

东莞城市学院人工智能学院

School of Artificial Intelligence

《数据结构》

实验报告

| 学号 |  |
| --- | --- |
| 姓名 |  |
| 专业、班级 | 2024级计科（专升本）A班 |
| 课程名称 | 《数据结构》 |
| 指导教师 | 李言一 |
| 学期 | 2024—2025第1学期 |
| 实验次序 | 实验一 |
| 上机时间 | 2024-9-25 周三 第5-6节 |
| 上机地点 |  |
| 分数 |  |

二○二三——二○二四学年 第一学期

实验一 顺序表的定义和运算

# 实验课时

课内：2课时

# 实验类型

操作性实验

# 实验目的

通过本次实验，掌握顺序表的创建、插入和删除等基本操作算法。

# 实验内容和步骤

## 一、源代码

（1）顺序表结构的定义及初始化顺序表的源码

定义部分、函数部分和主函数生效部分：

#include<stdlib.h>

#include<stdio.h>

#include<conio.h>

typedef int ListItem;

typedef ListItem\* addr;

typedef struct alist\* List;

void ItemShow(ListItem x)

{

printf("%d \n", x);

}

typedef struct alist

{

int n, curr;

int maxsize;

ListItem\* table;

}Alist;

int ListLength(List L) //函数为侧表长//

{

return L->n;

}

ListInsert(int k, ListItem x, List L) //插入元素函数，函数将输入 位置、元素内容、插入的表//

{

int i;

if (k<0 || k>L->n) //若插入的位置不对，将结束程序

{

return;

}

for (i = L->n - 1; i >= k; i--) //设置一个 数i，i=表的最大值-1（为了使 数组最后一位的下一位产生，并使最后一位移至新出现的最后一位）

{ //也就是做到了从原本k位的位置全部后移一位） 问题：最大值已经给出，意味着有了限制，那么超出最大限制的内存空

// 没有重新用Malloc函数分配，哪来的空间？

L->table[i + 1] = L->table[i]; //当原本k的位置也移到k+1后

}

L->table[k] = x; //把腾出来的空间给要插入的元素

L->n++; //

}

void printList(List L) //输出函数//

{

int i;

for (i = 0; i < L->n; i++)

{

ItemShow(L->table[i]); //若没到最后一位，输出，并下一位

}

}

List ListInit(int size)

{

List L = (List)malloc(sizeof \* L);

L->table = (ListItem\*)malloc(size \* sizeof(ListItem));

L->n = 0;

return L;

}

main()

{

int t, u, o, p;

char tt;

List a;

printf("程序开始.....\n");

printf("建立链表:\n");

printf("输入链表大小:（初步定义是5）");

scanf\_s("%d", &t);

a = ListInit(t);

printf("已成功建立，测试表长：\n");

u = ListLength(a);

printf("%d\n", u);

printf("测试结束，正确结果为0，现进行填充元素:\n请输入填充内容(依次为1～n):");

ListInsert(0, 50, a);

printf("尝试进行首次插入，检测情况：\n");

printList(a);

tt = getch();

}

（2）顺序表插入数据的源码：

#include<stdlib.h>

#include<stdio.h>

#include<conio.h>

typedef int ListItem;

typedef ListItem\* addr;

typedef struct alist\* List;

void ItemShow(ListItem x)

{

printf("%d \n", x);

}

typedef struct alist

{

int n, curr; //n 为表长（实际使用长度），curr为表当前位置）//

int maxsize; // 表的最大长度//

ListItem\* table; //定义为该表类型的指针类型，作用是建立起一个指向数组的指针

}Alist;

List ListInit(int size) //表的初始化，会从终端接收一个关于表的最大值的数值//

{

List L = (List)malloc(sizeof \* L);

L->table = (ListItem\*)malloc(size \* sizeof(ListItem)); /\*给table指针分指向一个 该数据结构类型（定义的）的类型的 由malloc函数分配的 表最大值× 类型结 构的大小的连续的空间 （起到了数组的作用->数据类型的×总数的大小的数组）\*/

L->n = 0; //初始时实际表长为0//

return L; //返回一个 名为L 实为 List 类型的指针/

}

int ListEmpty(List L) /\*该函数为测试表是否为空 但此处仅适用于刚开口，若表已使用该改为if L->n ！=0，return 1 \*/

{

return L->n == 0;

}

ListInsert(int k, ListItem x, List L)

{

int i;

if (k<0 || k>L->n)

{

return;

}

for (i = L->n - 1; i >= k; i--)

{

L->table[i + 1] = L->table[i];

}

L->table[k] = x;

L->n++;

}

void printList(List L) //输出函数//

{

int i;

for (i = 0; i < L->n; i++)

{

ItemShow(L->table[i]); //若没到最后一位，输出，并下一位

}

}

int ListLength(List L) //函数为侧表长//

{

return L->n;

}

main()

{

int t, u, o, p;

char tt;

List a;

printf("程序开始.....\n");

printf("建立链表:\n");

printf("输入链表大小:");

scanf\_s("%d", &t);

a = ListInit(t);

printf("已成功建立，测试表长：\n");

u = ListLength(a);

printf("%d\n", u);

printf("测试结束，正确结果为0，现进行填充元素:\n请输入填充内容(依次为1～n):");

ListInsert(0, 50, a);

printf("尝试进行首次插入，检测情况：\n");

printList(a);

printf("继续插入中...\n");

ListInsert(1, 1143, a); /\*顺序连续插入\*/

ListInsert(2, 501, a);

ListInsert(3, 502, a);

ListInsert(4, 9822, a);

ListInsert(5, 334, a);

printf("检测插入情况：\n");

printList(a);

printf("检测表长:");

o = ListLength(a);

printf("%d\n", o);

tt = getch();

}

（3）顺序表删除数据的源码：

#include<stdlib.h>

#include<stdio.h>

#include<conio.h>

typedef int ListItem;

typedef ListItem\* addr;

typedef struct alist\* List;

void ItemShow(ListItem x)

{

printf("%d \n", x);

}

typedef struct alist

{

int n, curr;

int maxsize;

ListItem\* table;

}Alist;

List ListInit(int size) //表的初始化，会从终端接收一个关于表的最大值的数值//

{

List L = (List)malloc(sizeof \* L);

L->table = (ListItem\*)malloc(size \* sizeof(ListItem));

L->n = 0;

return L; //返回一个 名为L 实为 List 类型的指针/

}

int ListEmpty(List L) /\*该函数为测试表是否为空 但此处仅适用于刚开口，若表已使用该改为if L->n ！=0，return 1 \*/

{

return L->n == 0;

}

int ListLength(List L) //函数为侧表长//

{

return L->n;

}

ListItem ListRetrieve(int k, List L) //反向锁定，给出位置找元素

{

if (k<1 || k>L->n) //进行对比，若给的维持不存在，返回0 结束程序//

{

return 0;

}

return L->table[k - 1]; //若找到了，记得因为 数组数字与现实数字的差1，要+1再返回内容

}

void printList(List L) //输出函数//

{

int i;

for(i=0;i<L->n;i++)

{

ItemShow(L->table[i]); //若没到最后一位，输出，并下一位

}

}

int ListLocate(ListItem x, List L) //锁定函数，锁定某元素的位置 该函数带入参数为表名称 与 某元素

{

int i; //设置一个单位，用于对表长单位对比

for (i = 0; i < L->n; i++) //在该单位暂<=表总长时，增加

{

if (L->table[i] == x) //若在某一位置的元素==需要锁定的元素，记录下该位置（i）//

{

return ++i; //因为元素位置在数组中是从0开始，所以要++1后再返回

}

}

return 0;

}

ListInsert(int k, ListItem x, List L)

{

int i;

if (k<0 || k>L->n)

{

return;

}

for (i = L->n - 1; i >= k; i--)

{

L->table[i + 1] = L->table[i];

}

L->table[k] = x;

L->n++;

}

ListItem ListDelete(int k, List L)

{

int i;

ListItem x;

if (k<1 || k>L->n)

{

return 0;

}

x = L->table[k - 1];

for (i = k; i < L->n; i++)

{

L->table[i - 1] = L->table[i];

}

L->n--;

return x;

}

main()

{

int t, u, o, p;

char tt;

List a;

printf("程序开始.....\n");

printf("建立链表:\n");

printf("输入链表大小:");

scanf\_s("%d", &t);

a = ListInit(t);

printf("已成功建立，测试表长：\n");

u = ListLength(a);

printf("%d\n", u);

printf("测试结束，正确结果为0，现进行填充元素:\n请输入填充内容(依次为1～n):");

ListInsert(0, 50, a);

printf("尝试进行首次插入，检测情况：\n");

printList(a);

printf("继续插入中...\n");

ListInsert(1, 1143, a);

ListInsert(2, 501, a);

ListInsert(3, 502, a);

ListInsert(4, 9822, a);

ListInsert(5, 334, a);

printf("检测插入情况：\n");

printList(a);

printf("检测表长:");

o = ListLength(a);

printf("%d\n", o);

printf("尝试测试表示是否为空");

p = ListEmpty(a);

printf("%d\n", p);

printf("调用第4位的元素：");

p = ListRetrieve(4, a);

printf("%d\n", p);

printf("求1143在元素表中的位置：");

p = ListLocate(1143, a);

printf("%d\n删除后两位元素：", p);

p = ListDelete(3, a);

printf("删除的元素是：%d\n再输出：\n", p);

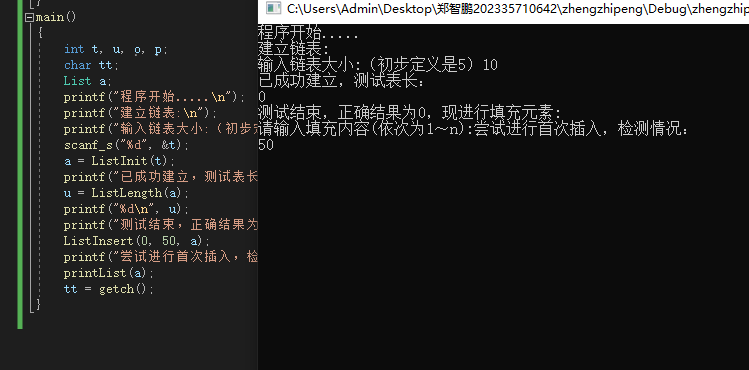
printList(a);

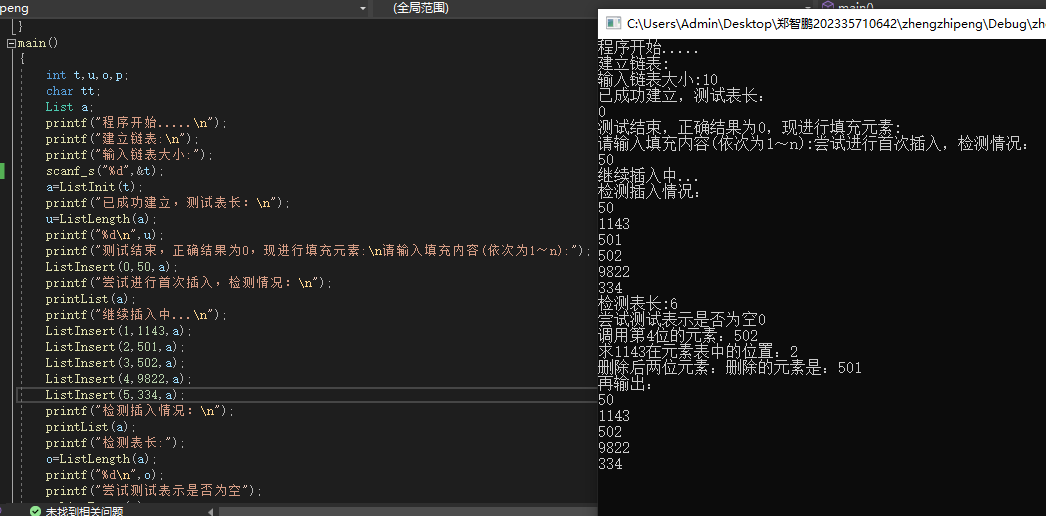
tt = getch();

return 0;

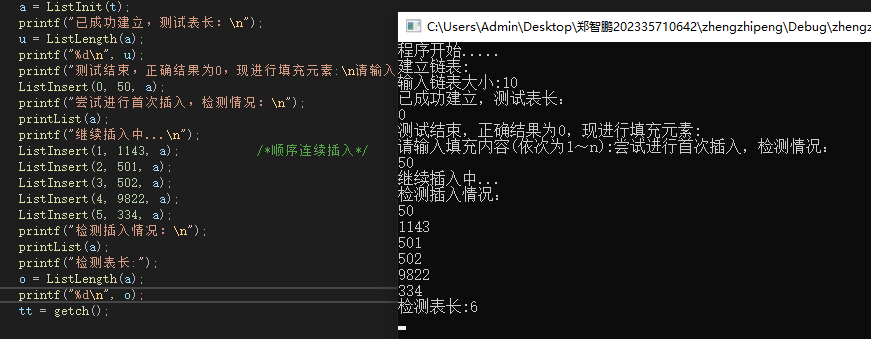
## }二、程序运行结果截图

（1）顺序表结构的定义及初始化顺序表的截图：

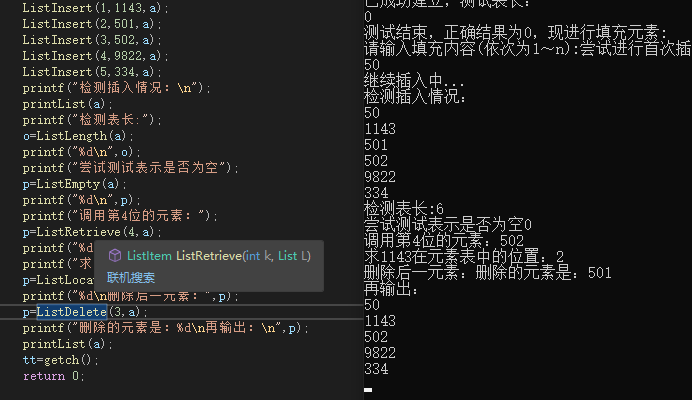




（2）顺序表插入数据的截图：



1. 顺序表删除数据的截图：



## 三、实验总结

数据结构是一门理论结合实践的一门课，光是学习理论是不足以完全领会结构内在的关系的，继而使学习过程产生迷茫与空洞，作为一门实践性与理论都非常重要而学科，我在上机过程中解决了很多我在书上看不懂和不理解的东西，我深刻的认识到上机操作的重要性，如果不上机，学习的理论也只是限制在理论上，没办法与实践相结合，在这次上机实践中，特别是顺序表的查排增删等运算，定义、建立等运算也让我吃了不少苦头，也加了很多注释来帮助自己观看和理解，中途也出现过申请空间时代码申请错误：L->table=(ListItem \*)malloc(sizeof(ListItem));不过仍然正常运行的问现象，不过这个不能因为正常运行就视而不见，还是要吧问题给解决：L->table=(ListItem \*)malloc(size\*sizeof(ListItem));

实践课上实操的实践给我带来了很多的书本上看不到的东西，所以我仍然期待下一次的实践课