Natural for UNIX

ファーストステップ

バージョン 8.4.1

2017 年 10 月

ADABAS & NATURAL

このマニュアルは Natural バージョン 8.4.1 およびそれ以降のすべてのリリースに適用されます。

このマニュアルに記載される仕様は変更される可能性があります。変更は以降のリリースノートまたは新しいマニュアルに記述されます。

ドキュメント IDは: NATUX-NNATFIRSTSTEPS-841-20200614JA

目次

前書き

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。

1

表記規則

規則	説明
太字	画面上の要素を表します。
モノスペースフォント	*folder.subfolder:service* という規則を使用して webMethods Integration Server 上のサービスの保存場所を表します。
大文字	キーボードのキーを表します。同時に押す必要があるキーは、プラス記号 (+) で結んで表記されます。
斜体	独自の状況または環境に固有の値を指定する必要がある変数を表します。本文で最初に出現する新しい用語を表します。
モノスペースフォント	入力する必要があるテキストまたはシステムから表示されるメッセージを表します。Program code.
{ }	選択肢のセットを表します。ここから1つ選択する必要があります。中カッコの内側にある情報のみを入力します。{ } 記号は入力しません。
|	構文行で相互排他的な2つの選択肢を区切ります。いずれかの選択肢を入力します。| 記号は入力しません。
[ ]	1 つ以上のオプションを表します。大カッコの内側にある情報のみを入力します。[ ] 記号は入力しません。
...	同じ種類の情報を複数回入力できることを示します。情報だけを入力してください。実際のコードに繰り返し記号 (...) を入力しないでください。

オンライン情報

Software AG マニュアルの **Web** サイト

マニュアルは、Software AG マニュアルの Web サイト (**http://documentation.softwareag.com**) で入手できます。このサイトでは Empower クレデンシャルが必要です。Empower クレデンシャルがない場合は、TECHcommunity Web サイトを使用する必要があります。

Software AG Empower 製品のサポート **Web** サイト

もしまだEmpowerのアカウントをお持ちでないのなら、こちらへempower@softwareag.com電子メールにて あなたのお名前、会社名、会社の電子メールアドレスをお書きの上、アカウントを請求してください。

いったんアカウントをお持ちになれば、Empower **https://empower.softwareag.com/** のeServiceセクションにてサポートインシデントをオンラインで開くことができます。

製品情報は、Software AG Empower 製品のサポート Web サイト (**https://empower.softwareag.com**) で入手できます。

機能および拡張機能に関するリクエストの送信、製品の可用性に関する情報の取得、製品のダウンロードを実行するには、Products に移動します。

修正に関する情報を取得し、早期警告、技術論文、Knowledge Base の記事を読むには、**Knowledge Center** に移動します。

もしご質問があれは、こちらの**https://empower.softwareag.com/public_directory.asp** グローバルサポート連絡一覧の、あなたの国の電話番号を選んで、わたくし共へご連絡ください。

Software AG TECHcommunity

マニュアルおよびその他の技術情報は、Software AG TECHcommunity Web サイト (**http://techcommunity.softwareag.com**) で入手できます。以下の操作を実行できます。

- TECHcommunity クレデンシャルを持っている場合は、製品マニュアルにアクセスできます。TECHcommunity クレデンシャルがない場合は、登録し、関心事の領域として [マニュアル] を指定する必要があります。
- 記事、コードサンプル、デモ、チュートリアルにアクセスする。
- Software AGの専門家によって承認されたオンライン掲示板フォーラムを使用して、質問したり、ベストプラクティスを話し合ったり、他の顧客が Software AG のテクノロジをどのように使用しているかを学んだりすることが可能です。
- オープンスタンダードや Web テクノロジを取り扱う外部 Web サイトにリンクできます。

データ保護

Software AG製品は、EU一般データ保護規則(GDPR)を尊重した個人データの処理機能を提供します。 該当する場合、適切な手順がそれぞれの管理ドキュメントに記載されています。

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。

必要条件

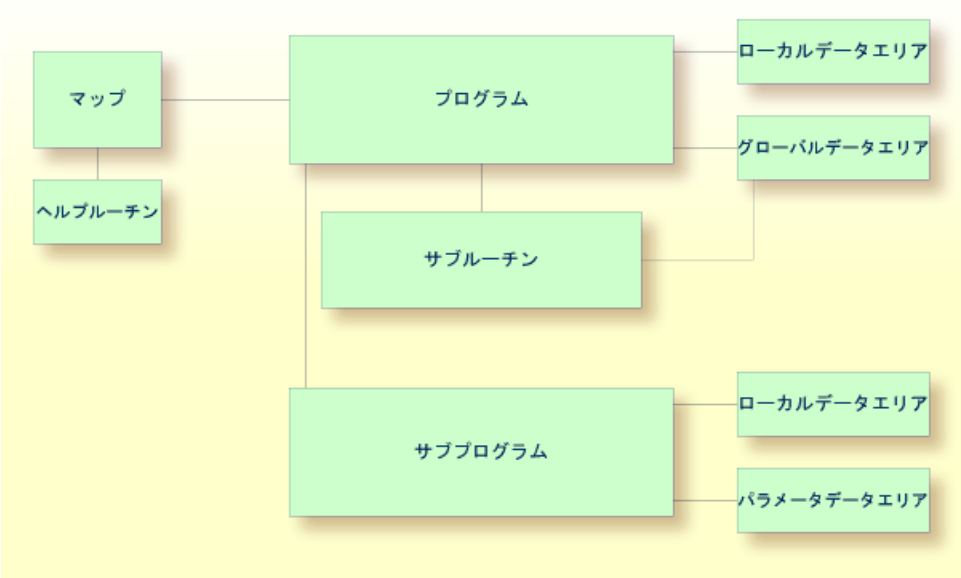
このチュートリアルのすべてのステップを実行するには、デモデータベース SAG-DEMO-DB がアクティブになっている必要があります。アクティブでない場合は、起動するよう管理者に依頼してください。

このチュートリアルで使用するサンプル DDM（EMPLOYEES および VEHICLES）が含まれているシステムライブラリ SYSEXDDM を steplib として定義しておく必要があります。steplib は、オブジェクトが現在のライブラリで見つからない場合にNaturalが検索するライブラリです。SYSEXDDMがsteplibとして定義されていないと、サンプルDDMのビューを定義しようとするときにエラーが発生します。その場合は、管理者に連絡してください。

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Hello World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。



注意: このチュートリアルでは、一般に文字型環境（メインフレームなど）で使用されるマップの作成方法について説明します。グラフィカルユーザーインターフェイスの場合は、ダイアログを作成します。ただし、これはこのチュートリアルの範囲外です。

最初の演習に進みます。「*Natural の基本*」

3 Natural の基本

Natural のメインメニューの呼び出し

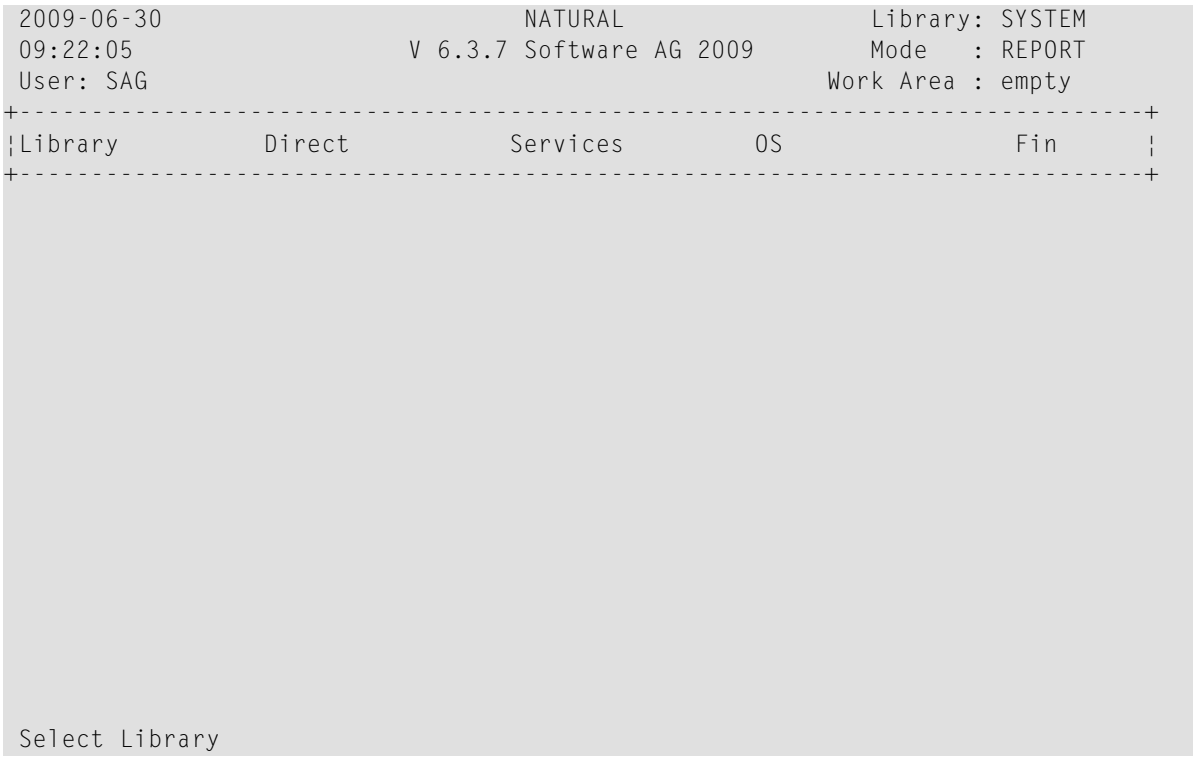
Natural およびそのメインメニューの呼び出し方法は、サイトにおけるシステムの構成方法によって異なります。ほとんどのインストールでは、以下のように Natural を呼び出します。

手順 3.1. Natural のメインメニューを起動するには

- UNIX システムのプロンプトで次のコマンドを入力します。

```
natural
```

メインメニューが表示されます。



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 の機能は、一連のメニューおよび選択ウィンドウから機能を選択するか、Natural システムコマンドを直接入力することによって実行できます。

メニューを選択するには、方向キーを使用します。必要なメニューが強調表示されたら、ENTER キーを押します。また、メニュー名の最初の文字を入力することもできます。この場合、ENTER キーを押す必要はありません。

結果のウィンドウにオプションのリストが表示されたら、方向キーを使用して必要なオプションを選択し、ENTER キーを押します。一部のウィンドウでは、機能の最初の文字を入力することもできます。この場合、ENTER キーを押す必要はありません。

以降の操作を行わずにウィンドウを閉じるには、ESC キーを押します。

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 キーを押すと、操作が確認されてコマンドが実行されるか、またはコマンドの実行を明示的に確認する追加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。

ライブラリは、異なる方法で作成できます。次の手順は、Naturalシステムコマンドを入力する［**Direct**］メニューの使用方法、および［**Library**］メニューを使用して同じ結果を得る方法を示します。

手順 3.2. ［Direct］メニューを使用してユーザーライブラリを作成するには

■ ［**Direct**］メニューを選択し、ENTER キーを押します。結果の［**Direct Command**］ウィンドウで次のように入力し、ENTER キーを押します。

```
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。

手順 3.3. ［Library］メニューを使用してユーザーライブラリを作成するには

1 ［**Library**］メニューを選択し、ENTER キーを押します。

既存のすべてのライブラリにアクセスできるウィンドウが表示されます。

```
 2009-06-30                          NATURAL                  Library: SYSTEM
 11:16:09                  V 6.3.7 Software AG 2009           Mode   : REPORT
 User: SAG                                                  Work Area : empty
+-------------------------------------------------------------------------------+
¦Library          Direct           Services         OS                Fin      ¦
+-------------------------------------------------------------------------------+
+----------+
¦ <LOGON>  ¦
¦ AA       ¦
¦ ABC      ¦
¦ ADR      ¦
¦ BBC      ¦
¦ BTX      ¦
¦ BZG      ¦
¦ CEC1     ¦
¦ CEC2     ¦
¦ CEC3     ¦
¦ CMAS     ¦
¦ CT       ¦
¦ CUST     ¦
¦ DAC      ¦
+----------+

 Select Library
```

2 方向キーを使用して最初のエントリである［**<LOGON>**］を選択し、ENTER キーを押します。

次のウィンドウが表示されます。既存のライブラリにログオンするか、または新しいライブラリを追加することができます。

```
 +-- New Library: --+
 ¦                  ¦
 +------------------+
```

3 "TUTORIAL" の名前を指定します。

ライブラリ名が8文字の場合は、ENTERキーを押す必要はありません。8文字である場合は、名前の最後の文字を入力すると、ライブラリに自動的にログオンします。

新しいライブラリにはオブジェクトがまだ存在していないため、ライブラリが現在空であることを示すメッセージが表示されます。

注意:

1. 既存のライブラリにログオンする場合にも、方向キーを使用してウィンドウ内のすべてのライブラリをスクロールできます。または、文字を入力して、その文字で始まるライブラリに進むこともできます。ENTERキーを押して、強調表示されたライブラリにログオンします。

2. オブジェクトがすでに存在するライブラリにログオンすると、そのライブラリのオブジェクトのリストが表示されます。前の演習で説明した［**Direct Command**］ウィンドウを使用してログオンした場合には、このリストは表示されません。

プログラミングモード

プログラミングモードは、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

- （［**Direct**］メニューを選択して）［**Direct Command**］ウィンドウを表示し、次のシステムコマンドを入力して ENTER キーを押します。

```
GLOBALS SM=ON
```

最初のプログラムに進みます。*Hello World!*

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 ［**Direct Command**］ウィンドウを表示し、次のいずれかを入力します。

```
EDIT PROGRAM
```

```
E P
```

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

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。

```
>> -----------Columns 001 072 <<  Program          Lines         User SAG
Command ===>                                      Mode   Struct Lib  TUTORIAL
****** ****************************** top of data ******************************
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
****** *************************** bottom of data ***************************
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Save  Exit  Run   Rfind Stow  -     +     Check Home  Undo  Canc
```

注意: まだ ENTER キーを押さないでください。ENTER キーを押すと、挿入モードが終了します。コードを入力するには、次の手順で挿入モードに戻る必要があります。"top of data" という語句を含む行の先頭で行コマンド「I」を入力し、ENTER キーを押します。

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
```

コードの最終行を追加した後で ENTER キーを 2 回押すと、挿入モードがオフになります。

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. では、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
```

注意: プログラムエディタで PF10（Home）キーを押して、コマンド行にカーソルを置くことができます。

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                                                13-05-16  13:27:42


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。

```
>> -----------Columns 001 072 <<  Program          Lines  4      User SAG
Command ===> run                                   Mode   Struct Lib  TUTORIAL
****** ****************************** top of data ******************************
000010 * The "Hello world!" example in Natural.
000020 *
000030 DISPLAY "Hello world!
000040 END /* End of program
+-------------------------- Error in File:          --------------------------+
¦NAT0305 Text string must begin and end on the same line.                     ¦
¦             VVVVVVVVVVVVVV                                                  ¦
¦0030 DISPLAY "Hello world!                                                   ¦
+-------------------------------------------------------------------------------+









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Save  Exit  Run   Rfind Stow  -     +     Check Home  Undo  Canc
```

3 エラーメッセージが表示されたウィンドウで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 ENTER キーを押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には（作業を翌日まで停止する場合など）、システムコマンド SAVE を使用できます。プログラムを初めて保存する場合は、名前も指定する必要があります。例えば、次のようになります。SAVE HELLO

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 HELLO
```

"HELLO" は格納するプログラムの名前です。

注意: プログラムにすでに名前が付いている場合は、（プログラム名を指定せずに）コマンド行で「STOW」と入力するか、PF6 キーを押すだけで十分です。

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。

```
>> -----------Columns 001 072 <<  Program HELLO    Lines  4      User SAG
Command ===> list dir hello                        Mode   Struct Lib  TUTORIAL
****** ****************************** top of data ******************************
+---------------------------- List Directory HELLO -----------------------------+
¦ Directory of Program HELLO                  Saved on ... 2009-06-30 16:37:00 ¦
¦ ------------------------------------------------------------------------------ ¦
¦ Library .... TUTORIAL   User-ID ...... SAG       Mode .. Structured           ¦
¦ OP-System .. SUN_SOLA                                                         ¦
¦ NAT-Ver .... V 6.3.7                                                          ¦
¦ Size ............................... 108 Bytes                                ¦
¦                                                                               ¦
¦ Directory of Program HELLO                  Cataloged on 2009-06-30 16:37:00 ¦
¦ ------------------------------------------------------------------------------ ¦
¦ Library .... TUTORIAL   User-ID ...... SAG       Mode .. Structured           ¦
¦ OP-System .. SunOS                          OP-Version ..5.8Generic_10852     ¦
¦ NAT-Ver .... V 6.3.7                                                          ¦
¦ Size ............................... 330 Bytes                                ¦
¦ Endian mode: Big                                                              ¦
¦                                                          ENTER to continue    ¦
+---------------------------------------------------------------------------------+

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Save  Exit  Run   Rfind Stow  -     +     Check Home  Undo  Canc
```

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 *
```

次のウィンドウが表示されます。作成したプログラムがリストされます。

```
>> -----------Columns 001 072 <<  Program HELLO    Lines  4      User SAG
Command ===> l *                                    Mode   Struct Lib  TUTORIAL
****** ***************************** top of data ******************************
000010 * The "Hello world!" example in Natural.
000020 *
000030 DISPLAY "Hello world!"
0+---------------------------------- List * * ----------------------------------+
*¦ Cmd Name     Type        SM S/C Userid   SRC Date         GP Date           ¦
 ¦ --- -------- ----------- -- --- -------- ---------------- ----------------- ¦
 ¦ <DIRECT COMMAND>                                                           ¦
 ¦     HELLO    Program     S  S/C SAG      16:37 2009-06-30 16:37 2009-06-30 ¦
 ¦                                                                            ¦
 ¦                                                                            ¦
 ¦                                                                            ¦
 ¦                                                                            ¦
 ¦                                                                            ¦
 ¦                                                                            ¦
 ¦                                                                            ¦
 ¦                                                                            ¦
 ¦                                                                            ¦
 ¦                                                                            ¦
E¦                                                                            ¦
 +----------------------------------------------------------------------------+
```

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。

```
+---------------------------+
¦ C Check                   ¦
¦ D Read                    ¦
¦ E Edit                    ¦
¦ L List                    ¦
¦ I List Dir                ¦
¦ H Hardcopy                ¦
¦ R Run                     ¦
¦ X Execute                 ¦
¦ S Stow                    ¦
¦ U Scratch                 ¦
¦ . End                     ¦
+---------------------------+
```

注意: ［**Scratch**］を使用してオブジェクトを削除します。

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

4 プログラムエディタに戻るには、ESC 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
```

エディタプロファイルのメインメニューが表示されます。

```
13:57:44         **** Program Editor Profile ****                    09.06.30
                            Main Menu

  Profile Name ... SYSTEM


        _  Save

        _  Modify

        _  Read

        _  Technical Info









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

ユーザー固有のエディタプロファイルがない場合、デフォルトプロファイル SYSTEM が表示されます。このデフォルト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 ［**Modify**］オプションをマークし、ENTER キーを押します。

または:

PF4 キーを押します。

次の画面が表示されます。

```
14:05:36          **** Program Editor Profile ****                  09.06.30
                         Modify Defaults

  User ID ........ SYSTEM


        _  PA/PF-Keys

        _  Commands

        _  Find

        _  General









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

3 ［**Commands**］オプションをマークし、ENTER キーを押します。

次の画面が表示されます。

```
14:04:39             **** Program Editor Profile ****                 09.06.30
                          Modify Editor Defaults

User ID ....... SYSTEM


aorder ............. OFF           hex ................ OFF
autosave ........... OFF           justify ............ LEFT
caps ............... OFF           limit .............. OFF
cols ............... OFF           log ................ OFF
decimal character ... .           mask line .......... OFF
empty .............. OFF           message line ....... ON
escape ............. OFF           mso ................ ON
escape character .... .           scroll mode ........ PAGE
fix ................ OFF           tabs ............... OFF
fixlen ............. 48            tabulator character . ^




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

次のオプションの設定をチェックします。

- **caps**
  データを大文字に変換するかどうかを指定します。
- **empty**
  スペース文字のみを含む行を自動的に削除するかどうかを指定します。
- **escape**
  行コマンドの前にエスケープ文字を使用するかどうかを指定します。

このチュートリアルでは、上記のオプションが "OFF" に設定されているとみなします。

4 上記のいずれかのオプションが現在 "ON" に定義されている場合は、 "OFF" で上書きします。

5 PF3 キーを押すか、コマンド行で「EXIT」と入力します。

6 エディタプロファイルのメインメニューが再表示されるまで、PF3キーを繰り返し押します。

7 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。

8 ［**Save**］オプションをマークし、ENTER キーを押します。

または:

PF5 キーを押します。

9 PF3 キーを押します。

または:

コマンド行で「EXIT」と入力します。

プログラムエディタが再表示されます。新しい設定がプログラムエディタで使用されるようになります。

次の演習に進みます。データベースへのアクセス

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：

```
>> -----------Columns 001 072 <<  Program PGM01    Lines  4      User SAG
Command ===>                                        Mode   Struct Lib  TUTORIAL
****** ***************************** top of data ******************************
D00010 * The "Hello world!" example in Natural.
D00020 *
D00030 DISPLAY "Hello world!"
D00040 END /* End of program
****** **************************** bottom of data ****************************
```

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。

```
D4
```

コマンドの後にある番号は、削除する行数を表します。

または:

コマンド行で次の行コマンドを入力し、削除する最初の行にカーソルを移動して、ENTER キーを押します。

```
:D4
```

エディタ画面のコマンド行で行コマンドを入力することもできます。この場合、コマンドの前にコロン（:）を付ける必要があります。カーソルでマークされた行に常に適用されます。

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。

前のHello World プログラムから行をすべて削除したため、プログラムエディタは現時点では次のようになっています。

```
>> -----------Columns 001 072 <<  Program PGM01    Lines          User SAG
Command ===>                                       Mode    Struct Lib  TUTORIAL
****** ****************************** top of data ******************************
****** ***************************** bottom of data *****************************
```

プログラムのコードを入力する前に、空行を挿入する必要があります。

手順 5.2. 空行を挿入するには

■ 「top of data」という語句を含む行の先頭で次の行コマンドを入力し、ENTER キーを押します。

```
I
```

これにより、1行の空行が挿入され、エディタが挿入モードに切り替わります。これで、プログラムコードを入力できるようになります（以下を参照）。ENTER キーを押すと、新しい行が挿入されます。新しく挿入した行（行番号の代わりにアポストロフィがいくつか表示されます）にデータを入力せずに ENTER キーを押すと、エディタの挿入モードが終了し空行が削除されます。

ヒント: （以下に示すように）LOCAL の下に空行を挿入するには、この行でアスタリスク（*）を入力します。（プログラムの最終行を入力してから ENTER キーを 2 回押した後で）挿入モードがアクティブでなくなると、アスタリスクを削除できます。この場合、空行は削除されません。

手順 5.3. 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 5.4. 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。

```
>> -----------Columns 001 072  <<  Program PGM01    Lines  6      User SAG
Command ===>                                        Mode   Struct Lib  TUTORIAL
****** ***************************** top of data ******************************
000010 DEFINE DATA
000020 LOCAL
000030
000040 END-DEFINE
000050 *
000060 END
****** **************************** bottom of data ****************************

>> -----------Columns 001 072  <<     View EMPLOYEE Lines  38     User SAG
Command ===>                                                    Lib  SYSTEM
****** ***************************** top of data ******************************
000001 DB: 020 FILE: 014  - EMPLOYEES                        DEFAULT SEQUENCE:
000002   1 AA PERSONNEL-ID                      A    8    D
000003 G 1 AB FULL-NAME
000004   2 AC FIRST-NAME                        A   20  N
000005   2 AD MIDDLE-I                          A    1  N
000006   2 AE NAME                              A   20    D
000007   1 AD MIDDLE-NAME                       A   20  N
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Save  Exit  Run   Rfind Stow  -     +     Check Home  Undo  Canc
```

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、次のコマンドまたはキーを使用します。

コマンドまたはキー	説明
SWAP	編集エリアのコマンド行から表示エリア（ビュー）のコマンド行、またはその逆にカーソルを移動します。また、目的のコマンド行にカーソルを単に移動することもできます。
+ または PF8	ビューの次のページに進みます。カーソルを表示エリア（ビュー）のコマンド行に置く必要があります。
- または PF7	ビューの前のページに進みます。カーソルを表示エリア（ビュー）のコマンド行に置く必要があります。
SPLIT END	画面分割モードを終了します。カーソルを編集エリア（プログラム）のコマンド行に置く必要があります。

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。

```
I9
```

9 行の空行が挿入されます。

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。ただし、カーソルが置かれている 1 行の空行は残ります（エディタは挿入モードのままになります）。ENTER キーを押すと、挿入モードが終了します。

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: トランザクションが中止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 5.5. 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                                                  09-06-30  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 5.6. 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。

Command	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	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。

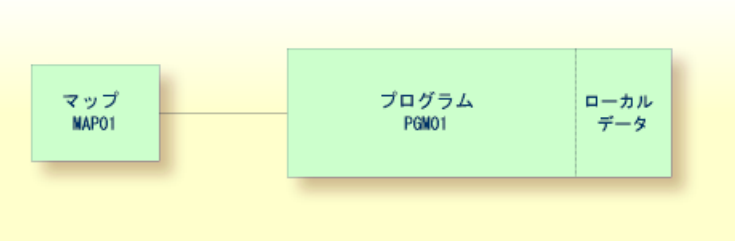
4 プログラムを格納します。

次の演習に進みます。*ユーザー入力*

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 の下に以下を挿入します。

```
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を表します（つまり、"ADKINSON" および "BENNETT"）が入力フィールドに表示されます。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）です。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が入力できます。

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。

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。

```
XXXXXXXXXX                                                          TT:TT:TT


                          Start XXXXXXXXXXXXXXXXXXX

                            End XXXXXXXXXXXXXXXXXXX
```

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。

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。

手順 6.2. マップを作成するには

■ 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。

```
EDIT MAP
```

```
E M
```

［**Natural Map Editor**］メニューが表示されます。

```
+-----------------------------------------------------------------------------+
¦                    NATURAL MAP EDITOR (Esc to select field)                 ¦
¦Create   Modify    Erase     Drag      Info OFF  Lines     Ops. Map  Quit    ¦
+-----------------------------------------------------------------------------+






















 Create a new field definition
```

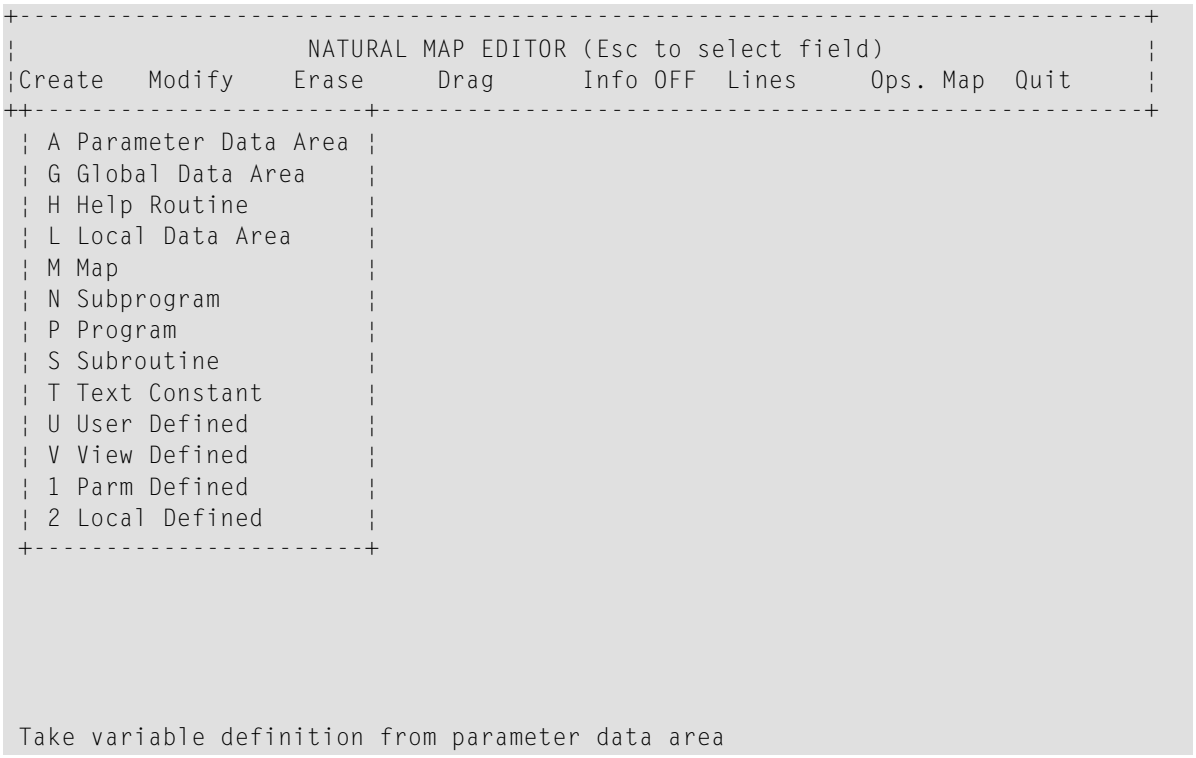
テキストフィールドの定義

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 6.3. テキストフィールドを定義するには

1 ［**Natural Map Editor**］メニューで［**Create**］を選択し、ENTER キーを押します。

作成可能なすべての項目をリストするウィンドウが表示されます。



2 方向キーを使用して［**Text Constant**］を選択し、ENTER キーを押します。

または:

T キーを押します。

空の画面が表示されます。カーソルを置いてテキストを入力するよう指示するメッセージが表示されます。

3 4行目の最初の位置（行および列番号は画面下部に表示されます）にカーソルを移動し、次のように入力します。

```
Start
```

まだ ENTER キーを押さないでください。

4 PF2 キーを押します。

このテキストフィールドに属性を設定できるウィンドウが表示されます。

```
Start
+-Text Attribute/Color Definition: --------------+
¦ Attribute: <No Attribute> Color: <No Color>    ¦
+-------------------------------------------------+
 MAP0083: Use cursor keys (UP, DOWN) for Attribute. (Esc=Cancel, Enter=OK)
```

［**Attribute**］フィールドが自動的に選択されます。

注意: LEFT-ARROW および RIGHT-ARROW を使用して、フィールド属性とカラー定義との間を切り替えることができます。このチュートリアルでは色は使用しません。

5 UP-ARROW または DOWN-ARROW を使用して、使用可能な属性をスクロールできます。ウィンドウに "Default" と表示されたら、ENTER キーを押します。"Default" は、フィールドが高輝度表示でなく強調表示でもないことを示します。

これで、フィールドが画面で選択されました。このことは、強調表示された背景によって示されます。

6 ESC キーを押すと、メニューが再表示されます。

ESC キーは、メニューの表示と非表示を切り替えるために使用されます。定義したテキストフィールドは表示されなくなります。削除されたのではなく、メニューの後ろに消えるだけです。ESC キーをもう一度押すと、メニューが非表示になりフィールドが再表示されます。

7 最初のテキストフィールドの追加と同じ手順で、2 つ目のテキストフィールドを追加（2 つ目のテキスト定数を定義し、［**Default**］属性を設定）します。［**Start**］フィールドの下にある行（行 5）にこのフィールドを配置し、次のように名前を付けます。

```
End
```

8 2 つ目のテキストフィールドを追加したら、ESC キーを押してメニューを再表示します。

新しい［**End**］フィールドがメニューの真下に表示されます。

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。

手順 6.4. データフィールドを定義するには

1 ［**Create**］メニューで、［**User Defined**］を選択します。

［**Extended Field Editing**］ウィンドウが表示されます。

```
Start
End
+-Extended Field Editing-----------------------------------------------------+
¦Field :                                                                     ¦
¦Format: A Len:            AL:             PM:                  ZP: N  SG: N ¦
¦Rules : O Rule Editing: N Array:          Array Editing: N     Mode:        ¦
¦AD:            CD:        CV:                                  DY: N  HE: N ¦
¦EM:                                                                         ¦
+-----------------------------------------------------------------------------+

 MAP0079: Position cursor and press Enter or format char.(Esc=Cancel) (012,019)
```

2 ［**Start**］フィールドの後にカーソルを置き（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。

次のウィンドウが表示されます。

```
+--------------------+
¦ A Alphanumeric     ¦
¦ B Binary           ¦
¦ D Date             ¦
¦ F Floating Point   ¦
¦ I Integer          ¦
¦ L Logical          ¦
¦ N Numeric          ¦
¦ P Packed Numeric   ¦
¦ T Time             ¦
¦ * System           ¦
+--------------------+
```

3 ［**A Alphanumeric**］を選択します。

選択したフォーマットは、［**Extended Field Editing**］ウィンドウの［**Format**］フィールドに表示されます。カーソルは［**AL**］フィールドに自動的に移動します。

4 ［**AL**］フィールドに「"20"」を入力します。

これは、データフィールドが長さ 20 文字で定義されることを示します。

5 TAB キーを押します。

マップでフィールド長が "X" 文字で示されます。

6 ［**AD**］フィールドが強調表示されるまで、TAB キーを繰り返し押します。

I/O 特性および充填文字を定義します。次の属性定義文字は、［**AD**］フィールドに現在表示されています。

```
ILMFHT'_'
```

I/O 特性は文字 "M" で示されます。充填文字は '_' で示されます。このチュートリアルでは、これらのデフォルト設定を使用します。次の手順は、この機能の使用方法を単に示しています。この時点で、以下に詳しく説明するように、［**Attribute Definition**］ウィンドウおよび［**Extended Field Editing**］ウィンドウも閉じることができます。

注意: ［**AD**］フィールドにおける属性定義文字の配列順序は変わることがあります。つまり、属性が異なる位置で生じる可能性があります。

7 PF2 キーを押して、属性定義を編集します。

［**Attribute Definition**］ウィンドウが表示されます。

```
+-Attribute Definition--+
¦Representation         ¦
¦Alignment              ¦
¦I/O Characteristics    ¦
¦Mandatory Characters   ¦
¦Length Characteristics ¦
¦Upper/Lower Case       ¦
¦Filler Character       ¦
+-----------------------+
```

8 ［**I/O Characteristics**］を選択します。

次のウィンドウが表示されます。

```
+------------------------+
¦ A Input, non_protected ¦
¦ M Output, modifiable   ¦
¦ O Output, protected    ¦
+-Esc=Quit, Enter=Toggle-+
```

9 文字 "M" が［**AD**］フィールドにすでに表示されている場合は、ESC キーを押してこのウィンドウを終了します。

または:

文字 "M" が［**AD**］フィールドに表示されていない場合は、［**M Output, modifiable**］を選択します。

注意: 同じ属性を何度も選択するとトグル効果が生じ、属性の定義または削除が切り替えられます。

10 ［**Attribute Definition**］ウィンドウで［**Filler Character**］を選択します。

新しい充填文字の入力を求めるプロンプトが表示されます。

充填文字を使用して、マップ内の入力フィールドの空の位置を埋めます。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。

11 現在のカーソル位置で下線（_）を入力します。

ENTER キーを押す必要はありません。下線文字を入力すると、メッセージが直ちに非表示になります。

12 ESC キーを押して、［**Attribute Definition**］ウィンドウを閉じます。

13 ENTER キーを押して、［**Extended Field Editing**］ウィンドウを閉じ、フィールドをマップに格納します。

注意: ENTER キーの代わりに ESC キーを押すと、新しいデータフィールド定義は保存されず、フィールドはマップに格納されません。

14 上記の手順と同じ方法で 2 つ目のデータフィールドを追加します。2 つ目のデータフィールドを［**End**］フィールドの横に配置し、20 文字の長さを定義し、上記と同じ属性定義文字を定義します。

データフィールドの名前の指定

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。

同じ名前がプログラムで使用されていることを確認します。#NAME-START および #NAME-END。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 6.5. データフィールドの名前を定義するには

1 方向キーを使用して、開始名のデータフィールドを選択します。

2 ESC キーを押すと、メニューが再表示されます。

3 メニューの［**Modify**］を選択します。

［**Extended Field Editing**］ウィンドウが表示されます。

```
+-Extended Field Editing-------------------------------------------------------+
¦Field : #1                                                                    ¦
¦Format: A Len: 20         AL: 20          PM:                   ZP: N  SG: N  ¦
¦Rules : O Rule Editing: N Array: None     Array Editing: N      Mode:  Undef  ¦
¦AD: ILMFHT'_'  CD:        CV:                                   DY: N  HE: N  ¦
¦EM:                                                                           ¦
+-------------------------------------------------------------------------------+
```

［**Field**］フィールドには、Natural によって割り当てられたフィールド名が含まれています。"#1"。

4 ［**Field**］フィールドに「"#NAME-START"」と入力します。

5 ENTER キーを押します。

6 終了名のデータフィールドについても上記の手順を繰り返し、フィールド名として「"#NAME-END"」と入力します。

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 6.6. システム変数を追加するには

1 ESC キーを押すと、メニューが再表示されます。

2 ［**Create**］メニューで、［**User Defined**］を選択します。

［**Extended Field Editing**］ウィンドウが表示されます。

3 画面の左上隅（行 001、列 001）にカーソルを置き、ENTER キーを押します。

次のウィンドウが表示されます。

```
+---------------------+
¦ A Alphanumeric      ¦
¦ B Binary            ¦
¦ D Date              ¦
¦ F Floating Point    ¦
¦ I Integer           ¦
¦ L Logical           ¦
¦ N Numeric           ¦
¦ P Packed Numeric    ¦
¦ T Time              ¦
¦ * System            ¦
+---------------------+
```

4 ［*** System**］を選択します。

次のウィンドウが表示されます。

```
+----------------------------+
¦ *APPLIC-ID           A8    ¦
¦ *APPLIC-NAME         A32   ¦
¦ *COM                 A128  ¦
¦ *CONVID              I4    ¦
¦ *CPU-TIME            I4    ¦
¦ *CURRENT-UNIT        A32   ¦
¦ *CURSOR              N6    ¦
¦ *CURS-COL            P3    ¦
¦ *CURS-LINE           P3    ¦
¦ *DAT4D               A10   ¦
¦ *DAT4E               A10   ¦
¦ *DAT4I               A10   ¦
¦ *DAT4J               A7    ¦
¦ *DAT4U               A10   ¦
+----------------------------+
```

DOWN-ARROW または UP-ARROW を使用して、システム変数のリストをスクロールします。

5 ［***DAT4I**］を選択し、ENTER キーを押します。

いくつかの "X" 文字が表示されます。

6 ENTER キーを押して、［**Extended Field Editing**］ウィンドウを閉じます。

7 上記の手順を繰り返し、システム変数 ***TIMX** を追加します。画面の右上隅（行 001、列 070）に配置します。このシステム変数用に **TT:TT:TT** が表示されます。ENTER キーを押して、［**Extended Field Editing**］ウィンドウを閉じることを忘れないでください。

フィールドの再位置決め

追加したフィールドの再位置決めを行います。

手順 6.7. フィールドを移動するには

1 メニューを表示中に ESC キーを押すと、非表示になります。

2 方向キーを使用して、開始名のデータフィールド（"X" 文字で示されます）を選択します。

データフィールドが強調表示されます。

3 ESC キーを押すと、メニューが再表示されます。

4 ［**Drag**］を選択します。

メニューが非表示になります。

5 方向キーを使用して、強調表示されたフィールドを行の中央に移動します。

6 ENTER キーを押すと、ドラッグモードが終了します。

7 開始名および終了名の残りのテキストフィールドとデータフィールドを、上記の手順で行の中央に移動します。開始名および終了名のフィールド間の空行はそのままにします。

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。

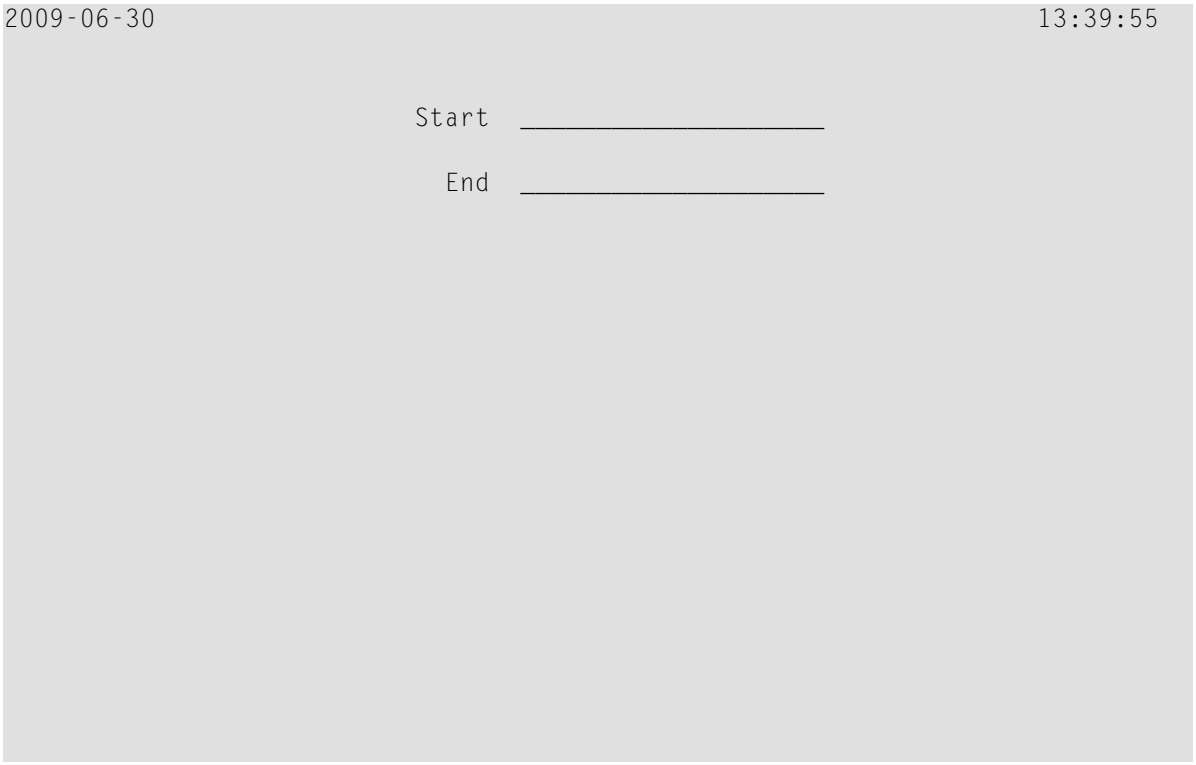
マップのテスト

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 6.8. マップをテストするには

1 ESC キーを押してメニューを再表示し、［**Ops. Map**］メニューで［**Test Map**］を選択します。

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 6.9. マップを Stow するには

1 ESC キーを押すと、メニューが再表示されます。

2 ［**Ops. Map**］メニューで、［**Stow Map**］を選択します。

次のウィンドウが表示されます。

```
+--Stow Map As: ---+
¦Name...:          ¦
¦Library: TUTORIAL ¦
+------------------+
```

3 マップ名として「"MAP01"」と入力し、ENTER キーを押します。

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 6.10. プログラムからマップを起動するには

1 ESC キーを押してメニューを再表示し、［**Quit**］を選択します。

Natural のメインメニューが再表示され、［**Direct Command**］ウィンドウで入力を求められます。

2 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
```

3 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
```

4 プログラムを実行します。

マップが表示されます。

5 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。

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 6.11. 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'の下に以下を挿入します。

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

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。

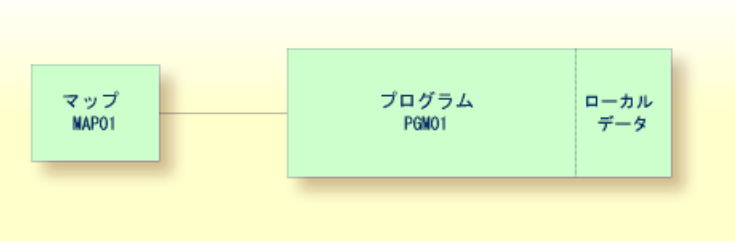
6 プログラムを格納します。

次の演習に進みます。*ループおよびラベル*

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 の下に以下を挿入します。

```
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' の下に以下を挿入します。

```
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 の下に以下を挿入します。

```
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である場合（指定した名前の従業員が見つからない場合）、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。

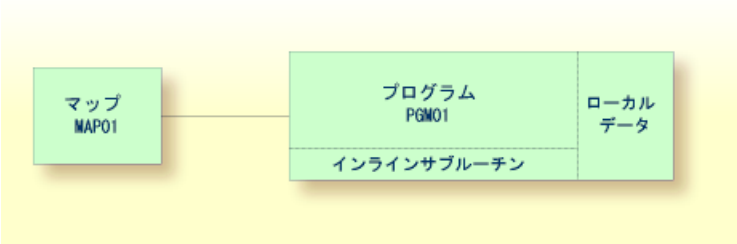
次の演習に進みます。*インラインサブルーチン*

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を使用することもできます。#MARK := '*'

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 ステートメントの前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、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。

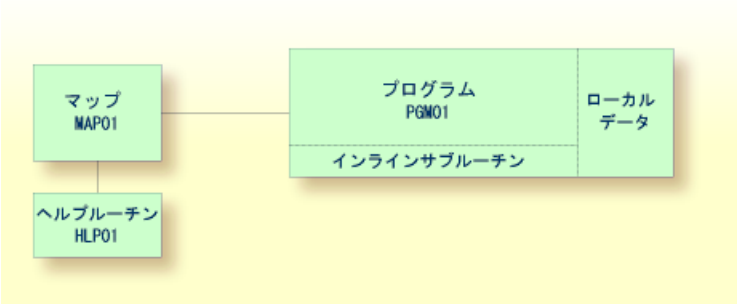
5 プログラムを格納します。

次の演習に進みます。*処理ルールとヘルプルーチン*

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**］メニューでマップエディタに戻ります。

```
EDIT MAP01
```

2 方向キーを使用して、開始名の入力フィールドを選択します。ESC キーを押してメニューを再表示し、［**Modify**］を選択します。

［**Extended Field Editing**］ウィンドウが表示されます。

```
+-Extended Field Editing-------------------------------------------------------+
¦Field : #NAME-START                                                          ¦
¦Format: A Len: 20          AL: 20          PM:                  ZP: N  SG: N ¦
¦Rules : 0 Rule Editing: N Array: None      Array Editing: N     Mode:  User  ¦
¦AD: ILMFHT'_'  CD:         CV:                                  DY: N  HE: N ¦
¦EM:                                                                          ¦
+-------------------------------------------------------------------------------+
```

3 TAB キーを使用して、［**Rule Editing**］フィールドに移動します。このフィールドで「"Y"」と入力するか、PF2 キーを押します。

次の画面が表示されます。

```
+-Current Field: #NAME-START-------------------------------------------------+
¦                      R U L E   E D I T I N G (Esc = Quit)                  ¦
¦Rules                                  Fields                               ¦
+-----------------------------------------------------------------------------+

                            End XXXXXXXXXXXXXXXXXXX









 Create or modify a rule for this field
```

4 ［**Rules**］を選択してから［**<CREATE>**］を選択します。

空のエディタウィンドウが表示されます。

5 次の処理ルールを入力します（空行を挿入するには、行コマンド I を使用します）。

```
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。

6 コマンド行で次のように入力します。

```
P=1
```

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。

```
EDIT:                                      S 02- ---------------Columns 001 074
Rank:  NEW RULE      Rule:                                        Typ: R  Mode: S
Cmnd=> p=1                                                     Scroll==> CSR
****** ******************************** top of data ******************************
000001 IF & = ' ' THEN
000002   REINPUT 'PLEASE ENTER A STARTING NAME.'
000003   MARK *&
000004 END-IF
****** ***************************** bottom of data *****************************









PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12--
P     U     End   P*    Rfind Rchng Up    Down        Right Left  Home
```

7 ENTER キーを押して、入力内容を保存します。

次の画面が再表示されます。ランク 1 の新しいルールが［**<CREATE>**］の下に表示されます。このルールを選択すると、コードの最初の行が画面下部に表示されます。

```
+-Current Field: #NAME-START-------------------------------------------------+
¦                      R U L E   E D I T I N G (Esc = Quit)                  ¦
¦Rules                                  Fields                               ¦
+--------+-------------------------------------------------------------------+
¦<CREATE>¦
¦       1¦             End XXXXXXXXXXXXXXXXXXX
+--------+









0010 IF & = ' ' THEN
0020   REINPUT 'PLEASE ENTER A STARTING NAME.'
0030   MARK *&

 Create or modify a rule for this field
```

8 ［Natural Map Editor］メニューが再表示されるまで、ESC キーを繰り返し押します。

9 マップをテストします。

10 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。

11 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。

12 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。

13 マップを Stow 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 ［Natural Map Editor］メニューから、［**Quit**］を選択します。

Natural のメインメニューが再表示され、［**Direct Command**］ウィンドウで入力を求められます。

2 以下のいずれかを入力します。

```
EDIT PROGRAM
```

```
E P
```

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 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、プログラムをヘルプルーチンに変更します。

```
SET TYPE H
```

"H" はヘルプルーチンを示します。

5 ヘルプルーチンを格納し「HLP01」という名前にします。

```
STOW HLP01
```

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を選択し、ESC キーを押して［Natural Map Editor］メニューを表示し、［**Modify**］を選択します。

フィールドに対する［**Extended Field Editing**］ウィンドウが表示されます。

3 TAB キーを使用して、［**HE**］フィールドに移動します。このフィールドで「"Y"」と入力するか、PF2 キーを押します。

ヘルプルーチン名の入力を求めるウィンドウが表示されます。

4 ［**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。

```
XXXXXXXXXX                                                      TT:TT:TT


                         Start XXXXXXXXXXXXXXXXXXX

                           End XXXXXXXXXXXXXXXXXXX




     +-HE-------------------------------------------------------------------+
     ¦HE: 'HLP01'                                                          ¦
     ¦                                                                     ¦
     +---------------------------------------------------------------------+

+-Extended Field Editing-----------------------------------------------------+
¦Field : #NAME-START                                                        ¦
¦Format: A Len: 20         AL: 20          PM:                  ZP: N  SG: N ¦
¦Rules : 1 Rule Editing: Y Array: None     Array Editing: N     Mode:  User  ¦
¦AD: MILFHT'_'  CD:        CV:                                  DY: N  HE: Y ¦
¦EM:                                                                        ¦
+-----------------------------------------------------------------------------+
 Helproutine and Parameters
```

5 ENTER キーを 2 回押して、変更内容を保存し、すべてのウィンドウを閉じます。

6 マップをテストします。

7 結果の出力で、開始名の入力フィールドに疑問符（?）を入力します。

定義したヘルプテキストが表示されます。

8 ENTER キーを押して、マップに戻ります。

9 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。

10 マップを Stow します。

11 ［Natural Map Editor］メニューから、［**Quit**］を選択します。

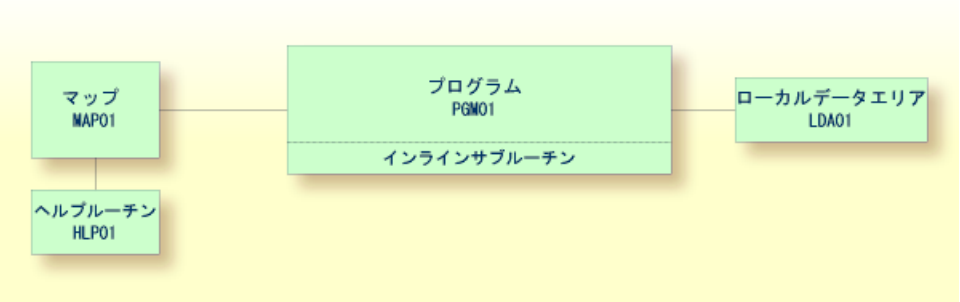
Natural のメインメニューが再表示され、［**Direct Command**］ウィンドウで入力を求められます。

次の演習に進みます。*ローカルデータエリア*.

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

■ ［**Direct Command**］ウィンドウに次のいずれかを入力します。

```
EDIT LOCAL
```

```
E L
```

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、画面の上に示されます。

```
                        Press <ESC> to enter command mode
 Mem: empty    Lib: TUTORIAL Type: LOCAL      Bytes:     0  Line:    0 of:   0
C T    Comment
  *    *** Top of Data Area ***
  *    *** End of Data Area ***
```

```
  F 1 HELP     F 2 CHOICE   F 3 QUIT     F 4 SAVE     F 5 STOW     F 6 CHECK
  F 7 READ     F 8 CLEAR    F 9 MEM TYPE F10 GEN      F11 FLD TYPE F12
```

デフォルトでは、データエリアエディタは起動時に編集モードになっています。編集モードからコマンドモード（およびその逆）に切り替えるには、ESC キーを押します。エディタを終了すると、次にエディタを開くときに、現在のモードが再開されます。

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	Name	フォーマット	データ長
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 データエリアの先頭を示す最初のエントリが、エディタで選択されていることを確認します。

ヒント: カーソルを移動して行を強調表示できない場合は、コマンドモードになっています。このモードでは、コマンド行が画面上部に表示されます。ESC キーを押して、編集モードに切り替えます。

2 選択した行の最初の列で、次の行コマンドを入力します。

```
I
```

ENTER キーを押す必要はありません。

次のウィンドウが表示されます。

```
+-----------------------+
¦ D Data Field          ¦
¦ B Block               ¦
¦ C Constant            ¦
¦ H Handle              ¦
¦ S Structure           ¦
¦ * Comment             ¦
+-----------------------+
```

3 ［**Data Field**］を選択します。

［**Data Field Definition**］ウィンドウが表示されます。

```
+-------------------- Data Field Definition ----------------------+
¦ Level:           1                                              ¦
¦ Field Name:                                                     ¦
¦ Field Format:                                                   ¦
¦ Field Length:                                                   ¦
¦ Arraydefinition:                                                ¦
¦ Edit Mask:                                                      ¦
¦                                                                 ¦
¦ Header Definition:                                              ¦
¦                                                                 ¦
¦ Initialization:                                                 ¦
¦ Value Clause:                                                   ¦
¦ Optional Param:  N                                              ¦
¦ Comment:                                                        ¦
+-------------------------------------------------------------------+
```

4 上の表に示す必須情報をすべて最初のフィールド（#NAME-START）に対して指定します。次のフィールドに移動するには、方向キーを使用します。

5 このフィールドの情報をすべて指定したら、ENTER キーを押して情報を保存します。

［**Data Field Definition**］ウィンドウは開いたままになります。入力フィールドは再び空になり、新しいデータフィールドを定義できます。

6 上記の手順で、すべての必須情報を残りのフィールド（#NAME-END および #MARK）に対して指定します。

7 すべてのフィールドが追加されたら、ESC キーを押します。

定義したフィールドがエディタに表示されます。

```
                      Press <ESC> to enter command mode
 Mem:          Lib: TUTORIAL Type: LOCAL      Bytes:   291  Line:    3 of:   3
C T L  Name of Datafield                F     Length Index/Comment             M
  *    *** Top of Data Area ***
     1 #NAME-START                      A         20
     1 #NAME-END                        A         20
     1 #MARK                            A          1
  *    *** End of Data Area ***
  F 1 HELP     F 2 CHOICE   F 3 QUIT     F 4 SAVE     F 5 STOW     F 6 CHECK
  F 7 READ     F 8 CLEAR    F 9 MEM TYPE F10 GEN      F11 FLD TYPE F12
```

ヒント: フィールド定義に誤りがある場合は、行コマンド E を使用して、選択したフィールドを編集したり、行コマンド D を使用して、選択したフィールドを削除したりすることができます。

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。

データエリアエディタで、インポートしたデータフィールドは、現在選択されているデータフィールドの下に挿入されます。

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 ［#MARK］行の最初の列で、次の行コマンドを入力します。

```
V
```

［**View Definition**］ウィンドウが表示されます。

```
                      Press <ESC> to enter command mode
 Mem:          Lib: TUTORIAL Type: LOCAL      Bytes:   291  Line:    3 of:   3
C T L  Name of Datafield               F     Length Index/Comment             M
  *    *** Top of Data Area ***
     1 #NAME-START                     A         20
     1 #NAME-END                       A         20
V    1 #MARK                           A          1
  *    *** End of Data Area ***

           +------------------- View Definition -------------------+
           ¦Name of View:                                          ¦
           ¦Name of DDM:                                           ¦
           ¦Comment:                                               ¦
           +---------------------------------------------------------+









  F 1 HELP     F 2 CHOICE   F 3 QUIT     F 4 SAVE     F 5 STOW     F 6 CHECK
  F 7 READ     F 8 CLEAR    F 9 MEM TYPE F10 GEN      F11 FLD TYPE F12
```

2 次の情報を入力して ENTER キーを押します。

```
+------------------- View Definition -------------------+
¦Name of View:          EMPLOYEES-VIEW                  ¦
¦Name of DDM:           EMPLOYEES                       ¦
¦Comment:                                               ¦
+---------------------------------------------------------+
```

表示されるウィンドウに、指定された DDM のフィールドが表示されます。

```
                      Press <ESC> to enter command mode
 Mem:          Lib: TUTORIAL Type: LOCAL      Bytes:   291  Line:    3 of:   3
C T L  Name of Datafield                F     Length Index/Comment             M
  *    *** Top of Data Area ***
     1 #NAME-START                      A         20
     1 #NAME-END                        A         20
V    1 #MARK                            A          1
  *    *** End of Data Area ***




+------------------------------- DDM: EMPLOYEES --------------------------------+
¦    1  AA  PERSONNEL-ID                A          8    D                      ¦
¦  G 1  AB  FULL-NAME                                                          ¦
¦    2  AC  FIRST-NAME                  A         20  N                       ¦
¦    2  AD  MIDDLE-I                    A          1  N                       ¦
¦    2  AE  NAME                        A         20    D                     ¦
¦    1  AD  MIDDLE-NAME                 A         20  N                       ¦
+-------------------------------------------------------------------------------+
     HD=PERSONNEL/ID


  F 1 HELP     F 2 CHOICE   F 3 QUIT     F 4 SAVE     F 5 STOW     F 6 CHECK
  F 7 READ     F 8 CLEAR    F 9 MEM TYPE F10 GEN      F11 FLD TYPE F12
```

3 、1 番目の列に「"X"」を入力して、以下のフィールドをマークします。

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

DDM 内でスクロールするには、方向キーを使用します。

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、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（方向キーを使用して、データエリアの上部にスクロールします）。

```
                       Press <ESC> to enter command mode
 Mem:          Lib: TUTORIAL Type: LOCAL      Bytes:   970  Line:    0 of:  10
C T    Comment
  *    *** Top of Data Area ***
     1 #NAME-START                        A          20
     1 #NAME-END                          A          20
     1 #MARK                              A           1
  V  1 EMPLOYEES-VIEW                                    EMPLOYEES
     2 PERSONNEL-ID                       A           8
  G  2 FULL-NAME
     3 NAME                               A          20
     2 DEPT                               A           6
  G  2 LEAVE-DATA
     3 LEAVE-DUE                          N           2
  *    *** End of Data Area ***









  F 1 HELP     F 2 CHOICE   F 3 QUIT     F 4 SAVE     F 5 STOW     F 6 CHECK
  F 7 READ     F 8 CLEAR    F 9 MEM TYPE F10 GEN      F11 FLD TYPE F12
```

［**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 ESC キーを押して、コマンドモードに入ります。

6 ローカルデータエリアを格納し、「"LDA01"」という名前を付けます。

```
STOW LDA01
```

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 の間にある変数をすべて削除します（行コマンドD を使用します）。

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 以前と同じ結果であることを確認するため（DEFINE DATA ステートメントでローカルデータエリアを参照しなかった場合）、開始名として「"JONES"」と入力し、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。

次の演習に進みます。*グローバルデータエリア*

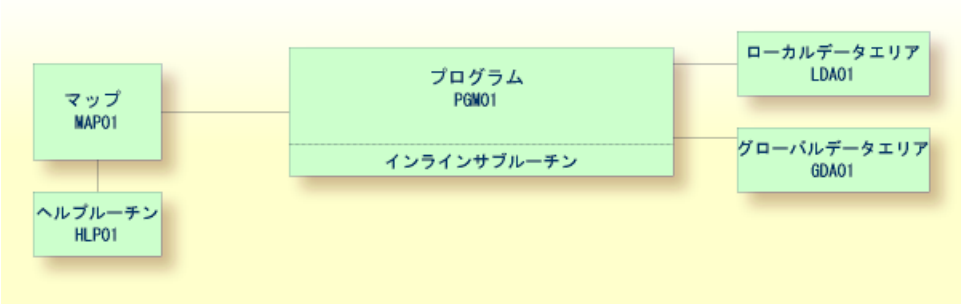
11 グローバルデータエリア

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、画面の上に示されます。

5 ESC キーを押して、編集モードに入ります。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
```

6 グローバルデータエリアは次のようになります。

```
                       Press <ESC> to enter command mode
 Mem: GDA01    Lib: TUTORIAL Type: GLOBAL     Bytes:   351  Line:    1 of:   8
C T L  Name of Datafield                F     Length Index/Comment             M
  *    *** Top of Data Area ***
     1 #MARK                            A          1
  V  1 EMPLOYEES-VIEW                                EMPLOYEES
     2 PERSONNEL-ID                     A          8
  G  2 FULL-NAME
     3 NAME                             A         20
     2 DEPT                             A          6
  G  2 LEAVE-DATA
     3 LEAVE-DUE                        N          2
  *    *** End of Data Area ***









  F 1 HELP     F 2 CHOICE   F 3 QUIT     F 4 SAVE     F 5 STOW     F 6 CHECK
  F 7 READ     F 8 CLEAR    F 9 MEM TYPE F10 GEN      F11 FLD TYPE F12
```

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。

```
Command:
 Mem: LDA01    Lib: TUTORIAL Type: LOCAL      Bytes:    85  Line:      of:   2
C T    Comment
  *    *** Top of Data Area ***
     1 #NAME-START                       A          20
     1 #NAME-END                         A          20
  *    *** End of Data Area ***
  F 1 HELP     F 2 CHOICE   F 3 QUIT     F 4 SAVE     F 5 STOW     F 6 CHECK
  F 7 READ     F 8 CLEAR    F 9 MEM TYPE F10 GEN      F11 FLD TYPE F12
```

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 以前（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なかった場合）と同じ結果であることを確認するため、開始名として「"JONES"」と入力し、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。

次の演習に進みます。*外部サブルーチン*

12 外部サブルーチン

これまでは、MARK-SPECIAL-EMPLOYEES ステートメントを使用して、サブルーチン DEFINE SUBROUTINE 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。

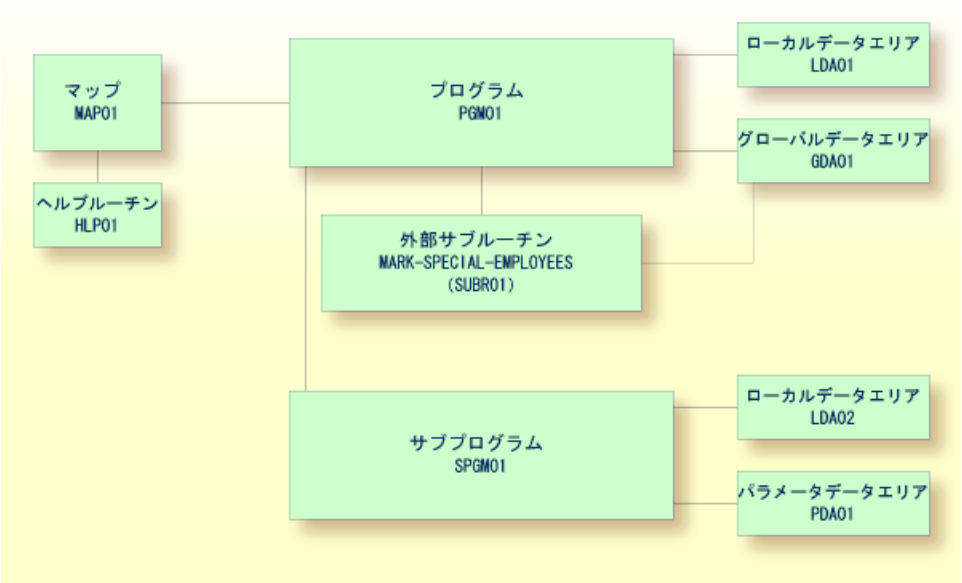
次の演習に進みます。サブプログラム

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 必要に応じて ESC キーを押し、編集モードに切り替えます。

3 #NAME-END の下に（行コマンド I を使用し、結果のウィンドウで［**Data Field**］を選択して）次のフィールドを定義します。

レベル	Name	フォーマット	データ長
1	#PERS-ID	A	8
1	#MAKE	A	20
1	#MODEL	A	20

すべてのフィールドが追加されたら、ESC 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。

```
                       Press <ESC> to enter command mode
 Mem: LDA01    Lib: TUTORIAL Type: LOCAL      Bytes:   376  Line:    5 of:   5
C T L  Name of Datafield                F     Length Index/Comment             M
  *    *** Top of Data Area ***
     1 #NAME-START                      A         20
     1 #NAME-END                        A         20
     1 #PERS-ID                         A          8
     1 #MAKE                            A         20
     1 #MODEL                           A         20
  *    *** End of Data Area ***
  F 1 HELP     F 2 CHOICE   F 3 QUIT     F 4 SAVE     F 5 STOW     F 6 CHECK
  F 7 READ     F 8 CLEAR    F 9 MEM TYPE F10 GEN      F11 FLD TYPE F12
```

4 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、行コマンド D を使用してフィールド #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。

```
Command:
 Mem: PDA01    Lib: TUTORIAL Type: PARAMETER  Bytes:   116  Line:      of:   3
C T    Comment
  *    *** Top of Data Area ***
     1 #PERS-ID                          A          8
     1 #MAKE                             A         20
     1 #MODEL                            A         20
  *    *** End of Data Area ***
```

```
  F 1 HELP     F 2 CHOICE   F 3 QUIT     F 4 SAVE     F 5 STOW     F 6 CHECK
  F 7 READ     F 8 CLEAR    F 9 MEM TYPE F10 GEN      F11 FLD TYPE F12
```

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 ESC キーを押して、編集モードに切り替えます。次に、データエリアの先頭を示す行の最初の列で、次の行コマンドを入力します。

```
V
```

4 結果のウィンドウで次のように入力し、ENTER キーを押します。

```
+------------------- View Definition -------------------+
¦Name of View:          VEHICLES-VIEW                   ¦
¦Name of DDM:           VEHICLES                        ¦
¦Comment:                                               ¦
+-------------------------------------------------------+
```

表示されるウィンドウに、指定された DDM のフィールドが表示されます。

```
                       Press <ESC> to enter command mode
 Mem: empty    Lib: TUTORIAL Type: LOCAL      Bytes:     0  Line:    0 of:   0
C T    Comment
V *    *** Top of Data Area ***
  *    *** End of Data Area ***









+------------------------------- DDM: VEHICLES --------------------------------+
¦    1  AA  REG-NUM                            A         15  N D                ¦
¦    1  AB  CHASSIS-NUM                        B          4  F                  ¦
¦    1  AC  PERSONNEL-ID                       A          8    D                ¦
¦  G 1  CD  CAR-DETAILS                                                         ¦
¦    2  AD  MAKE                               A         20  N D                ¦
¦    2  AE  MODEL                              A         20  N                  ¦
+-------------------------------------------------------------------------------+



  F 1 HELP     F 2 CHOICE   F 3 QUIT     F 4 SAVE     F 5 STOW     F 6 CHECK
  F 7 READ     F 8 CLEAR    F 9 MEM TYPE F10 GEN      F11 FLD TYPE F12
```

5 、1 番目の列に「"X"」を入力して、以下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、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（方向キーを使用して、データエリアの上部にスクロールします）。

```
                        Press <ESC> to enter command mode
 Mem:          Lib: TUTORIAL Type: LOCAL      Bytes:   485  Line:    0 of:   5
C T    Comment
  *    *** Top of Data Area ***
  V  1 VEHICLES-VIEW                                  VEHICLES
     2 PERSONNEL-ID                      A          8
  G  2 CAR-DETAILS
     3 MAKE                              A         20
     3 MODEL                             A         20
  *    *** End of Data Area ***
  F 1 HELP     F 2 CHOICE   F 3 QUIT     F 4 SAVE     F 5 STOW     F 6 CHECK
  F 7 READ     F 8 CLEAR    F 9 MEM TYPE F10 GEN      F11 FLD TYPE F12
```

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。