SDF REPORT

Mandapalli Rashmika - CS23BTECH11033
Thakur Ruchika Singh - CS23BTECH11061

April 21, 2024

1 Design

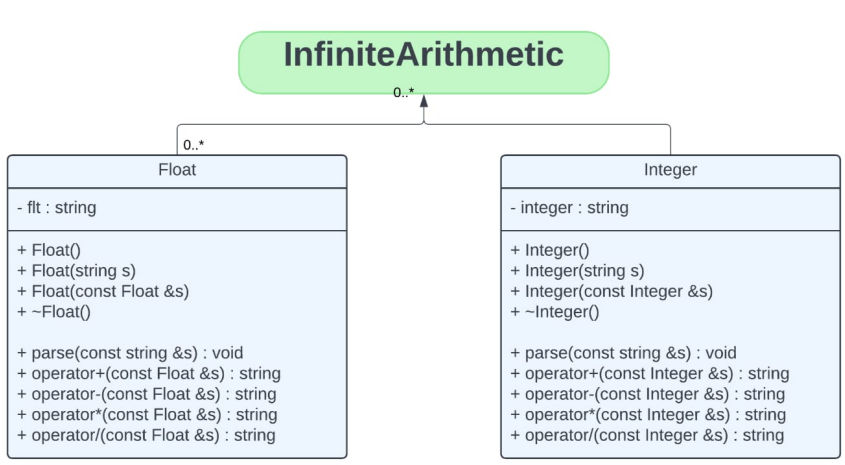

Figure 1: UML CLASS-DIAGRAM

Created a header file in which we separated integer overloading and float overloading by creating two separate classes.
Introduced special functions to balance the size of strings.

1.1 Integer class:

- ATTRIBUTES:
  int_num : stores the value of string which will be considered as integer.
- CONSTRUCTORS:
  integer():this initialises the integer value to 0
  integer(string s):the constructor takes the string s as input
  integer(const integer &copy): to copy the constructor
- DESTRUCTOR:
  ~ integer() : Destroys the instance
- METHODS:
  - parse(const integer s):
  - operator+(const integer s):overloaded + to return the sum as string.
  - operator-(const integer s):overloaded - to return the difference as string.
  - operator*(const integer s):overloaded * which used overloaded + to return the product as string.

1.2 Float class:

- ATTRIBUTES:
  float_num : stores the value of string which will be considered as float number.
- CONSTRUCTORS:
  Float(): this initialises the float value to 0.0
  Float(string s):the constructor takes the string s as input
  Float(const Float &copy): to copy the constructor
- DESTRUCTOR:
  ~ Float() : Destroys the instance
- METHODS:
  - parse(const Float s):
  - operator+(const Float s):overloaded + to return the sum as string.

- operator-(const Float s):overloaded - to return the difference as string.
- operator*(const Float s):overloaded * which used overloaded + to return the product as string.

1.3 Design

- For both integer and float infinite size calculations we considered them as strings.Because by strings we can easily access the length and characters.
- The same arithematic operators were overloaded to provide easiness for the calculation just as normal numbers.
- We created some functions like to balance the size if strings,to remove excess zeroes.These functions encapsulate common operations and improve code reusability.

2 README

The InfiniteArithemetic library created which provides the required functions to perform arithemetic operations on integers and float numbers without any size limit.

2.1 Infinite size integers

We created strings to store the digits and overloaded the operators " + , - and * " .We devised a logic to perform calculations on these strings as if they were integers by converting from the 'char' type to 'integer' and implemented a loop to iterate through the entire string.For + and - operations, we created a function to balance the size of the strings, making it easier to perform these operations via loops on strings. For the * operation, we utilized the overloaded + operator to simplify the code and logic.We created a function to remove excess zeroes in the result to make the answer accurate.

2.2 Infinite size float numbers

We created strings to store the digits and '.' and overloaded " + , - and * " .Similar to infinite size integers we devised a logic to perform calculations on these strings as if they were integers by converting from the 'char' type to 'integer' and implemented a loop to iterate through the entire string.For + and - operations, we created a function to balance the size of the strings on either side of '.', making it easier to perform these operations via loops on strings.Later we removed '.' through erase function and performed the overloaded + and - of integer class and then we inserted '.' from backside of the string.For the * operation, we utilized the overloaded + operator of Float class to simplify the code and logic.We created a function to remove excess zeroes before and after the result to ensure its accuracy.

3 Limitations

The library we created (InfiniteArithemetic) has accuracy but lacking in fast functioning. It is taking sometime to give the output.Also, it works only for float numbers and integers.

4 Verification approach

Various testcases were done to know whether the code is working for the conditions to make it accurate. We got to know the errors by printing every calculated value in multiplication overloading.For remaining we used break-points to correct the code.After solving we made changes by adding some functions to remove excess zeroes.

5 Key learning

This code gives clarity on usage and creating a library.The logic involves the perfect understanding of simple mathematics.We got to know that for the big codes like these we need to order the code perfectly to make it readable to the partner and comments are required to make the code readable.Perfect spacing is essential to understand the code easily.Not only accuracy but also the performance and readability are required for a code.

6 GIT commit screenshots

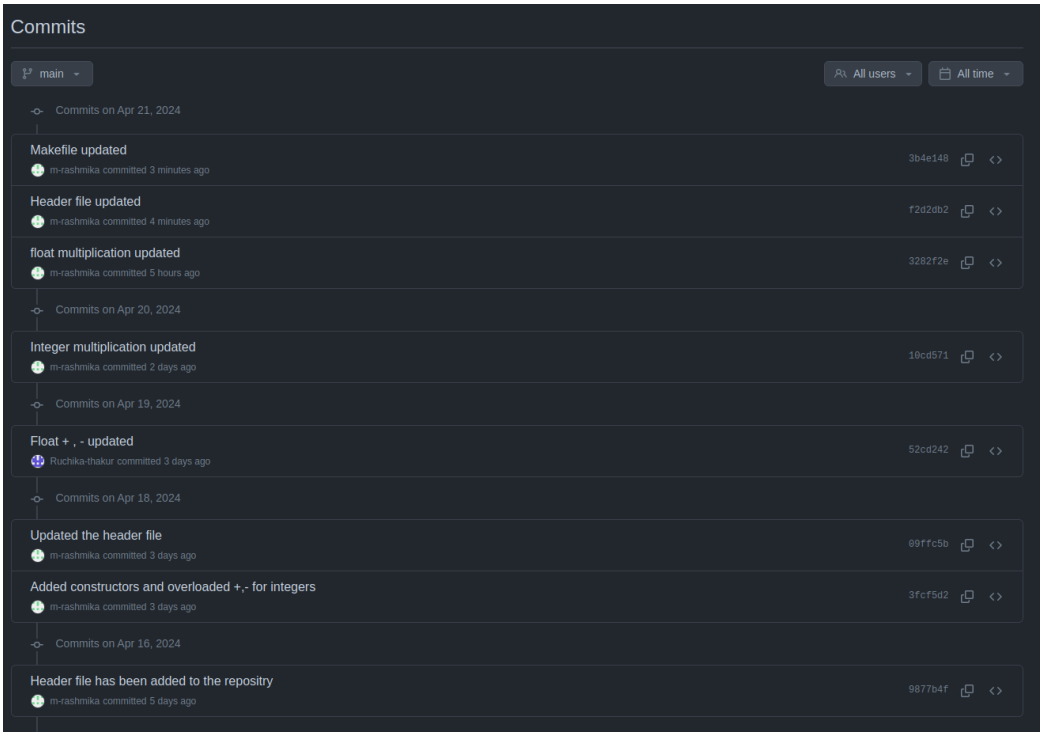

Figure 2: Git Commits