Data Mine Tech Ltd.
Data Bring New Insight to Your Business

2008/4/21

WPS スタートアップガイド（WPS2.2）

目標

WPS(World Programming System)の基本操作方法とプログラミングの基礎を習得します。

目次

1章 基本操作

この章は WPS の基本的使用法の習得を目標としてします。

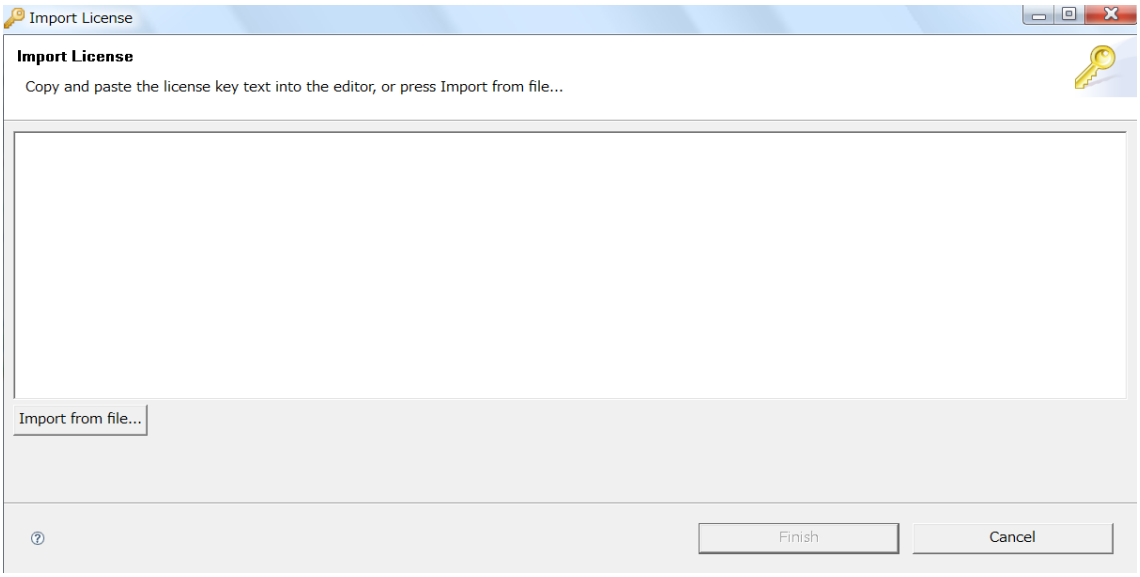
まずは WPS を使ってみましょう。

（１）WPS の起動

インストール時にデスクトップアイコンを作成した場合は、デスクトップアイコンをダブルクリックし、WPS Workbench を起動します。

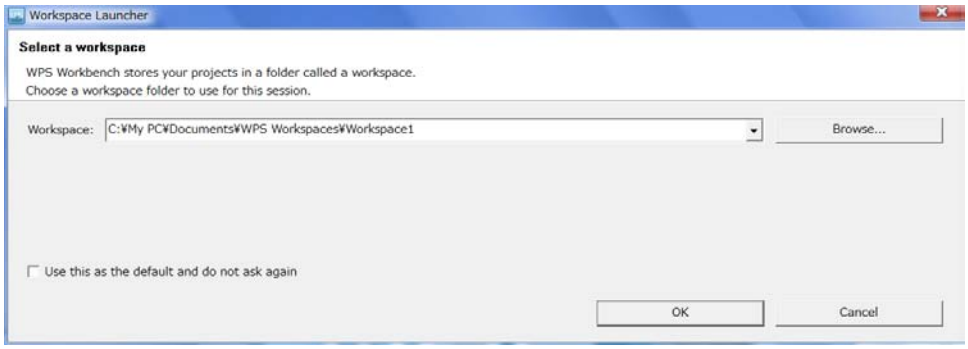
または、Windows スタートメニュー>プログラム>World Programming WPS 2 を選択して WPS Workbench を起動します。

インストール後はじめて WPS を起動した場合は、ライセンス情報の入力要求画面がオープンします。



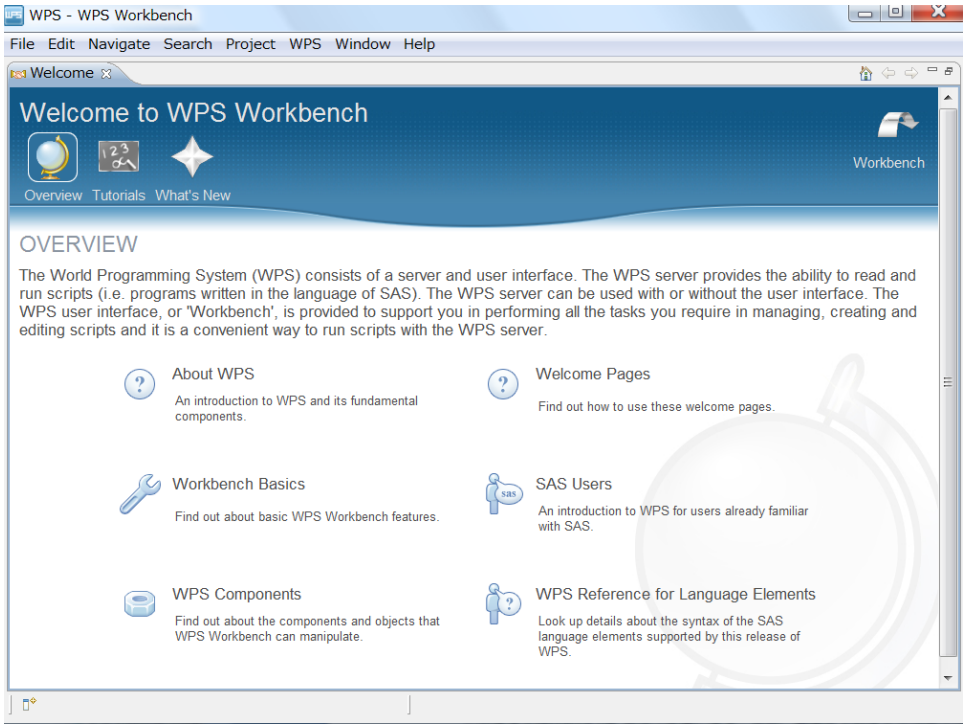
Import from file...からライセンス情報ファイルの場所を参照するか、ライセンステキストをすべて白地内の編集エリアにコピーペーストして Finish ボタンを押します。

次に Workspace の設定場所の確認画面が開きます。Workspace は既定ではマイドキュメントの中に作られます。 WPS では作成するプログラムやデータはプロジェクトと呼ばれる論理的な容器の中で管理しますが、Workspace は複数のプロジェクトを含むことができる物理的な記憶場所を意味しています。



もしも設定場所[1]を別の場所に変更したければBrowse... ボタンを押して変更先のフォルダを指定します。このままで良ければOKボタンを押します。 なお、画面左下のチェックボックスにチェックを入れておくと次回の起動時からこの確認画面はスキップされます。

さて、インストール後最初の起動時は、次の Welcome 画面が必ず出現します。



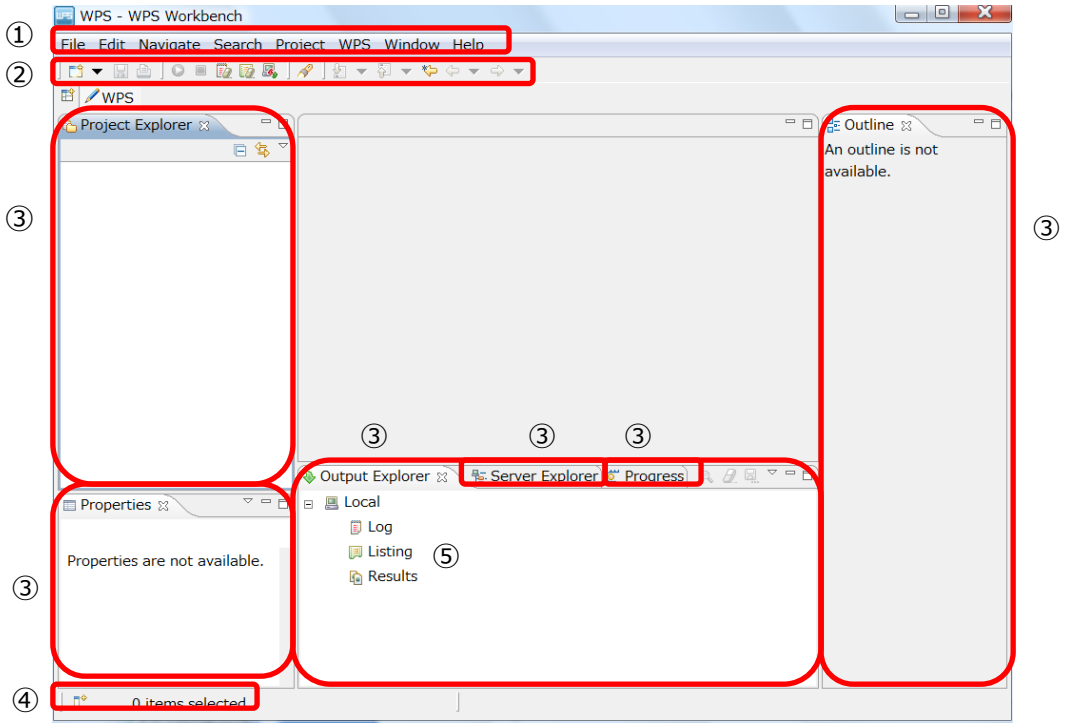
この段階で既に WPS は起動していますが、Welcome 画面は最初の起動時に必ず出現します。 Welcome 画面からは WPS の用語や使い方のヘルプなど、さまざまな情報が得られますのでとても便利ですが、とりあえず見ずにスキップして先に進みましょう。 なお、[別表 1] に Welcome メニューの簡単な説明を掲載しましたので参考にしてください。

[1] Workspace の既定の保存場所は WPS Workbench の Window>Preference...>WPS>Startup Options からも設定可能です。

Welcome画面は、左上のタブを右クリックして出現するリストのCloseを選択するか、タブの×の位置をクリック[2]して閉じます。 なお、この後WPSを一度終了した後、次回以降WPSを起動した場合は、保存された終了時の画面レイアウトが立ち上がり、Welcome画面は出現しません。もしも再度出現させたい場合は、メニューのHelp>Welcomeを選択してください。

（２）WPS 起動画面

Welcome 画面を閉じると、いよいよ WPS の初期起動画面（WPS ワークベンチ）が出現します。



WPS ワークベンチとは、WPS に備っている対話型のプログラム開発実行環境の呼び名です。 ユーザはこの WPS ワークベンチを通じて WPS 本体（WPS サーバと呼ばれます）と対話しながらデータ処理を実行していくことができます。 なお、後述するように、完成したプログラムのバッチ処理実行も可能となっており、その場合は WPS ワークベンチを使いません。 しかし、通常は WPS ワークベンチを使うことになりますので、まずは WPS ワークベンチを理解しておく必要があります。 以下 WPS ワークベンチの各オブジェクトの名前と役割を簡単に説明します。

①［メニュー］

File Edit Navigate Search Project WPS Window Help

WPS にさまざまな指示を与えるためのメニュー方式のコマンドです。

例えば、はじめて WPS を使う場合は、まず新しいプロジェクトを作成しなければなりません。プロジェクトの新規作成は File メニューから New アイテムを選択し、次に Project アイテムを選択します。（長たらしいので、以後、このテキストではメニューから指定を行うことを File>New> Project と表記することに

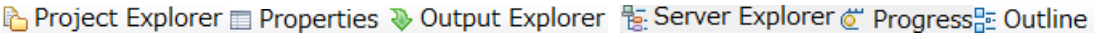
2 このテキストでは一貫して「クリック」とはマウスの「左」クリックを意味し、「右クリック」（マウスの「右」クリック）と区別しています。

します。）　その他、多くのコマンドがメニューから指定できますが、ここに書くと長くなります。 主なコマンドの名前と役割をまとめて［別表 2］に掲載しましたので、確認しておいてください。

② ［ツールバー］

良く使うコマンドをアイコンに割当てて指定しやすくしたものです。
例えば、一番左に配置されているオプション（選択）付きアイコン（ ）は New アイコンと呼ばれメニューの File>New 指定と同じです。各アイコンにマウスカーソルを乗せると割当てられたコマンドの名前が表示されるようになっていますので、確認しておいてください。

③ ［ビュー］



これらのウィンドウは WPS ではビューと呼ばれています。WPS ワークベンチにはさまざまな種類のビューが存在します。 次の（３）実行例の中で、基本的なビューの役割や操作を経験することになりますが、全体のビューの名称と役割を［別表 3］に掲載しましたので、確認しておいてください。
なお、個々のビューは Window>Show View メニューから出現させることができます。

④ ［メッセージ欄］

WPS ワークベンチからのメッセージがここに表示されます。 なお、WPS サーバ（WPS 本体のこと）や Windows からのメッセージはログビューやプロブレムビューに表示されます。

⑤ ［階層オブジェクト］

ビュー、ファイル、スクリプト [3]などの各種オブジェクトを複数配置可能できるビューの場合に、配置されているオブジェクト名が階層的に表示されます。 ビューの種類によって配置されるオブジェクトの種類が異なり、また、オブジェクトごとに、クリック、ダブルクリック、右クリックからのコンテキストメニューなどにより、操作できる内容が異なります。(内容表示、プロパティ表示、名前の変更、削除、階層の展開など)
上図に表示されているアウトプットエクスプローラビューの場合は、WPS からのメッセージ出力（ログ）、画面出力（リスティング）、HTML 形式結果出力（リザルト）といった役割を持つエディタアンドアウトプットビューに属する各アイテムが表示されています。

（３）実行例

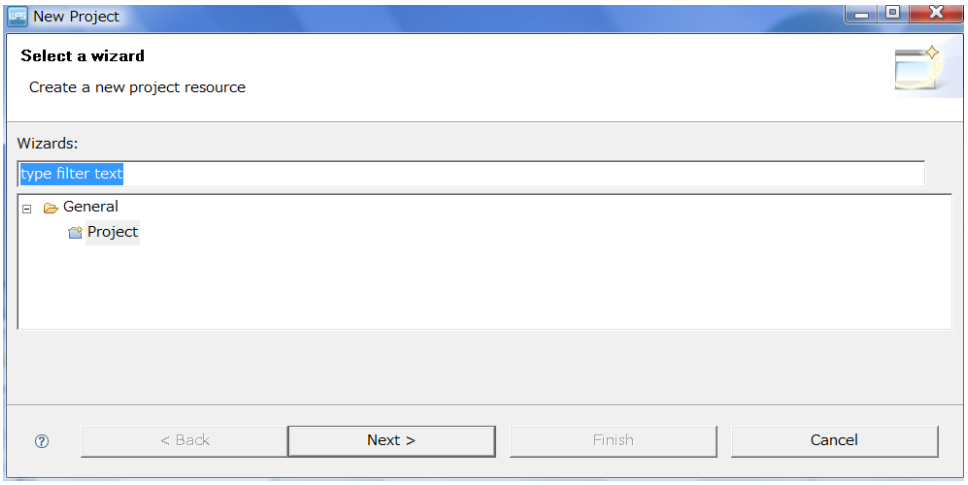
実際に WPS ワークベンチを使ったデータ処理例を体験します。
この例では以下の手順を実行します。

---

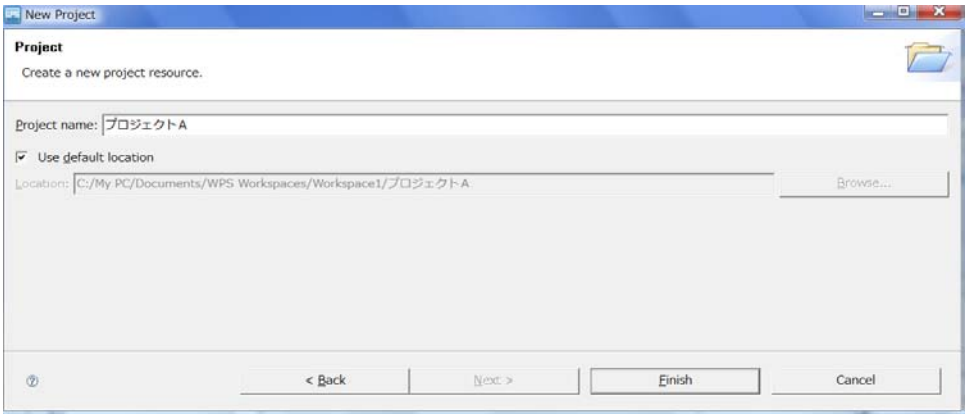
3 SAS 言語で書かれたプログラムを WPS ではスクリプトと呼んでいます。

(Step1) データ処理を行うための事前準備としてプロジェクトを１つ作成します。プロジェクトとは相互に関連する一連のデータやプログラムを管理する WPS の概念です。 これは WPS が管理するユーザファイルのディレクトリのようなものと考えてください。
(Step2) WPS インストレーションファイルに含まれているサンプルデータおよびサンプルスクリプトをプロジェクトに追加します。
(Step3) いくつかのサンプルスクリプトを実行し結果を確認します。

では、実際に行ってみましょう。
(Step1) メニューからFile>New>Projectを選択します。以下のNew Projectウィザードがオープンします。 Type filter text[4]と入力された箇所は無視し、ウィンドウ内のGeneralアイテムの１つ下の階層にあるProjectアイテムを選択してからNext>ボタンを押します。



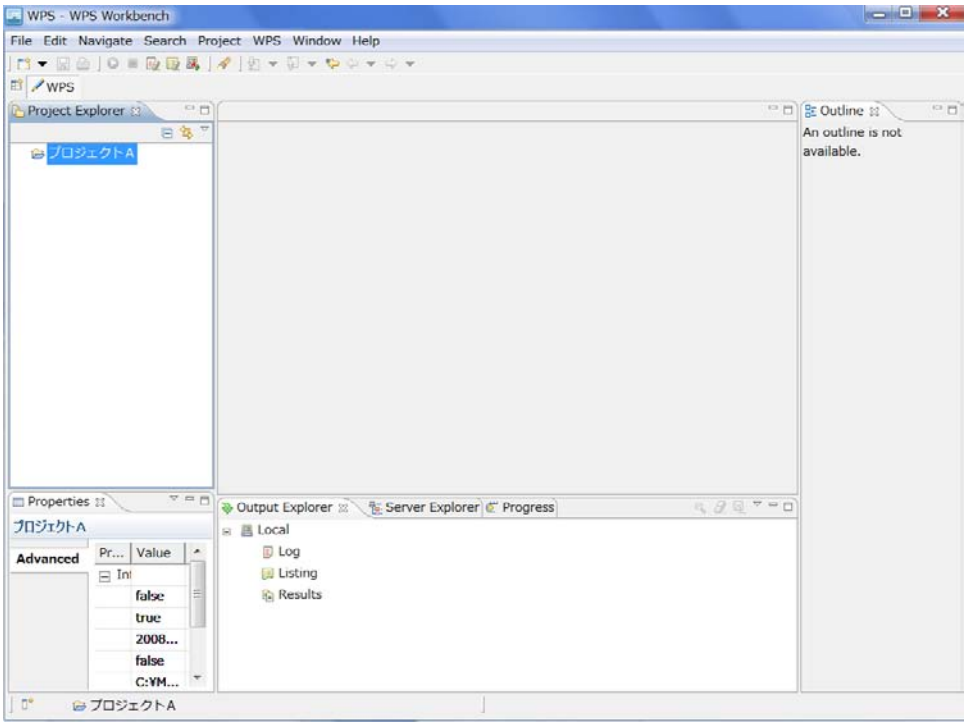
プロジェクト名の入力を要求するウィザードに切り替わります。任意の名前を入力可能ですが、この例ではプロジェクト A と入力し Finish ボタンを押します。



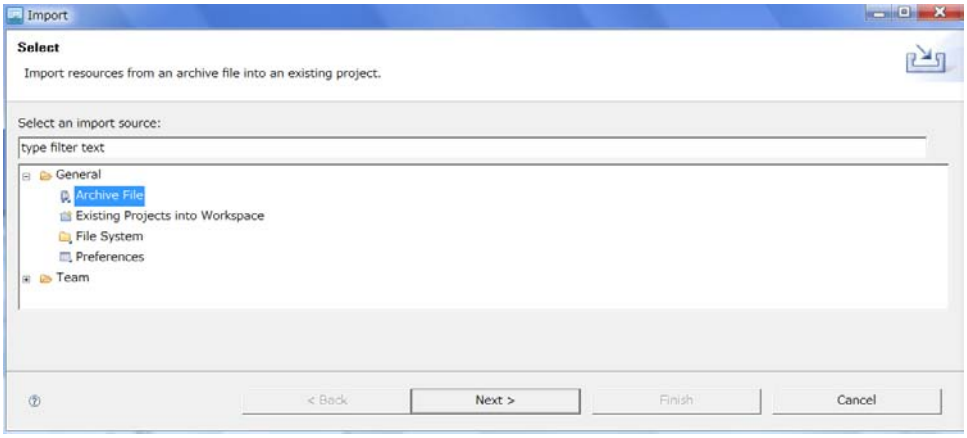
WPS ワークベンチのプロジェクトエクスプローラビュー内にプロジェクト A が出現したことを確認して

4 これは絞り込み検索のための入力欄です。

Step1 は終了です。



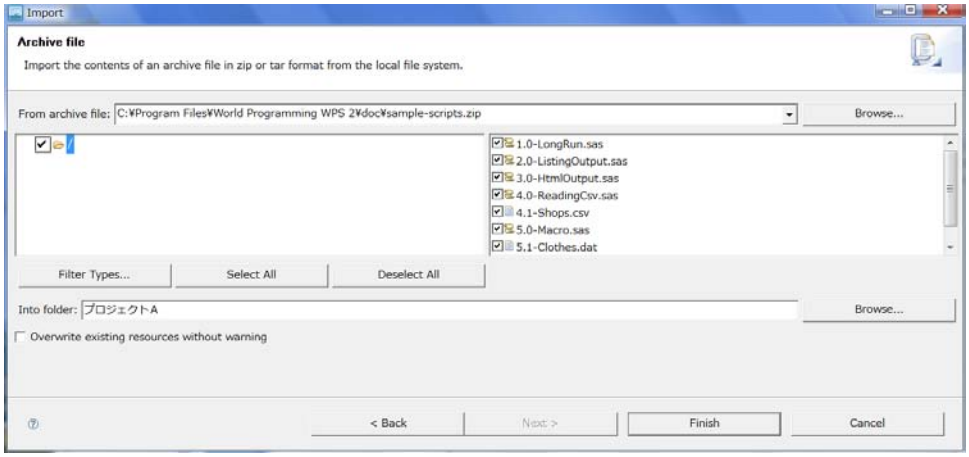
(Step2) プロジェクト A が選択された状態から File>Import またはプロジェクト A を右クリックして出現するコンテキストメニューの Import を選択します。 出現する Import ウィザードのウィンドウ内の General の左にある+を押して下階層を展開して現れるアイテムリストの中から、今回のサンプルデータ形式に合わせて Archive File（圧縮形式ファイル）を選択後 Next>ボタンを押します。



サンプルデータは WPS インストールディレクトリにあります。 Browse ボタンを押して Windows ディレクトリを辿って以下のファイルを選択します。

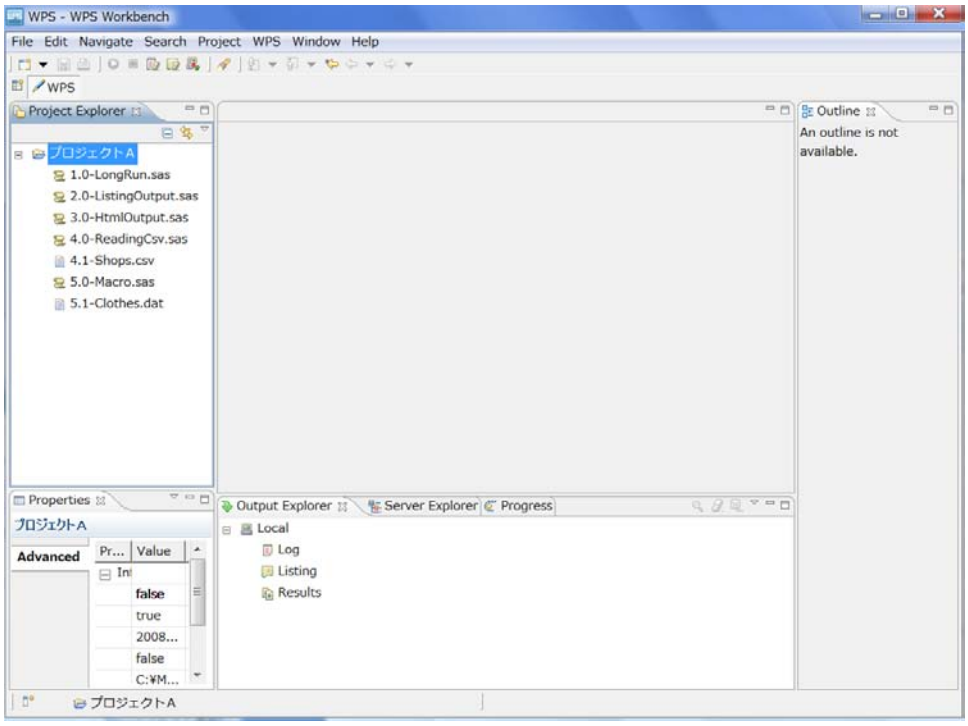
C:¥Program Files¥World Programming WPS 2¥doc¥sample-scripts.zip[5]

5 デフォルトのインストール先の場合です。なお、このテキストではファイル拡張子（ここでは.zip）は表示するように設定されているものとしています。



sample-scripts.zip には 7 個のファイルが圧縮保存されており、既定の設定ではすべて選択された状態になっています。そのまま Finish ボタンを押します。

プロジェクトエクスプローラ内のプロジェクト A を展開すると、その下に 7 個のファイルが加わっていることを確認します。 アイコンが付いた 5 個のファイルはスクリプトで、残り 2 個は入力データです。

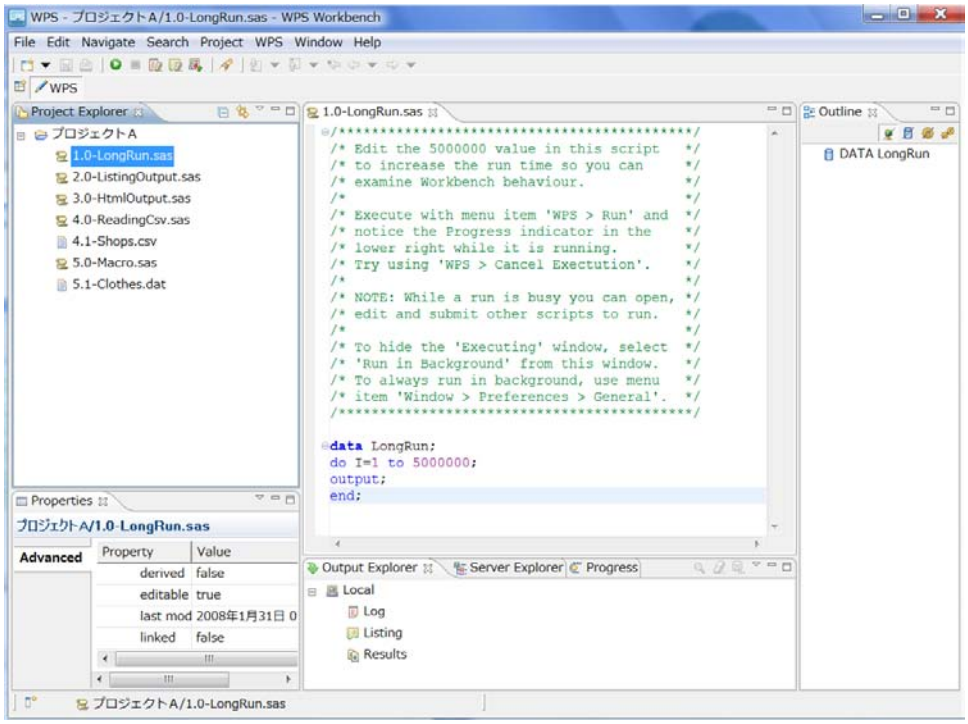


以上で Step2 は完了です。

(Step3) サンプルスクリプトを 1 つずつ実行します。

プロジェクト A 内の各アイテムをダブルクリックするか、右クリックして Open を選択すると、ファイル内容を表示するエディタビューがオープンします。

（ア）1.0-LongRun.sas

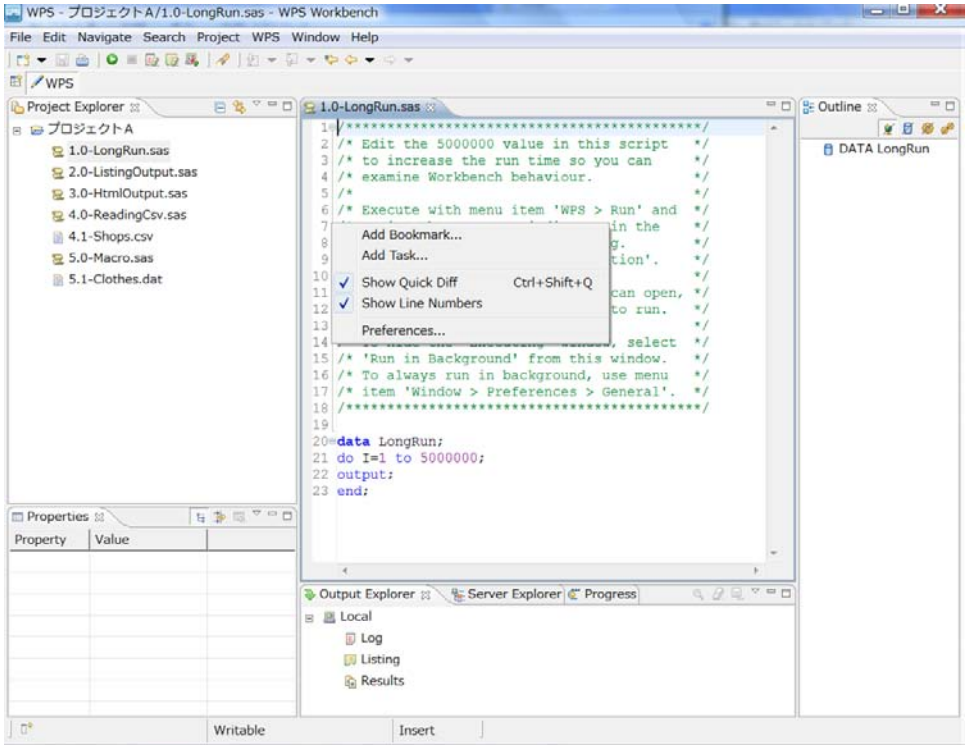


1.0-LongRun.sasの内容が表示されたエディタビューが中央に出現し、同時に右のアウトライン(Outline)ビューにこのスクリプトの構造が表示されます。 アウトラインビューは、このスクリプトがLongRunデータセットを作成する 1 つのDATAステップ [6]のみを含むことを表示しています。

ここで、プログラムコードを参照しやすくするため、エディタビューに行番号を付加しておきましょう。行番号を付加するには、エディタビュー内の左側の白い空白部分にマウスカーソルを当てた状態で右クリックし、出現するコンテキストメニューからShow Line Numbersを選択します。[7]

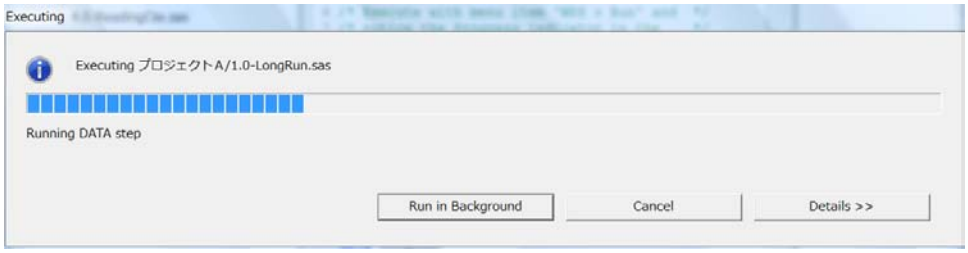
6 Data ステップは後述する SAS 言語の基本構成要素の 1 つであり、PL/1 と良く似たコンピュータ言語仕様を持っています。

7 WPS ワークベンチの詳細な設定は Windows>Preferences･･･から確認や変更が可能です。



さて、行番号が表示されたら、スクリプトの内容を確認してみましょう。 スクリプトは 20 行から 23 行までのわずか 4 行のSAS言語[8]で書かれたプログラムです。 SAS言語については第 2 章でより詳しく説明しますが、ここでは、このスクリプトは 500 万回のDOループ実行を行い、1 から 5000000 まで 1 ずつ増加する変数Iの値を持つ 500 万件のLongRun という名前のWPSデータセット[9]を作成するプログラム（DATAステップコード）だというように理解してください。

スクリプトを実行するにはWPS>Run Fileメニューを選択するか、ツールバーの実行アイコン（  ）を押します。[10] 実行するとWPSサーバから実行中のメッセージウィンドウが表示され、しばらくするとメッセージウィンドウは終了します。

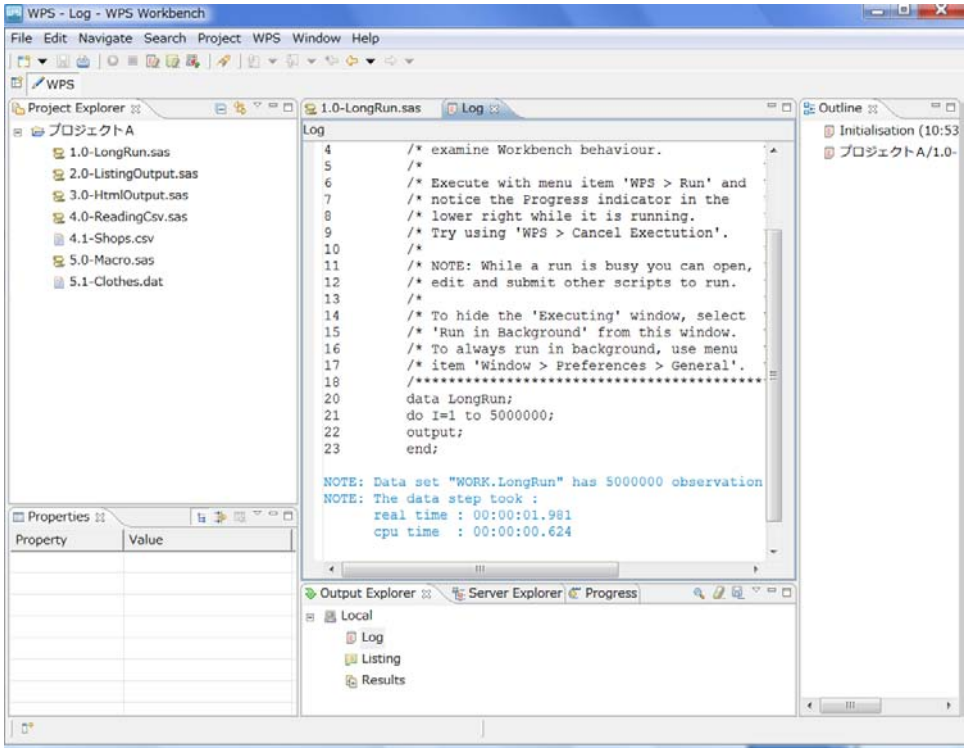


実行がうまくいったかどうかを確認するには、アウトプットエクスプローラ内のログ（Log）アイテムをダブルクリックし、ログビューをオープンします。

8 SAS 言語は米国 SAS Institute Inc.が開発した PL/1 言語に良く似た汎用アプリケーション開発用プログラミング言語です。

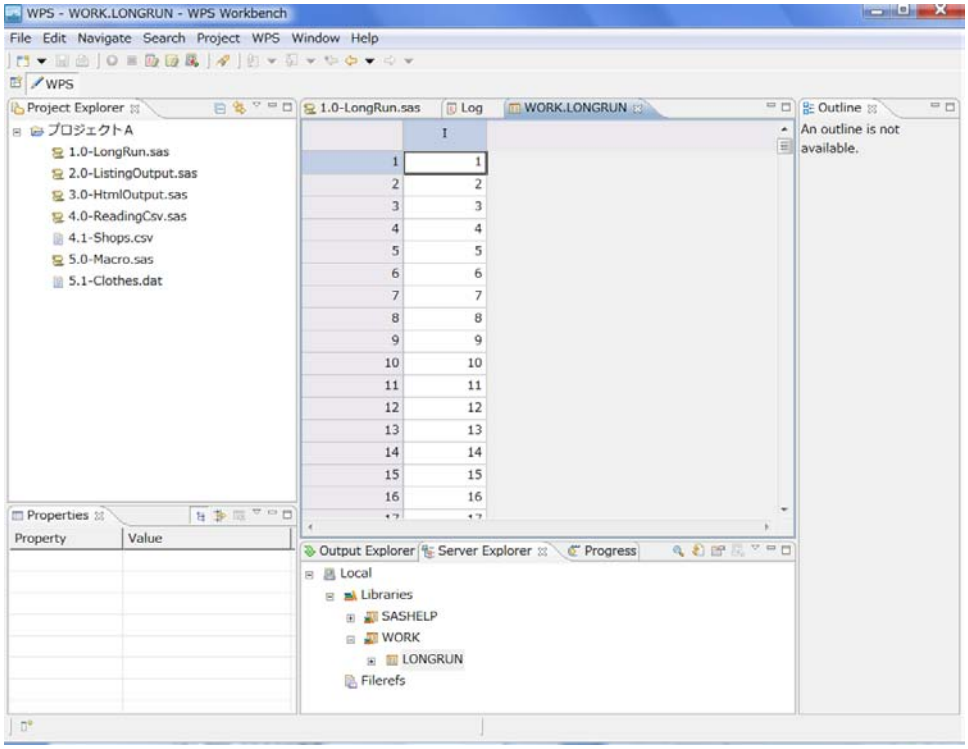
9 WPS データセットとは WPS が内部的に管理するデータセットの 1 つの種類であり、Windows におけるファイル拡張子は wpd です。

10 いずれもスクリプトの全行が一括実行されます。行の一部のみを選択した状態で実行すると、選択状態の行部分のみ実行されます。



ログビューが中央のウィンドウに表示されます。 ログにはエラーは出現しておらず、500 万件のオブザベーション（レコードのこと）を持つ LongRun データセットが作成されたことがノート（Note）表示されています。次に、LongRun データセットの中身を確認してみましょう。 スクリプトの実行結果はリスト出力、データセット出力などですが、これらはサーバエクスプローラ（Server Explorer）ビューにアイテムとして格納されています。

アウトプットエクスプローラタブの右に見えているサーバエクスプローラタブをクリックし、その中の Local>Libraries>WORK>LONGRUN をダブルクリックすると、LongRun データセットの中身を表示するアウトプットビューWORK.LONGRUN がオープンします。

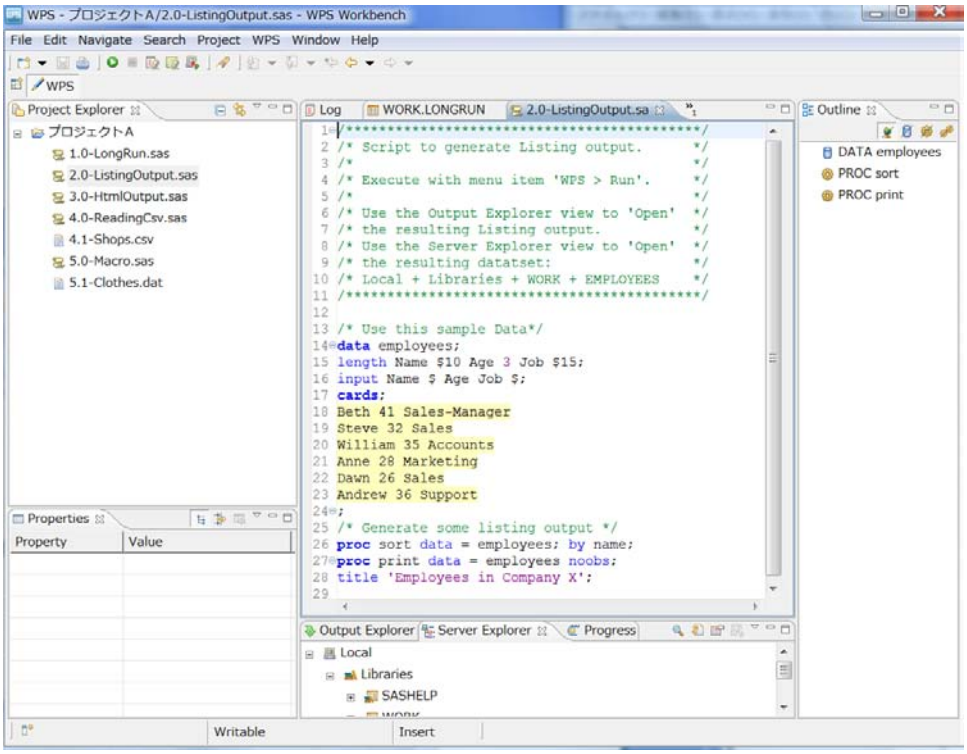


確認したらWORK.LONGRUNビューを閉じます。[11] 以上で最初のスクリプト 1.0-LongRun.sasの実行と結果の確認が完了です。

（イ）2.0-ListingOutput.sas

データセット内容のリスト出力を伴う実行例です。 プロジェクトエクスプローラ内の 2.0-ListingOutput.sas スクリプトをダブルクリックしてオープンします。

11 閉じないで次のスクリプトを実行すると、強制的に閉じるよう指示するウィンドウが出現します。



アウトラインビューを見ると、このスクリプトはemployeesデータセットを作成し、sortプロシジャ[12]とprintプロシジャを実行するものだとわかります。

スクリプトを見ると、14 行から 24 行までの DATA ステップは 18 行から 23 行までのプログラム中に書かれているデータ値を読んで employees データセットを作成するステップとなっています。26 行の sort プロシジャは変数 name のアルファベット順にオブザベーションを並べ替えて同じ名前のデータセットに保存するステップで、27 行の print プロシジャはデータセットの中身をリスト表示するステップです。

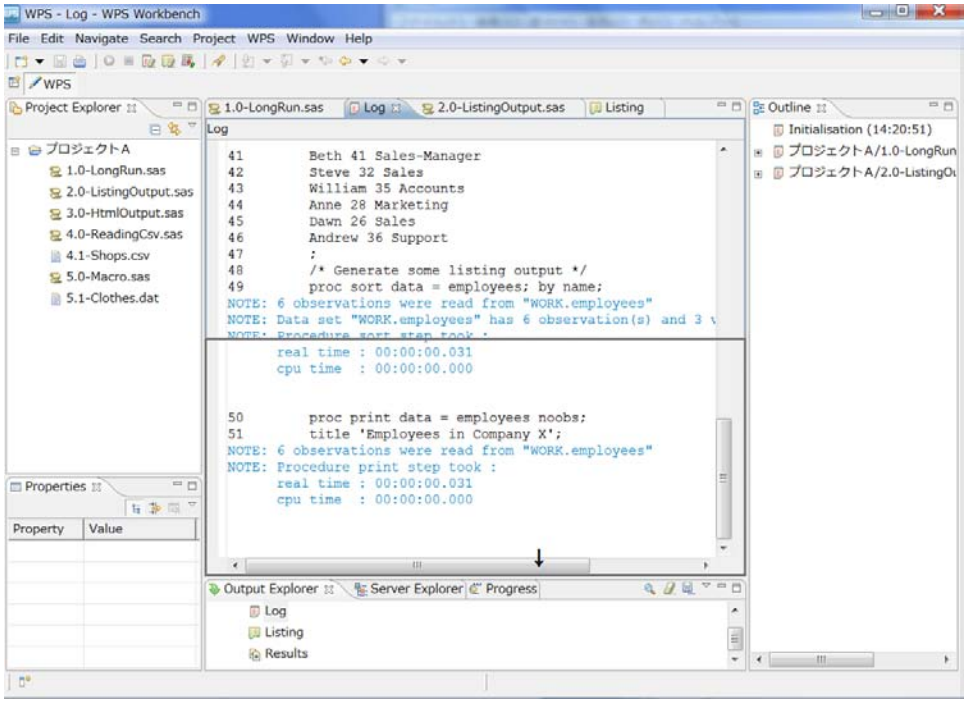
さて、実行アイコン（  ）を押してスクリプトを実行し、実行結果のログとリスト出力を確認してみましょう。

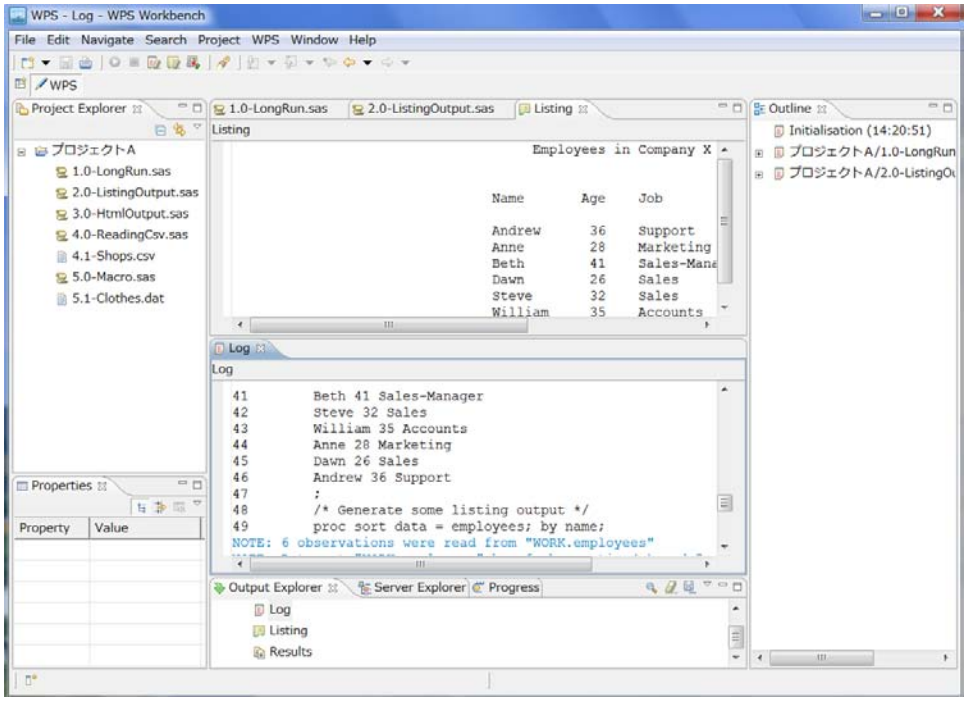
実行後、リスト出力（Listing）ビューがいずれかのウィンドウに出現します。 ここで、WPS ワークベンチのウィンドウ操作の練習のため、WPS ワークベンチの中央上部のウィンドウを 2 分割してログとリスト出力の 2 つのビューを同時に表示するよう調整してみましょう。

Logビューのタブをウィンドウの下の境界線近くまでドラッグします。黒の↓が出現したところでマウスボタンから手を離すとウィンドウが水平に 2 分割されます。[13]

12 プロシジャ（PROC で始まりプロシジャ名が続く）は Data ステップと並ぶ SAS 言語の基本構成要素の 1 つです。処理目的ごとにさまざまなプロシジャが用意されています。プロシジャは Data ステップのようなプログラミング言語仕様ではなくパラメータオプション選択仕様による実行モジュールです。

13 同様にビュータブをウィンドウの左右の端近くにドラッグし、黒の←または→が出現したところで手を離すとウィンドウが縦に 2 分割されます。





これで、ログとリスト出力が同時に見えるようになりました。リスト出力（Listing）を見るとデータセットのオブザベーションは変数 name の昇順に並んでいることがわかります。

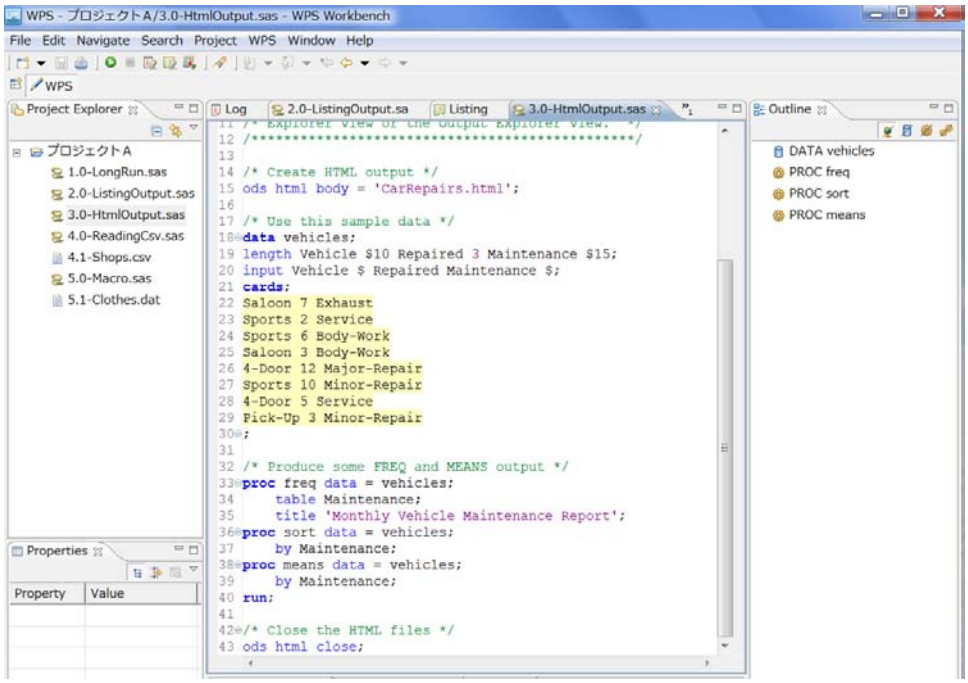
ビュータブをウィンドウ上部のタブ位置までドラッグすることにより同じウィンドウに束ねることができます。 Log ビュータブを元の位置までドラッグすると元に戻ります。

サーバエクスプローラビューから employees データセットができていることを確認してください。 以上で 2.0-ListingOutput.sas の実行と結果の確認は終了です。

（ウ）3.0-HtmlOutput.sas

HTML 形式の結果出力を行う実行例です。

この例に進む前に WPS>Clear Log と WPS>Cleat Listing を選択してログとリスト出力を一旦クリアしておきましょう。スクリプトをオープンします。



この例は 8 件の車の修理データに対して修理個所の集計レポートを作成し html 形式で出力する例を示しています。 ただし、このまま実行すると次のエラーが出現する場合があります。

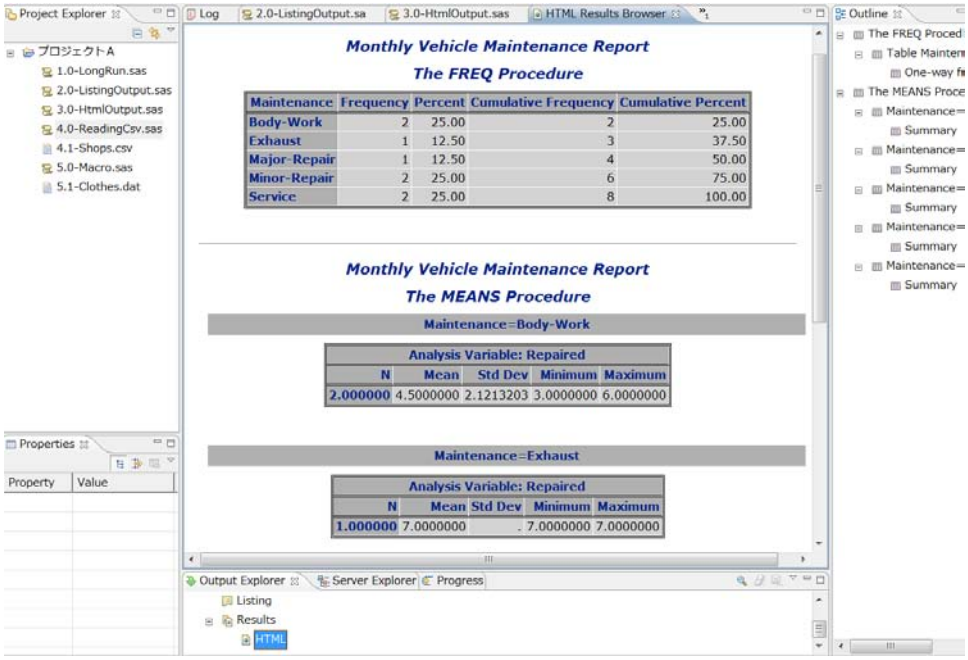
ERROR: Could not open ODS HTML BODY file 'C:¥Windows¥system32¥CarRepairs.html'. HTML output will not be available

これは 15 行目の html ファイルの作成先のデフォルトディレクトリが関係しています。 エラーが発生した場合は、次のようにフルパスで指定するように変更して再実行してください。（ただし、この例では C:¥temp ディレクトリが存在していることを前提としています）

Ods html body = '**C:¥temp¥**CarRepairs.html';[14]

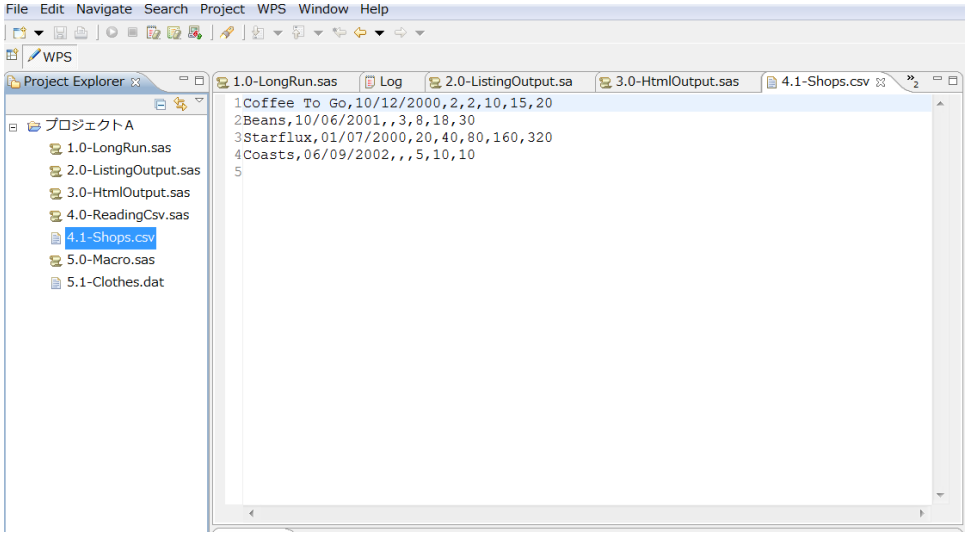
HTML 出力はアウトプットエクスプローラの Result>HTML アイテムをダブルクリックすることにより出現します。

14 ¥記号は¥記号で入力します。ただし、WPS のスクリプトでは¥記号は逆スラッシュ記号に変換されて表示されます。

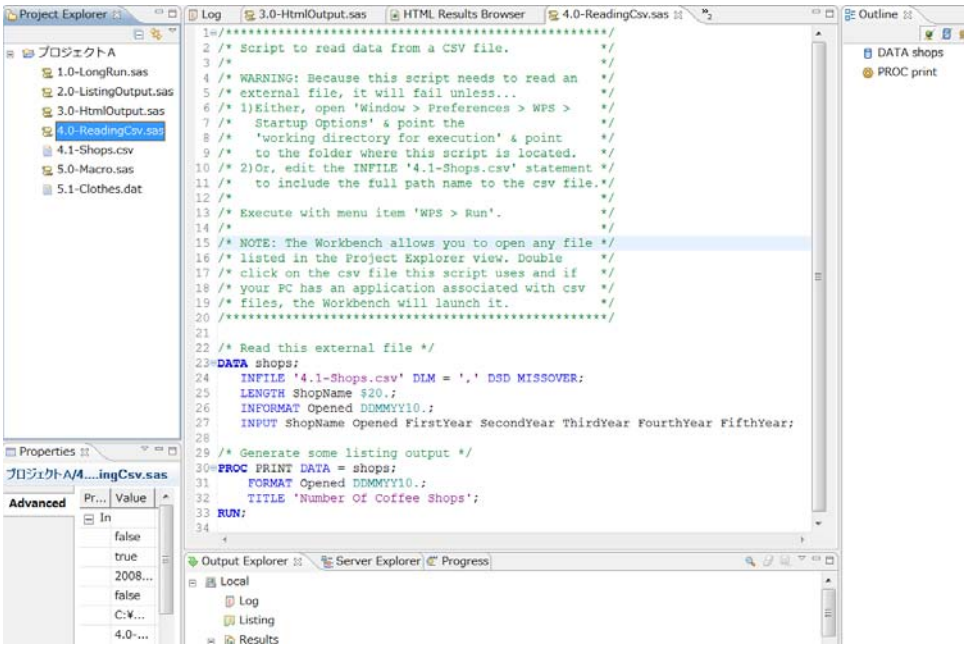


（エ）4.0-ReadingCsv.sas

CSV 形式ファイルデータの読み込み例です。 まず、読み込みたいファイル 4.1-Shops.csv を右クリックし、Open　With>Text Editor を選択し、内容をエディタビューで確認しましょう。



スクリプトをオープンします。



これも、24 行の指定ではエラーになる場合があります。 その場合は、以下のように修正してから実行します。

INFILE **'C:¥My PC¥Documents¥WPS Workspaces¥Workspace1¥プロジェクトA¥**4.1-shops.csv' DLM = ',' DSD MISSOVER;[15]

上記 INFILE ステートメントは読取り先のデータファイル名（'C:¥...¥4.1-Shops.csv'）、データ間の区切り文字（DLM=','（コンマ区切り））、レコード形式（DSD（Data-Separator-Data 形式、つまりデータ-区切り文字-データといったようにレコードが並んでいる））、レコード中のデータ数が読み取り指定した変数項目数より少ない場合の処理方針指定（MISSOVER（足りないデータがある変数項目は次のレコードのデータ値を読まさずに欠損値を充当するという方針の意味））を指定しています。 これらの指定により、CSV（Comma Separated Value）形式のファイルをうまく読むことができます。

15 ¥記号は¥記号で入力します。ただし、WPS のスクリプトでは¥記号は逆スラッシュ記号に変換されて表示されます。

```
1          DATA shops;
2             INFILE 'C:\My PC\Documents\WPS Workspaces\Workspace1\プロジェクトA\4.1-S
2        ! DLM = ',' DSD MISSOVER;
3             LENGTH ShopName $20.;
4             INFORMAT Opened DDMMYY10.;
5             INPUT ShopName Opened FirstYear SecondYear ThirdYear FourthYear Fift

NOTE: The file 'C:\My PC\Documents\WPS Workspaces\Workspace1\プロジェクトA\4.1-Shops
      File Name=C:\My PC\Documents\WPS Workspaces\Workspace1\プロジェクトA\4.1-Shops
      RECFM=V, LRECL=256
NOTE: 4 records were read from file C:\My PC\Documents\WPS
      Workspaces\Workspace1\プロジェクトA\4.1-Shops.csv
      The minimum record length was 27
      The maximum record length was 36
NOTE: Data set "WORK.shops" has 4 observation(s) and 7 variable(s)
NOTE: The data step took :
      real time : 00:00:00.006
      cpu time  : 00:00:00.000


7          /* Generate some listing output */
8          PROC PRINT DATA = shops;
9            FORMAT Opened DDMMYY10.;
10           TITLE 'Number Of Coffee Shops';
11         RUN;
NOTE: 4 observations were read from "WORK.shops"
```

Listing

sting

Number Of Coffee Shops

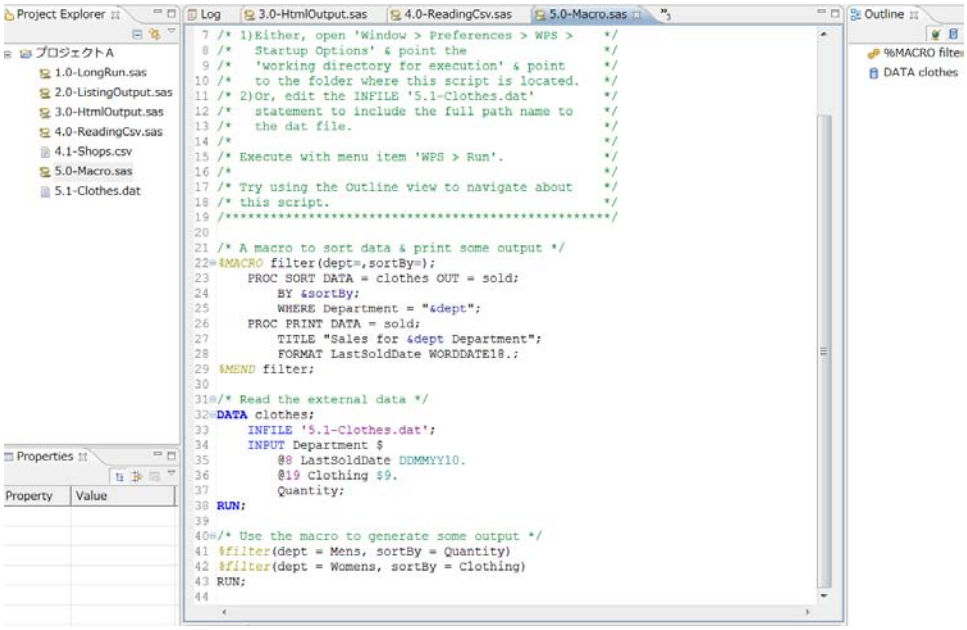
13:24 Wednesday, J

Obs	ShopName	Opened	First Year	Second Year	Third Year	Fourth Year
1	Coffee To Go	10/12/2000	2	2	10	15
2	Beans	10/06/2001	.	3	8	18
3	Starflux	01/07/2000	20	40	80	160
4	Coasts	06/09/2002	.	.	5	10

（オ）5.0-Macro.sas

SAS 言語のマクロ機能を用いたプログラミング例です。SAS マクロ機能はエクセルなどの通常のマクロ機能のようなパラメータを変えた単純な繰返し処理を行えるだけでなく、読み込んだデータ値などによって異なる処理を行うなど、高度に複雑な処理を事前にプログラミングしておくことができる SAS 言語機能の 1 つです。

この例では、22 行～29 行で服飾店売上データセット（clothes）を用い、特定の部門（&dept）の売上データを抽出し、特定の項目（&sortBy）で並び替えを行った上でリスト出力を行うマクロ filter を定義しています。 32 行～38 行では服飾店売上データセット clothes を作成し、41 行と 42 行でそれぞれパラメータ &dept と &sortBy に具体的な値を与えてマクロ filter を呼出しています。 41 行では紳士服を売上数量（Quantity）の少ない順、42 行では.婦人服を種類（clothing）のアルファベット順にリスト出力するプログラムを呼び出しています。



ここでもエラーを防ぐため、33 行を以下のように修正した上で実行する必要がある場合があります。

INFILE ' **C:¥My PC¥Documents¥WPS Workspaces¥Workspace1¥プロジェクトA¥**5.1-Clothes.dat';

実行がうまくいったかどうか、ログとリスト出力を確認してください。

（４）WPS の終了

File>Exit メニューを選択して WPS を終了します。終了時のウィンドウが保存されます。

（５）WPS スクリプトのバッチモード実行

WPS システムにはスクリプトを Windows コマンドプロンプトからバッチモードで実行する機能（wpsi.exe）が備わっています。wpsi.exe ファイルは既定のインストレーションの場合、以下のディレクトリに存在します。

C:¥Program Files¥World Programming WPS 2¥bin

基本的な wpsi コマンド使用法は、以下のとおりです。

>wpsi.exe <オプション> スクリプトファイル名

<オプション>には、以下の 1 つ以上を指定可能です。

-オプション名 [オプション値]

-config <コンフィグファイル名>

-set <環境変数名> <環境変数値>

詳細は WPS のヘルプを参照してください。

[実行例]

例えば、実行例（オ）5 .0-Macro.sas の中で読み取る外部ファイルをフルパスで指定するよう修正したスクリプト 5.0-Macro2.sas を実行するには、以下の内容をバッチファイル REI5.bat に保存し、コマンドプ

ロンプトから実行します。

（次の 3 行をファイル c:¥rei5.bat に保存しておきます）

```
set WPSLOC=C:¥Program Files¥World Programming WPS 2.0
set PRJLOC=C:¥My PC¥Documents¥WPS Workspaces¥Workspace1¥プロジェクトA
"%WPSLOC%¥bin¥wpsi.exe" "%PRJLOC%¥Macro2.sas"
```

ただし、5.0-Macro2.sas は 5.0-Macro.sas の中の外部ファイル指定をプロンプトで読む場合のフルパスで指定しなおしたもので、34 行を以下のように修正したものです。

```
INFILE ' /プロジェクトA/¥5.1-Clothes.dat';  /* 修正前 */
INFILE 'C:¥My PC¥Documents¥WPS Workspaces¥Workspace1¥プロジェクトA¥5.1-Clothes.dat';
/* 修正後 */
```

Window>プログラム>アクセサリ>コマンドプロンプトからコマンドライン入力画面を起動します。

```
c:¥>type rei5.bat
set WPSLOC=C:¥Program Files¥World Programming WPS 2
set PRJLOC=C:¥My PC¥Documents¥WPS Workspaces¥Workspace1¥プロジェクトA
"%WPSLOC%¥bin¥wpsi.exe" "%PRJLOC%¥5.0-Macro2.sas"
c:¥>rei5
```

```
1         /*************************************************/
2         /* Script to read some external data & use a macro */
3         /* to generate some output                       */
4         /*                                               */
5         /* WARNING: Because this script needs to read an   */
6         /* external file, it will fail unless...         */
7         /* 1)Either, open 'Window > Preferences > WPS >    */
8         /*   Startup Options' & point the                */
9         /*   'working directory for execution' & point     */
10        /*   to the folder where this script is located.   */
11        /* 2)Or, edit the INFILE '5.1-Clothes.dat'         */
12        /*   statement to include the full path name to    */
13        /*   the dat file.                               */
14        /*                                               */
15        /* Execute with menu item 'WPS > Run'.           */
16        /*                                               */
17        /* Try using the Outline view to navigate about    */
18        /* this script.                                  */
19        /*************************************************/
21        /* A macro to sort data & print some output */
22        %MACRO filter(dept=,sortBy=);
23              PROC SORT DATA = clothes OUT = sold;
24                      BY &sortBy;
25                      WHERE Department = "&dept";
26              PROC PRINT DATA = sold;
27                      TITLE "Sales for &dept Department";
28                      FORMAT LastSoldDate WORDDATE18.;
29        %MEND filter;
31        /* Read the external data */
32        DATA clothes;
33              INFILE 'C:¥My PC¥Documents¥WPS Workspaces¥Workspace1¥プロジェク
トA¥5.1-Clothes.dat';
34              INPUT Department $
35                      @8 LastSoldDate DDMMYY10.
36                      @19 Clothing $9.
37                      Quantity;
38        RUN;

NOTE: The file 'C:¥My PC¥Documents¥WPS Workspaces¥Workspace1¥プロジェクトA¥5.1-
Clothes.dat'
      is:
     File Name=C:¥My PC¥Documents¥WPS Workspaces¥Workspace1¥プロジェクトA¥5.1-
Clothes.dat
```

```
NOTE: The data step took :
      real time : 00:00:00.042
      cpu time  : 00:00:00.015

2                                       The WPS System        16:03 Wednesday,
January 30, 2008


40        /* Use the macro to generate some output */
41        %filter(dept = Mens, sortBy = Quantity)
NOTE: 4 observations were read from "WORK.clothes"
NOTE: Data set "WORK.sold" has 4 observation(s) and 4 variable(s)
NOTE: Procedure SORT step took :
      real time : 00:00:00.051
      cpu time  : 00:00:00.015


42        %filter(dept = Womens, sortBy = Clothing)
NOTE: 4 observations were read from "WORK.sold"
NOTE: Procedure PRINT step took :
      real time : 00:00:00.011
      cpu time  : 00:00:00.000


NOTE: 5 observations were read from "WORK.clothes"
NOTE: Data set "WORK.sold" has 5 observation(s) and 4 variable(s)
NOTE: Procedure SORT step took :
      real time : 00:00:00.018
      cpu time  : 00:00:00.000


43        RUN;
NOTE: 5 observations were read from "WORK.sold"
NOTE: Procedure PRINT step took :
      real time : 00:00:00.008
      cpu time  : 00:00:00.015


NOTE: Submitted statements took :
      real time : 00:00:00.254
      cpu time  : 00:00:00.093

c:¥>
```

実行結果のうち、リスト出力については、自動的に同じプロジェクト内に 5.0-Macro.lst ファイルが作成されその中に出力されます。

しかしながら、実行ログの方は上記のように、デフォルトではプロンプト画面に出力され、保存されません。最後の行を以下のように変更すると同じプロジェクト内の 5.0-Macro.log ファイルに保存されます。

```
"%WPSLOC%¥bin¥wpsi.exe" "%PRJLOC%¥5.0-Macro2.sas" > "%PRJLOC%¥5.0-Macro2.log"
```

2 章 プログラミングの基礎

本章では WPS のスクリプト言語である SAS 言語の基礎を学習します。

SAS言語はPL/1,C,Fortran,BASICなどと同じようにプログラミング言語の一種です。 ただし、元々はコンピュータ専門家ではない統計解析専門家のために開発されたデータ入力・加工・解析用の言語です。 そのため、PL/1,Cなどの手続き型言語と比較すると、はるかに使いやすい仕様となっています。[16] SAS言語は主にデータ加工を柔軟に行う目的を実現するために豊富な関数やプログラミング機能を有するDATAステップ [17]と特定のデータ分析・集計・レポートを行うために用意されているパラメータ指定仕様のPROCステップの 2 つの性格の異なるコンポーネントを有しており、これらを組み合わせてプログラミングを行うようになっています。 そのほかにもSQL言語仕様によるデータ検索を行うSQLプロシジャ [18]や高度なマクロ機能を実現するマクロ言語といったコンポーネントも含んでおり、必要に応じて使いわけていくことになります。

SAS 言語は非常に強力なデータ入力・加工機能を持っています。その分、すべての機能を最初から使いこなすことは不可能です。最初は基本的な機能から始めて、使いながら順に新しい機能を覚えていくというやり方がお勧めです。 ここでは最初に学ぶべき入門的な内容を理解することを目的としています。

（１） 基本文法

まず、基本的な SAS 言語の用語と文法を学びましょう。

（ア）DATA ステップと PROC ステップ

SAS 言語によるプログラミングは、DATA ステップと呼ばれる実行単位と PROC ステップと呼ばれる実行単位を組み合わせて行います。 DATA ステップは汎用的なデータ検索加工機能を実行するためのプログラミング言語部分であり、PROC ステップは基本統計やレポーティングその他の特定機能を実行するために用意されている組み込みモジュールです。 DATA ステップは DATA ステートメントで始まり、PROC ステップは PROC ステートメントで始まり、どちらも RUN ステートメントで明示的に終了します。 ただし、RUN ステートメントを記述しなくても､次の DATA ステップまたは PROC ステップを開始すればその前の DATA ステップまたは PROC ステップは暗黙的に終了します。

（イ）ステートメント

SAS 言語には通常のプログラミング言語と同じようにプログラミングステートメント（文）が用意されています。 1 個のステートメントは、通常、キーワード（定型語）で始まりセミコロン（;）で終了します。キーワードとセミコロン以外にステートメントに含まれる他の要素は、変数名、データセット名、関数名、フォーマット名、定数、パラメータなどの 1 文字以上の長さを持つワード（語）と演算子、記号およびブ

16 SAS は 1973 年ごろから開発がはじまり、当時は第四世代言語と呼ばれていました。

17 使いやすいだけでなく、扱えるデータ量やファイル形式などに制約が無く、また、複雑な処理も高級言語なみに実行できる機能を持っています。なお DATA ステップはコンパイル言語です。

18 PROC ステップの個々の構成要素をプロシジャと呼んでいます。一般の開発言語でいうところのサブルーティンに相当し、実際に実行可能形式（ロードモジュール）となっています。

ランク（空白）などの 1 文字の長さの特殊文字です。 ワードの区切り文字はブランク 1 個以上です。 ただし、演算子（+-*/など）や記号（¥やカッコや引用符など）も通常ワード間の区切り文字として認識されます。そのため、これらの特殊文字をワードとワードの間に記述する場合、ブランクはあってもなくてもかまいません。 また、1 つのステートメントを複数行にわたって書いても、1 つの行に複数のステートメントを書いてもかまいません。 ただし、ワードの途中で改行することは許されません。 なお、キーワードを含むワードのアルファベットは大文字（A～Z）・小文字（a～z）いずれで書いてもかまいません。
なお、WPS で使える演算子の種類を[別表 7]に表示しました。

例

Data Longrun;

このステートメントはキーワード DATA で始まっていますので、DATA ステートメントと呼びます。 このステートメントは以下のように分解できます。

“Data”(キーワード=ステートメントの開始) ,“ ”(ブランク＝区切り文字), ”Longrun”（ワード＝このステートメンメントでは作成するデータセット名を宣言する）,”;”（セミコロン=ステートメントの終了）

なお、WPS（バージョン 2.0）でサポートしている DATA ステップで使えるステートメントの一覧を[別表 4]に、PROC ステップの種類を[別表 5]に、グローバルステートメントを[別表 6]にそれぞれ掲載しました。

（ウ）データセット名、変数名の命名規則

WPSで内部的に管理するデータセットをWPSデータセットと呼び、WPSデータセットは表形式のデータ集合の形をしており、行（レコード）方向をオブザベーション、列（カラム）方向を変数と呼びます。[19] なお、本書においてデータセットという呼び名は、特に断りが無い限りWPSデータセットのことを指します。

データセット名、変数名のつけ方には以下の制約が課せられています。

データセット名、変数名の命名規則
・長さ 32 文字以内（すべて半角）
・先頭の 1 文字はアルファベット（A～Z）もしくはアンダースコア（_）のいずれかでなければならない
・2 文字目以降はアルファベット（A～Z）もしくはアンダースコア（_）もしくは数字（0～9）のいずれかが使える

名前の中に漢字やブランクが使えないことに注意してください。

なお、データセット名、変数名のアルファベットの大文字小文字の区別については、以下のように取り扱われます。

データセット名、変数名の大文字小文字の区別
・データセット名は大文字小文字の区別はなく、すべて WPS 内部で大文字として認識される
・変数名はDATAステップやPROCステップのプログラミングレベルおよび実行中は大文字小文字の区別は

19 オブザベーション、変数という呼び名は SAS 言語が元々統計解析用のアプリケーション開発言語として開発された歴史に由来しています。

なく同じ文字として認識される。 ただし、データセットに格納する変数名は最初に定義したとおりに大文字小文字を区別して保存される。[20]

同じ命名規則は、配列名、ステートメントに置くラベル名、マクロ定義名、マクロ変数名、カタログ名、ライブラリ参照名にも適用されます。なお、一部の命名規則（ユーザー定義フォーマット名など）は若干異なりますので注意が必要です。

（エ）変数のタイプと値

WPS の変数のタイプ（型）は以下のとおり、数値タイプと文字タイプの 2 通りしかありません。

変数のタイプ

・数値タイプ

内部的に 3 バイト～8 バイトの浮動小数点形式で格納される。[21]

・文字タイプ

1～32767 バイトの長さまでの文字列を値として持つことができる。

数値変数は固定小数点形式で持つことができませんので、丸め誤差にシビアなアプリケーションに WPS を使うような場合は注意が必要です。

文字変数に定数を与えるには、ダブルクオテーション（"）で囲むか、シングルクオテーション（'）で囲んで指定します。

例

name=”Robert Edison”

sex=’M’

（オ）日付の取り扱い

SAS 言語における日付の取り扱いは通常の数値タイプ変数値に対して以下のように取り扱われます。

日付の内部表現と外部表現

・日付値の内部表現

1960 年 1 月 1 日を起点とする経過日数を値として持つ。

・日付の外部表現と計算

日付に関するフォーマット、インフォーマットにより外部表現と内部表現との間を変換する。 また、日付関数を用いて計算処理を行う。

・時間値と日時値

1960 年 1 月 1 日午前 0 時 0 分 0 秒を起点とする経過秒数を値として持つ。

・時間値と日時値の外部表現と計算

日付値と同様に時間と日時に関するフォーマット、インフォーマットにより外部表現と内部表現との間を

20 CONTENTS プロシジャを用いて変数名をデータセット出力するような場合に注意が必要となります。

21 LENGTH ステートメントで長さを定義しなかった場合は 8 バイトに設定されます。

変換する。 また、時間や日時を扱う関数を用いて計算処理を行う。

ここで、フォーマットとは WPS 変数の値を画面上やファイル中のレコード上に書き出すとき、どのような編集形式で書き出すかを指定するものです。一方、インフォーマットとはテキスト行やファイル中のレコードから値を読み取るとき、読み取り元のデータ値がどのような編集形式で表現（または格納）されているかを指定するものです。WPS でサポートされているフォーマットとインフォーマットの一覧をそれぞれ[別表 9],[別表 10]に表示しました。

たとえば、変数 date に 1960 年 4 月 1 日を代入して内部値を見てみましょう。

```
Data _null_;
  date=input(“1960401”,yymmdd8.);
  put date=;
Run;
```

date=91 となります。 つまり、1960 年 4 月 1 日は 1960 年 1 月 1 日から 91 日後の日付であることを意味しています。

なお、上記プログラムにおいて input()の部分は WPS でサポートされている関数の 1 つを表します。[別表 8]に WPS でサポートしている関数一覧を表示しました。

（2）サンプルプログラムの説明

さて、以上の文法を頭に入れて 1 章で実行したサンプルプログラムコーディングに取り組んでみましょう。

（ア）1.0-LongRun.sas

```
data LongRun;
do I=1 to 5000000;
output;
end;
```

この例では明示的な DATA ステップの終わりを示す RUN ステートメントが省略されています。

この例に出現するステートメントの文法を学習します。

［DATA ステートメント］

DATA データセット名 1 [データセット名 2] [...];

DATA ステートメントは DATA ステップの開始を宣言し、このステップで作成する WPS データセットの名前を宣言します。

この例では LongRun という名前の WPS データセットを 1 つ作成することを宣言しています。

［DO ステートメント］

DO [変数名=初期値 TO 終了値] [BY 増分値] ;

DO ステートメントは他の言語にも登場する一般的なステートメントです。 必ず END ステートメントと対で使用され、DO ステートメントと END ステートメントに囲まれたステートメントを実行するように指

示します。[変数名=初期値 TO 終了値]が指定された場合は、繰り返し DO ステートメントと呼ばれ、指定した変数の値が初期値から終了値に達するまで値を[増分値]（指定がなければ 1）単位で増加させながらDOステートメントとENDステートメントに囲まれたステートメントを繰り返し実行する指示となります。

この例では変数Iの値を1から1の増分で5000000になるまで500万回繰り返し、その中で次のOUTPUTステートメントを実行することを指示しています。

［OUTPUT ステートメント］

OUTPUT [出力データセット名 1] [出力データセット名 2] [...];

このステートメントはその時の変数値を持つオブザベーションを作成先のデータセットに書き出すことを指示します。 出力データセット名を省略すると、DATA ステートメントで指定したすべてのデータセットが出力先となります。 この例では LongRun データセットが出力先となります。

［END ステートメント］

直前のDOステートメントと対で使われENDステートメントの前のステートメントまでを実行するよう指示します。

ここで、スクリプトを実行したときの実行ログを説明しておきます。

NOTE: Data set "WORK.LongRun" has 5000000 observation(s) and 1 variable(s)
NOTE: The data step took :
　　real time : 00:00:00.933
　　cpu time　: 00:00:00.655

ログはスクリプトの実行状況を報告する役割を持ち、エラー(ERROR)、警告(WARNING)、ノート(NOTE)の各メッセージにより実行状況を確認できます。

この例ではエラーと警告は発せられず、ノートのみのメッセージとなっており、少なくとも文法的なエラーや実行時のエラーは出現しなかったということを意味しています。 ただし、意図した結果が得られたかどうかは、ノートに記された作成されたデータセットのオブザベーション数と変数の数が 1 つのポイントとなります。 特にオブザベーション数が 0（ゼロ）になっていないかどうかを確認することが実際上は重要です。 エラーが無い場合でもこの部分は必ず確認しておきましょう。 この例では、データセット WORK.LomgRun[22]は 5000000 件のオブザベーションと 1 変数を有しているというメッセージですので問題はなさそうです。（最終的に問題がないかどうかは、1 章の例で示したようにサーバエクスプローラビューからデータセットの中身を見て判断します）

（イ）2.0-ListingOutput.sas

```
data employees;
length Name $10 Age 3 Job $15;
input Name $ Age Job $;
```

22 データセット名の頭についている WORK.は一時的なデータライブラリの中にデータセットが保存されることを意味します。

```
cards;
Beth 41 Sales-Manager
Steve 32 Sales
William 35 Accounts
Anne 28 Marketing
Dawn 26 Sales
Andrew 36 Support
;
proc print;run;
/* Generate some listing output */
proc sort data = employees; by name;
proc print data = employees noobs;
title 'Employees in Company X';
```

この例はプログラムと一緒にデータ行を入力して WPS データセット employees を作成する例となっています。

［LENGTH ステートメント］

LENGTH 変数名 1 [$]長さ [変数名 2 [$]長さ] [...] ;

作成する変数の属性（名前とタイプと長さ）の定義を実行します。 この例では作成するデータセット employees の中に含む変数として、以下の変数名とタイプと長さの定義を実行しています。

変数名	タイプ	長さ
Name	文字	10 バイト
Age	数値	3 バイト
Job	文字	15 バイト

この例にあるように、変数名の後に$を指定すると文字タイプとして定義されることになり、$が無いと数値タイプとして定義されます。

［INPUT ステートメント］

INPUT 変数名 1 [$] [変数名 2 [$]] [...];

INPUT ステートメントは多彩な指定方法を持ち、入力データのフォーマットに合わせて柔軟にデータ読取りを行えるステートメントです。

この例は最も単純なリスト入力と呼ばれる指定方法です。 リスト入力指定は、読み取りたいオブザベーションの各データ値がブランクを区切り文字として並んでいることを前提とし、指定の変数名の順にデータ値を読み取る場合に使います。したがってこの場合、Name,Age,Jobの順に読み取ることを指示しており、NameとJobは文字タイプとして読み取ることを指示しています [23]。 読み取り先はCARDSステートメント

23 この例では既に LENGTH ステートメントで変数のタイプ指定は済ましているため、厳密にはこの INPUT ステートメントに変数タイプ指定（$記号）は不

とヌルステートメント（;）に囲まれた範囲のテキスト行です [24]。
なお、リスト入力以外にデータ値が区切り文字なしに固定カラム位置に並んでいる場合に有効なカラム入力指定や複数行にわたるデータ値の読み取り順を自由に設定できるポインター入力指定も可能です。
（カラム入力指定）
INPUT 変数名 1 [$]スタートカラム位置-エンドカラム位置　[変数名 2 [$]スタートカラム位置-エンドカラム位置] [...];
（ポインター入力指定）
INPUT [#行番号] [@カラム位置] [+相対移動カラム数] 変数名 1 インフォーマット [[#行番号] [@カラム位置] [+相対移動カラム数] 変数名 2 インフォーマット] [...];
これらの指定方法の詳細はヘルプ等を参照してください。
［CARDS ステートメント］
CARDS;
このステートメントの次の行からセミコロン（;）行の 1 つ前の行までの範囲内にデータ値が存在することを WPS に知らせます。なお、同じ役割を持つステートメントで、セミコロン記号をデータ値として入力したい場合や 2 バイト文字を含むデータ値を正しく入力するために CARDS4 ステートメントが別にあります。
[ヌルステートメント]
;
CARDS ステートメントと対で指定します。ヌルステートメントは必ずセミコロン 1 個のみを指定してください。 データ値を含む同じ行にセミコロンを指定すると、セミコロンを含む行のデータ値はすべて無視されます。 なお、CARDS4 ステートメントに対応するヌルステートメントは;;;;（4 個の連続したセミコロン）です。

[PROC PRINT ステートメント]
PROC PRINT [DATA=入力データセット名] [その他のオプション];
PRINT プロシジャを呼び出します。PRINT プロシジャは DATA=入力データセットのオブザベーション値のリストをプリント出力するためのプロシジャです。DATA=指定は省略可能で、省略値は DATA=_LAST_ です。(_LAST_は最後に作成された WPS データセットの名前を保持している WPS システム変数の 1 つです）その他のオプションの 1 つである NOOBS オプションはプリント出力に通常表示されるオブザベーション番号を省略するよう指示するものです。
［PROC SORT ステートメント］
PROC SORT [DATA=入力データセット名] [OUT=出力データセット名] [その他のオプション];
SORT プロシジャを呼び出します。SORT プロシジャはデータセットのオブザベーションの並びを付随する BY ステートメントで指定するキー変数の値の順に並べ替えを行うプロシジャです。DATA=入力データ

要です。

24 後に登場する INFILE ステートメントが存在しない場合、WPS は CARDS ステートメントを探す仕組みです。

セット名も OUT=出力データセット名も省略可能ですが、OUT=を省略すると DATA=オプションで指定した入力データセットと同じ名前が出力データセット名として指定されたものとみなされますので、データセットの内容が置き換わることになる点に特に注意が必要です。誤って置き換えを行うことを防ぐ以外に、プログラムの視認性を高めるためにも DATA=オプションと OUT=オプションは両方とも省略せずに指定することをお勧めします。

[BY ステートメント]

BY [DESCENDING] 変数名 1 [[DESCENDING] 変数名 2] ... ;

SORT プロシジャでは必須指定です。並べ替えを行うキー変数を指定します。DESCENDING オプションを変数名の前に指定するとその変数の値の大きい順（文字変数ならソートシーケンスの順）に並べ替えを行う指示となります。

（ウ）3.0-HtmlOutput.sas

```
/* Create HTML output */
ods html body = '¥プロジェクトA¥CarRepairs.html';
/* Use this sample data */
data vehicles;
length Vehicle $10 Repaired 3 Maintenance $15;
input Vehicle $ Repaired Maintenance $;
cards;
Saloon 7 Exhaust
Sports 2 Service
Sports 6 Body-Work
Saloon 3 Body-Work
4-Door 12 Major-Repair
Sports 10 Minor-Repair
4-Door 5 Service
Pick-Up 3 Minor-Repair
;
/* Produce some FREQ and MEANS output */
proc freq data = vehicles;
 table Maintenance;
 title 'Monthly Vehicle Maintenance Report';
proc sort data = vehicles;
 by Maintenance;
proc means data = vehicles;
 by Maintenance;
```

```
run;
/* Close the HTML files */
ods html close;
```

[ODS HTML ステートメント]
最初の ODS HTML ステートメントは body=オプションにより出力先 HTML ファイル名を宣言しています。
最後の ODS HTML ステートメントは作成した HTML 出力ファイルをクローズします。
[DATA ステートメント] データセット Vehicles を作成することを宣言しています。
[LENGTH ステートメント] 3 個の変数 Vehicle,Repaired,Maintenance の属性（タイプと長さ）を宣言しています。
[INPUT ステートメント] CARDS ステートメントとヌルステートメント(;)に囲まれた範囲の 8 行のテキスト行から各変数値を指定順にの読み取るよう指示しています。
[PROC FREQ ステートメント]
PROC FREQ [DATA=入力データセット名] [その他のオプション];
FREQ プロシジャを呼び出します。FREQ プロシジャは付随する TABLE ステートメントで指定する分類変数の多重クロス集計を行うプロシジャです。
[TABLE ステートメント]
TABLE 変数名 1[*変数名 2[*変数名 3*[...]]] [...][/オプション];
FREQ プロシジャに付随する必須ステートメントです。クロス集計をとりたい変数組み合わせを*記号で繋いで指定します。 また、ブランクで区切って別の変数組み合わせ指定を行うこともできます。この例では Maintenance の値ごとの件数を集計してレポートするよう指示しています。数値タイプの変数も指定できますが、値の種類ごとに件数集計を行いますので、たくさんの値の種類を持つ数値変数の場合、集計結果リストが膨大となる可能性があります。そのため数値変数を TABLES ステートメントに指定する場合は注意が必要です。
[PROC MEANS ステートメント]
PROC MEANS [DATA=入力データセット名] [その他のオプション];
MEANS プロシジャを呼び出します。MEANS プロシジャは数値タイプの変数の平均、合計、標準偏差といった分布の集計値を計算します。集計を行う変数の指定は VAR ステートメントで指定しますが、MEANS プロシジャは PROC MEANS ステートメント以外すべてオプション指定です。VAR ステートメントを指定しない場合は、入力データセットに含まれるすべての数値タイプの変数が指定されたものとみなされます。
[BY ステートメント]
BY [DESCENDING] 変数名 1 [NOTSORTED] [[DESCENDING] 変数名 2 [NOTSORTED]] ... ;
PROC SORT プロシジャ以外のプロシジャと DATA ステップで使われる BY ステートメントは処理中のオブザベーションが BY ステートメントで指定された変数の昇順,降順（DESCENDING オプションが指定された場合）,昇順・降順というわけではないがグループ別（NOTSORTED オプションが指定された場合）に並んでいることを前提として、グループ別に処理することをプロシジャまたは DATA ステップコンパイ

ラに指示する役割を持っています。この例では Maintenance の値ごとにオブザベーションのグループを作りグループごとに数値変数の基本統計量を計算しアウトプウト画面（今回は html ファイル）に出力するよう指示しています。

（エ）4.0-ReadingCsv.sas

```
DATA shops;
   INFILE '4.1-Shops.csv' DLM = ',' DSD MISSOVER;
   LENGTH ShopName $20;
   INFORMAT Opened DDMMYY10.;
   INPUT ShopName Opened FirstYear SecondYear ThirdYear FourthYear
FifthYear;
/* Generate some listing output */
PROC PRINT DATA = shops;
 FORMAT Opened DDMMYY10.;
 TITLE 'Number Of Coffee Shops';
RUN;
```

ＣＳＶファイル 4.1-Shops.csv の内容

```
Coffee To Go,10/12/2000,2,2,10,15,20
Beans,10/06/2001,,3,8,18,30
Starflux,01/07/2000,20,40,80,160,320
Coasts,06/09/2002,,,5,10,10
```

ＣＳＶ形式のファイルで、各レコードの長さが一定でない（可変長レコードと呼ばれます）テキストファイルからデータを読み取る例です。

［INFILE ステートメント］

INFILE ファイル参照名または'実ファイル名' [オプション];

後に続く INPUT ステートメントで読み込む入力ファイルの存在場所・名前・属性などを指定します。

ファイル参照名を指定する場合は、INFILE ステートメントに先んじて FILENAME ステートメントを用い INFILE ステートメントで指定するファイル参照名を実ファイル名に結び付けておきます。 この例では'実ファイル名'を直接指定する方法をとっています。 なお、この例のように実ファイル名のパスが省略された場合、プロジェクトディレクトリの下にそのファイルが存在するものとみなされます。 オプションはファイルの属性を指定するようになっています。 また、この例は CSV ファイルを読むための標準的なオプションが指定されています。 DLM=',' オプションはデータ間の区切り文字としてコンマを使うことを指示しており、DSD オプションは読み込むべきレコードが可変長の Data-Separator-Data 形式、つまりデータと区切り文字が交互に並んだ表現形式となっていることを指示しています。 この DSD オプションを指

定しているため区切り文字が連続した箇所も対応する変数値を欠損値にセットして正しく読み込むことができるようになっています。 また、MISSOVER オプションは少々理解しにくいかもしれませんが、レコード中のデータ項目数が INPUT ステートメントで指定した変数項目数より少ない場合の処理方針を指示しています。 MISSOVER とは欠損値にセットして次に進めといった意味です。 もしも MISSOVER オプションを指定しないと、INPUT ステートメントで指定した最後の方の変数で対応するデータ値を同一レコード上に見つけられなかった変数は次のレコードのデータ値を読みに行ってしまいます。
なお、実践上大変重要なオプションがこの例では指定されていませんので、追記しておきます。 それは、次の論理レコード長属性を指定するオプションです。
LRECL=最大論理レコード長(バイト数)
このオプションを指定していない場合はLRECL=256 が暗黙的に指定されたものとして扱われます。 したがって、INPUTステートメントの実行時点では外部ファイルのレコード上で先頭から 256 バイト目を超える部分のデータは無視されます。 この例ではどのレコードも長さが 256 バイトより短いため問題は起きていませんが、実際に読み取りたいデータの最大論理レコード長は 256 バイトを越える場合が普通にありますので、注意が必要です。 可変長レコードを持つファイルに対してはLRECLオプションを実際の最大レコード長よりいくら大きめに指定しても不都合はありませんので、CSVファイルを読む場合のINFILEステートメントには、常に、LRECL=32000 というように大きな値を指定しておくことをお勧めします [25]。
［LENGTH ステートメント］
この例では変数 ShopName の値を長さ 20 バイトの文字タイプとして定義しています。 もしもこの指定が無い場合は、すべての文字タイプ変数の値は、デフォルトの長さである 8 バイトとして定義されるため、8 文字目を超えるデータ値の部分は切り捨てられてしまいます。
［INFORMAT ステートメント］
INFORMAT 変数名 1 入力フォーマット 1 [変数名 2 入力フォーマット 2] […];
変数ごとに読み取る外部ファイル上のデータ値の入力データ編集形式を指定します。
この例では変数Openedの値は外部ファイル上でDDMMYY10.という入力フォーマットで編集されているということをWPSに知らせています。DDMMYY10.という入力フォーマットは日付の値に関するSAS言語上の入力データ編集形式の 1 つです。 CSVファイルの最初のレコードの 2 番目の変数Opendの読み取り先の値は 10/12/2000 と表示されたテキスト値となっていますが、DDMMYY10.は日(10)、月(12)、年(2000)の順に並んだ全体の長さが 10 バイトの値を 2000 年 12 月 10 日を表すWPS変数Openedの値 [26] に変換して読み取るための指定です。
［INPUT ステートメント］
この例では CSV ファイルのレコード上の値の並びに対応して 7 個の変数を読み取るよう指示しています。

---

25 余分の長さを LRECL オプションで指定するのは CSV ファイルなどの可変長ファイルを読む場合のみ有効です。固定長ファイルを読む場合はファイルに合わせた論理レコード長を指定してください。

26 WPS 変数のタイプは数値と文字の 2 種類しか存在しませんので、日付を表す入力フォーマットで読み取られた変数値は、1960 年 1 月 1 日からの経過日数に変換され通常の数値タイプの変数値として格納されます。

参考として、固定長ファイル（4.1-Shops.dat）からデータを読み取る場合のコーディング例を以下表示します。

```
DATA shops;
   INFILE '4.1-Shops.dat' LRECL = 42;
   LENGTH ShopName $20;
   INFORMAT Opened DDMMYY10.;
   INPUT ShopName Opened FirstYear SecondYear ThirdYear FourthYear
FifthYear;
```

テキストファイル 4.1-Shops.dat の内容

```
----+----1----+----2----+----3----+----4----+
Coffee To Go        2000/12/10  2  2 10 15 20
Beans               2001/06/10  .  3  8 18 30
Starflux            2000/07/01 20 40 80160320
Coasts              2002/09/06  .  .  5 10 10
```

（３）次のステップへ

ここまで WPS の基本的な使い方と文法について述べてきましたが､本書で触れなかったたくさんのことが残っています。特に、以下のテーマについて学習することをお勧めします。

（１）データベースとのインターフェース機能(LIBNAME エンジン)
（２）データセットのマージとテーブルルックアップ(MERGE ステートメント＋BY ステートメント)
（３）ユーザ定義フォーマットとインフォーマット（PROC FORMAT）
（４）SQL プロシジャ(PROC SQL)
（５）マクロ言語機能(%MACRO ステートメント,%LET ステートメントなど)

[別表 1] WPS ワークベンチ Welcome メニューの内容

Overview サブメニュー –

WPS の 2 つの基本構成要素である WPS サーバと WPS ユーザインターフェース（WPS Workbench）の概要を説明する以下のメニューアイテムを表示します。

WPS は SAS 言語で書かれたプログラム（WPS ではスクリプトと呼ぶ）を読んで実行する WPS サーバと、Workbench ユーザインターフェースの 2 つの基本構成について説明しています。

WPS Workbench の各ビュー（プロジェクトエクスプローラ、エディタとアウトプット、アウトライン、プロパティ、アウトプットエクスプローラ、サーバエクスプローラ、プログレス、ヘルプなど）の基本的役割を説明しています。

WPS Workbench で管理される構成要素（プロジェクト、スクリプト、ログ、リスティング出力、HTML 出力、ライブラリ、データセット）について説明しています。

この Welcome 画面の説明です。

SAS 経験者のために SAS とのコンパティビリティ、既存 SAS プログラムの動作確認ユーティリティ（Analyzer Tools）、SAS から WPS への移行方法などの情報を提供しています。

Workbenchユーザガイド [27]、WPS言語レファレンス、WPS Workbenchユーザガイドへの入り口です。

[27] WPS Workbench は IBM 社が提供する eclipse オープンソースを用いて作成されているため、ユーザ独自の拡張も可能です。この Workbench ユーザガイドは IBM 社のドキュメントです。

Tutorials サブメニュー-

チュートリアルメニューアイテムに切り替わります。WPS では cheat sheets（カンニングシート）と呼んでいるチュートリアルビューが Workbench の中に出現し、その指示に従ってオブジェクトを操作していくことにより WPS の使い方を自習形式で習得していくことができます。


An Introduction to Using WPS

プロジェクトの開始、スクリプトの作成・実行、結果の確認、エラー出現と訂正などの一連のタスクを実行します。


FAQ: Running Scripts

スクリプト実行に関する良くある質問に対する回答をチュートリアル形式で学習します。


Customise the Workbench Layout

Workbench の各ビューの操作方法やレイアウト変更方法に関するチュートリアルです。


Sample Scripts

WPS インストレーションファイルに含まれているサンプルスクリプトを実行します。（本文で説明している内容です）

What’s New サブメニュー

現在のバージョンの新機能などに関する情報を表示するメニューアイテムに切り替わります。

[別表２] WPS ワークベンチ 主なメニューコマンドリスト

ファイル（File）

新しいまたは既存のプロジェクト・フォルダー・スクリプトの作成（New）・呼び出し（Open File）・保存（Save, Save As）、名前の変更（Rename）

各ビューウィンドウ内容の印刷（Print）

Workspace の変更（Switch Workspace）

ファイルのインポート（Import）／エクスポート（Export）

WPS の終了（Exit）

エディット（Edit）

元に戻す（Undo）と再実行（Redo）

アイテムの切取り（Cut）,コピー（Copy）,貼付け（Paste）および削除（Delete）

ナビゲート（Navigate）

前回の編集箇所へ移動（Last Edit Location）

行番号を指定して移動（Go To Line...）

サーチ（Search）

ファイル内の文字列探索（Search , File , Text）

プロジェクト（Project）

プロジェクトのオープン（Open）／クローズ（Close）

ＷＰＳ（WPS）

スクリプトの実行（Run）・実行中止（Cancel）

ログのクリア（Clear Log）・画面リスト出力のクリア（Clear Listing）

WPS サーバの再起動（Restart Server）

ウィンドウ（Window）

新しい Workbench ウィンドウをオープン（New Window）

新しいスクリプト編集ビューのオープン（New Editor）

ビューを指定して表示（Show View）

Workbench レイアウトのカスタマイズ（Customize Perspective...）と保存（Save Perspective As...）

各種ユーザ設定（Preferences...）

ヘルプ（Help）

Welcome 画面の呼び出し（Welcome）

ダイナミックヘルプ（Dynamic Help）

ショートカットキー（Key Assist...）

使用上のヒント（Tips and Tricks...）

チュートリアル（Cheat Sheets...）

ライセンス管理（View License , Apply License）

[別表 3] WPS ワークベンチ ビューの名前と役割

プロジェクトエクスプローラビュー（Project Explorer）

作業中のプロジェクト名とその中に含まれるファイルやフォルダーをツリー構造で表示します。

なお、 1 つの Workbench で表示できるのは 1 つのプロジェクトのみです。現在作業中のプロジェクトを閉じないで別のプロジェクトの作業を同時に行いたい場合は、Window>New Window コマンドを使って、新しい Workbench を開始します。

エディタアンドアウトプットビュー（Script1.sas Log Listing WORK.LongRun *4.1-Shops.csv など）

これらのビューはファイルの内容を編集したり確認したりするためのものです。複数のビューを開いたり閉じたりできます。

アウトラインビュー（Outline）

スクリプトまたはログに含まれる内容を構造化して表示します。

スクリプトに対しては Global ステートメント,DATA ステップ,PROC ステップ,マクロなどを単位として表示します。ログに対してはエラーやワーニング（警告）メッセージを単位として表示します。

プロパティビュー（Properties）

プロジェクトフォルダ,スクリプト,ライブラリ,データセットなど各オブジェクトに対するプロパティ（属性情報）を表示します。

アウトプットエクスプローラビュー（Output Explorer）

スクリプトの実行ログ（Log）実行結果リスト出力（Listing）実行結果 HTML 形式出力（HTML result File）を表示する各ビューをオープンします。なお、このビューに表示された情報のリセットは Restart Server コマンドにより行います。

サーバエクスプローラビュー（Server Explorer）

作成したデータセットの内容、フォーマットカタログ、ファイル参照名、ライブラリ参照名などを参照するビューです。なお、このビューに表示された情報のリセットは Restart Server コマンドにより行います

プログレスビュー（Progress）

実行中と実行待ちの状態あるスクリプトの情報を表示します。またこのビューから実行中のスクリプトの実行中止を指示したり、実行待ちの状態にあるスクリプトをキャンセルすることができます。

ヘルプビュー（Help）

作業場面に応じたヘルプ情報（コンテキストヘルプ）を表示します。メニューから Help>Dynamic Help を選択するか F1 キーを押します。

タスクビュー（Tasks）と **ブックマークビュー**（Bookmarks）

これらのビューを用いると、プロジェクト内の任意のスクリプトの任意の行に対してノート（備忘録）とプライオリティ（優先順位）を付けておく、またはブックマーク（しおり）を付けておくことができます。タスクノートやブックマークを追加するには、スクリプトビューにおいて、当該行を選択した状態で右クリックし、出現するコンテキストメニューから Add Task または Add Bookmark を選択します。タスクビ

ューまたはブックマークビューに記録されたタスクアイテムやしおりアイテムをダブルクリックすることにより、当該スクリプトの当該行をすぐに画面に呼び出すことができます。

サーチビュー（Search）

テキスト内の文字列パターンの検索および置換、およびファイルの検索を行うためのビューです。

この他にも、**コンソールビュー**（プラグインを追加するときのみ関係するビュー）、**プロブレムビュー**（システムからのエラーや警告メッセージの表示）、**ナビゲータビュー**（ナビゲーション表示）、**インターナルウェブブラウザビュー**（インターネットブラウザ）などがあります。

[別表 4] DATA ステップで使えるステートメント一覧

ステートメント名	役割
ABORT	DATAステップの実行を中断する
ARRAY	変数配列の宣言
asign（割り当て）	変数名=式;の形で指定する。 等号の左辺の変数に右辺の式の値を割り当てる
ATTRIB	1つの変数の属性（タイプ・長さ・フォーマット・インフォーマット・ラベル）をまとめて宣言する
BY	指定の変数のソート順にオブザベーションが並んでいることを示す。 BYグループ処理を行う場合に必須となる
CALL	CALLルーティン（複数の戻り値を許す関数）の呼び出し
CARDS	これ以降にカードイメージデータが記述されていることを示す
CARDS4	同　セミコロンや2バイト文字を含むデータを正しく読み取る場合にCARDSに代えて用いる
CONTINUE	DOループ（DOグループ）処理の中で用い、ENDステートメントまで強制移動させてDOループ処理にとどまることを指示する
DATA	DATAステップの開始とこのステップで作成する出力データセット名を宣言する
DATALINES	CARDSステートメントの別名
DATALINES4	CARDS4ステートメントの別名
DELETE	現在処理中のオブザベーションの処理を中断（出力データセットに書き込まない）して、次のオブザベーションの処理に移るためにDATAステップのはじめに戻る
DO	ENDステートメントと対で用い、条件式に合致した場合の実行範囲をDO～ENDで囲んで指定する。 囲まれた範囲をDOループまたはDOグループと呼ぶ
DO, iterative（繰り返しDO）	繰り返しDOステートメントの1つで、iterativeの部分には　変数名=開始値　TO　終了値　BY　増分値というDOループ処理の実行条件指定が入る
DO UNTIL	同　UNTIL(条件式)に指定した条件を満たさない範囲でDOループ処理を実行する
DO WHILE	同　WHILE（条件式）に指定した条件を満たしている範囲でDOループ処理を実行する
DROP	出力データセットに含めない変数を指定する
ELSE	IF～THENステートメントと共に用い、IF条件に合致しない場合の処理を記述する
END	DOステートメントと対で用い、DOループ処理範囲を指定する
ERROR	強制的にエラーを発生させる
FILE	データ値の出力先（外部ファイル名、リスティング、ログなど）を指定する
FORMAT	指定の変数に出力フォーマットを指定する
GO TO	指定のラベル名が書かれたプログラム位置に次の処理を強制移動させる
IF, Subsetting（サブセットIF）	指定の条件に合致するオブザベーションのみこれ以降の処理に進むことを許可する
IF～THEN	指定の条件に合致する場合の処理を記述する。 条件に合致しない場合の処理は続くELSEステートメントで記述する
INFILE	外部入力ファイル名を指定する
INFORMAT	指定の変数に入力フォーマットを指定する
INPUT	外部ファイルから指定の変数名の値を指定の入力形式で読み取る

KEEP	出力データセットに含める変数を指定する
LABEL	指定の変数に変数ラベルを定義する
Labels, Statement	ラベル名:(コロン)の指定により、GO TOやLINKステートメントにより強制移動させるプログラム位置を示す
LEAVE	DOループ(DOグループ)処理の中で用い、ENDステートメントの次のステートメントまで(ラベルを指定していた場合はラベル位置まで)強制移動させてDOループ処理を抜けることを指示する
LENGTH	作成する変数のタイプと長さを定義する。
LINK	指定のラベル名が書かれたプログラム位置からRETURNステートメントまでの範囲に記述されたサブルーティンに処理を強制移動させた後、移動前の位置に戻るよう指示する
LIST	変数の値をログに書き出す
MERGE	複数のデータセットを横に結合した形でオープンする
OUTPUT	指定の出力データセットに現在処理中のオブザベーションを書き出す
PAGE	改ページを指示する。 DATAステッププログラミングによるレポート作成用ステートメント
PUT	外部ファイルやリスティング出力やログへ指定の変数名の値を指定の出力形式で書き出す
PUTLOG	FILEステートメントの指定する書き出し先に無関係に、ログにメッセージを書き出す
RENAME	変数名を変更する
RETAIN	指定の変数値の現在値を次のオブザベーション処理に変わっても初期化せずに保持するよう宣言する
RETURN	最初のDATAステートメントに処理を戻す。 LINKステートメントからの分岐の場合はLINKの次のステートメントに処理を戻す
RUN	DATAステップの記述の終了を明示的に指定する
SELECT	条件選択のために条件を指定する
SET	データセットを入力のためにオープンする。 複数のデータセットを指定した場合MERGEと異なり縦に結合したイメージでオープンする
SKIP	ブランク行を書き出す。 DATAステッププログラミングによるレポート作成用ステートメント
STOP	DATAステップの処理を中止する
Sum	変数名+式の形で指定する。DATAステップのループ処理中の変数値は右辺の式の値の累積値を値として持つ
UPDATE	UPDATE Master Transact;の形の指定となり必ずBYステートメントと共に指定する。 Masterデータセットの値をTransactデータセットの値で更新する場合に用いる

[別表 5] PROC ステップの種類

プロシジャ名	役割
PROC APPEND	データセット最後のオブザベーションの後に他のデータセットのオブザベーションを追加する
PROC COMPARE	2つのデータセットの内容を比較する
PROC CONTENTS	データセットのコンテンツ情報(オブザベーション数などの一般属性と変数名や変数タイプなどの変数属性)を表示したりデータセットに出力する
PROC COPY	データセットをコピーする
PROC CORR	相関係数を計算する
PROC DATASETS	特定のライブラリに格納されているデータセット名のリストを表示したり、個々のデータセットの名前の変更・削除などを行う。 また、個々のデータセットに関する属性の表示や編集を行う
PROC DELETE	上記DATASETSプロシジャの機能の一部であるデータセットの削除を行う
PROC EXPORT	データセットを外部ファイル形式に変換する。 IMPORTプロシジャの逆の操作を行う
PROC FORMAT	ユーザー定義フォーマットを作成する
PROC FREQ	度数集計を行う。 n次元クロス集計も可能
PROC IMPORT	特定の形式(CSV形式など)の外部ファイルからデータを読み取りデータセットに変換する
PROC MEANS	変数ごとの基本統計(平均値、標準偏差など)を計算する
PROC OPTIONS	オプション設定を変更する
PROC PRINT	データセットの値をリスト表示する
PROC PRINTTO	リスティング出力の出力先をファイルに変更する
PROC RANK	変数値の順序を計算する
PROC SCORE	スコア係数とデータ値の積和によるスコアを計算する

PROC SORT	オブザベーションを指定の変数値の順に並び替える
PROC SQL	SQL言語によるデータ検索・加工を行う
PROC SUMMARY	MEANSと同じく変数ごとの基本統計（平均値、標準偏差など）を計算する
PROC TABULATE	度数・平均・百分率を含む多重クロス集計表を作成する
PROC TRANSPOSE	データセットを転置（行と列を交換）する
PROC UNIVARIATE	変数ごとの基本統計（平均値、標準偏差など）を計算する。 MEANS,SUMMARYより詳細

[別表 6] グローバルステートメント一覧

ステートメント名	役割
Comment（コメント）	*（アスタリスク記号）で始まるステートメント。 任意のコメントを記述できる
ENDSAS	WPSセッションを終了する
FILENAME	外部ファイル参照名を定義する
FILENAME_DDE	DDE（動的データ交換）機能によるアプリケーションとWPS間のデータ交換を行う
FILENAME_EMAIL	WPSからEmailを送る
FOOTNOTE	フットノートテキストを定義する
%INC	%INCステートメントの省略形
%INCLUDE	外部ファイルに書かれたソースコードを読み込み実行する
LIBNAME	データセットライブラリ参照名を定義する
MISSING	数値タイプ変数の入力値に書かれた欠損を表す文字を指定する。
ODS EXCLUDE	ODS機能による選択リストの中から除外項目を選ぶ
ODS HTML	HTML出力を管理する。 また、どのオブジェクトをHTML出力するかを制御する
ODS LISTING	特定項目のリスティング出力を管理する
ODS OUTPUT	特定項目のデータセット出力を管理する
ODS SELECT	ODS機能による選択リストの中から選択項目を選ぶ
ODS SHOW	ODS選択リストの表示
ODS TRACE	ODS出力に関するメッセージをログに書き出すかどうかを切り替える
OPTIONS	各種オプションの設定を変更する
RUN	DATAステップ、PROCステップのステートメントの指定を終了しステップを実行する
TITLE	タイトルテキストを定義する
PAGE	ログを新しいページに切り替える
SKIP	ログにブランク行を1行書いて改行する
X	コマンドプロンプトを呼び出す

[別表 7] 演算子一覧

分類	シンボル	別表記	意味	例
算術	+		足し算	c=a+b
	−		引き算	d=10−x
	*		掛け算	y=2*x
	/		割り算	z=x/y
	**		累乗計算	value=2**(x+1)
論理	&	AND	かつ	if (a=1) & (b=10)
	|	OR	または	if (a=1) | (b=10)
	^	NOT	否定	if ^(z=1)
比較	=	EQ	等しい	if a=b
	^=	NE	等しくない	if a NE b
	<	LT	より小さい（未満）	if a<b
	<=	LE	等しいかより小さい（以下）	if a LE 5
	>	GT	より大きい（超）	if b GT 15*x
	>=	GE	等しいかより大きい（以上）	if b>=16*x
符号	+		プラス（正の数）	y=+1

	-		マイナス(負の数)	y=-1
最小	><		小さい方の値	z=(x><y)
最大	<>		大きい方の値	H=(x<>y<>z)
文字列検索		IN	いずれかの文字列に一致する	if name in ("abc" "de")
文字列連結	||		左右の文字列をつなぐ	z="ABC"||"DEFG"||"HI"
文字列比較	:		比較演算子と共に用いる。 左右の文字列の長さを短い方に揃えてから比較する	a="12345"; b="123"; if a=:b (真)

[別表 8] 関数一覧

分類	関数名	意味	例
三角	ARCOS	アークコサイン	y=arcos(x);
	ARSIN	アークサイン	y=arsin(x);
	ATAN	アークタンジェント	y=atan(x);
	COS	コサイン	y=cos(x);
	COSH	ハイパボリックコサイン	y=cosh(x);
	SIN	サイン	y=sin(x);
	SINH	ハイパボリックサイン	y=sinh(x);
	TAN	タンジェント	y=tan(x);
	TANH	ハイパボリックタンジェント	y=tanh(x);
数学	ABS	絶対値	y=abs(x);
	EXP	指数	y=exp(x);
	LOG	自然対数(底=e)	y=log(x);
	LOG10	常用対数(底=10)	y=log10(x);
	LOG2	2を底とする対数	y=log2(x);
	MOD	割り算の余りを返す	y=mod(x,100);
	POW	累乗。 **演算子と同じ	x=pow(100,2);
	SIGN	符号を返す	s=sign(-156);
	SQRT	平方根	y=sqrt(x);
数値丸め	CEIL	整数値に切り上げ	y=ceil(x);
	FLOOR	整数値に切り捨て	y=floor(x);
	FUZZ	最も近い整数値との差が1E-12以内であればその整数値を返す	x=fuzz(x);
	INT	整数部分を取り出す	x_int=int(x);
	ROUND	四捨五入して指定の桁位置に丸める	x=round(125.321,0.1);
	ROUNDZ	四捨五入して指定の桁位置に丸める、fuzzing処理を行わない	x=roundz(125.321,0.1);
統計	CSS	修正済平方和	x=css(5,10,20,16,0,5);
	CV	変動係数(%表示)	x=cv(5,10,20,16,0,5);
	KURTOSIS	尖度	x=kurtosis(5,10,20,16,0,5);
	MAX	最大値	x=max(5,10,20,16,0,5);
	MEAN	平均値	x=mean(5,10,20,16,0,5);
	MIN	最小値	x=min(5,10,20,16,0,5);
	N	非欠損値の数を返す	n=n(1,3,.,5,10);
	NMISS	欠損値の数を返す	nmiss=nmiss(1,3,.,5,10);
	RANGE	範囲	x=range(5,10,20,16,0,5);
	SKEWNESS	歪度	x=skewness(5,10,20,16,0,5);
	STD	標準偏差	x=std(5,10,20,16,0,5);
	SUM	合計	x=std(5,10,20,16,0,5);
	USS	修正前平方和	x=uss(5,10,20,16,0,5);
	VAR	不偏分散	x=var(5,10,20,16,0,5);
配列	DIM	配列の要素数を返す	do i=1 to dim(z);
	HBOUND	定義された配列の最後の要素の参照番号を返す	do i=lbound(arrayname) to hbound(arrayname);
	LBOUND	定義された配列の最初の要素の参	do i=lbound(arrayname) to hbound(arrayname);

		照番号を返す	
日付と時間	DATE	今日の日付を返す（1960年１月１日を起点とした経過日数）、TODAYと同じ	today=date();
	DATEJUL	ユリウス暦表示)から標準の日付値に変換	d=datejul(2008366);
	DATEPART	日時値から日付値部分を取り出す	date=datepart("01JAN2008:12:10:00"DT);
	DATETIME	現在の日時値を返す（1960年１月１日を起点とした経過秒数）	now=datetime();
	DAY	日付値または日時値から日部分を取り出す	day=day("10AUG2008"D);
	DHMS	日付、時、分、秒から日時値を作成	val=dhms("10AUG2008"D,12,10,30);
	HMS	時、分、秒から時間値を作成	time=hms(12,0,0);
	HOUR	時間値または日時値から時間部分を取り出す	h=hour("01JAN2008:12:10:00"DT);
	INTCK	開始時点から終了時点までの経過時間をさまざまな時間単位で計算	keika_month=intck("month","10JAN2008"D,"25AUG2008"D);
	INTNX	指定の時間経過後の時点を返す	after=intnx("month","10JAN2008"D,3,"END");
	JULDATE	日付値を5桁のユリウス暦表現(yyddd)に変換	juldate=juldate("01JAN2008"D);
	JULDATE7	日付値を7桁のユリウス暦表現(yyyyddd)に変換	juldate=juldate7("01JAN2008"D);
	MDY	月、日、年から日付値を作成	date1=mdy(12,31,2007);
	MINUTE	時間値または日時値から分部分を取り出す	m=minute("01JAN2008:12:10:00"DT);
	MONTH	日付値または日時値から分部分を取り出す	month=month("01JAN2008:12:10:00"DT);
	QTR	日付値または日時値から四半期部分を取り出す	qtr=qtr("10AUG2008"D);
	SECOND	時間値または日時値から秒部分を取り出す	s=second("01JAN2008:12:10:00"DT);
	TIME	現在の時間値を返す	now=time();
	TIMEPART	日時値から時間部分を取り出す	time=timepart("01JAN2008:12:10:00"DT);
	TODAY	今日の日付を返す（1960年１月１日を起点とした経過日数）、DATEの別名	today=today();
	WEEKDAY	日付値または日時値から曜日を取り出す	week=weekday("10AUG2008"D);
	YEAR	日付値または日時値から年部分を取り出す	year=year("10AUG2008"D);
	YYQ	年、四半期から日付値を作成	yyq=yyq(2008,1);
ビット演算	BAND	ビット単位のAND	x=band(9Fx,11x);
	BLSHIFT	ビット列を左シフトする	x=blshift(01x,1);
	BNOT	ビット単位のNOT	x=bnot(01x);
	BOR	ビット単位のOR	x=bor(9fx,90x);
	BRSHIFT	ビット列を右シフトする	x=brshift(01x,31);
	BXOR	ビット単位のXOR	bxor(01x,55x);
マクロ	CALL EXECUTE	DATAステップの中から実行ルーティンを呼び出す	if x=1 then call execute("proc print;run;");
	CALL SYMDEL	グローバルマクロ変数を削除	(動作しない)
	CALL SYMPUT	DATAステップ変数値をマクロ変数値に割り当てる	call symput("mvar",char);
	SYMGET	マクロ変数値をDATAステップ変数値に変換	c=symget("c");
文字	BYTE	ASCII文字を返す、RANKの逆	char=byte(40x);
	COMPBL	連続するブランクを１個に圧縮する	char=compbl(a||" "||b);
	COMPRESS	指定の文字（デフォルトはブランク）を除外する	char=compress(a||b);
	CONTAINS	指定の部分文字列の有無をチェックする	check=contains(c,"abc");

	INDEX	文字値から指定の文字列の開始位置を返す	position=index("abcabdefgh","bde");
	INDEXC	文字値から指定のいずれかの文字の開始位置を返す	position=indexc("abcabdefgh","bde");
	INDEXW	文字値から指定のワードの開始位置を返す	position=indexw("abc,abde,fgh","abde",",");
	LEFT	文字値を左詰する	char=left("　　abc　");
	LENGTH	文字値の長さを返す	len=length(compress(x));
	LIKE	正規表現のあいまい検索	if like(c1,"_ABC%") then put "OK";
	LOWCASE	小文字変換	low=lowcase("ABC");
	MAXC	ブランクを除く最大の文字値	maxchar=maxc("Z1","ABC");
	MINC	ブランクを除く最小の文字値	minchar=minc("Z1","ABC");
	PROPCASE	特殊文字をデリミタとして語単位に先頭は大文字化、残りの文字は小文字化する	c=propcase("new/software@world","/@ ");
	RANK	1文字値のシーケンス番号を返す、BYTEの逆	seq=rank("z");
	REPEAT	1文字の繰り返し文字列を作成	char=repeat("z",10);
	REVERSE	文字値を逆順に並べ替える	rev_c=reverse(c);
	RIGHT	文字値を右詰する	char=right("　　abc　");
	SCAN	区切り文字で区切られたn番目の文字列を抽出	c=scan("new/software@world",2,"/@ ");
	SUBSTR	部分文字列の抽出	c=substr("abcdefg",3,2);
	TRANSLATE	特定の文字を別の文字に変換する	new=translate("abcdefgfcded","150","ceg");
	TRANWRD	特定の文字列を別の文字列に変換する	new=tranwrd("abcdefgfcded","cd","99");
	TRIM	文字値の後ろ側のブランクを削除する	c=trim(a)||trim(b);
	TRIMN	欠損値に対して長さ0の文字値を返す以外はTRIMと同じ	c=trimn(a);
	UPCASE	大文字変換	up=upcase("Abc");
	VERIFY	文字値が指定の文字のみ含むかどうかをチェック	chk=verify("abcdefgfcded","abcdefg");
乱数	CALL RANCAU	コーシー乱数（シードの詳細制御可能）	call rancau(seed,x);
	CALL RANNOR	正規乱数（シードの詳細制御可能）	call rannor(seed,x);
	CALL RANUNI	一様乱数（シードの詳細制御可能）	call ranuni(seed,x);
	RANCAU	コーシー乱数	x=rancau(seed);
	RANNOR	正規乱数	x=rannor(seed);
	RANUNI	一様乱数、UNIFORMと同じ	x=ranuni(seed);
	UNIFORM	一様乱数、RANUNIの別名	x=uniform(seed);
データセット操作	ATTRC	文字型属性の値をとる	dslabel=attrc(dsid,"label"):
	ATTRN	数値型属性の値をとる	nobs=attrn(dsid,"nobs");
	CLOSE	データセットをクローズ	dsid=close("work.a");
	EXIST	データセットやカタログが存在するかどうかをチェック	rc=exist(work.a,data);
	FETCH	オープンしたデータセットのオブザベーション読み取りポインタを次のオブザベーションに移動する	rc=fetch(dsid);
	FETCHOBS	オープンしたデータセットのオブザベーション読み取りポインタを指定のオブザベーションに移動する	rc=fetchobs(dsid,5,abs);
	GETVARC	FETCHされているオブザベーションの文字変数値を読み取る	cval=getvarc(dsid,varnum(dsid,"varc");
	GETVARN	FETCHされているオブザベーションの数値変数値を読み取る	xval=getvarn(dsid,varnum(dsid,"varx");
	LIBREF	ライブラリ参照名の存在をチェック（値0が返ると存在を意味する）	rc=libref("work");

	OPEN	データセットをオープン	dsid=open("work.a");
	PATHNAME	データライブラリ参照名またはファイル参照名の物理パスを返す	path1=pathname("work");
	SYSMSG	ファイルアクセス時のエラーメッセージまたは警告メッセージを獲得	msg=sysmsg();
	VARFMT	変数に定義されているフォーマット名を返す	fmt=varfmt(dsid,varnum(dsid,"a"));
	VARINFMT	変数に定義されているインフォーマット名を返す	infmt=varinfmt(dsid,varnum(dsid,"a"));
	VARLABEL	変数に定義されているラベル名を返す	label=varlabel(dsid,varnum(dsid,"a"));
	VARLEN	変数に定義されている長さを返す	len=varlen(dsid,varnum(dsid,"a"));
	VARNAME	変数に定義されている変数名を返す	name1=varname(dsid,1);
	VARNUM	変数名の定義されている番号を返す	num=varnum(dsid,"a"));
	VARTYPE	変数に定義されているタイプを返す	type=vartype(dsid,varnum(dsid,"a"));
その他	CALL SYSTEM	OSコマンドを呼び出す	call system("dir c:¥");
	CHOOSEC	文字列リストから指定の番号の文字列を抽出する	a=choosec(2,"abc","de","fgh");
	CHOOSEN	数値リストから指定の番号の数値を抽出する	x=choosen(5,120,35,11,16,280);
	DIF	nオブザベーション前の値との差をとる	d2=dif2(x);
	GETOPTION	オプション設定値を返す	ls=getoption("linesize");
	INPUT	インフォーマットを用いて値を変換	num=input("100",3.);
	LAG	nオブザベーション前の値を返す	l2=lag2(x);
	MISSING	欠損値かどうかをチェック	if missing(x) then put "MISSING VALUE";
	PUT	フォーマットを用いて値を変換	char=put(100,3.);
	SLEEP	実行を休止する	sleep=sleep(10,1);
	SOUNDEX	英語のみ関係する文字列分類アルゴリズム	a=soundex("hello");
	SOUNDSLIKE	2つの文字列を英語の発音で比較	r=soundslike("hello","helow");
	SPEDIS	2つの文字列のレーベンシュタイン距離を返す	distance=spedis("test","twist");
	SYSPARM	SYSPARM=オプション指定値を返す	sysparm=sysparm();
	SYSPROD	そのプロダクトのライセンス有無をチェックする	prod_chk=sysprod("WPS");
	SYSTEM	OSコマンドを呼び出しシステムリターンコードを返す	rc=system("dir c:¥");

[別表 9] フォーマット一覧

分類	フォーマット名	意味	wとdの範囲（デフォルト）	例	結果
数値コード変換	BINARYw.	数値をバイナリコード(0 or 1)で書き出す	1-64 (8)	x=256;put x binary10.;	0100000000
	HEXw.	数値を 16 進数で書き出す	1-16 (8)	x=512;put x hex8.;	00000200
	IBw.d	数値を整数バイナリ形式で書き出す	1-8 (4)	x=put(12345,ib8.);put x $hex16.;	3930000000000000
	IBRw.d	数値をOS環境下依存の整数バイナリ形式で書き出す	1-8 (4)	x=put(12345,ibr8.);put x $hex16.;	3930000000000000
	IEEEw.d	数値を IEEE 浮動小数点で書き出す	1-8 (8)	x=put(1,ieee8.);put x $hex16.;	3FF0000000000000
	OCTALw.	数値を 8 進数表記で書き出す	1-24 (3)	x=123456789012345;put x octal24.;	000000003404420603357571

	PDw.d	数値をパック 10 進数形式で書き出す	1-16 (1)	x=put(999,pd4.);put x hex8.;	00000999
	PDJULGw.	数値をz/OS上のパックジュリアン日付値 yyyydddF の形式で書き出す	3-16 (4)	x=put("31DEC2008"D,pdjulg8.);put x $hex16.;	000000002008366F
	PDJULIw.	数値をz/OS上の16進パックジュリアン日付値 ccyydddF の形式で書き出す	3-16 (4)	x=put("31DEC2008"D,pdjuli8.);put x $hex16.;	000000000108366F
	PIBw.d	数値をOS環境下依存の正の整数バイナリ形式で書き出す	1-8 (1)	x=put(2,pib1.);put x $hex2.;	02
	PIBRw.d	数値を Window OS 環境の正の整数バイナリ形式で書き出す	1-8 (1)	x=put(2,pib1.);put x $hex2.;	02
	PKw.d	数値を符号なしパック10進数形式で書き出す。	1-16 (1)	x=put(2,pk1.);put x $hex2.;	02
	RBw.d	数値をOSに依存する実数バイナリ形式で書き出す	2-8 (4)	x=put(2,rb2.);put x $hex4.;	0040
	S370FFw.d	数値をz/OS上の EBCDIC 文字形式で書き出す	1-32 (12)	x=put(1234,s370ff4.);put x $hex8.;	F1F2F3F4
	S370FIBw.d	数値をz/OS上の整数バイナリ形式で書き出す	1-8 (4)	x=put(1234,s370fib4.);put x $hex8.;	000004D2
	S370FIBUw.d	数値をz/OS上の符号なしバイナリ形式で書き出す	1-8 (4)	x=put(1234,s370fib4.);put x $hex8.;	000004D2
	S370FPDw.d	数値をz/OS上のパック10進数形式で書き出す	1-16 (1)	x=put(1234,s370fpd4.);put x $hex8.;	0001234C
	S370FPDUw.d	数値をz/OS上の符号なしパック 10 進数形式で書き出す	1-16 (1)	x=put(1234,s370fpdu4.);put x $hex8.;	0001234F
	S370FPIBw.d	数値をz/OS上の正の整数バイナリ形式で書き出す	1-8 (4)	x=put(1234,s370fpib4.);put x $hex8.;	000004D2
	S370FRBw.d	数値をz/OS上の実数バイナリ形式で書き出す	2-8 (4)	x=put(2,s370frb2.);put x $hex4.;	4120
	S370FZDw.d	数値をz/OS上のゾーン 10 進数形式で書き出す	1-32 (8)	x=put(1234,s370fzd4.);put x $hex8.;	F1F2F3C4
	S370FZDLw.d	数値をz/OS上の符号つきゾーン 10 進数形式で書き出す	1-32 (8)	x=put(1234,s370fzdl6.);put x $hex16.;	C0F0F1F2F3F4
	S370FZDSw.d	数値をz/OS上の符号を分離したゾーン 10 進数形式で書き出す	2-32 (8)	x=put(1234,s370fzds6.);put x $hex16.;	4EF0F1F2F3F4
	S370FZDTw.d	数値をz/OS上の符号を分離して後ろにつけたゾーン 10 進数形式で書き出す	2-32 (8)	x=put(1234,s370fzdt6.);put x $hex16.;	F0F1F2F3F44E
	S370FZDUw.d	数値をz/OS上の符号なしゾーン 10 進数形式で書き出す	1-32 (8)	x=put(1234,s370fzdu6.);put x $hex16.;	F0F0F1F2F3F4
	ZDw.d	数値をOS環境下依存のゾーン 10 進数形式で書き出す	1-16 (8)	x=put(12.5,zd5.1);put x $hex32.;	3030313245
数値編集	BESTw.d	数値を指定の長さで可能な限り詳細なフォーマットで書き出す	1-32 (12)	x=1111111111111111;put x best12.;	1.1111111E17

	COMMAw.d	整数部分は 3 桁おきにカンマを追加し、指定の小数点桁数まで書き出す	1-32 (6)	x=1000000;put x comma12.2;	1,000,000.00
	COMMAXw.d	整数部分は 3 桁おきにピリオドを追加し、小数点はカンマ表示し、指定の小数点桁数まで書き出す	1-32 (6)	x=1000000;put x commax12.2;	1.000.000,00
	Dw.d	変動範囲の大きな数値に対してなるだけ小数点位置を揃えて書き出す	1-32 (12)	x=12.518;put x d6.1;	12.52
	DOLLARw.d	数値を先頭に$、3 桁おきにカンマを付加して書き出す	2-32 (6)	x=1250.50;put x dollar12.2;	$1,250.50
	DOLLARXw.d	数値を先頭に$、3 桁おきにピリオドを付加し、小数点にカンマを使って書き出す	2-32 (6)	x=1250.50;put x dollarx12.4;	$1.250,5000
	Ew.	数値を科学(浮動)小数点法で書き出す	7-32 (12)	x=123.456789;put x e12.;	1.23457E+02
	FLOATw.d	数値を単精度浮動小数点で書き出す。	4-4 (4)	x=put(123.456789,float4.);put x $hex8.;	E0E9F642
	FRACTw.	数値を分数表示で書き出す	4-32 (10)	x=0.3;put x fract10.;	3/10
	NEGPARENw.d	数値をカンマ編集し、かつ、負の値の場合はカッコで囲む	1-32 (6)	x=-12567.8;put x negparen8.0.;	(12,568)
	NUMXw.d	数値を小数点記号としてカンマを用いて書き出す	1-32 (12)	x=-12567.8;put x numx8.1;	-12567,8
	PERCENTw.d	数値を%記号をつけたパーセンテージ表現で書き出す	4-32 (6)	x=0.0512;put x percent6.1;	5.1%
	PVALUEw.d	数値を p 値(帰無仮説を誤って棄却してしまう確率)表示形式で書き出す	3-32 (6)	p=0.0000000001;put p pvalue6.4;	<.0001
	ROMANw.	数値をローマ数字で書き出す	2-32 (6)	x=2008;put x roman12.;	MMVIII
	SSNw.	数値を社会保障番号(Social Security Number)形式に書き出す	11-11 (11)		
	w.d	数値を全体で w 文字、少数点以下 d 桁の形式で書き出す	1-32 (1)	x=10.091;put x 5.1;	10.1
	WORDFw.	数値を英語の数の読み方で分数表現つきで書き出す	5-32767 (10)	x=12.25;put x wordf64.;	twelve and 25/100
	WORDSw.	数値を英語の数の読み方で書き出す	5-32767 (10)	x=12.25;put x words64.;	twelve and twenty-five hundredths
	Zw.d	数値を全体で w 文字、少数点以下 d 桁の形式で書き出す。 先頭の 0 を書き出す点で w.d と異なる	1-32 (1)	x=10.091;put x z5.1;	010.1
日付・時間	DATEw.	日付値を ddmmmyy または ddmmmyyyy の形式で書き出す	5-9 (7)	x=0;put x date9.;	01JAN1960

DATEAMPMw.d	日時値を ddmmmyy:hh:mm:ss.ss AM または ddmmmyy:hh:mm:ss.ss PM の形式で書き出す	7-40 (19)	x=0;put x dateampm19.;	01JAN60:12:00:00 AM
DATETIMEw.d	日時値を ddmmmyy:hh:mm:ss.ss の形式で書き出す	7-40 (19)	x=0;put x datetime19.;	01JAN1960:00:00:00
DAYw.	日付値から日付部分を取り出し dd の形式で書き出す	2-32 (2)	x=0;put x day2.;	1
DDMMYYw.	日付値を ddmmyy または ddmmyyyy または dd/mm/yy または dd/mm/yyyy の形式で書き出す	2-10 (8)	x=0;put x ddmmyy10.;	01/01/1960
DDMMYYBw.	日付値を dd mm yy または dd mm yyyy の形式で書き出す	2-10 (8)	x=0;put x ddmmyyb10.;	01 01 1960
DDMMYYCw.	日付値を dd:mm:yy または dd:mm:yyyy の形式で書き出す	2-10 (8)	x=0;put x ddmmyyc10.;	01:01:1960
DDMMYYDw.	日付値を dd-mm-yy または dd-mm-yyyy の形式で書き出す	2-10 (8)	x=0;put x ddmmyyd10.;	01-01-1960
DDMMYYNw.	日付値を ddmmyy または ddmmyyyy の形式で書き出す	2-8 (8)	x=0;put x ddmmyyn8.;	01011960
DDMMYYPw.	日付値を dd.mm.yy または dd.mm.yyyy の形式で書き出す	2-10 (8)	x=0;put x ddmmyyp10.;	01.01.1960
DDMMYYSw.	日付値を dd/mm/yy または dd/mm/yyyy の形式で書き出す	2-10 (8)	x=0;put x ddmmyys10.;	01/01/1960
DOWNAMEw.	日付値から曜日を書き出す	1-32 (9)	date="01jan2008"d;put date downame9.;	Tuesday
DTDATEw.	日時値を ddmmmyy または ddmmmyyyy の形式で書き出す	5-9 (7)	x="01JAN1960:00:00:00"DT;put x dtdate9.;	01JAN1960
DTMONYYw.	日時値を mmmyy または mmmyyyy の形式で書き出す	5-7 (7)	x="01JAN1960:00:00:00"DT;put x dtmonyy7.;	JAN1960
DTWKDATXw.	日時値を day-of week, dd name-of month yy または day-of week, dd name-of month yyyy の形式で書き出す	3-37 (29)	x="01JAN1960:00:00:00"DT;put x dtwkdatx29.;	Friday, 1 January 1960
DTYEARw.	日時値を yy または yyyy の形式で書き出す	2-4 (4)	x="01JAN1960:00:00:00"DT;put x dtyear4.;	1960
DTYYQCw.	日時値を yy:q または yyyy:q の形式で書き出す	4-6 (4)	x="01JAN1960:00:00:00"DT;put x dtyyqc4.;	60:1
HHMMw.d	時間値を hh:mm.mm の形式で書き出す	2-20 (5)	x=60;put x hhmm.;	0:01
HOURw.d	時間値を hh.hh の形式で書き出す	2-20 (2)	x=3600;put x hour2.;	1
JULDAYw.	日付値からジュリアン日付値の日付部分 ddd を書き出す	3-32 (3)	x="31DEC2008"D;put x julday3.;	366

JULIANw.	日付値をジュリアン日付値 yyddd または yyyyddd の形式で書き出す	5-7 (5)	x="31DEC2008"D;put x julian5.;	08366
MMDDYYw.	日付値を mmddyy , mmddyyyy , mm/dd/yy または mm/dd/yyyy の形式で書き出す	2-10 (8)	x="31DEC2008"D;put x mmddyy10.;	12/31/2008
MMDDYYBw.	日付値を mm dd yy または mm dd yyyy の形式で書き出す	2-10 (8)	x="31DEC2008"D;put x mmddyyb10.;	12 31 2008
MMDDYYCw.	日付値を mm:dd:yy または mm:dd:yyyy の形式で書き出す	2-10 (8)	x="31DEC2008"D;put x mmddyyc10.;	12:31:2008
MMDDYYDw.	日付値を mm-dd-yy または mm-dd-yyyy の形式で書き出す	2-10 (8)	x="31DEC2008"D;put x mmddyyd10.;	12-31-2008
MMDDYYNw.	日付値を mmddyy または mmddyyyy の形式で書き出す	2-8 (8)	x="31DEC2008"D;put x mmddyyn8.;	12312008
MMDDYYPw.	日付値を mm.dd.yy または mm.dd.yyyy の形式で書き出す	2-10 (8)	x="31DEC2008"D;put x mmddyyp10.;	12.31.2008
MMDDYYSw.	日付値を mm/dd/yy または mm/dd/yyyy の形式で書き出す	2-10 (8)	x="31DEC2008"D;put x mmddyys10.;	12/31/2008
MMSSw.d	時間値を mm:ss.ss の形式で書き出す	2-20 (5)	x="01:10:30"T;put x mmss8.;	70:30
MMYYw.	日付値を mmMyy または mmMyyyy の形式で書き出す	5-32 (7)	x="31DEC2008"D;put x mmyy7.;	12M2008
MMYYCw.	日付値を mm:yy または mm:yyyy の形式で書き出す	5-32 (7)	x="31DEC2008"D;put x mmyyc7.;	12:2008
MMYYDw.	日付値を mm-yy または mm-yyyy の形式で書き出す	5-32 (7)	x="31DEC2008"D;put x mmyyd7.;	12-2008
MMYYNw.	日付値を mmyy または mmyyyy の形式で書き出す	4-32 (6)	x="31DEC2008"D;put x mmyyn6.;	122008
MMYYPw.	日付値を mm.yy または mm.yyyy の形式で書き出す	5-32 (7)	x="31DEC2008"D;put x mmyyp7.;	12.2008
MMYYSw.	日付値を mm/yy または mm/yyyy の形式で書き出す	5-32 (7)	x="31DEC2008"D;put x mmyys7.;	12/2008
MONNAMEw.	日付値から月名を書き出す	1-32 (9)	x="31DEC2008"D;put x monname9.;	December
MONTHw.	日付値から月部分を取り出し mm の形式で書き出す	1-32 (9)	x="31DEC2008"D;put x month2.;	12
MONYYw.	日付値を mmmyy または mmmyyyy の形式で書き出す	5-7 (5)	x="31DEC2008"D;put x monyy5.;	DEC08
QTRw.	日付値を四半期 q 形式で書き出す	1-32 (1)	x="31DEC2008"D;put x qtr1.;	4
QTRRw.	日付値を四半期 qr 形式で書き出す	3-32 (3)	x="31DEC2008"D;put x qtrr3.;	IV
TIMEw.d	時間値または日時値を hh:mm:ss.ss の形式で書き出す	2-20 (8)	x="01JAN1960:00:00:00"DT;put x time8.;	0:00:00

TIMEAMPMw.d	時間値または日時値を hh:mm:ss.ss AM または、hh:mm:ss.ss PM の形式で書き出す	2-20 (11)	x="01JAN1960:00:00:00"DT;put x timeampm11.;	12:00:00 AM
TODw.	日時値から hh:mm:ss.ss の形式で時間部分のみを書き出す	2-20 (8)	x="01JAN1960:00:00:00"DT;put x tod8.;	00:00:00
WEEKDATEw.	日付値を 曜日, 月名 dd, yy または 曜日, 月名 dd, yyyy の形式で書き出す	3-37 (29)	x="31DEC2008"D;put x weekdate3.;	Wed
WEEKDATXw.	日付値を 曜日, dd 月名 yy または 曜日, dd 月名 yyyy の形式で書き出す	3-37 (29)	x="31DEC2008"D;put x weekdatx29.;	Wednesday, 31 December 2008
WEEKDAYw.	日付値を 曜日を表す整数で書き出す	1-32 (1)	x="31DEC2008"D;put x weekday1.;	4
WORDDATEw.	日付値を 月名 dd, yy または 月名 dd, yyyy の形式で書き出す	3-32 (18)	x="31DEC2008"D;put x worddate18.;	December 31, 2008
WORDDATXw.	日付値を dd 月名 yy または dd 月名 yyyy の形式で書き出す	3-32 (18)	x="31DEC2008"D;put x worddatx18.;	31 December 2008
YEARw.	日付値から年部分を取り出し yy または yyyy の形式で書き出す	2-32 (4)	x="31DEC2008"D;put x year4.;	2008
YYMMw.	日付値を yyMmm または yyyyMmm の形式で書き出す	5-32 (7)	x="31DEC2008"D;put x yymm7.;	2008M12
YYMMCw.	日付値を yy:mm または yyyy:mm の形式で書き出す	5-32 (7)	x="31DEC2008"D;put x yymmc7.;	2008:12
YYMMDw.	日付値を yy-mm または yyyy-mm の形式で書き出す	5-32 (7)	x="31DEC2008"D;put x yymmd7.;	2008-12
YYMMNw.	日付値を yymm または yyyymm の形式で書き出す	4-32 (6)	x="31DEC2008"D;put x yymmn6.;	200812
YYMMPw.	日付値を yy.mm または yyyy.mm の形式で書き出す	5-32 (7)	x="31DEC2008"D;put x yymmp7.;	2008.12
YYMMSw.	日付値を yy/mm または yyyy/mm の形式で書き出す	5-32 (7)	x="31DEC2008"D;put x yymms7.;	2008/12
YYMMDDw.	日付値を yymmdd , yyyymmdd , yy/mm/dd または yyyy/mm/dd の形式で書き出す	2-10 (8)	x="31DEC2008"D;put x yymmdd10.;	2008-12-31
YYMMDDBw.	日付値を yy mm dd または yyyy mm dd の形式で書き出す	2-10 (8)	x="31DEC2008"D;put x yymmddb10.;	2008 12 31
YYMMDDCw.	日付値を yy:mm:dd または yyyy:mm:dd の形式で書き出す	2-10 (8)	x="31DEC2008"D;put x yymmddc10.;	2008:12:31
YYMMDDDw.	日付値を yy-mm-dd または yyyy-mm-dd の形式で書き出す	2-10 (8)	x="31DEC2008"D;put x yymmddd10.;	2008-12-31
YYMMDDNw.	日付値を yymmdd または yyyymmdd の形式で書き出す	2-8 (8)	x="31DEC2008"D;put x yymmddn8.;	20081231
YYMMDDPw.	日付値を yy.mm.dd または yyyy.mm.dd の形式で書き出す	2-10 (8)	x="31DEC2008"D;put x yymmddp10.;	2008.12.31

	YYMMDDSw.	日付値を yy/mm/dd または yyyy/mm/dd の形式で書き出す	2-10 (8)	x="31DEC2008"D;put x yymmdds10.;	2008/12/31
	YYMONw.	日付値を yymmm または yyyymmm の形式で書き出す	5-32 (7)	x="31DEC2008"D;put x yymon7.;	2008DEC
	YYQw.	日付値を yyQq または yyyyQq の形式で書き出す	4-32 (6)	x="31DEC2008"D;put x yyq6.;	2008Q4
	YYQCw.	日付値を yy:q または yyyy:q の形式で書き出す	4-32 (6)	x="31DEC2008"D;put x yyqc6.;	2008:4
	YYQDw.	日付値を yy-q または yyyy-q の形式で書き出す	4-32 (6)	x="31DEC2008"D;put x yyqd6.;	2008-4
	YYQNw.	日付値を yyq または yyyyq の形式で書き出す	3-32 (5)	x="31DEC2008"D;put x yyqn6.;	20084
	YYQPw.	日付値を yy.q または yyyy.q の形式で書き出す	4-32 (6)	x="31DEC2008"D;put x yyqp6.;	2008.4
	YYQSw.	日付値を yy/q または yyyy/q の形式で書き出す	4-32 (6)	x="31DEC2008"D;put x yyqs6.;	2008/4
	YYQRw.	日付値を yyQqr または yyyyQqr の形式で書き出す	6-32 (8)	x="31DEC2008"D;put x yyqr6.;	08QIV
	YYQRCw.	日付値を yy:qr または yyyy:qr の形式で書き出す	6-32 (8)	x="31DEC2008"D;put x yyqrc6.;	08:IV
	YYQRDw.	日付値を yy-qr または yyyy-qr の形式で書き出す	6-32 (8)	x="31DEC2008"D;put x yyqrd6.;	08-IV
	YYQRNw.	日付値を yyqr または yyyyqr の形式で書き出す	5-32 (7)	x="31DEC2008"D;put x yyqrn.;	2008IV
	YYQRPw.	日付値を yy.qr または yyyy.qr の形式で書き出す	6-32 (8)	x="31DEC2008"D;put x yyqrp6.;	08.IV
	YYQRSw.	日付値を yy/qr または yyyy/qr の形式で書き出す	6-32 (8)	x="31DEC2008"D;put x yyqrs6.;	08/IV
文字編集	$CHARw.	文字列を先頭のブランクは詰めずにそのまま書き出す	1-32767 (8)	c="|"||put(" ABC ",$char7.)||"|";put c;		ABC	
	$Fw.	文字列を文字列として書き出す。 $w.と同じ	1-32767 (1)	c="ABC";put c $F3.;	ABC
	$QUOTEw.	文字列をダブルクオテーションで囲んで書き出す	2-32767 (8)	c="ABC";put c $quote5.;	"ABC"
	$REVERJw.	文字列を逆順に書き出す。先頭および後ろのブランクは取り除かない	1-32767 (1)	c=put(" ABC ",$char7.);put "|" c $reverj7. "|";		CBA	
	$REVERSw.	文字列を逆順に書き出す。先頭のブランクは取り除かないが後ろは除く	1-32767 (1)	c=put(" ABC ",$char7.);put "|" c $revers7. "|";	|CBA	
	$UPCASEw.	文字列を大文字化して書き出す	1-32767 (1)	c="abc";put c $upcase3.;	ABC
	$w.	文字列を文字列として書き出す。 $Fw.と同じ	1-32767 (1)	c="ABC";put c $3.;	ABC
文字コード変換	$ASCIIw.	文字列を ASCII コード文字で書き出す。ただし PC 環境では$CHARw.と同じ	1-32767 (1)	c=put("ABC",$ascii3.);put c $hex6.;	414243

	$BINARYw.	文字列をバイナリコード(0 or 1)で書き出す	1-32767 (1)	c="ABC";put c $binary24.;	010000010100001001000011
	$EBCDICw.	文字列を EBCDIC コード文字で書き出す	1-32767 (1)	c=put("ABC",$ebcdic3.);put c $hex6.;	C1C2C3
	$HEXw.	文字列を 16 進コードで書き出す	1-32767 (1)	c="ABC";put c $hex6.;	414243
	$OCTALw.	文字列を 8 進コードで書き出す	1-24 (3)	c="ABC";put c $octal6.;	101102

注:数値フォーマットに共通: w.d の d を指定した場合、小数点以下 d 桁を表示する。

[別表 10] インフォーマット一覧

分類	フォーマット名	意味	w と d の範囲 (デフォルト)	例	結果
数値	BESTw.d	数値を読み込む。1個のピリオドは欠損値とみなす	1-32 (1)	x=input("1.1111111E17",best12.);put x;	1.1111111E17
	BITSw.d	文字列をビット列として読み込み対応する数値に変換する	1-64 (1)	x=input("A",bits8.);put x;	65
	COMMAw.d	数字とピリオド、そして先頭のプラス記号とマイナス記号以外のカンマ、ブランク、ダッシュ、ドル記号、%記号、右カッコを除去して読み込む。先頭の左カッコはマイナス記号に変換する	1-32 (1)	x=input("-$1,234.5",comma12.);put x;	-1234.5
	COMMAXw.d	数字とカンマ、そして先頭のプラス記号とマイナス記号以外のピリオド、ブランク、ダッシュ、ドル記号、%記号、右カッコを除去して読み込む。先頭の左カッコはマイナス記号に変換する	1-32 (1)	x=input("-$1.234,5",commax12.);put x;	-1234.5
	DOLLARw.d	数字とピリオド、そして先頭のプラス記号とマイナス記号以外のカンマ、ブランク、ダッシュ、ドル記号、%記号、右カッコを除去して読み込む。先頭の左カッコはマイナス記号に変換する	1-32 (1)	x=input("$1,250.50",dollar12.);put x;	1250.5
	DOLLARXw.d	数字とカンマ、そして先頭のプラス記号とマイナス記号以外のピリオド、ブランク、ダッシュ、ドル記号、%記号、右カッコを除去して読み込む。先頭の左カッコはマイナス記号に変換する	1-32 (1)	x=input("$1.250,50",dollarx12.);put x;	1250.5
	Ew.	科学(浮動)小数点法で書かれた数値を読み込む	1-32 (1)	x=input("1.23457E+02",e12.);put x;	123.457
	FLOATw.d	4 バイトのバイナリ浮動小数点形式で書かれた数値を読み込む	4-4 (4)	x=input("E0E9F642"X,float4.);put x;	123.45678711
	HEXw.	16 進数を表す文字列を数値として読み込む	1-16 (8)	x=input("FFFF",hex4.);put x;	65535
	IBw.d	Window 環境下の整数バイナリ形式で書かれた値を読み込む	1-8 (4)	x=input("39300000"X,ibr8.);put x;	12345
	IBRw.d	OS環境下依存の整数バイナリ形式で書かれた値を読み込	1-8 (4)	x=input("39300000"X,ibr8.);put x;	12345

		む			
	PERCENTw.	数値をパーセントとして読み込む。ピリオド以外のカンマ、ブランク、ダッシュ、パーセント記号と右カッコを無視し、先頭の左カッコはマイナス記号に変換する。パーセント記号は除去され値は 100 で割る	1-32 (6)	x=input("(5.12%",percent6.);put x;	-0.0512
	PIBw.d	OS環境依存の正の整数バイナリ形式の数値を読み込む	1-8 (1)	x=input("02"X,pib2.);put x;	2
	PIBRw.d	Window OS 環境下の正の整数バイナリ形式の数値を読み込む	1-8 (1)	x=input("02"X,pib2.);put x;	2
	PKw.d	符号なしパック10進数形式の数値を読み込む	1-16 (1)	x=input("02"X,pk2.);put x;	2
	RBw.d	OS環境に依存する実数バイナリ形式の数値を読み込む	2-8 (4)	x=input("0040"X,rb4.);put x;	2
	w.d	全体で w 文字、少数点以下 d 桁の形式で書かれた数値を読み込む。ピリオド1個(.)は欠損値として読み込む	1-32 (1)	x=input("10.091",6.3);put x;	10.091
	PDw.d	パック 10 進数形式で書かれた値を読み込む	1-16 (1)	x=input("00000999"X,pd8.);put x;	999
日付・時間	DATEw.	ddmmmyy または ddmmmyyyy の形式で書かれた値を日付値として読み込む	7-32 (7)	x=input("01JAN1960",date9.);put x;	0
	DATETIMEw	ddmmmyy:hh:mm:ss.ss または ddmmmyyyy:hh:mm:ss.ss の形式で書かれた値を日時値として読み込む	13-40 (18)	x=input("01JAN1960:00:01:00",datetime18.);put x;	60
	DDMMYYw.	ddmmyy , ddmmyyyy の形式、または dd/mm/yy , dd/mm/yyyy などの区切り文字付きの形式で書かれた値を日付値として読み込む	6-32 (6)	x=input("10/01/1960",ddmmyy10.);put x;	9
	JULIANw.	ジュリアン日付値 yyddd または yyyyddd の形式で書かれた値を日付値として読み込む	5-32 (5)	x=input("08366",julian5.);put x;	17897
	MMDDYYw.	mmddyy , mmddyyyy の形式、または mm/dd/yy , mm/dd/yyyy などの区切り文字付き形式で書かれた文字列を日付値として読み込む	2-10 (8)	x=input("12/31/2008",mmddyy10.);put x;	17897
	MONYYw.	mmmyy , mmmyyyy の形式、または mmm/yy , mmn/yyyy などの区切り文字付き形式で書かれた文字列を日付値として読み込む	5-32 (5)	x=input("DEC08",monyy5.);put x;	17867
	TIMEw.	hh:mm:ss.ss の形式で書かれた数値を時間値として読み込む	5-32 (8)	x=input("12:10:30",time8.);put x;	43830
	YYMMDDw.	yymmdd , yyyymmdd または yycmmcdd, yyyycmmcdd(c は /などの区切り文字) の形式で書かれた値を日付値として読み込む	6-32 (6)	x=input("2008-12-31",yymmdd10.);put x;	17897
	YYMMNw.	yymm または yyyymm の形式で書かれた値を日付値として読み込む	4-6 (6)	x=input("200812",yymmn6.);put x;	17867

文字	$CHARw.	文字列を先頭のブランクは詰めずにそのまま読み込む。また、空白付き空白付きでないにかかわらず、ピリオド1個は欠損値とみなさない	1-32767 (8)	c=input(" ABC ",$char10.);put c $char.;	ABC
	$CHARZBw.	文字列をバイナリゼロはブランクに変換して読み込む。その他は$CHARw. と同じ	1-32767 (8)	c=input("2041420043"X,$charzb10.);put c $char.;	AB C
	$UPCASEw.	文字列を大文字化して読み込む	1-32767 (8)	c=input("abc",$upcase3.);put c;	ABC
	$w.	先頭のブランクを無視して文字列を読み込む。 また、空白付き空白付きでないにかかわらず、ピリオド1個は欠損値とみなす	1-32767 (1)	c=input(" ABC",$6.);put c $char6.;	ABC
	$ASCIIw.	ASCII コード文字で書かれた文字列を読み込む。ただし PC 環境では$CHARw.と同じ	1-32767 (1)	c=input("414243"X,$ascii3.);put c;	ABC
	$BINARYw.	バイナリコード(0 or 1)で書かれた文字列を読み込む	1-32767 (1)	c=input("010000010100001001000011",$binary24.);put c ;	ABC
	$EBCDICw.	EBCDIC コード文字列を読み込む	1-32767 (1)	c=input("C1C2C3"X,$ebcdic6.);put c $char.;	ABC
	$HEXw.	16 進コードで書かれた文字列を読み込む	1-32767 (2)	c=input("414243",$hex6.);put c $char.;	ABC
	$PHEXw.	パック 16 進コードで書かれた文字列を読み込む	1-32767 (2)	c=input("414243",$phex6.);put c $char.;	343134
z/OS 数値	S370FFw.d	z/OS 環境下の EBCDIC 文字形式で書かれた数値を読み込む	1-32 (12)	x=input("F1F2F3F4"X,s370ff8.);put x;	1234
	S370FIBw.d	z/OS 環境下の整数バイナリ形式で書かれた数値を読み込む	1-8 (4)	x=input("000004D2"X,s370fib8.);put x;	1234
	S370FIBUw.d	z/OS 環境下の符号なしバイナリ形式で書かれた数値を読み込む	1-8 (4)	x=input("000004D2"X,s370fib8.);put x;	1234
	S370FPDw.d	z/OS 環境下のパック 10 進数形式で書かれた数値を読み込む	1-16 (1)	x=input("0001234C"X,s370fpd8.);put x;	1234
	S370FPDUw.d	z/OS 環境下の符号なしパック 10 進数形式で書かれた数値を読み込む	1-16 (1)	x=input("0001234F"X,s370fpdu8.);put x;	1234
	S370FPIBw.d	z/OS 環境下の正の整数バイナリ形式で書かれた数値を読み込む	1-8 (4)	x=input("000004D2"X,s370fpib8.);put x;	1234
	S370FRBw.d	z/OS 環境下の実数バイナリ形式で書かれた数値を読み込む	2-8 (6)	x=input("4120"X,s370frb4.);put x;	2
	S370FZDw.d	z/OS 環境下のゾーン 10 進数形式で書かれた数値を読み込む	1-32 (8)	x=input("F1F2F3C4"X,s370fzd8.);put x;	1234
	S370FZDBw.d	z/OS 環境下の 0 はブランクで表現された符号つきゾーン 10 進数形式で書かれた数値を読み込む。	1-32 (8)	x=input("C0F0F1F2F3F4"X,s370fzdl12.);put x;	1234
	S370FZDLw.d	z/OS 環境下の符号つきゾーン 10 進数形式で書かれた数値を読み込む	1-32 (8)	x=input("C0F0F1F2F3F4"X,s370fzdl12.);put x;	1234
	S370FZDSw.d	z/OS 環境下の符号を分離したゾーン 10 進数形式で書かれた数値を読み込む	2-32 (8)	x=input("4EF0F1F2F3F4"X,s370fzds12.);put x;	1234

	S370FZDTw.d	z/OS 環境下の符号を分離して後ろにつけたゾーン 10 進数形式で書かれた数値を読み込む	2-32 (8)	x=input("F0F1F2F3F44E"X,s370fzdt12.);put x;	1234
	S370FZDUw.d	z/OS 環境下の符号なしゾーン 10 進数形式で書かれた数値を読み込む	1-32 (8)	x=input("F0F0F1F2F3F4"X,s370fzdu12.);put x;	1234
	ZDw.d	OS 環境依存のゾーン 10 進数形式(最後のバイトに符号つきデジタル)で書かれた数値を読み込む	1-32 (1)		
	ZDBw.d	OS 環境依存の0がブランクで表現されているゾーン 10 進数形式で書かれた数値を読み込む	1-32 (1)		
z/OS 日付・時間	MSECw.	z/OS 環境下の 8 バイトマイクロ秒値を時間値として読み込む	1-8 (8)		
	PDJULGw.	z/OS 環境下のパックジュリアン日付値 yyyydddF の形式で書かれた値を日付値として読み込む	4-4 (4)	x=input(put("31DEC2008"D,pdjulg4.),pdjulg4.);put x;	17897
	PDJULIw.	z/OS 環境下の 16 進パックジュリアン日付値 ccyydddF の形式で書かれた値を日付値として読み込む	4-4 (4)	x=input(put("31DEC2008"D,pdjuli4.),pdjuli4.);put x;	17897
	PDTIMEw.	z_OS 環境下の RMF や SMF レコードに見られる 0hhmmssF 形式の 16 進パック時間値を読み込む	4-4 (4)		
	RMFDURw.	z_OS 環境下の mmsstttF 16 進形式の RMF レコードの時間値を読み込む	4-4 (4)		
	RMFSTAMPw.	z_OS 環境下の 0hhmmssFccyydddF 16 進形式の RMF レコードの時間値を読み込む	8-8 (8)		
	SMFSTAMPw.d	z_OS 環境下の hhhhhhhhccyydddF 16 進形式の SMF レコードの日時値を読み込む	8-8 (8)		
	TODSTAMPw.	z_OS 環境下の 8 バイトバイナリ整数形式の時間値(Time Of Day)を数値として読み込む	1-8 (8)		
	TUw.	z_OS 環境下の 4 バイトバイナリ整数形式の時間単位(Timer Unit)を数値として読み込む	4-4 (4)		