|  |  |
| --- | --- |
|  |  |
| 자료구조와C++프로그래밍 |  |
|  |  |
|  | **이름 : 장혁수****학과 : 소프트웨어학과****학번 : 2019125055****수업시간 : 월 10:30, 수09:00** |
|  | **과제번호 : 과제 2** |

### 문제 정의

먼저 크게 문제를 쪼개 보면3 가지 문제가 나온다고 할 수 있다.

1. 계수, 지수 순서대로 입력 받고 이를 다항식으로 구현하기
2. 다항식과 다항식을 곱해주는 MUL함수 구현
3. 다항식의 변수에 값을 대입한 결과를 계산해주는 Eval 함수를 구현

이에 대한 문제해결에 앞서 한번 다항식을 어떻게 표현할지 생각해보면 많은 방법이 있겠지만, 계수-지수 쌍으로 저장하는 방식으로 표현하고자 한다.

즉 , 각 항의 지수, 계수를 저장하는 변수(coef, exp)를 가지고 있는Term 클래스를 선언하고, 이 Term클래스 배열을 가리키는 포인터변수(termArray)와 Term 클래스 배열의 크기 또 항의 개수를 저장하는 변수(terms, capacity)를 가지고 있는 Polynomial 클래스를 또한 만들었다.

그리고 위 문제들의 해결방법을 간단히 생각해 보았다.

1. 다항식 A, B 를 계수, 지수 순서대로 다항식의 내용을 각각 입력 받고 A, B의 termArray에 저장한다.
2. A, B의 계수와 지수의 정보를 가지고 있는 A, B의 termArray에 접근해서 지수가 같은 항끼리 더해서 하나로 만들어주고 계수가 0인 항은 제거하고 또, 차수가 큰 순서대로 정렬하는 등의 작업을 한다. 이후 A, B를 출력한다.
3. 두 다항식 A, B의 곱을 구현하는데 먼저 A의 첫번째 항부터 시작해서 마지막 항까지 차례대로 B의 모든 항과 곱하고 그 결과 차례대로 생성된 다항식들을 계속 더한다. 이후 최종 다항식을 새로운 다항식 C에 저장하고 C를 출력한다.
4. 변수 x 값을 입력 받아 A, B, C 다항식에 x 값을 대입하고 그 결과를 각각 출력한다.

### 문제 해결

이제 위에서 말한 문제해결 방법을 각각 자세하게 풀어보자

★ 다항식 A, B 를 계수, 지수 순서대로 다항식의 내용을 각각 입력 받고 A, B의 termArray에 저장

먼저 Polynomial 클래스내에 NewTerm함수를 만들었다.

NewTerm 함수를 소개하자면, 먼저 인수로 계수, 지수 값을 받고 (\*this)의 termArray에 저장을 하는 함수이다. 정확히 말하자면, termArray가 가리키는 배열에 저장한다는 것이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위 코드는 NewTerm함수의 주요 코드이다.

설명하자면 먼저 (\*this)의 termArray에 공간이 충분한지 확인한다.

그리고 공간이 넉넉치 않는다면, capacity값을 2배 해주고, 그 값만큼 길이의 Term배열을 동적할당 받고 temp가 이것을 가리키게 한다.

여기서 포인트는 예외처리 메커니즘 try, catch를 이용해 try블록에서 동적할당에 실패 시 catch블록이 실행되도록 한다. 즉, 동적할당에 성공하면 다음 코드로 정상적으로 진행하고 실패하면 에러 메시지와 함께 프로그램이 종료되게 했다.

다음으로 Polynomial 클래스 내에 Input\_Polynomial이라는 다항식을 계수, 지수 순으로 입력받는 함수를 위 NewTerm함수를 이용해서 구현했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위와 같이 do-while문을 이용해서 먼저 입력 연산자 cin을 이용해서 계수, 지수 값을 입력 받고 위 선언한 변수에 각각 저장해준다.

이후 위에서 소개한 NewTerm 함수를 통해 계수, 지수 값을 (\*this)의 termArray에 저장해준다. 그리고 getchar함수로 하나의 문자를 입력 받고 그 문자가 ‘\n’이면 반복문을 끝내고 아니면 한번 더 계수, 지수 값을 입력 받는다.

여기서 문제점은 바로 주어진 규칙대로 입력을 안 하면 오류가 발생할 수 있다는 점이다. 예를 들어서 숫자 이외에 다른 문자를 입력한다든가 숫자를 입력하고 바로 enter를 치는 게 아닌 spacebar를 누르고 enter를 치는 등이 있을 수 있다. 하지만 규칙대로만 한다면 문제가 없기에 이렇게 구현했다.

텍스트이(가) 표시된 사진

자동 생성된 설명마지막으로 main함수에서 A, B, C Polynomial 객체를 선언하고, Input\_Polynomial함수를 이용해서 다항식 A, B를 각각 입력 받을 수 있다.

★ 입력 받은 다항식 A, B 를 정리 및 정렬 하고 출력하기

이렇게 다항식 A, B를 입력 받았는데, 보다시피 입력 받은 데이터들은 정돈되지 않은 데이터들이다. 즉, 정리와 정렬이라는 과정을 거쳐야지 그제서야 출력할 준비가 되었다고 할 수 있다.

여기서 먼저 Polynomial 클래스의 생성자를 보여주자면 다음과 같다.



그리고 아래와 같이 함수를 구현했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

즉, capacity 의 default value를 1로 지정해주었고, terms은 0, termArray는 capacity 길이만큼 동적 할당된 Term배열을 가리키게 초기화해주었다.

위와 같이 구현한 이유는 지금 설명할 다항식을 정리해주는 Classify\_Polynomial함수를 구현하기 위해서이다.

다항식을 정리해주는 Classify\_Polynomial함수는 말 그대로 만약 지수가 같은 항이 있다면 그 항들을 더해서 하나의 항으로 만들어주는 기능을 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

핵심코드를 보면 먼저, 정리된 다항식을 저장할 Polynomial 객체 P를 선언할 때 (\*this)의 terms값을 인수로 넣어서 객체 P의 capacity가 (\*this)의 terms 값이 되도록 하자. 왜냐하면 다항식을 정리하면 항의 개수가 줄면 줄었지, 늘지는 않기 때문이다.

이후 for문을 이용해 (\*this)의 termArray의 모든 항에 접근하는데, 차례대로 기준으로 세우고, 이후 또 for문을 이용해 기준이 되는 항 보다 더 큰 index를 가진 항들에게 접근해서 지수가 같으면 계수를 더한다. 즉, Coef\_Sum에 먼저 기준이 되는 항의 계수를 저장하고 지수가 같으면 계수를 더한다는 것이다.

그리고 이렇게 접근한 항들의 지수를 -1로 바꾸어서 그 항은 기준이 될 필요도 없고, 또한 지수가 같은 지 찾을 때 고려대상에서 확실히 벗어나게 했다.

계속해서 안쪽의 for문이 끝나면 Coef\_Sum의 값이 0과 같으면 저장할 가치가 없기에 바깥쪽 for문으로 패스하고 아니면 선언한 객체 P의 termArray에 변수 k를 이용해서 차례대로 저장한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그리고 이중 for문이 끝나고 나서 먼저 k 값이 0인지 아닌지 확인하는데 k가 0이면 결국 다항식은 0이라는 뜻인데 객체 P에는 데이터가 아무것도 없기에 P의 terms와 capacity에 1을 넣고, 첫번째 항의 지수, 계수 값을 0이라고 인위적으로 명시해 주었다.

반대로 k가 0이 아니면 0이 아닌 다항식인 경우로 P의 terms와 capacity에 k를 넣어주었다. 이때 terms뿐만 아니라 capacity에도 k를 넣은 이유는 최대한 메모리 공간을 절약하기 위해서이다.

이제 정리된 객체 P의 데이터들을 CopyObject 함수를 이용해서 (\*this)에 복사해 준다.

CopyObject함수는 Polynomial객체의 데이터 값들을 복사해주는 함수이다.

아래는 CopyObject함수의 주요 코드이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

간단하게 설명하면 일단 (\*this)의 termArray가 가리키는 동적 할당된 메모리를 해제한다. 그리고 참조하는 객체 P의 terms만큼 길이의 Term배열을 동적 할당하고 이것을 (\*this)의 termArray가 가리키게 한다.

여기서 비슷하게 try, catch 예외처리 메커니즘을 이용해 동적 할당이 정상적으로 이루어졌는지 확인한다.

이어서 (\*this)의 capacity, terms값을 객체 P의 terms값으로 대체하고, copy함수를 이용해 P의 termArray데이터들을 (\*this)의 termArray 에 복사한다.

이렇게 정리가 된 다항식을 다음으로 차수가 큰 순서대로 정렬시켜주는 Sort\_Polynomial 함수를 이용해서 정렬 시켜준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Sort\_Polynomial 함수는 위와 같이 기본적인 선택 정렬 알고리즘을 이용해서 비교해서 차수가 더 크면 값을 서로 교환하는 방식으로 구현했다.

마지막으로 이렇게 정리, 정렬된 깔끔한 다항식을 출력시키면 된다. 그런데 여기서 생각할 수 있는 특별한 경우가 있었고 다음과 같이 해결했다.

1. 계수가 1 또는 -1인 경우

* 이런 경우에는 1x 혹은 -1x처럼 출력하면 보기 이상하니 그냥 x, -x로 출력되게 하자

1. 상수인 경우

* 상수 일 때는 지수가 0일 때를 의미한다. 예를 들어서 다항식에서 “3”을 표현하면 “3x^0” 인데 이러면 또한 보기 이상하니 그냥 “3”이라고 출력되게 하자

예를 들어서 ‘1 3 -1 2 1 1 0 1’ 라고 입력될 시 ‘x^3○-○x^2○+○x^1○+○1’ 라고 출력되게 하자. 이때 ○는 spacebar를 뜻한다.

이후 나올 수 있는 케이스별로 구분하자면 다음과 같다.

경우 1 : 항의 개수가 1개

경우 1-1 : 상수

경우 1-2 : 계수가 1

경우 1-3 : 계수가 -1

경우 1-4 : 나머지 일반적인 경우

경우 2 : 항의 개수가 2이상

경우 2-1 : 첫번째 항

경우 2-1-1 : 계수가 1

경우 2-1-2 : 계수가 -1

경우 2-1-3 : 나머지 일반적인 경우

경우 2-2 : 두번째 항부터 마지막 전까지 항

경우 2-2-1 : 계수가 1 또는 -1

경우2-2-2 : 나머지 일반적인 경우

경우 2-3 : 마지막 항

경우 2-3-1 : 상수

경우 2-3-2 : 계수가 1 또는 -1

경우 2-3-3 : 나머지 일반적인 경우

여기서 기본적인 알고리즘은 반복문을 통해 모든 항에 접근하는데 접근할 때마다 그 항의 부호를 뺀 모습과 다음 항의 부호를 출력하는 형식인데 항의 개수가 1개이면 안되기에 처음에 조건문을 통해 구별했다. 그리고 항의 개수가 2개 이상일 때 나올 수 있는 경우 3가지를 따로 구현해 놓았고, 반복문을 이용해 경우에 맞춰서 출력하게 했다.

아래는 참고로 위에 설명을 토대로 만든 다항식을 출력시키는 Show\_Polynomial함수이다.

텍스트, 벽이(가) 표시된 사진

자동 생성된 설명

텍스트, 벽, 은색이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

부호를 빼는 방법으로 헤더파일 stdlib.h에서 abs함수(절댓값 함수)를 이용했다.

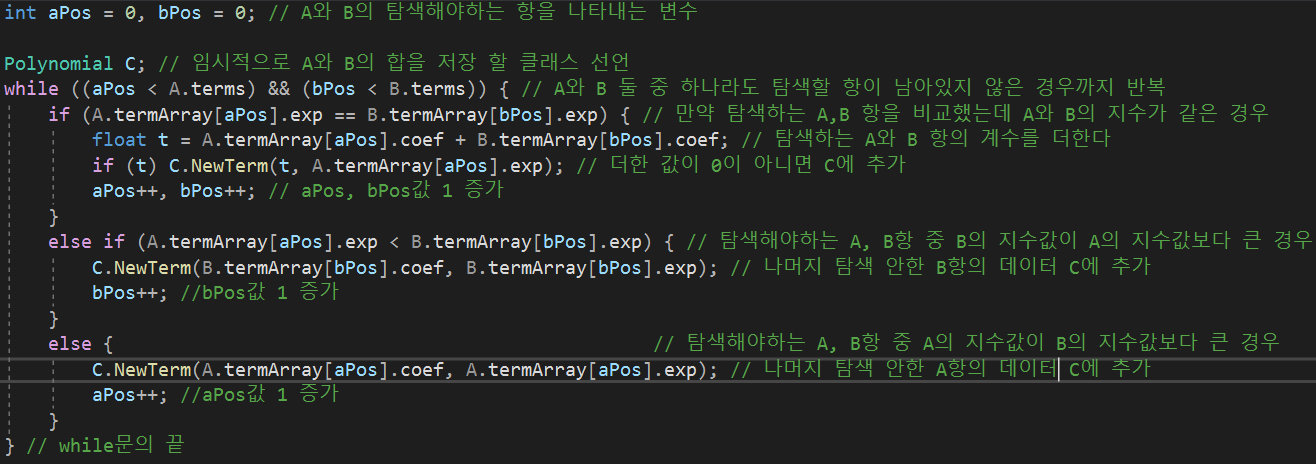
마지막으로 위에서 말한 함수들을 이용해 입력 받은 다항식 A, B를 main함수에서 활용하자면 다음과 같다. 즉, 정리, 정렬, 출력이라는 단계를 거쳐서 정돈된 다항식들을 출력했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

★ 두 다항식의 곱을 구현하는 Mul 함수를 만들고 이를 이용해서 다항식 A, B의 곱의 결과를 다항식 C에 저장하고 다항식 C를 출력하기

앞에서 말했듯이, Mul 함수의 알고리즘은 먼저 A의 첫번째 항부터 시작해서 마지막 항까지 차례대로 B의 모든 항과 곱하고 그 결과 차례대로 생성된 다항식들을 계속 더하는 것이다. 즉, Add라는 두 다항식의 합을 구현하는 함수를 먼저 만들어야 된다.



위는 Add함수의 주요 코드이다. 여기서 핵심은 aPos, bPos라는 탐색해야 하는 A, B항의 index를 저장하는 변수를 이용해 풀었다는 것이다.

여기서 주의할 점은 객체 A, B의 다항식이 정돈되지 않은 다항식인 경우 객체 C에 저장될 다항식도 정돈되지 않을 수 있다는 것이다.

일단은 객체 A, B의 다항식이 정돈되어 있다고 가정하고 문제에 접근하자.

계속 코드를 보면, 객체 A, B를 참조변수로 가져왔는데 aPos, bPos 값과 A, B의 terms값을 비교해 가면서 A와 B 둘 중 하나라도 탐색할 항이 없는 경우까지 반복을 한다. 그 과정에서 접근하는 항의 지수가 같으면 서로 더한 값을 C에 저장하고 같지 않으면 더 큰 지수를 가지고 있는 항을 C에 저장하기를 반복한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이후 for문을 이용해 아직 다 접근하지 못한 객체의 항들을 C에 저장하고 마지막에 CopyObject 함수를 이용해 객체 C의 데이터를 (\*this)에 복사한다. 이때 복사된 다항식은 정돈된 상태이다.

이제 Add 함수를 이용해서 Mul 함수를 구현하면 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

여기서 이중 for문을 이용해 참조하는 객체 A다항식의 첫번째 항부터 시작해서 마지막 항까지 차례대로 참조하는 객체 B 다항식의 모든 항과 곱하고 그 결과 차례대로 생성된 다항식(temp)를 계속 (\*this)에 더한 것이다.

이때 생각할 수 있는게 A, B다항식이 정돈된 상태라면 안쪽 for문에서의 연산이 끝난 temp 다항식은 저절로 정돈된 상태라는 것이다. 즉, (\*this)의 다항식이 또한 정돈된 상태라면 이후 Add함수를 통해 생성된 객체도 정돈될 것이라는 것이다.

하지만 여기서 한가지 예외 상황은 A, B 둘 중에 하나라도 다항식이 “0” 이라면 정리가 필요하다.

이에 다음같이 main함수에서 Mul함수를 호출하고 한번 정리 과정을 거치고 나서 출력해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

★ 변수 x 값을 입력 받아 A, B, C 다항식에 x 값을 대입하고 그 결과를 각각 출력하기

마지막으로 Eval함수를 이용해서 각 A, B, C 다항식의 변수에 값을 주었을 때 결과를 출력해주도록 하자

텍스트, 화면이(가) 표시된 사진

자동 생성된 설명

결국 핵심 알고리즘은 먼저 매개변수 x를 통해 변수의 값을 입력 받고, for문을 이용해 모든 항에 접근한 후 그 항의 계수와 지수를 이용해서 그 항의 값을 구한 뒤 total이라는 변수에 차곡차곡 저장하는 것이다. 또 이중 for문을 사용해 각 항의 거듭제곱을 구현했다.

이후 다음과 같이 main함수에서 변수 값을 x를 입력 받고 Eval함수를 적절히 썼다.

텍스트이(가) 표시된 사진

자동 생성된 설명

최종적으로 아래와 입력한 후 실행결과를 보여주자면 다음과 같다.

(구분하기 쉽게 계수는 초록색, 지수는 주황색으로 색칠했다.)

다항식 A 입력 : -2 6 3 2 -1 1 2 6

다항식 B 입력 : 0 4 1 1 4 0 0 0

대입할 값 x 입력 : 1

텍스트이(가) 표시된 사진

자동 생성된 설명

보다시피 원하는 대로 곱 연산, 값 계산 연산이 잘 이루어졌다 .

그리고 지수가 같은 항끼리는 서로 더해서 하나로 만들었는데 계수가 0이라서 삭제된 것과 처음에 입력했을 때 계수가 0이면 출력이 안된 것을 볼 수 있다.

마지막으로 상수이거나 계수가 1 또는 -1일 때에도 보기 좋게 출력되었다.

### 결론

새롭게 배운 점을 말하자면 new연산을 통해 동적할당에 실패한 경우 어떻게 해야 할 지에 대해서이다. 또한, 동적할당 된 클래스는 기본적으로 대입연산에서 자유롭지 못하다는 점을 알았다. 그래서 따로 생각해 CopyObject함수를 사용한 것이다. 비록 아직까지는 위 개념에 대해서 정확하게 이해는 못했지만, 앞으로 배워 나가면서 부족한 부분을 채워가겠다.

그리고 지금까지 계산한 다항식은 정말 기본적인 다항식의 모습이라고 할 수 있다. 예를 들어서 지수에 sin 또는 cos을 집어 넣는 등, 다항식을 복잡하게 한다면 정말 복잡하게 만들 수 있을 것이다. 실제로도 그런 다항식이 실생활에서 자주 쓰일 것 같다. 그런데도 내가 작성한 코드를 보면 매우 난잡한데, 복잡한 다항식의 경우는 상상하기도 싫다. 이에 앞으로 더욱 더 열심히 자료구조에 대해서 공부하고 코딩 능력을 기르려고 한다.