

# LILIB(Long Interval LIBrary)

松田 望

平成 24 年 9 月 4 日

## 1 概要

- C++ のライブラリ。
- LINUX 上の GCC でコンパイル可能。
- 多倍長実数クラス LongFloat を提供。
- 多倍長行列クラス LongMatrix を提供。
- 精度保証付き多倍長区間クラス LongInterval を提供。
- 精度保証付き多倍長区間行列クラス LongIntervalMatrix を提供。
- 区間値を整数の中心値・半径形式で保持。
- 多倍長演算の精度を任意に設定可能。
- 多倍長クラスはプログラム上、通常の「数」と同じように扱える。
- LongFloat および LongInterval の中心値に対する、四則演算および平方根の計算が、correct rounding である。

## 2 使用方法

1. LILIB を使用するユーザプログラム `src.cpp` をコンパイルするには、
  - `src.cpp`
  - `lilib.h`
  - `libli.a`の三つのファイルが必要です。
2. ライブラリファイル `libli.a` は、`lilib.zip` を展開したディレクトリで、`make` コマンドを実行すると生成されます。

3. `src.cpp` の冒頭に、

```
#include "lilib.h"
```

と書いてください。

4. クラス `LongFloat`, `LongMatrix`, `LongInterval`, `LongIntervalMatrix` を使用する前 (変数の宣言より前) に、関数

```
void lilib::setPrecision(int precision)
```

を一度だけ呼んでください。precision は、使用したい多倍長演算の精度 (10 進数の桁数) です。

5. 関数 `setPrecision` によって、指定された桁数以上の精度の多倍長演算が可能になります。実際に扱える桁数は、関数

```
int lilib::getPrecision()
```

によって取得できます。

## 3 サンプルプログラム

### 3.1 ソースコード

```
#include <iostream>
#include "lilib.h"

using namespace std;

int main(){
    lilib::setPrecision(100);
    cout << "Long precision is " << lilib::getPrecision() << "." << endl;
    cout << endl;

    LongInterval x;

    x = 1;
    cout << "x = " << x << endl;

    x /= 3;
    cout << "x = " << x << endl;
```

```

x *= 3;
cout << "x = " << x << endl;
cout << endl;

int m = 3, n = 2, i, j;
LongIntervalMatrix a(m, n), b, c;

for(i = 0; i < m; i++){
    for(j = 0; j < n; j++){
        a[i][j] = n * i + j;
    }
}

b = trans(a);
c = a * b;

cout << "a = " << endl << a << endl;
cout << "b = " << endl << b << endl;
cout << "c = " << endl << c << endl;

return 0;
}

```

### 3.2 実行結果

Long precision is 105.

```

x = < 1.000000, 0.000000>
x = < 3.333333e-1, 2.537942e-116>
x = < 1.000000, 7.613826e-116>

a =
< 0.000000, 0.000000>   < 1.000000, 0.000000>
< 2.000000, 0.000000>   < 3.000000, 0.000000>
< 4.000000, 0.000000>   < 5.000000, 0.000000>

b =

```

```

< 0.000000, 0.000000>   < 2.000000, 0.000000>   < 4.000000, 0.000000>
< 1.000000, 0.000000>   < 3.000000, 0.000000>   < 5.000000, 0.000000>

c =
< 1.000000, 0.000000>   < 3.000000, 0.000000>   < 5.000000, 0.000000>
< 3.000000, 0.000000>   < 1.300000e1, 0.000000> < 2.300000e1, 0.000000>
< 5.000000, 0.000000>   < 2.300000e1, 0.000000> < 4.100000e1, 0.000000>

```

## 4 LongFloat クラス

### 4.1 コンストラクタ

LongFloat()	値は不定。
LongFloat(int x) LongFloat(double x) LongFloat(LongFloat x)	x で初期化する。

### 4.2 使用可能な演算子

- +LongFloat
- -LongFloat
- LongFloat + int
- LongFloat - int
- LongFloat \* int
- LongFloat / int
- int + LongFloat
- int - LongFloat
- int \* LongFloat
- int / LongFloat
- LongFloat + LongFloat
- LongFloat - LongFloat
- LongFloat \* LongFloat
- LongFloat / LongFloat

- `LongFloat = int`
- `LongFloat += int`
- `LongFloat -= int`
- `LongFloat *= int`
- `LongFloat /= int`
- `LongFloat = LongFloat`
- `LongFloat += LongFloat`
- `LongFloat -= LongFloat`
- `LongFloat *= LongFloat`
- `LongFloat /= LongFloat`
- `LongFloat == int`
- `LongFloat != int`
- `LongFloat < int`
- `LongFloat > int`
- `LongFloat <= int`
- `LongFloat >= int`
- `int == LongFloat`
- `int != LongFloat`
- `int < LongFloat`
- `int > LongFloat`
- `int <= LongFloat`
- `int >= LongFloat`
- `LongFloat == LongFloat`
- `LongFloat != LongFloat`
- `LongFloat < LongFloat`
- `LongFloat > LongFloat`

- `LongFloat <= LongFloat`
- `LongFloat >= LongFloat`
- `std::ostream << LongFloat`

### 4.3 メンバ関数

<code>void setDouble(double x)</code>	値を $x$ にする。
<code>double getDouble()</code>	<code>double</code> 値を取得する。
<code>double getDouble(int round)</code>	丸めの方向を指定して <code>double</code> 値を取得する。 <code>round &lt; 0</code> なら、下への丸め。 <code>round == 0</code> なら、最近点への丸め。 <code>round &gt; 0</code> なら、上への丸め。
<code>std::string getString()</code>	値を表す文字列を取得する。
<code>std::string getInternalData()</code>	内部データを表す文字列を取得する。

### 4.4 非メンバ関数

<code>LongFloat abs(LongFloat x)</code>	$ x $
<code>LongFloat pow(LongFloat x, int n)</code>	$x^n$
<code>LongFloat sqrt(LongFloat x)</code>	$\sqrt{x}$

## 5 LongMatrix クラス

### 5.1 コンストラクタ

<code>LongMatrix()</code>	1 行 1 列の行列。値は不定。
<code>LongMatrix(int rows, int columns)</code>	rows 行 columns 列の行列。値は不定。
<code>LongMatrix(LongMatrix x)</code>	$x$ で初期化する。

### 5.2 使用可能な演算子

- `+LongMatrix`
- `-LongMatrix`
- `LongMatrix + int`
- `LongMatrix - int`
- `LongMatrix * int`

- LongMatrix / int
- int + LongMatrix
- int - LongMatrix
- int \* LongMatrix
- int / LongMatrix
- LongMatrix + LongFloat
- LongMatrix - LongFloat
- LongMatrix \* LongFloat
- LongMatrix / LongFloat
- LongFloat + LongMatrix
- LongFloat - LongMatrix
- LongFloat \* LongMatrix
- LongFloat / LongMatrix
- LongMatrix + LongMatrix
- LongMatrix - LongMatrix
- LongMatrix \* LongMatrix
- LongMatrix += int
- LongMatrix -= int
- LongMatrix \*= int
- LongMatrix /= int
- LongMatrix += LongFloat
- LongMatrix -= LongFloat
- LongMatrix \*= LongFloat
- LongMatrix /= LongFloat
- LongMatrix = LongMatrix
- LongMatrix += LongMatrix
- LongMatrix -= LongMatrix
- std::ostream << LongMatrix

### 5.3 メンバ関数

<code>void resize(int rows, int columns)</code>	サイズを rows 行 columns 列 にする。 値は不定になる。
<code>int rows()</code>	行数を取得する。
<code>int columns()</code>	列数を取得する。
<code>std::string getString()</code>	値を表す文字列を取得する。

### 5.4 非メンバ関数

<code>LongMatrix abs(LongMatrix a)</code>	$y_{ij} =  a_{ij} $ となる行列 $Y$ を取得する。
<code>LongMatrix sqrt(LongMatrix a)</code>	$y_{ij} = \sqrt{a_{ij}}$ となる行列 $Y$ を取得する。
<code>LongMatrix trans(LongMatrix a)</code>	転置行列 $A^T$ を取得する。
<code>LongMatrix zeros(int rows, int columns)</code>	rows 行 columns 列 の 全要素が 0 の行列を取得する。
<code>LongMatrix ones(int rows, int columns)</code>	rows 行 columns 列 の 全要素が 1 の行列を取得する。
<code>LongMatrix eye(int size)</code>	size 行 size 列 の単位行列を取得する。
<code>void qr(LongMatrix &amp;q, LongMatrix &amp;r, const LongMatrix &amp;a)</code>	QR 分解を行う。

## 6 LongInterval クラス

### 6.1 コンストラクタ

<code>LongInterval()</code>	値は不定。
<code>LongInterval(int x)</code> <code>LongInterval(double x)</code> <code>LongInterval(LongFloat x)</code> <code>LongInterval(LongInterval x)</code>	$x$ で初期化する。
<code>LongInterval(int mid, int rad)</code> <code>LongInterval(LongFloat mid, int rad)</code> <code>LongInterval(int mid, LongFloat rad)</code> <code>LongInterval(LongFloat mid, LongFloat rad)</code>	中心値を mid、 半径を rad で初期化する。

### 6.2 使用可能な演算子

- `+LongInterval`
- `-LongInterval`



- $\text{LongInterval} + \text{int}$
- $\text{LongInterval} - \text{int}$
- $\text{LongInterval} * \text{int}$
- $\text{LongInterval} / \text{int}$
- $\text{int} + \text{LongInterval}$
- $\text{int} - \text{LongInterval}$
- $\text{int} * \text{LongInterval}$
- $\text{int} / \text{LongInterval}$
- $\text{LongInterval} + \text{LongFloat}$
- $\text{LongInterval} - \text{LongFloat}$
- $\text{LongInterval} * \text{LongFloat}$
- $\text{LongInterval} / \text{LongFloat}$
- $\text{LongFloat} + \text{LongInterval}$
- $\text{LongFloat} - \text{LongInterval}$
- $\text{LongFloat} * \text{LongInterval}$
- $\text{LongFloat} / \text{LongInterval}$
- $\text{LongInterval} + \text{LongInterval}$
- $\text{LongInterval} - \text{LongInterval}$
- $\text{LongInterval} * \text{LongInterval}$
- $\text{LongInterval} / \text{LongInterval}$
- $\text{LongInterval} + \text{LongMatrix}$
- $\text{LongInterval} - \text{LongMatrix}$
- $\text{LongInterval} * \text{LongMatrix}$
- $\text{LongInterval} / \text{LongMatrix}$
- $\text{LongMatrix} + \text{LongInterval}$
- $\text{LongMatrix} - \text{LongInterval}$

- `LongMatrix * LongInterval`
- `LongMatrix / LongInterval`
- `LongInterval = int`
- `LongInterval += int`
- `LongInterval -= int`
- `LongInterval *= int`
- `LongInterval /= int`
- `LongInterval = LongFloat`
- `LongInterval += LongFloat`
- `LongInterval -= LongFloat`
- `LongInterval *= LongFloat`
- `LongInterval /= LongFloat`
- `LongInterval = LongInterval`
- `LongInterval += LongInterval`
- `LongInterval -= LongInterval`
- `LongInterval *= LongInterval`
- `LongInterval /= LongInterval`
- `std::ostream << LongInterval`

### 6.3 メンバ関数

<code>void setDouble(double x)</code>	中心値を $x$ 、半径を 0 にする。
<code>void setMidRad(int mid, int rad)</code> <code>void setMidRad(LongFloat mid, int rad)</code> <code>void setMidRad(int mid, LongFloat rad)</code> <code>void setMidRad(LongFloat mid, LongFloat rad)</code>	中心値を $mid$ 、半径を $rad$ にする。
<code>double getDouble()</code>	<code>double</code> 値を取得する。
<code>std::string getMidRad()</code>	中心値と半径を表す文字列を取得する。
<code>std::string getInfSup()</code>	下端と上端を表す文字列を取得する。
<code>std::string getInternalData()</code>	内部データを表す文字列を取得する。
<code>LongFloat mid()</code>	中心値を取得する。
<code>LongFloat rad()</code>	半径を取得する。
<code>LongFloat diam()</code>	直径を取得する。
<code>LongFloat inf()</code>	下端を取得する。
<code>LongFloat sup()</code>	上端を取得する。
<code>LongFloat mag()</code>	最大絶対値を取得する。
<code>LongFloat mig()</code>	最小絶対値を取得する。

### 6.4 非メンバ関数

<code>LongInterval pow(LongInterval x, int n)</code>	$x^n$
<code>LongInterval sqrt(LongInterval x)</code>	$\sqrt{x}$

## 7 LongIntervalMatrix クラス

### 7.1 コンストラクタ

<code>LongIntervalMatrix()</code>	1 行 1 列の行列。値は不定。
<code>LongIntervalMatrix(int rows, int columns)</code>	$rows$ 行 $columns$ 列の行列。値は不定。
<code>LongIntervalMatrix(LongMatrix x)</code>	$x$ で初期化する。
<code>LongIntervalMatrix(LongIntervalMatrix x)</code>	

### 7.2 使用可能な演算子

- `+LongIntervalMatrix`
- `-LongIntervalMatrix`
- `LongIntervalMatrix + int`
- `LongIntervalMatrix - int`

- `LongIntervalMatrix * int`
- `LongIntervalMatrix / int`
- `int + LongIntervalMatrix`
- `int - LongIntervalMatrix`
- `int * LongIntervalMatrix`
- `int / LongIntervalMatrix`
- `LongIntervalMatrix + LongFloat`
- `LongIntervalMatrix - LongFloat`
- `LongIntervalMatrix * LongFloat`
- `LongIntervalMatrix / LongFloat`
- `LongFloat + LongIntervalMatrix`
- `LongFloat - LongIntervalMatrix`
- `LongFloat * LongIntervalMatrix`
- `LongFloat / LongIntervalMatrix`
- `LongIntervalMatrix + LongInterval`
- `LongIntervalMatrix - LongInterval`
- `LongIntervalMatrix * LongInterval`
- `LongIntervalMatrix / LongInterval`
- `LongInterval + LongIntervalMatrix`
- `LongInterval - LongIntervalMatrix`
- `LongInterval * LongIntervalMatrix`
- `LongInterval / LongIntervalMatrix`
- `LongIntervalMatrix + LongMatrix`
- `LongIntervalMatrix - LongMatrix`
- `LongIntervalMatrix * LongMatrix`
- `LongMatrix + LongIntervalMatrix`

- LongMatrix - LongIntervalMatrix
- LongMatrix \* LongIntervalMatrix
- LongIntervalMatrix + LongIntervalMatrix
- LongIntervalMatrix - LongIntervalMatrix
- LongIntervalMatrix \* LongIntervalMatrix
- LongIntervalMatrix += int
- LongIntervalMatrix -= int
- LongIntervalMatrix \*= int
- LongIntervalMatrix /= int
- LongIntervalMatrix += LongFloat
- LongIntervalMatrix -= LongFloat
- LongIntervalMatrix \*= LongFloat
- LongIntervalMatrix /= LongFloat
- LongIntervalMatrix += LongInterval
- LongIntervalMatrix -= LongInterval
- LongIntervalMatrix \*= LongInterval
- LongIntervalMatrix /= LongInterval
- LongIntervalMatrix = LongMatrix
- LongIntervalMatrix += LongMatrix
- LongIntervalMatrix -= LongMatrix
- LongIntervalMatrix = LongIntervalMatrix
- LongIntervalMatrix += LongIntervalMatrix
- LongIntervalMatrix -= LongIntervalMatrix
- std::ostream << LongIntervalMatrix

### 7.3 メンバ関数

<code>void resize(int rows, int columns)</code>	サイズを rows 行 columns 列 にする。 値は不定になる。
<code>int rows()</code>	行数を取得する。
<code>int columns()</code>	列数を取得する。
<code>std::string getMidRad()</code>	中心値と半径を表す文字列を取得する。
<code>std::string getInfSup()</code>	下端と上端を表す文字列を取得する。
<code>LongMatrix mid()</code>	中心値を取得する。
<code>LongMatrix rad()</code>	半径を取得する。
<code>LongMatrix diam()</code>	直径を取得する。
<code>LongMatrix inf()</code>	下端を取得する。
<code>LongMatrix sup()</code>	上端を取得する。
<code>LongMatrix mag()</code>	最大絶対値を取得する。
<code>LongMatrix mig()</code>	最小絶対値を取得する。

### 7.4 非メンバ関数

<code>LongIntervalMatrix sqrt(LongIntervalMatrix a)</code>	$y_{ij} = \sqrt{a_{ij}}$ となる行列 $Y$ を取得する。
<code>LongIntervalMatrix trans(LongIntervalMatrix a)</code>	転置行列 $A^T$ を取得する。