第28讲：单链表的应用

目录

1. 单链表经典算法OJ题目
2. 基于单链表再实现通讯录项目

正文开始

单链表经典算法

1. 链表经典算法OJ题目

1.1 单链表相关经典算法OJ题1：移除链表元素

1.2 单链表相关经典算法OJ题2：反转链表

1.3 单链表相关经典算法OJ题3：合并两个有序链表

1.4 单链表相关经典算法OJ题4：链表的中间结点

1.5 循环链表经典应用-环形链表的约瑟夫问题

> 著名的Josephus问题
>
> 据说著名犹太历史学家 Josephus有过以下的故事：在罗马人占领乔塔帕特后，39 个犹太人与Josephus及他的朋友躲到一个洞中，39个犹太人决定宁愿死也不要被人抓到，于是决定了一个自杀方式，41个人排成一个圆圈，由第1个人开始报数，每报数到第3人该人就必须自杀，然后再由下一个重新报数，直到所有人都自杀身亡为止。
>
> 然而Josephus 和他的朋友并不想遵从，Josephus要他的朋友先假装遵从，他将朋友与自己安排在第16个与第31个位置，于是逃过了这场死亡游戏。

1.6 单链表相关经典算法OJ题5：分割链表

2. 基于单链表再实现通讯录项目

```
1 //SList.h
2 //
3 // Created by mm on 2023/6/13.
4 //
```

```
#include<stdio.h>
#include<stdlib.h>
#include<assert.h>
#include"contact.h"

typedef struct PersonInfo SLTDataType;
//typedef int SLTDataType;
struct SListNode{
 struct SListNode* next;
 SLTDataType data;
}
void SLTPrint(SLTNode* phead);

//头部插入删除/尾部插入删除
void SLTPushBack(SLTNode** pphead, SLTDataType x);
void SLTPushFront(SLTNode** pphead, SLTDataType x);
void SLTPopBack(SLTNode** pphead);
void SLTPopFront(SLTNode** pphead);

//查找
SLTNode* SLTFind(SLTNode* phead, SLTDataType x);
//在指定位置之前插入数据
void SLTInsert(SLTNode** pphead, SLTNode* pos, SLTDataType x);
//删除pos节点
void SLTErase(SLTNode** pphead, SLTNode* pos);
//在指定位置之后插入数据
void SLTInsertAfter(SLTNode* pos, SLTDataType x);
//删除pos之后的节点
void SLTEraseAfter(SLTNode* pos);
//销毁链表
void SListDesTroy(SLTNode** pphead);
```

```
//contact.h
#pragma once
#define NAME_MAX 100
#define SEX_MAX 4
#define TEL_MAX 11
#define ADDR_MAX 100

//前置声明
typedef struct SListNode contact;

//用户数据
typedef struct PersonInfo
```

```
{
        char name[NAME_MAX];
        char sex[SEX_MAX];
        int age;
        char tel[TEL_MAX];
        char addr[ADDR_MAX];
}PeoInfo;

//初始化通讯录
void InitContact(contact** con);//实际调用的是链表的初始化接口（可以简单做一个头结点的初始化）
//添加通讯录数据
void AddContact(contact** con);// 链表尾插/头插
//删除通讯录数据
void DelContact(contact** con);//链表的删除指定位置的数据
//展示通讯录数据
void ShowContact(contact* con);//链表的打印
//查找通讯录数据
void FindContact(contact* con);
//修改通讯录数据
void ModifyContact(contact** con);
//销毁通讯录数据
void DestroyContact(contact** con);
```

```
//contact.c
#define _CRT_SECURE_NO_WARNINGS
#include"contact.h"
#include"SList.h"

void LoadContact(contact** con) {
        FILE* pf = fopen("contact.txt", "rb");
        if (pf == NULL) {
                perror("fopen error!\n");
                return;
        }

        //循环读取文件数据
        PeoInfo info;
        while (fread(&info, sizeof(info), 1, pf))
        {
                SLTPushBack(con, info);
        }
        printf("历史数据导入通讯录成功! \n");
}

```

```c
void InitContact(contact** con) {
	LoadContact(con);//把本地的通讯录数据导入到链表结构
}

void AddContact(contact** con) {
	PeoInfo info;
	printf("请输入姓名：\n");
	scanf("%s", &info.name);
	printf("请输入性别：\n");
	scanf("%s", &info.sex);
	printf("请输入年龄：\n");
	scanf("%d", &info.age);
	printf("请输入联系电话：\n");
	scanf("%s", &info.tel);
	printf("请输入地址：\n");
	scanf("%s", &info.addr);

	SLTPushBack(con, info);
	printf("插入成功！\n");

}

contact* FindByName(contact* con, char name[]) {
	contact* cur = con;
	while (cur)
	{
		if (strcmp(cur->data.name, name) == 0) {
			return cur;
		}
		cur = cur->next;
	}
	return NULL;
}

void DelContact(contact** con) {
	char name[NAME_MAX];
	printf("请输入要删除的用户姓名：\n");
	scanf("%s", name);

	contact* pos = FindByName(*con, name);
	if (pos == NULL) {
		printf("要删除的用户不存在，删除失败！\n");
		return;
	}
	SLTErase(con, pos);
	printf("删除成功！\n");
}
```

```
69
70 void ShowContact(contact* con) {
71         printf("%-10s %-4s %-4s %15s %-20s\n", "姓名", "性别", "年龄", "联系电
   话", "地址");
72         contact* cur = con;
73         while (cur)
74         {
75                 printf("%-10s %-4s %-4d %15s %-20s\n",
76                         cur->data.name,
77                         cur->data.sex,
78                         cur->data.age,
79                         cur->data.tel,
80                         cur->data.addr);
81                 cur = cur->next;
82         }
83 }
84
85 void FindContact(contact* con) {
86         char name[NAME_MAX];
87         printf("请输入要查找的用户姓名: \n");
88         scanf("%s", name);
89
90         contact* pos = FindByName(con, name);
91         if (pos == NULL) {
92                 printf("要查找的用户不存在，查找失败! \n");
93                 return;
94         }
95
96         printf("查找成功! \n");
97         printf("%-10s %-4s %-4d %15s %-20s\n",
98                 pos->data.name,
99                 pos->data.sex,
100                pos->data.age,
101                pos->data.tel,
102                pos->data.addr);
103 }
104
105 void ModifyContact(contact** con) {
106         char name[NAME_MAX];
107         printf("请输入要修改的用户名称:\n");
108         scanf("%s", &name);
109
110         contact* pos = FindByName(*con, name);
111         if (pos == NULL) {
112                 printf("要查找的用户不存在，修改失败! \n");
113                 return;
114         }
```

```
115         printf("请输入要修改的姓名: \n");
116         scanf("%s", pos->data.name);
117         printf("请输入要修改的性别: \n");
118         scanf("%s", pos->data.sex);
119         printf("请输入要修改的年龄: \n");
120         scanf("%d", &pos->data.age);
121         printf("请输入要修改的联系电话: \n");
122         scanf("%s", pos->data.tel);
123         printf("请输入要修改的地址: \n");
124         scanf("%s", pos->data.addr);
125
126         printf("修改成功! \n");
127 }
128
129 void SaveContact(contact* con) {
130         FILE* pf = fopen("contact.txt", "wb");
131         if (pf == NULL) {
132                 perror("fopen error!\n");
133                 return;
134         }
135         //将通讯录数据写入文件
136         contact* cur = con;
137         while (cur)
138         {
139                 fwrite(&(cur->data), sizeof(cur->data), 1, pf);
140                 cur = cur->next;
141         }
142
143         printf("通讯录数据保存成功! \n");
144 }
145
146 void DestroyContact(contact** con) {
147         SaveContact(*con);//在通讯录销毁之前，先把历史数据保存到本地文件中contact.txt
148         SListDesTroy(con);
149 }
```

完