软件工程课程设计（II）实验报告

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 姓名 | 林镕琛 | 学号 | | 181491117 |
| 班级 | 软件1班 | 时间 | | 2020-05-23 |
| 题目 | 顺序表的操作 | | | |
| 实验成绩 |  | 指导教师 | 杜岳峰 | |
| 实验内容：   1. 实验程序：   /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  @Author：林镕琛  @Date： 2020-05-23  @Description：   顺序表的操作   给定数据：2，15，33，11，17，16，223，128  （1）创建顺序表  （2）顺序表进行顺序查找  （3）顺序表进行排序  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    #include <iostream>  #include<stdlib.h>  using namespace std;    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  @Function：定义结构体  @Parm：nodeElem 结点内容  @Parm：next 结点指针  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  typedef struct listNode{   int nodeElem;   struct listNode \*next;  };    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  @Function：通过尾插法创建顺序表  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  listNode \*listCreate(){   int elem;   listNode \*nodeHead, \*tail, \*node;   nodeHead = (listNode \*)malloc(sizeof(listNode));   tail = nodeHead;   cout<<"请输入数据,以0结束：";   cin>>elem;   while (elem != 0){   node = (listNode \*)malloc(sizeof(listNode));   node->nodeElem = elem;   tail->next = node;   tail = node ;   cin>>elem;   }   tail->next = NULL;   return nodeHead;  }    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  @Function：打印显示顺序表  @Parm: nodeHead 链表头指针  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void listPrint(listNode \*nodeHead){   listNode \* nodePoint = nodeHead->next; // 结点指针指向第一个有效的结点   cout<<"显示数据：";   while (nodePoint){   cout<<nodePoint->nodeElem<<" ";   nodePoint = nodePoint->next;   }   cout<<endl;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  @Function：结点顺序查找  @Parm：Elem 需要查找的结点内容  @Parm：nodeHead 头结点指针  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void nodeFind(listNode \*nodeHead,int elem){   listNode \* nodePoint= nodeHead->next; // 结点指针指向第一个有效的结点   while(nodePoint && (nodePoint->nodeElem != elem)){   nodePoint = nodePoint->next;   }   if (nodePoint != NULL){ // 判断结点指针是否指向末尾   cout<<"查找结果："<<elem<<"在链表中"<<endl;   }else{   cout<<"查找结果："<<elem<<"不在链表中"<<endl;   }  }    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  @Function：选择符进行链表排序  @Parm：nodeHead 头结点指针  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void listSelectSort(listNode \*nodeHead){   listNode \*prePoint = nodeHead->next; // 前一个结点指针   listNode \*aftPoint ; // 后一个结点指针   while (prePoint){   aftPoint = prePoint->next;   while (aftPoint){   if(aftPoint->nodeElem < prePoint->nodeElem){ // 比较结点值大小   int temp = aftPoint->nodeElem;   aftPoint->nodeElem = prePoint->nodeElem;   prePoint->nodeElem = temp;   }   aftPoint = aftPoint->next; // 后一个结点指针后移   }   prePoint = prePoint->next; // 前一个结点指针后移   }  }    int main()  {   listNode \*nodeHead; // 创建头结点   nodeHead = listCreate(); // 创建链表   cout<<"排序前，";   listPrint(nodeHead); // 显示排序之前链表   listSelectSort(nodeHead); // 链表排序   cout<<"排序后，";   listPrint(nodeHead); // 显示排序之后链表   int findNode;   cout<<"请输入需要查找的结点：";   cin>>findNode;   nodeFind(nodeHead,findNode);   return 0;  }   1. 实验运行结果： | | | | |