### 实验1 单链表

|  |  |
| --- | --- |
| **成绩** |  |

**专业班级 信息173班 学号 201711010228姓名 苏 栋 报告日期**

**实验类型**：●验证性实验 ○综合性实验 ○设计性实验

1. **实验目的或任务**

通过指导学生上机实践，对常用数据结构的基本概念及其不同的实现方法的理论得到进一步的掌握，并对在不同存储结构上实现不同的运算方式和技巧有所体会。

1. **实验教学基本要求**

1.了解实验目的及实验原理；

2.编写程序，并附上程序代码和结果图；

3.总结在编程过程中遇到的问题、解决办法和收获。

1. **实验教学的内容或要求**

1.编写函数,实现随机产生或键盘输入一组元素,建立一个带头结点的单链表（无序）

2.编写函数,实现遍历单链表

3.编写函数,实现把单向链表中元素逆置

4.编写函数,建立一个非递减有序单链表

5.编写函数,利用以上算法，建立两个非递减有序单链表,然后合并成一个非递减链表。

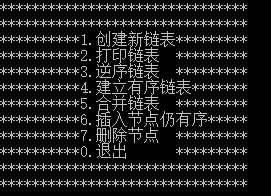
6.编写函数,在非递减有序单链表中插入一个元素使链表仍然有序

7.编写函数,实现在非递减有序链表中删除值为x的结点

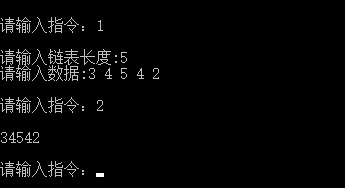
8.编写一个主函数,在主函数中设计一个简单的菜单,分别调试上述算法

**四、实验总结**

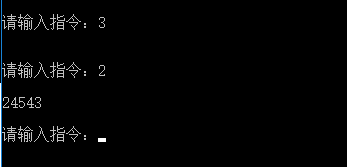
1. 编写函数,实现随机产生或键盘输入一组元素,建立一个带头结点的单链表（无序）



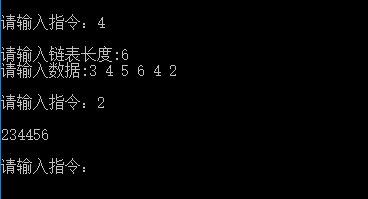
1. 编写函数,实现遍历单链表



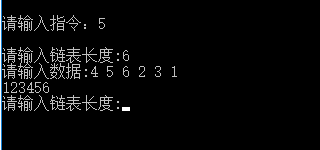
1. 编写函数,实现把单向链表中元素逆置



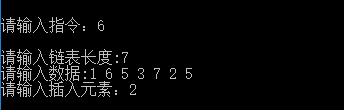
1. 编写函数,建立一个非递减有序单链表



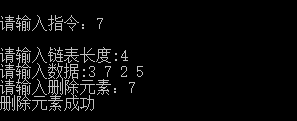
1. 编写函数,利用以上算法，建立两个非递减有序单链表,然后合并成一个非递减链表。



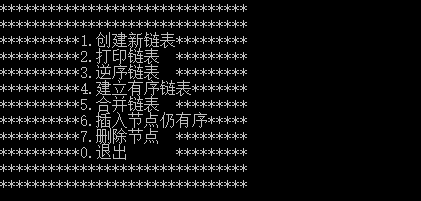
1. 编写函数,在非递减有序单链表中插入一个元素使链表仍然有序



1. 编写函数,实现在非递减有序链表中删除值为x的结点



8.编写一个主函数,在主函数中设计一个简单的菜单,分别调试上述算法



**实验总结：**

通过在对单链表的建立、查找、插入和删除等操作中，熟悉了 C 语言的相关用法，掌握结构体指针的相关用法，在这个过程中，了解到数据结构靠上课学习理论知识是远远不够的，必须实践，上机。

**实验代码：**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef int Datatype;

typedef struct Node

{

Datatype data;

struct Node\*Next;

}Node,\*ListNode;

Node\*NewLinkList()

{

Node\*head;

Node\*temp;

Node\*ptr;

int c;

int count;

head=(Node\*)malloc(sizeof(Node));

head->Next=NULL;

ptr=head;

printf("请输入链表长度:");

scanf("%d",&count);

printf("请输入数据:");

while(count--)

{

scanf("%d",&c);

temp=(Node\*)malloc(sizeof(Node));

if(!temp)

{perror("use malloc");

exit(1);

}

else

{

temp->data=c;

temp->Next=NULL;

ptr->Next=temp;

ptr=ptr->Next;

}

}

return head;

}

void Reverse(ListNode list) //逆序

{

Node\*p=list->Next;

list->Next=NULL;

while(p!=NULL)

{

Node\*q=p->Next;

p->Next=list->Next;

list->Next=p;

p=q;

}

}

Node\*OrderList() //创建有序链表

{

Node\*head;

Node\*ptr;

Node\*cur;

Node\*pcur;

head=NewLinkList();

ptr=head->Next;

while(ptr!=NULL)

{

cur=head->Next;

ptr=ptr->Next;

pcur=ptr;

while(pcur!=NULL)

{

if(cur->data>cur->Next->data)

{

Datatype temp=cur->data;

cur->data=cur->Next->data;

cur->Next->data=temp;

}

cur=cur->Next;

pcur=pcur->Next;

}

}

return head;

}

Node\*Merge(ListNode head1,ListNode head2) //合并两个有序链表

{

Node\*newhead;

Node\*p1=head1->Next;

Node\*p2=head2->Next;

Node\*ptr;

newhead=head1;

ptr=newhead;

while((p1!=NULL)&&(p2!=NULL))

{

if(p1->data<p2->data)

{

ptr->Next=p1;

p1=p1->Next;

}

else

{

ptr->Next=p2;

p2=p2->Next;

}

ptr=ptr->Next;

}

if(p1==NULL)

{

ptr->Next=p2;

}

else

{

ptr->Next=p1;

}

free(head2);

return newhead;

}

void insertOrderList(ListNode head, Datatype data)

{

Node\*temp;

Node\*cur=head->Next;

Node\*ptr=head->Next;

temp=(Node\*)malloc(sizeof(Node));

if(!temp)

{

perror("use malloc");

exit(1);

}

else

{

temp->data=data;

temp->Next=NULL;

}

while(ptr!=NULL)

{

if(data<=ptr->data)

{

temp->Next=cur->Next;

cur->Next=temp;

break;

}

cur=ptr;

ptr=ptr->Next;

}

if(NULL==temp->Next)

{

cur->Next=temp;

}

}

void Delete(ListNode head,Datatype k)//删除有序链表中值为k的节点

{

Node\*cur;

Node\*ptr;

cur=head->Next;

ptr=head->Next;

int flag=0;

while(ptr!=NULL)

{

if(k==ptr->data)

{

cur->Next=ptr->Next;

free(ptr);

ptr=NULL;

flag=1;

break;

}

cur=ptr;

ptr=ptr->Next;

}

if(flag==0)

{

printf("删除元素不存在\n");

}

else

{

printf("删除元素成功\n");

}

}

void Display(Node\*list)

{

Node\*ret=list->Next;

while(ret)

{

printf("%d",ret->data);

ret=ret->Next;

}

printf("\n");

}

void menu()

{

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*1.创建新链表 2.打印链表 \*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*3.逆序链表 4.建立有序链表\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*5.合并链表 6.插入节点仍有序\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*7.删除节点 0.退出\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

int main()

{

menu();

int input=0;

Node\*head=NULL;

Node\*head1;

Node\*head2;

do{

printf("\n");

printf("请输入指令：");

scanf("%d",&input);

printf("\n");

switch(input)

{

case 1:head=NewLinkList();break;

case 2:Display(head);break;

case 3:Reverse(head);break;

case 4:head=OrderList();break;

case 5:

head1=OrderList();

Display(head1);

head2=OrderList();

Display(head2);

head=Merge(head1,head2);

break;

case 6:

head=OrderList();

int data;

printf("请输入插入元素：");

scanf("%d",&data);

insertOrderList(head,data);

break;

case 7:

head=NewLinkList();

int d;

printf("请输入删除元素：");

scanf("%d",&d);

Delete(head,d);

break;

case 0:

break;

default:

printf("您的输入有误，请重新输入：");

}

}while(input);

system("pause");

return 0;

}