

第29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 //
```

```

5
6  #include<stdio.h>
7  #include<stdlib.h>
8  #include<assert.h>
9  #include"contact.h"
10
11  typedef struct PersonInfo SLTDataType;
12  //typedef int SLTDataType;
13  struct SListNode{
14      struct SListNode* next;
15      SLTDataType data;
16  }
17  void SLTPrint(SLTNode* phead);
18
19  //头部插入删除/尾部插入删除
20  void SLTPushBack(SLTNode** pphead, SLTDataType x);
21  void SLTPushFront(SLTNode** pphead, SLTDataType x);
22  void SLTPopBack(SLTNode** pphead);
23  void SLTPopFront(SLTNode** pphead);
24
25  //查找
26  SLTNode* SLTFind(SLTNode* phead, SLTDataType x);
27  //在指定位置之前插入数据
28  void SLTInsert(SLTNode** pphead, SLTNode* pos, SLTDataType x);
29  //删除pos节点
30  void SLTErase(SLTNode** pphead, SLTNode* pos);
31  //在指定位置之后插入数据
32  void SLTInsertAfter(SLTNode* pos, SLTDataType x);
33  //删除pos之后的节点
34  void SLTEraseAfter(SLTNode* pos);
35  //销毁链表
36  void SListDesTroy(SLTNode** pphead);

```

```

1  //contact.h
2  #pragma once
3  #define NAME_MAX 100
4  #define SEX_MAX 4
5  #define TEL_MAX 11
6  #define ADDR_MAX 100
7
8  //前置声明
9  typedef struct SListNode contact;
10
11  //用户数据
12  typedef struct PersonInfo

```

```

13 {
14     char name[NAME_MAX];
15     char sex[SEX_MAX];
16     int age;
17     char tel[TEL_MAX];
18     char addr[ADDR_MAX];
19 }PeoInfo;
20
21 //初始化通讯录
22 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)
23 //添加通讯录数据
24 void AddContact(contact** con); // 链表尾插/头插
25 //删除通讯录数据
26 void DelContact(contact** con); //链表的删除指定位置的数据
27 //展示通讯录数据
28 void ShowContact(contact* con); //链表的打印
29 //查找通讯录数据
30 void FindContact(contact* con);
31 //修改通讯录数据
32 void ModifyContact(contact** con);
33 //销毁通讯录数据
34 void DestroyContact(contact** con);

```

```

1 //contact.c
2 #define _CRT_SECURE_NO_WARNINGS
3 #include "contact.h"
4 #include "SList.h"
5
6 void LoadContact(contact** con) {
7     FILE* pf = fopen("contact.txt", "rb");
8     if (pf == NULL) {
9         perror("fopen error!\n");
10        return;
11    }
12
13    //循环读取文件数据
14    PeoInfo info;
15    while (fread(&info, sizeof(info), 1, pf))
16    {
17        SLTPushBack(con, info);
18    }
19    printf("历史数据导入通讯录成功! \n");
20 }
21

```

```

22 void InitContact(contact** con) {
23     LoadContact(con); //把本地的通讯录数据导入到链表结构
24 }
25
26 void AddContact(contact** con) {
27     PeoInfo info;
28     printf("请输入姓名: \n");
29     scanf("%s", &info.name);
30     printf("请输入性别: \n");
31     scanf("%s", &info.sex);
32     printf("请输入年龄: \n");
33     scanf("%d", &info.age);
34     printf("请输入联系电话: \n");
35     scanf("%s", &info.tel);
36     printf("请输入地址: \n");
37     scanf("%s", &info.addr);
38
39     SLTPushBack(con, info);
40     printf("插入成功! \n");
41
42 }
43
44 contact* FindByName(contact* con, char name[]) {
45     contact* cur = con;
46     while (cur)
47     {
48         if (strcmp(cur->data.name, name) == 0) {
49             return cur;
50         }
51         cur = cur->next;
52     }
53     return NULL;
54 }
55
56 void DelContact(contact** con) {
57     char name[NAME_MAX];
58     printf("请输入要删除的用户姓名: \n");
59     scanf("%s", name);
60
61     contact* pos = FindByName(*con, name);
62     if (pos == NULL) {
63         printf("要删除的用户不存在, 删除失败! \n");
64         return;
65     }
66     SLTErase(con, pos);
67     printf("删除成功! \n");
68 }

```

```
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
148     //contact.txt
149     SListDesTroy(con);
150 }

```

完