following functions are required to be implemented:

**Design**：

We need a login system that can help users log in, register and change their passwords. After successful registration and login, users can perform the following functions after entering the main menu: Add contact, view all contacts, find contact, delete contact, edit contact, sort. Unless the user opts out of the system, it will return to the main menu after performing the function. And the information will be stored in the corresponding user's file.

The program follows the order: first, a user inputs his account and password, the program compares them in the file which stores all users’ information. If the account and password matched successfully, the program will open a file that stores you contact information and the user can enter the main menu. Else you can re-enter the account and password. Or you can register a new account. If you don’t have an account before, you can register an account first. With an account and password, you can log in successfully.

The login system includes following function.

1.Log in.

If you have already had your account and password. You can input you account and password. The program can match it in the file. And if it matched successfully, the program opens a file and you will enter the main menu.

If the account you input doesn’t exist in the file, the program will help you to register a new account. And if the account and the password you have inputted don’t matched, you can re-enter or exit the system.

2.Register

You can create an account and a password.

3.Modify the password

According to the old password, you can modify the password into a new one. Of course, the new password can’t be same as the old one.

Then you enter into the main menu, the program first opens a file and read it. And you can operate your contact after it. There are six functions you can choose: add contact, view all contacts, find contact, delete contact, edit contact, sort.

Now we will state each function as the sequence.

1.Add contact.

You need to input all information of your contact. Name, age, gender, e-mail, telephone number. When you input the name, you need input a name which is not same as the name you input before. In a word, there can’t be same name in the address book. Then you need to input the age. When you input the gender you only can input man or woman. And the e-mail must have “@”, if what you input not have “@”, you need to re-enter it. after you input all information, the program will write it into the file.

2.View all contact

In the function you can view all information. After that you can choose the contact you want to edit, such as: modify, delete. You can choose according to the serial number. If don’t want to edit them you can be back to the main menu.

3.Delete contact

You can delete the contact according to the name and the sequence in the address book. Before you operate the function, all information will be outputted on the screen. You can read the sequence easily. If you input a name which is not existed in the address book, the program will remind you, and you need input a new one. When you successfully delete it, you can back to the main menu.

4.Search contact

In this function you can search a contact according to his name or telephone number, and if what you find doesn’t exit in the address book, you need re-enter it. And if you successfully find it, you can choose edit it or back.

5.Edit

Edit the information according to name. You can change every information you need.

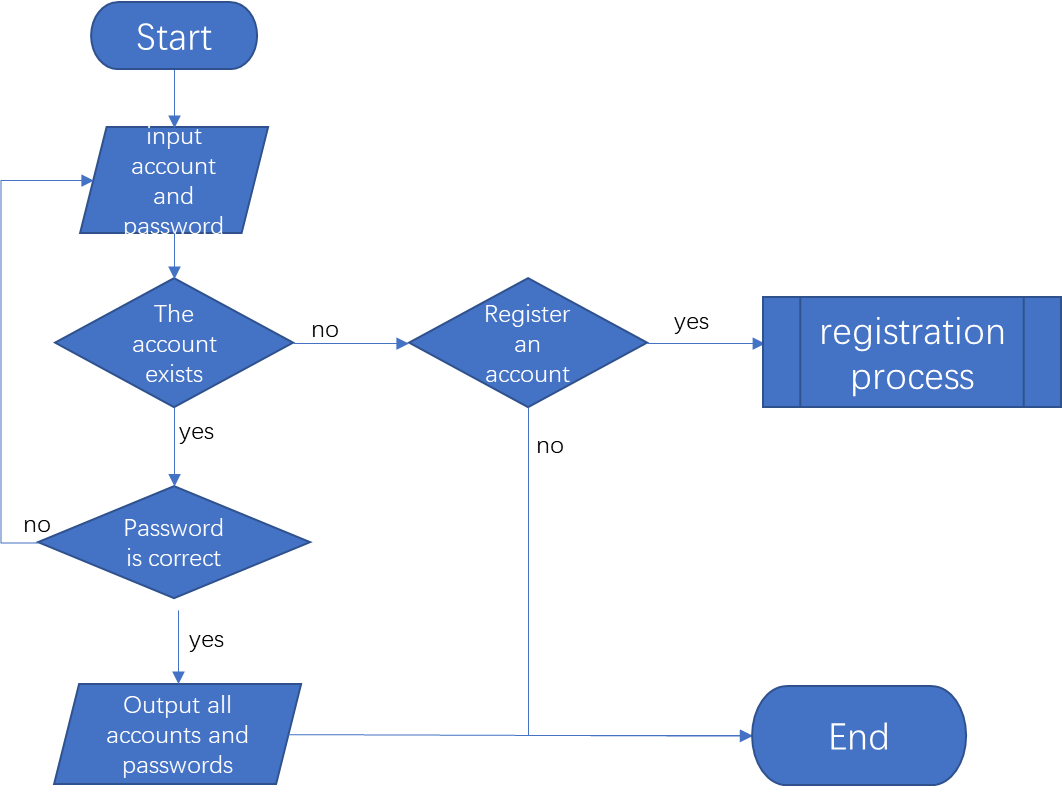
6.Sort

This function provides you to rank the contact, rank down or rank up.

**Implementation**

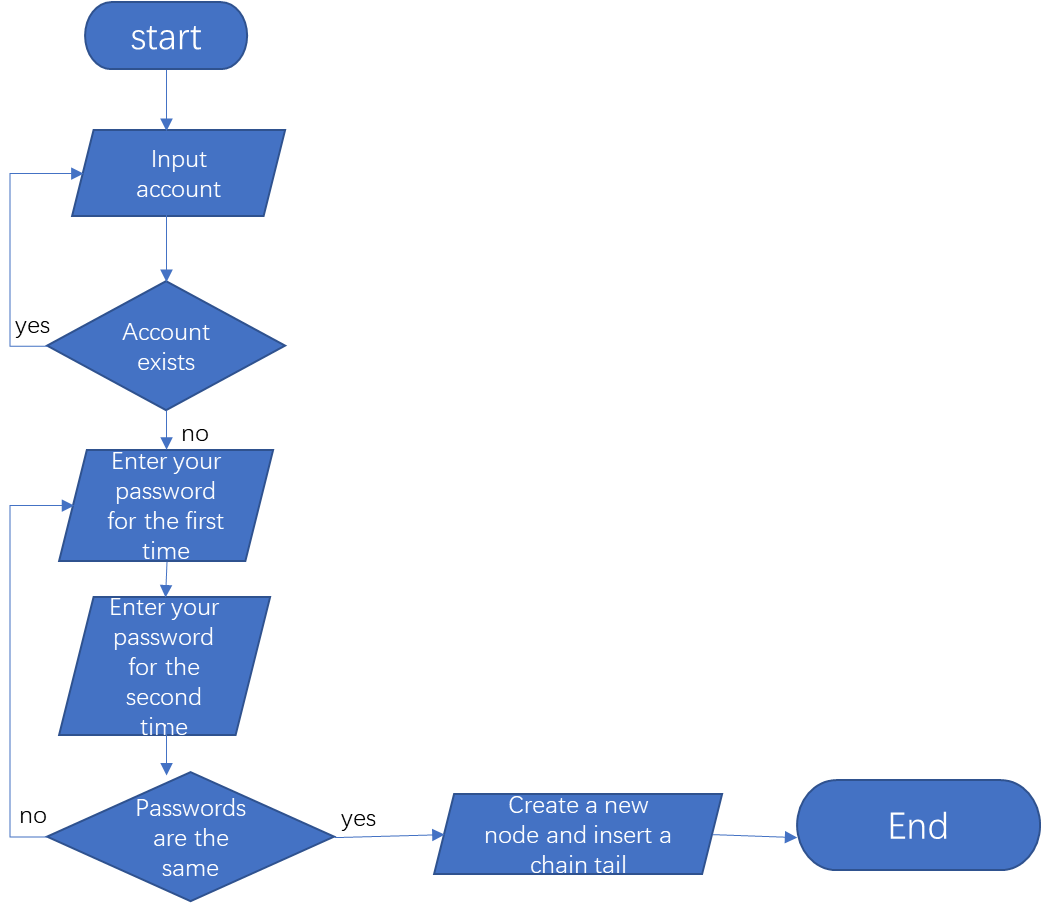
**Login**

In this function users can use his account and password to log in the system. If you successfully enter, you can operate the information in the main menu. And following part is the flow chart.



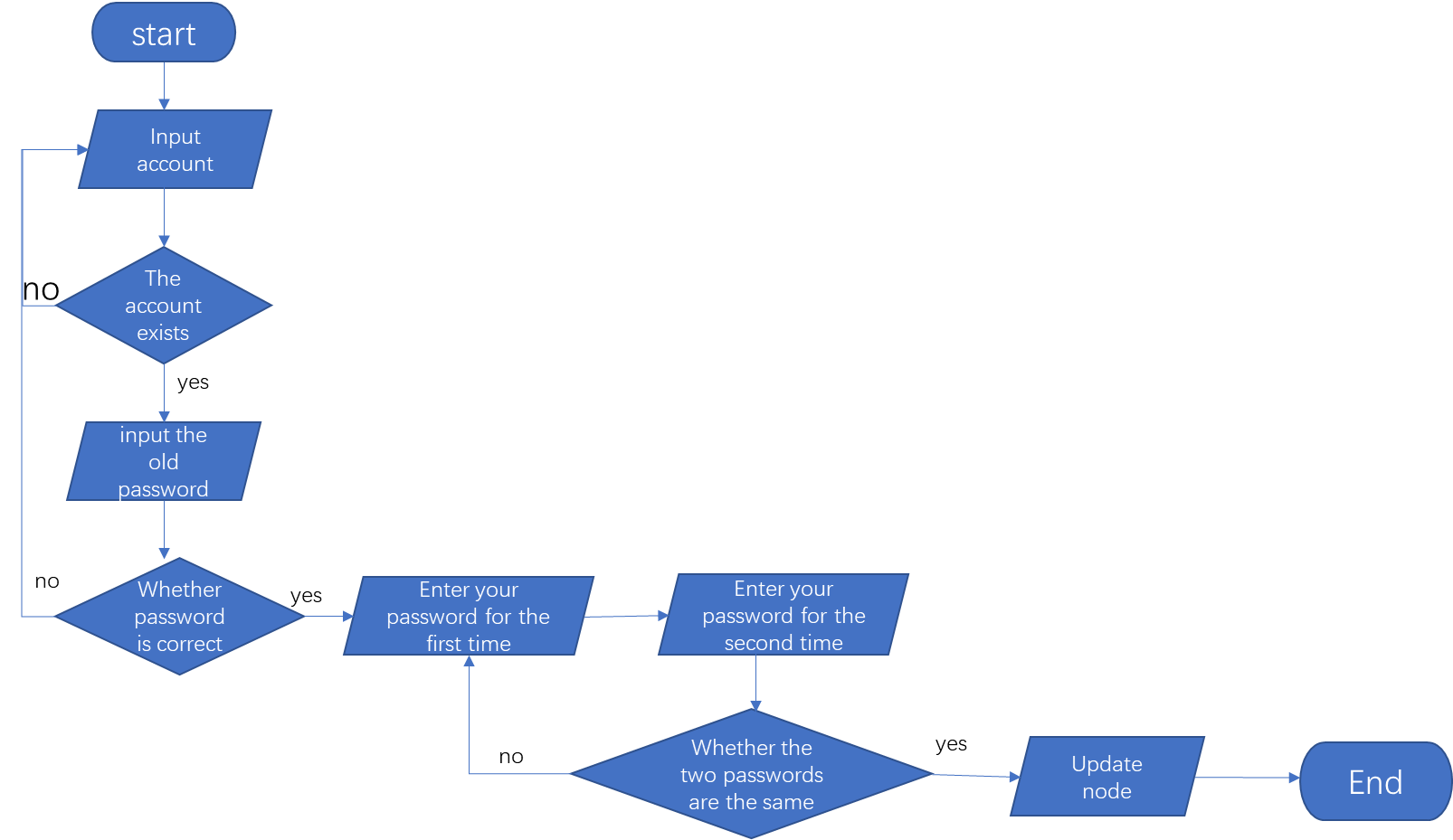
**Register**

In this function users can sign up for their own account, and they can get their own file to store their contact information. The registration information is stored in a file, and when the user logs in, the program matches the information entered by the user with the file, and if successful, the login is successful.



**modify the password**

In this feature, users can use their account and old password to get the right to change their passwords. And then they can login with their new password.



**The code**(containing login, register and modify the password):

char\* login()//Logon success returns 1, logon failure returns 0

{

int select = 0, i = 0;

char name[MAX], fname[MAX], password[MAX], fpassword[MAX];

FILE\* fp1;

// Read the file to create a linked list

land\* p = NULL, \*lhead = NULL;

fp1 = fopen("login.txt", "a+");

while (feof(fp1) == 0)// Make a linked list of the data in the file and execute it when there is something in the file

{

if (lhead == NULL)

{

lhead = (land\*)malloc(sizeof(land));

fscanf(fp1, "%s %s", lhead->account, lhead->password);

lhead->next = NULL;

}

else

{

p = (land\*)malloc(sizeof(land));

//Now->pNext = Head;

fscanf(fp1, "%s %s", p->account, p->password);

p->next = lhead;

lhead = p;

}

}

fclose(fp1);

while (1)

{

system("cls");

system("date/T");

system("time/T");

printf("\t\t\t\*\*\*Welcome to the address book mangement \*\*\*\*\*\n");

printf("\t\t\t\*\*\* 1.log in \*\*\*\*\*\n");

printf("\t\t\t\*\*\* 2.register \*\*\*\*\*\n");

printf("\t\t\t\*\*\* 3.modify the password \*\*\*\*\*\n");

printf("\t\t\tplease choose:");

scanf("%d", &select);

if (select == 1)

{

// login

while (1)

{

system("cls");

printf("Welcome to the address book mangement\nplease input your account：");

scanf("%s", name);

printf("please input your password：");

char pass[LEN + 1];

memset(pass, NULL, LEN + 1);

char ch;

if (count == 0) {

int inputLength = 0;

ch = \_getch();

while (isprint(ch)) {

if (ch != ' ') {

putchar('\*');

pass[inputLength] = ch;

inputLength++;

ch = \_getch();

}

if (!isprint(ch))

break;

if (inputLength == LEN)

break;

}

password[13] = pass[13];

printf("\n");

printf("what you have just input is: %s\n", pass);

}

p = lhead->next;

while (p != NULL)

{

i = 0;

if (strcmp(p->account, name) == 0)

{

i = 1;

break;

}

p = p->next;

}

if (i == 1)

{

if (strcmp(p->account, name) == 0 && strcmp(p->password, password) == 0)

{

p = lhead->next;

fp1 = fopen("login.txt", "w");

while (p != NULL)

{

fprintf(fp1, "%s %s\n", p->account, p->password);

p = p->next;

}

fclose(fp1);

printf("success!");

Sleep(1000);

return name;

}

else {

printf("account doesn't match with the password,please rewite");

Sleep(1000);

}

}

else

{

printf("don't find this account register now？(Y/N)");

getchar();

char c;

scanf("%c", &c);

if (c == 'y' || c == 'Y')

{

system("cls");

printf("Welcome to the address book mangement\nplease input your account");

scanf("%s", name);

printf("please input your password:");

scanf("%s", password);

char lpassword[MAX];

while (true)

{

printf("please input your password again：");

scanf("%s", lpassword);

if (strcmp(lpassword, password) == 0)

{

if (lhead->next == NULL)

{

p = lhead;

}

else {

p = lhead->next;

while (p->next != NULL) { p = p->next; }// Find as node }

p->next = (land\*)malloc(sizeof(land));

strcpy(p->next->account, name);

strcpy(p->next->password, password);

p->next->next = NULL;

printf("success");

Sleep(1000);

system("cls");

break;

}

else {

printf("The password entered twice is different. Please re-enter it：");

scanf("%s", password);

}

}

}

else {

printf("failed");

return "null";

}

}

}

break;

}

else if (select == 2)

{

while (1)

{

system("cls");

printf("Welcome to the address book mangement\nplease input your account：");

scanf("%s", name);

p = lhead->next;

while (p != NULL)

{

i = 0;

if (strcmp(p->account, name) == 0)

{

i = 1;

break;

}

p = p->next;

}

if (i == 1)

{

printf("The user you entered exist already. Please re-enter。");

Sleep(1000);

}

else {

break;

}

}

printf("please input your password:：");

scanf("%s", password);

char lpassword[MAX];

while (1)

{

printf("please input your password again：");

scanf("%s", lpassword);

if (strcmp(lpassword, password) == 0)

{

if (lhead->next == NULL)

{

p = lhead;

}

else {

p = lhead->next;

while (p->next != NULL) { p = p->next; }// Find as node

}

p->next = (land\*)malloc(sizeof(land));

strcpy(p->next->account, name);

strcpy(p->next->password, password);

p->next->next = NULL;

printf("success");

Sleep(1000);

system("cls");

break;

}

else {

printf("The passwords entered twice are different. Please re-enter them：");

scanf("%s", password);

}

}

//break;

}

else if (select == 3)// Change the password

{

while (1)

{

system("cls");

printf("welcome to modify the password\n");

printf("pleas your account");

scanf("%s", name);

p = lhead->next;

while (p != NULL)

{

if (strcmp(p->account, name) == 0)

{

break;

}

p = p->next;

}

if (p != NULL)

{

while (1)

{

char lp[MAX];

printf("please input your old password");

scanf("%s", password);

if (strcmp(password, p->password) != 0)

{

printf("your old password is wrong rewrite\n");

Sleep(1000);

continue;

}

break;

}

printf("please input your new password：");

scanf("%s", password);

char lpassword[MAX];

while (true)

{

printf("please input your new password again：");

scanf("%s", lpassword);

if (strcmp(lpassword, password) == 0)

{

strcpy(p->password, password);

printf("suceess");

Sleep(1000);

system("cls");

break;

}

else {

printf("The passwords entered twice are different. Please re-enter them：");

scanf("%s", password);

}

}

break;

}

else

{

printf("The user you entered does not exist. Please re-enter");

}

}

}

else {

//setbuf(stdin, NULL);

getchar();

printf("the serial number is wrong please rewrite");

Sleep(500);

system("cls");

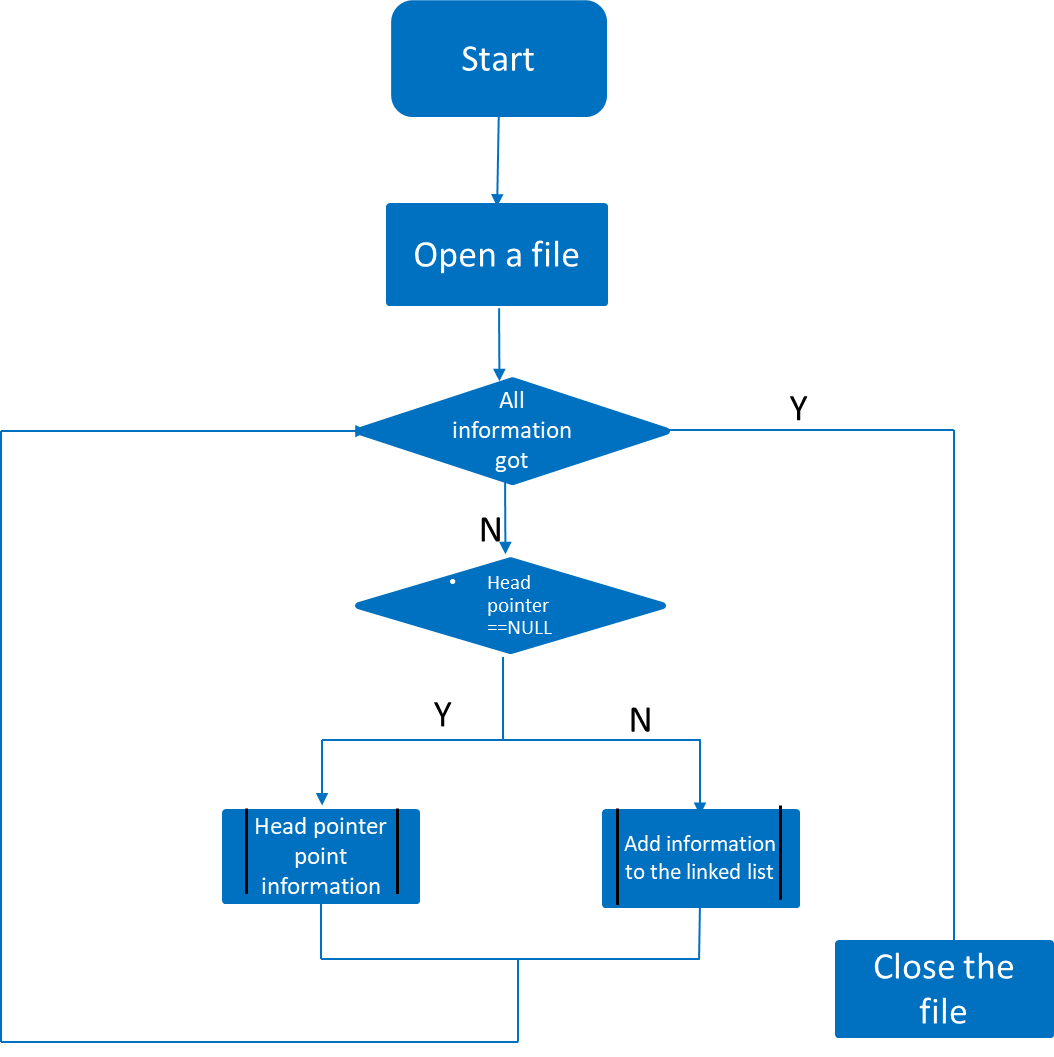
}

}

}

**Read the file**

This is the read file function, which converts the information in the file into a linked list in the program. This allows the user to manipulate this information.



**Main menu**

After entering the main menu, users can select the function they want to perform according to the number. The following functions can be performed: add contacts, browse all contact information, find contacts, delete contacts, edit the corresponding information of contacts, sort contacts and log out of the system.

 printf("\t\t\t\tYou have %d contacts\n", num1);

printf("\t\t\t\*\*\*please choose the founction\*\*\*\*\*\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t\*\*\* 0.Exit \*\*\*\*\*\n");

printf("\t\t\t\*\*\* 1.Add contact \*\*\*\*\*\n");

printf("\t\t\t\*\*\* 2.View all contacts \*\*\*\*\*\n");

printf("\t\t\t\*\*\* 3.Find contact \*\*\*\*\*\n");

printf("\t\t\t\*\*\* 4.delete conatct \*\*\*\*\*\n");

printf("\t\t\t\*\*\* 5.edit contact \*\*\*\*\*\n");

printf("\t\t\t\*\*\* 6.rank \*\*\*\*\*\n");

scanf("%d", &num2);

switch (num2)

**Add contact**

In this feature, users can add their contact information, while users cannot add two contacts with the same name. Contact mailbox must contain @, otherwise this input is illegal, the user needs to re-enter.

**The code**  case 1:

people\_t\* pNode;

pNode = (people\_t\*)malloc(sizeof(people\_t)); //ask for the reserved memory space for a node

system("cls");

printf("please input name");

scanf("%s", pNode->Name);

checknum = checkname(pNode->Name, Head);

while (getchar() != '\n');

while (checknum != 0)

{

printf("there has same name, rewrite\n");

scanf("%s", pNode->Name);

checknum = checkname(pNode->Name, Head);

}

printf("name valid\n");

printf("please input age (number)");

scanf("%d", &pNode->age);

while (getchar() != '\n');

printf("please input gender(man/woman)");

scanf("%s", pNode->Gender);

checknum = checkgender(pNode->Gender);

while (checknum == 0)

{

printf("not valid\n");

scanf("%s", pNode->Gender);

checknum = checkgender(pNode->Gender);

}

printf("success\n");

while (getchar() != '\n');

printf("please input e-mail（needs @）");

scanf("%s", pNode->Email);

p = find(pNode->Email, c);

while (p != 1)

{

printf("please input right e-mail！");

scanf("%s", pNode->Email);

p = find(pNode->Email, c);

}

while (getchar() != '\n');

printf("please input telephonenumber");

scanf("%s", pNode->Mobilephonenum);

while (getchar() != '\n');

system("cls");

outputpreinfor(pNode);

printf("Add?\n1.Y\n2.N\n");

scanf("%d", &num2);

if (num2 == 1)

{

pNode->pNext = Head;

Head = pNode;

printf("sucess\n");

writetext(Head, account);

}

printf("back to the main menu?\n1.Y\n2.N\n");

scanf("%d", &num2);

if (num2 == 1)

{

flag = 0;

system("cls");

break;

}

else flag = 2;

break;

**Correlation function**：

**Find a contact**：By comparing the input characters with the data in the database, the information of the same element is found and output.

**Code**：

int find(char \*s, char c)

{

char \*x = s;

int num = 0;

for (x = s; \*x != 0; x++)

{

if (\*x == c)

{

num++;

}

}

return num;

Delete contact：Find the corresponding element through comparison, delete the corresponding node, free the memory, and move the pointer to the next node.

Code：

people\_t\* pTemp = NULL;// Define temporary pointer

people\_t\* pr = NULL;

int index = 1;// The ordinal number of the element in a linked list

int i;

if (head == NULL)

{

printf("No contacts！\n");

}

pTemp = head;

while (pTemp != 0) // When the temporary pointer does not equal a null value, it is at most output to the linked list to the last element {

if (strcmp(pTemp->Name, inforamtion) == 0 || (strcmp(pTemp->Mobilephonenum, inforamtion)) == 0)// Find a contact

{

if (pTemp == head)

{

head = pTemp->pNext;

}

else

{

pr = head;

for (i = 1; i<index - 1; i++)

pr = pr->pNext;

pr->pNext = pTemp->pNext;

}

free(pTemp); // Free the memory of the deleted node

break;

}

else

{

pTemp = pTemp->pNext;// Move the temporary pointer to the next element

index++;

}

}

return(head);Edit

**the code**

case 5:

if (Head == NULL)

printf("Address book is NULL, please add first");

else

{

outputallinfor(Head);

printf("please input the name");

scanf("%s", information);

checknum = checkname(information, Head);

while (getchar() != '\n');

while (checknum == 0)

{

printf("Without this contact, please re-enter your first name\n");

scanf("%s", information);

checknum = checkname(information, Head);

}

target = localize(Head, information);

printf("which information to edit:\nname：%s\n gender：%s\n gender：%d\n e-mail：%s\n telephonenumber：%s\n ", target->Name, target->Gender, target->age, target->Email, target->Mobilephonenum);

printf("input the information\n1.name\n2.gender\n3.age\n4.e-mail\n5.telephonenumber\n");

scanf("%d", &num2);

switch (num2)

{

case 1:

printf("edit name to");

scanf("%s", name);

checknum = checkname(name, Head);

while (getchar() != '\n');

while (checknum != 0)

{

printf("The same contact already exists. Please enter the name again \n");

scanf("%s", name);

checknum = checkname(name, Head);

}

printf("name valid\n");

editpeople1(target, name);

printf("success");

writetext(Head, account);

break;

case 2:

printf("edit gender to");

scanf("%s", name);

editpeople3(target, name);

printf("修改成功");

writetext(Head, account);

break;

case 3:

printf("edit age to");

scanf("%d", &num3);

editpeople2(target, num3);

printf("success");

writetext(Head, account);

break;

case 4:

printf("edit e-mail to");

scanf("%s", name);

editpeople4(target, name);

printf("success");

writetext(Head, account);

break;

case 5:

printf("edit telephonenumber to");

scanf("%s", name);

editpeople5(target, name);

printf("success");

writetext(Head, account);

break;

}

}

printf("back to main menu\n1.Y\n2.N\n");

scanf("%d", &num2);

if (num2 == 1)

{

flag = 0;

system("cls");

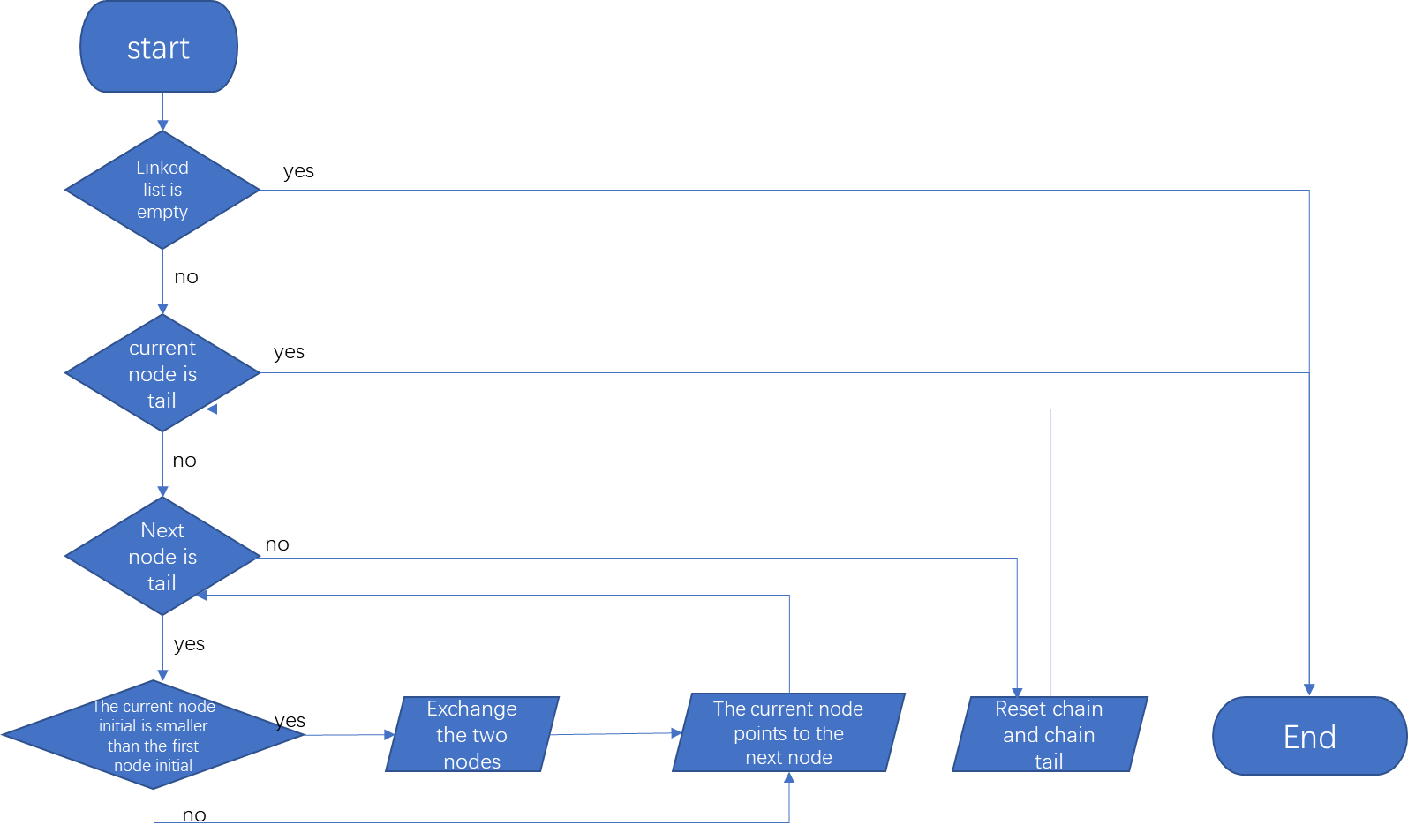
break;

}

**Rank**

In this part, users can sort all the contacts in the address book in ascending or descending order, making it easier to view.

Rankup:



The code:

people\_t\* rankup(people\_t\* head)// In ascending alphabetical order

{

people\_t\* cur, \*tail;

cur = head;

tail = NULL;

if (cur == NULL || cur->pNext == NULL) {

return head;

}

while (cur != tail) {

while (cur->pNext != tail) {

if (strcmp(cur->Name, cur->pNext->Name)<0) {

int temp = cur->age;

cur->age = cur->pNext->age;

cur->pNext->age = temp;

char ct[MAX];

strcpy(ct, cur->Name);

strcpy(cur->Name, cur->pNext->Name);

strcpy(cur->pNext->Name, ct);

char G[5];

strcpy(G, cur->Gender);

strcpy(cur->Gender, cur->pNext->Gender);

strcpy(cur->pNext->Gender, G);

strcpy(ct, cur->Email);

strcpy(cur->Email, cur->pNext->Email);

strcpy(cur->pNext->Email, ct);

strcpy(ct, cur->Mobilephonenum);

strcpy(cur->Mobilephonenum, cur->pNext->Mobilephonenum);

strcpy(cur->pNext->Mobilephonenum, ct);

}

cur = cur->pNext;

}

tail = cur;

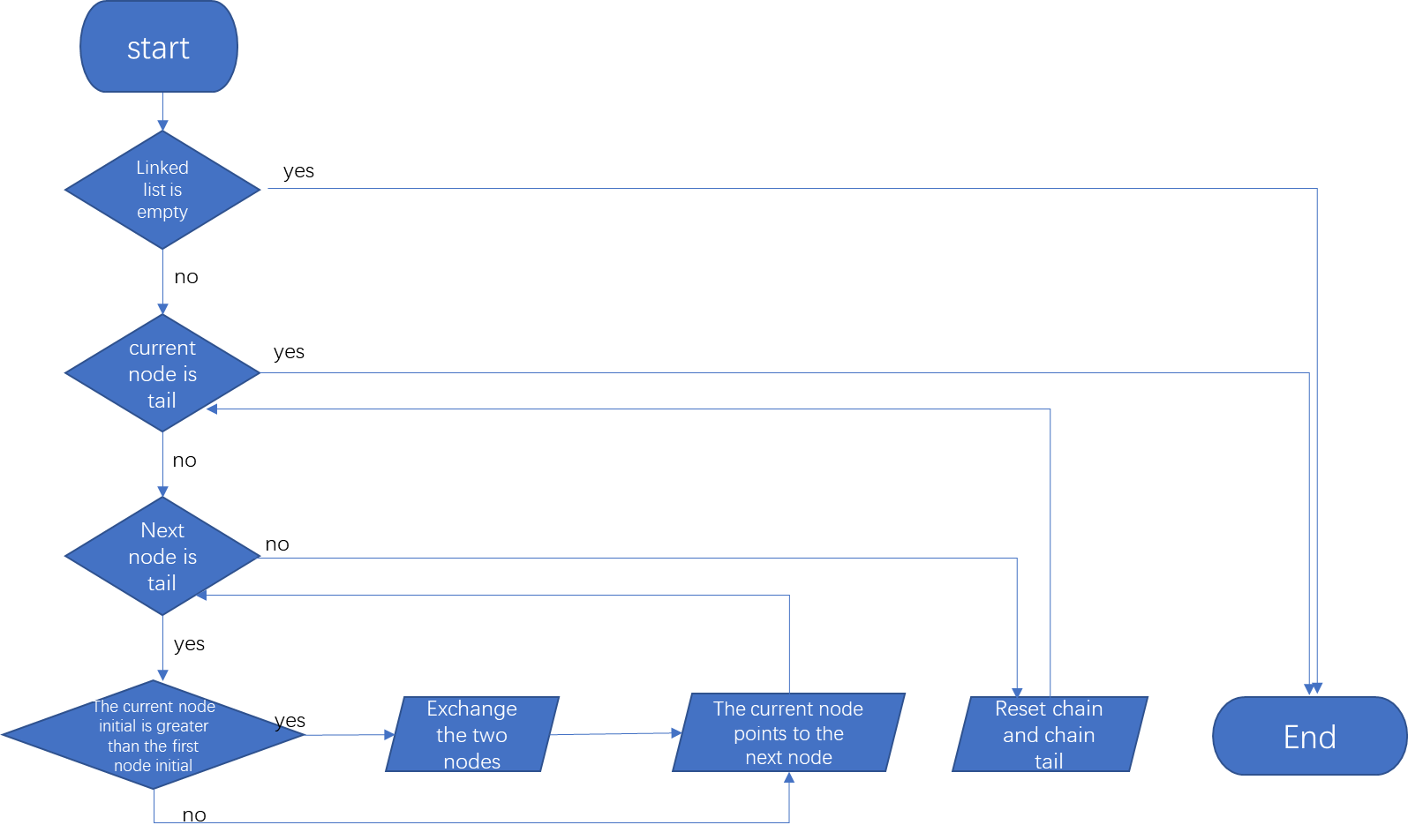
cur = head;

}

return head;

}

**Rankdown**



**The code:**

people\_t\* rankdown(people\_t\*head)// In descending alphabetical order

{

people\_t\* cur, \*tail;

cur = head;

tail = NULL;

if (cur == NULL || cur->pNext == NULL) {

return head;

}

while (cur != tail) {

while (cur->pNext != tail) {

if (strcmp(cur->Name, cur->pNext->Name)>0) {

int temp = cur->age;

cur->age = cur->pNext->age;

cur->pNext->age = temp;

char ct[MAX];

strcpy(ct, cur->Name);

strcpy(cur->Name, cur->pNext->Name);

strcpy(cur->pNext->Name, ct);

char G[5];

strcpy(G, cur->Gender);

strcpy(cur->Gender, cur->pNext->Gender);

strcpy(cur->pNext->Gender, G);

strcpy(ct, cur->Email);

strcpy(cur->Email, cur->pNext->Email);

strcpy(cur->pNext->Email, ct);

strcpy(ct, cur->Mobilephonenum);

strcpy(cur->Mobilephonenum, cur->pNext->Mobilephonenum);

strcpy(cur->pNext->Mobilephonenum, ct);

}

cur = cur->pNext;

}

tail = cur;

cur = head;

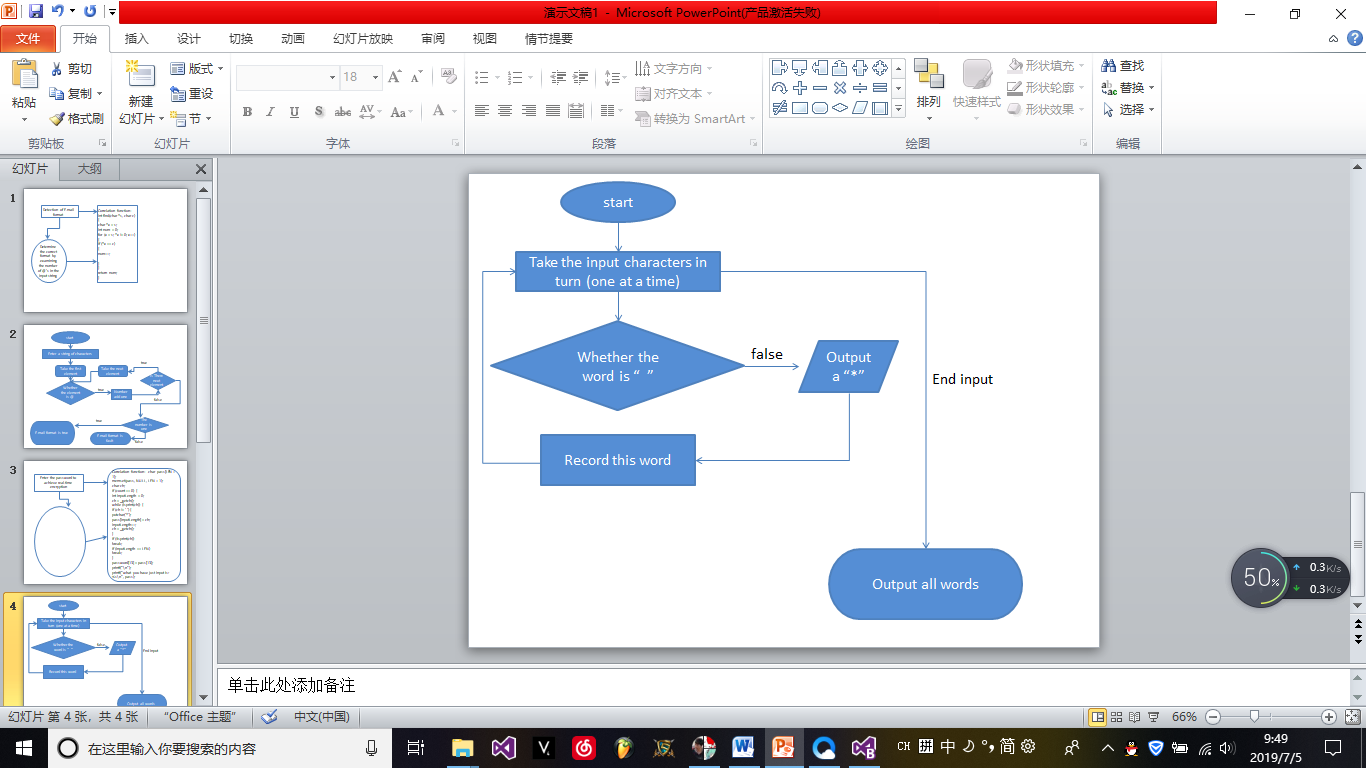
}

return head;

}

**Innovation**

Based on the traditional address book, we have implemented some innovations.

**Input real-time encryption：**

**code：**

**Correlation function**：

char pass[LEN + 1];

memset(pass, NULL, LEN + 1);

char ch;

if (count == 0) {

int inputLength = 0;

ch = \_getch();

while (isprint(ch)) {

if (ch != ' ') {

putchar('\*');

pass[inputLength] = ch;

inputLength++;

ch = \_getch();

}

if (!isprint(ch))

break;

if (inputLength == LEN)

break;

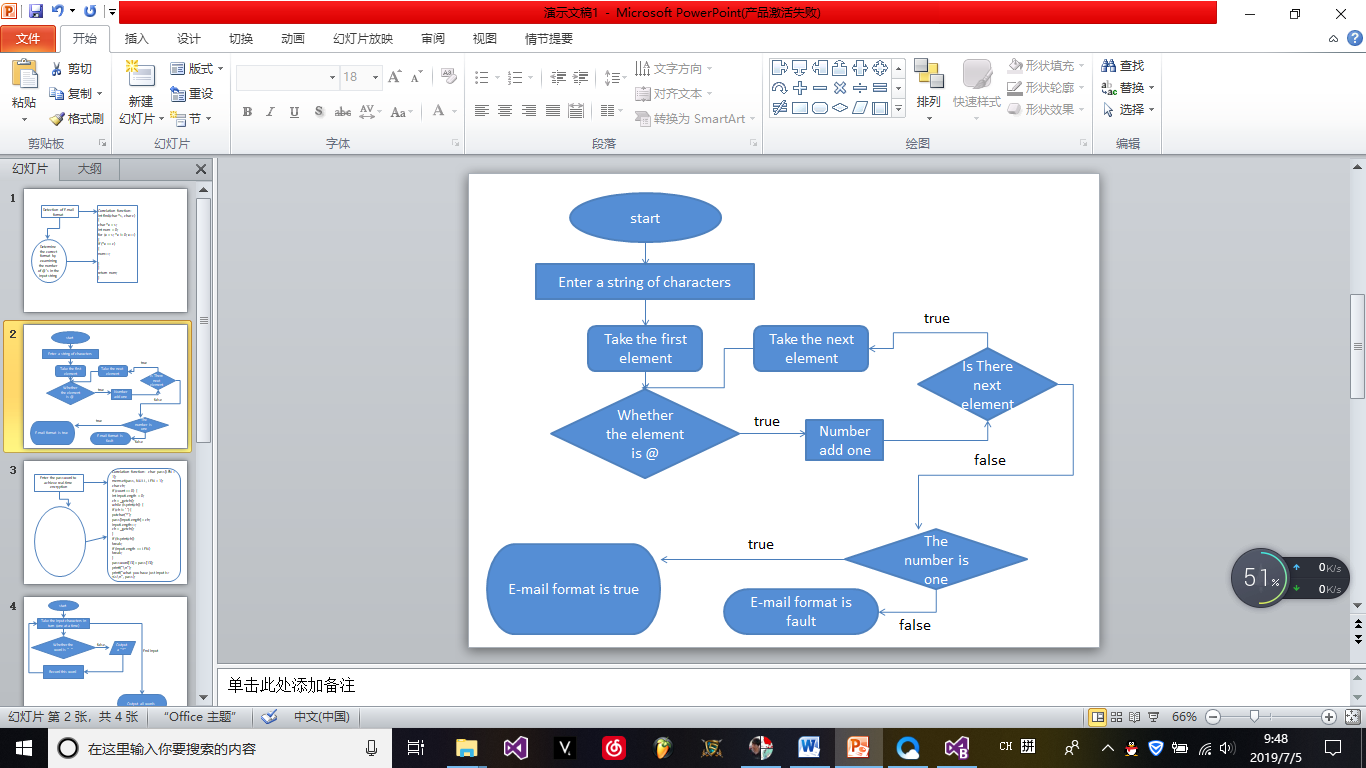
}

password[13] = pass[13];

printf("\n");

printf("what you have just input is: %s\n", pass);

**Check that the E-mail is formatted correctly：**



**code**：Correlation function：

int find(char \*s, char c)

{

char \*x = s;

int num = 0;

for (x = s; \*x != 0; x++)

{

if (\*x == c)

{

num++;

}

}

return num;

}

printf("please input e-mail（needs @）");

scanf("%s", pNode->Email);

p = find(pNode->Email, c);

while (p != 1)

{

printf("please input right e-mail！");

scanf("%s", pNode->Email);

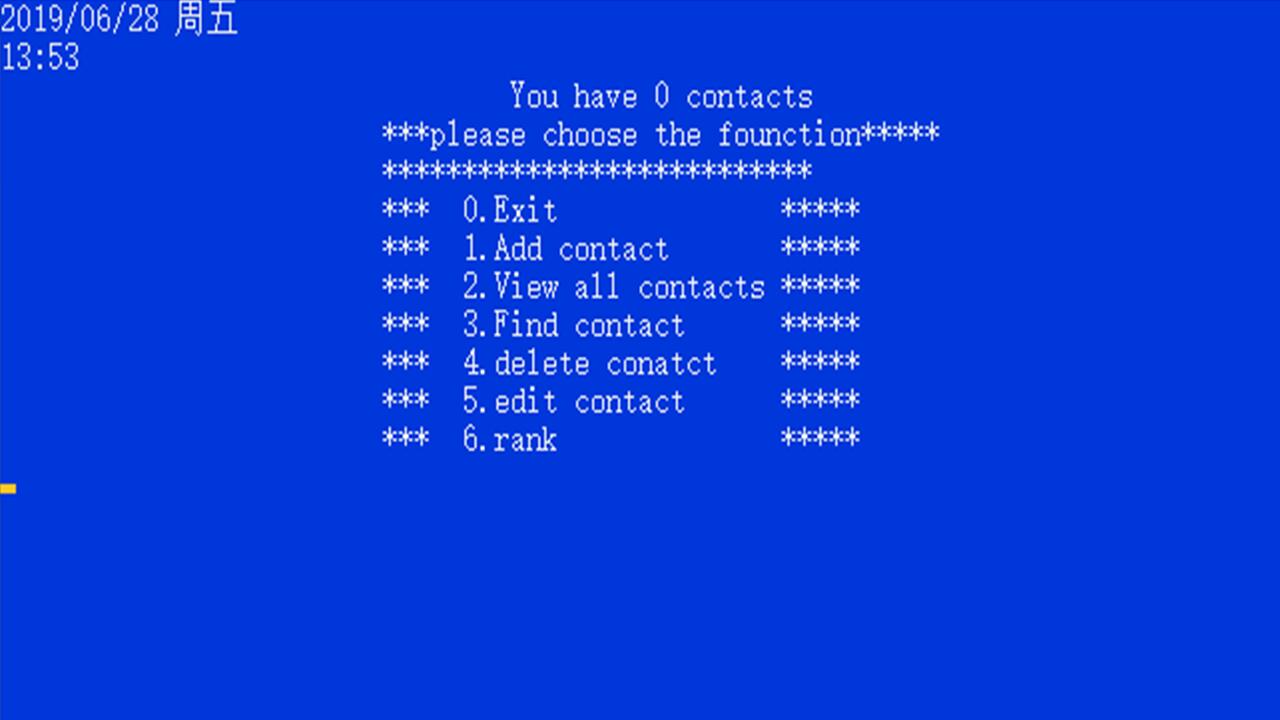
p = find(pNode->Email, c);

}

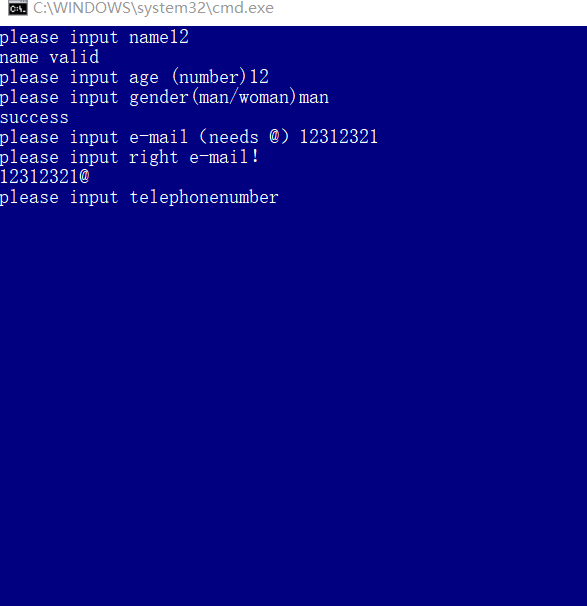
while (getchar() != '\n');

**Test**

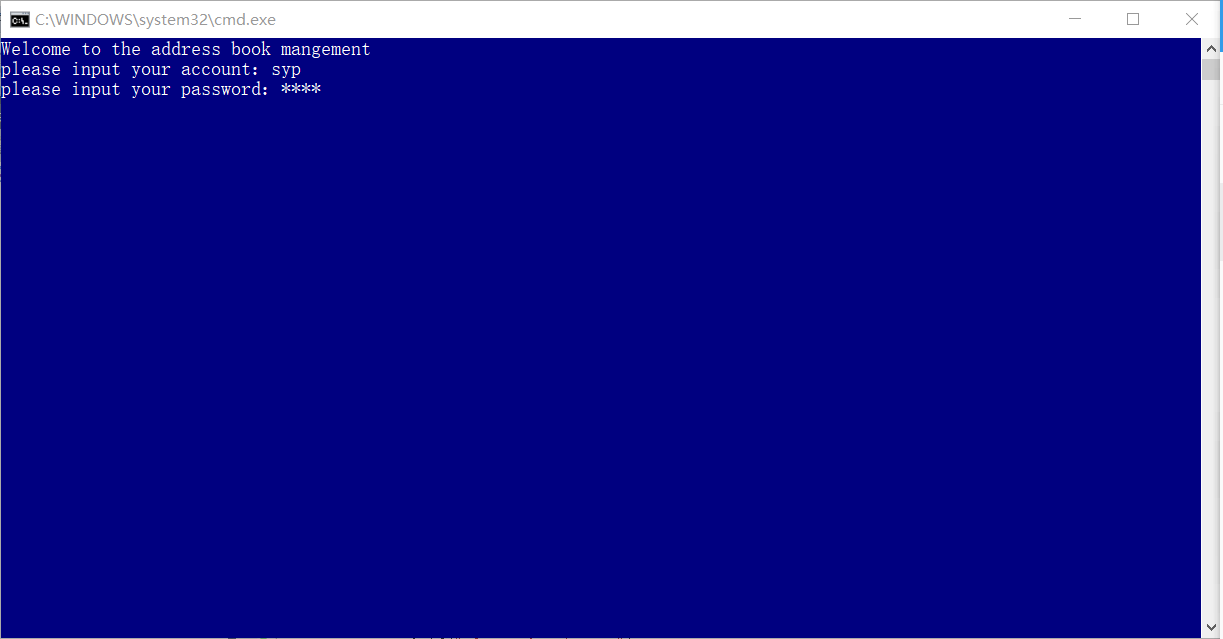
**Main menu**



**Fault tolerance**



**Password encryption**



**Conclusion**

Our program successfully achieved the management of the address book, the main use of the knowledge of the C language: linked list, read and write files, Pointers, structure and sorting algorithm. At the same time, our program has a powerful function, prepared a certain error statement, has a more humane interface.