System.Numerics.Multiprecision Library Reference

January 4, 2016

Contents

Description

Multiprecision is a multiple precision arithmetic library. The library is implemented using GNU Multiple Precision Arithmetic Library (http://gmplib.org/).

Namespaces

Namespace	Description
System.Numerics.Multiprecision	Contains arbitrary precision signed integer type BigInt.

1 Usage

1.0.1 Referencing the System.Numerics.Multiprecision library

Right-click a project node in IDE | Project References... | Add System Extension Library Reference... | enable *System.Numerics.Multiprecision* check box

or add following line to your project's .cmp file:

```
reference <ext/System.Numerics.Multiprecision/System.Numerics.Multiprecision.cml>;
```

2 Installation

The System.Numerics.Multiprecision library uses the GNU MP library that is precompiled for the following platforms:

- 32-bit Windows (x86)
- 64-bit Windows (x64)
- 32-bit PC Linux (i686)
- 64-bit PC Linux (x86_64)

Current version of GNU MP library at the time of writing is 6.1.0.
If you want to install newer version of the GNU MP library or if you have different platform, you can compile the GNU MP library from sources by using the following instructions.

2.0.1 Compiling the GNU MP library from sources in Windows

- Install MinGW-w64 GCC to a path that does not contain spaces. I have mine installed in `C:\mingw-w64`.

  Installer can be obtained from
  http://sourceforge.net/projects/mingw-w64/files/Toolchains%20targetting%20Win32/Personal%20Builds/mingw-builds/installer/mingw-w64-install.exe/download

- Install MSYS2. Installation instructions can be found in
  http://sourceforge.net/p/msys2/wiki/MSYS2%20installation/

  I have mine installed in `C:\msys64`.

- Start MSYS2 shell.

- In the MSYS2 shell obtain **tar** by executing

  ```
  pacman -S tar
  ```

- Obtain **make** by executing

  ```
  pacman -S make
  ```

- Mount your MinGW-directory (mine is in `C:/mingw-w64`) by executing

  ```
  mkdir /mingw
  mount C:/mingw-w64 /mingw
  ```

- Insert your MinGW bin-directory in front of the PATH by executing

  ```
  export PATH=/mingw/mingw64/bin:$PATH
  ```

- Now if you ask which GCC is used by executing

  ```
  which gcc
  ```

  you should get **/mingw/mingw64/bin/gcc**.

- Download the GNU MP library from http://gmplib.org/#DOWNLOAD in bz2 format and place it to your MSYS2 home directory (mine is in `C:\msys64\home\Seppo`).

- Extract the GNU MP library by executing

  ```
  tar xjf gmp-6.1.0.tar.bz2
  ```

- Change to gmp-directory by executing

  ```
  cd gmp-6.1.0
  ```

- Configure the library for 64-bit Windows by executing

  ```
  ./configure --enable-static --disable-shared --host=x86_64-w64-mingw32
  ```

  For 32-bit Windows the command is

  ```
  ./configure --enable-static --disable-shared --host=i686-w64-mingw32
  ```

  Shared libraries are not used in Cmajor so we are disabling them.

- Make the library by executing

  ```
  make
  ```

- Install the library by executing

  ```
  make install
  ```

- Optionally test the library by executing

  ```
  make check
  ```

- Now the header file **gmp.h** should be in `C:\msys64\usr\local\include` directory and the library file **libgmp.a** should be in `C:\msys64\usr\local\lib` directory. Copy gmp.h and libgmp.a under the Cmajor System.Numerics.GmpIntf extension library to directory

  ```
  %APPDATA%\Cmajor\system\ext\System.Numerics.GmpIntf\gmp\windows\x64
  ```

  for 64-bit Windows, or to

  ```
  %APPDATA%\Cmajor\system\ext\System.Numerics.GmpIntf\gmp\windows\x86
  ```

  for 32-bit Windows.

- That's all.

2.0.2 Compiling the GNU MP library from sources in Linux

- Download the GNU MP library from http://gmplib.org/#DOWNLOAD in bz2 format and place it to your home directory.
- Extract the GNU MP library by executing

  ```
  tar xjf gmp-6.1.0.tar.bz2
  ```

- Change to gmp-directory by executing

  ```
  cd gmp-6.1.0
  ```

- Configure the library by executing

  ```
  ./configure --enable-static --disable-shared
  ```

  Shared libraries are not used in Cmajor so we are disabling them.
- Make the library by executing

  ```
  make
  ```

- Install the library by executing

  ```
  make install
  ```

- Optionally test the library by executing

  ```
  make check
  ```

- Now the header file **gmp.h** should be in `/usr/local/include` directory and the library file **libgmp.a** should be in `/usr/local/lib` directory. Copy gmp.h and libgmp.a under the Cmajor System.Numerics.GmpIntf extension library to

  ```
  <your cmajor directory>/system/ext/System.Numerics.GmpIntf/gmp/linux/x86_64
  ```

  directory for 64-bit Linux or to

  ```
  <your cmajor directory>/system/ext/System.Numerics.GmpIntf/gmp/linux/i686
  ```

  directory for 32-bit Linux.
- That's all.

3 System.Numerics.Multiprecision Namespace

Contains arbitrary precision signed integer type BigInt.

3.1 Classes

Class	Description
BigInt	An arbitrary precision signed integer type.

3.1.1 BigInt Class

An arbitrary precision signed integer type.

Syntax

```
public class BigInt;
```

3.1.1.1 Member Functions

Member Function	Description
BigInt()	Default constructor. Creates an instance of arbitrary precision signed integer and initializes it to zero.
BigInt(const System.Numerics.Multiprecision.BigInt&)	Copy constructor.
operator=(const System.Numerics.Multiprecision.BigInt&)	Copy assignment.
BigInt(System.Numerics.Multiprecision.BigInt&&)	Move constructor.
operator=(System.Numerics.Multiprecision.BigInt&&)	Move assignment.
BigInt(const System.String&)	Creates an instance of arbitrary precision signed integer and initializes it from provided decimal digit string. Digit string may be prefixed by a sign character. Throws **System.ConversionException** if a character other than sign or 0-9 is encountered in digit string.
BigInt(const System.String&, int)	Creates an instance of arbitrary precision signed integer and initializes it from provided digit string. Digit string may be prefixed by a sign character. Digits must be in given base. Throws **System.ConversionException** if a character other than sign or valid digit character is encountered.
BigInt(int)	Creates an instance of arbitrary precision signed integer and initializes it to given signed integer value.
BigInt(uint)	Creates an instance of arbitrary precision signed integer and initializes it to given unsigned integer value.
Handle() const	Returns a handle to the GNU MP library arbitrary precision integer representation.

ToString() const	Returns the value of the **BigInt** as a string of decimal digits prefixed by minus sign if the value is negative.
ToString(int) const	Returns the value of the **BigInt** as a string of digits in given base prefixed by minus sign if the value is negative.
operator=(int)	Assigns the value of the **BigInt** to given signed integer value.
operator=(uint)	Assigns the value of the **BigInt** to given unsigned integer value.
~BigInt()	Frees memory occupied by the **BigInt** instance.

BigInt() Member Function

Default constructor. Creates an instance of arbitrary precision signed integer and initializes it to zero.

Syntax

```
public BigInt();
```

BigInt(const System.Numerics.Multiprecision.BigInt&) Member Function

Copy constructor.

Syntax

```
public BigInt(const System.Numerics.Multiprecision.BigInt& that);
```

Parameters

Name	Type	Description
that	const System.Numerics.Multiprecision.BigInt&	A BigInt to copy from.

operator=(const System.Numerics.Multiprecision.BigInt&) Member Function

Copy assignment.

Syntax

```
public void operator=(const System.Numerics.Multiprecision.BigInt& that);
```

Parameters

Name	Type	Description
that	const System.Numerics.Multiprecision.BigInt&	A BigInt to assign.

BigInt(System.Numerics.Multiprecision.BigInt&&) Member Function

Move constructor.

Syntax

```
public BigInt(System.Numerics.Multiprecision.BigInt&& that);
```

Parameters

Name	Type	Description
that	System.Numerics.Multiprecision.BigInt&&	A BigInt to move from.

operator=(System.Numerics.Multiprecision.BigInt&&) Member Function

Move assignment.

Syntax

```
public void operator=(System.Numerics.Multiprecision.BigInt&& that);
```

Parameters

Name	Type	Description
that	System.Numerics.Multiprecision.BigInt&&	A BigInt to assign.

BigInt(const System.String&) Member Function

Creates an instance of arbitrary precision signed integer and initializes it from provided decimal digit string. Digit string may be prefixed by a sign character. Throws **System.ConversionException** if a character other than sign or 0-9 is encountered in digit string.

Syntax

```
public BigInt(const System.String& str);
```

Parameters

Name	Type	Description
str	const System.String&	A string of decimal digits.

BigInt(const System.String&, int) Member Function

Creates an instance of arbitrary precision signed integer and initializes it from provided digit string. Digit string may be prefixed by a sign character. Digits must be in given base. Throws **System.ConversionException** if a character other than sign or valid digit character is encountered.

Syntax

```
public BigInt(const System.String& str, int base_);
```

Parameters

Name	Type	Description
str	const System.String&	A string of digits of base base_.
base_	int	Base of digits. Base may vary from 2 to 62.

Remarks

For bases up to 36, case is ignored; upper-case and lower-case letters have the same value. For bases 37 to 62, upper-case letter represent the usual 10..35 while lower-case letter represent 36..61.

BigInt(int) Member Function

Creates an instance of arbitrary precision signed integer and initializes it to given signed integer value.

Syntax

```
public BigInt(int that);
```

Parameters

Name	Type	Description
that	int	A signed integer value.

BigInt(uint) Member Function

Creates an instance of arbitrary precision signed integer and initializes it to given unsigned integer value.

Syntax

```
public BigInt(uint that);
```

Parameters

Name	Type	Description
that	uint	An unsigned integer value.

Handle() const Member Function

Returns a handle to the GNU MP library arbitrary precision integer representation.

Syntax

```
public void* Handle() const;
```

Returns

void*

Returns a handle to the GNU MP library arbitrary precision integer representation.

ToString() const Member Function

Returns the value of the BigInt as a string of decimal digits prefixed by minus sign if the value is negative.

Syntax

```
public System.String ToString() const;
```

Returns

System.String

Returns the value of the BigInt as a string of decimal digits prefixed by minus sign if the value is negative.

ToString(int) const Member Function

Returns the value of the BigInt as a string of digits in given base prefixed by minus sign if the value is negative.

Syntax

```
public System.String ToString(int base_) const;
```

Parameters

Name	Type	Description
base_	int	Base of digits. Base may vary from 2 to 62.

Returns

System.String

Returns the value of the BigInt as a string of digits in given base prefixed by minus sign if the value is negative.

operator=(int) Member Function

Assigns the value of the BigInt to given signed integer value.

Syntax

```
public void operator=(int that);
```

Parameters

Name	Type	Description
that	int	A signed integer value.

operator=(uint) Member Function

Assigns the value of the BigInt to given unsigned integer value.

Syntax

```
public void operator=(uint that);
```

Parameters

Name	Type	Description
that	uint	An unsigned integer value.

~BigInt() Member Function

Frees memory occupied by the BigInt instance.

Syntax

```
public ~BigInt();
```

3.1.1.2 Nonmember Functions

Function	Description
Abs(const System.Numerics.Multiprecision.-BigInt&)	Returns absolute value of given **BigInt** .
ClearBit(System.Numerics.Multiprecision.-BigInt&, uint)	Clear given bit of given **BigInt** .
SetBit(System.Numerics.Multiprecision.BigInt&, uint)	Set given bit of given **BigInt** .
TestBit(System.Numerics.Multiprecision.-BigInt&, uint)	Returns true if given bit of given **BigInt** is set, false otherwise.
ToggleBit(System.Numerics.Multiprecision.-BigInt&, uint)	Toggle given bit of given **BigInt** .
operator%(const System.Numerics.-Multiprecision.BigInt&, const System.Numerics.-Multiprecision.BigInt&)	Returns the remainder when given **BigInt** is divided by another.
operator&(const System.Numerics.-Multiprecision.BigInt&, const System.Numerics.-Multiprecision.BigInt&)	Returns bitwise AND of two **BigInt** values.
operator*(const System.Numerics.-Multiprecision.BigInt&, const System.Numerics.-Multiprecision.BigInt&)	Returns the product of given **BigInt** value multiplied by another.
operator+(const System.Numerics.-Multiprecision.BigInt&, const System.Numerics.-Multiprecision.BigInt&)	Returns the sum of given **BigInt** value added to another.
operator-(const System.Numerics.-Multiprecision.BigInt&)	Returns the negation of **BigInt** .
operator-(const System.Numerics.-Multiprecision.BigInt&, const System.Numerics.-Multiprecision.BigInt&)	Returns the difference of given **BigInt** value subtracted from another.
operator/(const System.Numerics.-Multiprecision.BigInt&, const System.Numerics.-Multiprecision.BigInt&)	Returns the quotient when given **BigInt** is divided by another.
operator<(const System.Numerics.-Multiprecision.BigInt&, const System.Numerics.-Multiprecision.BigInt&)	Returns true if the first **BigInt** is less than the second **BigInt** , false otherwise.
operator<<(System.IO.OutputStream&, const System.Numerics.Multiprecision.BigInt&)	Puts the value of the given **BigInt** to the given output stream as string of decimal digits prefixed by minus sign if the value is negative.

operator==(const System.Numerics.-Multiprecision.BigInt&, const System.Numerics.-Multiprecision.BigInt&)	Returns true if the first **BigInt** is equal to the second **BigInt** , false otherwise.
operator^(const System.Numerics.-Multiprecision.BigInt&, const System.Numerics.-Multiprecision.BigInt&)	Returns bitwise XOR of two **BigInt** values.
operator—(const System.Numerics.-Multiprecision.BigInt&, const System.Numerics.-Multiprecision.BigInt&)	Returns bitwise inclusive OR of two **BigInt** values.
operator~(const System.Numerics.-Multiprecision.BigInt&)	Returns bitwise complement of **BigInt** value.

Abs(const System.Numerics.Multiprecision.BigInt&) Function

Returns absolute value of given BigInt.

Syntax

```
public System.Numerics.Multiprecision.BigInt Abs(const System.Numerics.Multiprecision.BigInt&
x);
```

Parameters

Name	Type	Description
x	const System.Numerics.Multiprecision.BigInt&	A BigInt.

Returns

System.Numerics.Multiprecision.BigInt

Returns absolute value of given BigInt.

ClearBit(System.Numerics.Multiprecision.BigInt&, uint) Function

Clear given bit of given BigInt.

Syntax

```
public void ClearBit(System.Numerics.Multiprecision.BigInt& x, uint bitIndex);
```

Parameters

Name	Type	Description
x	System.Numerics.Multiprecision.BigInt&	A reference to a BigInt.
bitIndex	uint	Index of bit to clear.

SetBit(System.Numerics.Multiprecision.BigInt&, uint) Function

Set given bit of given BigInt.

Syntax

```
public void SetBit(System.Numerics.Multiprecision.BigInt& x, uint bitIndex);
```

Parameters

Name	Type	Description
x	System.Numerics.Multiprecision.BigInt&	A reference to a BigInt.
bitIndex	uint	Index of bit to set.

TestBit(System.Numerics.Multiprecision.BigInt&, uint) Function

Returns true if given bit of given BigInt is set, false otherwise.

Syntax

```
public bool TestBit(System.Numerics.Multiprecision.BigInt& x, uint bitIndex);
```

Parameters

Name	Type	Description
x	System.Numerics.Multiprecision.BigInt&	A reference to a BigInt.
bitIndex	uint	Index of bit to test.

Returns

bool

Returns true if given bit of given BigInt is set, false otherwise.

ToggleBit(System.Numerics.Multiprecision.BigInt&, uint) Function

Toggle given bit of given BigInt.

Syntax

```
public void ToggleBit(System.Numerics.Multiprecision.BigInt& x, uint bitIndex);
```

Parameters

Name	Type	Description
x	System.Numerics.Multiprecision.BigInt&	A reference to a BigInt.
bitIndex	uint	Index of bit to toggle.

operator%(const System.Numerics.Multiprecision.BigInt&, const System.Numerics.Multiprecision.Bi
Function

Returns the remainder when given BigInt is divided by another.

Syntax

```
public System.Numerics.Multiprecision.BigInt operator%(const System.Numerics.Multiprecision.BigIn
left, const System.Numerics.Multiprecision.BigInt& right);
```

Parameters

Name	Type	Description
left	const System.Numerics.Multiprecision.BigInt&	Divisor.
right	const System.Numerics.Multiprecision.BigInt&	Dividend.

Returns

System.Numerics.Multiprecision.BigInt

Returns the remainder when left is divided by right.

operator&(const System.Numerics.Multiprecision.BigInt&, const System.Numerics.Multiprecision.Big
Function

Returns bitwise AND of two BigInt values.

Syntax

```
public System.Numerics.Multiprecision.BigInt operator&(const System.Numerics.Multiprecision.BigIn
left, const System.Numerics.Multiprecision.BigInt& right);
```

Parameters

Name	Type	Description
left	const System.Numerics.Multiprecision.BigInt&	Left operand.
right	const System.Numerics.Multiprecision.BigInt&	Right operand.

Returns

System.Numerics.Multiprecision.BigInt

Returns bitwise AND of given BigInt values.

operator*(const System.Numerics.Multiprecision.BigInt&, const System.Numerics.Multiprecision.Big
Function

Returns the product of given BigInt value multiplied by another.

Syntax

```
public System.Numerics.Multiprecision.BigInt operator*(const System.Numerics.Multiprecision.BigIn
left, const System.Numerics.Multiprecision.BigInt& right);
```

Parameters

Name	Type	Description
left	const System.Numerics.Multiprecision.BigInt&	Left operand.
right	const System.Numerics.Multiprecision.BigInt&	Right operand.

Returns

System.Numerics.Multiprecision.BigInt

Returns the product of given BigInt value multiplied by another.

operator+(const System.Numerics.Multiprecision.BigInt&, const System.Numerics.Multiprecision.Big
Function

Returns the sum of given BigInt value added to another.

Syntax

```
public System.Numerics.Multiprecision.BigInt operator+(const System.Numerics.Multiprecision.BigIn
left, const System.Numerics.Multiprecision.BigInt& right);
```

Parameters

Name	Type	Description
left	const System.Numerics.Multiprecision.BigInt&	Left operand.
right	const System.Numerics.Multiprecision.BigInt&	Right operand.

Returns

System.Numerics.Multiprecision.BigInt

Returns the sum of given BigInt value added to another.

operator-(const System.Numerics.Multiprecision.BigInt&) Function

Returns the negation of BigInt.

Syntax

```
public System.Numerics.Multiprecision.BigInt operator-(const System.Numerics.Multiprecision.BigIn
x);
```

Parameters

Name	Type	Description
x	const System.Numerics.Multiprecision.BigInt&	A value.

Returns

System.Numerics.Multiprecision.BigInt

Returns the negation of BigInt.

operator-(const System.Numerics.Multiprecision.BigInt&, const System.Numerics.Multiprecision.Bigl Function

Returns the difference of given BigInt value subtracted from another.

Syntax

```
public System.Numerics.Multiprecision.BigInt operator-(const System.Numerics.Multiprecision.BigIn
left, const System.Numerics.Multiprecision.BigInt& right);
```

Parameters

Name	Type	Description
left	const System.Numerics.Multiprecision.BigInt&	Left operand.
right	const System.Numerics.Multiprecision.BigInt&	Right operand.

Returns

System.Numerics.Multiprecision.BigInt

Returns the difference of given BigInt value subtracted from another.

operator/(const System.Numerics.Multiprecision.BigInt&, const System.Numerics.Multiprecision.Big
Function

Returns the quotient when given BigInt is divided by another.

Syntax

```
public System.Numerics.Multiprecision.BigInt operator/(const System.Numerics.Multiprecision.BigIn
left, const System.Numerics.Multiprecision.BigInt& right);
```

Parameters

Name	Type	Description
left	const System.Numerics.Multiprecision.BigInt&	Divisor.
right	const System.Numerics.Multiprecision.BigInt&	Dividend.

Returns

System.Numerics.Multiprecision.BigInt

Returns the quotient when given BigInt is divided by another.

Remarks

Quotient is rounded towards zero.

operator<(const System.Numerics.Multiprecision.BigInt&, const System.Numerics.Multiprecision.Big
Function

Returns true if the first BigInt is less than the second BigInt, false otherwise.

Syntax

```
public bool operator<(const System.Numerics.Multiprecision.BigInt& left, const System.Numerics.M
right);
```

Parameters

Name	Type	Description
left	const System.Numerics.Multiprecision.BigInt&	Left operand.
right	const System.Numerics.Multiprecision.BigInt&	Right operand.

Returns

bool

Returns true if the first BigInt is less than the second BigInt, false otherwise.

operator<<(System.IO.OutputStream&, const System.Numerics.Multiprecision.BigInt&) Function

Puts the value of the given BigInt to the given output stream as string of decimal digits prefixed by minus sign if the value is negative.

Syntax

```
public System.IO.OutputStream& operator<<(System.IO.OutputStream& s, const System.Numerics.Multi
x);
```

Parameters

Name	Type	Description
s	System.IO.OutputStream&	An output stream.
x	const System.Numerics.Multiprecision.BigInt&	A BigInt value.

Returns

System.IO.OutputStream&

Returns a reference to the output stream.

operator==(const System.Numerics.Multiprecision.BigInt&, const System.Numerics.Multiprecision.B Function

Returns true if the first BigInt is equal to the second BigInt, false otherwise.

Syntax

```
public bool operator==(const System.Numerics.Multiprecision.BigInt& left, const System.Numerics.M
right);
```

Parameters

Name	Type	Description
left	const System.Numerics.Multiprecision.BigInt&	Left operand.
right	const System.Numerics.Multiprecision.BigInt&	Right operand.

Returns

bool

Returns true if the first BigInt is equal to the second BigInt, false otherwise.

operator^(const System.Numerics.Multiprecision.BigInt&, const System.Numerics.Multiprecision.Big
Function

Returns bitwise XOR of two BigInt values.

Syntax

```
public System.Numerics.Multiprecision.BigInt operator^(const System.Numerics.Multiprecision.BigIn
left, const System.Numerics.Multiprecision.BigInt& right);
```

Parameters

Name	Type	Description
left	const System.Numerics.Multiprecision.BigInt&	Left operand.
right	const System.Numerics.Multiprecision.BigInt&	Right operand.

Returns

System.Numerics.Multiprecision.BigInt

Returns bitwise XOR of two BigInt values.

operator|(const System.Numerics.Multiprecision.BigInt&, const System.Numerics.Multiprecision.BigI Function

Returns bitwise inclusive OR of two BigInt values.

Syntax

```
public System.Numerics.Multiprecision.BigInt operator|(const System.Numerics.Multiprecision.BigIn
left, const System.Numerics.Multiprecision.BigInt& right);
```

Parameters

Name	Type	Description
left	const System.Numerics.Multiprecision.BigInt&	Left operand.
right	const System.Numerics.Multiprecision.BigInt&	Right operand.

Returns

System.Numerics.Multiprecision.BigInt

Returns bitwise inclusive OR of two BigInt values.

operator~(const System.Numerics.Multiprecision.BigInt&) Function

Returns bitwise complement of BigInt value.

Syntax

```
public System.Numerics.Multiprecision.BigInt operator~(const System.Numerics.Multiprecision.BigI
x);
```

Parameters

Name	Type	Description
x	const System.Numerics.Multiprecision.BigInt&	Operand.

Returns

System.Numerics.Multiprecision.BigInt

Returns bitwise complement of BigInt value.