题目：

设有两个一元多项式A(x),B(x)，请完成运算A(x)+B(x)、A(x)-B(x)，要求多项式采用链表结构进行存储，多项式可以按照升幂（降幂）形式输入对应的系数、指数项，但对输入输出格式不做要求。

代码：

#include<stdio.h>

#include<malloc.h>

#include<stdlib.h>

typedef struct poly

{

int a;//系数

int b;//指数

struct poly\* next;

}poly;

void create(poly\*,FILE\*);//创建链表

void add(poly\*, poly\*, poly\*);//加法运算

void sub(poly\*, poly\*, poly\*);//减法运算

void write(poly\*, FILE\*);//写入文件

int main()

{

//建立多项式链表A

poly\* A = (poly\*)malloc(sizeof(poly));

if (A == NULL)

{

printf("error\n");

exit(-1);

}

A->next = NULL;

FILE\* fp\_1 = fopen("A", "r+");

if (fp\_1 == NULL)

{

printf("A error\n");

exit(-1);

}

create(A,fp\_1);

//建立多项式链表B

poly\* B = (poly\*)malloc(sizeof(poly));

if (B == NULL)

{

printf("error\n");

exit(-1);

}

B->next = NULL;

FILE\* fp\_2 = fopen("B", "r+");

if (fp\_2 == NULL)

{

printf("B error\n");

exit(-1);

}

create(B, fp\_2);

//进行加法运算，C是结果多项式

poly\* C = (poly\*)malloc(sizeof(poly));

if (C == NULL)

{

printf("error\n");

exit(-1);

}

C->next = NULL;

//选择加减法

char s;

while (1)

{

printf("请选择‘+’或‘-’：");

scanf("%c", &s);

getchar();

if (s == '+')

{

add(A, B, C);

break;

}

else if (s == '-')

{

sub(A, B, C);

break;

}

else

printf("error\n\*\*\*\*请重新输入\*\*\*\*\n");

}

//写入文件

FILE\* fp = fopen("C", "w+");

if (fp == NULL)

{

printf("C error\n");

exit(-1);

}

write(C, fp);

return 0;

}

//创建链表

void create(poly\* phead,FILE\* fp)

{

poly\* ptail = phead;

//读入文件数据

while(1)

{

//建立新结点

poly\* pnew = (poly\*)malloc(sizeof(poly));

if (pnew == NULL)

{

printf("error\n");

exit(-1);

}

pnew->next = NULL;

//读取数据

fscanf(fp, "%d %d\n", &pnew->a, &pnew->b);

ptail->next = pnew;

ptail = pnew;

//系数为0时跳出循环 结束读取

if (pnew->a == 0)

{

fclose(fp);

break;

}

}

return;

}

//加法运算

void add(poly\* A, poly\* B, poly\* C)

{

poly\* pA = A->next;

poly\* pB = B->next;

poly\* pC = C;

while (1)

{

//建立C的新结点

poly\* pnew = (poly\*)malloc(sizeof(poly));

if (pnew == NULL)

{

printf("error\n");

exit(-1);

}

pnew->a = 0;

pnew->b = 0;

pnew->next = NULL;

//分情况讨论AB系数指数大小：

//AB系数大小是否为0 (即A或B是否结尾)

if (pA->a == 0 && pB->a != 0)

{

pnew->a = pB->a;

pnew->b = pB->b;

pB = pB->next;

}

else if (pA->a != 0 && pB->a == 0)

{

pnew->a = pA->a;

pnew->b = pA->b;

pA = pA->next;

}

else if (pA->a == 0 && pB->a == 0)

{

break;

}

//AB指数大小比较

else if (pA->b > pB->b)

{

pnew->a = pA->a;

pnew->b = pA->b;

pA = pA->next;

}

else if (pA->b < pB->b)

{

pnew->a = pB->a;

pnew->b = pB->b;

pB = pB->next;

}

else if (pA->b == pB->b)

{

pnew->a = pA->a + pB->a;

pnew->b = pA->b;

pA = pA->next;

pB = pB->next;

}

pC->next = pnew;

pC = pnew;

}

return;

}

//减法运算

void sub(poly\* A, poly\* B, poly\* C)

{

poly\* pA = A->next;

poly\* pB = B->next;

poly\* pC = C;

while (1)

{

//建立C的新结点

poly\* pnew = (poly\*)malloc(sizeof(poly));

if (pnew == NULL)

{

printf("error\n");

exit(-1);

}

pnew->a = 0;

pnew->b = 0;

pnew->next = NULL;

//分情况讨论AB系数指数大小：

//AB系数大小是否为0 (即A或B是否结尾)

if (pA->a == 0 && pB->a != 0)

{

pnew->a = 0 - (pB->a);

pnew->b = pB->b;

pB = pB->next;

}

else if (pA->a != 0 && pB->a == 0)

{

pnew->a = pA->a;

pnew->b = pA->b;

pA = pA->next;

}

else if (pA->a == 0 && pB->a == 0)

{

break;

}

//AB指数大小比较

else if (pA->b > pB->b)

{

pnew->a = pA->a;

pnew->b = pA->b;

pA = pA->next;

}

else if (pA->b < pB->b)

{

pnew->a = 0 - (pB->a);

pnew->b = pB->b;

pB = pB->next;

}

else if (pA->b == pB->b)

{

pnew->a = (pA->a) - (pB->a);

pnew->b = pA->b;

pA = pA->next;

pB = pB->next;

}

pC->next = pnew;

pC = pnew;

}

return;

}

//写入文件

void write(poly\* C, FILE\* fp)

{

poly\* ptail = C->next;

int flag;

flag = 0;

while (ptail != NULL)

{

while (ptail->a == 0)

{

ptail = ptail->next;

}

if (flag == 0)

{

fprintf(fp, "%d", ptail->a);

fprintf(fp, "x^%d", ptail->b);

flag = 1;

ptail = ptail->next;

}

if (flag == 1)

{

while (ptail->a == 0)

ptail = ptail->next;

if (ptail->a > 0)

fprintf(fp, "+%d", ptail->a);

else if (ptail->a < 0)

fprintf(fp, "%d", ptail->a);

if(ptail->b == 1)

fprintf(fp, "x");

else if (ptail->b != 0)

fprintf(fp, "x^%d", ptail->b);

ptail = ptail->next;

}

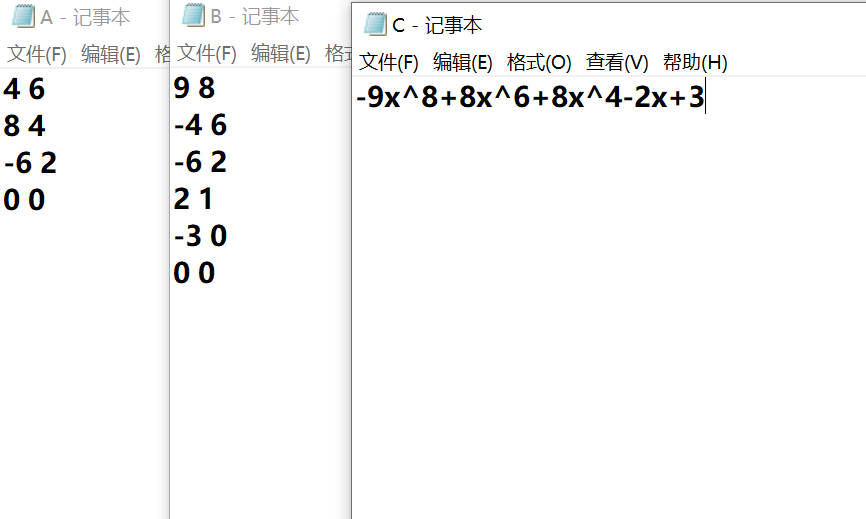
}

fclose(fp);

return;

}

运行结果：



总结：

采用链表存储是一个比较好的结构，考虑到多项式习惯用降幂排列，因此建立多项式时，可以按照降幂表示（升幂也是一样的）。两个多项式中指数相同的项对应系数相加，若相加的和不为零，则构成相加结果多项式中的一项，所有指数不相同的项均复制到相加结果多项式中。具体算法过程是：遍历第一个多项式，在遍历过程中，处理每一个单项：遍历第二个多项式；比较两个单项式的指数；若指数相同，则两个单项式的系数相加，并形成新的单项式添加到运算结果列表中；若指数不相同，则两个单项式都添加到运算结果列表中。