lttzz / **Course-Design**

Branch: **master** **Course-Design** / T3 / **3.cpp** Find file Copy path

lttzz 多项式(bug未修复) 54b8f88 2 days ago

1 contributor

198 lines (175 sloc) 4.56 KB

```
/*************************************************************************
        > File Name:            3.cpp
        > File Category: Course Design
        > Author:               lttzz
        > Mail:                         3344517687@qq.com
 ************************************************************************/

 /*
1 1 2 2 3 3  1 1 1 1 2 2 3 3  1 1 0 0
5 2 7 9 9 8 0 0

4x1+4x2+6x3
5x2+7x9+9x8
list1+list2 = 4x1+9x2+6x3+7x9+9x8
*/

//缺陷: 当第一个多项式为0的时候会出错
#include <iostream>
#include <cstdio>
using namespace std;

struct Node
{
        int xn, xe;
        struct Node *next;
};

void addNode(struct Node* &head, struct Node* &tail, int a, int b);
void readDate(struct Node* &head, struct Node* &tail);
void showList(struct Node *head);
void mergeNode(struct Node *head);
void mergeList(struct Node* head1, struct Node* tail1, struct Node* head2);
void copyList(struct Node* &bkhead, struct Node* &bktail, struct Node* &head, struct Node* &tail);
void initList(struct Node* &head, struct Node* &tail, struct Node* &bkhead, struct Node* &bktail);

/**
 * 没有用常见思路去直接把两个多项式加起来,而是简单封装了一下复用mergeNode()函数
 * 具体做法是将复制一份另个多项式,mergeNode()对于复制后的链表进行操作
 * 其中一个的尾指针指向另一个的首节点,显然此时做多项式的合并操作与将两个多项式相加结果相同
 */
int main(void)
{
        //freopen("./in.txt", "r", stdin);

        struct Node *head1 = NULL, *tail1 = NULL;
        struct Node *bkhead1 = NULL, *bktail1 = NULL;
        initList(head1, tail1, bkhead1, bktail1);


        struct Node *head2 = NULL, *tail2 = NULL;
        struct Node *bkhead2 = NULL, *bktail2 = NULL;
        initList(head2, tail2, bkhead2, bktail2);


        mergeList(bkhead1, bktail1, bkhead2);
        cout << "list1+list2 = ";
        showList(bkhead1);

        return 0;
}

/**
```

```
 * 链表添加节点
 * @param head 头指针
 * @param tail 尾指针
 * @param a     多项式系数
 * @param b     多项式指数
 */
void addNode(struct Node* &head, struct Node* &tail, int a, int b)
{
        struct Node *newnode = new struct Node;
        newnode->xn = a;
        newnode->xe = b;
        newnode->next = NULL;

        if (NULL == head)
        {
                head = tail = newnode;
        }
        else
        {
                tail->next = newnode;
                tail = newnode;
        }
}

/**
 * 读取数据以"0 0"结束,并调用addNode()函数,将读取到的数字插入链表尾部
 */
void readDate(struct Node* &head, struct Node* &tail)
{
        int a, b;               //所以说第一个多项式是0,就gg了
        while (cin >> a >> b)
        {
                if (0 == a && 0 == b)
                {
                        break;
                }

                addNode(head, tail, a, b);
        }
}

/**
 * 输出链表,对于加号的输出,用flag标识是否为链表第一个有效元素,若是则不输出加号并将标志置否,否则输出加号和数据域
 * 0x的n次方此处不做有效元素处理
 */
void showList(struct Node *head)
{
        bool flag = true;
        struct Node *P;
        for (P = head; P != NULL; P = P->next)
        {
                if (0 == P->xn)
                {
                        continue;
                }
                if (!flag)
                {
                        cout << "+";
                }
                else
                {
                        flag = false;
                }
                if (P->xn < 0)
                {
                        cout << "(" << P->xn << "x" << P->xe << ")";
                }
                else
                {
                        cout << P->xn << "x" << P->xe;
                }
        }
        cout << endl;
}
```

```
137
138 /**
139  * 暴力循环合并同类项同时释放被合并项的内存
140  */
141 void mergeNode(struct Node *head)
142 {
143         struct Node *cur1, *cur2, *bk;
144
145         for (cur1 = head; cur1 != NULL; cur1 = cur1->next)
146         {
147                 bk = cur2 = cur1;
148                 for (cur2 = cur2->next; cur2 != NULL; cur2 = cur2->next)
149                 {
150                         if (cur1->xe == cur2->xe)
151                         {
152                                 cur1->xn = cur1->xn + cur2->xn;
153                                 bk->next = bk->next->next;
154                                 delete cur2;
155                         }
156                         else
157                         {
158                                 bk = cur2;
159                         }
160                 }
161         }
162 }
163
164 /**
165  * 合并两个多项式
166  * 基于开头所写思路,两个多项式相加等效与两个多项式"拼起来"后做合并同类项操作
167  * @param head1 调用mergeNode()时用
168  */
169 void mergeList(struct Node* head1, struct Node* tail1, struct Node* head2)
170 {
171         tail1->next = head2;
172         mergeNode(head1);
173 }
174
175 /**
176  * 复制一份两个多项式
177  * @param bkhead 复制后头指针
178  * @param bktail 复制后尾指针
179  */
180 void copyList(struct Node* &bkhead, struct Node* &bktail, struct Node* &head, struct Node* &tail)
181 {
182         for (struct Node *cur = head; cur != NULL; cur = cur->next)
183         {
184                 addNode(bkhead, bktail, cur->xn, cur->xe);
185         }
186 }
187
188 /**
189  * 对多项式的操作全部写在了这个函数内,以缩短主函数长度
190  */
191 void initList(struct Node* &head, struct Node* &tail, struct Node* &bkhead, struct Node* &bktail)
192 {
193         readDate(head, tail);
194         copyList(bkhead, bktail, head, tail);
195         mergeNode(head);
196         showList(head);
197 }
```