**数据结构课程实验报告**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **学生姓名** | |  | **学 号** |  |
| **班 级** | | **计算机科学与技术** | **指导老师** |  |
| **实验名称** | | **线性表的链式存储结构** | **实验成绩** |  |
| **实验报告** | | | | |
| 实  验  概  述 | 实验目的：掌握线性表的链式存储结构及基本操作，深入了解顺序表的基本特性。 | | | |
| 实验要求：  （1）建立一个评委打分的单向链表。  （2）显示删除相关结点后的链表信息。  （3）显示要求的结果。 | | | |
| 实  验  内  容 | 实验设计思路、步骤和方法等：  （1）评委信息结点用结构变量存储，包含三个成员项，即姓名、年龄、评分。结构类型定义如下：  //定义评委信息  struct pw  {  char name[8]; //姓名  short age; //年龄  float score; //评分  };  （2）用头插法或尾插法建立带头结点的单链表，本实验采用尾插法。  （3）遍历链表并逐次比较求最高分和最低分。  （4）在链表中物理删除，即实际删除最高分和最低分结点；也可以进行逻辑删除，即在被删结点的数据域设置一个删除标记，本实验采用物理删除的方法。  （5）遍历链表，累加求和，计算总分及平均分，并输出相关信息。 | | | |
| 实验过程（实验中涉及的记录、数据、分析）：  //实验1.1 线性表的链接存储结构  #include  #include  #include  #define PWRS 5 //定义评委人数  //定义评委信息  struct pw  {  char name[8]; //姓名  short age; //年龄  float score; //评分  };  typedef struct pw PW;  //定义链表结点  struct node  {  PW data;  struct node \* next;  };  typedef struct node NODE;  NODE \*create(int n); //建立单链表  void input(NODE \*s,int i); //输入第i个评委信息  void output(NODE \*s); //输出评委信息  void traverse(NODE \*head); //遍历链表  void calc(NODE \*head); //计算及数据处理  void main()  {  NODE \*head=NULL;  head=create(PWRS); //建立评委信息单链表  printf("\n所有评委的评分信息如下:\n");  traverse(head); //输出所有评委的评分信息  calc(head); //计算成绩  printf("该参赛者去掉一个最高分和一个最低分后的有效评委的评分信息如下:\n");  traverse(head); //输出有效评委的评分信息  }  //尾插法建立带头结点的单链表  NODE \*create(int n)  {  NODE \*head,\*p,\*q;  int i;  p=(NODE\*)malloc(sizeof(NODE));  head=p; q=p; p->next=NULL;  for(i=1; i<=n; i++)  {  p=(NODE\*)malloc(sizeof(NODE));  input(p,i);  p->next=NULL;  q->next=p;  q=p;  }  return (head);  }  //输入评委信息，包括姓名、年龄和评分  void input(NODE \*s,int i)  {  printf("请输入第 %d 个评委的姓名、年龄和评分：",i);  scanf("%s%d%f",&s->data.name,&s->data.age,&s->data.score);  }  //输出评委信息  void output(NODE \*s)  {  printf("评委姓名：%6s 年龄：%d 评分：%6.2f\n",s->data.name,s->data.age,s->data.score);  }  //遍历链表，输出所有评委的评分信息  void traverse(NODE \*head)  {  NODE \*p=head->next; //指向第一个结点  while(p!=NULL)  {  output(p);  p=p->next;  }  printf("\n");  }  //输出最高分及最低分评委信息，删除最高分及最低分结点并计算参赛者的最后平均分  void calc(NODE \*head)  {  NODE \*q,\*p,\*pmin,\*pmax;  float sum=0; //总分  float ave=0; //平均分  //查找最高分和最低分并计算总分  p=head->next;  pmin=pmax=p;  while(p!=NULL)  {  sum+=p->data.score;  if(p->data.score>pmax->data.score) pmax=p; //pmax指向最高分结点  if(p->data.scoredata.score) pmin=p; //pmin指向最低分结点  p=p->next;  }  //输出最高分及最低分评委信息  printf("给出最高分的评委姓名：%6s 年龄：%d 评分：%6.2f\n",pmax->data.name,pmax->data.age,pmax->data.score);  printf("给出最低分的评委姓名：%6s 年龄：%d 评分：%6.2f\n",pmin->data.name,pmin->data.age,pmin->data.score);  printf("\n");  //去掉一个最高分和一个最低分，计算并输出参赛者的最后平均分  sum-=pmax->data.score;  sum-=pmin->data.score;  ave=sum/(PWRS-2);  printf("该参赛者去掉一个最高分和一个最低分后的平均得分为：%6.2f\n",ave);  printf("\n");  //在链表中删除最高分和最低分结点  for(q=head,p=head->next;p!=NULL;q=p,p=p->next)  {  if(p==pmin) { q->next=p->next; p=q; } //删除最低分结点  if(p==pmax) { q->next=p->next; p=q; } //删除最高分结点  }  } | | | |
| 实验结果：IMG_256 | | | |
| 实  验  小  结 | 实验的心得体会：   1. 线性表采用链式存储（链表）时，用结构变量存储结点，动态生成结点，用指针链接结点，能有效利用存储空间，插入删除方便。 2. （2）链表不能随机访问，是顺序访问方式，可从某结点访问到其后继结点，通常对单链表的遍历即从表头结点顺序访问到表尾结点，任何在链表上做的查找运算都是在遍历的基础上进行的。   （3）单链表操作的关键步骤包括：  1）建立链表的头插法：指针变量 p 开辟单元，生成结点，指针变量 q 始终指向头结点；操作为：p->next=q->next; q->next=p;  2）建立链表的尾插法：指针变量 p 开辟单元，生成结点，指针变量 q 始终指向尾结点；操作为： q->next=p; q=p;  3）插入：p结点的后面插入新结点s；操作为：s->next=p->next; p->next=s;  4）删除：p，q 指向相邻结点，q结点是p结点的后继，删除q结点；操作为：p->next=q->next;  5）遍历：p指向后继结点；操作为：p=p->next; | | | |
| 实验思考：线性表的应用方式以及应用前景如何，我们要深入学习。 | | | |
| 指  导  教  师  评  语 | 指导教师 日期 | | | |