ADAM

産業連関分析システム

(Version 6.0)

はじめに

今日，産業連関表を用いた分析は多くの分野で幅広く行われています．そして，分析の対象となる産業連関表は時と共に増え，その分析内容も多様さを増しています．産業連関分析に限らず一般に実証分析の第一歩はデータの入手であり，分析にコンピュータを用いることが普通となった今日ではその入力から始まります．そして，それに続く分析計算も多くの労力を必要とします．

分析の計算を担うコンピュータ環境は日々充実してきています．以前はメインフレーム（大型汎用機）でなければ産業連関分析はできなかったと言っても過言ではありません．しかし，今ではパーソナルコンピュータ（以下パソコン）を用いて分析を行うことができます．部門数が少なければスプレッドシート(表計算ソフト）で行うこともできます．自らプログラムを作成すればかなり多い部門数の分析を行うこともできます．産業連関分析のためのコンピュータ環境は，メインフレームの利用から個人所有のパソコンに移行しており，コンピュータがより身近な分析用具になってきたといえます．

スプレッドシートの出現により，自分自身の操作で様々な計算処理が可能であることを知り，またその充実したグラフ作成機能などがパソコンの利用に拍車をかけました．多様な産業連関分析のためのデータとプログラムを準備するにはある程度の経験を必要とします．今日では，データはＣＤや各機関の Web サイトからの入手が可能になってきていますが，その活用は容易ではありません．このために，スプレッドシートのような簡便さを持った分析システムの開発とそれに連動したデータの整備が切望されることになります．

産業連関分析システムの開発実績は，かなり古くからあります．ハーバード経済研究所のベンツ C.W.Benz によって研究・開発された PASSION（1971 年）はその草分けです．また日本経済新聞社データバンク局の NEEDS-TS の MERLIN(1974 年)が行列演算で分析を進める形を整えた最初のシステムです．これは，現在は稼働を中止していますが，メインフレームを用いたシステムです．また，マクロエコノメトリックス研究会の ECOMOMATE I-O も広く利用されているシステムです．また，最近では，(株)アクト・ブレーン社の「産業連関王 I・O king」も広く利用されています．安田秀穂氏が Excel マクロの Visual Basic で開発した経済波及効果算出ツール「波及さん」(『自治体の経済波及効果の算出』，学陽書房，2008 年３月)も注目すべきソフトでしょう．

ADAM（Advanced system for analysis of input-output models）は，産業連関分析をより効率的に行うためのシステムであり，次の３つを開発の基本軸としています．第１は，システムのデータの記述を素朴な形式にして，多種多様なデータのデータベース化を図れるようにしています．第２は，多くの利用者がこのシステムの利用を通して作成したデータの交換を可能にし，入力の手間の軽減と分析の効率化を互いに図ることのできる基盤の提供であります．第３は，このシステムは，これ自体での完結を目指すものでなく，システムをオープンにして，特殊な処理や結果のプレゼンテーションは他のより便利なアプリケ

ーションソフトウェアの利用を前提にし，データベースの効率的活用のもとに各自の分析目的に応じたデータ処理ができることをこのシステムの役割としています．このため，他のアプリケーションソフトウェアとの間のデータ交換機能の充実が重要であると考えています．

ADAM（version3）は，その前身である ADAM（version 1，version 2）の構想を生かし，そこに改良を加え，VC++言語ですべてを作り替えた新たなプログラムです．Version1 は DOS 環境で動作する GNU-C 言語によるものでした．Version2 は，Java 言語を用い，動作環境を Windows に変えましたが，処理速度に難がありました．処理速度の改善に力を注ぎ，言語 VC++での開発となりました．2000/XP/Vista の環境で動作します．また，ADAM（version4）は，分析を主としていた ADAM（version3）の機能をそのまま引継ぎ，要望がありました作表機能とデータの抽出・組み替え機能を追加したものです．なお，Version3， version4 は Microsoft Visual Studio 6.0 による開発でした．

ADAM（version5）は，Microsoft Visual Studio 2005 .NET Framework による開発であり，すべてを書き換えました．基本的な構想，機能は同じですが，さまざまな初期値の設定を幅広く用意し，整数値の桁数を 9 桁から 19 桁に拡張，内部処理コードを MBCS(マルチバイト文字セット)から Unicode に変更しました．また，Excel シートとのデータ交換を容易にしています．

今回の ADAM（version6）は，ユーザーコードやパスワードによる管理をすべて取りやめ、誰でもが自由に利用できる環境にしました．また，Excel シートとのデータ交換機能をなくしました．多くの利用者が使用している Microsoft Office のバージョンが多く、それに対応させることが複雑になったためです．Excel シートとのデータ交換が効率悪いこともその一因です．CSV 形式でのデータ交換機能をご利用ください。

2015 年 3 月
慶應義塾大学
新井 益洋

はじめに

概要

コマンド

トピックス

概要

◇特徴

ADAM の特徴の第１は，データベースとの連動である．分析においてデータの確保はその第一歩であり，多くの労力を必要とする避けることのできない過程である．産業連関表は公表されると以後変わることがないからデータベース化は容易であり，そのデータ量も膨大であるから，その効果は大きいといえよう．ADAM のデータベースは，初期入力データだけでなく，分析の過程で得られた各種統合表や計算結果も，すべて同じ形式で保存し後の利用に供することができるようになっている．これを可能にしたのはデータ記憶形式の単純化にある．ADAM では，このデータベースをデータライブラリと呼ぶ．

その第２は，統合処理の容易性である．分析に用いる表は，その目的に応じて統合される．基本表から統合する必要もあれば，統合中分類表からの統合で事足りることもある．どのような場合でも肝心なことは分析に合った統合処理が自由にできることである．ADAM の統合処理機構は柔軟であり，どのような大きさの表でも処理でき，かつ最小限の指示で目的を達成することができる．

その第３は，データ処理の容易性である．産業連関分析は行列演算を必須とするが，スカラー演算もできなければならない．　コードや名称などの文字列も扱えなければならない．ADAM は，行列演算の範囲である添字の指定を連続な場合だけでなく不連続な場合も考慮して集合で定義することにしている．また，投入係数算出の場合に見られるように，演算における添字要素の「１対多」の対応を具現化し，処理の指示を簡明にしている．また，頻繁に使用したり，変更される部分の置き換えを簡潔にするために，また複数のコマンドを一纏めにした新コマンドに似た記号名の定義のために，置換記号を導入している．さらに一連のコマンドをファイル化し，それを実行させることにより，事後の処理内容の確認と，若干の変更による新たな分析などの弾力的運用を可能にしている．

その第４は，他のアプリケーションソフトウェアとの連携を図ることである．データの入力処理から結果のプレゼンテーションまでを完全に網羅するシステムの開発は望まれることではあるが，その保守・更新までを考えるとそこに要する労力は膨大であり，極めて困難であると言わざるをえない．現存する様々なアプリケーションソフトウェアを見ても分かるように，それぞれは固有の特徴を持つことによってその存在を主張している．各アプリケーションソフトウェアが得意とする機能を巧みに組み合わせて利用することが，多くの分析者にとって最も賢い選択であろう．ADAM は，この観点に立ち，他のアプリケーションソフトウェア間のデータ交換機能の充実を目指している．現在はスプレッドシートとの連携機能，データの抽出・組み替え機能を用意しているに過ぎないが，徐々に充実を図って行きたい．

◇処理構造

ADAM は所定の形式で記述した命令（コマンド）にしたがって処理を行う．このコマンドはコンピュータ言語に似ており，まさに，コンピュータ言語による処理といえる．すなわち，データを読み込み，演算を行い，結果を書き出す，という手順を記述していくのである．一般のコンピュータ言語が持つ if 文に準ずる機能および goto 文，for 文も装備している．ADAM では，一つの処理を記述した一纏まりの処理コマンドをプロシージャと呼ぶ．

ADAM コマンドは簡明な構造を追求し，その扱いの容易性に重点を置いている．このため，複雑な処理や特殊な処理についての機能は十分でないといえよう．しかし，演算式上で与える添字要素の対応や工夫および各種関数の使用によって，処理可能な範囲をかなり広げることができると考えている．

ある一連の分析処理を「プロジェクト」と呼ぶ．プロジェクトごとにフォルダ名を与え，その場所にそのプロジェクトが必要とするデータファイルとプロシージャファイル（実行ファイル）のすべてを格納することを原則にしている．これは数多く行われる分析処理の管理を容易にするためである．このフォルダを「プロジェクトフォルダ」と呼ぶ．

しかし，頻繁に利用される基本的なデータや定型化されたプロシージャを各プロジェクトフォルダに，その都度複写し準備することは効率的でない．このため，プロジェクトに固有でない共用データと共用プロシージャを独立した専用フォルダに格納し，共同利用できるようにしている．それぞれを「データフォルダ」，「プロシージャフォルダ」と呼ぶ．「データフォルダ」，「プロシージャフォルダ」は複数指定することができる．

処理コマンドを実行していく過程で要求される読み込みデータは，先ず各プロジェクトフォルダ内が検索される．そこに存在すればそれが読み込まれる．もしプロジェクトフォルダ内に存在しなければ，そしてデータフォルダの指定があれば，その中が検索され，存在すればそれが読み込まれる．データフォルダが複数指定されていれば，指定された順に検索が行われる．何れのフォルダでも発見できなければ，エラーとなる．

データの書き出しは，すべて常にプロジェクトフォルダ内に行われる．他のフォルダに書き出すことはできない．上書き事故回避のための措置である．

プロシージャについては，現在実行中のプロシージャの中，プロジェクトフォルダ，プロシージャフォルダがこの順に検索され，最初に発見されたプロシージャが使用される．プロシージャフォルダが複数指定されていれば，指定された順に検索が行われる．何れのフォルダでも発見できなければ，エラーとなる．呼び出しプロシージャ名とそれに対応するファイル名の関係は ADAM コマンドのプロシージャの実行(proc)で述べる．

beginproc と endproc で括られていないプロシージャを主プロシージャと呼ぶ．ADAM は，主プロシージャの実行から始めるので，主プロシージャが実行プロシージャの中に必ず存在しなければならない．

一つの処理プロシージャをテキストエディタで準備をしておき，それを ADAM 実行画面で実行指示をする．実行指示は実行ボタンをクリックするか Enter キーを押す．もし，コマンドにエラーがあれば，そこで処理が止まる．テキストエディタで該当プロシージャを呼び出し，エラー箇所を修正した上で「上書き保存」を行い，再度 ADAM 実行画面で実行指示をする．この繰り返しで，所期の目的を達成する．

コンピュータ画面には，ADAM 実行画面とテキストエディタ画面を併存させた方が作業効率がよい．また，シートデータ出力をすれば，表計算ソフトでその内容を確認することになるので，３画面を併存させることもある．

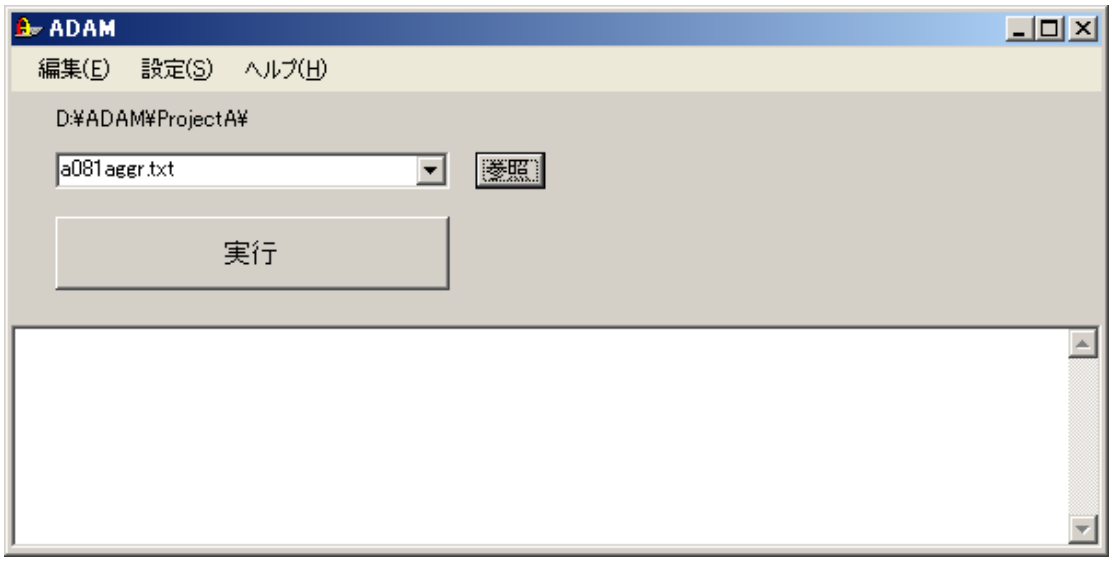
ADAM 実行画面上には，処理確認のために処理経過や処理結果の一部を表示コマンドで表示させることができる．

ADAM 実行画面の枠には，実行させたいプロシージャファイル名を指定する．枠の横にある[参照]ボタンをクリックし，ファイルダイアログで選択をすれば簡単である．このときのフォルダがプロジェクトフォルダと認識され，その名前が枠の上に表示される．

また，実行済ファイル名は自動的にコンボボックスに記録されるので，再実行させたいときは項目リストから実行ファイルを選択することができる．

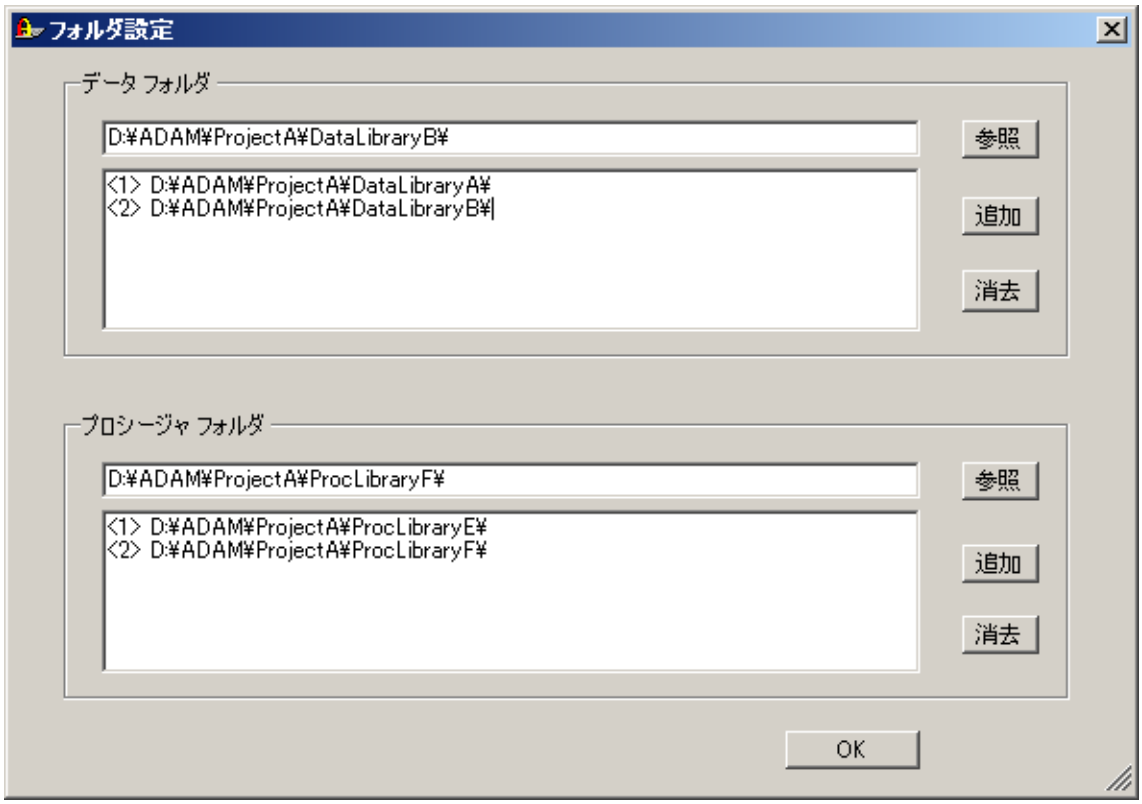
マニュアル表示は，ヘルプメニュー内にある．F1 キーでも表示させることができる．

ADAM の実行画面

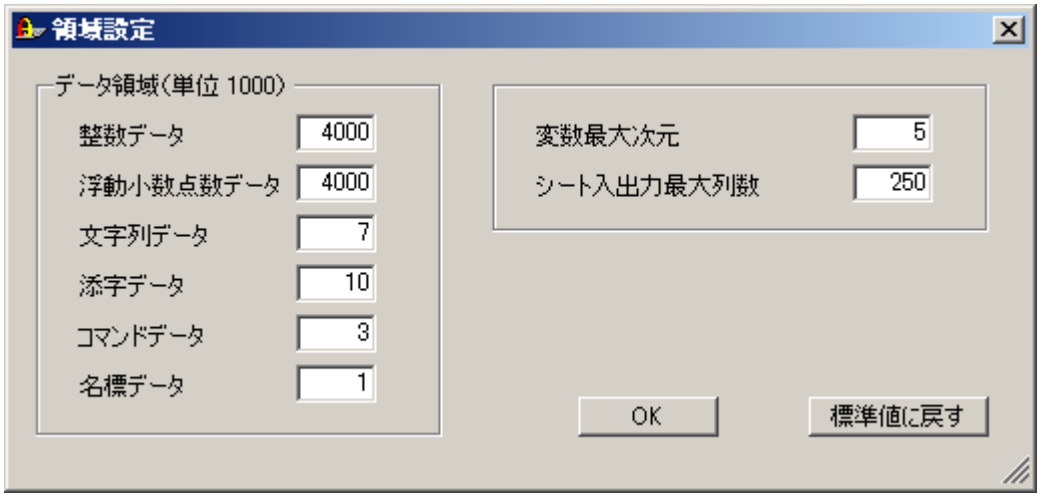


「データフォルダ」と「プロシージャフォルダ」は，ADAM 実行画面の「設定」からフォルダ設定画面を呼び出し，設定を行う．どちらのフォルダも所定のフォルダを参照ボタンで選択し，正しければ追加ボタンを押す．フォルダは最大５個まで指定することができ，優先順位は指定した順になる．指定を変更したいときは，消去ボタンで全体を消去し，再度指定を行う．フォルダ指定の必要がなければ指定をしなくてもよい．以下にフォルダ設定画面を示す．

ADAM のフォルダ設定画面



ADAM の領域設定画面



ADAM 実行時のデータ領域や変数の最大次元などは，ADAM 実行画面の「設定」から領域設定画面を呼び出し，設定を行う．

以下の領域設定画面に表示されている値は，それぞれの標準値であり，400～500 部門程度の分析計算および作表を想定したものである．その単位は 1000 である．

整数および浮動小数点数は 400 万データ，文字列は 7000 データ，1 つのコマンドで指定できる添字の総数は 1 万個，処理プロシージャは 3000 行，変数名，置換記号名，部門分類名などの名標は 1000 個などとなっている． 500×500 の行列データは，25 万データの記憶領域を必要とする．したがって，標準値の 400 万データ領域は，この大きさの行列であれば 16 個程度の領域でしかない，さらに，double や integer 文で宣言した行列以外に，実際の計算過程では，システムが一時的に使用する 2～3 個の同じ大きさの行列が必要となる．大規模計算を行うときは，宣言したデータ領域に，一時的使用領域を加え，また若干の余裕も持たせ，得られた総量をデータ領域枠に指定する．また，変数の最大次元は 5，シートの入出力の最大列数は 250 である．

領域設定の値を標準値に戻したいときは，「標準値に戻す」ボタンを押す．

◇基本事項

■コマンドの区切り

コマンドの最後にはセミコロンを付ける．これは複数行にわたるコマンドの指示や同一行内に複数のコマンドの指示を可能にするためである．改行記号はセミコロンの代わりにならない．

ただし，def コマンドだけは例外であり，改行記号または連続した２個のセミコロンのみがその終わりを示す．これはセミコロンをその値として使用できることを可能にするためである．

複数のオプションや変数，指示事項などがあるとき，それらの区切りは，特殊な場合を除き，コンマでも空白でもよい．

■コマンドのコメント

コマンドの任意の箇所にコメントを挿入することができる．コメント部分を記号「/*」と「*/」でくくる．括られた部分は完全な無記述扱いとなり，コマンドには一切影響を与えない．コメントは複数行にわたってもよい．ただし，引用符を用いた文字列の中ではこのコメント指示は無効である．

また，記号「//」を指示すれば，この記号から同行の改行記号までの部分がコメントとなる．同一行内のコメントであり，次行への影響はない．引用符文字列に優先する．

■データのコメント

データ入力コマンドで読み込まれるデータは，行の先頭文字が # であれば，その行全体がコメント扱いになり，その行の内容はすべて無視される．上述の「コマンドのコメント」の記述はできない．

■プロシージャファイル

ADAM は，一連の処理のコマンドをファイル化しておき，それを実行させ，所期の目的を達成する．このファイルをプロシージャファイル（あるいは実行ファイル）と呼ぶ．プロシージャファイルはテキストエディタで事前に編集をしておく．実行中のプロシージャから別のプロシージャ(ファイル)を呼び出すこともできる．

■置換記号

任意の記号名に自由な数値や文字列(これを置換テキストと呼ぶ)を対応（定義）させておき，その記号名をコマンド内に使用する．そして，コマンドの実行時にその部分を対応させた置換テキストに置き換えて実行する機能である．使用例を示そう．

```
def  sz  80  /* 部門の大きさ */
```

```
double   x[@sz,@sz];
```

def コマンドで記号名 sz を 80 と定義している．次に，その記号名を使用し浮動小数点数の配列変数を宣言している．この結果は x[80,80] となる．定義のときは記号名に @ を付けないが，それを使用するときは記号名の先頭に @ を付けなければならない．記号「/*」と「*/」で括られた部分はコメントであり，無視されるので，置換テキストの一部にならない．次の例をみてみよう．

```
def  a  @b.gr@c
def  b  ag
def  c  m
 …
@a  …;
```

最終行の @a …; は置換の結果 aggrm …; となる．これは統合のコマンドである．１行目の @b.gr@c のピリオドは，記号名 @b と固定文字列 gr とを分離する役割を果たすためのものであり，固定文字の一部とはならない．また，置換テキスト内に別の置換記号名が現れているが， def コマンドは置換記号の定義を行うことが目的なので何ら差し支えない．また，この定義段階では置き換えをしない．実際に置き換えが行われるのは，その置換記号が実際に使われるときである．したがって， @b や @c がその後に定義されていても何ら問題ない．

置換記号の使用には大きく次の２つがある．第１は，一律に変わるであろう数値や変数名，文字列部分を置換記号で表現することである．部門数や行列の大きさ，添字の指示などは，一連の処理コマンド内で随所に出現することになるが，同じ数値を何度も入力することは懸命でない．なぜなら，その値が変わったとき，それらすべてを変えなければならない．この労力は大変であると同時に変更間違いをすることが多々ある．また，違う配列変数に対して同じ処理を行いたい時などは，変数名に置換記号を用いてコマンドを作成すれば，容易にその目的を達成することができる．

第２は，置換記号名に一つないし複数のコマンドを対応させ，新しいコマンドに似た記号名を定義して，効率的な処理コマンドの指示を行うことである．ADAM は，置換テキストの中に別の置換記号名を指定することが可能であり，工夫によっては便利な処理コマンドの構築が可能となる．

■実行条件式

すべてのコマンドに対し，それを実行するか否かの条件を与えることができる．コマンドの前に丸括弧でくくりその条件を与える．多くの言語が持つ if(…) 文の if が無い記述と考えればよい．条件が成立すればそのコマンドは実行され，不成立であればそのコマン

ドは実行されない．実行条件式が与えられていないときは，そのコマンドは無条件で実行される．

ある条件付き実行コマンドが複数行あるときは，それら全体を中括弧｛｝でくくる．また，中括弧｛｝の中に新たな実行条件式を指示することもできるので，きめ細かな制御が可能である．

■関数

ADAM には幾つかの関数が用意されている．行列を転置したり，逆行列を求めたり，文字列コ ードを探索してその位置を知ったりするなどである．詳しくは，後出の「◇関数」を参照．

例：inverse(b[1:50,1:50])　index(rowcode,0211200)

■データの要素

ADAM が扱うデータの要素には，整数，浮動小数点数，文字列がある．整数値は小数を持たない約 19 桁の符号付き値であり，浮動小数点数は，仮数(有効桁数 15 桁)×10^{-308}～10^{+308}の符号付き値で小数を持つことができる．文字列は英数字および一部の特殊文字，全角文字で構成される文字列である．引用符で括っても括らなくてもよいが，空白やコンマ，セミコロン，コロン，括弧などを含むときは，引用符でくくる必要がある．

例：123　12.345　011101　″0111011″　″goods and services″

文字列が長いときは，改行をすることなく長いまま与えてもよいが，複数の文字列を演算子(+)で連結指示をすることができ，全体は１つの文字列として認識される．ただし，文字列連結演算子(+)は，引用符文字列の右引用符に続けて(空白を置かない)与えなければならない．引用符がない文字列の連結はできない．

```
例：″goods and ... ″+
      ″... services″;
```

■使用文字

コマンドや変数名,置換記号名は,英字で始め 2 文字目以降を英数字とアンダースコア(_),あるいは漢字などの全角文字で構成する．英大文字はすべて英小文字に変換される．したがって，英大文字と英小文字の違いによる識別指示はできない．ただし，ファイル名や置換記号，文字列値は何の変換も行われなし，全角文字を使用することも自由である．コマンド記述に全角文字を使用しても差し支えない．

■記号@

記号 @ は置換記号の識別のために使用される．このため，もしファイル名の一部などに記号 @ を一つの文字として用いると都合が悪いことになる．このときは，その部分を @@ と

することで回避できる． @@ は１字の @ となり，かつ置換記号の識別の対象とならなくなる．引用符でくくられた中でも機能する．

■引用符

文字列値の表記には引用符を用いる．しかし，空白やコンマ，セミコロン，コロン，括弧などを含まない文字列値の場合は引用符を用いなくてもよい．引用符には単一引用符と二重引用符のどちらを用いてもよい．単一引用符を用いた場合，その中に単一引用符を文字として使用したいときは単一引用符を二個その間を開けずに指定する．この時，その中の二重引用符は単なる文字として認識される．また同様に，二重引用符を用いた場合，その中に二重引用符を文字として使用したいときは，二重引用符を二個その間を開けずに指定する．この時，その中の単一引用符は単なる文字として認識される．

例：’producers’’ price’　″producers’price″　″producers’price″

■添字

添字は連続な場合だけでなく不連続な場合も考慮して集合で定義する．添字要素はコンマあるいは空白で区切る．スカラーおよび配列の整数変数も添字の指定として用いることができ，その変数が持つ値を指定したことになる．各添字要素の加減演算が可能である．また，添字の集合は中括弧｛｝でくくる．しかし，その指定が１つあるいは１組の連続指定であるときは中括弧を省略してもよい．

例:　5　1:7　{1,5,n+6}　m[k,j+p]

連続指定は開始と終了をコロンで連結する．さらにコロンで連結し増減値を指定することができる．

例:｛1,5,10:20｝　｛10:2:-2,20｝　1:65:1

検索関数 index() を用いれば，添字に部門コードを指定することができる．その形式は，

index(codevar,code)

である．ただし， codevar はその位置の確定に必要な部門コードを収めたベクトル変数であり， code は変換対象のコードまたはコードを収めた文字列変数である．

例:　｛1:index(rcd,9311000)｝　｛index(rcd,g[k]):index(rcd,g[k+1])｝

要素除外の指示ができる． exclude または exc の後ろに除外したい要素を指定する．この除外要素が，その直前の添字集合より除外される．そこに該当する除外要素が無ければその除外は行われないし，エラーにもならない．直前部分とは，exclude の左側部分の直近の左中括弧｛までの範囲である．

例：{1:10 {20:30 exc 25 28}}

次は似た指示であるが，エラーになるので注意を要する．

例：{1:10 {20:30 {exc 25 28}}}

■作表支援機能

作表支援機能は，産業連関表の作成を円滑に進めるために組み込まれたものである．その第１は，作成する表の行および列の部門分類の設定であり，それぞれの部門コードと名称を与え部門分類表を定義する(makesectortable)．これは部門分類の一元化と処理の簡便さを目的としたものである．この部門分類を「部門分類表定義名」で保存し，必要時に呼び出して使用する(sectortable)．

第２は，変数定義の簡易化である変数の定義は行および列の大きさを直接与えるのではなく，定義した部門分類表定義名で指示する．この変数を「部門分類付変数」と呼び，これまでの変数である「標準変数」と区別する．演算などのコマンドの指示は同じであるが，部門分類付変数については index(…)関数の指示が簡単になっている．通常の指示も使用可能であるが，変数に部門分類表が関連付けられているので，コードの先頭に$記号を付けただけの指示が可能になる．

第３は，作業途中の変数内容の保存と復元の機能である．作表は試行錯誤を重ねる複雑な作業の繰り返しである．細かな作業過程は，その都度その結果を確認しながら進めていくことになる．作業途中の中間結果(変数内容)の保存はファイル書き出し(put)で行い，内容確認後，その時点からの作業継続に必要となる変数復元はファイル読み込み(get)で行う．

第４は，与えたマージン表に基づき，形態別マージンを算出し，購入者価格評価表示から生産者価格評価表示への組み替え処理(margindrop)が，また生産者価格評価表示から購入者価格評価表示への組み替え処理(marginback)ができるようになっている．この処理で用いるマージン表は，全国基本表(総務省)のマージン表を基に作成する(makemargintable)．新設部門があればそのマージンを組み込むことができるようになっている．

第５は，作成表の内容確認をするためのプリント機能である．部門分類表のプリント(pfst)，マージン表のプリント(pfmargintable)，変数値比較のプリント(pf,pfa)，変数値順位のプリント(pfr)，報告書作成のプリント(pfreport)がある．ただし，ここでいうプリントは，プリンタに直接印刷することではなく，テキストデータ形式での印刷イメージをファイルに出力する機能である．実際の印刷は，テキストエディタの印刷機能を用いて行うことになる．

第６は，作表の最終段階におけるバランス調整機能である．表の産出計と投入計のコントロール・トータルとの差異が小さくなった段階でのバランス調整を行う機能である．多くの労力軽減ができる．ただし，安易な使用は避けなければならない．

コマンド

◇処理制御

■実行条件式

```
 ( δ)   command
!( δ)   command
 ( @@ss )  command
!( @@ss )  command
```

δ	条件式	
	算術演算子	# * / + -
	関係演算子	> >= <= < != = ==
	論理演算子	! && ||
ss	置換記号名	
command	任意のコマンド	

[処理]
実行条件式は，それに続くコマンドの処理を実行するか否かの条件を与えるものである．すべてのコマンドに与えることができる．

条件が成立すればそれに続くコマンドは実行され，不成立ならばそのコマンドは実行されない．実行条件式が与えられていなければ，そのコマンドは無条件で実行される．

与えた実行条件式のもとで複数のコマンドを制御したいときは，その全体を 中括弧{ } でくくる．これをブロック指示と呼ぶ．また，ブロック内で別の実行条件式を与えることも，ブロック指示を与えることもできる．

以下の各コマンドの説明では，この実行条件式については触れない．

[条件式]
条件式には２種類ある．第１は定数・変数・関数および算術演算子・関係演算子・論理演算子による条件記述であり，第２は置換記号名に＠＠を付けた条件記述である．条件記述全体は丸括弧でくくる．行列，ベクトルの演算や比較も可能である．ただし，数値と文字列の比較，文字列と文字列の算術演算などはできない．

条件を否定したいときは，その手前にエクスクラメーションマーク(!)を付ける．例えば，f=5，g=7 のとき，(f==g)は条件不成立であるが，!(f==g) は条件成立である．あるいは，

(f!=g)としても同様に条件成立となる.

条件記述 (@@ss)は置換記号 ss がその時点で定義されていれば条件成立であり，定義されていなければ条件不成立である．当然，式 !(@@ss)はその反対になる．置換記号の前にアットマーク(@)を２個付けなければいけない.例えば,事前に「def　fname　sname.txt」と定義されていれば，(@@fname)は条件成立であり，!(@@fname)は条件不成立である.

算術演算子 # は行列積である．関係演算子 == と= は共に等しい，!= は等しくないである．論理演算子 ！ は否定，&& は論理積(かつ)，|| は論理和(または)である．演算子の優先順位は以下の通りである．( )でくくられた中の演算子は優先順位が同じであることを示す.

#, (*,/), (+, -), (>, >=, <=, <), (!=, =, ==), !, &&, ||

ベクトルなどの条件確認は注意が必要である．例えば，輸入係数はどの部門も１を超えてはいけない．この確認を行うには，「全部門の輸入係数は１以下である」という正当な条件を否定することによって行う．正当な条件を満たさないということは，１以下でない部門があるということになるからである．n を部門数とすれば，以下の通りである.

!(m[1:n]<=1) message ″エラーメッセージ″;

条件式を (m[1:n]>1) とすれば，全部門が１以上のとき条件成立であり，所期の目的を達成できない.

[使用例]

```
int  m,n;
 …
m=…;
n=…;
 …
(m=0 || n>0)  a=c+5;  // m の値が 0 または n の値が 0 より大きいならば,
                      // コマンド a=c+5 が実行される.
(m+n>=7){ a=d*2;      // (m+n)の値が 7 以上ならば，コマンド a=d*2: が実行され,
          e=…;        // かつ，同じ条件のもとでコマンド e=…; 実行される.
                      // ブロック指示である.
        }
 …
def fname sname.txt
 …
(@@fname) input datafile(@fname)  … ; //置換記号 fname が定義されていれば
                                      //そのファイルの読み込みを行う.
```

■飛越し(goto)

```
 …
label:
 …
goto label;
------------------------
     label    ラベル名
```

[処理]
任意のコマンドにラベル名を付け，goto コマンドでその位置に実行を移すことができる．ラベル名は，英字を先頭にした英数字で構成し，その直後にコロン(:)を付ける．英大文字はすべて自動的に英小文字に変換される．このラベル名には，置換記号を使用できない．

同一プロシージャ内での移動だけであり，プロシージャ間の移動はできない．同一プロシージャ内でもブロック外からの移動指示は，エラーにはならないが，動作が保証されないので避けたほうがよい．ブロックは実行条件式および for 文を参照のこと．goto コマンドのラベル名の指定には，置換記号を使用してもよい．

[使用例]

```
 …
 int n=0;
 …
step4:
n+=1;                // n に 1 を加算する．n++としてもよい．
 …
(n<10) goto step4; // n の値が 10 より小さいときは，step4 に実行が移る．

 …
 (n=2){ …
        s3: …;
        …          // ブロック内への移動である．
        goto s3;
      }
 …
```

■ラベル名の表示(lslabel)

```
lslabel;
```

[処理]
登録されているラベル名の表示を行う.

[使用例]

```
 lslabel;
```

■メッセージの表示(message)

```
message  msg;
------------------------
        msg    文字列
```

[処理]
文字列の表示を行う. コマンド message は msg と省略してもよい. 複数の文字列を演算子(+)で連結してもよい.

[使用例]

```
 message  "投入係数の計算開始";
 msg      "レオンチェフ"+"逆行列";
```

■処理の終了(end,quit)

```
end;
quit;
```

[処理]
プロシージャの処理を終了させる. end,quit をそれぞれ e,q と省略してもよい.

このコマンドが無くても, 処理は主プロシージャ (beginproc で始まらないプロシージャ) の最後に到達すると自動的に終了する.

[使用例]

```
q;
end;
```

■プロシージャの定義(beginproc,endproc,return)

```
beginproc  name;
…
return;
…
endproc;
--------------------------------
           name    プロシージャ名
```

[処理]
一組のコマンド群を beginproc と endproc でくくり，それに名前を付けプロシージャを定義する．これを繰り返せば複数個のプロシージャを定義することができる．

beginproc と endproc で括られていないプロシージャを主プロシージャと呼ぶ．ADAM は，主プロシージャの実行から始めるので，主プロシージャが実行プロシージャの中に必ず存在しなければならない．

beginproc は行頭コマンドとして指示しなければならない．また，endproc; の後は無視されるので，他のコマンドを与えてはいけない．

プロシージャ名は，英字を先頭にした英数字とアンダースコア(_)で構成する．英大文字はすべて自動的に英小文字に変換される．置換記号を含んではいけない．

実行は endproc に到達すると，呼び出し位置に戻る．endproc でないところで強制的に呼び出し位置に戻したいときは，そこに return; コマンドを挿入する．

プロシージャの中から別のプロシージャを呼び出すことができる．

[使用例]

```
    …                  // 主プロシージャ.
    …                  // ADAM はこのプロシージャから実行を開始する.

   end;

   beginproc  proc1;  // proc1.
    …                 // そのコマンド群.
   endproc;

   beginproc  proc2;  // proc2.
```

```
 …                    // そのコマンド群.
return;
 …
endproc;
```

■プロシージャの実行(proc)

```
proc  name;
------------------------------
      name     プロシージャ名
```

[処理]
指定したプロシージャを実行し，このコマンドの直後に戻る．プロシージャ名の英大文字はすべて自動的に英小文字に変換される．置換記号を使用してもよい．

呼び出すプロシージャは，次の順に検索され，最初に発見されたプロシージャが使用される．プロシージャフォルダに複数のフォルダが指定されていれば，指定された順に検索される．

実行中のプロシージャ
プロジェクトフォルダ
プロシージャフォルダ（複数のフォルダ指定が可能）

フォルダを検索するときのファイル名はプロシージャ名と同じで，拡張子が付いていない名，またはプロシージャ名に次の拡張子を付けたものである．検索順序もこの順番であり，最初に発見されたファイルが使用される．

拡張子なし
.adm
.adam
.txt

発見されたファイルの中に，該当するプロシージャが無いとエラーになる．プロシージャ名とファイル名の対応には十分な確認が必要である．

[使用例]

```
 …              // <<実行中のプロシージャ>>
 proc a;          // プロシージャ a は,
 …                // この実行中のプロシージャ内のものが選択され実行される.
 proc b;          // プロシージャ b は,
```

```
…              // プロジェクトフォルダのものが選択され実行される.
beginproc a;   // プロシージャ a の開始
…
endproc;
             // <<プロジェクトフォルダのファイル>>
beginproc a;   // プロシージャ a の開始.
…              // 次の何れかのファイル名で登録されている.
endproc;       // ファイル名は, "a","a.adm","a.adam","a.txt"の何れかである.

beginproc b;   // プロシージャ b の開始.
…              // 次の何れかのファイル名で登録されている.
endproc;       // ファイル名は, "b","b.adm","b.adam","b.txt"の何れかである.

…            // 呼出しに置換記号を使用した例.
proc  proc1;
…
def pr proc2
proc  @pr;
```

■繰り返し処理（for 文）

```
for(initial; decision; increment){ commands }
-------------------------------------------
     initial      初期設定式
     decision     条件式
     increment    増分式
```

[処理]

繰り返し処理を行う．先ず，具体例を示そう．

```
for( i=1; i<=10; i++){
   …      // コマンド群
   …
}
```

第１段階は，変数 i が 1 に初期化される．第２段階は，条件式 i<=10 の確認が行われる．条件が成立すれば，第３段階，中括弧でくくられたコマンド群が実行される．コマンドが１つならば中括弧は省略できる．コマンド群の実行が終了すると，第４段階は，増分式 i++ が実行される．これは，変数 i に 1 を加える演算である．第５段階は，再び条件式 i<=10 の確認が行われ，条件が成立すれば，再びコマンド群が実行される．この手続きが繰り返される．条件式が不成立になったとき for 文の処理が終了し，次のコマンドに進むことになる．

[初期設定式]
初期設定式は演算式であり，セミコロンで区切る．複数の演算式を指示したいときはその間をカンマで区切り，最後はセミコロンにする．

[条件式]
条件式はコマンド群を実行するか否かを判断する条件を与え，セミコロンで区切る．変数・定数・関数および算術演算子，関係演算子，論理演算子を用いて記述を行う．

[増分式]
増分式は演算式である．複数の演算式を与えたいときは間をカンマで区切る．演算式は後述の「◇演算・代入」に詳しい説明がある．

初期設定式および増分式は，もし不要ならば省略することができる．ただし，区切のセミコロンは省略できない．条件式は省略できない．

コマンド群の処理中，状況によっては for 文を強制終了したい，あるいは，その途中で上の第４段階の増分式に強制的に実行を移行させたいことがある．前者のために break コマンド，後者のために continue コマンドがある．具体例を示そう．

```
  for( i=1; i<=10; i++ ){
     …
     (v[i]>0) continue;
     …
     (u[i]<0) break;
     …
  }
 …     // next-command;
```

コマンド群は i の値が１から 10 まで変化して繰り返される．途中，条件 v[i]>0 が成立すると，その行以下のコマンド群の実行はスキップされ，for 文の増分式に実行が移り，条件 i<=10 が成立していれば再びコマンド群の実行が始まる．また，条件 u[i]<0 が成立すれば，for 文の処理は打ち切られ， for 文に続く次のコマンドに実行が移る．

[使用例]

```
 for( i=1; i<=nrow; i++ ){
    for( j=1; j<=ncol; j++ ){
        …
       (a[i,j]<0.001) u[i,j]= …;
       …
    }
 }
```

■実行ファイルの一括実行(submit)

```
submit execfile_1
       …
       execfile_n;
    -----------------
     execfile_x    実行ファイル名
```

[処理]
複数の実行ファイルを一括処理する．大規模な処理を複数のステップに分割し，それぞれを別々の実行ファイルで構成しているとき，作業状況の変更などにより，それらを最初からすべてを再度実行したいことがよくある．このようなときは，一括処理したいそれらの実行ファイルを実行順序にしたがって指示をする．指示された実行ファイルがその順序で次々と実行される．

この submit コマンドを持つ実行ファイルは特殊であり，submit コマンドの後は何を指示しても無視されるので注意が必要である．

[使用例]

```
submit a03pxj
       a06exp
       a07cal;
```

■変数消去(cleardata)

```
cleardata;
```

[処理]
現在処理中プロシージャの全変数名とそのデータ，置換記号名，部門分類表名，想定部門分類表名が消去され，初期状態に戻る．一括して別処理を行うときなど，メモリ制約や使用変数名に整合性がないときに用いると便利である．

[使用例]

```
  cleardata;
```

◇置換記号

■置換記号の定義(def)

```
def  name  text
----------------------------
     name       置換記号名
     text       置換テキスト
```

[処理]
name は置換記号名であり，英数字および漢字，アンダースコア(_)で構成する．先頭文字が数字でもよい．数字，英字は半角文字，全角文字すべてを使用することができ，与えた文字がそのまま使用される．英字の大文字と小文字の変換は行われない．長さ制限はない．この name 欄に置換記号名を使用することはできない．

text は置換テキストである．置換テキストは，置換記号名の後ろの空白でない最初の文字から改行記号直前までの数値や文字列である．空白やコンマ，セミコロンもその一部となる．長さに制限はない．改行記号はその一部とならない．したがって，他のコマンドと違い，最後がセミコロンである必要はなく，最後に付けたセミコロンも置換テキストの一部と認識される．

ただし，置換テキストの終わりを改行記号でなく，行途中で止めたいときは，その位置に記号 ;;（連続した 2 個のセミコロン）を置く．記号;;は改行記号とみなされるので，置換テキストはその直前までとなる．そして，記号;;の直後からは新たなコマンドの記述ができる．記号;;は def コマンドでのみ使用可能である．

このコマンドは置換記号の定義であり，この段階では，置換テキスト内に置換記号名が使用されていてもそれは置換されない．

同名の置換記号名が既に定義されていれば，何の通知もなしに，既存の置換テキストは消去され，新しい置換テキストに変更される．置換テキストの記述がなければ，ここに指定した置換記号名は削除される．

置換記号名と固定文字列をこの順序で結合するときは，その区切りにピリオドを使用する．固定文字列値と置換記号名をこの順序で結合するときは，特に区切りはいらない．多くの場合，置換テキスト内に置換記号名を使用するときは，その直後にピリオドを付けるようにした方がよい．

[使用例]

```
  def  a    @b.g@b2.m
```

```
def  b    ag
def  b2   @g5.
def  g5   r
…
@a  …;          // @a  …; は, aggrm  …; となる
```

■置換記号の定義(ndef)

```
ndef  name  text
---------------------------
```

name	置換記号名
text	置換テキスト

[処理]
コマンド ndef は，コマンド def と同様の機能を持つが，置換記号名が未定義であるときにのみ定義が行われる．反対に，同名の置換記号名がその時点で既に定義されていれば何の処理も行われない．

[使用例]

```
ndef size  120
```

■置換記号の表示(lss)

```
lss  name;
```

[処理]
name に指定した文字が，定義されている置換記号名の先頭部分と比較され，一致するものだけが表示される．name を省略すれば，すべての置換記号が表示される．

[使用例]

```
lss;       // すべてが表示される.
lss  a;    // a で始まる置換記号名が表示される.
lss  hs;   // hs で始まる置換記号名が表示される.
```

◇変数

■変数の定義(integer,doble,string)

```
integer  varname …;      整数変数
double   varname …;      浮動小数点変数
string   varname …;      文字列変数
-------------------------------------------------------------
         varname         変数名であり，次の何れかの形式をとる
           vn              スカラー
           vn[d1]          ベクトル
           vn[d1,d2]       行列
           …
           vn[d1,…,d5]   5 次元配列
```

[処理]
変数の定義には，整数 integer(または int)，浮動小数点数 double，文字列 string(または char)の３種類があり，それぞれにスカラー，ベクトル，行列(2 次元配列)から 5 次元配列まで(必要ならば，設定画面でより高次元配列の定義が可能になる)の変数を定義することができる．d1,…,d5 のそれぞれは各次元の最大要素数であり，指定には加減演算を用いることができる．要素番号はすべて１から始まる．ベクトルはすべて列ベクトルである．行ベクトルは演算時に転置関数で対応する．変数の定義は，その変数を使用する前であれば，どこで行ってもよい．

変数名は，先頭文字を英字とし 2 文字目以降を英数字とアンダースコア(_)，あるいは漢字などの全角文字で構成する．英字はすべて小文字に自動的に変換され処理されるので，大文字と小文字の違いによる変数名の識別はできない．

整数はすべて約 19 桁(64 ビット)，-9223372036854775808～9223372036854775807 の範囲の値の処理が可能である．浮動小数点数は，仮数部(有効桁数)が約 15 桁，指数部が約±308 の範囲の値である．文字列はその長さに制限はない．英数字，特殊文字，全角文字で構成する．

変数の定義の方法には本節以外に，後述する「◇作表支援コマンド，■部門分類付変数の定義」があり，使用する部門分類表を基準にその大きさが自動設定される．ベクトルと行列のみの設定ができる．この変数を使用するときは，添字に「$指示」を使用することができる．例えば，x[$0111011,$011101]のように，部門分類コードを指定して部門を特定することができる．

本節の変数定義では，各次元の大きさを直接指定しているため，部門分類表に対応していない．したがって，$指示はできないことになる．この不便さを回避するために，想定する部門分類表を定義し，変数との対応を図り指示を可能にしている．すなわち，想定部門分類表が定義されていれば，その部門分類表に基づいて$指示が行えることになる．想定する部門分類表の定義は，assumedsectortable コマンドで行う．ここで注意すべきことは，部門分類付変数は使用する部門分類表と変数の対応が厳密に管理され，部門分類表の変更・修正に対しての同期化が確保されている．しかし，本節の変数は$指示が可能というだけで，作業途中で想定する部門分類表が変更されたとき，そしてそれに伴う遡及作業がなければ，分割された実行ファイル間での get コマンドおよび put コマンドの同期化については保証されないので，使用に際しては十分な注意が必要である．

変数宣言の際は，変数値が初期化される．整数変数および浮動小数点変数は 0，文字列変数は空になる．

繰り返し呼び出されるプロシージャ内の変数宣言はそのコマンドを通過する度に初期化される．変数名，変数種別，大きさが同じであれば，再定義とみなされるのでエラーにならない．

１つのスカラー変数の定義ときは，それに続けて演算式を書くことができる．

[使用例]

```
string  rowcode[150],cdx;
int     cons[110,120],num;
def     sz  120
double  x[@sz,@sz+20],invrs[100+20,150-30],ratio;
int     nod=125, e=200, n=m*5+c;
int     nof=count(@fd);
```

■変数値の表示(lsv)

```
lsv   format(w,d) varname;
     ------------------------------------------
       format(w,d)  表示形式であり，w は表示幅，d は小数表示桁数である
                    省略したときは，format(12,0)となる
       varname      変数名である．次の何れかの形式をとる
         vn                 スカラー
         vn[subs1]          ベクトル
         vn[subs1,subs2]    行列
```

[処理]
変数の値を表示させる指示である．varname は変数名であり，スカラー，ベクトル，行列変数の指示ができる．subs1,subs2 は表示したい要素の範囲を与える．

ベクトル，行列変数のとき，要素指定[ ]を省略してもよい．このときは，行添字に 1:20 が想定され，列添字に 1:20 が想定される．

[表示形式]
値の表示桁数を変えたいときは，format(w,d)で指定する．省略したときは，w=12，d=0 となる．

[使用例]

```
lsv  s;
lsv  b[1:8];
lsv  format(12,8)  a[3:10,{21,25,18:20}];
lsv  format(18,15) pa;
```

■変数名の表示(lsvn)

```
lsvn      name;
```

[処理]
name に指定した文字が，定義されている変数名の先頭部分と比較され，一致するものだけが表示される．name を省略すれば，すべての変数名が表示される．

[使用例]

```
lsvn;       // すべてが表示される.
lsvn  a;    // a で始まる変数名が表示される.
lsvn  es;   // es で始まる変数名が表示される.
```

◇統合処理

■行列データの統合(aggrm)

```
aggrm nz | all | nzs          // 統合データの出力要素(省略時 nzs)
      cwise | rwise           // 統合データの出力順序(省略時 cwise)
      indatafile(…)           // 読み込みデータのファイル名
      infield(…)              // 読み込みデータのフィールド指定
      infieldsep(・)          // 読み込みデータのフィールド区切り記号(省略時,)
      inrcodefile(…)          // 統合行コード対応のファイル名
      inccodefile(…)          // 統合列コード対応のファイル名
      outdatafile(…)          // 書き出しデータのファイル名
      outfield(…)             // 書き出しデータのフィールド指定
      outfieldsep(・)         // 書き出しデータのフィールド区切記号(省略時,)
      outrnamefile(…)         // 書き出し行部門名称のファイル名
      outcnamefile(…)         // 書き出し列部門名称のファイル名
      maxerror(…);            // 処理を中断するエラー件数(省略時 50)
```

[処理]
行列データの統合を行う．与えられた行および列の「原データの部門コード」と「新データの部門コード」の対応に基づき，原データを統合し，新データを作成する．「読み込みデータ」は原データに関する指示であり，「書き出しデータ」は新データに関する指示である．データや統合する部門コード対応はすべてテキストエディタを用いて，テキスト形式で事前にファイル化しておく必要がある．

原データおよび新データの部門の大きさについては制限がなく，すべて自動的に処理される．

[統合データの出力要素]
統合された行列データをファイルに書き出すとき，その要素を選択する指示である．nz は 0 でない要素だけを，all は全要素を，nzs は 0 でない要素と１行目および１列目の全要素を書き出す．いずれか１つを指定する．このオプションの指定を省略すると nzs になる．

[統合データの出力順序]
統合された行列データを列方向 cwise に書き出すか，行方向 rwise に書き出すかの指定である．このオプションの指定を省略すると cwise になる．

[読み込みデータのファイル名]
読み込みデータファイルは統合に使用する原データであり，indatafile(…)の括弧の中に

そのファイル名を与える．データファイルはプロジェクトフォルダあるいはデータフォルダになければならない．フルパスの指定はできない．

[読み込みデータのフィールド区切記号]
データファイルを構成する各データをフィールドと呼ぶ．フィールドとフィールドの区切記号は半角文字とする．英数字は指定できない．infieldsep(・)を用い，括弧の中に１文字の記号を指定する．区切記号を空白にしたいときは，infieldsep()とする．このオプションの指定を省略するとコンマになる．

[読み込みデータのフィールド指定]
このフィールド指定 infield(…)は原データに対する指示である．行列データの統合には，行部門コードと列部門コードおよびその数値が必要である．この３者が読み込みデータファイル indatafile(…)において左から何番目のフィールドであるかを，「行部門コード」，「列部門コード」，「統合対象数値」の順序でそれぞれのフィールド位置を指定する．例えば，infield(3,4,7)とすれば，行部門コードが第３番目，列部門コードが第４番目，そして統合対象数値が第７番目のフィールドのデータであることを示す．

[書き出しデータのファイル名]
書き出しデータファイルは統合された新データを書き出すときのファイル名であり，outdatafile(…)で与える．フルパスの指定はできない．結果はプロジェクトフォルダに書き出される．

[書き出しデータのフィールド区切記号]
データファイルを構成する各データをフィールドと呼ぶ．フィールドとフィールドの区切記号は outfieldsep(・)で指定する．括弧の中に１文字の半角文字記号を指定する．英数字は指定できない．区切記号を空白にしたいときは，outfieldsep()とする．このオプションの指定を省略するとコンマとなる．

[書き出しデータのフィールド指定]
このフィールド指定 outfield(…)は統合データの出力に対する指示である．書き出しの対象となるデータは，「行通し番号」，「列通し番号」，「新行部門コード」，「新列部門コード」，「統合数値」の５つがある．これらを書き出しデータファイル outdatafile(…)上に何番目のフィールドとして書き出すかをこの順序でそれぞれのフィールド位置を指定する．outfield(1,2,3,4,5)とすれば，行通し番号が第１番目，列通し番号が第２番目，新行部門コードが第３番目，新列部門コードが第４番目，そして統合数値が第５番目のフィールドにそれぞれ書き出されることを示す．

outfield(…)においてフィールド位置が 0 であれば，それに対応するデータは書き出されない．0 を除いた他のフィールド位置全体は，順序は任意であるが，連続した番号になっていなければならない．例えば，outfield(0,0,2,3,1)であるとか outfield(2,3,0,0,1)であ

る．前者は，行および列の通し番号を書き出さないで，統合数値，新行部門コード，新列部門コードがこの順に書き出される．また，後者は，行および列の新部門コードを書き出さないで，統合数値，行通し番号，列通し番号がこの順に書き出される．

[統合部門コードの対応]
統合の対応は，行部門コードと列部門コードを次の形式で別々に作成し，それぞれのファイル名を inrcodefile(…)と inccodefile(…)に指定する．

新部門コードに対し１つあるいは複数の原データの部門コード(原部門コード)を等号記号（=）で区切り対応させ，セミコロンを付ける．これを繰り返す．新部門コードと原部門コードの構成は，英数字と特殊文字（ただし，コンマ，セミコロン，コロン，括弧を除く）とする．また，複数の原部門コードを指定するときは空白またはコンマで区切る．１つの原部門コードを複数の新部門コードに対応させる分割統合処理はできない．

新部門コード１ = 原部門コード１　原部門コード２　原部門コード３ …;
新部門コード２ = 原部門コード４ … ;
…

原部門コードの指定部分が範囲であるときは，開始部門コードと終了部門コードをコロンで連結し範囲指定をすることができる．範囲指定は，それを構成する部門コードが不連続でも差し支えない．範囲指定の際，終了部門コード桁数は開始部門コードの桁数を超えてはいけない．終了部門コード桁数が開始部門コードより短いときは，その上部に不足桁数分の部門コードが開始部門コードより補われる．文字の大小関係は，Unicode にしたがう．その順序は特殊文字や漢字など全角文字は複雑であるが，英数字については，半角数字，全角数字，半角英小文字，全角英小文字，半角英大文字，全角英大文字の順である．

N010=01011:01099 …;
N020=A02011:A02325 …;
N030=A03011:25 …; //これは，N030=A03011:A03025 …; と同じである．

原部門コードに指定する部門コードの桁数は，原データがファイル上で持つ部門コード桁数である必要はない．統合目的に応じて矛盾なく識別と特定が可能ならば，１桁から原部門コードの桁数までを自由に指定できる．このとき当然であるが，指定桁数の長い部門コードが優先的に処理される．例えば，

N1=01;
…
N3=012301;
…

とすれば，先頭２桁が 01 の部門コードを持つ部門の大部分は新部門コード N1 に統合されるが，012301 だけは新部門コード N3 に統合される．

原部門コードおよび新部門コードに「対応なし」の null または NULL を指定することがで

きる.

null=原部門コード…;

は指定した原部門コードは無視され，統合の対象にならない．また，

新部門コード=null;

は対応する原データは無く，新部門コード部門だけを登録することになる.

[新部門コードの名称]

新部門コードに対し，その部門名称を新部門コードの左側に与えることができる．新部門名称と新部門コードの区切は空白またはコンマとする．部門名称に特殊文字（コンマ，セミコロン，コロン，括弧など）が含まれているときは，必ずその部門名称全体を引用符でくくらなければならない．特殊文字が含まれていない部門名称を引用符でくくっても差し支えない．新部門名称は演算子(+)を用いて，連結指定をすることができる.

新部門名称 1　　新部門コード 1 = 原部門コード 1　原部門コード 2 …;

″新部門名称″+″2″,　新部門コード 2　= 原部門コード 3 … ;

…

部門名称は，統合部門コードの部門分類が明らかになり，間違いが減り，作業効率を高める効果があるだけでなく，ここに与えた部門名称はファイルに書き出すことができ，そのファイルは後の作業で利用することができる．このため，部門名称は常に与えた方がよい.

行部門名称の書き出しファイル名を outrnamefile(…)に指定すれば，そこに新行部門コードと新行部門名称が書き出される．このオプションの指示を省略すれば，書き出しは行われない.

同様に，outcnamefile(…)にファイル名を指定すれば，そこに新列部門コードと新列部門名称が書き出される．このオプションの指示を省略すれば，書き出しは行われない.

[エラーによる処理の中断]

エラー総数が maxerror を超えると，処理が中断される．省略時は 50 である.

[使用例]

```
aggrm  indatafile(jio.dat)
       infield(3,4,5)
       inrcodefile(agrc4.txt)
       inccodefile(agcc4.txt)
       outdatafile(jio4.dat)
       outfield(1,2,3,4,5);
```

■ベクトルデータの統合(aggrv)

```
aggrv nz | all | nzs          // 統合データの出力形態(省略時 all)
      indatafile(…)           // 読み込みデータファイル名
      infield(…)              // 読み込みデータのフィールド指定
      infieldsep(・)          // 読み込みデータのフィールド区切記号(省略時,)
      incodefile(…)           // 統合対応部門コードファイル名
      outdatafile(…)          // 書き出しデータファイル名
      outfield(…)             // 書き出しデータのフィールド指定
      outfieldsep(・)         // 書き出しデータのフィールド区切記号(省略時,)
      outnamefile(…)          // 部門名称の書き出しファイル名
      maxerror(…);            // 処理を中断するエラー件数(無指定ときは 50)
```

[処理]
与えられた「原データの部門コード」と「新データの部門コード」の対応に基づき，原データを統合し新データを作成する．「読み込みデータ」は原データに関する指示であり，「書き出しデータ」は新データに関する指示である．データや統合部門コード対応のすべてはテキスト形式で事前にファイル化しておく必要がある．

処理内容は「行列データの統合コマンド aggrm」と基本的に同じである．以下，相違点だけ述べる．ここに無い事項の説明はコマンド aggrm を参照のこと．

[統合部門コードの対応]
ベクトルデータの統合では，統合部門コード対応が一組で済むことから，aggrm コマンド時の inrcodefile(…)と inccodefile(…)に代えてその部分が incodefile(…)となっている．

[読み込みデータのフィールド指定]
このフィールド指定 infield(…)は，原データに対する指示である．ベクトルデータでは，統合には部門コードと数値が必要である．この２者が読み込みデータファイル indatafile において，左から何番目のフィールドであるかを，「部門コード」，「統合対象数値」の順序でそれぞれのフィールド位置を指定する．infield(4,7)とすれば，部門コードが第４番目，統合対象数値が第７番目のフィールドのデータであることを示す．

[書き出しデータのフィールド指定]
書き出しデータのフィールド指定 outfield(…)は，ベクトルデータでは，「通し番号」，「新部門コード」,「統合数値」の３つがある．これらを書き出しデータファイル outdatafile 上に何番目のフィールドとしてに書き出すかをこの順序でそれぞれのフィールド位置を指定する．outdatafile(1,2,3)とすれば，通し番号が第１番目，新部門コードが第２番目，統合数値が第３番目のフィールドのデータとすることを示す．

outfield(…)において，該当箇所が０であれば，対応する内容は書き出されない．０を除いた他の番号全体は，順序は任意であるが，連続した番号になっていなければならない．outfield(0,2,1)であるとか outfield(2,0,1)である．前者は通し番号を書き出さないで，統合数値，新部門コードがこの順に書き出され，後者は，新部門コードを書き出さないで，統合数値，通し番号がこの順に書き出される．

[新部門コードの名称]
新部門コードに対し，その部門名称を新部門コードの左側に与えることができる．ベクトルデータでは，新部門コードと部門名称ファイルの書き出しは１つである．上の統合部門コード対応と同様に，aggrm コマンドの outrnamefile(…)と outcnamefile(…)に代えて，その部分が outnamefile(…)となっている．このオプションの指示が無ければ，書き出しは行われない．

[使用例]

```
aggrv  all
       indatafile(lab2.dat)
       infield(2,3)
       incodefile(labcode.txt)
       outdatafile(lab4.dat)
       outfield(1,2,3);
```

◇入力処理

■標準入力(input)

```
input  datafile(…)  fieldsep(・)
       #1(opt1)  #2(opt2) …
       (fn1,vn1[#1]) (fn2,vn2[#2]) (fn3,vn3[#1,#2]) …;
-------------------------------------------------------
       #1,#2,…          添字変数
       fn1,fn2,…        フィールド位置
       vn1,vn2,…        変数名
       (opt1),(opt2),…  添字の指定，次の形式がある
          (fn)
          (fn,index(codevec))
          (fn,vn,ascend)
          (fn,vn,decend)
          (fn,vn,unchange)
```

[処理]
指定したADAMの標準データ構造を持つファイルからデータの読み込みを行う．

[ADAMの標準データ構造]
ADAMのデータ構造は，１レコード（１行のデータ）内に目的とするデータとその位置を示す添字部分（番号あるいはコード）を一体として収めた形式である．それらの各データをフィールドと呼ぶ．したがって，データの読み込みは，添字となるフィールド位置，読み込むデータのフィールド位置，およびそれらを格納する変数名の指示を行うことになる．

[読み込みデータのファイル名]
読み込みデータファイルは，datafile(…)で括弧の中にファイル名を与える．

[読み込みデータのフィールド区切記号]
読み込みデータのフィールド区切記号は，fieldsep(・)で指定する．括弧の中に１文字の半角文字記号を指定する．英数字は指定できない．区切記号を空白にしたいときは，fieldsep()とする．このオプションの指定を省略するとコンマになる．

[添字変数]
添字変数 #k　k=1,2,…,99　はレコード上に記憶されている情報で決定する．大きく３つの形式がある．これに加えて，特殊な添字変数に#0がある．添字変数番号の使用順序に制限はない．

第１の形式は，#k(fn) であり，レコード上に添字番号のフィールドが存在しているとき，そのフィールド位置 fn を指定する．例えば，#1(2)であれば，第２フィールドの値が第１添字変数値になる．

第 2 の形式は，#k(fn, index(codevec))である．フィールド位置 fn から部門コードを読み込み，事前に読み込みまれている部門コードの序列を記憶したベクトル変数 codevec に基づいて添字の位置が決まる．#2(3, index(cdv))であれば，第 3 フィールドの部門コードを変数 cdv 内の部門コード序列で第２添字変数の位置が決定される．

第 3 の形式は,#k(fn, vn, ascend)であり,ascend(昇順)の部分を,decend(降順),unchange(無変更)と指示することができる．これらを，それぞれ a, d, u としてもよい．フィールド位置 fn から部門コードのすべてをベクトル変数 vn に読み込み，重複を排除し，ascend ならば昇順に, decend ならば降順に並べ替える. 無変更ならば並べ替えはしない. 得られた変数 vn の序列に基づいて添字の位置が決まる．また，#k(fn, vn)は#k(fn, vn, ascend)と解釈される．たとえば，#3(2, cdv, ascend)であれば，先ず, 第 2 フィールドの部門コードを変数 cdv に重複を排除して読み込み，それを昇順に並べ替える．その上で，各レコードの第 2 フィールドの部門コードで変数 cdv 内を検索し，第３添字の位置が決まることになる．

添字変数 #0 は読み込みデータの順序を保持する添字変数であり，何の宣言も必要なく(宣言を行うことはできない)，いつでも自由に使用することができる．ファイルに記録された順序通りに，データをベクトル変数に読み込むことを目的としている．任意の部門コード配列を与えるときなどに便利である．

[読み込み変数の指示]
変数へのデータの読み込みは，(fn, vn[…])の形式で行う．fn はフィールド位置であり，vn は値を格納する変数名である．変数 vn[…]の添字は，それぞれの変数の読み込み位置を特定するものであり，その変数が持つ次元数だけ必要となる．その変数がベクトルであれば１つの添字変数だけを与え, 行列であれば２つの添字変数を与える．多次元配列であればその次元数だけの添字変数を与えることになる．各添字は上の添字変数 #k ，あるいは固定値で与える．添字変数 #k は，上述のいずれかの方法で指定されていなければならない．固定値は，整数値，スカラーおよび配列の整数変数，index()関数，$指示，およびそれらの加減算で与える．(5, xx[#1, #2])は，第５フィールドの値を行列 xx に，#1 を行添字，#2 を列添字として読み込む．(5, xx[#1, index(cc, 0111)])は，第５フィールドの値を行列 xx に，#1 を行添字とし，列コード 0111 (部門コード序列 cc に基づいて決まる)の位置に読み込む．

一度に指定できる読み込み変数は，添字変数も含めて最大 30 個である．

[使用例]

```
   string   rcode[100],ccode[100];
```

```
   double   x[100,100];
   input    datafile(jtable1.dat) #1(1) #2(2)
            (3,rcode[#1]) (4,ccode[#2]) (5,x[#1,#2]);

   string   rcode[100],ccode[100];
   double   x[100,100];
   input    datafile(jtable2.dat)
            #11(1,rcode,ascend) #22(2,ccode,ascend) (3,x[#11,#22]);

   string   rcode[100],ccode[100];
   double   x[100,100];
   input    datafile(rowcode.txt)  #1(1) (2,rcode[#1])
   input    datafile(colcode.txt)  #1(1) (2,ccode[#1])
   input    datafile(jtable3.dat)
            #1(1,index(rcode)) #2(2,index(ccode)) (3,x[#1,#2]);

   sectortable k101st ;
   double   x(k101st);  //「部門分類付変数の定義」参照
   input    datafile(jts1.dat)
            #1(1,index(_rc_k101st)) (2,x[#1,$911000]) (3,x[#1,$913110]);

   assumedsectortable k101st ;
   double   x(k1010st);
   input    datafile(jts2.dat)
            #1(1,index(_cc_k101st)) (2,x[$9311000,#1]) (3,x[$9404000,#1]);

string   rcode[100],rname[100];
   input    datafile(codefile.dat)  (1,rcode[#0]) (2,rname[#0]);

   string   rcode[100],ccode[100];
   double   x[100,100];
   input    datafile(jtable4.dat)
            #1(1,index(rcode)) (3,x[#1,index(ccode,0111)]);
```

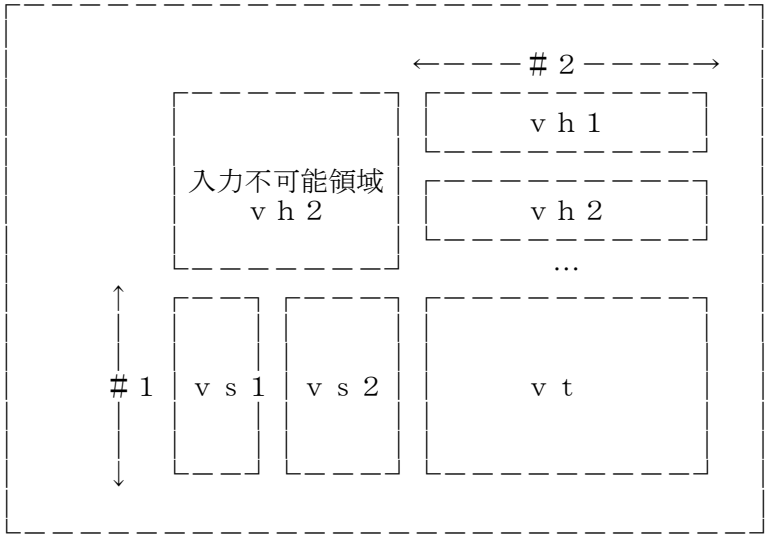
■シートデータ入力(insheet)

```
  insheet  datafile(…)  fieldsep(・)  #1(subs1)  #2(subs2)
         (vh1[#2]) (vh2[#2]) … (vs1[#1]) (vs2[#1]) … (vt[#1,#2]);
  ------------------------------------------------------------
           datafile(…)     読み込みデータのファイル名
```

#1	行添字変数
#2	列添字変数
subs1,subs2	添字指定
vh1,vh2,…	表頭ベクトル変数名
vs1,vs2,…	表側ベクトル変数名
vt	行列変数名

[処理]

シートデータ入力は，スプレッドシート（表計算ソフト）との連携のためのコマンドである．シートデータの入力は次図に示す配置を想定している．



[読み込みデータのファイル名]

読み込みデータファイルは，datafile(…)で括弧の中にファイル名を与える．

[読み込みデータのフィールド区切記号]

読み込みデータのフィールド区切記号は，fieldsep(・)で指定する．括弧の中に１文字の半角文字記号を指定する．英数字は指定できない．区切記号を空白にしたいときは，fieldsep()とする．このオプションの指定を省略するとコンマになる．

[添字変数]

添字変数 #1 は行添字であり，#2 は列添字である．読み込んだ値を変数の添字位置に格納する．読み込みできるシートの最大列数は ADAM の設定画面「シート入出力最大列数」である．たとえば，#1(1:57) #2(1:63)のように指示をする．添字の指示には index()関数，整数変数を使用することもできる．

assumedsectortable コマンドで部門分類表名が与えられていれば，$指示による部門コードを使用することもできる．このとき，#1 は部門分類表の行コードが，#2 は部門分類表の列コードがそれぞれ検索の対象となる．

[読み込み変数の指示]
変数値の読み込みは，(vn[…])の形式で行う．vn は値を格納する変数名である．#1,#2 は，それぞれの変数の読み込み位置を特定する添字であり，変数がベクトルであれば１つの添字だけを与え,行列であれば２つの添字を与える．

添字変数 #2 を付された変数 vh1,vh2,… は表頭変数であり，ベクトルである．列の部門コードや部門名称などであることが多い．読み込みは指定した変数の順序となる．

添字変数 #1 を付された変数 vs1,vs2,…は表側変数であり，ベクトルである．行の部門コードや部門名称などであることが多い．読み込みは指定した変数の順序となる．

添字変数 #1 と#2 を付された変数 vt は，行列データである．指示できる変数は１つだけである．

このコマンドで読み込んだ行コード・列コードに基づいて，その行列データ値を部門分類付変数の該当する行部門・列部門に移す有用な move コマンドがある．

[使用例]

```
def    sz  100
string h1[@sz],h2[@sz],s1[@sz],s2[@sz];
double x[@sz,@sz];
…
insheet  datafile(ctable.csv) #1(1:87),#2(1:83)
        (h1[#2]),(h2[#2]),(s1[#1]),(s2[#1]),(x[#1,#2]);

…
assumedtsectortable k101st;
insheet datafile(d108fd1.csv) fieldsep(,)
  #1({1:$8513013}) #2({1:50,$970000}) (rowcode[#1]) (rowname[#1])
                    (colcode[#2]) (colname[#2])  (t[#1,#2]);
```

◇出力処理

■標準出力(output)

```
output datafile(…)  fieldsep(・)
        #1(fn1,subs1)  #2(fn2,subs2) …
        (fn3,vn1[#1],fmt1) (fn4,vn2[#2],fmt2)
        (fn5,vn3[#1,#2],fmt3) …;
-------------------------------------------
        #1,#2,…          添字変数
        subs1,subs2,…    添字指定
        fn1,fn2,…        フィールド位置
        vn1,vn2,…        変数名
        fmt1,fmt2,…      フォーマット指定
```

[処理]
指定したファイルに，ADAM の標準データ構造に準拠して，データの書き出しを行う．

対象となる数値変数(整数および浮動小数点数)のすべての値が０のときは書き出されない．なお，書き出し対象に数値変数が含まれないときは，すべてが書き出される．

[ADAM の標準データ構造]
ADAM のデータ構造は，１レコード(１行のデータ)内に目的とするデータとその位置示す添字部分（番号あるいはコード）を一体として収めた形式である．したがって，データの書き出しは，添字部分と変数値，そしてそれらの書き出しフィールド位置を指示することになる．

[書き出しデータのファイル名]
書き出しデータファイルは，datafile(…)で括弧の中にファイル名を与える．ファイルはプロジェクトフォルダに書き出される．他のフォルダに書き出すことはできない．

[書き出しデータのフィールド区切記号]
書き出しデータのフィールド区切記号は，fieldsep(・)で指定する．括弧の中に１文字の半角文字記号を指定する．英数字は指定できない．区切記号を空白にしたいときは，fieldsep()とする．このオプションの指定を省略するとコンマになる．

[添字変数]
添字変数 #k k=1,2,…,99 はファイルに書き出す添字の範囲を特定する．#1(1,1:100)とすれば，添字変数 #1 は，1 から 100 まで変化し，かつその値をファイルレコードの第１フィ

ールドに書き出す．#1(,1:100)とすれば，添字変数の範囲は同じであるが，その値はファイルに書き出されない．#1(1,{1:100,102,105:107})とすれば，添字の変化範囲は 1～100，102，105～107 となり，その値は第１フィールドに書き出される．

[書き出し変数の指示]
添字変数を複数与えたときは，先に指示した添字変数が先に変化する．したがって，
#1(1,1:10) #2(2,1:7) (3,xx[#1,#2]) とすれば，変数 xx の値は列方向に書き出される．また，#2(2,1:7) #1(1,1:10) (3,xx[#1,#2]) とすれば，変数 xx の値は行方向に書き出されることになる．

変数値の書き出しは，(fn,vn[…],fmt)の形式で行う．fn はフィールド位置，vn は書き出す値の変数名，fmt はフォーマットである．フォーマットは変数の種類によって与え方が違う．変数 vn[…]の添字は，それぞれの変数の書き出す位置を特定するものであり，その変数が持つ次元数だけ必要となる．その変数がベクトルであれば１つの添字変数だけを与え，行列であれば２つの添字変数を与える．多次元配列であればその次元数だけの添字変数を与えることになる．各添字は上の添字変数 #k ，あるいは固定値で与える．添字変数 #k は，上述の方法で指定されていなければならない．固定値は，整数値，スカラーおよび配列の整数変数，index()関数，$指示，およびそれらの加減算で与える．(5,xx[#1,#2])は，行列 xx の行添字 #1，列添字 #2 の値を第５フィールドに書き出す．どちらの添字が先に変化するかは，上述の通りである．(6,xx[#1,index(rc,0111)])は，行列 xx の行添字 #1，列コード 0111 (部門コード序列 rc に基づいて決まる)の固定位置の値を第６フィールドに書き出す．

変数が整数のときのフォーマットは，書き出し最小桁数であり，1～19 桁の範囲で指定できる．省略時は１桁となる．(2,g[#1],3)で変数値が 37 であれば，書き出しは 037 となる． また，(2,g[#1])で変数値が 7 であれば，書き出しは 7 となる．

変数が浮動小数点数のときのフォーマットは，小数の書き出し最大桁数であり，0～29 桁の範囲で指定できる．省略したときは 12 桁となる．小数が 0 であれば，小数点も書き出されない．

変数が文字列のときのフォーマットは，書き出し最大文字数であり，1～999 の範囲で指定できる．省略したときは 120 字となる．全角文字も半角文字も１文字は１字である．

フィールド位置は，指定順序は任意であるが，全体は連続した番号になっていなければならない．一度に指定できる書き出し変数は，添字変数も含めて最大 30 個である．

[使用例]

```
string  rcode[100],ccode[100];
double  x[100,100];
```

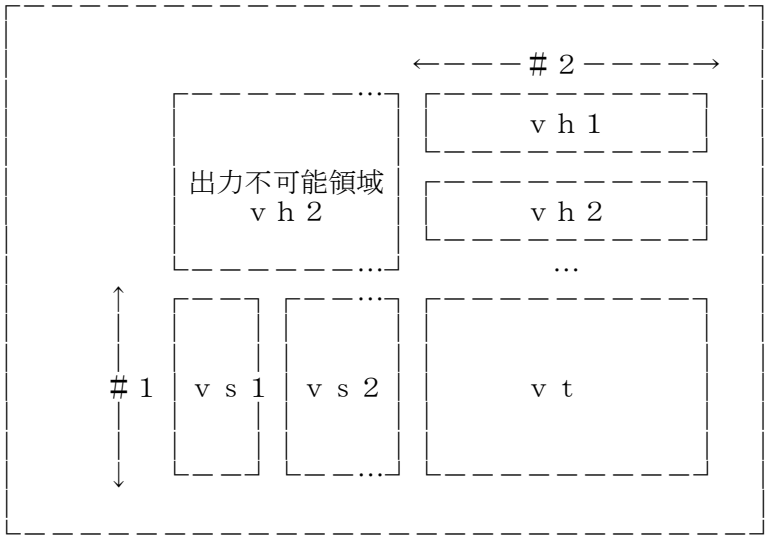
```
output  datafile(jtable6.dat) #11(1,1:85) #22(2,1:92)
        (3,rcode[#11]) (4,ccode[#22]) (5,x[#11,#22]);

string  rcode[100],ccode[100];
float   x[100,100];
output  datafile(jtable7.dat) #1(,1:85) #2(,1:92)
        (1,rcode[#1]) (2,ccode[#2]) (3,x[#1,#2]);

string  code[100],name[100];
output  datafile(jtable8.dat)  #1(1,1:85) (2,code[#1])

string  code[100],name[100];
output  datafile(jtable9.dat)  #1(,1:85) (1,code[#1]) (2,name[#1]);



string  rcode[100],ccode[100];
double  x[100,100];
output  datafile(jtable10.dat) #1(2,1:85)
        (1,rcode[#1]) (3,x[#1,index(ccode,0111)]);

sectortable k101st;
assumedsectortable k101st;
double x(k101st);
output datafile(k107.csv) fieldsep(,)
        #7(1,31:100) (2,_rc_k101st[#7])(3,x[#7,$011101]) (4,x[#7,$970000]);
```

■シートデータ出力(outsheet)

```
outsheet  datafile(…)  fieldsep(・)  #1(subs1)  #2(subs2)
       (vh1[#2],fmt) (vh2[#2],fmt) …
       (vs1[#1],fmt) (vs2[#1],fmt) … (vt[#1,#2],fmt);
```

#1	行添字変数
#2	列添字変数
subs1,subs2	添字指定
vh1,vh2,…	表頭ベクトル変数名
vs1,vs2,…	表側ベクトル変数名
vt	行列変数名

[処理]
シートデータ出力は，スプレッドシート（表計算ソフト）との連携のためのコマンドである．シートデータ出力は次図に示す配置を想定して行う．



[書き出しデータのファイル名]
書き出しデータファイルは，datafile(…)で括弧の中にファイル名を与える．

[書き出しデータのフィールド区切記号]
書き出しデータのフィールド区切記号は，fieldsep(・)で指定する．括弧の中に１文字の半角文字記号を指定する．英数字の指定はできない．区切記号を空白にしたいときは，fieldsep()とする．このオプションの指定を省略するとコンマになる．

[添字変数]
添字変数 #1 は行添字であり，#2 は列添字である．書き出す変数の値を添字変数で特定する．#1(1:57) #2(1:63)のように指示する．書き出しできるシートの最大列数は ADAM の設定画面「シート入出力最大列数」である．添字の指示には index()関数，整数変数を使用することもできる．

assumedsectortable コマンドで部門分類表名が与えられていれば，$指示を使用することもできる．このとき，#1 は部門分類表の行コードが，#2 は部門分類表の列コードがそれぞれ検索の対象となる．

[書き出し変数の指示]
添字変数 #2 を付された変数 vh1,vh2,… は表頭変数であり，ベクトルである．列の部門コードや部門名称などであることが多い．書き出しは指定した変数の順序となる．複数の変数を指定することができる．

添字変数 #1 を付された変数 vs1,vs2,…は表側変数であり，ベクトルである．行の部門コ

ードや部門名称などであることが多い．書き出しは指定した変数の順序となる．複数の変数を指定することができる．

添字変数 #1 と#2 を付された変数変数 vt は，行列データである．行列変数指定は１つだけである．

変数値の書き出しは，(vn[…],fmt)の形式で行う．vn は書き出す値の変数名，fmt はフォーマットである．フォーマットは変数の種類によって与え方が違う．vn[…]の…部分は，それぞれの変数の書き出し位置を特定する添字であり，変数がベクトルであれば１つの添字だけを与え,行列であれば２つの添字を与える．

変数が整数のときのフォーマットは，書き出し最小桁数であり，1～19 桁の範囲で指定できる．省略時は１桁となる．(2,g[#1],3)で変数値が 37 であれば，書き出しは 037 となる． また，(2,g[#1])で変数値が 7 であれば，書き出しは 7 となる．

変数が浮動小数点数のときのフォーマットは，小数の書き出し最大桁数であり，0～29 桁の範囲で指定できる．省略したときは 12 桁となる．小数が 0 であれば，小数点も書き出されない．

変数が文字列のときのフォーマットは，書き出し最大桁数であり，半角文字換算で，1～999 桁の範囲で指定できる.省略したときは 120 桁となる.全角文字も半角文字も１文字は１ 字である．

[使用例]

```
def    sz  100
string h1[@sz],h2[@sz],s1[@sz],s2[@sz];
double x[@sz,@sz];
…
outsheet  datafile(ctable.csv) #1(1:87),#2(1:83)
          (h1[#2]),(h2[#2]),(s1[#1]),(s2[#1]),(x[#1,#2]);

sectortable k101st;
…
defaultsectortable k101st;
outsheet datafile(t108fd1.csv) fieldsep(,)
      #1(1:$8513013) #2({1:50,$970000}) (rowcode[#1]) (rowname[#1])
                (colcode[#2]) (colname[#2]) (t[#1,#2]);
```

◇演算・代入

■演算

```
α  aop  β   op   γ   … ;
-------------------------------------------------------
        α           変数
        β  γ …      変数・定数
        aop         代入演算子      =  +=  -=  *=  /=
        op  …       算術演算子      +  -  *  /  #  ++  --
```

[処理]
右辺の演算が行われ，その結果が左辺の変数に代入される．右辺には複数の変数と定数の演算を指示できる．演算が可能な変数は，スカラー，ベクトル，行列(２次元配列)だけである．３次元以上の配列は代入処理（後述）のみが可能であり，演算はできない．右辺の変数の値は演算・代入後も変わらない．ただし，右辺の変数でもインクリメント・デクリメント演算子(++,--)を持つ変数はその値が変わる．for 文，宣言文などで，複数の演算式を１コマンドで与えたいときは，それらをカンマで区切ることができる．

演算可能な変数・定数には整数(有効桁数約 19 桁)と浮動小数点数(有効桁数約 15 桁，指数部が約±302)がある．右辺に整数と浮動小数点数が混在するときは，整数が浮動小数点数に変換され演算が行われる．

[代入演算子]
代入演算子には次のものがある．左辺と右辺の変数種別が整数あるいは浮動小数点数で，両者が違うときは自動的に変換が行われる．数値と文字列の変換は行われない．

```
=  +=  -=  *=  /=
```

a=b; は b の値を a の値とする単純な代入である．a+=b; は a=a+b; と同義であり，a*=b+c; は a=a*(b+c); と同義である．他も同様である．

[算術演算子]
算術演算子には次のものがある．

```
#    行列積                    (演算優先順位が高い)
*    乗算      /    除算              ↑
+    加算      -    減算      (演算優先順位が低い)
```

[除算の処理規則]
除算は，次のように処理される．この処理規則は後述の分配関数演算にも適用される．

- □ 分子分母が共に 0 でなければ除算が行われる．
- □ 分子が 0 ならば分母の値の如何を問わず，商は 0 となる．

□ 分子が 0 でなく分母が 0 ならば商は 0 となり，エラーが通知される．

[単項演算子]
単項演算子には次のものがある．演算優先順位は行列積より高い．

+　－　符号:単項演算子　　(演算優先順位が高い)

[インクリメント，デクリメント演算子]
インクリメント，デクリメント演算子には次のものがある．演算子は変数の後に付ける．

++　インクリメント　　--　デクリメント

この演算は，右辺の演算がすべて終了し，その値が左変数に代入された後で行われる．インクリメントは変数への１加算であり，デクリメントは１減算である．例えば，k=f+h++; は，f の値が 5 で，h の値が 3 ならば，k の値は 8，h の値は 4 になり，f の値は変わらない．代入演算子が無い演算式も指示可能である．例えば，i++; のときは，i の値が１加算され，i--; のときは，i の値が１減算される．for 文の増分式でよく用いられる．

[演算の順位]
演算は，算術演算子の優先順位の高い部分から処理される．また，丸括弧 ( )の使用により，演算順序は自由に変えることができる．

[演算・代入の要素対応]
演算・代入を行う要素の対応方式は ADAM の特徴の一つである．演算を行う変数の要素の対応は，その都度その変数間で決められる．その対応は次のように行われる．

□ 行と列は独立であり，行についても列についても与えた添字の順序で対応が行われ，添字値の大きさなどとは無関係である．
□ 当該変数に与えた添字の要素数がそれぞれ n 個あるときは，与えた順序でそれぞれが対応づけられる．
□ 一方の変数に与えた添字の要素数が n 個で，他方の変数に与えた添字の要素数が１つのときは，前者の n 個のすべての要素が後者の１つの要素に対応づけられる．
□ スカラー変数と定数の行および列の要素数は共に１と認識される．
□ ベクトル変数の指定した要素数は行の要素数であり，列の要素数は１と認識される．ただし，その変数を転置すれば逆になる．

[整数と整数の除算の注意]
一般のコンピュータ言語と同様に，整数と整数の除算の商は整数になる． k,m が共に整数変数であり，その値が k=7,m=3 であれば k/m の値は 2 となる．また，その値が k=7,m=8 であれば k/m の値は 0 となる．

［関数］

関数には次のものがある.

□abs(α)　絶対値

行列，ベクトル，スカラーαの絶対値を求める.

結果は同じ大きさの行列，ベクトル，スカラーである.

□balance(α,β,γ)　バランス調整

初期行列α,列ベクトルβ(制約条件行和,産出額),行ベクトルγ(制約条件列和,投入額)のもとで縦横(行列)のバランス調整を解析的に行う関数である．β,γ が整数変数のときは，丸めの誤差による端数調整処理が自動的に行われる.

結果は同じ大きさの行列である．詳しい使用法は，「◇作表支援コマンド，■バランス調整」を参照のこと.

□count(σ)　添字要素数

添字列σの要素数をカウントする.

結果はスカラーである.

添字指定の中での使用はできない.

□diag(β)　対角行列

ベクトルβを対角行列とする.

結果は正方行列である.

□distr(α,β)　行ベクトル分配

行列αの各行の構成割合で列ベクトルβの各要素値を分割する.

βが整数のときは，丸めの誤差による端数調整処理が,

最大構成割合を持つ要素位置で自動的に行われる.

結果は同じ大きさの行列である.

□distr(α,γ)　列ベクトル分配

行列αの各列の構成割合で行ベクトルγの各要素値を分割する.

βが整数のときは，丸めの誤差による端数調整処理が,

最大構成割合を持つ要素位置で自動的に行われる.

結果は同じ大きさの行列である.

□distr(α,s)　行列分配

行列αの構成割合でスカラーsの値を分割する.

sが整数のときは，丸めの誤差による端数調整処理が,

最大構成割合を持つ要素位置で自動的に行われる.

結果は同じ大きさの行列である.

□exp(α)　指数

行列，ベクトル，スカラーαの値の指数を求める．

結果は同じ大きさの行列，ベクトル，スカラーである．

□get diag(α)　対角要素

行列αの対角要素を取出す．

結果は列ベクトルとなる．

□index(β,code)　検索

文字列ベクトルβの中から文字列 code を検索し，その位置を求める．

結果はスカラーである．

□index(β,varname)　検索

文字列ベクトルβの中から変数 varname が持つ文字列を検索し，その位置を求める．

結果はスカラーである．

□int5(α)　四捨五入

行列，ベクトル，スカラーαの値の小数第１位を四捨五入する．

結果は同じ大きさの行列，ベクトル，スカラーである．

□int9(α)　切上げ

行列，ベクトル，スカラーαの値の小数第１位を切り上げる．

結果は同じ大きさの行列，ベクトル，スカラーである．

□inverse(α)　逆行列

行列αの逆行列を求める．

結果は同じ大きさの正方行列である．

inv(α)としてもよい．

□log(α)　自然対数

行列，ベクトル，スカラーαの値の自然対数を求める．

結果は同じ大きさの行列，ベクトル，スカラーである．

ln(α)としてもよい．

□log10(α)　常用対数

行列，ベクトル，スカラーαの値の常用対数を求める．

結果は同じ大きさの行列，ベクトル，スカラーである．

□nz(α)　負値零

行列，ベクトル，スカラーαの値が負ならば 0 とする．
結果は同じ大きさの行列，ベクトル，スカラーである．

□pow(α，γ)　べき乗
α の γ 乗を求める．α および γ は行列，ベクトル，スカラーである．
演算の要素対応は前述の「演算の要素対応」にしたがう．
結果は同じ大きさの行列，ベクトル，スカラーである．

□pz(α)　正値零
行列，ベクトル，スカラーαの値が正ならば 0 とする．
結果は同じ大きさの行列，ベクトル，スカラーである．

□sqrt(α)　平方根
行列，ベクトル，スカラーαの値の平方根を求める．
結果は同じ大きさの行列，ベクトル，スカラーである．

□sum(α)　行列和
行列あるいはベクトルαの指定部分の集計値を求める．
結果はスカラーとなる．

□sum1(α)　列和
行列αの指定部分の各列の集計値を求める．
結果は行ベクトルとなる．
sum1 の"1"は第１添字を集計する意である．

□sum2(α)　行和
行列αの指定部分の各行の集計値を求める．
結果は列ベクトルとなる．
sum2 の"2"は第２添字を集計する意である．

□trans(α)　転置行列
行列あるいはベクトルαを転置する．

□unit(n)　単位行列
大きさn(スカラー)の単位行列である．
結果は(n×n)の正方行列である．

[使用例]

```
def     sz  20
def     s   1:@sz
```

```
def     m   1:14
def     n   14
def     ct  18
double  x[@sz,@sz],a[@sz,@sz],b[@sz,@sz],z[@sz,@sz],u[@sz];
int     g[@sz,@sz];
u[@m]=1.0;
a[@m,@m]=x[@m,@m]/x[@ct,@m];
b[@m,@m]=inverse(diag(u[@m])-a[@m,@m]);
b[@m,@m]=inverse(unit(@n)-x[@m,@m]/x[@ct,@m]); /* 上の２行に同じ */
g[@m,1:5]=b[@m,@m]#z[@m,1:5];

int     v[15],a[10,15];
double  b[10,15];
a[1:10,1:15]=distr(b[1:10,1:15],v[1:10]);         // 行分配
a[1:10,1:15]=distr(b[1:10,1:15],trans(v[1:15])); // 列分配
```

■代入

```
α = β;
α = {c1 c2 …};
----------------------
     α   β       変数
    c1   c2      定数
```

[処理]
左辺と右辺の変数種別は一致していなければならない．ただし，整数と浮動小数点数については自動的に変換が行われる．

多次元変数の代入は，右辺と左辺の指定した要素数が一致する次元が対応し，代入処理が行われる．同じ要素数を持つ次元が複数あるときは，指示順序で対応する．また，左辺および右辺の次元数は一致していなくてよいが，対応しない左辺および右辺の次元の要素数は１でなければならない．

左辺が数値の配列変数であるとき，右辺に複数個の数値を並べ，それらの値を順次代入することができる．このとき，それら全体は中括弧｛｝でくくらなければならない．その対応は，配列変数の左側の次元が先に変化し，右側の次元が後から変化する．例えば，
mtx[1:3,1:2]={11,21,31,12,22,32,…}; は，mtx[1,1]=11; mtx[2,1]=21; mtx[3,1]=31; mtx[1,2]=12; ...となる．左辺と右辺の指示した要素数は一致していなければならない．ただし，右辺の要素数が１つのときは，左辺に指定した全要素がその値になる．

左辺が文字列の配列変数であるとき，右辺に複数個の文字列を並べたときも同様に処理される．ただし，それら全体は中括弧｛｝でくくらなければならない．右辺の文字列がコンマや空白など特殊文字を含むときは，それを引用符でくくる必要がある．

［使用例］

```
def  sz  20
def  sb  1:@sz
string s[@sz,@sz],c[@sz];
int  pp[@sz,@sz];
s[1:3,2:3,1:2]={a121,a221,a321,a131,a231,a331,
                a122,a222,a322,a132,a232,a332};
   // s[1,2,1]=a121; s[2,2,1]=a221; ... s[3,3,2]=a332;
s[@sb,@sb]="ab's"+" ab's";          // s の全要素が ab's ab's となる
```

◇特殊コマンド

■スカイライン(skyline)

```
skyline title(…) windowsize(w,h) model(m) option(…)
        color(…) sectorname(snv[…]) var[sbs1,sbs2];
```

---

title(…)	表示する表題
windowsize	表示ウインドウの大きさ
w	表示ウインドウの横幅(300～740, 省略時 720)
h	表示ウインドウの高さ(150～500, 省略時 260)
model	分析モデルの指定
m	t1,t2,m1,m2 の 4 種がある
option(…)	オプション（表題の表示位置, 図表の大きさ, 基準線座標）
color(…)	表示色の指定
sectorname	部門名称の指定
svn	変数名
var	各種生産誘発額を格納した変数名
sbs1	部門添字
sbs2	各種生産誘発額(モデル t1,t2:第１～3 列, モデル m1,m2:第１～4 列)

[処理]
生産誘発額を格納した変数 var は行列である．行は表示対象部門，列は各種生産誘発額であり，描画モデルによって違う．以下の描画モデルを参照のこと．スカイライン図は１つのプロシージャで 10 画面まで同時表示が可能である．

[表示する表題]
スカイライン図表に表示する表題は，title(…)で与える．空白，特殊文字を含むときは引用符でくくる．表示位置は，以下のオプションで与える．指示が無くてもよい．

[表示ウインドウの大きさ]
スカイライン図を表示するウインドウの大きさを指示する．単位はすべてピクセルである．横幅 w は 300～740 の範囲で指定する．省略時は 720 となる．高さ h は 150～500 の範囲で指定する．省略時 250 となる．実際の描画の大きさは option(…)で行う．

例： windowsize(720,250)

［分析モデルの指定］

分析モデルには，t1, t2, m1, m2 の４種がある．以下，説明しよう．産業連関表のバランス式は $\mathbf{x} = \mathbf{Ax} + \mathbf{d} + \mathbf{e} - \mathbf{m}$ と表記できる．ただし， $\mathbf{A}$ は投入係数，$\mathbf{x}$ は国内生産額，$\mathbf{d}$ は国内最終需要，$\mathbf{e}$ は輸出額，$\mathbf{m}$ は輸入額である．これより，均衡算出高モデルは以下のようになる．

モデル t ：$\mathbf{x} = \mathbf{Bd} + \mathbf{Be} - \mathbf{Bm}$

$= \mathbf{x}_D + \mathbf{x}_E - \mathbf{x}_M$

モデル m ：$\mathbf{x} = \mathbf{Bd} + (\mathbf{B_d e} + \mathbf{B}\hat{\mathbf{M}}\mathbf{AB_d e}) - (\mathbf{B}[\hat{\mathbf{M}}\mathbf{AB_d}(\mathbf{I} - \hat{\mathbf{M}}) + \hat{\mathbf{M}}]\mathbf{d} + \mathbf{B}\hat{\mathbf{M}}\mathbf{AB_d e})$

$= \mathbf{x}_D + (\mathbf{x}_E^* + \mathbf{x}_{ME}) - (\mathbf{x}_M^* + \mathbf{x}_{ME})$

ただし，$\mathbf{B} = [\mathbf{I} - \mathbf{A}]^{-1}$，　$\mathbf{B_d} = [\mathbf{I} - (\mathbf{I} - \hat{\mathbf{M}})\mathbf{A}]^{-1}$，　$\mathbf{m} = \hat{\mathbf{M}}(\mathbf{Ax} + \mathbf{d})$

（参考文献）宮川幸三(2005)「スカイラインチャートによる産業構造分析の新たな視点」
『産業連関』第 13 巻 2 号，環太平洋産業連関分析学会.

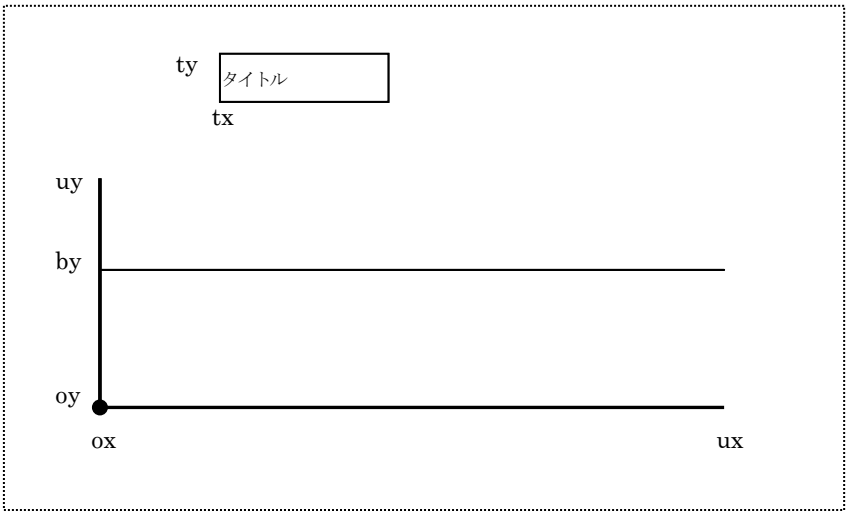
モデル t は従来の一般的分析モデルであり，モデル m は新たな宮川分析モデルである．ここで，$\mathbf{x}_D$ は国内最終需要によって誘発される生産分，$\mathbf{x}_E$ は輸出需要によって誘発される生産分，$\mathbf{x}_M$ は輸入財を国内で生産したと仮定したときに誘発される生産分である．また，$\mathbf{x}_E^*$ は $\mathbf{x}_E$ のうち純粋な国内需要によって誘発される生産分，$\mathbf{x}_{ME}$ は $\mathbf{x}_E$ のうち輸出によって誘発される輸入中間財を国内で生産したとき誘発される生産分，$\mathbf{x}_M^*$ は $\mathbf{x}_M$ のうち国内需要によって誘発される生産分である．

また，横軸の部門別シェアを，国内生産額 $\mathbf{x}$ にする時は「１」，国内最終需要によって誘発される生産額 $\mathbf{x}_D$ にするときは「２」とする．例えば，model(m2)とすれば，宮川分析モデルを用い，横軸の部門別シェアは国内最終需要によって誘発される生産額で目盛られることになる．

［オプション］

オプションは，表示ウインドウの中に描画する表題の表示位置，描画図表の大きさ，基準線の座標を与える． option(tx, ty, ox, oy, ux, uy, by)の各座標は下図の通り．

表示ウインドウ



ウインドウ枠の左下座標を原点(0,0)にしたときの位置であり，単位はすべてピクセルである．tx,ty は表題の表示位置座標，ox,oy は描画図の左下座標，ux,uy は描画図の右上座標，by は基準線の縦座標，である．ウインドウ枠内の値を与える．省略時は option(40,240,50,15,710,165,70) となる．

[表示色の指定]

表示色を指定する．

モデル t ：$\mathbf{x}_i = \mathbf{x}_{Di} + \mathbf{x}_{Ei} - \mathbf{x}_{Mi}$　i：部門

この両辺を$\mathbf{x}_{Di}$で割り，　$\theta_i = \dfrac{\mathbf{x}_i}{\mathbf{x}_{Di}}$，　$\theta_{Ei} = \dfrac{\mathbf{x}_{Ei}}{\mathbf{x}_{Di}}$，　$\theta_{Mi} = -\dfrac{\mathbf{x}_{Mi}}{\mathbf{x}_{Di}}$，　とすれば，

$\theta_i = 1 + \theta_{Ei} - \theta_{Mi}$

となる．

モデル m：$\mathbf{x}_i = \mathbf{x}_{Di} + (\mathbf{x}^*_{Ei} + \mathbf{x}_{MEi}) - (\mathbf{x}^*_{Mi} + \mathbf{x}_{MEi})$　i：部門

同様にして，　$\theta_i = \dfrac{\mathbf{x}_i}{\mathbf{x}_{Di}}$，　$\theta^*_{Ei} = \dfrac{\mathbf{x}^*_{Ei}}{\mathbf{x}_{Di}}$，　$\theta^*_{Mi} = -\dfrac{\mathbf{x}^*_{Mi}}{\mathbf{x}_{Di}}$，　$\theta_{MEi} = -\dfrac{\mathbf{x}_{MEi}}{\mathbf{x}_{Di}}$,とすれば，

$\theta_i = 1 + (\theta^*_{Ei} + \theta_{MEi}) - (\theta^*_{Mi} + \theta_{MEi})$

となる．

ここで指定する色の順序は以下の通りであり，括弧内は省略時の色である．ただし，0 は X 軸であり，1は 100%ラインである．また，モデル t のときは①～④，モデルmのときは①～⑤を指定しなければならない．

(1) $\theta_i < 1$のとき，

①$0 \sim \theta_i$ の部分(White)．

②該当部分は無い．

③$\theta_i \sim 1$の部分(Yellow)．

④$1 \sim (1+\theta_{Ei})$ または $1 \sim (1+\theta^*_{Ei})$ の部分(SkyBlue)．

⑤$(1+\theta^*_{Ei}) \sim (1+\theta^*_{Ei}+\theta_{MEi})$の部分(Blue)．

(2) $\theta_i > 1$のとき，

①$0 \sim 1$の部分(White)．

②$1 \sim \theta_i$ の部分(White)．

③該当部分は無い．

④$\theta_i \sim (1+\theta_{Ei})$ または $\theta_i \sim (1+\theta^*_{Ei})$ の部分(SkyBlue)．

⑤$(1+\theta^*_{Ei}) \sim (1+\theta^*_{Ei}+\theta_{MEi})$の部分(Blue)．

省略時は，４色のとき color(white, white, yelllow, skyBlue)，５色のとき color(white, white ,yelllow, skyblue, blue)である．

指定できる色名は次の通りであり，これ以外の色は指定できない．

aqua, aquamarine, black, blue, blueviolet, brown, cyan, darkblue,
darkcyan, darkgray, darkgreen, darkmagenta, darkorange, darkred,
darksalmon, darkviolet, deeppink, deepskyblue, gold, gray, green,

greenyellow, hotpink, ivory, lawngreen, lightblue, lightcyan,
lightgreen, lightgrey, lightpink, lightsalmon, lightskyblue,
lightyellow, lime, limegreen, magenta, mediumblue, mediumorchid,
mediumpurple, mediumvioletred, midnightblue,navy, olive, orange,
orangered, palegreen, pink, plum, purple, red, salmon, sandybrown,
seagreen, silver, skyblue, snow, springgreen, violet, wheat, white,
whitesmoke, yellow, yellowgreen

[部門名称の指定]
描画された各部門の上部に部門名称(部門コードの場合も含む)を表示させることができる．表示名称の長さは統一する必要はない．しかし，狭い部分に表示されるので，あまり長くない方がよい．表示不要の部門は空白にする．部門名称は重ならないように表示する位置を５段階に分け調整して行うので，上部に余白を確保しておく必要がある．部門名称の表示が不要ならば，このオプション指示は省略することができる．

表示名称を格納した文字列変数を与える．部門の大きさは，次の生産誘発額を格納した変数と一致していなければならない．

[各種生産誘発額を格納した変数名]
各種生産誘発額を格納した変数名を与える．行は部門であり，列は生産誘発額である．モデルによって与える列数と値が異なる．

一般モデル		宮川モデル	
第１列	$\mathbf{x}$	第１列	$\mathbf{x}$
第２列	$\mathbf{x}_D$	第２列	$\mathbf{x}_D$
第３列	$\mathbf{x}_E$	第３列	$\mathbf{x}_E^*$
第４列	$\mathbf{x}_M$	第４列	$\mathbf{x}_M^*$
		第５列	$\mathbf{x}_{ME}$

[使用例]

```
skyline  title("Japan 2005") model(t1) jp[1:20,1:4];
skyline  title("U.S.A") width(720) height(400) us[1:30,{1,3,4,5}];
skyline  title("Canada 2000") windowsize(600,300) model(m2)
             color(White,LightYellow,LightPink,Pink,Red)
             sectorname(name[1:30]) ca[1:30,{1,3,4,5,6}]
             option(40,250,20,15,550,200,60);
```

■ファイル変換(fconvert)

```
fconvert indatafile(…)          // 読み込みデータのファイル名
         outdatafile(…)         // 書き出しデータのファイル名
         infieldsep(・)         // 読み込みデータのフィールド区切記号
         outfieldsep(・)        // 書き出しデータのフィールド区切記号
         line(line1:line2)      // 処理対象データの行選択(line1～line2)
         select(cond …) …       // 処理対象データの条件選択
         insert(opt1,str) …     // 文字列の挿入
        (opt1 opt2) ( … ) … ;   // 変換指示
```

```
cond                        条件式
  value1 rop opt1 rop value2
                              value1,value2   定数, 文字列
                              rop             比較演算子(>,>=,<=,<,=,!=)
str                         挿入文字列
opt1                        入力データの指示
  inpos(ip1:ip2:topt)         範囲(ip1～ip2)
  infield(if:topt)            フィールド(if)
                              範囲･フィールド内の空白除去処理(topt)
                                t   データ前後の空白を取り除く
                                nt  データ前後の空白を取り除かない
                                ts  データの前空白を取り除く
                                te  データの後空白を取り除く
opt2                        出力データの指示
  outpos(op1:op2:aopt)        範囲(op1～op2)
  outfield(of)                フィールド(of)
                              範囲内の左寄せ/右寄せ処理(aopt)
                                l   データを左寄せにする
                                r   データを右寄せにする
```

[処理]

ファイル形式を変換する．コマンドは fconv でもよい．ファイル形式には，固定形式とフィールド形式がある．後者は，データとデータを区切記号で区切った形式のファイルのことであり，各データをフィールドと呼び，ADAM の標準データ形式である．コンマで区切られていれば，よく知られた CSV 形式である．

このコマンドは，固定形式ファイルからフィールド形式ファイルへの変換，フィールド形式ファイルから固定形式ファイルへの変換，データ位置の組み替え，必要データの抽出，データ前後の空白の除去，処理対象データの条件選択などの処理が可能である．

[読み込みデータのファイル名]
読み込みデータは本コマンドの原データであり，indatafile(…)にそのファイル名を与える．

[読み込みデータのフィールド区切記号]
読み込みデータがフィールド形式であるとき，その区切記号は infieldsep(・)で与える．括弧の中に１文字の記号を指定する．このオプションの指定を省略するとコンマになる．英数字は指定できない．読み込みデータが固定形式であれば，このオプションは無視される．

区切記号を空白にしたいときは infieldsep()とする．このときの空白区切記号は半角文字であり，全角文字でないこと明記する．データとデータの区切り空白が複数桁連続するときは，最初の空白が区切記号と認識され，それに続く２文字目以降の空白は次のデータの前空白とみなされる．この前空白を除去したいときは空白除去処理オプション(後出)を指示する必要がある．

[書き出しデータのファイル名]
書き出しデータは変換後の出力データであり，outdatafile(…)にファイル名を与える．

[書き出しデータのフィールド区切記号]
書き出しデータがフィールド形式のときの区切記号は outfieldsep(・)を用い，括弧の中に１文字の記号を指定する．区切記号を空白にしたいときは，outfieldsep()とする．このオプションの指定を省略するとコンマになる．英数字は指定できない．書き出しデータが固定形式であれば，このオプションは無視される．

[処理対象データの行選択]
処理の対象とするデータを行数で指定する．例えば，line(1:100)とすれば，第 1 行目～第 100 行目のデータが処理の対象となる．line(10)とすれば，第 10 行目から最後のデータまでが処理の対象となる．指定しなければ全データが処理の対象となる．

[処理対象データの条件選択]
処理対象データを選択する条件を指定する．例えば，select(100<=inpos(11:15)<200)とすれば，入力データ第 11～15 桁の値が 100 以上 200 未満のデータが処理対象となる．また，

```
select(100<=inpos(11:15)<200,inpos(1:1)=″A″)
```

とすれば，入力データ第 11～15 桁の値が 100 以上 200 未満であり，かつ入力データ第 1 桁が″A″であるデータが処理対象となる．

この select オプションを以下のように２つに分けて与える．

```
select(100<=inpos(11:15)<200)
select(inpos(1:1)=″A″)
```

このようにすると，入力データ第 11～15 桁の値が 100 以上 200 未満であるデータ，または入力データ第 1 桁が"A"であるデータが処理の対象になる．すなわち，一つの select オプション内の各条件は「かつ(and)」で結びつけられて評価され，並列される select オプションは，それぞれが独立に「または(or)」で評価されることになる．条件指示の整合性は確認されない．

各条件は，評価対象とする入力データを固定形式またはフィールド形式で与え(後出)，その片側または両側に比較演算子(>, >=, <=, <, =, !=)と判定値を用いて指示をする．判定値は定数か文字列である．文字列のときはその大小関係に注意が必要である．評価対象データに出力データを示す固定形式 outpos(…)またはフィールド形式 outfield(…)を指示することはできない．

[文字列の挿入]
出力データの一部に固定文字を挿入するオプションである．挿入したい位置を固定形式またはフィールド形式で与え(後出)，挿入文字列を与える．

```
insert(outpos(1:1),"A")
insert(outf(1),"C")
```

［変換指示］
変換指示は，原データから出力データへの転写･編集の対応である．原データと出力データの対応を一組とし，括弧でくくる．与える順序に制約はない．対応の指示は select オプションで与える条件の数も含め最大 50 組である，例えば，２組の場合は，

```
(inpos(11:20) outpos(1:10)), (outpos(11:15) inpos(1:5))
```

となる．この例は，原データの第 11～20 桁が出力データの第 1～10 桁に転写され，原データの第 1～5 桁が出力データの第 11～15 桁に転写される．inpos(…)は転写元の原データ範囲であり，outpos(…)は転写先の出力データ範囲である．in…と out…の指示順序は自由である．

原データが固定形式であれば inpos(…)，フィールド形式であれば infield(…)を用いる．同様に,出力データが固定形式であれば outpos(…),フィールド形式であれば outfield(…)を用いる．したがって，その対応の組み合わせは次の何れかとなる．

- □ 固定形式 inpos(…) から 固定形式 outpos(…)
- □ 固定形式 inpos(…) から フィールド形式 outfield(…)
- □ フィールド形式 infield(…) から 固定形式 outpos(…)
- □ フィールド形式 infield(…) から フィールド形式 outfield(…)

[原データの範囲 inpos(ip1:ip2:topt))]
範囲は，その開始桁 ip1 と終了桁 ip2 をコロンで結び指示をする．開始桁のみの指示は許されない．inpos(…)は inp(…)でもよい．topt は，指定した範囲内の空白除去処理オプション指示(後述)であり，コロンで結び指示をする．複数組の対応があるとき，同じデータを使用する重複指示があってもよい．

[原データのフィールド infield(if:topt))]
原データのフィールド位置 if を指示する．infield(…)は inf(…)でもよい．topt は，指定したフィールドのデータの空白除去処理オプション指示(後述)であり，コロンで結び指示をする．複数組の対応があるとき，同じデータを使用する重複指示があってもよい．

[空白除去処理オプション(topt)]
範囲･フィールド内での空白除去処理オプションは次の通りである．範囲またはフィールドの後にコロンを付け指示をする．指示がないときは，データ前後の空白を取り除かないオプション「:nt」 とみなされる．

- □ t データ前後の空白を取り除く
- □ nt データ前後の空白を取り除かない
- □ ts データ前の空白を取り除く
- □ te データ後の空白を取り除く

[出力データの範囲 outpos(op1:op2:aopt)]
出力データの範囲は，その開始桁 op1 と終了桁 op2 をコロンで結び指示をする．開始桁のみの指示は許されない．outpos(…)は outp(…)でもよい．aopt は，左寄せ/右寄せオプション指示(後述)であり，指定した範囲内にデータを左寄せにするか右寄せにするかの指示である．コロンで結び指示をする．複数組の対応があるとき，出力データの範囲に重なりや空き(不連続)があってはならない．

[出力データのフィールド outfield(of)]
出力データのフィールド位置 of を指示する．outfield(…)は outf(…)でもよい．複数組の対応があるとき，出力データのフィールド位置は，指示する順序は自由であるが，第 1 フィールドから始まり全体が連続していなければならない．

[左寄せ/右寄せオプション(aopt)]
範囲内での左寄せ/右寄せ処理オプションは次の通りである．範囲の後にコロンを付け指示をする．指示がないときは，左寄せ「:l」とみなされる．

- □ l データを左寄せにする(英子文字エル)
- □ r データを右寄せにする

[使用例]

```
fconvert  indatafile(td01.dat)  outdatafile(td02.dat)
```

```
            select(100<=inpos(21:25))
            select(inpos(21:25)<100,500<inpos(31:35)<700)
            (outpos(1:20:r),inpos(1:20:t));

fconvert  indatafile(td03.dat)  outdatafile(td04.dat)
            outfieldsep(,)
            (inpos(1:5:t),outf(1))    (inpos(6:10:nt),outf(2))
            (inpos(11:15:t),outf(3)) (inpos(16:20:nt),outf(4))
            (inpos(21:25),outf(5))    (inpos(26:26:nt) outf(6));

fconvert  indatafile(td05.dat)  outdatafile(td06.dat)
            infieldsep(,)  outfieldsep(,)
            (inf(2),outf(1)), (inf(6),outf(2)), (inf(5),outf(3));

fconv     indatafile(td09.dat)  outdatafile(td10.dat)
            (inf(1:te) outpos(1:5:r)) (inf(2) outpos(6:10:r))
            (inf(3:ts) outpos(11:15:r)) (inf(4) outpos(16:20:r));
```

◇作表支援コマンド

■部門分類表の作成(makesectortable)

```
makesectortable stname;        // 部門分類表定義の開始
  beginrowsector;              //
    rowcode-1   rowname-1;     //   ┐
    rowcode-2   rowname-2;     //   │行部門コード・名称の定義
      …           …            //   │
    rowcode-n   rowname-n;     //   ┘
  endrowsector;                //
  begincolumnsector;           //
    colcode-1   colname-1;     //   ┐
    colcode-2   colname-2;     //   │列部門コード・名称の定義
      …           …            //   │
    colcode-m   colname-m;     //   ┘
  endcolumnsector;             //
endsectortable;                // 部門分類表定義の終了
```

---

stname	部門分類表定義名
rowcode-i	行部門コード
rowname-i	行部門名称
colcode-j	列部門コード
colname-j	列部門名称

[処理]

行および列の部門コードと名称を定義し，「部門分類表」を作成する．そして，その内容は与えた「部門分類表定義名」で書き出しファイル保存される．拡張子は .adt となる．この保存された部門分類表ファイルは，作表作業で呼び出され使用されることになる．十分な検討と準備を行い，定義処理をする必要がある．作表途中でこの部門分類表に変更があれば，その時点までの全作業を再処理し直さなければならない．再処理についての備えは以下を参照のこと．

[実行ファイル名の付け方]

ADAM は，部門分類表の内容が変更されると，その部門分類表名のもとに定義されている既存の作業表のすべてが無効になる．作表は一貫した部門分類で処理する必要があり，作業途中での部門分類の変更は，全体の作業に大きな混乱を招くだけでなく，作表事故の原因ともなる．これを回避するための措置である．

しかし，作業途中で部門分類表に不備を見つかり，部門分類表の変更を余儀なくされることはよくある．このような事態に対処するためには，全作業の再処理を間違いなく実行できるように，全作業の実行ファイルを漏れなく順序正しく管理しておくことが肝要となる．

実行ファイル名の付け方を，a01xxxxx，a02yyyyy，…，b01zzzzz のように，先頭に英数字を組み合わせた処理順序番号を置き，その後に処理内容を示す情報を付加する．このようにすると，エクスプローラによる表示でも，全実行ファイルが整然と並び，再処理手順の管理は容易となる．再処理手順が少々長くても，全体の処理時間は僅かなものである．実行ファイルの一括実行(submit)の活用もある．また，実行ファイル(プロシージャ)の内容は，作表作業の完全にして正確無比な記録である．この観点からも，全実行ファイルをしっかりと保存管理することが重要である．

ADAM による処理は実行ファイルだけでない．その処理に必要なデータファイルやコードファイルなどの入力ファイル，結果を書き出す出力ファイルがある．これら３種のファイルが混在していると上述の再処理に手違いを生じることになる．この混乱を回避するための一例としては，実行ファイル(プロシージャ)の頭には c_ を，データファイルなどの頭には d_ を，後続ステップで使用する出力ファイルの頭には g_ を，後続ステップで使用しない出力ファイルの頭には p_ を，出力ファイルでも保存の必要がないものには t_ を付けるようにすると，エクスプローラによる表示でも整理されることになる．すなわち， a01 を処理順序番号とすれば，実行ファイル c_a01xxxx の実行に際しては，入力データファイル d_a01yyyy が使用され，結果ファイルが p_a01zzzz として書き出され，それぞれのファイル種別が峻別でき，どの実行ステップに関連しているかが一目瞭然である．一例ではあるが，ファイル名の実践的な命名手法である．

[部門分類表定義の開始]
部門分類表の定義の開始は makesectortable であり，その終了は endsectortable; である．与えた「部門分類表定義名」で部門分類表の内容が自動的にファイル保存される．続く作表作業でそれを呼び出し使用する．出力ファイル名は与えた定義名の後に拡張子 .adt が付加されたものになる．なお，書き出し先フォルダはプロジェクトフォルダであり，これ以外のフォルダに書き出し指示をすることはできない．

以下のコード・名称の定義を与える順序は，行部門，列部門の順でも，列部門，行部門の順でもよい．

[行部門コード・名称の定義]
行部門コードと名称の定義は，全体を beginrowsector; と endrowsector; でくくる．行部門コードと名称は，一部門ずつこの順序で与え，終わりにセミコロンを付ける．行部門コードと名称の区切りは空白またはコンマとする．また，与える部門の順序は，行部門コードが昇順(Unicode，その順序は数字，英字)になっていなければならない．部門数に制限はない．コードに全角文字の使用は可能であるが，半角英数字が望ましい．作業途中の指

示の容易性を考慮すると特殊文字(ハイフン，ピリオド，コロンなど)の使用は避けた方がよい．特殊文字を部門コードに使用すると，指示の度にその部門コードを引用符でくくらなければならなくなり，その負担は大きい．なお，最終報告書において部門コードを見やすくするためにハイフンを挿入する処置は，報告書作成時に簡単に対応することができる．

名称も同様に全角文字の使用は自由である．ただし，空白文字やハイフン，ピリオド，セミコロン，コロン，コンマ，括弧などの特殊文字が含まれるときは，その全体を引用符でくくる必要がある．

[列部門コードと名称の定義]
列部門コードと名称の定義は，全体を begincolumnsector; と endcolumnsector; でくくる．列部門コードと名称は，一部門ずつこの順序で与え，終わりにセミコロンを付ける．コードと名称の区切りは空白またはコンマとする．また，与える部門の順序および使用文字は上述の[行部門コードと名称の定義]と同様である．

[使用例]

```
makesectortable g_a01stbase;        // g_a01stbase は部門分類表定義名である．
  beginrowsector;
    0111011        米;
    0111012        稲わら;
    0111021       ″小麦（国産）″;
    0111011       ″小麦（輸入）″;
       …
    ″6111011-6″  卸売りマージン;
       …
    9700000       生産額;
  endrowsector;
  begincolumnsector;
    011101         米;
    011102         麦類;
    011103         穀類;
       …
    970000         生産額;
  endcolumnsector;
endsectortable;
```

■部門分類表の指定(sectortable)

```
sectortable stname;
-----------------------------------------------
          stname          部門分類表定義名
```

[処理]
使用する部門分類表の定義名を指定する．拡張子.adt を指定してはいけない．指定した部門分類表はファイルから読み込まれ，続くコマンドで使用されることになる．複数の部門分類表を同時に使用したいときは，このコマンドを複数行与え指定する．なお，読み込みフォルダはプロジェクトフォルダであり，これ以外のフォルダから読み込むことはできない．ファイル名は，部門分類表定義名に拡張子 .adt が付加されたものである．

[コードと名称の変数名]
このコマンドで部門分類表が読み込まれると，行部門コードと名称および列部門コードと名称の変数名は，自動的に次のように生成される．この変数名は ADAM コマンドで自由に用いることができる．ただし，xxxx は部門分類表定義名である．

- □ 行部門コード　_rc_xxxx
- □ 行部門名称　_rn_xxxx
- □ 列部門コード　_cc_xxxx
- □ 列部門名称　_cn_xxxx

[使用例]

```
sectortable g_a01stbase;

double ta(g_a01stbase), … ;  // ta(g_a01stbase)は「■部門分類付変数」の
       …                     // 定義を参照.
int    tzz(g_a01stbase),xx[700,600], …;
       …
ta[1:100,1:30] = …;

…=ta[index(_rc_g_a01stbase,A0111011):index(_rc_g_a01stbase,F9700000),1:30];
…=ta[$A0111011:$F9700000,1:30]; // 前行例と同義.
                                // $指示は部門分類付変数の定義を参照.
q;

sectortable  g_a01stbase;
sectortable  g_a02stbase2;

double ta(g_a01stbase),ta2(p_a02stbase2),xx[700,600], …;
```

```
int     tzz(g_a01stbase),tzz2(g_a02stbase2), …;
        …
ta2[$A0111011:$F9700000,1:30]=ta[$A0111011:$F9700000,1:30];

q;
```

■想定部門分類表の指定(assumedsectortable)

```
assumedsectortable stname;
```

---

stanme	部門分類表定義名

[処理]
変数と連動しないところで想定する部門分類表を指定するコマンドである．変数を部門分類付変数として定義をすれば，その変数は部門分類表と自動的に連動するので，添字に「$指示」を安全に使用することができる．しかし，シートデータ入出力の添字変数(#1 および#2)などは連動する部門分類表がないため添字に「$指示」を使用することができない．変数と連動しない局面において「$指示」を可能にするため，そこに想定する部門分類表を指定することが，このコマンドの機能である．

[使用例]

```
assumedsectortable g_a01stbase;
```

■部門分類付変数の定義(integer,doble,string)

```
integer   varname …;      整数変数
double    varname …;      浮動小数点変数
string    varname …;      文字列変数
```

---

varname	変数名であり，次の何れかの形式をとる
vn(stname)	行列
vn(stname,m)	行列
vn(stname,r)	行部門分類ベクトル
vn(stname,c)	列部門分類ベクトル
vn	変数名
stname	部門分類表定義名である．

[処理]
部門分類付変数を定義する．変数名の後に丸括弧で適用する部門分類表定義名を与える．

部門分類表定義名 stname は，sectortable コマンドで事前に読み込み指示がなくてもプロジェクトフォルダ内に該当ファイルがあれば，自動的に読み込みが行われる．

[部門分類付変数]
部門分類付変数は，ADAM の標準変数と区別した呼び名である．変数定義は，同じ行に部門分類付変数と標準変数が混在していてもよい．

部門分類付変数は適用する部門分類表のコードと名称が，行部門分類と列部門分類の双方であれば部門分類付行列(m)，行部門分類だけならば行部門分類付ベクトル(r)，列部門だけならば列部門部門付ベクトル(c)となる．m，r，c は変数種別である．ただし，ベクトルの形はすべて列ベクトルである．演算コマンドで行ベクトルとして指示をするときは転置関数を用いる．

変数の行および列の要素数は，部門分類表のそれぞれの大きさになる．

[部門分類付変数の添字]
ADAM の標準的な添字指示はそのまま指示可能である．部門分類付変数は部門分類表が付加された変数であるため，添字の index(…)関数の部分を簡単に指示できる．部門分類付変数が使用できる添字の簡単な例を示す．

```
sectortable g_a01stbase;
ta[1:110,1:30]
ta[{1:100,108:110},1:30]
ta[index(_rc_ g_a01stbase,0111011),index(_cc_ g_a01stbase,011101)]
ta[$0111011,$011101]
```

上の例の１行目は部門分類表の指定である．２行目と３行目は，見慣れた例である．４行目は「■部門分類表の指定（sectortable）」で見たように，自動生成された変数名(部門分類表定義名が g_a01stbase であるから，_rc_ g_a01stbase，_cc_ g_a01stbase はそれぞれは行コード変数名と列コード変数名）を用いた index(…)関数の使用である．５行目は４行目と全く同義であり，index()関数部分を，先頭に記号($)を付したコードだけの指示に換えている．記法の簡便化であり，以下，この指示を「$指示」と呼ぶ．この記法は部門分類付変数にのみ安全に使用できる．assumedsectortable コマンドによる想定部門分類表の指定があれば，標準変数についても「$指示」を使用することは可能である．しかし，注意が必要である．詳しくは，「■変数の定義」を参照のこと．

「$指示」の変数名 _rc_xxxx および _cc_xxxx は，指示した位置に合ったものが自動的に用いられる．変数種別が行列であれば行位置は _rc_xxxx が，列位置は _cc_xxx が使用され，行部門分類付ベクトルであれば _rc_xxxx が，列部門分類付ベクトルであれば _cc_xxx が使用される．

[使用例]

```
sectortable g_a01stbase;
double ta(g_a01stbase),tr(g_a01stbase,r),tc(g_a01stbase,c);
       …
ta[$0111011:$9700000,$011101:$970000] = …;
       …
tr[$0111011] = …;  // $0111011 は index(_rc_g_a01stbase,0111011) と同義である.
tc[$011101] = …;  // $011101 は index(_cc_g_a01stbase,011101) と同義である.
       …
q;
```

■変数値の書き出し(put)

```
put varname-1(filename-1)  varname-2(filename-2)  … ;
-----------------------------------------------------
    varname-i           変数名
    filename-i          ファイル名
```

[処理]

作業途中の中間結果をファイルに書き出しその内容を保存する．書き出し先フォルダはプロジェクトフォルダであり，これ以外のフォルダに書き出すことはできない．部門分類付変数および標準変数を指定できる．ただし，ベクトルと行列だけである．

作表は試行錯誤を重ねる複雑な作業の繰り返しであり，作表処理を１つの実行ファイルで済ますことは不可能である．膨大な作業を細かな作業に分割し，一歩一歩その結果を確認しながら進めていくことになる．その手順は，先ず作業途中の中間結果である変数値をファイルに書き出し，その内容を保存する．処理結果内容を確認後，保存したファイルを読み込み，その変数値を復元し，その時点からの処理作業を続行することになる．

[変数名とファイル名]

保存したい変数名を指定する．変数名の後に丸括弧を付け，その中にファイル名を与えることができる．拡張子を付けることはできない．ファイル名の後には拡張子 .adv が自動的に付加される．ファイル名を省略したときは，変数名がファイル名として用いられる．保存変数は非常に多くなることが予想され，同変数名の書き出しで，既存の変数ファイルを上書きしてしまわないように，ファイル名のつけ方には十分な配慮が必要である．ファイル名を指定しないときは，更なる注意が必要となる．ファイル名の頭に前述のように g_ と処理順序番号を付けるのがよいであろう．どの作業過程で保存された変数値であるかが一目瞭然となる．

[使用例]

```
sectortable g_a01stbase;              // このプロシージャのファイル名は
       …                              // c_a03xxxx であるとする.
double ta(g_a01stbase),xx[700,600], … ;
int    tzz(g_a01stbase), …;
       …
ta[1:100,1:30] = …;
       …
put ta(g_a03ta),tzz(g_a03tzz);
q;
```

■変数値の読み込み(get)

```
get varname-1(filename-1)  varname-2(filename-2)  … ;
```

---

varname-i	変数名
filename-i	ファイル名

[処理]

上述の中間結果の書き出しコマンド(put)で書き出した中間結果をファイルから読み込み,変数値を復元する. 読み込み先のフォルダはプロジェクトフォルダであり, これ以外のフォルダから読み込むことはできない.

このコマンドに指定した変数は, 変数の定義が自動的に行われるので, 変数の定義をしてはいけない.

[変数名とファイル名]

値を復元する変数名を指定する. 変数名の後に丸括弧を付け, その中にファイル名を与えることができる. 拡張子を付けることはできない. ファイル名の後には拡張子 .adv が自動的に付加される. ファイル名を省略したときは, 変数名がファイル名として用いられる.

[使用例]

```
sectortable g_a01stbase;              // このプロシージャのファイル名は
       …                              // c_a04xxxx であるとする.
double xx[700,600], … ;
       …
get ta(g_a03ta),tzz(g_a03tzz);        // 変数 ta,tzz は自動的に定義される.
       …
ta[1:100,1:30] = …;
       …
q;
```

■マージン表の作成(makemargintable)

```
makemargintable                          // マージン表作成の開始
    sectortable(…)                       // 部門分類表定義名
    margintable(…)                       // 新規マージン表定義名
    datafile(…)                          // 原データファイル名
    fieldsep(・)                         // フィールド区切記号
    corefield(rcfn,ccfn,prfn)            // 行, 列, 取引額フィールド位置
    margin(new-mgrsc,mgfn,…)             // 新規マージン行部門の設定
      …
    margin(…);
   beginrowsector(rowitemsize);          // ┐
     nrc rc-1*wt, … ;                    // │新データと原データの行部門対応
     …                                   // │一律ウエイトの設定
   endrowsector;                         // ┘
   begincolumnsector(columnitemsize);    // ┐
     ncc cc-1*wt, … ;                    // │新データと原データの列部門対応
     …                                   // │一律ウエイトの設定
   endcolumnsector;                      // ┘
   beginweight(weightitemsize);                // ┐
     nrc-1:nrc-2 ncc-1:ncc-2*(wt-1, …,wt-k); // │マージンの
     …                                         // │形態別ウエイトの設定
   endweight;                                  // ┘
   beginextra;                                 // ┐
     nrc ncc pr,mg-1, …, mg-k;                 // │新規マージンの直接追加
     …                                         // │
   endextra;                                   // ┘
endmargintable;                                // マージン表作成の終了
```

---

sectortable(…)	部門分類表定義名
margintable(…)	新規マージン表定義名
datafile(…)	原データファイル名
fieldsep(・)	原データのフィールド区切記号
corefield(rcfn,ccfn,prfn)	原データのフィールド
rcfn	行コードのフィールド位置
ccfn	列コードのフィールド位置
prfn	生産者価格表示取引額のフィールド位置
margin(new-mgrsc,mgfn,…)	新規マージン行部門の設定
new-mgrsc	新規形態別マージンの行部門コード

mgfn	原データの形態別マージンのフィールド位置
nrc, nrc-i	新規マージン表の行部門コード
ncc, ncc-i	新規マージン表の列部門コード
rc-1	原データ行部門コード
cc-1	原データ列部門コード
pr	生産者価格表示取引額
wt, wt-i	一律，形態別ウエイト
mg-i	形態別マージン額
rowitemsize	原データの行部門項目数
columnitemsize	原データの列部門項目数
weightitemsize	新規マージン表のウエイト項目数

[処理]

作表過程で行うマージン処理に使用するマージン表を作成し，与えた「マージン表定義名」で保存する．保存先のフォルダはプロジェクトフォルダであり，これ以外のフォルダに書き出すことはできない．

このマージン表は財取引にともなうマージンを形態別に算出し，購入者価格評価表示から生産者価格評価表示への組み替え処理(margindrop)，生産者価格評価表示から購入者価格評価表示への組み替え処理(marginback)に使用する．ここで作成するマージン表は，全国基本表(総務省)のマージン値(あるいはそれに準ずる表値)を基本にしての組み替えや加工を前提にしている．また，新部門のマージン額の直接組み込みが可能である．全体は次の４部から構成され，その部の指示順序は任意であり，weight 部と extra 部は必要がなければ省略することができる．

- □ rowsector 部　　beginrowsector … endrowsector
- □ columnsector 部　　begincolumnsector … endcolumnsector
- □ weight 部　　beginweight … endweight
- □ extra 部　　beginextra … endextra

[makemargintable]

このコマンドの処理に必要な基本項目を与える．項目の指示順序は，margin オプション指示の前に sectortable オプション指示が必要であるが，他は任意である．なお，このコマンドの終了は，全体の最後におく endmargintable である．使用例を示し説明する．

```
makemargintable
  sectortable(g_a01stbase)    // ①部門分類表定義名
  margintable(g_a02mgtbase)   // ②新規マージン表定義名
  datafile(d_jiobase.dat)     // ③原データファイル名
  fieldsep(,)                 // ④原データのフィールド区切記号
  corefield(2,1,3)            // ⑤行コード，列コード，取引額フィールド位置
  margin("6111011-6",5)       // ⑥形態別新規マージンコードと
```

```
margin("6112011-6",6)      // ⑦          原データのフィールド位置の対応
margin("7112011-7",8)      // ⑧
margin("7122011-7",9)      // ⑨
margin("7142012-7",10)     // ⑩
margin("7143011-7",11)     // ⑪
margin("7151013-7",12)     // ⑫
margin("7161011-7",13)     // ⑬
margin("7171011-7",14) ;   // ⑭
```

① sectortable(…) は，ここで作成される新規マージン表の基準になる部門分類表定義名を与える．したがって，このコマンドのこの処理時には，その前に sectortable コマンドによる部門分類表の指定が必要である．このオプションは省略できない．上の例では，その定義名が g_a01stbase となっている．

② margintable(…) は，ここで作成された新規マージン表を識別する「新規マージン表定義名」であり，ファイル保存の際のファイル名になる．ファイル名は拡張し.adr が付いて，g_a02mgtbase.adr となる．このオプションは省略できない．上の例では g_a02mgtbase であり，拡張子は付けない．

③ datafile(…) は，原データのマージンファイル名である．総務省の全国基本表を想定している．プロジェクトフォルダまたはデータフォルダを検索する．このオプションは省略できない．上の例では， d_jiobase.dat がそのファイル名である．

④ fieldsep(・) は，原データのフィールド区切記号である．１文字の半角文字記号を与える．英数字は指定できない．空白指定であれば fieldsep()とする．上の例ではカンマを指定している．このオプションを省略するとカンマが想定される．

⑤ corefield(rcfn,ccfn,prfn) は，原データ内における行コードと列コードおよび生産者価格表示取引額のそれぞれのデータの在りかを示すフィールド位置であり，この順序で指定する．すなわち，rcfn は行コードのフィールド位置，ccfn は列コードのフィールド位置，prfn は生産者価格表示取引額のフィールド位置である．それらの区切は空白でもカンマでもよい．このオプションは省略できない．上の例では，行コードが第２フィールド，列コードが第１フィールド，生産者価表示取引額が第３フィールドに存在していることを示している．

⑥～⑭ margin(new-mgrsc,mgfn,…) は，これから作成する新規マージン表における形態別マージンの行部門コードとそれに対応する原データ内のデータ位置をこの順序で与える．new-mgrsc は新規マージン表における形態別マージンの行部門コード，mgfn はそれに対応する原データ内のデータの在りかを示すフィールド位置である．区切は空白でもカンマでもよい．新規マージン形態の数だけその指示が必要である．このオプション指示の前に sectortable オプションおよび corefield オプションの指示が必要である．

上の例では，形態別行部門コードには -6，-7 の枝番が付されおり，基本分類部門で理解すれば，⑥が卸売であり，原データ内の対応するデータのフィールド位置が 5 である．以下同様に，⑦が小売でそのフィールド位置が 6，⑧が鉄道でそのフィールド位置が 8，そして⑭が倉庫でそのフィールド位置が 14，と対応が指示されている．部門コードは部門分類表に定義されているコードでなければならない．部門コードに英数字以外の文字を含むときはその全体を引用符でくくる必要がある．

複数のマージンを統合したいときは，新行部門コードに続けて該当フィールド位置を羅列する．極端な例示ではあるが，もしマージン形態を商業マージン(″61-6″)と運輸マージン(″71-7″)の２種に統合してしまいたいのであれば，それぞれの行部門コードに続けて次のように指示すれば，同一行内に与えたフィールド位置のマージンがすべて統合される．

```
margin(″61-6″,5,6)
margin(″71-7″,8,9,10,11,12,13,14)
```

[rowsector 部 beginrowsector … endrowsector]
この部は，新規マージン表の行部門と原データの行部門との対応を行う．新行部門に複数の原データ行部門を対応させ，また必要ならばウエイトを与え，原データには無い行部門のマージンを設定することができる．

コマンド beginrowsector(rowitemsize)；に与える原データの行部門項目数 rowitemsize は，この部内に指示する原データの行部門項目の総数の上限である．省略時は 800 である．

新規マージン表の行部門と原データの行部門の対応は，行部門コードを用いてこの順序で行う．対応させる原データ部門が複数の場合はそれらを羅列する．それらの区切は空白でもカンマでもよい．最後にセミコロンをつける．羅列した部門のマージンはすべて統合され，その結果，マージン額の構成が変化することもある．

羅列した原データ行部門のそれぞれに必要ならばウエイトを与え，マージン額の構成に部門別の重みを指示することができる．ウエイトを乗じたい原データ行部門コードの後にアスタリスク(*)をつけ，続けてウエイトを正値で与える．ウエイトを省略すると１が想定される．このウエイトは，生産者価格表示取引額および形態別マージンのすべてに一律に乗じられる．

形態別マージン率は，ウエイトつきで統合された生産者価格表示取引額および形態別マージン額を用いて算出される．このため，部門対応とウエイトによって形態別マージン率が大きく変化することもある．

次の例は，最も単純な場合であり，新データと原データの行部門コード対応のみを行い，原データのマージンをそのまま使用してマージン表を作成する．

```
beginrowsector(600);
  0111011  0111011;
  0111012  0111012;
   …
  8519091  8519099;
   …
endrowsector;
```

次の例は，新規部門が統合小分類のケースであり，該当基本分類のマージンが統合される．

```
beginrowsector(600);
  0111  0111011,0111012,…;   // 原データ行部門コードのすべてを羅列する．
  0112  0112011,0112012,…;
  0113  0113001;
  0114  0114011,…;
   …
endrowsector;
```

ウエイト指示は次のように行う．

```
beginrowsector(600);
  0111  0111011*80,0111012*20,…;
  0114  0114011,…;
   …
endrowsector;
```

[columnsector 部 begincolumnsector … endcolumnsector]
この部は，新規マージン表の列部門と原データの列部門との対応を行う．新列部門に複数の原データ列部門を対応させ，また必要ならばウエイトを与え，原データには無い列部門のマージンを設定することができる．

コマンド begincolumnsector(columnitemsize)；に与える原データの列部門項目数 columnitemsize は，この部内に指示する原データの列部門項目の総数の上限である．省略時は 600 である．

新規マージン表の列部門と原データの列部門の対応は，部門コードを用いてこの順で行う．対応させる原データ列部門が複数の場合はそれらを羅列する．それらの区切は空白でもカンマでもよい．最後にセミコロンをつける．羅列した列部門のマージンはすべて統合され，その結果，マージン額の構成が変化することもある．

羅列した原データ列部門のそれぞれに必要ならばウエイトを与え，マージン額の構成に部門別の重みを指示することができる．ウエイトを乗じたい原データ列部門コードの後にア

スタリスク(*)をつけ，続けてウエイトを正値で与える．ウエイトを省略すると１が想定される．このウエイトは，生産者価格表示取引額および形態別マージンのすべてに一律に乗じられる．

形態別マージン率は，ウエイトつきで統合された生産者価格表示取引額および形態別マージン額を用いて算出される．このため，部門対応とウエイトによって形態別マージン率が大きく変化することもある．

```
begincolumnsector(500);
  011101    011101;
   …
  6411011   641101;
  6411012   641101;
   …
  851909    851909;
   …
endcolumnsector;

begincolumnsector(500);
  01001     011101*80,013101*20;
   …
  15001     151101,151401,151902,151909;
   …
endcolumnsector;
```

上の例は，前半は最も単純な場合である．後半の 01001 列部門は原データの２箇所の部門を統合し，かつそれぞれに 80，20 のウエイトを乗じてマージンを設定している．15001 は原データの複数列部門を統合し，マージンを設定している．

次の例は，新規部門が統合小分類であり，該当基本分類のマージンが統合される．

```
begincolumnsector;
  0111   011101,011102,011103;    // 原データ列部門コードのすべてを記述する
  0112   011201,011202;
   …
endcolumnsector;
```

[weight 部 beginweight … endweight]

この部は，新規マージンに対し形態別ウエイトを与える．地域や部門によっては航空輸送がなく，その分道路輸送が多いなどの特性を考慮しなければならない．たとえば，航空マージンを 0 にし，道路マージンを 10%増やすなどである．このときは，道路マージンウエイトを 1.1 とし，航空マージンを 0.0 にする．

コマンド beginweight(weightitemsize); に与える weightitemsize は，この部内に指示する新規マージン表のウエイト項目総数の上限である．省略時は 800 である．

この指示は新規部門別に与える．範囲指定も可能である．行部門コード，列部門コード，形態別マージンウエイトをこの順序で与える．形態別マージンウエイトは，列部門コードの後にアスタリスク(*)をおき，上の margin で与えた形態別マージン行部門の順序にしたがって形態の数だけ与え全体を括弧でくくる．それらのコードと数値は空白あるいはカンマで区切り，最後にセミコロンをつける．指定した部門の重複確認は行われないので，部門指定は慎重に行わなければならない．例を示す．

```
beginweight;
  0101:3900 0101:3900*(1.1,1,1,1,1,1,1,0.0,1,1);
   …
  3319091 3351909*(1.1,1,1,1,1,1,1,0.0,1,1);
   …
endweight;
```

[extra 部 beginextra … endextra]

この部は，直接追加したい新規部門のマージンを与える．行部門コード，列部門コード，生産者価格表示取引額，形態別マージン額をこの順序で与える．形態別マージン額は上の margin で与えた形態別マージン行部門の順序に従って形態の数だけ与えることになる．それらのコードと数値は空白あるいはカンマで区切り，最後にセミコロンをつける．追加データの量に制限はない．使用例を示す．

```
beginextra;
  A8519091 K851909 12345,678,2345,234,56,6,0,0,0,34;
   …
endextra;
```

[使用例]

```
sectortable  g_a01stbase;
makemargintable  sectortable(g_a01stbase)  margintable(g_a02mgtbase) … ;
  beginrowsector;
     …
  endrowsector;
  begincolumnsector;
     …
  endcolumnsector;
  beginweight;
     …
  endweight;
  beginextra;
```

```
      …
  endextra;
endmargintable;
q;
```

■マージン処理：購入者価格⇒生産者価格(margindrop)

```
margindrop margintable(mtname) var[…];
----------------------------------
    margintable
      mtname          マージン表定義名
    var               変数名
```

[処理]
マージン表に基づき，マージンを形態別に算出し，購入者価格評価表示から生産者価格評価表示への組み替え処理を行う．

mtname は使用するマージン表定義名であり，makemagintable コマンドを用いて事前に作成されていなければならない．

var[…] は組み替え処理対象の変数であり，購入者価格評価値である必要がある．部門分類付変数の行列として定義されていなければならない．

指定した添字部門について形態別マージンが求められ，それらは列部門ごとに集計され，集計値が該当マージン部門の値になる．

部門の指定は，行部門は自由であるが，列部門は昇順に与えなければならない．行部門も列部門も連続している必要はない．

組み替え処理は指示した変数内で行われ，結果が同変数に得られ，変数の内容が大きく変化することもある．変化するセルは，指定した各列部門内の行部門とマージン部門である．

[使用例]

```
sectortable g_a01stbase;
double ta(g_a01stbase),xx(g_a01stbase);
ta[…]=…;
  …
xx[…]=ta[…];
margindrop  margintable(g_a02mgtbase) xx[{1:250,270:380},1:450];
q;
```

■マージン処理：生産者価格⇒購入者価格(marginback)

```
marginback margintable(mtname) var[…];
-----------------------------------
    mtname              マージン表定義名
    var                 変数名
```

[処理]
マージン表に基づき，生産者価格評価表示から購入者価格評価表示への組み替え処理を行う．また，全マージン部門の値を０にする．

mtname は使用するマージン表定義名であり，makemagintable コマンドを用いて事前に作成されていなければならない．

var[…] は組み替え処理対象の変数であり，生産者価格評価値である必要がある．部門分類付変数の行列として定義されていなければならない．

指定した添字部門について，各セルの組み替え係数(マージン表の購入者価格評価値/生産者価格評価値)を処理対象セルに乗じて求める．また，マージン部門の値は０になる．

部門の指定は，行部門は自由であるが，列部門は昇順に与えなければならない．行部門も列部門も連続している必要はない．

組み替え処理は指示した変数内で行われ，結果が同変数に得られ，変数の内容は大きく変化することもある．変化するセルは，与えた各列部門内の行部門とマージン部門である．

[使用例]

```
sectortable g_a01stbase;
double ta(g_a01stbase),xx(g_a01stbase);
  …
ta[…]=…;
  …
xx[…]=ta[…];
marginback  margintable(g_a02mgtbase) xx[{1:250,270:380},1:450];
q;
```

■部門分類表のプリント(pfst)

```
pfst file(…)
     sectortable(stname)
     psize(nol,noc,nos);
-------------------------------------------------------
     file            出力先ファイル名
     sectortable     部門分類表
       stname          部門分類表定義名
     psize           頁行数と行文字数，区切り行数
       nol             頁行数(20～200，省略時 85)
       noc             行文字数(80～300，省略時 114)
       nos             区切り行数(1～100，省略時 5)
```

[処理]
部門分類表の行部門および列部門のコードと名称のプリントを行う．ここでいうプリントは，プリンタに直接印字することではない．テキストデータ形式で印刷イメージをファイルに出力する機能である．出力ファイルの内容は，テキストエディタでそれを確認することができ，必要に応じて，テキストエディタの印刷機能を用いて，その内容を印刷する．コマンド指示の頁行数での自動改ページが行われる．以下，「プリント」はこの意で用いる．

[出力先ファイル名]
プリント(印刷イメージ)の出力先ファイル名を与える．省略することはできない．出力先フォルダは，常にプロジェクトフォルダであり，他のフォルダに出力することはできない．ファイル名に拡張子を付加してもよい．

[部門分類表定義名]
プリントしたい部門分類表定義名を指定する．ここに与える部門分類表定義名は部門分類表の指定コマンド sectortable を用いて，事前に，指定しておく必要がある．

[頁行数と行文字数，区切り行数]
psize(nol,noc,nos) は頁行数と行文字数および区切り行数を与える．nol は一頁のプリント最大行数であり，表題や空白行もそれに含まれる． noc は一行の最大文字数である．nos は区切り行数であり，この区切り行数ごとに区切りのための空白行が挿入され，表示を見やすくする．省略時は，psize(85,114,5) となる．ただし，nol は 20～200，noc は 80～300，nos は 1～100 の範囲でなければならない．

既述のように，結果の確認と実際の印刷はテキストエディタで行う．印刷したときの見やすさ，作業のしやすさを考慮して，テキストエディタの印刷仕様の設定値に合わせて psize オプションを与える必要がある．

テキストエディタの印刷仕様の設定値を頁行数 81，行文字数 114 と想定した一例は，以下はの通りである．

用紙 A4，印刷フォント MS ゴシック 9 ポイント，ヘッダー・フッターなし
上下余白 13mm，左右余白 13mm，行数は用紙サイズから自動計算

[使用例]

```
sectortable g_a01stbase;  //このプロシージャのファイル名は c_a03pfst.txt である.
pfst file(p_a03pfst.txt) sectortable(g_a01stbase) psize(,,10);
q;
```

■マージン表のプリント(pfmargintable)

```
pfst file(…)
     margintable(mtname)
     title(…)
     rowcode(rcfrom:rcto)
     columncode(ccfrom:ccto)
     psize(nol,noc,nos)
     format(w,d);
```

---

file	出力先ファイル名
margintable	マージン表定義名の指示
mtname	マージン表定義名
title	表題
rowcode	行部門プリント対象範囲の指示
rcfrom	開始行部門コード
rcto	終了行部門コード
columncode	列部門プリント対象範囲の指示
ccfrom	開始列部門コード
ccto	終了列部門コード
psize	頁行数と行文字数，区切り行数
nol	頁行数(20～200，省略時 81)
noc	行文字数(80～300，省略時 114)
nos	区切り行数(1～100，省略時 5)
format	変数値の表示幅と小数表示桁数

w	表示幅(10～12，省略時 10)
d	小数表示桁数(3～w-3，省略時 7)

[処理]
指定した行部門と列部門の各セルについて，マージン表が持つ生産者価格評価表示取引額および形態別マージン額を購入者価格評価表示取引額で除した率がプリントされる．ここでいうプリントは，プリンタに直接印字することではない．テキストデータ形式で印刷イメージをファイルに出力する機能である．出力ファイルの内容は，テキストエディタでそれを確認することができ，必要に応じて，テキストエディタの印刷機能を用いて，その内容を印刷する．コマンド指示の頁行数での自動改ページが行われる．以下，「プリント」はこの意で用いる．

[出力先ファイル名]
プリント(印刷イメージ)の出力先ファイル名を与える．省略することはできない．出力先フォルダは，常にプロジェクトフォルダであり，他のフォルダに出力することはできない．ファイル名に拡張子を付加してもよい．

[マージン表定義名]
mtname はプリント対象とするマージン表定義名であり，makemagintable コマンドで事前に作成されていなければならない．

[表題]
表題は title(…)で与える．各ページの表題となる．空白文字やカンマなどを含むときは，その全体を二重引用符(″ ″)でくくる．プリントされる表題文字数は行文字数の制限をうける．表題の指示が省略されたときは，当該コマンドの第１行目の内容がそのままプリントされる．

[開始行部門コードと終了行部門コード]
rowcode(rcfrom:rcto)はプリント対象とする行部門の範囲を与える．rcfrom はその開始であり，rcto はその終了である．その間をコロンで結ぶ．rcfrom のみの指示も可能である．

[開始列部門コードと終了列部門コード]
column(ccfrom:ccto)はプリント対象とする列部門の範囲を与える．ccfrom はその開始であり，ccto はその終了である．その間をコロンで結ぶ．ccfrom のみの指示も可能である．

[頁行数と行文字数，区切り行数]
psize(nol,noc,nos) は頁行数と行文字数および区切り行数を与える．nol は一頁のプリント最大行数であり，表題や空白行もそれに含まれる． noc は一行の最大文字数である．nos は区切り行数であり，この区切り行数ごとに区切りのための空白行が挿入され，表示を見

やすくする．省略時は，psize(81,114,5) となる．ただし，nol は 20～200，noc は 80～300，nos は 1～100 の範囲でなければならない．

既述のように，結果の確認と実際の印刷はテキストエディタで行う．印刷したときの見やすさ，作業のしやすさを考慮して，テキストエディタの印刷仕様の設定値に合わせて psize オプションを与える必要がある．

テキストエディタの印刷仕様の設定値を頁行数 81，行文字数 114 と想定した一例は，以下はの通りである．

> 用紙 A4，印刷フォント MS ゴシック 9 ポイント，ヘッダー・フッターなし
> 上下余白 13mm，左右余白 13mm，行数は用紙サイズから自動計算

[マージン率の表示幅と小数表示桁数]
format(w,d) はマージン率の表示幅 w と小数表示桁数 d を与える．省略時は format(10,7) である．表示幅 w は 10～12，小数表示桁数 d は 3～w-3 でなければならない．

[使用例]

```
pfmargintable  file(g_a01stbase.txt) margintable(g_a02mgtbase)
               rowcode(2110110:2110210) columncode(011011:011021)
               title(“マージン率”) psize(81,114,5) format(10,8);
q;
```

■変数値比較のプリント(pf,pfa)

```
pf file(…)
   title(…)
   st1(…)  st2(…) …
   value1 rop varname1[…] rop value2
   value1 rop varname2[…] rop value2
     …
   value1 rop dif   rop value2
   value1 rop ndif  rop value2
   value1 rop rate  rop value2
   value1 rop nrate rop value2
   psize(nol,noc,nos)
   format(w,d)
   format1(w,d) format2(w,d) …
   rwise  cwise
```

```
   consist  consist2 ;
-------------------------------------------------------------------------
    file                出力先ファイル名
    title               表題
    sti                 副題(i=1,2, …,8)
    varanmei            変数名(i=1,2, …,8)
    dif   ndif          差変数
    rate  nrate         比変数
                        条件式
      rop                 比較演算子(>,>=,<=,<,=,!=)
      value1,value2       スカラー値(定数, スカラー変数)
    psize           頁行数と行文字数, 区切り行数
      nol               頁行数(20～200, 省略時 81)
      noc               行文字数(50～300, 省略時 114)
      nos               区切り行数(1～100, 省略時 5)
    format,formati  変数値の表示幅と小数表示桁数
      w                 表示幅(6～15, 省略時 10)
      d                 小数表示桁数(0～w-3, 省略時 0)
    rwise   cwise       処理方向(rwise:行方向, cwise:列方向, 省略時 cwise)
    consist consist2    整合性確認オプション
```

[処理]

作表内容を確認するための変数値比較のプリントを行う．ここでいうプリントは，プリンタに直接印字することではない．テキストデータ形式での印刷イメージをファイルに出力する機能である．出力ファイルの内容は，テキストエディタでそれを確認することができ，必要に応じて，テキストエディタの印刷機能を用いて，その内容を印刷する．以下，「プリント」はこの意で用いる．

プリントできる内容は，複数の表の値(最大８表，差変数および比変数を含む)，第１変数値と第２変数値の差や比であり，それらの値に条件を与えてプリント対象を限定することもできる．表題や副題を与えることができ，コマンド指示による頁行数での自動改ページも行われる．また，表示を見やすくするために，数行ごとに空白行を挿入することができる．

コマンドには pf，pfa がある．コマンド pf は，複数の変数に条件式が与えられたとき，何れか１つが成立すれば，対象となる全変数値がプリントされる．条件が与えられていない変数は条件成立判定の対象にならない．ただし，条件式が全く指示されていないときは，指定された全体が無条件プリントされる．

コマンド pfa は，複数の変数に条件式が与えられたとき，それらすべての条件が成立したときにのみ対象となる全変数値がプリントされる．条件が与えられていない変数は条件成立判定の対象にならない.

[条件式]
条件式は，以下に述べる変数，差変数(dif,ndif)および比変数(rate,nrate)にプリント対象条件を与える式である．変数の前または後，あるいは前後に比較演算子(rop)と範囲を限定するスカラー値(value1,value2)を与える．スカラー値は定数かスカラー変数の何れかである．比較演算子には，>,>=,<=,<,=,==,!= がある．== は = と同義である．!= は等しくないの意である.

[出力先ファイル名]
プリント(印刷イメージ)の出力先ファイル名を与える．省略することはできない．出力先フォルダは常にプロジェクトフォルダであり，他のフォルダに出力することはできない．ファイル名に拡張子を付加してもよい.

[表題]
表題は title(…)で与える．各ページの表題となる．空白文字やカンマなどを含むときは，その全体を二重引用符(″ ″)でくくる．プリントされる表題文字数は行文字数の制限をうける．表題の指示が省略されたときは，当該コマンドの第１行目の内容がそのままプリントされる.

[副題]
副題は，後述の各変数に対応させ，その内容を示す見出しであり，最多８変数に与えることができる．第１変数に対しては st1(…)で与え，第２変数に対しては st2(…)で与える．同様に st3(…)，…，st8(…)の指示ができる．それぞれに与えることのできる文字数は，後述の表示幅(format(w,d))の w を超えることができない．全角文字を与えてもよい．特殊文字や空白文字などを含むときは，その全体を二重引用符(″ ″)でくくる．副題の指示が省略されたときは，その部分に当該変数名が入る.

[変数名]
変数は最多８変数(差変数，比変数を含む)まで与えることができ，その値をプリントすることができる．ただし，後述の差変数(dif,ndif)や比変数(rate,nrate)は，第１変数と第２変数から得られる値に限定される．第３変数から第８変数は内容表示のための変数であるが，条件式を付与することは可能である.

第１変数は部門分類付変数でなければならない．他は，部門分類付変数であってもなくてもよい．ただし，第 1 変数以外で部門分類付変数の場合，部門分類表は第１変数と同じでなければならない．変数はベクトルか行列でなければならない．また，それら全変数の次元は一致していなければならない.

プリントされる部門コードおよび部門名称は，第１変数が持つ部門分類表が用いられる．

各変数にはプリント範囲の添字を与えることができる．添字の要素数は全変数が一致していなければならない．第１変数の添字指示は省略することはできない．第２変数以降の添字は省略が可能であり，省略したときは第１変数の添字がそのまま適用される．

[差変数]
差変数は，第１変数と第２変数の値の差を示す変数であり，これには dif と ndif がある．第１変数値を v1 とし，第２変数値を v2 とすれば，dif は|v1-v2|であり，ndif は v1-v2 である．すなわち，前者は変数値差の絶対値であり，後者は単純な変数値差である．

[比変数]
比変数は，第１変数と第２変数の値の比を示す変数であり，rate と nrate がある．第１変数値を v1 とし，第２変数値を v2 とすれば，rate は|v1/v2*100|であり，nrate は v1/v2*100 である．すなわち，前者は変数値比を 100 倍した絶対値であり，後者は単に変数値比を 100 倍した値である．ただし，v1!=0,v2=0 のときの 比変数の値は 9999,v1=0,V2=0 のときは 0 となる．

[頁行数と行文字数，区切り行数]
psize(nol,noc,nos) は頁行数と行文字数および区切り行数を与える．nol は一頁のプリント最大行数であり，表題や空白行もそれに含まれる． noc は一行の最大文字数である．nos は区切り行数であり，この区切り行数ごとに区切りのための空白行が挿入され，表示を見やすくする．省略時は，psize(81,114,5) となる．ただし，nol は 20～200，noc は 50～300，nos は 1～100 の範囲でなければならない．

既述のように，結果の確認と実際の印刷はテキストエディタで行う．印刷したときの見やすさ，作業のしやすさを考慮して，テキストエディタの印刷仕様の設定値に合わせて psize オプションを与える必要がある．

テキストエディタの印刷仕様の設定値を頁行数 81，行文字数 114 と想定した一例は，以下はの通りである．

> 用紙 A4，印刷フォント MS ゴシック 9 ポイント，ヘッダー・フッターなし
> 上下余白 13mm，左右余白 13mm，行数は用紙サイズから自動計算

[変数値の表示幅と小数表示桁数]
format(w,d)，format*i*(w,d) は変数値の表示幅 w と小数表示桁数 d を与える．表示幅 w は前の項目との間隔を考慮する必要がある．表示幅 w は 6～15，小数表示桁数 d は 0～w-3 でなければならない．format*i* は，対応する変数に対し個別に与えるものであり，format

は個別に与えられない変数に対し適用されるものである. 省略したときは format(10,0) となる.

[処理方向]
変数値の処理方向を行方向 rwise とするか, 列方向 cwise とするかを指示する. 省略したときは列方向となる.

[整合性確認オプション]
整合性確認の指示には, consist と consist2 がある. それぞれは, 第１変数値を v1 とし, 第２変数値を v2 としたとき, 次の条件の何れかが成立したときにプリントさせる指示である. != は等しくないの意である.

consist	v1*v2<0; v1!=0,v2=0; v1=0,v2!=0
consist2	v1*v2<0; v1!=0,v2=0

同時に他の変数に与えられた条件は, この条件とは独立に動作する.

[使用例]

```
sectortable g_a01stbase;
       …
get ta(g_a03ta),tb(g_a03tb);     // 変数 ta,tb は自動的に定義される.
      …
pf ta[1:500,1:400] tb dif rate>150  file(p_a12pfout1.txt) ;
q;

       …
pfa file(p_a12pfout2.txt) ta[1:500,1:400]>0.01 tb dif 100>rate>150
   title("A表とB表の差") st1("A表")  st2("B表");
q;

       …
pf file(p_a12pfout3.txt) ta[1:500,1:400] tb tc td dif consist
   title("A表とB表の整合性, C表, D表")
   st1("A表") st2("B表") st3("C表") st4("D表");
q;
          …
pf file(p_a12pfout4.txt) te[1:500,1:400] tf ndif>0
```

■変数値順位のプリント(pfr)

```
pfr file(…)
    title(…)
    st1(…)  st2(…) …
    varname1[…]  varname2[…]  …
    rank(r)
    coverage(c)
    psize(nol,noc,nos)
    format(w,d)
    format1(w,d) format2(w,d) …
    rwise  cwise
    nsum   asum ;
```

---

file	出力先ファイル名
title	表題
st*i*	副題(*i*=1,…,4)
varanme*i*	変数名(*i*=1,…,4)
rank	上位部門数
r	上位部門数(産出(あるいは投入)比率の大きい順位)
coverage	累積比率
c	産出(あるいは投入)累積比率(%, 0～100)
psize	頁行数と行文字数, 区切り行数
nol	頁行数(20～200, 省略時 81)
noc	行文字数(50～300, 省略時 114)
nos	区切り行数(1～100, 省略時 5)
format,format*i*	変数値の表示幅と小数表示桁数
w	表示幅(6～15, 省略時 10)
d	小数表示桁数(0～w-3, 省略時 0)
rwise　cwise	処理方向(rwise:行方向, cwise:列方向, 省略時 cwise)
nsum　asum	合計値(nsum:通常集計, asum:絶対値集計, 省略時 nsum)

[処理]

作表内容を確認するため, 変数値の産出(あるいは投入)比率の順位によるプリントを行う機能である. ここでいうプリントは, プリンタに直接印字することではない. テキストデータ形式で印刷イメージをファイルに出力する機能である. 出力ファイルの内容は, テキストエディタでそれを確認することができ, 必要に応じて, テキストエディタの印刷機能を用いて, その内容を印刷する. 以下, 「プリント」はこの意で用いる.

プリントできる内容は，第１変数については変数値，産出(あるいは投入)比率およびその累積であり，第２～4 変数については変数値，産出(あるいは投入)比率，その累積と順位である．比率が高い上位部門数や累積比率を条件として与えプリント対象を限定することができる．表題や副題を与えることができ，コマンド指示による頁行数での自動改ページが行われる．また，表示を見やすくするために，数行ごとに空白行を挿入することができる．

[出力先ファイル名]
プリント(印刷イメージ)の出力先ファイル名を与える．省略することはできない．出力先フォルダは常にプロジェクトフォルダであり，他のフォルダに出力することはできない．ファイル名に拡張子を付加してもよい．

[表題]
表題は title(…)で与える．各ページの表題となる．空白文字やカンマなどを含むときは，その全体を二重引用符(″ ″)でくくる．プリントされる表題文字数は行文字数の制限をうける．表題の指示が省略されたときは，当該コマンドの第１行目の内容がそのままプリントされる．

[副題]
副題は，後述の各変数に対応させ，その内容を示す見出しであり，最多４変数に与えることができる．第１変数に対しては st1(…)で与え，第２変数に対しては st2(…)，第３変数に対しては st3(…)，第４変数に対しては st4(…)で与える．それぞれに与えることのできる文字数は，後述の表示幅(format(w,d))を w とすれば，第１変数は w+12 字以内，第２変数以降は w+18 字以内である．全角文字を与えてもよい．特殊文字や空白文字を含むときは，その全体を二重引用符(″ ″)でくくる．副題の指示が省略されたときは，その部分に当該変数名が入る．

[変数名]
変数は最多４変数まで与えることができ，その値をプリントすることができる．第１変数は部門分類付変数でなければならない．他は，部門分類付変数であってもなくてもよい．ただし，第 1 変数以外で部門分類付変数の場合，部門分類表は第１変数と同じでなければならない．変数はベクトルか行列でなければならない．また，それら全変数の次元は一致していなければならない．

各変数にはプリント範囲の添字を与えることができる．添字の要素数は全変数が一致していなければならない．第１変数の添字指示は省略することはできない．第２変数以降の添字は省略が可能であり，省略したときは第１変数の添字がそのまま適用される．

第１変数の産出(あるいは投入)比率の大きい部門順にプリントされる．第２変数以降は，それ自身が持つ部門順位でなく，第１変数の部門順にプリントされる．部門コードおよび部門名称は第１変数が持つ部門分類表が用いられる．

比率は，与えたプリント範囲の値の合計値を分母にして求められる．与える範囲によって比率は違うことになるので，比率の見方には注意が必要である．また，合計値(後述)には通常集計と絶対値集計がある．

プリントできる内容は，第１変数については変数値，産出(あるいは投入)比率およびその累積であり，第２変数以降については変数値，産出(あるいは投入)比率，その累積と順位である．第２変数以降の産出(あるいは投入)比率順位は第１変数と違うことがあるので，第２変数以降には順位が付けられる．

変数名は，既述のように最多４変数まで与えることができるが，与える変数の数が多くなるとプリントされる行文字数が長くなる．その長さがプリントできるように，使用する用紙の大きさや文字の大きさを考慮しなければならない．後述 psize オプションの行文字数を参照のこと．

[上位部門数]
上位部門数 r は産出(あるいは投入)比率の大きい順にプリントさせたい部門数である．
rank(20) とすれば，産出(あるいは投入)比率が大きい上位 20 部門がプリントされる．

[累積比率]
プリントは第１変数の産出(あるいは投入)比率の大きい順に行われる．このとき，産出(あるいは投入)比率の累積が，この累積比率 c (パーセント)までの部門がプリントされる．
coverage(60) とすれば，累積比率が 60% までの部門がプリントされることになる．

[頁行数と行文字数，区切り行数]
psize(nol,noc,nos) は頁行数と行文字数および区切り行数を与える．nol は一頁のプリント最大行数であり，表題や空白行もそれに含まれる． noc は一行の最大文字数である．nos は区切り行数であり，この区切り行数ごとに区切りのための空白行が挿入され，表示を見やすくする．省略時は，psize(81,114,5) となる．ただし，nol は 20～200，noc は 50～300，nos は 1～100 の範囲でなければならない．

既述のように，結果の確認と実際の印刷はテキストエディタで行う．印刷したときの見やすさ，作業のしやすさを考慮して，テキストエディタの印刷仕様の設定値に合わせて psize オプションを与える必要がある．

テキストエディタの印刷仕様の設定値を頁行数 81，行文字数 114 と想定した一例は，以下はの通りである．

用紙 A4，印刷フォント MS ゴシック 9 ポイント，ヘッダー・フッターなし
上下余白 13mm，左右余白 13mm，行数は用紙サイズから自動計算

[変数値の表示幅と小数表示桁数]

format(w,d), format*i*(w,d) は変数値の表示幅 w と小数表示桁数 d を与える. 表示幅 w は前の項目との間隔を考慮する必要がある. 表示幅 w は 6～15, 小数表示桁数 d は 0～w-3 でなければならない. format*i*は, 対応する変数に対し個別に与えるものであり, format は個別に与えられない変数に対し適用されるものである. 省略したときは format(10,0) となる.

[処理方向]

変数値の処理方向を行方向 rwise とするか, 列方向 cwise とするかを指示する. 省略したときは列方向となる.

[合計値]

比率は, 与えたプリント範囲の値の合計値を分母にして求められる. この合計値の算出には, 値を単純に集計する通常集計 nsum と, すべての値を絶対値で集計する絶対値集計 asum がある. 正負の値が混在する部門は, 絶対値集計が便利であると考えられる. 省略時は通常集計である.

[使用例]

```
sectortable g_a01stbase;
       …
get ta(g_a07ta),tb(g_a08tb);      // 変数 ta,tb は自動的に定義される.
      …
pfr file(p_a12pfout1.txt) ta[1:500,1:400] tb rank(20) coverage(70) ;
      …
pfr file(p_a12pfout2.txt) ta[1:500,1:400] tb rank(10) coverage(50) rwise
   title("A表とB表の産出比率") st1("A表")  st2("B表");
      …
pfr file(p_a12pfout3.txt) ta[1:500,1:400] tb cwise asum
    title("A表とB表の投入比率") st1("A表")  st2("B表");
q;
```

■報告書のプリント(pfreport)

```
pfreport file(…)
         header(…)
         footer(…)
         firstpage(…)
         value1 rop varname[…] rop value2
         rwise  cwise
         option1(norows,nocols)
         option2(item1_ind,item1_psp,item1_loc,item1_plb,item1_pla,
                 item2_ind,item2_psp,item2_loc,item2_bln,item2_pln)
         scmark(sc,newsc)
         format(w,d) ;
```

---

file	出力先ファイル名
header	ヘッダー(省略時空白)
footer	フッター(省略時空白)
firstpage	先頭頁番号(省略時 1)
varname	変数名
	条件式
rop	比較演算子(>,>=,<=,<,=,!=)
value1,value2	スカラー値(定数, スカラー変数)
rwise　cwise	処理方向(rwise:行方向, cwise:列方向, 省略時 cwise)
option1	オプション１
norows	頁行数(5～500, 省略時 112)
nocols	頁列数(1～20, 省略時 4)
option2	オプション２
item1_ind	第１項目字下げ文字数(0～20, 省略時 1)
item1_psp	第１項目挿入空白文字数(0～10, 省略時 3)
item1_loc	第１項目名称表示文字数(4～60, 省略時 16)
item1_plb	第１項目前挿入空白行数(0～20, 省略時 2)
item1_pla	第１項目後挿入空白行数(0～20, 省略時 1)
item2_ind	第２項目字下げ文字数(0～20, 省略時 3)
item2_psp	第２項目挿入空白文字数(0～10, 省略時 2)
item2_loc	第２項目名称表示文字数(4～60, 省略時 14)
item2_bln	第２項目区切り単位行数(0～100, 省略時 10)
item2_pln	第２項目区切り挿入空白行数(0～10, 省略時 1)
scmark	特殊コードオプション
sc	特殊コード区切り識別文字
newsc	特殊コード置換文字

format	変数値の表示幅と小数表示桁数
w	表示幅(6～15, 省略時 10)
d	小数表示桁数(0～w-3, 省略時 0)

[処理]
作表結果を報告書形式でプリントする．ここでいうプリントは，プリンタに直接印字することではない．プリントイメージをファイルに出力する機能である．報告書は，ここで作成された出力ファイルを「報告書作成 EXCEL マクロ(VBA)」(以下 EXCEL マクロ)で印刷することを前提としている．以下，「プリント」はこの意で用いる．

[出力先ファイル名]
プリント(印刷イメージ)の出力ファイル名を与える．省略することはできない．出力先フォルダは，常にプロジェクトフォルダであり，他のフォルダに出力することはできない．ファイル名の拡張子は .csv とする．

[ヘッダー]
ヘッダーは header(…)で与える．報告書の識別に用いる．このヘッダーは各頁の左上に表示される．表示位置やフォントとその大きさなどは，EXCEL マクロおよび"FRAME"シート(報告書のプロトタイプ)で自由に変更することができる．ヘッダーの指示が省略されたときは空白となる．

[フッター]
フッターは footer(…)で与える．報告書の識別に用いる．このフッターは各頁の右下に表示される．表示位置やフォントや大きさなどは，EXCEL マクロおよび"FRAME"シート(報告書のプロトタイプ)で自由に変更することができる．フッターの指示が省略されたときは空白となる．

[先頭頁番号]
先頭頁番号は，報告書の先頭頁に表示される頁番号である．省略されたときは１となる．本コマンド pfreport で出力されるファイル内の実頁もこの先頭頁番号から始まる．たとえば，先頭頁の表示を「11」としたいとき firstpage(11) とする．EXCEL マクロを実行すると，最初にプリントの開始頁と終了頁の入力指示が現れ，指示した範囲の報告書がプリントされる．これは分割印刷を可能にするための機能である．

[変数名]
プリントしたい変数の変数名を与える．変数は行列またはベクトルを対象としている．また，ここで指定する変数は部門分類付変数でなければならない．行部門コードや列部門コードを見やすくするための中区切"-"などは，報告書専用部門分類表の定義を事前に行い，準備をしておかなければならない．

[条件式]
条件式はプリント対象の条件を与え，その条件が成立した値のみをプリントさせようにするものである．変数の前または後，あるいは前後に比較演算子(rop)と範囲を限定するスカラー値(value1,value2)で与える．スカラー値は定数かスカラー変数の何れかである．比較演算子には，>,>=,<=,<,=,==,!= がある．== は = と同義である．!= は等しくないの意である．

[処理方向]
行列変数におけるプリントの処理方向を指定する。行方向 rwise と列方向 cwise がある．列方向のときは，第１項目(見出し部門表示)が列部門，第２項目(数値表示)が行部門となる．行方向のときは，第１項目は行部門，第２項目は列部門となる．省略したときは列方向となる．以下のオプション２を参照．

[オプション１：頁行数，頁列数]
option1(norows,nocols) は，報告書の各頁のプリント行数と列数を指定する．norows は頁行数であり，nocols は頁列数である．頁行数は 5～500(省略時 112)，頁列数は 1～20(省略時 4)の範囲で与える．

[オプション２：挿入空白文字数および行数]
option2(…)はプリント時に挿入する空白文字数や空白行数を指定する．行列変数のプリントに関しては第１項目および第２項目のオプションが有効であり，ベクトル変数に関しては第２項目のみが有効である．以下は，報告書の一部である．

```
┌──────────────────────────────┬──
│  2319-01  ゴム製履物                  │ ①第１項目(見出し部門表示)
│                                       │ ②
│    2032-029  その他の環式中間物    6  │ ③第２項目(数値表示)
│    2033-011  合成ゴム            900  │
│      …                                │
│    9700-000  国内生産額          54626 │ ④
│                                       │ ⑤
│  2319-02  プラスチック製履物          │ ⑥第１項目(見出し部門表示)
│                                       │ ⑦
│    0116-092  生ゴム(輸入)          5  │ ⑧
│      …                                │
│                                       │
```

第１項目 item1 は見出し部門表示であり(上例①⑥)，この項目に関しては５種のオプションがある．以下の文字数はすべて半角文字幅であり，全角文字幅は半角文字幅の２倍である．item1_ind は，直前縦罫線と部門コードの間に置く空白文字数であり，0～20 の範囲で与える(省略時 1)．item1_psp は，部門コードと部門名称の間に置く空白文字数であり，0～10 の範囲で与える(省略時 3)．item1_loc は，部門名称の表示文字数であり，4～60 の範囲で与える(省略時 16)．item1_plb は，第２項目と第１項目の間(上例⑤)に挿入する空白行数であり，0～20 の範囲で与える(省略時 2)．item1_pla は，第１項目と第２項目の間(上例②⑦)に挿入する空白行数であり，0～20 の範囲で与える(省略時 1)．

第２項目 item2 は数値表示であり(上例③④⑧)，この項目に関しても５種のオプションがある．以下の文字数はすべて半角文字幅であり，全角文字幅は半角文字幅の２倍である．item2_ind は，直前縦罫線と部門コードの間に置く空白文字数であり，0～20 の範囲で与える(省略時 3)．item2_psp は，部門コードと部門名称の間に置く空白文字数であり，0～10 の範囲で与える(省略時 2)．item2_loc は，部門名称の表示文字数であり，4～60 の範囲で与える(省略時 14)．item2_bln と item2_pln は関連しており，連続する第２項目の数値表示を何行かおきに(区切り単位行数)，空白行(空白行数)を入れ見やすくするための機能である．item2_bln は区切り単位行数であり，0～100 の範囲で与える(省略時 10)．また，item2_pln は，その区切りとして挿入する空白行数であり，0～10 の範囲で与える(省略時 1)．

[特殊コードオプション]
特殊コードは，屑・副産物・各種マージンに関する情報コードであり，行部門コードに付与されている．特殊コードを表象する基本分類での報告書作成には注意を要する．処理方向が cwise のときは，数値表示となる第２項目が行部門となり，特殊コードはそこに付されているため，何ら問題は生じない．しかし，処理方向 rwise のときは行部門コードに付与されている特殊コードを数値表示となる第２項目の列部門コードに付け替えなければならない．このため特殊コードを識別するための工夫が必要になってくる．さらに，部門コードを見やすくするための中区切"-"などがあるときは，これとの見分けも必要になる．

以下は，報告書用の部門分類表作成の一部である．この例を用いて説明を行う．

```
makesectortable   st_base;
beginrowsector;
  …
"5212-021",  "廃棄物処理（産業）";
"6111-011",  "卸売";
"6111-011#6","卸売";
"6112-011",  "小売";
"6112-011#6","小売";
"6211-011",  "公的金融（帰属利子）";
  …
"7112-011",  "鉄道貨物輸送";
"7112-011#7","鉄道貨物輸送";
  …
"7122-011",  "道路貨物輸送";
"7122-011#7","道路貨物輸送";
"7131-011","自家輸送（旅客自動車）";
  …
```

この部門コードには中区切"-"が入っている．そして特殊コードの区切りは"#"になっている．この場合の特殊コードオプションは scmark("#","-")とする．特殊コード区切り識別文字"#"に続くコードは特殊コードと認識され，さらにこの記号"#"は，特殊コード置換文字"-"に置き換わる．したがって，報告書では 6111-011#6 は 6111-011-6　となる.

特殊コード区切り識別文字と特殊コード置換文字はそれぞれ半角一文字でなければならない.

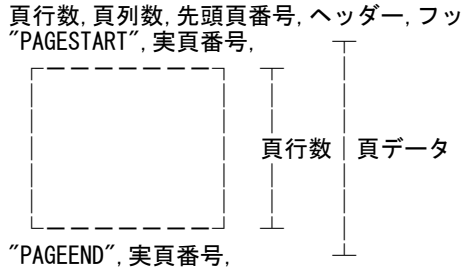
[変数値の表示幅と小数表示桁数]

format(w,d)は変数値の表示幅 w と小数表示桁数 d を与える．表示幅 w は前の項目との間隔を考慮する必要がある．表示幅 w は 6～15，小数表示桁数 d は 0～w-3 でなければならない．省略したときは format(10,0) となる．

小数表示桁数が 0 でないとき，本コマンドの出力ファイルには指示通りの桁数の小数が書き出されているが，このファイルを EXCEL に読み込ませ表示させると，右端の 0 が取り除かれるため小数点位置が不揃いになる．報告書で小数点位置を揃えてプリントさせたいときは，"FRAME"シート上で「セルの書式設定」を用いて小数点以下の桁数を設定する必要がある．

[出力ファイルの形式]

このコマンドで出力されるファイル形式は以下のようになっている．ファイルの第１行目には頁行数，頁列数，先頭頁番号，ヘッダー，フッターが書き出され．その後に頁データが出力される．各頁データの第１行目には"PAGESTART"，最後には"PAGEEND"の文字列が実頁番号とともに出力される．



[報告書作成プロトタイプ adm5_pfreport.xls]

adm5_pfreprot.xls は頁列数４の場合の報告書作成プロトタイプである．この中には，"DATA","OUTPUT","FRAME"の３シートと以下の EXCEL マクロがセットになっている．EXCEL マクロは報告書形式に合わせ必要に応じて変更することは自由である．後述の「報告書の作成手順」を参照．

```
Sub 報告書作成()
'
'  産業連関表 報告書作成マクロ
'
Application.ScreenUpdating = False
'-------------------------------------------------★ファイル名の指定★-----
  Windows("adm5_pfreport.XLS").Activate
'-----------------------------------------------------パラメータ入力-----
  Sheets("DATA").Select
  NOROWS = Cells(1, 1) + 2                          'ページ当りの DATA 行数
  NOCOLS = Cells(1, 2)                              'ページ当りの DATA 列数
  FIRSTPAGE = Cells(1, 3)                                '先頭ページ番号
  HEADER = Cells(1, 4)                                   'ヘッダー文字列
  FOOTER = Cells(1, 5)                                   'フッター文字列
  STARTPAGE = Val(InputBox("印刷開始ページを入力してください．"))
  ENDPAGE = Val(InputBox("印刷終了ページを入力してください．"))
'---------------------------- 各ページごとに表作成・印刷の繰り返し処理-----
  For CPAGE = STARTPAGE To ENDPAGE
     A = (CPAGE - FIRSTPAGE) * NOROWS + 2         '該当ページの先頭行位置
     B = (CPAGE - FIRSTPAGE + 1) * NOROWS + 1     '該当ページの終了行位置
     Sheets("DATA").Select
     If Cells(A, 1) = "PAGESTART" And Cells(A, 2) = CPAGE And _
```

```
        Cells(B, 1) = "PAGEEND" And Cells(B, 2) = CPAGE Then
        '-----------------------------------------------表枠のコピー処理-----
        Sheets("FRAME").Select
        Cells.Select
        Selection.Copy
        Sheets("OUTPUT").Select
        Cells.Select
        ActiveSheet.Paste
        '-----------------------------------------データの貼り付け処理-----
        Sheets("DATA").Select
        Range(Cells(A + 1, 1), Cells(A + NOROWS - 2, NOCOLS * 2)).Select
        Application.CutCopyMode = False
        Selection.Copy
        Sheets("OUTPUT").Select
        Cells(5, 1).Select
        Selection.PasteSpecial Paste:=xlValues, Operation:=xlNone, _
                               SkipBlanks:=False, Transpose:=False
        '-------------------------ページ数とタイトルの付与と貼り付け処理-----
        Cells(1, 1) = HEADER
        Cells(119, 4) = CPAGE
        Cells(119, 8) = FOOTER
        '-----------------------------------------------------印刷処理-----
        Range("A1:H119").Select
        With ActiveSheet.PageSetup
            .LeftHeader = ""
            .CenterHeader = ""
            .RightHeader = ""
            .LeftFooter = ""
            .CenterFooter = ""
            .RightFooter = ""
            .LeftMargin = Application.InchesToPoints(0.68)
            .RightMargin = Application.InchesToPoints(0.43)
            .TopMargin = Application.InchesToPoints(0.63)
            .BottomMargin = Application.InchesToPoints(0.61)
            .HeaderMargin = Application.InchesToPoints(0.512)
            .FooterMargin = Application.InchesToPoints(0.512)
            .PrintHeadings = False
            .PrintGridlines = False
            .PrintComments = xlPrintNoComments
           '.PrintQuality = 300
            .CenterHorizontally = False
            .CenterVertically = False
            .Orientation = xlPortrait
            .Draft = False
            .PaperSize = xlPaperA4
            .FirstPageNumber = xlAutomatic
            .Order = xlDownThenOver
            .BlackAndWhite = False
            .Zoom = False
            .FitToPagesWide = 1
            .FitToPagesTall = 1
        End With
        Selection.PrintOut Copies:=1, Preview:=False, Collate:=True
      Else
        MsgBox ("処理がエラーとなりました. ")
      End If
  Next
  MsgBox ("処理が終了しました. ")
Application.ScreenUpdating = True
End Sub
```

[報告書の作成手順]

① 報告書データの作成(pfreport コマンド)

報告書データを pfreport コマンドで作成する.

②報告書作成プロトタイプファイルのコピー

報告書作成プロトタイプファイル (…¥ADAM¥proclib¥adm5_pfreport.xls) をプロジェクトフォルダにコピーする．コピー後，そのファイル名を管理識別しやすい名前に変更し，EXCEL で立ち上げる．

③報告書データの読み込み

EXCEL を新規に立ち上げ，報告書データ(pfreport の出力ファイル)を読み込む．あるいはこのファイルをダブルクリックする．

④報告書データの貼り付け

③で読み込んだ報告書データのすべてを報告書作成 EXCEL の"DATA"シートに貼り付ける．

⑤報告書作成 EXCEL マクロの変更

②で行ったファイル名の変更処理にともない，EXCEL マクロの該当箇所を変更する必要がある．マクロを開き編集画面で行う．EXCEL マクロの７行目のファイル名“adm5_pfreport.xls” を新しいファイル名に変更する（EXCEL マクロの「★ファイル名の指定★」箇所）．

⑥報告書作成 EXCEL マクロをテスト

報告書作成 EXCEL マクロを実行する．結果は直接プリンタに出力される．実行操作を開始すると，プリントの開始頁と終了頁の入力指示が現れるので数頁のテスト印刷指示を行い，印刷状況を確認する．不具合があれば"FRAME"シートの形式を整え再度実行する．これを繰り返し，印刷内容を精査する．なお，"OUTPUT"シートは印刷イメージを展開する作業用であり，修正など手を加えることはない．

⑦報告書作成 EXCEL マクロを実行

テスト印刷の最終確認を経て，報告書作成 EXCEL マクロを実行する．報告書の頁数が多いときは，開始頁と終了頁の指示を用いて分割印刷を行うとプリンタの長時間占有もなく便利である．

⑧報告書データが多量のとき

④の報告書データが多量で EXCEL に一度に読み込むことができないときは，報告書データをテキストエディタなどで分割して処理を行う必要がある．このとき，分割した報告書データの第１行目の「頁行数，頁列数，先頭頁番号，ヘッダー，フッター」の追加，先頭頁番号の変更などが必要になる．

⑨EXCEL のセキュリティレベル

EXCEL のセキュリティレベルが高いと，マクロを実行することができない．セキュリティレベルを変更する必要がある．その変更は，[ツール]，[オプション]，[セキュリティ]，[マクロセキュリティ]，[セキュリティレベルを中]の手順で行う．

[使用例]

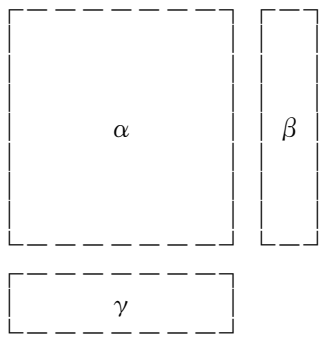
```
sectortable g_a01stbase;
get y(g_a20xij);

pfreport file(p_f40pfreport.csv), header("投入表") footer("基本－投入"),
        y[1:556,1:465];
```

```
q;

sectortable g_a01stbase;
get a(g_a24jaij);

pfreport file(p_f41pfreport.csv), header("投入係数表") footer("基本－投入")
         rwise, firstpage(101), a[1:366,1:277]>0.01, format(,6)
         option1(112,4)  option2(1,3,16,2,1,3,2,1,10,1);
q;
```

■バランス調整(balance)

```
balance(α, β, γ)
```

α	初期行列
β	列ベクトル(制約条件, 行合計(産出額計))
γ	行ベクトル(制約条件, 列合計(投入額計))



[処理]
作表の最終段階において生じる縦横(行列)のバランス調整を解析的に行う機能である．ラグランジュの未定乗数法を用いている．バランス調整は，基本的には諸データの基で慎重に進めるべきものである．しかし，最後まで完全手作業を貫くことは膨大な時間を必要とする．差異が小さくなった段階で，本機能を使用することは現実的な選択であるといえよう．

[ラグランジュの未定乗数法(KEO-RAS 法)]
行列のバランス調整(matrix balancing)は，基準となる行列および新たな行合計と列合計を与え，両者の合計に一致するように，新たな行列を推定する問題である．これには，繰り返し収束計算による RAS 法がよく知られている．しかし，この方法は時には収束しないこともある．

ADAM が導入したラグランジュの未定乗数法は，解析的なバランス調整法である．これは，与えた基準となる行列から得られる縦比率・横比率と，与えた新たな行合計と列合計(以下，制約条件値)に基づいて推計される新行列から得られる新縦比率・新横比率について，それぞれの加重残差自乗和を最小化させるように新行列を決定する方法である．そして，それぞれの加重を縦比率および横比率の自乗の逆数にしたものが KEO-RAS である．この方法は，延長表推計で長く使用されてきた．(注)KEO：Keio Economic Observatory，慶應義塾大学産業研究所の英文名称である．

[balance 関数の使用法]
まず，使用例を示そう．

```
bm[1:510,1:470]=balance(mx[1:510,1:470],vo[1:510],trans(vi[1:470]));
```

mx[…]はバランスがとれていない表である．vo[…]，vi[…]は，それぞれ産出生産額(行和)および投入生産額(列和)の制約条件値である．バランス調整後の値は，bm[…]に得られる．産出生産額の合計と投入生産額の合計は，当然であるが一致していなければならない．

指示した表の 0 でないセルは，すべてバランス調整の対象になる．したがって，確度が高くバランス調整の対象にしたくないセル値は,退避保存を行った上でそのセル値を 0 にし，さらにそのセル値を該当する制約条件値の産出生産額および投入生産額から差し引く．そして，バランス調整後，これらの退避保存したセル値を元の位置に戻し，処理を完了する．

このバランス調整機能の使用に当たっては留意すべきことがある．移輸入競争型において，中間需要や最終需要のすべてが移輸入で賄われ，当該部門の生産額が 0 のケース，および屑・副産物部門のように発生と投入が相殺し，生産額(行和)が定義的に 0 になるケースである．このときは，新横比率が定義できないので解くことができない．このため，ADAM では制約条件値が 0 の部門は，その行(あるいは列)の全要素を自動的に 0 にし，調整の対象にしていない． したがって，これらの部門の推計値は事前に退避保存しておき，バランス調整後，退避保存した値を元の位置に戻すようにしなければならない．生産額が 0 でなくても，その値が極めて小さく，産出額や移輸入額が大きい部門も同様に扱った方がよい．

産出生産額と投入生産額の制約条件値が共に整数変数であれば，調整後の全セル値は四捨五入が行われ整数値化され，その上で，その表の産出合計および投入合計が制約条件値に合うように端数調整処理が自動的に行われる．ただし，調整未了セルが生じることがあり，そのときは,ADAM 画面にその旨の表示が行われる．生じたとしてもその件数は若干である．それらは手作業修正で調整を図る必要がある．

最近の表は値の桁数が大きくなってきている．特に制約条件値が生産額となる場合はその値が大きく，整数値での処理には十分な注意が必要になってきた．これまでは，PC のハードウェアー特性もあり，整数値は約 9 桁(32 ビット，-2147483648～2147483647)の処理が限界であった．その値が 2147483647 を超えたとき，値が負値に一転してしまう．この上限値より 1 大きい 2147483648 は， PC 内では -2147483648 (負値)になっていしまう．

今回の ADAM(version5)はこの点を考慮し，前述「◇変数」のように，整数はすべて約 19 桁(64 ビット)の範囲の値の処理できるようにした.

前述のように，ADAM が導入したラグランジュの未定乗数法は解析的なバランス調整法である．バランス調整の対象となる表は負値も含まれており，行和あるいは列和を小さくしてしまうこともある．時には，それが調整に悪い影響を与えることもある．調整前と調整後の表の差異を確認するなど，調整後の結果は常に十分な精査をすることが肝要である.

[使用例]

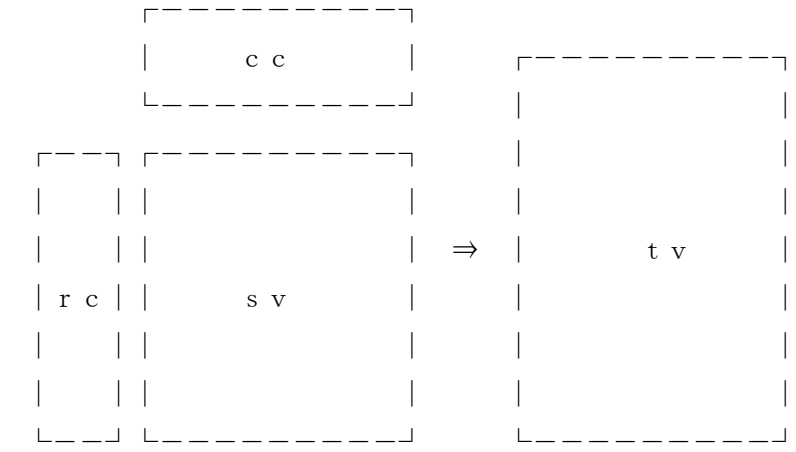
```
sectortable g_a01stbase;
double bm(g_a01stbase),mx(g_a01stbase);
int vo(g_a01stbase,r),vi(g_a01stbase,c);
  …
bm[1:510,1:470]=balance(mx[1:510,1:470],vo[1:510],trans(vi[1:470]));
  …
q;
```

■データ移動(move)

```
move sv[…]  rc  cc  tv;
```

---

sv	原データ変数名(行列変数)
rc	行部門コード変数名(文字列ベクトル変数)
cc	列部門コード変数名(文字列ベクトル変数)
tv	移動先データ変数名(部門分類付変数)



[処理]

原データ変数 sv の行と列には，行部門コード rc と列部門コード cc が対応しているものと想定する(下図参照)．シートデータ入力で読み込みを行えば結果はこのようになる．一方，部門分類付変数 tv がある．原データの行部門コードと列部門コードに基づいて，その行列データ値を部門分類付変数の対応する行部門・列部門の位置に移す．コマンドに与える変数の順序は上述の通りでなければならない．

[原データ変数]

原データ変数は行列変数でかつ数値変数(整数変数，浮動小数点変数)でなければならない．変数の後に移動対象の範囲を限定する添字を与える．例えば，sv[1:200,{1:5,8,10}]とする．

[行部門コード変数・列部門コード変数]

ともに文字列ベクトル変数であり，原データ変数に対応しているものとする．また，そのコード体系は,目的からして,移動先データ変数が持つ部門分類表に準拠したものでなければならない．移動対象の範囲を限定する添字は与えない．すべて原データ変数に与えた範囲に自動連動する．

[移動先データ変数]

移動先データ変数は部門分類表付行列変数でかつ数値変数(整数変数,浮動小数点変数)でなければならない．移動対象の範囲を限定する添字は与えてはいけない．原データの行部門コード・列部門コードに基づいて，移動先データ変数の該当位置が検索され移動が行われる．原データ変数が浮動小数点変数で移動先データ変数が整数変数であるとき小数はすべて切り捨てられるので注意を要する．四捨五入などの処理が必要ならば，事前に処理をしておかなければならない．

[使用例]

```
move sd[1:200,{1:20,23:28}],  rc,  cc,  td;
```

トピックス

◇有用な手法

■データ変換(標準⇒CSV)

[例題の内容]
ADAM の標準データ形式ファイルからデータを読み込み, CSV データ形式でファイルに書き出す. 行部門名称, 列部門名称もファイから読み込み, 同ファイルに書き出す.

```
//
//  標準形式データ ⇒ CSV 形式データ
//
def indatafile   io95.dat
def inrnamefile  rname.txt
def incnamefile  cname.txt
def outdatafile  out.csv

def sz  120
def nor index(rowcode,9700)
def noc index(colcode,9700)

double  x[@sz,@sz];
string  rowcode[@sz],rowname[@sz];
string  colcode[@sz],colname[@sz];

x[1:@sz,1:@sz]=0;
input  datafile(@indatafile) fieldsep(,)
       #1(1) #2(2) (3,rowcode[#1]) (4,colcode[#2]) (5,x[#1,#2]);

input  datafile(@inrnamefile) fieldsep(,)
       #1(1,index(rowcode)) (2,rowname[#1]);

input  datafile(@incnamefile) fieldsep(,)
       #1(1,index(colcode)) (2,colname[#1]);

outsheet datafile(@outdatafile) fieldsep(,)
  #1(1:@nor) #2(1:@noc) (rowcode[#1]) (rowname[#1])
                        (colcode[#2]) (colname[#2])  (x[#1,#2],5);
q;
```

■データ変換(CSV⇒標準)

[例題の内容]
ADAM の CSV データ形式ファイルからデータを読み込み，標準データ形式でファイルに書き出す．行部門名称，列部門名称もファイルに書き出す．

```
//
//  CSV 形式データ ⇒ 標準形式データ
//
def sz  120
def nor 73
def noc 97

def indatafile    infile.csv
def outdatafile   io95.dat
def outrnamefile  rname.txt
def outcnamefile  cname.txt

double  x[@sz,@sz];
string  rowcode[@sz],rowname[@sz];
string  colcode[@sz],colname[@sz];

insheet datafile(@indatafile) fieldsep(,)
  #1(1:@nor) #2(1:@noc) (rowcode[#1]) (rowname[#1])
                        (colcode[#2]) (colname[#2])  (x[#1,#2]);

output  datafile(@outdatafile) fieldsep(,)
       #1(1,1:@nor) #2(2,1:@noc) (3,rowcode[#1]) (4,colcode[#2]) (5,x[#1,#2]);

output  datafile(@outrnamefile) fieldsep(,)
       #1(,1:@nor) (1,rowcode[#1]) (2,rowname[#1]);

output  datafile(@outcnamefile) fieldsep(,)
       #1(,1:@noc) (1,colcode[#1]) (2,colname[#1]);
q;
```

◇インストールと起動方法

■動作環境

Windows
Internet Explorer 6.0 以上
ADAM オンラインマニュアル使用のため.
インストールされていないときは, …¥ADAM¥manual¥index.html を
直接クリックすれば, マニュアルを見ることができる.

■インストール

□ADAM version5 以前のバージョンを利用している環境では, 新バージョンとの錯綜を防止するため, 次の処理を行う必要がある.

1. これまで分析に用いていたデータおよびプロシージャを確実に待避保存する.
2. デスクトップ上のアイコンを削除する.
3. インストール済のフォルダ「ADAM」および関連ファイルを他の場所に移動する.

□ADAM バージョン 6 のインストール

1. ダウンロードしたファイルを解凍する.
   setup.exe と ADAMSetup.msi の２ファイルが展開される.
2. setup.exe を起動する.
   Windows インストーラと .NET Framework が自動的にダウンロードされる(初回のみ), 引き続き ADAM のインストールが開始される. 途中, フォルダの選択画面が表示されるので, インストール場所を指示する(…¥ADAM¥ とする).
3. ADAM ショートカットの作成.
   インストール先 ADAM フォルダの ADAM.exe を㊨クリック⇒[ショートカットの作成]を選択する. 作成された[ADAM へのショートカット]をデスクトップに移動する.
4. ADAM フォルダ中の構成は次の通り.

```
ADAM
  ├datalib          データライブラリーフォルダ
  ├manual           マニュアルフォルダ
  ├proclib          プロシージャライブラリーフォルダ
  ├sample           プロジェクトフォルダ(サンプル)
  ├──ADAM.exe       プログラム
  ├──Config.inf     ユーザ登録情報(利用を開始すると作成される)
  ├──microsoft.office.interop.excell.dll   システム dll
  ├──Microsoft.Vbe.Interop.dll             システム dll
```

```
├──office.dll                                システム dll
├──useid.txt       ユーザ ID(利用を開始すると作成される)
```

□ADAM バージョン 6 の再インストール(更新)

1. ダウンロードしたファイルを解凍する.
   setup.exe と ADAMSetup.msi の２ファイルが展開される.
2. setup.exe を起動する.
   既存 ADAM に対して修復/削除の画面が表示される. 削除を指示する.
   終了後, 再度 setup.exe を起動する. 途中, フォルダの選択画面が表示されるので, インストール場所を指示する.

■起動方法

1. デスクトップ上の[ADAM へのショートカット]をダブルクリックする.
2. 「実行ファイル名」の枠に, 事前に編集した実行ファイル名を指定する.
   横にある[参照]ボタンをクリックし, ファイルダイアログで選択をすれば簡単である. 実行済みファイル名はコンボボックスに記録されるので, 項目リストから実行ファイルを選択し再実行させることができる.
   その実行ファイルが格納されているフォルダがプロジェクトフォルダと認識され, それが枠の上に表示される.
3. 必要があれば, データフォルダとプロシージャフォルダの指定をフォルダ設定画面で行い, 領域設定画面で領域の設定を行う.
4. 上の指定が完了したならば, [実行]ボタンをクリックする (または, [Enter]キーを押す).
5. 処理コマンドにエラーがあれば, そこで処理が止まる.
   テキストエディタを呼び出し, コマンドのエラー箇所を修正した上で「上書き保存」を行い (テキストエディタ画面は閉じない方がよい), 再度 ADAM 画面で実行する. この繰り返しで, 所期の目的を達成する.
6. コンピュータ画面には, ADAM 画面とテキストエディタ画面を併存させた方が, 作業効率が上がる.

緊急処置

ADAM プログラムが動作しなくなったときは, 緊急処置として,
[Ctrl]キーと[Alt]キーと[Deletel]キーを同時に押し, ADAM プログラムの動作を停止させる.

◇Version 5 の変更点

■Version 5.0.1 の変更点 [2008.7.1]

[整数値の有効桁数を拡張]

整数値の有効桁数が約 19 桁になった(以前のバージョンは約 9 桁).

[各種の設定方法の変更]

データライブラリ，プロシージャライブラリの指定方法が変更になった．また，変数の領域設定指定方法も変更になった.

[unit()関数の追加]

単位行列の指示を簡単にするために unit()関数が追加された.

[複数データ一括代入の指示方法の変更]

代入文の右辺に複数データを与えるとき，それらのデータを中括弧でくくるように変更された.

[for 文コマンドの導入]

for 文の使用を可能にした．また，continue 文 と break 文の使用も可能にした.

[インクリメント・デクリメントの後置演算子の導入]

インクリメントとデクリメント後置演算子(++,--)の使用を可能にした.

[日付変数の取消]

日付変数が使用できなくなった.

[文字列連結指示の変更]

文字列が長く複数行に続くときの指示方法が変更になった.

[マージン表のプリントコマンド(pfmargintable)の追加]

マージン表の内容をプリントするコマンドが追加された.

[Excel ファイルからの読み込みコマンド(inexcel)の追加]

コマンド inexcel を追加し，Excel ファイルからの読み込みを可能にした.

[Excel ファイルへの書き出しコマンド(outexcel)の追加]

コマンド outexcel を追加し，Excel ファイルへの書き出しを可能にした.

[Skyline コマンドの変更]

スカイライン図の描画に新たなモデルが追加された.

[UserID の変更]

UserID が変更になった．以前の UserID は使用できなくなった.

◇Version 6 の変更点

■Version 6 の変更点 [2015.3.30]

[UserID・パスワードを廃止]

ユーザーコードやパスワードによる管理をすべて取りやめた.

[Excel ファイルからの読み込みコマンド(inexcel) 書き出しコマンド(outexcel)の廃止]

Excel ファイルからの読み込みコマンドと書き出しコマンドを廃止した.

◇地域内分析

■統合データ確認プロシージャ　　adm105_dc

統合データについて次の事項の確認を行う.

・各列の和が生産額(投入)と一致しているか

・各行の和が生産額(産出)と一致しているか

・生産額の投入と産出が一致しているか

もし，一致していないときは，その状況が表示される.

詳細は，使用方法は ¥ADAM¥proclib¥adm105_dc.adm 内の使用法を参照.

■生産誘発分析基本処理プロシージャ　　adm106_ipbase

与えられた情報から次の値を求める.

モデルは，X=inv([I-(I-M)A])×(I-M)F である.

・投入係数行列

・輸入係数

・レオンチェフ逆行列

・影響力係数

・感応度係数

・最終需要項目別生産誘発額

詳細は，…¥ADAM¥proclib¥adm106_ipbase.adm 内の使用法を参照.

■生産誘発分析Ⅰ型プロシージャ　　adm106_ip1

生産誘発分析処理Ⅰ型プロシージャ（一括需要型).

需要ベクトルを一括型で与える.

モデルは，X=inv([I-(I-M)A])×(I-M)F である.

プロシージャ adm106_ipbase で求めた値を用いて行うので，このプロシージャを事前に呼び出す必要がある.

第１次生産誘発額(第１次波及効果)，第２次生産誘発額(第２次波及効果)を求めることができる．詳細は，…¥ADAM¥proclib¥adm106_ip1.adm 内の使用法を参照.

■生産誘発分析Ⅱ型プロシージャ　　adm106_ip2

生産誘発分析処理Ⅱ型プロシージャ（独自推計需要型).

需要額ベクトルには，直接需要額ベクトルと独自推計需要額ベクトルの２種類を与える.

モデルは，X=inv([I-(I-M)A])×(I-M)F である.

プロシージャ adm106_ipbase で求めた値を用いて行うので，このプロシージャを事前に呼び出す必要がある.

第１次生産誘発額(第１次波及効果)，第２次生産誘発額(第２次波及効果)を求めることができる．詳細は，…¥ADAM¥proclib¥adm106_ip2.adm 内の使用法を参照.

■スカイライン作図プロシージャ　　adm107_skyline

与えられた情報からスカイライン図を表示する.

処理できる表の大きさは最大 120 部門までである.

詳細は, ¥ADAM¥proclib¥adm107_skyline.adm 内の使用法を参照.

◇２地域間分析

■統合データ確認プロシージャ　　adm205_dc

統合データについて次の事項の確認を行う.

・各列の和が生産額(投入)と一致しているか

・各行の和が生産額(産出)と一致しているか

・生産額の投入と産出が一致しているか

もし, 一致していないときは, その状況が表示される.

詳細は, …¥ADAM¥proclib¥adm205_dc.adm 内の使用法を参照.

■生産誘発分析基本処理プロシージャ　　adm206_ipbase

与えられた情報から次の値を求める.

モデルは, X=inv([I-(I-M)A])×(I-M)F である.

・投入係数行列

・輸入係数

・レオンチェフ逆行列

・影響力係数

・感応度係数

・最終需要項目別生産誘発額

詳細は, …¥ADAM¥proclib¥adm206_ipbase.adm 内の使用法を参照.

■生産誘発分析Ｉ型Ａ地域プロシージャ　　adm206_ip1a

生産誘発分析処理Ｉ型Ａ地域プロシージャ（一括需要型).

Ａ地域の需要ベクトルを一括型で与える.

モデルは, X=inv([I-(I-M)A])×(I-M)F である.

プロシージャ adm206_ipbase で求めた値を用いて行うので, このプロシージャを事前に呼び出す必要がある.

第１次生産誘発額(第１次波及効果), 第２次生産誘発額(第２次波及効果)を求めることができる. 詳細は, …¥ADAM¥proclib¥adm206_ip1a.adm 内の使用法を参照.

■生産誘発分析Ｉ型Ｂ地域プロシージャ　　adm206_ip1b

生産誘発分析処理Ｉ型Ｂ地域プロシージャ（一括需要型).

Ｂ地域の需要ベクトルを一括型で与える.

モデルは，X=inv([I-(I-M)A])×(I-M)F である．
プロシージャ adm206_ipbase で求めた値を用いて行うので，このプロシージャを事前に呼び出す必要がある．
第１次生産誘発額(第１次波及効果)，第２次生産誘発額(第２次波及効果)を求めることができる．詳細は，…¥ADAM¥proclib¥adm206_ip1b.adm 内の使用法を参照．

■生産誘発分析Ⅱ型Ａ地域プロシージャ　　adm206_ip2a

生産誘発分析処理Ⅱ型Ａ地域プロシージャ（独自推計需要型）．
需要額ベクトルには，直接需要額ベクトルと独自推計需要額ベクトルの２種類を与える．
モデルは，X=inv([I-(I-M)A])×(I-M)F である．
プロシージャ adm206_ipbase で求めた値を用いて行うので，このプロシージャを事前に呼び出す必要がある．
第１次生産誘発額(第１次波及効果)，第２次生産誘発額(第２次波及効果)を求めることができる．詳細は，…¥ADAM¥proclib¥adm206_ip2a.adm 内の使用法を参照．

■生産誘発分析Ⅱ型Ｂ地域プロシージャ　　adm206_ip2b

生産誘発分析処理Ⅱ型Ｂ地域プロシージャ（独自推計需要型）．
需要額ベクトルには，直接需要額ベクトルと独自推計需要額ベクトルの２種類を与える．
モデルは，X=inv([I-(I-M)A])×(I-M)F である．
プロシージャ adm206_ipbase で求めた値を用いて行うので，このプロシージャを事前に呼び出す必要がある．
第１次生産誘発額(第１次波及効果)，第２次生産誘発額(第２次波及効果)を求めることができる．詳細は，¥ADAM¥proclib¥adm206_ip2b.adm 内の使用法を参照．