

電卓 説明書

2020/04/16

三浦 高志

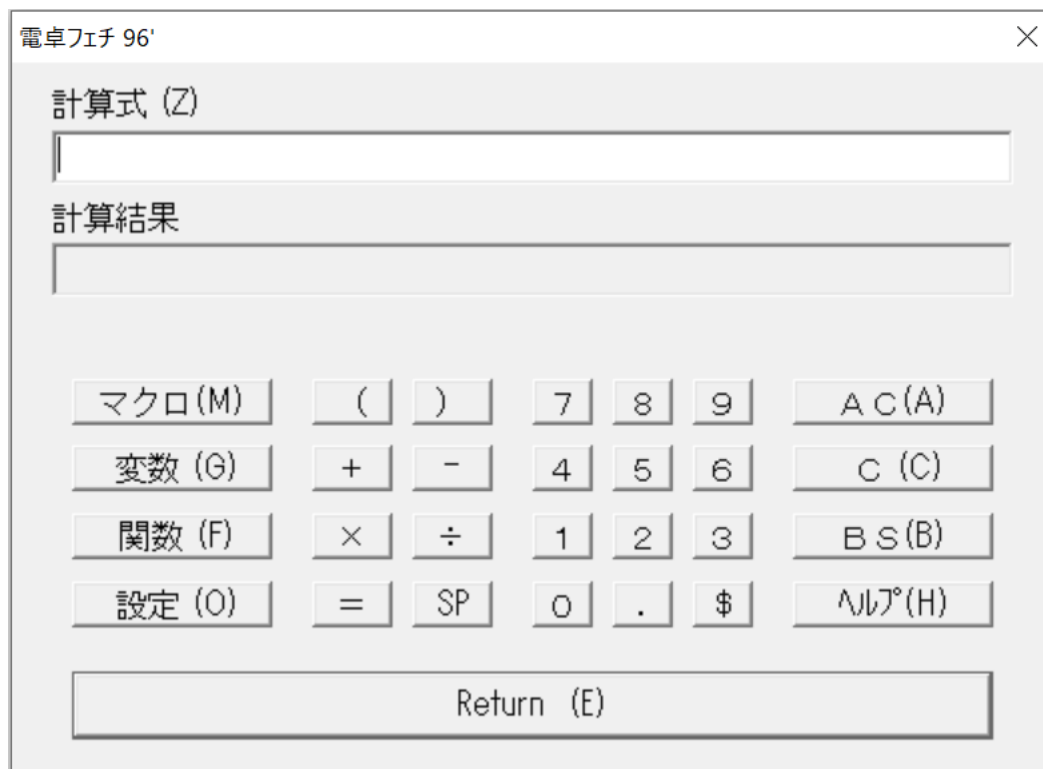
目次

目次.....	1
電卓プログラム	3
電卓プログラムのボタン	4
電卓プログラムの特徴.....	4
組み込み関数	5
計算式の入力と関数や定数の利用	6
マクロ（自作の関数）	7
マクロのセーブ	7
マクロのロード	7
マクロの例	8
マクロファイルの取り扱い.....	9
電子回路の設計における利用例.....	10
電卓プログラムの改良希望.....	16
関数の複素数対応.....	17
指数関数.....	17
三角関数・双曲線関数・対数関数・逆双曲線関数.....	18
逆三角関数・べき関数・平方根関数その他.....	19
新人研修	20
課題として提供したソースリストと作業の指示書.....	20
指示書.....	20
ソースリスト	22
COMPLEX.h	22
NODES.h.....	26
NODES.cpp	30
DENTAKU.cpp.....	32
完成したプログラムのソースリスト	39
DT_.h	39
Data.h	41
DT_DLG.h	47
Complx.h.....	50
Compute.h	63
FUNCDDL.h	65
GVARDLG.h	67
MACEDDLG.h	69

MACRODLG.h	71
NODES.h	73
OPTDLG.h	81
RESOURCE.h	83
USERMAC.h	86
STDAFX.h	89
Compute.cpp	90
DT_.cpp	102
Dt_dlg.cpp	105
FUNCDLG.cpp	120
GVARDLG.cpp	125
MACEDDLG.cpp	134
MACRODLG.cpp	136
NODES.cpp	144
OPTDLG.cpp	148
STDAFX.cpp	151
USERMAC.cpp	152

電卓プログラム

2020/03/03 三浦 高志



この電卓プログラムは1996年に入社した新人プログラマー用の課題として作成しました。私が電卓の骨組みとなる部分をC++言語のヘッダファイルとC++ソースプログラムファイルとして作成して、最終的な機能やGUIの要望を添付して新人プログラマーに渡しました。プログラマーは数ステップの段階ごとに完成したプログラムを提出し、私がそれを添削しながら、完成度を上げていくという新人研修でした。

Windows用に様々な電卓プログラムが公開されていますが、誰かがこのプログラムを改良して、複素数の計算が出来て、自作のマクロ（関数）も作成できて、さらにベクトルや行列の計算など也能するプログラムが開発されたら嬉しいと思います。

Microsoft Visual Studio 6.0 で開発した電卓プログラムのソースを添付しますので、ぜひ面白い電卓に改良して公開してください。

電卓プログラムのボタン

プログラム名は「Dt_.exe」

保存されているフォルダーにあるアイコンをダブルクリックして起動します。

アイコンをタスクバーに入れて起動することもできます。

電卓プログラムのボタン

「ヘルプ」はこの「電卓プログラム 説明書.pdf」が開きます。

「SP」はスペースが入るので、カーソルを式の右端に戻してから続きを入力します。

「AC」は記憶しているすべての変数をクリアします。

「C」は入力中の計算式をクリアします。

「BS」は入力中の計算式を1文字消して後ろに戻ります。

「0」～「9」は数字入力ボタンです。

「.」は小数点の区切りを入力します。

「\$」は自分で定義した「マクロ」を呼び出して計算結果を表示します。

「(」と「)」は計算式入力でカッコを使用する時に使用します。

「=」は変数に値を設定する時に使用します。

電卓プログラムの特徴

1. 複素数の計算ができます。
2. 多数の関数が登録されており、多くの関数は複素数の変数に対応しています。
3. 多数のメモリーを使用することが出来ます。メモリーの名前は英数文字1～10文字以下で、電卓プログラムが使用する定数とは異なる変数名が利用できます。

変数名は大文字でも小文字でも同一の変数と見なされます。

電卓が利用する定数名は以下のものがあります。

I 虚数単位	0 + 1*I	t テラ	1,000,000,000,000	p ピコ	0.000000000001
pi 円周率	3.141592654	g ギガ	1,000,000,000	n ナノ	0.000000001
e ネイピア数	2.718281828	meg メガ	1,000,000	u マイクロ	0.000001
		k キロ	1,000	m ミリ	0.001

4. 自作の「マクロ」を任意の関数名を付けて定義することが出来ます。マクロ名をfunc1とすると、数式入力で\$func1と入力して「return」をクリックすると計算結果が表示されます。マクロには任意の変数や関数を利用することが出来ます。
5. 自作のマクロを任意のフォルダーに保存したり読みだすことが出来ます。
6. 「設定」をクリックして、三角関数の角度の単位を「度」または「ラジアン」に設定できます。また、数字表示で3桁ごとにカンマを入れるか入れないかを切り替え出来ます。また、数字の有効桁数（小数点以下の桁数ではない）を指定できます。

組み込み関数

組み込み関数

関数名	解説
real(a)	複素数 a の実数部
image(a)	aの虚数部
int(a)	aの実数部と虚数部の小数部を切り捨てる
ceil(a)	aの実数部と虚数部の小数部を切り上げる
conj(a)	aの共役複素数
sin(a)	aの正弦関数値
cos(a)	aの余弦関数値
tan(a)	aの正接関数値
asin(a)	aの逆正弦関数値
acos(a)	aの逆余弦関数値
atan(a)	aの逆正接関数値
sinh(a)	aの双曲正弦関数
cosh(a)	aの双曲余弦関数
tanh(a)	aの双曲正接関数
asinh(a)	aの逆双曲正弦関数
acosh(a)	aの逆双曲余弦関数
atanh(a)	aの逆双曲正接関数
abs(a)	aの絶対値
norm(a)	aのノルム
norm2(a)	aの2乗ノルム
arg(a)	aの偏角
exp(a)	eのa乗
ln(a)	aの自然対数値
log(a)	aの常用対数値
log10(a)	aの常用対数値
sqrt(a)	aの平方根
db(a)	aのデシベル値
undb(a)	デシベル値aの倍率
badr(a)	偏差aに対する不良率
defl(a)	不良率aに対する偏差
pow(a,b)	aのb乗
unpow(a,b)	aがbの何乗かの値
rtpow(a,b)	aが何のb乗かの値
estn(a,b)	不良率a、データ数bに対する飽和データ数
build(a, b)	実数 a, b から、複素数 $a + b*i$ を作る

数式入力画面では、関数名（変数名）または関数名（数式）の形式で入力する

「関数」を押して任意の関数名を選択すると、関数の説明が表示されます。

組み込み関数

計算式の入力と関数や定数の利用

計算式の入力では、複数の式を「;」で区切って入力することが出来ます。

変数aとbに値を設定して、a*bを計算するなら、「a=356;b=525;a*b」のように最後の式の後ろには「;」が無くても構いません。

計算式	計算結果	解説
a=3.2	3.2	変数に値を代入
a=3;exp(a)	20.0855	指数関数
a=1+2*i;exp(a)	-1.1312 + 2.47173 * I	指数関数に複素数を使用
a=30;cos(a)	0.866025	三角関数
sin(1+2*i)	6.56594e-002 + 3.62631 * I	三角関数に複素数を使用
unpow(7, 3)	1.77124 →	$7 = 3^{1.77124}$
rtpow(7, 3)	1.91293 →	$7 = 1.91293^3$

電卓の定数や単位を利用した計算式

計算式	計算結果
r=10;s=pi*r*r	314.159
a=1+3*i;b=2-i;a*b	5+5*I
C=0.03*u;R=2.4*k;freq=1/(2*pi*C*R)	2210.49
C=20*p;L=50*u;freq=1/(2*pi*sqrt(C*L))	5.03E+06

計算式の入力では、スペースを「x」（乗算）の代わりに利用することが出来ます。

通常、変数aとbの乗算は、a*b と入力しますが、a b でも同じ計算結果が得られます。

変数名や関数名およびマクロ名は大文字で入力しても小文字として扱われます。

「;」の後ろなどにスペースを入れると、計算式が見やすくなります。

「SP」をクリックするとスペースが入力されますが、カーソルを計算式の右端でクリックしてから式の入力を続ける必要があります。

キーボードのスペースキーを利用した方が良いと思います。しかも、キーボードで「*」を入力するよりも楽な気がします。

計算式	計算結果
r = 10; s = pi r r	314.159
a=1+3 i; b=2-i; a b	5+5*I
C=0.03 u; R=2.4 k; freq=1/(2 pi C R)	2210.49
C=20 p; L=50 u; freq=1/(2 pi sqrt(C L))	5.03E+06

マクロ（自作の関数）

マクロ（自作の関数）

例えば、変数aと変数bの平均値を計算してcに値を設定するマクロheikinを作る場合を説明する。

「マクロ」をクリックして、「新規作成」をクリックする。

マクロ名「heikin」を入力する。

式に $c=(a+b)/2$ を入力する。

説明文に、「aとbの平均値をcに入れる」と入力する

「OK」をクリックする。もう一度、「OK」をクリックする。

「計算式」に $a=123;b=456;\$heikin$ と入力してリターンすると、289.5が表示される。

マクロのセーブ

「マクロ」をクリックして、「セーブ」をクリックする。

任意のフォルダを選択して、任意のファイル名を入力して「保存」をクリックする。

「変数」をクリックして、「セーブ」をクリックする。

任意のフォルダを選択して、任意のファイル名を入力して「保存」をクリックする。

電卓プログラムを一度停止する。

もう一度電卓プログラムを起動する。

マクロのロード

「マクロ」をクリックして、「ロード」をクリックする。

フォルダを選択して、ファイルをクリックして「開く」をクリックする。

「マクロ一覧」にマクロ名が表示される。

マクロ名を選択すると、マクロの内容が表示される。

「マクロの編集」をクリックすると、マクロの内容を編集することもできる。

「変数」をクリックして、「ロード」をクリックする。

フォルダを選択して、ファイルをクリックして「開く」をクリックする。

「計算式」に $\$heikin$ を入力すると、289.5が表示される。

組み込み関数の $\exp(a)$ や $\text{pow}(a, b)$ などは1個か2個の引数を入力しますが、「マクロ」は希望だけのメモリーを使用して計算ができるので、引数がありません。

マクロ $\$heikin$ では、2つのメモリーの値を使って計算した結果を別のメモリーに入力しています。

マクロ（自作の関数）

マクロの例

ある数値 r が変数 tt に対して、未知数 a と b により $r = a \cdot tt + b$ で表されたとする。

変数が $t1$ において $r1$ 、変数が $t2$ において $r2$ の場合には、以下の連立方程式ができる。

$$\begin{cases} r1 = t1 \cdot a + b \\ r2 = t2 \cdot a + b \end{cases}$$

これから、 a と b を求めるマクロは次のように定義できる。

「マクロ」をクリックして、「マクロ名」を `reqatb` と入力する。

「計算式」に `t3=t2-t1;a=(r2-r1)/t3;b=(r1*t2-r2*t1)/t3;c=a+b*i` と入力する。

説明文には「 $r=a \cdot t+b$ の a, b を求め、 $a+b \cdot i$ を表示する」と入力する。

計算式に `t1, r1, t2, r2` を入力して、`$reqatb` を入力すると、 $a+b \cdot i$ が表示される。

表示された数値は複素数の形式で、実数部は a の値、虚数部は b の値である。

「計算式」に `t1=12;r1=0.967;t2=22;r2=0.82;$reqatb` と入力すると、

$-1.47e-002 + 1.1434 \cdot i$ と表示される。

これで、 $a = -0.0147$, $b = 1.1434$ が求められた。

マクロは定義された計算式を実行すると同時に、計算結果をメモリーに代入することもできる。そして、マクロで定義された最後の計算式の結果が表示される。

このマクロの例では、最後に定義された $a+b \cdot i$ の計算結果が表示される。この計算式がなければ、 b が表示される。未知数は2個なので、両方を同時に表示するために、複素数形式に変換する計算式を最後に定義した。メモリー c にも同じ数値が格納されている。変数 c に代入しなくても表示は同じである。

電卓プログラムを解凍したフォルダーには、サンプルのマクロ `macro-関数.UMF` が入っています。「マクロ」をクリックしてロードしてください。次に、「変数」をクリックして `macro-メモリー.UVF` をロードしてください。こちらは、上で説明したマクロ `$reqatb` で使用するメモリーが記録されています。計算式に `$reqatb` と入力すれば、上の結果が表示されます。

また、「マクロ」をクリックすると「マクロ一覧」に複数のマクロ名が表示されます。

`$build` はメモリー x と y から複素数 $x+i \cdot y$ を計算します。`$norm` は複素数 a の2乗ノルム $x^2 + y^2$ を計算します。計算式に `x=2;y=3;a=conj($build)` と入力すると、 $2-3 \cdot i$ が表示されます。このように関数の引数として、マクロ名を含む数式を利用できます。

次に、`$norm` と入力すると、 13 が表示されます。

複数のマクロを1つのファイルに集めて用途別に関数ライブラリが作れます。

ver.1.2 では、組み込み関数 `build(a,b)` ($a+b \cdot i$ を出力)、`norm(a)` (a のノルムを計算)、`norm2(a)` (a の2乗ノルムを計算)を追加しました。

マクロ（自作の関数）

マクロファイルの取り扱い

電卓プログラムを起動した直後は、マクロもメモリー（変数）もない状態です。

自作のマクロを利用する時には「マクロ」をクリック→「ロード」からマクロのファイルをロードして「マクロ一覧」に表示されているマクロを利用してください。

同じマクロファイルにさらにマクロを追加したい時は、「マクロ」→「新規作成」から、マクロ名の入力、式の入力、説明文の入力を行ってから、「セーブ」してください。

「マクロ一覧」に表示されているマクロの一部を削除する場合は、そのマクロを選択してから「マクロの削除」をクリックします。

違う目的のためにマクロのファイルを作成する場合は、プログラムを起動直後に「マクロ」→「新規作成」からマクロを作成して「セーブ」してください。

作成したマクロの内容を修正する場合は「マクロ」をクリックして、マクロを選択して「マクロの編集」により編集して、「セーブ」してください。

メモリー（変数）は計算式に適当な変数名(**sum, width, height** など)を入力すると、登録されます。変数名だけを入力すると、値は **0** として登録されます。

値を変更したい時は計算式に、**width=4500** のように入力します。

電卓プログラムが使用する定数に値を設定しようとすると、エラーが表示されます。

メモリーのファイルは必要に応じて、「変数」をクリックして「セーブ」または「ロード」してください。

変数一覧に表示されているメモリーに値を設定する場合は、変数を選択して値を入力してから「アサイン」をクリックします。

電卓プログラムが使用する定数に値を設定しようとすると、エラーが表示されます。

自分が作った変数を削除する時は、「変数」をクリックして変数を選択してから「変数の削除」をクリックします。

電卓プログラムが使用する定数を削除しようとすると、エラーが表示されます。

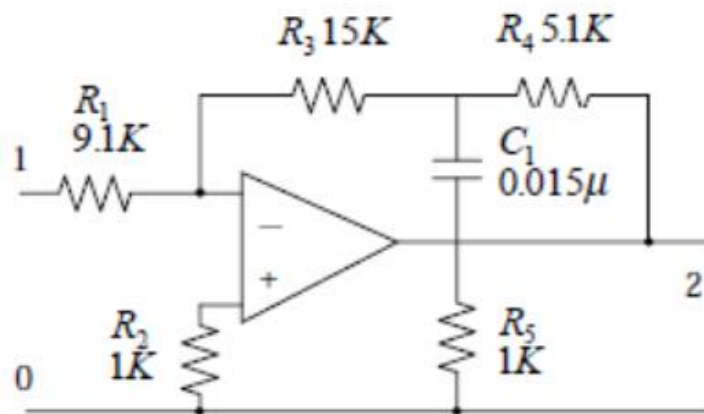
電子回路の設計における利用例

アナログ TV の時代には、テレビの音声信号が電波ノイズ（特に高周波数域）で音質劣化するのを防ぐために、あらかじめ音声信号にプリエンファシス処理を行っていました。この処理は音声信号の周波数が f_{c2} を超えるほど信号を拡大して、周波数が f_{c1} を超えると一定の大きさに落ち着いて行くようなフィルタです。

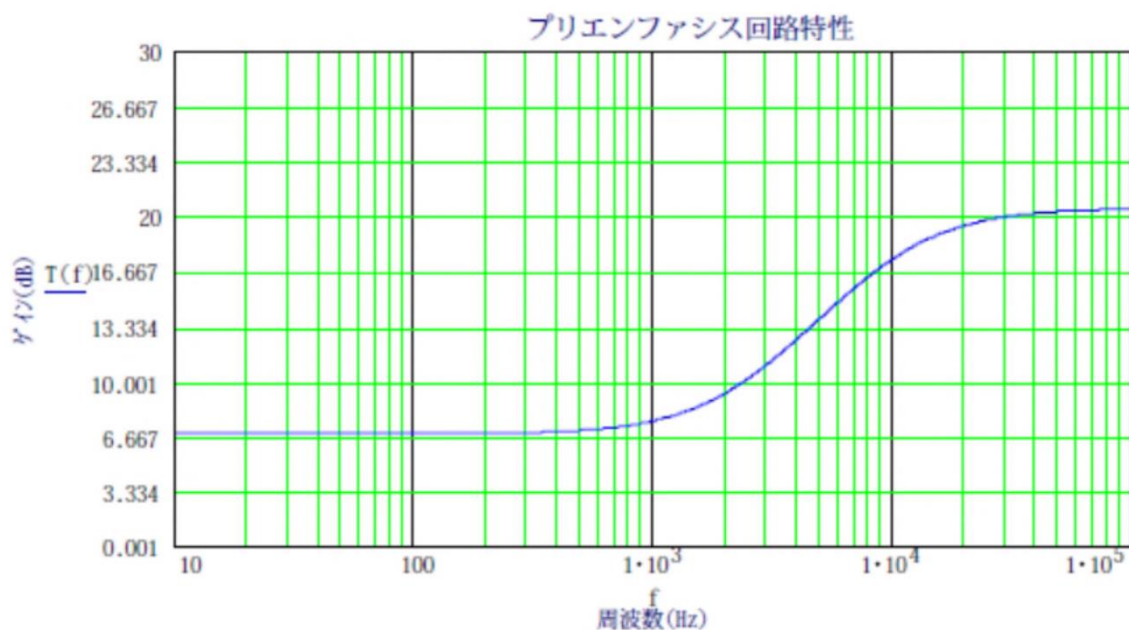
テレビ受像機側ではこの逆の処理を行うディエンファシス処理を行って、高周波数のノイズは低減しながら、音声信号は元の大きさに戻していました。

アナログ TV 規格では、 $f_{c2} = 2,122 \text{ Hz}$ (時定数 $t_{c2}=75\mu$)、 $f_{c1} = 10,610\text{Hz}$ ($t_{c1}=15\mu$)です。

下図がプリエンファシス回路の一例です。



このプリエンファシス回路の周波数特性は次のようなものです。



マクロ（自作の関数）

上のグラフを見ると、周波数が 1KHz あたりから少しずつゲインが増加して、2KHz を少し超えた所（おそらく $f_{c2} = 2122 \text{ Hz}$ ）でゲインが約 3dB 増加しているのが分かります。周波数がさらに上昇して 10 KHz（おそらく $f_{c1} = 10,610 \text{ Hz}$ ）を超えると、ゲインの増加が衰え始めて 30 KHz 位ではほぼ一定のゲインに落ち着くのが分かります。

この電卓プログラムを使って、周波数特性が正確に f_{c1} と f_{c2} になっているかを、回路図の素子値を使って計算してみます。

この回路の伝達関数は次のように書けます。（ s はラプラス演算子）

$$G(s) = -\frac{R_3 + R_4}{R_1} \cdot \frac{1 + sC_1(R_5 + \frac{R_3 \cdot R_4}{R_3 + R_4})}{1 + sC_1R_5}$$

この伝達関数を周波数 f を使って複素数表現の数式に簡略に書き直します。

$$G(f) = -gain \cdot \frac{1 + i \cdot 2\pi \cdot f \cdot t_{c2}}{1 + i \cdot 2\pi \cdot f \cdot t_{c1}} = -gain \cdot \frac{1 + i \cdot \frac{f}{f_{c2}}}{1 + i \cdot \frac{f}{f_{c1}}}$$

ゲインは $G(f)$ の絶対値なので、次式で得られます。

$$|G(f)| = gain \cdot \frac{\left| 1 + i \cdot \frac{f}{f_{c2}} \right|}{\left| 1 + i \cdot \frac{f}{f_{c1}} \right|} = gain \cdot \sqrt{\frac{1 + \left(\frac{f}{f_{c2}}\right)^2}{1 + \left(\frac{f}{f_{c1}}\right)^2}}$$

ここで、

$$f_{c1} = \frac{1}{2\pi \cdot C_1 \cdot R_5}; f_{c2} = \frac{1}{2\pi \cdot C_1 \cdot (R_5 + \frac{R_3 \cdot R_4}{R_3 + R_4})}$$

また、

$$gain = \frac{R_3 + R_4}{R_1}; \frac{f_{c1}}{f_{c2}} = \frac{(R_5 + \frac{R_3 \cdot R_4}{R_3 + R_4})}{R_5}$$

$|G(f)|$ の式から、周波数が $f=0$ の場合には、ゲインは $gain$ になることが分かります。

アナログ TV 規格では $\frac{f_{c1}}{f_{c2}} = 5$ なので、周波数が低いうちはゲインはほぼ $gain$ のままで、

周波数が $f_{c2} = \frac{1}{2\pi \cdot C_1 \cdot (R_5 + \frac{R_3 \cdot R_4}{R_3 + R_4})}$ を超えるとゲインが徐々に上昇していくことが分かります。

そして、さらに周波数が上昇して、 $f_{c1} = \frac{1}{2\pi \cdot C_1 \cdot R_5}$ を超えると一定値に落ち着いて行くことが

分かります。最終的なゲインは $f \rightarrow \infty$ にすると、

マクロ（自作の関数）

$$\text{gainhigh} = \text{gain} \cdot \frac{f_{c1}}{f_{c2}} = \frac{R_3 + R_4}{R_1} \cdot \frac{(R_5 + \frac{R_3 \cdot R_4}{R_3 + R_4})}{R_5}$$

になり、 $f=0$ に比べると $\frac{f_{c1}}{f_{c2}} = 5 \rightarrow 13.98 \text{ dB}$ だけゲインが上昇することが分かります。

それでは、電卓プログラムを使って 2 つの周波数とゲインを計算してみましょう。

4 個のマクロを作成します。

1. マクロ名：peset

式： $c1=0.015 \text{ u}; r1=9.1 \text{ k}; r3=15 \text{ k}; r4=5.1 \text{ k}; r5=1 \text{ k};$

説明文：プリエンファシス回路の定数セット

2. マクロ名：calfc

式： $rr=r5+r3 \cdot r4/(r3+r4); tc1=c1 \cdot r5; tc2=c1 \cdot rr; fc1=1/(2 \pi \cdot tc1); fc2=1/(2 \pi \cdot$

$tc2); \text{gain}=(r3+r4)/r1; \text{gainhigh}=\text{gain} \cdot rr/r5; \text{gdb}=\text{db}(-\text{gain}); \text{ghdb}=\text{db}(-\text{gainhigh})$

説明文：プリエンファシスフィルタのカットオフ周波数 $fc2$, $fc1$ とゲイン gain , gainhigh を計算する

3. マクロ名：fc2set

式： $\text{gain}=2; rr34=75 \text{ u}/c1 \cdot r5; r4=rr34 \cdot r3/(r3-rr34); r1=(r3+r4)/\text{gain}; \calfc

説明文：ゲインを 2 に設定して、 $fc2$ が TV 規格に合うように、 $r4$ と $r1$ を変更して、 $fc1$, $fc2$, gain などを計算する

4. マクロ名：pecomp

式： $\$peset; \$\text{calfc}; \$\text{fc2set}$

説明文：プリエンファシスフィルタの素子値設定から修正まで

計算式に $\$peset$ を入力すると、回路図の値がセットされます。

「変数」をクリックして、 $c1$, $r1$, $r3$, $r4$, $r5$ に値が設定されていることが分かります。

計算式に $\$calfc$ を入力すると、回路図の値で $fc1$, $fc2$, gain , gainhigh が計算されます。

同時に gain のデシベル値 gdb と gainhigh のデシベル値 ghdb も計算されます。

確認すると、

$fc2 = 2207.74$, $fc1 = 10610.33$, $\text{gdb} = 6.88 \text{ db}$, $\text{ghdb} = 20.52 \text{ db}$

が確認できます。

周波数特性のグラフを見ると、低周波数のゲインは $\text{gdb} = 6.88 \text{ db}$ に近く、周波数が高くなりゲインが一定値に近付いた時は $\text{ghdb} = 20.52 \text{ db}$ に近い値に見えます。

それから、 $fc1$ の計算値は正確に規格値ですが、 $fc2$ は少し誤差があります。

マクロ（自作の関数）

誤差は小さいと思いますが、マクロの練習のために **fc2** を規格値に合わせて確認します。

fc1 が正しいので、 $f_{c1} = \frac{1}{2\pi \cdot C_1 \cdot R_5}$; $f_{c2} = \frac{1}{2\pi \cdot C_1 \cdot (R_5 + \frac{R_3 \cdot R_4}{R_3 + R_4})}$ より、 $C_1 = 0.015 \text{ u}$; $R_5 = 1 \text{ k}$ は現在値

のままとします。**fc2** は $R_3 = 15 \text{ k}$; $R_4 = 5.1 \text{ k}$ で調節できますが R_3 は現在値のままで、

R_4 を調節します。 $(R_5 + \frac{R_3 \cdot R_4}{R_3 + R_4}) = \frac{t_{c2}}{C_1} = \frac{75 \text{ u}}{0.015 \text{ u}} = 5 \text{ k}$ より $\frac{R_3 \cdot R_4}{R_3 + R_4} = 5 \text{ k} - R_5 = 4 \text{ k}$

従って、 $rr = 4 \text{ k}$ とすると、 $\frac{R_3 \cdot R_4}{R_3 + R_4} = rr$ より、 $R_4 = \frac{R_3 \cdot rr}{R_3 - rr}$ が求められます。

ゲインは、 $gain = \frac{R_3 + R_4}{R_1} = 2$ より $R_1 = \frac{R_3 + R_4}{gain}$ が求められます。

計算式に **\$fc2set** を入力すると、**fc2** が 2,122 Hz ($t_{c2}=75\text{u}$) になるように、**r4** を調整してから、ゲインが 2 になるように **r1** を調整します。

確認すると、

fc2 = 2122.07, **fc1 = fc1 = 10610.33**, **gdb = 6.02 db**, **ghdb = 20 db**,

r1 = 10227.27, **r4 = 5454.55**

のように確認が出来ます。これで、**fc1**, **fc2**, ゲインを希望通りの値に調整することが出来ました。

最後に、あと 1 つマクロを作成して、周波数ごとのゲインを計算して、回路図の素子値による周波数特性と TV 規格に合わせた時の周波数特性をグラフにして比較します。

ゲインは、 $|G(f)| = gain \cdot \sqrt{\frac{1 + (\frac{f}{f_{c2}})^2}{1 + (\frac{f}{f_{c1}})^2}}$ で計算できるので、周波数 **frq** の時のゲイン **gdb** を

計算するマクロは次のように書けます。

マクロ名：**caldbvsfrq**

式：

num=1+pow((frq/fc2),2);den=1+pow((frq/fc1),2);gcal=gain*sqrt(num/den);gdb=db(gcal)

説明文：周波数 **frq** ごとのゲイン **gdb** を計算する

マクロ（自作の関数）

計算式に\$peset;\$scalfcを入力して、回路図の素子値で変数をセットします。次に、
frq=10;\$caldbvsfrqを入力して、周波数ごとのゲイン gdb を求めて Excel の表に記録しま
す。frq=100 k まで計算したら、計算式に\$fc2setを入力して、素子値を TV 規格に調整し
てから、先程と同様に frq を変えながら周波数ごとのゲインを計算して記録します。

周波数 Hz	図面の素子値 ゲインdb	TV規格の素 子値 ゲインdb	誤差 db
10	6.88	6.02	0.86
100	6.89	6.03	0.86
200	6.92	6.06	0.86
500	7.09	6.25	0.84
1,000	7.66	6.85	0.81
2,000	9.33	8.63	0.7
5,000	13.89	13.31	0.58
10,000	17.45	16.92	0.53
20,000	19.49	18.97	0.52
50,000	20.34	19.82	0.52
100,000	20.47	19.95	0.52

回路図の素子値では、周波数 10 Hz から 100 KHz の範囲で TV 規格との誤差は 1db 未満
に収まっています。テレビのボリュームで音量を変えても周波数特性は変化しないの
で、誤差の平均値 0.69db だけ図面のゲインを下げるか、TV 規格のゲインを上げて比較す
ると周波数特性の違いがはっきり判断できると思います。

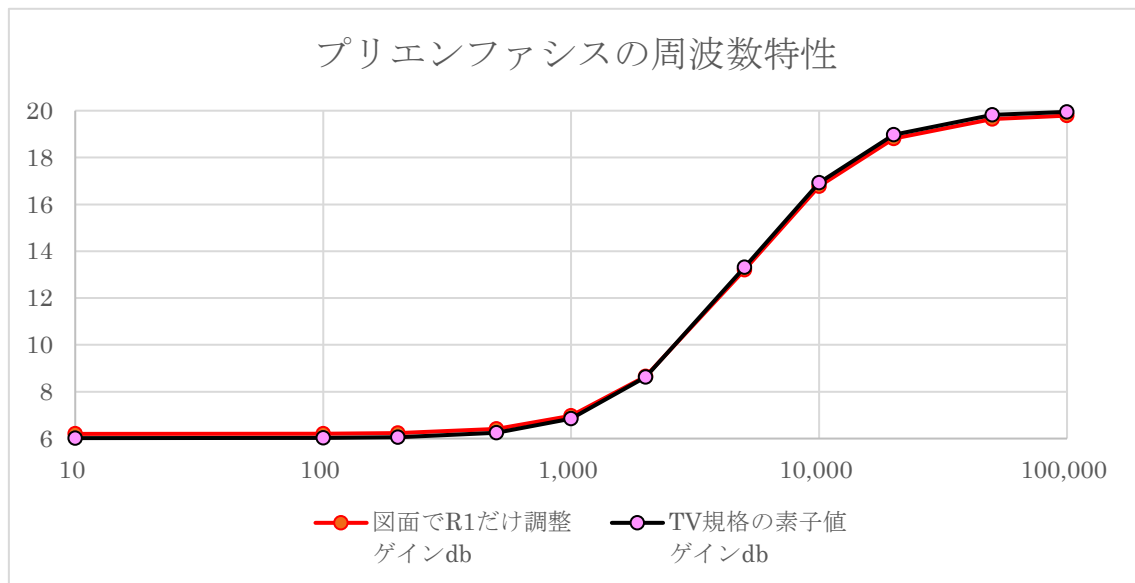
回路図のゲインを 0.69db だけ下げるために r1= 9845 に設定して、先程と同様に周波数特
性を計算すると、下表のように TV 規格との誤差は±0.2 db 未満になりました。

つまり、回路図の周波数特性は現在の素子値で、十分に TV 規格に近いということです。

周波数 Hz	図面でR1だけ 調整 ゲインdb	TV規格の素 子値 ゲインdb	誤差 db
10	6.2	6.02	0.18
100	6.21	6.03	0.18
200	6.23	6.06	0.17
500	6.41	6.25	0.16
1,000	6.97	6.85	0.12
2,000	8.65	8.63	0.02
5,000	13.20	13.31	-0.11
10,000	16.77	16.92	-0.15
20,000	18.81	18.97	-0.16
50,000	19.65	19.82	-0.17
100,000	19.79	19.95	-0.16

マクロ（自作の関数）

周波数特性の比較グラフ（Excel の表より）



電子回路を製造するには、コストと精度を考慮に入れる必要があります。抵抗やキャパシタなどの素子値は、通常は E シリーズという飛び飛びの値の部品を利用します。安価なのは E24 シリーズですが、要求される精度を実現できない場合には、E48～E192 の高価な部品を利用します。回路図では E24 シリーズを使い f_{c2} の誤差は 4% 程度ですが、安価に製造できます。これを TV 規格にすると E192 シリーズになってコストが上がります。回路図で E24 シリーズのまま誤差を減らすために、 r_3 と r_4 の組み合わせ 17 通りを計算した所、幸運にも誤差が 0 となる唯一の組み合わせを発見しました。その結果は、 $r_3=27\text{ k}$; $r_4=4.7\text{ k}$; $r_1=16\text{ k}$ です。マクロ「peset2」で確認できます。

マクロのサンプル「macro-関数.UMF」はそのままでは読めないもので、テキストに変換して読みやすく編集して、「macro-関数.txt」と「macro-関数.pdf」を同梱しました。電子回路のシミュレーションについては、「Vector」のページから「[Sim for DOS](#)」というフリーソフトに同梱されている「回路シミュレーション.pdf」を参考にしてください。このソフトは初期バージョンを 1985 年頃に開発して 1997 年に Vector に登録しました。16bit の DOS 窓ソフトは windows 10 では起動出来なくなりましたので、32bit の DOS 窓用に作りなおして 2020 年 4 月に差し替えしました。このプログラムの複素関数を拡張して電卓プログラムの課題用に利用しました。電卓プログラム「Dt_.exe」は、研修の電卓が完成後にさらに複素関数を追加しました。また、「[McAct2W](#)」と「[ActDoc](#)」はフィルタの学習の参考になると思いますので、興味があればダウンロードしてください。作者のページ <https://www.vector.co.jp/vpack/browse/person/an008575.html>

電卓プログラムの改良希望

もしも誰かがこの電卓プログラムを改良してくれるのなら、ベクトルや行列の計算関数の追加も嬉しいですが、むしろ、マクロを強化する関数の実装をお願いしたいと思います。

1. `for(n=1 to 100){計算式; 計算式}` の形式による繰り返し関数の組み込み
2. `Fopen{test}` の形式によるテキストファイル (`test.txt`) のオープン。マクロ終了時に自動的に `close` されるものとする。
3. `Fprint{n, a, b}` の形式によってテキストファイルに数値を出力して改行する。数値と数値の間はスペースまたはタブで隙間を開ける。ファイルを Excel で利用可能にする。
4. マクロ関数の中で他のマクロを実行する時には、すでに電卓プログラム中に存在しているものとする。無い場合にはエラーメッセージを表示してマクロを停止する。

拡張したマクロの利用例

電卓プログラムを解凍したフォルダにはマクロファイル「`macro-関数.UMF`」と変数データ「`macro-メモリー.UVF`」が含まれているので、電卓を起動したらこれらのファイルをロードしておきます。

このマクロファイルには「`abtor`」というマクロ関数が含まれています。このマクロはある数値 r が変数 tt に対して、未知数 a と b により $r = a \cdot tt + b$ で表される時に、 a と b が決定された時に、与えられた tt に対する r を出力する関数です。

上記のファイルをロードすると、 $a = -0.0147$, $b = 1.1434$ が設定されています。

そして、 tt を 12 から 24 まで 1 ずつ増加した時の r の値をファイルに出力するマクロは次のように書けます。

マクロ名「`tt-r 数値表`」

式

```
Fopen{ tt-r 数値表;tt=12;for(n=1to13) {$abtor; Fprint{tt, r};tt=tt+1 }
```

説明文

$tt=12$ から 24 に対する r の数値をファイル「`tt-r 数値表.txt`」に出力する

このようなマクロ機能が実装されたら、`sin` 関数や `log` 関数その他の関数の詳細な数値表を簡単にファイルに作成することが可能になります。

マクロ名「`sin 関数 数値表`」

式

```
Fopen{ sin 関数 数値表;x=0;for(n=1to901) {Fprint{x, sin(x),x+0.05,sin(x+0.05)};x=x+0.1 }
```

説明文

角度 $x=0$ から 90 度まで $\sin x$ の値を 0.05 度刻みで「`sin 関数 数値表.txt`」に出力する

指数関数

関数の複素数対応

オイラーの公式 (e をネイピア数、i を虚数単位とする)

$$e^{i \cdot x} = \cos(x) + i \cdot \sin(x)$$

これから、

$$e^{-i \cdot x} = \cos(-x) + i \cdot \sin(-x) = \cos(x) - i \cdot \sin(x)$$

従って、

$$\begin{cases} e^{i \cdot x} = \cos(x) + i \cdot \sin(x) \\ e^{-i \cdot x} = \cos(x) - i \cdot \sin(x) \end{cases}$$

この連立方程式から、**sin(x)**と**cos(x)**を求めると、

$$\begin{cases} \sin(x) = \frac{e^{i \cdot x} - e^{-i \cdot x}}{2 \cdot i} \\ \cos(x) = \frac{e^{i \cdot x} + e^{-i \cdot x}}{2} \end{cases}$$

次に、オイラーの公式の x を複素数にすると、

$$e^{x+i \cdot y} = e^x \cdot e^{i \cdot y} = e^x \cdot \{\cos(y) + i \cdot \sin(y)\}$$

となる。

指数関数

従って、**指数関数** $\exp(a) = e^a$ は、変数 a が複素数 $x + i \cdot y$ の場合には、

$$\exp(x + i \cdot y) = e^{x+i \cdot y} = e^x \cdot \{\cos(y) + i \cdot \sin(y)\}$$

のように、実数の関数を使って表現することが出来る。

また、双曲線関数は指数関数を使って次のように表すことができる。

$$\begin{cases} \sinh(x) = \frac{e^x - e^{-x}}{2} \\ \cosh(x) = \frac{e^x + e^{-x}}{2} \end{cases}$$

従って、正弦関数と余弦関数を複素数に拡張すると、

$$\begin{cases} \sin(x + i \cdot y) = \frac{e^{i \cdot (x+i \cdot y)} - e^{-i \cdot (x+i \cdot y)}}{2 \cdot i} \\ \cos(x + i \cdot y) = \frac{e^{i \cdot (x+i \cdot y)} + e^{-i \cdot (x+i \cdot y)}}{2} \end{cases}$$

三角関数・双曲線関数・対数関数・逆双曲線関数

三角関数・双曲線関数・対数関数・逆双曲線関数

ここで、正弦関数を整理する。

$$\begin{aligned}
 \sin(x + i \cdot y) &= \frac{e^{i \cdot (x + i \cdot y)} - e^{-i \cdot (x + i \cdot y)}}{2 \cdot i} \rightarrow \frac{-i}{2} \cdot \{e^{i \cdot (x + i \cdot y)} - e^{-i \cdot (x + i \cdot y)}\} \\
 &\rightarrow \frac{-i}{2} \cdot \{e^{-y} \cdot (\cos x + i \cdot \sin x) - e^y \cdot (\cos x - i \cdot \sin x)\} \\
 &\rightarrow \sin x \cdot \frac{e^y + e^{-y}}{2} + i \cdot \cos x \cdot \frac{e^y - e^{-y}}{2} \rightarrow \sin(x) \cdot \cosh(y) + i \cdot \cos(x) \cdot \sinh(y)
 \end{aligned}$$

よって**正弦関数** $\sin(a)$ は、変数 a が複素数 $x + i \cdot y$ の場合には、

$$\sin(x + i \cdot y) = \sin(x) \cdot \cosh(y) + i \cdot \cos(x) \cdot \sinh(y)$$

のように、実数の関数を使って表現できる。

同様にして、**余弦関数** $\cos(a)$ は次のように整理できる。

$$\cos(x + i \cdot y) = \cos(x) \cdot \cosh(y) - i \cdot \sin(x) \cdot \sinh(y)$$

同様にして、**双曲線関数**は次のように整理できる。

$$\begin{cases} \sinh(x + i \cdot y) = \cos(y) \cdot \sinh(x) + i \cdot \sin(y) \cdot \cosh(x) \\ \cosh(x + i \cdot y) = \cos(y) \cdot \cosh(x) + i \cdot \sin(y) \cdot \sinh(x) \end{cases}$$

これらの結果より、**正接関数**は次式で与えられる。

$$\begin{cases} \tan(x + i \cdot y) = \frac{\sin(x + i \cdot y)}{\cos(x + i \cdot y)} \\ \tanh(x + i \cdot y) = \frac{\sinh(x + i \cdot y)}{\cosh(x + i \cdot y)} \end{cases}$$

自然対数関数では、変数が $x + i \cdot y = \sqrt{x^2 + y^2} \cdot (\cos \theta + i \cdot \sin \theta)$ で表される時、

$$\log(x + i \cdot y) = \frac{\log(x^2 + y^2)}{2} + i \cdot \operatorname{atan}\left(\frac{y}{x}\right)$$

常用対数関数は、

$$\log_{10}(x + i \cdot y) = \frac{\log(x + i \cdot y)}{\log(10)} = \frac{1}{\log(10)} \cdot \left\{ \frac{\log(x^2 + y^2)}{2} + i \cdot \operatorname{atan}\left(\frac{y}{x}\right) \right\}$$

逆双曲線関数は次のように求められます。

$$\begin{cases} \operatorname{asinh}(x + i \cdot y) = \log\left\{(x + i \cdot y) + \sqrt{(x + i \cdot y)^2 + 1}\right\} \\ \operatorname{acosh}(x + i \cdot y) = \log\left\{(x + i \cdot y) + \sqrt{(x + i \cdot y)^2 - 1}\right\} \\ \operatorname{atanh}(x + i \cdot y) = \frac{1}{2} \cdot \log\left(\frac{1 + x + i \cdot y}{1 - x - i \cdot y}\right) \end{cases}$$

逆三角関数・べき関数・平方根関数その他

逆三角関数・べき関数・平方根関数その他

逆三角関数は、次のように求められます。

$$\begin{cases} \operatorname{asin}(x + i \cdot y) = -i \cdot \operatorname{asinh}(-y + i \cdot x) \\ \operatorname{acos}(x + i \cdot y) = i \cdot \operatorname{acosh}(x + i \cdot y) \\ \operatorname{atan}(x + i \cdot y) = -i \cdot \operatorname{atanh}(-y + i \cdot x) \end{cases}$$

べき乗関数は、 a と b を複素数として、

$$\operatorname{pow}(a, b) = a^b = e^{\{b \cdot \log(a)\}}$$

平方根関数は、 a を複素数として、

$$\operatorname{sqrt}(a) = a^{\frac{1}{2}} = e^{\{\frac{1}{2} \log(a)\}}$$

 $\operatorname{unpow}(a, b)$ は、 a が b の x 乗 ($b^x = a$) の場合、その x を出力します。

$$\operatorname{unpow}(a, b) = \frac{\log(a)}{\log(b)}$$

 $\operatorname{rtpow}(a, b)$ は、 a が x の b 乗 ($x^b = a$) の場合、その x を出力します。

$$\operatorname{rtpow}(a, b) = \operatorname{pow}\left(a, \frac{1}{b}\right)$$

 $\operatorname{real}(a)$ は、複素数 a の実部を出力します。

$$\operatorname{real}(x + i \cdot y) = x$$

 $\operatorname{imag}(a)$ は、複素数 a の虚部を出力します。

$$\operatorname{imag}(x + i \cdot y) = y$$

 $\operatorname{conj}(a)$ は、複素数 a の共役複素数を出力します。

$$\operatorname{conj}(x + i \cdot y) = x - i \cdot y$$

 $\operatorname{abs}(a)$ は、 $a = x + y \cdot i$ とすると、 $\sqrt{x^2 + y^2}$ を出力する。複素数 a の絶対値。 $\operatorname{int}(a)$ は、複素数 a の実部と、虚部の小数点以下を切り捨てる。 $\operatorname{ceil}(a)$ は、複素数 a の実部と、虚部の小数点以下を切り上げる。 $\operatorname{norm}(a)$ も、 $a = x + y \cdot i$ とすると、 $\sqrt{x^2 + y^2}$ を出力する。ベクトル a の長さ (ノルム)。 $\operatorname{norm2}(a)$ は、 $a = x + y \cdot i$ とすると、 $x^2 + y^2$ を出力する。ベクトル a の 2 乗ノルム。 $\operatorname{arg}(a)$ は、複素数 a の偏角を出力する。 $\operatorname{build}(a, b)$ は、複素数 a と b の実部をそれぞれ x と y とする時、 $x + i \cdot y$ を出力する。 $\operatorname{dbfrac}(a, b)$ は、複素数 a / b のデシベル値を出力する。 $\operatorname{recip}(a)$ は、複素数 a の逆数を出力する。 $\operatorname{swap}(a)$ は、 $a = x + y \cdot i$ とすると、 $y + i \cdot x$ を出力する。 $\operatorname{cswap}(a)$ は、 $\operatorname{conj}(\operatorname{swap}(a))$ の処理を行い、 $y - i \cdot x$ を出力する。 $\operatorname{swapc}(a)$ は、 $\operatorname{swap}(\operatorname{conj}(a))$ の処理を行い、 $-y + i \cdot x$ を出力する。

新人研修

課題として提供したソースリストと作業の指示書

指示書

C++言語学習用教材 1

1996/01/08

出題 三浦 高志

ディレクトリ `cpp__tst` には、以下のファイルが入っています。

1. `dentaku.cpp`
2. `nodes.h`
3. `nodes.cpp`
4. `complex.h`
5. `dentaku.ide`

5. の `dentaku.ide` は上の 1 から 4 までのファイルをボーランド C 4. 5 でメイ
クするためのプロジェクトファイルです。

注意：ビジュアル C 5. 0 で開発する場合は、`dentaku.ide` は不要です。

DOS 用のプログラムですが、このプロジェクトファイルでは `easwin` のオブジェ
クトが生成されます。この様にすると、DOS 用のアプリでも `windows` 上でデバッグ
が出来るようになります。

1 から 4 のソースプログラムを改良して、複素数対応の関数電卓プログラムを完成して下
さい。プログラムの改良は以下のステップで行って下さい。各ステップごとにディレクトリ
を作成して、そのステップのソースファイルを保存して下さい。

ステップ 1。 `Id` クラスから `Func` クラスを派生させて、次の関数が利用出来る様に
して下さい。

- | | |
|-----------------------------|-----------------------------------|
| 1. <code>real</code> (式) | 式の値の実数部を返す |
| 2. <code>image</code> (式) | 式の値の虚数部を返す |
| 3. <code>abs</code> (式) | 式の値の絶対値を返す |
| 4. <code>arg</code> (式) | 式の値の偏角を返す |
| 5. <code>exp</code> (式) | $e^{\text{式}}$ の値を返す |
| 6. <code>log</code> (式) | 式の値の自然対数値を返す |
| 7. <code>pow</code> (式1、式2) | $(\text{式1})^{(\text{式2})}$ の値を返す |

8. `s q r t` (式) 式の値の平方根を返す
これらの関数はすべて、`c o m p l e x . h`において定義されているので、
関数へのポインタを利用して使う様にしてください。

ステップ 2。ユーザ定義の手続きを登録して、利用出来る様にしてください。

ユーザ定義の手続き名を`m i u r a`とし、
その中で次の手続きが定義されているとします。

```
a = s q r t ( b )
```

この時このユーザ定義関数を実行するには、次の様に入力するものとします。

```
$ m i u r a
```

ユーザ定義の実行後は、計算結果の値を保持するものとします。

(関数`e v a l ()`の結果として)

また、この手続きで代入されたユーザ定義の変数は計算結果の値に更新
されるものとします。

次に、複数の処理が定義可能な様に拡張して下さい。

```
a = s q r t ( b ) ; d = l o g ( a )
```

この場合には、関数`e v a l ()`の結果として最後に実行した手続きの
結果を保持するものとします。

ユーザ定義の手続きの最大文字数は128文字までとします。

ステップ 3。ユーザ定義の変数及び手続きをファイルにセーブして、
再利用出来る様にしてください。セーブ・ロードの追加。
ユーザ定義の変数や手続きの削除が可能となるように改良して下さい。

ステップ 4。表示の精度（小数点以下の桁数）、カンマ（小数点以上の数字に
3桁毎にカンマをつける）などのモード切り替えのコマンドを追加して
下さい。ユーザ定義の変数の一覧表示や手続きの一覧表示もほしいですね。
この時ついでに、計算結果の値で虚数部が0の時には実数部のみ
を表示し、実数部が0の時には「I (値)」のみを表示するオートスイッチ
も追加して下さい。

ステップ 5。上記の改良を行ったプログラムを`w i n d o w s`用に移植して下さい。
もちろん、`w i n d o w s`に付属している関数電卓の外観と機能及び操作性
を超えることを目標にがんばって下さい。
オンラインヘルプで使い方が分かるようにして下さい。

課題として提供したソースリストと作業の指示書

ソースリスト

COMPLEX.h

```
#include <math.h>

#define topexp 150
#define min_num 1.e-150
#define max_num 1.e150
#define max_num2 1.e300

class complx
{
double x, y;
public:
    complx(double r = 0, double i = 0)
    {x = r; y = i;}

    friend complx operator -(complx a)
    { return complx(-a.x, -a.y);}

    friend complx operator *(complx a, complx b)
    { return complx(a.x * b.x - a.y * b.y, a.x * b.y + a.y * b.x);}

    friend complx operator *(complx a, double b)
    { return complx(a.x * b, a.y * b);}

    friend complx operator *(double b, complx a)
    { return complx(a.x * b, a.y * b);}

    friend complx operator +(complx a, double b)
    { return complx(a.x + b, a.y);}

    friend complx operator +(double b, complx a)
    { return complx(a.x + b, a.y);}
```

課題として提供したソースリストと作業の指示書

```

friend   complx   operator +(complx a, complx      b)
{
    return   complx(a.x + b.x, a.y + b.y);}

friend   complx   operator /(complx a, double b)
{
    if(fabs(b) < min_num){
        b = (b > 0.0)?1.:-1.;
        return   complx(a.x*b*max_num, a.y*b*max_num);
    }
    return   complx(a.x / b, a.y / b);
}

friend   complx   operator /(double b, complx      a)
{
    double   c;
    if((fabs(a.x) > max_num) || (fabs(a.y) > max_num)){
        c = max_num2;
        return   complx(b*a.x/c, -b*a.y/c);
    }
    if((fabs(a.x) < min_num) && (fabs(a.y) < min_num)){
        c = max_num2;
        return   complx(b*a.x*c, -b*a.y*c);
    }
    c = a.x*a.x + a.y*a.y;
    return   complx(b*a.x/c, -b*a.y/c);
}

friend   complx   operator /(complx a, complx      b)
{
    double   c;
    if((a.x == 0.0) && (a.y == 0.0))      return   complx(0.0, 0.0);
    if((fabs(b.x) > max_num) || (fabs(b.y) > max_num)){
        c = max_num2;
        return   complx((b.x/c*a.x+b.y/c*a.y), (b.x/c*a.y-b.y/c*a.x));
    }
}

```


課題として提供したソースリストと作業の指示書

```

        if((fabs(b.x) < min_num) && (fabs(b.y) < min_num)){
            c = max_num2;
            return complx((b.x*c*a.x+b.y*c*a.y), (b.x*c*a.y-b.y*c*a.x));
        }
        c = b.x*b.x + b.y*b.y;
        return complx((a.x * b.x + a.y * b.y)/c, (a.y * b.x - a.x * b.y)/c);
    }

friend complx operator -(complx a, double b)
{
    return complx(a.x - b, a.y);}

friend complx operator -(double b, complx a)
{
    return complx(b - a.x, -a.y);}

friend complx operator -(complx a, complx b)
{
    return complx(a.x - b.x, a.y - b.y);}

friend double real(complx a)
{
    return a.x; }

friend double imag(complx a)
{
    return a.y; }

friend double abs(complx a)
{
    if((fabs(a.x) < min_num) && (fabs(a.y) < min_num)) return 0.0;
    return sqrt(a.x * a.x + a.y * a.y);
}

friend double arg(complx a)
{
    return atan2(a.y, a.x); }

friend complx exp(complx a)
{
    double sc;
    sc = a.x;

```

課題として提供したソースリストと作業の指示書

```

        if(sc > topexp)    return          complx(max_num * cos(a.y), max_num
* sin(a.y));

        if(sc < -topexp)  return  complx(0., 0.);
        sc = exp(sc);
        return  complx(sc*cos(a.y), sc*sin(a.y));
    }

friend  complx  log(complx      a)
{
    double  ud;
        if((fabs(a.x) < min_num) && (fabs(a.y) < min_num)){
            return  complx(min_num, 0.0);
        }
        if((fabs(a.x) > max_num) || (fabs(a.y) > max_num))    ud = max_num2;
        else    ud = a.x * a.x + a.y * a.y;
        return  complx((log(ud))/2, atan2(a.y, a.x));
    }

friend  complx  pow(complx      a, complx      b)
{
    return  exp(b * log(a));  }

friend  complx  sqrt(complx      a)
{
    complx  c, d;
        if((fabs(a.x) < min_num) && (fabs(a.y) < min_num))    return
complx(0.0, 0.0);
        c = log(a)/2.;    d = exp(c);
        return  complx(d.x, d.y);
    }

};

```

課題として提供したソースリストと作業の指示書

NODES.h

```
#include <string.h>
#include "complex.h"

class Node
{
protected:
    Node() {}
    void error();
public:
    virtual ~Node() {}
    virtual complx eval() {error(); return 0;}
};

class Binop : public Node
{
protected:
    Node *left, *right;
    ~Binop() {delete left; delete right;}
    Binop(Node *l, Node *r) {left = l; right = r;}
};

class Plus : public Binop
{
public:
    Plus(Node *l, Node *r) : Binop(l,r){}
    complx eval()
    {
        complx l = left->eval();
        return l + right->eval();
    }
};

class Minus : public Binop
{

```

課題として提供したソースリストと作業の指示書

```

public:
    Minus(Node *l, Node *r) : Binop(l,r){}
    complx eval()
    {
        complx l = left->eval();
        return l - right->eval();
    }
};

class Times : public Binop
{
public:
    Times(Node *l, Node *r) : Binop(l,r){}
    complx eval()
    {
        complx l = left->eval();
        return l * right->eval();
    }
};

class Div : public Binop
{
public:
    Div(Node*l, Node *r) : Binop(l,r){}
    complx eval()
    {
        complx l = left->eval();
        return l / right->eval();
    }
};

class Uminus : public Node
{
    Node *operand;
public:
    Uminus(Node *o) {operand = o;}

```

課題として提供したソースリストと作業の指示書

```

        ~Uminus()      {delete operand;}
        complx eval()  {return -operand->eval();}
};

class Num      :      public Node
{
    complx value;
public:
        Num(complx v)      {value = v;}
        complx eval()      {return value;}
};

class Id;

class Nament
{
    char *name;
    complx value;
    Nament *next;
    Nament(char *nm, Nament *n){
        name = strcpy(new char[strlen(nm)+1], nm);
        value = 0;
        next = n;}
    friend Id;
};

class Id      :      public Node
{
    //static Nament *syntab;
    Nament *entry;
    Nament *look(char *);
public:
        Id(char *nm)      {entry = look(nm);}
        complx set(complx i)      {return entry->value = i;}
        complx eval()      {return entry->value;}
};

```

課題として提供したソースリストと作業の指示書

```
class   Assign   :           public   Binop
{
public:
    Assign(Id*t, Node *e)       :           Binop(t, e)           {}
    complx eval()
    {       return   ((Id       *)left)->set(right->eval());}
};
```

課題として提供したソースリストと作業の指示書

NODES.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include "nodes.h"

void    Node::error()
{
    printf("ERROR Node!!!\n");
    exit(254);
}

Nament *Id::look(char    *nm)
{
    static    Nament *symtab;
    if(symtab->name == NULL){
        symtab = new    Nament("i",symtab);
        symtab->value = complx(0, 1.0);
        symtab = new    Nament("p",symtab);
        symtab->value = complx(1e-12, 0);
        symtab = new    Nament("n",symtab);
        symtab->value = complx(1e-9, 0);
        symtab = new    Nament("u",symtab);
        symtab->value = complx(1e-6, 0);
        symtab = new    Nament("m",symtab);
        symtab->value = complx(1e-3, 0);
        symtab = new    Nament("k",symtab);
        symtab->value = complx(1e3, 0);
        symtab = new    Nament("meg",symtab);
        symtab->value = complx(1e6, 0);
        symtab = new    Nament("g",symtab);
        symtab->value = complx(1e9, 0);
        symtab = new    Nament("t",symtab);
        symtab->value = complx(1e12, 0);
    }
    for(Nament    *p = symtab; p; p = p->next)
```

課題として提供したソースリストと作業の指示書

```
        if(strcmp(p->name, nm) == 0)      return  p;
return  symtab = new Nament(nm, symtab);
}
```


課題として提供したソースリストと作業の指示書

DENTAKU.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "Nodes.h"

//複素数電卓に改良

static int token; //現在
のトークン
static char lexeme[81]; // I DとNUMに対する属性
enum {ID = 257, NUM, EOLN, BAD}; //Idの処理のために変更

char* add_num(char*&, int&);
char* next_char(char*&, int&);

Node *E(), *T(), *F();
static Num ONE(1.0); // Idの処理のために追加
static Node *One = &ONE;

char* add_num(char* &s, int& c)
{
    do
        *s++ = c;
    while(isdigit(c = tolower(getchar())));
    return s;
}

char* next_char(char* &s, int& c)
{
    *s++ = c;
    c = tolower(getchar());
    return s;
}
```

課題として提供したソースリストと作業の指示書

```

/*
    S c a n n e r : 入力ストリーム中の次のトークンを返す。
                    整数定数及び識別子に対する属性を、
                    大域変数 'lexeme'に渡す。

*/
int    scan()
{
    int    c;
    char    *s = lexeme;
    while(1){ //小文字に変換してからトークンに分類する
        switch(c = tolower(getchar())){
            case    '+'      :           case    '-'      :
            case    '*'      :           case    '/'      :
            case    '('      :           case    ')'      :
            case    '='      :
                return    c;
            case    ' '      :           case    '\t'      :
                return    c;           //continue;           --> Idの処理のために変更
            case    '\n'      :
                return    EOLN;
            default :
                if(isdigit(c)){
                    add_num(s, c);
                    switch(c){
                        case    '.' :
                            next_char(s, c);
                            if(isdigit(c))
                                add_num(s, c);
                            else    if(c == '.')
                                if(c != 'e')    break;
                        case    'e' :
                            next_char(s, c);
                            if(c == '-')

```

課題として提供したソースリストと作業の指示書

```

next_char(s, c);

return BAD;

if((c == '-') || (c == 'e'))

add_num(s, c);
break;

}
*s = '\0';
ungetc(c, stdin);
return NUM;
}
if(isalpha(c)){
do
*s++ = c;
while(isalnum(c = tolower(getchar())));
*s = '\0';
ungetc(c, stdin);
return ID;
}
return BAD;
}
}

/*
エラーが起きたら、単純にあきらめる。
*/

void error()
{
printf("ERROR!!!\n");
exit(255);
}

/*
簡単な数式文法の予測的構文解析系：
E(Expression:式),T(Term:項),F(Factor:因子)

```

課題として提供したソースリストと作業の指示書

```

E  --> T { (+ | -) T }
T  --> F { (* | /) F }
//      F  --> ID | NUM | (E) | ID=E | -F
//      以下の様に拡張する
F  --> ID | NUM | (E) | GVAR=E | -F |
        ID F | F ID | FUNC (E)
ID  -> GVAR | UNIT | FUNC
UNIT -> I | p | n | u | m | k | m e g | g | t
FUNC -> s i n | c o s | t a n | a s i n |
        a c o s | a t a n | s q r t | e x p | l o g | l o g 1
0 |
        s i n h | c o s h | t a n h | a s i n h | a c o s h |
        a t a n h | d b | u d b | u n p w | ~ | a b s | a r g
|
        r e a l | i m a g e | . . . .
GVAR -> NAME & !UNIT & !FUNC
NAME  -> ALPHA {ALPHA | NUM}

*/

Node  *E()
{
Node  *root = T();
while(1){
    switch(token){
        case  '+'      :
            token = scan();
            root = new      Plus(root, T());
            break;
        case  '-'      :
            token = scan();
            root = new      Minus(root, T());
            break;
        default  :
            return  root;
    }
}

```

課題として提供したソースリストと作業の指示書

```

    }
}

Node *F()
{
    Node *root;
    switch(token){
        case ID:
            root = new Id(lexeme);
            token = scan();
            switch(token){ // Idの処理のために変更
                case '+' : case '-' :
                case '*' : case '/' :
                case '%n' :
                    //ungetc(token, stdin);
                    root = new Times(root, One); break;
                case '=' :
                    token = scan();
                    root = new Assign((Id *)root, E());
                default : break;
            }
            return root;
        case NUM:
            root = new Num(atof(lexeme));
            token = scan();
            return root;
        case '(':
            token = scan();
            root = E();
            if(token != ')') error();
            token = scan();
            return root;
        case '-':
            token = scan();
            return new Uminus(F());
        default :
    }
}

```

課題として提供したソースリストと作業の指示書

```

        error();
    }
}

Node *T()
{
Node *root = F();
while(1){
    switch(token){
        case '*':           case '':           case
        't' : //Idの処理のために変更
                token = scan();
                root = new Times(root, F());
                break;
        case '/':           :
                token = scan();
                root = new Div(root, F());
                break;
        default :
                return root;
    }
}
}

/*
    Driver : 初期化を行い、エラーがない間は繰り返す。
*/

void main()
{
Node *root;
while(1){
    token = scan();
    if( (root = E()) && token == EOLN){
        double r, i;
        r = real(root->eval());
    }
}
}

```

課題として提供したソースリストと作業の指示書

```
        i = imag(root->eval());
        if(r*i != 0.0)      printf("%lf I %lf\n", r, i);
        else if(i == 0.0)   printf("%lf\n", r);
        else                printf("%lf\n", i);
        delete    root;
    }
    else {
        error();
    }
}
}
```

完成したプログラムのソースリスト

完成したプログラムのソースリスト

```

DT_.h

// DT_.h : DT_ アプリケーションのメイン ヘッダー ファイルです
#ifdef _AFXWIN_H_
    #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"           // メイン シンボル

////////////////////////////////////

// CDT_App:
// このクラスの動作の定義に関しては DT_.cpp ファイルを参照してください。
//

class CDT_App : public CWinApp
{
public:
    CDT_App();

// オーバーライド
    // ClassWizard によって生成された仮想関数がオーバーライドします
    //{{AFX_VIRTUAL(CDT_App)
public:
    virtual BOOL InitInstance();
    //}}AFX_VIRTUAL

// インプリメンテーション

    //{{AFX_MSG(CDT_App)
        // NOTE - ClassWizard はこの位置にメンバ関数を追加または削除し
ます。
        //      この位置に生成されるコードを編集しないでください。
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()

```


完成したプログラムのソースリスト

```
};
```

```
////////////////////////////////////
```

完成したプログラムのソースリスト

Data.h

```

/*=====
各種データ
=====*/

const  NAME_LEN = 10, INFO_LEN = 513;
const  PARAM1 = 0, PARAM2 = 1, TABLE_END = -1;

struct  UNIT_TABLE{
    char    name[ NAME_LEN ];
    char    info[ INFO_LEN ];
    complex value;
};

struct  FUNC_TABLE1{
    char    name[ NAME_LEN ];
    complx  (*pFunc)( complx );
    char    info[ INFO_LEN ];
};

struct  FUNC_TABLE2{
    char    name[ NAME_LEN ];
    complx  (*pFunc)( complx, complx );
    char    info[ INFO_LEN ];
};

/*=====
入力用文字列 ( 記号、数字ボタン )
=====*/

#ifdef  _INIT
int  KEY_CHAR[] = {
    '(', ')', '+', '-', '*', '/', '=', ' ',
    '0', '1', '2', '3',    '4', '5', '6', '7', '8', '9',
    '!', '$'

```

完成したプログラムのソースリスト

```

};
#else
extern  KEY_CHAR[];
#endif

/*=====
                        定数の管理
=====*/

#ifdef  _INIT
UNIT_TABLE  UnitTable[] = {
    {"i",   "単位定数 i (虚数単位) ", {0,      1}},
    {"p",   "単位定数 p (ピコ) ",      {1e-12, 0}},
    {"n",   "単位定数 n (ナノ) ",      {1e-9,   0}},
    {"u",   "単位定数 u (マイクロ) ",  {1e-6,   0}},
    {"m",   "単位定数 m (ミリ) ",      {1e-3,   0}},
    {"k",   "単位定数 k (キロ) ",      {1e3,    0}},
    {"meg", "単位定数 meg (メガ) ",    {1e6,    0}},
    {"g",   "単位定数 g (ギガ) ",      {1e9,    0}},
    {"t",   "単位定数 t (テラ) ",      {1e12,   0}},
    {"pi",  "数学定数 pi (円周率) ", {M_PI,   0}},
    {"e",   "数学定数 e (常用対数の底) ", {M_E,    0}},
    {"",    ""},
};
#else
extern  UNIT_TABLE  UnitTable[];
#endif

/*=====
                        関数の管理
=====*/

/*-----
                        関数テーブル (引き数 1 個)
-----*/

```

完成したプログラムのソースリスト

```

#ifdef _INIT
FUNC_TABLE1 FuncTable1[] = {
    {"abs", abs,      "a の絶対値を返します。"},
    {"arg", arg,      "a の偏角を返します。"},

    {"asin", asin,    "a の逆正弦関数値を返します。"},
    {"acos", acos,    "a の逆余弦関数値を返します。"},
    {"atan", atan,    "a の逆正接関数値を返します。"},
    {"asinh",        asinh, "a の逆双曲正弦関数値を返します。"},
    {"acosh",        acosh, "a の逆双曲余弦関数値を返します。"},
    {"atanh",        atanh, "a の逆双曲正接関数値を返します。"},

    {"badr", badr,    "偏差 a に対する不良率(%)を返します。"},
    {"ceil", ceil,    "小数点以下を切り上げます。"},
    {"conj", conj,    "a の共役複素数を返します。"},
    {"db",          db,      "倍率値 a のデシベル値を返します。"},
    {"defl", defl,    "不良率 a (%)に対する偏差を返します。"},
    {"exp", exp,      "e の a 乗 を返します。"},

    {"image",        imag,   "a の虚数部を返します。"},
    {"int", _int,     "小数点以下を切り捨てます。"},
    {"ln",          log,     "a の自然対数値を返します。"},
    {"log", log,      "a の自然対数値を返します。"},
    {"log10",        log10,   "a の常用対数値を返します。"},
    {"norm",         norm,    "a のノルムを返します。"},
    {"norm2",        norm2,   "a の 2 乗ノルムを返します。"},
    {"real", real,    "a の実数部を返します。"},
    {"recip", recip,   "a の逆数を出力します。"},

    {"sin", sin,      "a の正弦関数値を返します。"},
    {"cos", cos,      "a の余弦関数値を返します。"},
    {"tan", tan,      "a の正接関数値を返します。"},
    {"sinh", sinh,    "a の双曲正弦関数値を返します。"},
    {"cosh", cosh,    "a の双曲余弦関数値を返します。"},
    {"tanh", tanh,    "a の双曲正接関数値を返します。"},

```

完成したプログラムのソースリスト

```

        { "sqrt", sqrt,      "a の平方根を返します。" },
        { "swap",  swap,    "a = x + i*y から、y + i*x を出力します。" },
        { "cswap", cswap,   "a = x + i*y から、y - i*x を出力します。" },
        { "swapc", swapc,   "a = x + i*y から、-y + i*x を出力します。" },
        { "undb",  undb,    "デシベル値 a の倍率値を返します。" },
        { "" }

};

/*-----
関数テーブル (引き数 2 個)
-----*/
FUNC_TABLE2 FuncTable2[] = {
    { "build",      build,  "a + b*i を返す。" },
    { "dbfrac",     dbfrac, "複素数 a / b のデシベル値を返します。" },
    { "estn",       estn,   "不良率 a (%), データ数 b から飽和データ数を返す。" },
    { "pow",        pow,    "a の b 乗を計算します。" },
    { "rtpow",      rtpow,  "rtpow(a, b) は a が 何 の b 乗かを返しま
す。" },
    { "unpow",      unpow,  "unpow(a, b) は a が b の 何乗かを返しま
す。" },
    { "" }

};

#else

extern FUNC_TABLE1  FuncTable1[];
extern FUNC_TABLE2  FuncTable2[];

#endif

/*=====
セーブ・ロード時のファイル拡張子
=====*/
#define MACRO_FILE  ".UMF"
#define GVAR_FILE   ".UVF"
#ifdef _INIT

```

完成したプログラムのソースリスト

```

        char    FileFilter1[] = {"マクロデータ(*.UMF) | *.UMF| |"}; //    User    Macro
Format
        char    FileFilter2[] = {"変数データ(*.UWF) | *.UWF| |"};          //    User
Valuable Format
#else
        extern char    FileFilter1[];
        extern char    FileFilter2[];
#endif

/*=====
        入力用文字列（記号、数字ボタン）
=====*/

enum    {
        SYNTAX_ERROR,
        ASSIGN_ERROR,
        KAKKO_ERROR,
        FUNC_ERROR1,
        FUNC_ERROR2,
        MACRO_ERROR1,
        MACRO_ERROR2,
        DIVISION_ERROR
};

#ifdef    _INIT
        int        ErrorNumber;
        CString    ErrorToken;
        CString    ErrorMessage[] = {
                "式の構文エラー",
                "定数には値を代入出来ません。",
                "'')'がありません。",
                "関数のパラメータが異常です。",
                "関数名の後には'('が必要です。",
                "マクロが見つかりません。",
                "マクロが再帰しています。",
                "0 で除算しました。"
        };
#else
        extern    int        ErrorNumber;

```

完成したプログラムのソースリスト

```
extern  CString ErrorToken;  
extern  CString ErrorMessage[];  
#endif
```

完成したプログラムのソースリスト

DT_DLG.h

```
// DT_dlg.h : ヘッダー ファイル
//

const    BT_ID_TOP = IDC_BUTTON4;        //記号、数字入力ボタンの先頭ID
const    BT_ID_END = IDC_BUTTON23;       //記号、数字入力ボタンの終了ID
const    DEFAULT_FLOAT = 6;

/*****
[ クラス名 ] CDT_Dlg
[基底クラス] CDialog ← CWnd ← CCmdTarget ← CObject
[ 機能 ] メインのダイアログ・各ボタンの入力チェック
[ メンバ ] m_Com          ・ 計算を行う CCompute オブジェクト
              m_EditBox1    ・ 式入力エディットボックス( DDX )
              m_EditBox2    ・ 計算結果表示エディットボックス( DDX )
*****/
class CDT_Dlg : public CDialog
{
// 構築
public:
    CDT_Dlg(CWnd* pParent = NULL); // 標準のコンストラクタ
    CDT_Dlg::~CDT_Dlg();

private:
    CCompute      m_Com;
    int           m_Float;
    int           m_CommaMode;

    char          buff[ 512 ];        // 小数→文字 変換用

protected:
    void          ClearEditBox( int BoxNum = 0 );

    BOOL          InsertComma( double f, CString* Buff );
    CString       InsertCommaSub( char* pNumStr, int NumLen );

```


完成したプログラムのソースリスト

```
void    AssignError();
void    DefForcus();

public:
// ダイアログ データ
//{{AFX_DATA(CDT_Dlg)
enum { IDD = IDD_DT_DIALOG };
CString m_EditBox1;
CString m_EditBox2;
//}}AFX_DATA

// ClassWizard によって生成された仮想関数がオーバーライドします
//{{AFX_VIRTUAL(CDT_Dlg)
public:
virtual void WinHelp(DWORD dwData, UINT nCmd = HELP_CONTEXT);
protected:
virtual void DoDataExchange(CDataExchange* pDX);  // DDX/DDV のサポート
virtual BOOL OnCommand(WPARAM wParam, LPARAM lParam);
//}}AFX_VIRTUAL

// インプリメンテーション
HICON m_hIcon;

// 生成されたメッセージ マップ関数
//{{AFX_MSG(CDT_Dlg)
virtual BOOL OnInitDialog();
afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
afx_msg void OnDestroy();
afx_msg void OnPaint();
afx_msg HCURSOR OnQueryDragIcon();
afx_msg void OnB01Gvar();
afx_msg void OnB02Func();
afx_msg void OnB03Option();
afx_msg void OnBt27Enter();
afx_msg void OnBt24Ac();
```

完成したプログラムのソースリスト

```
afx_msg void OnBt25C();  
afx_msg void OnBt26Bs();  
afx_msg void OnMacro();  
//}}AFX_MSG  
DECLARE_MESSAGE_MAP()  
};
```

完成したプログラムのソースリスト

Complx.h

```

/*-----
COMPLEX.H
-----*/
#include<math.h>

#define topexp      150
#define min_num      1.e-150
#define min_num2     1.e-300
#define max_num      1.e150
#define max_num2     1.e300
#define M_E          2.71828182845904523536
#define M_PI          3.14159265358979323846
#define M_2_PI        6.28318530717958647693
#define M_PI_2        1.57079632679489661923
#define M_PI_4        0.785398163397448309616

#ifdef _RAD_INIT
    int    rad = 0;
#else
    extern int rad;
#endif

// 追加 2000/06/09
struct complex{
    double  x, y;
};

class    complx
{
private:
    double  x, y;
public:
    //// コンストラクタ
    complx( double  r = 0, double  i = 0 )  {x = r;  y = i;}

```

完成したプログラムのソースリスト

```

    complx( complex a )      {x = a.x; y = a.y;}

///// 実数、虚数部に分ける
double  Real()  { return x; }
double  Image() { return y; }
// double  Fnc(double x)    { return exp(-x*x/2); }

/////  +-×÷
friend  complx  operator-(complx a)
{
    return complx( -a.x, -a.y ); }

friend  complx  operator*(complx      a, complx      b)
{
    return  complx(a.x * b.x - a.y * b.y, a.x * b.y + a.y * b.x);}

friend  complx  operator*(complx      a, double      b)
{
    return  complx(a.x * b, a.y * b);}

friend  complx  operator*(double b, complx      a)
{
    return  complx(a.x * b, a.y * b);}

friend  complx  operator+(complx      a, double      b)
{
    return  complx(a.x + b, a.y);}

friend  complx  operator+(double b, complx      a)
{
    return  complx(a.x + b, a.y);}

friend  complx  operator+(complx      a, complx      b)
{
    return  complx(a.x + b.x, a.y + b.y);}

friend  complx  operator/(complx a, double      b)
{
    if(fabs(b) < min_num){
        b = (b > 0.0)?1.:-1.;
        return  complx(a.x*b*max_num, a.y*b*max_num);
    }
}

```

完成したプログラムのソースリスト

```

        return  complx(a.x / b, a.y / b);
    }

friend  complx  operator/(double b, complx      a)
{
    double  c, d, e;
    d = (a.x > 0)?a.x:-a.x;      e = (a.y > 0)?a.y:-a.y;
    if((d > max_num) || (e > max_num)){
        c = min_num2;
        return  complx(b*a.x*c, -b*a.y*c);
    }
    if((d < min_num) && (e < min_num)){
        c = max_num2;
        return  complx(b*a.x*c, -b*a.y*c);
    }
    c = a.x*a.x + a.y*a.y;
    return  complx(b*a.x/c, -b*a.y/c);
}

friend  complx  operator/(complx a, complx      b)
{
    double  c, e, f;
    complx  d;
    if((a.x == 0.0) && (a.y == 0.0))      return  complx(0.0, 0.0);
    if((b.x == 0.0) && (b.y == 0.0)){
        c = max_num2;
        d.x = ((a.x > 0.0)?c:-c);
        d.y = ((a.y > 0.0)?c:-c);
        return  d;
    }
    e = (b.x > 0)?b.x:-b.x;      f = (b.y > 0)?b.y:-b.y;
    if((e > max_num) || (f > max_num)){
        c = min_num2;
        e = (b.x > 0)?c:-c; f = (b.y > 0)?c:-c;
        d.x = e * a.x + f * a.y;
        d.y = e * a.y - f * a.x;
    }
}

```

完成したプログラムのソースリスト

```

        return d;
    }
    if((e < min_num) && (f < min_num)){
        c = max_num2;
        e = (b.x > 0)?c:-c; f = (b.y > 0)?c:-c;
        d.x = e * a.x + f * a.y;
        d.y = e * a.y - f * a.x;
        return d;
    }
    c = b.x*b.x + b.y*b.y;
    d.x = (a.x * b.x + a.y * b.y)/c;
    d.y = (a.y * b.x - a.x * b.y)/c;
    return d;
}

friend  complx  operator-(complx a, double      b)
{
    return  complx(a.x - b, a.y);}

friend  complx  operator-(double b, complx      a)
{
    return  complx(b - a.x, -a.y);}

friend  complx  operator-(complx a, complx      b)
{
    return  complx(a.x - b.x, a.y - b.y);}

///// 関数
friend  complx  real( complx a )
{
    complx buff( a.x, 0 );      return  buff;    }

friend  complx  imag( complx a )
{
    complx buff( a.y, 0 );      return  buff;    }

friend  complx  abs(complx      a)
{
    if((fabs(a.x) < min_num) && (fabs(a.y) < min_num))return
0.0;

```

完成したプログラムのソースリスト

```

        complx buff( sqrt(a.x * a.x + a.y * a.y), 0 );
        return  buff;
    }

friend  complx  norm(complx  a)
{
    if((fabs(a.x) < min_num) && (fabs(a.y) < min_num))return
0.0;

    complx buff( sqrt(a.x * a.x + a.y * a.y), 0 );
    return  buff;
}

friend  complx  norm2(complx  a)
{
    if((fabs(a.x) < min_num) && (fabs(a.y) < min_num))return
0.0;

    complx buff( a.x * a.x + a.y * a.y, 0 );
    return  buff;
}

friend  complx  db(complx  a)
{
    complx buff( sqrt(a.x * a.x + a.y * a.y), 0 );
    return  log10(buff) * 20.;
}

friend  complx  undb(complx  a)
{
    return  pow(10., a.x / 20);
}

friend  complx  unpow(complx  a, complx  b)
{
    return  log(a)/log(b);
}

```

完成したプログラムのソースリスト

```

friend   complx   rtpow(complx   a, complx       b)
{
    return   pow(a, 1/b);
}

friend   complx   conj(complx   a)
{
    a.y = -a.y;
    return   a;
}

friend   complx   arg(complx   a)
{
    if(rad)   return complx( atan2(a.y, a.x), 0 );
    else      return complx( atan2(a.y, a.x) * 180 / M_PI, 0 );
}

friend   complx   exp(complx   a)
{
    double   sc;
    sc = a.x;
    if(sc > topepx)   return          complx(max_num *
cos(a.y), max_num * sin(a.y));
    if(sc < -topepx)   return   complx(0., 0.);
    sc = exp(sc);
    return   complx(sc*cos(a.y), sc*sin(a.y));
}

friend   complx   log(complx   a)
{
    double   ud;
    if((fabs(a.x) < min_num) && (fabs(a.y) < min_num)){
        return   complx(min_num, 0.0);
    }
    if((fabs(a.x) > max_num) || (fabs(a.y) > max_num)) ud =
max_num2;

```


完成したプログラムのソースリスト

```

        else    ud = a.x * a.x + a.y * a.y;
        return  complx((log(ud))/2, atan2(a.y, a.x));
    }

friend  complx  log10(complx    a)
{
    complx  c;
    double  ud, ud2;
    if((fabs(a.x) < min_num) && (fabs(a.y) < min_num)){
        return  complx(min_num, 0.0);
    }
    if((fabs(a.x) > max_num) || (fabs(a.y) > max_num)) ud =
max_num2;

    else    ud = a.x * a.x + a.y * a.y;
    ud2 = 10.0;
    ud = log(ud2);
    c = log(a)/ud;
    return  c;
}

friend  complx  sin(complx    a)
{
    double  x, y;
    complx  d;
    if(rad)  x = a.x;
    else    x = a.x / 180. * M_PI;
    y = a.y;
    d.x = sin(x) * cosh(y);
    d.y = cos(x) * sinh(y);
    return  d;
}

friend  complx  cos(complx    a)
{
    double  x, y;
    complx  d;

```

完成したプログラムのソースリスト

```

        if(rad)  x = a.x;
        else    x = a.x / 180. * M_PI;
        y = a.y;
        d.x = cos(x) * cosh(y);
        d.y = -sin(x) * sinh(y);
        return  d;
    }

friend  complx  tan(complx      a)
{
    return  sin(a) / cos(a);
}

friend  complx  sinh(complx      a)
{
    double  x, y;
    complx  d;
    x = a.x;
    y = a.y;
    d.x = cos(y) * sinh(x);
    d.y = sin(y) * cosh(x);
    return  d;
}

friend  complx  cosh(complx      a)
{
    double  x, y;
    complx  d;
    x = a.x ;
    y = a.y ;
    d.x = cos(y) * cosh(x);
    d.y = sin(y) * sinh(x);
    return  d;
}

friend  complx  tanh(complx      a)

```

完成したプログラムのソースリスト

```

    {
        return  sinh(a) / cosh(a);
    }

friend  complx  asinh(complx  a)
{
    return  log(a + sqrt(a * a + 1));
}

friend  complx  acosh(complx  a)
{
    return  log(a + sqrt(a * a - 1));
}

friend  complx  atanh(complx  a)
{
    return  log((1+a)/(1-a)) / 2;
}

friend  complx  asin(complx  a)
{
    complx  c, d, e(0,1);
    c.x = -a.y;
    c.y =  a.x;
    d = - asinh(c) * e;
    while(d.x > M_PI_2)      d.x -= M_2_PI;
    while(d.x < -M_PI_2)     d.x += M_2_PI;
    if(d.x > M_PI_2) d.x = M_PI - d.x;
    if(!rad) d.x *= 180. / M_PI;
    return  d;
}

friend  complx  acos(complx  a)
{
    complx  d, e(0,1);
    d = acosh(a) * e;

```

完成したプログラムのソースリスト

```

        while(d.x > M_PI)      d.x -= M_2_PI;
        while(d.x < 0)      d.x += M_2_PI;
        if(d.x > M_PI)      d.x = M_2_PI - d.x;
        if(!rad)  d.x *= 180. / M_PI;
        return  d;
    }

friend  complx  atan(complx      a)
{
    complx  c, d, e(0,1);
    c.x = -a.y;
    c.y =  a.x;
    d = - atanh(c) * e;
    while(d.x > M_PI_2)      d.x -= M_2_PI;
    while(d.x < -M_PI_2)      d.x += M_2_PI;
    if(d.x > M_PI_2)  d.x = d.x - M_PI;
    if(!rad)  d.x *= 180. / M_PI;
    return  d;
}

friend  complx  pow(complx      a, complx      b )
{
    return  exp(b * log(a));  }

friend  complx  sqrt(complx      a)
{
    complx  c, d;
    if((fabs(a.x) < min_num) && (fabs(a.y) < min_num))return
complx(0.0, 0.0);
    c = log(a)/2.;      d = exp(c);
    return  complx(d.x, d.y);
}

friend  complx  _int(complx      a)
{
    a.x = floor( a.x );
    a.y = floor( a.y );

```

完成したプログラムのソースリスト

```

        return a;
    }

friend  complx  ceil(complx      a)
{
    a.x = ceil( a.x );
    a.y = ceil( a.y );
    return a;
}

friend  complx  badr(complx      a)
{
double  x = a.x, h, fx = 0, X, FF;
int      i, n = 1000;
    h = x / (double)n; X = -h;
    for(i = 0; i <= n; i++){
        X += h;  FF = exp(-X*X/2.);
        if((i == 0) || (i == n)){
            fx += FF;          continue;
        }
        if(i & 1) fx += 4. * FF;
        else      fx += 2. * FF;
    }
    fx *= (h / 3.) / sqrt(2. * (double)M_PI);
    fx = (1. - 2. * fx) * 100.;
    a.x = fx;
    a.y = 0;
    return a;
}

friend  complx  defl(complx      a)
{
double  p = a.x, x1 = 0, x2 = 4.0, x3, p1;
int      i;
complx  er;
    for(i = 0; i < 20; i++){

```

完成したプログラムのソースリスト

```

        x3 = (x1 + x2) / 2.;
        er = badr(complx(x3, 0)); p1 = er.x;
        if(p1 > p)          x1 = x3;
        else                x2 = x3;
    }
    a.x = x1;
    a.y = 0;
    return a;
}

friend complx estn(complx a, complx b)
{
    complx c = defl(a);
    double r = 4. / c.x;
    a.x = r * b.x;
    a.y = 0;
    return a;
}

friend complx build(complx x, complx y)
{
    complx c;
    c.x = x.x; c.y = y.x;
    return c;
}

friend complx recip(complx a)
{
    double c, d, e;
    if(a.x == 0 && a.y == 0){e=0; d=1/e;return d;}
    d = (a.x >= 0)?a.x:-a.x;    e = (a.y >= 0)?a.y:-a.y;
    if((d > max_num) || (e > max_num)){
        c = min_num2;
        return complx(c, -c);
    }
    if((d < min_num) && (e < min_num)){

```

完成したプログラムのソースリスト

```

        c = max_num2;
        return  complx(c, -c);
    }
    return  1. / a;
}

friend  complx  swap(complx  x )
{
    complx  c;
        c.x = x.y; c.y = x.x;
    return  c;
}

friend  complx  cswap(complx  x )
{
    complx  c;
        c.x = x.y; c.y = -x.x;
    return  c;
}

friend  complx  swape(complx  x )
{
    complx  c;
        c.x = -x.y; c.y = x.x;
    return  c;
}

friend  complx  dbfrac(complx  x, complx      y )
{
    double  d, e;
        if(y.x == 0 && y.y == 0){e=0; d=1/e;return d;}
    return  db(x)-db(y);
}

};

```

完成したプログラムのソースリスト

Compute.h

```

/*//////////////////////////////////////////////////////////////
    CCompute クラスの定義
//////////////////////////////////////////////////////////////*/

const    TOKEN_MAXLEN = 10;    //トークン文字列の最大長
const    PARAM_MAX = 2;        // 関数の引数の最大数

/*****
[ クラス名 ] CCompute
[基底クラス] なし
[ 機能 ] 式の文字列を受け取り計算を行う。
[ メンバ ] GetUserMacro ・マクロの管理を行う CUserMacroオブジェクト
                のポインタを返す。
                GetChar      ・ pSorceStrから 1 文字読み込む。
                UnGetChar    ・ pSorceStrに 1 文字返す。
*****/
class    CCompute
{
    public:
        CCompute();          ~CCompute();
        BOOL    GoCompute( CString str, complx* Answer );
        CUserMacro*    GetUserMacro(){ return &m_UserMacro; }

    private:
        CString SorceString;    // 計算式文字列
        char*    pSorceStr;      // 計算式文字列のポイン
タ
        CString TokenStr;        // 読みこんだトークンの
格納
        Token    token;          // トークンの識別

        CUserMacro    m_UserMacro;    // マクロのデータ管理

```


完成したプログラムのソースリスト

```

protected:
    char    GetChar()
    {
        if(*pSourceStr)    return (*( pSourceStr++ ));
        else                return
'¥0';
    }

    void    UnGetChar( char c = '¥0' )
    {
        pSourceStr--;
        *pSourceStr = c;
    }

    Token    Scan();
    char    Add_Char( Token c );
    char    Add_Num(  Token c );
    Node*    E();
    Node*    T();
    Node*    F();
    int      IsFunc( CString pIdName, int* token );
    Node*    CallFunc( int  FuncIndex );
    Node*    CallUserMacro( CString name );
    BOOL    MacroLoopCheck( CString MacroName );
};

```

完成したプログラムのソースリスト

FUNCDLG.h

```
// funcdlg.h : ヘッダー ファイル
//

/////////////////////////////////////////////////////////////////
// CFuncDlg ダイアログ

/*****
[ クラス名 ] CDT_Dlg
[ 基底クラス ] CDialog <- CWnd <- CCmdTarget <- CObject
[ 機能 ] メインのダイアログ・各ボタンの入力チェック
[ メンバ ] pBuff          ・ リストボックスの処理用バッファ
           GetFuncName    ・ 選択された関数名を返す
*****/
class CFuncDlg : public CDialog
{
// コンストラクション
public:
    CFuncDlg(CWnd* pParent = NULL);    // 標準のコンストラクタ
    char*   GetFuncName(){ return pBuff; }

private:
    char    pBuff[512];

protected:
    void     CFuncDlg::GetFuncInfo( CString Name, CString* Info );

public:
// ダイアログ データ
    //{AFX_DATA(CFuncDlg)
    enum { IDD = IDD_FUNC };
    CString m_FuncInfo;
    //}AFX_DATA
```

完成したプログラムのソースリスト

```
// オーバーライド
// ClassWizard は仮想関数を生成しオーバーライドします。
//{{AFX_VIRTUAL(CFuncDlg)
protected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV サポート
//}}AFX_VIRTUAL

// インプリメンテーション
protected:

// 生成されたメッセージ マップ関数
//{{AFX_MSG(CFuncDlg)
virtual BOOL OnInitDialog();
afx_msg void OnDbclkFuncList1();
virtual void OnOK();
afx_msg void OnSelchangeFuncList1();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};
```

完成したプログラムのソースリスト

GVARDLG.h

```
// gvardlg.h : ヘッダー ファイル
//

/////////////////////////////////////////////////////////////////
// CGvarDlg ダイアログ

/*****
[ クラス名 ] CGvarDlg
[ 基底クラス ] CDialog <- CWnd <- CCmdTarget <- CObject
[ 機能 ] 変数の一覧表示・選択、値の代入、、、機能の呼び出し
[ メンバ ] pBuff          ・ 作業用バッファ
           GetGvarName    ・ 選択された変数名を返す
*****/
class CGvarDlg : public CDialog
{
// コンストラクション
public:
    CGvarDlg(CWnd* pParent = NULL);    // 標準のコンストラクタ
    char*    GetGvarName(){ return pBuff; }

private:
    char    pBuff[512];

protected:
    BOOL    GetListBoxText( char* pbuff );
    int     OutGvarList();

// ダイアログ データ
    //{{AFX_DATA(CGvarDlg)
    enum { IDD = IDD_GVAR };
    CString m_EditBox_Real;
    CString m_EditBox_Image;
    CString m_Info;
    //}}AFX_DATA
```

完成したプログラムのソースリスト

```
// オーバーライド
// ClassWizard は仮想関数を生成しオーバーライドします。
//{{AFX_VIRTUAL(CGvarDlg)
protected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV サポート
//}}AFX_VIRTUAL

// インプリメンテーション
protected:

// 生成されたメッセージ マップ関数
//{{AFX_MSG(CGvarDlg)
virtual BOOL OnInitDialog();
afx_msg void OnSelchangeGvarList();
afx_msg void OnGvarAssign();
virtual void OnOK();
afx_msg void OnDbclckGvarList();
afx_msg void OnGvarDelete();
afx_msg void OnGvarLoad();
afx_msg void OnGvarSave();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};
```

完成したプログラムのソースリスト

MACEDDLG.h

```
// maceddlg.h : ヘッダー ファイル
//

/////////////////////////////////////////////////////////////////
// CMacEdDlg ダイアログ

/*****
[ クラス名 ] CMacEdDlg
[ 基底クラス ] CDialog <- CWnd <- CCmdTarget <- CObject
[ 機能 ] マクロの編集を行う
[ メンバ ] m_MacroName    ・マクロ名入力エディットボックス
           m_MacroCode    ・マクロコード入力エディットボックス
           m_MacroInfo    ・マクロ機能説明入力エディットボックス
*****/
class CMacEdDlg : public CDialog
{
// コンストラクション
public:
    CMacEdDlg(CWnd* pParent = NULL);    // 標準のコンストラクタ

// ダイアログ データ
   //{{AFX_DATA(CMacEdDlg)
    enum { IDD = IDD_MACRO_EDIT };
    CString m_MacroName;
    CString m_MacroCode;
    CString m_MacroInfo;
    }}AFX_DATA

// オーバーライド
    // ClassWizard は仮想関数を生成しオーバーライドします。
   //{{AFX_VIRTUAL(CMacEdDlg)
protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV サポート
```

完成したプログラムのソースリスト

```
//}}AFX_VIRTUAL

// インプリメンテーション
protected:

    // 生成されたメッセージ マップ関数
   //{{AFX_MSG(CMacEdDlg)
        // NOTE: ClassWizard はこの位置にメンバ関数を追加します。
   //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

完成したプログラムのソースリスト

MACRODLG.h

```
// macrodlg.h : ヘッダー ファイル
//

/////////////////////////////////////////////////////////////////
// CMacroDlg ダイアログ

/*****
[ クラス名 ] CMacEdDlg
[ 基底クラス ] CDialog <- CWnd <- CCmdTarget <- CObject
[ 機能 ] マクロの選択、編集、削除、ロード、セーブ、
          各機能の呼び出し。
[ メンバ ] SetUserMacro ・マクロ管理オブジェクトのセット
          GetMacroName ・選択されたマクロ名の拾得
*****/
class CMacroDlg : public CDialog
{
// コンストラクション
public:
    CMacroDlg(CWnd* pParent = NULL);    // 標準のコンストラクタ
    void SetUserMacro( CUserMacro* p ){ pUserMacro = p; }
    char* GetMacroName(){ return Buff; }

private:
    char Buff[512];                    // リストボックス処理用
    バッファ
    CUserMacro* pUserMacro;           // マクロ管理オブジェク
    トのポインタ

protected:
    int OutMacroList();
    BOOL GetListBoxText( char* pBuff );
    int ReadString( CFile* pFile, CString* pString );

    void MacroSave( CFile* pFile, CString name, CString code, CString info );
```


完成したプログラムのソースリスト

```
public:
// ダイアログ データ
//{{AFX_DATA(CMacroDlg)
enum { IDD = IDD_MACRO };
CString m_MacroInfo;
//}}AFX_DATA

// オーバーライド
// ClassWizard は仮想関数を生成しオーバーライドします。
//{{AFX_VIRTUAL(CMacroDlg)
protected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV サポート
//}}AFX_VIRTUAL

// インプリメンテーション
protected:

// 生成されたメッセージ マップ関数
//{{AFX_MSG(CMacroDlg)
virtual BOOL OnInitDialog();
afx_msg void OnMacroDelete();
afx_msg void OnMacroEdit();
afx_msg void OnDbclckMacroList1();
afx_msg void OnMacroLoad();
afx_msg void OnMacroSave();
afx_msg void OnNewmacroEdit();
virtual void OnOK();
afx_msg void OnSelchangeMacroList1();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};
```

完成したプログラムのソースリスト

NODES.h

```

/*=====
Nodes Header
=====*/

enum    { ID = 257, NUM, EOLN, BAD, UNIT, GVAR, USER, FUNC, FUNC2, FUNC3  };
typedef int    Token;

/*****
[ クラス名 ] Node
[基底クラス] なし
[ 機能 ] 各派生関数の数値拾得関数 eval() の仮想関数を装備
[ メンバ ] Err          ・通常時 = TRUE, エラー時 = FALSE
                      error      ・エラーが起きたかチェック
*****/
class    Node
{
protected:
    BOOL    Err;
public:
    Node() { Err = TRUE; }
    BOOL    error(){ return Err; }
    virtual ~Node() {}
    virtual complx eval() { error(); return 0;}
};

/*****
[ クラス名 ] Binop
[基底クラス] Node
[ 機能 ] 各計算時のバッファ left, right の装備、割り当てられた
          メモリの解放。
[ メンバ ] left          ・左辺部格納用ポインタ
                      right      ・右辺部格納用ポインタ
*****/
class    Binop    :    public    Node
{

```

完成したプログラムのソースリスト

```

        public:
            Binop( Node      *l, Node *r)      {left = l; right = r; }
            ~Binop(){ delete left;      delete right;      }

        protected:
            Node      *left,      *right;
};

/*****
[ クラス名 ] Plus
[基底クラス] Binop ← Node
[ 機能 ] 加算
[ メンバ ] eval()      ・ 計算結果を返す
*****/
class Plus      :      public      Binop
{
    public:
        Plus(Node      *l, Node *r)      :      Binop(l,r){}
        complx eval()
        {
            complx l = left->eval();
            return l + right->eval();
        }
};

/*****
[ クラス名 ] Minus
[基底クラス] Binop ← Node
[ 機能 ] 減算
[ メンバ ] eval()      ・ 計算結果を返す
*****/
class Minus      :      public      Binop
{
    public:
        Minus(Node      *l, Node *r)      :      Binop(l,r){}
        complx eval()
        {

```

完成したプログラムのソースリスト

```

        complx l = left->eval();
        return l - right->eval();
    }
};

/*****
[ クラス名 ] Times
[基底クラス] Binop <- Node
[ 機能 ] 乗算
[ メンバ ] eval() ・ 計算結果を返す
*****/
class Times : public Binop
{
public:
    Times(Node *l, Node *r) : Binop(l,r){}
    complx eval()
    {
        complx l = left->eval();
        return l * right->eval();
    }
};

/*****
[ クラス名 ] Div
[基底クラス] Binop <- Node
[ 機能 ] 除算
[ メンバ ] eval() ・ 計算結果を返す
*****/
class Div : public Binop
{
public:
    Div(Node*l, Node *r) : Binop(l,r)
    {
        if( (r->eval()).Real() == 0 && (r->eval()).Image() == 0 )
            Err = FALSE; // 0で除算した場合
    }
};

```

完成したプログラムのソースリスト

```

        complx eval()
        {
            complx l = left->eval();
            return l / right->eval();
        }
};

/*****
[ クラス名 ] Uminus
[基底クラス] Binop <- Node
[ 機能 ] 符号 (ー)
[ メンバ ] eval() ・ 計算結果を返す
*****/
class Uminus : public Node
{
private:
    Node *operand;
public:
    Uminus(Node *o) {operand = o;}
    ~Uminus() {delete operand;}
    complx eval() {return -operand->eval();}
};

/*****
[ クラス名 ] Num
[基底クラス] Node
[ 機能 ] 数字格納
[ メンバ ] eval() ・ 数値を返す
*****/
class Num : public Node
{
private:
    complx value;
public:
    Num(complx v) {value = v;}
    complx eval() {return value;}
};

```

完成したプログラムのソースリスト

```

};

/*****
[ クラス名 ] Nament
[基底クラス] なし
[ 機能 ] 変数格納用。リストで管理される。
[ メンバ ] name    ・変数名
              value  ・数値
              next   ・リスト構築用
              attrib ・属性 ( GVAR = 変数、UNIT = 定数。代入禁
止 )
*****/
class Id;
class Nament
{
    private:
        CString name;
        complx value;
        CString info;
        Nament *next;
        int      attrib;

        Nament( char *nm, Nament *n, int atr = GVAR, complx val = 0,
char* inf = NULL )
        {
            name = nm;
            value = val;
            attrib = atr;
            next = n;
            if( inf )info = inf; else info = "ユーザー変数";
        }
        ~Nament(){ if( next != NULL )delete next; }

    friend Id;
};

```

完成したプログラムのソースリスト

```

/*****
    変数処理用グローバル変数
*****/
#ifdef  _ID_INIT
    Nament*ID_Symtab = NULL;        // 変数リストの先頭ポインタ
#else
    extern  Nament*ID_Symtab;
#endif

/*****
[ クラス名 ] Id
[基底クラス] Node
[ 機能 ] 変数リストの構築、変数の管理
[ メンバ ] set          ・ 変数に値を代入
              eval       ・ 変数の値を返す。
              GetAttrib   ・ 変数属性の拾得
              entry       ・ 変数格納オブジェクト(Nament)のポインタ
              --- class Nament のメンバを参照可 ---
*****/
class  Id      :      public  Node
{
    public:
        Id(){}
        Id( char *nm ) { entry = look( nm ); }
        complx  set( complx      i )      { return entry->value = i; }
        complx  eval()    { return entry->value; }
        int      GetAttrib(){ return entry->attrib; }
        CString  GetGvarInfo(){ return entry->info; };

        void      DeleteList(){ if( ID_Symtab != NULL ){ delete ID_Symtab;
ID_Symtab = NULL; } }

        BOOL      GetGvarData( int index, CString* name, complx* value, CString*
info = NULL      );
        BOOL      DeleteGvar( CString GvarName );
    private:
        Nament* entry;

```

完成したプログラムのソースリスト

```

        protected:
            Nament *look(char *);
            void    MakeDefault( Nament* &syntab );
};

/*****
[ クラス名 ] Func
[基底クラス] Id ← Node
[ 機能 ] 関数の実行
[ メンバ ] value    ・ 計算結果の格納
              eval    ・ 計算結果を返す。
*****/
class Func : public Id
{
    private:
        complx value;
    public:
        Func( int index, Node* Paran1 );
        Func( int index, Node* Paran1, Node* Param2 );
        complx eval() { return value; }
};

/*****
[ クラス名 ] Assign
[基底クラス] Binop ← Node
[ 機能 ] 変数への値の代入、定数への代入禁止チェック
[ メンバ ] eval    ・ 代入したを返す。
*****/
class Assign : public Binop
{
    public:
        Assign(Id*t, Node *e) : Binop(t, e)
        { if( t->GetAttrib() != GVAR )Err = FALSE; }
        complx eval()
        { return ((Id *)left)->set(right->eval()); }
};

```


完成したプログラムのソースリスト

```
};
```

完成したプログラムのソースリスト

OPTDLG.h

```

// optdlg.h : ヘッダー ファイル
//

const    FLOAT_MAX = 10;
const    FLOAT_MIN = 1;

/////////////////////////////////////////////////////////////////
// COptDlg ダイアログ

class COptDlg : public CDialog
{
// コンストラクション
public:
    COptDlg(CWnd* pParent = NULL);    // 標準のコンストラクタ

    void    GetStatus( int* rad, int* comma, int* f )
    {   *rad = m_RadMode;   *comma = m_CommaMode;   *f = m_Float; }
    void    SetStatus( int* rad, int* comma, int* f )
    {   m_RadMode = *rad;   m_CommaMode = *comma;   m_Float = *f; }
private:
    int      m_RadMode;
    int      m_CommaMode;

protected:

// ダイアログ データ
   //{{AFX_DATA(COptDlg)
    enum { IDD = IDD_OPTION };
    int             m_Float;
    //}}AFX_DATA

```

完成したプログラムのソースリスト

```
// オーバーライド
// ClassWizard は仮想関数を生成しオーバーライドします。
//{{AFX_VIRTUAL(COptDlg)
protected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV サポート
//}}AFX_VIRTUAL

// インプリメンテーション
protected:

// 生成されたメッセージ マップ関数
//{{AFX_MSG(COptDlg)
virtual BOOL OnInitDialog();
virtual void OnOK();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};
```

完成したプログラムのソースリスト

RESOURCE.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by DT_rc
//

#define IDM_ABOUTBOX                0x0010
#define IDD_ABOUTBOX                100
#define IDS_ABOUTBOX                101
#define IDD_DT_DIALOG               102
#define IDR_MAINFRAME               128
#define IDD_GVAR                    129
#define IDD_FUNC                    130
#define IDD_FUNC_EDIT               131
#define IDD_MACRO_EDIT              131
#define IDD_OPTION                  132
#define IDD_MACRO                   133
#define IDC_EDIT1                   1000
#define IDC_EDIT2                   1001
#define IDC_BUTTON1                  1002
#define IDC_GVAR_DELETE              1002
#define IDC_BUTTON2                  1003
#define IDC_GVAR_LOAD                1003
#define IDC_BUTTON3                  1004
#define IDC_GVAR_SAVE                1004
#define IDC_BUTTON4                  1005
#define IDC_BUTTON5                  1006
#define IDC_BUTTON6                  1007
#define IDC_BUTTON7                  1008
#define IDC_BUTTON8                  1009
#define IDC_BUTTON9                  1010
#define IDC_BUTTON10                 1011
#define IDC_BUTTON11                 1012
#define IDC_BUTTON12                 1013
#define IDC_BUTTON13                 1014
#define IDC_BUTTON14                 1015
```

RESOURCE.h

完成したプログラムのソースリスト

```
#define IDC_BUTTON15          1016
#define IDC_BUTTON16          1017
#define IDC_BUTTON17          1018
#define IDC_BUTTON18          1019
#define IDC_BUTTON19          1020
#define IDC_BUTTON20          1021
#define IDC_BUTTON21          1022
#define IDC_BUTTON22          1023
#define IDC_BUTTON23          1024
#define IDC_BUTTON23          1024
#define IDC_BT24_AC           1025
#define IDC_BT25_C            1026
#define IDC_BT25_             1026
#define IDC_BT26_BS           1027
#define IDC_BT27_ENTER        1028
#define IDC_LIST1             1029
#define IDC_GVAR_LIST         1029
#define IDC_RADIO3            1032
#define IDC_OPTION_RADIO1     1032
#define IDC_RADIO4            1033
#define IDC_OPTION_RADIO2     1033
#define IDC_RADIO5            1034
#define IDC_CHECK1            1035
#define IDC_OPTION_CHECK1     1035
#define IDC_B01_GVAR          1036
#define IDC_B02_FUNC          1037
#define IDC_B03_OPTION        1038
#define IDC_FUNC_EDIT         1039
#define IDC_MACRO_EDIT        1039
#define IDC_EDIT3             1040
#define IDC_NEWMACRO_EDIT     1041
#define IDC_MACRO_DELETE      1042
#define IDC_MACRO             1043
#define IDC_MACRO_LOAD        1044
#define IDC_MACRO_SAVE        1045
#define IDC_MACRO_LIST1       1046
```

完成したプログラムのソースリスト

```
#define IDC_FUNC_LIST1          1047
#define IDC_STATIC1            1050
#define IDC_FUNC_INFOTEXT      1051
#define IDC_MACRO_INFO        1052
#define IDC_GVAR_EDIT1        1053
#define IDC_GVAR_EDIT2        1054
#define IDC_GVAR_ASSIGN        1055
#define IDC_OPTION_EDIT1       1056
#define IDC_GVAR_INFO          1057

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifdef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        134
#define _APS_NEXT_COMMAND_VALUE        32771
#define _APS_NEXT_CONTROL_VALUE        1058
#define _APS_NEXT_SYMED_VALUE          101
#endif
#endif
```

完成したプログラムのソースリスト

USERMAC.h

```

const    MACRO_LOOP_MAX = 100; // マクロの中のマクロ最大数

/*****
[ クラス名 ] CUserCode
[基底クラス] なし
[ 機能 ] マクロ情報の格納
[ メンバ ] Name          ・ マクロ名
              Code          ・ マクロのコード部
              Information    ・ マクロの機能説明
              pBefour        ・ リスト構築用
*****/
class    CUserMacro;
class    CUserCode {
public:
    CUserCode()
    { pBefour = NULL; }
    CUserCode( CString name, CString code, CString info, CUserCode*
pbefour )
    { Name = name; Name.MakeLower();    Code = code;    Information = info;
pBefour = pbefour; }
    ~CUserCode(){ if( pBefour != NULL )delete pBefour; }
private:
    CString      Name;
    CString      Code;
    CString      Information;
    CUserCode*   pBefour;
friend    CUserMacro;
};

/*****
マクロ処理用グローバル変数
*****/

```

完成したプログラムのソースリスト

```

#ifdef _INIT_MACRO
    CUserCode      Macro_Symtab;                // マクロリスト
    の先頭
    CString MacroHistory[ MACRO_LOOP_MAX ]; // 実行したマクロの名前格納
    int      MacroHistCnt;                      //
    MacroHistory のインデックス用
#else
    extern  CUserCode Macro_Symtab;
    extern  CString MacroHistory[ MACRO_LOOP_MAX ];
    extern  int      MacroHistCnt;
#endif

/*****
[ クラス名 ] CUserMacro
[基底クラス] なし
[ 機能 ] マクロリストの構築、管理
[ メンバ ] AllDelete() ・マクロ名
          --- CUserCode のメンバを参照可 ---
*****/
class CUserMacro {
public:
    CUserMacro(){}
    ~CUserMacro(){}

    void      AllDelete()
                { if( Macro_Symtab.pBefour != NULL )delete
Macro_Symtab.pBefour; Macro_Symtab.pBefour = NULL;      }

    BOOL      MakeMacro( CString name, CString code, CString info );
    BOOL      GetMacro( CString name, CString* code, CString* info
= NULL );

    BOOL      DeleteMacro( CString name );
    BOOL      GetMacroInfo( int index, CString* name, CString* code,
CString* info = NULL );
private:
    //CUserCode      Symtab;
protected:

```


完成したプログラムのソースリスト

```
CUUserCode* Find( CString name );  
BOOL CheckName( CString Name );  
};
```

完成したプログラムのソースリスト

STDAFX.h

```
// stdafx.h : 標準のシステム インクルード ファイル、  
//          または参照回数が多く、かつあまり変更されない  
//          プロジェクト専用のインクルード ファイルを記述します。  
//  
  
#include <afxwin.h>          // MFC の標準コンポーネント  
#include <afxext.h>          // MFC の拡張部分
```

完成したプログラムのソースリスト

Compute.cpp

```

/*=====
    Compute.cpp
=====*/

#include "stdafx.h"
#include "DT.h"

#define      _RAD_INIT
#include "Complx.h"

#include "NODES.h"
#include "UserMac.h"
#include "Compute.h"
#include "Data.h"

/*****
[ 機能 ] p_t = 1 のときは スペース、タブを '*' として扱う。
          p_t = 0 のときはしない。
*****/
int      p_t;

/*****
[関数名] Compute(コンストラクタ)
[ 属性 ] public
[ 機能 ]
*****/
CCompute::CCompute()
{
}

/*****
[関数名] ~Compute(デストラクタ)
[ 属性 ] public
[ 機能 ]

```

完成したプログラムのソースリスト

```

*****/
CCompute::~CCompute()
{
}

/*****/
[関数名] GoCompute
[ 属性 ] public
[ 機能 ] 文字列 pSorce で指定される数式を計算して結果を Answer
         に格納する。通常は TRUE を計算に失敗した場合は FALSE を
         返す。
*****/
BOOL  CCompute::GoCompute( CString str, complx* Answer )
{
    char    buf[256];
    strcpy(buf, (LPCSTR)str);
    SorceString = str;
    pSorceStr = buf; //(char *)(const char *)SorceString;

    Node    *root;
    BOOL    Error = TRUE;

    p_t = 0;
    token = Scan();
    do {
        if( (root = E()) && (token == EOLN || token == ';') ){
            *Answer = root->eval();
            Error = TRUE;
        }else {
            Error = FALSE;;
            if( root != NULL ) delete root;        break;
        }
        if( root != NULL ) delete root;
        p_t = 0;
    }while( (token = Scan()) != EOLN );
    return Error;
}

```

完成したプログラムのソースリスト

}

/*****

[関数名] Add_NUm

[属性] protected

[機能] pSorceStr から数字を読み込んで TokenStr に入れる。

*****/

char CCompute::Add_Num(Token c)

{

do

TokenStr += (char)c;

while(isdigit(c = tolower(GetChar())));

return c;

}

/*****

[関数名] Add_Char

[属性] protected

[機能] pSorceStr から 1 文字読み込んで TokenStr 付け加える。

*****/

char CCompute::Add_Char(Token c)

{

TokenStr += (char)c;

c = tolower(GetChar());

return c;

}

/*****

[関数名] Scan()

[属性] protected

[機能] 式文字列 pSorceStr から 1 トークン読み込んでその種類を返す。文字、数字の時はTokenStrに格納する。

(トークン) (戻り値) (TokenStr)

記号(+,-,*,/,...) ascii code なし

完成したプログラムのソースリスト

'\$'	USER	なし
ヌル文字	EOLN	なし
数字	NUM	格納
文字	ID	格納

```

*****/
Token  CCompute::Scan()
{
    Token  c;
    TokenStr.Empty();
    while( 1 ){
        switch( c = tolower( GetChar() ) ){
            case  '+'      :      case  '-'      :      case
            '*'      :      case  '/'      :
            '='      :      case  ','      :      case
            ';'      :
                p_t = 0; return  c;
            case  '('      :      case  ')'      :
                p_t = 1; return  c;
            case  '¥n':      case  '¥r':
                continue;
            case  ' '      :      case  '¥t'      :
                if(p_t){
                    int      c1 = GetChar();  UnGetChar(c1);
                    switch(c1){
                        case  '+':      case  '-':
                        case  '=':      case  ',':
                        case  ' ':
                            continue;
                        default :
                            p_t = 0; return  c;
                    }
                } else continue;
            case  '$':
                return  USER;
            case  '¥0':

```

完成したプログラムのソースリスト

```

        return  EOLN;
    default  :
        if( isdigit( c ) ){
            c = Add_Num( c );
            switch( c ){
                case  '.' :
                    c = Add_Char( c );
                    if( isdigit( c ) )    c =
Add_Num( c );
                    else    if( c == '.')
return  BAD;

                case  'e' :
                    c = Add_Char( c );
                    if( c == '-' )    c =
Add_Char( c );
                    if( (c == '-') || (c == 'e') )
return  BAD;

                    c = Add_Num( c );
                    break;
            }
            UnGetChar( c );
            p_t = 1; return  NUM;
        }
        if( iscsymf( c ) ){
            do {
                TokenStr += c;
                c = tolower( GetChar() );
            }while( iscsym( c ) );
            UnGetChar( c );
            p_t = 1; return  ID;
        }
        return  BAD;
    }
}
}

```

完成したプログラムのソースリスト

```

/*****
[関数名] E
[ 属性 ] protected
[ 機能 ] 構文解析 ( +、 - )
*****/
Node* CCompute::E()
{
    Node    *root = T();
    while(1){
        switch( token ){
            case  '+'      :
                token = Scan();
                root = new      Plus(root, T() );
                break;
            case  '-'      :
                token = Scan();
                root = new      Minus(root,T() );
                break;
            default :
                return  root;
        }
    }
}

```

```

/*****
[関数名] T
[ 属性 ] protected
[ 機能 ] 構文解析 ( ×、 ÷ )
*****/
Node* CCompute::T()
{
    Node    *root = F();
    while(1){
        switch( token ){

```


完成したプログラムのソースリスト

```

        case '*' : case ' ' : case
        '¥t' :
            token = Scan();
            root = new Times( root, F() );
            break;
        case '/' :
            token = Scan();
            root = new Div( root, F() );
            if( root->error() == FALSE ){ token = BAD; ErrorNumber
= DIVISION_ERROR; }
            break;
        default :
            return root;
    }
}
}
}

```

/******

[関数名] F

[属性] protected

[機能] 構文解析 (数字、変数、関数、括弧,,)

*****/

Node* CCompute::F()

```

{
    CCompute    a;
    complx      cmp;
    Node    *root = NULL;
    switch( token ){
        case NUM:
            root = new Num( atof( (const char *)TokenStr  ) );
            token = Scan();
            return root;
        case '(':
            token = Scan();
            root = E();
            if( token == BAD ){ ErrorNumber = SYNTAX_ERROR; return

```

完成したプログラムのソースリスト

```

root; }

        if( token != ')' ){ ErrorNumber = KAKKO_ERROR; token = BAD;

return root; }

        token = Scan();
        return root;
    case   '-':
        token = Scan();
        return new Uminus( F() );
    case   ID:
        int      func_index;
        func_index = IsFunc( TokenStr, &token );
        switch( token ) {
            case   FUNC:   case   FUNC2:
                root = CallFunc( func_index );
                if( token != BAD ) token = Scan();  // エラー
                break;
            case   GVAR:
                root = new Id( (char *) (const char
*)TokenStr );

                ErrorToken = TokenStr;
                if( (token = Scan()) == '=' ) {
                    token = Scan();
                    root = new      Assign((Id *)root,

E());

                    if( root->error() == FALSE ) {
                        ErrorNumber =

                        token = BAD;      break;

// 代入出来ない

                }
            }else {
                root = new Times(root, new

Num( 1.0 ) );      break;

            }
        default: break;

```

完成したプログラムのソースリスト

```

        }
        return root;

    case    USER:
        if( (token = Scan()) != ID )
        { ErrorNumber = MACRO_ERROR1; token = BAD;    return
root; }

        root = CallUserMacro( TokenStr );

        if( root == NULL ){ token = BAD;    return root; }
        token = Scan();
        return root;

    default :
        ErrorNumber = SYNTAX_ERROR;
        token = BAD;
        return root;

    }
    token = BAD;
    return    root;
}

```

```

/*****

```

```

[関数名] IsFunc

```

```

[ 属性 ] protected

```

```

[ 機能 ] TokenStr が関数名かどうかを識別。結果をtokenに格納する。

```

関数名でないときは token に GVAR(変数)を格納する。

戻り値として何の関数かがわかる「関数番号」を返す。

```

*****/

```

```

int          CCompute::IsFunc( CString pIdName, int* token )
{
    int loop;
    *token = FUNC;  //引数 1 個
    for( loop = 0; *(FuncTable1[ loop ].name); loop++ )
        if( pIdName.Compare( FuncTable1[ loop ].name ) == 0 )return loop;
}

```

完成したプログラムのソースリスト

```

        *token = FUNC2; //引数 2 個
        for( loop = 0; *(FuncTable2[ loop ].name); loop++ )
            if( pIdName.Compare( FuncTable2[ loop ].name ) == 0 )return loop;

        *token = GVAR; //変数
        return 0;
    }

/*****
[関数名] CallFunc
[ 属性 ] protected
[ 機能 ] 関数番号 FuncIndex で指定される関数を呼び出す。
        関数名でないときは token に GVAR(変数)を格納する。
        戻り値として何の関数かがわかる「関数番号」を返す。
*****/
Node* CCompute::CallFunc(int FuncIndex )
{
    const ARG_TYPE1 = 0, ARG_TYPE2 = 1;
    int loop = 0;
    int ParamNum = token - (int)FUNC;
    Node *root, *pParam[ PARAM_MAX ];

    if( (token = GetChar()) != '(' ) {
        ErrorNumber = FUNC_ERROR2;
        token = BAD; return NULL;
    }

    do{
        token = Scan();
        pParam[ loop ] = E();
        if( token != ',' && token != ')' ) {
            ErrorNumber = FUNC_ERROR1;
            token = BAD; return NULL;
        }
    }while( ParamNum > loop++ );
    if( token != ')' ){ ErrorNumber = FUNC_ERROR1; token = BAD; return NULL; }
}

```

完成したプログラムのソースリスト

```
// 引き数の個数でコンストラクタを呼び分ける
switch( ParamNum ){
    case ARG_TYPE1:root = new Func( FuncIndex, pParam[0] );    break;
    //引数 1 個
    case ARG_TYPE2:root = new Func( FuncIndex, pParam[0], pParam[1] );
break; //引数 2 個
    default:
        token = BAD;
        ErrorNumber = FUNC_ERROR1;
        return NULL;    // 引数の数があわない (エラー)

}
return    root;
}
```

```
/******
```

[関数名] CallUserMacro

[属性] protected

[機能] name がマクロ名かどうかチェック後、マクロを実行するCall

UserMacroSubを呼び出す。

実行できた場合は計算結果を格納するNumオブジェクトのポインタを、それ以外はNULLを返す。

```
*****/
```

```
Node*    CCompute::CallUserMacro( CString name    )
```

```
{
```

```
    complex        value;
```

```
    CString        MacroCode;
```

```
    CCompute        Com;
```

```
    if( m_UserMacro.GetMacro( name, &MacroCode ) == TRUE ){
```

```
        if( MacroLoopCheck( name ) == TRUE ){
```

```
            MacroHistCnt++;
```

```
            if( Com.GoCompute( (char*)(const char*)MacroCode, &value )
```

```
== TRUE ){
```

```
                MacroHistCnt--;
```

完成したプログラムのソースリスト

```

        return new Num( value );
    }else
        return NULL;
    }else{
        ErrorNumber = MACRO_ERROR2;
        return NULL;
    }
}else {
    ErrorNumber = MACRO_ERROR1;
    return NULL;
}
}

/*****
[関数名] MacroLoopCheck
[ 属性 ] protected
[ 機能 ] MacroNameで示されるマクロが再帰するかどうかチェック。
        しない場合は TRUE を、再帰する場合と階層が MACRO_LOOP_MAX
        より深くなる場合は FALSE を返します。
*****/
BOOL  CCompute::MacroLoopCheck( CString MacroName )
{
    for( int loop = 0; loop < MacroHistCnt; loop++ ) {
        if( MacroName == MacroHistory[loop] )return FALSE;
    }
    MacroHistory[ MacroHistCnt ] = MacroName;
    if( MacroHistCnt < MACRO_LOOP_MAX )
        return TRUE;
    else
        return FALSE;
}

```

完成したプログラムのソースリスト

DT_.cpp

// DT_.cpp : アプリケーション用クラスの機能定義を行います。

//

#include "stdafx.h"

#include "DT.h"

#include "Complx.h"

#include "NODES.h"

#include "UserMac.h"

#include "Compute.h"

#include "DT_dlg.h"

#ifdef _DEBUG

#undef THIS_FILE

static char BASED_CODE THIS_FILE[] = __FILE__;

#endif

//

// CDT_App

BEGIN_MESSAGE_MAP(CDT_App, CWinApp)

//{{AFX_MSG_MAP(CDT_App)

// NOTE - ClassWizard はこの位置にマッピング用のマクロを追加または削除します。

// この位置に生成されるコードを編集しないでください。

//}}AFX_MSG

ON_COMMAND(ID_HELP, CWinApp::OnHelp)

END_MESSAGE_MAP()

//

// CDT_App クラスの構築

完成したプログラムのソースリスト

```

CDT_App::CDT_App()
{
    // TODO: この位置に構築用コードを追加してください。
    // ここに InitInstance 中の重要な初期化処理をすべて記述してください。
}

////////////////////////////////////
// 唯一の CDT_App オブジェクト

CDT_App theApp;

////////////////////////////////////
// CDT_App クラスの初期化

BOOL CDT_App::InitInstance()
{
    // 標準的な初期化処理
    // もしこれらの機能を使用せず、実行ファイルのサイズを小さくしたければ
    // 以下の特定の初期化ルーチンの中から不必要なものを削除してください。

    Enable3dControls();
    LoadStdProfileSettings(); // 標準 INI ファイルをロードします。(MRU を含
む)

    CDT_Dlg dlg;
    m_pMainWnd = &dlg;
    int nResponse = dlg.DoModal();
    if (nResponse == IDOK)
    {
        // TODO: この位置にはダイアログが <OK> で消されたときのコード
        を記述してください
    }
    else if (nResponse == IDCANCEL)
    {

```


完成したプログラムのソースリスト

```
        // TODO: この位置にはダイアログが <キャンセル> で消されたときのコードを記述してください
```

```
    }
```

```
        // ダイアログ閉じられてからアプリケーションのメッセージ ポンプを開始するよりは、
```

```
        // アプリケーションを終了するために FALSE を返してください。
```

```
        return FALSE;
```

```
    }
```

完成したプログラムのソースリスト

Dt_dlg.cpp

// DT_dlg.cpp : インプリメンテーション ファイル

//

#include "stdafx.h"

#include "DT.h"

#include "Complex.h"

#include "UserMac.h"

#include "FuncDlg.h"

#include "MacroDlg.h"

#include "GvarDlg.h"

#include "OptDlg.h"

#include "Nodes.h"

#include "Compute.h"

#include "DT_dlg.h"

#define _INIT

#include "Data.h"

#ifdef _DEBUG

#undef THIS_FILE

static char BASED_CODE THIS_FILE[] = __FILE__;

#endif

////////////////////////////////////

// アプリケーションのバージョン情報で使われている CAboutDlg ダイアログ

char FilSysPath[256]; // プログラムの起動パスをコピーしておく。

char SysPath[256], AbsPath[256], Drv[256], Dir[256];

Dt_dlg.cpp

105 / 155

完成したプログラムのソースリスト

```

class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

    // ダイアログ データ
   //{{AFX_DATA(CAboutDlg)
    enum { IDD = IDD_ABOUTBOX };
   //}}AFX_DATA

    // インプリメンテーション
protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV のサポート
   //{{AFX_MSG(CAboutDlg)
    virtual BOOL OnInitDialog();
   //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
   //{{AFX_DATA_INIT(CAboutDlg)
   //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CAboutDlg)
   //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
   //{{AFX_MSG_MAP(CAboutDlg)
        // メッセージ ハンドラがありません。
    }}AFX_MSG_MAP

```

完成したプログラムのソースリスト

```

        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CAboutDlg メッセージ ハンドラ

BOOL CAboutDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    CenterWindow();

    // TODO: 特別なバージョン情報 ダイアログの初期化を行うときはこの場所に追加してく
    ださい。

    return TRUE; // TRUE を返すとコントロールに設定したフォーカスは失われません。
}

////////////////////////////////////
// CDT_Dlg ダイアログ

CDT_Dlg::CDT_Dlg(CWnd* pParent /*=NULL*/)
    : CDialog(CDT_Dlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CDT_Dlg)
    m_EditBox1 = _T("");
    m_EditBox2 = _T("");
    //}}AFX_DATA_INIT
    // LoadIcon は Win32 の DestroyIcon サブシーケンスを要求しません。
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CDT_Dlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CDT_Dlg)
    DDX_Text(pDX, IDC_EDIT1, m_EditBox1);

```

完成したプログラムのソースリスト

```

        DDX_Text(pDX, IDC_EDIT2, m_EditBox2);
        //}}AFX_DATA_MAP
    }

BEGIN_MESSAGE_MAP(CDT_Dlg, CDialog)
    //{{AFX_MSG_MAP(CDT_Dlg)
        ON_WM_SYSCOMMAND()
        ON_WM_DESTROY()
        ON_WM_PAINT()
        ON_WM_QUERYDRAGICON()
        ON_BN_CLICKED(IDC_B01_GVAR, OnB01Gvar)
        ON_BN_CLICKED(IDC_B02_FUNC, OnB02Func)
        ON_BN_CLICKED(IDC_B03_OPTION, OnB03Option)
        ON_BN_CLICKED(IDC_BT27_ENTER, OnBt27Enter)
        ON_BN_CLICKED(IDC_BT24_AC, OnBt24Ac)
        ON_BN_CLICKED(IDC_BT25_C, OnBt25C)
        ON_BN_CLICKED(IDC_BT26_BS, OnBt26Bs)
        ON_BN_CLICKED(IDC_MACRO, OnMacro)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/*****
[関数名] ~CDT_Dlg
[ 属性 ] public
[ 機能 ] 変数・定数リストのメモリ解放
*****/
CDT_Dlg::~CDT_Dlg()
{
    Id      id;
    id.DeleteList();
}

/*****
[関数名] ClearEditBox
[ 属性 ] protected
[ 機能 ] 式、計算結果の表示を消す。

```

完成したプログラムのソースリスト

```

*****/
void CDT_Dlg::ClearEditBox( int BoxNum )
{
    m_EditBox1.Empty();
    m_EditBox2.Empty();
    UpdateData( FALSE );
}

////////////////////////////////////
// CDT_Dlg メッセージ ハンドラ

/*****
[関数名] OnInitDialog
[ 属性 ] protected
[ 機能 ] ダイアログの初期化( m_CommaMode, m_Float の初期化)
*****/
BOOL CDT_Dlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    CenterWindow();

    // “バージョン情報...” メニュー項目をシステム メニューへ追加します。

    // IDM_ABOUTBOX はシステム メニューの範囲内でなければなりません。
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu* pSysMenu = GetSystemMenu(FALSE);
    CString strAboutMenu;
    strAboutMenu.LoadString(IDS_ABOUTBOX);
    if (!strAboutMenu.IsEmpty())
    {
        pSysMenu->AppendMenu(MF_SEPARATOR);
        pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
    }
}

```

完成したプログラムのソースリスト

```
// TODO: 特別な初期化を行うときにはこの場所に追加してください。
m_Float = DEFAULT_FLOAT;          // 小数部表示桁数
m_CommaMode = 0;                  // コンマ挿入モードOFF

GetModuleFileName(NULL, AbsPath, 256);
_splitpath(AbsPath, Drv, Dir, NULL, NULL);
strcpy(SysPath, Drv); strcat(SysPath, Dir);
strcpy(FilSysPath, SysPath);

return TRUE; // TRUE を返すとコントロールに設定したフォーカスは失われません。
}

void CDT_Dlg::OnSysCommand(UINT nID, LPARAM lParam)
{
    if ((nID & 0xFFF0) == IDM_ABOUTBOX)
    {
        CAboutDlg dlgAbout;
        dlgAbout.DoModal();
    }
    else
    {
        CDialog::OnSysCommand(nID, lParam);
    }
}

void CDT_Dlg::OnDestroy()
{
    WinHelp(0L, HELP_QUIT);
    CDialog::OnDestroy();
}

// もしダイアログボックスに最小化ボタンを追加するならば、アイコンを描画する
// コードを以下に記述する必要があります。MFC アプリケーションは document/view
// モデルを使っているので、この処理はフレームワークにより自動的に処理されます。

void CDT_Dlg::OnPaint()
```

完成したプログラムのソースリスト

```

{
    if (IsIconic())
    {
        CPaintDC dc(this); // 描画用のデバイス コンテキスト。

        SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

        // クライアント矩形領域内の中央。
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // アイコンを描画します。
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}

// システムコールは、ユーザーが最小化ウィンドウをドラッグしている間の
// カーソル表示を行います。
HCURSOR CDT_Dlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}

/*****
[関数名] OnMacro( マクロボタン・クリック )
[ 属性 ] protected
[ 機能 ] マクロダイアログの表示、選択されたマクロの式への追加。
*****/

```


完成したプログラムのソースリスト

```

void CDT_Dlg::OnMacro()
{
    // TODO: この位置にコントロール通知ハンドラのコードを追加してください
    CMacroDlg      MacroDlg;
    MacroDlg.SetUserMacro( m_Com.GetUserMacro() );
    if( MacroDlg.DoModal() == IDOK ) {
        UpdateData( TRUE );
        m_EditBox1 += MacroDlg.GetMacroName();
        UpdateData( FALSE );
    }

    DefForcus();
}

/*****
[関数名] OnB01Gvar( 変数ボタン・クリック )
[ 属性 ] protected
[ 機能 ] 変数ダイアログの表示、選択された変数の式への追加
*****/
void CDT_Dlg::OnB01Gvar()
{
    CGvarDlg      GvarDlg;
    if( GvarDlg.DoModal() == IDOK ) {
        UpdateData( TRUE );
        m_EditBox1 += GvarDlg.GetGvarName();
        UpdateData( FALSE );
    }

    DefForcus();
}

/*****
[関数名] OnB02Func( 関数ボタン・クリック )
[ 属性 ] protected
[ 機能 ] 関数ダイアログの表示、選択された関数の式への追加。
*****/
void CDT_Dlg::OnB02Func()

```

完成したプログラムのソースリスト

```

{
    CFuncDlg      FuncDlg;
    if( FuncDlg.DoModal() == IDOK ) {
        UpdateData( TRUE );
        m_EditBox1 += FuncDlg.GetFuncName();
        UpdateData( FALSE );
    }
    DefForcus();
}

/*****
[関数名] OnB03Option( 設定ボタン・クリック )
[ 属性 ] protected
[ 機能 ] 設定ダイアログの表示。
*****/
void CDT_Dlg::OnB03Option()
{
    COptDlg OptionDlg;
    OptionDlg.SetStatus( &rad, &m_CommaMode, &m_Float );
    if( OptionDlg.DoModal() == IDOK ) {
        OptionDlg.GetStatus( &rad, &m_CommaMode, &m_Float );
    }

    DefForcus();
}

/*****
[関数名] OnBt24Ac( ACボタン・クリック )
[ 属性 ] protected
[ 機能 ] 変数の初期化、表示を消す
*****/
void CDT_Dlg::OnBt24Ac()
{
    if( MessageBox("全ての変数を初期化します","変数の初期化", MB_OKCANCEL ) ==
IDOK ) {

        Id      id;

```

完成したプログラムのソースリスト

```

        id.DeleteList();
        OnBt25C();
    }
    GotoDlgCtrl( GetDlgItem( IDC_EDIT1 ) );
}

/*****
[関数名] OnBt25C( Cボタン・クリック )
[ 属性 ] protected
[ 機能 ] 表示を消す
*****/
void CDT_Dlg::OnBt25C()
{
    ClearEditBox();
    GotoDlgCtrl( GetDlgItem( IDC_EDIT1 ) );
}

/*****
[関数名] OnBt26Bs( BSボタン・クリック )
[ 属性 ] protected
[ 機能 ] 式の最後の1文字を削除
*****/
void CDT_Dlg::OnBt26Bs()
{
    UpdateData( TRUE );
    int len = m_EditBox1.GetLength();
    if( len ) m_EditBox1 = m_EditBox1.Left( len - 1 );
    UpdateData( FALSE );
    GotoDlgCtrl( GetDlgItem( IDC_EDIT1 ) );
}

/*****
[関数名] OnCommand( 0～9, +～/ ,,,,ボタン・クリック )
[ 属性 ] protected
[ 機能 ] 式の最後にクリックされたボタンの文字を追加
*****/

```

完成したプログラムのソースリスト

```

BOOL CDT_Dlg::OnCommand(WPARAM wParam, LPARAM lParam)
{
    if( wParam >= BT_ID_TOP && wParam <= BT_ID_END ) {
        UpdateData( TRUE );
        m_EditBox1 += KEY_CHAR[ wParam - BT_ID_TOP ];
        UpdateData( FALSE );
    }
    return CDialog::OnCommand(wParam, lParam);
}

/*****
[関数名] OnBt27Enter( Enter ボタン・クリック )
[ 属性 ] protected
[ 機能 ] エディットボックスから式文字列を拾い計算後、表示
*****/
void CDT_Dlg::OnBt27Enter()
{
    UpdateData( TRUE );

    complex Answer;
    double re, im, re_abs, im_abs;

    MacroHistCnt = 0;
    ErrorNumber = 0;

    if( m_Com.GoCompute( m_EditBox1, &Answer ) ) {
        char buf[7];
        m_EditBox2.Empty();
        re = Answer.Real();          re_abs = fabs(re);
        im = Answer.Imag();          im_abs = fabs(im);
        if(re_abs < 1e-15) re = 0;
        if(im_abs < 1e-15) im = im_abs = 0;

        if( re == 0 && im == 0 ) m_EditBox2 = "0";

        //// 実数部 ////

```

完成したプログラムのソースリスト

```

        if( re != 0 ) {
            if( m_CommaMode ) {
                if( InsertComma( re, &m_EditBox2 ) == FALSE ) {
                    _gcvrt( re, m_Float, buff );
                    m_EditBox2 = buff;
                }
            }else {
                _gcvrt( re, m_Float, buff );
                m_EditBox2 = buff;
            }
        }

        //// 虚数部 ////
        if( im != 0 ) {
            CString Num;
            if( m_CommaMode ) {
                if( InsertComma( im_abs, &Num ) == FALSE ) {
                    _gcvrt( im_abs, m_Float, buff );
                    Num = buff;
                }
            }else {
                _gcvrt( im_abs, m_Float, buff );
                Num = buff;
            }

            if(im > 0) strcpy(buf," + ");
            if(im < 0) strcpy(buf," - ");
            if( re == 0 ){if(im < 0)      m_EditBox2 = &buf[1];}
            else      m_EditBox2 += buf;
            m_EditBox2 += Num;
            m_EditBox2 += " * I";
        }
    }
    else {
        //// エラー処理 ////
        m_EditBox2 = "ERROR: ";
        if(ErrorNumber == ASSIGN_ERROR ) AssignError();
    }
}

```

完成したプログラムのソースリスト

```

        else      m_EditBox2 += ErrorMessage[ ErrorNumber ];
    }

    // 式エディットボックスを反転
    GotoDlgCtrl( GetDlgItem( IDC_EDIT1 ) );
    UpdateData( FALSE );
}

/*****
[関数名] InsertComma
[ 属性 ] protected
[ 機能 ] 小数 f をコンマ入りで文字列に変換し、Buffに格納する。
        通常は TRUE を f が 1 未満の時はは FALSE を返す。
*****/
BOOL  CDT_Dlg::InsertComma( double f, CString* Buff )
{
    int      dec, sign;
    const    MAX_COMMA_NUM = 50;

    if( f > max_num )return FALSE;
    strcpy( buff, _fcvt( f, m_Float, &dec, &sign ) );
    if( dec <= 0 )return FALSE;

    Buff->Empty();
    if( sign != 0 )*Buff = '-' ;
    *Buff += InsertCommaSub( buff + dec - 1, dec );
    *Buff += '.';

    char* p = buff + dec;
    while( *p )p++;
    for( p--; p >= buff + dec; p-- ) {
        if( *p == '0' )*p = '¥0';
        if( *p > '0' )break;
    }
    *Buff += buff + dec;
}

```

完成したプログラムのソースリスト

```

        return TRUE;
    }

/*****
[関数名] InsertCommaSub
[ 属性 ] protected
[ 機能 ] fcvで変換された文字列の小数点位置 pNumStr と整数部
        の桁数 NumLen を受け取り,カンマ入りの文字列作成して
        返す。
*****/
CString CDT_Dlg::InsertCommaSub( char* pNumStr, int NumLen )
{
    CString num;
    int cnt = 0;
    for( int loop = 0; loop < NumLen; loop++ ) {
        cnt++;
        if( cnt > 3 ){ cnt = 1; num += ','; }
        num += *pNumStr;
        pNumStr++;
    }
    num.MakeReverse();
    return num;
}

/*****
[関数名] AssignError
[ 属性 ] protected
[ 機能 ] 定数に値を代入使用とした場合のエラー表示
*****/
void CDT_Dlg::AssignError()
{
    Id      id( (char*)(const char*)ErrorToken );
    m_EditBox2 += id.GetGvarInfo();
    m_EditBox2 += " には値を代入出来ません。";
}

```

完成したプログラムのソースリスト

```

/*****
[関数名] DefFocus
[ 属性 ] protected
[ 機能 ] フォーカスを式エディットボックスに移す
*****/
void CDT_Dlg::DefFocus()
{
    UpdateData(TRUE);
    CString BuffStr = m_EditBox1;
    SetDlgItemText( IDC_EDIT1, "" );
    GotoDlgCtrl( GetDlgItem( IDC_EDIT1 ) );
    m_EditBox1 = BuffStr;
    UpdateData(FALSE);
}

/*****
[関数名] WinHelp (ヘルプの起動)
[ 属性 ] protected
[ 機能 ] ヘルプが呼ばれたら、ヘルプの目次を呼び出す。
*****/
void CDT_Dlg::WinHelp(DWORD dwData, UINT nCmd)
{
    char name[256];
    if(nCmd == 1){
        sprintf(name, "%s電卓プログラム 説明書.pdf", FilSysPath);
        ShellExecute(NULL, "open", name, "", "", SW_SHOW);
    }
}

```


完成したプログラムのソースリスト

FUNCDLG.cpp

```
// funcdlg.cpp : インプリメンテーション ファイル
//

#include "stdafx.h"
#include "DT.h"

#include "complx.h"
#include "Data.h"

#include "funcdlg.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

//////////////////////////////////////
// CFuncDlg ダイアログ

CFuncDlg::CFuncDlg(CWnd* pParent /*=NULL*/)
: CDialog(CFuncDlg::IDD, pParent)
{
   //{{AFX_DATA_INIT(CFuncDlg)
    m_FuncInfo = _T("");
    //}}AFX_DATA_INIT
}

void CFuncDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CFuncDlg)
    DDX_Text(pDX, IDC_FUNC_INFOTEXT, m_FuncInfo);
    //}}AFX_DATA_MAP
}
```

完成したプログラムのソースリスト

```

        //}}AFX_DATA_MAP
    }

BEGIN_MESSAGE_MAP(CFuncDlg, CDialog)
    //{{AFX_MSG_MAP(CFuncDlg)
        ON_LBN_DBLCLK(IDC_FUNC_LIST1, OnDbclkFuncList1)
        ON_LBN_SELCHANGE(IDC_FUNC_LIST1, OnSelchangeFuncList1)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CFuncDlg メッセージ ハンドラ

/*****
[関数名] OnInitDialog
[ 属性 ] protected
[ 機能 ] ダイアログの初期化、リストボックスへの関数名の追加
*****/
BOOL CFuncDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    // TODO: この位置にその他の初期化用コードを追加してください。
    int    loop, ListIndex = 0;
    loop = 0;
    CString func_name;

    while( *(FuncTable1[loop].name) ) {
        func_name = FuncTable1[loop].name;
        func_name += "(a)";
        SendDlgItemMessage( IDC_FUNC_LIST1, LB_INSERTSTRING, ListIndex++,
            (LONG)(LPSTR)(const char *) func_name );
        loop++;
    }
}

```

完成したプログラムのソースリスト

```

        loop = 0;
        while( *(FuncTable2[loop].name) ) {
            func_name = FuncTable2[loop].name;
            func_name += "(a,b)";
            SendDlgItemMessage( IDC_FUNC_LIST1, LB_INSERTSTRING, ListIndex++,
                (LONG)(LPSTR)(const char *) func_name );
            loop++;
        }
        return TRUE;
    }
}

```

```

/*****

```

[関数名] OnOK

[属性] protected

[機能] 選択された関数名をリストボックスから拾い、pBuffに格納後、ダイアログを閉じる。

```

*****/

```

```

void CFuncDlg::OnOK()
{
    int      index = (int)SendDlgItemMessage( IDC_FUNC_LIST1, LB_GETCURSEL, 0,
0 );

    if( index >= 0 )
        SendDlgItemMessage( IDC_FUNC_LIST1, LB_GETTEXT, index,
( LONG )( LPSTR )pBuff );
    else      *pBuff = '¥0';
    CDialog::OnOK();
}

```

```

/*****

```

[関数名] OnDbclckFuncList1(リストボックス・ダブルクリック)

[属性] protected

[機能] OnOKを呼び出す。

```

*****/

```

```

void CFuncDlg::OnDbclckFuncList1()
{
    OnOK();
}

```

完成したプログラムのソースリスト

}

/*****

[関数名] OnSelchangeFuncList1(リストボックス内の選択が変更)

[属性] protected

[機能] 関数の機能説明を表示

*****/

void CFuncDlg::OnSelchangeFuncList1()

{

// TODO: この位置にコントロール通知ハンドラのコードを追加してください

int index = (int)SendDlgItemMessage(IDC_FUNC_LIST1, LB_GETCURSEL, 0,

0);

SendDlgItemMessage(IDC_FUNC_LIST1, LB_GETTEXT, index,

(LONG)(LPSTR)pBuff);

char* p = pBuff;

while(*p++ != '(');*(p-1) = '¥0';

GetFuncInfo(pBuff, &m_FuncInfo);

UpdateData(FALSE);

}

/*****

[関数名] GetFuncInfo

[属性] protected

[機能] Nameで示される関数の機能説明を Info に格納。

*****/

void CFuncDlg::GetFuncInfo(CString Name, CString* Info)

{

int loop = 0;

while(*(FuncTable1[loop].name)){

if(Name == FuncTable1[loop].name){ *Info = FuncTable1[loop].info;

return; }

loop++;

}

完成したプログラムのソースリスト

```
        loop = 0;
        while( *(FuncTable2[loop].name) ) {
            if( Name == FuncTable2[loop].name ){ *Info = FuncTable2[loop].info;
return; }
            loop++;
        }
    }
```

完成したプログラムのソースリスト

GVARDLG.cpp

```
// gvardlg.cpp : インプリメンテーション ファイル
//
```

```
#include "stdafx.h"
#include "DT.h"
```

```
#include "complx.h"
#include "nodes.h"
```

```
#include "gvardlg.h"
#include "Data.h"
```

```
#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif
```

```
////////////////////////////////////
// CGvarDlg ダイアログ
```

```
CGvarDlg::CGvarDlg(CWnd* pParent /*=NULL*/)
: CDialog(CGvarDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CGvarDlg)
    m_EditBox_Real = _T("");
    m_EditBox_Image = _T("");
    m_Info = _T("");
    //}}AFX_DATA_INIT
}
```

完成したプログラムのソースリスト

```

void CGvarDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CGvarDlg)
    DDX_Text(pDX, IDC_GVAR_EDIT1, m_EditBox_Real);
    DDX_Text(pDX, IDC_GVAR_EDIT2, m_EditBox_Image);
    DDX_Text(pDX, IDC_GVAR_INFO, m_Info);
   //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CGvarDlg, CDialog)
   //{{AFX_MSG_MAP(CGvarDlg)
    ON_LBN_SELCHANGE(IDC_GVAR_LIST, OnSelchangeGvarList)
    ON_BN_CLICKED(IDC_GVAR_ASSIGN, OnGvarAssign)
    ON_LBN_DBLCLK(IDC_GVAR_LIST, OnDblclkGvarList)
    ON_BN_CLICKED(IDC_GVAR_DELETE, OnGvarDelete)
    ON_BN_CLICKED(IDC_GVAR_LOAD, OnGvarLoad)
    ON_BN_CLICKED(IDC_GVAR_SAVE, OnGvarSave)
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/*****
[関数名] GetListBoxText
[ 属性 ] protected
[ 機能 ] リストボックス内の選択されているテキストをpBuffに格納
         通常は TRUE を、選択されていない場合は FALSE を返す。
*****/
BOOL CGvarDlg::GetListBoxText( char* pbuff )
{
    int      index = (int)SendDlgItemMessage( IDC_GVAR_LIST,  LB_GETCURSEL, 0,
0 );
    if( index >= 0 ) {
        SendDlgItemMessage( IDC_GVAR_LIST,  LB_GETTEXT, index,
( LONG )( LPSTR )pbuff );
        return  TRUE;
    }
}

```

完成したプログラムのソースリスト

```

        }else    return FALSE;
    }

/*****
[関数名] OutGvarList
[ 属性 ] protected
[ 機能 ] リストボックスに変数一覧を表示
*****/
int    CGvarDlg::OutGvarList()
{
    Id    id("i");
    CString name;    complx    value;

    SendDlgItemMessage( IDC_GVAR_LIST, LB_RESETCONTENT, 0,0 );
    int    loop = 0;
    while( id.GetGvarData( loop, &name, &value ) == TRUE ) {
        SendDlgItemMessage( IDC_GVAR_LIST, LB_INSERTSTRING, loop,
            (LONG)(LPSTR)(const char *) name );
        loop++;
    }
    return loop;
}

////////////////////////////////////
// CGvarDlg メッセージ ハンドラ

/*****
[関数名] OnInitDialog
[ 属性 ] protected
[ 機能 ] ダイアログの初期化
*****/
BOOL CGvarDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    // TODO: この位置にその他の初期化用コードを追加してください。

```


完成したプログラムのソースリスト

```

        OutGvarList();
        return TRUE;
    }

/*****
[関数名] OnSelchangeGvarList   (リストボックスの選択変更)
[ 属性 ] protected
[ 機能 ] 選択されている変数の値を表示
*****/
void CGvarDlg::OnSelchangeGvarList()
{
    // TODO: この位置にコントロール通知ハンドラのコードを追加してください
    const FLOAT_MAX = 10;
    if( GetListBoxText( pBuff ) == TRUE ){
        Id      id( pBuff );
        complex value = id.eval();
        _gcv( value.Real(),  FLOAT_MAX, pBuff );
        m_EditBox_Real = pBuff;
        _gcv( value.Image(), FLOAT_MAX, pBuff );
        m_EditBox_Image = pBuff;

        m_Info = id.GetGvarInfo();
    }
    UpdateData( FALSE );
}

/*****
[関数名] OnGvarAssign   (アサインボタン・クリック)
[ 属性 ] protected
[ 機能 ] 変数の値（実数、虚数）エディットボックスの値を変数に代入。
         定数の場合は出来ない。
*****/
void CGvarDlg::OnGvarAssign()
{
    // TODO: この位置にコントロール通知ハンドラのコードを追加してください

```

完成したプログラムのソースリスト

```

        if( GetListBoxText( pBuff ) == TRUE ) {
            Id      id( pBuff );
            char    *p1, *p2;
            UpdateData( TRUE );
            complx  value(  strtod( (const char *)m_EditBox_Real, &p1 ),
                           strtod( (const char
*)m_EditBox_Image,&p2 )          );

            if( *p1 == '¥0' && *p2 == '¥0' ) {
                if( id.GetAttrib() == GVAR )
                    id.set( value );
                // 代入する
                else {
                    CString  UnitStr = id.GetGvarInfo();
                    UnitStr += " には値を代入出来ません。";
                    MessageBox( UnitStr, "エラー" );
                    // 代入不可
                }
            }
        }
    }else {
        m_EditBox_Real.Empty();          m_EditBox_Image.Empty();
    }
    OnSelchangeGvarList();
}

/*****
[関数名] OnOK   (OKボタン・クリック)
[ 属性 ] protected
[ 機能 ] pBuff に選択された変数名を格納して終了
*****/
void CGvarDlg::OnOK()
{
    // TODO: この位置にその他の検証用のコードを追加してください
    GetListBoxText( pBuff );
    CDialog::OnOK();
}

```

完成したプログラムのソースリスト

```

}

/*****
[関数名] OnDbclckGvarList( リストボックス・ダブルクリック )
[ 属性 ] protected
[ 機能 ] OnOK を呼び出す
*****/
void CGvarDlg::OnDbclckGvarList()
{
    // TODO: この位置にコントロール通知ハンドラのコードを追加してください
    OnOK();
}

/*****
[関数名] OnGvarDelete() (削除ボタンクリック)
[ 属性 ] protected
[ 機能 ] 選択された変数をリストから削除する。
*****/
void CGvarDlg::OnGvarDelete()
{
    // TODO: この位置にコントロール通知ハンドラのコードを追加してください
    GetListBoxText( pBuff );
    Id      id( pBuff );
    if( id.GetAttrib() == GVAR ) {
        CString UnitStr = "変数 ";
        UnitStr += pBuff;
        UnitStr += " を削除します。";
        if( MessageBox( UnitStr, "変数の削除", MB_OKCANCEL ) == IDOK )
            if( id.DeleteGvar( pBuff ) == TRUE ) OutGvarList();
    }
    else {
        CString UnitStr = id.GetGvarInfo();
        UnitStr += " は削除出来ません。";
        MessageBox( UnitStr, "エラー" );
    }
    // 代入不可
}
}

```

完成したプログラムのソースリスト

```

/*****
[関数名] OnGvarLoad
[ 属性 ] protected
[ 機能 ] 変数情報をロード
*****/
void CGvarDlg::OnGvarLoad()
{
    // TODO: この位置にコントロール通知ハンドラのコードを追加してください
    CFileDialog      FileDlg( TRUE, NULL,NULL, OFN_HIDEREADONLY |
    OFN_OVERWRITEPROMPT, FileFilter2 );
    if( FileDlg.DoModal() == TRUE ) {
        CFile      File( FileDlg.GetPathName(), CFile::modeRead );

        CString  name;
        char      c;
        double    x, y;
        int       FileEnd;
        while(1){
            name.Empty();    x = 0;    y = 0;
            if( File.Read( &c, 1 ) ) {
                name += c;
                while( 1 ) {
                    File.Read( &c, 1 );
                    if( c == '¥0' )break;
                    name += c;
                }
                File.Read( &x, sizeof(double) );
                FileEnd = File.Read( &y, sizeof(double) );

                Id      id( (char *)(const char *)name );
                id.set( complx( x, y ) );

            }else break;
        };
    }
}

```

完成したプログラムのソースリスト

```

        OutGvarList();
    }
}

/*****
[関数名] OnGvarSave
[ 属性 ] protected
[ 機能 ] 変数情報をセーブ
*****/
void CGvarDlg::OnGvarSave()
{
    // TODO: この位置にコントロール通知ハンドラのコードを追加してください
    CFileDialog      FileDlg( FALSE, NULL, NULL, OFN_HIDEREADONLY |
    OFN_OVERWRITEPROMPT, FileFilter2 );
    if( FileDlg.DoModal() == TRUE ) {
        CString  FileName = FileDlg.GetPathName();
        if( FileDlg.GetFileExt() == "" )FileName += GVAR_FILE;

        CFile      File( FileName, CFile::modeCreate | CFile::modeWrite );
        int loop = 0;
        CString  name;
        complx  value;
        Id      id;
        double   x,y;

        while( id.GetGvarData( loop, &name, &value ) ) {
            Id      id( (char *)(const char *)name );
            if( id.GetAttrib() != GVAR )break;

            x = value.Real();  y = value.Image();
            File.Write( (const char *)name, name.GetLength()+1 );
            File.Write( &x, sizeof(double) );
            File.Write( &y, sizeof(double) );
            loop++;
        }
    }
}

```

完成したプログラムのソースリスト

```
}
```

完成したプログラムのソースリスト

MACEDDLG.cpp

```
// maceddlg.cpp : インプリメンテーション ファイル
//
```

```
#include "stdafx.h"
#include "DT.h"
#include "maceddlg.h"
```

```
#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif
```

```
////////////////////////////////////
// CMacEdDlg ダイアログ
```

```
CMacEdDlg::CMacEdDlg(CWnd* pParent /*=NULL*/)
: CDialog(CMacEdDlg::IDD, pParent)
{
   //{{AFX_DATA_INIT(CMacEdDlg)
    m_MacroName = _T("");
    m_MacroCode = _T("");
    m_MacroInfo = _T("");
   //}}AFX_DATA_INIT
}
```

```
void CMacEdDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    {{{AFX_DATA_MAP(CMacEdDlg)
    DDX_Text(pDX, IDC_EDIT1, m_MacroName);
    DDX_Text(pDX, IDC_EDIT2, m_MacroCode);
    DDV_MaxChars(pDX, m_MacroCode, 512);
    }}}
}
```

完成したプログラムのソースリスト

```
        DDX_Text(pDX, IDC_EDIT3, m_MacroInfo);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CMacEdDlg, CDialog)
    //{{AFX_MSG_MAP(CMacEdDlg)
        // NOTE: ClassWizard はこの位置にメッセージ マップ用のマクロを
        追加します。
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CMacEdDlg メッセージ ハンドラ
```


完成したプログラムのソースリスト

MACRODLG.cpp

// macrodlg.cpp : インプリメンテーション ファイル

//

#include "stdafx.h"

#include "DT.h"

#include "complx.h"

#include "UserMac.h"

#include "MacEdDlg.h"

#include "macrodlg.h"

#include "Data.h"

#ifdef _DEBUG

#undef THIS_FILE

static char BASED_CODE THIS_FILE[] = __FILE__;

#endif

#define ERROR_MESSAGE01 "Macro Name Error !"

////////////////////////////////////

// CMacroDlg ダイアログ

CMacroDlg::CMacroDlg(CWnd* pParent /*=NULL*/)

: CDialog(CMacroDlg::IDD, pParent)

{

//{{AFX_DATA_INIT(CMacroDlg)

m_MacroInfo = _T("");

//}}AFX_DATA_INIT

}

void CMacroDlg::DoDataExchange(CDataExchange* pDX)

完成したプログラムのソースリスト

```

{
    CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CMacroDlg)
    DDX_Text(pDX, IDC_MACRO_INFO, m_MacroInfo);
   //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CMacroDlg, CDialog)
   //{{AFX_MSG_MAP(CMacroDlg)
    ON_BN_CLICKED(IDC_MACRO_DELETE, OnMacroDelete)
    ON_BN_CLICKED(IDC_MACRO_EDIT, OnMacroEdit)
    ON_LBN_DBLCLK(IDC_MACRO_LIST1, OnDbclkMacroList1)
    ON_BN_CLICKED(IDC_MACRO_LOAD, OnMacroLoad)
    ON_BN_CLICKED(IDC_MACRO_SAVE, OnMacroSave)
    ON_BN_CLICKED(IDC_NEWMACRO_EDIT, OnNewmacroEdit)
    ON_LBN_SELCHANGE(IDC_MACRO_LIST1, OnSelchangeMacroList1)
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/*****
[関数名] OutMacroList
[ 属性 ] protected
[ 機能 ] 定義されていりマクロ名をリストボックスに追加
*****/
int CMacroDlg::OutMacroList()
{
    int    loop = 0;
    CString name, code;

    SendDlgItemMessage( IDC_MACRO_LIST1, LB_RESETCONTENT, 0, 0 );
    while( pUserMacro->GetMacroInfo( loop, &name, &code )) {
        SendDlgItemMessage( IDC_MACRO_LIST1, LB_INSERTSTRING, loop,
            (LONG)(LPSTR)(const char *)name );
        loop++;
    }
}

```

完成したプログラムのソースリスト

```

        return    loop;
    }

/*****
[関数名] GetListBoxText
[ 属性 ] protected
[ 機能 ] リストボックス内の選択されているテキストをpBuffに格納
         通常は TRUE を、選択されていない場合は FALSE を返す。
*****/
BOOL CMacroDlg::GetListBoxText( char* pBuff )
{
    int      index = (int)SendDlgItemMessage( IDC_MACRO_LIST1,  LB_GETCURSEL,
0, 0 );
    if( index >= 0 ) {
        SendDlgItemMessage( IDC_MACRO_LIST1,  LB_GETTEXT, index,
( LONG )( LPSTR )pBuff );
        return    TRUE;
    }else    return FALSE;
}

////////////////////////////////////
// CMacroDlg メッセージ ハンドラ

/*****
[関数名] OnInitDialog
[ 属性 ] protected
[ 機能 ] リストボックスにマクロ名一覧を表示
*****/
BOOL CMacroDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    // TODO: この位置にその他の初期化用コードを追加してください。
    OutMacroList();
    return TRUE;
}

```

完成したプログラムのソースリスト

```

/*****
[関数名] OnMacroEdit   (マクロの編集ボタン・クリック)
[ 属性 ] protected
[ 機能 ] 選択されたマクロをマクロ編集ダイアログで編集後、更新する。
         マクロ名が変な時はメッセージボックスを出す。
*****/
void CMacroDlg::OnMacroEdit()
{
    // TODO: この位置にコントロール通知ハンドラのコードを追加してください
    CMacEdDlg    EditDlg;
    if( GetListBoxText(Buff) == TRUE ){
        EditDlg.m_MacroName = Buff;
        pUserMacro->GetMacro( EditDlg.m_MacroName, &EditDlg.m_MacroCode,
&EditDlg.m_MacroInfo );
        if( EditDlg.DoModal() == TRUE ){
            if( pUserMacro->MakeMacro( EditDlg.m_MacroName,
EditDlg.m_MacroCode, EditDlg.m_MacroInfo ) == TRUE )OutMacroList();
            else MessageBox( ERROR_MESSAGE01 );
        }
    }
}

/*****
[関数名] OnNewmacroEdit   (新規編集ボタン・クリック)
[ 属性 ] protected
[ 機能 ] 新しいマクロをマクロ編集ダイアログで編集後、登録する。
         マクロ名が変な時はメッセージボックスを出す。
*****/
void CMacroDlg::OnNewmacroEdit()
{
    // TODO: この位置にコントロール通知ハンドラのコードを追加してください
    CMacEdDlg    EditDlg;
    if( EditDlg.DoModal() == TRUE ){
        if( pUserMacro->MakeMacro( EditDlg.m_MacroName,
EditDlg.m_MacroCode, EditDlg.m_MacroInfo ) == TRUE )OutMacroList();
        else MessageBox( ERROR_MESSAGE01 );
    }
}

```

完成したプログラムのソースリスト

```

    }
}

/*****
[関数名] OnMacroDelete (マクロの削除ボタン・クリック)
[ 属性 ] protected
[ 機能 ] 選択されたマクロを削除する。
*****/
void CMacroDlg::OnMacroDelete()
{
    CMacEdDlg    EditDlg;
    if( GetListBoxText(Buff) == TRUE ){
        CString MacroStr = "マクロ ";
        MacroStr += Buff;
        MacroStr += " を削除します。";
        if( MessageBox( MacroStr, "マクロの削除", MB_OKCANCEL ) == IDOK )
        {
            pUserMacro->DeleteMacro( Buff );
            OutMacroList();
        }
    }
}

/*****
[関数名] OnMacroLoad (LOADボタン・クリック)
[ 属性 ] protected
[ 機能 ] ファイルダイアログで得られたファイル名のファイルから
        マクロ定義を読み込み、追加する。
*****/
void CMacroDlg::OnMacroLoad()
{
    // TODO: この位置にコントロール通知ハンドラのコードを追加してください
    CFileDialog    FileDlg( TRUE, NULL,NULL, OFN_HIDEREADONLY |
OFN_OVERWRITEPROMPT, FileFilter1 );
    if( FileDlg.DoModal() == TRUE ){
        CFile      File( FileDlg.GetPathName(), CFile::modeRead );

```

完成したプログラムのソースリスト

```

// これを有効にすると前回のマクロ情報がキャンセルされます、
//pUserMacro->AllDelete();

CString name, code, info;
int      FileEnd = 1;
while( FileEnd ) {
    ReadString( &File, &name );
    ReadString( &File, &code );
    info.Empty();
    FileEnd = ReadString( &File, &info );
    pUserMacro->MakeMacro( name, code, info );
}
OutMacroList();
}
}

/*****
[関数名] ReadString
[ 属性 ] protected
[ 機能 ] pFileのファイルから1文字列読み込み、pString に格納する。
        ファイルエンドに到達すると 0 を返す。
*****/
int      CMacroDlg::ReadString( CFile* pFile, CString* pString )
{
    char    c;
    int      FileEnd = 1;
    pString->Empty();
    while( FileEnd ) {
        FileEnd = pFile->Read( &c, 1 );
        if( c == '|' )break;
        *pString += c;
    }
    return  FileEnd;
}

```

完成したプログラムのソースリスト

```

/*****
[関数名] OnMacroSave ( SVAE ボタン・クリック)
[ 属性 ] protected
[ 機能 ] ファイルダイアログで得られたファイル名のファイルに
         マクロ定義を書き込む。
*****/
void CMacroDlg::OnMacroSave()
{
    // TODO: この位置にコントロール通知ハンドラのコードを追加してください
    CFileDialog FileDlg( FALSE, NULL, NULL, OFN_HIDEREADONLY |
OFN_OVERWRITEPROMPT, FileFilter1 );
    if( FileDlg.DoModal() == TRUE ) {
        CString FileName = FileDlg.GetPathName();
        if( FileDlg.GetFileExt() == "" )FileName += MACRO_FILE;

        CFile File( FileName, CFile::modeCreate | CFile::modeWrite );
        int loop = 0;
        CString name, code, info;
        while( pUserMacro->GetMacroInfo( loop, &name, &code, &info ) ) {
            File.Write( (const char *)name, name.GetLength() );
            File.Write( "|", 1 );
            File.Write( (const char *)code, code.GetLength() );
            File.Write( "|", 1 );
            File.Write( (const char *)info, info.GetLength() );
            File.Write( "|", 1 );
            loop++;
        }
    }
}

```

```

/*****
[関数名] OnOK ( OK ボタン・クリック)
[ 属性 ] protected

```

完成したプログラムのソースリスト

[機能] 選択されたマクロ名に '\$'を追加して Buff に格納すして
終了。

```

*****/

```

```

void CMacroDlg::OnOK()

```

```

{

```

```

    if( GetListBoxText(Buff+1) == TRUE ){

```

```

        *Buff = '$';

```

```

    }else *Buff = '¥0';

```

```

    CDialog::OnOK();

```

```

}

```

```

/*****

```

[関数名] OnDblclkMacroList1 （ リストボックス・ダブルクリック）

[属性] protected

[機能] OnOkを実行

```

*****/

```

```

void CMacroDlg::OnDblclkMacroList1()

```

```

{

```

```

    OnOK();

```

```

}

```

```

/*****

```

[関数名] OnSelchangeMacroList1 （ リストボックスの選択が変更）

[属性] protected

[機能] 選択されているマクロの機能説明を表示

```

*****/

```

```

void CMacroDlg::OnSelchangeMacroList1()

```

```

{

```

```

    // TODO: この位置にコントロール通知ハンドラのコードを追加してください

```

```

    if( GetListBoxText(Buff) == TRUE ){

```

```

        CString code;

```

```

        pUserMacro->GetMacro( Buff, &code, &m_MacroInfo );

```

```

        UpdateData( FALSE );

```

```

    }

```

```

}

```


完成したプログラムのソースリスト

NODES.cpp

```

#include "Stdafx.h"
#include "complx.h"

#define      _ID_INIT
#include "nodes.h"

#include "Data.h"

/*****
[関数名] look
[ 属性 ] protected
[ 機能 ] 1. ID_SymtabがNULLの時、MakeDefault を呼び出す。
          2. 変数名 nm を変数リストの中から探し、一致する Nament
             のポインタを返す。無い場合は新規に作成する。
*****/
Nament *Id::look(char    *nm)
{
    if( ID_Symtab == NULL)    MakeDefault( ID_Symtab );

    for( Nament    *p = ID_Symtab; p; p = p->next )
        if( p->name.Compare( nm ) == 0)    return  p;

    ID_Symtab = new Nament(nm, ID_Symtab, GVAR );
    return  ID_Symtab;
}

/*****
[関数名] MakeDefault
[ 属性 ] protected
[ 機能 ] 変数リストに定数のデータをUNIT属性(代入禁止)で追加する。
*****/
void    Id::MakeDefault( Nament* &syntab )

```

完成したプログラムのソースリスト

```

{
    int    loop = 0;
    while( *(UnitTable[ loop ].name) ) {
        symtab = new Nament( UnitTable[ loop ].name,  symtab, UNIT,
                               UnitTable[ loop ].value, UnitTable[ loop ].info );
        loop++;
    }
}

/*****
[関数名] GetGvarData
[ 属性 ] public
[ 機能 ] 変数リストの前から index 番目で指定される変数の名前と
         数値を name,value に格納する。
         通常は TRUE を、変数が無い時は FALSE を返す。
*****/
BOOL  Id::GetGvarData( int index, CString* name, complx* value, CString* info  )
{
    Nament*p = ID_Symtab;
    if( p == NULL )return FALSE;
    for( int loop = 0; loop < index; loop++ ) {
        if( (p = p->next) == NULL )return FALSE;
    }
    *name = p->name;      *value = p->value;
    if( info )*info = p->info;
    return TRUE;
}

/*****
[関数名] DeleteGvar
[ 属性 ] protected
[ 機能 ] 変数名 GvarName で指定される変数をリストから削除します。
         通常は TRUE を、削除に失敗すると FALSE を返します。
*****/
BOOL  Id::DeleteGvar( CString GvarName  )

```

完成したプログラムのソースリスト

```

{
    Nament* befour = NULL;

    for( Nament* p = ID_Symtab; p; p = p->next ) {
        if( p->name == GvarName ) {
            if( p->attrib != GVAR ) return FALSE;
            if( befour == NULL ) {
                Nament* Buff = ID_Symtab->next;
                ID_Symtab->next = NULL;
                delete ID_Symtab;
                ID_Symtab = Buff;
                return TRUE;
            }
            befour->next = p->next;
            p->next = NULL;
            delete p;
            return TRUE;
        }
        befour = p;
    }
    return FALSE;
}

/*****
[関数名] Func(コンストラクタ)
[ 属性 ] public
[ 機能 ] FuncIndexで指定される関数に Param1,2,,, で指定される引数
         を与えて実行し、結果をメンバ変数 value に格納する。
*****/
Func::Func( int FuncIndex, Node* Param1 )
{
    value = (*(FuncTable1[ FuncIndex ].pFunc))( Param1->eval() );
    delete Param1;
}

```

完成したプログラムのソースリスト

```
Func::Func( int FuncIndex, Node* Param1, Node* Param2 )
{
    value = (*(FuncTable2[ FuncIndex ].pFunc))( Param1->eval(), Param2->eval() );
    delete  Param1; delete  Param2;
}
```

完成したプログラムのソースリスト

OPTDLG.cpp

```
// optdlg.cpp : インプリメンテーション ファイル
//

#include "stdafx.h"
#include "DT.h"

#include "complx.h"
#include "optdlg.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

//////////////////////////////////////
// COptDlg ダイアログ

COptDlg::COptDlg(CWnd* pParent /*=NULL*/)
    : CDialog(COptDlg::IDD, pParent)
{
   //{{AFX_DATA_INIT(COptDlg)
    m_Float = 0;
    //}}AFX_DATA_INIT
}

void COptDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(COptDlg)
    DDX_Text(pDX, IDC_OPTION_EDIT1, m_Float);
    DDV_MinMaxInt(pDX, m_Float, 0, 15);
    //}}AFX_DATA_MAP
}
```

完成したプログラムのソースリスト

```

}

BEGIN_MESSAGE_MAP(COptDlg, CDialog)
   //{{AFX_MSG_MAP(COptDlg)
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// COptDlg メッセージ ハンドラ

/*****
[関数名] OnInitDialog
[ 属性 ] protected
[ 機能 ] ダイアログの初期化 (各ボタンのセット)
*****/
BOOL COptDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    // TODO: この位置にその他の初期化用コードを追加してください。
    if( m_RadMode )
        ((CButton *)GetDlgItem( IDC_OPTION_RADIO2 ))->SetCheck( 1 );
    else
        ((CButton *)GetDlgItem( IDC_OPTION_RADIO1 ))->SetCheck( 1 );

    if( m_CommaMode )((CButton *)GetDlgItem( IDC_OPTION_CHECK1 ))->SetCheck( 1 );

    UpdateData( FALSE );

    return TRUE;
}

```

完成したプログラムのソースリスト

```

/*****
[関数名] OnOK
[ 属性 ] protected
[ 機能 ] 設定されたデータの更新
*****/
void COptDlg::OnOK()
{
    // TODO: この位置にその他の検証用のコードを追加してください

    if( ((CButton *)GetDlgItem( IDC_OPTION_RADIO2 ))->GetCheck() )m_RadMode = 1;
    else      m_RadMode = 0;
    if( ((CButton *)GetDlgItem( IDC_OPTION_CHECK1 ))->GetCheck() )m_CommaMode
= 1;
    else      m_CommaMode = 0;

    UpdateData( TRUE );
    CDialog::OnOK();
}

```

完成したプログラムのソースリスト

STDAFX.cpp

```
// stdafx.cpp : 標準インクルードファイルを含むソースファイル
//      DT_pch : 生成されるプリコンパイル済ヘッダー
//      stdafx.obj : 生成されるプリコンパイル済タイプ情報

#include "stdafx.h"
```


完成したプログラムのソースリスト

USERMAC.cpp

```

/*=====
    CUserMacro クラス
=====*/

#include "Stdafx.h"
#define      _INIT_MACRO
#include "UserMac.h"

/*****
[関数名] CheckName
[ 属性 ] protected
[ 機能 ] マクロ名が正しいかどうかチェック
*****/
BOOL  CUserMacro::CheckName( CString Name )
{
    char*   p = (char*)(const char*)Name;
    if( isalpha( *p ) == 0 )return FALSE;

    p++;
    while( *p ){
        if( iscsym( *p ) == 0 )return FALSE;
        p++;
    }
    return  TRUE;
}

/*****
[関数名] Find
[ 属性 ] protected
[ 機能 ] マクロリストの中から name と同じ名前のマクロをさがして
        ポインタを返す。無い場合は NULL を返す。
*****/
CUserCode*  CUserMacro::Find( CString name )

```

完成したプログラムのソースリスト

```

{
    for( CUserCode* p = Macro_Symtab.pBefour; p; p = p->pBefour ){
        if( p->Name == name )return p;
    }
    return NULL;
}

/*****
[関数名] MakeMacro
[ 属性 ] protected
[ 機能 ] name と同名のマクロがある場合は code, info で更新し、
        無い場合は新規に作成する。
        通常は TRUE を返し、名前チェックにひつかかると FALSE を返す
*****/
BOOL CUserMacro::MakeMacro( CString name, CString code, CString info )
{
    CUserCode* p;
    if( CheckName(name) == TRUE ){
        if( (p = Find(name)) == NULL ){
            Macro_Symtab.pBefour = new CUserCode( name, code, info,
Macro_Symtab.pBefour );
        }else {
            p->Code = code;
            p->Information = info;
        }
        return TRUE;
    }
    return FALSE;
}

/*****
[関数名] GetMacro
[ 属性 ] protected
[ 機能 ] name で指定されるマクロのコード部、機能説明を code, info
        に格納する。通常は TRUE を、マクロが見つからない時は
        FALSE を返す。、

```

完成したプログラムのソースリスト

```

*****/
BOOL  CUserMacro::GetMacro( CString name, CString* code, CString* info )
{
    CUserCode*    p;
    if( (p = Find(name)) ) {
        *code = p->Code;
        if( info != NULL ) *info = p->Information;
        return  TRUE;
    }else return  FALSE;
}

```

```

/*****
[関数名] DeleteMacro
[ 属性 ] protected
[ 機能 ] name で指定されるマクロをリストから削除する。
        通常は TRUE を、マクロが見つからない時は FALSE を返す。、

```

```

*****/
BOOL  CUserMacro::DeleteMacro( CString name )
{
    CUserCode*    p  = Macro_Symtab.pBefour;
    CUserCode* pBack = &Macro_Symtab;

    while( p ) {
        if( p->Name == name ) {
            pBack->pBefour = p->pBefour;
            p->pBefour = NULL;
            delete  p;      return  TRUE;
        }
        pBack = p;      p = p->pBefour;
    }
    return  FALSE;
}

```

```

/*****
[関数名] GetMacroInfo
[ 属性 ] protected

```

完成したプログラムのソースリスト

[機能] マクロリストの頭から `index` 個目のマクロ情報を `name`,
`code`, `info` に格納する。通常は `TRUE` をマクロが無い時は
`FALSE` を返す。

```

*****/
BOOL  CUserMacro::GetMacroInfo( int index, CString* name, CString* code, CString*
info )
{
    CUserCode*  p = &Macro_Symtab;
    for( int loop = 0; loop <= index; loop++ )
        if( p = p->pBefour ) == NULL )return FALSE;
    *name = p->Name;          *code = p->Code;
    if( info != NULL )*info = p->Information;
    return  TRUE;
}

```