2021-2 자료구조및실습 실습과제 03

학번	2020136129	이름	최수연

1) 각 문제에 대한 분석과 및 해결 방법

1. 교재 119~120쪽의 프로그래밍 프로젝트 P3.4를 구현하시오.

- **최고 차수 반환**
- **덧셈 기능**
- **뺄셈 기능 (덧셈 함수를 이용할 수 있는 방법)**
- **곱셈 기능**
- **화면 출력 기능**

단, display() 함수를 개선할 수 있는 방법을 고민해서 추가하시오.
예) 0인 항 표현 안하기, 음수 계수 표현방법 개선 등

[문제분석 및 해결방법]

본 문제는 교수님께서 힌트를 많이 주셔서 궁극적으로 뺄셈 기능, 곱셈 기능, 화면 출력 기능(해당 다항식에 미지수를 넣어 계산한 값 출력), display() 함수 개선할 수 있는 법만 코드로 작성하면 되었다.

먼저 **최고 차수 반환**의 경우 교재 120p의 방법2를 사용하였는데, 인덱스가 0, 1, 2 순으로 가는 것에 맞춰 x의 차수를 인덱스 번호에 맞춘다. 다시 말해, x의 최고 차수가 2이면, 최고 차수대로 x의 계수가 입력되어도, 저장될 때는 인덱스 번호와 x의 차수가 동일하게 들어간다. (e.g. a.coef[0]에는 x의 차수가 0인 항의 계수가 들어감) 그러나 display() 함수로 출력될 때는 for 문을 사용하여 최고 차수의 항부터 거꾸로 1씩 감소하며 차수가 0인 항의 계수로 출력이 된다.

덧셈 기능의 경우 a를 self로 받고 a가 b보다 더 클 경우, 다항식 a를 새로 생성한 객체 p에 넣는다. (b차수+1) 기준으로 for 문을 돌려 p에 다항식 b를 더한다. 만약 b가 더 클 경우, 다항식 b를 새로 생성한 객체 p에 넣는다. (a차수+1) 기준으로 for 문을 돌려 p에 다항식 a를 더한다.

다음 **뺄셈 기능**의 경우 덧셈의 코드를 변형하여 구할 수 있다. a를 self로 받고, 무조건 (a- - b)만 가능하다는 전제하에 a가 더 클 경우, 다항식 a를 새로 생성한 객체 p에 넣는다. (b차수+1) 기준으로 for 문을 돌려 p에 다항식 b를 뺀다. 만약 b가 더 클 경우, 다항식 b를 각항마다 (-1)을 곱하여 새로 생성한 객체 p에 넣는다. (a차수+1) 기준으로 for 문을 돌려 p에 다항식 a를 뺀다.

다음으로 **곱셈 기능**의 경우 이중 for 문을 사용하는데, 먼저 변수 deg를 선언하여 deg에 self와 b의 차수를 더한 값을 넣어둔다. 그리고 for 문을 사용하여 deg만큼 루프를 돌며 p.coef를 초기화한다. 다음 이중 for 문을 사용하여 바깥 for 문은 i(self차수)까지의 범위로 설정하고, 안쪽 for 문은 n(b차수)까지의 범위로 설정하여, 다항식 p 인덱스(i+n)의 값에 차수 i의 self와 차수 n의 b를 더하여 p에 더해준다.

다음 **화면 출력 기능**의 경우 다항식에 지정된 미지수 x를 대입하였을 때 대입한 결과값이 반환되도록 하는 함수를 구현하였다. 먼저 객체 p를 생성하여 p에 self로 받은 다항식을 복사한다. 그리고 변수 y를 생성하여 y에 차수가 0인 항의 계수를 먼저 y에 넣은 후, for 문을 사용하여 해당 self의 차수부터 1까지 반복하여 y에 해당 차수만큼 미지수 x값을 제곱하고, 이 값을 해당 차수의 계수와 곱하여 y에 더해준다. for 문이 종료되면 y 값을 반환한다.

마지막으로 **display() 함수 개선 코드**를 생각해보았는데, 해당 코드에서는 0인 항을 표현하지 않도록 하는 코드를 추가하였다. 먼저 설정한 차수 deg에 해당하는 값이 0이 아닐 경우, 먼저 출력한다. 그 후 for 문을 사용하여 deg –1부터 1까지 –1씩 진행하여 계수가 0인 항은 출력하지 않고 그냥 넘어가고, 계수가 0이 아니면 + 붙여 출력하도록 함. for 문이 끝난 후 다음 if 문에서 만약 0인 항의 계수가 0이 아니면 + 붙여 출력하고, 그렇지 않고 만약 0인 항의 계수가 0이면 그냥 다음 줄로 넘어가도록 한다. (해당 마지막 코드를 추가한 이유는 앞의 코드에서 end=''를 사용하였기 때문이다.)

2) 자신이 구현한 주요 코드

```
def display(self, msg = "f(x) = "):                          #0인 항 표현을 없앤 display 함수 개선 코드
    print("   ", msg, end='')
    deg = self.degree()
    if(self.coef[deg] != 0):
        print("%5.1f x^%d " % (self.coef[deg], deg), end='')
    for n in range(deg - 1, 0, -1):                          #deg - 1부터 1까지 -1씩 진행 #방법2
        if(self.coef[n] == 0):                               #계수가 0인 항은 출력하지 않음
            continue                                         #해당 차수의 계수가 0이면 그냥 넘어감
        elif(self.coef[n] != 0):                             #계수가 0이 아니면 출력
            print("+ %5.1f x^%d " % (self.coef[n], n), end='')

    if(self.coef[0] != 0):                                   #계수가 0인 항이 0이 아니면 출력
        print("+ %4.1f" % self.coef[0])
    elif(self.coef[0] == 0):                                 #계수가 0인 항이 0이면 그냥 다음줄로 넘어감
        print()
```

```
def sub(self, b):                          #뺄셈 => 무조건 (a - b)로 뺄셈 계산이 되도록 함
    p = Polynomial()
    if self.degree() > b.degree() :        ### self(a)의 차수가 더 클 경우
        p.coef = list(self.coef)           #self 다항식을 p에 넣음
        for i in range(b.degree()+1) :     #차수별로 b의 계수를 p(self)에 빼줌
            p.coef[i] -= b.coef[i]
    else :                                 ### b의 차수가 더 클 경우
        p.coef = list(b.coef)              #b 다항식을 p에 넣음
        for i in range(b.degree()+1) :     #p를 모두 음수로 바꿈
            p.coef[i] = (-1)*p.coef[i]
        for i in range(self.degree()+1) :  #p(b)와 self(a)를 더함
            p.coef[i] += self.coef[i]
    return p                               #p를 반환
```

```
def mul(self, b):                          #곱셈
    p = Polynomial()
    deg = self.degree() + b.degree()       #다항식 곱셈의 최고 차수는 각 다항식의 최고 차수를 합한
값
    for k in range(deg + 1) :              #deg 차수만큼 초기화
        p.coef.append(0)
    for i in range(self.degree() + 1) :    #이중 for문 사용하여 각 차수마다의 계수를 곱하여 넣음
        for n in range(b.degree() + 1) :
            p.coef[i + n] += self.coef[i] * b.coef[n]
    return p
```

```
def eval (self, x):                          #다항식에 미지수 대입
    p = Polynomial()
    p.coef = list(self.coef)
    y = p.coef[0]                            #차수가 0인 항의 계수를 먼저 y에 넣음
    for n in range(self.degree(), 0, -1) :   #차수에 맞춰 곱하여 더함
            y += p.coef[n]*(x ** n)
    return y
```

3) 다양한 입력에 대한 테스트 결과

1. 교재 119~120쪽의 프로그래밍 프로젝트 P3.4를 구현하시오.

```
최고차항부터 차수를 순서대로 입력하시오: 5 -3 12
최고차항부터 차수를 순서대로 입력하시오: 2 -6 0 -4
   A(x) =   5.0 x^2 +  -3.0 x^1 + 12.0
   B(x) =   2.0 x^3 +  -6.0 x^2 + -4.0
   ADD(x) =   2.0 x^3 +  -1.0 x^2 +  -3.0 x^1 +  8.0
   SUB(x) =  -2.0 x^3 +  11.0 x^2 +  -3.0 x^1 + 16.0
   MUL(x) =  10.0 x^5 + -36.0 x^4 +  42.0 x^3 + -92.0 x^2 +  12.0 x^1 + -48.0
   ADD(2) =  14.0
```

$A(x) = 5x^2 - 3x + 12$
$B(x) = 2x^3 - 6x^2 - 4$
$ADD(x) = 2x^3 - x^2 - 3x + 8$
$SUb(x) = -2x^3 + 11x^2 - 3x + 16$
$ML(x) = 10x^5 - 36x^4 + 42x^3 - 92x^2 + 12x - 48$
$ADD(2) = 14$

4) 코드에 대한 설명 및 해당 문제에 대한 고찰

이번 문제에서 음수 계수에 대한 표현이 좀 아쉬웠는데 해당 부분을 개선할 수 있는 법에 대해 더 고민해봐야 할 것 같다. 또한 다항식의 곱셈 기능 구현할 때 생각보다 어려워서 고민을 많이 했던 것 같다. 그런데 계속 생각하던 방식 외에 다른 방식으로 접근해보니 코드로 구현하기 더 수월해진 것 같아서 끝내 곱셈 기능을 구현해낼 수 있었던 것 같다. 하지만 이번 과제 코드가 완벽하다고 생각하지 않기 때문에 이에 대해 더 깔끔하게 코드를 수정할 수 있도록 다시 공부해봐야 할 것 같다.

5) 이번 과제에 대한 느낀점

이번 과제는 지난 과제들보다 신경 써야 할 부분이 많아서 헷갈리기도 했고 생각보다 어렵다고 느꼈다. 그래도 하나씩 구현해내는 과정에서 뿌듯함을 많이 느꼈던 것 같다. 하지만 이번 과제를 너무 급하게 하다 보니 충분한 시간을 두고 고민하지는 못했는데, 시간적 여유를 두고 과제를 했으면 좀 더 깔끔하고 좋은 코드를 만들 수 있었을 것 같았다는 아쉬움이 남는다. 다음 과제부터는 좀 더 빨리 시작해야겠다.

6) 궁금한 점이나 건의사항

딱히 없습니다.

7) 자신이 구현한 전체 코드

```
class Polynomial :
    def __init__(self):
        self.coef= []    #생성자

    def degree (self):
        return len(self.coef) - 1    #다항식이 3개의 항을 가지면 차수는 2

    def display(self, msg = "f(x) = "):
        print("   ", msg, end='')
        deg = self.degree()
        if(self.coef[deg] != 0):
            print("%5.1f x^%d " % (self.coef[deg], deg), end='')

        for n in range(deg - 1, 0, -1):          #deg - 1부터 1까지 -1씩 진행 #방법2
            if(self.coef[n] == 0):               #계수가 0인 항은 출력하지 않음
                continue                         #해당 차수의 계수가 0이면 그냥 넘어감
            elif(self.coef[n] != 0):             #계수가 0이 아니면 출력
                print("+ %5.1f x^%d " % (self.coef[n], n), end='')

        if(self.coef[0] != 0):                   #계수가 0인 항이 0이 아니면 출력
            print("+ %4.1f" % self.coef[0])
        elif(self.coef[0] == 0):                 #계수가 0인 항이 0이면 그냥 다음줄로 넘어감
            print()

    def add(self, b):    #덧셈(교수님께서 알려주신 코드)
        p = Polynomial()
        if self.degree() > b.degree() :
            p.coef = list(self.coef)
            for i in range(b.degree()+1) :
                p.coef[i] += b.coef[i]
        else :
            p.coef = list(b.coef)
            for i in range(self.degree()+1) :
                p.coef[i] += self.coef[i]
        return p

    def sub(self, b):                            #뺄셈 => 무조건 (a - b)로 뺄셈 계산이 되도록 함
        p = Polynomial()
        if self.degree() > b.degree() :          ### self(a)의 차수가 더 클 경우
            p.coef = list(self.coef)             #self 다항식을 p에 넣음
            for i in range(b.degree()+1) :       #차수별로 b의 계수를 p(self)에 빼줌
                p.coef[i] -= b.coef[i]
        else :                                   ### b의 차수가 더 클 경우
            p.coef = list(b.coef)                #b 다항식을 p에 넣음
```

```
            for i in range(b.degree()+1) :        #p를 모두 음수로 바꿈
                p.coef[i] = (-1)*p.coef[i]
            for i in range(self.degree()+1) :     #p(b)와 self(a)를 더함
                p.coef[i] += self.coef[i]
        return p                                   #p를 반환

    def mul(self, b):                             #곱셈
        p = Polynomial()
        deg = self.degree() + b.degree()  #다항식 곱셈의 최고 차수는 각 다항식의 최고 차수를 합한 값
        for k in range(deg + 1) :             #deg 차수만큼 초기화
            p.coef.append(0)
        for i in range(self.degree() + 1) :  #이중 for문 사용하여 각 차수마다의 계수를 곱하여 넣음
            for n in range(b.degree() + 1) :
                p.coef[i + n] += self.coef[i] * b.coef[n]
        return p

    def eval (self, x):                           #다항식에 미지수 대입
        p = Polynomial()
        p.coef = list(self.coef)
        y = p.coef[0]                             #차수가 0인 항의 계수를 먼저 y에 넣음
        for n in range(self.degree(), 0, -1) :  #차수에 맞춰 곱하여 더함
                y += p.coef[n]*(x ** n)
        return y

def read_poly():
    instr = input("최고차항부터 차수를 순서대로 입력하시오: ")     # "1 2 0"
    strlist = instr.split()                       #[ "1", "2", "0" ]
    p = Polynomial()
    for coef in strlist :
        val = float(coef)                          #계수 ==> 실수
        p.coef.insert(0, val)                     #맨 앞에 넣음 [0, 2, 1] #1이 최고차항 #방법2
    return p

# 테스트 코드
a = read_poly()
b = read_poly()
c = a.add(b) # a + b
d = a.sub(b) # a - b
e = a.mul(b) # a * b
a.display("A(x) = ")
b.display("B(x) = ")
c.display("ADD(x) = ")
d.display("SUB(x) = ")
e.display("MUL(x) = ")
print(" ADD(2) = ", c.eval(2))
```