Date: 22 February 2022

Design Principles of Programming Languages

Group – 9

Group Members

1. BT20CSE113: Anubhav Govind
2. BT20CSE139: Utkarsh Srivastava
3. BT20CSE144: Anukalp Pandey

Document for BigInt data type implementation in C++ and Member Functions for Addition and Subtraction.

BigInt – Data type

The BigInt data type is an implementation to represent the integers that are larger than maximum allowed size of any containers already available in C++, i.e., 64-bits.

Implementation

In this implementation the BigInt data type is implement using string which can grow as long as possible until the main memory is exhausted, which basically means there is no upper limit for the size of a BigInt number.

The implementation is done using a class named BigInt in which there is a private string which can be initialized using constructors in the public section or by taking an input string from the user or simply by assigning value by "=" operator.

Constructors

The Constructors are of 4 types in which 2 are parameterized constructors, a default constructor and a copy constructor.

The first parameterized constructor uses long long int data types to assign the value to the BigInt number so that we can typecast any data type to BigInt and then add it to some other BigInt number i.e., adding an int or long int to BigInt number.

The second parameterized constructor takes string as the argument to assign that value to the BigInt number

Copy constructor is used to copy the value of another BigInt data type. And default constructor assigns value "0" to the BigInt.

Member Functions

Further the public section contains the overloaded functions “+” for addition with another BigInt or long long int value, “-” for Subtraction from another BigInt or long long int, “=” for providing another way to assign the value to the BigInt number, “+=” for shortcut addition and “-=” for shortcut subtractions.

There are a few friend functions for “+”, “-”, “+=” and “-=” for the cases when the calling function is a long long int or any other similar data types.

Besides these there are two friend overloaded “<<” and “>>” functions for displaying the number and taking input from the user as and when required.

Other approaches

While Brainstorming for the best data type to use for implementing our BigInt data type we also found that the best method to implement BigInt is to store the number in form of base 2^64 manner like binary number is in base 2 and decimal is in base 10.

The biggest container possible in the C++ is long long int which can hold values of upto 64-bit memory. So one can store any value upto 2^64 –1 in that container so we can create an array (or better vector) of long long int which will contain the values till 2^63 – 1 in each container and the index will denote the power of (2^64).

This approach is more memory efficient than string method but it is more complicated than the string method at the same time. Also, when we save space we actually buy it with time and vice versa, so string implementation is faster with respect to the base 2^64 method, hence we chose string method over base 2^64.

Acknowledgement

We would like to thank our Professor, Dr. Milind Penurkar for giving this project which has increased our understanding of data types implementation as well as OOP concepts. We will keep making this types of project to enhance our knowledge of core concepts in the field of Computer Languages.