实验二　线性表

1. 实验目的
2. 掌握线性表的逻辑结构；
3. 顺序表和链表的基本操作的实现；
4. 掌握利用C/C++编程语言实现数据结构的编程方法；
5. 通过上机时间加强利用数据结构解决实际应用问题的能力；
6. 实验相关知识
7. 线性表的顺序存储结构的实现；
8. 线性表的链式存储结构的实现；
9. 线性表的应用——一元多项式的表示及相加。
10. 实验内容与要求
11. 利用顺序表或链表表示两个一元多项式，并完成两多项式的乘法运算。按指数的升序输入第一个一元多项式polya各项的指数和系数，且以输入0 0结束，按指数的升序输入第一个一元多项式polyb各项的指数和系数。输出两一元多项式乘积的一元多项式polyc，并进行算法时间复杂度的分析

例1：

【测试用例】

|  |  |  |
| --- | --- | --- |
| 输入 | 0 2 1 3 4 -3 0 0  2 4 6 6 0 0 | 0 2 1 3 2 -3 0 0  1 1 2 4 0 0 |
| 输出 | 2 8 3 12 7 18 10 -18 | 1 2 2 11 3 9 4 -12 |

【设计要求】在给出的代码素材polymul.cpp文件中补充完整以下函数，实现多项式相乘的计算。

void polyadd(Polylist poly,int coef,int exp)

void polymul(Polylist polya, Polylist polyb,Polylist polyc)

1. 利用循环单链表求解约瑟夫环问题（即n个人围成一个圆圈，然后从第一个人开始，按：1,2,3,…,m报数，数到m的人出圈，并有出圈者的下一个人重新开始报数，数到m又要出圈，如此类推，直到所有人都出圈，打印出圈的次序，其中n和m为输入数据）

【测试用例】

|  |  |  |
| --- | --- | --- |
| 输入 | 9 3 | 6 2 |
| 输出 | 8 7 3 5 9 6 2 1 4 | 2 4 6 3 1 5 |

【设计要求】在给出的代码素材josephus.cppp文件中补充完整main函数，求解约瑟夫环中出列的人的编号。

1. 程序代码及运行结果

1、【程序代码】

void polyadd(Polylist poly,int coef,int exp)

{

Polynode \*pre;

pre=(Polynode \*)malloc(sizeof(Polynode));

while(poly->next!=NULL&&poly->exp<exp)

{

if(exp==poly->next->exp)

{

poly->next->exp=exp;

poly->next->coef+=coef;

if(poly->next->coef==0)

{

pre=poly->next;

poly->next=pre->next;

free(pre);

}

return;

}

poly=poly->next;

}

if(coef==0)

{

return;

}

pre->coef=coef;

pre->exp=exp;

pre->next=poly->next;

poly->next=pre;

}

void polymul(Polylist polya, Polylist polyb,Polylist polyc)

{

Polylist pre;

pre=polyb;

int coef,exp;

while(polya->next!=NULL)

{

polya=polya->next;

while(polyb->next!=NULL)

{

polyb=polyb->next;

exp=polya->exp+polyb->exp;

coef=polya->coef\*polyb->coef;

polyadd(polyc,coef,exp);

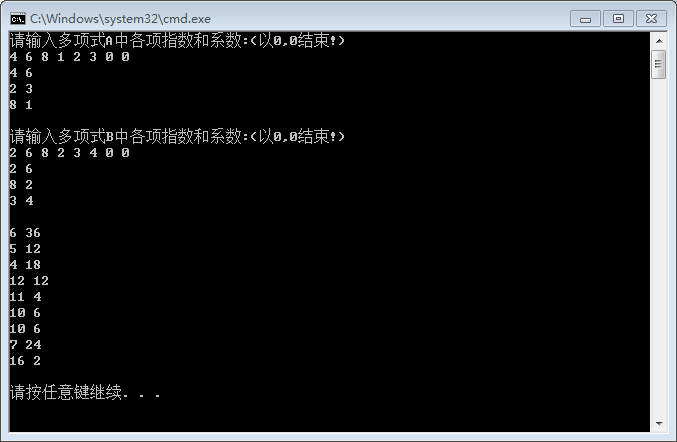
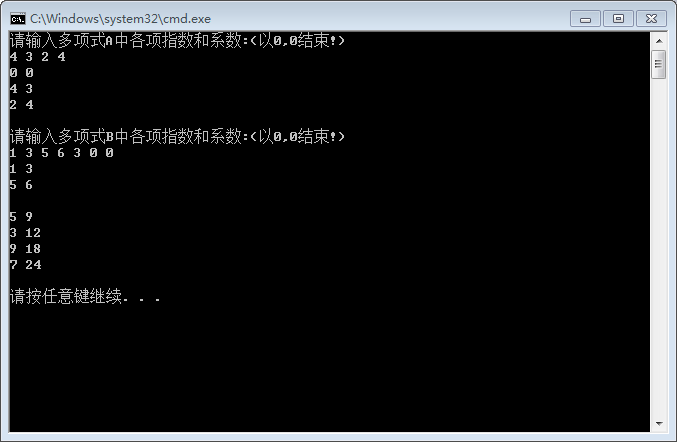
}

polyb=pre;

}

}

【运行结果】



【算法时间复杂度分析】

T(n)=O(n²)，多项式A与多项式B相乘需扫描多项式A长度\*多项式B长度次

2、【程序代码】

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

#define OK 1

#define ERROR 0

#define TRUE 1

#define FALSE 0

typedef struct Node /\*结点类型定义\*/

{

int data;

struct Node \* next;

}Node, \*link; /\* LinkList为结构指针类型\*/

int main()

{

int n, m, count;

link joselink, current, s;

scanf("%d %d", &n, &m);

joselink = (Node \*)malloc(sizeof(Node));

current = joselink;

joselink->next = NULL;

joselink->data = 1;

while (current->data < n)

{

s = (Node \*)malloc(sizeof(Node));

s->data = current->data + 1;

current->next = s;

current = current->next;

}

current->next = joselink;

link r;

count = 0;

while (current->next != current)

{

count++;

if (count == m)

{

r = current->next;

current->next = r->next;

printf("%d ", r->data);

free(r);

count = 0;

}

else

{

current = current->next;

}

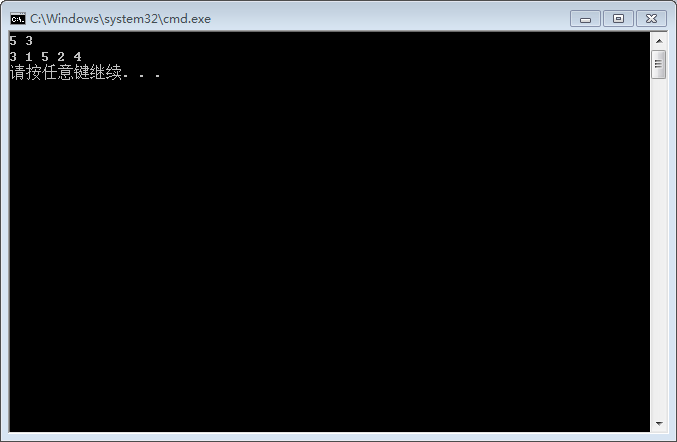
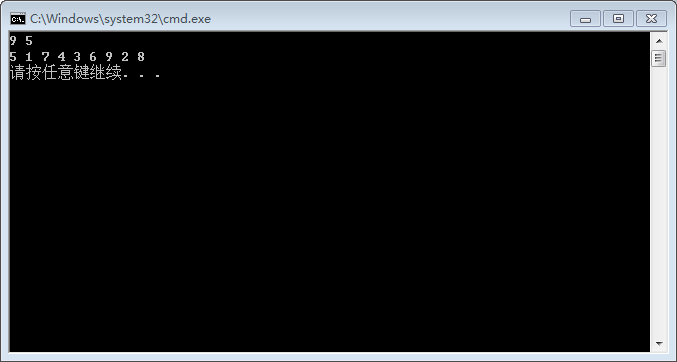
}

printf("%d\n", current->data);

return 0;

}

【运行结果】



五、实验心得体会

因为线性表属于新内容，所以对于我来说掌握有点慢，因为个人在前两节也没学好，所以在线性表上也下了比较大得功夫，特别在指针方面，这也是因为差不多忘记上学期的知识了，在之后一定要经常自己打代码练习不然就会忘记。