|  |  |  |
| --- | --- | --- |
| **软件学院实验报告** | | |
| 姓名： 胡逢彬 学号： 2225060803 专业： 网络工程 年级： 2022级 | | |
| 课程名称 | | 数据结构 |
| 实验名称 | | 实验2 顺序表的实现 |
| 实验的准备阶段 | 实验内容 | （1）实验目的  通过该实验，深入理解顺序表的逻辑结构、物理结构等概念，掌握顺序表基本操作的编程实现，注意顺序表插入、删除等操作过程中数据元素的移动现象，培养学生编写程序时，要考虑程序的健壮性，全面考虑问题，熟练掌握通过函数参数返回函数结果的办法。  （2）实验内容  编程实现顺序表下教材第二章定义的线性表的基本操作，并根据已经实现的基本操作（函数），通过调用函数，实现两个非递减有序的线性表的合并，***注意，***合并时，如果有重复的元素(一个表内部的重复元素和两个表之间的重复元素)，请保留一个。  （3）实验要求  （a）求前驱是指，输入一个元素值（而不是位置），求该元素在顺序表中的直接前驱元素值。求后继是指：输入一个元素值（而不是位置），求该元素在顺序表中的直接后继元素值；（b）为了方便修改数据元素的类型，请使用类型重定义，可以方便修改线性表中的数据元素的类型；（c）大部分函数的返回结果应是函数执行是否成功的一种状态，执行成功了，才返回具体的结果值；（d）对每个功能进行测试时，要求把不合法的情况也测试一下。具体见下面的测试用例；（e）采用菜单形式对应各个操作，使其编成一个完整的小软件，参考界面如下。***注意：***程序运行过程中菜单不要做成刷屏的效果，测试过的数据不要清除，这样方便截图和查看。    注：  销毁是指free(L.elem); L.elem=NULL; L.length=0; L.listsize=0; return TRUE。  清空是指：L.length=0 ;return TRUE。  （4）验收/测试用例  通过菜单调用各个操作，测试点：   * 没有初始化前进行其他操作，程序是否能控制住；即，如果没有初始化线性表，其他的功能是无法正常进行的，如果选择进行其他操作，要提示先进行初始化； * 先选择菜单1，初始化一个顺序表（初始化顺序表，是指初始化一个空的线性表，里面的元素个数是0）； * 选择菜单10，插入数据(位置， 数据)，要测插入位置不合法的情况如：（0,1）、（2,1），正确插入3个数据（1，20）、（1，10）、（3,30）； * 显示顺序表中的数据，屏幕输出10， 20， 30； * 判空，屏幕输出顺序表非空； * 输出顺序表长度，屏幕输出3； * 获取指定位置元素，要测指定位置在【1，3】范围之外的情况和之内的情况都要测试，非法的情况要做出合理的提示； * 定位，输入：40， 输出：不存在，输入20，输出位置为2； * 求直接前驱，要测求第一个元素的前驱、不存在顺序表中的元素的直接前驱，其他元素的直接前驱；输入10，输出：第一个元素没有前驱，输入20，输出前驱是10，输入40，输出该元素不存在； * 求直接后继，要测最后一个元素的后继、不存在顺序表中的元素的直接后继，其他元素的直接后继；同上求前驱； * 删除，要测位置在【1，3】范围之外的情况和之内的情况，非法的情况要做出合理的提示； * 清空操作后再测长度，判断是否为空；清空后，测试菜单6到11的功能，看是否能够正确提示。 * 销毁顺序表，销毁线性表之后还能不能做插入，删除等操作，如果选其他操作，就要提示线性表已经销毁不存在； * 测试合并操作，第一个线性表中的元素是（2,3,3,4，5），第二个线性表中的内容是（1,4,5，6,6,7），合并后的结果，请输出。 |
| 实验类型 | 验证性 |
| 实验的重点、难点 | 重点： 顺序表的定义和实现  难点： 两个非递减有序表的去重合并 |
| 实验环境 | VC++6.0 |
| 实验的实施阶段 | 实验步骤及完成任务情况 | 一、设计思想  （注意：请同学们看完要求，把本段红色文字删除！！！）  写出整体程序框架是怎么安排（菜单结构的设计）的，另外分别写出上述13个函数的设计思想和思路方法。  二、主要源代码  （注意：请同学们看完要求，把本段红色文字删除！！！）  如果是非原创  可以适当修改：变量名，菜单提示语  要多加注释。  如果是参考的其他同学的代码，请一定要多增加注释，表示自己看懂了。  (5分)  #include<iostream>  #include<stdio.h>  #include<stdlib.h>  using namespace std;  # define LIST\_INIT\_SIZE 100 //符号常量 // 线性表存储空间的初始分配量  # define LISTINCREMENT 10  # define OK 1  # define ERROR 0  # define NO 2  typedef int Status ;  typedef int ElemType;  typedef struct {  ElemType\*elem;  int length;  int listsize;  } SqList;  SqList L;  int flag;//是否初始化的标志  //初始化  Status InitList(SqList &L) {  //构造一个空的线性表Ｌ  L.elem=(ElemType\* )malloc(LIST\_INIT\_SIZE\*sizeof(ElemType)); //为数据元素开辟一维数组空间  if(!L.elem)return ERROR;//存储分配失败  L.length=0; // 空表长度为０  L.listsize=LIST\_INIT\_SIZE; //初始存储容量  return OK;  }  //插入  Status ListInsert(SqList &L,int i,ElemType e) {  if(i<1||i>L.length+1)return ERROR;  if(L.length>=L.listsize) {  ElemType\*newbase=(ElemType\*)realloc(L.elem,(L.listsize+LISTINCREMENT)\*sizeof(ElemType));  if(!newbase)return ERROR;  L.elem=newbase;  L.listsize+=LISTINCREMENT;  }  ElemType \*q;  q=&L.elem[i-1];  for(ElemType\*p=&(L.elem[L.length-1]); p>=q; --p)  \*(p+1)=\*p;  \*q=e;  ++L.length;  return OK;  }  //删除  Status DeleteElem(SqList &L,int i) {  if(i<1||i>L.length)  return ERROR;  ElemType\*q=&(L.elem[L.length-1]);  for(ElemType\*p=&(L.elem[i-1]); p<q; p++)  \*p=\*(p+1);  L.length--;  return OK;  }  //销毁  Status DestrotyList(SqList &L) {  free(L.elem);  L.elem=0;  L.length=0;  L.listsize=0;  cout<<"销毁成功"<<endl;  return OK;  }  //清空  Status EmptyList(SqList &L) {  L.length=0;  cout<<"表清空成功"<<endl;  return OK;  }  //判空  bool JudgeEmptyList(SqList &L) {  if(L.length==0) {  cout<<"表为空"<<endl;  return true;  } else {  cout<<"表不为空"<<endl;  return false;  }  }  //求长度  int ListLength(SqList L) {  return L.length;  }  //显示  Status ShowList(SqList &L) {  for(int i=0; i<L.length; i++)  printf("%d ",L.elem[i]);  cout<<endl;  return OK;  }  //获取指定位置元素  Status ElemGet(SqList L,int i) {  if(i<1||i>L.length) {  cout<<"输入位置不合法"<<endl ;  return ERROR;  } else {  cout<<"该位置元素为："<<L.elem[i-1]<<endl;  return OK;  }  }  //获取元素位置  Status LocateGet(SqList L,ElemType e,int &Loc) {  for(int i=0; i<L.length; i++) {  if(L.elem[i]==e) {  Loc=i+1;  return OK;  }  }  return ERROR;  }  //求前驱  Status PriorElem(SqList L,ElemType e,ElemType &pir) {  for(int i=0; i<L.length; i++) {  if(L.elem[i]==e) {  if(i==0)return ERROR;  else {  pir=L.elem[i-1];  return OK;  }  }  }  return NO;  }  //求后继  Status NextElem(SqList L,ElemType e,ElemType &next) {  for(int i=0; i<L.length; i++) {  if(L.elem[i]==e) {  if(i==L.length-1)return ERROR;  else {  next=L.elem[i+1];  return OK;  }  }  }  return NO;  }  //合并  Status ListMerge(SqList La,SqList Lb,SqList &Lc) {  ElemType \*pa=&La.elem[0];  ElemType \*pb=&Lb.elem[0];  ElemType \*pc=&Lc.elem[0];  ElemType \*qa=&(La.elem[La.length-1]);  ElemType \*qb=&(Lb.elem[Lb.length-1]);  while (pa <= qa && pb <= qb) { //合并线性表A、B。  if (\*pa <= \*pb) {  \*pc = \*pa;  pa++;  pc++;  } else {  \*pc = \*pb;  pb++;  pc++;  }  Lc.length++;  }  while(pa<=qa) {  \*pc=\*pa;  pa++;  pc++;  Lc.length++;  }  while(pb<=qb) {  \*pc=\*pb;  pb++;  pc++;  Lc.length++;  }  for(int i=0; i<Lc.length-1; i++) {  for(int j=i+1; j<Lc.length; j++) {  if(Lc.elem[i]==Lc.elem[j]) {  DeleteElem(Lc,j);  j--;  }  }  }  return OK;  }  void menu() {  cout<<" ====顺序表的实现===="<<endl;  cout<<" --输入数字来实现相应的功能-- "<<endl;  cout<<"\*\*\*\*\*\*\*\*1--- 初始化 \*\*\*\*\*\*\*\*"<<endl;  cout<<"\*\*\*\*\*\*\*\*2--- 销毁线性表 \*\*\*\*\*\*\*\*"<<endl;  cout<<"\*\*\*\*\*\*\*\*3--- 清空线性表 \*\*\*\*\*\*\*\*"<<endl;  cout<<"\*\*\*\*\*\*\*\*4--- 判空 \*\*\*\*\*\*\*\*"<<endl;  cout<<"\*\*\*\*\*\*\*\*5--- 求线性表长度 \*\*\*\*\*\*\*\*"<<endl;  cout<<"\*\*\*\*\*\*\*\*6--- 获取指定位置的元素 \*\*\*\*\*\*\*\*"<<endl;  cout<<"\*\*\*\*\*\*\*\*7--- 获取元素位置 \*\*\*\*\*\*\*\*"<<endl;  cout<<"\*\*\*\*\*\*\*\*8--- 求前驱 \*\*\*\*\*\*\*\*"<<endl;  cout<<"\*\*\*\*\*\*\*\*9--- 求后继 \*\*\*\*\*\*\*\*"<<endl;  cout<<"\*\*\*\*\*\*\*\*10-- 插入 \*\*\*\*\*\*\*\*"<<endl;  cout<<"\*\*\*\*\*\*\*\*11-- 删除 \*\*\*\*\*\*\*\*"<<endl;  cout<<"\*\*\*\*\*\*\*\*12-- 显示线性表 \*\*\*\*\*\*\*\*"<<endl;  cout<<"\*\*\*\*\*\*\*\*13-- 合并两个非递减有序的线性表 \*\*\*\*\*\*\*\*"<<endl;  cout<<"\*\*\*\*\*\*\*\* 输入一个负数退出程序 \*\*\*\*\*\*\*\*"<<endl;  }  int main() {  int n=1;  while(n) {  menu();  int s;  cout << "请输入指令编号（1--13），（退出时输入负数）：\n";  cin>>s;  switch(s) {  case 1:  InitList(L);  if(InitList(L)==OK) {  cout<<"初始化成功"<<endl;  flag=1;  } else  cout<<"内存分配失败，请重新操作"<<endl;  break;  case 2:  if(flag==1) {  DestrotyList(L);  flag=2;  } else  cout<<"你尚未初始化一个表"<<endl;  break;  case 3:  if(flag==1) {  EmptyList(L);  } else  cout<<"你尚未初始化一个表"<<endl;  break;  case 4:  if(flag==1) {  (JudgeEmptyList(L));  } else  cout<<"你尚未初始化一个表"<<endl;  break;  case 5:  if(flag==1) {  cout<<"表长为:"<<ListLength(L)<<endl;  } else  cout<<"你尚未初始化一个表"<<endl;  break;  case 6:  if(flag==1) {  int i;  cout<<"请输入元素的位置：";  cin>>i;  ElemGet(L,i);  } else  cout<<"你尚未初始化一个表"<<endl;  break;  case 7:  if(flag==1) {  ElemType e;  int Loc=0;  cout<<"请输入元素：";  cin>>e;  if(LocateGet(L,e,Loc)==OK) {  cout<<"该元素的位置为："<<Loc<<endl;  } else  cout<<"该元素不存在"<<endl;  } else  cout<<"你尚未初始化一个表"<<endl;  break;  case 8:  if(flag==1) {  ElemType e,pir;  cout<<"请输入元素:";  cin>>e;  if(PriorElem(L,e,pir)==OK)  cout<<"该元素的前驱为："<<pir<<endl;  if(PriorElem(L,e,pir)==ERROR)  cout<<"第一个元素不存在前驱"<<endl;  if(PriorElem(L,e,pir)==NO)  cout<<"该元素不存在"<<endl;  } else  cout<<"你尚未初始化一个表"<<endl;  break;  case 9:  if(flag==1) {  ElemType e,next;  cout<<"请输入元素：";  cin>>e;  if(NextElem(L,e,next)==OK)  cout<<"该元素的后继为："<<next<<endl;  if(NextElem(L,e,next)==ERROR)  cout<<"最后一个元素不存在后继"<<endl;  if(NextElem(L,e,next)==NO)  cout<<"该元素不存在"<<endl;  } else  cout<<"你尚未初始化一个表"<<endl;  break;  case 10:  if(flag==1) {  int i;  ElemType e;  cout<<"请输入插入的位置和元素:";  cin>>i>>e;  // cin>>i;  // cout<<"请输入插入的数据:";  // cin>>e;  if(ListInsert(L,i,e)==OK)  cout<<"插入成功"<<endl;  else  cout<<"插入位置不合法"<<endl;  } else  cout<<"你尚未初始化一个表"<<endl;  break;  case 11:  if(flag==1) {  int i;  cout<<"请输入删除的元素位置:";  cin>>i;  if(DeleteElem(L,i)==OK)  cout<<"删除成功"<<endl;  else  cout<<"输入位置不合法"<<endl;  } else  cout<<"你尚未初始化一个表"<<endl;  break;  case 12:  if(flag==1) {  cout<<"线性表中的元素为："<<endl;  ShowList(L);  } else  cout<<"你尚未初始化一个表"<<endl;  break;  case 13:  SqList La,Lb,Lc;  InitList(La);  InitList(Lb);  InitList(Lc);  int a;  cout<<"请输入线性表A中的元素"<<endl;  while(cin>>a) {  La.elem[La.length]=a;  La.length++;  if(getchar()=='\n')break;  }  int b;  cout<<"请输入线性表B中的元素"<<endl;  while(cin>>b) {  Lb.elem[Lb.length]=b;  Lb.length++;  if(getchar()=='\n')break;  }  ListMerge(La,Lb,Lc);  cout<<"合并后的线性表C为："<<endl;  ShowList(Lc);  break;  default:  if(s<0) {  n = 0;  cout<<"程序退出成功，欢迎下次使用~~"<<endl;  break;  } else  cout<<"您输入的指令有误，请重新输入~"<<endl;  }  system("pause");  system("cls");  }  } |
| 实验结果的处理阶段 | 实验结果 | (3分)  IMG_256  IMG_256  IMG_256  IMG_256IMG_256  IMG_256  IMG_256  IMG_256 |
| 实验结果总结 | 问题1：许多输出语句堆积，页面不简洁，于是增加system("cls")  导致原本想要的输出语句被清空  解决方法：通过增加语句  system("pause");  先按任意键再清空 |