

**数据结构课程实验报告**

**（1）**

**姓名：顾宬书**

**学号：2135061913**

**专业：通信**

**完成日期：2023.3.5**

**目录**

[目录 - 1 -](#_Toc434170429)

[1 设计要求 - 2 -](#_Toc434170430)

[2 程序功能框图 - 2 -](#_Toc434170431)

[3 数据结构说明 - 2 -](#_Toc434170432)

[4 重要算法核心代码 - 2 -](#_Toc434170433)

[5 测试运行界面 - 2 -](#_Toc434170434)

[6 完整源程序 - 3 -](#_Toc434170435)

**数据结构实验一（线性表）**

**<通讯录的制作> 设计要求**

【问题描述】

编写一个小型通讯录管理系统，完成相关人员信息的建立、显示、查找、插入、删除等操作。要有良好的界面和较强的容错性。

【基本要求】

应包含以下几方面的功能：

1. 输入信息
2. 显示信息
3. 查找信息，以姓名为关键字
4. 删除信息
5. 信息存储到磁盘文件
6. 信息从磁盘文件导入

【数据】

通讯录的每条信息至少包含以下几项内容：

1. 姓名
2. 街道
3. 城市
4. 国家
5. 电话号码

【实现方法】

使用双向链表进行设计。需要用到C语言中文件读写函数进行磁盘文件访问。

**重要算法核心代码**

typedef struct ListNode {

    char\* name;

    char\* street;

    char\* city;

    char\* country;

    int tel;

    struct ListNode\* next;

    struct ListNode\* pre;

    int isEnd;

}Node;

Node\* search(char\* name, Node\* list) {

    Node\* ptr = list;

    while (ptr->isEnd != 1) {

        if (strcmp(name, ptr->name) == 0) {

            return ptr;

        }

        else {

            ptr = ptr->next;

        }

    }

    return NULL;

}

void append(Node\* target, Node\* list) {

    Node\* end = list;

    while (end->isEnd != 1) {

        end = end->next;

    }

    if (end->pre != NULL) {

        end->pre->next = target;

        target->pre = end->pre;

        target->next = end;

        end->pre = target;

        return;

    }

    else {

        list->city = target->city;

        list->country = target->country;

        list->name = target->name;

        list->street = target->street;

        list->tel = target->tel;

        list->isEnd = 0;

        Node\* newEnd = (Node\*)malloc(sizeof(Node));

        init(newEnd);

        newEnd->isEnd = 1;

        newEnd->pre = list;

        list->next = newEnd;

        list->pre = NULL;

    }

}

void del(Node\* target, Node\* list) {

    if (list->isEnd == 1) {

        printf("Erro\n");

        return;

    }

    if (target->pre != NULL) {

        target->pre->next = target->next;

        target->next->pre = target->pre;

        free(target);

        return;

    }

    else {

        list = list->next;

        free(list->pre);

        list->pre = NULL;

        return;

    }

}

**完整源程序**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct ListNode {

    char\* name;

    char\* street;

    char\* city;

    char\* country;

    int tel;

    struct ListNode\* next;

    struct ListNode\* pre;

    int isEnd;

}Node;

void init(Node\* node) {

    node->name = (char\*)malloc(sizeof(char) \* 0xFFFF);

    node->street = (char\*)malloc(sizeof(char) \* 0xFFFF);

    node->city = (char\*)malloc(sizeof(char) \* 0xFFFF);

    node->country = (char\*)malloc(sizeof(char) \* 0xFFFF);

    node->tel = 0;

    node->next = NULL;

    node->pre = NULL;

    node->isEnd = 0;

}

Node\* search(char\* name, Node\* list) {

    Node\* ptr = list;

    while (ptr->isEnd != 1) {

        if (strcmp(name, ptr->name) == 0) {

            return ptr;

        }

        else {

            ptr = ptr->next;

        }

    }

    return NULL;

}

void append(Node\* target, Node\* list) {

    Node\* end = list;

    while (end->isEnd != 1) {

        end = end->next;

    }

    if (end->pre != NULL) {

        end->pre->next = target;

        target->pre = end->pre;

        target->next = end;

        end->pre = target;

        return;

    }

    else {

        list->city = target->city;

        list->country = target->country;

        list->name = target->name;

        list->street = target->street;

        list->tel = target->tel;

        list->isEnd = 0;

        Node\* newEnd = (Node\*)malloc(sizeof(Node));

        init(newEnd);

        newEnd->isEnd = 1;

        newEnd->pre = list;

        list->next = newEnd;

        list->pre = NULL;

    }

}

void del(Node\* target, Node\* list) {

    if (list->isEnd == 1) {

        printf("Erro\n");

        return;

    }

    if (target->pre != NULL) {

        target->pre->next = target->next;

        target->next->pre = target->pre;

        free(target);

        return;

    }

    else {

        list = list->next;

        free(list->pre);

        list->pre = NULL;

        return;

    }

}

Node\* createList(Node\* head) {

    Node\* list = head;

    head->next = (Node\*)malloc(sizeof(Node));

    init(head->next);

    head->next->isEnd = 1;

    head->next->pre = head;

    return list;

}

void show(Node\* node) {

    if (node == NULL) {

        printf("Erro\n");

        return;

    }

    printf("name: %s\n", (node->name));

    printf("street: %s\n", (node->street));

    printf("city: %s\n", (node->city));

    printf("country: %s\n", (node->country));

    printf("tel : %d\n", node->tel);

}

void menu() {

    printf("1:search\n2:input\n3:del\n4:output\n5:append\n6:quit\n");

    return;

}

void input(Node\* list) {

    FILE\* f = fopen("input.txt", "a+");

    if (feof(f)) {

        return;

    }

    while (!feof(f)) {

        Node\* newNode = (Node\*)malloc(sizeof(Node));

        init(newNode);

        fscanf(f, "%s", newNode->name);

        fscanf(f, "%s", newNode->street);

        fscanf(f, "%s", newNode->city);

        fscanf(f, "%s", newNode->country);

        fscanf(f, "%d", &(newNode->tel));

        append(newNode, list);

    }

    if (f != NULL)

        fclose(f);

    return list;

}

void output(Node\* list) {

    FILE\* f = fopen("input.txt", "w");

    Node\* ptr = list;

    while (ptr->isEnd != 1) {

        fprintf(f, "%s ", ptr->name);

        fprintf(f, "%s ", ptr->street);

        fprintf(f, "%s ", ptr->city);

        fprintf(f, "%s ", ptr->country);

        fprintf(f, "%d", ptr->tel);

        fprintf(f, "\n");

        ptr = ptr->next;

    }

    fclose(f);

    return;

}

int main() {

    Node\* mainList = (Node\*)malloc(sizeof(Node));

    init(mainList);

    mainList->isEnd = 1;

    strcpy(mainList->name, "end");

    int choice = 0;

    while (1) {

        menu();

        scanf("%d", &choice);

        if (choice == 1) {

            if (mainList != NULL) {

                char\* name = (char\*)malloc(sizeof(char)\*0xFFFFF);

                scanf("%s", name);

                Node\* n = search(name, mainList);

                show(n);

            }

            else {

                printf("Please input information fist.\n");

            }

        }

        else if (choice == 2) {

            input(mainList);

        }

        else if (choice == 3) {

            char\* name = (char\*)malloc(sizeof(char) \* 0xFFFFF);

            scanf("%s", name);

            if (search(name, mainList) != NULL) {

                del(search(name, mainList), mainList);

            }

            else {

                printf("Nobody found.\n");

            }

        }

        else if (choice == 4) {

            if (mainList != NULL) {

                output(mainList);

            }

            else {

                printf("Please input information first.\n");

            }

        }

        else if (choice == 5) {

            Node\* newNode = (Node\*)malloc(sizeof(Node));

            init(newNode);

            scanf("%s", newNode->name);

            scanf("%s", newNode->street);

            scanf("%s", newNode->city);

            scanf("%s", newNode->country);

            scanf("%d", &(newNode->tel));

            append(newNode, mainList);

        }

        else if(choice == 6) {

            return 0;

        }

        else {

            continue;

        }

    }

}