Bigint

Layout

The layout follows the same universal rules for indentation, such as indenting after loops and if statements etc. White space was added between wherever I felt it was necessary to organise the code. Operator implementations were organised with white space between things in this order:

- Declaring Bigint result
- Declaring variables for the next calculating loop
- First calculating loop
- Second calculating loop (if there is one)
- Variables relevant for the conversion to string loop
- Conversion to string loop
- Assignment of string to Bigint result and return result.

Names

I felt it to be more necessary to name anything with actual meaning than for more convenience. As a result, all the names give the variable or function meaning and clearly show what it is supposed to be doing. Any names with more than one word follow the convention that the first word is not capitalised at the first letter, but the other words are.

Structure

Variables are only declared when necessary, such as right before the loop in which they are used. Int is the only data type for numerical values because it is the only type that should be used in the context of the assignment. Everything should be in the correct order to ensure that control flow will work properly.

Documentation

The purpose of many functions and how they work within the program are documented in the code, by long comments written about them right before the code itself. There can be more to be said:

The algorithm for addition is made by me, but the other arithmetic operators I used web resources to work out how to complete them. For subtraction, I was able to make it by reading a method of working out how to do subtraction on paper from *dummies.com – How to Borrow When Subtracting* (*https://www.dummies.com/education/math/basic-math/how-to-borrow-when-subtracting/*). For multiplication, I was able to find pseudocode on 'long multiplication' from *Wikipedia.org – Multiplication algorithm* (*https://en.wikipedia.org/wiki/Multiplication_algorithm#Long_multiplication*). For division, I found C++ code on long division that can work in the context of this assignment. Then it was modified so that it is appropriate within that context, such as the removal of character to integer conversions because this conversion can occur later instead and avoiding the declaration of unnecessary variables by using the class variables. This code is found on *geeksforgeeks.org – Divide large number represented as string* (*https://www.geeksforgeeks.org/divide-large-number-represented-string/?fbclid=IwAR36zplMsKUglI9ZYBbqWbkzo9TvWIqnAk0y4INkLTWQ47QFifCrFYmLaC8*). The code I have in main.cpp was mostly taken from my hp-35 practical file and using the imported stack instead.

Another point that should be made is that I did the conversion from digits into a string within the arithmetic operators, rather than within the output operator (<<). This is because it felt easiest to do it that way earlier on and because it did not compromise on efficiency or convenience (I could just copy and paste it into the other arithmetic operators), but most importantly it worked still. Also, it would be easier to understand the debugging when looking for errors when a stack is being used, because the output operator is only used to display the final result and seeing the string of the correct number is quicker.

References

GeeksforGeeks. 2021. *Divide large number represented as string - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/divide-large-number-represented-string/?fbclid=IwAR36zplMsKUglI9ZYBbqWbkzo9TvWIqnAk0y4INkLTWQ47QFifCrFYmLaC8> [Accessed 21 October 2021].

dummies. 2021. *How to Borrow When Subtracting*. [online] Available at: <https://www.dummies.com/education/math/basic-math/how-to-borrow-when-subtracting/> [Accessed 14 October 2021].

En.wikipedia.org. 2021. *Multiplication algorithm - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Multiplication_algorithm#Long_multiplication> [Accessed 18 October 2021].