哈尔滨工业大学计算机科学与技术学院

实验报告

课程名称：数据结构与算法

课程类型：必修

实验项目：线性表的链式存储结构与应用

实验题目：一元多项式的代数

实验日期：2021.01.01

班级：1803501

学号：L170300901

姓名：卢兑玩

设计成绩	报告成绩	指导老师

一、实验目的

1. 掌握线性表顺序存储结构的特点及线性表在顺序存储结构中各种基本操作的实现。
2. 掌握线性表链式存储结构的特点及线性表在链式存储结构中各种基本操作的实现。
3. 重点巩固和体会线性表在链式存储结构上的各种操作和应用。

二、实验要求及实验环境

实验要求

1.能够输入多项式(可以按各项的任意输入顺序,建立按指数降幂排列的多项式)和输出多项式 （按指数降幂排列）,以文件形式输入和输出，并显示。

2.能够计算多项式在某一点 x=x0 的值，其中 x0 是一个浮点型常量，返回结果为浮点数。

3.能够给出计算两个多项式加法、减法、乘法和除法运算的结果多项式，除法运算的结果包括商 多项式和余数多项式。

4.要求尽量减少乘法和除法运算中间结果的空间占用和结点频繁的分配与回收操作。(提示：利 用循环链表结构和可用空间表的思想，把循环链表表示的多项

式返还给可用空间表，从而解决上述问题)。

实验环境

Windows10, Codeblocks20.03

三、设计思想（本程序中的用到的所有数据类型的定义，主程序的流程图及各程序模块之间的调用关系）

```
typedef struct poly_node* pn; //구조체 연결리스트 포인트형 정의(定义指针型)
typedef struct poly_node {
    double coef;
    int expon;
    pn next;
```

}poly_node;

void Delete(pn head);//释放链表

void Create(double c, int e, pn head); // 输入多项式的函数

void Print(pn head); // 把链表打印多项式

float Insert(float x0, pn head); // 返回，输入 x0 时的函数返回值

void Sum(pn head_A, pn head_B, pn head_C); //多项式 A 跟多项式 B 进行加法运算之后的结果多项式保存到链表 C

pn Sub(pn head_A, pn head_B); //返回多项式 A-多项式 B 的结果多项式

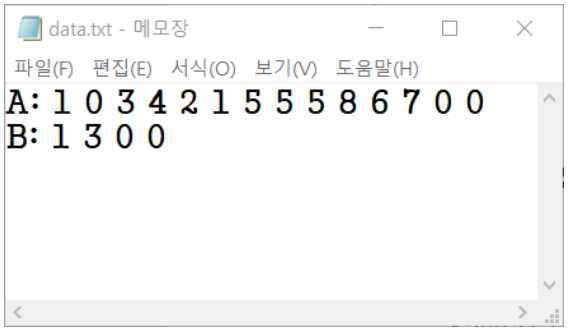
void Multiply(pn head_A, pn head_B, pn head_C);//多项式 A 跟 B 进行乘法运算之后的结果多项式保存到链表 C

void Divison(pn head_A, pn head_B, pn head_C, pn head_D); //打印多项式 A 跟多项式 B 进行除法的结果

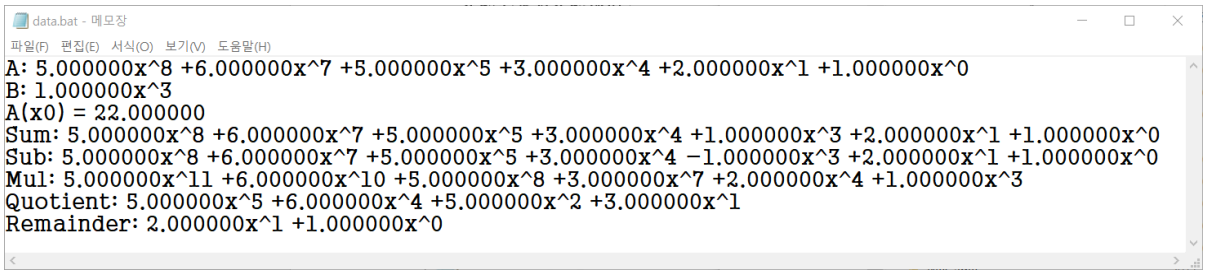
void Read(pn head_A, pn head_B); //从文件读取多项式 A,B

void Write(pn head_A, pn head_B, float x0); //结果保存到文件里

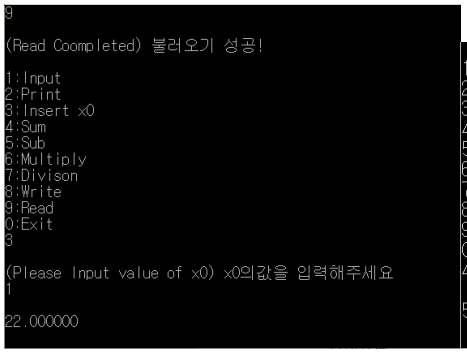
四、测试结果

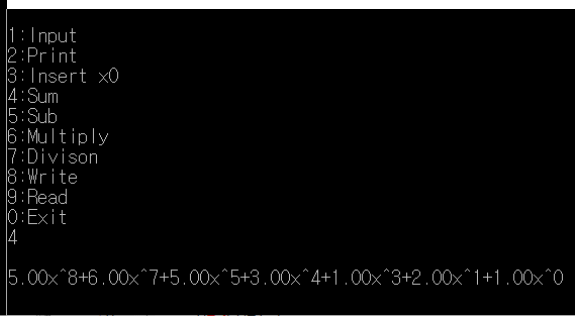


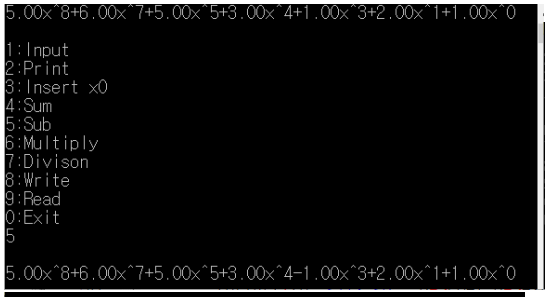
输入文件

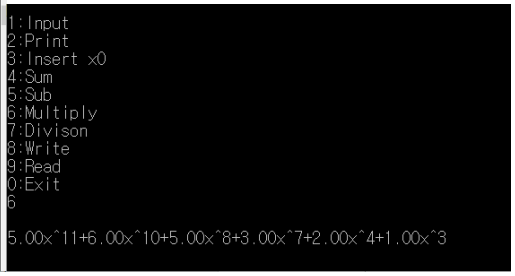


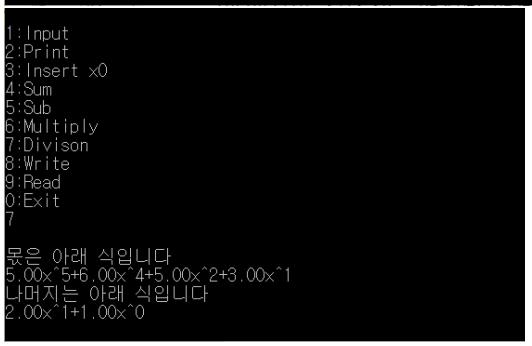
打印文件











五、经验体会与不足

通过本次实验，可以知道我的编程能力并能练习编程方法及思路。比如说把数学上的除法运算写成 c 语言这样的过程，我觉得并不简单。用循环过程当中要判断的部分不少，而且占用空间的问题也不少了。经过利用链表的过程，能知道程序什么时候会发生关于内存空间的问题或者无穷循环的问题。觉得不足的部分也不少，首先用 codeblocks 写这次实验代码，再 codeblocks 上运行基本上没什么问题，不过在 VS 上运行总是发生一些问题。而且我的代码里面乱的部分也不少，一个运算过程当中利用两三个不同的函数。可能是这个乱码引起报错的。觉得优化代码之后可以写成一个函数。要练习编程解决数据，算法等等的问题

六、附录：源代码（带注释）

```
#include <stdio.h>

#include <stdlib.h>

#include <math.h>
```

```
#pragma warning(disable : 4996)
#define base "D://data.bat"
#define baseR "D://data.txt"

typedef struct poly_node* pn; //구조체 연결리스트 포인트형 정의(定义指针型)
typedef struct poly_node {
    double coef;
    int expon;
    pn next;
}poly_node;



void Delete(pn head);
void Create(double c, int e, pn head); //연결리스트로 다항식을 입력（输入多项式的函数）
void Print(pn head); //연결리스트를 다항식의 형식으로 프린트（把链表打印多项式）
float Insert(float x0, pn head);    //x0 값의 입력으로 나오는 결과（返回，输入 x0 时的函数返回值）
void Sum(pn head_A, pn head_B, pn head_C); //A 다항식과 B 다항식의 덧셈결과를 C 연결리스트 항목으로 저장
pn Sub(pn head_A, pn head_B); //A 다항식에서 B 다항식을 뺀값을 되돌려줌
void Multiply(pn head_A, pn head_B, pn head_C); //A 다항식과 B 다항식의 곱셈 결과를 C 연결리스트에 저장
void Divison(pn head_A, pn head_B, pn head_C, pn head_D);   //A 다항식을 B 다항식으로 나눈결과를 도출
void Read(pn head_A, pn head_B);  //파일에서부터 A 다항식과 B 다항식을 읽어옴 从文件读取多项式 A,B
void Write(pn head_A, pn head_B, float x0); //파일로 A 다항식과 B 다항식을 저장 结果保存到文件里
```

```
void Create(double c, int e, pn head) //연결리스트로 다항식을 입력（输入多项式的函数）
{
    int flag = 1;
    pn p = (pn)malloc(sizeof(poly_node));
    pn x = head, y;
    p->coef = c;
    p->expon = e;
    while (head->next && (head->next->expon >= p->expon))
    {
        head = head->next;
        if (head->expon == p->expon)
        {
            head->coef += p->coef;
            flag = 0;
            break;
        }
    }
    if (flag)
    {
        p->next = head->next;
        head->next = p;
    }
    while (x->next)
    {
        if (x->next->coef == 0)
```

```
        {
            y = x->next;
            x->next = y->next;
            y->expon = NULL;
        }
        if (x->next)
            x = x->next;

    }
}
void Print(pn head) //연결리스트를 다항식의 형식으로 프린트（把链表打印多项式）
{
    do {
        if (head->next&&head->next->coef<0)
            printf("-");
        if (head->next)
            head = head->next;
        printf("%.2lfx^%d", fabs(head->coef), head->expon);
        if (head->next)
        {
            if (head->next->coef>0)
                printf("+");
        }
    } while (head->next);
}
float Insert(float x0, pn head)    //x0 값의 입력으로 나오는 결과（返回，输入 x0 时的函数返回值）
{
    float sum = 0;
```

```
    do {
        head = head->next;
        sum += (float)head->coef*(float)pow(x0, head->expon);
    } while (head->next);
    return sum;
}
void Sum(pn head_A, pn head_B, pn head_C) //A 다항식과 B 다항식의 덧셈결과를 C 연결리스트 항목으로 저장
                                          //多项式 A 跟多项式 B 进行加法运算之后的结果多项式保存到链表 C
{
    int sum;
    head_A = head_A->next;
    head_B = head_B->next;
    while (head_A)
    {
        Create(head_A->coef, head_A->expon, head_C);
        head_A = head_A->next;
    }
    while (head_B)
    {
        Create(head_B->coef, head_B->expon, head_C);
        head_B = head_B->next;
    }
    pn x = head_C, y;
    while (x->next)
    {
        if (x->next->coef == 0)
        {
```

```
            y = x->next;
            x->next = y->next;
            y->expon = NULL;
        }
        if (x->next)
            x = x->next;

    }
}
pn Sub(pn head_A, pn head_B) //A 다항식에서 B 다항식을 뺀값을 되돌려줌
                             //返回多项式 A-多项式 B 的结果多项式
{
    pn head_C = (pn)malloc(sizeof(poly_node));
    pn head_D = (pn)malloc(sizeof(poly_node));
    while (head_B->next)
    {
        head_B = head_B->next;
        Create(-(head_B->coef), head_B->expon, head_C);
    }
    Sum(head_A, head_C, head_D);
    Delete(head_C);
    return head_D;
}
void Multiply(pn head_A, pn head_B, pn head_C) //A 다항식과 B 다항식의 곱셈 결과를
C 연결리스트에 저장
                                               ////多项式 A 跟多项式 B 进行乘法运
算之后的结果多项式保存到链表 C

{
```

```
    pn p = head_B;
    while (head_A->next)
    {
        head_A = head_A->next;
        head_B = p;
        while (head_B->next)
        {
            head_B = head_B->next;
            Create(head_A->coef*head_B->coef,   head_A->expon   +   head_B->expon,
head_C);
        }
    }
}
void Divison(pn head_A, pn head_B, pn head_C, pn head_D)   //A 다항식을 B 다항식으로
나눈결과를 도출
                                                          //打印多项式 A 跟多项
式 B 进行除法的结果
{
    if (head_A->next&&head_B->next) {
        pn      head_U      =      (pn)malloc(sizeof(poly_node)),      head_Q      =
(pn)malloc(sizeof(poly_node));
        pn n = (pn)malloc(sizeof(poly_node)), m = head_A, k = head_B;
        pn ok = (pn)malloc(sizeof(poly_node));
        ok = head_U;
        head_A = head_A->next;
        while (head_A)
        {
            Create(head_A->coef, head_A->expon, head_U);
            head_A = head_A->next;
```

```
    }
    n = head_B->next; head_Q = head_U;
    head_U = head_U->next;
    while (head_U && (head_U->expon) >= (n->expon))
    {
        pn head_O = (pn)malloc(sizeof(poly_node));
        pn head_z = (pn)malloc(sizeof(poly_node));
        pn head_P = (pn)malloc(sizeof(poly_node));
        head_z = head_B;
        Create((head_U->coef) / (n->coef), (head_U->expon) - (n->expon), head_O);
        Create(head_O->next->coef, head_O->next->expon, head_C);
        Multiply(head_z, head_O, head_P);
        head_U = Sub(head_Q, head_P);
        head_Q = head_U;
        Delete(head_P);
        Delete(head_O);
        head_U = head_U->next;

    }
    while (head_U)
    {
        Create(head_U->coef, head_U->expon, head_D);
        head_U = head_U->next;
    }
}
else
    printf("(Please Input A&B) A 다항식과 B 다항식을 먼저 입력해주세요");
```

```
}

void Delete(pn head)    //안쓰는 연결리스트 초기화 및 삭제(공간절약) 释放链表
{
    pn temp = head;
    while (temp != NULL)
    {
        head = head->next;
        temp->next = NULL;
        temp->coef = 0;
        temp->expon = NULL;
        free(temp);
        temp = head;
    }
}
void Read(pn head_A, pn head_B)  //파일에서부터 A 다항식과 B 다항식을 읽어옴 从文件读取多项式A,B
{
    Delete(head_A->next), Delete(head_B->next);
    head_A->next = NULL, head_B->next = NULL;
    FILE *fp;
    double coef; char c;
    int expon, flag = 1;
    fp = fopen(baseR, "r");
    if (fp)
    {
        fscanf(fp, "A: ");
        while (!feof(fp))
        {
```

```
            fscanf(fp, "%lf %d ", &coef, &expon);
            Create(coef, expon, head_A);
            if (!coef && !expon)
                break;
        }
        fscanf(fp, "B: ");
        while (!feof(fp))
        {
            fscanf(fp, "%lf %d ", &coef, &expon);
            Create(coef, expon, head_B);
            if (!coef && !expon)
                break;
        }
        printf("(Read Coompleted) 불러오기 성공!");
    }
    else
        printf("(Read Failed) 불러오기 실패!");
    fclose(fp);
}
void Write(pn head_A, pn head_B, float x0) //파일로 A 다항식과 B 다항식을 저장 结果保存到文件里
{
    pn head_C = (pn)malloc(sizeof(poly_node));
    head_C->next = NULL; pn c = head_C;
    pn head_D = (pn)malloc(sizeof(poly_node));
    head_D->next = NULL; pn d = head_D;
    pn head_E = (pn)malloc(sizeof(poly_node));
    head_E->next = NULL; pn e = head_E;
    pn head_F = (pn)malloc(sizeof(poly_node));
```

```
head_F->next = NULL; pn f = head_F;
pn head_G = (pn)malloc(sizeof(poly_node));
head_G->next = NULL; pn g = head_G;
FILE *fp;
fp = fopen(base, "w");
fprintf(fp, "A: ");
pn a = head_A->next, b = head_B->next;
while (a)
{
    fprintf(fp, "%lfx^%d ", a->coef, a->expon);
    if (a->next)
    {
        if (a->next->coef >= 0)
            fprintf(fp, "+");
    }
    a = a->next;
}
fprintf(fp, "\n");
fprintf(fp, "B: ");
while (b)
{

    fprintf(fp, "%lfx^%d ", b->coef, b->expon);
    if (b->next)
    {
        if (b->next->coef >= 0)
            fprintf(fp, "+");
    }
    b = b->next;
```

```
}
fprintf(fp, "\n");
if (x0 != 0)
    fprintf(fp, "A(x0) = %f\n", Insert(x0, head_A));
else
    fprintf(fp, "please input x0\n", x0);
fprintf(fp, "Sum: ");
Sum(head_A, head_B, c);
c = head_C->next;
while (c)
{
    fprintf(fp, "%lfx^%d ", c->coef, c->expon);
    if (c->next)
    {
        if (c->next->coef >= 0)
            fprintf(fp, "+");
    }
    c = c->next;
}

fprintf(fp, "\nSub: ");
head_D = Sub(head_A, head_B);
d = head_D->next;
while (d)
{
    fprintf(fp, "%lfx^%d ", d->coef, d->expon);
    if (d->next)
    {
        if (d->next->coef >= 0)
```

```
            fprintf(fp, "+");
    }
    d = d->next;
}

Multiply(head_A, head_B, head_E);
e = head_E->next;
fprintf(fp, "\nMul: ");
while (e)
{
    fprintf(fp, "%lfx^%d ", e->coef, e->expon);
    if (e->next)
    {
        if (e->next->coef >= 0)
            fprintf(fp, "+");
    }
    e = e->next;
}

Divison(head_A, head_B, head_F, head_G);
fprintf(fp, "\nQuotient: ");
f = f->next;
while (f)
{
    fprintf(fp, "%lfx^%d ", f->coef, f->expon);
    if (f->next)
    {
        if (f->next->coef >= 0)
            fprintf(fp, "+");
```

```
    }
    f = f->next;
}
fprintf(fp, "\nRemainder: ");
g = g->next;
while (g)
{
    fprintf(fp, "%lfx^%d ", g->coef, g->expon);
    if (g->next)
    {
        if (g->next->coef >= 0)
            fprintf(fp, "+");
    }
    g = g->next;
}
if (head_C->next)
    Delete(head_C);
if (head_D->next)
    Delete(head_D);
if (head_E->next)
    Delete(head_E);
if (head_F->next)
    Delete(head_F);
if (head_G->next)
    Delete(head_G);
if (fp)
    printf("(Write Completed) 저장 성공!");
else
    printf("(Write Failed) 저장 실패!");
```

```
    fclose(fp);
}

int main()
{
    char sel, num;
    char x0[30] = { 0 }, exponv[30], coefv[30];
    pn head_A = (pn)malloc(sizeof(poly_node));
    pn head_B = (pn)malloc(sizeof(poly_node));
    head_A->next = NULL; head_A->expon = 0;
    head_B->next = NULL; head_B->expon = 0;
    do {
        pn head_C = (pn)malloc(sizeof(poly_node));
        pn head_D = (pn)malloc(sizeof(poly_node));
        head_C->next = NULL;
        head_D->next = NULL;
        printf("1:Input\n2:Print\n3:Insert
x0\n4:Sum\n5:Sub\n6:Multiply\n7:Divison\n8:Write\n9:Read\n0:Exit\n");
        scanf("%c", &num);
        getchar();
        printf("\n");
        switch (num) {
        case '1':
            printf("(Please Input A or B) 다항식 A or B?\n");
            scanf("%c", &sel);
            getchar();
            if ((sel == 'a' || sel == 'A') && head_A->next)
            {
                Delete(head_A->next);
```

```
            head_A->next = NULL;
        }
        else if ((sel == 'b' || sel == 'B') && head_B->next)
        {
            Delete(head_B->next);
            head_B->next = NULL;
        }
        if (!(sel == 'a' || sel == 'A' || sel == 'b' || sel == 'B'))
        {
            printf("(Please Input A or B) A 혹은 B를 입력해주세요");
            break;
        }
        do {

            printf("(Please Input Coefficient) 계수를 입력해주세요:");
            scanf("%s", &coefv);
            getchar();
            if (!atof(coefv))
                break;
            printf("(Please Input Exponent) 지수를 입력해주세요:");
            scanf("%s", &exponv);
            getchar();
            if (sel == 'A' || sel == 'a')
                Create((float)atof(coefv), atoi(exponv), head_A);
            else if (sel == 'B' || sel == 'b')
                Create((float)atof(coefv), atoi(exponv), head_B);
        } while (coefv);
        break;
    case '2':
```

```
        printf("(Please Input A or B) 다항식 A or B?\n");
        scanf("%c", &sel);
        getchar();
        if (sel == 'A' || sel == 'a'&&head_A->next)
            Print(head_A);
        else if (sel == 'B' || sel == 'b'&&head_B->next)
            Print(head_B);
        else
            printf("(Please Input A or B) A 혹은 B를 입력해주세요");
        break;
    case '3':
        if (head_A->next&&head_B->next)
        {
            printf("(Please Input value of x0) x0의값을 입력해주세요\n");
            scanf("%s", &x0);
            getchar();
            printf("\n%f", Insert((float)atof(x0), head_A));
        }
        else
            printf("(Please Input A) A를 입력해주세요");
        break;
    case '4':
        Sum(head_A, head_B, head_C);
        Print(head_C);
        break;
    case '5':
        head_C = Sub(head_A, head_B);
        Print(head_C);
        break;
```

```
case '6':
    Multiply(head_A, head_B, head_C);
    Print(head_C);
    break;
case '7':
    Divison(head_A, head_B, head_C, head_D);
    printf("몫은 아래 식입니다\n");
    Print(head_C);
    printf("\n 나머지는 아래 식입니다\n");
    Print(head_D);
    break;
case '8':
    if (head_A->next&&head_B->next)
        Write(head_A, head_B, (float)atof(x0));
    else
        printf("(Please Input A and B) A 와 B 를 입력해주세요");
    break;
case '9':
    Read(head_A, head_B);
    break;
case '0': num = 0; break;
default:
    printf("(Please Input 0~9 number) 0~9 사이의 숫자를 입력해주세요");
    break;
}
if (head_C->next)
    Delete(head_C);
if (head_D->next)
    Delete(head_D);
```

```
        printf("\n\n");
    } while (num);
    if (head_A->next)
        Delete(head_A);
    if (head_B->next)
        Delete(head_B);
    return 0;
}
```