Infinite Arithmetic Precision Report

April 21, 2024

Introduction

We defined a namespace which contains two classes Integer and Float. At the progress of project we started with Integer which contains 2 objects

1. String container which stores the numbers as characters
2 . Sign which says whether the number is positive or negative.

Function members in Class Integer:

1) $+, -$ operator overloading:(which performs addition and subtraction)

In building up the algorithm of + operator we first concentrated ourselves on adding 2 positive integers. Since we will be manipulating the string we preferred copying rather than referencing. For adding 2 strings of unequal size. We added 0's so that value do not change implies addition do not change. Then stored the added char in a string of size 1 unit greater than maximum sized string. We iterated a loop from back so char stoting data > 10 are to be subtracted by 10 added one value to next place digit.

Next we built the subtraction (overloading operator) which involved the process same as addition that is subtracting individual digits storing on it. Another string of size greater than the maximum size of two integer strings is defined

Problem in Subtraction
We can only subtract the number that is of greater size as first integer so that we get positive integer as difference, so we interchange and insert underscore at last.

Problem in subtraction of 2 integers of same sign with first one greater than second one.

Here we get first character of resultant string -1 for this I made every character minus of previous then created an algorithm such that at '1' is forward to the array from 0 to end.

Using this subtraction,addition collectively.
1. We do addition of a positive and negative integer.
2. We do addition of a negative and positive integer.

For subtraction of two negative integers we erase negative sign and two positive ints and again add negative sign.

Similarly for subtraction
1. For positive,negative int we strip negative and add both integers.
2. For negative,positive int we strip negative and add both integers and insert negative sign.
3. For negative,negative int we strip negative and sub second from first and insert negative sign.

2) Multiplication overloading operator:

1. First for two positive ints product we use 2 for loops taking an integer class which is of zero value.We proceed from lats digit of 2nd number.The loop adds the first alst digit number of times then we go to 2nd digit from last of 2nd number, which adds on 1st number with pushed from back indirectly multiplying it 10 times, abd adds it 2nd last digit number of times this continues to give products No passing (removing of 0's needed) since addition which implicitly takes care of multiplication removes 0.

2. If negative and positive taken in the negative erased and multiplication 2 +ve ints get performed and finally

3. If negative,negative we erase 2 -ve and returned multiplied value directly.

3) Comparison operator ¿:
This operator though not give in project is needed for execution of division.
In this comparison I only overloaded to so that it gives correct output when 2 +ve ints taken which (0,1) o when false and 1 when true.

3) Division operator —:

1. Division of 2 +ve ints
a) If 1st int is less than 2nd int then I directly hard cooled it to zero.
b) If 1st greater than 2nd int then I used long division algorithm,first take a sub string of size of 2nd int and define it as string which abbreviates to complement string. I use 2 for loops for borrowing next left out digit other for possible "j" which subtracts such that remainder which is less than 2nd int remainder which

is less than 2nd int remainder and borrowed digit from right are joined and made to go string com this continues until last borrow.

2. Negative integers are handled same as multiplication.

Function members in Class Float:

String data members
1.string str
2.int sign
3.int point$_position$
4.$intpa(noofdigitsafterpoint)$
5.$pb(noofdigitsbeforepoint)$

0.1 Addition of two floats

1. Considering 2 positive ints
No of digits after point are made equal by adding 0's at the end.
No of digts before point are made equal by adding o's at the start.
Now points are removed from both the strings and integer instances are created for both strings and added. Now point is placed accordingly and we thus obtain the result needed
Limitations
We may notice addition 0's at the end of output
2. Other sub cases involving negative floats are dealt with

0.2 Subtraction of two floats

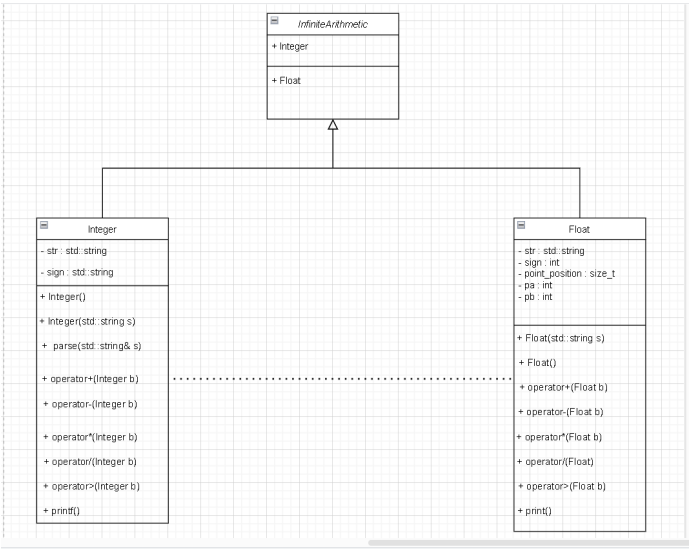
Same as addition but instances of integers are subtracted.

0.3 Multiplication of 2 floats

Points are removed from the float string, now the integer instances of the strings are multiplied and stored in integer mul
Now point is placed accordingly and we return float of the string
Other cases are dealt with

0.3.1 Division of 2 floats

For the both floating numbers points are removed for the first floating number 1000 0's are added at back and integer instances are divided and point is located



accordingly

1 Limitations

When the answer is below 1 then we get an out of range error

2 Learnings and some key techniques used

When the float string does not contain any point point is introduced at the back using npos