**REPORT**

로고이(가) 표시된 사진

자동 생성된 설명

**과제명 : 다항식 , 중위식 변환**

1. **다항식 덧셈 : 입력 수정(함수)**
2. **중위식 변환 :** 
   1. **중위식 -> 전위식**
   2. **변환식 같은 변수에 저장.**
   3. **한자리 수 이상 계산**

**과목명 : 자료구조 심화학습**

**교수명 : 심종익 교수님**

**학 년 : 3학년**

**학 번 : 201901366**

**성 명 : 서희준**

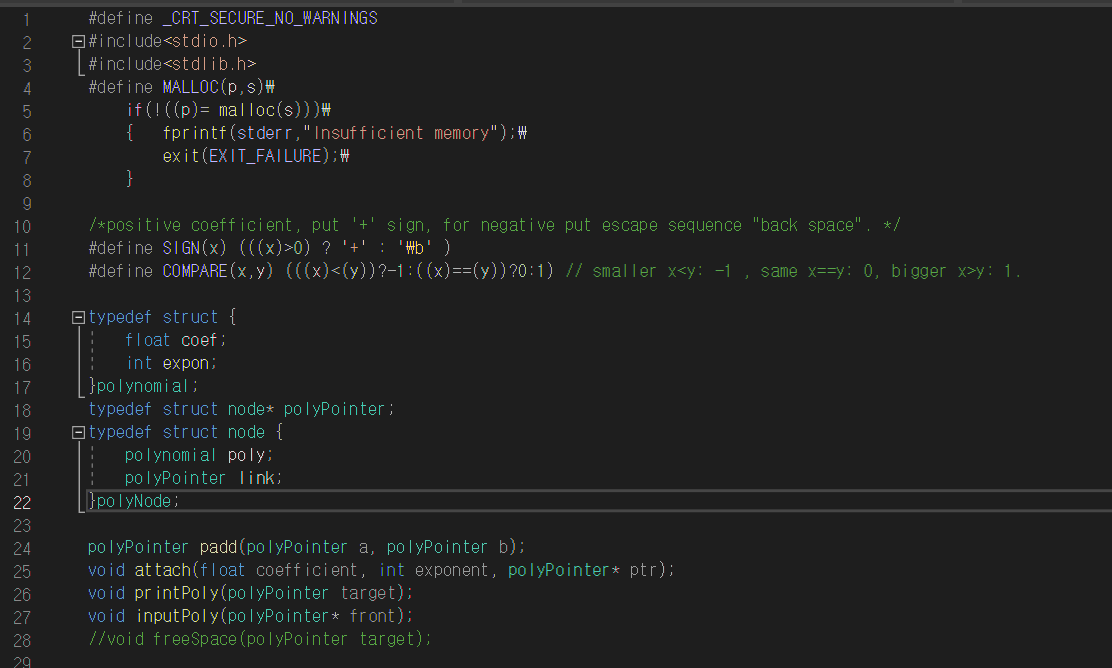
**제출일 : 23.07.08.(토)**

**텍스트이(가) 표시된 사진

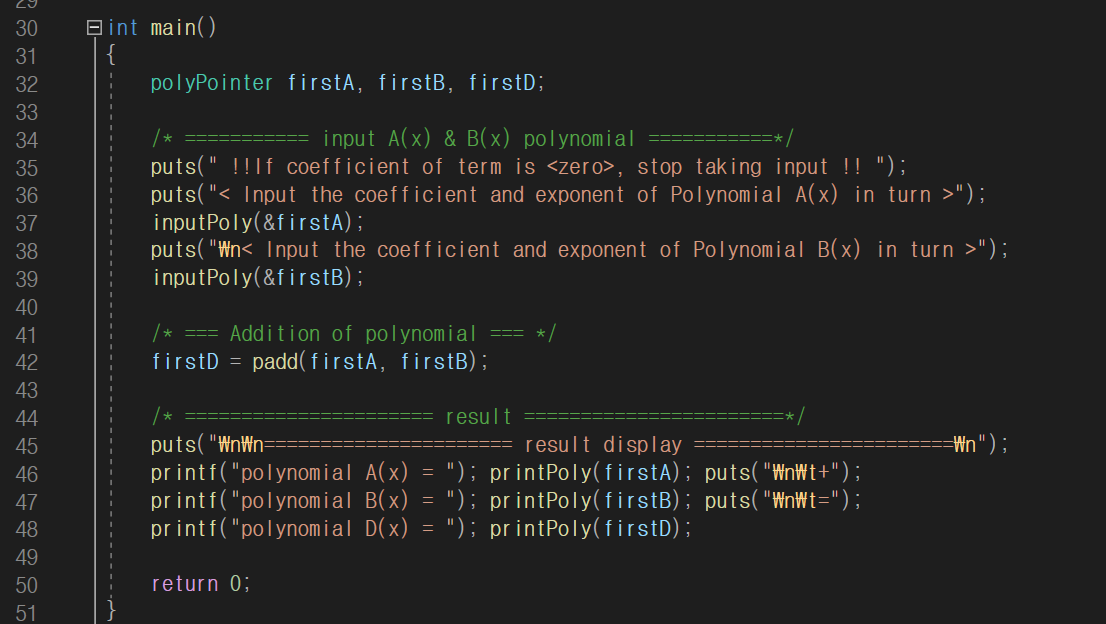
자동 생성된 설명**

**<목차>**

1. **다항식 덧셈 : 입력 파트(수정) -> attach 함수 이용**
2. **중위식 변환 :**
3. **중위식 -> 전위식**
4. **변환식 같은 변수에 저장**
5. **한자리 수 이상 정수 계산**
6. **다항식 덧셈 입력 파트 수정: attach 함수 이용**

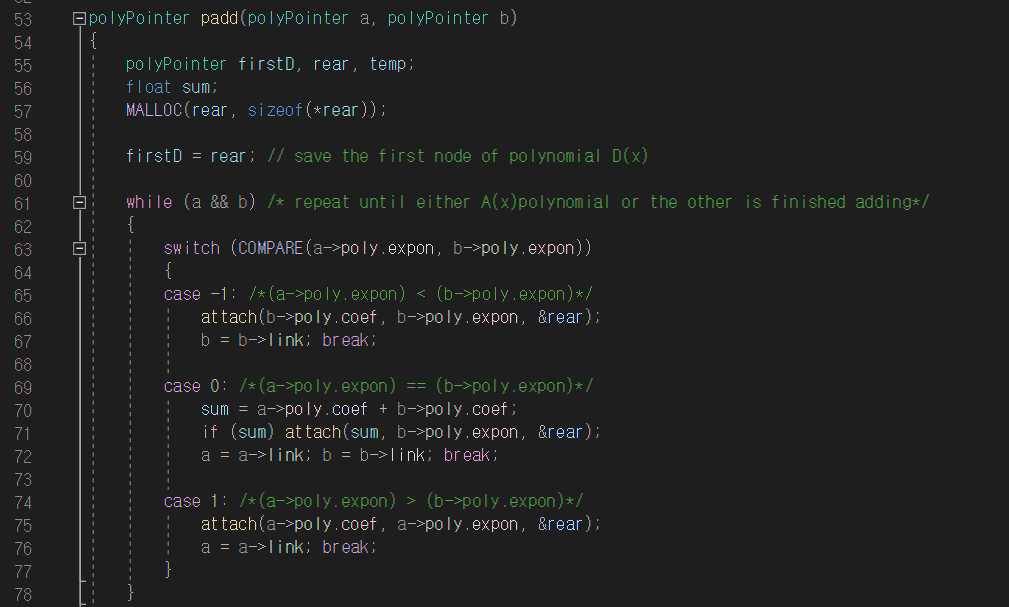
****

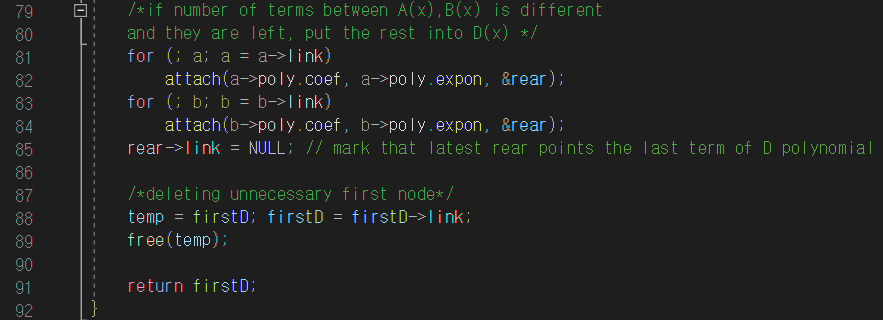
기존 프로그램의 main함수에서 반복되는 부분을 함수로 변경하였다. 초기에 다항식을 입력 받는 부분과 다항식을 출력하는 부분은 반복되기에 함수에 첫번째 노드의 주소를 전달하는 방법을 사용한다. 수정된 메인 함수는 아래와 같다.

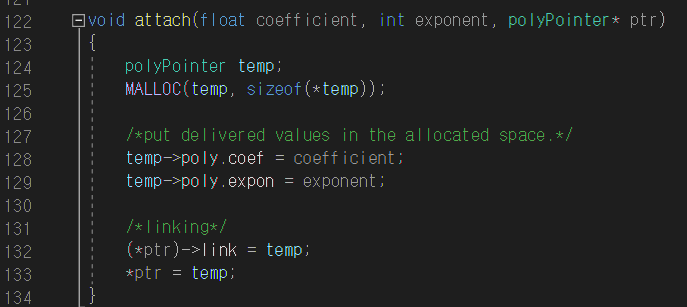
****

line 37, 39 : inputPoly함수에 firstA, firstB 포인터 변수의 주소를 전달함으로써, 함수의 반환 값이 없이도 firstA와 firstB가 각 다항식의 첫번째 노드를 가리킬 수 있도록 했다.

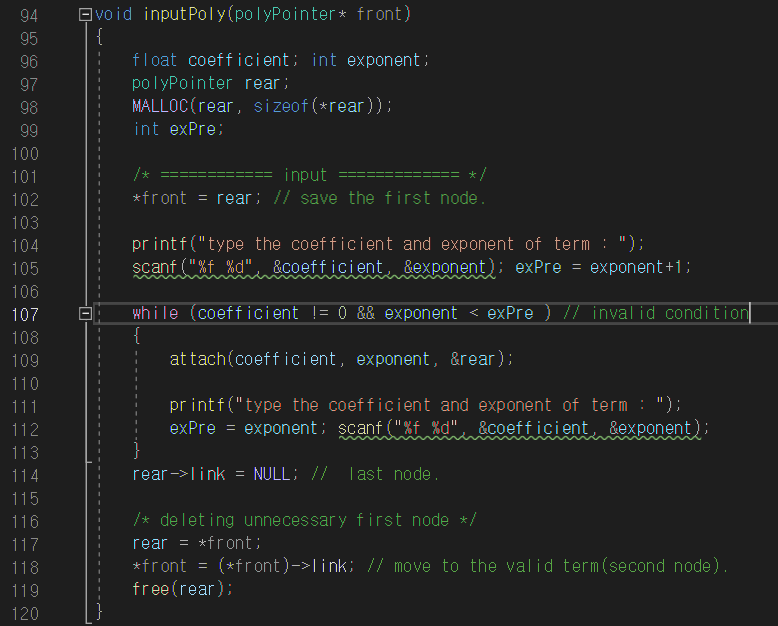
line 46, 47, 48 : printPoly함수에 firstA, firstB, firstD가 가리키는 주소를 전달하여, 각 다항식을 출력했다. 기존 프로그램에서는 메모리 반환 과정을 메인 함수에서 처리했으나, 이 프로그램에서는 printPoly함수에 반환과정을 포함하였다.

****

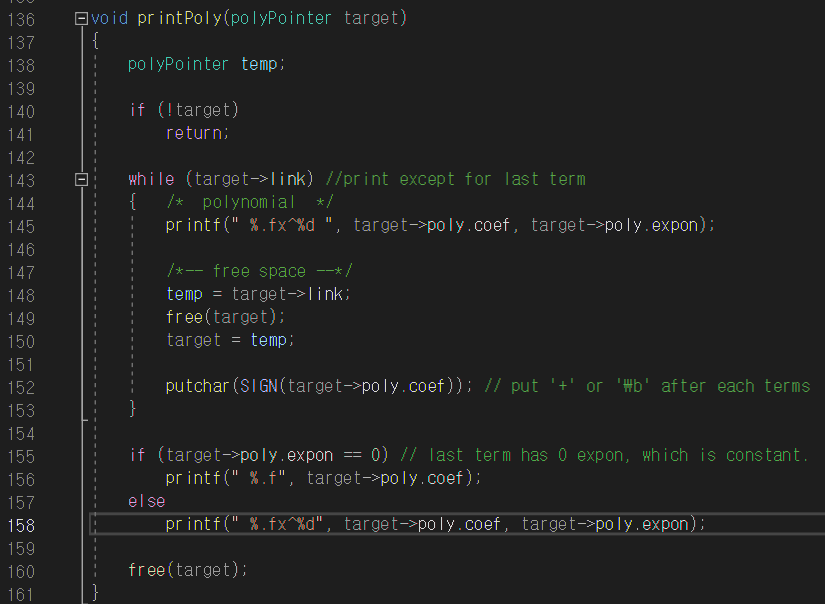
****

****

위의 attach, padd함수들은 동일하나, attach함수는 다항식을 입력 받을 때 이용한다. attach함수는 A, B다항식을 더해 D다항식을 만드는데 사용하였다. Line 125에서 각 항에 대한 공간을 할당 받고, line 128,129에서 그 값을 저장한다. 마지막으로, 그 다음 항의 연결을 위해 line 132,133에서 링크과정이 있다. 이는 다항식의 개별 항을 만들고, 연결하는 함수이기 때문에 A, B 다항식을 만들 때도 활용할 수 있다. 입력함수 inputPoly()는 아래와 같다.

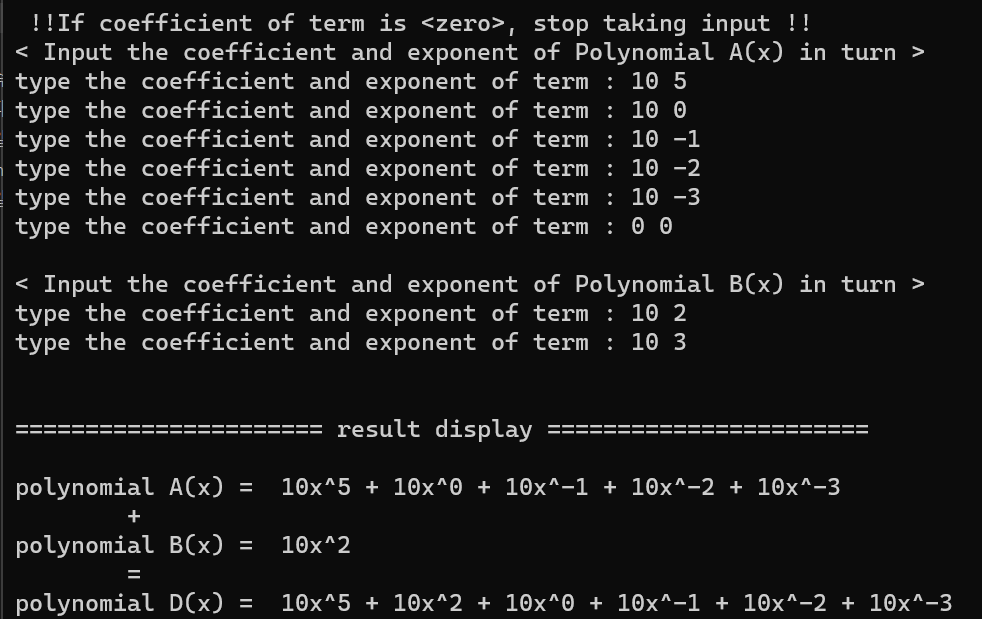


지수와 계수의 값을 입력 받을 변수 exponent와 coefficient함수를 선언하여, 입력 받은 값이 유효할 때 attach()함수를 호출한다. 이때 다항식의 첫번째 노드의 주소를 main함수의 각 포인터에 저장하기 위해 더블 포인터 매개변수를 사용한다. 이로써, main함수의 firstA, firstB 포인터 변수의 공간에 직접 참조가 가능하다. 하지만, inputPoly()함수에서 만들어지는 첫번째 항의 위치를 기억하기 위해선 ‘\*’연산자를 사용해야 한다. Line 102를 보면, 다항식의 첫번째 노드를 저장한다. 이후 while문에서 각 항을 입력 받아, 다항식을 생성한다. while문에서도 조건문을 수정하였다. 기존에는 무한루프에 계수가 0일 때 break문을 사용하였다면, 이 함수에서는 계수가 0이거나, 입력한 계수가 이전 항의 계수보다 작거나 같을 때 거짓이 되도록 하였다. 이 프로그램의 가정된 입력조건으로는 2가지가 있다. 첫번째는 계수가 0인 항은 공간을 절약하기 위해 입력하지 않는 것이고, 두번째는 각 항의 지수를 내림차순으로 입력해야 한다는 것이다. 단순히 attach함수를 호출하여, 항을 만들고 내림차순으로 링크하기 때문에 위와 같은 조건에서 벗어나는 입력은 잘못된 값임을 명시해야 한다. 이를 에러출력으로 표시할 수도 있지만, 이를 while문의 조건문으로 사용해서 잘못된 입력을 제외하는 방법을 선택했다. 이전 항의 지수의 값을 저장하기 위해 exPre 변수를 추가로 선언하였고, 첫 번째 항의 경우 이전 항이 없으므로, line105과 같이 1회 while문 밖에서 입력 받았다.

****

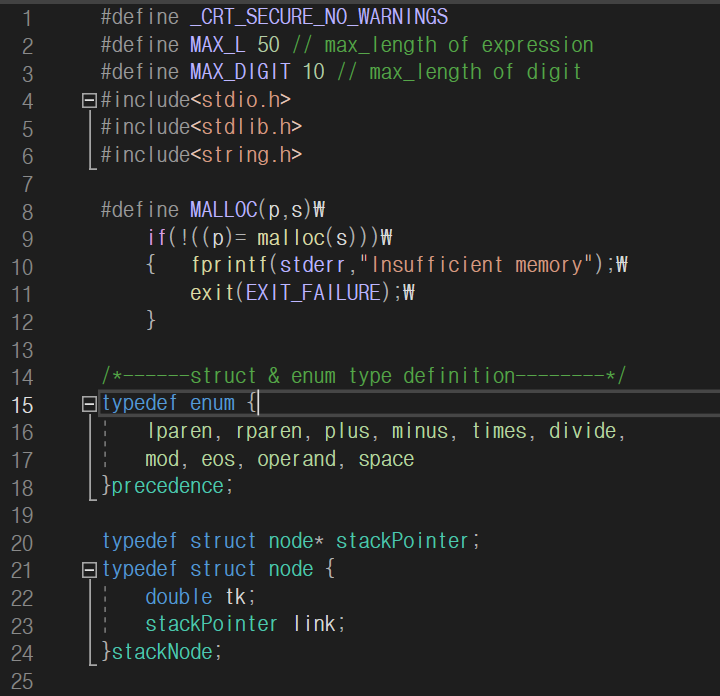
다항식을 출력하는 부분도 반복되기 때문에, 함수로 만들어 다항식을 출력한다. 이때 메모리를 반환하는 부분도 추가하였다. 다항식 출력 시 부호가 각 항 뒤에 붙기 때문에 마지막 항은 while문 밖에서 출력한다. 따라서 조건문을 line 143과 같이 하여 마지막 항을 가리킬 때 거짓이 되도록 하였다. 또 한가지 수정한 사항으로, 다항식이 없을 때를 가정하여 출력없이 함수를 종료하는 부분을 line 140과 같이 추가하였다.

**<결과>**

****

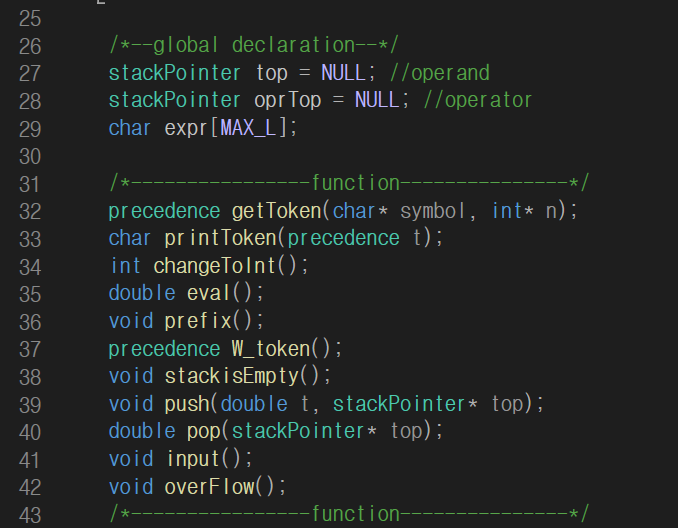
입력 조건에 벗어난 입력 시 반복 루프가 종료되는 것을 확인할 수 있다. 지수가 내림차순이고, 계수가 0이 아니면, 계속 입력 받아 덧셈 후 출력된다.

1. **중위식 변환 : 중위식->전위식(같은 변수에 저장), 한자리 수 이상 계산**



이번에는 중위식을 전위식으로 바꿔, 전위식으로 계산하는 프로그램이다.

line 3 : 한자리 수 이상의 정수를 가지고, 연산하기 위해 배열의 크기를 10으로 한정하였다. 이는 문자열의 마지막 NULL을 제외한 9자리 정수까지 허용한다.



중위식을 전위식으로 바꾸고, 전위식으로 계산하는 과정에서 두개의 스택을 사용할 것이다. 따라서, top을 두개로 독립 유지한다.

함수를 보면 changeToInt(), input(), overflow()함수가 추가되었다. 이전에 한자리 수 프로그램을 계산할 때는 “push((double)symbol - '0'); “처럼 간단하게 문자를 숫자로 변환했지만, 이 프로그램에서는 한자리 수 이상의 정수를 다루기 때문에 문자열(한자리 수 이상 정수)을 정수형으로 변환해주는 함수를 독립적으로 생성했다. 앞서, 언급한 바와 같이 chageToInt()함수에서 문자열(한자리 수 이상 정수)을 정수형으로 변환해줄 때 그 자릿수를 9자리(MAX\_DIGIT)로 한정했다. 따라서 중위식을 입력 받을 때 숫자부분을 나타내는 문자열이 9자리를 초과하면 안 된다. 이 부분을 input()함수와 overFlow()함수를 이용해 에러 처리하였다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

메인 함수는 3가지 부분으로, 중위식을 입력 받고 출력, 전위식으로 변환하고 출력, 전위식을 계산하여 출력한다.

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

이전 프로그램에서는 “scanf("%s", expr);”과 같이 한 번에 문자열로 중위식을 입력 받았지만, 현 프로그램에서는 한자리 이상의 정수를 다루기 때문에 line 68과 같이 한 글자씩 입력 받는다. 반복문은 식의 최대 크기인 MAX\_L보다 작고, “ENTER”가 입력되기 전까지 반복한다.

for문 안에 조건문들은 입력 받을 정수를 9자리 정수로 제한하기 위함이다. Line 69와 같이 숫자(문자)가 입력되면, 연속적으로 입력된 숫자를 카운트한다. 그리고, 숫자(문자)들 사이에 연산자가 있을 때 그 값을 0으로 초기화한다. 만약 카운트한 숫자가 9자리를 초과하면, 9자리를 초과한 정수를 의미하므로, 에러처리(call overflow())한다. 마지막으로, 입력 받은 문자배열을 문자열로 만들어주기 위해, 마지막 count번 공간에 0을 저장한다.

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

스택 구조를 사용할 때, 사용될 pop(), push(), stackisEmpt()함수는 동일하다. 이전에 설명한 바와 같이 중위식을 후위식이나 전위식으로 바꿀 때, 스택에 푸시되는 대상은 연산자이다. 반대로 변환된 식을 가지고 연산을 할 때는 피연산자가 푸시된다. 계산 시 피연산자는 op2/op1의 결과가 될 수도 있는데, 이 때 부동소수점을 가지는 실수가 그 결과가 될 수 있으므로, 스택의 데이터 자료형을 double로 선언하였다. 따라서, push(), pop()함수의 반환형과 매개변수도 double형이다.

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

다음은 prefix()함수이다.

중위식에서 전위식으로 변환하는 개념은 다음과 같다.

중위식을 오른쪽부터 읽어드리는 것이다. 하지만, 출력은 왼쪽부터 출력되도록 해야 한다. 즉, 가장 오른쪽이 가장 마지막에 출력되고, 가장 늦게 읽어드린 부분이 가장 먼저 출력되야 한다. 이는 FILO (first in, last out) 구조로 스택 구조이다. 이러한 출력을 위해 연산자를 비교하는 스택 이외에 출력을 위한 스택을 하나 더 이용한다. 앞서, 설명한 바와 같이 전역으로 선언된 top이 두개인 이유이다. 그렇다면, 출력을 위해 스택에 저장되는 순서를 알아야 한다. 숫자는 읽어드리는 순서대로 “출력” 스택에 저장하고, 연산자는 “비교” 스택에 먼저 저장한 후, pop()될 때 “출력” 스택에 저장한다.

중위식을 오른쪽부터 읽어드리기 때문에 각 연산자에 대한 우선 순위 또한 달라져야 한다. ‘)’ 오른쪽 괄호가 isp는 가장 낮고, icp가 가장 높도록 설정해야 한다. 또 다른 한가지는 연산자의 우선순위를 비교할 때 “isp>icp”의 경우만 pop()된다. 예를 들어 “5+4-3”의 중위식은 “출력” 스택에 3 4 5 + - 순서로 저장되기 때문에 결론적으로 “- + 5 4 3”의 전위식이 생성된다.

한자리 수 이상의 정수를 계산하는 과정은 다음과 같다.

“출력” 스택에 푸시할 때 그 값을 구분하기 위해 “ “(공백)을 이용한다. 값들 사이에 공백을 푸시해서 값을 구분하는 것이다. 예를 들어 123/2\*2를 구분한다고 하면, “\* / 123 2 \* 3”로 구분해야 할 것이다. 따라서, 123과 같이 입력되는 정수들은 1,2,3 문자를 “출력” 스택에 연속적으로 푸시 후 공백을 푸시한다. 그 과정은 getToken()함수에 추가되어 있다.

문자열을 오른쪽부터 읽어드리기 위해 line 134와 같이 strlen을 이용한다. “1+3/2”의 중위식이면 길이가 5이므로, 가장 마지막 인덱스는 4일 것이다. 읽는 인덱스를 i변수로 유지한다.

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

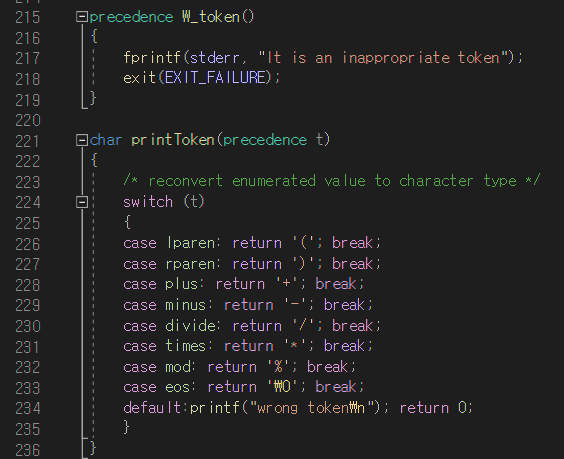
자동 생성된 설명

바로 이어 getToken()함수를 살펴보면, “i--"로, 오른쪽부터 좌측으로 문자열을 읽어드린다. 여기서 달라진 부분은 중위식에서 숫자(문자)를 읽어드리는 부분이다. 숫자를 나타내는 문자일 경우 123과 같이 그 다음 문자도 숫자인지를 파악해야 한다. 따라서, line 204와 같이 그 다음 문자가 0부터 9사이의 문자가 아닐 때까지 그 다음 문자를 읽어드린다. 연 속적으로 읽을 수 있는 숫자의 경우 123과 같이 한자리 수 이상의 정수이다. 따라서, 읽는 과정이 끝난 후에 “operand”를 반환해서 prefix()함수에서 공백을 추가한다. 이때 공백을 prefix()에서 추가해야 하는 이유는 getToken함수를 전위식 계산에서 재사용하기 위함이다. 공백을 기준으로 각 문자들을 정수로 변환할 때 연속된 숫자는 연속적으로 읽어드리는 과정이 필요하다.

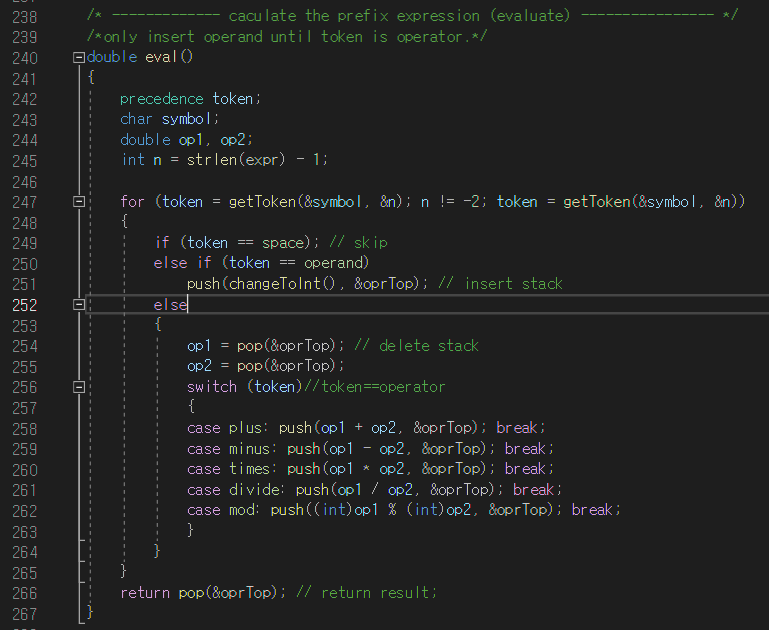
텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

중위식의 모든 문자들을 읽어드린 후에는 “비교” 스택에 있는 연산자를 모두 pop하여 “출력” 스택에 저장한다. 마지막으로 “출력” 스택에 반전된 전위식을 expr함수에 저장한다.

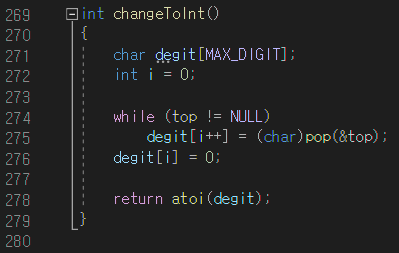


printToken함수와 W\_token()함수는 이전 프로그램과 동일하다.



전위식을 가지고 연산하는 과정도 우측에서 좌측으로 읽어드리며 계산한다.

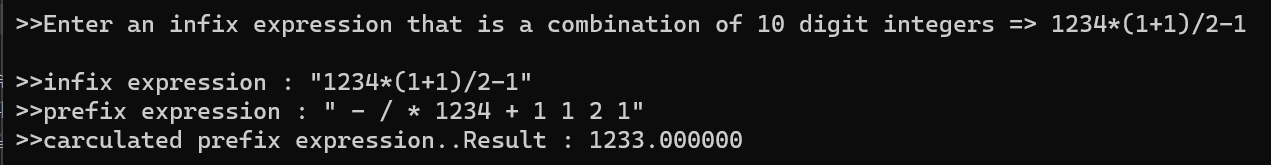
expr배열에 저장된 전위식 문자열은 “ “공백이 포함되어 있다. precedence(enum)타입에 space를 추가해서, 전위식을 getToken함수에서 읽어드릴 때, eval()에서 공백을 처리할 수 있도록 하였다.



피 연산자일 때 마찬가지로, getToken()함수에서 문자를 연속적으로 읽어드린다. 이때 기억해야 할 것은, 읽어드린 문자들을 이전에 “출력”으로 사용한 스택에 푸시했다는 것이다. 따라서, eval()함수에서는 operand가 반환되었을 때, “출력” 스택에 입력된 문자를 chageToInt()함수에서 pop()하여 정수형으로 변환한다. 이때 pop한 문자열을 저장할 배열의 크기가 MAX\_DIGIT이기 때문에 최대 9자리 정수까지 입력을 제한해야 한다. 이러한 이유로 input()함수가 필요했다.

연속된 숫자(문자)들을 getToken()함수에서 연속적으로 푸시할 때 “top”을 이용하였다. 따라서, eval()함수에서 계산된 피 연산자는 oprTop을 이용한 스택에 저장한다. 결론적으로 계산된 값 하나만 “oprTop”이 가리키는 위치에 있을 것이다.

<결과>



최대 9자리 정수를 중위식으로 입력할 수 있고, 9자리를 벗어날 시 아래와 같이 출력된다.

