地球流体電脳ライブラリ

MATH1

(数学処理下位パッケージ)

地球流体電脳倶楽部

2018 年 07 月 20 日 (DCL-7.3.3)

目次

第1章　概要

1.1　はじめに

MATH1 (数学処理下位パッケージ) は, 地球流体電脳ライブラリ群のなかでも, もっとも基本的なサブルーチンおよび関数を集めたパッケージである.

MATH1 は, いくつかのサブパッケージからなる.

- **SYSLIB：** 内部変数管理, メッセージ出力.
- **OSLIB：** システム依存ルーチン.
- **FNCLIB：** 基本関数 (最大整数, 剰余など).
- **SUBLIB：** 基本サブルーチン (自然数列の生成など).
- **CHRLIB：** 文字列の左・右詰め, 反転, 空白処理.
- **XFCLIB：** 文字列の数値化.
- **LRLLIB：** 実数値の比較.
- **BLKLIB：** 実数値と順序列.
- **GNMLIB：** きりのよい打ち切り数.
- **INTLIB：** 実数に近い整数.
- **INDXLIB：** 配列要素の検索.
- **IFALIB：** 整数の欠損値処理付最大最小など.
- **RFALIB：** 実数の欠損値処理付最大最小など.
- **RFBLIB：** 実数列の内積, 共分散, 相関係数.
- **VIALIB：** 1 つの整数型配列への作用素.
- **VIBLIB：** 2 つの整数型配列への作用素.
- **VRALIB：** 1 つの実数型配列への作用素.
- **VRBLIB：** 2 つの実数型配列への作用素.
- **CTRLIB：** 座標変換/回転.
- **MAPLIB：** 地図投影変換.
- **GT2DLIB:** 2 次元補間/変換.

これらの中の多くのサブパッケージでは, 「欠損値の処理」や「実数の誤差を考慮した大小関係の判定」ができるようになっている. なお, これらのサブパッケージの中でソースレベルでの機種依存性があるのは OSLIB のみである. (SYSLIB はシステムに依存する定数を管理している.)

以下本章では, MATH1 の特色である内部変数管理, 欠損値処理, 並びに, 誤差を含めた判断について若干の解

説をおこない, さらに, そのようなパッケージのよってきたる FORTRAN 77 の構造について説明を行なうことにする.

1.2 内部変数

地球流体電脳ライブラリでは, MATH1/SYSLIB の GL*p*GET/GL*p*SET のような**内部変数管理ルーチン**が多く使われている. (ここで, *p* は R/I/L/C のどれかで, それぞれ, 実数, 整数, 論理変数, 文字変数用である.) **内部変数管理ルーチン** とは, 設定された変数の値を保持し, 問い合わせに答えて値を返す機能を持ったルーチンで, 「掲示版」のような役目をするものである. この様なルーチンを使う理由は,

- 複数のサブルーチンで情報を共有するため.
- サブルーチンの引数の個数を最小限にするため.

である. 複数のサブルーチンで情報を共有するには, COMMON BLOCK を使うこともできるが, これを多用するとプログラムの可読性を落とすことにつながるので, 電脳ライブラリでは極力 COMMON BLOCK の使用を避けている. また, サブルーチンの引数を少くすると融通の効かないサブルーチンになってしまうが, かといって, むやみに引数の数を増やすとかえって使いにくくなってしまう. このような問題を解決するのが **内部変数管理ルーチン** である. もともと「パッケージの内部で使われる変数」という意味で, 内部変数という言葉を使っているが, その変数はパッケージ外からも設定/参照できるので, その有効範囲からすると C 言語の「外部変数」に似た性格を持つものである.

内部変数管理ルーチンは xx*p*GET, xx*p*SET という名前 (xx は通常パッケージの先頭 2 文字) を持ち, あらかじめ, システムが用意した変数の値 (システムデフォルト) を保持している. この値は xx*p*GET によって参照, xx*p*SET によって変更することがでる. 電脳ライブラリの多くのルーチンでは, 必要なパラメタの多くを xx*p*GET によって取得しており, ユーザーが何も指定しなければシステムデフォルトを使い, xx*p*SET で値を指定すれば, その値を使うようになっている.

この内部変数は実行時オプションによって変更することもできる. 実行時オプションとは, プログラム実行時に外部環境と交信することによって内部変数への介入を可能にするような指定である. 具体的には, 外部ファイル, 環境変数, コマンドライン引数などによる交信手段を念頭においた概念であるが, それがどのように実現されているかは, システムに依存する. なお, **実行時オプションの効力は, システムデフォルトよりも強く, xx*p*SET の指定よりも弱い.**

実行時オプションに比べて xx*p*SET による設定は強力なので, その設定を実行時に変更することができないが, 実行時オプションより効力の弱い xx*p*STX というルーチンも用意されている. これを使えば, プログラム中でデフォルト値 (ユーザーデフォルト) を設定し, 実行時に変更することが可能になる.

内部変数の設定手段を, その効力の強さの順に並べると,

1: xx*p*SET
2: 実行時オプション
3: xx*p*STX
4: システムデフォルト

となる.

実行時オプションは,「オプション名」と「オプションの値」の組合せからなり, 複数個指定できる. オプション名は原則として, 介入しようとする内部変数を管理している xx*p*GET/xx*p*SET の最初の 2 文字 'xx' と内部変数名 'PNAME' を組み合わせて xx_PNAME となる. 区切り文字自身は、環境変数 DCLENVCHAR を設定することで変更できる.

環境変数を通してオプションを指定する場合, 例えば UNIX の C シェル環境で GL*p*GET/GL*p*SET が管理する内部変数 'MSGLEV' を 1 に変更したいときには,

```
> setenv GL_MSGLEV 1
```

とする. bash シェル環境では

```
> export GL_MSGLEV=1
```

である. このとき**環境変数名はすべて大文字でなければならない.**

また, コマンドライン引数を通してオプションを指定する場合は, 原則としてオプション名に - 記号を付け, さらにオプション名とオプションの値の間に '=' を入れて, -xx_PNAME=value という形式で指定する (環境変数名と違って大文字・小文字の区別はない). 例えば, sample というプログラム実行時に

```
> sample -gl_msglev=1
```

とすると, 上記の環境変数の場合と同じ結果が得られる. ここで, **'=' 記号の前後に空白を入れてはいけない.**

外部ファイルを通してオプションを指定する場合は,

```
1  gl_msglev  1
2  gl_lmiss .true.
```

のように (行のはじめの番号は便宜的につけてある), オプション名とオプションの値を並べて書いたファイルを, たとえば .dclrc としてカレントディレクトリに用意してやればよい. ここで, オプション名は第 1 カラム目から書きはじめ, オプションの値との間は 1 個以上の空白によって区切らなければならない (タブなどを入れてはいけない). 読み込むファイルを検索するルールについては, 第 2 節を参照のこと.

なお, 環境変数, コマンドライン引数, 外部ファイルを通して同じオプションを指定した場合には, 原則としてその優先順位はコマンドライン引数, 環境変数, 外部ファイルを通して指定したもの順になる. (コマンドライン引数による指定が一番強い.)

1.3 欠損値処理

自然を相手に観測などする場合, 観測機器の不調等により正常にデータが取れない場合がある. そのような場合, 本来データを書くべきところに, 特定のある数値 (例えば 999.) を書いて, 正常にデータが取れていないことを示すことがある. この「正常なデータでない」ことを意味する数値が欠損値である. 地球流体電脳ライブラリの多くのパッケージでは, このような欠損値を含むデータを直接扱うことができる.

例えば, 次のようなデータがあったとしよう.

日最高気温の月平均値

月 \ 観測点	札幌	仙台	東京	名古屋	京都	福岡	鹿児島
1 月	-1.1	5.0	9.5	8.4	999.	9.3	12.2
7 月	24.8	25.6	28.8	999.	999.	30.6	31.4

1 月の京都, 7 月の名古屋, 京都の 999. はデータが正常に得られなかったことを示している. 1 月, 7 月それぞれの月のデータを配列 T1, T7 に読み込んで平均値を算出するとき, 999. まで有効なデータとして扱ってしまうと, とんでもない値が算出されてしまう. そこで, 以下のように GL*p*GET/GL*p*SET が管理する内部変数 'LMISS' を.TRUE. として (初期値は.FALSE.) RFALIB の RAVE を用いれば, 999. を除いた平均値が算出できる.

```
1         CALL GLLSET('LMISS', .TRUE.)
2         TAVE1 = RAVE(T1, 7, 1)
3         TAVE7 = RAVE(T7, 7, 1)
```

なお, 欠損値処理をおこなうことができる関数では, すべての配列要素が欠損値のときは欠損値が返される.

また, VRBLIB の VRADD を用いて配列 T1 と T7 の和を TX として求めるときは,

```
1         CALL GLLSET('LMISS', .TRUE.)
2         CALL VRADD(T1, T7, TX, 7, 1, 1, 1)
```

としてやればよい. 'LMISS' を.TRUE. としたとき, VRADD は T1 および T7 の配列要素の少なくともどちらかが欠損値ならば欠損値を返すようになっているので, この場合, 名古屋, 京都に対応する TX の配列要素は欠損値となる.

欠損値は GL*p*GET/GL*p*SET が管理する内部変数'RMISS' で指定され, 初期値は 999. である. 999. という値は, 気温としては本来ありえない値なので, 気温データなどの場合には問題はないが, この値がデータの範囲に入ってしまうような場合には, 'RMISS' の値を変更しておく必要がある.

1.4 誤差を含めた判断

計算機内部で実数値は近似的に扱われるため, どうしても誤差を含んでしまう. 例えば,

```
1         LOGICAL LEQ
2         X = 1.
3         Y = X/3.
4         LEQ = X .EQ. (Y*3.)
```

というプログラムでは LEQ は.TRUE. にならない. しかし, もともと X や Y は近似値なのであるから, それらを比較するときも近似的に比較するのが合理的であり, 実際そのような比較をしたい場合も多い. すなわち, X と Y*3. が正確に一致するかどうかではなく, 近似的に一致するかどうか調べたい場合も多い.

そのような場合のために, LRLLIB が用意されている. 例えば,

```
1         LOGICAL LEQ, LREQ
2         X = 1.
3         Y = X/3.
4         CALL GLLSET('LEPSL', .TRUE.)
5         LEQ = LREQ(X, Y*3.)
```

とすると, 近似的な比較が可能になる. LREQ は LRLLIB の関数で, 2 つの値が計算機の精度内で近似的に一致するかどうか調べるものである. ここでいう計算機の精度とは GL*p*GET/GL*p*SET の管理する内部変数'REPSL' に'RFACT' をかけた値である. 'REPSL' には, 実際の実数の内部表現の精度に安全率として 10 倍をかけた値が設定されており, 'RFACT' には初期値として 1 が設定されている. 'REPSL' はライブラリのインストール時に決るもので, この値をプログラムの中で変更することはできないが, 計算精度が悪くて, この値よりもラフな比較をしたい時には, 内部変数 'RFACT' に 1 以上の数値を指定すればよい.

LRLLIB だけでなく, MATH1 のいくつかのサブパッケージには同様の機能がある.

1.5 FORTRAN77 規格について

FORTRAN は 1950 年代から使われている, もっとも古い高級言語である. 1966 年に FORTRAN66 として規格が定められ, その後, 1978 年に全面的に改訂されて FORTRAN77 となった.

このように, 長い歴史を持つ言語だけに, FORTRAN で書かれたライブラリーは膨大な量にのぼる. 反面, C 言語など新しい言語が数多く存在する現在では, 古い歴史を引きずった堅い (融通の効かない) 言語という印象も免れない. しかしながら, FORTRAN には科学計算用の言語として, 他の言語にはない使いやすさがある. 単に「蓄積が大きい」というだけでなく, この「使いやすさ」こそ, FORTRAN が 30 年以上も使い続けられてきた真の理由である.

FORTRAN に限らず, どんな言語でもそれを使いこなすのは, なかなか大変である. 幸にも, FORTRAN の仕様は比較的小さい (文法書が薄い) ので学習は容易なほうであるが, それでも FORTRAN を “極める” のは容易なことではない. MATH1 を含む地球流体電脳ライブラリでも, 入門書に書いてない「奥義」をかなり使っている. もちろん, これらの知識がなくても電脳ライブラリを使うことは可能である. しかし, 電脳ライブラリをよりよく理解し上手に使いこなすためには是非とも知っておいていただきたい. また, これらの知識は応用範囲が広いので, 一般の FORTRAN プログラムを書く際にも, きっと役にたつはずである.

1.5.1 FORTRAN の方言

計算機言語には各国の標準機関 (日本では JIS) 及び国際標準機関 (ISO) により定められた規格がある. FORTRAN も例外ではなく,

- **アメリカ規格:** ANSI X3.9-1978 Programming Language FORTRAN
- **国際規格:** IS 1539 Programming Languages — FORTRAN
- **日本工業規格:** JIS C 6201 電子計算機プログラム言語 FORTRAN

という言語規格があるが, 実質的には全て同じもので, いわゆる FORTRAN77 の規格である.

言語規格というのは「日本語の文法」の様なもので, 計算機に自分が意図した動作をさせるためにはこれをよく知っていなければならない. これは, 当り前のようで, 意外に軽視されがちである. なぜならば, 普通, 多くの人々は FORTRAN を勉強する時, 英語の勉強のように, 文法を厳しく教わりながら勉強するわけではないからである. どちらかというと, 日本語を覚えたときと同じ様に, 「見よう見まね」または「試行錯誤」により FORTRAN を勉強するのが普通である. そうなると, 多くの人が生まれた土地の方言を, 方言と気づかずに覚えてしまうのと同じ様に, 特定の計算機の特定のコンパイラだけが解釈できるプログラムを平気で書いてしまうことになる.

FORTRAN コンパイラは通常, 標準語としての FORTRAN77 規格 $+\alpha$ の機能を持っている. この $+\alpha$ の部分は, 拡張機能と呼ばれ, このような拡張機能を持つコンパイラは FORTRAN77 上位互換 (アッパーコンパチ) コンパイラと呼ばれる. これは, 標準語だけで書かれたプログラムでも, $+\alpha$ の機能を使ったプログラムも, どちらもコンパイルできるという意味ではコンパイラの「機能が高い」のであるが, 逆に $+\alpha$ の機能を使ったプログラムは, そのコンパイラでないとコンパイルできないことになる. したがって, 我々は $+\alpha$ の部分を「方

言」と呼ぶ.

日常会話の中では, 方言は特に問題にならず, むしろ標準語では表現できないニュアンスまで伝えることができて便利である場合が多い. これは, 方言が標準語の拡張機能として働いているためであるが, ビジネス上の会話で方言を使うと, とんでもない結果をもたらす. 多くの人は「明日, 東京にいかずー」という約束をしたら, 明日, 東京にいくべきか否か迷うに違いない. (これは中部地方の方言で, 東京にいこう, という意味である)

これと同じ様に, FORTRAN の「方言」は特定の計算機を使いこなすには便利な場合も多いが, むやみに使うと非常に危険である. しかし, 生まれた土地を離れたことがないと, 自分の言葉が方言かどうかわからないのと同じ様に, 特定の計算機しか使ったことがないと, 正確な FORTRAN の文法を身につけるのは難しい. なぜなら, いつも使っている FORTRAN77 のコンパイラがちゃんとコンパイルしてくれれば, そのプログラムは FORTRAN77 規格に準拠したプログラムであると思ってしまうからである. しかし, これは明かな間違いである. どんなコンパイラでも大なり小なり「方言」を持っており, その「方言」を使ったプログラムは, たとえ, FORTRAN77 規格のコンパイラでコンパイルできたとしても, FORTRAN77 規格準拠のプログラムではない.

そこで, FORTRAN で何とかプログラムが書けるようになった初心者も, この道 10 年のベテランも, 市販の文法書以外に「岩波 FORTRAN 辞典」を座右におかれることを強くお勧めする. これは単なる文法書ではなく, 規格が意味するところや, 歴史的背景などが解説してあって, 辞典としてだけでなく, 読物としても面白い書物である.

電脳ライブラリはいくつかの例外を除いて, 極力この「方言」を避けて書いてある. どうしても「方言」使わなければならないようなところは, 直接「方言」を使わず「電脳標準語」を定義して, 各プログラムの中では「電脳標準語」を使うようにしている.

例えば, SYSLIB の中の `MSGDMP` はエラーを検出したとき, システムに依存したエラー処理をおこなって (たとえば, エラーのトレースバック情報を出力して) プログラムを強制終了する `OSABRT` というサブルーチンを呼ぶ. 各コンパイラは必ずといってよいほど, この種のサブルーチンを「方言」として持っており, `OSABRT` は単にそのコンパイラ固有のサブルーチン (方言) を呼んでいるだけである. しかし, その「方言」を使わず,「電脳標準語」である `OSABRT` を使って他のサブルーチンを書いておけば, `OSABRT` を各コンパイラ用に書きなおすだけで, そのようなプログラム全てを移植できることになる.

この移植のしやすさ (可搬性) こそ電脳ライブラリの最大の特徴であり, ライブラリ開発者が最も苦労した部分でもある. この特徴を最大限に生かすためにも, FORTRAN の文法をよく理解して, 不必要な「方言」を使わないよう心がけて欲しい.

以下の節では不必要な方言を避けるために知っておくべき知識, 知っていると便利なことがら, そして, プログラミング上の盲点となるような落し穴について多少の解説をしておく.

1.5.2 不定の概念

FORTRAN 規格の基本的な概念に「不定」という概念がある. これは最も重要な概念の一つでありながら, 最もわかりにくい概念でもある. FORTRAN を勉強したことのある人ならば, 文法書の中の

○○の時, 変数××は不定となる.

という記述を見て, 首をかしげた経験が 1 度はあると思う. 最初にこれを読むと, 「××に乱数が代入される」とか「数値がメモリ上から消滅する」ということを想像してしまうが, 決してそういう意味ではない. 大抵の場合, ○○の時, ××がどうなるか, コンパイラはちゃんと知っている.

この奇怪な文章を理解するには, FORTRAN の規格の文章が誰に対する文章であるかということを, よく理解しておく必要がある. FORTRAN 規格というのは, 我々プログラマに対する文章であると同時に, FORTRAN コンパイラをつくる人に対する文章でもある. したがって, この文章は 2 通りに翻訳できるように書いてあるのである. 上記の文章を, コンパイラをつくる人に対する文章に翻訳すれば,

○○の時, 変数××が占めていた記憶領域をどのように使ってもよい.

ということになり, 我々プログラマに対しては,

コンパイラをつくる人に, 上のように言ってしまったので, ××がどうなるかわからない.

ということになる.

もし, この文章をコンパイラを「作った人」から「使う人」へのメッセージだと解釈してしまうと, 「自分で作っておきながら, わからない, どういうことだ」と言いたくなってしまう. 特に, コンパイラに付属の文法書を読んでいると, そのような錯覚に落ちいりやすいが, FORTRAN の文法書とはあくまでも,

コンパイラを作る人と, それを使う人の両者の責任関係を明確にするための法律

なのである.

「不定」とは逆に, 規格によって値が保証されている状態を「確定」という. 規格上, プログラムの中で引用できる変数は「確定」された変数だけである. 以下, 変数が「不定」となる例である.

- **変数が未定義のとき:**
  これは, 最もわかり易い「不定」の場合である. 変数に値が代入なければ, 当然その変数にどのような値が入っているかわからない.

  コンパイラによっては全ての変数を初期化 (通常は 0 を代入) するものもある. FORTRAN 規格はあくまで「不定」である (どうしてもよい) ので, そのようなコンパイラでも規格に違反しているわけではない. しかし, これは規格によって保証されているものではなく, 一種の「方言」であるので, これに頼ってプログラムを書くと後で必ず後悔する.

  最初から変数に値を入れておきたい時には DATA 文を使う. 次の例を参照のこと.
- **サブルーチンの中の RETURN 文または END 文を実行したとき:**
  サブルーチンの中で使われる変数は, いくつかの例外を除いて, サブルーチンの実行が終了した段階で「不定」となる.

  これは少々わかりにくいが, 要するに, 同じサブルーチンを 2 回呼んだとき, 1 回目で値が代入された変数を, 2 回目に参照しようとしても, 1 回目の値が保存されている保証はない, ということで

ある.

例えば, 次のプログラムを見て欲しい.

```
*------------------ main program ----------------
      DO 100 I=1, 100
        CALL BEAT
  100 CONTINUE
      END
*-----------------------------------------------
      SUBROUTINE BEAT
      DATA N /0/
      SAVE
      N=N+1
      IF(N.EQ.10) THEN
        WRITE(6,*) ' I hate you !'
        N=0
      ENDIF
      END
```

このサブルーチン BEAT は呼ばれた回数 (N) を覚えていて, 10 回呼ばれるごとにメッセージを出力するものである.

変数 N は BEAT が呼ばれる前に, DATA 文によって 0 という値で確定している. BEAT が 1 回呼ばれると, この変数 N に 1 が加えられるが, もし, 9 行目の SAVE 文がなければ, 16 行目の END 文を実行した段階で, FORTRAN の規定により N は不定となる.

SAVE 文は, サブルーチン内の変数の値をサブルーチンの実行が終っても保持するように指示する宣言文である. このプログラムのように変数 N の値が次回の呼び出しまで保持されないと正常に動かないようなプログラムでは必ず書かなければならない.

コンパイラによっては (というより多くのコンパイラでは) サブルーチンの中で使われる局所変数を保存するが, これも「方言」である.

ただし, DATA 文で指定された変数は, その値が書き換えられない限り, RETURN 文または END 文を実行しても「不定」にはならない.

因みに, 地球流体電脳ライブラリの GL*p*GET/GL*p*SET が, 掲示板の役目を果たせるのは, この DATA 文と SAVE 文のおかげである. (1.5.4 節参照.)

なお, SAVE 文を指定すると, 通常, 実行ファイルのサイズは大きくなるが, 実行速度は若干速くなることが多い.

- **異なる型の変数が結合しているとき:**

  異なる型の変数が EQUIVALENCE 文などにより結合しており, その一方の変数が確定したとき, もう一方の変数は不定となる. 例えば,

```
      INTEGER IX
      REAL    RX
      EQUIVALENCE (IX, RX)
      RX = 1.
      WRITE(6,*) IX
      END
```

  というプログラムで, 実変数 RX が代入されると, 整変数 IX の値は不定となる. したがって, 不定の変数を引用している WRITE 文は, 文法違反である.

しかし, この時 IX は, ちゃんと「ある値」を持っている. ただし, 整数や実数の表現形式がコンパイラによって異なるため, この「ある値」が, どのような値になるか FORTRAN 規格からは予測できないという意味で「不定」なのである.

したがって, 文法上このプログラムの結果はわからないが, 通常は正常に動作して「ある値」を書き出すことになる. (1.5.4 節参照.)

1.5.3 ENTRY 文

FORTRAN でサブルーチンを書く時, いろいろな場合を想定して, なるべく汎用的なサブルーチンにしようとすると, 引数がどんどん増えて困ることがある. 大事な引数は 2–3 個なのに, どうでもよいような引数がずらずらと並んでいると, それだけで CALL .... と書く手が重くなってくる.

そのような時に ENTRY 文を使うと解決できる場合がある. ENTRY 文とは, サブルーチンの途中から実行を始めるための文である.

```
1   *----------------- main program -----------------
2         REAL X(10)
3         .......
4         CALL TABLE(X, 10)
5         ........
6         END
7   *-----------------------------------------------
8         SUBROUTINE TABLE(X,N)
9         REAL       X(N)
10        DATA       IOU /6/
11        SAVE
12        WRITE(IOU,'(10F12.4)') X
13        RETURN
14        ENTRY IOUSET(IOU0)
15        IOU = IOU0
16        RETURN
17        END
```

このサブルーチン TABLE は 1 次元配列 X を出力するものであるが, TABLE だけを呼べば, DATA 文で指定された IOU の機番 (6) に出力される. ファイルなど別の機番に出力させたい場合は, TABLE を呼ぶ前に IOUSET を呼んで, IOU を書き換えておけばよい. つまり, 1 つのサブルーチンに, 印刷すべき変数だけを与える入口と, その他のパラメータを設定する入口の, 2 つの入口を作るわけである.

ここで, 注意すべきことは SUBROUTINE 文または ENTRY 文で指定された引数は, その SUBROUTINE 文または ENTRY 文から入った時でないと使えないことである. 例えば, このプログラムの ENTRY 文から入った時 (IOUSET が呼ばれた時), 変数 N を引用することはできない. 同じように, TABLE が呼ばれた時に IOU0 を引用することはできないので, IOUSET が呼ばれたときに IOU に代入しておかなければならない. また, このようなプログラムでも SAVE 文と DATA 文は不可欠である.

1.5.4 GL*p*GET/GL*p*SET の構造

電脳ライブラリの「パラメータ掲示版」である SYSLIB の GL*p*GET/GL*p*SET は, 毎回指定する必要のないパラメータをうまく隠蔽して見えないようにしておくという前述のテクニックを極限にまで利用したライブラリなのである. GL*p*GET/GL*p*SET は他のプログラムにパラメータを渡すだけで, あとは何もしない. その構造は, 以下のようになっている (実際のソースより簡略化されている).

(注: Ver.5 の電脳ライブラリでは, 実行時パラメータによって内部変数への介入が可能になったため GL*p*GET/GL*p*SET は内部でさらに下請けルーチンを呼ぶようになった. 以下では, Ver.4 の電脳ライブラリにおける GL*p*GET/GL*p*SET を用いて説明をおこなう. Ver.5 の下請けルーチンの構造は, 基本的に Ver.4 の GL*p*GET/GL*p*SET と同じである.)

```
1  *---------------------------------------------------------------------
2  *     GLPGET / GLPSET
3  *---------------------------------------------------------------------
4        SUBROUTINE GLPGET(CP,IPARA)
5        CHARACTER CP*(*)
6        PARAMETER (NPARA=17)
7        INTEGER   IX(NPARA)
8        REAL      RX(NPARA)
9        LOGICAL   LX(NPARA)
10       CHARACTER CPARA(NPARA)*8
11       EQUIVALENCE (IX,RX,LX)
12       SAVE
13       DATA CPARA( 1)/'NBITSPW '/, IX( 1)/32/
14       ........
15       DATA CPARA(12)/'LMISS   '/, LX(12)/.FALSE./
16       DATA CPARA(13)/'IMISS   '/, IX(13)/999/
17       DATA CPARA(14)/'RMISS   '/, RX(14)/999.0/
18       ........
19       DO 10 N=1,NPARA
20         IF (CP.EQ.CPARA(N)) THEN
21           IPARA=IX(N)
22           RETURN
23         END IF
24    10 CONTINUE
25       エラー処理
26       STOP
27 *---------------------------------------------------------------------
28       ENTRY GLPSET(CP,IPARA)
29       DO 15 N=1,NPARA
30         IF (CP.EQ.CPARA(N)) THEN
31           IX(N)=IPARA
32           RETURN
33         END IF
34    15 CONTINUE
35       エラー処理
36       STOP
37       END
```

ここで, CPARA という文字変数にパラメータ名が登録されており, その名前に対応した値が IX, RX, LX という変数に入っている. (これら 3 つの変数は EQUIVALENCE 文により結合されているので, 実際には 1 つの記憶領域しか持たない)

さて, IFALIB の関数 IMAX は, 欠損値以外の最大値を求めるものであるが, IMAX は欠損値を示す整数値を引数から得る代りに,

```
1        CALL GLPGET('IMISS', IMISS)
```

という文を実行する. この時, GLPGET では, CPARA の中から'IMISS' という名前を探して (24–29 行目) その名前に対応する値 IMISS として返す. この時返される値は, DATA 文によって指定されているが, 必要であれば, あらかじめこれらの値を GLISET で変更しておくこともできる.

この方法が前の 1.5.3 節の方法 (各プログラムが IMISS 設定用の ENTRY を持つ方法) より優れている点は, GLPGET が持つ情報を複数のサブルーチンから参照できることである. 例えば, 欠損値を示す値は何も初期値としては 999 が与えられているが, 999 がデータ範囲に入っているような場合には, これを変更しておかなければならない. そのような時にでも GLPSET を一度呼ぶだけで, 欠損値処理をする全てのサブルーチンの動作を制御することが可能になる.

1.5.5 文字型変数とその他の変数の違い

前節では実変数, 整変数, 論理変数を混同した書き方をしても, 大きな問題にならなかった. これは, FORTRAN 規格により, これら 3 つの型のデータが全て「1 数値記憶単位を占める」と規定されているからである. これに対して, 文字型のデータは 1 文字が「1 文字記憶単位を占める」と規定されている. 多くの処理系においては「1 数値記憶単位」は 32 ビットの「1 ワード」に対応し, 「1 文字記憶単位」は 8 ビットの「1 バイト」に対応するが, 規格の中で具体的な長さは規定されていない. したがって, 文字型とその他の変数を EQUIVALENCE 文で結合したりすると, 規格上, 「数値記憶単位」が「文字記憶単位」のいくつ分に相当するかわからないため, そのような結合は禁じられている.

さらに, これは規格上の話だけではなく, 上記のような 1 ワードが 4 バイトに相当する標準的な処理系で, ユーザーが「数値記憶単位」と「文字記憶単位」の対応をよく考えた上で結合したとしても実害が生じる場合がある. これは, 文字型変数が文字単位で任意の長さがとれることから, 文字型変数の扱い方が, それ以外の変数 (実数, 整数, 論理変数) と大きく異なる場合があるからである. 特にサブルーチンの引数の渡し方が異なる場合が多い.

もともと FORTRAN66 の規格には文字型変数はなく, 文字を扱う場合にはキャラクターコード列 (整数) として扱われていた (4 文字を 1 つの整数として扱っていた). そのため 4 文字単位の長さの文字列しか扱えず不便であったが, FORTRAN77 から文字型変数として正式に認知されるようになった. これに伴って, 任意の長さの文字列が扱えるようになるなど, 文字の扱いが非常に楽になった反面, 上記のような制約も増えた. もっとも, かなり多くのコンパイラでは文字型とそれ以外の変数の結合が可能であるが, これも「方言」の一つである.

実際の観測データなどを扱う分野では, 文字と数字が混在したデータを扱うことも少なくないので, 伝統的にこのような EQUIVALENCE 文を使っている場合もあるが, 実際にこのような結合ができないコンパイラもあるので, 注意が必要である.

さらに, 文字型変数は他の型の変数と違って, 以下の例のように, 同じ文字要素を式の右辺と左辺に書く (3 行目) ことが禁じられている.

```
1         CHARACTER*12 CX
2         CX = 'DENNOU'
3         CX = CX(1:6)//'DENNOU'
```

文字以外の変数では, 同様のことが許されているので, つい, このような代入文を書いてしまいそうになるが, このような場合には,

```
1         CHARACTER*12 CX
2         CX = 'DENNOU'
3         CX(7:12) = 'DENNOU'
```

とすれば, 問題はない.

1.5.6 フォーマットの動的変更

FORMAT 文を用いて WRITE 文などで出力する書式を指定する場合, 書式の中に変数が使えないため, 「プログラムの中で必要な桁数を判断して, 出力する桁数を変える」という様な芸当ができない. このあたりが,

古い言語の堅さの 1 つなのであるが, 「だから FORTRAN はきらいだ」という前に, 以下の説明を読んで欲しい.

FORTRAN77 では WRITE 文の書式識別子に FORMAT 文の文番号ではなく, 以下のように, 文字変数を指定することができる. この機能を使えば, 書式の動的な変更が可能になる.

```
1         CHARACTER CFMT*12
2         ...
3         CFMT = '(T12, F5.2)'
4         WRITE(6,CFMT) X
5         ...
```

この例では, 文字変数 CFMT に定数を代入しているだけなので, 動的に書式を変更するようにはなっていないが, CFMT は変数であるから, プログラムの中で変更するのは容易である.

桁数などの数値変数を文字変数に書き込むには, やはり WRITE 文を使う. WRITE 文の出力機番を書くところに文字変数を書くと, その文字変数を “内部ファイル” として, 普通の WRITE 文と同じ様に編集, 出力ができる.

```
1         CHARACTER CFMT*12
2         ...
3         CFMT = '(T12, Fx.2)'
4         WRITE(CFMT(8:8),'(I1)') N
5         WRITE(6,CFMT) X
6         ...
```

この例では, 整変数 N を文字変数 CFMT の 8 桁目に書き込み, X を出力する際の桁数としている. このあたりのテクニックは FORTRAN66 の時代には使えなかったので知らない人も多いが, 知っていると非常に便利である.

1.5.7 多次元配列の記憶順序

FORTRAN77 では多次元 (7 次元まで) の配列を定義できるが, 実際の記憶装置は多次元構造を持っているわけではなく, 1 次元的に管理されている. 多次元配列を 1 次元的に展開する順序は, FORTRAN77 の規格で定めてあるので, 多次元配列を 1 次元配列と結合する (多次元配列に 1 次元配列の別名を与える) ことが可能である. 例えば

```
1         REAL         X(6), Y(2,3)
2         COMPLEX      Z(3)
3         EQUIVALENCE (X,Y,Z)
```

という場合, 変数 X, Y, Z はどれも長さが 6 語 (48 バイト) の配列で, EQUIVALENCE 文により同じ記憶領域を占める. この時, それぞれの変数の並び方は

X(1)	X(2)	X(3)	X(4)	X(5)	X(6)
Y(1,1)	Y(2,1)	Y(1,2)	Y(2,2)	Y(1,3)	Y(2,3)
Re(Z(1))	Im(Z(1))	Re(Z(2))	Im(Z(2))	Re(Z(3))	Im(Z(3))

という様に, 2 次元配列 Y は添字の左側の数字から変るように 1 次元的に展開される. また, 複素数データ Z は 2 つの実数が並んだ形で展開される. したがって, X(3) の数値は Y(1,2), Re(Z(2)) と全く同じになる.

この規則は多くのプログラムで積極的に使われており, 多次元配列を 1 次元配列として扱ったり, 複素数データを実数データを見なして処理したりすることが行われている. この規則は, 計算機のハードウェアに近い部

分の規則なので, 計算機によって異なるように思えるかもしれないが, FORTRAN77 規格で定められた「標準語」である.

1.5.8 サブルーチンへのデータ引渡

引数によるデータの引渡では, 前述の配列の記憶順序を意識したテクニックがよく使われる. MATH1 の IFALIB, RFALIB, VRALIB などでは, 1 次元配列の要素を飛び飛びに使うための引数 (JX など) が必ずある. これは, 多次元配列を同じサブルーチンまたは関数で処理するためのものである. これらのルーチンの動作を理解するために, 以下のプログラムを見て欲しい.

```
1         REAL X(100)
2         ......
3         CALL SUM(X, 100, 1, XSUM)
4         ......
5         END
6         SUBROUTINE SUM(X, N, JX, XSUM)
7         REAL X(*)
8         XSUM = 0.
9         DO 100 I=1, JX*(N-1)+1, JX
10          XSUM = XSUM + X(I)
11    100 CONTINUE
12        END
```

このサブルーチンは 1 次元配列 X の和を求めて, XSUM として返すものである. この同じサブルーチンを 2 次元配列 Y に対して

```
1         REAL Y(10,10)
2         ........
3         CALL SUM(Y(2,1), 10, 10, YSUM)
4         ......
5         END
```

という様に使うとサブルーチン SUM は, 以下のように 2 次元配列要素 Y(2,1) を先頭とする記憶領域を 1 次元配列 X に割り当てる.

Y(2,1)	Y(3,1)	···	Y(10,1)	Y(1,2)	Y(2,2)	···	Y(10,2)	Y(1,3)	Y(2,3)	···
X(1)	X(2)	···	X(9)	X(10)	X(11)	···	X(19)	X(20)	X(21)	···

その 1 次元配列 X の要素を 10 (JX) 個おきに 10 個の和をとると, Y(2,1), Y(2,2), ···, Y(2,10) という具合に, 二次元配列 Y の第 2 添字についての和をとることになる.

ここで, サブルーチンの中では配列 X の長さはわからず, 単に, X(1) を先頭とする 1 つながりのデータとして扱われていることに注意して欲しい. 配列がサブルーチンに渡されるとき, メインプログラムの配列の中身がサブルーチンの配列の中身にコピーされるのではなく, メインプログラムの中の配列の番地だけが渡され, サブルーチンの方はその番地を基準に処理を行うのである. このようなことを意識してプログラムを書くと, 少ない引数で柔軟な処理が可能になる.

もう一つの応用例を紹介しよう.

```
1         SUBROUTINE SUM2(X, IMAX, IX, JX, XSUM)
2         REAL X(IMAX,JX)
3         XSUM = 0.
4         DO 100 I=1,IX
5         DO 100 J=1,JY
6           XSUM = XSUM + X(I,J)
```

```
7      100 CONTINUE
8          END
```

これは 2 次元配列 X(I,J) の一部の和をとるものであるが, これを

```
1          REAL X(102,102)
2          ........
3          CALL SUM2(X(2,2), 102, 100, 100, XSUM)
4          ........
```

という具合に使うと, 102 × 102 の 2 次元配列のデータのうち, 周辺を残して真中の 100 × 100 の部分, すなわち, (2,2)–(101,101) を対角線とする部分の和をとることができる. MATH1 のルーチンで, このような使い方を想定しているものはないが, GRPH2 のコンターを描くルーチンなどで応用すると便利である.

なお, サブルーチンに渡すことのできる引数のなかに,「手続き名」がある. これは, 他の引数と異なり「変数」ではないので, 動的な変更はできないが, これが指定できるおかげで, MATH1 の VRFNA/VIFNA の様に, サブルーチンを呼ぶ際に使うべき関数を指定することが可能になる. この場合, 実引数に指定する手続き名は, サブルーチンを呼ぶプログラムの中で EXTERNAL 文または INTRINSIC 文により手続き名であることを宣言しておかなければならない.

1.5.9 EXTERNAL 文と INTRINSIC 文

EXTERNAL 文は, そこに指定された関数名が外部関数であることを宣言するものであり, INTRINSIC 文は, そこに指定された関数名が組み込み関数であることを宣言するものである. これらは, 上記のように関数名をサブルーチンの実引数として使用する時に必ず必要なものであるが, それ以外も重要な意味がある.

SIN や COS のような組み込み関数は, FORTRAN 規格によって処理系が用意しなければならないが, FORTRAN 規格はさらに処理系が規格外の組み込み関数を用意することを許している. したがって, 各処理系は FORTRAN 規格に定められた組み込み関数よりも多くの組み込み関数を持っているのが普通である. そして当然のことながら, それら拡張された組み込み関数の名前に関しては, 処理系によって全く異なる. ということは, FORTRAN 規格にない関数名で, EXTERNAL 文や INTRINSIC 文によって宣言されていない関数名は, 処理系によって組み込み関数として解釈されたり外部関数として解釈されたりすることがある.

例えば, 自分で外部関数を定義して, それを引用しているつもりでも, たまたまその処理系が同名の組み込み関数を持っていた場合には, ユーザー定義の関数ではなく, 処理系が用意した関数が呼ばれてしまう可能性がある. 関数名の長さは 6 文字以内と制限されている上, 人間の発想も多様性に限りがあるので, この手の衝突は意外に多い. しかも, この種の原因でプログラムが暴走したりすると, その原因を突き止めるのはかなり難しい. (自分のプログラムそのものには全く誤りがないのだから)

そこで, 処理系に依存しないプログラムを書くためには, そのプログラム単位で引用している外部関数名は全て EXTERNAL 文に指定することが望ましい. また, FORTRAN 規格にない処理系固有の関数を引用している場合には, INTRINSIC 文に指定すべきである.

なお, 当然のことながら, 電脳ライブラリが用意する関数は全て「外部関数」である. 処理系から見れば電脳ライブラリを含めて全て「ユーザープログラム」であるから, 自分が書かなかったからといって組み込み関数と混同することのないように注意されたい.

1.5.10 COMMON 文

前述の EQUIVALENCE 文が同じプログラムの中の変数を結合させるものであったのに対し, COMMON 文は異なったプログラム単位 (サブルーチンなど) の中の変数同志を結合させるものである.

初期の FORTRAN では引数によるデータの引渡しに時間がかかったため, COMMON 文によりメインプログラムからサブルーチンへデータを引き渡す方法が多用された歴史がある. しかし, 現在ではデータの引渡しに要する時間は, どちらの方法でもほとんど変らない. それより, COMMON 文を使うと, どのサブルーチンで値が代入されているのかわかりにくくなってしまうので, 可読性を高めるという見地から, COMMON 文によるサブルーチンへのデータ引渡はお勧めできない.

電脳ライブラリでも余程特殊な理由 (内部の作業領域確保など) がない限り, COMMON 文は使用していない. 各サブルーチンへのデータの引渡は全て引数を使っている.

1.5.11 その他

その他, FORTRAN でプログラムを書く場合の注意を以下に列挙する.

- **整数のオーバーフロー :**

  ほとんどのシステムでは, 整変数がオーバーフローしても一切の警告を発する事なく, 上位の桁を切捨てて何事もなかったかのように計算を続ける.

  これは非常に危険な事であるが, 乱数を発生させるプログラムなどでは意識的にオーバーフローさせている場合もある.

  とにかく, 大きな整数を扱う場合には, 注意が必要である.

- **整数の切捨て及び剰余 :**

  実数値を整変数に代入すると, 当然小数点以下は切捨てられるが, その切捨ては常に「0 に近くなるように」切捨てられる. これは, 正の範囲では数学的な切捨てと同じであるが, 負の範囲では異なる. (-1.5 が切捨てられると -1 になってしまう) また同様に, 組み込み関数の `MOD` も負の範囲で数学的な剰余の定義と異なる.

  数学的な切捨て及び剰余を求めたい場合には, FNCLIB または INTLIB のルーチンを使う.

- **実数のべき乗 :**

  べき乗計算 `X = B**N` では, 基数 `B` の丸め誤差が `N` 倍になって現れるので, 丸め誤差だけでも `N` が大きいとかなりの誤差を生じる. さらに, 計算の過程で生じる誤差が加わると, 思わぬ誤差を生じてしまう. しかも, 処理系によってこの誤差の生じかたが大きく異なるのでたちが悪い.

  べき乗の精度が気になるところでは, その部分だけ倍精度演算をするなどの安全対策を心がけた方がよい. 特に, 基数 `B` の精度に気を付けること. たとえ, 1 つの計算機で所定の精度が得られても, 他の計算機で同じ結果が得られる保証はない.

- **定数の精度 :**

  FORTRAN 規格では「定数は, それを表す文字列そのものが型と値を決める」とされている. した

がって, 定数の書き方によって値だけでなく, その精度も決められることになる. 例えば

```
1       DOUBLE PRECISION X
2       ....
3       X = 3.8*X*(1.-X)
4       ....
```

と書くと, せっかく倍精度計算をしているのにも関わらず, 定数 3.8 や 1. が単精度の定数なので, 全体として倍精度計算の精度が出ない. 正確にいうと 3.8 という定数は X との乗算が行われる前に, 倍精度実数に変換されるのであるが, その倍精度実数の値はあくまで単精度の 3.8 であり, 倍精度の 3.8D0 とは値が異なる. このようなところでは, 必ず

```
1       X = 3.8D0*X*(1.D0-X)
```

と書かなければならない. これは, DATA 文中の定数も同じである.

第2章　SYSLIB：内部変数管理, メッセージ出力

2.1　概要

MATH1 下位基本数学処理ライブラリの (そしてまた電脳ライブラリの) 最下位に位置するパッケージで, 内部変数の管理やメッセージの出力をする. システムに依存する定数を管理したり, ファイル操作をおこなったりするルーチンが含まれているが, ソースレベルでの機種依存性はない.

2.2　サブルーチンのリスト

GL*p*GET(CP,IPARA)	内部変数を参照する.
GL*p*SET(CP,IPARA)	内部変数を変更する.
GL*p*STX(CP,IPARA)	内部変数を変更する (実行時オプションによる変更を許す).
MSGDMP(CLEV,CSUB,CMSG)	メッセージを出力する.
RTPGET(CPFIX,CP,IPARA,N)	実行時オプションから内部変数を取得する.
RTCGET(CPFIX,CP,CPARA,N)	実行時オプションから内部変数 (文字変数) を取得する.
CFSRCH(CPL,NP,CFL,NF,CFN)	パス名の先頭部分と末尾部分のリストの組合せから, 存在するファイル名を探す.

2.3　関数のリスト

LCHREQ(CH1,CH2)	大文字・小文字の区別なく文字列を比較する.
IUFOPN()	利用可能な入出力装置番号を返す.

2.4　サブルーチンの説明

2.4.1　GL*p*GET/GL*p*SET(GL*p*STX)

1. 機能

   MATH1 で (さらには電脳ライブラリ全体で) 使用する内部変数を参照/変更する. (GL*p*STX は実行時オプションによる変更を許す.) 内部変数はすべて参照可能であるが, システムに依存するような内部変数は変更できない.

2. 呼び出し方法

   ```
   CALL GLpGET(CP,IPARA)
   CALL GLpSET(CP,IPARA)
   CALL GLpSTX(CP,IPARA)
   ```

3. パラメーターの説明

CP	(C*8)	内部変数の名前.
IPARA	(I,R,L)	内部変数の値.

以下に CP として指定できる名前のリストを記す.

'NBITSPW'	(I)	1 語のビット長. システム依存 (ふつうは 32).
'NCHRSPW'	(I)	1 語の文字長. システム依存 (ふつうは 4).
'INTMAX'	(I)	1 語で表現できる最大の整数. システム依存 (備考参照).
'REALMAX'	(R)	1 語で表現できる最大の実数. システム依存 (備考参照).
'REALMIN'	(R)	1 語で表現できる最小の正の実数. システム依存 (備考参照).
'IIUNIT'	(I)	標準入力装置番号. システム依存 (ふつうは 5).
'IOUNIT'	(I)	標準出力装置番号. システム依存 (ふつうは 6).
'MSGUNIT'	(I)	メッセージを出力する出力装置番号 (初期値は'IOUNIT' と同じ).
'MAXMSG'	(I)	出力する最大メッセージ数 (初期値は 20).
'MSGLEV'	(I)	出力するメッセージレベル. 0 以下ならすべてのレベルのメッセージを出力する; 1 なら警告, エラーメッセージを出力する; 2 以上ならエラーメッセージだけ出力する (初期値は 0).
'NLNSIZE'	(I)	一行に出力するメッセージの最大文字数 (初期値は 78). メッセージを出力する際, 各行の第一カラムにはブランクを書き出すので, 実際の一行の長さは 'NLNSIZE'+1 である.
'LMISS'	(L)	欠損値処理をおこなうかどうかを指定する. .TRUE. ならおこなう, .FALSE. ならおこなわない (初期値は.FALSE.).
'IMISS'	(I)	欠損値処理をおこなうときの整数型の欠損値を指定する (初期値は 999).
'RMISS'	(R)	欠損値処理をおこなうときの実数型の欠損値を指定する (初期値は 999.0).
'IUNDEF'	(I)	不定であることを示す整数型の値 (初期値は–999).
'RUNDEF'	(I)	不定であることを示す実数型の値 (初期値は–999.0).
'LEPSL'	(L)	実数値を比較するとき, 誤差を含めた判断をするかどうかを指定する (第 8,9,10,11 章参照). .TRUE. なら誤差を含めた判断をする; .FALSE. なら誤差を含めた判断をしない (初期値は.FALSE.).
'REPSL'	(R)	誤差を含めた判断をするときの相対誤差の基準値. システム依存 (備考参照).
'RFACT'	(R)	誤差を含めた判断をするとき, 相対誤差の基準値にかけるスケーリングファクター (初期値は 1.0). 'REPSL'×'RFACT' が相対誤差として用いられる.

4. 備考

(a) 内部変数を管理するための下請けルーチンとして以下のものがある.

GLPQNP(NCP)	内部変数の総数 NCP を求める.
GLPQID(CP,IDX)	内部変数 CP の位置 IDX を求める.
GLPQCP(IDX,CP)	IDX の位置にある内部変数の名前 CP を参照する.
GLPQVL(IDX,IPARA)	IDX の位置にある内部変数の値 IPARA を参照する.
GLPSVL(IDX,IPARA)	IDX の位置にある内部変数の値 IPARA を変更する.

(b) GL*p*GET は上に述べた GLPQID を呼んで内部変数の位置を求め, GLPQVL によって値を参照する; GL*p*SET は GLPQID を呼んで内部変数の位置を求め, GLPSVL によって値を設定する. したがって指定した内部変数名が見つからないとき, エラーメッセージは GLPQID が出力する.

(c) IPARA としては適切な型の定数または変数を指定すること.

(d) 内部変数の名前は, 大文字・小文字の区別なく文字列を比較する文字関数 LCHREQ を用いてチェックされるので, LCHREQ が正しく移植されていれば, 小文字で指定してもよい.

(e) 'REALMIN' があらわす最小の正の実数は, 仮数部の最上位の桁が 0 でないものとする. これは必ずしも UNDER FLOW を起こす直前の値ではないので注意すること. すなわち, 通常実数型の変数は仮数部の最上位の桁が 0 でないように規格化されるが, システムによってはこのような規格化をすると指数部が足りなくなるような小さな値でも, 仮数部の上位の桁を 0 にすることで表現するものがある. このような値は一応実数として扱われるが, 通常の実数の精度はない. つまり 'REALMIN' は通常の精度を持つ最小の正の実数である.

(f) 'REPSL' があらわす相対誤差の基準値は, 実数表現の相対誤差の最大値の 10 倍とする. この 10 倍という値には任意性があるが計算結果の精度の目安として安全率を 10 倍としたものである. 実数表現の相対誤差は内部表現の形から論理的にもとめることができるが, 実際に実数の仮数部の最下位 bit を on/off することでも求めることができる. 1 語の中の仮数部の位置はシステムによるが, 1 語の最下位 bit が仮数部の最下位 bit であることが多い. src/env2/repsl にこのようなチェックをするプログラムが含まれている.

(g) システムに依存するパラメーターのうち'INTMAX', 'REALMAX', 'REALMIN', 'REPSL' は大きく分けて, メインフレーム系の大型計算機と UNIX 系の計算機とでは次の値をとることが多い.

	メインフレーム系	UNIX 系
'INTMAX'	2147483647	2147483647
'REALMAX'	0.7237005E+76	3.4028236E+38
'REALMIN'	0.5397605E-78	1.1754944E-38
'REPSL'	0.95E-5	1.2E-6

2.4.2 GLCGET/GLCSET(GLCSTX)

1. 機能

   MATH1 で使用する文字型の内部変数を参照/変更する. (GLCSTX は実行時オプションによる変更を許す.)

2. 呼び出し方法

```
CALL GLCGET(CP,CPARA)
CALL GLCSET(CP,CPARA)
CALL GLCSTX(CP,CPARA)
```

3. パラメーターの説明

CP (C*8) 内部変数の名前.
CPARA (C*(*)) 文字型の内部変数の値.

以下に CP として指定できる名前のリストを記す.

'DCLRC' (C) 実行時オプションを読み込む外部ファイル名 (標準ライブラリにおける値は.dclrc). ただし, この内部変数の値がそのままファイル名として用いられるわけではない. 備考を参照のこと.
'DUPATH' (C) ユーザー用の各種データベースファイルをおくパス名. 標準ライブラリにおける初期値は' ' (空白: これはカレントディレクトリをあらわす).
'DSPATH' (C) システムがあらかじめ用意した各種データベースファイルをおくパス名. 標準ライブラリにおける値は, インストール時に$(DCLDBASE) が示す値となる.

4. 備考
   (a) 内部変数を管理するための下請けルーチンとして以下のものがある.
   GLCQNP(NCP) 内部変数の総数 NCP を求める.
   GLCQID(CP,IDX) 内部変数 CP の位置 IDX を求める.
   GLCQCP(IDX,CP) IDX の位置にある内部変数の名前 CP を参照する.
   GLCQVL(IDX,CPARA) IDX の位置にある内部変数の値 IPARA を参照する.
   GLCSVL(IDX,CPARA) IDX の位置にある内部変数の値 IPARA を変更する.
   (b) GLCGET は上に述べた GLCQID を呼んで内部変数の位置を求め, GLCQVL によって値を参照する; GLCSET は GLCQID を呼んで内部変数の位置を求め, GLCSVL によって値を設定する. したがって指定した内部変数名が見つからないとき, エラーメッセージは GLCQID が出力する.
   (c) 内部変数の名前は, 大文字・小文字の区別なく文字列を比較する文字関数 LCHREQ を用いてチェックされるので, LCHREQ が正しく移植されていれば, 小文字で指定してもよい.
   (d) このルーチンが管理する内部変数を, 外部ファイルを通した実行時オプションによって変更することはできない. (なぜなら, 再帰的なループに入り込んでしまうため.)
   (e) 'DCLRC' が示す内部変数は, その内部変数の値がファイル名としてそのまま利用されるわけではない. 実際に利用されるファイル名は GLQFNM が決める. 標準ライブラリではまずカレントディレクトリを探す. 次に内部変数'DUPATH' (ユーザーの指定するパス名) のさすディレクトリを探す. 最後に内部変数'DSPATH'(システムがあらかじめ用意するパス名) のさすディレクトリを探す.

2.4.3 MSGDMP

1. 機能
   メッセージを出力する. 指定するパラメーターに応じてプログラムは強制終了または続行の制御を受ける.
2. 呼び出し方法
   CALL MSGDMP(CLEV,CSUB,CMSG)
3. パラメーターの説明

CLEV (C*1) メッセージのレベルを'E','W','M' のうちから 1 つ指定する. メッセージレベルの意味と作用は以下のとおり.

'E': 重大なエラー. プログラムは強制終了する.

'W': 警告. 何らかの処置をおこなってプログラムは続行する.

'M': メッセージ. 何の変更もせずに (あるいは特に問題とならない変更をして) プログラムは続行する.

CSUB (C*6) MSGDMP を呼んでいるサブルーチン名を指定する.

CMSG (C*(*)) 出力するメッセージ.

4. 備考

(a) 'E' が指定されたとき, システムに依存したエラー処理をおこなって強制終了するルーチン OSABRT を呼ぶ. (詳しくは 3.3.5 節参照のこと.)

(b) サブルーチンの作用としては, 'W' と 'M' は同じ作用をする. 2 つのパラメーターの違いはユーザーの側で使い分けること.

(c) CMSG の最大文字数は 174 文字.

(d) メッセージを出力する出力装置番号は内部変数'MSGUNIT' (初期値は 6) で決まる.

(e) 出力する最大メッセージ数は内部変数'MAXMSG' (初期値は 20) で決まる.

(f) 1 行に書く文字数は内部変数'NLNSIZE' (初期値は 78) で決まる.

(g) 出力するメッセージのレベルは内部変数'MSGLEV' (初期値は 0) で決まる.

(h) 下位に MSZDMP という名前のサブルーチンがある.

2.4.4 RTPGET/RTCGET

1. 機能

実行時オプションから内部変数を取得する.

2. 呼び出し方法

```
CALL RTPGET(CPFIX,CP,IPARA,N)
CALL RTCGET(CPFIX,CP,CPARA,N)
```

3. パラメーターの説明

CPFIX	(C*(*))	変数名の前につける接頭辞.
CP	(C(*)*8)	変数名.
IPARA	(I,R,L(*))	変数の値.
CPARA	(C(*)*80)	変数の値.
N	(I)	内部変数の数.

4. 備考

(a) 接頭辞とは, たとえば GL*p*GET/GL*p*SET の管理する変数の場合 'GL_' のように, xx*p*GET/xx*p*SET の 'xx' 部分と '_' の組合せからなる.

(b) CP, CPARA として指定する文字型配列は, 1 要素長の長さがそれぞれ 8, 80 でなければならない.

(c) 実行時オプションで該当する変数が指定されていなければ, IPARA, CPARA は変更されない.

(d) 実行時オプションは, 標準的には環境変数, コマンドライン引数, 外部ファイルを通して入手するこ

とを念頭においているが, 具体的な実装方法は機種に依存する. しかし, OSLIB (第 3 節参照) が正しく移植されていることを前提として, 実際には以下のルーチンが下請けをおこなっている.

RTPENV(CPFIX,CP,IPARA)	環境変数の値を得る.
RTPOPT(CPFIX,CP,IPARA)	コマンドラインオプションの値を得る.
RTPXFL(CPFIX,CP,IPARA)	外部ファイルからオプションの値を得る.
RTCENV(CPFIX,CP,CVAL)	環境変数の値 (文字型) を得る.
RTCOPT(CPFIX,CP,CVAL)	コマンドラインオプションの値 (文字型) を得る.
RTCXFL(CPFIX,CP,CVAL)	外部ファイルからオプションの値 (文字型) を得る.

(e) 実行時オプションの指定方法として, 環境変数, コマンドライン引数, 外部ファイルが使用できる場合, その効力は, コマンドライン引数, 環境変数, 外部ファイルの順である. (すべて指定されたら, コマンドライン引数による指定を採用する.)

(f) 実行時オプションが定義できない機種では, 実行時オプションが何も指定されなかったものとみなされる.

2.4.5 CFSRCH

1. 機能

   パス名の先頭部分と末尾部分のリストの組合せから, 存在するファイル名を探す.

2. 呼び出し方法

   CFSRCH(CPL,NP,CFL,NF,CFN)

3. パラメーターの説明

CPL	(C(NP)*80)	パス名の先頭部分のリスト. 長さ 80 の文字型配列.
NP	(I)	配列 CPL の長さ.
CFL	(C(NF)*80)	パス名の末尾部分のリスト. 長さ 80 の文字型配列.
NF	(I)	配列 CFL の長さ.
CFN	(C*(*))	最初に見つかったファイル名を返す文字型変数.

4. 備考

   (a) たとえばファイル名 /usr/local/bin/dclfrt について, 先頭部分とは /usr/local/bin/ をさす; 末尾部分とは dclfrt をさす.

   (b) このサブルーチンは, CPL(1) から CPL(NP) の先頭部分について, CFL(1) から CFL(NF) の末尾部分との組合せからなるパス名をもつファイルが存在するかどうかを調べて, 最初に見つかったファイル名を返す. そのようなファイル名が存在しなければ空白からなる文字列を返す.

   (c) パス名の先頭部分として空白からなる文字列を与えると, カレントディレクトリについて, パス名の末尾部分からなるファイルを検索する.

   (d) このルーチンはシステムに依存する. つまり, CFSRCH は UNIX のようなツリー構造を持ったファイルシステムを念頭において作成されており, メインフレーム系のようなファイルシステムにおいてはファイル名検索のルールを別途定義してやる必要がある. しかし, 現在用意されているルーチンでも, 内部的には単に 2 種類の文字型リストを組合せてファイルが存在するかどうかを調べているだけなので, 「先頭部分」, 「末尾部分」という定義にこだわらずに, ファイル名を構成する要素としてこれらを与えてやればそのまま利用することができる. ただし, 「パス名の先頭部分として

空白を指定するとカレントディレクトリを検索する」という仕様の制約から, 空白を指定しないようにするか, 空白が指定されたときの動作を別途定義する (たとえば, 何もしない) 必要がある.

2.5 関数の説明

2.5.1 LCHREQ

1. 機能

   大文字・小文字の区別なく 2 つの文字列を比較する.

2. 呼び出し方法

   ```
   LCHREQ(CH1,CH2)
   ```

3. パラメーターの説明

LCHREQ	(L)	2 つの文字列が (大文字・小文字の区別なく) 等しいとき .TRUE. を返す論理関数値.
CH1, CH2	(C*(*))	比較する文字列.

4. 備考

   (a) CH1 と CH2 の長さが異なるときは, 長い方の文字列と同じ長さになるまで, 短い方の文字列の後ろにブランクがあるものとして比較する.

2.5.2 IUFOPN

1. 機能

   存在し, 接続されていない (すなわち利用可能な) もっとも小さい入出力装置番号を返す.

2. 呼び出し方法

   ```
   IUFOPN()
   ```

3. パラメーターの説明

IUFOPN	(I)	装置番号を返す整数値.

4. 備考

   (a) この関数は, 1 から順に 99 までの装置番号に対応する入出力装置の状態を問い合わせ, 存在し, 接続されていない (オープンされていない) 最初の装置番号を返す. すべて利用不可能なときは 0 を返す.

第3章　OSLIB：システム依存ルーチン

3.1　概要

システムに依存するサブルーチンをまとめたパッケージ. FORTRAN の規格外ではあるが多くの処理系でサポートされていると思われる手続きの中でよく使われそうなもののインターフェイス (サブルーチン名) を揃えることがこのパッケージの目的でである. C で標準的にサポートされているルーチンは, その規格に準拠している. なお, このパッケージの中の手続きがサポートされないようなシステムでは, ダミールーチンとしてもよいものも多い. くわしくは, 以下の「サブルーチンの説明」を参照されたい.

MATH1 のパッケージの中でシステム依存関数を含むのは, この OSLIB だけである.

3.2　サブルーチンのリスト

`OSEXEC(CMD)`	OS コマンド `CMD` を実行する.
`OSGENV(CENAME,CEVAL)`	環境変数 `CENAME` の値を `CVAL` として得る.
`OSQARN(N)`	コマンドライン引数の数 `N` を得る.
`OSGARG(N,CHAR)`	`N` 番目のコマンドライン引数 `CHAR` を得る.
`OSABRT`	エラー処理をおこなってプログラムを強制終了する.

3.3　サブルーチンの説明

3.3.1　OSEXEC

1. 機能
   OS コマンドを実行する.
2. 呼び出し方法
   ```
   CALL OSEXEC(CMD)
   ```
3. パラメーターの説明
   `CMD`　`(C*(*))`　コマンド名.
4. 備考
   (a) このサブルーチンに対応する ANSI C 関数は `system` である.

3.3.2　OSGENV

1. 機能
   環境変数の値を取得する.

2. 呼び出し方法

```
CALL OSGENV(CENAME,CVAL)
```

3. パラメーターの説明

```
CENAME  (C*(*))  環境変数名.
CVAL    (C*(*))  環境変数の値.
```

4. 備考

(a) このサブルーチンに対応する ANSI C 関数は getenv である.

3.3.3 OSQARN

1. 機能

コマンドライン引数の数を返す.

2. 呼び出し方法

```
CALL OSQARN(N)
```

3. パラメーターの説明

```
N  (I)  コマンドライン引数の数.
```

4. 備考

(a) コマンドライン引数が得られないシステムでは, 0 が返される.

(b) このサブルーチンに対応する ANSI C 関数はない.

(c) HITACHI の FORTRAN では、コンパイルオプションで挙動が変わるので注意.

3.3.4 OSGARG

1. 機能

N 番目のコマンドライン引数を返す.

2. 呼び出し方法

```
CALL OSGARG(N,CHAR)
```

3. パラメーターの説明

```
N     (I)       コマンドライン引数の位置.
CHAR  (C*(*))   N 番目のコマンドライン引数.
```

4. 備考

(a) コマンドライン引数が得られないシステムでは, 常に空白の文字列が返される.

(b) 0 番目のコマンドライン引数は, 通常コマンド名であるが, コマンド名が得られないシステムでは空白となる.

(c) このサブルーチンに対応する ANSI C 関数はない.

(d) HITACHI の FORTRAN では、コンパイルオプションで挙動が変わるので注意。

3.3.5 OSABRT

1. 機能

システムに依存したエラー処理をおこなってプログラムを強制終了する.

2. 呼び出し方法

```
CALL OSABRT
```

3. パラメーターの説明

なし.

4. 備考

(a) たとえば, トレースバック情報などを出力してプログラムを強制終了するように実装される.

第4章　FNCLIB：基本関数 (最大整数, 剰余など)

4.1　概要

基本的な関数パッケージ.

4.2　関数のリスト

IGUS(RX)	RX を越えない最大の整数 ([RX]) を返す.
IMOD(IX,ID)	数学的な剰余 (IX mod ID) を求める.
RMOD(RX,RD)	数学的な剰余 (RX mod RD) を求める.
REXP(RX,IB,IE)	$RX \times IB^{IE}$ を求める.
RFPI()	円周率 π を返す.
RD2R(X)	角度の変換 (度からラジアン) をおこなう.
RR2D(X)	角度の変換 (ラジアンから度) をおこなう.

4.3　関数の説明

4.3.1　IGUS

1. 機能

   与えた実数を越えない最大の整数を求める.

2. 呼び出し方法

   IGUS(RX)

3. パラメーターの説明

RX	(R)	調べる実数値.
IGUS	(I)	最大の整数 [RX] 関数値.

4. 備考

   (a) FORTRAN の組込み関数 INT(X) は, X の値の小数点以下を切り捨てた値, つまり $|X| \geq 1$ ならば |X| 以下の最大の整数に X と同じ符号をつけた値, $|X| < 1$ ならばゼロを返す. たとえば INT(4.5)=4, INT(-4.5)=-4 である. いっぽう IGUS は, たとえば IGUS(4.5)=4, IGUS(-4.5)=-5 となる.

4.3.2 IMOD/RMOD

1. 機能

   数学的な剰余を求める. IMOD は整数用 (IX mod ID), RMOD は実数用 (RX mod RD) である.

2. 呼び出し方法

   IMOD(IX,ID)

   RMOD(RX,RD)

3. パラメーターの説明

IX	(I)	調べる整数値.
ID	(I)	除数.
IMOD	(I)	剰余関数値.
RX	(R)	調べる実数値.
RD	(R)	除数.
RMOD	(R)	剰余関数値.

4. 備考

   (a) FORTRAN の組込み関数 MOD(X,Y) は, X-INT(X/Y)*Y を返す. たとえば MOD(4,3)=1, MOD(-4,3)=-1 である. いっぽう IMOD は, たとえば IMOD(4,3)=1, IMOD(-4,3)=2 となる. また RMOD は, たとえば RMOD(4.0,3.0)=1.0, RMOD(-4.0,3.0)=2.0 となる.

   (b) ID≠0, RD≠0 でなければならない.

4.3.3 REXP

1. 機能

   RX × $\mathrm{IB}^{\mathrm{IE}}$ を求める.

2. 呼び出し方法

   REXP(RX,IB,IE)

3. パラメーターの説明

RX	(R)	機能欄参照.
IB	(I)	機能欄参照.
IE	(I)	機能欄参照.
REXP	(R)	求める関数値.

4. 備考

   (a) FORTRAN の指数計算は, 機種によってかなり精度の悪いものがあるのでこのルーチンは作成された. 実際, REXP は RX*IB**REAL(IE) と書くのに比べて一般的にけた落ちが少ないが, もっとも高速で精度の高い計算をするためには倍精度で指数計算をおこなうのがよい.

   (b) IB≠0 でなければならない.

4.3.4 RFPI

1. 機能

円周率 π を返す.

2. 呼び出し方法

```
RFPI()
```

3. パラメーターの説明

RFPI	(R)	円周率 π を返す実数関数値.

4. 備考

(a) なし.

4.3.5 RD2R/RR2D

1. 機能

角度の変換をおこなう.

2. 呼び出し方法

```
RD2R(X)
RR2D(X)
```

3. パラメーターの説明

X	(R)	RD2R に対しては「度」, RR2D に対しては「ラジアン」を単位とする実数値.
RD2R	(R)	「ラジアン」を単位とした値を返す実数関数値.
RR2D	(R)	「度」を単位とした値を返す実数関数値.

4. 備考

(a) なし.

第5章　SUBLIB：基本サブルーチン (自然数列の生成など)

5.1　概要

基本的なサブルーチンパッケージ.

5.2　サブルーチンのリスト

VIGNN(IX,N,JX)	自然数列を生成して整数型配列で返す.
VRGNN(RX,N,JX)	自然数列を生成して実数型配列で返す.
DXFLOC(ND,NS,NP,NCP)	添字の値から配列要素の位置を求める.
DXILOC(ND,NS,NP,NCP)	配列要素の位置から添字の値を求める.

5.3　サブルーチンの説明

5.3.1　VIGNN/VRGNN

1. 機能

   自然数列を生成して整数型配列/実数型配列で返す.

2. 呼び出し方法

   ```
   CALL VIGNN(IX,N,JX)
   CALL VRGNN(RX,N,JX)
   ```

3. パラメーターの説明

IX	(I)	自然数列を返す整数型配列.
RX	(R)	自然数列を返す実数型配列.
N	(I)	数列を構成する配列要素の個数.
JX	(I)	数列を構成する配列要素の間隔.

4. 備考

   (a) なし.

5.3.2　DXFLOC/DXILOC

1. 機能

   DXFLOC：添字の値から配列要素の位置を求める.

   DXILOC：配列要素の位置から添字の値を求める.

2. 呼び出し方法

```
CALL DXFLOC(ND,NS,NP,NCP)
CALL DXILOC(ND,NS,NP,NCP)
```

3. パラメーターの説明

ND	(I)	配列の次元数.
NS	(R)	配列の寸法を収めた長さ ND の整数型配列.
NP	(I)	配列の添字を収めた長さ ND の整数型配列. DXFLOC では入力パラメータ; DXILOC では出力パラメータ.
NCP	(I)	配列要素の位置. DXFLOC では出力パラメータ; DXILOC では入力パラメータ.

4. 備考

(a) FORTRAN では, たとえば, 3 次元配列 X(3,4,5) が宣言してあったとき, X(2,2,2) であらわされる配列要素の位置は記憶列の先頭から数えて 17 番めである. (FORTRAN では, 記憶列の先頭から順に, X(1,1,1), X(2,1,1), X(3,1,1), ..., X(1,2,1), X(2,2,1), X(3,2,1), ..., X(1,1,2), X(2,1,2), X(3,1,2), ..., のように添字が動く.) DXFLOC では, ND=3, NS(1)=3, NS(2)=4, NS(3)=5, NP(1)=2, NP(2)=2, NP(3)=2, とすると, NCP=17 が返される; DXILOC では, ND=3, NS(1)=3, NS(2)=4, NS(3)=5, NCP=17 とすると, NP(1)=2, NP(2)=2, NP(3)=2 が返される.

(b) 与える引数の妥当性 (たとえば, NP の値が 1 以上で対応する NS の値より小さいか, など) はチェックされない.

第6章　CHRLIB：文字列の左・右詰め, 反転, 空白処理

6.1　概要

文字列を扱う関数・サブルーチンパッケージ.

以下でいう' 空白' とは, NULL 文字および空白文字 (ASCII コード系においてはそれぞれ CHAR(0), CHAR(32) に対応) のことである.

6.2　サブルーチンのリスト

CLADJ(CHR)　文字列を左詰めする.
CRADJ(CHR)　文字列を右詰めする.
CRVRS(CHR)　文字列を反転する.

6.3　関数のリスト

LENB(C), LENY(C)　先行する空白の数を求める.
LENC(C), LENZ(C)　後続する空白を除いた文字数を求める.

6.4　サブルーチンの説明

6.4.1　CLADJ/CRADJ

1. 機能
   文字列の左詰め/右詰めをおこなう.
2. 呼び出し方法
   ```
   CALL CLADJ(CHR)
   CALL CRADJ(CHR)
   ```
3. パラメーターの説明
   CHR　(C*(*))　操作をほどこす文字列. 入力パラメータでもあり出力パラメータでもある.
4. 備考
   (a) 有効な文字列の長さは LEN(CHR) (LEN は FORTRAN の組み込み関数) で決まる.
   (b) 文字列の左詰め/右詰めとは, 先行/後続する空白の数だけ文字列を左/右にシフトすることをいう. シフトした後に残る領域には空白文字が埋められる.

(c) CLADJ, CRADJ は途中の空白を詰めない.

6.4.2 CRVRS

1. 機能
   文字列を反転する.
2. 呼び出し方法
   CALL CRVRS(CHR)
3. パラメーターの説明

CHR	(C*(*))	操作をほどこす文字列. 入力パラメータでもあり出力パラメータでもある.

4. 備考
   (a) 有効な文字列の長さは LEN(CHR) (LEN は FORTRAN の組み込み関数) で決まる.

6.5 関数の説明

6.5.1 LENB/LENY

1. 機能
   先行する空白の数を求める.
2. 呼び出し方法
   LENB(C)
   LENY(C)
3. パラメーターの説明

C	(C*(*))	調べる文字列.
LENB, LENY	(I)	先行する空白の数を返す関数値.

4. 備考
   (a) LENB と LENY の違いは, C がすべて空白からなるとき LENB は LEN(C)–1 を返すのに対し, LENY は LEN(C) を返す (LEN は FORTRAN の組込み関数).

6.5.2 LENC/LENZ

1. 機能
   後続する空白を除いた文字数を求める.
2. 呼び出し方法
   LENC(C)
   LENZ(C)
3. パラメーターの説明

C	(C*(*))	調べる文字列.
LENC, LENZ	(I)	後続する空白を除いた文字数を返す関数値.

4. 備考

(a) LENC と LENZ の違いは, C がすべて空白からなるとき LENC は 1 を返すのに対し, LENZ は 0 を返す.

第7章　XFCLIB：文字列の数値化

7.1　概要

文字列を数値化するサブルーチンパッケージ

7.2　関数のリスト

IFROMC(CX)　整数文字列 CX のあらわす整数値を返す.
LFROMC(CX)　論理定数を表現する文字列 CX のあらわす論理値を返す.
RFROMC(CX)　指数表現を含む実数文字列 CX のあらわす実数値を返す.

7.3　関数の説明

7.3.1　IFROMC

1. 機能
   整数文字列のあらわす整数値を返す.
2. 呼び出し方法
   ```
   IFROMC(CX)
   ```
3. パラメーターの説明

CX	(C*(*))	調べる文字列.
IFROMC	(I)	求める整数値.

4. 備考
   (a) 下請けの関数として符号なし整数文字列を解釈する JFROMC がある.

7.3.2　LFROMC

1. 機能
   論理定数を表現する文字列のあらわす論理値を返す.
2. 呼び出し方法
   ```
   LFROMC(CX)
   ```
3. パラメーターの説明

CX	(C*(*))	調べる文字列.
LFROMC	(L)	求める論理値.

4. 備考
   (a) 論理定数は FORTRAN における表現方法にしたがう. すなわち .TRUE., .FALSE. を解釈する.

7.3.3 RFROMC

1. 機能

   指数表現を含む実数文字列のあらわす実数値を返す.

2. 呼び出し方法

   ```
   RFROMC(CX)
   ```

3. パラメーターの説明

CX	(C*(*))	調べる文字列.
RFROMC	(R)	求める実数値.

4. 備考

   (a) 下請けの関数として指数部なしの実数文字列を解釈する FFROMC がある.

第8章　LRLLIB：実数値の比較

8.1　概要

2 つの実数値を比較する論理型関数パッケージ．以下では総称名を示す．たとえば LREQ に対しては, ほかに LREQ0, LREQ1 という名前の関数も存在する．LREQ0 は誤差を含んだ比較をおこなわない; LREQ1 は誤差を含んだ比較をおこなう．LREQ は内部変数'LEPSL' が.TRUE. なら誤差を含んだ比較をおこなう; .FALSE. なら誤差を含んだ比較をおこなわない (初期値は.FALSE.)．誤差を含んだ比較をおこなうときの相対誤差は, 内部変数'REPSL' と'RFACT' の積で決まる．(詳しくは 2.4.1 節参照)

8.2　関数のリスト

LREQ(X,Y)	X と Y が等しいかどうか調べる．
LRNE(X,Y)	X と Y が等しくないかどうか調べる．
LRLT(X,Y)	X が Y より小さいかどうか調べる．
LRLE(X,Y)	X が Y 以下かどうか調べる．
LRGT(X,Y)	X が Y より大きいかどうか調べる．
LRGE(X,Y)	X が Y 以上かどうか調べる．
LREQA(X,Y,EPSL)	X が誤差 EPSL の範囲で Y と等しいかどうか調べる．
LRNEA(X,Y,EPSL)	X が誤差 EPSL の範囲で Y と等しくないかどうか調べる．
LRLTA(X,Y,EPSL)	X が誤差 EPSL の範囲で Y より小さいかどうか調べる．
LRLEA(X,Y,EPSL)	X が誤差 EPSL の範囲で Y 以下かどうか調べる．
LRGTA(X,Y,EPSL)	X が誤差 EPSL の範囲で Y より大きいかどうか調べる．
LRGEA(X,Y,EPSL)	X が誤差 EPSL の範囲で Y 以上かどうか調べる．

8.3　関数の説明

8.3.1　LREQ/LRNE/LRLT/LRLE/LRGT/LRGE

1. 機能

   2 つの実数値の大小関係を調べる．

2. 呼び出し方法

   ```
   LREQ(X,Y)
   LRNE(X,Y)
   LRLT(X,Y)
   LRLE(X,Y)
   ```

LRGT(X,Y)

LRGE(X,Y)

3. パラメーターの説明

X, Y	(R)	調べる 2 つの実数値.
LREQ	(L)	X と Y が等しいとき.TRUE. である論理関数値.
LRNE	(L)	X と Y が等しくないとき.TRUE. である論理関数値.
LRLT	(L)	X が Y より小さいとき.TRUE. である論理関数値.
LRLE	(L)	X が Y 以下のとき.TRUE. である論理関数値.
LRGT	(L)	X が Y より大きいとき.TRUE. である論理関数値.
LRGE	(L)	X が Y 以上のとき.TRUE. である論理関数値.

4. 備考

(a) なし.

8.3.2 LREQA/LRNEA/LRLTA/LRLEA/LRGTA/LRGEA

1. 機能

誤差を陽に指定して 2 つの実数値の大小関係を調べる.

2. 呼び出し方法

LREQA(X,Y,EPSL)

LRNEA(X,Y,EPSL)

LRLTA(X,Y,EPSL)

LRLEA(X,Y,EPSL)

LRGTA(X,Y,EPSL)

LRGEA(X,Y,EPSL)

3. パラメーターの説明

X, Y	(R)	調べる 2 つの実数値.
EPSL	(R)	大小関係を調べるときの誤差.
LREQA	(L)	X と Y が等しいとき.TRUE. である論理関数値.
LRNEA	(L)	X と Y が等しくないとき.TRUE. である論理関数値.
LRLTA	(L)	X が Y より小さいとき.TRUE. である論理関数値.
LRLEA	(L)	X が Y 以下のとき.TRUE. である論理関数値.
LRGTA	(L)	X が Y より大きいとき.TRUE. である論理関数値.
LRGEA	(L)	X が Y 以上のとき.TRUE. である論理関数値.

4. 備考

(a) なし.

第9章　BLKLIB：実数値と順序列

9.1　概要

指定した値が, 昇順に並んだ配列の何番目の区間に入っているかを調べる関数パッケージ. 大小比較はLRLLIB を使っておこなうので, 誤差を含んだ比較ができる (第 8 章参照). ただし, 区間間隔が誤差より小さい場合これらの関数は意味をなさないので注意すること.

9.2　関数のリスト

IBLKLT(RX,N,RR)	RX$(i-1) \le$ RR $<$ RX(i) を満たす i を求める.
	ただし RR $<$ RX(1) のとき 1 を返す; RX(N) $\le$ RR のとき N+1 を返す.
IBLKLE(RX,N,RR)	RX$(i-1) <$ RR $\le$ RX(i) を満たす i を求める.
	ただし RR $\le$ RX(1) のとき 1 を返す; RX(N) $<$ RR のとき N+1 を返す.
IBLKGT(RX,N,RR)	RX$(i) <$ RR $\le$ RX$(i+1)$ を満たす i を求める.
	ただし RR $\le$ RX(1) のとき 0 を返す; RX(N) $<$ RR のとき N を返す.
IBLKGE(RX,N,RR)	RX$(i) \le$ RR $<$ RX$(i+1)$ を満たす i を求める.
	ただし RR $<$ RX(1) のとき 0 を返す; RX(N) $\le$ RR のとき N を返す.

9.3　関数の説明

9.3.1　IBLKLT/IBLKLE/IBLKGT/IBLKGE

1. 機能

   指定した値が, 昇順に並んだ配列の何番目の区間に入っているかを調べる.

2. 呼び出し方法

   ```
   IBLKLT(RX,N,RR)
   IBLKLE(RX,N,RR)
   IBLKGT(RX,N,RR)
   IBLKGE(RX,N,RR)
   ```

3. パラメーターの説明

RX	(R)	昇順に並んだ実数型配列.
N	(I)	配列の寸法.
RR	(R)	調べる実数値.
IBLKLT	(I)	RX$(i-1) \leq$ RR $<$ RX(i) を満たす i を返す. ただし RR $<$ RX(1) のとき 1, RX(N) $\leq$ RR のとき N+1 を返す.
IBLKLE	(I)	RX$(i-1) <$ RR $\leq$ RX(i) を満たす i を返す. ただし RR $\leq$ RX(1) のとき 1, RX(N) $<$ RR のとき N+1 を返す.
IBLKGT	(I)	RX$(i) <$ RR $\leq$ RX$(i+1)$ を満たす i を返す. ただし RR $\leq$ RX(1) のとき 0, RX(N) $<$ RR のとき N を返す.
IBLKGE	(I)	RX$(i) \leq$ RR $<$ RX$(i+1)$ を満たす i を返す. ただし RR $<$ RX(1) のとき 0, RX(N) $\leq$ RR のとき N を返す.

4. 備考

(a) 区間間隔が誤差より小さい場合これらの関数は意味をなさないので注意すること.

第10章　GNMLIB：きりのよい打ち切り数

10.1　概要

きりのよい数を求めるサブルーチン・関数パッケージ. 大小比較は LRLLIB (第 8 章) を使っておこなうので, 誤差を含んだ比較ができる.

ここでいうきりのよい数とは, 1.0, 1.2, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0, 6.0, 8.0 $\times 10^n$ で表現できる数または 0 である. きりのよい数のリストはユーザーが設定・参照することもできる.

10.2　サブルーチンのリスト

GNLT(RX,BX,IP)	RX より小さいきりのよい数のうちで最大のものを求める.
GNLE(RX,BX,IP)	RX 以下のきりのよい数のうちで最大のものを求める.
GNGT(RX,BX,IP)	RX より大きいきりのよい数のうちで最小のものを求める.
GNGE(RX,BX,IP)	RX 以上のきりのよい数のうちで最小のものを求める. 結果は BX $\times 10^{\mathrm{IP}}$ の形で返される. きりのよい数のリストは次のサブルーチンで設定・参照できる.
GNSBLK(XB,NB)	きりのよい数のリストを設定する.
GNQBLK(XB,NB)	現在設定されているリストを参照する.
GNSAVE	現在設定されているリストを保存する.
GNRSET	GNSAVE が保存したリストで再設定する.

10.3　関数のリスト

RGNLT(RX)	RX より小さいきりのよい数のうちで最大のものを求める.
RGNLE(RX)	RX 以下のきりのよい数のうちで最大のものを求める.
RGNGT(RX)	RX より大きいきりのよい数のうちで最小のものを求める.
RGNGE(RX)	RX 以上のきりのよい数のうちで最小のものを求める.

10.4　サブルーチンの説明

10.4.1　GNLT/GNGT

1. 機能

   与えた実数値より小さい/大きいきりのよい数のうちで最大/最小のものを求める. 結果は BX $\times 10^{\mathrm{IP}}$ の形で返される.

2. 呼び出し方法

```
CALL GNLT(RX,BX,IP)
CALL GNGT(RX,BX,IP)
```

3. パラメーターの説明

RX (R) 調べる実数値.
BX (R) きりのよい数の仮数部.
IP (I) きりのよい数の指数部.

4. 備考

(a) きりのよい数のリストは GNSBLK, GNQBLK で設定・参照できる.

10.4.2 GNLE/GNGE

1. 機能

与えた実数値以下/以上のきりのよい数のうちで最大/最小のものを求める. 結果は $\mathtt{BX} \times 10^{\mathtt{IP}}$ の形で返される.

2. 呼び出し方法

```
CALL GNLE(RX,BX,IP)
CALL GNGE(RX,BX,IP)
```

3. パラメーターの説明

RX (R) 調べる実数値.
BX (R) きりのよい数の仮数部.
IP (I) きりのよい数の指数部.

4. 備考

(a) きりのよい数のリストは GNSBLK, GNQBLK で設定・参照できる.

10.4.3 GNSBLK/GNQBLK

1. 機能

きりのよい数のリストを設定/参照する.

2. 呼び出し方法

```
CALL GNSBLK(XB,NB)
CALL GNQBLK(XB,NB)
```

3. パラメーターの説明

XB (R) きりのよい数を昇順に納めた実数型配列. XB(1)=1.0, XB(NB)=10.0 となるように指定する.
NB (I) 配列の寸法.

4. 備考

(a) NB は 20 以下でなければならない.

10.4.4 GNSAVE/GNRSET

1. 機能

   現在設定されているきりのよい数のリストを保存する/保存してあったリストで再設定する.

2. 呼び出し方法

   ```
   CALL GNSAVE
   CALL GNRSET
   ```

3. パラメーターの説明

   なし

4. 備考

   (a) なし

10.5 関数の説明

10.5.1 RGNLT/RGNLE/RGNGT/RGNGE

1. 機能

   きりのよい数を求める.

2. 呼び出し方法

   ```
   RGNLT(RX)
   RGNLE(RX)
   RGNGT(RX)
   RGNGE(RX)
   ```

3. パラメーターの説明

RX	(R)	調べる実数値.
RGNLT	(R)	RX より小さいきりのよい数のうちで最大のものを与える関数値.
RGNLE	(R)	RX 以下のきりのよい数のうちで最大のものを与える関数値.
RGNGT	(R)	RX より大きいきりのよい数のうちで最小のものを与える関数値.
RGNGE	(R)	RX 以上のきりのよい数のうちで最小のものを与える関数値.

4. 備考

   (a) きりのよい数のリストは GNSBLK, GNQBLK で設定・参照できる.

第11章　INTLIB：実数に近い整数

11.1　概要

与えた実数に近い整数を求める関数パッケージ. 大小比較は LRLLIB を使っておこなうので, 誤差を含んだ比較ができる (第 8 章参照).

11.2　関数のリスト

IRLT(RX)　RX より小さい最大の整数を求める.
IRLE(RX)　RX 以下の最大の整数を求める.
IRGT(RX)　RX より大きい最小の整数を求める.
IRGE(RX)　RX 以上の最小の整数を求める.

11.3　関数の説明

11.3.1　IRLT/IRLE/IRGT/IRGE

1. 機能
   与えた実数値に近い整数値をもとめる.
2. 呼び出し方法
   IRLT(RX)
   IRLE(RX)
   IRGT(RX)
   IRGE(RX)
3. パラメーターの説明

RX	(R)	調べる実数値.
IRLT	(I)	RX より小さい最大の整数関数値.
IRLE	(I)	RX 以下の最大の整数関数値.
IRGT	(I)	RX より大きい最小の整数関数値.
IRGE	(I)	RX 以上の最小の整数関数値.

4. 備考
   (a) なし.

第12章　INDXLIB：配列要素の検索

12.1　概要

与えられた配列中に, 指定した値と同じ値をもつ配列要素が存在するかどうかを調べ, その位置や個数を返す関数パッケージ.

ここでいう, 配列における' 位置' とは, それまでに調べた (今調べたものも含む) 配列要素の個数のことである. 指定した値と同じ値をもつ配列要素が存在しないとき, 位置の値は 0 とする. したがって, 調べる配列要素の個数を N とすると, 位置を調べる関数が返す値は, 0 以上 N 以下である.

12.2　関数のリスト

INDXCF(CX,N,JD,CH)	CH (文字長は 1) が最初に現れる位置を求める.
INDXCL(CX,N,JD,CH)	CH (文字長は 1) が最後に現れる位置を求める.
INDXNF(CX,N,JD,CH)	CH (文字長は 1 以上) が最初に現れる位置を求める.
INDXNL(CX,N,JD,CH)	CH (文字長は 1 以上) が最後に現れる位置を求める.
INDXMF(CX,N,JD,CH)	CH (文字長は 1 以上) が最初に現れる位置を求める. 大文字・小文字を区別しない.
INDXML(CX,N,JD,CH)	CH (文字長は 1 以上) が最後に現れる位置を求める. 大文字・小文字を区別しない.
INDXIF(IX,N,JD,II)	II が最初に現れる位置を求める.
INDXIL(IX,N,JD,II)	II が最後に現れる位置を求める.
INDXRF(RX,N,JD,RR)	RR が最初に現れる位置を求める.
INDXRL(RX,N,JD,RR)	RR が最後に現れる位置を求める.
NINDXC(CX,N,JD,CH)	CH (文字長は 1) が何個あるかを求める.
NINDXN(CX,N,JD,CH)	CH (文字長は 1 以上) が何個あるかを求める.
NINDXM(CX,N,JD,CH)	CH (文字長は 1 以上) が何個あるかを求める. 大文字・小文字を区別しない.
NINDXI(IX,N,JD,II)	II が何個あるかを求める.
NINDXR(RX,N,JD,RR)	RR が何個あるかを求める.

12.3　関数の説明

12.3.1　INDXCF/INDXCL

1. 機能

   指定した文字 (文字長は 1) が最初/最後に現れる位置を求める.

2. 呼び出し方法

```
INDXCF(CX,N,JD,CH)
INDXCL(CX,N,JD,CH)
```

3. パラメーターの説明

CX	(C*(*))	調べる文字列.
N	(I)	調べる文字の個数.
JD	(I)	調べる文字の間隔.
CH	(C*1)	指定した文字 (文字長は 1).
INDXCF	(I)	最初に現れる位置を与える関数値.
INDXCL	(I)	最後に現れる位置を与える関数値.

4. 備考

(a) なし.

12.3.2 INDXNF/INDXNL

1. 機能

指定した文字 (文字長は 1 以上) が最初/最後に現れる位置を求める.

2. 呼び出し方法

```
INDXNF(CX,N,JD,CH)
INDXNL(CX,N,JD,CH)
```

3. パラメーターの説明

CX	(C*(*))	調べる文字列.
N	(I)	調べる文字の個数.
JD	(I)	調べる文字の間隔.
CH	(C*(*))	指定した文字 (文字長は 1 以上).
INDXNF	(I)	最初に現れる位置を与える関数値.
INDXNL	(I)	最後に現れる位置を与える関数値.

4. 備考

(a) JD は, 調べる文字の先頭位置の間隔である. したがって, CH の長さが JD より長いとき調べる文字の範囲に重なりができる.

(b) CX の長さは N×JD+(LEN(CH)−JD) 以上でなければならない.

12.3.3 INDXMF/INDXML

1. 機能

指定した文字 (文字長は 1 以上) が最初/最後に現れる位置を求める. 大文字・小文字を区別しないで調べる.

2. 呼び出し方法

```
INDXMF(CX,N,JD,CH)
INDXML(CX,N,JD,CH)
```

3. パラメーターの説明

CX	(C*(*))	調べる文字列.
N	(I)	調べる文字の個数.
JD	(I)	調べる文字の間隔.
CH	(C*1)	指定した文字 (文字長は 1 以上).
INDXMF	(I)	最初に現れる位置を与える関数値.
INDXML	(I)	最後に現れる位置を与える関数値.

4. 備考
   (a) JD は, 調べる文字の先頭位置の間隔である. したがって, CH の長さが JD より長いとき調べる文字の範囲に重なりができる.
   (b) CX の長さは N×JD+(LEN(CH)−JD) 以上でなければならない.
   (c) 文字の比較は LCHREQ (2.5.1 節参照) によっておこなわれる.

12.3.4 INDXIF/INDXIL

1. 機能
   指定した整数値が最初/最後に現れる位置を求める.
2. 呼び出し方法
   ```
   INDXIF(IX,N,JD,II)
   INDXIL(IX,N,JD,II)
   ```
3. パラメーターの説明

IX	(I)	調べる整数型配列.
N	(I)	調べる配列要素の個数.
JD	(I)	調べる配列要素の間隔.
II	(I)	指定した整数値.
INDXIF	(I)	最初に現れる位置を与える関数値.
INDXIL	(I)	最後に現れる位置を与える関数値.

4. 備考
   (a) なし.

12.3.5 INDXRF/INDXRL

1. 機能
   指定した実数値が最初/最後に現れる位置を求める.
2. 呼び出し方法
   ```
   INDXRF(RX,N,JD,RR)
   INDXRL(RX,N,JD,RR)
   ```
3. パラメーターの説明

RX	(R)	調べる実数型配列.
N	(I)	調べる配列要素の個数.
JD	(I)	調べる配列要素の間隔.
RR	(R)	指定した実数値.
INDXRF	(I)	最初に現れる位置を与える関数値.
INDXRL	(I)	最後に現れる位置を与える関数値.

4. 備考

(a) なし.

12.3.6 NINDXC

1. 機能

指定した文字 (文字長は 1) が何個あるかを求める.

2. 呼び出し方法

```
NINDXC(CX,N,JD,CH)
```

3. パラメーターの説明

CX	(C*(*))	調べる文字列.
N	(I)	調べる文字の個数.
JD	(I)	調べる文字の間隔.
CH	(C*1)	指定した文字 (文字長は 1).
NINDXC	(I)	文字の個数を与える関数値.

4. 備考

(a) なし.

12.3.7 NINDXN

1. 機能

指定した文字 (文字長は 1 以上) が何個あるかを求める.

2. 呼び出し方法

```
NINDXN(CX,N,JD,CH)
```

3. パラメーターの説明

CX	(C*(*))	調べる文字列.
N	(I)	調べる文字の個数.
JD	(I)	調べる文字の間隔.
CH	(C*1)	指定した文字 (文字長は 1 以上).
NINDXN	(I)	文字の個数を与える関数値.

4. 備考

(a) JD は, 調べる文字の先頭位置の間隔である. したがって, CH の長さが JD より大きいとき調べる文字の範囲に重なりができる.

(b) CX の長さは N×JD+(LEN(CH)−JD) 以上でなければならない.

12.3.8 NINDXM

1. 機能

   指定した文字 (文字長は 1 以上) が何個あるかを求める. 大文字・小文字を区別しないで調べる.

2. 呼び出し方法

   ```
   NINDXM(CX,N,JD,CH)
   ```

3. パラメーターの説明

CX	(C*(*))	調べる文字列.
N	(I)	調べる文字の個数.
JD	(I)	調べる文字の間隔.
CH	(C*1)	指定した文字 (文字長は 1 以上).
NINDXM	(I)	文字の個数を与える関数値.

4. 備考

   (a) JD は, 調べる文字の先頭位置の間隔である. したがって, CH の長さが JD より大きいとき調べる文字の範囲に重なりができる.

   (b) CX の長さは N×JD+(LEN(CH)−JD) 以上でなければならない.

   (c) 文字の比較は LCHREQ (2.5.1 節参照) によっておこなわれる.

12.3.9 NINDXI/NINDXR

1. 機能

   指定した整数値/実数値が何個あるかを求める.

2. 呼び出し方法

   ```
   NINDXI(IX,N,JD,II)
   NINDXR(RX,N,JD,RR)
   ```

3. パラメーターの説明

IX	(I)	調べる整数型配列.
RX	(R)	調べる実数型配列.
N	(I)	調べる配列要素の個数.
JD	(I)	調べる配列要素の間隔.
II	(I)	指定した整数値.
RR	(R)	指定した実数値.
NINDXI	(I)	整数の個数を与える関数値.
NINDXR	(I)	実数の個数を与える関数値.

4. 備考

   (a) なし.

第13章　IFALIB：整数の欠損値処理付最大最小など

13.1　概要

1 つの整数型配列に対して, 1 つの整数値が決まるような作用をもつ関数パッケージ. 以下では総称名を示す. たとえば IMAX に対しては, ほかに IMAX0, IMAX1 という名前の関数も存在する. IMAX0 は欠損値処理をおこなわない; IMAX1 は欠損値処理をおこなう. IMAX は内部変数'LMISS' が.TRUE. なら欠損値処理をおこなう; .FALSE. なら欠損値処理をおこなわない (初期値は.FALSE.). また欠損値処理をおこなうときの欠損値は, 内部変数'IMISS' で決まる (初期値は 999). (詳しくは 2.4.1 節参照)

13.2　関数のリスト

IMAX(IX,N,JX)	最大値を求める.
IMIN(IX,N,JX)	最小値を求める.
ISUM(IX,N,JX)	総和を求める.

13.3　関数の説明

13.3.1　IMAX/IMIN/ISUM

1. 機能

   最大値/最小値/総和をもとめる.

2. 呼び出し方法

   ```
   IMAX(IX,N,JX)
   IMIN(IX,N,JX)
   ISUM(IX,N,JX)
   ```

3. パラメーターの説明

IX	(I(*))	処理する整数型配列.
N	(I)	処理する配列要素の個数.
JX	(I)	処理する配列要素の間隔.
IMAX	(I)	最大値を与える関数値.
IMIN	(I)	最小値を与える関数値.
ISUM	(I)	総和を与える関数値.

4. 備考

   (a) なし.

第14章　RFALIB：実数の欠損値処理付最大最小など

14.1　概要

1 つの実数型配列に対して, 1 つの実数値が決まるような作用をもつ関数パッケージ. 以下では総称名を示す. たとえば RMAX に対しては, ほかに RMAX0, RMAX1 という名前の関数も存在する. RMAX0 は欠損値処理をおこなわない; RMAX1 は欠損値処理をおこなう. RMAX は内部変数'LMISS' が.TRUE. なら欠損値処理をおこなう; .FALSE. なら欠損値処理をおこなわない (初期値は.FALSE.). また欠損値処理をおこなうときの欠損値は, 内部変数'RMISS' で決まる (初期値は 999.0). (詳しくは 2.4.1 節参照)

14.2　関数のリスト

RMAX(RX,N,JX)	最大値を求める.
RMIN(RX,N,JX)	最小値を求める.
RSUM(RX,N,JX)	総和を求める.
RAVE(RX,N,JX)	平均を求める.
RVAR(RX,N,JX)	分散を求める.
RSTD(RX,N,JX)	標準偏差を求める.
RRMS(RX,N,JX)	root mean square $(((\Sigma RX^2)/N)^{1/2})$ を求める.
RAMP(RX,N,JX)	大きさ $((\Sigma RX^2)^{1/2})$ を求める.
VRMAX(RX,NS,NP,NQ,ND)	多次元配列の (一部分の) 最大値を求める.
VRMIN(RX,NS,NP,NQ,ND)	多次元配列の (一部分の) 最小値を求める.

14.3　関数の説明

14.3.1　RMAX/RMIN/RSUM/RAVE/RVAR/RSTD/RRMS/RAMP

1. 機能

   最大値 / 最小値 / 総和 / 平均 / 分散 / 標準偏差 / root mean square $(((\Sigma RX^2)/N)^{1/2})$ / 大きさ $((\Sigma RX^2)^{1/2})$ をもとめる.

2. 呼び出し方法

   ```
   RMAX(RX,N,JX)
   RMIN(RX,N,JX)
   RSUM(RX,N,JX)
   RAVE(RX,N,JX)
   ```

```
RVAR(RX,N,JX)
RSTD(RX,N,JX)
RRMS(RX,N,JX)
RAMP(RX,N,JX)
```

3. パラメーターの説明

RX	(R(*))	処理する実数型配列.
N	(I)	処理する配列要素の個数.
JX	(I)	処理する配列要素の間隔.
RMAX	(R)	最大値を与える関数値.
RMIN	(R)	最小値を与える関数値.
RSUM	(R)	総和を与える関数値.
RAVE	(R)	平均を与える関数値.
RVAR	(R)	分散を与える関数値.
RSTD	(R)	標準偏差を与える関数値.
RRMS	(R)	root mean square $(((\Sigma RX^2)/N)^{1/2})$ を与える関数値.
RAMP	(R)	大きさ $((\Sigma RX^2)^{1/2})$ を与える関数値.

4. 備考

(a) なし.

14.3.2 RVMAX/RVMIN

1. 機能

多次元配列の (一部分の) 最大値 / 最小値をもとめる.

2. 呼び出し方法

```
VRMAX(RX,NS,NP,NQ,ND)
VRMIN(RX,NS,NP,NQ,ND)
```

3. パラメーターの説明

RX	(R(*))	処理する実数型配列.
NS	(I(*))	配列の各次元の長さを与える整数型配列.
NP	(I(*))	各次元の処理をおこなう最初の位置を与える整数型配列.
NQ	(I(*))	各次元の処理をおこなう最後の位置を与える整数型配列.
ND	(I)	配列の次元数.
RVMAX	(R)	最大値を与える関数値.
RVMIN	(R)	最小値を与える関数値.

4. 備考

(a) たとえば 3 次元配列 X(10,20,30) が宣言してあったとき, 1 次元目について [2, 9], 2 次元目について [3, 12], 3 次元目について [5, 20] の添字が動く範囲の最大値を求めるためには, NS(1)=10, NS(2)=20, NS(3)=30, NP(1)=2, NP(2)=3, NP(3)=5, NQ(1)=9, NQ(2)=12, NQ(3)=20, ND=3 として RVMAX を呼んでやればよい.

第15章　RFBLIB：実数列の内積, 共分散, 相関係数

15.1　概要

2 つの実数型配列に対して, 1 個の実数値が決まるような作用をもつ関数パッケージ. このパッケージの関数は, RFALIB (第 14 章) などと異なり欠損値処理はおこなわない. 欠損値処理をおこないたいときは, MISC1 の中の VSTLIB を用いるとよい.

15.2　関数のリスト

RPRD(RX,RY,N,JX,JY)	内積を求める.
RCOV(RX,RY,N,JX,JY)	共分散を求める.
RCOR(RX,RY,N,JX,JY)	相関係数を求める.

15.3　関数の説明

15.3.1　RPRD/RCOV/RCOR

1. 機能

   内積/共分散/相関係数をもとめる.

2. 呼び出し方法

   RPRD(RX,RY,N,JX,JY)

   RCOV(RX,RY,N,JX,JY)

   RCOR(RX,RY,N,JX,JY)

3. パラメーターの説明

RX, RY	(R(*))	処理する実数型配列.
N	(I)	処理する配列要素の個数.
JX, JY	(I)	配列 RX, RY において, 処理する配列要素の間隔.
RPRD	(R)	内積を与える関数値.
RCOV	(R)	共分散を与える関数値.
RCOR	(R)	相関係数を与える関数値.

4. 備考

   (a) 相関係数 RCOR については, RX または RY の分散が 0 のときは, 内部変数'RMISS' で決まる欠損値 (初期値は 999.0) を返す.

第16章　VIALIB：1つの整数型配列への作用素

16.1　概要

1つの整数型変数に対して1つの整数値がきまるような関数を, 1つの整数型配列の各要素に対して作用させ, 結果を整数型配列として得るようなサブルーチンパッケージ. 以下では総称名を示す. たとえばVIFNAに対しては, ほかにVIFNA0, VIFNA1という名前のサブルーチンも存在する. VIFNA0は欠損値処理をおこなわない; VIFNA1は欠損値処理をおこなう. VIFNAは内部変数'LMISS'が.TRUE.なら欠損値処理をおこなう; .FALSE.なら欠損値処理をおこなわない (初期値は.FALSE.). また欠損値処理をおこなうときの欠損値は, 内部変数'IMISS'が決める (初期値は999). (詳しくは 2.4.1 節参照)

16.2　サブルーチンのリスト

VIFNA(IX,IY,N,JX,JY,IFNA)	IXにIFNAを作用させてIYに代入する.
VIINC(IX,IY,N,JX,JY,II)	IXにIIを加えてIYに代入する.
VIFCT(IX,IY,N,JX,JY,II)	IXにIIを掛けてIYに代入する.
VICON(IX,IY,N,JX,JY,II)	IIをIYに代入する.
VISET(IX,IY,N,JX,JY)	IXをIYに代入する.

これらの簡易版として次のようなサブルーチンがある.

IADD(IX,N,JX,II)	IXにIIを加える.
IMLT(IX,N,JX,II)	IXにIIを掛ける.
ISET(IX,N,JX,II)	IXにIIを代入する.

16.3　サブルーチンの説明

16.3.1　VIFNA

1. 機能

   整数型配列IXに整数型関数IFNAを作用させて整数型配列IYに代入する.

2. 呼び出し方法

   CALL VIFNA(IX,IY,N,JX,JY,IFNA)

3. パラメーターの説明

IX, IY	(I(*))	処理する整数型配列.
N	(I)	処理する配列要素の個数.
JX, JY	(I)	配列 IX, IY において, 処理する配列要素の間隔.
IFNA	関数名	引数が 1 個である整数型関数名.

4. 備考

(a) 関数名は, それが外部関数なら EXTERNAL 文で, 組込み関数なら INTRINSIC 文で宣言しておかなければならない.

16.3.2 VIINC/VIFCT

1. 機能

整数型配列 IX に整数値 II を加えて/掛けて整数型配列 IY に代入する.

2. 呼び出し方法

```
CALL VIINC(IX,IY,N,JX,JY,II)
CALL VIFCT(IX,IY,N,JX,JY,II)
```

3. パラメーターの説明

IX, IY	(I(*))	処理する整数型配列.
N	(I)	処理する配列要素の個数.
JX, JY	(I)	配列 IX, IY において, 処理する配列要素の間隔.
II	(I)	加える/掛ける整数値.

4. 備考

(a) なし.

16.3.3 VICON

1. 機能

整数値 II を整数型配列 IY に代入する.

2. 呼び出し方法

```
CALL VICON(IX,IY,N,JX,JY,II)
```

3. パラメーターの説明

IX, IY	(I(*))	処理する整数型配列.
N	(I)	処理する配列要素の個数.
JX, JY	(I)	配列 IX, IY において, 処理する配列要素の間隔.
II	(I)	代入する整数値.

4. 備考

(a) IX は欠損値処理をおこなうかどうかの判断に用いる. つまり欠損値処理をおこなうような設定のとき, IX の配列要素が欠損値なら, 代入する整数値は欠損値となる.

16.3.4 VISET

1. 機能

整数型配列 IX を整数型配列 IY に代入する.

2. 呼び出し方法

```
CALL VISET(IX,IY,N,JX,JY)
```

3. パラメーターの説明

IX, IY	(I(*))	処理する整数型配列.
N	(I)	処理する配列要素の個数.
JX, JY	(I)	配列 IX, IY において, 処理する配列要素の間隔.

4. 備考

(a) なし.

16.3.5 IADD/IMLT

1. 機能

VIINC/VIFCT の簡易版サブルーチン.

2. 呼び出し方法

```
CALL IADD(IX,N,JX,II)
CALL IMLT(IX,N,JX,II)
```

3. パラメーターの説明

IX	(I(*))	処理する整数型配列.
N	(I)	処理する配列要素の個数.
JX	(I)	配列 IX において, 処理する配列要素の間隔.
II	(I)	加える/掛ける整数値.

4. 備考

(a) なし.

16.3.6 ISET

1. 機能

VICON の簡易版サブルーチン.

2. 呼び出し方法

```
CALL ISET(IX,N,JX,II)
```

3. パラメーターの説明

IX	(I(*))	処理する整数型配列.
N	(I)	処理する配列要素の個数.
JX	(I)	配列 IX において, 処理する配列要素の間隔.
II	(I)	代入する整数値.

4. 備考

(a) なし.

第17章　VIBLIB：2つの整数型配列への作用素

17.1　概要

2つの整数型変数に対して1つの整数値がきまるような関数を, 2つの整数型配列の各要素に対して作用させ, 結果を整数型配列として得るようなサブルーチンパッケージ. 以下では総称名を示す. たとえばVIFNBに対しては, ほかにVIFNB0, VIFNB1という名前のサブルーチンも存在する. VIFNB0は欠損値処理をおこなわない; VIFNB1は欠損値処理をおこなう. VIFNBは内部変数'LMISS'が.TRUE.なら欠損値処理をおこなう; .FALSE.なら欠損値処理をおこなわない (初期値は.FALSE.). また欠損値処理をおこなうときの欠損値は, 内部変数'IMISS'が決める (初期値は999). (詳しくは 2.4.1 節参照)

17.2　サブルーチンのリスト

VIFNB(IX,IY,IZ,N,JX,JY,JZ,IFNB)	IXとIYにIFNBを作用させてIZに代入する.
VIADD(IX,IY,IZ,N,JX,JY,JZ)	IXとIYを加えてIZに代入する.
VISUB(IX,IY,IZ,N,JX,JY,JZ)	IXからIYを引いてIZに代入する.
VIMLT(IX,IY,IZ,N,JX,JY,JZ)	IXとIYを掛けてIZに代入する.
VIDIV(IX,IY,IZ,N,JX,JY,JZ)	IXをIYで割ってIZに代入する.

17.3　サブルーチンの説明

17.3.1　VIFNB

1. 機能

   整数型配列IXとIYに整数型関数IFNBを作用させて整数型配列IZに代入する.

2. 呼び出し方法

   ```
   CALL VIFNB(IX,IY,IZ,N,JX,JY,JZ,IFNB)
   ```

3. パラメーターの説明

IX, IY, IZ	(I(*))	処理する整数型配列.
N	(I)	処理する配列要素の個数.
JX, JY, JZ	(I)	配列IX, IY, IZにおいて, 処理する配列要素の間隔.
IFNB	関数名	引数が2個である整数型関数名.

4. 備考

   (a) 関数名は, それが外部関数ならEXTERNAL文で, 組込み関数ならINTRINSIC文で宣言しておかなければならない.

17.3.2 VIADD/VISUB

1. 機能

   整数型配列 IX と/から整数型配列 IY を加えて/を引いて整数型配列 IZ に代入する.

2. 呼び出し方法

```
CALL VIADD(IX,IY,IZ,N,JX,JY,JZ)
CALL VISUB(IX,IY,IZ,N,JX,JY,JZ)
```

3. パラメーターの説明

IX, IY, IZ	(I(*))	処理する整数型配列.
N	(I)	処理する配列要素の個数.
JX, JY, JZ	(I)	配列 IX, IY, IZ において, 処理する配列要素の間隔.

4. 備考

   (a) なし.

17.3.3 VIMLT/VIDIV

1. 機能

   整数型配列 IX と/を整数型配列 IY を掛けて/で割って整数型配列 IZ に代入する.

2. 呼び出し方法

```
CALL VIMLT(IX,IY,IZ,N,JX,JY,JZ)
CALL VIDIV(IX,IY,IZ,N,JX,JY,JZ)
```

3. パラメーターの説明

IX, IY, IZ	(I(*))	処理する整数型配列.
N	(I)	処理する配列要素の個数.
JX, JY, JZ	(I)	配列 IX, IY, IZ において, 処理する配列要素の間隔.

4. 備考

   (a) なし.

第18章　VRALIB：1つの実数型配列への作用素

18.1　概要

1つの実数型変数に対して1つの実数値がきまるような関数を, 1つの実数型配列の各要素に対して作用させ, 結果を実数型配列として得るようなサブルーチンパッケージ. 以下では総称名を示す. たとえばVRFNAに対しては, ほかにVRFNA0, VRFNA1という名前のサブルーチンも存在する. VRFNA0は欠損値処理をおこなわない; VRFNA1は欠損値処理をおこなう. VRFNAは内部変数'LMISS'が.TRUE.なら欠損値処理をおこなう; .FALSE.なら欠損値処理をおこなわない (初期値は.FALSE.. また欠損値処理をおこなうときの欠損値は, 内部変数'RMISS'が決める (初期値は999.0). (詳しくは2.4.1節参照)

18.2　サブルーチンのリスト

VRFNA(RX,RY,N,JX,JY,RFNA)	RXにRFNAを作用させてRYに代入する.
VRINC(RX,RY,N,JX,JY,RR)	RXにRRを加えてRYに代入する.
VRFCT(RX,RY,N,JX,JY,RR)	RXにRRを掛けてRYに代入する.
VRCON(RX,RY,N,JX,JY,RR)	RRをRYに代入する.
VRSET(RX,RY,N,JX,JY)	RXをRYに代入する.

これらの簡易版として次のようなサブルーチンがある.

RADD(RX,N,JX,RR)	RXにRRを加える.
RMLT(RX,N,JX,RR)	RXにRRを掛ける.
RSET(RX,N,JX,RR)	RXにRRを代入する.

18.3　サブルーチンの説明

18.3.1　VRFNA

1. 機能

   実数型配列RXに実数型関数RFNAを作用させて実数型配列RYに代入する.

2. 呼び出し方法

   CALL VRFNA(RX,RY,N,JX,JY,RFNA)

3. パラメーターの説明

RX, RY	(R(*))	処理する実数型配列.
N	(I)	処理する配列要素の個数.
JX, JY	(I)	配列 RX, RY において, 処理する配列要素の間隔.
RFNA	関数名	引数が 1 個である実数型関数名.

4. 備考

(a) 関数名は, それが外部関数なら EXTERNAL 文で, 組込み関数なら INTRINSIC 文で宣言しておかなければならない.

18.3.2 VRINC/VRFCT

1. 機能

実数型配列 RX に実数値 RR を加えて/掛けて実数型配列 RY に代入する.

2. 呼び出し方法

```
CALL VRINC(RX,RY,N,JX,JY,RR)
CALL VRFCT(RX,RY,N,JX,JY,RR)
```

3. パラメーターの説明

RX, RY	(R(*))	処理する実数型配列.
N	(I)	処理する配列要素の個数.
JX, JY	(I)	配列 RX, RY において, 処理する配列要素の間隔.
RR	(R)	加える/掛ける実数値.

4. 備考

(a) なし.

18.3.3 VRCON

1. 機能

実数値 RR を実数型配列 RY に代入する.

2. 呼び出し方法

```
CALL VRCON(RX,RY,N,JX,JY,RR)
```

3. パラメーターの説明

RX, RY	(R(*))	処理する実数型配列.
N	(I)	処理する配列要素の個数.
JX, JY	(I)	配列 RX, RY において, 処理する配列要素の間隔.
RR	(R)	代入する実数値.

4. 備考

(a) RX は欠損値処理をおこなうかどうかの判断に用いる. つまり欠損値処理をおこなうような設定のとき, RX の配列要素が欠損値なら, 代入する実数値は欠損値となる.

18.3.4 VRSET

1. 機能

実数型配列 RX を実数型配列 RY に代入する.

2. 呼び出し方法

```
CALL VRSET(RX,RY,N,JX,JY)
```

3. パラメーターの説明

RX, RY	(R(*))	処理する実数型配列.
N	(I)	処理する配列要素の個数.
JX, JY	(I)	配列 RX, RY において, 処理する配列要素の間隔.

4. 備考

(a) なし.

18.3.5 RADD/RMLT

1. 機能

VRINC/VRFCT の簡易版サブルーチン.

2. 呼び出し方法

```
CALL RADD(RX,N,JX,RR)
CALL RMLT(RX,N,JX,RR)
```

3. パラメーターの説明

RX	(R(*))	処理する実数型配列.
N	(I)	処理する配列要素の個数.
JX	(I)	配列 RX において, 処理する配列要素の間隔.
RR	(R)	加える/掛ける実数値.

4. 備考

(a) なし.

18.3.6 RSET

1. 機能

VRCON の簡易版サブルーチン.

2. 呼び出し方法

```
CALL RSET(RX,N,JX,RR)
```

3. パラメーターの説明

RX	(R(*))	処理する実数型配列.
N	(I)	処理する配列要素の個数.
JX	(I)	配列 RX において, 処理する配列要素の間隔.
RR	(R)	代入する実数値.

4. 備考

(a) なし.

第19章　VRBLIB：2つの実数型配列への作用素

19.1　概要

2つの実数型変数に対して1つの実数値がきまるような関数を, 2つの実数型配列の各要素に対して作用させ, 結果を実数型配列として得るようなサブルーチンパッケージ. 以下では総称名を示す. たとえばVRFNBに対しては, ほかにVRFNB0, VRFNB1という名前のサブルーチンも存在する. VRFNB0は欠損値処理をおこなわない; VRFNB1は欠損値処理をおこなう. VRFNBは内部変数'LMISS'が.TRUE.なら欠損値処理をおこなう; .FALSE.なら欠損値処理をおこなわない (初期値は.FALSE.). また欠損値処理をおこなうときの欠損値は, 内部変数'RMISS'が決める (初期値は999.0). (詳しくは2.4.1節参照)

19.2　サブルーチンのリスト

VRFNB(RX,RY,RZ,N,JX,JY,JZ,RFNB)	RXとRYにRFNBを作用させてRZに代入する.
VRADD(RX,RY,RZ,N,JX,JY,JZ)	RXとRYを加えてRZに代入する.
VRSUB(RX,RY,RZ,N,JX,JY,JZ)	RXからRYを引いてRZに代入する.
VRMLT(RX,RY,RZ,N,JX,JY,JZ)	RXとRYを掛けてRZに代入する.
VRDIV(RX,RY,RZ,N,JX,JY,JZ)	RXをRYで割ってRZに代入する.

19.3　サブルーチンの説明

19.3.1　VRFNB

1. 機能

   実数型配列RXとRYに実数型関数RFNBを作用させて実数型配列RZに代入する.

2. 呼び出し方法

   ```
   CALL VRFNB(RX,RY,RZ,N,JX,JY,JZ,RFNB)
   ```

3. パラメーターの説明

RX, RY, RZ	(R(*))	処理する実数型配列.
N	(I)	処理する配列要素の個数.
JX, JY, JZ	(I)	配列RX, RY, RZにおいて, 処理する配列要素の間隔.
RFNB	関数名	引数が2個である実数型関数名.

4. 備考

   (a) 関数名は, それが外部関数ならEXTERNAL文で, 組込み関数ならINTRINSIC文で宣言しておかなければならない.

19.3.2 VRADD/VRSUB

1. 機能

   実数型配列 RX と/から実数型配列 RY を加えて/を引いて実数型配列 RZ に代入する.

2. 呼び出し方法

```
CALL VRADD(RX,RY,RZ,N,JX,JY,JZ)
CALL VRSUB(RX,RY,RZ,N,JX,JY,JZ)
```

3. パラメーターの説明

RX, RY, RZ	(R(*))	処理する実数型配列.
N	(I)	処理する配列要素の個数.
JX, JY, JZ	(I)	配列 RX, RY, RZ において, 処理する配列要素の間隔.

4. 備考

   (a) なし.

19.3.3 VRMLT/VRDIV

1. 機能

   実数型配列 RX と/を実数型配列 RY を掛けて/で割って実数型配列 RZ に代入する.

2. 呼び出し方法

```
CALL VRMLT(RX,RY,RZ,N,JX,JY,JZ)
CALL VRDIV(RX,RY,RZ,N,JX,JY,JZ)
```

3. パラメーターの説明

RX, RY, RZ	(R(*))	処理する実数型配列.
N	(I)	処理する配列要素の個数.
JX, JY, JZ	(I)	配列 RX, RY, RZ において, 処理する配列要素の間隔.

4. 備考

   (a) なし.

第20章　CTRLIB：座標変換/回転

20.1　概要

座標変換に関するサブルーチンパッケージ．変換する座標系はルーチン名の末尾 2 文字に略号であらわされる．ここで扱う座標系の略号は以下の通り．

	次元	英語名	略号
直角座標	2	cartesian coordinates	C
極座標	2	polar coordinates	P
楕円座標	2	elliptic coordinates	E
直角双曲線座標	2	rectangular hyperbolic	H
双極座標	2	bipolar coordinates	B
直角座標	3	cartesian coordinates	C
球座標	3	spherical coordinates	S
楕円面座標	3	ellipsoidal coordinates	E

放物線座標との変換は直角双曲線座標の逆変換である．

角度に関する引数の単位はすべてラジアンである．

20.2　サブルーチンのリスト

`CT2PC(R,THETA,X,Y)`	2 次元極座標を直角座標に変換する．
`CT2CP(X,Y,R,THETA)`	直角座標を 2 次元極座標に変換する．
`CT2EC(U,V,X,Y)`	楕円座標を直角座標に変換する．
`CT2BC(U,V,X,Y)`	双極座標を直角座標に変換する．
`CT2HC(U,V,X,Y)`	直角双曲線座標を直角座標に変換する．
`CT2CH(X,Y,U,V)`	直角座標を直角双曲線座標に変換する．
`CT3SC(R,THETA,PHI,X,Y,Z)`	3 次元球面座標を直角座標に変換する．
`CT3CS(X,Y,Z,R,THETA,PHI)`	直角座標を 3 次元球面座標に変換する．
`CR2C(THETA,X0,Y0,X1,Y1)`	2 次元直角座標を回転する．
`CR3C(THETA,PHI,PSI,X0,Y0,Z0,X1,Y1,Z1)`	3 次元直角座標を回転する．
`CR3S(THETA,PHI,PSI,THETA0,PHI0,THETA1,PHI1)`	球面座標を回転する．

20.3 サブルーチンの説明

20.3.1 CT2PC/CT2CP

1. 機能

   2 次元極座標と直角座標の変換をする.

2. 呼び出し方法

   ```
   CALL CT2PC(R,THETA,X,Y)
   CALL CT2CP(X,Y,R,THETA)
   ```

3. パラメーターの説明

X,Y	(R)	直角座標.
R,THETA	(R)	極座標.

4. 備考

   (a) 定義式

$$x = r\cos(\theta)$$
$$y = r\sin(\theta)$$

20.3.2 CT2EC

1. 機能

   楕円座標と直角座標の変換をする.

2. 呼び出し方法

   ```
   CALL CT2EC(U,V,X,Y)
   ```

3. パラメーターの説明

X,Y	(R)	直角座標.
U,V	(R)	楕円座標.

4. 備考

   (a) 定義式

$$x = \cosh(u)\cos(v)$$
$$y = \sinh(u)\sin(v)$$

20.3.3 CT2BC

1. 機能

   双極座標と直角座標の変換をする.

2. 呼び出し方法

   ```
   CALL CT2BC(U,V,X,Y)
   ```

3. パラメーターの説明

X,Y (R) 直角座標.

U,V (R) 双極座標.

4. 備考

(a) 定義式

$$x = \frac{\sinh(v)}{\cosh(v) + \cos(u)}$$
$$y = \frac{\sin(u)}{\cosh(v) + \cos(u)}$$

(b) $\cosh(v) + \cos(u) = 0$ のとき, 内部変数'RUNDEF' で決まる不定を示す値 (初期値は-999) を返す.

20.3.4 CT2HC/CT2CH

1. 機能

直角双曲線座標と直角座標の変換をする.

2. 呼び出し方法

```
CALL CT2HC(U,V,X,Y)
CALL CT2CH(X,Y,U,V)
```

3. パラメーターの説明

X,Y (R) 直角座標.

U,V (R) 直角双曲線座標.

4. 備考

(a) 定義式

$$u = x^2 - y^2$$
$$v = 2xy$$

(b) X,Y と U,V を入れ換えると, 放物線座標と直角座標の変換となる.

20.3.5 CT3CS/CT3SC

1. 機能

3 次元球面座標と直角座標の変換をする.

2. 呼び出し方法

```
CALL CT3SC(R,THETA,PHI,X,Y,Z)
CALL CT3CS(X,Y,Z,R,THETA,PHI)
```

3. パラメーターの説明

X,Y,Z (R) 直角座標.

R,THETA,PHI (R) 極座標.

4. 備考

(a) 定義式

$$x = r\sin(\theta)\cos(\varphi)$$
$$y = r\sin(\theta)\sin(\varphi)$$
$$z = r\cos(\theta)$$

20.3.6 CR2C

1. 機能

   2 次元直角座標を回転する.

2. 呼び出し方法

   `CALL CR2C(THETA,X0,Y0,X1,Y1)`

3. パラメーターの説明

`THETA`	`(R)`	回転角.
`X0,Y0`	`(R)`	回転前の座標値.
`X1,Y1`	`(R)`	回転後の座標値.

4. 備考

   (a) 定義式

$$x_1 = \cos(\theta)x_0 + \sin(\theta)y_0$$
$$y_1 = -\sin(\theta)x_0 + \cos(\theta)y_0$$

20.3.7 CR3C

1. 機能

   3 次元直角座標を回転する.

2. 呼び出し方法

   `CALL CR3C(THETA,PHI,PSI,X0,Y0,Z0,X1,Y1,Z1)`

3. パラメーターの説明

`THETA,PHI,PSI`	`(R)`	Euler の回転角 (θ, φ, ψ).
`X0,Y0,Z0`	`(R)`	回転前の座標値.
`X1,Y1,Z1`	`(R)`	回転後の座標値.

4. 備考

   (a) 定義式

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} = \begin{pmatrix} \cos\varphi\cos\theta\cos\psi - \sin\varphi\sin\psi, & \sin\varphi\cos\theta\cos\psi + \cos\varphi\sin\psi, & -\sin\theta\cos\psi \\ -\cos\varphi\cos\theta\sin\psi - \sin\varphi\cos\psi, & -\sin\varphi\cos\theta\sin\psi + \cos\varphi\cos\psi, & \sin\theta\sin\psi \\ \cos\varphi\sin\theta, & \sin\varphi\sin\theta, & \cos\theta \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}$$

   (b) Euler の角 (θ, φ, ψ) で回転させるとは, z 軸の回りに φ 回転させ, y 軸の回りに θ 回転させた後に, z 軸の回りに ψ 回転させるのに等しい.

   (c) Euler の角については, 数学辞典などを参照のこと.

   (d) 座標軸ではなく, 座標値を (θ, φ, ψ) だけ回転させるには
   `CALL CR3C(-THETA,-PSI,-PHI,X0,Y0,Z0,X1,Y1,Z1)` とする.

20.3.8 CR3S

1. 機能

   球面座標を回転する.

2. 呼び出し方法

```
CALL CR3S(THETA,PHI,PSI,THETA0,PHI0,THETA1,PHI1)
```

3. パラメーターの説明

THETA,PHI,PSI	(R)	Euler の回転角 (θ, φ, ψ).
THETA0,PHI0	(R)	回転前の座標値.
THETA1,PHI1	(R)	回転後の座標値.

4. 備考

   (a) なし.

第21章　MAPLIB：地図投影変換

21.1　概要

地図投影変換に関する関数パッケージ. このパッケージでは地球の半径を 1 とし, 投影点の緯度を (0,0) または 北極 とした時の地図投影関数を定義する. 投影点での面積が変わらないように (?) 定義されている. また, 全ての関数に逆関数が定義されている.

角度に関する引数の単位は全てラジアンである.

多くの逆投影関数に関して, 与えた値が定義域外のとき, 内部変数'RUNDEF' で決まる不定を示す値 (初期値は-999) を返す.

21.2　サブルーチンのリスト

MPFCYL(XLON,YLAT,X,Y)	正距円筒図法
MPFMER(XLON,YLAT,X,Y)	メルカトル図法.
MPFMWD(XLON,YLAT,X,Y)	モルワイデ図法.
MPFMWL(XLON,YLAT,X,Y)	モルワイデ図法もどき.
MPFHMR(XLON,YLAT,X,Y)	ハンメル図法
MPFEK6(XLON,YLAT,X,Y)	エッケルト第 6 図法
MPFKTD(XLON,YLAT,X,Y)	北田楕円図法
MPFCON(XLON,YLAT,X,Y)	円錐図法
MPSCON(YLAT1)	標準緯線の指定
MPFCOA(XLON,YLAT,X,Y)	ランベルト正積円錐図法
MPSCOA(YLAT1)	標準緯線の指定
MPFCOC(XLON,YLAT,X,Y)	ランベルト正角円錐図法
MPSCOC(YLAT1,YLAT2)	標準緯線の指定
MPFBON(XLON,YLAT,X,Y)	ボンヌ図法
MPSBON(YLAT1)	標準緯線の指定
MPFOTG(XLON,YLAT,X,Y)	正射図法
MPSOTG(RSAT)	軌道半径の設定
MPFPST(XLON,YLAT,X,Y)	平射図法
MPFAZM(XLON,YLAT,X,Y)	正距方位図法
MPFAZA(XLON,YLAT,X,Y)	ランベルト正積方位図法

これらのルーチン以外に, すべての変換関数に対して逆関数が定義されている. それらのルーチン名は, 正変換関数の名前の 3 文字めを I にしたものである.

21.3 サブルーチンの説明

21.3.1 MPFCYL/MPICYL

1. 機能

   正距円筒図法の投影, 逆投影を行なう.

2. 呼び出し方法

```
CALL MPFCYL(XLON,YLAT,X,Y)
CALL MPICYL(X,Y,XLON,YLAT)
```

3. パラメーターの説明

XLON,YLAT	(R)	経度, 緯度.
X,Y	(R)	x, y 座標.

4. 備考

   (a) これは正積でも正角でもない投影法である.

   (b) 経線に沿った長さが正しく投影される.

   (c) 定義式

$$x = \lambda$$
$$y = \varphi$$

21.3.2 MPFMER/MPIMER

1. 機能

   メルカトール図法の投影, 逆投影を行なう.

2. 呼び出し方法

```
CALL MPFMER(XLON,YLAT,X,Y)
CALL MPIMER(X,Y,XLON,YLAT)
```

3. パラメーターの説明

XLON,YLAT	(R)	経度, 緯度.
X,Y	(R)	x, y 座標.

4. 備考

   (a) これは正角図法である.

   (b) 定義式

$$x = \lambda$$
$$y = \log\tan(\pi/4 + \varphi/2)$$

   (c) $-\pi \leq x < \pi$ の範囲に投影される.

21.3.3 MPFMWD/MPIMWD

1. 機能

   モルワイデ図法の投影, 逆投影を行なう.

2. 呼び出し方法

```
CALL MPFMWD(XLON,YLAT,X,Y)
CALL MPIMWD(X,Y,XLON,YLAT)
```

3. パラメーターの説明

XLON,YLAT	(R)	経度, 緯度.
X,Y	(R)	x, y 座標.

4. 備考

   (a) これは正積図法である.

   (b) 定義式

$$x = \frac{2\sqrt{2}\cos(\alpha)}{\pi}\lambda$$
$$y = \sqrt{2}\sin(\alpha)$$

   ここで, α は

$$2\alpha + \sin(2\alpha) = \pi\sin\varphi$$

   の解である.

   (c) 長径 (x 軸方向) $2\sqrt{2}$, 短径 $\sqrt{2}$ の楕円内に投影される.

21.3.4 MPFMWL/MPIMWL

1. 機能

   モルワイデ図法 (もどき) の投影, 逆投影を行なう.

2. 呼び出し方法

```
CALL MPFMWL(XLON,YLAT,X,Y)
CALL MPIMWL(X,Y,XLON,YLAT)
```

3. パラメーターの説明

XLON,YLAT	(R)	経度, 緯度.
X,Y	(R)	x, y 座標.

4. 備考

   (a) この投影法は正確なモルワイデ図法ではない.

   (b) 定義式

$$x = \frac{2\sqrt{2}\cos(\varphi)}{\pi}\lambda$$
$$y = \sqrt{2}\sin(\varphi)$$

   (c) 長径 (x 軸方向) $2\sqrt{2}$, 短径 $\sqrt{2}$ の楕円内に投影される.

21.3.5 MPFHMR/MPIHMR

1. 機能

   ハンメル図法の投影, 逆投影を行なう.

2. 呼び出し方法

```
CALL MPFHMR(XLON,YLAT,X,Y)
CALL MPIHMR(X,Y,XLON,YLAT)
```

3. パラメーターの説明

XLON,YLAT	(R)	経度, 緯度.
X,Y	(R)	x, y 座標.

4. 備考

   (a) これは正積図法である.

   (b) 定義式

$$x = \frac{2\sqrt{2}\cos(\varphi)\sin(\lambda/2)}{\sqrt{1+\cos(\varphi)\cos(\lambda/2)}}$$
$$y = \frac{\sqrt{2}\sin(\varphi)}{\sqrt{1+\cos(\varphi)\cos(\lambda/2)}}$$

   (c) 長径 (x 軸方向) $2\sqrt{2}$, 短径 $\sqrt{2}$ の楕円内に投影される.

21.3.6 MPFEK6/MPIEK6

1. 機能

   エッケルト第 6 図法の投影, 逆投影を行なう.

2. 呼び出し方法

```
CALL MPFEK6(XLON,YLAT,X,Y)
CALL MPIEK6(X,Y,XLON,YLAT)
```

3. パラメーターの説明

XLON,YLAT	(R)	経度, 緯度.
X,Y	(R)	x, y 座標.

4. 備考

   (a) これは正積図法である.

   (b) 定義式

$$x = \frac{a(1+\cos(\alpha))}{2}\lambda$$
$$y = a\alpha$$

   ここで, $a = 2/\sqrt{\pi+2}$, α は

$$\alpha + \sin(\alpha) = \frac{\pi+2}{2}\sin\varphi$$

   の解である.

   (c) 極は $y = \pm\pi a/2$, 長さ πa の線分に投影される.

21.3.7 MPFKTD/MPIKTD

1. 機能

   北田楕円図法の投影, 逆投影を行なう.

2. 呼び出し方法

```
CALL MPFKTD(XLON,YLAT,X,Y)
CALL MPIKTD(X,Y,XLON,YLAT)
```

3. パラメーターの説明

XLON,YLAT	(R)	経度, 緯度.
X,Y	(R)	x, y 座標.

4. 備考

   (a) これは正積図法である.

   (b) 定義式

$$x = \frac{9a\cos(\alpha)}{5\pi}\lambda$$
$$y = a\sin(\alpha)$$

   ここで, $a = 2\sqrt{10\pi/(12\pi + 9\sqrt{3})}$, α は

$$2\alpha + \sin(2\alpha) = \left(\frac{2\pi}{3} + \frac{\sqrt{3}}{2}\right)\sin\varphi$$

   の解である.

   (c) 極は $y = \pm\sqrt{3}a/2$ の長さ a の線分に投影される.

   (d) この図法は教育用地図帳に多く使用されている.

21.3.8 MPFCON/MPICON/MPSCON

1. 機能

   正距円錐図法の投影, 逆投影を行なう.

2. 呼び出し方法

```
CALL MPFCON(XLON,YLAT,X,Y)
CALL MPICON(X,Y,XLON,YLAT)
CALL MPSCON(YLAT1)
```

3. パラメーターの説明

XLON,YLAT	(R)	経度, 緯度.
X,Y	(R)	x, y 座標.
YLAT1	(R)	標準緯線の緯度

4. 備考

   (a) これは正積でも正角でもない図法である.

   (b) 経線に沿った長さが正しく投影される.

   (c) 定義式

$$x = r\sin(k\lambda)$$
$$y = -r\cos(k\lambda)$$

ここで, r, k は

$$k = \cos(\theta_1)$$
$$r = \theta - \theta_1 + \tan(\theta_1)$$

である (θ は余緯度, θ_1 は標準緯線の余緯度).

(d) MPFCON, MPICON を呼ぶ前に, MPSCON で標準緯線の緯度を設定しておかなければならない.

21.3.9 MPFCOA/MPICOA/MPSCOA

1. 機能

   ランベルト正積円錐図法の投影, 逆投影を行なう.

2. 呼び出し方法

```
CALL MPFCOA(XLON,YLAT,X,Y)
CALL MPICOA(X,Y,XLON,YLAT)
CALL MPSCOA(YLAT1)
```

3. パラメーターの説明

XLON,YLAT	(R)	経度, 緯度.
X,Y	(R)	x, y 座標.
YLAT1	(R)	標準緯線の緯度

4. 備考

   (a) これは正積図法である.

   (b) 定義式

$$x = r\sin(k\lambda)$$
$$y = -r\cos(k\lambda)$$

ここで, r, k は

$$k = \cos^2(\theta_1/2)$$
$$r = \frac{2}{\sqrt{k}}\sin\left(\frac{\theta}{2}\right)$$

である (θ は余緯度, θ_1 は標準緯線の余緯度).

(c) MPFCOA, MPICOA を呼ぶ前に, MPSCOA で標準緯線の緯度を設定しておかなければならない.

21.3.10 MPFCOC/MPICOC/MPSCOC

1. 機能

   ランベルト正角円錐図法の投影, 逆投影を行なう.

2. 呼び出し方法

```
CALL MPFCOC(XLON,YLAT,X,Y)
CALL MPICOC(X,Y,XLON,YLAT)
CALL MPSCOC(YLAT1,YLAT2)
```

3. パラメーターの説明

XLON,YLAT	(R)	経度, 緯度.
X,Y	(R)	x, y 座標.
YLAT1,YLAT2	(R)	標準緯線の緯度

4. 備考

(a) これは正角図法である.

(b) 定義式

$$x = r\sin(k\lambda)$$
$$y = -r\cos(k\lambda)$$

ここで, r, k は

$$k = \frac{\log\sin(\theta_2) - \log\sin(\theta_1)}{\log\sin(\theta_2/2) - \log\sin(\theta_1/2)}$$
$$r = \frac{\sin(\theta_1)}{k\tan^k(\theta_1/2)}\tan^k(\theta/2)$$

である (θ は余緯度, θ_i は標準緯線の余緯度).

(c) MPFCOC, MPICOC を呼ぶ前に, MPSCOC で標準緯線の緯度を設定しておかなければならない.

(d) この図法は中緯度の天気図の投影法として世界的に使われている.

21.3.11 MPFBON/MPIBON/MPSBON

1. 機能

ボンヌ図法の投影, 逆投影を行なう.

2. 呼び出し方法

```
CALL MPFBON(XLON,YLAT,X,Y)
CALL MPIBON(X,Y,XLON,YLAT)
CALL MPSBON(YLAT1)
```

3. パラメーターの説明

XLON,YLAT	(R)	経度, 緯度.
X,Y	(R)	x, y 座標.
YLAT1	(R)	標準緯線の緯度

4. 備考

(a) これは正積図法である.

(b) 定義式

$$x = r\sin(k\lambda)$$
$$y = -r\cos(k\lambda)$$

ここで, r, k は

$$k = \sin(\theta)/r$$
$$r = \theta - \theta_1 + \tan(\theta_1)$$

である (θ は余緯度, θ_1 は標準緯線の余緯度).

(c) MPFBON, MPIBON を呼ぶ前に, MPSBON で標準緯線の緯度を設定しておかなければならない.

(d) YLAT1=90 のとき, ヴェルネル図法 (ハート型図法) になる.

21.3.12 MPFOTG/MPIOTG/MPSOTG

1. 機能

   正射図法および Satellite View の投影, 逆投影を行なう.

2. 呼び出し方法

```
CALL MPFOTG(XLON,YLAT,X,Y)
CALL MPIOTG(X,Y,XLON,YLAT)
CALL MPSOTG(RSAT)
```

3. パラメーターの説明

XLON,YLAT	(R)	経度, 緯度.
X,Y	(R)	x, y 座標.
RSAT	(R)	地球の半径を単位とする衛星の軌道半径.

4. 備考

   (a) これは正積でも正角でもない図法である.

   (b) RSAT に 1 より大きい値を指定すると, その距離からの Satellite View になる. 1 以下の場合は, 通常の正射図法となる.

   (c) 定義式

$$x = r\sin(\lambda)$$
$$y = -r\cos(\lambda)$$
$$r = \frac{\cos(\theta)}{(1 - C\sin(\theta))}$$

   ここで C は RSAT>1 の時, RSAT の逆数, それ以外の時は $C = 0$ で通常の正射図法となる.

   (d) 半径 1 の円内に投影される.

21.3.13 MPFPST/MPIPST

1. 機能

   ポーラーステレオ図法の投影, 逆投影を行なう.

2. 呼び出し方法

```
CALL MPFPST(XLON,YLAT,X,Y)
CALL MPIPST(X,Y,XLON,YLAT)
```

3. パラメーターの説明

XLON,YLAT	(R)	経度, 緯度.
X,Y	(R)	x, y 座標.

4. 備考
   (a) これは正角図法である.
   (b) 定義式

$$x = r\sin(\lambda)$$
$$y = -r\cos(\lambda)$$
$$r = 2\tan(\theta/2)$$

21.3.14 MPFAZM/MPIAZM

1. 機能
   正距方位図法の投影, 逆投影を行なう.
2. 呼び出し方法
```
CALL MPFAZM(XLON,YLAT,X,Y)
CALL MPIAZM(X,Y,XLON,YLAT)
```
3. パラメーターの説明

XLON,YLAT	(R)	経度, 緯度.
X,Y	(R)	x, y 座標.

4. 備考
   (a) これは正積でも正角でもない図法である.
   (b) 方位線に沿った長さが正しく投影される.
   (c) 定義式

$$x = r\sin(\lambda)$$
$$y = -r\cos(\lambda)$$

   ここで, r は

$$r = \theta$$

   である.
   (d) 半径 π の円内に投影される.
   (e) 図の中心は北極である.

21.3.15 MPFAZA/MPIAZA

1. 機能
   ランベルト正積方位図法の投影, 逆投影を行なう.
2. 呼び出し方法
```
CALL MPFAZA(XLON,YLAT,X,Y)
CALL MPIAZA(X,Y,XLON,YLAT)
```
3. パラメーターの説明

`XLON,YLAT`	(R)	経度, 緯度.
`X,Y`	(R)	x, y 座標.

4. 備考

(a) これは正積図法である.

(b) 定義式

$$x = r\sin(\lambda)$$
$$y = -r\cos(\lambda)$$

ここで, r は

$$r = 2\sin(\theta/2)$$

である.

(c) 半径 2 の円内に投影される.

(d) 図の中心は北極である.

(e) この図法は地震学や構造地質学で広く使われている.

第22章　GT2DLIB：2次元補間/変換

22.1　概要

本ライブラリーは2次元の補間、並びにそれを用いた2次元の座標変換を扱う。座標間の対応は離散的な格子点で指定し、間は補間する。補間アルゴリズムは双線形補間を用いる。このため逆変換は解析的に計算される。将来的には双3次補間もサポートする可能性があるが、その場合逆変換は反復法によることになる。
補間のサブルーチンは、座標変換のみならず一般の用途に適するよう作ってある。
座標変換は、$(\xi,\eta)->(x,y)$ において、離散的な点 $\xi_i(i=0,1,2,..),\eta_j(j=0,1,2,..)$ で

$$x_{i,j}=x(\xi_i,\eta_j)$$
$$y_{i,j}=y(\xi_i,\eta_j)$$

という形で表される格子点での対応関係により定義される。従って「正変換」においては、変換元の座標のデータは2つの1次元配列、変換先の座標のデータは2つの2次元配列となる。逆変換はその逆である。なお、2つの2次元配列から2つの2次元配列へと対応づけられる座標変換は、中間的に1次元配列ベースの「矩形」格子を設定すれば、逆変換、正変換の2段階で行うことが出来る。これは、幾何的には、基準となるデカルト座標系を用いることに相当する。

22.2　関数のリスト

`G2FBLI(P, Q, Z00, Z10, Z01, Z11, Z)`	双線形補間
`G2FBL2(P,Q, X00,X10,X01,X11, Y00,Y10,Y01,Y11, X,Y)`	2変数同時の双線形補間
`G2IBL2(X,Y, X00,X10,X01,X11, Y00,Y10,Y01,Y11, P,Q)`	G2FBL2の逆変換
`G2SCTR(NX, NY, UXA,UYA, TXA,TYA)`	2次元の座標変換の設定
`G2FCTR(UX, UY, TX, TY)`	2次元の座標変換
`G2ICTR(TX, TY, UX, UY)`	G2FCTRの逆変換
`G2QCTR(LINIT)`	初期化済かの問合わせ
`LG2INQ(TX,TY, TX00,TX10,TX01,TX11, TY00,TY10,TY01,TY11)`	四辺形の内側にあるか判定

22.3　関数の説明

22.3.1　G2FBLI

1. 機能

   双線形補間. 入力座標 (P, Q) はともに [0, 1] の区間に正規化されたものをとる.

2. 呼び出し方法

```
CALL G2FBLI(P, Q, Z00, Z10, Z01, Z11, Z)
```

3. パラメーターの説明

P,Q	(R)	補間すべき点の座標. 4 隅が (0,0), (1,0), (1,1), (0,1) となるよう正規化されてるものとする. (入力)
Z00, Z10, Z01, Z11	(R)	P,Q の 4 隅 (上述)、即ち、それぞれ (P,Q) = (0,0), (1,0), (1,1), (0,1) における変数の値 (入力)
Z	(R)	補間結果 (出力)

4. 備考

(a) 中身は Z = (1-P)*(1-Q)*Z00 + P*(1-Q)*Z10 + (1-P)*Q*Z01 + P*Q*Z11 である (一行のみ).

(b) P, Q とも [0,1] の区間内にある場合は補間, そうでなければ外挿 となる.

(c) パラメターの説明から明らかであろうが, (P,Q) = (0,0), (1,0), (1,1), (0,1) の場合、出力はそれぞれ Z00, Z10, Z01, Z11 に等しくなる.

22.3.2 G2FBL2

1. 機能

2 変数同時の双線形補間. 入力座標 (P, Q) はともに [0, 1] の区間に正規化 されたものをとる.

2. 呼び出し方法

```
CALL G2FBL2(P,Q, X00,X10,X01,X11, Y00,Y10,Y01,Y11, X,Y)
```

3. パラメーターの説明

P, Q	(R)	補間すべき点の座標. 4 隅が (0,0), (1,0), (1,1), (0,1) となるよう正規化されてるものとする. (入力)
X00, X10, X01, X11	(R)	P,Q の 4 隅 (上述)、即ち、それぞれ (P,Q) = (0,0), (1,0), (1,1), (0,1) における一つめの変数の値 (入力)
Y00, Y10, Y01, Y11	(R)	P,Q の 4 隅 (上述)、即ち、それぞれ (P,Q) = (0,0), (1,0), (1,1), (0,1) における二つめの変数の値 (入力)
X	(R)	X00, X10, X01, X11 に関する補間結果 (出力)
Y	(R)	Y00, Y10, Y01, Y11 に関する補間結果 (出力)

4. 備考

(a) G2FBLI を 2 回呼んでいるだけである。よって X = (1-P)*(1-Q)*X00 + P*(1-Q)*X10 + (1-P)*Q*X01 + P*Q*X11 Y = (1-P)*(1-Q)*Y00 + P*(1-Q)*Y10 + (1-P)*Q*Y01 + P*Q*Y11

(b) 2 変数から 2 変数への変換であるから逆変換が存在する. 逆変換を行うサブルーチンは G2IBL2 である.

22.3.3 G2IBL2

1. 機能

G2FBL2 (2 変数同時の双線形補間) の逆変換. 解析解である.

2. 呼び出し方法

```
CALL G2IBL2(X,Y, X00,X10,X01,X11, Y00,Y10,Y01,Y11, P,Q)
```

3. パラメーターの説明

X,Y, X00,X10,X01,X11, Y00,Y10,Y01,Y11	(R)	(入力)
P, Q	(R)	(出力)

4. 備考
   (a) パラメーターの説明は G2FBL2 を参照のこと.
   (b) 連立２次方程式を解析的に解く. (実質的に１次方程式になる場合等、 多くの場合分けを含む)

22.3.4 G2SCTR

1. 機能
   2 次元の座標変換の設定 (初期化). 格子点間の対応で定義する.
2. 呼び出し方法
   ```
   CALL G2SCTR(NX, NY, UXA,UYA, TXA,TYA)
   ```
3. パラメーターの説明

NX, NY	(R)	配列の長さ (入力)
UXA	(R)	変換元の座標 (独立変数その 1). 長さ NX の 1 次元配列 (入力).
UYA	(R)	変換元の座標 (独立変数その 2). 長さ NY の 1 次元配列 (入力).
TXA	(R)	変換先の座標 (独立変数その 1). 長さ NX*NY の 2 次元配列 (入力). 第 1 要素 (TX(1,1)) を RUNDEF とすることで省略できる. 省略の場合,TX(I,J)=UX(I) が用いられる.
TYA	(R)	変換先の座標 (独立変数その 2). 長さ NX*NY の 2 次元配列 (入力). 第 1 要素 (TY(1,1)) を RUNDEF とすることで省略できる. 省略の場合,TY(I,J)=UY(J) が用いられる.

4. 備考
   (a) 2 次元配列 (TXA,TYA) のセーブは、C 言語で書かれた下請の関数が行う。動的割り当てによるため、サイズに制限がない。一方、一次元の UXA,UYA は静的に確保されたサブルーチン内の配列にセーブされる。最大の長さは GRPH2 の UWPACK と同様、コンパイル時に変更可能な定数 MAXNGRID である。デフォルト長は 4000。

22.3.5 G2FCTR

1. 機能
   2 次元の座標変換を行う.
2. 呼び出し方法
   ```
   CALL G2FCTR(UX, UY, TX, TY)
   ```
3. パラメーターの説明

UX	(R)	変換前の座標（その 1） (入力)
UY	(R)	変換前の座標（その 2） (入力)
TX	(R)	変換後の座標（その 1） (出力)
TY	(R)	変換後の座標（その 2） (出力)

4. 備考

(a) 予め G2SCTR で初期化しておかないとエラーを発生する。

(b) G2SCTR で設定した UXA, UYA の範囲外の入力に対しては、 警告を出しつつ、外挿を行う。

22.3.6 G2ICTR

1. 機能

   G2FCTR (2 次元の座標変換) の逆変換を行う.

2. 呼び出し方法

```
CALL G2ICTR(TX, TY, UX, UY)
```

3. パラメーターの説明

   TX (R) 逆変換前の座標（その 1） (入力)

   TY (R) 逆変換前の座標（その 2） (入力)

   UX (R) 逆変換後の座標（その 1） (出力)

   UY (R) 逆変換後の座標（その 2） (出力)

4. 備考

   (a) 予め G2SCTR で初期化しておかないとエラーを発生する。

   (b) G2SCTR で設定した TXA, TYA の範囲外の入力に対しては、 エラーを発生する。

22.3.7 G2QCTR

1. 機能

   初期化済か否かの問合わせ. なお、初期化せずに G2FCTR, G2ICTR を呼べばエラーがでるようになっているので、問合わせは必須ではない。

2. 呼び出し方法

```
CALL G2QCTR(LINIT)
```

3. パラメーターの説明

   LINIT (L) 初期化済なら .TRUE. まだなら .FALSE. を返す。(出力)

4. 備考

   (a) G2SCTR が既に呼ばれていれば .TRUE.

22.3.8 LG2INQ

1. 機能

   ある点が四辺形の内側にあるか判定する。凸四辺形限定。

2. 呼び出し方法

```
LIN = LG2INQ(TX,TY, TX00,TX10,TX01,TX11, TY00,TY10,TY01,TY11)
```

3. パラメーターの説明

   TX,TY (R) 判定すべき点の座標 (入力)

   TX00,TX10,TX01,TX11, TY00,TY10,TY01,TY11 (R) 四辺形の 4 隅の座標 (入力)

   戻り値 (L) 内側または境界上にあれば .TRUE. 外側なら .FALSE. を返す.

4. 備考

(a) G2ICTR が下請として用いる.

(b) 凹四辺形には対応していない。座標変換においては、凹四辺形は多価性のある変換にしか現われない。多価の場合は、逆変換はそもそも一意に決まらないので、利用価値が低い。将来的には、凹四辺形に対応してもよい。

22.3.9 下請

FUNCTION G2SGRD, FUNCTION G2QGRD

G2SCTR の下請であり、動的な配列確保のため C で書かれている。それ以外に独立な関数とするべき理由はないので説明は割愛する。

第23章　CLSPLIB：色空間変換

23.1　概要

色空間を扱うための関数がそろえられている. UIPACK のためにデータと描画イメージの変換に使用される。

23.2　関数のリスト

CLLSRG　HLS に対応する RGB 値を返す.
CLRGLS　RGB に対応する HLS 値を返す.
CLSVRG　HSV に対応する RGB 値を返す.
CLRGSV　RGB に対応する HSV 値を返す.
INORML　整数の配列を規格化する.
RNORML　実数の配列を規格化する.

23.3　関数の説明

23.3.1　CLLSRG

1. 機能
   HLS に対応する RGB 値を返す.
2. 呼び出し方法
   ```
   CALL CLLSRG(H,L,S,R,G,B,N,M)
   ```
3. パラメーターの説明
   H,L,S　(R)　HLS 値.
   R,G,B　(I)　対応する RGB 値.
   N,M　(I)　配列の X,Y 方向の大きさ.
4. 備考
   (a) なし

23.3.2　CLRGLS

1. 機能
   RGB に対応する HLS 値を返す.
2. 呼び出し方法
   ```
   CALL CLRGLS(R,G,B,H,L,S,N,M)
   ```
3. パラメーターの説明

R,G,B (I) RGB 値.
H,L,S (R) 対応する HLS 値.
N,M (I) 配列の X,Y 方向の大きさ.

4. 備考
   (a) なし

23.3.3 CLSVRG

1. 機能
   HSV に対応する RGB 値を返す.
2. 呼び出し方法
   ```
   CALL CLSVRG(H,S,V,R,G,B,N,M)
   ```
3. パラメーターの説明
   H,S,V (R) HSV 値.
   R,G,B (I) 対応する RGB 値.
   N,M (I) 配列の X,Y 方向の大きさ.
4. 備考
   (a) なし

23.3.4 CLRGSV

1. 機能
   RGB に対応する HSV 値を返す.
2. 呼び出し方法
   ```
   CALL CLRGSV(R,G,B,H,S,V,N,M)
   ```
3. パラメーターの説明
   R,G,B (I) RGB 値.
   H,S,V (R) 対応する HSV 値.
   N,M (I) 配列の X,Y 方向の大きさ.
4. 備考
   (a) なし

23.3.5 INORML

1. 機能
   整数の配列を規格化する.
2. 呼び出し方法
   ```
   CALL INORML(V,W,N,M,X,Y)
   ```
3. パラメーターの説明

V (I) 元の値.
W (R) 規格化された結果.
N,M (I) 配列の X,Y 方向の大きさ.
X,Y (R) 規格化する最小最大値.

4. 備考
   (a) なし

23.3.6 RNORML

1. 機能
   実数の配列を規格化する.
2. 呼び出し方法
   ```
   CALL RNORML(V,W,N,M,X,Y)
   ```
3. パラメーターの説明
   V (R) 元の値.
   W (R) 規格化された結果.
   N,M (I) 配列の X,Y 方向の大きさ.
   X,Y (R) 規格化する最小最大値.
4. 備考
   (a) なし