Natural for Mainframes

ファーストステップ

バージョン 4.2.5

October 2009

Natural

This document applies to Natural バージョン 4.2.5 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

目次

1 ファーストステップ

このチュートリアルでは、Natural を使用したプログラミングおよび Natural エディタの使用について簡単に概要を説明します。

重要: 次のトピックを以下に示す順に読み、トピックに含まれているすべての演習をこのチュートリアルで示す順に行うことが重要です。 演習をスキップすると、問題が生じる可能性があります。

	このチュートリアルについて	前提条件およびこのチュートリアルで習得する内容。
	Natural の基本	Natural のメインメニューを起動する方法。 このチュートリアルで使用するライブラリを作成する方法。 Natural のプログラミングモードおよびこのチュートリアルで必要なモードに関する情報。
	Hello World!	最初の簡単なプログラムを作成、実行、および格納する方法。 現在のライブラリの内容を表示する方法。 エディタプロファイルを制御する一部のオプションに関する情報。
	データベースへのアクセス	特定のデータをデータベースから読み込み出力を表示する方法。
	ユーザー入力	情報の入力を求めるプロンプトをユーザーに対して表示する方法およびユーザー入力のマップを設計する方法。 ユーザーが指定しない場合でも特定の値（ここでは終了名）を常に確実に使用する方法。
	ループおよびラベル	繰り返しループおよび異なるループのラベルを定義する方法。 特定の情報（ここではユーザーが入力する開始名）が見つからない場合にメッセージを表示する方法。
	インラインサブルーチン	インラインサブルーチン（プログラムで直接コーディングするサブルーチン）を定義および起動する方法。
	処理ルールとヘルプルーチン	処理ルール（ここではユーザーが開始名を指定しない場合に表示するメッセージ）およびヘルプルーチン（ここではユーザーが開始名を入力する必要があるフィールドのヘルプテキスト）を定義する方法。
	ローカルデータエリア	フィールド定義をプログラムからプログラム外のローカルデータエリアに再配置する方法。

	グローバルデータエリア	複数のプログラムまたはルーチンで共有できるグローバルデータエリアを定義する方法。
	外部サブルーチン	外部サブルーチン（プログラム外に別のオブジェクトとして格納されるサブルーチン）を定義および起動する方法。
	サブプログラム	サブプログラムのパラメータデータエリアを定義する方法。 サブプログラムを定義および起動する方法。

2 このチュートリアルについて

初めてのユーザーの場合は、このチュートリアル全体を実行して、Naturalプログラミング環境の特定の機能についての基本的な理解を習得することをお勧めします。

チュートリアルで提供されている例の画面のレイアウトと、ここで説明しているNaturalの動作は、ユーザーの結果と異なる場合があります。 例えば、コマンドまたはメッセージ行が異なる画面の位置に表示されたり、Naturalコマンドの実行がセキュリティコントロールによって保護されている場合があります。 環境のデフォルト設定は、Natural管理者が設定したシステムパラメータによって異なります。

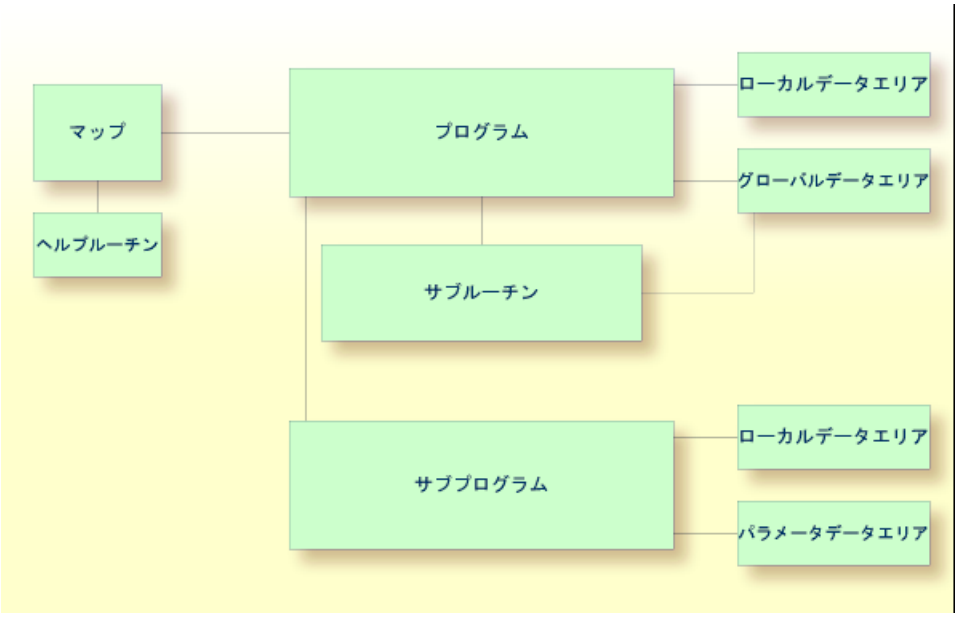
前提条件

このチュートリアルのすべてのステップを実行するには、Natural のオブジェクト例だけでなく、Adabas デモファイル `EMPLOYEES` および `VEHICLES` をインストールする必要があります。 これらのファイルがインストールされていない場合は、インストールするよう管理者に依頼してください。

サンプルのアプリケーションについて

このチュートリアルでは、モジュールのグループとしてアプリケーションをどのように構成できるかを示します。 アプリケーションの構築方法の例を示すわけではありません。

最初の簡単なHelloWorldプログラムを作成した後、データベースから従業員情報を読み込み出力を表示するプログラムを作成します。 出力用の開始名および終了名の入力を指示するプロンプトをユーザーに対して表示します。 プログラムの特定の部分を外部モジュールに移動することにより、プログラムをステップごとに強化していきます。 このチュートリアルの演習をすべて完了すると、アプリケーションは次のような構成になります。



最初の演習「*Natural の基本*」に進みます。

3 Natural の基本

Natural のメインメニューの呼び出し

Natural のメインメニューから、Natural の開発機能、環境設定、ユーティリティおよびライブラリ例にアクセスできます。

手順 3.1. Natural のメインメニューを起動するには

1 サイトにおける手順に従って Natural を起動します。

環境のデフォルト設定に応じて、Natural のメインメニューあるいは NEXT または MORE コマンドプロンプトが表示されます。

2 上記のいずれかのコマンドプロンプトが表示されたら、次のように入力します。

```
MAINMENU
```

メインメニューが表示されます。

```
09:51:48                    *****  NATURAL  *****                    2007-03-20
User SAG                        - Main Menu -                Library SYSTEM


                    Function

                 _  Development Functions
                 _  Development Environment Settings
                 _  Maintenance and Transfer Utilities
                 _  Debugging and Monitoring Utilities
                 _  Example Libraries
                 _  Other Products
                 _  Help
                 _  Exit Natural Session









Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                  Canc
```

ライブラリ

アプリケーションの作成に必要な Natural オブジェクトはすべて、Natural システムファイルの Natural ライブラリに格納されます。 システムファイルには、システムプログラム用のもの（FNAT）とユーザー作成プログラム用のもの（FUSER）があります。

したがって、Naturalではシステムライブラリとユーザーライブラリが区別されます。 システムライブラリは "SYS" という文字で始まり、Software AG 専用に確保されています。 ユーザーライブラリには、アプリケーションを構成するユーザー定義オブジェクト（プログラムおよびマップなど）がすべて含まれます。 ユーザーライブラリの名前を "SYS" という文字で始めることはできません。

Naturalのメインメニュー（および他の多くの画面）の右上隅にある［**Library**］フィールドに、現在ログオンしているライブラリの名前が表示されます。

コマンドの発行

Natural コマンドの入力では、大文字と小文字が区別されません。 Natural コマンドを入力したら、Enter キーを押します。 Enter キーを押すことにより、操作が確認されてコマンドが実行されるか、コマンドの実行を明示的に確認するための別の確認ウィンドウが表示されます。

ユーザーライブラリの作成

TUTORIAL という名前のユーザーライブラリを作成します。 このライブラリに、このチュートリアルで作成するすべての Natural オブジェクトを格納します。

手順 3.2. ユーザーライブラリを作成するには

- コマンド行（Naturalのメインメニューで［**Command ===>**］で示される）で次のように入力します。

  ```
  LOGON TUTORIAL
  ```

  "TUTORIAL" は作成するライブラリの名前です。

  LOGON は次のいずれかの目的のために使用するシステムコマンドです。

  - 既存のライブラリにログオンします。
  - 指定した名前のライブラリが存在しない場合に新しいライブラリを作成します。

  または:

画面の右上隅で、現在のライブラリ名をログオンするライブラリ名で上書きし、Enter キーを押します。

例：

```
11:33:26                    *****  NATURAL  *****                  2007-03-20
User SAG                        - Main Menu -               Library TUTORIAL
```

［Development Functions］メニュー

［**Development Functions**］メニューを使用して、Natural オブジェクトを作成および修正することができます。

▶手順 3.3. ［Development Functions］メニューを表示するには

- Natural のメインメニューで、［**Development Functions**］の横にある入力フィールドに任意の文字を入力し、Enter キーを押します。

  または:

  カーソル選択を使用します。つまり、［**Development Functions**］の横にある入力フィールドにカーソルを置き、Enter キーを押します。

  ［**Development Functions**］メニューが表示されます。

```
12:15:05                     *****  NATURAL  *****                  2007-03-20
User SAG                   - Development Functions -          Library TUTORIAL

                                                                 Mode Reporting
                                                          Work area empty
                         Code  Function

                          C    Create Object
                          E    Edit Object
                          R    Rename Object
                          D    Delete Object
                          X    Execute Program
                          L    List Object(s)
                          S    List Subroutines Used
                          ?    Help
                          .    Exit

                    Code .. _    Type .. _
```

```
                                Name .. _____________________________________


Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Menu  Exit                                                    Canc
```

プログラミングモード

プログラミングモードは、［**Development Functions**］メニューの右上隅にある［**Mode**］フィールドに示されます。

Natural には、2 つの異なるプログラミングモードがあります。

- ストラクチャードモード
  ストラクチャードモードは、明確で適切に定義されたプログラム構成で複雑なアプリケーションを実装するときに使用します。 ストラクチャードモードは単独で使用することをお勧めします。
- レポーティングモード
  レポーティングモードが役に立つのは、複雑なデータやプログラミング構成を必要としない、アドホックレポートおよび小さなプログラムを作成する場合のみです。

⚠ **重要:** このチュートリアルでは、ストラクチャードモードがアクティブになっている必要があります。 プログラムをレポーティングモードで実行しようとすると、`END-IF`、`END-READ`、および `END-REPEAT` でエラーになります。

レポーティングモードが現在アクティブになっている場合は、以下の手順に従ってください。

手順 3.4. レポーティングモードからストラクチャードモードに切り替えるには

- コマンド行で次のシステムコマンドを入力し、Enter キーを押します。

  ```
  GLOBALS SM=ON
  ```

  または:

  ［**Development Functions**］メニューの右上隅で、［**Mode**］フィールドの最初の位置（"Reporting" と表示されている）を次の文字で上書きし、Enter キーを押します。

  ```
  S
  ```

例：

```
12:17:20                      *****  NATURAL  *****                    2007-03-20
User SAG                   - Development Functions -          Library TUTORIAL

                                                                 Mode Seporting
                                                             Work area empty
```

最初のプログラム「*Hello World!*」に進みます。

4 Hello World!

プログラムの作成

"Hello World!" と表示する最初の簡単なプログラムを作成します。 このプログラムは、前に作成したライブラリに格納されます。

手順 4.1. 新しいプログラムを作成するには

1 TUTORIAL という名前のライブラリにログオンしていることを確認します。

2 ［**Development Functions**］メニューの下部で、次の情報を入力して Enter キーを押します。

```
                  Code .. C    Type .. P
                               Name .. HELLO_____________________________



Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Menu  Exit                                                    Canc
```

"C" は［**Create Object**］機能、"P" はオブジェクトタイププログラム、"HELLO" は作成するプログラムの名前を表します。

ヒント: ファンクションコード「C」と入力すると、［**Type**］フィールドにアスタリスク（*）も入力できます。 Enter キーを押すと、すべてのオブジェクトタイプのリストおよびオブジェクトタイプに対応する文字が表示されます。

プログラムエディタが表示されます。 現時点では空になっています。

3 プログラムエディタで次のコードを入力します。

```
* The "Hello world!" example in Natural.
*
DISPLAY "Hello world!"
END /* End of program
```

コメント行はアスタリスク（*）で始まり、少なくとも 1 つの空白または 2 つ目のアスタリスクが後に続きます。 空白または 2 つ目のアスタリスクを入力し忘れると、システム変数を指定したとみなされ、エラーになります。

プログラムに空行を挿入する場合は、コメント行として定義する必要があります。これは、異なるプラットフォーム（Windows、メインフレーム、UNIX、または OpenVMS）からプログラムにアクセスする場合に便利です。 例えば、メインフレームバージョンの Natural では、デフォルトにより、Enter キーを押すと空行が自動的に削除されます。

また、ステートメント行の最後にコメントを挿入することもできます。 この場合、コメントはスラッシュで始まり、アスタリスクが後に続きます（/*）。

出力に表示するテキストは、DISPLAY ステートメントで定義されます。 表示テキストは引用符で囲みます。

END ステートメントは、Natural プログラムの物理的な終わりをマークするために使用します。 各プログラムは END で終了する必要があります。

Enter キーを押すと、小文字がすべて大文字に変換されることがあります。 この動作は、エディタプロファイル（後述）で定義されます。

プログラムの実行

システムコマンド RUN では、プログラムコードのエラーをチェックするシステムコマンド CHECK が自動的に呼び出されます。 エラーが検出されなければ、その時点でプログラムがコンパイルされ、実行されます。

注意:

1. CHECK を別のコマンドとして使用することもできます。
2. Naturalには、格納バージョンのプログラムを使用するシステムコマンド EXECUTE も用意されています（プログラムの格納についてはこのチュートリアルで後述）。 これに対し、RUN コマンドでは、最新の修正を含むプログラムが常に使用されます。

手順 4.2. プログラムを実行するには

1 プログラムエディタのコマンド行で、次のいずれかを入力します。

```
RUN
```

```
R
```

システムコマンドは略記される場合があります。 R は RUN の省略形です。

コマンド行は、環境定義に応じて画面の上部または下部のどちらかに表示されます。

```
> RUN                                    > +  Program     HELLO    Lib TUTORIAL
```

コードが構文的に正しければ、定義したテキストが出力に表示されます。

```
MORE
Page      1                                                  05-03-11  12:07:25


Hello world!
```

2 Enter キーを押して、プログラムエディタに戻ります。

プログラムエラーの修正

Hello World プログラムでエラーを作成してから、プログラムをもう一度実行します。

手順 4.3. エラーを修正するには

1 DISPLAY ステートメントを含む行で 2 つ目の引用符を削除します。
2 上記の手順で、プログラムをもう一度実行します。

エラーが検出されると、エラーメッセージが表示されます。

```
NAT0305 Text string must begin and end on the same line.
>                                       > +  Program     HELLO    Lib TUTORIAL
All    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0010 * The "Hello world!" example in Natural.
  0020 *
E 0030 DISPLAY "HELLO WORLD!
  0040 END /* End of program

  0050
  0060
  0070
  0080
  0090
  0100
  0110
  0120
  0130
  0140
  0150
  0160
  0170
  0180
  0190
  0200
       ....+....1....+....2....+....3....+....4....+....5....+... S 4    L 1
```

エラーを含むステートメント行が強調表示され、"E" でマークされます。

3 エラーを修正します。つまり、欠けている引用符を行の最後に挿入します。

4 プログラムをもう一度実行して、次のエラーを見つけます。

この場合、他のエラーは検出されず出力が表示されます。

5 Enter キーを押して、プログラムエディタに戻ります。

プログラムの格納

プログラムを格納すると、プログラムがコンパイルされ、ソースコードと生成プログラムの両方が Natural システムファイルに格納されます。

RUN コマンドと同様に、システムコマンド STOW でも CHECK コマンドが自動的に呼び出されます。プログラムは構文的に正しい場合にのみ格納されます。

注意: プログラムに構文エラーが含まれていてもプログラムに加えた変更を保存する（作業を翌日まで停止する場合など）には、システムコマンド SAVE を使用できます。

手順 4.4. プログラムを Stow するには

■ プログラムエディタのコマンド行で、次のように入力します。

```
STOW
```

プログラムに関する情報の表示

LIST コマンドは、オブジェクトに対してソースコードのみが使用できるのか、ソースコードと生成プログラムの両方が使用できるのかを調べるのに便利です。

手順 4.5. プログラムに関する情報を表示するには

1 プログラムエディタのコマンド行で、次のいずれかを入力します。

```
LIST DIR HELLO
```

```
L DIR HELLO
```

次の画面が表示されます。［**Cataloged on**］の情報は、オブジェクトが格納された場合にのみ表示されます。

```
13:15:45               ***** NATURAL LIST COMMAND *****              2007-03-20
User SAG                      - List Directory -                Library TUTORIAL

Directory of Program HELLO                      Saved on ... 2007-03-20 13:15:36
------------------------------------------------------------------------------
Library .... TUTORIAL   User-ID ...... SAG       Mode ....... Structured
TP-System .. COMPLETE   Terminal-ID .. DAEFTCA9
Op-System .. MVS/ESA    Transaction .. NATvr
NAT-Ver .... v.r.s
Source size ........................     100 Bytes

Directory of Program HELLO                      Cataloged on 2007-03-20 13:15:36
------------------------------------------------------------------------------
Library .... TUTORIAL   User-ID ...... SAG       Mode ....... Structured
TP-System .. COMPLETE   Terminal-ID .. DAEFTCA9
Op-System .. MVS/ESA    Transaction .. NATvr
NAT-Ver .... v.r.s
Used GDA ...
Size of global data ...       0 Bytes  Size in DATSIZE ......     560 Bytes
Size in buffer pool ...    2620 Bytes

Size of OPT-Code ......       0 Bytes
Initial OPT string ....
                                                          ENTER to continue
```

注意: 上の例の *vr* および *v.r.s* という表記は、Natural の現在のバージョン番号を表します。「*用語集*」のバージョンの定義も参照してください。

2 Enter キーを押して、プログラムエディタに戻ります。

現在のライブラリの内容の表示

LIST コマンドは、現在のライブラリに存在するすべての Natural オブジェクトのリストを表示する場合にも使用できます。例えば、このチュートリアルの途中で最初からやり直すため、1つまたは複数の Natural オブジェクトを削除する場合などに便利です。

手順 4.6. Natural オブジェクトのリストを表示するには

1 プログラムエディタのコマンド行で、次のいずれかを入力します。

```
LIST *
```

```
L *
```

次の画面が表示されます。 作成したプログラムがリストされます。

```
13:34:27                ***** NATURAL LIST COMMAND *****              2007-03-20
User SAG                  - LIST Objects in a Library -          Library TUTORIAL

Cmd  Name       Type         S/C  SM Version  User ID    Date        Time
---  *_________ *__________  *__  *  *_______ *________  *__________ *________
__   HELLO      Program      S/C  S  v.r.s    SAG        2007-03-20  13:15:36
                                                                    1 Objects found
Top of List.
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Print Exit  Sort        --    -     +     ++          >     Canc
```

2 どのコマンドが使用可能であるかを調べるには、プログラムの横にある［**Cmd**］列で疑問符（?）を入力します。

次のウィンドウが表示されます。

```
+-------- COMMANDS ---------+
!                            !
!   ED   Edit                !
!   LI   List                !
!   LD   List Dir            !
!   PR   Print               !
!   LE   List expanded       !
!   RU   Run                 !
!   ST   Stow                !
!   CA   Catalog             !
!   DE   Delete              !
!   RE   Rename              !
!   .    End                 !
!                            !
!                            !
!                            !
!                            !
!                            !
!   __ HELLO                 !
+----------------------------+
```

3 ここでは、変更を適用しません。コマンドを指定せずにウィンドウを閉じるには、PF3 キーを押します。

4 プログラムエディタに戻るには、PF3 キーをもう一度押します。

エディタプロファイルのオプションの設定

Natural プログラムエディタまたはデータエリアエディタで作業するときは、ユーザーごとにエディタプロファイルを定義できます。このチュートリアルでは、SYSTEM という名前のエディタプロファイルのデフォルト設定を使用します。一部の重要な設定について、次に説明します。

▶手順 4.7. エディタプロファイルのオプションをチェックするには

1 プログラムエディタのコマンド行で、次のように入力します。

```
PROFILE
```

次の画面が表示されます。

```
13:35:43                ***** NATURAL EDITORS *****                2007-03-20
                            - Editor Profile -

Profile Name .. SYSTEM__

PF and PA Keys
  PF1 ... HELP_____________   PF2 ... _________________   PF3 ... EXIT_____________
  PF4 ... _________________   PF5 ... _________________   PF6 ... _________________
  PF7 ... -________________   PF8 ... +________________   PF9 ... _________________
  PF10 .. SC=______________   PF11 .. _________________   PF12 .. CANCEL___________
  PF13 .. _________________   PF14 .. _________________   PF15 .. MENU_____________
  PF16 .. _________________   PF17 .. _________________   PF18 .. _________________
  PF19 .. --_______________   PF20 .. ++_______________   PF21 .. _________________
  PF22 .. _________________   PF23 .. _________________   PF24 .. _________________
  PA1 ... _________________   PA2 ... SCAN_____________   PA3 ... _________________

Automatic Functions
  Auto Renumber .. Y   Auto Save Numbers .. 0__   Source Save into .. EDITWORK

Additional Options .. N

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  AddOp Save  Flip                          Del   Canc
```

ユーザー固有のエディタプロファイルがない場合、デフォルトのプロファイル SYSTEM が表示されます。 このデフォルトのプロファイルを使用して、ユーザー固有のプロファイルを作成できます。ユーザー固有のプロファイルがすでにある場合、そのプロファイルが SYSTEM プロファイルの代わりに表示されます。

2 ［**Additional Options**］フィールドで「Y」と入力し、Enter キーを押します。

または:

PF4 キーを押します。

次のウィンドウが表示されます。

```
+--------------------- ADDITIONAL OPTIONS ----------------------+
!                                                               !
!                                                               !
!   +  Editor Defaults ......... N                              !
!   +  General Defaults ........ N                              !
!   +  Colour Definitions ...... N                              !
!                                                               !
!                                                               !
!                                                               !
!                                                               !
```

```
!                                                                  !
+------------------------------------------------------------------+
```

3 ［**Editor Defaults**］および［**General Defaults**］フィールドで「Y」と入力し、Enter キーを押します。

エディタのデフォルトを示す次のウィンドウが表示されます。

```
+---------------------- EDITOR DEFAULTS -----------------------+
!                                                              !
!    Escape Character for Line Command .. .                    !
!    Empty Line Suppression ............. Y                    !
!    Source Size Information ............ N                    !
!    Source Status Message .............. N                    !
!    Absolute Mode for SCAN/CHANGE ...... N                    !
!    Range Mode for SCAN/CHANGE ......... N                    !
!    Direction Indicator ................ +                    !
!                                                              !
!                                                              !
+--------------------------------------------------------------+
```

行コマンドに定義されたエスケープ文字が表示されます。このチュートリアルでは、デフォルト文字であるピリオド（.）が使用されています。

また、［**Empty Line Suppression**］オプションは "Y" に設定されています。 この場合、プログラムエディタ内の空行はすべて、Enter キーを押すと自動的に削除されます。 このオプションが "N" に設定されていると、空行は削除されません。

4 このチュートリアルでは、すべてのオプションを上記のように設定する必要があります。Enter キーを押すと、次のウィンドウが表示されます。

次のウィンドウが表示されます。

```
+---------------------- GENERAL DEFAULTS ----------------------+
!                                                              !
!    Editing in Lower Case .............. N                    !
!    Dynamic Conversion of Lower Case ... Y                    !
!    Position of Message Line ........... TOP                  !
!    Cursor Position in Command Line .... N                    !
!    Stay on Current Screen ............. N                    !
!    Prompt Window for Exit Function .... Y                    !
!    ISPF Editor as Program Editor ...... N                    !
!    Leave Editor with Unlock ........... N                    !
!                                                              !
+--------------------------------------------------------------+
```

［**Editing in Lower Case**］オプションが "Y" に設定され、［**Dynamic Conversion of Lower Case**］オプションが "N" に設定されていると、ソースコードは入力どおりになります。 た

だし、この機能も、すべての入力の大文字変換を強制的に行うシステム環境固有の設定によって異なり、これは Natural によって左右されません。

5 必要に応じて、上記のオプションを小文字変換に変更し、Enter キーを押します。 Enter キーをもう一度押して［**Additional Options**］ウィンドウに戻り、Enter キーを再度押してこのウィンドウを閉じます。

6 ユーザー固有のプロファイルが作成されていない場合は、プロファイル名 SYSTEM をユーザー ID で上書きし、Enter キーを押します。

ユーザー固有のプロファイルがすでにある場合、次の手順に進みます。

7 PF5 キーを押して変更内容をデータベースに保存し、PF3 キーを押してエディタプロファイルを終了します。

注意: PF キーを押す代わりに、対応するコマンドをコマンド行に入力することもできます。 例えば、上記の場合は、SAVE および EXIT コマンドを入力できます。

または:

変更内容を適用しない場合は、PF3 キーを押してエディタプロファイルを終了します。

出口関数では、異なるオプションを持つウィンドウが表示されます。 変更内容を保存した場合にも表示されます。

```
+----------- EXIT Function -----------+
!                                      !
!   _ Save and Exit                    !
!   _ Exit without Saving              !
!   _ Resume Function                  !
!                                      !
!                                      !
+--------------------------------------+
```

8 出口関数を呼び出す直前に変更内容を保存した場合は、オプション［**Exit without Saving**］を選択しても問題ありません。 Enter キーを押して、プログラムエディタに戻ります。

注意: PF5 キーを押した後または SAVE コマンドを発行した後でさらに変更を加えて［**Exit without Saving**］オプションを選択すると、最後に加えた変更は現在のセッションでのみ有効となり、データベースには保存されません。

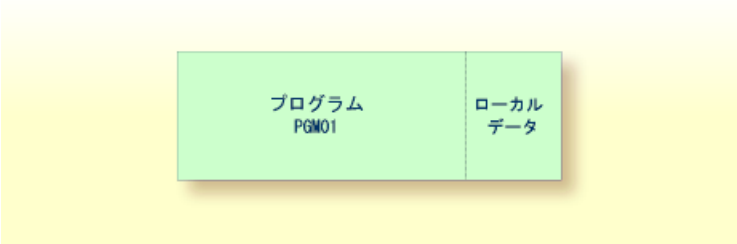
プログラムがもう一度表示されます。新しい設定がプログラムエディタ（および後述のデータエリアエディタ）で使用されるようになります。

次の演習「データベースへのアクセス」に進みます。

5 データベースへのアクセス

特定のデータをデータベースファイルから読み込み、対応する出力を表示する簡単なプログラムを作成します。

以下の演習を完了すると、サンプルのアプリケーションは1つのモジュールでのみ構成されます（プログラムで使用されるデータフィールドはプログラム内で定義されます）。



新しい名前でのプログラムの保存

このチュートリアルの残りの部分で使用する新しいプログラムを作成します。 新しいプログラムは、Hello World プログラムを新しい名前で保存することによって作成します。

手順 5.1. プログラムを新しい名前で保存するには

1 プログラムエディタのコマンド行で、次のいずれかを入力します。

```
SAVE PGM01
```

```
SA PGM01
```

現在のプログラムが新しい名前 PGM01 で保存されます。 プログラム名 HELLO は、引き続きプログラムエディタに表示されます。

2 プログラムエディタのコマンド行で次のように入力して、新しく作成したプログラムをプログラムエディタに読み込みます。

```
READ PGM01
```

プログラムエディタに表示されるプログラム名が PGM01 に変わります。

3 プログラムエディタですべてのコードを削除します。 これを行うには、削除する各行の先頭で次の行コマンドを入力し、Enter キーを押します。

```
.D
```

例：

```
>                                      > +  Program     PGM01    Lib TUTORIAL

All    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0010 .DThe "Hello world!" example in Natural.
  0020 .D
  0030 .DSPLAY "Hello world!"
  0040 .DD /* End of program
  0050
```

または:

最初の行の先頭で次の行コマンドを入力し、Enter キーを押します。

```
.D(4)
```

カッコ内にある番号は、削除する行数を表します。

ビューを使用した必須データの定義

プログラムで使用するデータベースファイルおよびフィールドは、プログラムの先頭にある DEFINE DATA と END-DEFINE との間で指定する必要があります。

Naturalがデータベースファイルにアクセスできるようにするには、物理データベースファイルの論理定義が必要です。このような論理ファイル定義は、データ定義モジュール（DDM）と呼ばれます。 DDM には、ファイルの個々のフィールドに関する情報が含まれます。 DDM は通常、Natural 管理者が定義します。

Natural プログラムでデータベースフィールドを使用できるようにするには、ビューで DDM からフィールドを指定する必要があります。 このチュートリアルでは、EMPLOYEES データベースファイル用の DDM を使用します。

手順 5.2. DEFINE DATA ブロックを指定するには

■ プログラムエディタで次のコードを入力します。

```
DEFINE DATA
LOCAL

END-DEFINE
*
END
```

LOCAL は、次の手順で定義する変数がこのプログラムにのみ適用されるローカル変数であることを示します。

手順 5.3. DDM のデータフィールドを画面分割で表示するには

1 プログラムエディタのコマンド行で、次のように入力します。

```
SPLIT VIEW EMPLOYEES SHORT
```

SHORT は、データフィールドが短縮形でリストされる（Adabas ショートネームおよび対応する Natural フィールド名のみが表示される）ことを示します。

画面が2つのセクションに分割されます。DDMのデータフィールドは、画面の下半分に表示されます。 画面の下半分でデータを編集することはできません。

```
>                                        > +  Program     PGM01    Lib TUTORIAL
All    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0010 DEFINE DATA
  0020 LOCAL
  0040 END-DEFINE
  0050 *
  0060 END
  0070
  0080
  0090
  0100
  0110
       ....+....1....+....2....+....3....+....4....+....5....+... S 5    L 1
Split Top  View EMPLOYEES                    DBID     0 FNR     1 Def seq
    1  AA  PERSONNEL-ID                     A    8    D CNNNNNNN
  G 1  AB  FULL-NAME                                    NAME INFORMATION
    2  AC  FIRST-NAME                       A   20  N   FIRST/CHRISTIAN NA
    2  AD  MIDDLE-I                         A    1  N   MIDDLE INITIAL
    2  AE  NAME                             A   20    D SURNAME/FAMILY NAM
    1  AD  MIDDLE-NAME                      A   20  N   SECOND/MIDDLE NAME
    1  AF  MAR-STAT                         A    1  F   M=MARRIED
    1  AG  SEX                              A    1  F
    1  AH  BIRTH                             D    6  N D BIRTH-DATE (YYYY-M
```

2 ビューでページをスクロールし、どのデータフィールドが使用され、それがどのように定義されているかを確認することができます。 これには、次のコマンドを使用します。

コマンド	説明
SPLIT + または S +	ビューの次のページに進みます。
SPLIT - または S -	ビューの前のページに進みます。
SPLIT . または S .	画面分割モードを終了します。

次の手順では、画面分割モードが終了していることが前提となります。

3 LOCAL を含む行の最初の位置にカーソルをおき、次のように入力します。

```
.I
```

フルスクリーンモードでは、9行の空行が挿入されます。 画面分割モードでは、4行の空行しか挿入されません。

4 LOCAL の下に次のコードを入力します。

```
1 EMPLOYEES-VIEW VIEW OF EMPLOYEES
  2 FULL-NAME
    3 NAME (A20)
  2 DEPT (A6)
  2 LEAVE-DATA
    3 LEAVE-DUE (N2)
```

5 Enter キーを押します。

残りの空行が除去されます。

注意: エディタプロファイルのデフォルト設定が変更された場合、つまり［**Empty Line Suppression**］オプションが"N"に設定された場合、残りの空行は除去されません。

最初の行には、ビューの名前およびフィールドが取得されるデータベースファイルの名前が含まれています。 最初の行は、常にレベル1で定義されます。 レベルは、行の先頭に示されます。 DDMのデータベースフィールドの名前は、レベル2および3で定義されます。

レベルはフィールドグルーピングと合わせて使用します。2またはそれ以上のレベル番号が付いたフィールドは、それより小さいレベル番号が付いた直前のグループの一部であるとみなされます。 グループで定義すると、グループ名を使用して一連のフィールド（または1つのフィールドだけでもよい）を参照することができます。 これは一連の連続フィールドを参照するのに便利で効果的な方法です。

各フィールドのフォーマットおよび長さは、カッコで囲んで示されます。 "A" は英数字を示し、"N" は数字を示します。

データベースからのデータの読み込み

必須データの定義が完了したので、READループを追加します。このループは、定義されたビューを使用してデータベースファイルからデータを読み込みます。各ループでは、データベースファイルから1人の従業員を読み込みます。 この従業員の名前、部署、および休暇の残り日数が表示されます。 すべての従業員が表示されるまで、データが読み込まれます。

注意: 最後のトランザクションがデータベースからバックアウトされたことを示すエラーメッセージが表示されることがあります。これは通常、Adabasによって決められる非アクティビティタイムリミットを超えた場合に発生します。 このようなエラーが発生した

場合は、最後に実行されたアクションを単に繰り返す（RUN コマンドを再発行するなど）必要があります。

手順 5.4. データベースからデータを読み込むには

1 END-DEFINE の下に次の行を挿入します（空行を挿入するには、上記の手順で .I コマンドを使用します）。

```
READ EMPLOYEES-VIEW BY NAME
*
  DISPLAY NAME 3X DEPT 3X LEAVE-DUE
*
END-READ
```

BY NAME は、データベースから読み込むデータが名前別にアルファベット順にソートされることを示します。

DISPLAY ステートメントは、出力を列形式で配列します。 指定されたフィールドごとに列が作成され、列の上にヘッダーが表示されます。 3X は、列間に 3 文字分の空白が挿入されることを示します。

2 プログラムを実行します。

次の出力が表示されます。

```
MORE
Page      1                                                  05-05-18  16:06:49

        NAME           DEPARTMENT   LEAVE
                          CODE       DUE
--------------------   ----------   -----

ABELLAN                PROD04        20
ACHIESON               COMP02        25
ADAM                   VENT59        19
ADKINSON               TECH10        38
ADKINSON               TECH10        18
ADKINSON               TECH05        17
ADKINSON               MGMT10        28
ADKINSON               TECH10        26
ADKINSON               SALE30        36
ADKINSON               SALE20        37
ADKINSON               SALE20        30
AECKERLE               SALE47        31
AFANASSIEV             MGMT30        26
AFANASSIEV             TECH10        35
AHL                    MARK09        30
AKROYD                 COMP03        20
ALEMAN                 FINA03        20
```

DISPLAY ステートメントの結果、列ヘッダー（DDM から取得）に下線が引かれ、下線とデータの間に 1 行の空行が挿入されます。 各列の幅は、DEFINE DATA ブロックで定義された値（ビューで定義された値）と同じになります。

各ページ上部のタイトルにはページ番号および日時が含まれ、これも DISPLAY ステートメントによって表示されます。

3 Enter キーを繰り返し押して、すべてのページを表示します。

すべての従業員が表示されると、プログラムエディタに戻ります。

ヒント: すべての従業員が表示される前にプログラムエディタに戻るには、［MORE］プロンプトで「EDIT」または省略形である「E」を入力します。 また、［MORE］プロンプトで、現在の Natural 操作を中断する端末コマンド「%」を入力することもできます。 デフォルトでは、各端末コマンドは制御文字 % で始まります。 ただし、管理者が別の制御文字を定義している可能性もあります。

データベースからの選択したデータの読み込み

前の出力は非常に長いため、それを制限します。 "Adkinson" で始まり "Bennett" で終わる範囲の名前のデータのみを表示します。 これらの名前はデモデータベースに定義されています。

手順 5.5. 出力をデータの範囲に制限するには

1 新しい変数を使用する前に、それを定義する必要があります。 したがって、LOCAL の下に次の行を挿入します。

```
1 #NAME-START        (A20) INIT <"ADKINSON">
1 #NAME-END          (A20) INIT <"BENNETT">
```

これらはユーザー定義変数であり、デモデータベースには定義されていません。 名前の先頭にあるハッシュ（#）は、ユーザー定義変数とデモデータベースに定義されているフィールドを区別するために使用されていますが、必須文字ではありません。

INIT はフィールドのデフォルト値を定義します。 デフォルト値は、山カッコおよび引用符で囲んで指定する必要があります。

2 READ ステートメントの下に次の行を挿入します。

```
STARTING FROM #NAME-START
ENDING AT #NAME-END
```

プログラムは次のようになります。

```
DEFINE DATA
LOCAL
  1 #NAME-START        (A20) INIT <"ADKINSON">
  1 #NAME-END          (A20) INIT <"BENNETT">
  1 EMPLOYEES-VIEW VIEW OF EMPLOYEES
    2 FULL-NAME
      3 NAME (A20)
    2 DEPT (A6)
    2 LEAVE-DATA
      3 LEAVE-DUE (N2)
END-DEFINE
*
READ EMPLOYEES-VIEW BY NAME
  STARTING FROM #NAME-START
  ENDING AT #NAME-END
*
  DISPLAY NAME 3X DEPT 3X LEAVE-DUE
*
END-READ
*
END
```

プログラムコードは1画面ページを超えています。次のコマンドまたはキーを使用すると、プログラムソース内で移動できます。

コマンド	説明
BOT	プログラムの最後にジャンプします。
TOP	プログラムの最初に戻ります。
キー	**説明**
PF8 または Enter	プログラムを 1 ページ下にスクロールします。
PF7	プログラムを 1 ページ上にスクロールします。

3 プログラムを実行します。

出力が表示されます。Enter キーを繰り返し押すと、数ページ後（Bennett という名前の最後の従業員のデータが表示された後）でプログラムエディタに戻ります。

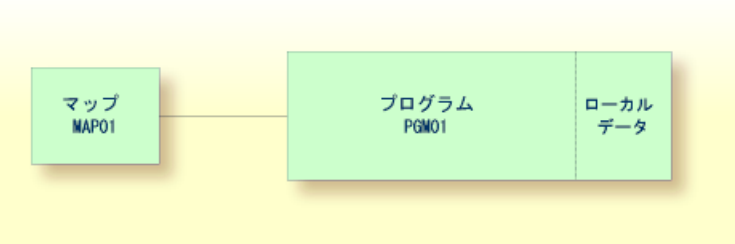
4 プログラムを格納します。

次の演習「ユーザー入力」に進みます。

6 ユーザー入力

データ、つまり出力の開始名と終了名の入力プロンプトをユーザーに対して表示する方法を習得します。

以下の演習を完了すると、サンプルのアプリケーションは次のモジュールで構成されます。



ユーザー入力の許可

プログラムを修正して、開始名および終了名の入力フィールドが出力に表示されるようにします。 これには、INPUT ステートメントを使用します。

手順 6.1. 入力フィールドを定義するには

1 END-DEFINE の下に次の行を挿入します。

```
INPUT (AD=MT)
  "Start:" #NAME-START /
  "End:   " #NAME-END
```

セッションパラメータ AD は「属性定義」を表し、値"M"は「変更可能出力フィールド」、値 "T" は 「小文字から大文字への変換」 を表します。

AD=MT の値 "M" は、INIT で定義されたデフォルト値（"ADKINSON" および "BENNETT"）が入力フィールドに表示されることを示します。ユーザーは異なる値を入力できます。"M" 値を省略すると、デフォルト値が定義されていても入力フィールドには表示されません。

AD=MT の値 "T" は、小文字の入力がすべて大文字に変換されてから処理されることを示します。 デモデータベースファイル内の名前はすべて大文字で定義されているため、これは重要です。 "T" 値を省略すると、すべての名前を大文字で入力する必要があります。 大文字で入力しないと、指定した名前が検出されなくなります。

"Start:" および "End:" はテキストフィールド（ラベル）であり、 引用符で囲んで指定します。

#NAME-START および #NAME-END はデータフィールド（入力フィールド）であり、任意の開始名および終了名をユーザーを入力できます。

スラッシュ（/）は、後続フィールドを新しい行に表示することを示します。

プログラムは次のようになります。

```
DEFINE DATA
LOCAL
  1 #NAME-START        (A20) INIT <"ADKINSON">
  1 #NAME-END          (A20) INIT <"BENNETT">
  1 EMPLOYEES-VIEW VIEW OF EMPLOYEES
    2 FULL-NAME
      3 NAME (A20)
    2 DEPT (A6)
    2 LEAVE-DATA
      3 LEAVE-DUE (N2)
END-DEFINE
*
INPUT (AD=MT)
  "Start:" #NAME-START /
  "End:  " #NAME-END
*
READ EMPLOYEES-VIEW BY NAME
  STARTING FROM #NAME-START
  ENDING AT #NAME-END
*
  DISPLAY NAME 3X DEPT 3X LEAVE-DUE
*
END-READ
*
END
```

2 プログラムを実行します。

定義したフィールドが出力に表示されます。

```
Start: ADKINSON
End:   BENNETT
```

3 デフォルト名を使用し、Enter キーを押します。

従業員のリストが表示されます。

4 プログラムエディタに戻るまで繰り返しEnterキーを押すか、EscMOREプロンプトで「EDIT」と入力します。

5 プログラムを格納します。

ユーザー入力のマップの設計

入力プロンプトをユーザーに表示するさまざまな方法を説明します。 マップエディタを使用して、前にプログラムで定義した同じフィールドを含むマップを作成します。 マップは別のオブジェクトであり、ユーザーインターフェイスのレイアウトをアプリケーションのビジネスロジックから分離するために使用します。

ここで作成するマップは次のようになります。

```
Ob _                                    Ob D CLS ATT  DEL    CLS ATT  DEL
.                                       .     T  D    Blnk    T  I    ?
.                                       .     A  D    _       A  I    )
.                                       .     A  N    ^       M  D    &
.                                       .     M  I    :       O  D    +
.                                       .     O  I    (
.                                       .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
(XXXXXXXXXX                                                          (XXXXXXXX


                          Start :XXXXXXXXXXXXXXXXXXX

                            End :XXXXXXXXXXXXXXXXXXX









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

マップの最初の行には、現在の日時を示すシステム変数が含まれます。 開始名および終了名をユーザーが指定できる 2 つのデータフィールド（入力フィールド）があります。 データフィールドの前にはテキストフィールド（ラベル）があります。

フィールドの長さは "X" 文字の数で示されます。 デリミタは異なるタイプのフィールドを区別するために使用されます。 サンプルマップでは、次のデフォルトデリミタを使用します。

デリミタ	フィールドタイプ
(	システム変数。
:	データフィールド

上記のマップには、次の手順が必要です。

- マップの作成
- テキストフィールドの定義
- データフィールドの定義
- データフィールドの名前の指定
- システム変数の追加
- フィールドの再位置決め
- マップのテスト
- マップの格納

マップの作成

マップエディタを起動し、マップを設計します。 マップエディタには、［**Edit Map**］メニューを使用してアクセスします。

手順 6.2. ［Edit Map］メニューにアクセスするには

1 プログラムエディタのコマンド行で「.」（ピリオド）を入力して、［**Development Functions**］メニューに戻ります。

2 ［**Development Functions**］メニューの下部で、次の情報を入力して Enter キーを押します。

```
                  Code .. E     Type .. M
                                Name .. ____________________________________



Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Menu  Exit                                                      Canc
```

"E" は［**Edit Object**］機能、"M" はオブジェクトタイプマップを示します。

注意: ［**Create Object**］用のコード「C」を入力することもできます。ただし、この場合は、オブジェクト名の入力プロンプトが表示されます。

［**Edit Map**］メニューが表示されます。

```
16:43:41                ***** NATURAL MAP EDITOR *****              2007-03-20
User SAG                         - Edit Map -                   Library TUTORIAL

                    Code    Function
                    ----    --------------------------------
                      D     Field and Variable Definitions
                      E     Edit Map
                      I     Initialize new Map
                      H     Initialize a new Help Map
                      M     Maintenance of Profiles & Devices
                      S     Save Map
                      T     Test Map
                      W     Stow Map
                      ?     Help
                      .     Exit



             Code .. I     Name .. _________       Profile .. SYSPROF_

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Test  Edit
```

▶手順 6.3. 新しいマップを初期化するには

1 ［**Edit Map**］メニューの下部で、次の情報を入力して Enter キーを押します。

```


             Code .. I     Name .. MAP01___       Profile .. SYSPROF_

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Test  Edit
```

［**Define Map Settings**］画面が表示されます。

```
16:43:42               Define Map Settings for MAP                  2007-03-20

 Delimiters           Format                          Context
 -----------------    ----------------------------    --------------------------
 Cls Att CD  Del      Page Size ..... 23              Device Check .... _________
  T   D      BLANK    Line Size ..... 79              WRITE Statement   _
  T   I      ?        Column Shift ... 0 (0/1)        INPUT Statement   X
  A   D      _        Layout ......... ________       Help  ____________________
  A   I      )         dynamic ....... N (Y/N)         as field default N (Y/N)
  A   N      ^        Zero Print ..... N (Y/N)
  M   D      &        Case Default ... UC (UC/LC)
  M   I      :        Manual Skip .... N (Y/N)        Automatic Rule Rank  1
  O   D      +        Decimal Char ... .              Profile Name .... SYSPROF
  O   I      (        Standard Keys .. N (Y/N)
                      Justification .. L (L/R)        Filler Characters
                      Print Mode ..... __             ------------------------
                                                      Optional, Partial ....
                      Control Var .... ________       Required, Partial ....
                                                      Optional, Complete ...
                                                      Required, Complete ...

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                   Let
```

この画面で、マップのデフォルト設定（マップのサイズなど）を定義します。 この画面ではデリミタ文字を変更できます。 ただし、このチュートリアルではデフォルトのデリミタ文字を使用します。 充填文字のみを定義します（次の手順を参照）。

デリミタ文字は、フィールドに割り当てられたクラスと属性の組み合わせを示します。 このチュートリアルでは、太字で示すデリミタを使用します。

```
Delimiters
 -----------------
 Cls Att CD  Del
  T   D      BLANK
  T   I      ?
  A   D      _
  A   I      )
  A   N      ^
  M   D      &
  M   I      :
  O   D      +
  O   I      (
```

- コロンは、変更可能な入力および出力フィールド（**Cls** 列の "M"）および高輝度フィールド（**Att** 列の "I"）を示します。

- 左カッコは、出力専用フィールド（**Cls** 列の "O"）および高輝度フィールド（**Att** 列の "T"）を示します。
- デリミタ文字を使用しない（画面に "BLANK" と表示される）と、フィールドがデフォルト属性（**Att** 列の "D"）を持つテキスト定数（**Cls** 列の "T"）であるとみなされます。

2 次に示すように、充填文字オプションごとに下線（_）を入力します。

```
Filler Characters
 ------------------------
 Optional, Partial .... _
 Required, Partial .... _
 Optional, Complete ... _
 Required, Complete ... _
```

充填文字は、マップ内の入力フィールドの空の位置を埋めるために使用されます。これにより、ユーザーは入力時にフィールドの正確な位置と長さを確認することができます。

3 Enter キーを押して、変更内容を保存します。

4 Enter キーをもう一度押して、マップ編集エリアを呼び出します。

マップ編集エリアは、画面分割モードで表示されます。

```
Ob _                                    Ob D CLS ATT  DEL    CLS ATT  DEL
.                                       .     T  D    Blnk    T  I    ?
.                                       .     A  D    _       A  I    )
.                                       .     A  N    ^       M  D    &
.                                       .     M  I    :       O  D    +
.                                       .     O  I    (
.                                       .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----


Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

注意: ［**Define Map Settings**］画面は、新しいマップを初期化する場合にのみ表示されます。 マップを編集しているときにマップ編集エリアで PF2（Mset）キーを押すと、［**Define Map Settings**］画面が呼び出されます。

画面分割モードでは、画面の上半分に有効なデリミタ文字が表示されます。 画面の下半分は、マップを作成するための編集エリアです。

プログラムエディタおよびデータエリアエディタ（このチュートリアルで後述）と異なり、マップエディタには Natural システムコマンドを入力できるコマンド行またはコマンドプロンプトはありません。 マップエディタの多くの機能は、行コマンドまたはフィールドコマンドを使用して（以下を参照）、または PF キーを使用して実行されます。

PF9 キーを押すと、画面分割モードとフルスクリーンモード（編集エリアをフルサイズで表示）を切り替えることができます。

テキストフィールドの定義

2 つのテキストフィールド（定数またはラベルとも呼ばれる）をマップに追加します。

手順 6.4. テキストフィールドを定義するには

1 マップ編集エリアで、4 行目の最初の位置にカーソルを移動し、次のように入力します。

```
Start
```

2 次の行の最初の位置にカーソルを移動し、次のように入力します。

```
End
```

マップは次のようになります。

```
Ob _                                    Ob D CLS ATT  DEL    CLS ATT  DEL
.                                       .     T  D    Blnk    T  I    ?
.                                       .     A  D    _       A  I    )
.                                       .     A  N    ^       M  D    &
.                                       .     M  I    :       O  D    +
.                                       .     O  I    (
.                                       .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----


Start
End
```

```
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

注意: 最後の変更を取り消すには、Enter キーを押す前に PF12 キーを押します。

データフィールドの定義

2つのデータフィールドをマップに追加します。これらは、開始名および終了名をユーザーが指定できる入力フィールドです。

データフィールドは2つの方法で定義できます。従来の方法であるデータフィールドに正しい長さを定義する必要があります。または、ユーザーフレンドリなデータフィールドをリストから選択するだけで、正しい長さが自動的に使用されます。 これらの 2 つの方法について、以下で説明します。

手順 6.5. 長さを指定する必要のあるデータフィールドを定義するには

1 **Start**テキストフィールドの後に次のように入力します（テキストフィールドとデータフィールドの間にスペースを残します）。

```
:X(20)
```

コロン（:）はデリミタ文字であり、データフィールドが変更可能で、強化されていることを示します。データフィールドの長さは20文字で定義されています。フィールドの長さは "X" 文字で示されています。

2 Enter キーを押します。

マップは次のようになります。

```
Ob _                                    Ob D CLS ATT  DEL    CLS ATT  DEL
.                                       .     T  D    Blnk    T  I    ?
.                                       .     A  D    _       A  I    )
.                                       .     A  N    ^       M  D    &
.                                       .     M  I    :       O  D    +
.                                       .     O  I    (
.                                       .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----



Start :XXXXXXXXXXXXXXXXXXX
```

```
End




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

手順 6.6. データフィールドをリストから選択するには

1 画面の左上にある［**Ob**］フィールドで次のように入力し、Enter キーを押します。

```
P PGM01
```

プログラム PGM01 で現在使用しているデータフィールドが画面に表示されます。 マップで使用できるフィールドの前には、番号が付きます。

```
Ob P PGM01                              Ob D CLS ATT  DEL    CLS ATT  DEL
1 #NAME-START               A20         .     T  D    Blnk    T  I    ?
2 #NAME-END                 A20         .     A  D    _       A  I    )
. EMPLOYEES-VIEW            *V1         .     A  N    ^       M  D    &
. FULL-NAME                 *2          .     M  I    :       O  D    +
3 NAME                      A20         .     O  I    (
4 DEPT                      A6          .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----



Start :XXXXXXXXXXXXXXXXXXX
End




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

PGM01 で定義されているすべてのデータフィールドが画面に表示されるわけではありません。 データフィールドリストのページをスクロールするには、［**Ob**］フィールド（文字 "P" が現在含まれているフィールド）で次のいずれかの配置コマンドを入力します。

コマンド	説明
+	リストの次のページに進みます。
-	リストの前のページに進みます。
++	リストの最後にジャンプします。
--	リストの先頭にジャンプします。

2 **End** テキストフィールドの後に次のように入力し（テキストフィールドとデータフィールドの間にスペースを残します）、Enter キーを押します。

```
:2
```

コロン（:）はデリミタ文字であり、データフィールドが変更可能で、強化されていることを示します。 2 は #NAME-END に割り当てられた番号です。

正しい長さ（この場合は 20 文字）でデータフィールドが自動的に定義されます。 フィールドの長さは "X" 文字で示されています。

データフィールドの名前の指定

次の説明は、手動で定義した開始名のデータフィールドに対してのみ適用されます。 リストから選択した終了名のデータフィールドには適用されません。ユーザー定義変数用に新しいデータフィールドを作成すると、Natural によってフィールド名が割り当てられます。 このフィールド名には数値が含まれます。 新しく作成したフィールドの名前を、プログラムで定義された変数名に調整する必要があります。

同じ名前 #NAME-START および #NAME-END がプログラムで使用されていることを確認します。 これらのフィールドの出力（ユーザー入力）は、プログラム内の対応するユーザー定義変数に渡されます。

▶手順 6.7. データフィールドの名前を定義するには

1 開始名のデータフィールドの最初の位置から開始して、次のように入力し、Enter キーを押します。

```
.E
```

または:

データフィールド内の任意の場所にカーソルを置き、PF5 を押します。

指定したフィールドに対する拡張フィールド編集セクションが表示されます。

```
Fld #001                                                          Fmt A20
--------------------------------------------------------------------------------
AD= MIT'_'____     ZP=          SG=         HE= _____________________   Rls 0
AL= ______         CD= __       CV= _____________________________   Mod Undef
PM= __  DF=                     DY= _____________________________
EM= _____________________________ SB= _____________________________

001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----



Start .EXXXXXXXXXXXXXXXXXX
End :XXXXXXXXXXXXXXXXXXX









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      HELP  Mset  Exit  <---  --->  --    -     +           <     >     Let
```

画面の左上隅にある［**Fld**］フィールドに、Natural によって割り当てられたフィールド名 "#001" が表示されます。

2 ［**Fld**］フィールドに「#NAME-START」と入力します。

3 拡張フィールド編集セクションを終了するには、PF3 を押します。

終了値のデータフィールドは前の演習でリストから選択されているため、終了名について上記の手順を繰り返す必要はありません。 この場合、#NAME-END は定義済みです。 必要に応じて、上記で説明したように、行コマンド .E を使用してこれをチェックできます。

注意: #NAME-END には、#NAME-START で定義されなかった追加の値 "L" がセッションパラメータ AD に存在します。 "L" は、フィールドの値が左揃えで表示されることを示します。 これは英数字フィールドのデフォルト値であるため、#NAME-START に対して定義する必要はありません。

システム変数の追加

Natural システム変数には、現在のライブラリ、ユーザー、日時など、現在の Natural セッションに関する情報が含まれます。 これらは、Natural プログラム内のどこからでも参照できます。 システム変数はすべてアスタリスク（*）で始まります。

日時のシステム変数をマップに追加します。 プログラムが実行されると、現在の日時がマップに表示されます。

手順 6.8. システム変数を追加するには

1 先頭行の最初の位置にカーソルを移動し、次のように入力します。

```
(*DAT4I
```

左カッコは、出力専用および強調表示としてシステム変数を識別するデリミタ文字です。

2 2 番目の行の最初の位置にカーソルを移動し、次のように入力します。

```
(*TIMX
```

マップは次のようになります。

```
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
(*DAT4I
(*TIMX

Start :XXXXXXXXXXXXXXXXXXX
End :XXXXXXXXXXXXXXXXXXX
```

3 Enter キーを押します。

システム変数名の代わりに "X" 文字が表示されます。

フィールドの再位置決め

行コマンドおよびフィールドコマンドを使用して、追加したフィールドの再位置決めを行います。

手順 6.9. 1 つのフィールドを移動するには

1 2行目のシステム変数の最初の位置から開始して、次のフィールドコマンドを入力します。

```
.M
```

例：

```
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
(XXXXXXXXXX
.MXXXXXXX

Start :XXXXXXXXXXXXXXXXXXX
End :XXXXXXXXXXXXXXXXXXX
```

フィールドの先頭からフィールドコマンドが入力されています。 これは、入力するフィールドに対してのみ適用されます。

2 システム変数の移動先となる位置（先頭行の列 70）にカーソルを移動します。

3 Enter キーを押します。

システム変数がカーソル位置に移動されます。

手順 6.10. 空行を挿入するには

1 4 行目（開始名）の最初の位置から開始して、次の行コマンドを入力します。

```
..I(1)
```

例：

```
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
(XXXXXXXXXX                                                        (XXXXXXXX


..I(1):XXXXXXXXXXXXXXXXXXX
End :XXXXXXXXXXXXXXXXXXX
```

行の先頭から行コマンドが入力されています。 これは、入力する行全体に対して適用されます。

2 Enter キーを押します。

行コマンドを入力した行の下に空行が挿入されます。

手順 6.11. 行を中央揃えにするには

1 4 行目（開始名）の最初の位置から開始して、次の行コマンドを入力します。

```
..C
```

例：

```
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
(XXXXXXXXXX                                                        (XXXXXXXX


..Crt :XXXXXXXXXXXXXXXXXXX

End :XXXXXXXXXXXXXXXXXXX
```

2 Enter キーを押します。

行が中央揃えになります。

手順 6.12. 複数のフィールドを移動するには

1 6 行目（**End**）のテキストフィールドの最初の位置およびこの行のデータフィールドの最初の位置から開始して、次のフィールドコマンドを入力します。

```
.M
```

例：

```
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
(XXXXXXXXXX                                                        (XXXXXXXX


                          Start :XXXXXXXXXXXXXXXXXXX

.Md .MXXXXXXXXXXXXXXXXXX
```

2 テキストフィールドの開始位置（6 行目の列 30）にカーソルを移動します。

3 Enter キーを押します。

両方のフィールドが移動します。

このセクションの最初に示したようなマップが表示されます。

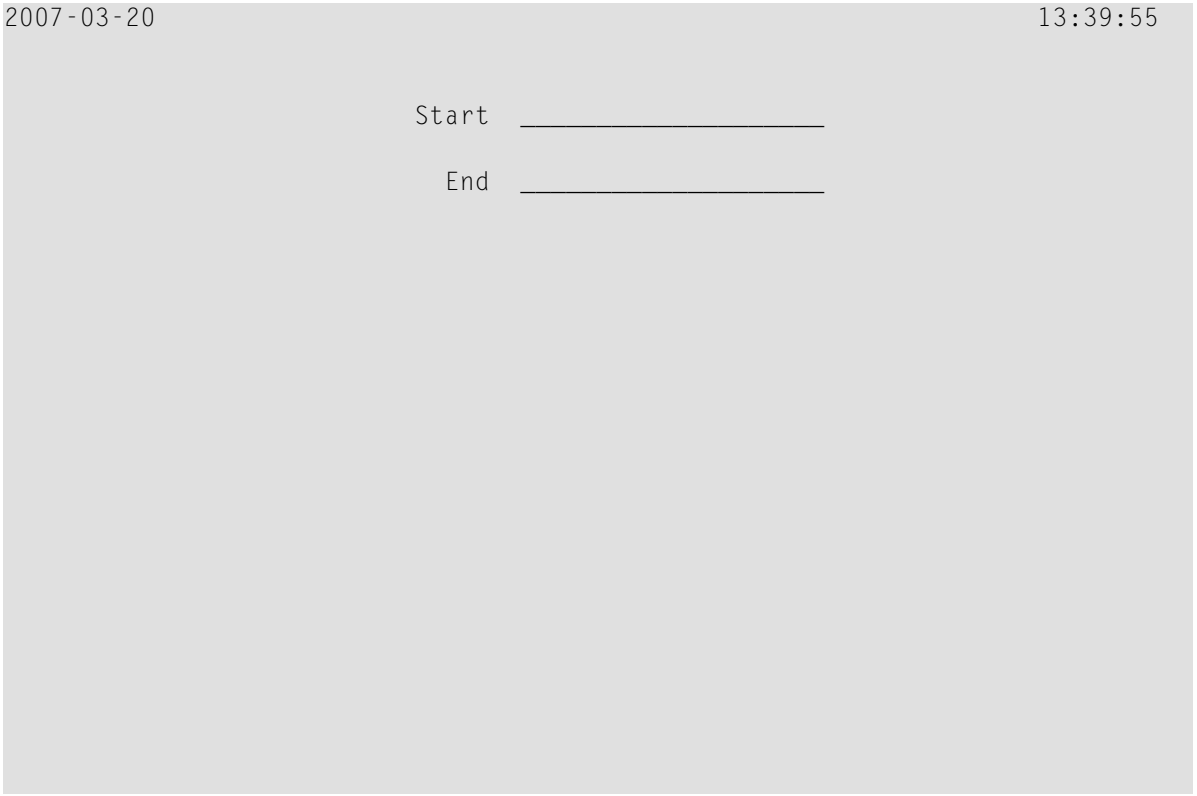
マップのテスト

マップが意図したとおりに動作するかどうかをテストします。

手順 6.13. マップをテストするには

1 PF4 キーを押します。

次の出力が表示されます。



開始名の入力フィールドはマップの最初の入力フィールドであるため、自動的に選択されます。 どちらの入力フィールドにも、充填文字が含まれています。

注意: 挿入モードで操作する場合、テキストを入力する前に充填文字を削除する必要があります。 デフォルトである上書きモードでは、この操作は不要です。

2 Enter キーを押して、マップエディタに戻ります。

マップの格納

マップのテストが正常に行われたら、プログラムで検出できるようにするため、マップを格納する必要があります。

手順 6.14. マップを Stow するには

1 PF3 を押して［**Edit Map**］メニューに戻ります。

2 ［**Code**］フィールドに次のように入力し、Enter キーを押します。

```
W
```

プログラムからのマップの起動

マップの格納が完了したら、`WRITE` または `INPUT` ステートメントを使用して Natural プログラムから起動できます。

手順 6.15. プログラムからマップを起動するには

1 ［**Edit Map**］メニューのコマンド行で次のいずれかを入力して、プログラムエディタに戻ります。

```
EDIT PGM01
```

```
E PGM01
```

2 前に定義した `INPUT` 行を次の行で置き換えます。

```
INPUT USING MAP 'MAP01'
```

これにより、設計したマップが起動されます。

マップをユーザー定義変数と区別するため、マップ名を一重引用符で囲む必要があります。

プログラムは次のようになります。

```
DEFINE DATA
LOCAL
  1 #NAME-START        (A20) INIT <"ADKINSON">
  1 #NAME-END          (A20) INIT <"BENNETT">
  1 EMPLOYEES-VIEW VIEW OF EMPLOYEES
    2 FULL-NAME
      3 NAME (A20)
    2 DEPT (A6)
    2 LEAVE-DATA
      3 LEAVE-DUE (N2)
END-DEFINE
*
INPUT USING MAP 'MAP01'
*
READ EMPLOYEES-VIEW BY NAME
  STARTING FROM #NAME-START
  ENDING AT #NAME-END
*
  DISPLAY NAME 3X DEPT 3X LEAVE-DUE
*
END-READ
*
END
```

3 プログラムを実行します。

マップが表示されます。

4 プログラムエディタに戻るまで繰り返しEnterキーを押すか、Esc MORE プロンプトで「EDIT」と入力します。

5 プログラムを格納します。

終了名を常に使用するための操作

プログラムをコーディングしても、終了名が指定されていなければデータは検出されません。

開始名および終了名の初期値を削除すると、これらの名前をユーザーが常に指定する必要があります。 ユーザーが終了名を指定しなくても、終了名が常に使用されるようにするため、対応するステートメントを追加します。

手順 6.16. 終了名を使用するには

1 DEFINE DATAブロックで、フィールド#NAME-STARTおよび#NAME-ENDのデフォルト値（INIT）を削除します。対応する行は次のようになります。

```
1 #NAME-START        (A20)
1 #NAME-END          (A20)
```

2 INPUT USING MAP 'MAP01'の下に、次の行を挿入します。

```
IF #NAME-END = ' ' THEN
    MOVE #NAME-START TO #NAME-END
END-IF
```

#NAME-ENDフィールドが空白になっている（ユーザーが終了名を入力しなかった）場合は、開始名が自動的に終了名として使用されます。

注意: ステートメントMOVE #NAME-START TO #NAME-ENDを使用する代わりに、ASSIGNまたはCOMPUTEステートメントの次の変形#NAME-END := #NAME-STARTを使用することもできます。

プログラムは次のようになります。

```
DEFINE DATA
LOCAL
  1 #NAME-START        (A20)
  1 #NAME-END          (A20)
  1 EMPLOYEES-VIEW VIEW OF EMPLOYEES
    2 FULL-NAME
      3 NAME (A20)
    2 DEPT (A6)
    2 LEAVE-DATA
      3 LEAVE-DUE (N2)
END-DEFINE
*
INPUT USING MAP 'MAP01'
*
IF #NAME-END = ' ' THEN
  MOVE #NAME-START TO #NAME-END
END-IF
*
READ EMPLOYEES-VIEW BY NAME
  STARTING FROM #NAME-START
  ENDING AT #NAME-END
*
  DISPLAY NAME 3X DEPT 3X LEAVE-DUE
*
END-READ
```

```
*
END
```

3 プログラムを実行します。

4 結果のマップで、開始名の入力を求めるフィールドに「JONES」と入力し、Enter キーを押します。

結果のリストには、"Jones" という名前の従業員のみが表示されます。

5 Enter キーを押して、プログラムエディタに戻ります。

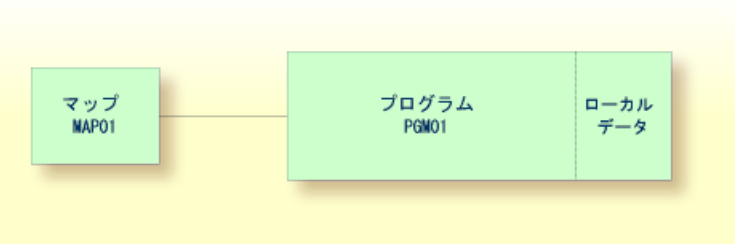
6 プログラムを格納します。

次の演習「*ループおよびラベル*」に進みます。

7 ループおよびラベル

ループおよびラベルを追加してプログラムを強化します。

以下の演習を完了すると、サンプルのアプリケーションを構成するモジュールは前の章と同じになります。



反復使用の許可

現時点では、マップが表示され、リストが表示された時点でプログラムは終了します。 プログラムを再スタートしなくても新しい従業員リストを直ちに表示できるようにするため、対応するプログラムコードを REPEAT ループに挿入します。

▶手順 7.1. 繰り返しループを定義するには

1 END-DEFINE の下に次の行を挿入します。

```
RP1. REPEAT
```

REPEATは、繰り返しループの開始を定義します。 RP1.は、繰り返しループを終了するときに使用されるラベルです（以下で定義）。

2 END ステートメントの前に次の行を挿入して、繰り返しループの終了を定義します。

```
END-REPEAT
```

3 INPUT USING MAP 'MAP01' の下に、次の行を挿入します。

```
IF #NAME-START = '.' THEN
  ESCAPE BOTTOM (RP1.)
END-IF
```

IF ステートメントは、END-IF で終了する必要があり、#NAME-START フィールドの内容をチェックします。 このフィールドに「.」（ピリオド）を入力すると、ループの終了に ESCAPE BOTTOM ステートメントが使用されます。 ループの後にある最初のステートメント（この場合は END）から処理が続行します。

ループにラベル（ここではRP1）を割り当てると、このループをESCAPE BOTTOMステートメントで参照できます。 ループはネストすることができるため、どのループを終了するか指定する必要があります。 指定がない場合は、最も内側にあるアクティブなループが終了します。

プログラムは次のようになります。

```
DEFINE DATA
LOCAL
  1 #NAME-START        (A20)
  1 #NAME-END          (A20)
  1 EMPLOYEES-VIEW VIEW OF EMPLOYEES
    2 FULL-NAME
      3 NAME (A20)
    2 DEPT (A6)
    2 LEAVE-DATA
      3 LEAVE-DUE (N2)
END-DEFINE
*
RP1. REPEAT
*
  INPUT USING MAP 'MAP01'
*
  IF #NAME-START = '.' THEN
    ESCAPE BOTTOM (RP1.)
  END-IF
*
  IF #NAME-END = ' ' THEN
    MOVE #NAME-START TO #NAME-END
  END-IF
*
  READ EMPLOYEES-VIEW BY NAME
    STARTING FROM #NAME-START
    ENDING AT #NAME-END
*
    DISPLAY NAME 3X DEPT 3X LEAVE-DUE
*
  END-READ
*
END-REPEAT
*
END
```

注意: 読みやすくするため、REPEATループのコンテンツはインデントされています。

4 プログラムを実行します。

5 結果のマップで、開始名の入力を求めるフィールドに「JONES」と入力し、Enter キーを押します。

結果のリストに、"Jones" という名前の従業員が表示されます。 Enter キーを押します。 REPEAT ループにより、マップが再表示されます。 終了名にも「JONES」と入力されていることがわかります。

6 マップを終了するには、開始名の入力を求めるフィールドに「.」（ピリオド）を入力し、Enter キーを押します。このフィールドに表示されたままになっている名前の残りの文字は、必ず削除してください。

7 プログラムを格納します。

情報が見つからないことを示すメッセージの表示

データベースで検出できない開始名をユーザーが入力したときに表示するメッセージを定義します。

手順 7.2. 指定の従業員が見つからない場合に表示するメッセージを定義するには

1 READ ステートメントを含む行にラベル RD1. を追加します。行は次のようになります。

```
RD1. READ EMPLOYEES-VIEW BY NAME
```

2 END-READ の下に次の行を挿入します。

```
IF *COUNTER (RD1.) = 0 THEN
   REINPUT 'No employees meet your criteria.'
END-IF
```

READ ループで検出されたレコード数をチェックするため、システム変数 *COUNTER を使用します。 この変数の値が 0 である（指定した名前の従業員が見つからない）と、REINPUT ステートメントで定義されたメッセージがマップ下部に表示されます。

READ ループを識別するため、ループにラベル（ここでは RD1.）を割り当てます。 データベースにアクセスする複雑なプログラムでは多くのループが含まれている場合があるため、参照するループを指定する必要があります。

プログラムは次のようになります。

```
DEFINE DATA
LOCAL
  1 #NAME-START        (A20)
  1 #NAME-END          (A20)
  1 EMPLOYEES-VIEW VIEW OF EMPLOYEES
    2 FULL-NAME
      3 NAME (A20)
    2 DEPT (A6)
    2 LEAVE-DATA
      3 LEAVE-DUE (N2)
END-DEFINE
*
RP1. REPEAT
*
  INPUT USING MAP 'MAP01'
*
  IF #NAME-START = '.' THEN
    ESCAPE BOTTOM (RP1.)
  END-IF
*
  IF #NAME-END = ' ' THEN
    MOVE #NAME-START TO #NAME-END
  END-IF
*
  RD1. READ EMPLOYEES-VIEW BY NAME
    STARTING FROM #NAME-START
    ENDING AT #NAME-END
*
    DISPLAY NAME 3X DEPT 3X LEAVE-DUE
*
  END-READ
*
  IF *COUNTER (RD1.) = 0 THEN
    REINPUT 'No employees meet your criteria.'
  END-IF
*
END-REPEAT
*
END
```

3 プログラムを実行します。

4 結果のマップで、デモデータベースで定義されていない開始名（「XYZ」など）を入力し、Enter キーを押します。

マップにメッセージが表示されます。

5 マップを終了するには、開始名の入力を求めるフィールドに「.」（ピリオド）を入力し、Enterキーを押します。このフィールドに表示されたままになっている名前の残りの文字は、必ず削除してください。

6 プログラムを格納します。

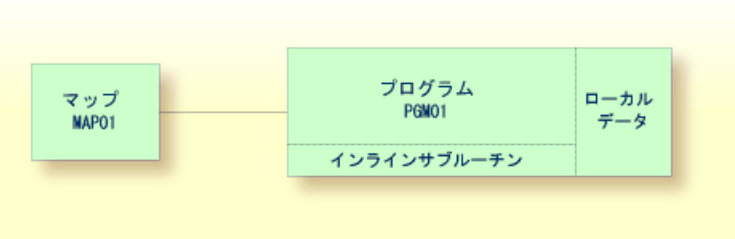
次の演習「*インラインサブルーチン*」に進みます。

8 インラインサブルーチン

Naturalでは、プログラム内で直接定義されるインラインサブルーチンと、プログラム外で別のオブジェクトとして格納される外部サブルーチン（このチュートリアルで後述）という2種類のサブルーチンが区別されます。

#MARK という名前のユーザー定義変数にアスタリスク（*）を移動するインラインサブルーチンをプログラムに追加します。 このサブルーチンは、従業員が20日以上の休暇を取った場合に起動されます。

以下の演習を完了すると、サンプルのアプリケーションは次の構造になります。



インラインサブルーチンの定義

サブルーチンをプログラムに追加します。

手順 8.1. サブルーチンを定義するには

1 ユーザー定義変数 #NAME-END の下に次の行を挿入します。

```
1 #MARK               (A1)
```

この変数はサブルーチンで使用されます。 したがって、最初に定義しておく必要があります。

2 サブルーチンを定義するため、END ステートメントの前に次の行を挿入します。

```
DEFINE SUBROUTINE MARK-SPECIAL-EMPLOYEES
  MOVE '*' TO #MARK
END-SUBROUTINE
```

このサブルーチンが実行されると、アスタリスク（*）が #MARK に移動します。

注意: ステートメント MOVE '*' TO #MARK の代わりに、ASSIGN または COMPUTE ステートメントの次の変形 #MARK := '*' を使用することもできます。

3 DISPLAY ステートメントを次のように変更します。

```
DISPLAY NAME 3X DEPT 3X LEAVE-DUE 3X '>=20' #MARK
```

これにより、出力に新しい列が表示されます。見出しは ">=20" です。該当する従業員が 20 日以上の休暇を取った場合、列にアスタリスク（*）が表示されます。

インラインサブルーチンの実行

インラインサブルーチンの定義が完了したので、インラインサブルーチンを実行するコードを指定できます。

手順 8.2. インラインサブルーチンを実行するには

1 DISPLAY ステートメントの下に次の行を挿入します。

```
IF LEAVE-DUE >= 20 THEN
  PERFORM MARK-SPECIAL-EMPLOYEES
ELSE
  RESET #MARK
END-IF
```

20 日以上の休暇を取った従業員が検出されると、MARK-SPECIAL-EMPLOYEES という名前の新しいサブルーチンが実行されます。 従業員の休暇が 20 日未満の場合、#MARK のコンテンツは空白にリセットされます。

プログラムは次のようになります。

```
DEFINE DATA
LOCAL
  1 #NAME-START        (A20)
  1 #NAME-END          (A20)
  1 #MARK              (A1)
  1 EMPLOYEES-VIEW VIEW OF EMPLOYEES
    2 FULL-NAME
      3 NAME (A20)
    2 DEPT (A6)
    2 LEAVE-DATA
      3 LEAVE-DUE (N2)
END-DEFINE
*
RP1. REPEAT
*
  INPUT USING MAP 'MAP01'
*
  IF #NAME-START = '.' THEN
```

```
    ESCAPE BOTTOM (RP1.)
  END-IF
*
  IF #NAME-END = ' ' THEN
    MOVE #NAME-START TO #NAME-END
  END-IF
*
  RD1. READ EMPLOYEES-VIEW BY NAME
    STARTING FROM #NAME-START
    ENDING AT #NAME-END
*
    IF LEAVE-DUE >= 20 THEN
      PERFORM MARK-SPECIAL-EMPLOYEES
    ELSE
      RESET #MARK
    END-IF
*
    DISPLAY NAME 3X DEPT 3X LEAVE-DUE 3X '>=20' #MARK
*
  END-READ
*
  IF *COUNTER (RD1.) = 0 THEN
    REINPUT 'No employees meet your criteria.'
  END-IF
*
END-REPEAT
*
DEFINE SUBROUTINE MARK-SPECIAL-EMPLOYEES
  MOVE '*' TO #MARK
END-SUBROUTINE
*
END
```

2 プログラムを実行します。

3 結果のマップで、「JONES」と入力して Enter キーを押します。

従業員のリストに列が追加されて表示されます。

4 プログラムエディタに戻るには、MORE プロンプトで「EDIT」と入力します。

5 プログラムを格納します。

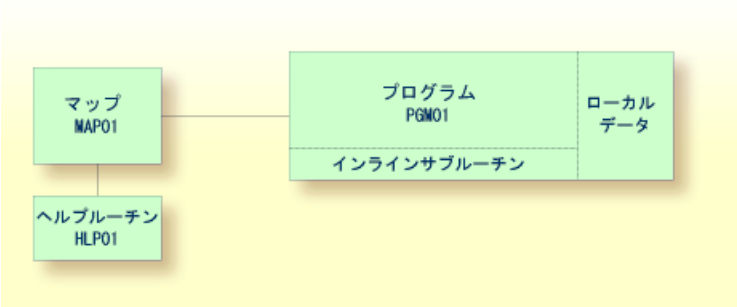
6 コマンド行で「.」（ピリオド）を入力して、［**Development Functions**］メニューに戻ります。

次の演習「*処理ルールとヘルプルーチン*」に進みます。

9 処理ルールとヘルプルーチン

処理ルールとヘルプルーチンは、マップ内のフィールドに対して定義されます。

以下の演習を完了すると、サンプルのアプリケーションは次のモジュールで構成されます（処理ルールは別のモジュールとして定義できず、常にマップの一部になります）。



処理ルールの定義

ユーザーが開始名を指定せずに Enter キーを押した場合に表示するメッセージを定義します。

手順 9.1. 処理ルールを定義するには

1 ［**Development Functions**］メニューで次のように入力して、マップエディタに戻ります。

```
                Code .. E    Type .. _

                             Name .. MAP01_____________________________
```

2 開始名の入力フィールドの最初の位置から開始して、次のように入力します。

```
.P
```

例：

```
                          Start .PXXXXXXXXXXXXXXXXXX

                            End :XXXXXXXXXXXXXXXXXX
```

3 Enter キーを押します。

選択したフィールドについて次の画面が表示されます。

```
Variables used in current map                                                Mod
#NAME-START(A20)                                                              U
#NAME-END(A20)                                                                U




Rule _____________________________________ Field #NAME-START
>                                          > + Rank 0    S      L 1     Struct Mode
ALL    ....+....10...+....+....+....30...+....+....+....50...+....+....+....70.
  0010
  0020
  0030
  0040
  0050
  0060
  0070
  0080
  0090
  0100
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test        --    -     +     Full  Sc=         Let
```

4 次の処理ルールを入力します。

```
IF & = ' ' THEN
  REINPUT 'Please enter a starting name.'
  MARK *&
END-IF
```

処理ルール内のアンパサンド（&）は、フィールド名にダイナミックに置き換えられます。この場合は、#NAME-START に置き換えられます。#NAME-START が空白の場合は、REINPUT ステートメントで定義されたメッセージが表示されます。

MARK は REINPUT ステートメントのオプションです。 構文は MARK *fieldname* です。 MARK は、REINPUT ステートメントの実行時にカーソルが置かれるフィールドを指定します。 この場合は、カーソルが #NAME-START フィールドに置かれます。

5 ［**Rank**］フィールドに「1」と入力します。

```
Rule _____________________________________ Field #NAME-START
>                                          > + Rank 1    S 2    L 1     Struct Mode
ALL    ....+....10...+....+....+....30...+....+....+....50...+....+....+....70.
  0010 IF & = ' ' THEN
  0020   REINPUT 'Please enter a starting name.'
  0030   MARK *&
  0040 END-IF
  0050
```

ランクにより、異なるフィールドに対するルールの処理順序が定義されます。 ランク 1 のルールはすべて最初に処理され、その後にランク 2 などが続きます。

6 Enter キーを押して、入力内容を保存します。 次に PF3 キーを押してマップに戻ります。

注意: 処理ルールを再表示する場合は、コマンド .P1（ランク 1 のルールを表示）または .P*（このフィールドに定義されたすべてのルールのリストを表示）を使用する必要があります。

7 マップをテストします。

8 結果の出力で、任意の開始名を入力して Enter キーを押します。

出力画面が閉じます。

9 マップをもう一度テストします。 名前を入力せずに、Enter キーを押します。

処理ルールで定義したメッセージが、マップに表示されます。

10 出力画面を終了するには、開始名の入力を求めるフィールドに「.」（ピリオド）を入力し、Enter キーを押します。

11 マップを Stow します（PF3 を押して［**Edit Map**］メニューに戻り、［**Code**］フィールドで「W」と入力します）。

ヘルプルーチンの定義

ヘルプルーチンは、開始名の入力フィールドにカーソルが置かれているときにユーザーがヘルプキーを押すと表示されます。

まず、ヘルプルーチンを定義してから、それを特定のフィールドに関連付けます。

手順 9.2. ヘルプルーチンを作成するには

1 ［**Edit Map**］メニューで PF3 キーを押して、［**Development Functions**］メニューに戻ります。

2 ［**Development Functions**］メニューの下部で、次の情報を入力して Enter キーを押します。

```
                  Code .. C    Type .. H

                               Name .. HLP01_____________________________
```

"C" は［**Create Object**］機能、"H" はオブジェクトタイプのヘルプルーチン、および "HLP01" は新しいヘルプルーチンの名前を表します。

空のエディタが表示されます。

3 次のように入力します。

```
WRITE 'Type the name of an employee'
END
```

4 ヘルプルーチンを格納し HLP01ます。

手順 9.3. ヘルプルーチンをマップ上のフィールドに関連付けるには

1 ヘルプルーチンを入力した画面のコマンド行で次のように入力して、マップエディタに戻ります。

```
E MAP01
```

2 開始名のデータフィールドの最初の位置から開始して、次のように入力し、Enter キーを押します。

```
.E
```

または:

データフィールド内の任意の場所にカーソルを置き、PF5 を押します。

フィールドの拡張フィールド編集セクションが表示されます。

3 ［**HE**］フィールドで「'HLP01'」（一重引用符も含めます）と入力します。

これは、ヘルプルーチンの保存に使用した名前です。

```
Fld #NAME-START                                                    Fmt A20
------------------------------------------------------------------------------
AD= MIT'_'____     ZP=         SG=        HE= 'HLP01'_____________   Rls 2
AL= _____          CD= __      CV= ___________________________________   Mod User
PM= __  DF=                    DY= ___________________________________
EM= ____________________________ SB= ___________________________________
```

4 拡張フィールド編集セクションを終了するには、PF3 を押します。

5 マップをテストします。

6 結果の出力で、開始名の入力フィールドに疑問符（?）を入力し、Enter キーを押します。

定義したヘルプテキストが表示されます。

7 Enter キーを押して、マップに戻ります。

8 マップを終了するには、開始名の入力を求めるフィールドに「.」（ピリオド）を入力し、Enter キーを押します。

9 マップを Stow します（PF3 を押して［**Edit Map**］メニューに戻り、［**Code**］フィールドで「W」と入力します）。

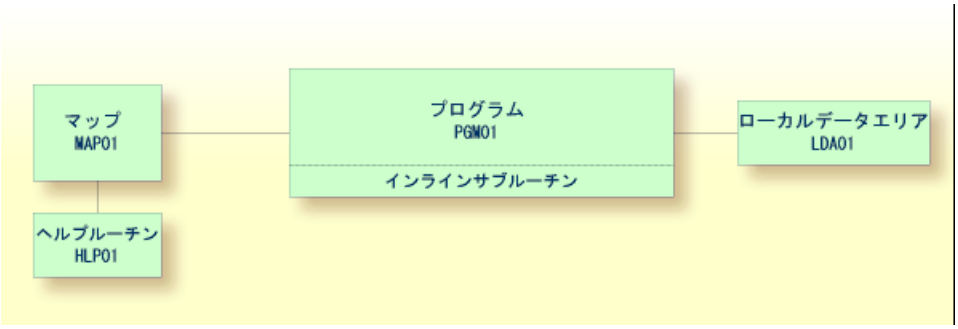
10 PF3 キーを押して［**Development Functions**］メニューに戻ります。

次の演習「*ローカルデータエリア*」に進みます。

10 ローカルデータエリア

現時点では、プログラムで使用するフィールドは、プログラム内の DEFINE DATA ステートメントで定義されています。 このフィールド定義をプログラム外のローカルデータエリア（LDA）に配置し、プログラムの DEFINE DATA ステートメントを使用して、ローカルデータエリアを名前で参照することもできます。 再利用および明確なアプリケーション構造という観点から見ると、通常、プログラム外のデータエリアにフィールドを定義することが推奨されます。

情報を DEFINE DATA ステートメントからローカルデータエリアに再配置します。 以下の演習を完了すると、サンプルのアプリケーションは次のモジュールで構成されます。



ローカルデータエリアの作成

必須フィールドを指定するデータエリアエディタを起動します。

手順 10.1. データエリアエディタを起動するには

■ ［**Development Functions**］メニューの下部で、次の情報を入力して Enter キーを押します。

```
Code .. C    Type .. L

             Name .. LDA01_____________________________
```

"C" は［**Create Object**］機能、"L" はオブジェクトタイプのローカルデータエリア、および "LDA01" は新しいローカルデータエリア名を表します。

データエリアエディタが表示されます。 オブジェクトタイプは "Local" に設定されています。 これは、画面の左上に示されます。

```
Local    LDA01      Library TUTORIAL                      DBID 11177 FNR     8
Command                                                                  > +
I T L  Name                              F Length     Miscellaneous
All -- --------------------------------- - ---------- ------------------------>
                                                                  S 0    L 1
```

データフィールドの定義

次のフィールドを定義します。

レベル（L 列）	名前	フォーマット（F 列）	長さ
1	#NAME-START	A	20
1	#NAME-END	A	20
1	#MARK	A	1

これらは、DEFINE DATA ステートメントで前に定義したユーザー定義変数です。

手順 10.2. データフィールドを定義するには

1 プログラムおよびデータエリアエディタを同じ画面に表示するには、次のように入力して画面分割モードを起動します。

```
SPLIT P PGM01
```

画面が 2 つのセクションに分割されます。 プログラムは画面の下半分に表示されます。 このモードではプログラムを変更できません。 データエリアエディタでユーザー定義変数を挿入する場合の参照として、プログラムを使用できます。 プログラムの次のページまたは前のページに進むには、コマンド SPLIT + および SPLIT - を使用します。

2 上の表に示す必須情報をすべて指定します。

ローカルデータエリアが次のように表示されます。

```
Local    LDA01      Library TUTORIAL                      DBID 11177 FNR     8
Command                                                                  > +
I T L  Name                             F Length     Miscellaneous
All -- -------------------------------- - ---------- ------------------------->
     1 #NAME-START                      A         20
     1 #NAME-END                        A         20
     1 #MARK                             A          1
```

```
------------------------------------------------------------------- S 0    L 1
Program     PGM01     Library TUTORIAL
  0010 DEFINE DATA
  0020 LOCAL
  0030   1 #NAME-START        (A20)
  0040   1 #NAME-END          (A20)
  0050   1 #MARK              (A1)
  0060   1 EMPLOYEES-VIEW VIEW OF EMPLOYEES
```

3 次のコマンドを入力して、画面分割モードを終了します。

```
SPLIT .
```

DDM からの必須データフィールドのインポート

プログラムの DEFINE DATA ステートメントで前に定義したデータフィールドをインポートします。 フィールドは、Natural データビューからデータエリアエディタに直接読み込まれます。 データビューは、データ定義モジュール（DDM）で定義されたデータベースフィールドを参照します。

▶手順 10.3. DDM からデータフィールドをインポートするには

1 すでに定義した変数の下の行で、**T** 列から開始して次のように入力します。

```
.V(EMPLOYEES)
```

例：

```
Local    LDA01       Library TUTORIAL                        DBID 11177 FNR
 8
 Command                                                                > +
 I T L  Name                             F Length     Miscellaneous

 All -- -------------------------------- - ---------- ------------------------->
      1 #NAME-START                       A         20

      1 #NAME-END                         A         20

      1 #MARK                             A          1

   . V( EMPLOYEES)
```

2 Enter キーを押します。

EMPLOYEES ビューが表示されます。

```
SYSGDA 4461: Mark fields to incorporate into data area.
Local     LDA01     Library TUTORIAL                      DBID 11177 FNR     8
View EMPLOYEES
I T L  Name                             F Length     Miscellaneous
- - -- -------------------------------- - ---------- -------------------------
     2 PERSONNEL-ID                     A          8 /* CNNNNNNN
  G  2 FULL-NAME                                     /* NAME INFORMATION
     3 FIRST-NAME                       A         20 /* FIRST/CHRISTIAN NAME
     3 MIDDLE-I                         A          1 /* MIDDLE INITIAL
     3 NAME                             A         20 /* SURNAME/FAMILY NAME
     2 MIDDLE-NAME                      A         20 /* SECOND/MIDDLE NAME
     2 MAR-STAT                         A          1 /* M=MARRIED
     2 SEX                              A          1
     2 BIRTH                            D            /* BIRTH-DATE (YYYY-MM-
     2 N§BIRTH                          I          2 /* INDICATOR OF BIRTH
  G  2 FULL-ADDRESS
  M  3 ADDRESS-LINE                     A         20 (1:8)/* ALL ADDRESS LINES
     3 CITY                             A         20 /* MAIN CITY/TOWN
     3 ZIP                              A         10 /* POSTAL ADDRESS CODE
     3 POST-CODE                        A         10 /* POSTAL ADDRESS CODE
     3 COUNTRY                          A          3 /* COUNTRY CODE
  G  2 TELEPHONE
- - -- -------------------------------- - ---------- -------------------------
```

3 **I** 列に任意の文字を入力して、次のフィールドをマークします。

```
PERSONNEL-ID
FULL-NAME
NAME
DEPT
LEAVE-DATA
LEAVE-DUE
```

これらのフィールドがすべてビューの最初のページに表示されるわけではありません。ビューで次のページにスクロールし、Enter キーを押します。

注意: フィールド PERSONNEL-ID は、後でサブプログラムを作成するときに使用します。

4 必須フィールドをすべてマークしたら、データエリアエディタが再表示されるまで Enter キーを押して処理を続行します。

ローカルデータエリアが次のように表示されます。

```
SYSGDA 4462: 6 field(s) of view EMPLOYEES included.
Local     LDA01     Library TUTORIAL                      DBID 11177 FNR     8
Command                                                                   > +
I T L  Name                            F Length     Miscellaneous
All -- --------------------------------- - ---------- -------------------------->
     1 #NAME-START                     A         20
     1 #NAME-END                       A         20
     1 #MARK                           A          1
  V  1 EMPLOYEES-VIEW                                 EMPLOYEES
     2 PERSONNEL-ID                    A          8 /* CNNNNNNN
  G  2 FULL-NAME                                      /* NAME INFORMATION
     3 NAME                            A         20 /* SURNAME/FAMILY NAME
     2 DEPT                            A          6 /* DDDDSS
  G  2 LEAVE-DATA                                     /* LEAVE/VACATION INFO
     3 LEAVE-DUE                       N        2.0 /* VACATION DAYS/YEAR
                                                                              
                                                                              
                                                                              
                                                                              
                                                                              
                                                                              
                                                                              
------------------------------------------------------------------- S 10   L 1
```

ビューの名前に、"-VIEW" が自動的に追加されます。 これは、プログラムですでに使用している名前と同じものです。

T 列は、変数タイプを示します。 ビューは "V" で示され、各グループは "G" で示されます。

5 ローカルデータを格納します。

プログラムからのローカルデータエリアの参照

ローカルデータエリアを格納すると、Natural プログラムから参照できます。

定義したローカルデータエリアを使用するため、プログラムの DEFINE DATA ステートメントを変更します。

手順 10.4. プログラムでローカルデータエリアを使用するには

1 データエリアエディタのコマンド行で次のように入力して、プログラムエディタに戻ります。

```
E PGM01
```

2 DEFINE DATA ステートメントで、LOCAL と END-DEFINE 間にある変数をすべて削除します（行コマンド .D を使用）。

3 LOCAL 行を次のように変更して、ローカルデータエリアへの参照を追加します。

```
LOCAL USING LDA01
```

プログラムは次のようになります。

```
DEFINE DATA
  LOCAL USING LDA01
END-DEFINE
*
RP1. REPEAT
*
  INPUT USING MAP 'MAP01'
*
  IF #NAME-START = '.' THEN
    ESCAPE BOTTOM (RP1.)
  END-IF
*
  IF #NAME-END = ' ' THEN
    MOVE #NAME-START TO #NAME-END
  END-IF
*
  RD1. READ EMPLOYEES-VIEW BY NAME
    STARTING FROM #NAME-START
    ENDING AT #NAME-END
*
    IF LEAVE-DUE >= 20 THEN
      PERFORM MARK-SPECIAL-EMPLOYEES
    ELSE
      RESET #MARK
```

```
    END-IF
*
    DISPLAY NAME 3X DEPT 3X LEAVE-DUE 3X '>=20' #MARK
*
  END-READ
*
  IF *COUNTER (RD1.) = 0 THEN
    REINPUT 'No employees meet your criteria.'
  END-IF
*
END-REPEAT
*
DEFINE SUBROUTINE MARK-SPECIAL-EMPLOYEES
  MOVE '*' TO #MARK
END-SUBROUTINE
*
END
```

4 プログラムを実行します。

5 前（DEFINE DATA ステートメントでローカルデータエリアを参照していない場合）と同じ結果になることを確認するため、開始名に「JONES」と入力して、Enter キーを押します。

6 プログラムエディタに戻るには、MORE プロンプトで「EDIT」と入力します。

7 プログラムを格納します。

次の演習「グローバルデータエリア」に進みます。

11 グローバルデータエリア

グローバルデータエリア（GDA）で定義されたデータは、複数のプログラム、外部サブルーチン、およびヘルプルーチンで共有することができます。

グローバルデータエリアにあるデータ要素の値に加えた変更は、そのグローバルデータエリアを参照するすべての Natural オブジェクトに影響を与えます。 したがって、グローバルデータエリアのソースを変更した場合は、そのグローバルデータエリアを参照する、作成済みのすべての Natural オブジェクトをもう一度格納する必要があります。オブジェクトを格納する順序は重要です。まず、グローバルデータエリアを格納してから、プログラムを格納する必要があります。この順序を逆にすると、グローバルデータエリアにある新しい要素を検出できなくなるため、プログラムを格納できなくなります。

プログラムおよび後で作成する外部サブルーチンで共有するグローバルデータエリアを作成します。 グローバルデータエリアのベースとして、作成済みのローカルデータエリアの一部の情報を使用します。

以下の演習を完了すると、サンプルのアプリケーションは次のモジュールで構成されます。

既存のローカルデータエリアからのグローバルデータエリアの作成

既存のデータエリアを編集し、それを異なる名前および異なるタイプで保存することにより、既存のデータエリアから新しいデータエリアを作成できます。 元のデータエリアは変更されないまま残り、新しいデータエリアを編集できます。 フィールド #NAME-START および #NAME-END はグローバルデータエリアには必要ないため、削除します。

▶手順 11.1. グローバルデータエリアを作成するには

1 プログラムエディタのコマンド行で次のように入力して、ローカルデータエリアに戻ります。

```
E LDA01
```

2 データエリアを新しい名前で保存するには、データエリアエディタのコマンド行で次のように入力します。

```
SA GDA01
```

現在のデータエリアが新しい名前 GDA01 で保存されます。 LDA01 という名前のローカルデータエリアは、データエリアエディタに引き続き表示されます。

3 次のコマンドを入力して、GDA01 をデータエリアエディタにロードします。

```
E GDA01
```

4 ローカルデータエリアをグローバルデータエリアに変更するには、次のコマンドを入力します。

```
SET TYPE G
```

"G" はグローバルデータエリアを表します。

オブジェクトタイプが "Global" に変わります。 これは、画面の左上に示されます。

5 行コマンド .D を使用して、次のフィールドを削除します。

```
#NAME-START
#NAME-END
```

行コマンドは、削除するフィールドを含む行の **T** 列から開始して入力されています。 上記のフィールドは連続した 2 行で定義されているため、行コマンド .D(2) を使用すると、両方のフィールドを同時に削除できます。

6 Enter キーを押します。

グローバルデータエリアは次のようになります。

```
Global    GDA01     Library TUTORIAL                      DBID 11177 FNR     8
Command                                                                  > +
I T L  Name                            F Length     Miscellaneous
All -- --------------------------------- - ---------- ------------------------>
     1 #MARK                           A          1
  V  1 EMPLOYEES-VIEW                                  EMPLOYEES
     2 PERSONNEL-ID                    A          8 /* CNNNNNNN
  G  2 FULL-NAME                                       /* NAME INFORMATION
     3 NAME                            A         20 /* SURNAME/FAMILY NAME
     2 DEPT                            A          6 /* DDDDSS
  G  2 LEAVE-DATA                                      /* LEAVE/VACATION INFO
     3 LEAVE-DUE                       N        2.0 /* VACATION DAYS/YEAR
```

```
                                                                        
                                                                        
                                                                        
---------------------------------------------------------------- S 8    L 1
```

7 グローバルデータエリアを格納します。

ローカルデータエリアへの適合

グローバルデータエリアに含まれるフィールドは、ローカルデータエリアに必要ありません。したがって、#NAME-START および #NAME-END を除くすべてのフィールドをローカルデータエリアから削除します。

▶手順 11.2. フィールドを削除するには

1 データエリアエディタのコマンド行で次のように入力して、ローカルデータエリアに戻ります。

```
E LDA01
```

2 #NAME-START および #NAME-END を除くすべてのフィールドを、行コマンド .D を使用して削除します。

ビューのトップレベルのエントリ（ビュー名の前にある"V"で示される）を削除すると、このビューに属するすべてのフィールドが自動的に削除されます。

3 変更したローカルデータエリアを格納します。

ローカルデータエリアが次のように表示されます。

```
SYSGDA 4454: Data area stowed successfully.
Local     LDA01     Library TUTORIAL                      DBID 11177 FNR     8
Command                                                                 > +
I T L  Name                            F Length     Miscellaneous
All -- ------------------------------- - ---------- ------------------------>
     1 #NAME-START                     A         20
     1 #NAME-END                       A         20
```

```
------------------------------------------------------------------ S 2    L 1
```

プログラムからのグローバルデータエリアの参照

グローバルデータエリアを格納すると、Natural プログラムから参照できます。

定義したグローバルデータエリアも使用するように、プログラムの DEFINE DATA ステートメントを変更します。

手順 11.3. プログラムでグローバルデータエリアを使用するには

1 データエリアエディタのコマンド行で次のように入力して、プログラムエディタに戻ります。

```
E PGM01
```

2 LOCAL USING LDA01 の上の行に、次のように挿入します。

```
GLOBAL USING GDA01
```

グローバルデータエリアは、常にローカルデータエリアより先に定義する必要があります。そのようにしないと、エラーが発生します。

プログラムは次のようになります。

```
DEFINE DATA
  GLOBAL USING GDA01
  LOCAL USING LDA01
END-DEFINE
*
RP1. REPEAT
*
  INPUT USING MAP 'MAP01'
*
  IF #NAME-START = '.' THEN
    ESCAPE BOTTOM (RP1.)
  END-IF
*
```

```
  IF #NAME-END = ' ' THEN
    MOVE #NAME-START TO #NAME-END
  END-IF
*
  RD1. READ EMPLOYEES-VIEW BY NAME
    STARTING FROM #NAME-START
    ENDING AT #NAME-END
*
    IF LEAVE-DUE >= 20 THEN
      PERFORM MARK-SPECIAL-EMPLOYEES
    ELSE
      RESET #MARK
    END-IF
*
    DISPLAY NAME 3X DEPT 3X LEAVE-DUE 3X '>=20' #MARK
*
  END-READ
*
  IF *COUNTER (RD1.) = 0 THEN
    REINPUT 'No employees meet your criteria.'
  END-IF
*
END-REPEAT
*
DEFINE SUBROUTINE MARK-SPECIAL-EMPLOYEES
  MOVE '*' TO #MARK
END-SUBROUTINE
*
END
```

3 プログラムを実行します。

4 前（DEFINE DATA ステートメントでグローバルデータエリアを参照していない場合）と同じ結果になることを確認するため、開始名に「JONES」と入力して、Enter キーを押します。

5 プログラムエディタに戻るには、MORE プロンプトで「EDIT」と入力します。

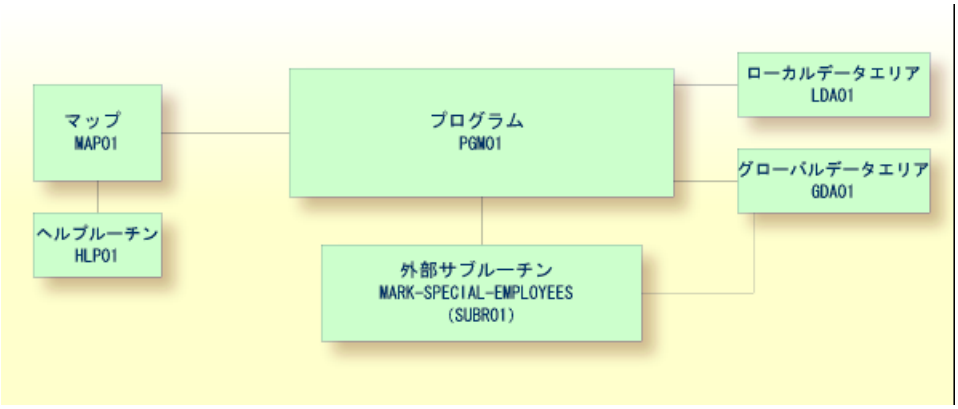
6 プログラムを格納します。

次の演習「*外部サブルーチン*」に進みます。

12 外部サブルーチン

これまでは、DEFINE SUBROUTINE ステートメントを使用して、サブルーチン MARK-SPECIAL-EMPLOYEES をプログラム内で定義していました。この演習では、サブルーチンを別のオブジェクトとしてプログラムの外部に定義します。

以下の演習を完了すると、サンプルのアプリケーションは次のモジュールで構成されます。



外部サブルーチンの作成

プログラムの既存のコードを外部サブルーチンで再使用するため、プログラムを新しい名前で保存し、タイプをサブルーチンに変更して、不要な行をすべて削除します。

外部サブルーチンの DEFINE SUBROUTINE ステートメントは、プログラム内のインラインサブルーチンと同じ方法でコーディングします。

手順 12.1. 外部サブルーチンを作成するには

1 プログラムエディタのコマンド行で、次のように入力します。

```
SA SUBR01
```

現在のプログラムが新しい名前 SUBR01 で保存されます。プログラムは引き続きエディタに表示されます。

2 次のコマンドを入力して、新しく作成したオブジェクトをエディタにロードします。

```
E SUBR01
```

オブジェクトタイプはプログラムのままです。

3 プログラムを外部サブルーチンに変更するには、次のコマンドを入力します。

```
SET TYPE S
```

"S" はサブルーチンを表します。

画面に表示されるオブジェクトタイプが "Subroutine" に変わります。

4 行コマンド .D を使用して、次の行を除くすべての行を削除します。

```
DEFINE DATA
  GLOBAL USING GDA01
  LOCAL USING LDA01
END-DEFINE
*
DEFINE SUBROUTINE MARK-SPECIAL-EMPLOYEES
  MOVE '*' TO #MARK
END-SUBROUTINE
*
END
```

テキストのブロックを削除することもできます。 これには、次の手順を実行します。

1. ブロックの最初の行の先頭で、行コマンド「.X」を入力します。
2. ブロックの最後の行の先頭で、行コマンド「.Y」を入力します。
3. Enter キーを押します。

   削除する行のブロックが "X" および "Y" でマークされます。 （マークを削除するには、コマンド行で「RESET」を入力します。）
4. マークされたブロックを削除するには、コマンド行で「DX-Y」を入力します。

5 サブルーチンを格納します。

プログラムからの外部サブルーチンの参照

PERFORM ステートメントは、内部サブルーチンおよび外部サブルーチンの両方を呼び出します。内部サブルーチンがプログラム内で見つからないと、Natural は同じ名前の外部サブルーチンを自動的に実行しようとします。 Natural では、サブルーチンコードで定義された名前（サブルーチン名）が検索され、サブルーチンの保存時に指定した名前（Natural オブジェクト名）が検索されるわけではありません。

外部サブルーチンの定義が完了したので、インラインサブルーチン（外部サブルーチンと同じ名前を持つ）をプログラムから削除する必要があります。

手順 12.2. プログラムで外部サブルーチンを使用するには

1 サブルーチンが現在表示されているエディタのコマンド行で次のように入力して、プログラムエディタに戻ります。

```
E PGM01
```

2 次の行を削除します。

```
DEFINE SUBROUTINE MARK-SPECIAL-EMPLOYEES
  MOVE '*' TO #MARK
END-SUBROUTINE
```

プログラムは次のようになります。

```
DEFINE DATA
  GLOBAL USING GDA01
  LOCAL USING LDA01
END-DEFINE
*
RP1. REPEAT
*
  INPUT USING MAP 'MAP01'
*
  IF #NAME-START = '.' THEN
    ESCAPE BOTTOM (RP1.)
  END-IF
*
  IF #NAME-END = ' ' THEN
    MOVE #NAME-START TO #NAME-END
  END-IF
*
  RD1. READ EMPLOYEES-VIEW BY NAME
    STARTING FROM #NAME-START
    ENDING AT #NAME-END
*
    IF LEAVE-DUE >= 20 THEN
      PERFORM MARK-SPECIAL-EMPLOYEES
    ELSE
      RESET #MARK
    END-IF
*
    DISPLAY NAME 3X DEPT 3X LEAVE-DUE 3X '>=20' #MARK

  END-READ
*
  IF *COUNTER (RD1.) = 0 THEN
    REINPUT 'No employees meet your criteria.'
  END-IF
*
END-REPEAT
*
END
```

3 プログラムを実行します。

4 開始名として「JONES」と入力し、Enter キーを押します。

結果のリストには、20 日以上の休暇を取った各従業員にアスタリスクが引き続き表示されます。

5 プログラムエディタに戻るには、MORE プロンプトで「EDIT」と入力します。

6 プログラムを格納します。

手順 12.3. 同等のサブルーチン名をリストするには

1 プログラムエディタのコマンド行で、次のいずれかのコマンドを入力します。

```
LIST EXTENDED SUBROUTINE *
```

```
L EXT S *
```

次の画面が表示されます。 すべての外部サブルーチンオブジェクト（メンバ）、および現在の Natural ライブラリおよびシステムファイルで使用できる同等のロングネームがリストされます。

```
12:21:09               ***** NATURAL LIST COMMAND *****              2007-03-20
User SAG                 - LIST Objects in a Library -          Library TUTORIAL

Cmd Subroutine/Class Name            Type  S/C Member    Cat Date    Cat Time
--- *________________________________ S____ --- *_______ *__________ *________
__  MARK-SPECIAL-EMPLOYEES           Subro S/C SUBR01    2007-03-20  12:11:56









                                                                  1 Objects found
Top of List.
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Print Exit  Sort        --    -     +     ++          >     Canc
```

特定の範囲のサブルーチン名を表示する場合は、ヘッダー **Subroutine/Class Name** または **Member** の下にあるフィールドで、検索値の後にアスタリスク（*）を続けて入力します。例：

```
Cmd Subroutine/Class Name            Type  S/C Member    Cat Date    Cat Time
--- MARK*____________________________ S____ --- *_______ *__________ *________
__  MARK-SPECIAL-EMPLOYEES           Subro S/C SUBR01    2007-03-20  12:11:56
```

2 PF3 キーを押して、プログラムエディタに戻ります。

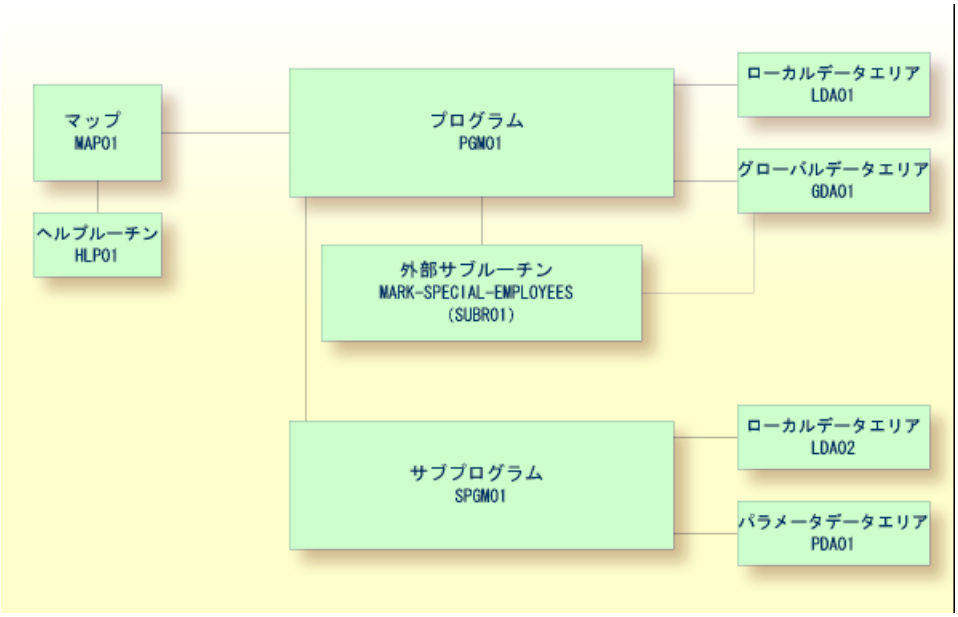
次の演習「サブプログラム」に進みます。

13 サブプログラム

サブプログラムを起動する CALLNAT ステートメントを含めるようにプログラムを拡張します。 サブプログラムでは、メインプログラムで識別された従業員が、デモデータベースの一部でもある VEHICLES ファイルに対する FIND 要求のベースとなります。 その結果、メインプログラムからの従業員情報だけでなくサブプログラムからの自動車情報も出力に含まれます。

新しいサブプログラムでは、追加のローカルデータエリアおよびパラメータデータエリアを作成する必要があります。

以下の演習を完了すると、サンプルのアプリケーションは次のモジュールで構成されます。



ローカルデータエリアの変更

前に作成したローカルデータエリアに他のフィールドを追加します。 これらのフィールドは、後で作成するサブプログラムで使用します。

手順 13.1. ローカルデータエリアに他のフィールドを追加するには

1 ローカルデータエリアに戻ります。

```
E LDA01
```

2 #NAME-END の下に次のフィールドを定義します。

レベル（［L］列）	名前	フォーマット（［F］列）	長さ
1	#PERS-ID	A	8
1	#MAKE	A	20
1	#MODEL	A	20

ローカルデータエリアが次のように表示されます。

```
Local     LDA01     Library TUTORIAL                      DBID 11177 FNR     8
Command                                                                     > +
I T L  Name                             F Length     Miscellaneous
All -- -------------------------------- - ---------- ------------------------->
     1 #NAME-START                      A         20
     1 #NAME-END                        A         20
     1 #PERS-ID                         A          8
     1 #MAKE                            A         20
     1 #MODEL                           A         20
















------------------------------------------------------------------ S 5    L 1
```

3 ローカルデータエリアを格納します。

既存のローカルデータエリアからのパラメータデータエリアの作成

パラメータデータエリア（PDA）は、Naturalプログラムおよび後で作成するサブプログラム間で受け渡すデータパラメータを指定するために使用します。 パラメータデータエリアは、サブプログラムで参照されます。

ローカルデータエリアにわずかな変更を加えるだけで、パラメータデータエリアを作成できます。つまり、ローカルデータエリアの2つのデータフィールドを削除してから、変更したデータ

エリアをパラメータデータエリアとして保存します。 元のローカルデータエリアはそのまま残ります。

手順 13.2. パラメータデータエリアを作成するには

1 ローカルデータエリアで、フィールド #NAME-START および #NAME-END を削除します。

2 データエリアエディタのコマンド行で、次のように入力します。

```
SA PDA01
```

現在のデータエリアが新しい名前PDA01で保存されます。既存のローカルデータエリアは、引き続きエディタに表示されます。

3 次のコマンドを入力して、新しく作成したデータエリアをエディタにロードします。

```
E PDA01
```

4 ローカルデータエリアをパラメータデータエリアに変更するには、次のコマンドを入力します。

```
SET TYPE A
```

"A" はパラメータデータエリアを表します。

オブジェクトタイプが "Parameter" に変わります。 これは、画面の左上に示されます。 パラメータデータエリアが次のように表示されます。

```
Parameter PDA01     Library TUTORIAL                      DBID 11177 FNR     8
Command                                                                  > +
I T L  Name                            F Length     Miscellaneous
All -- --------------------------------- - ---------- ------------------------>
     1 #PERS-ID                        A          8
     1 #MAKE                           A         20
     1 #MODEL                          A         20
                                                                    S 3    L 1
```

5 パラメータデータエリアを格納します。

異なるビューを含む別のローカルデータエリアの作成

2 つ目のローカルデータエリアを作成し、VEHICLES データベースファイルの DDM からフィールドをインポートします。

このローカルデータエリアは、サブプログラムで参照されます。

手順 13.3. ローカルデータエリアを作成するには

1 データエリアエディタのコマンド行で、次のコマンドを入力します。

```
CLEAR
```

データエリアエディタが空になります。

2 データエリアのタイプを変更するには、コマンド行で次のように入力します。

```
SET TYPE L
```

"L" はローカルデータエリアを表します。

3 編集エリアの最初の行で、**T** 列から開始して次のように入力します。

```
.V(VEHICLES)
```

4 Enter キーを押します。

VEHICLES ビューが表示されます。

```
SYSGDA 4461: Mark fields to incorporate into data area.
Local               Library TUTORIAL                      DBID 11177 FNR     8
View VEHICLES
I T L  Name                            F Length     Miscellaneous
- - -- ------------------------------- - ---------- -------------------------
     2 REG-NUM                         A         15 /* CAR'S REGISTR. NUMBE
     2 CHASSIS-NUM                     I          4 /* MANUFACTURER NUMBER
     2 PERSONNEL-ID                    A          8 /* IDENT. OF CAR USER
  G  2 CAR-DETAILS                                   /* DESCRIPTION OF THE C
     3 MAKE                            A         20
     3 MODEL                           A         20
     3 COLOR                           A         10
     3 COLOUR                          A         10
     2 YEAR                            N        4.0 /* MANUFACTURING YEAR
     2 CLASS                           A          1 /* P=PRIVAT
     2 LEASE-PUR                       A          1 /* L=LEASED
     2 DATE-ACQ                        N        8.0 /* DATE THE CAR WAS ACQ
```

```
     2 CURR-CODE                          A          3 /* CURRENCY OF CAR COST
  M  2 MAINT-COST                         P        7.0 (1:60)/* MAINTENANCE COST
     2 MODEL-YEAR-MAKE                    A         24 /* YEAR + CAR MAKE /* SP



- - -- -------------------------------- - ---------- -------------------------
```

5 **I** 列に任意の文字を入力して、次のフィールドをマークします。

```
PERSONNEL-ID
CAR-DETAILS
MAKE
MODEL
```

6 必須フィールドをすべてマークしたら、データエリアエディタに戻るため Enter キーを押します。

ローカルデータエリアが次のように表示されます。

```
Local               Library TUTORIAL                      DBID 11177 FNR     8
Command                                                                   > +
I T L  Name                             F Length     Miscellaneous
All -- -------------------------------- - ---------- -------------------------->
  V  1 VEHICLES-VIEW                                  VEHICLES
     2 PERSONNEL-ID                     A          8 /* IDENT. OF CAR USER
  G  2 CAR-DETAILS                                    /* DESCRIPTION OF THE CAR
     3 MAKE                             A         20
     3 MODEL                            A         20
                                                                  S 5    L 1
```

7 コマンド行で次のように入力して、新しいローカルデータエリアを保存します。

```
SA LDA02
```

8 新しいローカルデータエリアを格納します。

サブプログラムの作成

パラメータデータエリアおよびローカルデータエリアを使用して VEHICLES ファイルから情報を取得するサブプログラムを作成します。 サブプログラムではプログラム PGM01 から渡された職員 ID を受け取り、VEHICLES ファイルの検索ベースとしてこの ID を使用します。

手順 13.4. サブプログラムを作成するには

1 データエリアエディタのコマンド行で、次のコマンドを入力します。

```
E N
```

"N" はサブプログラムを表します。

空のプログラムエディタが起動されます。 オブジェクトタイプはサブプログラムに設定されています。

2 次のように入力します。

```
DEFINE DATA
  PARAMETER USING PDA01
  LOCAL USING LDA02
END-DEFINE
*
FD1. FIND (1) VEHICLES-VIEW
  WITH PERSONNEL-ID = #PERS-ID
  MOVE MAKE (FD1.) TO #MAKE
  MOVE MODEL (FD1.) TO #MODEL
  ESCAPE BOTTOM
END-FIND
*
END
```

このサブプログラムから、従業員の社用車の車種およびモデルが特定の職員 ID に対して返されます。

FIND ステートメントは、検索条件 #PERS-ID に基づいて、データベースから一連のレコード（ここでは 1 レコード）を選択します。

フィールド #PERS-ID には、プログラム PGM01 から渡された PERSONNEL-ID の値が入ります。サブプログラムでは、VEHICLES ファイルの検索ベースとしてこの値が使用されます。

3 サブプログラムを格納します。

```
STOW SPGM01
```

プログラムからのサブプログラムの参照

サブプログラムは、CALLNAT ステートメントを使用してメインプログラムから呼び出されます。サブプログラムは、CALLNAT ステートメントによってのみ呼び出すことができます。単独で実行することはできません。 サブプログラムには、呼び出し側オブジェクトが使用するグローバルデータエリアへのアクセスがありません。

メインプログラムから指定のサブプログラムへは、サブプログラムの DEFINE DATA PARAMETER ステートメントで参照される一連のパラメータを介してデータが渡されます。

サブプログラムのパラメータデータエリアで定義される変数は、CALLNAT ステートメントの変数と同じ名前でなくてもかまいません。 パラメータはアドレスによって渡されるため、順序、フォーマット、および長さが一致していれば十分です。

定義したサブプログラムが使用されるように、メインプログラムを変更します。

手順 13.5. メインプログラムでサブプログラムを使用するには

1 コマンド行で次のように入力して、プログラムエディタに戻ります。

```
E PGM01
```

2 DISPLAY ステートメントの直前に、次のように挿入します。

```
RESET #MAKE #MODEL
CALLNAT 'SPGM01' PERSONNEL-ID #MAKE #MODEL
```

RESET ステートメントにより、#MAKE および #MODEL の値が空値に設定されます。

3 DISPLAY ステートメントを含む行を削除し、次のように置き換えます。

```
WRITE TITLE
      / '*** PERSONS WITH 20 OR MORE DAYS LEAVE DUE ***'
      / '***     ARE MARKED WITH AN ASTERISK        ***'//
*
DISPLAY   1X '//N A M E' NAME
          1X '//DEPT'    DEPT
          1X '/LV/DUE'   LEAVE-DUE
             ' '         #MARK
          1X '//MAKE'    #MAKE
          1X '//MODEL'   #MODEL
```

WRITE TITLE ステートメントで定義されたテキストが、各出力ページの先頭に表示されます。WRITE TITLE ステートメントは、デフォルトのページタイトルを無効にします。各ページの先頭にこれまで表示されていた情報（ページ番号、日時）は表示されなくなります。各スラッシュ（/）により、後続の情報が新しい行に表示されます。

サブプログラムから追加の自動車情報が返されるため、出力列のサイズを変更する必要があります。 ヘッダーは短くなります。 アスタリスクを表示する列（#MARK）には、ヘッダーは表示されません。列間には1文字分の空白が挿入されます（1X）。ヘッダー内の各スラッシュにより、後続の情報が同じ列の新しい行に表示されます。

プログラムは次のようになります。

```
DEFINE DATA
  GLOBAL USING GDA01
  LOCAL USING LDA01
END-DEFINE
*
RP1. REPEAT
*
  INPUT USING MAP 'MAP01'
*
  IF #NAME-START = '.' THEN
    ESCAPE BOTTOM (RP1.)
  END-IF
*
  IF #NAME-END = ' ' THEN
    MOVE #NAME-START TO #NAME-END
  END-IF
*
  RD1. READ EMPLOYEES-VIEW BY NAME
    STARTING FROM #NAME-START
    ENDING AT #NAME-END
*
    IF LEAVE-DUE >= 20 THEN
      PERFORM MARK-SPECIAL-EMPLOYEES
    ELSE
      RESET #MARK
    END-IF
*
    RESET #MAKE #MODEL
    CALLNAT 'SPGM01' PERSONNEL-ID #MAKE #MODEL
*
    WRITE TITLE
      / '*** PERSONS WITH 20 OR MORE DAYS LEAVE DUE ***'
      / '***     ARE MARKED WITH AN ASTERISK        ***'//
*
    DISPLAY   1X '//N A M E' NAME
              1X '//DEPT'    DEPT
              1X '/LV/DUE'   LEAVE-DUE
                 ' '         #MARK
              1X '//MAKE'    #MAKE
              1X '//MODEL'   #MODEL
*
  END-READ
*
  IF *COUNTER (RD1.) = 0 THEN
```

```
    REINPUT 'No employees meet your criteria.'
  END-IF
*
END-REPEAT
*
END
```

4 プログラムを実行します。

5 開始名として「JONES」と入力し、Enter キーを押します。

結果のリストは次のようになります。

```
MORE

                *** PERSONS WITH 20 OR MORE DAYS LEAVE DUE ***
                ***     ARE MARKED WITH AN ASTERISK        ***



                            LV
       N A M E         DEPT  DUE           MAKE                 MODEL
 -------------------- ------ --- - -------------------- --------------------

 JONES                SALE30  25 * CHRYSLER             IMPERIAL
 JONES                MGMT10  34 * CHRYSLER             PLYMOUTH
 JONES                TECH10  11   GENERAL MOTORS       CHEVROLET
 JONES                MGMT10  18   FORD                 ESCORT
 JONES                TECH10  21 * GENERAL MOTORS       BUICK
 JONES                SALE00  30 * GENERAL MOTORS       PONTIAC
 JONES                SALE20  14   GENERAL MOTORS       OLDSMOBILE
 JONES                COMP12  26 * DATSUN               SUNNY
 JONES                TECH02  25 * FORD                 ESCORT 1.3
```

6 プログラムエディタに戻るには、MORE プロンプトで「EDIT」と入力します。

7 プログラムを格納します。

これで、チュートリアルがすべて完了しました。

索引

N

ち