

**数据结构实验报告**

**实验报告题目：** **顺序表基本算法的实现**

**学生姓名**：刘凌杰

**专 业： 软件工程**

**学 号**： 201720180411

**指导教师： 邹 国 华**

**2018年 10 月 17 日**

**一. 实验内容：**

* **链表的初始化**
* **输入数据**
* **输出数据**
* **链表的插入操作**
* **链表的删除操作**
* **链表的查找操作**
* **链表的合并**
* **链表的节点换值**
* **链表的数据排序**
* **获取链表的长度**
* **链表的倒置**
* **链表的销毁**

**二. 实验算法实现：**

**链表节点**

typedef int DataType;

typedef struct Node

{

DataType data;

struct Node \*next;

}Node;

**类链表**

class Chain

{

private:

Node \*head;

public:

Chain()

{

head=NULL;

}

~Chain()

{

}

int getsize(); //获取链表长度

bool insert(int pos,DataType data); //插入链表

Node\* getptr(int pos); //获取指定位置的元素地址

int getplace(DataType d); //查找元素位置

void print(); //显示数据

bool earse(int pos); //删除链表

bool reverse(); //链表倒置

void Union(Chain d); //两个链表连接

void destroylist(); //销毁链表

bool sort(); //排序

bool change(int pos,DataType data); //换值

};

1.获取链表长度

//获取链表长度

int Chain::getsize()

{

int n=0;

Node \*p=head;

while(p)

{

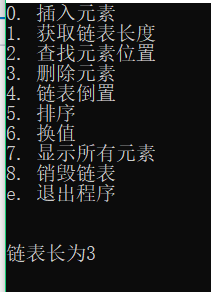
p=p->next;

n++;

}

return n;

}



2. 获取指定位置的元素地址

//获取指定位置的元素地址

Node\* Chain::getptr(int pos)

{

Node \*p=head;

if(p==NULL||pos<=0||pos>getsize())

{

return NULL;

}

for(int i=1;i<pos;i++)

{

p=p->next;

}

return p;

}

3.查找元素位置

//查找元素位置

int Chain::getplace(DataType d)

{

if(head==NULL)

{

cout<<"空链表！"<<endl;

return 0;

}

int n=1;

Node \*p=head;

while(p)

{

if(p->data==d)

return n;

p=p->next;

n++;

}

if(n==getsize()+1)

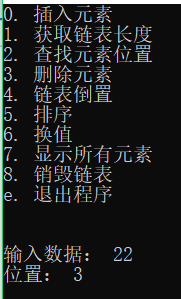
{

cout<<"无该数据！"<<endl;

return 0;

}

}



4. 插入链表

//插入链表

bool Chain::insert(int pos,DataType data)

{

if(pos>=1&&pos<=getsize()+1) //判断是否越界

{

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

p->data=data;

p->next=NULL;

if(pos==1||!head) //插入在最前面时

{

p->next=head;

head=p;

return true;

}

Node \*q=getptr(pos-1);//获取前一个元素的指针

Node \*r=q->next;

q->next=p;//前一个元素的指针成员指向第pos元素

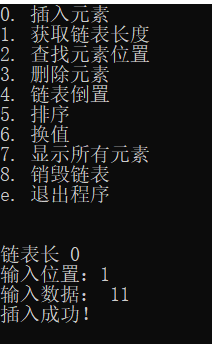
p->next=r;

return true;

}

return false;

}



5．删除链表.

//删除链表

bool Chain::earse(int pos)

{

if(pos<0||pos>getsize())

return false;

Node \*p=head;

if(pos==1) //删除第一个元素的情况

{

head=(head)->next;

free(p);

p=NULL;

return true;

}

p=getptr(pos-1); //指向第一元素

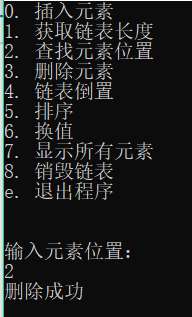
Node \*q=p->next->next; //指向第三元素

free(p->next); //指向第二元素

p->next=q; //重点（该边一个值要通过指向这个值来改变）

return true;

}



6.链表倒置

//链表倒置

bool Chain::reverse()

{

if((!head)||(!(head)->next))

return false;

Node \*p=head;

Node \*q=p->next;

Node \*r=q->next;

p->next=NULL;

q->next=p;

while(r)

{

p=q;

q=r;

r=q->next;

q->next=p;

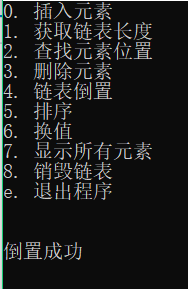
}

q->next=p;

head=q;

return true;

}



7.合并两个链表

//两个链表连接

void Chain::Union(Chain d)

{

if(!head)

{

head=d.head;

}

else

{

Node \*p=head;

while(p->next)

{

p=p->next;

}

p->next=d.head;

}

}

8.销毁链表

//销毁链表

void Chain::destroylist()//不用指针的指针会发生错误

{

Node \*p=head;

while(p)

{

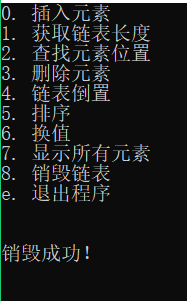
head=head->next;

free(p);

p=head;

}

}



9.排序

//排序

bool Chain::sort()

{

if(!head||!head->next) //冒泡排序法(从大到小）

return false;

Node\* p=head;

DataType a;

int n=getsize()-1;

for(int i=1;i<=n;i++)

{

Node\* q=p;

for(int j=i;j<=n;j++)

{

if(q->data<q->next->data)

{

a=q->data;

q->data=q->next->data;

q->next->data=a;

}

q=q->next;

}

}

/\*

while(p->next) //选择排序法(从小到大）

{

Node\* q=p->next;

while(q)

{

if((p->data)>(q->data))

{

a=p->data;

p->data=q->data;

q->data=a;

}

q=q->next;

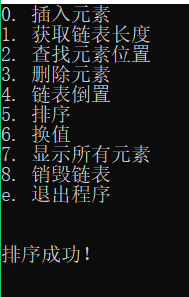
}

p=p->next;

}\*/

return true;

}



10. 指定节点换值

//换值

bool Chain::change(int pos,DataType data)

{

if(pos<0||pos>getsize()||head==NULL)

return false;

Node \*p=head;

for(int i=1;i<=pos-1;i++)

{

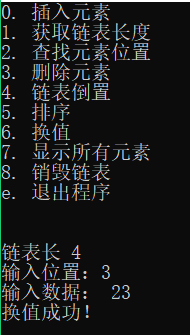
p=p->next;

}

p->data=data;

return true;

}



11.输出

//显示数据

void Chain::print()

{

if(!head)

{

cout<<"空链表！"<<endl;

return ;

}

Node \*p=head;

while(p)

{

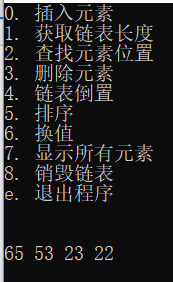
cout<<p->data<<" ";

p=p->next;

}

cout<<endl;

}



12.main函数

int main()

{

char ch;

Chain c;

int i;

DataType data;

while(1)

{

cout<<"0. 插入元素"<<endl;

cout<<"1. 获取链表长度"<<endl;

cout<<"2. 查找元素位置"<<endl;

cout<<"3. 删除元素"<<endl;

cout<<"4. 链表倒置"<<endl;

cout<<"5. 排序"<<endl;

cout<<"6. 换值"<<endl;

cout<<"7. 显示所有元素"<<endl;

cout<<"8. 销毁链表"<<endl;

cout<<"e. 退出程序"<<endl<<endl<<endl;

ch=getch();

switch(ch)

{

case '0':

cout<<"链表长 "<<c.getsize()<<endl;

cout<<"输入位置：";

cin>>i;

cout<<"输入数据： ";

cin>>data;

if(c.insert(i,data))

cout<<"插入成功！"<<endl;

else

cout<<"插入失败！输入位置不存在！"<<endl;

getch();

system("cls");

break;

case'1':

cout<<"链表长为"<<c.getsize ()<<endl;

getch();

system("cls");

break;

case'2':

cout<<"输入数据： ";

cin>>data;

cout<<"位置： "<<c.getplace(data)<<endl;

getch();

system("cls");

break;

case'3':

cout<<"输入元素位置： "<<endl;

cin>>i;

if(c.earse (i))

cout<<"删除成功"<<endl;

else

cout<<"删除失败！输入位置不存在！"<<endl;

getch();

system("cls");

break;

case'4':

if(c.reverse())

cout<<"倒置成功!"<<endl;

else

cout<<"空链表 或 只有一个元素！"<<endl;

getch();

system("cls");

break;

case'5':

if(c.sort())

cout<<"排序成功！"<<endl;

else

cout<<"空链表 或 只有一个元素！"<<endl;

getch();

system("cls");

break;

case'6':

cout<<"链表长 "<<c.getsize()<<endl;

cout<<"输入位置：";

cin>>i;

cout<<"输入数据： ";

cin>>data;

if(c.change(i,data))

cout<<"换值成功！"<<endl;

else

cout<<"位置不存在 或 空链表！"<<endl;

getch();

system("cls");

break;

case'7':

c.print ();

getch();

system("cls");

break;

case'8':

c.destroylist ();

cout<<"销毁成功！"<<endl;

getch();

system("cls");

break;

case 'e':

exit(0);

break;

default:

system("cls");

cout<<"输入序号无效！"<<endl;

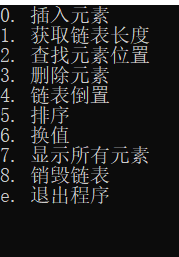
}

}

system("pause");

return 0;

}



**三．实验总结：**

经过一段时间的学习，链表算法的实现基本掌握了，就是代码打得还不太熟练，还有待提高，有些功能不太熟要问同学或查资料，尤其在做插入、删除运算时链表表的长度忘记改变。以后在这方面我会多花点时间，让自己更加熟练。