Cairo university

Faculty of computers and information

CS213
Object-oriented-programming
Assignment#2
2023

Ahmed Hossam

Mohamed Shebl

Mostafa Tarek

Dr Mohammed El-Ramly

m.elramly@fci-edu.eg

Work break down table

Mostafa Tarek	Ahmed Hossam	Mohamed Shebl

bool operator<(BigReal anotherReal)
bool operator>(BigReal anotherReal);
bool operator=(BigReal anotherReal);
friend ostream& operator<<(ostream& out, BigReal num);
	BigReal operator+ (BigReal& other)
	BigReal operator- (BigReal& other)
		isValidReal()
		BigReal (doublerealNumber=0.0)
		BigReal (string realNumber)
		BigReal(const BigReal& other)
		void setNum (string realNumber)
		int size()
		int sign()

Algorithms for BigReal class

isValidReal () algorithm:

1. Start by checking the sign character **k**:

    - If **k** is not equal to **'+'** and not equal to **'-'**, return **false** because it's an invalid sign.

2. Next, loop through each character in the **Number** string:

    - For each character at position **i**:

        - If the character is not a digit (0-9) and not a decimal point **'.'**, return **false** because it's an invalid character in the number.

3. Check for the presence of more than one decimal point in the string:

    - If the count of decimal points in the string is greater than 1, return **false** because it's not a valid number (more than one decimal point is not allowed).

4. Verify that the string does not end with a decimal point:

    - If the last character of the **Number** string is a decimal point, return **false** because it's not a valid number (a decimal point at the end is not allowed).

5. If none of the above conditions are met, you start checking the real part of the number:

    - Create an empty string **realPart** to store the real part of the number.

    - Loop through the characters in the **Number** string until you find the decimal point (if present) or the end of the string.

    -Check the following conditions:

- If the real part has more than one character and there is no decimal point in the number, and the first character of the real part is '0', return **false** because it's an invalid number (leading zero in a non-floating-point number is not allowed).

- If the real part has more than one character, there is one decimal point in the number, and the first character of the real part is '0', return **false** because it's an invalid number (leading zero in a floating-point number is not allowed).

6. If none of the above conditions are met, return **true** because the input is a valid BigReal number.

The Algorithm of “+” operator :

1) Copy the real and decimal parts of the current and other BigReal numbers into separate strings: `real1`, `real2`, `decimal1`, and `decimal2`.

2) Call a custom function `fun` to prepare the real and decimal parts, potentially modifying them and setting values for `n` and `y`.

3) Calculate the total number of digits required for addition, including one extra digit for a possible carry, and initialize a remainder variable `rem` to 0.

4) Declare and initialize three strings: `num1`, `num2`, and `add`.

5) Check if both numbers have the same sign:

a) If they have the same sign:

- Combine the real and decimal parts into two strings with a decimal point, `num1` and `num2`.
- Perform addition from the rightmost digit to the left, handling carry if necessary.
- Handle any remaining carry and format the result.
- Determine the sign of the result and return a new BigReal object with the calculated value.

b) If they have different signs:

- Determine which number has a greater magnitude.

- Set `num1` and `num2` based on the order of magnitude.
- Perform subtraction from the rightmost digit to the left, handling borrowing if necessary.
- Handle any leading zeros, remove the trailing decimal point, and format the result.
- Determine the sign of the result and return a new BigReal object with the calculated value.

The algorithm for “-“ operator :

1) Copy the real and decimal parts of the current and other BigReal numbers into separate strings: `real1`, `real2`, `decimal1`, `decimal2`, and calculate values for `n` and `y`.

2) Call a custom function `fun` to prepare the real and decimal parts, potentially modifying them and setting values for `n` and `y`.

3) Calculate the total number of digits required for subtraction, including one extra digit for a possible borrow, and initialize a string `subtract`.

4) Check the sign of both numbers:

a) If both numbers have the same positive sign:

- Determine the relative magnitude of the two numbers and set `num1` and `num2` accordingly.
- Perform subtraction from the rightmost digit to the left, handling borrowing if necessary.
- Reverse and format the result, considering leading zeros and trailing decimal points.
- Determine the sign of the result and return a new BigReal object with the calculated value.

b) If both numbers have the same negative sign:

- Determine the relative magnitude of the two numbers and set `num1` and `num2` accordingly.
- Perform subtraction from the rightmost digit to the left, handling borrowing if necessary.
- Reverse and format the result, considering leading zeros and trailing decimal points.
- Determine the sign of the result and return a new BigReal object with the calculated value.

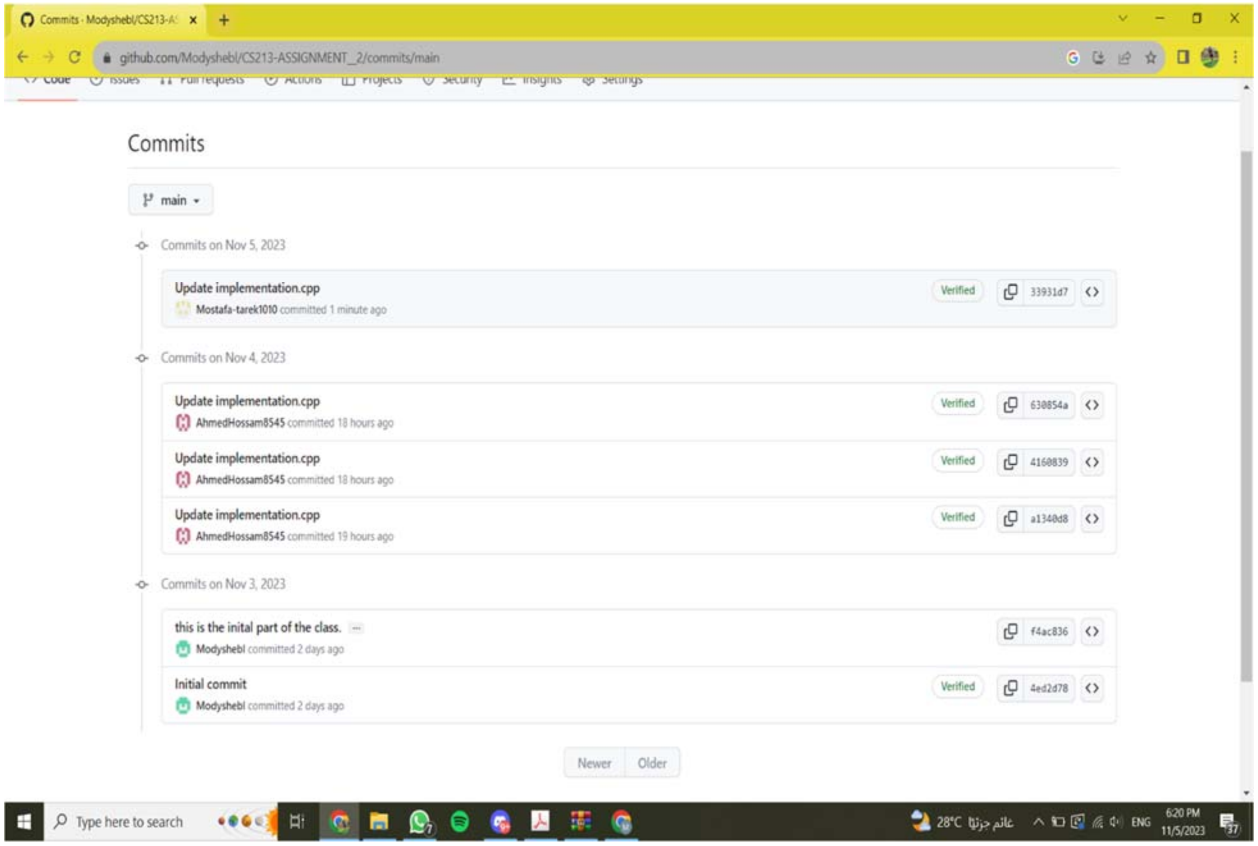
c) If the first number is positive and the second number is negative:

- Add the absolute values of the two numbers to handle subtraction.
- Perform addition from the rightmost digit to the left, handling carry if necessary.
- Reverse and format the result, considering leading zeros and trailing decimal points.
- Return a new BigReal object with a positive sign, as the first number is larger.

d) If the first number is negative and the second number is positive:

- Add the absolute values of the two numbers to handle subtraction.
- Perform addition from the rightmost digit to the left, handling carry if necessary.
- Reverse and format the result, considering leading zeros and trailing decimal points.
- Return a new BigReal object with a negative sign, as the second number is larger.

Github Screenshot


Commits · Modysheb/CS213-A
github.com/Modyshebl/CS213-ASSIGNMENT_2/commits/main
Code Issues Pull requests Actions Projects Security Insights Settings

Commits

main

Commits on Nov 5, 2023

Update implementation.cpp
Mostafa-tarek1010 committed 1 minute ago — Verified — 33931d7

Commits on Nov 4, 2023

Update implementation.cpp
AhmedHossam8545 committed 18 hours ago — Verified — 630854a

Update implementation.cpp
AhmedHossam8545 committed 18 hours ago — Verified — 4160839

Update implementation.cpp
AhmedHossam8545 committed 19 hours ago — Verified — a1340d8

Commits on Nov 3, 2023

this is the inital part of the class.
Modyshebl committed 2 days ago — f4ac836

Initial commit
Modyshebl committed 2 days ago — Verified — 4ed2d78