《数据结构》上机报告

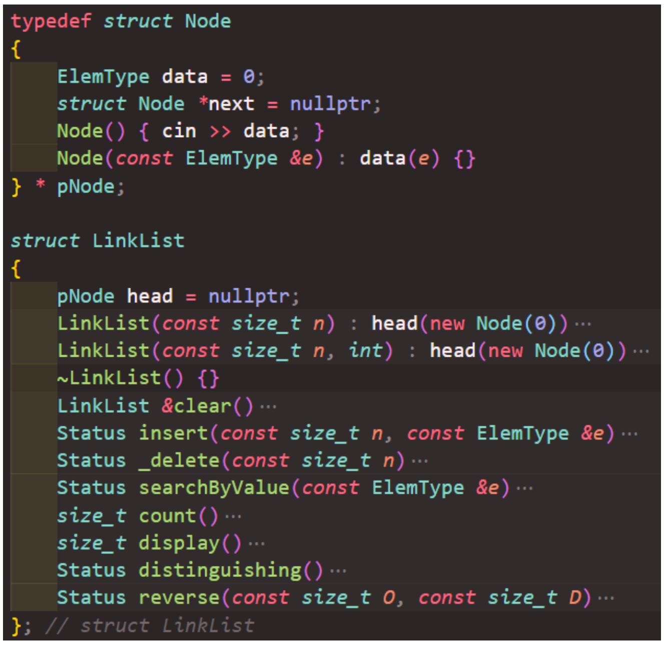
2020 年 10 月 13 日

姓名： 林日中 学号： 1951112 班级： 10072602 得分：

实验题目	链表实验报告	
问题描述	链表是一种物理存储单元上非连续、非顺序的存储结构，数据元素的逻辑顺序是通过链表中的指针链接次序实现的。链表由一系列结点（链表中每一个元素称为结点）组成，结点可以在运行时动态生成。每个结点包括两个部分：一个是存储数据元素的数据域，另一个是存储下一个结点地址的指针域。相比于线性表顺序结构，操作复杂。由于不必须按顺序存储，链表在插入的时候可以达到O(1)的复杂度，比另一种线性表顺序表快得多，但是查找一个节点或者访问特定编号的节点则需要O(*n*)的时间，而线性表和顺序表相应的时间复杂度分别是O(log*n*)和O(1)。 实验目的： 1、掌握线性表的链式表示（单链表、循环链表、双向循环链表）； 2、掌握链表实现线性表的基本操作，如建立、查找、插入、删除、去重、逆置、头部插入、尾部插入、销毁等； 3、掌握有序线性表的插入、删除、合并操作，及一元多项式的表示、相加和乘法等；	
基本要求	(1) 链表的基本操作； (2) 链表的逆置； (3) 链表的去重； (4) 一元多项式的表示、相加和乘法；	
	已完成基本内容（序号）：	1，2，3，4
选做要求		
	已完成选做内容（序号）：	无
数据结构设计	本次上机分别针对实验内容1，2，3和实验内容4分别设计了**两组**数据结构。 第一组数据结构包括基础数据结构结点struct Node，分别存储了表示存储数据的ElemType data和下一结点指针的Node *next；和由表示链表的struct LinkList，存储了头指针pNode head。 第二组数据结构包括多项式的项struct term，存储了系数double coef和指数int expn；基础数据结构结点struct Node，存储了当前结点的数据ElemType data和存储下一结点指针的Node *next；和由表示链表的struct polynominal，存储了头指针pNode head。	

功能
(函数)
说明

第一组：



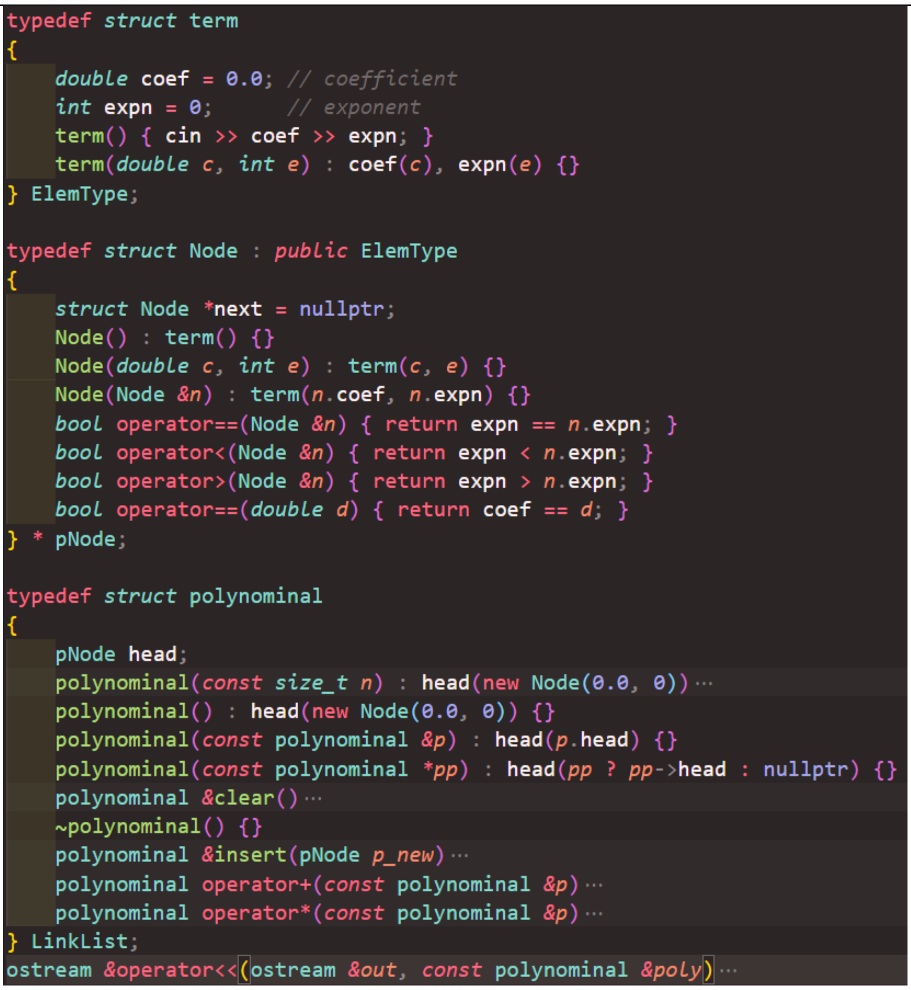
```
typedef struct Node
{
    ElemType data = 0;
    struct Node *next = nullptr;
    Node() { cin >> data; }
    Node(const ElemType &e) : data(e) {}
} * pNode;

struct LinkList
{
    pNode head = nullptr;
    LinkList(const size_t n) : head(new Node(0))···
    LinkList(const size_t n, int) : head(new Node(0))···
    ~LinkList() {}
    LinkList &clear()···
    Status insert(const size_t n, const ElemType &e)···
    Status _delete(const size_t n)···
    Status searchByValue(const ElemType &e)···
    size_t count()···
    size_t display()···
    Status distinguishing()···
    Status reverse(const size_t O, const size_t D)···
}; // struct LinkList
```

struct Node 的两个成员函数都为构造函数，并且是同名的重载函数，功能是直接通过由实参传入、或从输入流对象 cin 中读入的方式给结点的 data 赋初值。

struct LinkList 的两个构造函数互为重载，分别实现了头插法和尾插法，通过第一个参数后有没有一个整数作为区分。函数 clear()实现了对链表中所有元素的清空，使其恢复为空链表。成员函数 insert(), _delete(), searchByValue(), count(), display(), distinguishing(), reverse()分别实现了对链表中元素的插入、删除、查找、计数、展示、去重、逆置操作。

第二组：

<table>
<tr>
<td></td>
<td>



```
typedef struct term
{
    double coef = 0.0; // coefficient
    int expn = 0;      // exponent
    term() { cin >> coef >> expn; }
    term(double c, int e) : coef(c), expn(e) {}
} ElemType;

typedef struct Node : public ElemType
{
    struct Node *next = nullptr;
    Node() : term() {}
    Node(double c, int e) : term(c, e) {}
    Node(Node &n) : term(n.coef, n.expn) {}
    bool operator==(Node &n) { return expn == n.expn; }
    bool operator<(Node &n) { return expn < n.expn; }
    bool operator>(Node &n) { return expn > n.expn; }
    bool operator==(double d) { return coef == d; }
} * pNode;

typedef struct polynominal
{
    pNode head;
    polynominal(const size_t n) : head(new Node(0.0, 0))···
    polynominal() : head(new Node(0.0, 0)) {}
    polynominal(const polynominal &p) : head(p.head) {}
    polynominal(const polynominal *pp) : head(pp ? pp->head : nullptr) {}
    polynominal &clear()···
    ~polynominal() {}
    polynominal &insert(pNode p_new)···
    polynominal operator+(const polynominal &p)···
    polynominal operator*(const polynominal &p)···
} LinkList;
ostream &operator<<(ostream &out, const polynominal &poly)···
```

struct term 的两个同名构造函数，功能分别是通过由实参传入、或从输入流对象 cin 中读入的方式给成员变量 double coef 和 int expn 赋初值。

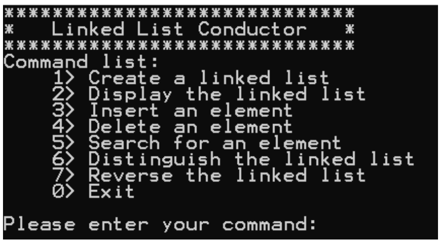
struct Node 以公有继承的方式继承了 struct term，并且增加了成员变量 Node *next，用于存储指向下一结点的指针。3 个构造函数分别实现了构造空结点、由实参传入、或从输入流对象 cin 中读入的方式给各成员变量赋初值。对运算符==、<、>的重载函数分别返回指数的大小比较结果，使两个结点的指数 expn 更易比较。

struct polynomial 有一个成员变量 pNode head，存储指向该多项式链表的头指针。各构造函数分别实现了从标准输入流中读入 n 组数据、不读数据、由现有多项式克隆构造新的多项式链表。成员函数 clear()实现了清除多项式中所有元素的功能；insert()函数实现了多项式的项的插入，并且能够处理相同指数项的合并、系数为 0 的结点的清除，避免出现同类项未合并、为 0 的项未删除等情况。对运算符+和*的重载分别实现了多项式的加法和乘法。

</td>
</tr>
<tr>
<td>开发环境</td>
<td>Windows 10, Visual Studio Code with g++, C++ language</td>
</tr>
</table>

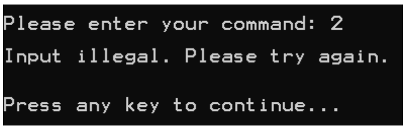
调试分析

第一组：



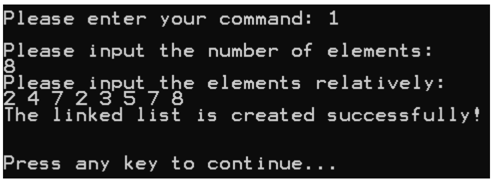
```
*******************************
*    Linked List Conductor    *
*******************************
Command list:
    1> Create a linked list
    2> Display the linked list
    3> Insert an element
    4> Delete an element
    5> Search for an element
    6> Distinguish the linked list
    7> Reverse the linked list
    0> Exit

Please enter your command:
```

此为主菜单。



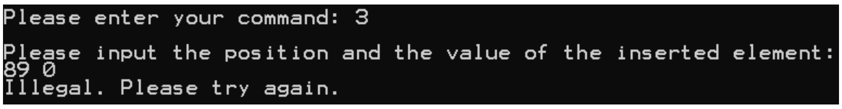
```
Please enter your command: 2

Input illegal. Please try again.

Press any key to continue...
```

若未创建链表就进行除创建和退出的其他操作，直接掷出错误提示信息，并重新进入菜单。



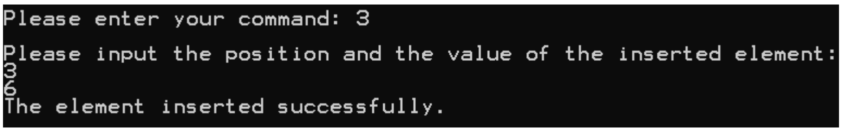
```
Please enter your command: 1

Please input the number of elements:
8
Please input the elements relatively:
2 4 7 2 3 5 7 8
The linked list is created successfully!

Press any key to continue...
```

成功以尾插法创建链表。链表元素包括两组重复数据（2 和 7）。



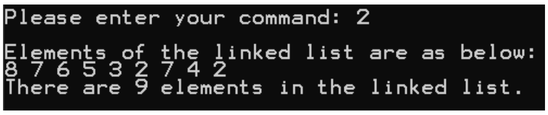
```
Please enter your command: 3

Please input the position and the value of the inserted element:
89 0
Illegal. Please try again.
```

插入的位置为 0 或超出当前链表长度时，掷出非法提示信息。



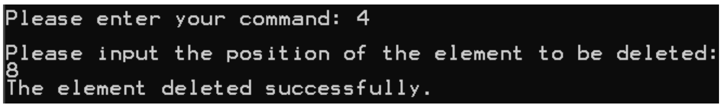
```
Please enter your command: 3

Please input the position and the value of the inserted element:
3
6
The element inserted successfully.
```

插入成功。



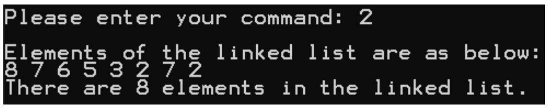
```
Please enter your command: 2

Elements of the linked list are as below:
8 7 6 5 3 2 7 4 2
There are 9 elements in the linked list.
```

展示链表内所有元素。



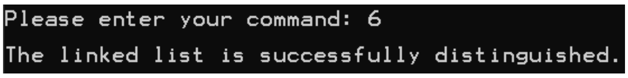
```
Please enter your command: 4

Please input the position of the element to be deleted:
8
The element deleted successfully.
```

删除元素。



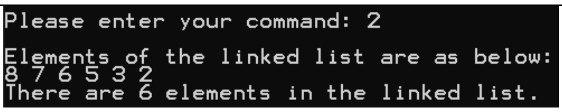
```
Please enter your command: 2

Elements of the linked list are as below:
8 7 6 5 3 2 7 2
There are 8 elements in the linked list.
```

观察到成功删除。



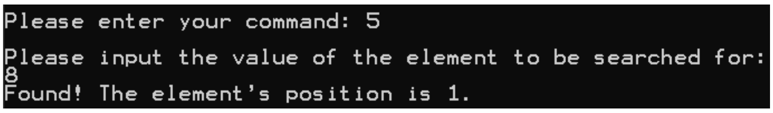
```
Please enter your command: 6

The linked list is successfully distinguished.
```

去重。



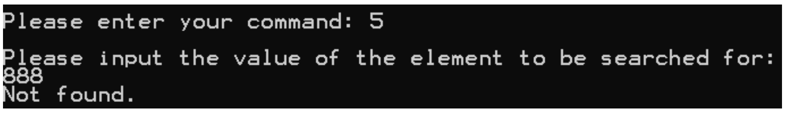
```
Please enter your command: 2

Elements of the linked list are as below:
8 7 6 5 3 2
There are 6 elements in the linked list.
```

观察到成功去重。



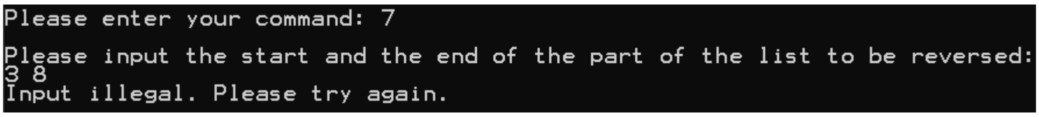
```
Please enter your command: 5

Please input the value of the element to be searched for:
8
Found! The element's position is 1.
```

按值查找元素，找到并反馈其在链表中的顺序。



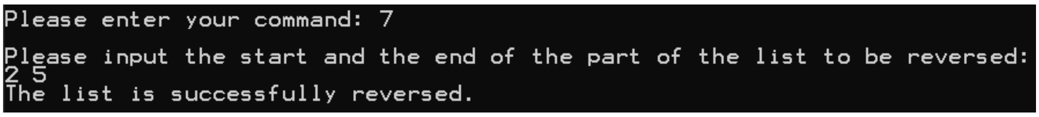
```
Please enter your command: 5

Please input the value of the element to be searched for:
888
Not found.
```

按值查找元素，提示未找到。



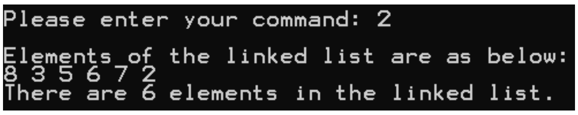
```
Please enter your command: 7

Please input the start and the end of the part of the list to be reversed:
3 8
Input illegal. Please try again.
```

输入需要逆置部分的首尾序号非法。



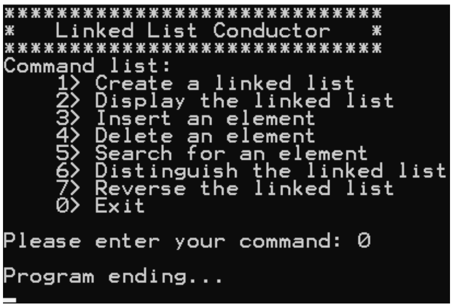
```
Please enter your command: 7

Please input the start and the end of the part of the list to be reversed:
2 5
The list is successfully reversed.
```

逆置 2 至 5 结点。



```
Please enter your command: 2

Elements of the linked list are as below:
8 3 5 6 7 2
There are 6 elements in the linked list.
```

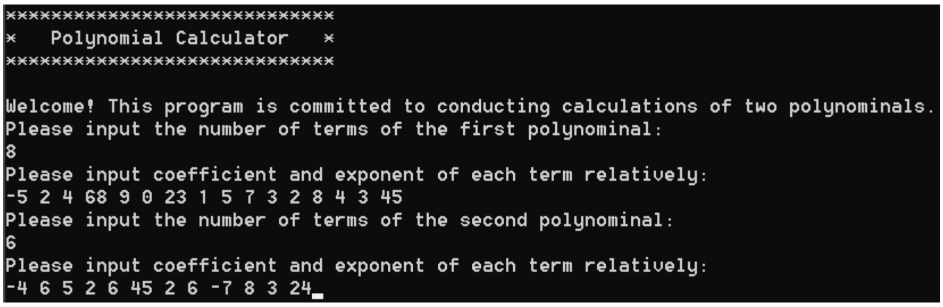
观察到成功逆置。



```
*****************************
*   Linked List Conductor   *
*****************************
Command list:
    1> Create a linked list
    2> Display the linked list
    3> Insert an element
    4> Delete an element
    5> Search for an element
    6> Distinguish the linked list
    7> Reverse the linked list
    0> Exit

Please enter your command: 0

Program ending...
```

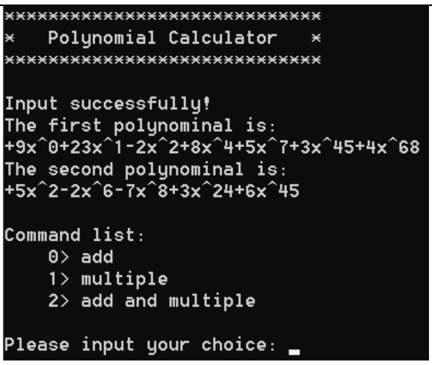
退出程序。

第二组：



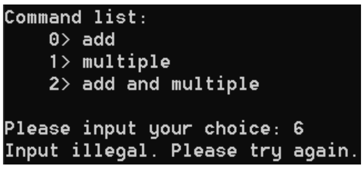
```
******************************
*   Polynomial Calculator   *
******************************

Welcome! This program is committed to conducting calculations of two polynominals.
Please input the number of terms of the first polynominal:
8
Please input coefficient and exponent of each term relatively:
-5 2 4 68 9 0 23 1 5 7 3 2 8 4 3 45
Please input the number of terms of the second polynominal:
6
Please input coefficient and exponent of each term relatively:
-4 6 5 2 6 45 2 6 -7 8 3 24
```

输入数据。所输入多项式的各项为乱序，并且含有指数相同的项。



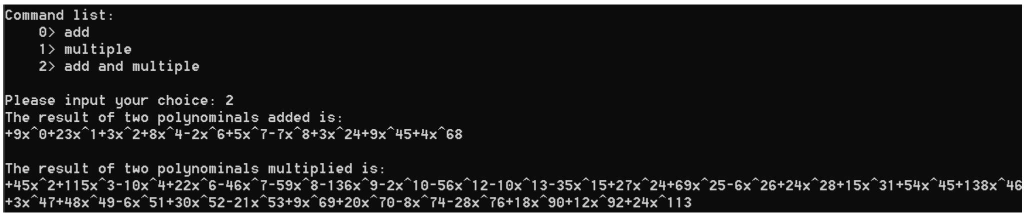
```
*****************************
*   Polynomial Calculator   *
*****************************

Input successfully!
The first polynominal is:
+9x^0+23x^1-2x^2+8x^4+5x^7+3x^45+4x^68
The second polynominal is:
+5x^2-2x^6-7x^8+3x^24+6x^45

Command list:
    0> add
    1> multiple
    2> add and multiple

Please input your choice: _
```

展示所输入多项式，发现同类项被合并，并且系数为 0 的项被删去。提示命令列表。



```
Command list:
    0> add
    1> multiple
    2> add and multiple

Please input your choice: 6
Input illegal. Please try again.
```

非法输入。



```
Command list:
    0> add
    1> multiple
    2> add and multiple

Please input your choice: 2
The result of two polynominals added is:
+9x^0+23x^1+3x^2+8x^4-2x^6+5x^7-7x^8+3x^24+9x^45+4x^68

The result of two polynominals multiplied is:
+45x^2+115x^3-10x^4+22x^6-46x^7-59x^8-136x^9-2x^10-56x^12-10x^13-35x^15+27x^24+69x^25-6x^26+24x^28+15x^31+54x^45+138x^46
+3x^47+48x^49-6x^51+30x^52-21x^53+9x^69+20x^70-8x^74-28x^76+18x^90+12x^92+24x^113
```

输出结果并结束程序。

综上，两组实验都很好地完成了题目对单向链表功能的要求，功能完好，使用方便，并且能够较好地处理非法数据的输入并反馈相关错误提示，程序界面友好，具有比较恰当的人性化提醒，具有一定的鲁棒性。

心得体会

一、 实验总结

在本次上机实验中，我复习了理论课上学到的线性表的定义和作用，将课本上线性表的链表存储结构和与其相关的建立、遍历、查找、插入、删除和去重等函数样例封装成了 struct LinkList 结构体。

链表的优点有，插入删除速度快，内存利用率高，不会浪费内存；大小没有固定，拓展很灵活。链表的缺点有不能随机查找，必须从第一个开始遍历，查找效率低。

从存储空间利用率角度出发：顺序表的存储空间是静态分配的，在执行程序前需预先分配一定长度的连续的存储空间。当线性表的长度 n 不能预先确定时，可能会分配的过大或过小，则导致存储空间的浪费或溢出；链表的存储空间是动态分配的，无需预先分配存储空间，只要内存中尚有可分配空间就不会发生溢出。但指针域需占额外的存储空间。所以当线性表长度变化较大或难以估计时，宜采用链表存储结构；当线性表长度变化不大，且能预先估计存储量大小时，为节省存储空间，宜采用顺序表存储结构。

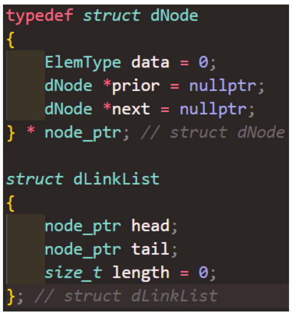
从时间效率角度考虑：若对线性表进行的主要操作是查找，很少进行插入和删除操作时，宜采用顺序表存储结构；若对线性表进行插入和删除操作频繁时，宜采用链表作为存储结构。

二、 双向循环链表的存储结构描述

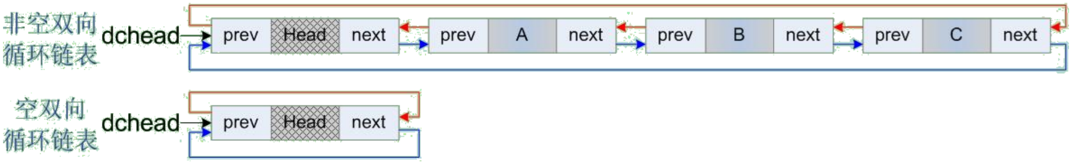
双向链表通常采用带表头结点的循环链表形式，即双向循环链表。双向循环链表在双向链表的基础上，将表头结点的前驱指针指向尾结点，尾结点的后驱指针指向头结点，首

尾相连形成一个双向环。双向循环链表可方便地获取当前结点的前驱结点，不必像单向循环链表那样从头开始遍历；而其循环的特性又可方便地从任一结点出发单向遍历整个链表，不必像双向链表那样根据方向而使用不同的指针域。

带头结点的双向循环链表如下图所示：



```
typedef struct dNode
{
    ElemType data = 0;
    dNode *prior = nullptr;
    dNode *next = nullptr;
} * node_ptr; // struct dNode

struct dLinkList
{
    node_ptr head;
    node_ptr tail;
    size_t length = 0;
}; // struct dLinkList
```

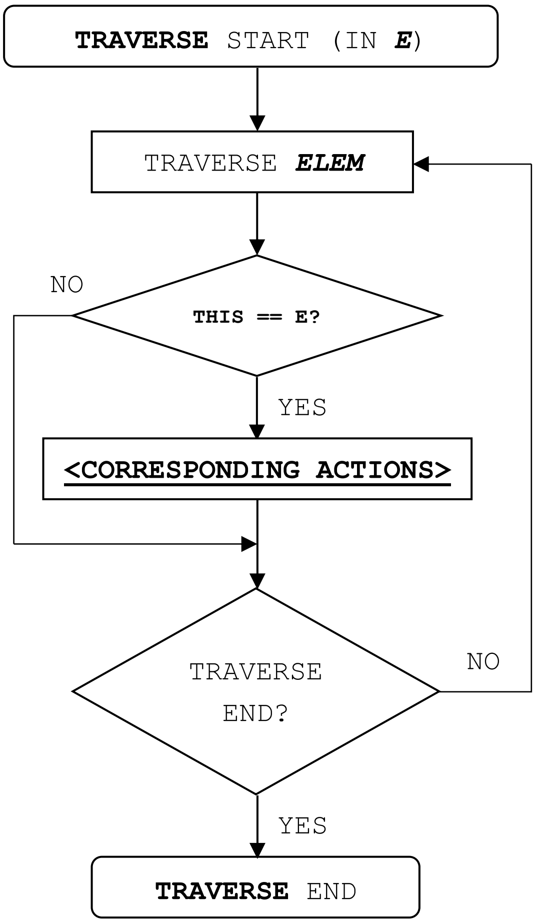


三、　　性能分析

（1）　链表的遍历相关操作（建立、查找、插入、删除等操作）

时间复杂度为O(n)。

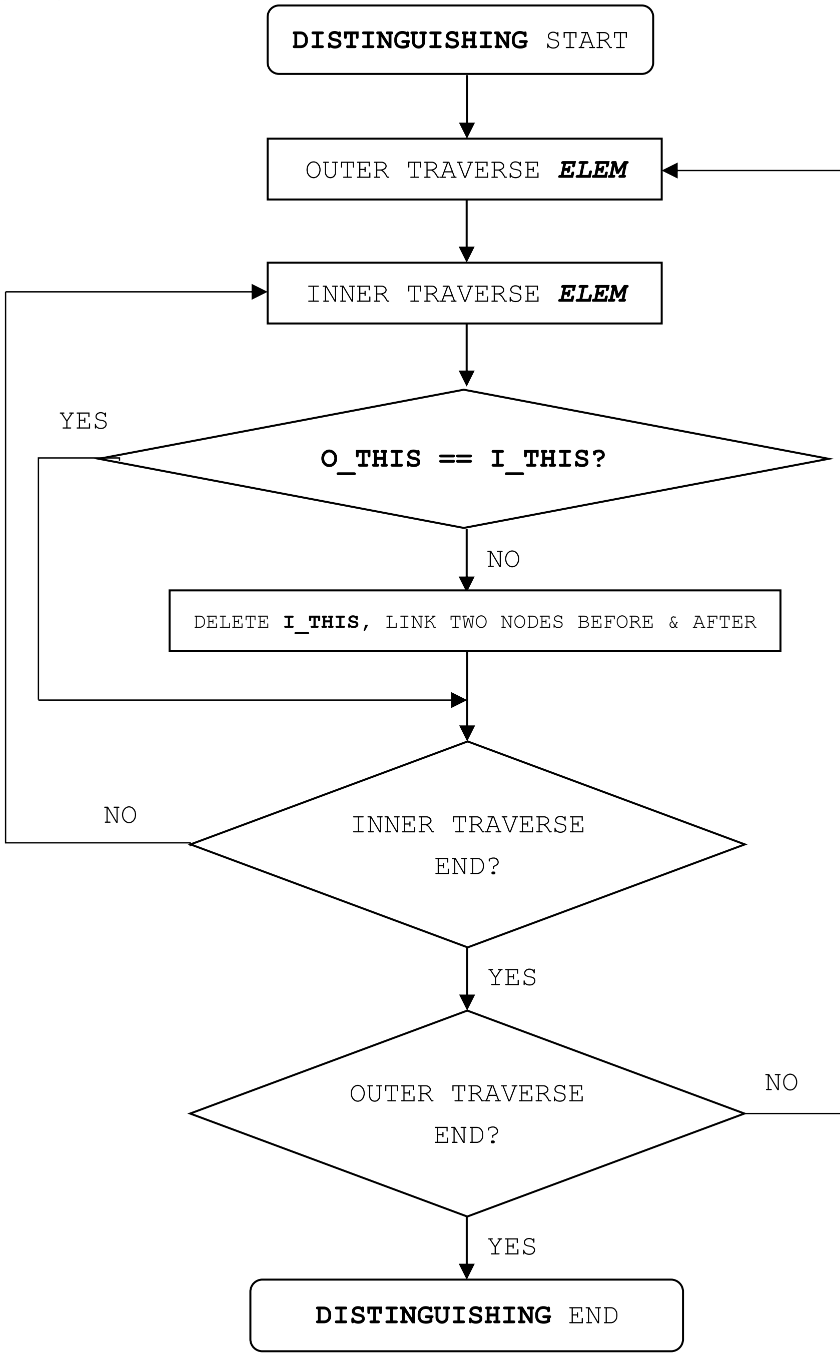
流程图如下：



（2）　链表的去重

时间复杂度为O(n^2)，空间复杂度为O(1)。

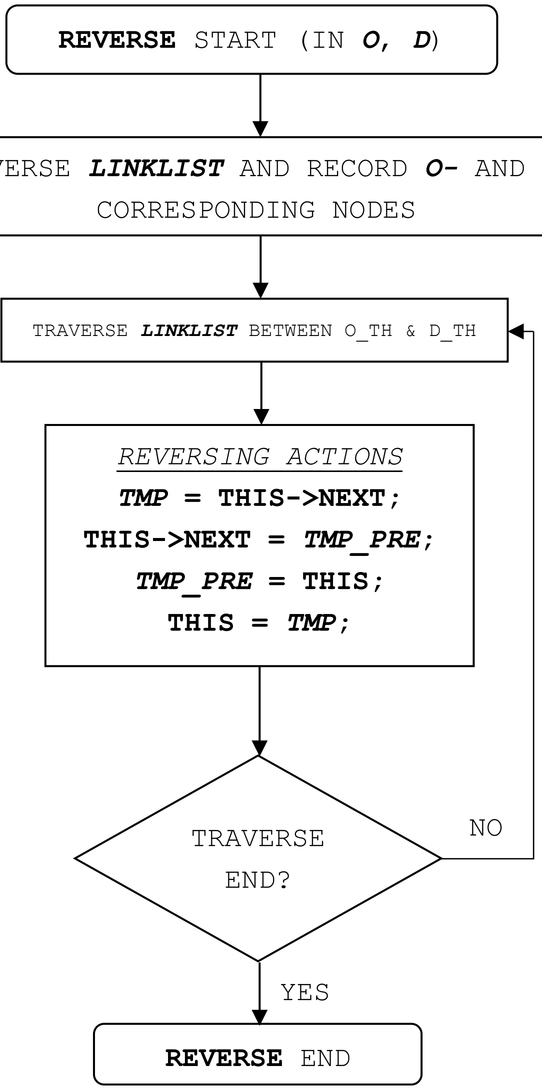
流程图如下：



（3） 链表的逆置

时间复杂度为O(n)，空间复杂度为O(1)。

流程图如下：



第一组源码

```
#ifdef __GNUC__
#include <bits/stdc++.h>
#endif

#ifdef _MSC_VER
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <cstdlib>
#include <cstring>
#include <cmath>
#endif

#include <windows.h>
#include <conio.h>

using namespace std;

#define TRUE 1
```

```
#define FALSE 0
#define OK 1
#define ERROR 0
#define INFEASIBLE -1
typedef int Status;
typedef int Boolean;
typedef int ElemType;

typedef struct Node
{
    ElemType data = 0;
    struct Node *next = nullptr;
    Node() { cin >> data; }
    Node(const ElemType &e) : data(e) {}
} * pNode;

struct LinkList
{
    pNode head = nullptr;
    LinkList(const size_t n) : head(new Node(0))
    { // create a linked list through head-inserted method
        pNode p = nullptr;
        for (size_t i = n; i > 0; i--)
        {
            p = new Node();
            p->next = head->next;
            head->next = p;
        }
    }
    LinkList(const size_t n, int) : head(new Node(0))
    { // create a linked list through tail-inserted method
        pNode p = head;
        for (size_t i = 0; i < n; i++)
        {
            p->next = new Node();
            p = p->next;
        }
    }
    ~LinkList() {}
    LinkList &clear()
    { // destroy the linked list
        while (head)
        {
            pNode p = head->next;
```

```cpp
            delete head;
            head = p;
        }
        return *this;
    }
    Status insert(const size_t n, const ElemType &e)
    { // insert a Node whose value is e, before n_th position
        pNode p = head, q = nullptr;
        size_t i = 0;
        for (; i < n - 1 && p; i++)
        {
            p = p->next;
        }
        if (i > n - 1 || !p)
        {
            return ERROR;
        }
        q = new Node(e);
        q->next = p->next;
        p->next = q;
        return OK;
    }
    Status _delete(const size_t n)
    { // delete n_th Node
        pNode p = head, q = nullptr;
        size_t i = 0;
        for (; i < n - 1 && p->next; i++)
        {
            p = p->next;
        }
        if (i > n - 1 || !(p->next))
        {
            return ERROR;
        }
        q = p->next;
        p->next = q->next;
        delete q;
        return OK;
    }
    Status searchByValue(const ElemType &e)
    {
        pNode p = head->next;
        for (int i = 1; p; i++)
        {
            if (p->data == e)
            {
                return i;
            }
            p = p->next;
        }
        return INFEASIBLE;
    }
    size_t count()
    {
        size_t n = 0;
        pNode p = head->next;
        while (p)
        {
            n++;
            p = p->next;
        }
        return n;
    }
    size_t display()
    {
        pNode p = head->next;
        size_t _count = 0;
        while (p)
        {
            cout << p->data << " ";
            p = p->next;
            _count++;
        }
        cout << endl;
        return _count;
    }
    Status distinguishing()
    {
        pNode p = head->next, q, k;
        while (p->next)
        {
            q = p;
            k = q->next;
            while (k)
            {
                if (p->data == k->data)
                {
                    q->next = k->next;
                    delete k;
                    k = q->next;
                }
                else
                {
                    q = k;
                    k = k->next;
                }
            }
            if (p->next)
            {
                p = p->next;
            }
        }
        return OK;
    }
    Status reverse(const size_t O, const size_t D)
    { // reverse the linked list from O_th to D_th elements
        if (O < 1U || O >= D)
        {
            return ERROR;
        }
        size_t len = 0;
        pNode cur = head->next, o = nullptr, o_pre = nullptr, d_next = nullptr;

        while (cur)
        {
            ++len;
            if (len == O - 1)
            {
                o_pre = cur;
            }
            if (len == D + 1)
            {
                d_next = cur;
            }
            cur = cur->next;
        }
        if (len < D)
        {
            return ERROR;
        }
        o = o_pre ? o_pre->next : head->next;

```

```
        pNode tmp_pre = d_next, tmp = nullptr;

        while (o != d_next)
        {
            tmp = o->next;
            o->next = tmp_pre;
            tmp_pre = o;
            o = tmp;
        }

        if (o_pre)
        {
            o_pre->next = tmp_pre;
        }
        else
        {
            head->next = tmp_pre;
        }
        return OK;
    }
}; // struct LinkList

inline int menu()
{
    system("cls");
    cout << "*****************************" << endl;
    cout << "*   Linked List Conductor   *" << endl;
    cout << "*****************************" << endl;
    cout << "Command list: " << endl;
    cout << "    1> Create a linked list" << endl;
    cout << "    2> Display the linked list" << endl;
    cout << "    3> Insert an element" << endl;
    cout << "    4> Delete an element" << endl;
    cout << "    5> Search for an element" << endl;
    cout << "    6> Distinguish the linked list" << endl;
    cout << "    7> Reverse the linked list" << endl;
    cout << "    0> Exit" << endl;
    cout << endl;
    cout << "Please enter your command: ";
    return _getche() - '0';
}

int main()
{
```

```cpp
    size_t n = 0, position = 0;
    size_t left = 0, right = 0;
    int order = 0;
    ElemType value;
    LinkList *L = nullptr;

    while (true)
    {
        switch (order = menu())
        {
        case 0: // exit
            cout << endl
                 << endl
                 << "Program ending..." << endl;
            if (L)
            {
                L->clear();
                delete L;
            }
            Sleep(1000);
            exit(0);
            break;
        case 1: // create
            cout << endl
                 << endl;
            if (L)
            {
                delete L;
                L = nullptr;
            }
            cout << "Please input the number of elements: " << endl;
            cin >> n;
            cout << "Please input the elements relatively: " << endl;
            L = new LinkList(n);
            cout << "The linked list is created successfully! " << endl;
            break;
        default:
            if (L)
            {
                switch (order)
                {
                case 2: // display
                    cout << endl
                         << endl
```

```
                     << "Elements of the linked list are as below: " << endl;
                n = L->display();
                cout << "There are " << n << " elements in the linked list. " << endl;
                break;
            case 3: // insert
                cout << endl
                     << endl
                     << "Please input the position and the value of the inserted element: " << endl;
                cin >> position >> value;
                if (L->insert(position, value))
                {
                    cout << "The element inserted successfully. " << endl;
                }
                else
                {
                    cerr << "Illegal. Please try again. " << endl;
                }
                break;
            case 4: // delete
                cout << endl
                     << endl
                     << "Please input the position of the element to be deleted: " << endl;

                cin >> position;
                if (L->_delete(position))
                {
                    cout << "The element deleted successfully. " << endl;
                }
                else
                {
                    cout << "Illegal. Please try again. " << endl;
                }
                break;
            case 5: // search
                cout << endl
                     << endl
                     << "Please input the value of the element to be searched for:" << endl;
                cin >> value;
                position = L->searchByValue(value);
                if (position == (size_t)INFEASIBLE)
                {
                    cerr << "Not found. " << endl;
```

```cpp
                        }
                        else
                        {
                            cout << "Found! The element's position is " << position << "." << endl;
                        }
                        break;
                    case 6: // distinguishing
                        cout << endl
                             << endl;
                        L->distinguishing();
                        cout << "The linked list is successfully distinguished. " << endl;
                        break;
                    case 7: // reverse
                        cout << endl
                             << endl
                             << "Please input the start and the end of the part of the list to be reversed: " << endl;
                        cin >> left >> right;
                        if (L->reverse(left, right))
                        {
                            cout << "The list is successfully reversed. " << endl;
                        }
                        else
                        {
                            cerr << "Input illegal. Please try again. " << endl;
                        }
                        break;
                    }
                }
                else
                {
                    cout << endl
                         << endl;
                    cerr << "Input illegal. Please try again." << endl;
                }
                break;
            }
            cout << endl
                 << endl
                 << "Press any key to continue..." << endl;
            (void)_getch();
        }

```

```
    if (L)
    {
        L->clear();
        delete L;
    }

    return 0;
}
```

第二组源码

```
#ifdef __GNUC__
#include <bits/stdc++.h>
#endif

#ifdef _MSC_VER
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <cstdlib>
#include <cstring>
#include <cmath>
#endif

#include <windows.h>
#include <conio.h>

using namespace std;

#define TRUE 1
#define FALSE 0
#define OK 1
#define ERROR 0
#define INFEASIBLE -1
typedef int Status;
typedef int Boolean;
typedef struct term
{
    double coef = 0.0; // coefficient
    int expn = 0;      // exponent
    term() { cin >> coef >> expn; }
    term(const double c, const int e) : coef(c), expn(e) {}
} ElemType;

typedef struct Node : public ElemType
{
    struct Node *next = nullptr;
    Node() : term() {}
    Node(const double c, const int e) : term(c, e) {}
    Node(const Node &n) : term(n.coef, n.expn) {}
    bool operator==(const Node &n) { return expn == n.expn; }
    bool operator<(const Node &n) { return expn < n.expn; }
    bool operator>(const Node &n) { return expn > n.expn; }
    bool operator==(const double d) { return coef == d; }
} * pNode;

typedef struct polynominal
{
    pNode head;
    polynominal(const size_t n) : head(new Node(0.0, 0))
    {
        for (size_t i = 0; i < n; i++)
        {
            pNode t = new Node();
            insert(t);
        }
    }
    polynominal() : head(new Node(0.0, 0)) {}
    polynominal(const polynominal &p) : head(p.head) {}
    polynominal(const polynominal *pp) : head(pp ? pp->head : nullptr) {}
    polynominal &clear()
    { // clear all elements and recover to an empty linked list
        while (head)
        {
            pNode p = head->next;
            delete head;
            head = p;
        }
        return *this;
    }
    ~polynominal() {}
    polynominal &insert(pNode p_new)
    { // insert an element into the linked list on the existed rule
        pNode m = head, n = head->next;

        do
        {
            if (!p_new)
            {
                break;
            }
            if (!(p_new->coef))
            {
                delete p_new;
                break;
            }
            if (!head->next)
            {
                head->next = p_new;
            }
            else
            {
                while (true)
                {
                    if (!n->next && *n < *p_new)
                    {
                        n->next = p_new;
                        break;
                    }
                    else if (*n > *p_new)
                    {
                        m->next = p_new;
                        p_new->next = n;
                        break;
                    }
                    else if (*n == *p_new)
                    {
                        n->coef += p_new->coef;
                        delete p_new;
                        break;
                    }
                    else
                    {
                        m = n;
                        n = n->next;
                    }
                }
            }
        } while (false);
        return *this;
    }
    polynominal operator+(const polynominal &p)
    {
        polynominal result;
```

```cpp
        pNode pLeft = this->head->next, pRight = p.head->next, pp = nullptr;
        while (pLeft && pRight)
        {
            if (*pLeft == *pRight)
            {
                pp = new Node(pLeft->coef + pRight->coef, pLeft->expn);
                pLeft = pLeft->next;
                pRight = pRight->next;
            }
            else if (*pLeft < *pRight)
            {
                pp = new Node(*pLeft);
                pLeft = pLeft->next;
            }
            else // if (*pLeft > *pRight)
            {
                pp = new Node(*pRight);
                pRight = pRight->next;
            }
            result.insert(pp);
        }
        while (pLeft)
        {
            pp = new Node(*pLeft);
            result.insert(pp);
            pLeft = pLeft->next;
        }
        while (pRight)
        {
            pp = new Node(*pRight);
            result.insert(pRight);
            pRight = pRight->next;
        }

        return result;
    }
    polynominal operator*(const polynominal &p)
    {
        polynominal result;
        pNode pLeft = this->head->next, pRight = p.head->next;

        while (pLeft)
        {
            pRight = p.head->next;
            while (pRight)
            {
                pNode p = new Node(pLeft->coef * pRight->coef, pLeft->expn + pRight->expn);
                result.insert(p);
                pRight = pRight->next;
            }
            pLeft = pLeft->next;
        }
        return result;
    }
} LinkList;
ostream &operator<<(ostream &out, const polynominal &poly)
{
    pNode p = poly.head->next;
    while (p)
    {
        out << showpos << p->coef << "x^";
        out << noshowpos << p->expn;
        p = p->next;
    }
    return out;
}

inline void menu()
{
    system("cls");
    cout << "*****************************" << endl
         << "*   Polynomial Calculator   *" << endl
         << "*****************************" << endl
         << endl;
}

int main()
{
    size_t m, n;
    int choice;
    menu();
    cout << "Welcome! This program is committed to conducting calculations of two polynominals." << endl;

    cout << "Please input the number of terms of the first polynominal: " << endl;
    cin >> m;
    cout << "Please input coefficient and exponent of each term relatively: " << endl;
    LinkList l_m(m);

    cout << "Please input the number of terms of the second polynominal: " << endl;
    cin >> n;
    cout << "Please input coefficient and exponent of each term relatively: " << endl;
    LinkList l_n(n);
    LinkList add(nullptr), mul(nullptr);

    bool pass = false;
    while (!pass)
    {
        menu();
        cout << "Input successfully! " << endl;
        cout << "The first polynominal is: " << endl
             << l_m << endl;
        cout << "The second polynominal is: " << endl
             << l_n << endl;

        cout << endl
             << "Command list: " << endl
             << "    0> add" << endl
             << "    1> multiple" << endl
             << "    2> add and multiple" << endl
             << endl
             << "Please input your choice: ";

        switch (choice = _getche() - '0')
        {
        case 0:
            add = l_m + l_n;
            cout << endl
                 << "The result of two polynominals added is: " << endl;
            cout << add << endl;
            pass = true;
            break;
        case 1:
            mul = l_m * l_n;
            cout << endl
                 << "The result of two polynominals multiplied is: " << endl;
            cout << mul << endl;
            pass = true;
            break;
        case 2:
            add = l_m + l_n;
            cout << endl
```

```
                 << "The result of two polynominals added is: " << endl;
            cout << add << endl;
            mul = l_m * l_n;
            cout << endl
                 << "The result of two polynominals multiplied is: " << endl;
            cout << mul << endl;
            pass = true;
            break;
        default:
            cerr << endl
                 << "Input illegal. Please try again. " << endl;
            Sleep(1000);
        }
    }
    add.clear();
    mul.clear();

    (void)_getch();
    return 0;
}
```