哈尔滨工业大学计算学部

实验报告

课程名称：数据结构与算法

课程类型：专业核心基础课（必修）

实验项目：线性表的链式存储结构与应用

实验题目：一元多项式计算器

实验日期：2024 年 4 月 5 日

班级：22WL022

学号：2022110829

姓名：杨明达

设计成绩	报告成绩	任课老师
		张岩

一、实验目的

设计线性表的动态或静态链式存储结构，并实现一个一元多项式的计算器。

二、实验要求及实验环境

实验要求：以动态或者静态链表存储一元多项式，在此基础上按要求完成对一元多项式的运算。（为保证多项式的值的准确性，多项式的系数可以用分数表示，涉及到两个分数相除时，结果也可以用分数表示。）

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。其中，除法运算为选做。

3. 能够计算多项式在某一点 $x = x_0$ 的值，其中 x_0 是一个浮点型常量，返回结果为浮点数。要求，所设计的算法为线性时间算法。

4. 要求尽量减少乘除法运算中间结果空间占用和结点频繁分配与回收操作。

实验环境：Windows 11 && Visual Studio Code

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、核心算法的主要步骤）

1．逻辑设计

1.1 主程序的流程图

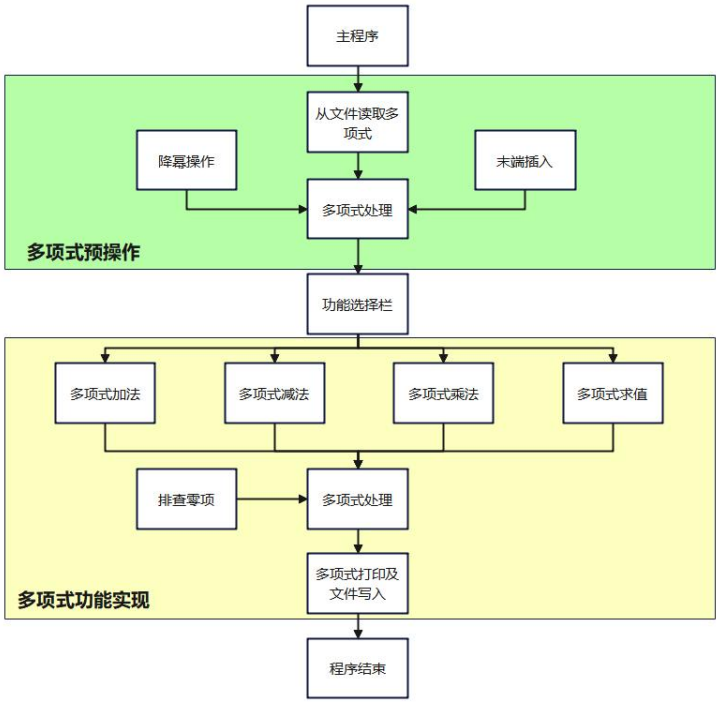


图 1 主程序的流程图

1.2 各程序模块的调用关系

（1）从文件读取多项式函数调用 FILE 相关函数、多项式结点的创建、多项式末端插入模块。

（2）多项式预操作中的多项式处理模块主要由多项式末端插入和多项式降幂排序实现。（本实验假设输入的多项式无同幂项和非法输入项）

（3）多项式降幂排序函数包含多项式各结点的链的删除和创建。

（4）多项式加法、减法和乘法函数调用了末端插入函数、多项式指数比较、多项式系数为零判断。

（5）多项式求值函数的核心算法由秦九韶算法实现。

（6）多项式文件写入函数调用 FILE 相关函数。

（7）主函数调用多项式文件读取、写入、多项式降幂排序、多项式打印、多项式加法、减法、乘法、求值函数。

1.3 核心算法

（1）末端插入接入新节点函数

在本次实验中，该函数被调用次数尤为多，成为该程序的核心算法。

该算法的具体思想大致为创建一个新结点，将系数和指数赋值到结点上，将结点指向 NULL，再将参数结点指向新结点。

```
1.  // 末端插入接入新结点
2.  void Attch(float c, int e, polypointer d) {
    polypointer x = (polypointer)malloc(sizeof(polypointer));
3.      x->coef = c;
4.      x->exp = e;
5.      x->link = NULL;
6.      d->link = x;
7.      d = x;
8.  }
```

（2）多项式求值（秦九韶算法）

由于实验要求多项式求值所设计的算法为线性时间算法，故本实验引入秦九韶算法实现该功能。

该算法的具体思想是将多项式按照幂次的从高到低的顺序表示为一个系数数组，在利用累积法从高次项开始，将多项式分解为若干步，每步由一次加法和一次乘法组成，能以时间复杂度为$O(n)$的效果实现功能。

```
// 计算多项式在 x0 处数值(秦九韶算法)
float Calculate(polypointer p, float x) {
    p = p->link;
    int volume = p->exp;
    float array[volume + 1];
    memset(array, 0, volume + 1);
    while (p != NULL) {
        array[p->exp] = p->coef;
        p = p->link;
    }
    float sum = array[volume];
    for (int i = volume; i > 0; i--) {
        sum = x * sum + array[i - 1];
    }
    return sum;
}
```

2. 物理设计（即存储结构设计）

本次实验主要采用动态链表存储多项式，该存储结构包含三个数据类型，分别为系数（float 类型），指数（int 类型）和指向下一个结点的指针。动态链表的示意图如下图所示。

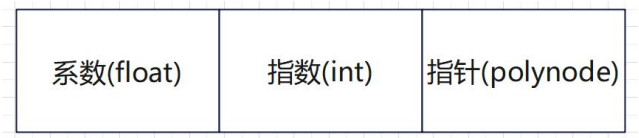
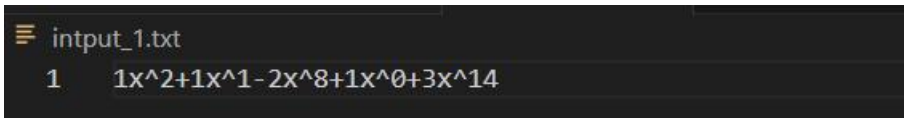


图 2 动态链表存储

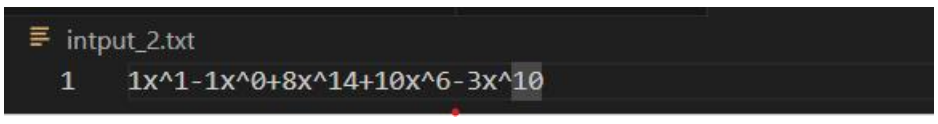
基于上述存储结构，实现动态链表的创建、添加结点、链表排序等操作。

四、测试结果（包括测试数据、结果数据及结果的简单分析和结论，可以用截图得形式贴入此报告）

1. 文件 intput_1.txt



2. 文件 intput_2.txt



3. 多项式加法

```
请选择要实现的功能：1.多项式加法；2.多项式减法；3.多项式乘法；4.计算多项式在特定点的值；0.退出
1
多项式1和多项式2分别为：
3.000000x^14-2.000000x^8+1.000000x^2+1.000000x^1+1.000000x^0
8.000000x^14-3.000000x^10+10.000000x^6+1.000000x^1-1.000000x^0
相加后的结果多项式为：
11.000000x^14-3.000000x^10-2.000000x^8+10.000000x^6+1.000000x^2+2.000000x^1
请选择要实现的功能：1.多项式加法；2.多项式减法；3.多项式乘法；4.计算多项式在特定点的值；0.退出
```

4. 多项式减法

```
请选择要实现的功能：1.多项式加法；2.多项式减法；3.多项式乘法；4.计算多项式在特定点的值；0.退出
2
多项式1和多项式2分别为：
3.000000x^14-2.000000x^8+1.000000x^2+1.000000x^1+1.000000x^0
8.000000x^14-3.000000x^10+10.000000x^6+1.000000x^1-1.000000x^0
相减后的结果多项式为：
-5.000000x^14+3.000000x^10-2.000000x^8-10.000000x^6+1.000000x^2+2.000000x^0
请选择要实现的功能：1.多项式加法；2.多项式减法；3.多项式乘法；4.计算多项式在特定点的值；0.退出
```

5. 多项式乘法

```
请选择要实现的功能：1.多项式加法；2.多项式减法；3.多项式乘法；4.计算多项式在特定点的值；0.退出
3
多项式1和多项式2分别为：
3.000000x^14-2.000000x^8+1.000000x^2+1.000000x^1+1.000000x^0
8.000000x^14-3.000000x^10+10.000000x^6+1.000000x^1-1.000000x^0
相乘后的多项式为：
24.000000x^28-9.000000x^24-16.000000x^22+30.000000x^20+6.000000x^18+8.000000x^16+11.000000x^15-15.000000x^14-3.000000x^12-3.000000x^11-3.000000x^10-2.000000x^9+
12.000000x^8+10.000000x^7+10.000000x^6+1.000000x^3-1.000000x^0
请选择要实现的功能：1.多项式加法；2.多项式减法；3.多项式乘法；4.计算多项式在特定点的值；0.退出
```

6. 多项式求值（对 intput_1 求值）

```
请选择要实现的功能：1.多项式加法；2.多项式减法；3.多项式乘法；4.计算多项式在特定点的值；0.退出
4
多项式1和多项式2分别为：
3.000000x^14-2.000000x^8+1.000000x^2+1.000000x^1+1.000000x^0
8.000000x^14-3.000000x^10+10.000000x^6+1.000000x^1-1.000000x^0
请选择待计算的多项式：1.多项式1；2.多项式2
1
请输入点x0的数值：
2
多项式1在x0=2.000000上的值为48647.000000
请选择要实现的功能：1.多项式加法；2.多项式减法；3.多项式乘法；4.计算多项式在特定点的值；0.退出
```

6. 多项式求值（对 intput_2 求值）

```
请选择要实现的功能：1.多项式加法；2.多项式减法；3.多项式乘法；4.计算多项式在特定点的值；0.退出
4
多项式1和多项式2分别为：
3.000000x^14-2.000000x^8+1.000000x^2+1.000000x^1+1.000000x^0
8.000000x^14-3.000000x^10+10.000000x^6+1.000000x^1-1.000000x^0
请选择待计算的多项式：1.多项式1；2.多项式2
2
请输入点x0的数值：
2
多项式1在x0=2.000000上的值为128641.000000
请选择要实现的功能：1.多项式加法；2.多项式减法；3.多项式乘法；4.计算多项式在特定点的值；0.退出
```

7. 文件 output.txt

```
output.txt
1   多项式1为：
2   3.000000x^14-2.000000x^8+1.000000x^2+1.000000x^1+1.000000x^0
3   多项式2为：
4   8.000000x^14-3.000000x^10+10.000000x^6+1.000000x^1-1.000000x^0
5   多项式1+多项式2为：
6   11.000000x^14-3.000000x^10-2.000000x^8+10.000000x^6+1.000000x^2+2.000000x^1
7   多项式1-多项式2为：
8   -5.000000x^14+3.000000x^10-2.000000x^8-10.000000x^6+1.000000x^2+2.000000x^0
9   多项式1*多项式2为：
10  24.000000x^28-9.000000x^24-16.000000x^22+30.000000x^20+6.000000x^18+8.000000x^16+11.000000x^15-15.000000x^14-3.000000x^12-3.000000x^11-3.000000x^10-2.000000x^9+12.000000x^8+10.000000x^7+10.000000x^6+1.000000x^3-1.000000x^0
11  多项式1在x0=2.000000上的值为48647.000000
12  多项式2在x0=2.000000上的值为128641.000000
13
```

五、经验体会与不足

经验体会：通过编写和调试实现一元多项式计算器程序，熟练掌握了链式存储结构的实现及相关操作，巩固了对链式存储结构的认识，提高了算法的效率和空间利用率，熟练 c 语言文件操作函数。

不足：面对情况多样的多项式输入类型，程序还是不能够有适应它们的能力，多项式运算函数功能实现也不够简洁，造成时间和空间的浪费。

六、附录：源代码（带注释）

```c
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int number;
int poly_number;
typedef struct polynode {
    float coef;
    int exp;
    struct polynode *link;
} polynode;
typedef polynode *polypointer;
// 末端插入接入新结点
void Attch(float c, int e, polypointer d) {
    polypointer x = (polypointer)malloc(sizeof(polypointer));
    x->coef = c;
    x->exp = e;
    x->link = NULL;
    d->link = x;
    d = x;
}
// 从文件中读取多项式
polypointer Read_polypointer(FILE *file) {
    polypointer p = (polypointer)malloc(sizeof(polypointer));
    polypointer d = p;
    p->link = NULL;
    float c;
    int e;
    char symbol;
    symbol = fgetc(file);
    if (symbol == '-') {
        rewind(file);
```

```
33.             fscanf(file, "%c%fx^%d", &symbol, &c, &e);
34.             Attch(-c, e, d);
35.             d = d->link;
36.         } else {
37.             rewind(file);
38.             fscanf(file, "%fx^%d", &c, &e);
39.             Attch(c, e, d);
40.             d = d->link;
41.         }
42.         while (fscanf(file, "%c%fx^%d", &symbol, &c, &e) != EOF) {
43.             if (symbol == '-') {
44.                 Attch(-c, e, d);
45.                 d = d->link;
46.             } else if (symbol == '+') {
47.                 Attch(c, e, d);
48.                 d = d->link;
49.             }
50.         }
51.         return p;
52.     }
53.     // 对多项式按指数降幂排序
54.     polypointer Sort_polypointer(polypointer p) {
55.         polypointer pre, q, end;
56.         end = NULL;
57.         pre = p;
58.         while ((p->link->link) != end) {
59.             pre = p;
60.             q = p->link;
61.             while (q->link != end) {
62.                 if (q->exp < q->link->exp) {
63.                     pre->link = q->link;
64.                     q->link = q->link->link;
65.                     pre->link->link = q;
66.                     q = pre->link;
67.                 }
68.                 q = q->link;
69.                 pre = pre->link;
70.             }
71.             end = q;
72.         }
73.         return p;
74.     }
75.     // 排查多项式中的系数为 0 的项
76.     polypointer Select_zero(polypointer p) {
```

```
77.     polypointer q;
78.     q = p;
79.     while (q->link != NULL) {
80.         if (q->link->coef == 0) {
81.             q->link = q->link->link;
82.         } else {
83.             q = q->link;
84.         }
85.     }
86.     return p;
87. }
88. // 多项式加法
89. polypointer Add(polypointer a, polypointer b) {
90.     polypointer p, q, d, c;
91.     float y;
92.     p = a->link;
93.     q = b->link;
94.     c = (polypointer)malloc(sizeof(polypointer));
95.     d = c;
96.     while ((p != NULL) && (q != NULL)) {
97.         if (p->exp == q->exp) {
98.             y = p->coef + q->coef;
99.             if (y) {
100.                Attch(y, p->exp, d);
101.                d = d->link;
102.                p = p->link;
103.                q = q->link;
104.            } else if (y == 0) {
105.                p = p->link;
106.                q = q->link;
107.            }
108.        } else if (p->exp > q->exp) {
109.            Attch(p->coef, p->exp, d);
110.            d = d->link;
111.            p = p->link;
112.        } else if (p->exp < q->exp) {
113.            Attch(q->coef, q->exp, d);
114.            d = d->link;
115.            q = q->link;
116.        }
117.    }
118.    while (p != NULL) {
119.        Attch(p->coef, p->exp, d);
120.        d = d->link;
```

```
121.         p = p->link;
122.     }
123.     while (q != NULL) {
124.         Attch(q->coef, q->exp, d);
125.         d = d->link;
126.         q = q->link;
127.     }
128.     d = NULL;
129.     return c;
130. }
131. // 多项式减法
132. polypointer Subtract(polypointer a, polypointer b) {
133.     polypointer p, q, d, c;
134.     float y;
135.     p = a->link;
136.     q = b->link;
137.     c = (polypointer)malloc(sizeof(polypointer));
138.     d = c;
139.     while ((p != NULL) && (q != NULL)) {
140.         if (p->exp == q->exp) {
141.             y = p->coef - q->coef;
142.             if (y) {
143.                 Attch(y, p->exp, d);
144.                 d = d->link;
145.                 p = p->link;
146.                 q = q->link;
147.             } else if (y == 0) {
148.                 p = p->link;
149.                 q = q->link;
150.             }
151.         } else if (p->exp > q->exp) {
152.             Attch(p->coef, p->exp, d);
153.             d = d->link;
154.             p = p->link;
155.         } else if (p->exp < q->exp) {
156.             Attch(-q->coef, q->exp, d);
157.             d = d->link;
158.             q = q->link;
159.         }
160.     }
161.     while (p != NULL) {
162.         Attch(p->coef, p->exp, d);
163.         d = d->link;
164.         p = p->link;
```

```
    }
    while (q != NULL) {
        Attch(-q->coef, q->exp, d);
        d = d->link;
        q = q->link;
    }
    d = NULL;
    return c;
}
// 多项式乘法
polypointer Multiply(polypointer a, polypointer b) {
    polypointer p, q, d, c, flag;
    float y;
    int z;
    int key = 0;
    p = a->link;
    q = b->link;
    c = (polypointer)malloc(sizeof(polypointer));
    c->link = NULL;
    d = c;
    while (p != NULL) {
        while (q != NULL) {
            key = 0;
            y = p->coef * q->coef;
            z = p->exp + q->exp;
            flag = c->link;
            while (flag != NULL) {
                if (z == flag->exp) {
                    key = 1;
                    flag->coef = flag->coef + y;
                }
                flag = flag->link;
            }
            if (key == 0) {
                Attch(y, z, d);
                d = d->link;
            }
            q = q->link;
        }
        q = b->link;
        p = p->link;
    }
    c = Sort_polypointer(c);
    c = Select_zero(c);
```

```
209.     return c;
210. }
211. // 计算多项式在 x0 处数值(秦九韶算法)
212. float Calculate(polypointer p, float x) {
213.     p = p->link;
214.     int volume = p->exp;
215.     float array[volume + 1];
216.     memset(array, 0, volume + 1);
217.     while (p != NULL) {
218.         array[p->exp] = p->coef;
219.         p = p->link;
220.     }
221.     float sum = array[volume];
222.     for (int i = volume; i > 0; i--) {
223.         sum = x * sum + array[i - 1];
224.     }
225.     return sum;
226. }
227. // 打印多项式
228. void Output(polypointer p) {
229.     polypointer first;
230.     p = p->link;
231.     first = p;
232.     while (p != NULL) {
233.         if (p != first) {
234.             if (p->coef > 0) {
235.                 printf("+");
236.             }
237.         }
238.         printf("%fx^%d", p->coef, p->exp);
239.         p = p->link;
240.     }
241.     printf("\n");
242. }
243. // 将多项式写出文件中
244. void Write(polypointer p) {
245.     if (!p) {
246.         printf("输入出现错误\n");
247.     }
248.     p = p->link;
249.     FILE *file = NULL;
250.     file = fopen("output.txt", "a");
251.     if (number == 1) {
252.         fprintf(file, "多项式 1+多项式 2 为:\n");
```

```
253.     } else if (number == 2) {
254.         fprintf(file, "多项式 1-多项式 2 为:\n");
255.     } else if (number == 3) {
256.         fprintf(file, "多项式 1*多项式 2 为:\n");
257.     }
258.     fclose(file);
259.     file = fopen("output.txt", "a");
260.     char symbol;
261.     symbol = getc(file);
262.     if (symbol == '-') {
263.         rewind(file);
264.         fprintf(file, "-%fx^%d", p->coef, p->exp);
265.     } else {
266.         rewind(file);
267.         fprintf(file, "%fx^%d", p->coef, p->exp);
268.     }
269.     p = p->link;
270.     while (p != NULL) {
271.         if (p->coef < 0) {
272.             fprintf(file, "%fx^%d", p->coef, p->exp);
273.         } else {
274.             fprintf(file, "+%fx^%d", p->coef, p->exp);
275.         }
276.         p = p->link;
277.     }
278.     fprintf(file, "\n");
279.     fclose(file);
280. }
281. // 将多项式在 x0 处的数值写入文件
282. void Write_x0(polypointer p, float x0, float result_calculate) {

283.     FILE *file = NULL;
284.     file = fopen("output.txt", "a");
285.     if (poly_number == 1) {
286.                     fprintf(file,  " 多 项 式 1 在 x0=%f 上 的 值
    为%f\n", x0, result_calculate);
287.     }
288.     if (poly_number == 2) {
289.                     fprintf(file,  " 多 项 式 2 在 x0=%f 上 的 值
    为%f\n", x0, result_calculate);
290.     }
291.     fclose(file);
292. }
293.
```

```
294. int main() {
295.     polypointer intput_1, intput_2;
296.     FILE *f1 = fopen("intput_1.txt", "r");
297.     FILE *f2 = fopen("intput_2.txt", "r");
298.     if (f1 == NULL) {
299.         printf("未成功打开文件 intput_1.txt\n");
300.         exit(1);
301.     } else {
302.         intput_1 = Read_polypointer(f1);
303.         fclose(f1);
304.     }
305.     if (f2 == NULL) {
306.         printf("未成功打开文件 intput_2.txt\n");
307.         exit(1);
308.     } else {
309.         intput_2 = Read_polypointer(f2);
310.         fclose(f2);
311.     }
312.     intput_1 = Sort_polypointer(intput_1);
313.     intput_2 = Sort_polypointer(intput_2);
314.     FILE *file = NULL;
315.     file = fopen("output.txt", "a");
316.     fprintf(file, "多项式 1 为:\n");
317.     fclose(file);
318.     Write(intput_1);
319.     file = fopen("output.txt", "a");
320.     fprintf(file, "多项式 2 为:\n");
321.     fclose(file);
322.     Write(intput_2);
323.     polypointer result[5] = {NULL};
324.     float result_calculate;
325.     while (1) {
326.         printf(
327.             "请选择要实现的功能：1.多项式加法；2.多项式减法；3.多项式乘
    法；4."
328.             "计算多项式在特定点的值；0.退出\n");
329.         scanf("%d", &number);
330.         if (number == 1) {
331.             printf("多项式 1 和多项式 2 分别为：\n");
332.             Output(intput_1);
333.             Output(intput_2);
334.             result[1] = Add(intput_1, intput_2);
335.             printf("相加后的结果多项式为：\n");
336.             Output(result[1]);
```

```
337.             Write(result[1]);
338.         } else if (number == 2) {
339.             printf("多项式 1 和多项式 2 分别为：\n");
340.             Output(intput_1);
341.             Output(intput_2);
342.             result[2] = Subtract(intput_1, intput_2);
343.             printf("相减后的结果多项式为：\n");
344.             Output(result[2]);
345.             Write(result[2]);
346.         } else if (number == 3) {
347.             printf("多项式 1 和多项式 2 分别为：\n");
348.             Output(intput_1);
349.             Output(intput_2);
350.             result[3] = Multiply(intput_1, intput_2);
351.             printf("相乘后的多项式为：\n");
352.             Output(result[3]);
353.             Write(result[3]);
354.         } else if (number == 4) {
355.             printf("多项式 1 和多项式 2 分别为：\n");
356.             Output(intput_1);
357.             Output(intput_2);
358.             printf("请选择待计算的多项式：1.多项式 1；2.多项式 2\n");
359.             scanf("%d", &poly_number);
360.             if (poly_number == 1) {
361.                 printf("请输入点 x0 的数值：\n");
362.                 float x0;
363.                 scanf("%f", &x0);
364.                 result_calculate = Calculate(intput_1, x0);
365.                                 printf("多项式 1 在 x0=%f 上的值
    为%f\n", x0, result_calculate);
366.                 Write_x0(intput_1, x0, result_calculate);
367.             } else if (poly_number == 2) {
368.                 printf("请输入点 x0 的数值：\n");
369.                 float x0;
370.                 scanf("%f", &x0);
371.                 result_calculate = Calculate(intput_2, x0);
372.                                 printf("多项式 1 在 x0=%f 上的值
    为%f\n", x0, result_calculate);
373.                 Write_x0(intput_1, x0, result_calculate);
374.             } else {
375.                 printf("输入值非法");
376.             }
377.         } else if (number == 0) {
378.             break;
```

```
379.         } else {
380.             printf("输入值非法");
381.         }
382.     }
```