《数据结构与算法实验》第 5 次实验

学院： 专业： 年级：
姓名： 学号： 日期： 2022 年 3 月 26 日

设计实验：一元多项式相加

一、问题描述

已知 $A(x)=a_0+a_1x+a_2x^2+\cdots+a_nx^n$ 和 $B(x)=b_0+b_1x+b_2x^2+\cdots+b_mx^m$ ，并且在 A(x)和 B(x)中指数相差很多，求 A(x)=A(x)+B(x)

二、基本要求

- 设计储存结构表示一元多项式；
- 设计算法实现一元多项式相加；
- 分析算法的时间复杂度和空间复杂度。

三、设计思想

一元多项式。求和实质上是合并同类项的过程，其运算规则为：

(1)若两项的指数相同，则系数相加；

(2)若两项的指数不等，则将两项加在结果中。

一元多项式 A(x)=a0+a1x+a2x2+…+anxn 由 n+1 个系数唯一确定，因此，可以用一个线性表(a0,a1,a2,…,an) 来表示，每一项的指数 i 隐含在其系数 ai 的序号里。但是，当多项式的指数很高且变化很大时，在表示多项式的线性表中就会存在很多 0 元素。一个较好的储存方法是只储存非 0 元素，但是需要在储存非 0 元素系数的同时储存相应的指数，这样，一元多项式的每一个非 0 项可由系数和指数唯一表示。

由于两个多项式相加后会改变多项式的系数和指数，因此不适合采用顺序表，我们选用单链表储存，每一个非 0 项对应单链表中的一个结点，且单链表应按照指数递增有序排列。

1. 有参构造函数

按照指数升幂次序依次输入系数（不为 0）和指数，使用尾插法，每次将新输入的

结点插到最后。这样得到了一个从头到位按照指数升幂排列的链表。

2. 加法函数

这个函数是本次实验的核心函数。对于多项式 A(x)，B(x)，把它们存入单链表 A,B 之后，我们考虑将相加的结果存入单链表 C。

首先，用两个指针 pre,qre 指向链表 A 和 B 的第一个非空结点，指针 p,q 分别指向 pre,qre 的下一个结点，然后进行如下算法：

- 判断 p,q 是否为空结点。
- 当 p,q 均不是空结点时，比较 p,q 的指数，如果不相等则进行排序；如果相等则将两者的系数相加，储存。然后让指针均后移，指向下一个结点。
- 当存在空结点时，直接指向下一个结点。

3. 输出函数要注意的问题

- 非首项的系数若为正数，输出的时候有 + 号。
- x 的 1 次方的指数不输出。
- 非 0 次项的系数 1 和-1 的 1 不输出。
- 考虑相加后多项式为 0 的情况。

3.抽象数据类型设计

设计一个 C++类。

成员变量：系数 coef，指数 exp，指向后一个结点的指针 next。

成员函数：无参构造函数（构造一个只有空头结点的链表），有参构造函数（传入参数长度，预处理线性表），析构函数，输出函数，加法函数。

四、代码实现

1.设计头文件-LinkList.h

```
#ifndef LinkList_H
#define LinkList_H

struct Node
{
    double coef;
    int exp;
```

```
    Node *next;
};

class LinkList
{
    public:
        LinkList();
        LinkList(double a[],int b[],int n);
        ~LinkList();//Îö¹¹
        void PrintList();
        friend void Add(LinkList &A,LinkList &B);
    private:
        Node *first;
};

#endif
```

2. 设计成员函数-LinkList.cpp

```
#include <iostream>
#include <cmath>
using namespace std;
#include "LinkList.h"

LinkList::LinkList()
{
    first = new Node;
    first -> next = NULL;
}

LinkList::LinkList(double a[],int b[],int n)
{
    Node *r,*s;
    first = new Node;
    r = first;
    for(int i = 0;i < n;i++)
    {
        s = new Node;
        s -> coef = a[i];
        s -> exp = b[i];
        r -> next = s;
        r = s;
    }
    r -> next = NULL;
}
```

```
LinkList::~LinkList()
{
    Node *q;
    while(first != NULL)
    {
        q = first;
        first = first -> next;
        delete q;
    }
}

void LinkList:: PrintList()
{
    Node *p = first -> next;
    if (p == NULL)
    {
        cout<<"0"<<endl;
        return;
    }
    while (p != NULL)
    {
        if (p -> coef > 0 && p != first -> next)
        {
            cout<<"+";
        }
        if (p -> coef == 1 && p -> exp != 0)
        {
            ;
        }
        else if (p -> coef == -1)
        {
            if(p -> exp == 0) cout<<"-1";
            else cout<<"-";
        }
        else cout<<p -> coef;
        if(p -> exp > 0) cout<<"x";
        if(p -> exp > 1) cout<<"^"<<p -> exp;
        p = p -> next;
    }
    cout<<endl;
}

void Add(LinkList &A,LinkList &B)
{
```

```
    Node *pre = A.first,*p = pre->next;
    Node *qre = B.first,*q = qre->next;
    while (p != NULL && q != NULL)
    {
        if (p -> exp < q -> exp)
        {
            pre = p;
            p = p -> next;
        }
        else if (p -> exp > q -> exp)
        {
            Node *v = q -> next;
            pre -> next = q;
            q -> next = p;
            q = v;
        }
        else
        {
            p -> coef = p -> coef + q -> coef;
            if (p -> coef == 0)
            {
                pre -> next = p -> next;
                delete p;
                p = pre -> next;
            }
            else
            {
                pre = p;
                p = p -> next;
            }
            qre -> next = q -> next;
            delete q;
            q = qre -> next;
        }
    }
    if(q != NULL) pre -> next = q;
    delete B.first;
}
```

3. 设计主函数-LinkList.cpp

```
#include <iostream>
using namespace std;
#include "LinkList.h"

int main()
```

```
{
    int n=0,m=0;
    //多项式 A
    cout<<"多项式 A: "<<endl;
    cout<<"请输入多项式的项数: ";
    cin>>n;
    double a1[n];
    int b1[n];
    cout<<"请按照升幂依次输入每一项的系数和次数: "<<endl;
    for(int i = 0;i < n;i++)
    {
        cout<<"第"<<(i+1)<<"项: ";
        cin>>a1[i];
        cin>>b1[i];
        cout<<endl;
    }
    LinkList A(a1,b1,n);
    //多项式 B
    cout<<"多项式 B: "<<endl;
    cout<<"请输入多项式的项数: ";
    cin>>m;

    double a2[m];
    int b2[m];
    cout<<"请按照升幂依次输入每一项的系数和次数: "<<endl;
    for(int j = 0;j < m;j++)
    { cout<<"第"<<(j+1)<<"项: ";
      cin>>a2[j];
      cin>>b2[j];
      cout<<endl;
    }
    LinkList B(a2,b2,m);
    //A+B
    cout<<"已知: "<<endl;
    cout<<"A=";A.PrintList();
    cout<<endl;
    cout<<"B=";B.PrintList();
    cout<<endl;
    Add(A,B);
    cout<<"则 A+B=";A.PrintList();
    return 0;
}
```

五、运行测试

测试结果如图。

```
多项式A:
请输入多项式的项数：4
请按照升幂依次输入每一项的系数和次数：
第1项：1 0

第2项：-10 6

第3项：2 8

第4项：7 14

多项式B:
请输入多项式的项数：5
请按照升幂依次输入每一项的系数和次数：
第1项：-1 4

第2项：10 6

第3项：-3 10

第4项：8 14

第5项：4 18

已知：
A=1-10x^6+2x^8+7x^14

B=-x^4+10x^6-3x^10+8x^14+4x^18

则A+B=1+2x^8-3x^10+15x^14+4x^18

--------------------------------
Process exited after 41.16 seconds with return value 0
请按任意键继续. . .
```

六、总结与心得

一元多项式相加主要是关注“一一对应”，难点是控制指针的移动和次数、系数、项数之间的关系。一元多项式的加法说起来其实也只是“合并同类项”，次数相同的系数相加，次数不同的直接记录即可。而在对结果进行输出的时候，也要进行分类讨论，考虑对于一项来说，要判断它的次数是否是 0，系数为 1 的时候应该要隐藏等等细节。对于数据结构和链表的设计，要考虑其边界条件的处理，对于 NULL 的 p 指针，其并未指向一个结点，p->coef 和 p->exp 其实是根本不存在的。这个程序的设计实用性较强，更加方便人们对数学问题进行研究。