有効数字を表示する電卓の作成

加 藤 初 儀*・川 上 光 博†

A calculator to display significant digits

Hatsuyoshi KATO, Mitsuhiro KAWAKAMI

概 要

本校の「マルチメディア教育システム」導入に伴い，物理科目においてもパソコンを利用した教材作成に取り組むこととなった。その第一歩として，四則演算の計算過程と誤差伝搬を筆算形式で表示できる電卓の作成を試みた。第一学年の学生に対して視覚的に誤差伝搬及び有効数字の基本的概念の理解させ，物理講義の内容を補うことを目的とする。

なお，この電卓はマイクロソフト社製の Visual C ＋＋ 5.0 を用い，DOS/V マシン上でコーディングされている。

Abstract

With introducing "multimedia education system" at our college, we are making materials for the physics education on a personal computer. As the first step, we have tried to make a calculator which displays calculation processes of computations and error propagations like on paper for the four basic arithmetics. The calculator helps the first grade students visually understand the error propagations and the basic concept of the significant digits. In coding the calculator, we use Microsoft(R) Visual C ＋＋ 5.0 on a DOS/V machine.

1．はじめに

現在，本校一般教科におけるカリキュラムの物理の単位数は，第１学年に２単位，第２学年で３単位である。特に，第１学年では数学等のカリキュラムに於いて，物理の基本的計算技術の講義が並行して行われているため，物理の時間で直ちに教科書に従って講義を行うことができない状況にある。このため，第１学年のはじめの講義では，四則演算に対する有効数字の取り扱いに密接に関連する計算誤差伝搬を取り上げている。

四則演算自体は小学校から中学校卒業までの各段階で，単純な数値による運算方法から代数的な取り扱い方法までの基本的事項を一通り各学生が学んでいる。しかし，学生達は特別の注意を払った反省もなく，経験的に計算が実行できるようになっている者が大多数である。このため，有効数字や有効桁の基本的概念に慣れておらず，その修得に多くの時間を要する学生達が多い。

以上の状況のため，誤差伝搬による四則演算結果の正確な誤差の取り扱い[１]に慣れるための一助として「誤差電卓」の作成を DOS/V マシン上でマイクロソフト社の Visual C ＋＋ 5.0 を用いて作成中である[２，３]。

2．通常電卓の作成

誤差の計算を行うには当然，四則演算を行える簡単な電卓が必要となってくる。そこで先ず電卓の作成から開始した。図１が通常電卓の外観である。ごく一般的な電卓の機能を参考に，四則演算及び平方根[√‾]，逆数[1/X]，指数表記変換[EXP]，サインチェンジ[＋/－]ボタンを付加した。メモリ計算機能や，関数電卓のように様々な関数計算を行う機能は持っていないが，この電卓の目的である「計算による誤差伝搬を視覚的に判りやすく表示する」ための最低限の機能は盛り込

＊ 助教授 一般教科
† 技 官 （一般教科）

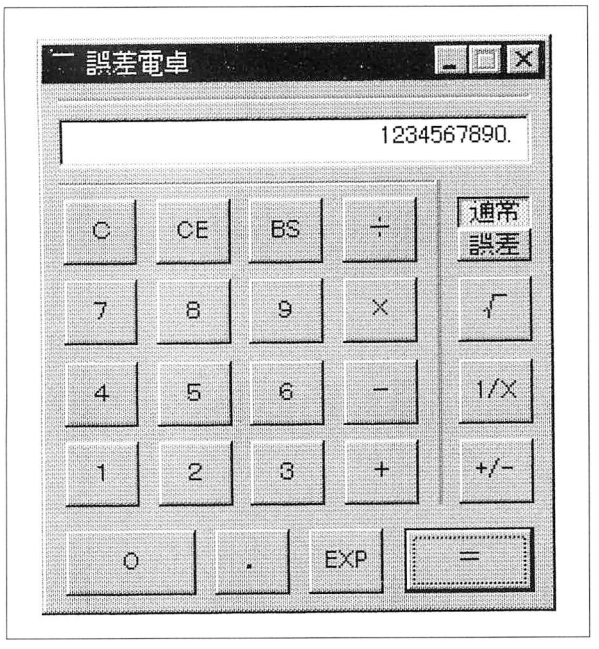

図1　通常電卓の外観

まれている。

電卓の表示部はエディットボックス，通常電卓と誤差電卓との切換はプッシュボタン型のラジオボタン，残りはボタンコントロールを使用している。以下に，各コントロールの簡単な説明を行う。

(a). 数値表示部：エディットコントロールを使用しているが，数値の直接入力は禁止している。表示桁数は最大15桁であり，CString 型メンバ変数(m _text1)を SetWindowText ()関数で表示している。

(b). [通常][誤差]ボタン：ラジオボタンを使用して排他的に択一処理を行い，電卓の種類を選択する。起動時には[通常]が選択されている。

(c). [0]～[9],[.]ボタン：各ボタンに対応した数字または小数点を文字列の引数として入力処理を行う関数に渡し，m _text1に逐次加算していく。

(d). [÷][×][−][+][=]ボタン：四則演算ボタンと等号ボタン。m _text1(文字列)を実数変換して演算処理した後，結果を再度文字列として m _text1に格納し，表示する。

(e). [C]ボタン：電卓によっては[AC]と表記されている場合もある。全てのフラグと計算結果をクリアし初期状態に戻す。

(f). [CE]ボタン：電卓によっては[C]と表記されている場合もある。現在表示されている数値のみをクリアする。

(g). [BS]ボタン：BackSpace。現在入力している値を1文字，元に戻す。戻れるのは現在の数値を入力し始めた時点までである。

(h). [√]ボタン：表示されている数値の平方根を求める。

(i). [1/X]ボタン：表示されている数値の逆数を求める。

(j). [+/-]ボタン：表示されている数値の符号を反転させる。

(k). [EXP]ボタン：表示されている数値を10進指数表記に変換し指数部の入力待ち状態にする。なお，指数部の入力上下限は±306以内としている。これは double 型の値の範囲が1.7E ±307となっているためである。

四則演算は，「最初に入力された値（もしくは現在，計算結果として格納されている値）」と「2番目（直前）に入力された値」の単純な加減乗除であり，演算子の優先順位は考慮していない。これは，一般的な電卓の動作を参考にしたものであるが，ウィンドウズプログラムの利点である「計算式の貼り付け」や，「ドラッグ&ドロップによる数値計算」も逆ポーランド記法を用いて，可能な限り盛り込んでいきたい。

2．誤差電卓の作成

次に，数値計算における計算課程とその誤差伝搬を表示させるための子ウィンドウ（subDlg）を作成した。図2が誤差電卓の外観である。

「計算結果」ダイアログはモードレスダイアログであり，[誤差]ボタン押下により生成・表示し，[通常]ボタン押下により破棄する。コントロールはエディットボックスのみであり，[誤差]ボタンが選択されている限りダイアログの表示を行うようになっている[4]。エディットボックスは CString 型メンバ変数（m _text2）を持ち，これに計算過程と計算結果を文字列として代入し，UpdateData ()関数で更新･表示させるようにした。

計算過程及び計算結果の出力は，普段我々が行う手計算による表記と同じにした。これは計算による誤差伝搬を視覚的に判りやすく表示するように考慮したためで，実際の手計算による誤差（有効数字）の算出にも効果的である。ただし，現時点では，記述方法の似た加減乗算の表記は暫定的に完成してはいるが，除算に関してはその計算表記が大きく異なるため，未実装の状態である。

図3は誤差電卓の大まかなフローチャートである。なお，表1は図3～図6で用いられる主要な変数の一覧である。flag _fc が TRUE（通常）か FALSE（誤差）かで分岐しているが，違いは「計算結果」のサブウィンドウを生成・出力する

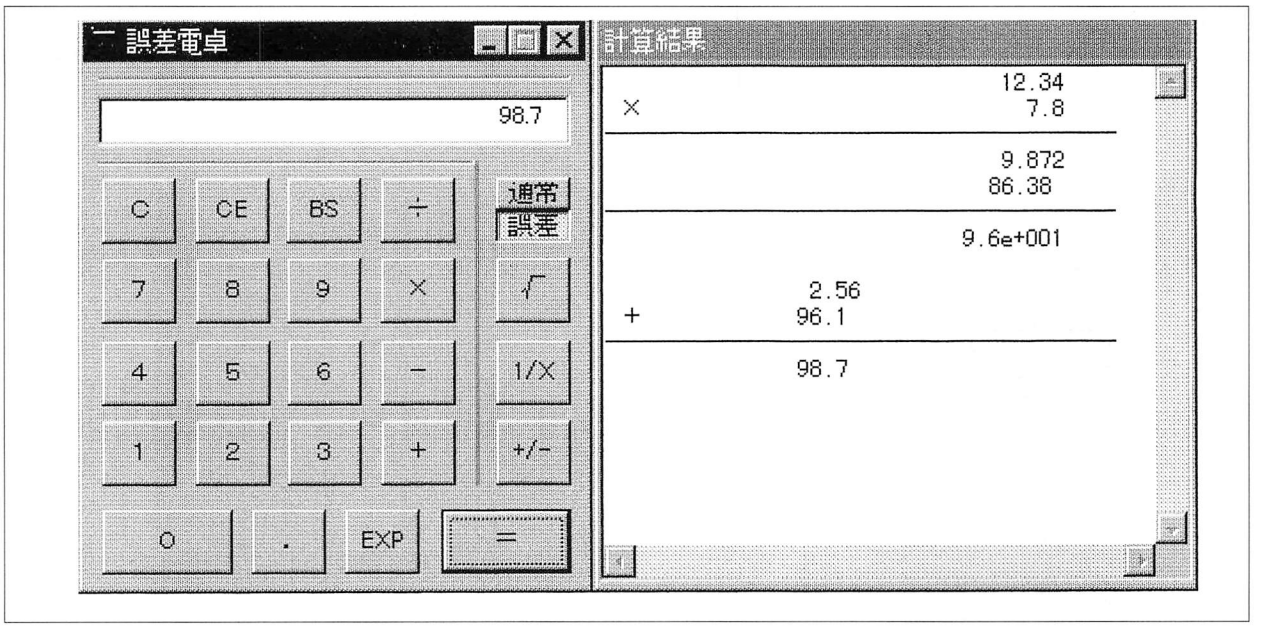

図 2　誤差電卓の外観計算課程及び結果を右側へ出力する

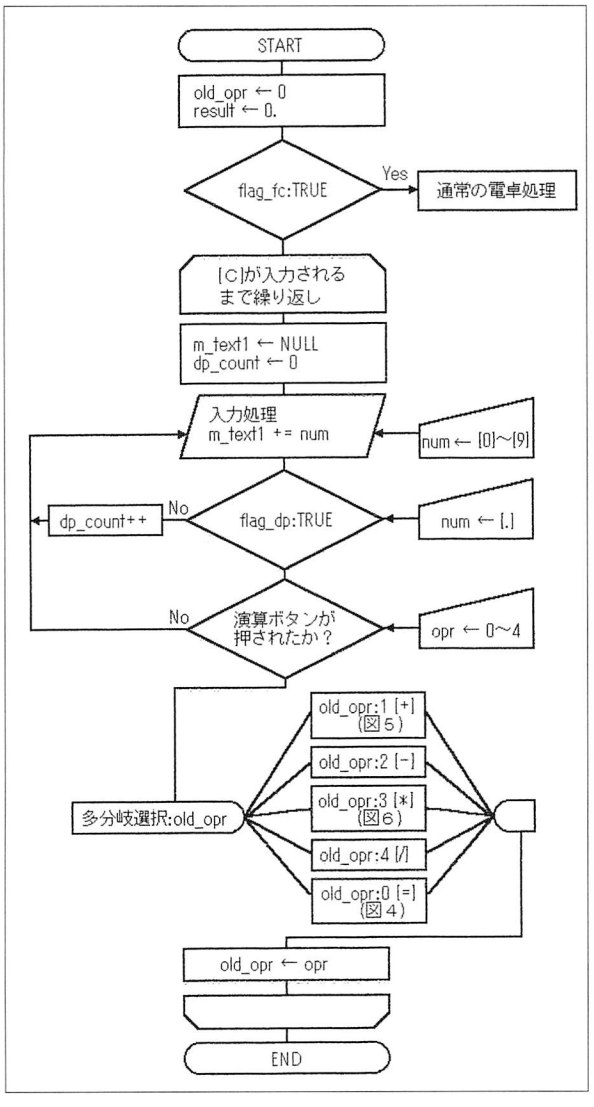

図 3　誤差電卓の全体的な流れ

フローチャートで用いられる変数一覧

引数		
CString	num	[0]〜[9],[.]が押されると入力処理に渡される
int	opr	演算ボタンが押されると演算処理に渡される
グローバル変数		
CString	m_text1	エディットコントロールのメンバ変数
	m_text2	同上
int	g_len1	第1値の全体長
	f_len1	第1値の小数部の長さ
	g_len2	第2値の全体長
	f_len2	第2値の小数部の長さ
	dp_count	入力時に小数部をカウント
	old_opr	演算子のスタック
	flag_fc	電卓の種類:[通常]=TRUE,[誤差]=FALSE
	flag_dp	小数点を押したか？:Yes=FALSE,No=TRUE
double	result	m_text1を実数変換して得られた計算結果
ローカル変数		
CString	dm_text2	計算過程や結果を一時的に格納
char	wk_str[24]	第2値を桁ごとの整数に分解するための配列
int	n	カウンタ
	prec	出力精度(小数部の桁数)
	width	有効桁数

表 1　フローチャートで用いられる主要な変数一覧

かどうかだけである。処理自体はほぼ同様なフローのため，「通常の電卓処理」の部分は省略している。一連の計算は[C]ボタンが押下されるまで繰り返され，数値を入力した後，いづれかの演算ボタンが押されることで old _opr の値により分岐する。 old _opr は前回押された演算ボタンであり初期値は 0 [=]である。分岐によって各演算処理が行われた後， old _opr に新しい演算子をスタックしてループする。

3．加算における誤差伝搬

図 3 における加算部分のフローチャートを図 5 に示す。なお説明が前後するが，第 1 値の入力直後は加減乗除の選択に関わらず， old _opr の初

期値により等号処理が必ず行われる。そのフローチャートが図4である。例えば今，最初に入力した値（以後，aとする）と2番目に入力した値（以後，bとする）の加算によって得られた演算結果をcとして，

$$a + b = c$$

を考える。aは小数部の桁数がf_len1からなり，

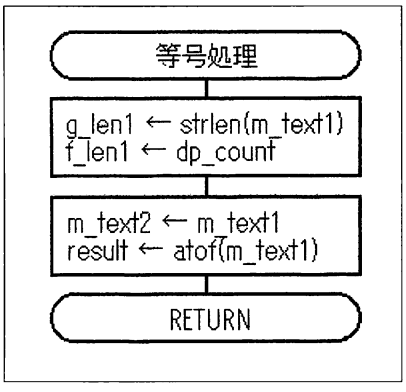

図4　等号処理

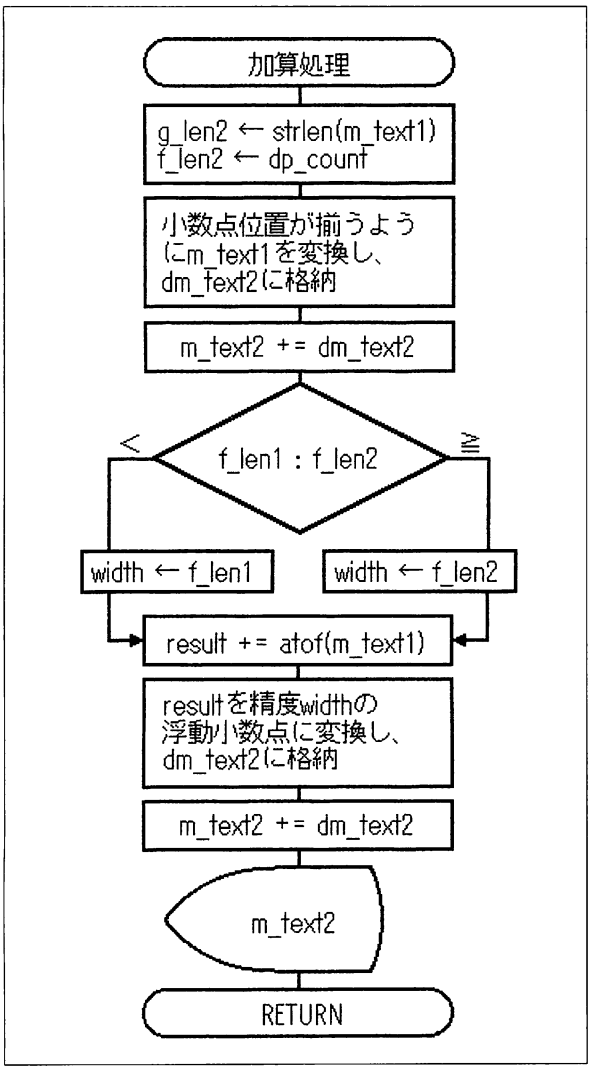

図5　加算処理

bは小数部の桁数がf_len2からなるとすると，演算結果cの有効数字における末尾の桁位は，f_len1とf_len2を比較してより小さい方の値となるので，それより下位の桁は誤差を持つとする。よって，f_len1とf_len2を比較してより小さい方の値を精度（width）とし，結演算果cの出力を浮動小数点変換すればよい。なお，現時点では各値の小数点位置をaの位置に揃えるよう最小フィールド幅を調整しているため，aの桁数によっては余白が空き過ぎてしまう場合がある。そこで，aとbそれぞれの桁数を計算して余白が適正な幅になるよう工夫する必要がある。

4．乗算における誤差伝搬

図3における乗算部分のフローチャートを図6に示す。

先の加減算と同じく，aとbの乗算によって得られた演算結果をcとして，

$$a \times b = c$$

を考慮する。aは全桁数（g_len1），小数部の桁数（f_len1）からなり，bは全桁数（g_len2），小数部の桁数（f_len2）からなるとすると，演算結果cの有効数字（width）は，g_len1とg_len2を比較して小さい方の値（またはその－1）となる筈である。そこで演算結果の出力を指数変換表記とし，仮数部の桁数を（width）で表示することとした。また，計算過程の各段はbの各桁ごとにaを乗じたものである。これらの積の小数部の桁数は，aとbそれぞれの小数部の桁数の和に等しく（またはそれ以下に）なるから，精度をf_len1＋f_len2（f_len2は桁ごとにデクリメントする）の浮動小数点に変換して表示すればよい。

aまたはbの一方が指数表記の場合，もう一方も同じく指数表記に変換して計算するようにした方が，計算過程を桁揃えする上で都合が良い。この場合，aとbそれぞれの仮数部についてその全桁数と小数部の桁数を求め，上記と同様にして有効数字を求めるようにした。

ただし，この方法では有効数字を考慮した計算結果を簡単に得ることはできるが，計算過程や計算結果がどの桁まで誤差を含んでいるのかすぐには判断し難い欠点がある。この欠点を解決する方法として，誤差を含む数字は色分けして表示するやり方を検討している。数値ａｂを1次元のchar型配列，計算過程及び計算結果を2次元のchar型配列として保持し，同時に同じサイズの

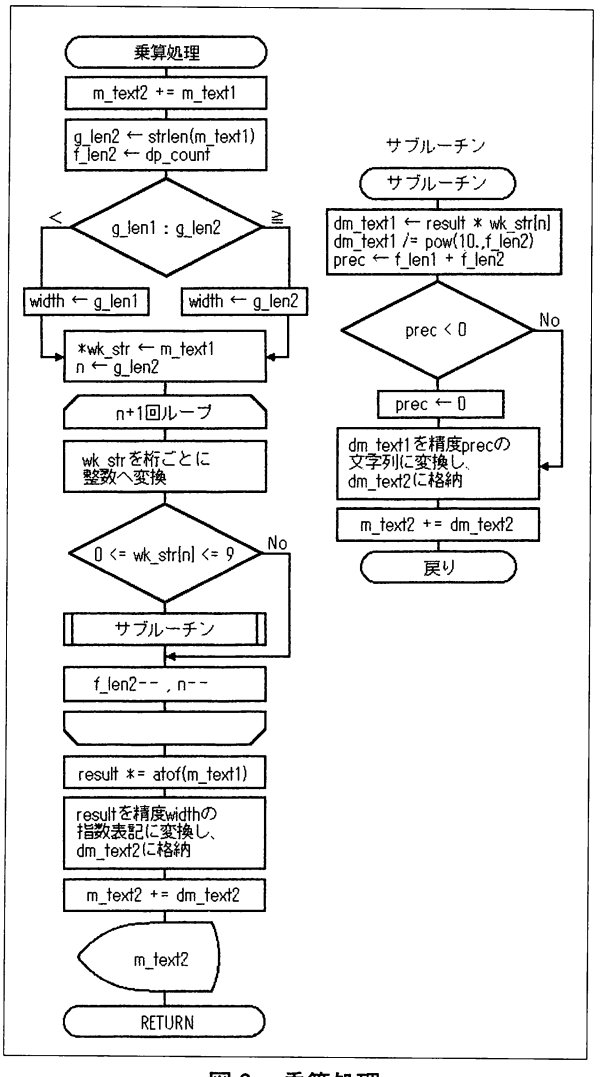

図6　乗算処理

int 型配列をそれぞれ用意して誤差の有無をビット立てする。2次配列：bit _table [Y][X]において，1の部分が誤差を含む桁を表す。

加減算も乗算もともに，誤差を含む桁はビット和で判定できるので，その位置の数字を別の色（赤など）で色分けすれば，視覚的に誤差の位置が判りやすいと思われる。

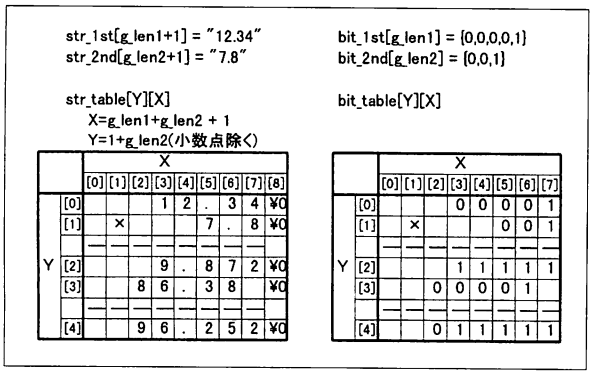

図7　2次配列を用いた誤差伝搬の判別例

5．今後の課題等

誤差電卓は現時点では完成していない部分が多数あり，特に除算に関する計算の表示は他の演算のそれと大きく異なるため，そのアルゴリズム作成に思考錯誤を繰り返している。また本文中にも指摘した問題点を解決して早急にプログラムを完成させる予定である。

なお，以上に述べてきた Visual C ++ 5.0 での基本的な言語概念は，高度に体系化された MFC ライブラリを用いてコーディングするように設計されている。オブジェクト指向言語である C ++ とウィンドウを細部まで操作可能とする機能を有している。しかし，ユーザー側とのインターフェースを簡潔にするために，コーディングを行うプログラマー側の負う労力が大きい。このため，C 言語を既知のものとしても，本電卓の作成のまでには半年以上の時間を要した[5，6]。特に，システムとプログラマー間のコーディングに区別がないことは，上級者にとって有益であるが入門者には戸惑うことが多かった。また，多数の MFC ライブラリの関数を駆使することが容易でない。これより，Visual C ++ 5.0 に対して，プログラマーの負担を軽減する工夫を期待する。

参考文献

[1] 一瀬正巳，「誤差論」，培風館（1993）.
[2] 林晴比古，「新 Visual C ++ 5.0 入門 ビギナー編」，ソフトバンク（1998）.
[3] 林晴比古，「新 Visual C ++ 5.0 入門 シニア編」，ソフトバンク（1998）.
[4] Donald Kato，「標準 Visual C ++」，技術評論社（1998）.
[5] マグロウヒル編集部，「詳説 Turbo C ++ for Windows」，マグロウヒル（1994）.
[6] 柴田望洋，「C プログラマーのための C ++ 入門」，ソフトバンク（1997）.

（平成10年11月25日受理）