

**计算机与信息 学院实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验课程： | 数据结构 | | | | |
| 实验编号： | 实验四 | | | | |
| 实验名称： | 单链表 | | | | |
| 实验人员： | 学号 | 18111303044 | | | |
| 姓名 | 邵一波 | | | |
| 班级 | 2018级计算机类一班 | | | |
| 实验日期： | 2019-10-10 | | | | |
| 实验室： | 六号楼2060302 | | | | |
|  |  | | | | |
| 实验评价： |  | | | | |
| 实验成绩： | |  | 评价日期： |  |
|  | 指导教师： | |  | | |

|  |
| --- |
| **注：具体内容可根据专业特点和实验性质略作调整，页面不够可附页。**   1. **需求分析** 2. 输入的形式和输入值的范围 3. 输入int整数0 – 12用于在菜单中选择要测试的功能 4. 功能3需要另外输入要插入的元素数量（int），并依次输入具体的元素（int） 5. 功能4 需要另外输入要插入的元素数量（char），并依次输入具体的元素（char） 6. 功能7需要输入元素的位置（int，小于等于表长） 7. 功能8需要输入具体的元素值（int） 8. 功能9需要依次输入要插入的位置（int，数值应小于等于表长）以及具体元素值（int） 9. 功能10需要输入要删除的元素的位置（int，小于等于表长） 10. 输出的形式 11. 功能1初始化单链表L      1. 功能2判断链表L是否为空      1. 功能3尾插法插入元素        1. 功能4头插法插入元素        1. 功能5输出链表L      1. 功能6输出链表L的长度      1. 功能7输出链表L某个位置的元素      1. 功能8输出某个元素的位置        1. 功能9在某个位置上插入元素      1. 功能10删除链表L某个位置的元素、      1. 功能11销毁链表L      1. 功能12将单链表L按数据元素值由小到大排序      1. 输入”0”退出程序     (3) 程序所能达到的功能   1. 初始化单链表L 2. 依次采用尾插法插入a,b,c,d,e元素 3. 输出链表L 4. 输出L的长度 5. 判断L是否为空 6. 输出链表L的第三个元素 7. 输出元素d的位置 8. 在第4个位置上插入元素f 9. 输出链表L 10. 删除链表的第3个元素 11. 输出链表L 12. 销毁链表L 13. 头插法建立单链表L，其中L的数据元素值依次为12,56,7,3,89,21,123,20,45,34 14. 输出单链表L 15. 将单链表L按数据元素值由小到大排序 16. 输出排序后新单链表L 17. 销毁单链表L 18. 测试数据。 19. 1 20. 5   a  b  c  d  e   1. 5 2. 6 3. 2 4. 7   3   1. 8   d   1. 9   4  f   1. 5 2. 10   3   1. 5 2. 11 3. 1   4  10  12  56  7  3  89  21  123  20  45  34   1. 5 2. 12 3. 5 4. 11 5. 0 6. **概要设计**   (1)抽象数据类型的定义  **CommonDef.h**  #define ERROR NULL  #define false 0  #define true 1  typedef int ElementType;  typedef struct LNode \* PtrToLNode;  typedef PtrToLNode Position;  typedef PtrToLNode List;  struct LNode  {  ElementType Data;  PtrToLNode Next;  };  (2函数的定义  **LinklistDef.h**  /\* 初始化 \*/  List makeEmpty();  /\* 判定是否为空表 \*/  int isEmpty( List L );  /\* 按位置序号查找 \*/  int Find\_L( List L, int i, ElementType \* X);  /\* 按元素值查找 \*/  int LocateElem( List L,ElementType X);  /\* 带头结点的插入 \*/  int Insert( List L, ElementType X, Position P );  /\* 按位置序号插入 \*/  int ListInsert\_L( List L,int i,ElementType X);  /\* 头插法依次插入 \*/  int Insert\_L( List L );  /\* 尾插法依次插入 \*/  int Insert\_Tail( List L );  /\* 带头结点的删除 \*/  int Delete( List L, Position P );  /\* 按位置序号删除 \*/  int ListDelete(List L,int i,int \* e);  /\* 带头结点的打印 \*/  void Print\_L( List L );  /\* 带头结点的长度 \*/  int Length\_L( List L );  /\* 销毁带头节点的链表 \*/  int DestoryList\_L( List \* L );  /\* 带头结点的排序 \*/  int Sort\_L( List L );  int ifInt;  (3)主程序的流程图     1. **详细设计**   **LinkListTestApp.c**  #include "LinklistDef.h"  //用于展示功能菜单  void showHelp();  int main()  {  int locate;  ElementType X;  List L;  int ifExit = 0;  int selector;  showHelp();  while(!ifExit)  {  printf("\n请输入您的选择(以0退出): ");  scanf("%d",&selector);  system("cls");  showHelp();  switch(selector)  {  case 0: ifExit = 1;system("cls");break;  case 1: L = makeEmpty();break;  case 2:  if(isEmpty(L))  printf("此表为空.\n");  else  printf("此表非空.\n");  break;  case 3:  if(Insert\_Tail(L))  {  system("cls");  showHelp();  printf("尾插完成.\n");  }  break;  case 4:  if(Insert\_L(L))  {  system("cls");  showHelp();  printf("头插完成.\n");  }  break;  case 5:  Print\_L(L);break;  case 6:  printf("长度: %d\n",Length\_L(L));break;  case 7:  printf("你想输出第几个元素?\n");  scanf("%d",&locate);  if(Find\_L(L,locate,&X))  printf("第%d个元素是 %c.\n",locate,X);  break;  case 8:  printf("你想输出哪个元素的位置?\n");  getchar();  scanf("%c",&X);  locate = LocateElem(L,X);  if(locate != 0)  printf("元素%c第一次出现的位置是 %d.\n",X,locate);  else  printf("未找到!\n");  break;  case 9:  printf("你想在哪个位置插入元素? ");  scanf("%d",&locate);  getchar();  printf("请输入你要插入的元素: ");  scanf("%c",&X);  if(ListInsert\_L(L,locate,X))  printf("插入成功!\n");  else  printf("插入失败!\n");  break;  case 10:  printf("你想删除哪个位置的元素? ");  scanf("%d",&locate);  if(ListDelete(L,locate,&X))  printf("元素 %c 删除成功!\n",X);  else  printf("删除失败!\n");  break;  case 11:  if(DestoryList\_L(&L))  printf("已销毁!\n");  break;  case 12:  if(Sort\_L(L))  printf("排序成功!\n");  break;  default: printf("您的输入有误，请重新输入0到12的数字!\n");  }  }  return 0;  }  void showHelp()  {  printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");  printf("\t\t\t 功能菜单\n");  printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");  printf("1.初始化单链表L\t\t\t2.判断链表L是否为空\n");  printf("3.尾插法插入元素\t\t4.头插法插入元素\n");  printf("5.输出链表L\t\t\t6.输出链表L的长度\n");  printf("7.输出链表L某个位置的元素\t8.输出某个元素的位置\n");  printf("9.在某个位置上插入元素\t\t10.删除链表L某个位置的元素\n");  printf("11.销毁链表L\t\t\t12.将单链表L按数据元素值由小到大排序\n");  printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");  }  **LinklistApp.c**  #include "LinklistDef.h"  /\* 初始化 \*/  List makeEmpty()  {  Position L = (Position)malloc(sizeof(struct LNode));  L->Next = NULL;  printf("初始化完成!\n");  return L;  }  /\* 判定是否为空表 \*/  int isEmpty( List L )  {  if(L->Next)  return false;  else  return true;  }  /\* 按位置序号查找 \*/  int Find\_L( List L, int i, ElementType \* X)  {  int j;  //寻找第i个结点  for(j = 0;L && j != i;j++)  L = L->Next;  if(j != i)  return false;  else  {  \*X = L->Data;  }  return true;  }  /\* 按元素值查找 \*/  int LocateElem( List L,ElementType X)  { //找到返回其位置，未找到返回0;  int i;  L = L->Next;  for(i = 1; L && L->Data != X ;i++, L = L->Next);  if(!L)  return false;  else  return i;  }  /\* 带头结点的插入 \*/  int Insert( List L, ElementType X, Position P )  { /\* 这里默认L有头结点 \*/  Position tmp, pre;    /\* 查找P的前一个结点 \*/  for ( pre=L; pre&&pre->Next!=P; pre=pre->Next );  if ( pre==NULL ) { /\* P所指的结点不在L中 \*/  printf("插入位置参数错误\n");  return false;  }  else { /\* 找到了P的前一个结点pre \*/  /\* 在P前插入新结点 \*/  tmp = (Position)malloc(sizeof(struct LNode)); /\* 申请、填装结点 \*/  tmp->Data = X;  tmp->Next = P;  pre->Next = tmp;  return true;  }  }  /\* 按位置序号插入 \*/  int ListInsert\_L( List L,int i,ElementType X)  {  Position p=L;  int j;  //寻找第i个结点  for(j = 0;p && j != i;j++)  p=p->Next;  if(j != i)  return false;  return Insert(L,X,p);  }  /\* 头插法依次插入 \*/  int Insert\_L( List L )  {  int temp;  Position p;  int i,n;  ifInt = 1;  printf("你想插入多少元素? ");  scanf("%d",&n);  for(i=0;i<n;i++)  {  p = (Position)malloc(sizeof(struct LNode));  printf("元素 %d: ",i+1);  scanf("%d",&temp);  p->Data = (char)temp;  p->Next = L->Next;//p指向下一个  L->Next = p;//再将p给单链表L的表头  }  return true;  }  /\* 尾插法依次插入 \*/  int Insert\_Tail( List L )  {  int i,n;  Position p;  while(L->Next)  L = L->Next;  ifInt = 0;  printf("你想插入多少元素? ");  scanf("%d",&n);  for(i=0;i<n;i++)  {  getchar();  p = (Position)malloc(sizeof(struct LNode));  printf("元素 %d: ",i+1);  scanf("%c",&(p->Data));  L->Next = p;  L = p;  }  L->Next = NULL;  return true;  }    /\* 带头结点的删除 \*/  int Delete( List L, Position P )  { /\* 这里默认L有头结点 \*/  Position pre;    /\* 查找P的前一个结点 \*/  for ( pre=L; pre&&pre->Next!=P; pre=pre->Next ) ;  if ( pre==NULL || P==NULL) { /\* P所指的结点不在L中 \*/  printf("删除位置参数错误\n");  return false;  }  else { /\* 找到了P的前一个结点pre \*/  /\* 将P位置的结点删除 \*/  pre->Next = P->Next;  free(P);  return true;  }  }  /\* 按位置序号删除 \*/  int ListDelete(List L,int i,ElementType \* e)  {  Position p=L;  int j;  if(i > Length\_L(L) || i < 0)  return false;  //寻找第i个结点  for(j = 0;j != i;j++)  p=p->Next;  \*e = p->Data;  return Delete(L,p);  }  /\* 带头结点的打印 \*/  void Print\_L( List L )  {  L = L->Next;  while(L)  {  if(!ifInt)  printf("%c ",L->Data);  else  printf("%d ",(int)L->Data);  L = L->Next;  }  printf("\n");  }  /\* 带头结点的长度 \*/  int Length\_L( List L )  {  int length;  L = L->Next;  for(length = 0;L;length++)  L = L->Next;  return length;  }  /\* 销毁带头节点的链表 \*/  int DestoryList\_L( List \* L )  { //连同头结点一并销毁  Position pre = (\*L);  Position p = (\*L)->Next;  for(;p;p = p->Next)  {  free(pre);  pre = p;  }  free(pre);  \*L = NULL;  return true;  }  /\* 带头结点的排序 \*/  int Sort\_L( List L )  {  Position head, prep1, p1, prep2, p2, premin, min, temp;  if((L->Next == NULL) || (L->Next->Next == NULL))  return false;  head = L;  for(prep1 = head, p1 = prep1->Next; p1->Next != NULL; prep1 = prep1->Next, p1 = prep1->Next)  {  //保存最小节点  premin = prep1;  min = p1;  for(prep2 = p1, p2 = prep2->Next; p2 != NULL; prep2 = prep2->Next, p2 = prep2->Next)  {  if(min->Data > p2->Data)  {  premin = prep2;  min = p2;  }  }  if(p1 != min)  {  if(p1->Next == min)  {  temp = min->Next;  prep1->Next = min;  min->Next = p1;  p1->Next = temp;  }else{  temp = min->Next;  prep1->Next = min;  min->Next = p1->Next;  premin->Next = p1;  p1->Next = temp;  }  }  }  return true;  }   1. **调试分析** 2. 调试过程中遇到的问题是如何解决的以及对设计与实现的回顾讨论和分析    1. 题目中的排序较为复杂，参考了数组中的选择排序进行实现 3. ①若要通过函数改变某一变量的值，需要使用指针变量，并在调用是使用变量地址 4. **测试数据与结果** |