ファイル名　　MultiPrecision.vb

モジュール名　　MultiPrecision

（モジュール：FftDouble と共に用いること）

メソッド名　　MpADD

引数　　ByVal u() As Byte, ByVal pu As Integer,
ByVal v() As Byte, ByVal pv As Integer,
ByVal m As Integer,
ByVal w() As Byte

説明　　u(pu)から始まるmバイトの符号なしの整数と、v(pv)から始まるmバイトの符号なしの整数を加算し、結果をw(0)から始まるm+1バイトに格納する。

u(pu)とv(pv)が最上位バイトで、w(0)にはさらに桁上がりしたデータが格納される。u(pu+m-1)、v(pv+m-1)及びw(m)が最下位バイトで同じ位を示す。

u()、v()、w()が同じ配列を参照してもよい。演算結果は保証される。

メソッド名　MpSUB

引数
```
ByVal u() As Byte, ByVal pu As Integer,
ByVal v() As Byte, ByVal pv As Integer,
ByVal m As Integer,
ByVal w() As Byte, ByRef isign As Integer
```

説明

u(pu)から始まるmバイトの符号なしの整数から、v(pv)から始まるmバイトの符号なしの整数を減算し、結果をw(0)から始まるmバイトに格納する。

u(pu)、v(pv)及びw(0)が最上位バイト、u(pu+m-1)、v(pv+m-1)及びw(m-1)が最下位バイトで、それぞれ同じ位を示す。

v()> u()の場合は、isignが-1で返り、w()には2の補数表現の負の結果が格納される。それ以外はisignは0である。

u()、v()、w()が同じ配列を参照してもよい。演算結果は保証される。

メソッド名	MpNeg
引数	ByVal u() As Byte, ByVal pu As Integer, ByVal m As Integer
説明	u(pu)から始まるmバイトの符号なしの整数を、2の補数表現の負数に置き換える。u(pu)が最上位バイト、u(pu+m-1)が最下位バイトである。

メソッド名	MpSAD
引数	ByVal u() As Byte, ByVal pu As Integer, ByVal m As Integer, ByVal iv As Integer, ByVal w() As Byte
説明	u(pu)から始まる m バイトの符号なしの整数に、iv の 0 以上 255 以下の整数を加算し、結果を w(0)から始まる m+1 バイトに格納する。 u(pu)が最上位バイトで、w(0)にはさらに桁上がりしたデータが格納される。u(pu+m-1)が最下位バイトで、ここに iv が最初に加算される。 u()、w()が同じ配列を参照してもよい。演算結果は保証される。

メソッド名	MpSMU
引数	ByVal u() As Byte, ByVal pu As Integer, ByVal m As Integer, ByVal iv As Integer, ByVal w() As Byte
説明	u(pu)から始まる m バイトの符号なしの整数に、iv の 0 以上 255 以下の整数を乗じ、結果を w(0)から始まる m+1 バイトに格納する。 u(pu)が最上位バイトで、w(0)にはさらに桁上がりしたデータが格納される。u(pu+m-1)と w(m)が最下位バイトで同じ位を示す。 u()、w()が同じ配列を参照してもよい。演算結果は保証される。

メソッド名　MpSDV

引数
```
ByVal u() As Byte, ByVal pu As Integer,
ByVal m As Integer,
ByVal iv As Integer,
ByVal q() As Byte, ByRef ir As Integer
```

説明

u(pu)から始まる m バイトの符号なしの整数を、iv の 0 以上 255 以下の整数で除し、q(0)から始まる m バイトに商を、ir に剰余を格納する。

u(pu)と q(0)が最上位バイト、u(pu+m-1)と q(m-1)が最下位バイトで、それぞれ同じ位を示す。

u()、q()が同じ配列を参照してもよい。演算結果は保証される。

メソッド名　　MpMUL

引数
```
ByVal u() As Byte, ByVal pu As Integer,
ByVal m As Integer,
ByVal v() As Byte, ByVal pv As Integer,
ByVal n As Integer,
ByVal w() As Byte
```

説明

u(pu)から始まる m バイトの符号なしの整数と、v(pv)から始まる n バイトの符号なしの整数を乗じ、結果を w(0)から始まる m+n バイトに格納する。

u(pu)、v(pv)、w(0)が最上位バイトだが、一般に同じ位ではない。
u(pu+m-1)、v(pv+n-1)と w(m+n-1)が最下位バイトで同じ位を示す。

u()、v()、w()が同じ配列を参照してもよい。演算結果は保証される。

フーリエ変換、逆フーリエ変換のコンボリューション演算を利用しており、
N バイト同士の乗算であっても、$O(N^2)$オーダーではなく、
$O(N \times \log N \times \log\log N)$オーダーの演算である。

メソッド名　　MpINV

引数　　ByVal u() As Byte, ByVal pu As Integer,
ByVal m As Integer,
ByVal v() As Byte

説明　　u(pu)から始まるmバイトのデータを、最初の1バイトu(pu)（ゼロでないこと）を整数部、残りのu(pu+1)からu(pu+m-1)までを小数部とみなして、逆数をv()に格納する。

v()は、v(0)を整数部、残り全てが小数部である。このメソッドを呼び出すときのv()のバイト数で計算精度が決まる。u()の表すデータが厳密に1でない限り、v()は1未満の結果となり、v(0)はゼロである。

u()、v()が同じ配列を参照してもよい。演算結果は保証される。

数学　　Uに対して、漸化式

$$V_{i+1} = V_i(2 - UV_i)$$

で示されるV_iは、$i \to \infty$のとき$V_i \to 1/U$である。

メソッド名　　MpSQRT

引数　　ByVal u() As Byte, ByVal pu As Integer,
ByVal m As Integer,
ByVal v() As Byte, ByVal w() As Byte

説明　　u(pu)から始まるmバイトのデータを、最初の1バイトu(pu)（ゼロでないこと）を整数部、残りのu(pu+1)からu(pu+m-1)までを小数部とみなして、平方根をv()に、平方根の逆数をw()に格納する。

v()は、v(0)を整数部、残り全てが小数部である。このメソッドを呼び出すときのv()のバイト数で計算精度が決まる。w()についても同様。

u()とv()、あるいはu()とw()が同じ配列を参照してもよい。演算結果は保証される。v()とw()が同じ配列の場合は、どちらも平方が格納される。

数学　　Uに対して、漸化式

$$W_{i+1} = \frac{1}{2} W_i (3 - U W_i^2)$$

で示されるW_iは、$i \to \infty$のときUの平方根の逆数に漸近する。

$$W_i \to \frac{1}{\sqrt{U}}$$

収束した結果にUを乗じれば平方根Vを得る。

$$V = W \times U = \frac{1}{\sqrt{U}} \times U = \sqrt{U}$$

メソッド名　　MpSQRT（オーバーロード）

引数　　ByVal u() As Byte, ByVal pu As Integer,
ByVal m As Integer,
ByVal v() As Byte, ByVal w() As Byte,
ByVal wi() As Byte

説明　　前述の MpSQRT に比べて、引数に wi が追加となっている。平方根の逆数について予めある程度収束した値が用意できる場合に、漸化式の Wi の初期値として wi を用いることにより、計算が速くなる。

メソッド名	`MpDIV`
引数	`ByVal u() As Byte, ByVal v() As Byte,` `ByRef q() As Byte, ByRef r() As Byte`
説明	`u(0)`から始まる `u.length` バイトの符号なしの整数を、`v(0)`から始まる `v.length` バイトの符号なしの整数で除し、商を `q(0)`から始まる `u.length-v.length+1` バイトに、剰余を `r(0)`から始まる `v.length` バイトに格納する。 `u.length` は `v.length` 以上でなければならない。
数学	まず V の逆数 $1/V$ を `MpINV` で求め、U と $1/V$ を乗じて商 Q を求める。 剰余 R は U から VQ を差し引いて求める。

ファイル名　　RadixConv.vb

モジュール名　　RadixConv

（モジュール：MultiPrecision と共に用いること）

関数名	`RadixToDec`
戻り値型	`String`
引数	`ByVal u() As Byte, ByVal pu As Integer,` `ByVal m As Integer`

説明

`u(pu)`から始まる `m` バイトの符号なしの整数を 10 進数表現の文字列に変換して返す。

`m` がモジュールレベル定数 `BYTES_THRESHOLD`（ソースコード参照のこと）以下の場合は、$O(N^2)$の手法で変換する。さらに大きなデータの場合は、再帰呼び出しによる方法を用いる。

関数名	`RadixToDecFrac`
戻り値型	`String`
引数	`ByVal u() As Byte, ByVal pu As Integer,` `ByVal dc As Integer`

説明

`u(pu)`から始まるバイト列を、`u(pu)`の前に小数点のある小数部とみなし、`dc` 桁の `10` 進数に変換して文字列で返す。バイト列の長さの目安として、`dc` / $\log_{10}(256) + 5$ バイト必要である。

`dc` がモジュールレベル定数 `DIGITS_THRESHOLD`（ソースコード参照）以下の場合は、$O(N^2)$の手法で変換する。

さらに大きなデータの場合は、10 の dc 乗を乗じた上で、整数部に対して `RadixToDec` を呼び出して変換する。ただし、この手法は、相当に大きな桁（数万桁）になって効果が現れる。数千桁ではオーバーヘッドが大きすぎて逆効果である。

`DIGITS_THRESHOLD`（ソースコード参照）の値は、コンピュータの環境（CPU の種類、クロック数、メモリ量など）によって最適値が異なる可能性がある。