学号 WA2214014 专业 人工智能 姓名 杨跃浙
实验日期 5 月 20 号 教师签字 成绩

实验报告

【实验名称】线性表

【实验目的】

了解并掌握线性表中顺序表的初始化、插入、删除和链表的初始化、插入、删除和反转。

【实验原理】

初始化一个数据元素为整型的顺序表，通过控制台 scanf 函数将每个节点赋值，并输出该顺序表每个结点的值并对顺序表插入和删除结点

初始化一个数据元素为整型的链表，通过控制台 scanf 函数将每个节点赋值，并输出该顺序表每个结点的值并对链表插入和删除节点最终反转一个链表的结点

【实验内容】

初始化一个数据元素为整型的顺序表，通过控制台 scanf 函数将每个节点赋值，并输出该顺序表每个结点的值

对顺序表插入和删除结点

初始化一个数据元素为整型的链表，通过控制台 scanf 函数将每个节点赋值，并输出该顺序表每个结点的值

对链表插入和删除节点

反转一个链表的结点

```
#include <iostream>
using namespace std;
#define OK 1
#define ERROR 0
#define OVERFLOW -2
#define MAXSIZE 100
typedef int Status;
typedef int ElemType;
typedef struct
{
	ElemType* elem;
	int length;
} SqList;
typedef struct LNode
{
	ElemType data;
	struct LNode* next;
}LNode, * LinkList;
Status Init_SqList(SqList& L)
{
	L.elem = new ElemType[MAXSIZE];
	if (!L.elem) exit(OVERFLOW);
	L.length = 0;
	return OK;
}
Status Insert_SqList(SqList& L, int i, ElemType e)
{
	if (i<1 || i>L.length) return ERROR;
	if (L.length == MAXSIZE) return OVERFLOW;
	for (int j = L.length - 1; j >= i - 1; j--)
		L.elem[j + 1] = L.elem[j];
	L.elem[i - 1] = e;
	L.length++;
	return OK;
}
Status Delete_SqList(SqList& L, int i, ElemType& e)
{
	if (i<1 || i>L.length) return ERROR;
	for (int j = i; j <= L.length - 1; j++)
		L.elem[j - 1] = L.elem[j];
	L.length--;
	return OK;
```

```
}
Status Init_LinkList(LinkList& L)
{
    L = new LNode;
    if (!L) return ERROR;
    L->next = NULL;
    return OK;
}
void CreateList_R(LinkList& L, int n)
{
    L = new LNode;
    L->next = NULL;
    LNode* r = L;
    for (int i = 0; i < n; i++)
    {
        LNode* p = new LNode;
        cin >> p->data;
        p->next = NULL;
        r->next = p;
        r = p;
    }
}
Status Print_LinkList(LinkList& L)
{
    LNode* p = L->next;
    while (p)
    {
        cout << p->data << "\t";
        p = p->next;
    }
    cout << endl;
    return OK;
}
Status Print_SqList(SqList L)
{
    for (int i = 1; i <= L.length; i++)
        cout << L.elem[i - 1]<<"\t";
    cout << endl;
    return OK;
}
Status Insert_LinkList(LinkList L, int i, ElemType e)
{
    LNode* p = L;
    int j = 0;
    while (p && (j < i - 1))
```

```
    {
        p = p->next;
        j++;
    }
    if (!p || j > i - 1) return ERROR;
    LNode* s = new LNode;
    s->data = e;
    s->next = p->next;
    p->next = s;
    return OK;
}
Status CreateSqList(SqList &L,int n)
{
    for (int i=0; i < n; i++)
    {
        cin >> L.elem[i];
        L.length++;
    }
    return OK;
}
Status Delete_LinkList(LinkList L, int i)
{
    LNode* p = L;
    int j = 0;
    while (p && (j < i - 1))
    {
        p = p->next;
        j++;
    }
    if (!p || j > i - 1) return ERROR;
    LNode* q = p->next;
    p->next = q->next;
    delete q;
    return OK;
}
Status Overturn(LinkList L)
{
    LinkList p = L->next;
    L->next = NULL;
    while (p)
    {
        LNode* r = p;
        p = p->next;
        r->next = L->next;
        L->next = r;
```

```
        }
        return OK;
}
int main()
{
        int n,num;
        SqList Sq;
        LinkList Link;
        Init_SqList(Sq);
        CreateSqList(Sq,5);
        Print_SqList(Sq);
        Insert_SqList(Sq, 3, 10);
        Print_SqList(Sq);
        Delete_SqList(Sq, 3, num);
        Print_SqList(Sq);
        Init_LinkList(Link);
        CreateList_R(Link, 5);
        Print_LinkList(Link);
        Insert_LinkList(Link, 3, 10);
        Print_LinkList(Link);
        Overturn(Link);
        Print_LinkList(Link);
        Delete_LinkList(Link, 4);
        Print_LinkList(Link);
        return 0;
}
```

【小结或讨论】

通过该次实验我掌握了顺序表和链表的查找、插入和删除以及链表的创建（前插和后插）等基本操作，并能够设计出线性表应用的常用算法，比如链表的反转等。