数据结构实验报告

评分

满分——10分

学号：2015111833 姓名：邓金红 专业：计算机科学与技术

知识范畴：线性表 完成日期：2017年03月24日

实验题目：基于链表的多项式乘法

实验内容及要求：

从字符文件输入两个多项式的非零系数及对应的指数，建立多项式的链式存储结构，计算这两个多项式的乘积，输出乘积多项式的全部非零系数及对应的指数到另一字符文件中。

要求输入输出字符文件中的数据格式自拟；编程语言采用C/C++。

实验目的：掌握单向链表的基本操作以及基于链表的多项式加法与乘法。

数据结构设计简要描述：

采用带附加头结点单向链表；每个结点包括两个整型的数据域，分别为多项式的非零系数和对应指数，以及一个指针域。

算法设计简要描述：

实现两个多项式的乘法，必须先实现两个多项式的加法。同时考虑到输入数据为非升序指数排列，应增加排序算法。所以依次实现按指数排序的排序算法，多项式加法，多项式乘法。

两个多项式相加算法类似于两个升序表的归并算法，算法考虑利用两个源多项式结点生成和式的结点，但要求两个源多项式结点按指数e升序连接，算法结束后，两个源多项式成为空链表。

两个多项式乘法算法原理性如下：

R(x)=0;

for(i=0; i<n; i++)

{ T(x)=P(x)cix^di; //T(x)结点数与P(x)相同

R(x)=R(x)+T(x); //利用加法算法

}

输入/输出设计简要描述：

本实验采取文件输入/输出形式，输入文件input.txt输出文件output.txt。其中input.txt的数据以空格(或CR或TAB)分隔的若干不等于0的整数，直到输入0时停止输入，按整数输入次序建立结点并顺序连接结点，建立多项式链表ha。再空格(或CR或TAB)分隔的若干不等于0的整数，直到输入0时停止输入，按整数输入次序建立结点并顺序连接结点，建立多项式链表hb。测试数据总共有互不相同的三组。

编程语言说明：

使用CodeBlocks编程。 主要代码采用C语言实现 ；动态存储分配采用C++的new和delete操作符实现；输入与输出采用C++的cin和cout流；程序注释采用C/C++规范。

主要函数说明：

Polyn create() //按照先入先出发创建顺序表，即先输入的数据存储在前面

void print(Polyn h) //打印出多项式

void upsort(Polyn h) //按照指数升序排列，默认输入数据没有相同的指数

Polyn add(Polyn h1, Polyn h2)//多项式加法的实现

Polyn mul(Polyn hp, Polyn hq)//多项式减法的实现

程序测试简要报告：

输入数据存在于input.txt中，现按照测试样例拆分分析：

1. **测试实例1**

***程序输入***

2 2

4 4

0 0

5 5

3 3

0 0

***程序输出***

case1 :

hp排序后：

2 2

4 4

hp排序后：

3 3

5 5

hp\*hq的结果:

6 5

22 7

20 9***结论***

程序输出结果与期望输出结果相符。

1. **测试实例2**

***程序输入***

3 3

2 2

1 1

0 0

7 7

8 8

9 9

0 0

***程序输出***

case2 :

hp排序后：

1 1

2 2

3 3

hp排序后：

7 7

8 8

9 9

hp\*hq的结果:

7 8

22 9

46 10

42 11

27 12

***结论***

程序输出结果与期望输出结果相符。

1. **测试实例3**

***程序输入***

1 1

3 3

2 2

4 4

6 6

0 0

7 7

9 9

2 2

5 5

0 0

***程序输出***

case3 :

hp排序后：

1 1

2 2

3 3

4 4

6 6

hp排序后：

2 2

5 5

7 7

9 9

hp\*hq的结果:

2 3

4 4

6 5

13 6

10 7

34 8

34 9

30 10

76 11

27 12

78 13

54 15

***结论***

程序输出结果与期望输出结果相符。

源程序代码：

// 链表实现两个多项式乘法

#include <iostream>

#include <fstream>

using namespace std;

#define CreateNode(p) p=(Polyn)malloc(sizeof(PNode))

#define DeleteNode(p) free((void \*)(p))

//结构体的定义

typedef struct Polynode

{

int c; //系数

int e; //指数

struct Polynode \*next; //指针

}PNode, \*Polyn;

ifstream is("input.txt"); //从文件input.txt读取数据

ofstream os("output.txt"); //从文件output.txt写入程序运行结果数据

//按照先入先出发创建顺序表，即先输入的数据存储在前面

Polyn create()

{

Polyn last, p;

Polyn h = new PNode; h->next = NULL;

last = h;

int c, e;

while(is >> c >> e && c!= 0)

{

p = new PNode;

p->c = c;

p->e = e;

last->next = p;

last = p;

}

last->next = NULL;

return h;

}

//打印出多项式

void print(Polyn h)

{

Polyn p = h->next;

while(p)

{

os << p->c << " " << p->e << endl;

p = p->next;

}

}

//按照指数升序排列，默认输入数据没有相同的指数

void upsort(Polyn h)

{

Polyn q, s, pr, p;

q = h->next;

h->next = NULL; //断开头结点

while(q)

{

s = q;q = q->next; //摘下节点

pr = h; p = h->next; //初始化主从动指针

while(p && p->e < s->e )

{

pr = p;

p = p->next;

}

//following insert node \*s

pr->next = s;

s->next = p;

}

}

/\*

两个多项式相加算法类似于两个升序表的归并算法；

以下算法考虑利用两个源多项式结点生成和式的结点；

算法要求两个源多项式结点按指数e升序连接；

算法结束后，两个源多项式成为空链表。

\*/

Polyn add(Polyn h1, Polyn h2)//函数返回和式头指针

{

Polyn p1, p2, p3, h, p;

p1 = h1->next; p2 = h2->next;

CreateNode(h); p3 = h;//h为和式附加头结点指针

while(p1 && p2)

{

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

{

p = p1;

p1 = p1->next;

}else if(p2->e < p1->e)

{

p = p2;

p2 = p2->next;

}else//p1->e==p2->e为真时

{

p1->c = p1->c + p2->c;

if( p1->c == 0)//p1->c==0应删除两个结点

{

p = p1; p1 = p1->next; DeleteNode(p);

p = p2; p2 = p2->next; DeleteNode(p);

continue;

}

p = p2; p2 = p2->next; DeleteNode(p);

p = p1; p1 = p1->next;

}

p3->next = p; p3 = p;//插入\*p结点至和式末尾

}

if(p1)

p3->next = p1;

else if(p2)

p3->next = p2;

else

p2->next = NULL;

return h;

}

/\*

//原理性算法如下

R(x)=0;

for(i=0; i<n; i++)

{ T(x)=P(x)cixdi; //T(x)结点数与P(x)相同

R(x)=R(x)+T(x); //利用加法算法

}

\*/

Polyn mul(Polyn hp, Polyn hq)

{

Polyn hr, ht, p, q, pt;

CreateNode(hr); hr->next = NULL;//R(x)=0;

CreateNode(ht); ht->next = NULL;//T(x)=0;

q = hq->next;

while(q)//实现for(i=0; i<n; i++)

{

pt = ht;

p = hp->next;

while(p)

{

//以下实现T(x)=P(x)cixdi;

CreateNode(pt->next);

pt = pt->next;

pt->c = p->c \* q->c;

pt->e = p->e + q->e;

p = p->next;

}

pt->next = NULL;

q = q->next;

//以下实现R(x)=R(x)+T(x)

p = add(hr, ht);

DeleteNode(hr);

hr = p;

}

DeleteNode(ht);

return hr;

}

int main()

{

int T = 3; //测试样例次数

while(T--)

{

Polyn hp, hq, hmul;

hp = create();

hq = create();

upsort(hp);

upsort(hq);

os << "case" << 3-T << " :"<< endl;

os << "hp排序后：" << endl;

print(hp);

os << "hp排序后：" << endl;

print(hq);

hmul = mul(hp, hq);

os << "hp\*hq的结果:" << endl;

print(hmul);

}

return 0;

}