ExcelVBA 用日本語ライブラリの開発

森　　重　雄*

Development of the Japanese program library for Excel VBA programming

Shigeo MORI

要　旨

ExcelVBA 用の日本語ライブラリを開発した。この日本語ライブラリは，Excel 環境下で動作する応用プログラムの開発を容易にするもので，すべての関数とプロシジャを日本語名で呼び出せる。これにより，初心者のみならず熟練者でも扱いやすく，また保守性の高い応用プログラムを開発できるようになる。Excel は非常に豊富な機能をもつが，逆にそれが障害となってエンドユーザによる利用を阻害している。日本語ライブラリはこれを改善することを狙った，エンドユーザ向けのライブラリである。

本報告では，この日本語ライブラリ開発の背景，開発方針，構成，実現方法，効果などについて述べる。

1．はじめに

企業内の情報処理は，販売管理や生産管理などの基幹業務における定型処理と，現場業務に密接に関連する非定型業務に分類できる。基幹業務の機械化はどの企業においても一応完成し，システム開発部門はその管理改善と保守を中心として取組んでいる。しかし，現場業務における非定型業務は機械化が難しく，システム開発部門の余力もないことから機械化が遅れ，経営の効率化を阻害している。

非定型業務の機械化は，現場のノウハウと密接に関係するので，業務に詳しい現場担当者が担当することが望ましい。しかし，現場担当者はプログラミングを専門の仕事とせず，また多忙の中でプログラミングをする余裕もない。

一方教育分野では，それぞれの専門分野において学生の理解を促すための，情報技術を応用した教材を必要としているが，市販の教材は経済的または適用上問題があって，自前での開発を迫られている。しかし，多忙な業務の中で，それらの開発をしている余裕はない。

筆者は，これらのユーザができるだけ簡単に利用できる，開発用プラットフォームを模索した。そして，表計算機能をもつ Excel と，その表と連動してプログラミングが可能な ExcelVBA の組み合わせが最適と考え，これらを基盤とした日本語ライブラリの開発を進めてきた。この度，開発目標に対して一応の充足をみたのでこれについて報告する。

＊　教　授　情報工学科

2．開発の背景

2．1　日本語プログラミングの必要性

日本語によるプログラミングの必要性をまとめると次のようになる。

(1)　連想の容易性

漢字の情報量は英字に比べて非常に多い。漢字をプログラムの中で記述できるということは，プログラムに文書としての機能をもたせることになる。エンドユーザは，毎日プログラムを組むわけではない。作成したプログラムの修正を，作成者が忘れた頃に行わなければならない。本業が別にあるから仕様書を書く余裕はない。プログラムが漢字で書かれていれば，記憶を呼び戻し，仕様を連想することが容易になる。

(2)　名称決定の容易性

プログラムを作成する場合，変数名や手続き名，関数名などの名称を決定しなければならない。英字でしか定義できない言語系では，プログラマは日本語名の英訳，またはローマ字化に頭を悩ます。時には辞書を引いて調べる。ここでの検討時間および調査時間はプログラミングの生産性を低下させる。

(3) アルゴリズム展開の容易性

日本語でのプログラミングでは思考結果を同じ言葉で表現できるので，アルゴリズムの組立ておよび展開が極めて容易である。これは，バグが少なく，保守の容易なプログラムの開発を促す。バグが発生してその原因を究明する場合も，英字のプログラムよりは探しやすい。

(4) 伝承の容易性

時には，あるユーザが作成したプログラムを他のユーザに引継がなければならないこともある。その場合，後継者にその内容を教え，後継者はそれを理解しなければならない。このとき，そのプログラムが日本語で書かれていれば，技術の伝承が容易になる。

2．2 開発基盤としての表計算ソフトウェア

最近の表計算ソフトウェアは，応用プログラムの開発基盤としての高度な機能を備えている。

(1) 応用プログラム開発環境の提供

表計算ソフトウェア自身が，プログラミング言語，エディタ，デバッグ機能，グラフィカルユーザインタフェース機能，ヘルプ機能，外部インタフェース機能などの豊富かつ高度な機能をもつ開発環境を提供している。普通の開発言語と同等以上の機能を有しながら，普通の開発言語よりも低価格で販売されているため，経済的にも導入が容易である。

なお，表計算ソフトウェアのプログラミング言語はマクロ言語とも呼ばれている。

(2) ビジュアルかつ動的な表示が可能

応用プログラムの出力先として表計算ソフトウェアのシートを使用できる。このため，罫線描画，色付け，文字フォントの属性設定，スクロール，強調，グラフ表示など，視覚に訴えた表示ができる。また，時系列的に情報を変化させて表示することもできる。

(3) 表計算ソフトウェア自身の編集機能を流用可

応用プログラムに渡すデータを，利用者が表計算ソフトウェア自身の編集機能を使って作成できる。また，その値を変更して再実行させることもできる。このため，ファイル変換のような処理が不要となる。また，入力したデータが即処理されるため，情報のターンアラウンドが速くなる。

また，得られた結果などを別のシートやファイルへ複写でき，条件を変えたときの結果の比較ができることや，途中の結果を履歴として保存できるなどの利点がある。

(4) アルゴリズム実現の容易性

ワークシートのセルデータは，二次元の添え字で参照できるので，従来のファイル操作が不要である。また，ワークシートの関数を利用できるので，これと組合わせることによってアルゴリズムを簡略化できる。

(5) 日本語のサポート

ExcelVBA の場合，応用プログラム開発のために日本語をサポートしている。変数名の他，サブプロシジャ名，関数名，各種のオブジェクト名に日本語を使用できる。これは他の開発言語にない大きな特長であり，エンドユーザ向けの開発基盤として最適なものと評価できる。

2．3 ExcelVBA の問題

Excel を応用プログラムの開発基盤とする場合，開発者は表計算の機能，VBA の言語仕様，および Excel 独自のインタフェース仕様を理解しなければならない。しかし，これには以下の問題がある。

(1) オブジェクト指向

ExcelVBA はオブジェクト指向の言語である。オブジェクト指向の採用によって，プログラムの再利用度が高まり，開発生産性が向上する。しかし，ExcelVBA を利用者する側にとっては，この理解のために，従来の非オブジェクト指向言語を習得するより数倍の労力を強いられる。また，ステートメントの記述量がどうしても多くなる。これでは生産性向上に逆行する。

(2) 膨大な機能

ExcelVBA の機能は非常に多い。開発するプログラムにはこれらのどの機能を使用すればよいのか，経験者でも迷うことがある。

(3) 応用の難しさ

目的の処理を実現するために，ExcelVBA の複数の機能を組合わせなければならない。また，ExcelVBA の個々の機能の前提条件を把握し，その条件に合った使い方をしなければならない。

このため，初心者が応用するには非常に難しい。

(4) 不親切な解説書

市販の多くの解説書やExcelVBAのオンラインヘルプ機能は，ExcelVBA開発者の視点でまとめられていて，応用プログラムおよび利用者の観点からは説明されていない。

使用例がなかったり，あっても参考にならないなど，非常に不親切である。

(5) 英語

ExcelVBAのステートメント，オブジェクト，プロパティ，関数，組込み定数，イベントなど，予約語はすべて英語である。日常の仕事や生活で英語を使用しない利用者にとっては，この理解が負担となる。また，経験者にとっても，実現したいことに対応する命令などを連想することが難しい。

3．ExcelVBA日本語ライブラリ開発方針

筆者はExcelVBA日本語ライブラリを以下の方針で開発した。

3．1　日本語重視

(1) 各種の名称を日本語とする。
(2) 連想しやすい名称を設定する。
(3) ソースリストを文章化できること。

3．2　利用者重視

(1) 説明書を完備すること。
(2) プロシジャや関数の使用例を掲載すること。
(3) プロシジャや関数の検索を容易にすること。

3．3　実務利用重視

(1) 実現したい機能を主眼として開発すること。
(2) ExcelVBAをブラックボックス化すること。

3．4　生産性と保守性重視

(1) プロシジャや関数を汎用化すること。
(2) 呼び出し方の自由度が高いこと。
(3) 呼び出し方に一貫性があること。
(4) 機能の拡張に対応できること。

4．日本語ライブラリの構成

日本語ライブラリは，次の複数のモジュールで構成される。本稿執筆時点の登録本数は約230本である。

(1) 変数操作モジュール
(2) 文字列操作モジュール
(3) セル操作モジュール
(4) 行列番号操作モジュール
(5) ブック操作モジュール
(6) シート操作モジュール
(7) ファイル操作モジュール
(8) グラフ操作モジュール
(9) 数値操作モジュール
(10) 数式操作モジュール
(11) GUI操作モジュール
(12) 画面操作モジュール
(13) メニュー操作モジュール
(14) 日付時間操作モジュール
(15) データ操作モジュール
(16) 印刷操作モジュール
(17) HTML操作モジュール
(18) NTコマンド生成モジュール
(19) 図形操作モジュール

5．呼び出し規約

日本語ライブラリのプロシジャや関数を呼び出す場合の形式は次のとおりである。

5．1　プロシジャの呼び出し

プロシジャ名　引数1，引数2，…，引数n

5．2　関数の呼び出し

(1) 書き方1（戻り値を受取る）
戻り値＝関数名（引数1，引数2，…，引数n）

(2) 書き方2（戻り値を参照）
関数名（引数1，引数2，…，引数n）

(3) 書き方3（戻り値がオブジェクト）
関数名（引数1，引数2，…，引数n）．プロパティ
関数名（引数1，引数2，…，引数n）．メソッド

(4) 書き方4（戻り値を参照せず）
関数名引数1，引数2，・・・，引数n

5．3　引数の省略

呼び出すときの引数の省略形を次のように定義した。

(1) 途中の引数の省略

定義：呼び出し名　引数１，[引数２]，引数３
実行：呼び出し名　引数１，,引数３

定義：呼び出し名　引数１ [,引数２] [,引数３]
実行：呼び出し名　引数１，,引数３

(2) 後部の引数の省略

定　義：呼び出し名　引数１[,引数２][,引数３]
実行１：呼び出し名　引数１，引数２
実行２：呼び出し名　引数１

(3) 全引数の省略

定　義：呼び出し名　[引数１] [,引数２] [, 引数３]
実行：呼び出し名

６．日本語ライブラリの特長

６．１　呼び出し時の引数省略化

５．３に示すように，日本語ライブラリの関数等を呼び出すときに，引数を省略できるようにしている。省略された引数について，呼ばれる側ではその仮定値を設定する。

これにより，引数の値が仮定値であれば，その引数の指定を省略でき，プログラミングが容易になる。

ところで,呼ばれる側で仮定値を設定するには，引数が省略されているかを判定しなければならない。そこで，引数が省略されているかを判定し，省略されていれば仮定値を返し，省略されていなければその変数の値をそのまま返す関数を開発した。この関数により，５行で定義しなければならない処理を1行で定義できるようになった。

６．２　呼び出しの自由度向上

一つのプロシジャや関数において，複数の呼び出し方法をサポートした。これによって，変数の型をあまり意識しなくてもよくなり，呼び出し時の自由度が向上した。

たとえば，ワークシートのセル（"A1"）を特定するには以下の三つの方法がある。

Range ("A1")
Cells (1, 1)
Cells (1,"A")

利用者は，アドレスの指定の仕方によって，プロパティを選択し，それらに応じた引数を渡さなければならない。

日本語ライブラリでは，以下のような呼び出し形式を可能にした。

呼び出し名　"A1"
呼び出し名　1, 1
呼び出し名　1,"A"
呼び出し名　レンジオブジェクト

リスト１　引数省略時の仮定値設定関数の例

```
'---------- 引数によるオブジェクト設定関数 ----------
Function 引数によるオブジェクト設定(引数 As Variant, 仮定値 As Variant) As Variant
    If TypeName(引数) = "Error" Then
            Set 引数によるオブジェクト設定 = 仮定値          '呼び出し側の，ひとつ前の呼び出し側がその引数を省略している場合
        Else
            Set 引数によるオブジェクト設定 = 引数
    End If
End Function


'-------  引数によるオブジェクト設定関数を使用しない場合 ----------
Function セル値取得B(引数1 As Variant, Optional 引数2 As Variant, Optional 引数3 As Variant) As Variant
    If TypeName(引数1) = "String" Then
            If IsMissing(引数2) Then
                    セル値取得B = ActiveSheet.Range(引数1).Value
                Else
                    セル値取得B = 引数2.Range(引数1).Value
            End If
        Else
            If IsMissing(引数3) Then
                    セル値取得B = ActiveSheet.Cells(引数1, 引数2).Value
                Else
                    セル値取得B = 引数3.Cells(引数1, 引数2).Value
            End If
    End If
End Function


'---------- 引数によるオブジェクト設定関数を使用した場合 ----------
Function セル値取得B(引数1 As Variant, Optional 引数2 As Variant, Optional 引数3 As Variant) As Variant
    If TypeName(引数1) = "String" Then
            セル値取得B = 引数によるオブジェクト設定(引数2, ActiveSheet).Range(引数1).Value
        Else
            セル値取得B = 引数によるオブジェクト設定(引数3, ActiveSheet).Cells(引数1, 引数2).Value
    End If
End Function
```

6．3　文書化指向プログラミング

ソースプログラム上で見たライブラリのプロシジャ名や関数名が，一連の手続きの中で文章的に捉えられるように設定した。

たとえば，対象のワークシートがある場合に注目して検査する場合，

```
If シートあり(シート名) Then
        シートがあるときの処理
    Else
        シートがないときの処理
    End If
```

また，対象のワークシートがない場合に注目して検査する場合，

```
If シートなし(シート名) Then
        シートがないときの処理
    Else
        シートがあるときの処理
    End If
```

という記述を可能にした。

さらに，操作対象を明確に把握できる名称設定を行った。たとえば，セルの値や属性すべてを複写する場合は，

```
セル複写
```

セルの値を複写する場合は，

```
セル値複写
```

とした。

6．4　記述量削減

セル操作などでは，ある処理を行うときに，それに関連した処理を続けて行うことが多い。たとえば，セルに数値を格納する場合，書式，配置，文字の色，フォントの種類などを続けて設定する。その場合，ExcelVBA の表現形式で記述すると記述行数が大幅に増える。

日本語ライブラリでは，引数を増やすことによって，関連する処理も一つの呼び出しの中で行えるようにした。これにより，プログラムの記述量が削減される。

6．5　数値計算における誤差への対応

浮動小数を扱う数値計算では，必ず誤差の問題が発生する。したがって，数値同士の比較においては，誤差を考慮しなければならない。また，数値の絶対値がある値より小さければ，ゼロとみなすことも必要である。

日本語ライブラリでは，浮動小数の数値を扱うプロシジャや関数には誤差の基準も引数として加えるようにした。また，誤差を考慮した比較関数や，引数の変数の絶対値が引数の誤差の値より小さい場合にゼロを返す関数を開発した。

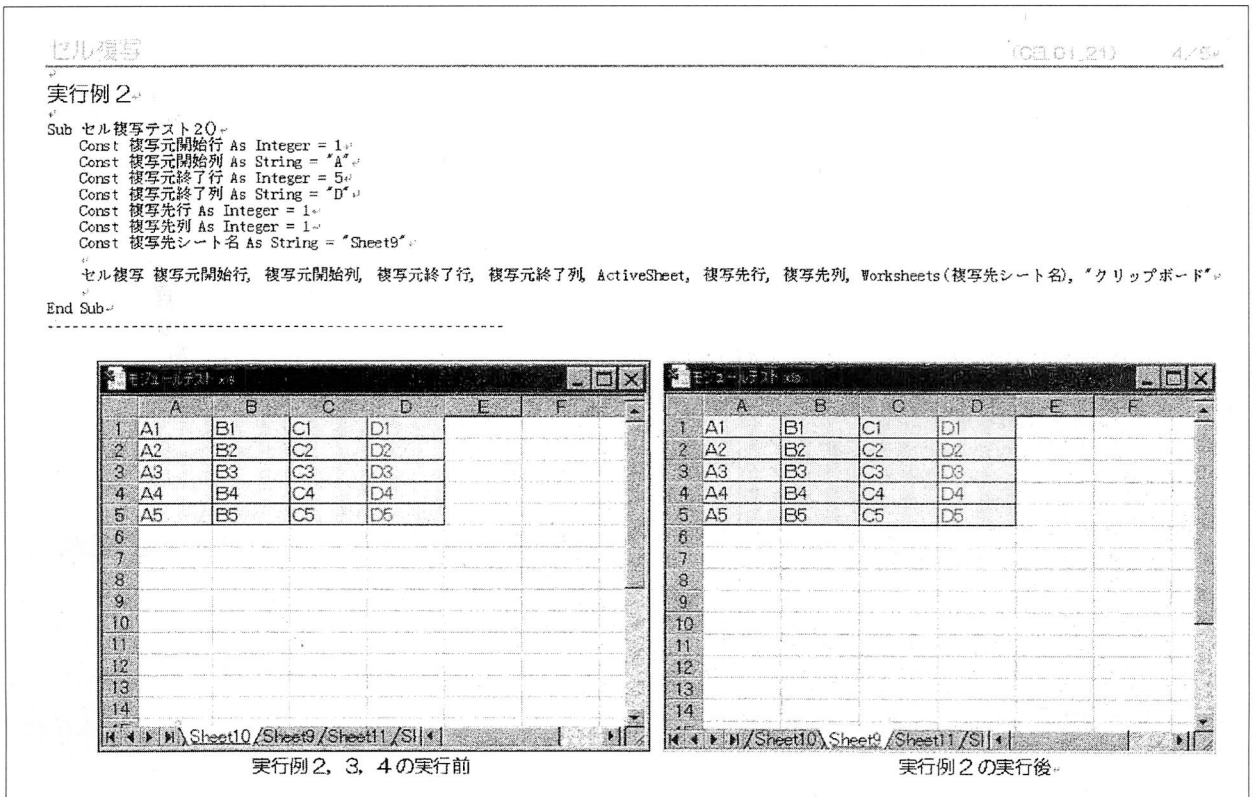

図１　仕様書に記載した実行例

6．6　仕様書の充実

日本語ライブラリのすべてのプロシジャおよび関数について仕様書を作成した。仕様書は，呼び出し方法，引数の型，引数の説明，注意事項，組込みモジュール，参照しているモジュール，ソースプログラムリスト，使用例で構成される。特に使用例については複数の例を示し，実行前と実行後の結果も掲載して理解を促すようにした。

6．7　仕様書検索機能

仕様書はWordで作成されており，画面上で読むことができる。さらにハイパーリンクで検索できるよう配慮している。ハイパーリンクによる検索の手順は以下のようになる。

(1)　一覧検索

利用者は，まずExcelのワークシートにあるライブラリの一覧表から目的の関数などを検索し，その関数のハイパーリンクをクリックする。このクリックにより，関数の仕様書がWordの画面上で表示される。

(2)　項目検索

仕様書には説明部分を検索できるよう，各要所にブックマークを登録してある。利用者は項目について詳細を知りたい場合は，そのハイパーリンクをクリックする。これにより，その説明部分が画面に表示される。

(3)　横断検索

仕様書には他の関数へのハイパーリンクも登録してある。また，項目によっては他の仕様書のブックマークへもリンクさせている。これによって，関数をまたがる検索もできる。

(4)　リンク先からの戻り

リンク先から元の仕様書へ戻る場合は，ExcelまたはWordのウェブバーにある［戻る］というアイコンをクリックすればよい。

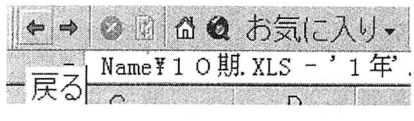

図2　ハイパーリンク先からの戻り

(5)　キーワードによるライブラリ検索

特定のキーワードを指定して，目的のプロシジャや関数を検索したい場合，日本語ライブラリ一覧表を使用する。一覧表には，Excelのオートフィルタのボタンが設定されているので，利用者は名称や説明の列にあるこのボタンをクリックし，キーワードを入力して絞り込み検索を行える。多段階の絞り込みも可能である。

これによって，目的に該当するプロシジャや関数の検索作業を改善できた。

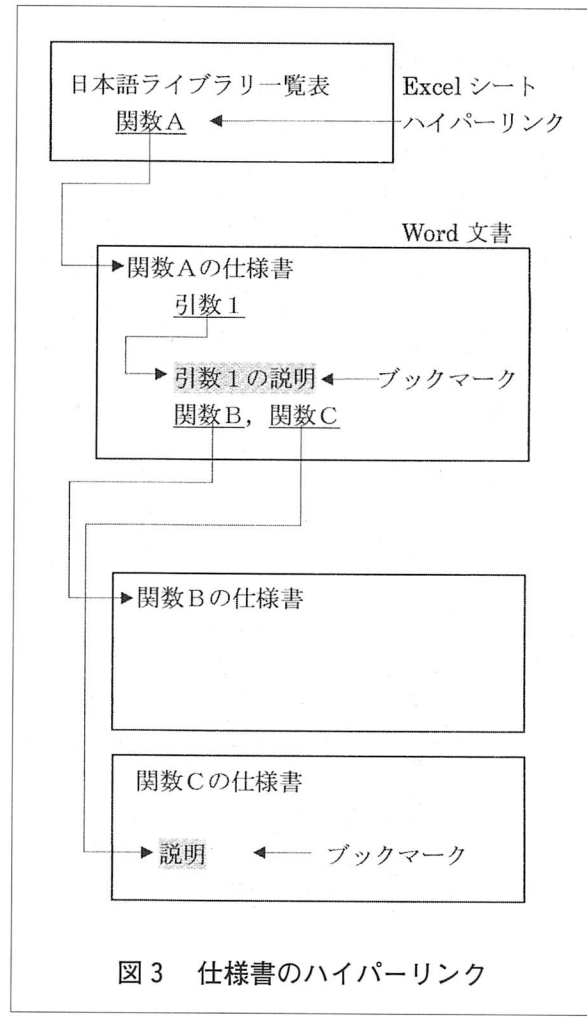

図3　仕様書のハイパーリンク

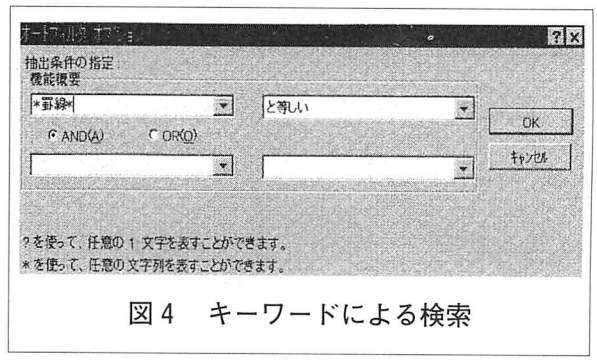

図4　キーワードによる検索

7．開発したプロシジャおよび関数

開発したプロシジャおよび関数を表１に示す。

表１　日本語ライブラリの構成一覧表

以下	参照アドレス分離	チャートオブジェクト検出	フォルダ--なし
以上	シートあり	チャートオブジェクト選択	複数記号による後続文字列抽
--致	シート関数設定	超過	複数記号による文字列の配列
色コード取得	シート取得	直線グラフ系列更新	複数行セル値設定
色番号色名取得	シート数取得	直線グラフ系列削除	複数組記号間文字列置換
色番号取得	シート追加	直線グラフ系列追加	複数文字列検出
色番号数値取得	シートなし	直線グラフ作成	複数文字列削除
色名取得	下付文字列設定	通算日数取得	複数文字列置換
上付下付文字列化	実数値 Hex ダンプ	通算和暦年取得	複数列セル値設定
上付文字列設定	GIF グラフ生成	月名取得	ブックあり
閏年	出力ファイル名取得	月リスト取得	ブックオープン
英大文字変換	上位からの区切り記号挿入	テキストボックス値取得	ブッククローズ
英頭文字取得	条件による変数値設定	テキストボックス値消去	ブックなし
英小文字変換	数式によるセル式設定	テキストボックス値設定	ブック保存
英字列数値変換	数式変数値設定	Dec 基数変換	ブック名抽出
オプションボタンオン	数値英字列変換	DecHex 変換	ふりがな取得
オプションボタン選択	スピンボタン値格納	トグルボタンオン	プロット領域設定
オプションボタン名取得	整数値 Hex ダンプ	トグルボタン選択	HexDec 変換
親オブジェクト取得	西暦年取得	トグルボタンリセット	変数値による棒グラフ色設定
下位からの区切り記号挿入	セル色コード取得	トグルボタン名取得	右不足文字追加
拡張子抽出	セル検出	ドライブ番号抽出	未満
型依存列番号取得	セルコメント消去	斜罫線描画	文字列位置取得
カタカナ変換	セルコメント設定	入力ファイル名取得	文字列検出
キーによる行挿入	セルコメント非表示	年リスト取得	文字列削除
記号間文字列置換	セルコメント表示	配列取得	文字列置換
記号間文字列抽出	セル書式取得	配列データ検出	文字列の配列への分割
逆探索文字列位置取得	セル書式設定	配列データ取得	文字列フォントサイズ設定
行削除	セル選択	配列データ設定	文字列分割
行挿入	セル値取得	配列文字列検出	文字列 Hex ダンプ
行ソート	セル値取得 B	配列文字列取得	曜日名取得
行列番号統合	セル値消去	文字配列設定	曜日リスト取得
行列番号分離	セル値設定	整数配列設定	横罫線描画
グラフ X 軸設定	セル値設定 B	実数配列設定	リストボックス行選択
グラフ Y 軸設定	セル値複写	通貨配列設定	リストボックス値取得
グラフ系列セル範囲取得	セル背景色設定	一般配列設定	リストボックスリスト作成
グラフ選択	セルパターン設定	パス終端編集	レンジ行列番号分離
グラフタイトル設定	セル貼付	パラメータシミュレート	レンジ取得
グラフフォント属性設定	セルフォント取得	パラメータ取得	和暦年取得
罫線あり	セルフォント設定	引数条件による値設定	複数記号による文字列分割
罫線消去	セル複写	引数による値設定	列削除
罫線なし	セル複製	引数による値設定 B	列挿入
罫線描画	セル文字透明化	引数によるオブジェクト設定	HTML リンク文字列生成
結合セル解除	セル文字配置取得	範囲行列番号統合	印刷範囲取得
結合セル行列番号分離	セル文字配置設定	範囲行列番号分離	印刷範囲設定
結合セル設定	全角変換	半角変換	印刷範囲解除
後続文字列抽出	全罫線描画	左不足文字追加	NT コマンド生成
項目列取得	先行文字列抽出	日付取得	グラフの場所設定
コマンドボタン無効化	選択セル行列番号取得	日付リスト取得	画面更新開始
コマンドボタン有効化	選択セル行列番号分離	表によるメニュー登録削除	画面更新停止
コンボボックス行設定	選択セルレンジ取得	ひらがな変換	メニューバーあり
コンボボックス値取得	対向記号文字列置換	ファイルあり	メニューバーなし
コンボボックスリスト作成	縦罫線描画	ファイルなし	メニューバー登録
最終実効引数位置取得	チェックボックスオン	ファイル名称抽出	メニューバー削除
最終セル行取得	チェックボックス選択	ファイル名称等抽出	メニューバー表示
最終セル列取得	チェックボックス名取得	不一致	メニューバー名取得
参照アドレス統合	チェックボックスリセット	フォルダーあり	メニュー情報登録

8．プロシジャおよび関数の例

開発したプロシジャおよび関数の例をリスト2に示す。セルパターン設定プロシジャはセルに背景色や塗りつぶしのパターンを設定する。先行文字列抽出関数は，対象文字列の特定文字列の前にある文字列を抽出する。

9．日本語ライブラリの参照

日本語ライブラリをExcelVBAのアドインファイルという形式で提供する。アドインファイルはコンパイル済みのプログラムファイルである。利用者が日本語ライブラリを使用してプログラムを作成する場合，開発するプロジェクトで日本語ラ

リスト2　プロシジャと関数の例

```
'---------- セルパターン設定 ---------- V2.1, 1999.09.02----------
'//  セルパターン設定  行, 列, シート, 色, パターン, パターン色
'//  セルパターン設定  セルアドレス, シート, 色, パターン, パターン色
'//  セルパターン設定  レンジ, 色, パターン, パターン色
'
Sub セルパターン設定(引数1 As Variant, Optional 引数2 As Variant, Optional 引数3 As Variant, _
                     Optional 引数4 As Variant, Optional 引数5 As Variant, _
                     Optional 引数6 As Variant, Optional 引数7 As Variant)
    Dim セル As Range
    Dim パターン As Long
    Dim パターン色 As Variant
    Dim セル色 As Variant
    Dim 色 As Variant
    Const なし As Long = xlNone

    If TypeName(引数1) = レンジOBJ Then
            Set セル = 引数1
            セル色 = 引数による値設定(引数2, 自動設定)
            パターン = 引数による値設定(引数3, xlSolid)
            パターン色 = 引数による値設定(引数4, 自動設定)
        ElseIf TypeName(引数1) = 文字列型 Then
            Set セル = 引数によるオブジェクト設定(引数2, ActiveSheet).Range(引数1)
            セル色 = 引数による値設定(引数3, 自動設定)
            パターン = 引数による値設定(引数4, xlSolid)
            パターン色 = 引数による値設定(引数5, 自動設定)
        ElseIf TypeName(引数1) = 短整数型 Or TypeName(引数1) = 長整数型 Then
            Set セル = 引数によるオブジェクト設定(引数3, ActiveSheet).Cells(引数1, 引数2)
            セル色 = 引数による値設定(引数4, 自動設定)
            パターン = 引数による値設定(引数5, xlSolid)
            パターン色 = 引数による値設定(引数6, 自動設定)
    End If
    色 = 色番号数値取得(セル色)
    If TypeName(色) = 短整数型 Then
        セル.Interior.ColorIndex = 色
    Else
        セル.Interior.Color = 色
    End If
    色 = 色番号数値取得(パターン色)
    If TypeName(色) = 短整数型 Then
        セル.Interior.PatternColorIndex = 色
    Else
        セル.Interior.PatternColor = 色
    End If
    セル.Interior.Pattern = パターン
End Sub

'----------先行文字列抽出---------- V2.0, 1999.02.24 ----------
'//              文字列数は全角、半角の各1文字を1とする。
'// 抽出文字列 = 先行文字列抽出(対象文字列, 検索文字列, 検索開始位置, 次の検索開始位置, 抽出文字列長)
'
Function 先行文字列抽出(対象文字列 As String, 検索文字列 As String, Optional 検索開始位置 As Variant, _
                        Optional 次の検索開始位置 As Variant, Optional 抽出文字列長 As Variant) As String
    Dim 検索文字位置 As Long
    Dim 文字 As String
    Dim 検索文字列数 As Long

    位置 = Variant引数による値設定(検索開始位置, 1)
    先行文字列抽出 = ""
    文字 = Mid(対象文字列, 位置, 1)
    検索文字列数 = Len(検索文字列)
    検索文字位置 = InStr(位置, 対象文字列, 検索文字列, 0)
    If 検索文字位置 > 0 Then
            先行文字列抽出 = Mid(対象文字列, 位置, 検索文字位置 - 位置)
            次の検索開始位置 = 検索文字位置 + 検索文字列数
            抽出文字列長 = 検索文字位置 - 位置
      Else
            先行文字列抽出 = Mid(対象文字列, 位置)
            次の検索開始位置 = 0
            抽出文字列長 = Len(先行文字列抽出)
    End If
End Function
```

イブラリの参照設定をすればよい。一度参照設定すると，それ以降の実行の際に自動的に日本語ライブラリが取り込まれて参照できるようになる。このため，ソースプログラムを取り込むことは不要である。日本語ライブラリは，実行システムに一つ複写しておけばよい。

WindowsNT のサーバを使用しているシステムの場合，サーバのファイルに一つ格納しておき，これを各クライアントパソコンで参照設定すればよい。

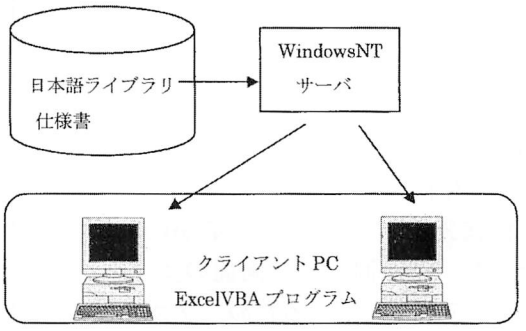

図 5　サーバクライアント環境での利用

図 6　日本語ライブラリの参照設定

10. 日本語ライブラリの評価

10. 1　ファイル容量と性能

(1)　アドインファイル

日本語ライブラリアドインファイルは1.1MB である。このファイルはメモリに常駐して参照される。

(2)　仕様書ファイル

日本語ライブラリ仕様書ファイルの容量は653MB である。

(3)　性能

日本語ライブラリは，通常の画面会話処理であれば，166MHz 程度のインテル系マイクロプロセッサで支障なく利用できる。しかし，プロシジャや関数のオーバヘッドが多くなるので，性能を意識する場合は300MHz 以上のプロセッサ性能のもとで利用することが望ましい。

10. 2　効果

(1)　使用プロシジャおよび関数決定の容易性

プロシジャおよび関数の名称を連想しやすいものにしたため，それらの検索と選択が容易になる。

(2)　保守性向上

日本語でプログラミングでき，さらに文章的に記述できるため，プログラムの保守性が向上する。

(3)　仕様書検索の容易性

仕様書の検索をハイパーリンクで行え，かつ横断的な検索もできるため，検索の作業を改善できる。

(4)　適用の容易性

仕様書に豊富な事例と図を掲載したので，利用者の理解と実務への適用判断を助ける。

(5)　ExcelVBA 機能の仮想化

ExcelVBA の機能からの独立性を高めてあるので，ExcelVBA の経験が未熟な利用者でも比較的理解しやすい。

10. 3 問題点

(1) 仕様書ファイル容量

仕様書ファイルの所要量が653MB と膨大になった。実行例として画面のイメージをビットマップで掲載しているため，ファイル容量が大幅に増えたものである。

最近の磁気ディスクの大容量化を考慮すれば，不安はないと思われるが，pdf 形式などへ変換して容量を削減することも検討しなければならない。

WindowsNT サーバに仕様書ファイルを格納しておけば，個々のクライアントパソコンに置く必要はなくなる。

(2) 性能

引数省略時の仮定値設定を可能にしたため，プロシジャや関数のオーバヘッドが増えている。これについては，プロセッサの発展が解決してくれるものと期待している。

(3) 冗長性

プログラムを文書として閲覧できるようにプロシジャや関数の名称を決定したため，冗長性が大きくなった。これはについては，二律背反の性格上やむを得ないものと判断する。

11. おわりに

筆者は，現在 ExcelVBA 用日本語ライブラリを使用して，学生向けの各種教材を開発している。その成果の一つが，本紀要に掲載した線形計画法自習教材である。日本語ライブラリにより，ExcelVBA 特有の予約語による記述量を削減できた。

今回は，日本語ライブラリにおける基礎的なプロシジャと関数の開発について報告した。今後もこの充足を進めてゆくが，並行してアプリケーション指向のライブラリ開発も行う予定である。

参考文献

1）森　重雄：ExcelVBA 日本語ライブラリ仕様書，1999

2）MicrosoftCorporation ： Office97 VisualBasic プログラマーズガイド，アスキー出版局，1997

3）Eric Wells·Steve Harshbargar·Micro Modeling Associates ： Microsoft Excel97 デベロッパーズハンドブック，アスキー出版局，1997

（平成11年11月30日受理）