

> 一研为定，万山无阻

@[toc]

第四次 直播 单链表的头插与尾插

- 使用 C++ 的引用进行读写数据

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include <stdlib.h>
4  void modify_pointer(char& p)
5  {
6      p = (char*)malloc(100); //申请空间
7      fgets(p, 100, stdin); //如果使用fgets传入的是一个指针变量，中间参数是指针指向的空间大小
8  }
9
10 int main()
11 {
12     char* p;
13     modify_pointer(p);
14     puts(p);
15     return 0;
16 }
```

CSDN @QuantumYou

顺序表的定义

顺序表

- 1、插入和删除操作移动大量元素。
- 2、数组的大小不好确定
- 3、占用一大段连续的存储空间，造成很多碎片。

单链表：逻辑上相邻的元素在物理上不相邻

单链表结点的定义：

```
typedef struct LNode{           //单链表结点类型
    ElemType data;              //数据域
    struct LNode *next;         //指针域
} LNode, *LinkList;
```

CSDN @QuantumYou

LinkList 等价于 struct LNode *

- **头指针**：链表中第一个结点的存储位置，用来标识单链表。
- **头结点**：在单链表第一个结点之前附加的一个结点，为了操作上的方便

顺序表的增删改查

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define MaxSize 50
5  typedef int ElemType;
```

```

6 //静态分配
7 typedef struct{
8     ElemType data[MaxSize];
9     int length;//当前顺序表中有多少个元素
10 }SqList;
11 //动态分配
12 #define InitSize 100
13 typedef struct{
14     ElemType *data;
15     int capacity;//动态数组的最大容量
16     int length;
17 }SeqList;
18 //i代表插入的位置，从1开始，e要插入的元素
19 bool ListInsert(SqList &L,int i,ElemType e)
20 {
21     if(i<1||i>L.length+1)//判断要插入的位置是否合法
22         return false;
23     if(L.length>=MaxSize)//超出空间了
24         return false;
25     for(int j=L.length;j>=i;j--)//移动顺序表中的元素
26         L.data[j]=L.data[j-1];
27     L.data[i-1]=e;//数组下标从零开始，插入第一个位置，访问的下标为0
28     L.length++;
29     return true;
30 }
31 //删除使用元素e的引用的目的是拿出对应的值
32 bool ListDelete(SqList &L,int i,ElemType &e)
33 {
34     if(i<1||i>L.length)//如果删除的位置是不合法
35         return false;
36     e=L.data[i-1];//获取顺序表中对应的元素，赋值给e
37     for(int j=i;j<L.length;j++)
38         L.data[j-1]=L.data[j];
39     L.length--;//删除一个元素，顺序表长度减1
40     return true;
41 }
42 //查找成功，返回位置，位置从1开始，查找失败，返回0
43 int LocateElem(SqList L,ElemType e)
44 {
45     int i;
46     for(i=0;i<L.length;i++)
47         if(L.data[i]==e)
48             return i+1;//加1就是元素在顺序表中的位置
49     return 0;
50 }
51 //打印顺序表元素
52 void PrintList(SqList &L)
53 {
54     for(int i=0;i<L.length;i++)
55     {
56         printf("%4d",L.data[i]);
57     }
58     printf("\n");
59 }
60 int main()
61 {
62     SqList L;//顺序表的名称
63     bool ret;//查看返回值，布尔型是True,或者False

```

```

64     ElemType del; //要删除的元素
65     //首先手动在顺序表中赋值
66     L.data[0]=1;
67     L.data[1]=2;
68     L.data[2]=3;
69     L.length=3; //总计三个元素
70     ret=ListInsert(L,2,60);
71     if(ret)
72     {
73         printf("插入成功\n");
74         PrintList(L);
75     }else{
76         printf("插入失败\n");
77     }
78     ret=ListDelete(L,1,del);
79     if(ret)
80     {
81         printf("删除成功\n");
82         printf("删除元素值为 %d\n",del);
83         PrintList(L);
84     }else{
85         printf("删除失败\n");
86     }
87     ret=LocateElem(L,60);
88     if(ret)
89     {
90         printf("查找成功\n");
91         printf("元素位置为 %d\n",ret);
92     }else{
93         printf("查找失败\n");
94     }
95     system("pause"); //停在控制台窗口
96 }
97

```

头插法

- 使用头插法新建链表

```

1
2  LinkList CreatList1(LinkList &L) //list_head_insert
3  {
4      LNode *s; int x;
5      L=(LinkList)malloc(sizeof(LNode)); //带头结点的链表
6      L->next=NULL; //L->data里边没放东西
7      scanf("%d",&x); //从标准输入读取数据
8      //3 4 5 6 7 9999
9      while(x!=9999){
10         s=(LNode*)malloc(sizeof(LNode)); //申请一个新空间给s, 强制类型转换
11         s->data=x; //把读取到的值, 给新空间中的data成员
12         s->next=L->next; //让新结点的next指针指向链表的第一个元素 (第一个放我们数据的
元素)
13         L->next=s; //让s作为第一个元素
14         scanf("%d",&x); //读取标准输入
15     }
16     return L;
17 }

```

尾插法

- 使用尾插法尾插法新建链表

```
1  LinkList CreatList2(LinkList &L)//list_tail_insert
2  {
3      int x;
4      L=(LinkList)malloc(sizeof(LNode));//带头节点的链表
5      LNode* s, * r = L;//LinkList s,r=L;也可以, r代表链表表尾结点, 指向链表尾部
6      //3 4 5 6 7 9999
7      scanf("%d",&x);
8      while(x!=9999){
9          s=(LNode*)malloc(sizeof(LNode));
10         s->data=x;
11         r->next=s;//让尾部结点指向新结点
12         r=s;//r指向新的表尾结点
13         scanf("%d",&x);
14     }
15     r->next=NULL;//尾结点的next指针赋值为NULL
16     return L;
17 }
```

Tips: next 指针, 没有赋值为NULL造成的

```
while(L!=NULL)
```

```
printf("%3d",L->data);//打印当前结点数据
L=L->next;//指向下一个结点
```

```
printf("\n");
```

E道C督学营》课程
线性表的链式表示
main()

```
LinkList L;//链表头, 是结构体指针类型
LinkList search;//用来存储拿到的某一个节点
//CreatList1(L); //输入数据可以为3 4 5 6 7 9999 尾插法新建链表
```

已引发异常

引发了异常: 读取访问权限冲突。
L 是 0xCDCDCDCD。

[复制详细信息](#)

异常设置

☒ 引发此异常类型时中断

从以下位置引发时除外:

☐ 2.3 线性表的链式表示.exe

[打开异常设置](#) | [编辑条件](#)

CSDN @QuantumYou

单链表的查找

- 关于 `q->next = p->next` 的理解, `->` 是指针访问成员变量 (地址) `p->next` 整体访问结构体空间里的一个成员

按序号查找

关键代码如下 (伪代码) 注意特殊情况 (边界值的考虑)

```
1  LNode *p = L->next ;
2  int j = 1 ;
3  while(p&& j<i){
4      p=p->next ;
5      j++ ;
6  }
7  return p ;
```

按值查找

关键代码如下（伪代码） 注意特殊情况（边界值的考虑）

```
1  LNode *p = L->next ;
2  while( p!= NULL && p->data != e){
3      p=p->next ;
4  }
5  return p ;
```

单链表的操作

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef int ElemType;
5  typedef struct LNode{
6      ElemType data;
7      struct LNode *next; //指向下一个结点
8  }LNode, *LinkList;
9  //头插法新建链表
10 LinkList CreatList1(LinkList &L) //list_head_insert
11 {
12     LNode *s; int x;
13     L=(LinkList)malloc(sizeof(LNode)); //带头结点的链表
14     L->next=NULL; //L->data里边没放东西
15     scanf("%d",&x); //从标准输入读取数据
16     //3 4 5 6 7 9999
17     while(x!=9999){
18         s=(LNode*)malloc(sizeof(LNode)); //申请一个新空间给s，强制类型转换
19         s->data=x; //把读取到的值，给新空间中的data成员
20         s->next=L->next; //让新结点的next指针指向链表的第一个元素（第一个放我们数据的
元素）
21         L->next=s; //让s作为第一个元素
22         scanf("%d",&x); //读取标准输入
23     }
24     return L;
25 }
26 //尾插法新建链表
27 LinkList CreatList2(LinkList &L) //list_tail_insert
28 {
29     int x;
30     L=(LinkList)malloc(sizeof(LNode)); //带头节点的链表
31     LNode* s, * r = L; //LinkList s,r=L;也可以，r代表链表表尾结点，指向链表尾部
32     //3 4 5 6 7 9999
33     scanf("%d",&x);
34     while(x!=9999){
35         s=(LNode*)malloc(sizeof(LNode));
36         s->data=x;
37         r->next=s; //让尾部结点指向新结点
38         r=s; //r指向新的表尾结点
39         scanf("%d",&x);
40     }
41     r->next=NULL; //尾结点的next指针赋值为NULL
42     return L;
43 }
```

```

44 //按序号查找结点值
45 LNode *GetElem(LinkList L,int i)
46 {
47     int j=1;
48     LNode *p=L->next;
49     if(i==0)
50         return L;
51     if(i<1)
52         return NULL;
53     while(p&& j<i)
54     {
55         p=p->next;
56         j++;
57     }
58     return p;
59 }
60 //按值查找
61 LNode *LocateElem(LinkList L,ElemType e)
62 {
63     LNode *p=L->next;
64     while(p!=NULL&&p->data!=e)
65         p=p->next;
66     return p;
67 }
68 //新结点插入第i个位置
69 bool ListFrontInsert(LinkList L,int i,ElemType e)
70 {
71     LinkList p=GetElem(L,i-1);
72     if(NULL==p)
73     {
74         return false;
75     }
76     LinkList s=(LNode*)malloc(sizeof(LNode));//为新插入的结点申请空间
77     s->data=e;
78     s->next=p->next;
79     p->next=s;
80     return true;
81 }
82 //删除第i个结点
83 bool ListDelete(LinkList L,int i)
84 {
85     LinkList p=GetElem(L,i-1);
86     if(NULL==p)
87     {
88         return false;
89     }
90     LinkList q;
91     q=p->next;
92     p->next=q->next;//断链
93     free(q);//释放对应结点的空间
94     return true;
95 }
96 //打印链表中每个结点的值
97 void PrintList(LinkList L)
98 {
99     L=L->next;
100     while(L!=NULL)//NULL是为了代表一张空的藏宝图
101     {

```

```

102         printf("%3d",L->data); //打印当前结点数据
103         L=L->next; //指向下一个结点
104     }
105     printf("\n");
106 }
107
108 //2.3 线性表的链式表示
109 int main()
110 {
111     LinkList L; //链表头，是结构体指针类型
112     LinkList search; //用来存储拿到的某一个节点
113     //CreatList1(L); //输入数据可以为3 4 5 6 7 9999,头插法新建链表
114     CreatList2(L); //输入数据可以为3 4 5 6 7 9999
115     PrintList(L); //链表打印
116     //search=GetElem(L,2);
117     //if(search!=NULL)
118     //{
119     //    printf("按序号查找成功\n");
120     //    printf("%d\n",search->data);
121     //}
122     //search=LocateElem(L,6); //按值查询
123     //if(search!=NULL)
124     //{
125     //    printf("按值查找成功\n");
126     //    printf("%d\n",search->data);
127     //}
128     //ListFrontInsert(L,2,99); //新结点插入第i个位置
129     //PrintList(L);
130     //ListDelete(L,4); //删除第4个结点
131     //PrintList(L);
132 }

```