Large Integer class documentation

Contents

[Introduction 1](#_Toc327385523)

[Overview 1](#_Toc327385524)

[1. Initialization section: 1](#_Toc327385525)

[2. Function and methods: 1](#_Toc327385526)

[3. The usage examples 2](#_Toc327385527)

[4. Description of the *LargeInteger.exe command line* 3](#_Toc327385528)

# Introduction

CLargeInteger class is designed for implementation arithmetic operations between large integers such as subtraction, addition, multiplication, division. This class also allows logical operations between large integers.

# Overview

## Initialization section:

*CLargeInteger* class can perform creating instance with several types of initialization.

**CLargeInteger** value; *//* Default constructor, after creating value equal “0”   
**CLargeInteger** value = 1024; *// C*onstructor with integer initialization value

**CLargeInteger** value = “-1125899906842624”; *// C*onstructor with string initialization

**CLargeInteger** new\_value(value); *// Copy constructor*

## Function and methods:

*CLargeInteger* class has following pubic method:

|  |  |
| --- | --- |
| CLargeInteger() | Default constructor for CLargeInteger |
| CLargeInteger( const CLargeInteger &src ) | Copy constructor for CLargeInteger |
| CLargeInteger( const int integer\_value); | Constructor with integer initialization |
| CLargeInteger( const char \*\_char\_values ) | Constructor with string initialization |
| ~CLargeInteger() | Default destructor |
| static bool isIntegerValue(const char \*buff) | Returs is input char array can be integer value |
| const char \*Print() | Returning of complete value in char array |
| CLargeInteger operator = (const CLargeInteger &src) | Overloaded operator = |
| CLargeInteger operator + (const CLargeInteger &src) | Overloaded operator + |
| CLargeInteger operator - (const CLargeInteger &src) | Overloaded operator - |
| CLargeInteger operator \* (const CLargeInteger &src) | Overloaded operator \* |
| CLargeInteger operator / (const CLargeInteger &src) | Overloaded operator / |
| CLargeInteger operator += (const CLargeInteger &src) | Overloaded operator += |
| CLargeInteger operator -= (const CLargeInteger &src) | Overloaded operator -= |
| CLargeInteger operator \*= (const CLargeInteger &src) | Overloaded operator \*= |
| CLargeInteger operator /= (const CLargeInteger &src) | Overloaded operator /= |
| bool operator == (const CLargeInteger &src) | Overloaded operator == |
| bool operator > (const CLargeInteger &src) | Overloaded operator > |
| bool operator >= (const CLargeInteger &src) | Overloaded operator >= |
| bool operator < (const CLargeInteger &src) | Overloaded operator < |
| bool operator <= (const CLargeInteger &src) | Overloaded operator <= |

## The usage examples

You can use this class for the following mathematical operation with Large Integer class:

#include <iostream>

#include "LargeInteger.h"

int main( int args\_count, char \*\*args )

{

CLargeInteger val1 = "23847623498738874576";

CLargeInteger val2 = "-562423498738745872";

// Base operation

CLargeInteger i\_val1 = val1 + val2;

CLargeInteger i\_val2 = val1 - val2;

CLargeInteger i\_val3 = val1 \* val2;

CLargeInteger i\_val4 = val1 / val2;

// Logical operation

bool compare\_res1 = ( val1 == val2 );

bool compare\_res2 = ( val1 > val2 );

bool compare\_res3 = ( val1 >= val2 );

bool compare\_res4 = ( val1 < val2 );

bool compare\_res5 = ( val1 <= val2 );

// Console output

std::cout << std::endl;

std::cout << "---------------------------------------------------------" << std::endl;

std::cout << "| Large integer program demonstration |" << std::endl;

std::cout << "---------------------------------------------------------" << std::endl;

std::cout << std::endl;

std::cout << "value1 = " << val1.Print() << std::endl;

std::cout << "value2 = " << val2.Print() << std::endl;

std::cout << std::endl;

std::cout << "value1 + value2 = " << i\_val1.Print() << std::endl;

std::cout << "value1 - value2 = " << i\_val2.Print() << std::endl;

std::cout << "value1 \* value2 = " << i\_val3.Print() << std::endl;

std::cout << "value1 / value2 = " << i\_val4.Print() << std::endl;

std::cout << std::endl;

std::cout << "( value1 == value2 ) = " << ((compare\_res1) ? "true" : "false") << std::endl;

std::cout << "( value1 > value2 ) = " << ((compare\_res2) ? "true" : "false") << std::endl;

std::cout << "( value1 >= value2 ) = " << ((compare\_res3) ? "true" : "false") << std::endl;

std::cout << "( value1 < value2 ) = " << ((compare\_res4) ? "true" : "false") << std::endl;

std::cout << "( value1 <= value2 ) = " << ((compare\_res5) ? "true" : "false") << std::endl;

return 0;

}

The result of the example:

|  |
| --- |
| ---------------------------------------------------------  | Large integer program demonstration |  ---------------------------------------------------------  value1 = 23847623498738874576  value2 = -562423498738745872  value1 + value2 = 23285200000000128704  value1 - value2 = 24410046997477620448  value1 \* value2 = -13412463844765049844320727646945750272  value1 / value2 = -42  ( value1 == value2 ) = false  ( value1 > value2 ) = true  ( value1 >= value2 ) = true  ( value1 < value2 ) = false  ( value1 <= value2 ) = false |

## Description of the *LargeInteger.exe command line*

Program for work with large integers, works via command line.

The command line format:

|  |
| --- |
| **LargeInteger.exe** [/?] [value1 op value2]    **value1**  : Specifies an integer value 1  **op** : Type of operation such as +, -, \*, /  **value2** : Specifies an integer value 2 |

Example of using command line:

**LargeInteger.exe** 562423498738745872 + 23847623498738874576

The result of the program

