Big Rational Arithmetic and Robust Geometric Primitives

Spatial data is often large and default implementations of datatypes like Integer, Float and Double in C++ aren't equipped to deal with data in this format, since it frequently exceeds their bounds of computation. A new data type is thus required to support computations on spatial data which sometimes needs to be accurate to a precision level which might not be supported by already available datatypes.

'BigRational' will allow us to store and perform operations on the type of spatial data that gives trouble to the default implementations. Such an implementation requires a robust arithmetic operations to be defined, like the mathematical operators "+, - , /, * and %". In addition to these, I/O operations (input and output) for a BigRational number are also required.

Any data type in a language can be converted to a different datatype by type casting. One BigRational number should have the ability to be converted into a different type. Thus, the system shall support conversions from BigFloat to BigInt, BigDouble to BigInt and so on.

Datatypes and Operations:

The following classes, datatypes and operations will be present in the interface:

Datatypes and Classes

<u>BigInteger</u>: This datatype will be used to represent integers of dynamic length. The length of the integer is defined by the operation that is used to generate that "Big Number". As an example, if we calculate the factorial of 1000, the result is 2568 digits. The result of 1000! can be stored in a BigInteger variable of 2568 length.

<u>BigFloat</u>: This datatype will be used to represent floating point numbers that are within a specified range. The length and precision will be defined by the use of that number in an application.

<u>BigDouble</u>: This datatype will be used to represent larger floating point numbers that aren't supported by 'BigFloat' that are within a specified range. The length and precision will be defined by the use of that number in an application

Mathematical Operations

Addition '+': A binary operator that performs the addition operation on any of the given 'Big' input types.

Multiplication '*': A binary operator that performs the multiplication operation on any of the given 'Big' input types.

Subtraction '-': A binary operator that performs the subtraction operation on any of the given 'Big' input types.

Division '/': A binary operator that performs the division operation on any of the given 'Big' input types.

Mod '%': A binary operator that performs the modular operation on any of the given 'Big' input types.

All these operations need to work within different "Big…" types.

I/O Operations

Obtaining user input: We will have to define an operation that obtains the user input and translates it into a representation that can then be used to store it in memory.

Displaying numbers of that datatype: This would require an operation that translates the representation of the number in memory into a format that can be displayed.

Type Conversion Operations

Some situations may require the conversion of one datatype to another in the manner described earlier. This means that operations that can convert the datatypes from one representation to another will be necessary.

toBigInt: This operation converts the datatype from its current representation to the BigInt datatype.

toBigFloat: This operation converts the datatype from its current representation to the BigFloat datatype.

toBigDouble: This operation converts the datatype from its current representation to the BigDouble datatype.