**C++ MyCalculator**

201935223 김민석

202031322 김소빈

202136031 정명균

**프로그램 구조 (클래스 다이어그램)**



**실행 방법**

1. Stringstream 을 이용하여 공백을 인식하고, 공백마다 끊어서 문자열을 따로 스택에 저장하기 때문에 각각의 연산자 / 피연산자가 끝날 때 마다 공백을 입력해 주어야 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. while루프를 사용해 연산식을 계속 입력할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 0으로 나누는 것을 시도하면 오류 메시지를 출력한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. exit을 입력하면 프로그램을 종료할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 프로젝트 폴더의 result.txt 파일에 출력 결과가 동일하게 저장된 것을 확인할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 피연산자 연산자 피연산자 순으로 공백으로 끊어주며 작성한다. 연산자 또는 피연산자가 연속될 경우 계산 순서 등이 섞여 오류가 발생할 수 있다.

오류 상황 1) 피연산자 연산자 피연산자 피 연산자 순의 입력

연산자에 비해 피연산자의 숫자가 과하게 많을 때이다. 스택을 이용해 연산자와 피연산자를 저장했다가 뒤쪽에 넣은 순서대로 다시 pop 해서 연산하기 때문에 (스택의 성질) 예시의 경우 뒤쪽에 있는 피연산자 둘과 입력 받은 연산자로 연산을 실행하고, 그 값을 출력하게 된다. 4 + 3 1을 입력해

연산자 스택에는 + 하나가 저장되고 피연산자 스택에는 4 3 1 순서로 저장된다.

이후 스택이기 때문에 저장한 순서의 역순으로 뽑아내므로 1 3 4 순서로 뽑아내 연산하기 때문에 1 과 3 을 + 연산에 맞춰서 연산하는 것이다. 그 결과 값으로 4 가 출력된다.

텍스트이(가) 표시된 사진

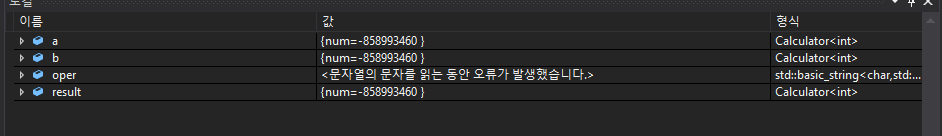
자동 생성된 설명

오류 상황 2) 연산자의 수가 피연산자의 수와 같거나 더 많을 때.

예시의 경우 피연산자 5와 2, 연산자 + - 를 입력받았는데 연산자를 입력받아 저장하는 스택인 oper에서 오류가 발생하면서 오류 메시지를 띄우게 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명



텍스트이(가) 표시된 사진

자동 생성된 설명

**개발 과정 (기본 기능)**

1. Template 함수 상속에 대하여 공부한다.

2. 연산자 함수를 가상함수로 정의하는 Base 함수와 Base 함수를 상속받는 Calculator 함수를 만들고 Calculator 함수에서 연산자 함수를 재정의한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

3. 각각의 연산자를 overloading 해준다.

텍스트이(가) 표시된 사진

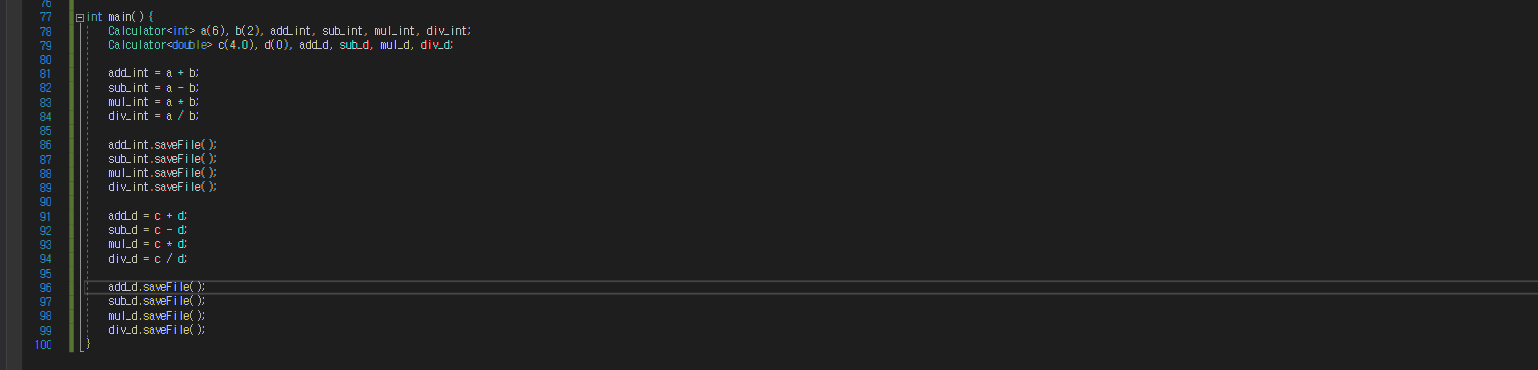
자동 생성된 설명

4. 이때, 나눗셈에서는 0으로 나누는 것으로 인한 오류에 대해 처리를 하기 위해서 try와 catch를 사용해 오류처리를 하도록 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

5. 메인함수에서 Caculator 함수에 대해 int와 double 각각 두가지 자료형을 받도록 오버로딩하고, 각각 자료형으로 변수들을 선언해준다.

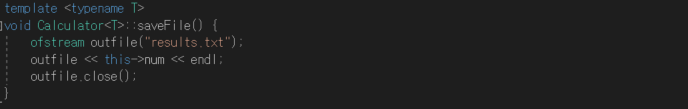


그 후 계산 결과를 results.txt 문서에 저장하도록 한다.

**개발 과정 (추가 기능)**

기존 코드의 문제점

1. saveFile이라는 함수로 파일로 출력하는 기능을 구현했지만 매번 덮어쓰기 때문에 새로운 결과를 출력할 때마다 초기화되는 문제가 발생한다. 따라서 연산을 여러 개 수행하면 마지막 연산의 결과만 파일에 출력되게 된다.



1. 0으로 나누는 오류를 처리한 후 오류 메시지를 연산자 함수 operator/에서 출력하도록 했다. 하지만 파일에 출력할 때가 아닌 연산자 함수가 호출이 될 때 오류 메시지가 출력이 되어 버리는 문제가 발생한다. 따라서 같은 연산에서 오류 메시지가 한 번 출력되고 파일에 계산 결과를 출력하는 함수인 saveFile을 호출하면 잘못된 계산 결과값이 한 번 더 출력되게 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

수정된 코드에서 해결한 방법

1. 파일을 open 함수로 열고 std::ios\_base::app 옵션(append 옵션)을 사용해 파일에 덮어쓰기가 아닌 추가를 하도록 변경했다.
2. 클래스 멤버 변수 errorFlag를 만들어 오류가 발생하면 플래그가 true로 바뀌어 saveFile함수에서 출력을 처리하도록 변경했다. 즉, 오류 메시지를 출력하거나 정상적으로 계산 결과를 출력하거나 둘 중 하나만 출력하도록 중복 출력의 문제를 해결했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

추가 기능 1

cin을 사용해 사용자로부터 연산식을 입력 받도록 main함수에 코드를 새로 작성했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

먼저 사용할 변수들을 선언했다. aInt, bInt와 aDouble, bDouble은 각각 int형과 double형 피연산자를 저장하기 위한 변수이다. userInput은 cin으로 사용자로부터 입력 받은 문자열을 담기 위한 변수이고, 이 문자열을 연산자와 피연산자로 나누어 연산자 문자열은 op에, 피연산자 문자열을 a, b에 담기게 된다. operators는 userInput문자열을 나누는 과정에서 연산자를 찾기 위한 비교 연산을 수행하는 과정에서 필요한 연산자들을 담아 놓은 문자열이다. isInt는 사용자가 입력한 연산식에서 피연산자가 int형이면 true를, double이면 false의 값을 가지게 된다. 마지막 두 줄의 Calculator형으로 선언된 변수들은 Calculator<int>와 Calculator<double>로 구분되어 각각 실제 연산에 사용될 피연산자와 결과값을 담은 객체들이다.



우선 cin을 사용해 사용자로부터 연산식을 문자열로 입력 받는다. 이때 연산식에는 공백이 없어야 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

사용자로부터 입력 받은 문자열 userInput에서 연산자를 찾는 과정이다. string::iterator를 사용해 비교할 연산자가 담긴 문자열 operators를 for 루프로 하나씩 순회한다. find함수를 사용해 userInput에 해당 연산자가 있다면 found변수에 연산자가 발견된 위치의 인덱스를 저장하고 for루프를 종료한다.

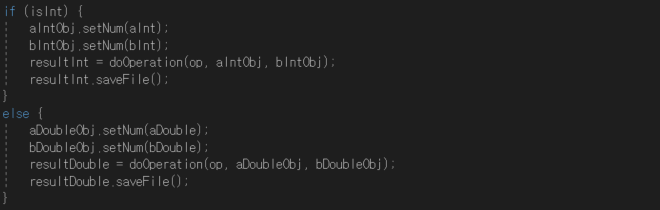


substr 함수를 사용해 연산자의 위치를 기준으로 앞, 뒤에 있는 피연산자를 잘라내어 각각 변수 a, b에 담는다.

텍스트이(가) 표시된 사진

자동 생성된 설명

If-else문으로 userInput에 점(“.”)이 있다면 피연산자가 double, 없다면 int인 것으로 판단해 각각 double 또는 int형으로 변환을 해 해당 변수에 담아준다. isInt변수도 피연산자의 자료형에 따라 false 또는 true 로 바꿔준다.



If-else문에서 isInt변수를 사용해 int로 연산을 하는지 double로 연산을 하는지 구분을 해준다. 이후에 setNum이라는 클래스 멤버 함수로 객체에 피연산자에 해당하는 숫자를 입력해주고 doOperation 함수로 연산을 수행한 다음 saveFile함수로 파일에 결과를 출력해 준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

doOperation함수에서는 첫번째 인자로 사용자가 입력한 문자열에서 찾아낸 연산자가 담긴 문자열을 str이라는 변수로 전달 받는다. If-else문으로 str이 어떤 연산자인지를 찾고 해당 연산을 수행한다. 마지막으로 결과값이 담긴 Calculator 객체 tmp를 리턴한다.

추가 기능 2

1. 계산 히스토리 목록처럼 만들고 싶어서 계산식+계산결과를 파일에 함께 출력하도록 변경했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

saveFile함수가 userInput 문자열을 인수로 받아 결과값과 함께 출력하도록 변경했다.

1. 파일에 출력하는 내용을 콘솔에도 출력하도록 변경했다.

구현은 동일하게 saveFile함수에서 파일 출력 후 cout으로 동일한 내용을 한 번 더 출력하도록 추가 했다.

1. While 루프를 추가해 사용자가 연산이 한 번 끝나도 계속 입력할 수 있도록 변경했고, ‘exit’이라고 입력을 하면 프로그램이 종료되도록 변경했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

While 루프를 추가하고 사용자로부터 입력 받은 문자열이 “exit”과 일치하면 while루프에서 break해 프로그램이 종료되도록 했다.

추가 기능 3

스택을 사용해 다항 연산을 처리할 수 있도록 변경했다.

스택을 사용하여 처리하는 부분은 다음 웹사이트의 글을 참고했다.( <https://penglog.tistory.com/99>)

텍스트이(가) 표시된 사진

자동 생성된 설명

우선 연산자를 나타내는 구조체를 선언했다. 구조체는 연산자 우선순위를 나타내는 변수 p와 연산자 자체를 나타내는 변수 o로 구성되어 있다. numInt와 numDouble은 각각 Calculator<int>와 Calculator<double>의 피연산자가 담길 숫자 스택이고, op는 앞에서 선언한 연산자 구조체가 담길 연산자 스택이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

기존의 doOperation함수를 doOperationInt와 doOperationDouble로 나누어 int일때와 double일 때의 연산을 각각 수행하도록 했다. 함수 내부도 스택을 사용하여 연산을 하기 때문에 숫자 스택에서 숫자 두 개를 pop해서 연산을 수행하도록 변경되었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

main함수의 while 루프 내부를 살펴 보면 스택 연산 때문에 공백이 있는 문자열을 사용자로부터 받도록 변경되었기 때문에 cin이 아닌 getline함수를 사용하였다. getline으로 받은 문자열을 >>연산으로 나누기 위해 stringstream형으로 변환해 주었다.

기존 코드의 userInput문자열을 처리해 연산자와 피연산자로 나누는 부분들은 모두 버리고 find함수를 사용해 사용자가 입력한 연산식의 피연산자가 int형인지 double형인지 구분하는 if-else문만 가져와 사용했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

나머지 코드 부분은 스택 연산을 처리하는 부분이다. >> 연산으로 ss에서 공백을 만날 때마다 끊어서 문자열을 tok에 저장하고tok을 분석해 연산자면 우선 순위를 부여해 연산자 스택에 push하고, 피연산자면 숫자 스택에 push한다. 연산 수행시에는 연산자 우선 순위에 따라 연산을 수행한다.

추가 기능 3 구현 후 문제 및 해결 방법

기존 코드에서는 연산을 한 번만 수행 했기 때문에 문제가 되지 않았지만 연산을 여러 번 수행 하면서 0으로 나눴을 때 오류 메시지 처리 부분에서 문제가 발생했다. 예를 들어 4 / 0과 같은 연산을 수행하면 오류 메시지가 정상적으로 출력이 되지만 4 + 4 / 0 + 4와 같은 식으로 0으로 나눈 후에도 추가로 연산을 수행하면 오류 메시지가 아닌 이상한 결과가 출력되는 문제가 생겼다.



기존에 오류 발생 상태를 나타내는 errorFlag변수는 Calculator클래스의 멤버 변수로 선언되어 있었다. 하지만 0으로 나눈 후에 연산을 한 번 더 수행하게 되면 Calculator 클래스가 한 번 더 생성이 되면서 errorFlag변수가 초기화 되는 것이 문제가 되었다. 따라서 errorFlag 변수를 클래스의 멤버 변수가 아닌 전역 변수로 설정하여 초기화되는 문제를 해결했다.

추가 기능 4

제곱 연산을 수행할 수 있도록 코드를 추가했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

Calculator 클래스에 operator^를 추가로 선언하고 cmath라이브러리를 사용해 제곱 연산을 구현했다.

**문제 해결 과정**

1. <문제> 템플릿 클래스를 상속받고 상속된 클래스 내에서 연산자 중복 함수를 선언 시 오버로딩하는 방법을 찾는데 어려움이 있었다.

<해결 방법> 템플릿 클래스를 상속받는 부분은 교수님이 공유해 주신 예제를 참고해 해결할 수 있었다. 연산자 중복 함수와 관련된 부분은 인터넷에 추가적으로 검색도 해보고 수업 자료도 찾아 가면서 서로 시도해본 방법을 공유하면서 해답을 찾았다. 결국 연산자 중복도 함수로 선언하는 것이기 때문에 보통 멤버 함수와 동일하게 virtual로 가상 선언이 가능함을 깨닫게 되었다.

1. <문제> cin으로 사용자 입력을 받을 때 피연산자의 int, double형을 구분하는 데 어려움이 있었다.

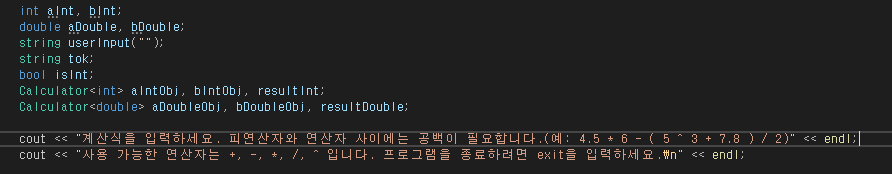
<해결 방법> 사용자가 연산식을 입력하기 전에 int와 double중에 피연산자를 어떤 것으로 입력할 것인지 고르게 하면 간단하게 해결할 수 있었지만 깔끔하게 계산식만 받고 싶어서 다른 방법이 없을까 고민해 보게 되었다. 따라서 사용자로부터 입력 받은 문자열을 여러 스트링 관련 함수를 사용해 처리하게 되었고, 추가적으로 발생한 템플릿 자료형과 관련된 호환성 문제들은 int형, double형별로 변수를 각각 선언하고 if-else문으로 각 자료형 별 수행할 연산을 구분하는 방법으로 해결을 했다.

1. <문제> 다항 연산을 구현하는데 어려움이 있었다.

<해결 방법> 기존 문자열을 처리하는 방법에서 다항 연산도 처리할 수 있도록 어떻게 확장을 하면 좋을지에 대한 고민이 많이 있었다. 인터넷에 검색을 하던 중 스택으로 다항 연산을 구현하는 글을 발견했고 기존 코드에 적용해 보기로 했다. 이번에도 템플릿 자료형 때문에 어려움이 있었지만 각자 시도해 본 방법들을 공유하면서 해답을 발견하게 되었다. 결국 두 가지의 해답을 발견하게 되었고, 동작 원리는 거의 동일했기 때문에 둘 중 덜 복잡한 방법을 채택해 최종 코드에 적용하게 되었다.

**디버깅 및 main함수 주요 동작 과정**

1. 입력을 받기 위한 변수 선언



1. 사용자에게 입력 받기

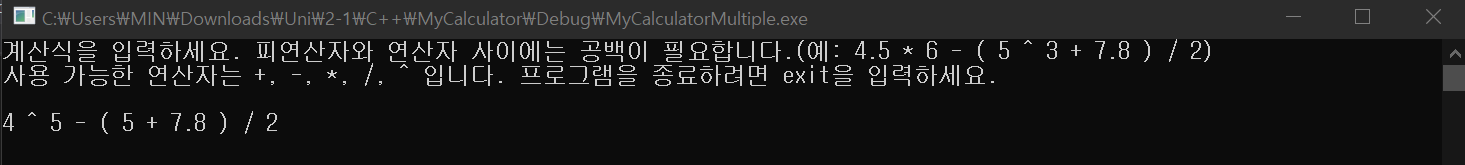
텍스트이(가) 표시된 사진

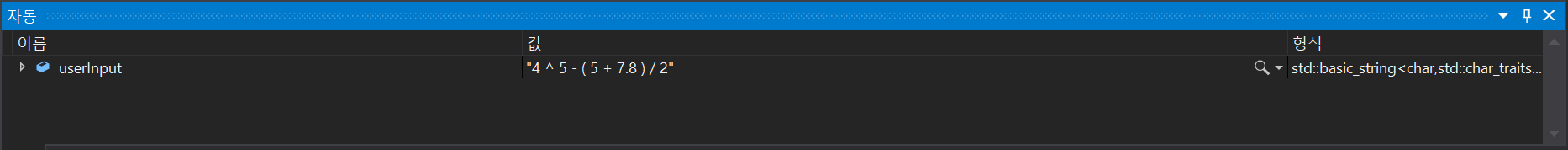
자동 생성된 설명

유저가 종료를 선택하지 않는다면 계속해서 계산을 하기 위해 while(true)로 무한 루프를 형성한다.

그리고 string으로 선언된 userinput에 유저로부터 입력을 받는다.

비교함수인 strcmp를 사용해 userinput에 받은 내용과 exit이 같다면 break로 더 이상 진행하지 않고 반복문을 빠져 나온다.





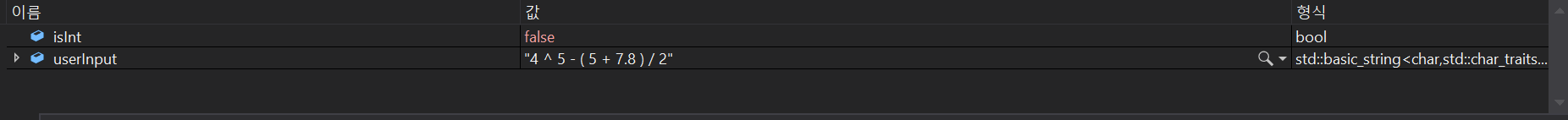
사용자로부터 받은 문자열이 userInput에 담긴 것을 확인할 수 있다. userInput이 “exit”과 일치하지 않기 때문에 다음 단계로 넘어간다.

1. 계산기에 입력 받은 값을 정수와 실수로 분리

텍스트이(가) 표시된 사진

자동 생성된 설명

점(“.”)이 있다면 실수, 없다면 정수로 분류한다.



문자열에서 점을 찾지 못했기 때문에 isInt에 false 가 저장된 것을 확인할 수 있다.

1. 연산자 우선순위 설정

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

스택을 이용한 계산기에서 중요한 것은 연산자 스택과 숫자 스택을 만드는 것이다.

왼쪽 괄호 ‘(’을 만나면 연산자 스택에 무조건 push를 한다. 이게 pop이 되려면 ‘)’을 만날 때 만이다. 하지만 ‘)’는 ‘(’가 없이는 들어갈 수 없다.

왼쪽에 ‘(’이 있고 ‘)’이 들어온 경우는 ( 2+ 4) 처럼 하나의 쌍이 완성됐기에 연산을 시작한다. 연산자의 맨 위가 새로운 ‘(’가 아닐 때까지만 이를 진행한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

연산자의 우선 순위를 위해 연산자 입력을 받은 것에 따라 우선순위를 숫자로 정해준다.

제곱이 가장 높은 우선순위를 갖고, 곱셈과 나눗셈, 덧셈과 뺄샘 순으로 우선 순위를 갖는다.

텍스트이(가) 표시된 사진

자동 생성된 설명

연산자 스택이 비어있지 않고, 연산자의 우선순위가 top에 있는 연산자보다 낮다면 연산을 수행한다.

그게 아니라면 op.push를 통해 연산자의 우선순위와 연산자를 push한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위의 경우가 아닌 경우는 숫자로 간주하여 int판정을 통해 int형과 double으로 바꿔줘서 숫자 스택에 push를 한다.

텍스트, 시계이(가) 표시된 사진

자동 생성된 설명

위의 과정을 다 거치고도 남아있는 것은 연산자 스택이 빌 때까지 연산을 시행한다.

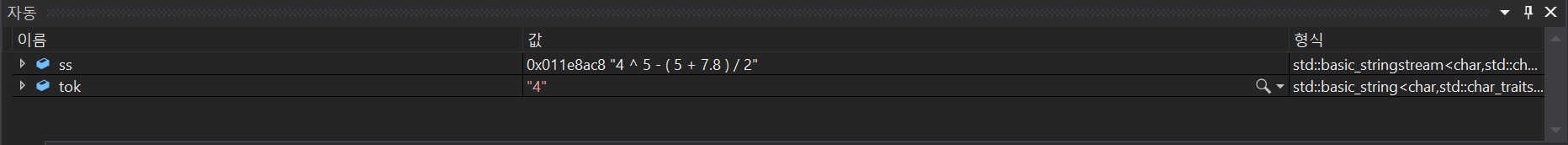
텍스트이(가) 표시된 사진

자동 생성된 설명

결과값을 출력한다.

각 스택의 맨 위의 있는 숫자가 결과값이다.

**<스택 연산 디버깅>**

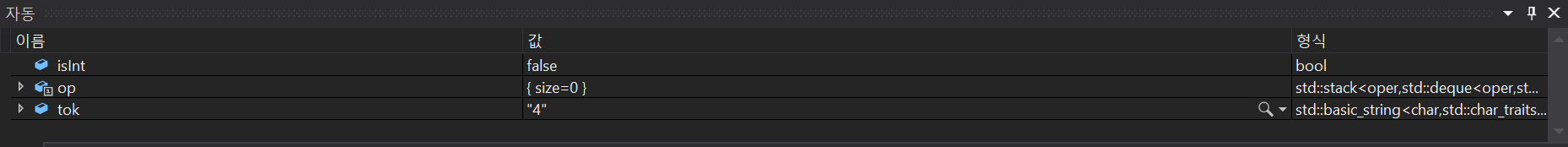


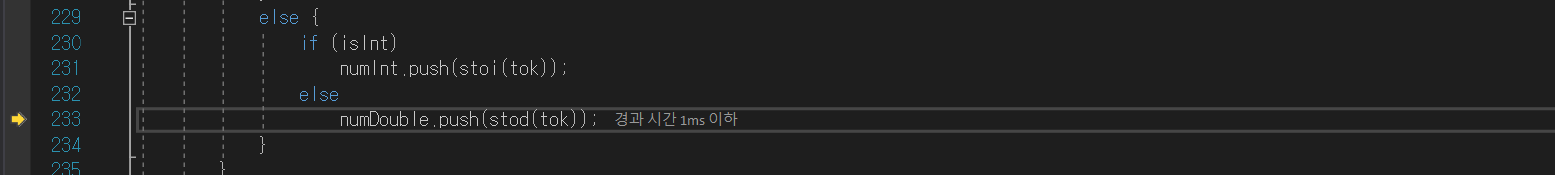
tok에는 우선 첫 번째 피연산자 4가 담긴다.

텍스트이(가) 표시된 사진

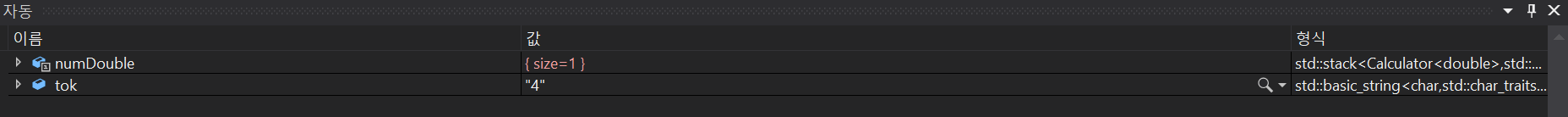
자동 생성된 설명

tok이 연산자가 아니므로 숫자 스택에 push하는 부분으로 넘어간다.

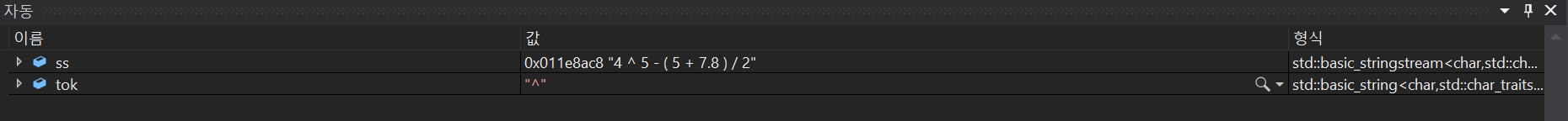




isInt가 false이므로 double형 숫자 스택에 push한다.



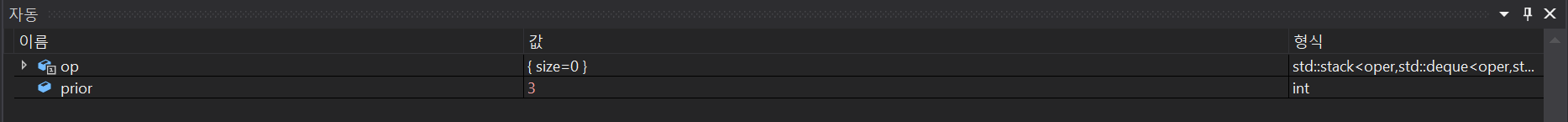
스택 사이즈가 증가한 것을 확인할 수 있다.



다음으로 tok이 받은 문자열은 연산자 “^”이다.

텍스트이(가) 표시된 사진

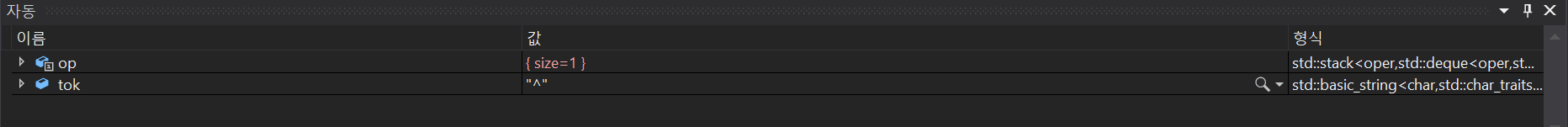
자동 생성된 설명



연산자 우선순위를 부여하는 부분으로 넘어가서 연산자 우선순위를 나타내는 변수 prior에 가장 높은 우선순위 3이 담긴 것을 확인할 수 있다.

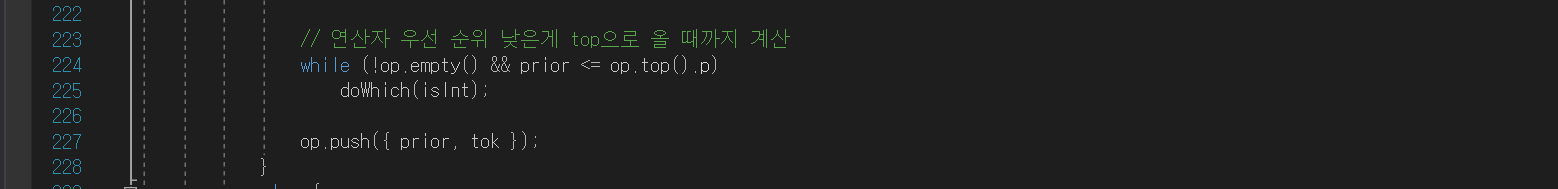
텍스트이(가) 표시된 사진

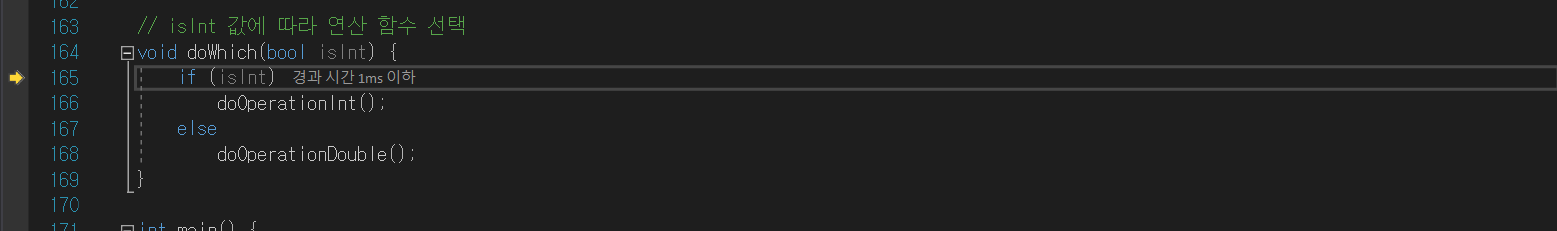
자동 생성된 설명



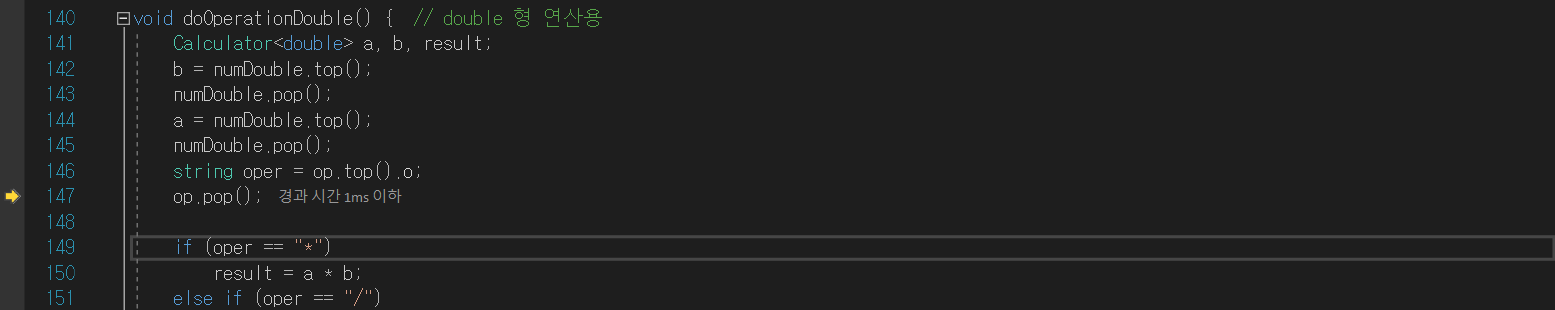
연산자 스택 op에 push해 스택의 사이즈가 증가한 것을 확인할 수 있다.

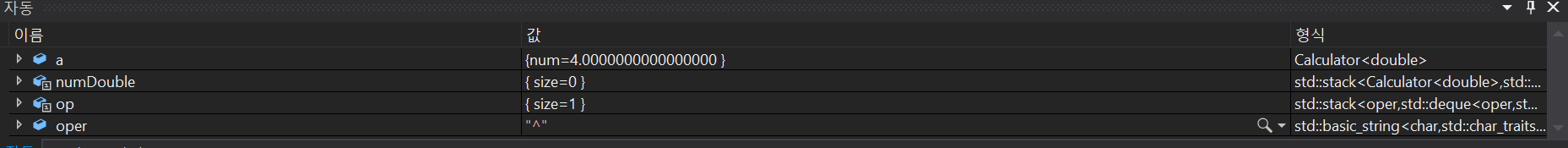
같은 과정이 반복되므로 tok이 “-“일 때로 넘어갔다.



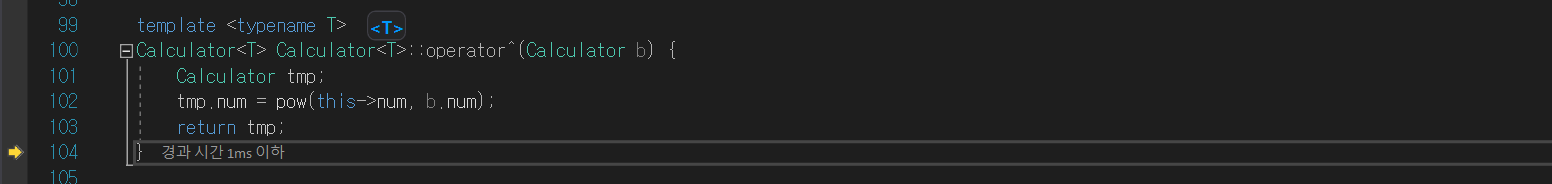


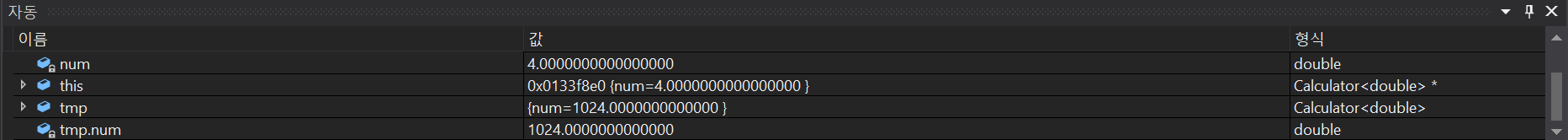
빼기 연산보다 제곱 연산이 우선 순위가 높으므로 연산을 수행하는 부분으로 넘어간다.



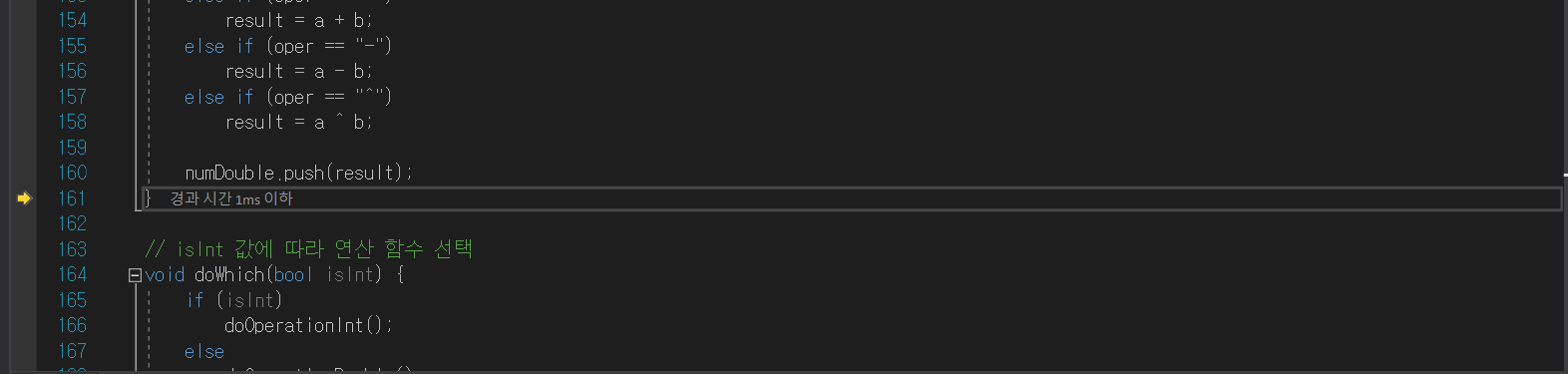


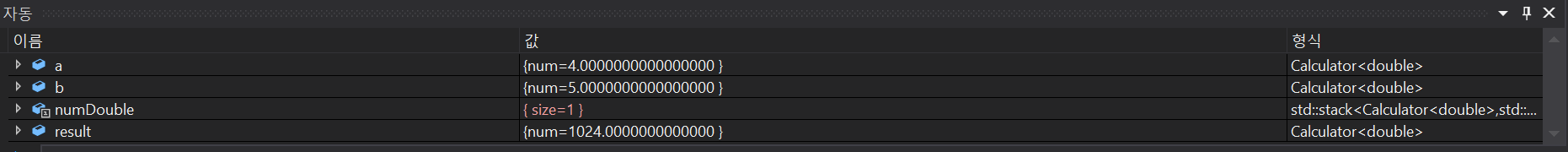
연산 함수 doOperationDouble에서 숫자 스택과 연산자 스택에서 top의 원소를 가져오고 pop하여 스택의 크기가 감소한 것을 확인할 수 있다.





제곱 연산을 연산자 중복 함수에서 수행해 tmp.num에 결과 값이 담긴 것을 확인할 수 있다.

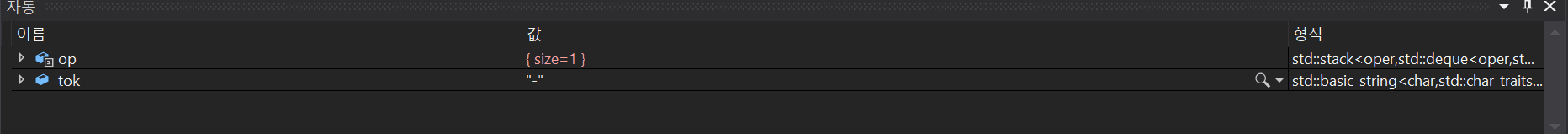




숫자 스택에 결과값을 다시 push해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

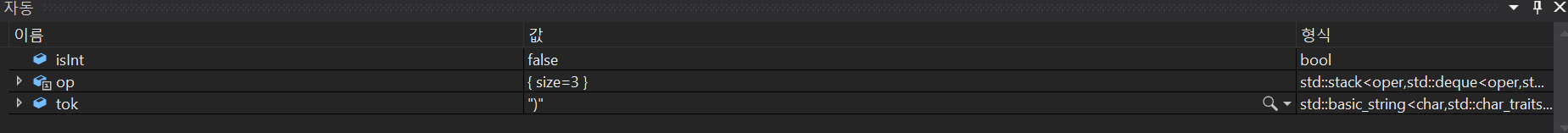


연산을 마치고 main함수로 돌아와 연산자 “-“을 연산자 스택에 push한다.

같은 과정이 반복되므로 tok이 “)”일 때로 넘어간다.

텍스트이(가) 표시된 사진

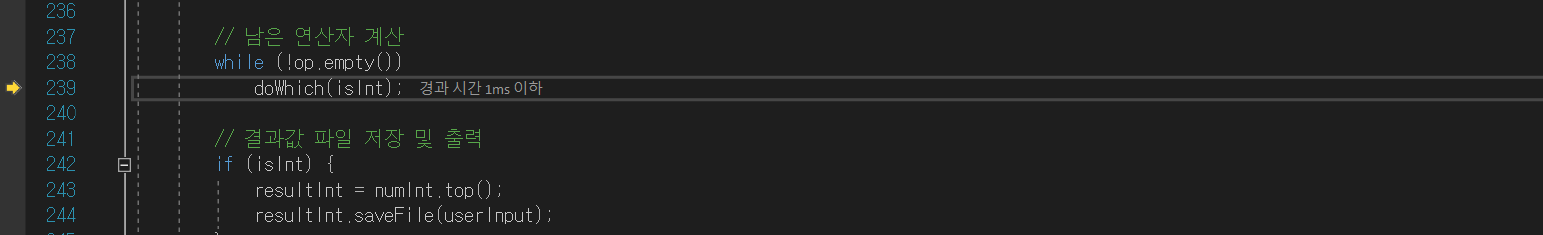
자동 생성된 설명

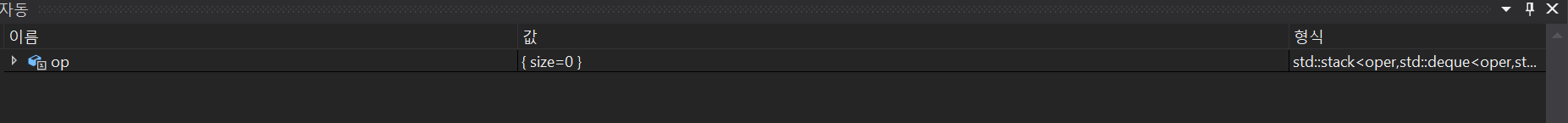


tok이 “)”이고 연산자 스택에 마지막으로 담긴 연산자가 “(“가 아니므로 괄호 안의 연산을 수행한다.

연산과정은 앞에 과정과 동일하기 때문해 생략한다.

4 ^ 5 – ( 5 + 7.8 ) / 2의 연산식에서 연산자의 우선순위대로 연산을 진행해 빼기와 나누기 연산을 제외하고 모든 연산을 수행한 상태이다. 마지막으로 1024 – 12.8 / 2의 연산을 수행해야 한다.

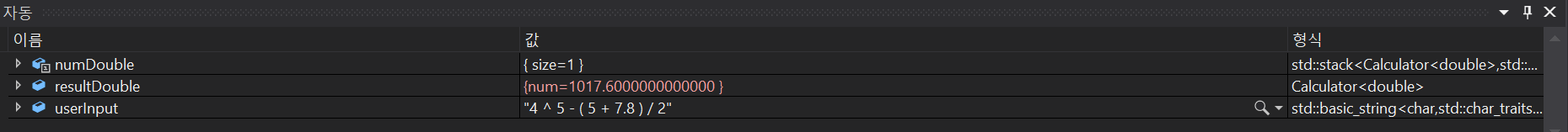
연산자 스택이 비어 있지 않으므로 남은 연산을 수행한다.



나머지 연산을 수행 후 연산자 스택이 비었으므로 다음 단계로 넘어간다.

텍스트이(가) 표시된 사진

자동 생성된 설명



숫자 스택에 남아 있는 숫자가 결과값이므로 결과값을 resultDouble 변수에 담아주고 saveFile함수로 파일에 연산식과 결과를 출력한다.

**느낀점**

김민석 : 팀장 역할을 맡게 되어 팀 프로젝트를 진행하면서 혼자 개발을 하는 것 보다 확실히 어려운 점이 있다고 느꼈다. 역할 분담을 적절히 하는 것, 팀원들과 진행 과정 중에 충분히 소통을 하는 것, 여러 의견들을 종합해서 한 가지 방향을 제시하는 것과 같이 추가적으로 생각해야 할 부분들이 많이 있었다. 하지만 배울 수 있는 점도 많이 있었다. 팀원들이 제시한 방법으로 혼자였다면 생각하지 못했을 방법으로 개발 과정 중 발생한 문제들을 해결할 수 있었고, 각자 짜온 코드를 보면서 같은 문제를 여러 방법으로 해결할 수 있다는 것을 깨달았다. 팀으로 프로젝트를 진행하면서 쉽지는 않았지만 많은 것을 배울 수 있는 좋은 기회였다.

김소빈 : 솔직히 맡은 부분은 그렇게 많지 않았지만 그럼에도 불구하고 내게는 어려운 부분이 많이 있어서, 부족한 부분이 많다는 것을 깨달았다. 그치만 차근차근 알려주고 설명해 주는 팀원들 덕분에 그래도 조금씩은 해 나갈 수 있어서 너무 고맙고 감사했다. 이 과제를 처음 받을 당시에는 정말 막막했지만 차근차근 풀어나가보면서 협동성이 왜 중요한지에 대해서도 깨닫게 되었고, 교수님께서 어떤 마음으로 이과제를 주셨는지 느낄 수 있었다.

정명균 : 팀 프로젝트를 해본적은 있지만 프로그래밍을 팀 프로젝트로 해보는 것은 처음이었고 오프라인에서 자주 만나 회의하며 할 수 없다는 점에서 불편한 점이 생길 수 밖에 없었다고 생각한다. 그러나 카카오톡이나 웹엑스 미팅 등으로 자주 연락을 주고 받음으로써 어느정도 보완이 된 것 같다.

또, 프로그래밍 말고도 정보검색 및 공유나 의견 제시 등도 팀 프로젝트에 큰 도움이 된다는 걸 상기시켜줬던 것 같다.

무엇보다 이번 팀 프로젝트는 다른 사람의 프로그래밍 과정이나 프로그래밍 할 때 생각하는 방향 등을 나와 비교하면서 볼 수 있던 것 같아 새로운 방향성을 나에게 제시해주는 좋은 경험이었다고 생각한다.