

# 单链表操作的实现

## 1. 需求分析

单链表的基本操作包括了创建、插入、删除、查找、遍历。

(1) 输入的形式和输入值的范围：程序是按数字键选择对应功能的，在各项功能中，创建、插入、删除、查找需要输入数据，输入的数据为数字。

(2) 输出的形式：在单链表初始化后显示初始化成功，在检查链表是否为空时显示是否为空的结果，在将清空链表后显示链表已清空，在插入删除数据后会显示链表，在遍历后会显示所有的元素，在查找后会显示指定的元素。

(3) 程序所能达到的功能：完成单链表的初始化，检查链表是否为空，建立单链表，清空单链表，求链表长度，遍历链表，按位置查找元素，查找指定元素的位置，按位置插入元素，在指定元素后面插入，按位置删除元素，按值删除元素。

(4) 测试数据：

- 初始化后输入需要链表的长度，再依次输入该节点的元素，
- 选择求链表长度；
- 选择检查链表是否为空；
- 选择按位置查找元素，输入 1；

- 选择在指定元素后面插入，依次输入 g,d;
- 选择按位置删除元素，输入 1;
- 选择清空，选择遍历链表;

## 2. 概要设计

(1) 抽象数据类型:

ADT LinkList{

数据对象  $D = \{a_i \mid a_i \in Elemset, i = 1, 2, \dots, n, n \geq 0\}$

数据关系  $R = \{ \langle a_{i-1}, a_i \rangle \mid a_{i-1}, a_i \in D, i = 2, \dots, n \}$

基本操作

InitList(L)

操作结果: 创建了一个空链表

Listempty (L)

初始条件: 链式线性表 L 已存在。

操作结果: 若 L 为空表, 则返回 TRUE, 否则返回 FALSE

ClearList(L)

初始条件: 链表已存在

操作结果: 将链表清为空链表

ListLength(L)

初始条件: 链式线性表 L 已存在,  $1 \leq i \leq \text{ListLength}(L)$ ,

操作结果: 在 L 中第 i 个位置之前插入新的数据元素 e, L 的长度加 1

ListInsert (L)

初始条件: 链表已存在

操作结果：插入一个元素到链表中

ListDelete (L)

初始条件：链表已存在

操作结果：删除链表中的元素

ListTraverse(L)

初始条件：链表已存在

操作结果：按顺序打印链表元素；

}ADT LinkedList

（2）本程序包含的函数：

主函数 main( )

插入一个元素 ListInsert ( )

删除一个元素 ListDelete ( )

获取链表的长度 ListLength ( )

打印链表 DispList( )

使用 C 语言实现

### 3. 详细设计

功能函数

```
(1) #ifndef _LINKLIST2022_H_
(2) #define _LINKLIST2022_H_
(3)
(4)
(5) #include <stdio.h>
(6) #include <stdlib.h>
(7) typedef int Status; /* Status是函数的类型, 其值是函数结果状态代码, 如OK等 */
(8)
(9) typedef struct Node {
(10)     int data;
```

```

(11)     struct Node *next;
(12) } Node;
(13) typedef struct Node LinkList; /* 定义LinkList */
(14)
(15)
(16) /* 初始化链式线性表 */
(17) //Status InitList(LinkList *L) {
(18) // L = (LinkList *)malloc(sizeof(LinkList)); /* 产生头结点,并使L指向此头结点
*/
(19) //
(20) // L->next = NULL; /* 指针域为空 */
(21)
(22) // return 1;
(23) //}
(24)
(25) /* 初始条件: 链式线性表L已存在。操作结果: 若L为空表,则返回TRUE,否则返回
FALSE */
(26) int ListEmpty(LinkList *L) {
(27)     if (L->next == NULL)
(28)         return 0;
(29)     else
(30)         return 1;
(31) }
(32)
(33) /* 初始条件: 链式线性表L已存在。操作结果: 将L重置为空表 */
(34) Status ClearList(LinkList *&L) {
(35)     LinkList *p, *q;
(36)     p = L->next;          /* p指向第一个结点 */
(37)     while (p != NULL) {    /* 没到表尾 */
(38)         q = p->next;
(39)         free(p);
(40)         p = q;
(41)     }
(42)     L->next = NULL;        /* 头结点指针域为空 */
(43)     return 1;
(44) }
(45)
(46) /* 初始条件: 链式线性表L已存在。操作结果: 返回L中数据元素个数 */
(47) int ListLength(LinkList *L) {
(48)     int i = 0;
(49)     LinkList *p ;
(50)     p = L->next; /* p指向第一个结点 */
(51)     for (;;) {
(52)         i++;

```

```

(53)      p = p->next;
(54)      if (p->next == NULL)
(55)          break;
(56)  }
(57)  return i;
(58) }
(59)
(60) /* 初始条件: 链式线性表L已存在,  $1 \leq i \leq \text{ListLength}(L)$ , */
(61) /* 操作结果: 在L中第i个位置之前插入新的数据元素e, L的长度加1 */
(62) Status ListInsert(LinkList *L, int i, int e) {
(63)     int j;
(64)     LinkList *p, *s;
(65)     p = (LinkList *)malloc(sizeof(LinkList));
(66)     p = L;
(67)     j = 1;
(68)     while (p && j < i) { /* 寻找第i个结点 */
(69)         p = p->next;
(70)         ++j;
(71)     }
(72)     if (!p || j > i)
(73)         return 0; /* 第i个元素不存在 */
(74)     s = (LinkList *)malloc(sizeof(LinkList));
(75)     /* 生成新结点(C语言标准函数) */
(76)
(77)     s->data = e;
(78)     s->next = p->next; /* 将p的后继结点赋值给s的后继 */
(79)     p->next = s; /* 将s赋值给p的后继 */
(80)     return 1;
(81) }
(82)
(83) /* 初始条件: 链式线性表L已存在,  $1 \leq i \leq \text{ListLength}(L)$  */
(84) /* 操作结果: 删除L的第i个数据元素, L的长度减1 */
(85) Status ListDelete(LinkList *L, int i) {
(86)     int j;
(87)     LinkList *p, *q;
(88)     p = L;
(89)     j = 1;
(90)     while (p->next && j < i) { /* 遍历寻找第i个元素 */
(91)         p = p->next;
(92)         ++j;
(93)     }
(94)     if (!(p->next) || j > i)
(95)         return 0; /* 第i个元素不存在 */
(96)     q = p->next;

```

```

(97)    p->next = q->next;           /* 将q的后继赋值给p的后继 */
(98)    free(q);                   /* 让系统回收此结点，释放内存 */
(99)    return 1;
(100) }
(101)
(102) /* 初始条件：链式线性表L已存在 */
(103) /* 操作结果：依次对L的每个数据元素输出 */
(104) Status ListTraverse(LinkList *L) {
(105)    LinkList *pi ;
(106)    pi = (LinkList *)malloc(sizeof(LinkList));
(107)    pi = L;
(108)    for (;;) {
(109)
(110)        printf("%d\n", pi->data);
(111)
(112)        if (pi->next == NULL)
(113)            break;
(114)        pi = pi->next;
(115)    }
(116)    printf("\n");
(117)    return 0;
(118) }
(119) #endif

```

## (120) 主函数

```

(121)
(122) #include <stdlib.h>
(123) #include <stdio.h>
(124) #include "linklist2022.h"
(125)
(126)
(127)
(128)
(129) int main() {
(130)    int m, n;
(131)    printf("请输入链表长度: ");
(132)    scanf("%d", &m);
(133)    LinkList *head, *end;
(134)    head = (LinkList *)malloc(sizeof(LinkList));
(135)    //    end = (LinkList *)malloc(sizeof(LinkList));
(136)
(137)    head->next = NULL;

```

```
(138) head->data = 0;
(139) end = head;
(140) // p = head;
(141) for (int i = 0; i < m; i++) {
(142)     int da;
(143)     printf("输入该节点的元素: ");
(144)     scanf("%d", &da);
(145)     LinkList *p;
(146)     p = (LinkList *)malloc(sizeof(LinkList));
(147)
(148)     p->data = da;
(149)     end->next = p;
(150)     end = p;
(151)
(152)     // end->next = NULL;
(153)
(154)
(155) }
(156) end->next = NULL;
(157) /*
(158) LinkList *pu;
(159) pu = (LinkList *)malloc(sizeof(LinkList));
(160) pu = head;
(161) for (;;) {
(162)     printf("%d\n", pu->data);
(163)     if (pu->next == NULL) {
(164)         break;
(165)     }
(166)
(167)     pu = pu->next;
(168)
(169)
(170)
(171)
(172) }*/
(173)
(174) for (;;) {
(175)     printf("请输入要执行的操作: \n");
(176)     printf("1. 检查链表是否为空\n");
(177)     printf("2. 输出链表元素个数\n");
(178)     printf("3. 在第i个元素前插入e这个元素\n");
(179)     printf("4. 删除第i个数据元素\n");
(180)     printf("5. 打印链表\n");
(181)     printf("6. 将链表置为空表\n");
```

```

(182)     scanf("%d", &n);
(183)     if (n == 1) {
(184)         int jud = ListEmpty(head);
(185)         if (jud == 0) {
(186)             printf("链表为空\n");
(187)         } else if (jud == 1) {
(188)             printf("链表非空\n");
(189)         }
(190)     } else if (n == 2) {
(191)         int len = ListLength(head);
(192)         printf("链表长度为: %d\n", len);
(193)     } else if (n == 3) {
(194)         int a1, a2;
(195)         printf("输入插入位置和插入元素\n");
(196)         scanf("%d%d", &a1, &a2);
(197)
(198)         ListInsert(head, a1, a2);
(199)     } else if (n == 4) {
(200)         int a3;
(201)         printf("输入要删除的元素位置\n");
(202)         ListDelete(head, a3);
(203)     } else if (n == 5) {
(204)         ListTraverse(head) ;
(205)     } else if (n == 6) {
(206)         ClearList(head);
(207)         break;
(208)     }
(209)
(210) }
(211) return 0;

(212) }

```

### (213) 分析程序代码的质量

- 正确性。在一定的数据范围内，该程序能实现所需功能，所以正确性是没有问题的。
- 健壮性。在一定的数据输入范围内，该程序能较好地实现链表的操作。但是如果输入数据非法，该程序还是可能会产生一些预想不到的输出结果。比如虽然在链表长度输入上做出了不能




为其他符号和小于 0 的数字限定，但是在元素赋值输入是还是不能将宽字符的输入考虑在内，如果输入中文字符，链表将无法正确显示链表内容。所以，该程序的健壮性有待进一步的提高。当输入有误时，应返回一个表示错误的值，并终止程序的执行，以便在更高的抽象层次上进行处理。

## 4. 使用说明

单链表模板类的全部实现内容均包含在一个独立的 C 语言头文件 `linklist2022` 中，要在程序中使用这个类只需包含该头文件即可。这个类实现了创建、插入、删除、查找、遍历等操作。

## 5. 测试程序的运行结果



```
请输入链表长度: 3
输入该节点的元素: 2
输入该节点的元素: 1
输入该节点的元素: 4
请输入要执行的操作:
1. 检查链表是否为空
2. 输出链表元素个数
3. 在第i个元素前插入e这个元素
4. 删除第i个数据元素
5. 打印链表
6. 将链表置为空表
```

## 6. 心得体会

对于这种模板类的问题，在设计时应考虑通用性，可以通过将所需要的函数封装在一个 `.h` 文件中，随用随取，这样的话在改变主函数功能的时候可以最大限度地减少不必要的麻烦。

# 城市链表

## 1. 需求分析

城市链表的基本操作包括了显示、插入、删除、查找。

输入的形式和输入值的范围：程序是按数字键选择对应功能的，在各项功能中，插入、删除、查找、范围查找需要输入数据，输入的数据应为字符串和经纬度。

（1）输出的形式：在数据文件打开成功后显示数据已导入，在显示数据时显示所有导入的数据，在插入删除数据后会显示操作成功，在查找后会显示指定的城市经纬度。

（2）程序所能达到的功能：完成城市经纬度数据的导入，插入城市数据，删除城市数据，查找城市经纬度。

（3）测试数据：

- 选择显示数据；
- 选择插入城市数据，依次输入华盛顿 -77 38
- 选择删除城市数据，输入北京市的经纬度
- 选择查找城市经纬度，输入北京
- 选择通过经纬度查找城市名称

## 2. 概要设计

（1）抽象数据类型：

ADT CityList{

数据对象  $D = \{a_i \mid a_i \in Elemset, i = 1, 2, \dots, n, n \geq 0\}$

数据关系  $R = \{ \langle a_{i-1}, a_i \rangle \mid a_{i-1}, a_i \in D, i = 2, \dots, n \}$

基本操作

`readcity(&L, filename)`

操作结果：导入数据创建了一个城市链表

`createnode(&L, newCity)`

初始条件：链表已存在

操作结果：新增一个城市

`Check1(&L, cityName)`

初始条件：链表和城市已存在

操作结果：查找链表中匹配的经纬度并返回其值

`Check2(&L, jd, wd)`

初始条件：链表和城市已存在

操作结果：查找链表中匹配的城市并返回其值

`Delete(&L, cityName)`

初始条件：链表和城市已存在

操作结果：删除链表中的城市

}ADT CityList

(2) 本程序包含的函数：

主函数 `main()`

插入一个城市 `createnode()`

删除一个城市 `citydelete()`

查找城市名称 `check1()`

查找城市经纬度 `check2()`

数据导入函数 `readcity()`

### 3. 详细设计

主要操作的实现：

```
(1) #ifndef FUNCTION_H
(2) #define FUNCTION_H
(3)
(4)
(5) #include <stdio.h>
(6) #include <string.h>
(7) #include <stdlib.h>
(8)
(9)
(10) typedef struct stu {
(11)     char name[12];
(12)     double jd;
(13)     double wd;
(14)     struct stu *next;
(15) } city;
(16)
(17)
(18) int readcity(city *&end) { //读入城市信息，并储存在链表中
(19)     FILE *fp = fopen("city.txt", "a+");
(20)
(21)
(22)     while (fscanf(fp, "%s %lf %lf\n", &end->name, &end->jd, &end->wd) != EOF)
//尾插增加新城市
(23)         city *p ;
(24)         p = (city *)malloc(sizeof(city));
(25)
(26)         // p->next = NULL;
(27)         end->next = p;
(28)
(29)         end = p;
(30)         end->next = NULL;
(31)         //free(p);
(32)
(33)
```

```
(34)     }
(35)
(36)     fclose(fp);
(37)     return 1;
(38) }
(39)
(40)
(41)
(42)
(43) void createnode(city *&end) { //创建一个城市结点
(44)     city *pou;
(45)     pou = (city *)malloc(sizeof(city));
(46)
(47)     end->next = pou;
(48)     scanf("%s", end->name);
(49)     scanf("%lf%lf", &end->jd, &end->wd);
(50)     end = pou;
(51)     end->next = NULL;
(52)
(53) }
(54) void putall(city *c) { //打印所有的城市信息
(55)
(56)     for (;;) {
(57)         printf("城市名称: %s, 经度: %lf, 纬度: %lf\n", c->name, c->jd, c->wd);
(58)         c = c->next;
(59)         if (c->next == NULL) {
(60)             break;
(61)         }
(62)     }
(63)
(64)
(65) }
(66)
(67) void check1(int x, int y, city *c) { //通过经纬度查询城市信息
(68)     int jud = 0;
(69)     for (;;) {
(70)         if (c->jd == x && c->wd == y) {
(71)             printf("查询到城市为: %s\n", c->name);
(72)             jud = 1;
(73)         } else {
(74)             c = c->next;
(75)         }
(76)         if (c->next == NULL)
(77)             break;
```

```
(78) }
(79) if (jud == 0) {
(80)     printf("未查询到该城市! \n");
(81) }
(82) }
(83)
(84) void check2(char a[], city *c) { //通过城市名查询城市信息
(85)     int jud = 0;
(86)     for (;;) {
(87)         for (int i = 0; i < 5; i++) {
(88)             if (a[i] != c->name[i]) {
(89)                 break;
(90)             }
(91)             if (i == 4) {
(92)                 jud = 1;
(93)             }
(94)         }
(95)         if (jud = 1) {
(96)             printf("查找的城市经纬度为: %lf,%lf\n", c->jd, c->wd);
(97)             break;
(98)         } else {
(99)             c = c->next;
(100)        }
(101)    }
(102)    if (jud == 0) {
(103)        printf("未查询到该城市! \n");
(104)    }
(105) }
(106)
(107) void citydelete(int x, int y, city *&head) {
(108)     city *del, *mid;
(109)     int jud = 0;
(110)
(111)
(112)     del = head;
(113)     for (;;) {
(114)         if (del->jd == x && del->wd == y) {
(115)             mid = del;
(116)             del = del->next;
(117)             free(mid);
(118)             jud = 1;
(119)
(120)         } else {
(121)             del = del->next;
```

```

(122)     }
(123)     if (del->next == NULL)
(124)         break;
(125)     }
(126)     if (jud == 0) {
(127)         printf("本不存在该城市! \n");
(128)     } else if (jud == 1) {
(129)         printf("删除成功! ");
(130)     }
(131)
(132) }

(133) #endif

```

## 主函数

```

#include <stdio.h>
#include <stdlib.h>
#include "function.h"

int main() {
    city *head, *end;
    head = (city *)malloc(sizeof(city));
    end = head;
    end->next = NULL;
    readcity(end); //将文件中的信息读进链表中
    printf("创建成功! \n");
    printf("1. 打印当前城市集合\n");
    printf("2. 创建一个新的城市\n");
    printf("3. 通过坐标查询城市\n");
    printf("4. 通过城市名查经纬度\n");
    printf("5. 通过经纬度删除城市\n");
    for (;;) {
        int n;
        printf("请输入您要进行的操作: \n");
        scanf("%d", &n);
        if (n == 1) {
            putall(head);

        } else if (n == 2) {
            printf("输入新创建的城市名和经度纬度: ");
            createnode(end);

```

```

    } else if (n == 3) {
        printf("输入查询城市的经度和纬度: ");
        double x;
        double y;
        scanf("%lf%lf", &x, &y);
        check1(x, y, head);
    } else if (n == 4) {
        printf("输入查询城市的名称: ");
        char nm[10];
        scanf("%s", nm);
        check2(nm, head);
    } else if (n == 5) {
        printf("输入删除城市的经纬度: ");
        double x, y;
        scanf("%lf%lf", &x, &y);
        citydelete(x, y, head);
    }
}

return 0;
}

```

## 4. 调试分析

调分析程序代码的质量

- 正确性。在一定的数据范围内，该程序能实现所需功能，所以正确性是没有问题的。

## 5. 使用说明

城市类及部分操作函数的全部实现内容均包含在一个独立的 C 语言头文件 `function.h` 中，要在程序中使用这个类只需包含该头文件即可。这个头文件实现了城市类的导入数据、创建、删除、查找、遍历等操作。



## 6. 测试程序的运行结果

```
创建成功!
1. 打印当前城市集合
2. 创建一个新的城市
3. 通过坐标查询城市
4. 通过城市名查经纬度
5. 通过经纬度删除城市
请输入您要进行的操作:
1
```

(1)显示数据:

选择 1, 输出:

```
创建成功!
1. 打印当前城市集合
2. 创建一个新的城市
3. 通过坐标查询城市
4. 通过城市名查经纬度
5. 通过经纬度删除城市
请输入您要进行的操作:
1
城市名称: 北京, 经度: 116.280000, 纬度: 39.540000
城市名称: 青岛, 经度: 120.190000, 纬度: 36.040000
城市名称: 天津, 经度: 117.100000, 纬度: 39.100000
城市名称: 郑州, 经度: 113.420000, 纬度: 34.440000
城市名称: 石家庄, 经度: 114.260000, 纬度: 38.030000
城市名称: 开封, 经度: 114.230000, 纬度: 34.520000
城市名称: 保定, 经度: 115.280000, 纬度: 38.530000
城市名称: 洛阳, 经度: 112.260000, 纬度: 34.430000
城市名称: 唐山, 经度: 118.090000, 纬度: 39.370000
城市名称: 许昌, 经度: 113.480000, 纬度: 34.000000
城市名称: 秦皇岛, 经度: 119.370000, 纬度: 39.540000
城市名称: 新乡, 经度: 113.540000, 纬度: 35.180000
城市名称: 张家口, 经度: 114.550000, 纬度: 40.510000
城市名称: 武汉, 经度: 114.200000, 纬度: 30.370000
城市名称: 承德, 经度: 117.520000, 纬度: 40.590000
城市名称: 宜昌, 经度: 111.170000, 纬度: 30.420000
城市名称: 太原, 经度: 112.330000, 纬度: 37.510000
城市名称: 沙市, 经度: 112.170000, 纬度: 30.160000
城市名称: 大同, 经度: 113.130000, 纬度: 40.070000
城市名称: 长沙, 经度: 112.550000, 纬度: 28.120000
城市名称: 临汾, 经度: 111.310000, 纬度: 36.050000
城市名称: 衡阳, 经度: 112.340000, 纬度: 26.550000
城市名称: 长治, 经度: 113.130000, 纬度: 36.050000
城市名称: 湘潭, 经度: 112.510000, 纬度: 27.540000
城市名称: 呼和浩特, 经度: 111.380000, 纬度: 40.480000
城市名称: 常德, 经度: 111.390000, 纬度: 29.000000
城市名称: 包头, 经度: 110.000000, 纬度: 40.350000
城市名称: 广州, 经度: 113.160000, 纬度: 23.100000
城市名称: 海拉尔, 经度: 119.430000, 纬度: 49.140000
城市名称: 汕头, 经度: 116.400000, 纬度: 23.210000
城市名称: 沈阳, 经度: 123.230000, 纬度: 41.480000
城市名称: 韶关, 经度: 113.340000, 纬度: 24.480000
城市名称: 大连, 经度: 121.380000, 纬度: 38.540000
城市名称: 海口, 经度: 110.100000, 纬度: 20.030000
城市名称: 鞍山, 经度: 123.000000, 纬度: 41.040000
城市名称: 南宁, 经度: 108.210000, 纬度: 22.470000
城市名称: 锦州, 经度: 121.090000, 纬度: 41.090000
城市名称: 桂林, 经度: 110.000000, 纬度: 25.180000
城市名称: 长春, 经度: 125.180000, 纬度: 43.550000
城市名称: 柳州, 经度: 109.190000, 纬度: 24.200000
城市名称: 吉林, 经度: 126.36.000000, 纬度: 43.480000
```

(2)插入城市数据

选择 2，依次输入华盛顿 -77 38，再输入 1 进行打印输出：

```
请输入您要进行的操作：
2
输入新创建的城市名和经纬度：华盛顿
-77 38
请输入您要进行的操作：
1
城市名称：北京, 经度：116.280000, 纬度：39.540000
城市名称：青岛, 经度：120.190000, 纬度：36.040000
城市名称：天津, 经度：117.100000, 纬度：39.100000
城市名称：郑州, 经度：113.420000, 纬度：34.440000
城市名称：石家庄, 经度：114.260000, 纬度：38.030000
城市名称：开封, 经度：114.230000, 纬度：34.520000
城市名称：保定, 经度：115.280000, 纬度：38.530000
城市名称：洛阳, 经度：112.260000, 纬度：34.430000
城市名称：唐山, 经度：118.090000, 纬度：39.370000
城市名称：许昌, 经度：113.480000, 纬度：34.000000
城市名称：秦皇岛, 经度：119.370000, 纬度：39.540000
城市名称：新乡, 经度：113.540000, 纬度：35.180000
城市名称：张家口, 经度：114.550000, 纬度：40.510000
城市名称：武汉, 经度：114.200000, 纬度：30.370000
城市名称：承德, 经度：117.520000, 纬度：40.590000
城市名称：宜昌, 经度：111.170000, 纬度：30.420000
城市名称：太原, 经度：112.330000, 纬度：37.510000
城市名称：沙市, 经度：112.170000, 纬度：30.160000
城市名称：大同, 经度：113.130000, 纬度：40.070000
城市名称：长沙, 经度：112.550000, 纬度：28.120000
城市名称：临汾, 经度：111.310000, 纬度：36.050000
城市名称：衡阳, 经度：112.340000, 纬度：26.550000
城市名称：长治, 经度：113.130000, 纬度：36.050000
城市名称：湘潭, 经度：112.510000, 纬度：27.540000
```

```
城市名称：拉萨, 经度：91.020000, 纬度：29.390000
城市名称：宁波, 经度：121.340000, 纬度：29.530000
城市名称：日喀则, 经度：88.490000, 纬度：29.160000
城市名称：温州, 经度：120.380000, 纬度：28.000000
城市名称：西安, 经度：108.550000, 纬度：34.150000
城市名称：金华, 经度：119.490000, 纬度：29.100000
城市名称：宝鸡, 经度：107.090000, 纬度：34.210000
城市名称：合肥, 经度：117.160000, 纬度：31.510000
城市名称：延安, 经度：109.260000, 纬度：36.350000
城市名称：芜湖, 经度：118.200000, 纬度：31.210000
城市名称：兰州, 经度：103.500000, 纬度：36.030000
城市名称：安庆, 经度：117.020000, 纬度：30.320000
城市名称：天水, 经度：105.330000, 纬度：34.350000
城市名称：福州, 经度：119.190000, 纬度：26.020000
城市名称：酒泉, 经度：98.300000, 纬度：39.440000
城市名称：厦门, 经度：118.040000, 纬度：24.260000
城市名称：西宁, 经度：101.490000, 纬度：36.370000
城市名称：泉州, 经度：118.370000, 纬度：24.540000
城市名称：银川, 经度：106.130000, 纬度：38.280000
城市名称：南昌, 经度：115.530000, 纬度：28.410000
城市名称：乌鲁木齐, 经度：87.360000, 纬度：43.460000
城市名称：九江, 经度：115.590000, 纬度：29.430000
城市名称：哈密, 经度：93.270000, 纬度：42.500000
城市名称：赣州, 经度：114.560000, 纬度：25.550000
城市名称：喀什, 经度：75.590000, 纬度：39.270000
城市名称：济南, 经度：117.020000, 纬度：36.400000
城市名称：和田, 经度：79.550000, 纬度：37.070000
城市名称：烟台, 经度：121.200000, 纬度：37.330000
城市名称：台北, 经度：121.310000, 纬度：25.020000
城市名称：华盛顿, 经度：-77.000000, 纬度：38.000000
```

### (3)查找城市经纬度

选择 4，输入长沙，输出：

```

创建成功！
1. 打印当前城市集合
2. 创建一个新的城市
3. 通过坐标查询城市
4. 通过城市名查经纬度
5. 通过经纬度删除城市
请输入您要进行的操作：
4
输入查询城市的名称：长沙
查找的城市经纬度为：116. 280000, 39. 540000
请输入您要进行的操作：

```

#### (4)删除城市数据

选择 5，输入经纬度-77，38，输出：

```

城市名称： 厦门, 经度： 118. 040000, 纬度： 24. 260000
城市名称： 西宁, 经度： 101. 490000, 纬度： 36. 370000
城市名称： 泉州, 经度： 118. 370000, 纬度： 24. 540000
城市名称： 银川, 经度： 106. 130000, 纬度： 38. 280000
城市名称： 南昌, 经度： 115. 530000, 纬度： 28. 410000
城市名称： 乌鲁木齐, 经度： 87. 360000, 纬度： 43. 460000
城市名称： 九江, 经度： 115. 590000, 纬度： 29. 430000
城市名称： 哈密, 经度： 93. 270000, 纬度： 42. 500000
城市名称： 赣州, 经度： 114. 560000, 纬度： 25. 550000
城市名称： 喀什, 经度： 75. 590000, 纬度： 39. 270000
城市名称： 济南, 经度： 117. 020000, 纬度： 36. 400000
城市名称： 和田, 经度： 79. 550000, 纬度： 37. 070000
城市名称： 烟台, 经度： 121. 200000, 纬度： 37. 330000
城市名称： 台北, 经度： 121. 310000, 纬度： 25. 020000
城市名称： 华盛顿, 经度： -77. 000000, 纬度： 38. 000000
请输入您要进行的操作：
5
输入删除城市的经纬度：-77 38
删除成功！请输入您要进行的操作：
1
城市名称： 北京, 经度： 116. 280000, 纬度： 39. 540000
城市名称： 青岛, 经度： 120. 190000, 纬度： 36. 040000
城市名称： 天津, 经度： 117. 100000, 纬度： 39. 100000
城市名称： 郑州, 经度： 113. 420000, 纬度： 34. 440000
城市名称： 石家庄, 经度： 114. 260000, 纬度： 38. 030000
城市名称： 开封, 经度： 114. 230000, 纬度： 34. 520000
城市名称： 保定, 经度： 115. 280000, 纬度： 38. 530000
城市名称： 洛阳, 经度： 112. 260000, 纬度： 34. 430000
城市名称： 唐山, 经度： 118. 090000, 纬度： 39. 370000
城市名称： 许昌, 经度： 113. 480000, 纬度： 34. 000000

```

再次显示数据可以发现北京被删除，华盛顿数据已经添加并更新：

## 7. 心得体会

(1)在本实验中再次运用了逐步调试的方法进行程序分析，加深了对函数编写和链表使用的熟练度；

(2)在本实验的设计过程中再次用到了文件的读写,但对于文件的删除还是不太熟练,需要在今后实验的编写中多加练习。

# 约瑟夫环

## 1. 需求分析

本问题是求解一个有终止的递推问题， $n$  个人围成一圈，从第一个人开始报数，数到  $m$  的人出列，再由下一个人重新从 1 开始报数，数到  $m$  的人再出圈，依次类推，直到所有的人都出圈，请输出依次出圈人的编号。输入的形式和输入值的范围：程序全程输入数字，游戏人数需要大于 1，循环终止参数需要大于 0。

(1) 输出的形式：在循环链表初始化后显示循环链表的内容，然后迭代计算依次输出退出游戏的人，最后输出胜利的人。

(2) 程序所能达到的功能：程序应该具有创建、显示循环链表，可视化迭代计算的功能。

(3) 由于问题比较简单，故直接放在主函数中进行运行。

## 2. 概要设计

(1) 抽象数据类型：

ADT JoCircle{

数据对象  $D = \{a_i \mid a_i \in Elemset, i = 1, 2, \dots, n, n \geq 0\}$

数据关系  $R = \{ \langle a_{i-1}, a_i \rangle \mid a_{i-1}, a_i \in D, i = 2, \dots, n \} \cup \{ \langle a_n, a_1 \rangle \}$

基本操作

Del (&L,M)

初始条件：存活人数大于 1

操作结果：将一个结点从循环链表中删除并且输出  
被删除的结点编号

}ADT JoCircle

图 3 函数模块及关系图

### 3. 详细设计

代码如下：

```
#include <stdio.h>
#include <stdlib.h>
#include "ysf.h"

int main() {
    node *head, *end, *chosed;
    head = (node *)malloc(sizeof(node));
    end = head;
    head->num = 1;
    end->next = NULL;
    int n;
    printf("请输入总数: ");
    scanf("%d", &n);
    for (int i = 2; i <= n; i++) {
        node *c;
        c = (node *)malloc(sizeof(node));
        c->num = i;
        end->next = c;
        end = end->next;
        end->next = NULL;
    }
    end->next = head;
    chosed = head;
    int m;
    printf("请输入第一个密码: ");
    scanf("%d", &m);
```

```

printf("请依次输入每位的密码: ");

for (int i = 0; i < n; i++) {
    scanf("%d", &chose->key);
    chose = chose->next;
}

chose = head;
printf("退出的编号依次为: ");
for (int i = 0; i < n; i++) {
    int n = del(chose, m);
    m = n;
}

}}

#ifndef YSF_H
#define YSF_H

#include <stdio.h>
#include <stdlib.h>
#include "ysf.h"

int main() {
    node *head, *end, *chose;
    head = (node *)malloc(sizeof(node));
    end = head;
    head->num = 1;
    end->next = NULL;
    int n;
    printf("请输入总数: ");
    scanf("%d", &n);
    for (int i = 2; i <= n; i++) {
        node *c;
        c = (node *)malloc(sizeof(node));
        c->num = i;
        end->next = c;
        end = end->next;
        end->next = NULL;
    }
}

```

```

end->next = head;
chose = head;
int m;
printf("请输入第一个密码: ");
scanf("%d", &m);
printf("请依次输入每位的密码: ");

for (int i = 0; i < n; i++) {
    scanf("%d", &chose->key);
    chose = chose->next;
}

chose = head;
printf("退出的编号依次为: ");
for (int i = 0; i < n; i++) {
    int n = del(chose, m);
    m = n;
}

}
#endif

```

## 4. 调试分析

### (1) 分析程序代码的质量

- 正确性。在一定的数据范围内，该程序能实现所需功能，所以正确性是没有问题的。
- 健壮性。该程序能够处理输入数据非法的情况，将输入限制在合法数据内，健壮性目前没有问题。

## 5. 使用说明

.h 文件中存放了删除和输出函数，其他部分较为简单没有通过函数进行功能的分块

## 6. 测试程序的运行结果

输入 7, 20, 3, 1, 7, 2, 4, 8, 4



## 输出结果

```

请输入总数: 7
请输入第一个密码: 20
请依次输入每位的密码: 3 1 7 2 4 8 4
退出的编号依次为: 6 1 4 7 2 3 5
-----
Process exited after 13.42 seconds with return value 0
请按任意键继续. . .

```

## 7. 心得体会

循环链表在约瑟夫问题上的优势就是可以动态拆链，而数组如果要进行删除操作的话非常麻烦，而且在循环上数组也没有循环链表方便，

## 附录：源程序文件清单

各程序源代码随本实验报告电子版一起打包，存放在 `code` 子目录。

文件清单如下：

Function.h ..... 城市类定义实现及相关操作

main.cpp ..... 城市链表主函数

city.txt ..... 城市链表测试程序所需数据

LinkedList2022.c ..... 单链表主函数

LinkedList2022.h ..... 单链表功能及函数

ysf.cpp ..... 约瑟夫问题代码

ysf.h ..... 约瑟夫问题函数