Natural

Erste Schritte

Version 8.2.8

April 2023

ADABAS & NATURAL

Dieses Dokument gilt für Natural ab Version 8.2.8.

Hierin enthaltene Beschreibungen unterliegen Änderungen und Ergänzungen, die in nachfolgenden Release Notes oder Neuausgaben bekanntgegeben werden.

Dokument-ID: NATMF-NNATFIRSTSTEPS-828-20230425DE

Inhaltsverzeichnis

Vorwort

Dieses Tutorial bietet eine einfache und kurze Einleitung in die Programmierung mit Natural und in die Benutzung der Natural-Editoren.

⚠ **Wichtig:** Es ist wichtig, dass Sie die folgenden Themen in der unten angegebenen Reihenfolge lesen und dass Sie alle darin enthaltenen Übungen in derselben Reihenfolge wie in diesem Tutorial abarbeiten. Wenn Sie eine Übung überspringen, könnte dies zu Problemen führen.

Über dieses Tutorial	Voraussetzungen und was Sie im Laufe dieses Tutorials lernen.
Grundlagen der Benutzung	Wie Sie das Hauptmenü von Natural aufrufen. Wie Sie die Library erstellen, mit der Sie in diesem Tutorial arbeiten werden. Informationen zu den Programmiermodi von Natural und über den Modus, der für dieses Tutorial erforderlich ist.
Hello World!	Wie Sie Ihr erstes kurzes Programm erstellen, ausführen und speichern. Wie Sie den Inhalt der aktuellen Library anzeigen. Informationen über einige Optionen, von denen Ihr Editorprofil beeinflusst wird.
Datenbankzugriff	Wie Sie bestimmte Informationen aus einer Datenbank lesen und ausgeben.
Benutzereingaben	Wie Sie den Benutzer zur Eingabe von Informationen auffordern und wie Sie eine Map für die Benutzereingabe entwerfen. Wie Sie gewährleisten, dass immer derselbe Wert benutzt wird (hier: ein Endname), auch wenn er nicht vom Benutzer eingegeben wurde.
Verarbeitungsschleifen und Labels	Wie Sie wiederholte Verarbeitungsschleifen und Labels für verschiedene Schleifen definieren. Wie eine Meldung angezeigt wird, wenn eine bestimmte Information (hier: der vom Benutzer eingegebene Startname) nicht gefunden wird.
Interne Subroutinen	Wie Sie eine interne Subroutine (das heißt: eine Subroutine, die direkt ins Programm geschrieben wird) definieren und aufrufen.
Verarbeitungsregeln und Helproutinen	Wie Sie eine Verarbeitungsregel (hier: eine Meldung, die erscheint, wenn der Benutzer keinen Startnamen angibt) und eine Helproutine (hier: ein Hilfetext für das Feld, in dem der Benutzer einen Startnamen angeben muss) definieren.
Local Data Areas	Wie Sie die Felddefinitionen aus dem Programm in eine Local Data Area außerhalb des Programms verlagern.
Global Data Areas	Wie Sie eine Global Data Area definieren, die von mehreren Programmen oder Routinen benutzt werden kann.
Externe Subroutinen	Wie Sie eine externe Subroutine definieren und aufrufen (das heißt: eine Subroutine, die als separates Objekt außerhalb des Programms gespeichert ist).
Subprogramme	Wie Sie eine Parameter Data Area für ein Subprogramm definieren. Wie Sie ein Subprogramm definieren und aufrufen.

1 Über diese Dokumentation

Dokumentationskonventionen

Konvention	Beschreibung
Fettschrift	>Kennzeichnet Elemente auf einem Bildschirm.
`Nichtproportionale Schrift`	Kennzeichnet Namen und Orte von Diensten im Format *`Ordner.Unterordner.Dienst`*, Programmierschnittstellen (APIs), Namen von Klassen, Methoden und Properties in Java.
`Kursivschrift`	Kennzeichnet: Variablen, für die Sie situations- oder umgebungsspezifische Werte angeben müssen. Neue Begriffe, wenn sie erstmals im Text auftreten. Verweise auf andere Dokumentationsquellen.
`Nichtproportionale Schrift`	Kennzeichnet: Text, den Sie eingeben müssen. Meldungen, die vom System angezeigt werden. Programmcode.
{ }	Zeigt eine Reihe von Auswahlmöglichkeiten an, von denen Sie eine auswählen müssen. Geben Sie nur die innerhalb der geschweiften Klammern vorhandenen Informationen ein. Geben Sie nicht die Klammersymbole { } ein.
|	Trennt zwei sich gegenseitig ausschließende Auswahlmöglichkeiten in einer Syntaxzeile voneinander ab. Geben Sie eine der Auswahlmöglichkeiten ein. Geben Sie nicht das Symbol | ein.
[ ]	Zeigt eine oder mehrere Optionen an. Geben Sie nur die innerhalb der eckigen Klammern vorhandenen Informationen ein. Geben Sie nicht die Klammersymbole [ ] ein.
...	Zeigt an, dass Sie mehrere Auswahlmöglichkeiten desselben Typs eingeben können. Geben Sie nur die Informationen ein. Geben Sie nicht die drei Auslassungspunkte (...) ein.

Online-Informationen und Support

Produktdokumentation

Sie finden die Produktdokumentation auf unserer Dokumentationswebsite unter **https://documentation.softwareag.com**.

Zusätzlich können Sie auch über **https://www.softwareag.cloud** auf die Dokumentation für die Cloud-Produkte zugreifen. Navigieren Sie zum gewünschten Produkt und gehen Sie dann, je nach Produkt, zu „Developer Center“, „User Center“ oder „Documentation“.

Produktschulungen

Sie finden hilfreiches Produktschulungsmaterial auf unserem Lernportal unter **https://knowledge.softwareag.com**.

Tech Community

Auf der Website unserer Tech Community unter **https://techcommunity.softwareag.com** können Sie mit Experten der Software AG zusammenarbeiten. Von hier aus können Sie zum Beispiel:

- Unsere umfangreiche Wissensdatenbank durchsuchen.
- In unseren Diskussionsforen Fragen stellen und Antworten finden.
- Die neuesten Nachrichten und Ankündigungen der Software AG lesen.
- Unsere Communities erkunden.
- Unsere öffentlichen Repositories auf GitHub and Docker unter **https://github.com/softwareag** und **https://hub.docker.com/publishers/softwareag>** besuchen und weitere Ressourcen der Software AG entdecken.

Produktsupport

Support für die Produkte der Software AG steht lizenzierten Kunden über unser Empower-Portal unter **https://empower.softwareag.com>** zur Verfügung. Für viele Dienstleistungen auf diesem Portal benötigen Sie ein Konto. Wenn Sie noch keines haben, dann können Sie es unter **https://empower.softwareag.com/register>** beantragen. Sobald Sie ein Konto haben, können Sie zum Beispiel:

- Produkte, Aktualisierungen und Programmkorrekturen herunterladen.
- Das Knowledge Center nach technischen Informationen und Tipps durchsuchen.
- Frühwarnungen und kritische Alarme abonnieren.
- Supportfälle öffnen und aktualisieren.
- Anfragen für neue Produktmerkmale einreichen.

Datenschutz

Die Produkte der Software AG stellen Funktionen zur Verarbeitung von personenbezogenen Daten gemäß der Datenschutz-Grundverordnung (DSGVO) der Europäischen Union zur Verfügung. Gegebenenfalls sind in der betreffenden Systemverwaltungsdokumentation entsprechende Schritte dokumentiert.

2 Über dieses Tutorial

Wenn Sie Natural zum ersten Mal benutzen, sollten Sie dieses Tutorial durcharbeiten, um Grundkenntnisse über bestimmte Merkmale der Natural-Programmierumgebung zu erhalten.

Das Layout der in diesem Tutorial beschriebenen Beispielbildschirme und das hier beschriebene Verhalten von Natural kann von Ihren Resultaten abweichen. So kann zum Beispiel die Kommandozeile oder Meldungszeile an einer anderen Position im Bildschirm erscheinen oder die Ausführung eines Natural-Kommandos kann durch Sicherheitseinstellungen geschützt sein. Die Standardeinstellungen in Ihrer Umgebung sind abhängig von den Systemparametern, die von Ihrem Natural-Administrator gesetzt wurden.

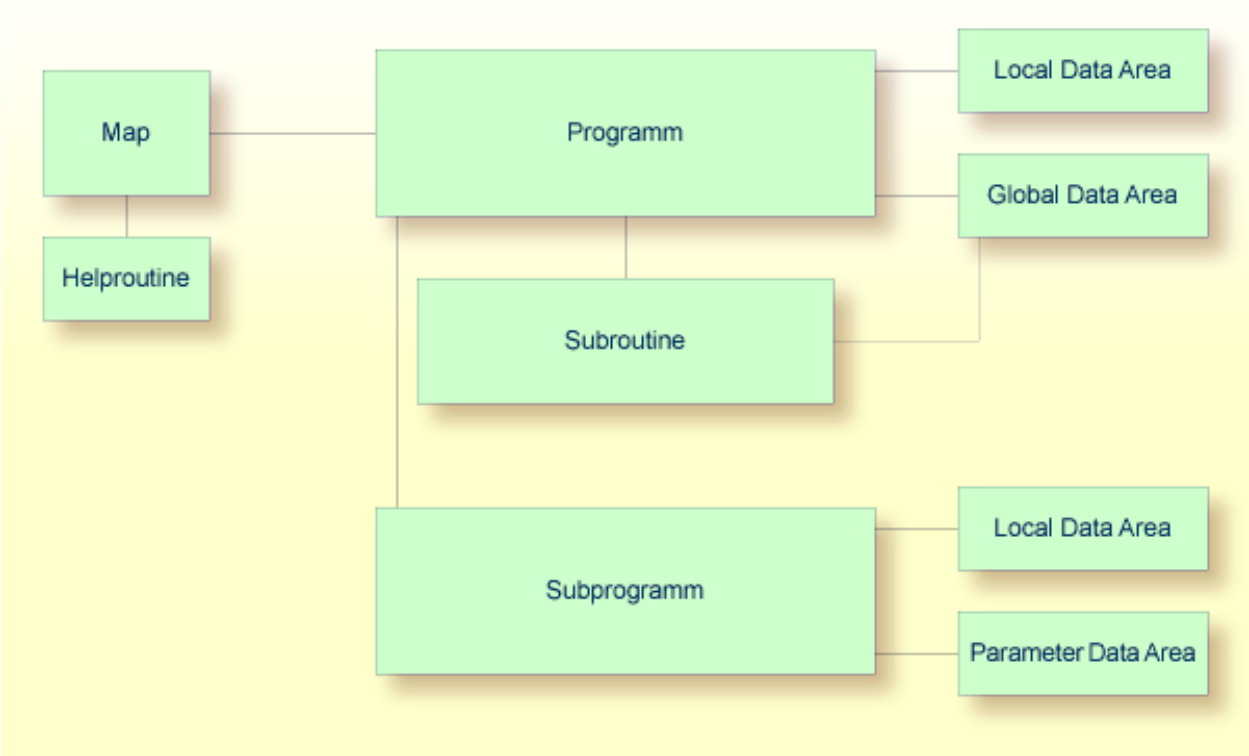
Voraussetzungen

Damit alle Schritte dieses Tutorials ausgeführt werden können, müssen die Adabas-Demodateien `EMPLOYEES` und `VEHICLES` installiert sein, sowie die Natural-Beispielobjekte. Falls dies noch nicht installiert ist, bitten Sie Ihren Administrator, dies zu tun.

Über die Beispielanwendung

Dieses Tutorial zeigt, wie Sie eine Anwendung als eine Gruppe von Modulen strukturieren können. Es ist nicht dazu gedacht, Ihnen zu zeigen, wie eine Anwendung aufgebaut werden sollte.

Zuerst werden Sie Ihr erstes kurzes "Hello World"-Programm schreiben. Danach schreiben Sie ein Programm, mit dem Mitarbeiter- (Employees-) Informationen aus einer Datenbank gelesen und angezeigt werden. Der Benutzer wird aufgefordert einen Start- und einen Endnamen für die Ausgabe anzugeben. Sie werden Ihr Programm Schritt für Schritt erweitern und dabei bestimmte Programmteile in externe Module verschieben. Wenn Sie mit allen Schritten dieses Tutorials fertig sind, dann hat Ihre Anwendung die folgende Struktur:



Sie können nun mit der ersten Übung beginnen: *Grundlagen der Benutzung*.

3 Grundlagen der Benutzung

Das Natural-Hauptmenü aufrufen

Das Natural-Hauptmenü (Main Menu) ermöglicht den Zugriff auf die Entwicklungsfunktionen, Umgebungseinstellungen, Utilities und Beispiel-Libraries von Natural.

Das Natural-Hauptmenü aufrufen

1 Starten Sie Natural mit den Kommandos, die zu diesem Zweck in Ihrem System bereit gestellt werden.

In Abhängigkeit von den Standardeinstellungen in Ihrer Umgebung erscheint entweder das Natural-Hauptmenü, die `NEXT`-Zeile oder die `MORE`-Zeile.

2 Wenn eine der oben genannten Zeilen erscheint, geben Sie Folgendes an dieser Eingabeaufforderung ein:

```
MAINMENU
```

Das Hauptmenü erscheint.

```
09:51:48                    *****  NATURAL  *****                    2012-07-17
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  ↩
 ↩
```

Libraries

Alle Natural-Objekte, die zum Erstellen einer Anwendung erforderlich sind, werden in Natural-Libraries und Natural-Systemdateien gespeichert. Es gibt eine Systemdatei für Systemprogramme (FNAT) und eine Systemdatei für Benutzerprogramme (FUSER).

Natural unterscheidet zwischen System-Libraries und Benutzer-Libraries. Die System-Libraries, die mit den Buchstaben "SYS" beginnen, sind ausschließlich Zwecken der Software AG vorbehalten. Eine Benutzer-Library enthält alle Benutzerobjekte (beispielsweise Programme und Maps), aus denen eine Anwendung besteht. Der Name einer Benutzer-Library darf nicht mit den Buchstaben "SYS" beginnen. Das Feld **Library** in der oberen rechten Ecke des Natural-Hauptmenüs (und von vielen anderen Bildschirmen) zeigt den Namen der Library, in der Sie gerade angemeldet sind.

Kommandos eingeben

Bei der Eingabe eines Natural-Kommandos wird nicht zwischen Groß- und Kleinschreibung unterschieden. Drücken Sie EINGABE nachdem Sie ein Natural-Kommando eingegeben haben. Mit EINGABE wird die Aktion bestätigt und das Kommando ausgeführt; es kann aber auch ein zusätzliches Bestätigungsfenster erscheinen, in dem Sie die Kommandoausführung explizit bestätigen müssen.

Benutzer-Library erstellen

Sie werden jetzt eine Benutzer-Library mit dem Namen TUTORIAL erstellen. Diese Library wird alle Natural-Objekte enthalten, die Sie im Laufe dieses Tutorials erstellen werden.

Benutzer-Library erstellen

- Geben Sie Folgendes in der Kommandozeile (**Command ===>** im Natural-Hauptmenü) ein:

  ```
  LOGON TUTORIAL
  ```

  wobei "TUTORIAL" der Name der Library ist, die Sie erstellt haben.

  LOGON ist ein Systemkommando, das für die folgenden Zwecke benutzt wird:

  - um sich in einer bereits existierenden Library anzumelden,
  - um eine neue Library zu erstellen, wenn eine Library mit dem angegebenen Namen noch nicht existiert.

Oder:

Überschreiben Sie den Namen der aktuellen Library in der oberen rechten Ecke des Bildschirms mit dem Namen der Library, in der Sie sich anmelden möchten und drücken Sie EINGABE.

Beispiel:

```
11:33:26                    *****  NATURAL  *****                    2012-07-17
User SAG                        - Main Menu -                 Library TUTORIAL
```

Das Menü Development Functions

Mit dem Menü **Development Functions** können Sie Natural-Objekte erstellen und ändern.

Das Menü Development Functions aufrufen

- Geben Sie im Natural-Hauptmenü ein beliebiges Zeichen im Eingabefeld neben **Development Functions** ein und drücken Sie EINGABE.

  Oder:

  Benutzen Sie die Cursor-Auswahl, das heißt: stellen Sie den Cursor in das Eingabefeld neben **Development Functions** und drücken Sie EINGABE.

  Das Menü **Development Functions** erscheint.

```
11:56:21                    *****  NATURAL  *****                    2012-07-17
User SAG                  - Development Functions -          Library TUTORIAL
                                                                Mode Structured
                                                            Work area empty

            Code  Function            Code  Function

             C    Create Object        L    List Objects or Single Source
             E    Edit Object          O    List Source with Expanded Sources
             X    Execute Program      N    List Extended Object Names
             R    Rename Object        I    List Directory Information
             D    Delete Objects       U    List Used Subroutines, etc.
             S    Scan Objects         ?    Help
                                       .    Exit


     Code .. _    Type .. _   Name .. ________________________________



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

Programmiermodi

Der Programmiermodus wird im **Mode**-Feld in der oberen rechten Ecke des Menüs **Development Functions** angezeigt.

Bei Natural gibt es zwei Programmiermodi:

- **Structured Mode**
  Der Structured Mode ist für komplexe Anwendungen gedacht, bei denen es auf eine klare und sinnvoll gegliederte Programmstruktur ankommt. Grundsätzlich empfiehlt es sich, ausschließlich im Structured Mode zu arbeiten.
- **Reporting Mode**
  Der Reporting Mode eignet sich nur für die Erstellung einfacher Reports und Programme, die keine komplexe Daten- und Programmstruktur erfordern.

⚠ **Wichtig:** Dieses Tutorial setzt voraus, dass der Structured Mode aktiv ist. Falls Sie versuchen Ihr Programm im Reporting Mode auszuführen, dann werden Fehler durch `END-IF`, `END-READ` und `END-REPEAT` verursacht.

Gehen Sie wie unten beschrieben vor, falls der Reporting Mode gerade aktiv ist.

Vom Reporting Mode in den Structured Mode umschalten

- Geben Sie das folgende Systemkommando in der Kommandozeile ein und drücken Sie EINGABE.

```
GLOBALS SM=ON
```

Oder:

Überschreiben Sie in der oberen rechten Ecke des Menüs **Development Functions** die erste Position des **Mode**-Feldes (in dem "Reporting" angezeigt wird) mit dem folgenden Buchstaben und drücken Sie EINGABE:

```
S
```

Beispiel:

```
12:17:20                    *****  NATURAL  *****                    2012-07-17
User SAG                  - Development Functions -          Library TUTORIAL ↩

                                                                Mode Seporting
                                                           Work area empty
```

Sie können nun mit Ihrem ersten Programm beginnen: *Hello World!*

4 Hello World!

Programm erstellen

Sie werden nun Ihr erstes kurzes Programm schreiben, das "Hello World!" anzeigt. Es wird in der Library gespeichert, die Sie zuvor erstellt haben.

Neues Programm erstellen

1 Achten Sie darauf, dass Sie in der Library `TUTORIAL` angemeldet sind.

2 Geben Sie unten im Menü **Development Functions** die folgenden Informationen ein und drücken Sie EINGABE:

```
    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" steht für die Funktion **Create Object** (Objekt erstellen), "P" steht für den Objekttyp Programm und "HELLO" ist der Name des zu erstellenden Programms.

Tipp: Wenn Sie den Funktionscode `C` eingeben, können Sie auch einen Stern (*) im **Type**-Feld angeben. Wenn Sie EINGABE drücken, erscheint eine Liste aller Objekttypen und aller Buchstaben, die für diese Objekttypen benutzt werden können.

Der Programmeditor erscheint. Er ist zurzeit leer.

3 Geben Sie den folgenden Code im Programmeditor ein:

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

Eine Kommentarzeile beginnt mit einem Stern (*), dem mindestens ein Leerzeichen oder ein zweiter Stern folgt. Wenn Sie das Leerzeichen oder den zweiten Stern vergessen, geht Natural davon aus, dass Sie eine Systemvariable angegeben haben; dies verursacht einen Fehler.

Wenn Sie Leerzeilen in Ihr Programm einfügen möchten, dann sollten Sie sie als Kommentarzeilen definieren. Dies ist hilfreich, wenn Sie Ihr Programm auf einer anderen Plattform (Windows, Großrechner, UNIX oder OpenVMS) weiterbearbeiten wollen. Mit der Großrechnerversion von Natural werden beispielsweise alle leeren Zeilen automatisch gelöscht, sobald Sie EINGABE drücken (Standardeinstellung).

Sie können auch einen Kommentar am Ende einer Statement-Zeile einfügen. In diesem Fall muss der Kommentar mit einen Schrägstrich beginnen, dem ein Stern folgt (/*).

Der Text, der in der Ausgabe erscheinen soll, wird mit dem Statement `DISPLAY` angegeben. Er steht in Anführungszeichen.

Das Statement `END` markiert das physische Ende des Natural-Programms. Jedes Programm muss mit `END` abgeschlossen werden.

Wenn Sie EINGABE drücken, kann es passieren, dass alle Kleinbuchstaben in Großbuchstaben umgesetzt werden. Dieses Verhalten ist im Editorprofil definiert (dies wird später erklärt).

Programm mit RUN ausführen

Das Systemkommando `RUN` ruft automatisch das Systemkommando `CHECK` auf, welches das Programm auf Fehler überprüft. Wenn kein Fehler gefunden wird, wird das Programm im Flug kompiliert und ausgeführt.

Anmerkungen:

1. `CHECK` steht ebenfalls als separates Systemkommando zur Verfügung.
2. Es gibt bei Natural auch das Systemkommando `EXECUTE`, welches die mit dem Systemkommando `STOW` gespeicherte Version Ihres Programms benutzt (dieses Kommando wird später in diesem Tutorial erklärt). Im Gegensatz zu `EXECUTE` werden bei `RUN` immer die letzten Programmänderungen berücksichtigt.

Programm mit `RUN` ausführen

1 Geben Sie eines der folgenden Kommandos in der Kommandozeile des Programmeditors ein:

```
RUN
```

```
R
```

Systemkommandos können abgekürzt werden. `R` ist die Abkürzung für `RUN`.

In Abhängigkeit von Ihren Umgebungseinstellungen befindet sich die Kommandozeile entweder am oberen oder unteren Rand des Bildschirms.

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

Wenn Ihr Code syntaktisch korrekt ist, wird der von Ihnen definierte Text ausgegeben.

```
MORE
Page      1                                                  10-11-22  12:07:25


Hello world!
                                                                                ↩
 ↩
```

2 Drücken Sie EINGABE um zum Programmeditor zurückzukehren.

Programmfehler korrigieren

Sie werden jetzt einen Fehler in Ihr Programm einbauen und das Programm anschließend noch einmal mit `RUN` ausführen.

Fehler korrigieren

1 Löschen Sie das zweite Anführungszeichen in der Zeile, die das `DISPLAY`-Statement enthält.

2 Führen Sie das Programm noch einmal wie oben beschrieben aus.

Wenn ein Fehler gefunden wird, wird eine Fehlermeldung angezeigt.

```
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  ↩
 ↩
```

Die fehlerhafte Zeile ist hervorgehoben und am Anfang mit einem "E" gekennzeichnet.

3 Korrigieren Sie den Fehler. Das heißt: fügen Sie das fehlende Anführungszeichen wieder am Ende der Zeile ein.

4 Führen Sie das Programm erneut aus, um den nächsten Fehler zu finden.

In diesem Fall werden keine weiteren Fehler gefunden und die Ausgabe wird angezeigt.

5 Drücken Sie EINGABE, um zum Programmeditor zurückzukehren.

Programm mit STOW speichern

Wenn Sie ein Programm mit `STOW` speichern, wird es kompiliert und der Quellcode sowie das generierte Programm werden in der Systemdatei abgelegt.

Wie beim Systemkommando `RUN` wird auch beim Systemkommando `STOW` automatisch das Systemkommando `CHECK` aufgerufen. Ein Programm kann nur dann mit `STOW` gespeichert werden, wenn es syntaktisch korrekt ist.

Anmerkung: Wenn Sie Ihre Programmänderungen speichern wollen, obwohl das Programm syntaktische Fehler enthält (wenn Sie Ihre Arbeit zum Beispiel bis zum nächsten Arbeitstag unterbrechen wollen), dann können Sie das Systemkommando `SAVE` benutzen.

Programm mit `STOW` speichern

- Geben Sie Folgendes in der Kommandozeile des Programmeditors ein:

```
STOW
```

Informationen über ein Programm anzeigen

Das Systemkommando `LIST` ist hilfreich, wenn Sie herausfinden wollen, ob für ein Objekt nur der Quellcode vorhanden ist oder ob beides, der Quellcode und das generierte Programm, vorhanden ist.

Informationen über ein Programm anzeigen

1 Geben Sie eines der folgenden Kommandos in der Kommandozeile des Programmeditors ein:

```
LIST DIR HELLO
```

```
L DIR HELLO
```

Der folgende Bildschirm erscheint. Die Informationen bei **Cataloged on** stehen nur dann zur Verfügung, wenn das Objekt mit `STOW` gespeichert wurde.

```
13:15:45              ***** NATURAL LIST COMMAND *****             2012-07-17
User SAG                      - List Directory -                Library TUTORIAL

Directory of Program HELLO                      Saved on ... 2012-07-17 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 2012-07-17 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
```

Anmerkung: Die Bezeichnungen *vr* und *v.r.s* im Beispielbildschirm oben stehen für die aktuelle Versionsnummer von Natural. Siehe auch die Definition von *Version* im *Glossary*.

2 Drücken Sie EINGABE um zum Programmeditor zurückzukehren.

Inhalt der aktuellen Library anzeigen

Das Systemkommando `LIST` kann dazu benutzt werden, eine Liste aller Natural-Objekte in der aktuellen Library anzuzeigen. Dies ist zum Beispiel dann hilfreich, wenn Sie im Laufe dieses Tutorials eines oder mehrere Ihrer Natural-Objekte löschen wollen, um wieder von vorne zu beginnen.

Liste der Natural-Objekte anzeigen

1 Geben Sie eines der folgenden Kommandos in der Kommandozeile des Programmeditors ein:

```
LIST *
```

```
L *
```

Der folgende Bildschirm erscheint. Das Programm, das sie eben erstellt haben, wird aufgelistet.

```
13:34:27               ***** NATURAL LIST COMMAND *****              2012-07-17
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        2012-07-17  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 Um herauszufinden, welche Kommandos zur Verfügung stehen, geben Sie ein Fragezeichen (?) in der **Cmd**-Spalte neben Ihrem Programm ein.

Das folgende Fenster erscheint.

```
+-------- 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 Ändern Sie jetzt noch nichts. Drücken Sie PF3 um das Fenster zu schließen, ohne dabei ein Kommando anzugeben.

4 Drücken Sie noch einmal PF3, um zum Programmeditor zurückzukehren.

Editorprofil-Optionen angeben

Wenn Sie mit dem Programmeditor oder Data-Area-Editor von Natural arbeiten, kann für jeden Benutzer ein Editorprofil definiert werden. In diesem Tutorial werden die Standardeinstellungen des Editorprofils mit dem Namen `SYSTEM` benutzt. Einige wichtige Einstellungen werden unten erwähnt.

Editorprofil-Optionen prüfen

1 Geben Sie Folgendes in der Kommandozeile des Programmeditors ein:

```
PROFILE
```

Der folgende Bildschirm erscheint.

```
13:35:43                ***** NATURAL EDITORS *****                2012-07-17
                            - 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  Reset                          Del   Canc
```

Wenn es für einen Benutzer noch kein Editorprofil gibt, dann wird das Standardprofil `SYSTEM` angezeigt. Mit diesem Standardprofil kann ein Benutzerprofil erstellt werden. Wenn ein Benutzerprofil bereits existiert, dann wird es anstelle des Profils `SYSTEM` angezeigt.

2 Geben Sie "Y" im Feld **Additional Options** (zusätzliche Optionen) ein und drücken Sie EINGABE.

Oder:

Drücken Sie PF4.

Das folgende Fenster erscheint.

```
+--------------------- Additional Options ----------------------+
!                                                               !
!                                                               !
!   +  Editor Defaults ......... N                              !
!   +  General Defaults ........ N                              !
!   +  Color Definitions ....... N                              !
!                                                               !
!                                                               !
!                                                               !
!                                                               !
!                                                               !
+---------------------------------------------------------------+
```

3 Geben Sie "Y" in den Feldern **Editor Defaults** (Standardeinstellungen für den Editor) und **General Defaults** (Allgemeine Standardeinstellungen) ein und drücken Sie EINGABE.

Zuerst erscheint das folgende Fenster.

```
+----------------------- Editor Defaults ------------------------+
|                                                                |
|    Escape Character for Line Command .. .                      |
|    Empty Line Suppression ............. Y                      |
|    Empty Line Suppression for Text .... N                      |
|    Source Size Information ............ N                      |
|    Source Status Message .............. Y                      |
|    Absolute Mode for SCAN/CHANGE ...... N                      |
|    Range Mode for SCAN/CHANGE ......... N                      |
|    Direction Indicator ................ +                      |
|                                                                |
+----------------------------------------------------------------+
```

Hinter **Escape Character for Line Command** sehen Sie das Umschaltzeichen, das für die Zeilenkommandos definiert ist. Dieses Tutorial geht davon aus, dass das Standardzeichen verwendet wird; dies ist der Punkt (.).

Dieses Tutorial geht auch davon aus, dass die Option **Empty Line Suppression** (Unterdrückung von Leerzeilen) auf "Y" gesetzt ist. In diesem Fall werden alle Leerzeilen im Programmeditor automatisch gelöscht, sobald Sie EINGABE drücken. Die Leerzeilen werden nicht gelöscht, wenn diese Option auf "N" gesetzt ist.

4 Achten Sie darauf, dass für dieses Tutorial alle Optionen so gesetzt sind, wie oben angezeigt. Drücken Sie EINGABE, um das nächste Fenster anzuzeigen.

Das folgende Fenster erscheint.

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

Wenn die Option **Editing in Lower Case** (Editieren in Kleinbuchstaben) auf "Y" gesetzt ist und die Option **Dynamic Conversion of Lower Case** (dynamische Umsetzung in Kleinbuchstaben) auf "N" gesetzt ist, dann bleibt der Quellcode so erhalten, wie Sie ihn eingeben. Diese Eigenschaft ist jedoch abhängig von den Einstellungen in Ihrer Systemumgebung; wenn

diese Einstellungen so gewählt sind, dass eine Umsetzung in Großbuchstaben erzwungen wird, dann kann dies nicht von Natural beeinflusst werden.

5 Wenn Sie es möchten, können Sie die oben genannten Optionen für die Umsetzung in Kleinbuchstaben ändern und EINGABE drücken. Drücken Sie noch einmal EINGABE, um zum Fenster **Additional Options** zurückzukehren, und drücken Sie dann noch einmal EINGABE, um dieses Fenster zu schließen.

6 Wenn ein Benutzerprofil noch nicht erstellt wurde, überschreiben Sie den Profilnamen `SYSTEM` mit Ihrer User-ID und drücken Sie EINGABE.

Wenn ein Benutzerprofil bereits existiert, fahren Sie fort mit dem nächsten Schritt.

7 Drücken Sie PF5, um Ihre Änderungen in der Datenbank zu speichern, und drücken Sie dann PF3, um das Editorprofil zu verlassen.

Anmerkung: Statt eine PF-Taste zu drücken können Sie auch das entsprechende Kommando in der Kommandozeile eingeben. Im hier beschriebenen Fall können Sie zum Beispiel die Kommandos `SAVE` und `EXIT` eingeben.

Oder:

Wenn Sie Ihre Änderungen nicht in der Datenbank speichern wollen und sie nur in der aktuellen Session benutzen möchten, drücken Sie PF3 um das Editorprofil zu verlassen.

Beim Verlassen des Editorprofils wird ein Fenster mit verschiedenen Optionen angezeigt.

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

Wählen Sie die Option **Exit without Saving** (Verlassen ohne Speichern), um die Änderungen nur in der aktuellen Session zu benutzen.

Oder:

Wenn Sie das Editorprofil verlassen wollen ohne die Änderungen zu speichern, drücken Sie PF12.

Ihr Programm wird wieder angezeigt. Falls Sie das Editorprofil geändert haben, dann werden die neuen Einstellungen jetzt im Programmeditor benutzt (und auch im Data-Area-Editor, der später in diesem Tutorial beschrieben wird).

Sie können nun mit den nächsten Übungen fortfahren: *Datenbankzugriff*.

5 Datenbankzugriff

Sie werden nun ein kurzes Programm schreiben, mit dem bestimmte Daten aus einer Datenbankdatei gelesen und angezeigt werden.

Wenn Sie mit den Übungen in diesem Kapitel fertig sind, wird Ihre Beispielanwendung aus einem einzigen Modul bestehen (die Datenfelder, die vom Programm benutzt werden, sind direkt in diesem Programm definiert).

Programm unter einem neuen Namen speichern

Sie werden jetzt ein neues Programm erstellen, das im weiteren Verlauf dieses Tutorials benutzt wird. Es wird erstellt, indem Sie Ihr "Hello World"-Programm unter einem neuen Namen speichern.

Programm unter einem neuen Namen speichern

1 Geben Sie eines der folgenden Kommandos in der Kommandozeile des Programmeditors ein:

```
SAVE PGM01
```

```
SA PGM01
```

Das aktuelle Programm wird mit dem neuen Namen `PGM01` gespeichert. Das Programm mit dem Namen `HELLO` wird aber immer noch im Programmeditor angezeigt.

2 Geben Sie das folgende Kommando in der Kommandozeile des Programmeditors ein, um das neu erstellte Programm in den Programmeditor einzulesen:

```
READ PGM01
```

Der im Programmeditor angezeigte Programmname ändert sich in `PGM01`.

3 Löschen Sie den gesamten Code im Programmeditor. Geben Sie hierzu das folgende Zeilenkommando am Anfang jeder zu löschenden Zeile ein und drücken Sie EINGABE:

```
.D
```

Beispiel:

```
>                                    > +  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                                                                       ↩
 ↩
```

Oder:

Geben Sie das folgende Zeilenkommando am Anfang der ersten Zeile ein und drücken Sie EINGABE:

```
.D(4)
```

wobei die Zahl in Klammern für die Anzahl der zu löschenden Zeilen steht.

Benötigte Daten mit einem View definieren

Die Datenbankdatei und die Felder, die in Ihrem Programm benutzt werden sollen, müssen am Anfang des Programms zwischen `DEFINE DATA` und `END-DEFINE` angegeben werden.

Damit Natural auf den Inhalt einer Datenbankdatei zugreifen kann, wird eine logische Definition der physischen Datenbankdatei benötigt. Eine solche logische Dateidefinition wird „Data Definition Module“ (DDM) genannt. Das DDM enthält Informationen über die einzelnen Felder der Datei. DDMs werden in der Regel vom Natural-Administrator definiert.

Damit die Datenbankfelder in einem Natural-Programm benutzt werden können, müssen Sie die Felder des DDMs in einem View angeben. In diesem Tutorial wird das DDM für die Datenbankdatei `EMPLOYEES` benutzt.

`DEFINE DATA`-Block angeben

- Geben Sie den folgenden Code im Programmeditor ein:

```
DEFINE DATA
LOCAL

END-DEFINE
*
END
```

`LOCAL` bedeutet, dass die Variablen, die Sie im nächsten Schritt definieren werden, lokale Variablen sind, die nur in diesem Programm zur Verfügung stehen.

Die Datenfelder des DDMs in einem geteilten Bildschirm (Split Screen) anzeigen

1 Geben Sie Folgendes in der Kommandozeile des Programmeditors ein:

```
SPLIT VIEW EMPLOYEES SHORT
```

`SHORT` bedeutet, dass die Datenfelder in der Kurzform aufgelistet werden sollen (das heißt: es werden nur die Adabas-Kurznamen und die entsprechenden Natural-Felder angezeigt).

Der Bildschirm ist jetzt in zwei Teile unterteilt. Die Datenfelder des DDMs werden im unteren Teil des Bildschirms angezeigt. Die Daten im unteren Teil des Bildschirms können nicht editiert werden.

```
>                                      > +  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     O 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 Mit den folgenden Kommandos können Sie den View durchblättern, um zu überprüfen, welche Felder benutzt werden und wie sie definiert sind:

Kommando	Beschreibung
SPLIT + oder S +	Im View vorwärts blättern.
SPLIT - oder S -	Im View rückwärts blättern.
SPLIT . oder S .	Den Split-Screen-Modus beenden.

Im nächsten Schritt wird davon ausgegangen, dass der Split-Screen-Modus beendet wurde.

3 Stellen Sie den Cursor an die erste Position der Zeile, in der LOCAL steht und geben Sie Folgendes ein:

```
.I
```

Im Vollbildmodus werden 9 Leerzeilen eingefügt. Im Split-Screen-Modus wären nur 4 Leerzeilen eingefügt worden.

4 Geben Sie unter LOCAL den folgenden Code ein:

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

5 Drücken Sie EINGABE.

Die übrigen Leerzeilen werden entfernt.

Anmerkung: Die übrigen Leerzeilen werden nicht entfernt, wenn die Standardeinstellung im Editorprofil geändert wurde, das heißt: wenn die Option **Empty Line Suppression** auf "N" gesetzt wurde.

Die erste Zeile enthält den Namen des Views und den Namen der Datenbankdatei, aus der die Felder genommen werden. Dies wird immer auf der ersten Ebene (Level 1) definiert. Der Level wird am Anfang der Zeile definiert. Die Namen der Datenbankfelder aus dem DDM sind auf den Levels 2 und 3 definiert.

Levels werden im Zusammenhang mit Feldgruppierungen benutzt. Felder mit einem Level von 2 oder höher werden als Teil der direkt davor stehenden Gruppe angesehen, die auf einem niedrigeren Level steht. Die Definition einer Gruppe ermöglicht das Referenzieren von mehreren Feldern (dies kann auch nur ein einziges Feld sein), indem man den Gruppennamen benutzt. Dies ist eine komfortable und effiziente Methode zum Referenzieren von mehreren aufeinander folgenden Feldern.

Format und Länge jedes Feldes sind in Klammern angegeben. "A" steht für alphanumerisch und "N" steht für numerisch.

Daten aus einer Datenbank lesen

Jetzt, nachdem Sie alle erforderlichen Daten definiert haben, werden Sie eine READ-Schleife in Ihr Programm einfügen. Hiermit werden die Daten mit Hilfe des definierten Views aus der Datenbankdatei gelesen. Mit jeder Schleife wird ein Mitarbeiter aus der Datenbankdatei gelesen. Name, Abteilung (Department) und die restlichen Urlaubstage (Leave Due) für diesen Mitarbeiter werden angezeigt. Die Daten werden so lange gelesen, bis alle Mitarbeiter angezeigt wurden.

Anmerkung: Es kann passieren, dass eine Fehlermeldung erscheint, die besagt, dass die letzte Transaktion aus der Datenbank zurückgezogen wurde (transaction backed out). Dies passiert in der Regel dann, wenn das Adabas-Zeitlimit für Nichtaktivität überschritten wurde. Wenn ein solcher Fehler auftritt, sollten Sie Ihre letzte Aktion einfach wiederholen (geben Sie zum Beispiel das Kommando RUN noch einmal ein).

Daten aus einer Datenbank lesen

1 Geben Sie folgendes unter END-DEFINE ein (benutzen Sie zum Einfügen von Leerzeilen das Kommando .I wie oben beschrieben):

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

BY NAME bedeutet, dass die aus der Datenbank gelesenen Daten alphabetisch nach den Namen sortiert werden sollen.

Mit dem DISPLAY-Statement wird die Ausgabe im Spaltenformat angeordnet. Für jedes angegebene Feld wird eine Spalte erzeugt und jede Spalte enthält eine Überschrift. 3X bedeutet, dass 3 Leerzeichen zwischen den Spalten eingefügt werden sollen.

2 Führen Sie das Programm mit RUN aus.

Die folgende Ausgabe wird angezeigt.

```
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
```

Das `DISPLAY`-Statement sorgt dafür, dass die Spaltenüberschriften (die aus dem DDM genommen werden) unterstrichen sind und dass sich zwischen der Unterstreichung und den Daten eine Leerzeile befindet. Jede Spalte hat die Breite, die im `DEFINE DATA`-Block definiert wurde (das heißt: die Breite, die im View definiert ist).

Der Titel oben auf jeder Seite (mit Seitennummer, Datum und Uhrzeit) wird ebenfalls vom `DISPLAY`-Statement erzeugt.

3 Drücken Sie wiederholt EINGABE, um alle Seiten nacheinander anzuzeigen.

Sie kehren zum Programmeditor zurück, nachdem alle Mitarbeiter angezeigt wurden.

Tipp: Wenn Sie zum Programmeditor zurückkehren möchten, bevor alle Mitarbeiter angezeigt wurden, geben Sie `EDIT` oder dessen Abkürzung `E` in der `MORE`-Zeile ein. Sie können auch das Terminalkommando `%.` in der `MORE`-Zeile eingeben; dies bricht die aktuelle Natural-Verarbeitung ab. Standardmäßig beginnt jedes Terminalkommando mit einem Prozentzeichen (%) als Steuerzeichen. Es ist jedoch möglich, dass Ihr Administrator ein anderes Steuerzeichen definiert hat.

Bestimmte Daten aus einer Datenbank lesen

Da die Ausgabe im Moment sehr lang ist, werden Sie sie nun eingrenzen. Es sollen nur noch die Daten für den Namenbereich angezeigt werden, der mit "Adkinson" beginnt und mit "Bennett" endet. Diese Namen sind in der Demodatenbank definiert.

Ausgabe auf einen Namensbereich eingrenzen

1 Bevor Sie neue Variablen benutzen können, müssen Sie sie definieren. Geben Sie deshalb Folgendes unter `LOCAL` ein:

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

Dies sind Benutzervariablen; sie sind nicht in der Demodatenbank definiert. Das Rautenzeichen (#) am Anfang des Namens wird dazu benutzt, die Benutzervariablen von den Feldern in der Demodatenbank zu unterscheiden; es ist jedoch nicht zwingend notwendig, dieses Zeichen zu verwenden.

`INIT` definiert den Vorgabewert für ein Feld. Der Vorgabewert muss in spitzen Klammern und Anführungszeichen angegeben werden.

2 Geben Sie Folgendes unter dem `READ`-Statement ein:

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

Ihr Programm sollte nun folgendermaßen aussehen:

```
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
```

Ihr Programmcode ist jetzt länger als eine Bildschirmseite. Um im Programm zu navigieren, können Sie die folgenden Kommandos oder Tasten benutzen:

Kommando	Beschreibung
BOT	Geht an das Ende des Programms.
TOP	Geht an den Anfang des Programms.
Taste	**Beschreibung**
PF8 oder EINGABE	Blättert eine Seite im Programm nach unten.
PF7	Blättert eine Seite im Programm nach oben.

3 Führen Sie das Programm mit RUN aus.

Die Ausgabe wird angezeigt. Wenn Sie wiederholt EINGABE drücken, werden Sie bemerken, dass Sie nach wenigen Seiten zum Programmeditor zurückkehren (d.h. nachdem die Daten für den letzten Mitarbeiter mit dem Namen Bennett angezeigt wurden).

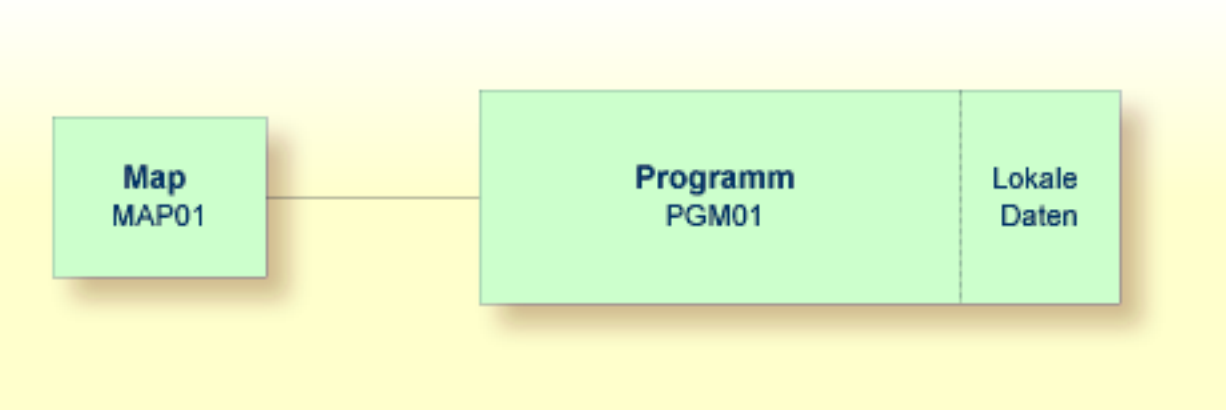
4 Speichern Sie das Programm mit STOW.

Sie können nun mit den nächsten Übungen fortfahren: *Benutzereingaben*.

6 Benutzereingaben

Sie lernen nun, wie Sie einen Benutzer zur Eingabe von Daten auffordern (d.h. der Benutzer soll einen Start- und Endnamen für die Ausgabe eingeben).

Wenn Sie mit den Übungen in diesem Kapitel fertig sind, wird Ihre Beispielanwendung aus den folgenden Modulen bestehen:



Benutzereingaben ermöglichen

Sie werden Ihr Programm jetzt so verändern, dass Eingabefelder für die Start- und Endnamen in der Ausgabe angezeigt werden. Hierzu benutzen Sie das `INPUT`-Statement.

Eingabefelder definieren

1 Geben Sie Folgendes unter `END-DEFINE` ein:

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

Der Session-Parameter `AD` steht für „Attribute Definition“ (Attributdefinition). Sein Wert "M" steht für „Modifiable output field“ (modifizierbares Ausgabefeld) und der Wert "T" steht für „translate lowercase to uppercase“ (Übersetzen von Kleinbuchstaben in Großbuchstaben).

Der Wert "M" in `AD=MT` bedeutet, dass die mit `INIT` definierten Vorgabewerte (d.h.: "ADKINSON" und "BENNETT") in den Eingabefeldern angezeigt werden. Der Benutzer kann andere Werte eingeben. Wenn Sie den Wert "M" weglassen, bleiben die Eingabefelder leer, obwohl Vorgabewerte definiert wurden.

Der Wert "T" in `AD=MT` bedeutet, dass jede Eingabe, die in Kleinbuchstaben gemacht wurde, vor der weiteren Verarbeitung in Großbuchstaben übersetzt wird. Dies ist wichtig, weil die Namen in der Demodatenbank komplett in Großbuchstaben definiert wurden. Wenn Sie den

Wert "T" weglassen, müssen Sie alle Namen komplett in Großbuchstaben angeben. Andernfalls wird der angegebene Name nicht gefunden.

`"Start:"` und `"End:"` sind Textfelder (Bezeichnungen). Sie werden in Anführungszeichen eingegeben.

`#NAME-START` und `#NAME-END` sind Datenfelder (Eingabefelder), in denen der Benutzer den gewünschten Start- und Endnamen eingeben kann.

Der Schrägstrich (/) bedeutet, dass die nachfolgenden Felder in einer neuen Zeile angezeigt werden sollen.

Ihr Programm sollte nun folgendermaßen aussehen:

```
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 Führen Sie das Programm mit `RUN` aus.

Die Ausgabe enthält die Felder, die Sie gerade definiert haben.

```
Start: ADKINSON                                                          ↩

End:   BENNETT
```

3 Behalten Sie die Vorgabewerte bei und drücken Sie EINGABE.

Die Liste der Mitarbeiter wird nun angezeigt.

4 Drücken Sie wiederholt EINGABE bis Sie wieder im Programmeditor sind, oder geben Sie `EDIT` in der `MORE`-Zeile ein.

5 Speichern Sie das Programm mit `STOW`.

Map für die Benutzereingabe gestalten

Sie lernen jetzt eine weitere Möglichkeit kennen, den Benutzer zur Eingabe aufzufordern. Sie werden mit dem Map-Editor eine Map erstellen, die dieselben Felder enthält, die Sie vorher in Ihrem Programm definiert haben. Eine Map ist ein separates Objekt; sie wird benutzt, um das Layout der Benutzeroberfläche von der Geschäftslogik der Anwendung zu trennen.

Die Map, die Sie jetzt erstellen werden, wird folgendermaßen aussehen:

```
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   ↩
```

Die erste Zeile der Map enthält Systemvariablen für das aktuelle Datum und die aktuelle Uhrzeit. Es gibt zwei Datenfelder (Eingabefelder), in denen der Benutzer einen Startnamen und einen Endnamen eingeben kann. Vor den Datenfeldern befinden sich Textfelder (Bezeichnungen).

Die Länge eines Feldes wird durch eine Reihe von "X"-Zeichen dargestellt. Für die Unterscheidung der verschiedenen Feldtypen werden Delimiter-Zeichen benutzt. In unserer Beispiel-Map werden die folgenden Delimiter-Zeichen verwendet:

Delimiter	Feldtyp
(	Systemvariable.
:	Datenfeld.

Für die oben gezeigte Map sind die folgenden Schritte erforderlich:

- Map erstellen
- Textfelder definieren
- Datenfelder definieren
- Namen für Datenfelder definieren
- Systemvariablen einfügen
- Felder verschieben
- Map testen
- Map mit STOW speichern

Map erstellen

Sie werden jetzt den Map-Editor aufrufen, mit dem Sie das Layout Ihrer Map gestalten. Der Map-Editor kann über das Menü **Edit Map** aufgerufen werden.

Das Menü Edit Map aufrufen

1 Geben Sie in der Kommandozeile des Programmeditors einen Punkt (.) ein, um zum Menü **Development Functions** zurückzukehren.

2 Geben Sie unten im Menü **Development Functions** die folgenden Informationen ein und drücken Sie EINGABE:

```
    Code .. E    Type .. M   Name .. ________________________________



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

"E" steht für die Funktion **Edit Object** (Objekt editieren), und "M" steht für den Objekttyp Map.

Anmerkung: Es wäre auch möglich gewesen, den Code "C" für **Create Object** (Objekt erstellen) einzugeben. In diesem Fall wären Sie aber dazu aufgefordert worden, einen Objektnamen anzugeben.

Das Menü **Edit Map** erscheint.

```
16:43:41                ***** NATURAL MAP EDITOR *****              2009-06-30
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                                          ↩
 ↩
```

Eine neue Map initialisieren

1 Geben Sie unten im Menü **Edit Map** die folgenden Informationen ein und drücken Sie EINGABE:

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

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

Der Bildschirm **Define Map Settings** erscheint.

```
16:43:42              Define Map Settings for MAP                  2012-07-17

 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
```

In diesem Bildschirm definieren Sie die Standardeinstellungen einer Map (zum Beispiel, die Größe der Map). Es ist auch möglich, die Delimiter-Zeichen in diesem Bildschirm zu ändern. Für dieses Tutorial werden Sie jedoch die Standard-Delimiter-Zeichen verwenden. Sie werden nur die Füllzeichen (Filler Characters) definieren (siehe den nächsten Schritt).

Ein Delimiter-Zeichen steht für eine Kombination aus Klasse und Attribut, die einem Feld zugeordnet ist. In diesem Tutorial werden die fett dargestellten Delimiter-Zeichen verwendet:

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

- Der Doppelpunkt bedeutet, dass ein Feld ein modifizierbares Ein- und Ausgabefeld ist ("M" in der Spalte **Cls**) und dass es hervorgehoben (intensiviert) ist ("I" in der Spalte **Att**).
- Die öffnende Klammer bedeutet, dass ein Feld nur zur Ausgabe (Output) dient ("O" in der Spalte **Cls**) und dass es hervorgehoben (intensiviert) ist ("I" in der Spalte **Att**).

- Wenn kein Delimiter-Zeichen benutzt wird (dies ist im Bildschirm mit "BLANK" angegeben), so bedeutet das, dass das Feld eine Textkonstante ("T" in der Spalte **Cls**) mit Standardattributen (Default Attributes) ist ("D" in der Spalte **Att**).

2 Geben Sie für jede Füllzeichenoption einen Unterstrich (_) ein, so wie es unten dargestellt ist:

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

Füllzeichen werden in der Map benutzt, um leere Positionen in den Eingabefeldern zu füllen, damit der Benutzer bei der Dateneingabe die exakte Position und Länge eines Feldes erkennen kann.

3 Drücken Sie EINGABE um die Änderungen zu speichern.

4 Drücken Sie noch einmal EINGABE, um den Editierbereich der Map aufzurufen.

Der Editierbereich der Map wird im Split-Screen-Modus angezeigt.

```
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   ↩
 ↩
```

Anmerkung: Der Bildschirm **Define Map Settings** erscheint nur, wenn Sie eine neue Map initialisieren. Wenn Sie eine Map editieren, können Sie den Bildschirm **Define**

Map Settings aufrufen, indem Sie im Editierbereich der Map (siehe den Bildschirm oben) PF2 (Mset) drücken.

Im Split-Screen-Modus werden in der oberen Bildschirmhälfte die gültigen Delimiter-Zeichen angezeigt. In der unteren Bildschirmhälfte befindet sich der Editierbereich, in dem Sie die Map erstellen werden.

Anders als im Programmeditor und Data-Area-Editor (der später in diesem Tutorial beschrieben wird), gibt es beim Map-Editor keine Kommandozeile oder Aufforderung für die Eingabe von Natural-Systemkommandos. Viele Funktionen im Map-Editor werden mit Zeilen- oder Feldkommandos ausgeführt (siehe unten) oder mit PF-Tasten.

Mit PF9 schalten Sie um zwischen Split-Screen-Modus und ungeteiltem Bildschirm, in dem der Editierbereich in voller Größe angezeigt wird.

Textfelder definieren

Sie werden jetzt zwei Textfelder (auch Konstanten oder Bezeichnungen genannt) in der Map definieren.

Textfelder definieren

1 Stellen Sie den Cursor im Editierbereich der Map auf die erste Position der vierten Zeile und geben Sie Folgendes ein:

```
Start
```

2 Stellen Sie den Cursor auf die erste Position der nächsten Zeile und geben Sie Folgendes ein:

```
End
```

Ihre Map sollte nun folgendermaßen aussehen:

```
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   ↩
 ↩
```

Anmerkung: Wenn Sie Ihre letzten Änderungen rückgängig machen wollen, drücken Sie PF12 bevor Sie EINGABE drücken.

Datenfelder definieren

Sie werden jetzt zwei Datenfelder in der Map definieren. Dies sind die Eingabefelder, in denen der Benutzer den Start- und Endnamen eingeben kann.

Sie können die Datenfelder auf zwei verschiedene Arten definieren: auf die herkömmliche Art, wobei es in Ihrer Verantwortung liegt, die Länge des Datenfelds korrekt anzugeben, oder auf eine benutzerfreundliche Art, wobei Sie das Datenfeld einfach aus einer Liste auswählen und die Länge bereits korrekt definiert ist. Diese beiden Arten sind unten beschrieben.

Datenfeld definieren und dabei die Länge selbst angeben

1 Geben Sie Folgendes hinter dem Textfeld **Start** ein (lassen Sie ein Leerzeichen zwischen dem Textfeld und dem Datenfeld):

```
:X(20)
```

Der Doppelpunkt (:) ist das Delimiter-Zeichen, das das Datenfeld als modifizierbar und intensiviert kennzeichnet. Das Datenfeld wird mit einer Länge von 20 Zeichen definiert. Die Länge des Feldes wird durch die "X"-Zeichen dargestellt.

2 Drücken Sie EINGABE.

Ihre Map sollte nun folgendermaßen aussehen:

```
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   ↩
 ↩
```

Datenfeld aus einer Liste auswählen

1 Geben Sie Folgendes im Feld **Ob** ein (dieses Feld befindet sich oben links im Bildschirm) und drücken Sie EINGABE:

```
P PGM01
```

Die Datenfelder, die zurzeit vom Programm `PGM01` benutzt werden, werden jetzt im Bildschirm angezeigt. Vor den Feldern, die in der Map benutzt werden können, steht eine Nummer.

```
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   ↩
 ↩
```

Nicht alle im Programm PGM01 definierten Datenfelder werden im Bildschirm angezeigt. Um die Liste der Datenfelder durchzublättern, geben Sie eines der folgendes Kommandos im Feld **Ob** ein (d.h. im dem Feld, das zurzeit den Buchstaben "P" enthält).

Kommando	Beschreibung
+	In der Liste nach vorne blättern.
-	In der Liste nach hinten blättern.
++	An das Ende der Liste gehen.
--	An den Anfang der Liste gehen.

2 Geben Sie Folgendes hinter dem Textfeld **End** ein (lassen Sie ein Leerzeichen zwischen dem Textfeld und dem Datenfeld) und drücken Sie EINGABE:

```
:2
```

Der Doppelpunkt (:) ist das Delimiter-Zeichen, das das Datenfeld als modifizierbar und intensiviert kennzeichnet. Die Nummer 2 ist #NAME-END zugeordnet.

Das Datenfeld ist automatisch mit der korrekten Länge definiert (20 Zeichen in diesem Fall). Die Länge des Feldes wird durch die "X"-Zeichen dargestellt.

Namen für Datenfelder definieren

Folgendes gilt nur für das Datenfeld für den Startnamen, das Sie manuell definiert haben. Es gilt nicht für das Datenfeld für den Endnamen, das Sie aus einer Liste ausgewählt haben: Wenn Sie ein neues Datenfeld für eine Benutzervariable erstellen, wird von Natural ein Name für dieses Feld vergeben. Dieser Feldname enthält eine Nummer. Sie müssen die Namen der neu erstellten Felder an die Namen anpassen, die in Ihrem Programm definiert wurden.

Sie werden jetzt dafür sorgen, dass dieselben Namen wie in Ihrem Programm benutzt werden: `#NAME-START` und `#NAME-END`. Die Ausgabe dieser Felder (d.h. die Benutzereingabe) wird an die entsprechenden Benutzervariablen in Ihrem Programm übergeben.

Namen für Datenfelder definieren

1 Geben Sie Folgendes ab der ersten Position des Datenfeldes für den Startnamen ein und drücken Sie EINGABE:

```
.E
```

Oder:

Stellen Sie den Cursor im Datenfeld an eine beliebige Position und drücken Sie PF5.

Für das markierte Feld werden erweiterte Feldinformationen angezeigt:

```
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 ↩
```

Das Feld **Fld** in der oberen linken Bildschirmecke enthält den Feldnamen, der von Natural vergeben wurde: "#001".

2 Geben Sie "#NAME-START" im Feld **Fld** ein.

3 Drücken Sie PF3, um die Funktion zu beenden.

Da das Datenfeld für den Endname in der vorherigen Übung aus einer Liste ausgewählt wurde, ist es nicht erforderlich, die Schritte oben für den Endnamen zu wiederholen. In diesem Fall ist #NAME-END bereits definiert. Wenn Sie möchten, können Sie dies mit dem Kommando .E wie oben beschrieben überprüfen.

Anmerkung: Bei #NAME-END hat der Session-Parameter AD den zusätzlichen Wert "L", der bei #NAME-START nicht definiert ist. "L" bedeutet, dass der Wert dieses Feldes linksbündig angezeigt wird. Da dies bei alphanumerischen Feldern die Vorgabe ist, ist es nicht nötig, dies für #NAME-START zu definieren.

Systemvariablen einfügen

Natural-Systemvariablen enthalten Information über die aktuelle Natural-Session, wie zum Beispiel die aktuelle Library, Benutzer, oder Datum und Uhrzeit. Diese Informationen können an jedem beliebigen Punkt in einem Natural-Programm referenziert werden. Alle Systemvariablen beginnen mit einem Stern (*).

Sie werden jetzt Systemvariablen für Datum und Uhrzeit in der Map einfügen. Wenn das Programm mit RUN ausgeführt wird, werden das aktuelle Datum und die aktuelle Uhrzeit in der Map angezeigt.

Systemvariablen einfügen

1 Stellen Sie den Cursor an die erste Position der ersten Zeile und geben Sie Folgendes ein:

```
(*DAT4I
```

Die öffnende Klammer ist das Delimiter-Zeichen, das die Systemvariable als intensiviertes Ausgabefeld kennzeichnet.

2 Stellen Sie den Cursor an die erste Position der zweiten Zeile und geben Sie Folgendes ein:

```
(*TIMX
```

Ihre Map sollte nun folgendermaßen aussehen:

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

Start :XXXXXXXXXXXXXXXXXXX
End :XXXXXXXXXXXXXXXXXXX                                                    ↩
 ↩
```

3 Drücken Sie EINGABE.

"X"-Zeichen werden jetzt statt der Systemvariablennamen angezeigt.

Felder verschieben

Sie werden die Felder, die Sie eingegeben haben, jetzt mit Zeilenkommandos und Feldkommandos verschieben.

Ein Feld verschieben

1 Geben Sie das folgende Feldkommando beginnend in der ersten Position der Systemvariablen in der zweiten Zeile ein:

```
.M
```

Beispiel:

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

Start :XXXXXXXXXXXXXXXXXXX
End :XXXXXXXXXXXXXXXXXXX                                                    ↩
 ↩
```

Ein Feldkommando wird am Anfang eines Feldes eingegeben. Es gilt nur für das Feld, in dem Sie es eingeben.

2 Stellen Sie den Cursor an die Position, an die Sie die Systemvariable verschieben möchten (Spalte 70 in der ersten Zeile).

3 Drücken Sie EINGABE.

Die Systemvariable wird an die Cursorposition verschoben.

Leerzeile einfügen

1 Geben Sie das folgende Zeilenkommando beginnend in der ersten Position der vierten Zeile (Startname) ein:

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

Beispiel:

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


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

Ein Zeilenkommando wird am Anfang einer Zeile eingegeben. Es gilt für die komplette Zeile, in der Sie es eingeben.

2 Drücken Sie EINGABE.

Eine Leerzeile wird unter der Zeile eingefügt, in der Sie das Zeilenkommando eingegeben haben.

Zeile zentrieren

1 Geben Sie das folgende Zeilenkommando beginnend in der ersten Position der vierten Zeile (Startname) ein:

```
..C
```

Beispiel:

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


..Crt :XXXXXXXXXXXXXXXXXXX

End :XXXXXXXXXXXXXXXXXXX ↩
```

2 Drücken Sie EINGABE.

Die Zeile wird zentriert.

Mehrere Felder verschieben

1 Geben Sie das folgende Feldkommando beginnend in der ersten Position des Textfeldes in der sechsten Zeile (**End**) ein und in der ersten Position des Datenfeldes in dieser Zeile:

```
.M
```

Beispiel:

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


                          Start :XXXXXXXXXXXXXXXXXXX

.Md .MXXXXXXXXXXXXXXXXXXX   ↩
```

2 Stellen Sie den Cursor an die Position, an der das Textfeld beginnen soll (Spalte 30 in der sechsten Zeile).

3 Drücken Sie EINGABE.

Beide Felder werden verschoben.

Die Map sollte jetzt so aussehen wie am Anfang dieses Abschnitts.

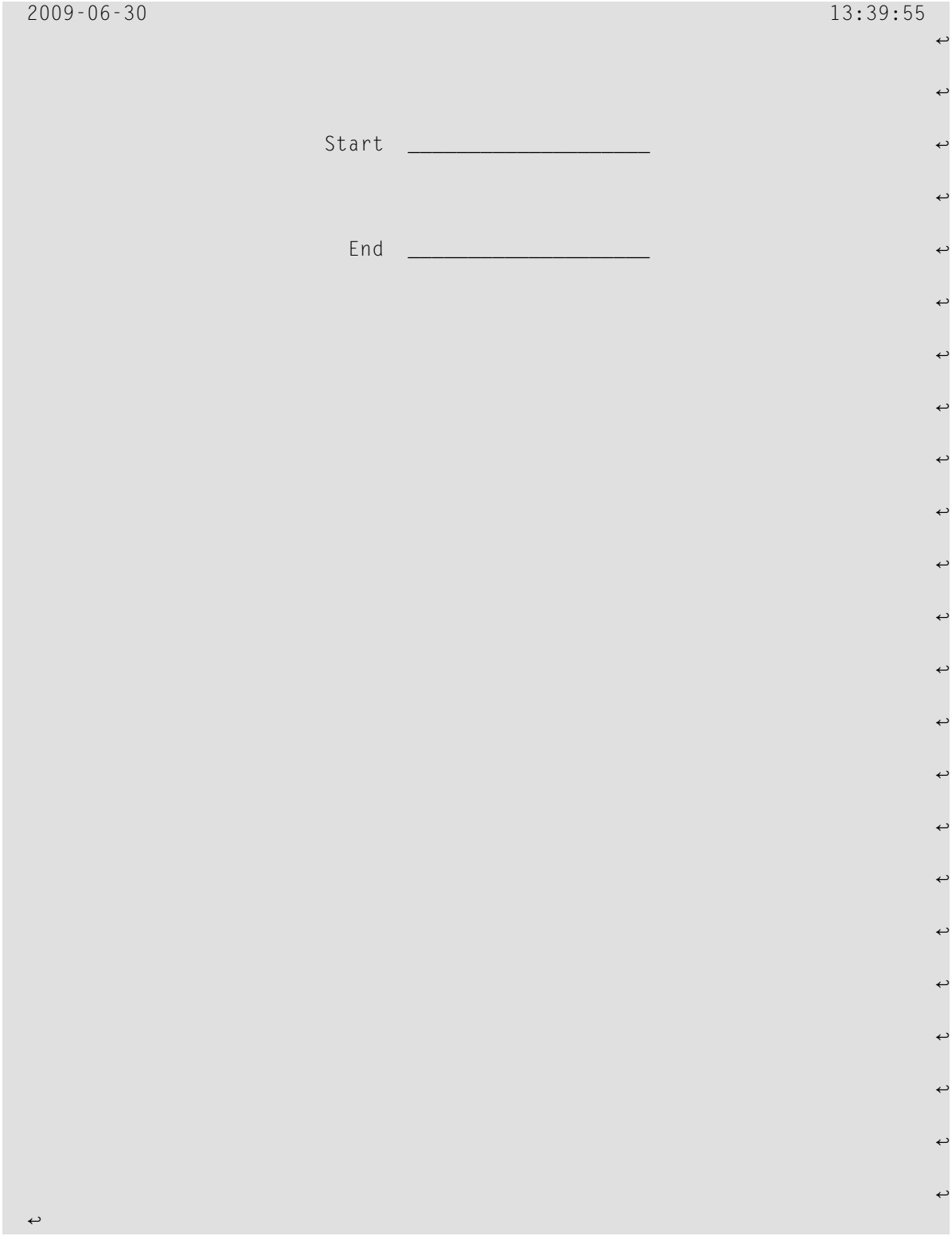
Map testen

Um zu überprüfen, ob Ihre Map wie beabsichtigt funktioniert, werden Sie sie jetzt testen.

Map testen

1 Drücken Sie PF4.

Die folgende Ausgabe wird angezeigt.



Das Eingabefeld für den Startnamen ist automatisch markiert, da es das erste Eingabefeld in der Map ist. Beide Eingabefelder enthalten Füllzeichen.

Anmerkung: Wenn der Einfügemodus aktiv ist, muss der Benutzer die Füllzeichen löschen, damit die Eingabe von Text möglich ist. Im Überschreibemodus, der standardmäßig aktiv ist, ist dies nicht erforderlich.

2 Drücken Sie EINGABE, um zum Map-Editor zurückzukehren.

Map mit STOW speichern

Wenn die Map erfolgreich getestet wurde, müssen Sie sie mit `STOW` speichern, damit sie von Ihrem Programm gefunden werden kann.

Map mit `STOW` speichern

1 Drücken Sie PF3, um zum Menü **Edit Map** zurückzukehren.

2 Geben Sie Folgendes im Feld **Code** ein und drücken Sie EINGABE:

```
W
```

Map aus dem Programm aufrufen

Sobald eine Map mit `STOW` gespeichert wurde, kann Sie mit einem `WRITE`- oder `INPUT`-Statement aus einem Natural-Programm heraus aufgerufen werden.

Map aus dem Programm aufrufen

1 Kehren Sie zum Programmeditor zurück, indem Sie eines der folgenden Kommandos in der Kommandozeile des Menüs **Edit Map** eingeben.

```
EDIT PGM01
```

```
E PGM01
```

2 Ersetzen Sie die vorher definierten `INPUT`-Zeilen durch die folgende Zeile:

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

Hiermit wird die von Ihnen erstellte Map aufgerufen.

Der Name der Map muss in einfache Anführungszeichen gesetzt werden, um die Map von einer Benutzervariablen zu unterscheiden.

Ihr Programm sollte nun folgendermaßen aussehen:

```
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 Führen Sie das Programm mit `RUN` aus.

Ihre Map wird nun angezeigt.

4 Drücken Sie wiederholt EINGABE bis Sie zum Programmeditor zurückkehren, oder geben Sie `EDIT` in der `MORE`-Zeile ein.

5 Speichern Sie das Programm mit `STOW`.

Immer einen Endnamen benutzen

So wie Ihr Programm jetzt kodiert ist, werden keine Daten gefunden, wenn kein Endname angegeben wird.

Sie werden jetzt die Anfangswerte für Start- und Endname entfernen; danach müssen diese Namen immer vom Benutzer angegeben werden. Um zu gewährleisten, dass ein Endname immer benutzt wird, auch wenn er vom Benutzer nicht eingegeben wurde, werden Sie Ihrem Programm nun ein entsprechendes Statement hinzufügen.

Den Endnamen benutzen

1 Gehen Sie zum `DEFINE DATA`-Block und entfernen Sie die Anfangswerte (`INIT`) für die Felder `#NAME-START` und `#NAME-END`, so dass die entsprechenden Zeilen folgendermaßen aussehen:

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

2 Geben Sie Folgendes unter `INPUT USING MAP 'MAP01'` ein:

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

Wenn das Feld `#NAME-END` leer ist (d.h. wenn der Benutzer keinen Endnamen angibt), wird der Startname automatisch als Endname benutzt.

Anmerkung: Statt des Statements `MOVE #NAME-START TO #NAME-END` können Sie auch die folgende Variante des `ASSIGN`- oder `COMPUTE`-Statements benutzen: `#NAME-END := #NAME-START`.

Ihr Programm sollte nun folgendermaßen aussehen:

```
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 Führen Sie das Programm mit `RUN` aus.

4 Geben Sie in der daraufhin erscheinenden Map "JONES" in dem Feld für den Startnamen ein und drücken Sie EINGABE.

In der daraufhin erscheinenden Liste werden jetzt nur die Mitarbeiter mit dem Namen "Jones" angezeigt.

5 Drücken Sie EINGABE, um zum Programmeditor zurückzukehren.

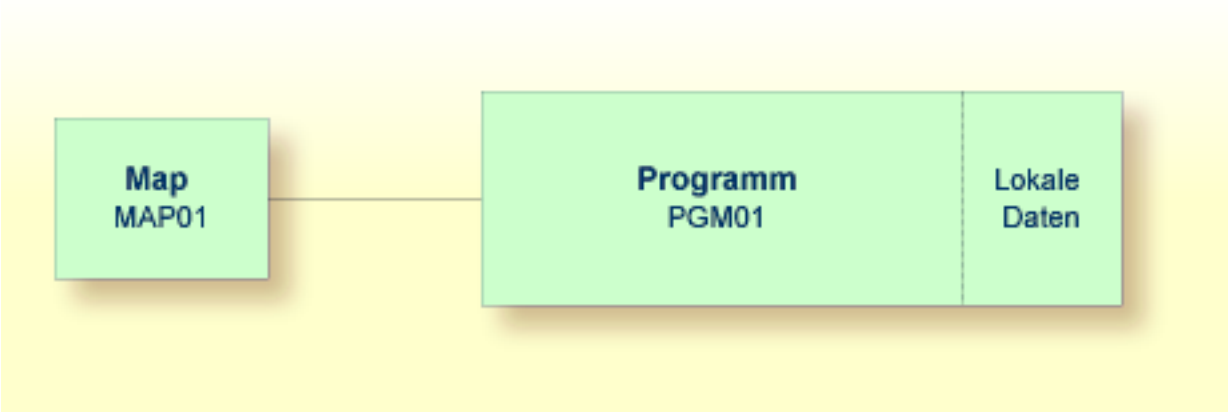
6 Speichern Sie das Programm mit `STOW`.

Sie können nun mit den nächsten Übungen fortfahren: ***Verarbeitungsschleifen und Labels***.

7 Verarbeitungsschleifen und Labels

Sie werden Ihr Programm jetzt erweitern, indem Sie Verarbeitungsschleifen und Labels hinzufügen.

Wenn Sie mit den Übungen in diesem Kapitel fertig sind, wird Ihre Beispielanwendung immer noch aus denselben Modulen wie im vorherigen Kapitel bestehen:



Wiederholte Benutzung erlauben

So wie Ihr Programm jetzt aufgebaut ist, wird es sofort nach dem Anzeigen der Map und dem Anzeigen der Mitarbeiterliste beendet. Damit der Benutzer sofort wieder eine neue Mitarbeiterliste anzeigen kann, ohne das Programm neu zu starten, werden Sie den entsprechenden Programmcode nun in eine `REPEAT`-Schleife setzen.

Verarbeitungsschleife definieren

1 Geben Sie Folgendes unter `END-DEFINE` ein:

```
RP1. REPEAT
```

`REPEAT` definiert den Beginn der Verarbeitungsschleife. `RP1.` ist ein Label, das beim Verlassen der Verarbeitungsschleife benutzt wird (dies wird weiter unten definiert).

2 Definieren Sie das Ende der Verarbeitungsschleife, indem Sie Folgendes vor dem `END`-Statement eingeben:

```
END-REPEAT
```

3 Geben Sie Folgendes unter `INPUT USING MAP 'MAP01'` ein:

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

Das `IF`-Statement, das mit `END-IF` beendet werden muss, prüft den Inhalt des Feldes `#NAME-START`. Wenn ein Punkt (.) in diesem Feld eingeben wird, wird das Statement `ESCAPE BOTTOM` zum Verlassen der Schleife benutzt. Die Verarbeitung wird mit dem ersten Statement nach der Schleife fortgesetzt (in diesem Fall ist das `END`).

Indem Sie einer Schleife ein Label zuordnen (hier ist es `RP1.`), können Sie sich mit dem Statement `ESCAPE BOTTOM` auf diese Schleife beziehen. Da Schleifen geschachtelt werden können, sollten Sie angeben, welche Schleife Sie verlassen möchten. Andernfalls wird nur die innerste aktive Schleife verlassen.

Ihr Programm sollte nun folgendermaßen aussehen:

```
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
```

Anmerkung: Zur besseren Lesbarkeit wurde der Inhalt der REPEAT-Schleife eingerückt.

4 Führen Sie das Programm mit RUN aus.

5 Geben Sie in der daraufhin erscheinenden Map "JONES" in dem Feld für den Startnamen ein und drücken Sie EINGABE.

In der daraufhin erscheinenden Liste werden nur die Mitarbeiter mit dem Namen "Jones" angezeigt. Drücken Sie EINGABE. Wegen der REPEAT-Schleife wird die Map wieder angezeigt. Sie können nun sehen, dass "JONES" als Endname eingetragen wurde.

6 Geben Sie einen Punkt (.) in dem Feld für den Startnamen ein und drücken Sie EINGABE, um die Map zu verlassen. Vergessen Sie nicht, die restlichen Zeichen des Namens zu löschen, der noch in diesem Feld angezeigt wird.

7 Speichern Sie das Programm mit STOW.

Meldung für nicht gefundene Informationen anzeigen

Sie werden jetzt die Meldung definieren, die angezeigt werden soll, wenn der Benutzer einen Startnamen angibt, der in der Datenbank nicht gefunden werden kann.

Meldung für nicht gefundene Mitarbeiter definieren

1 Geben Sie das Label RD1. am Anfang der Zeile ein, die das READ-Statement enthält:

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

2 Geben Sie Folgendes unter END-READ ein:

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

Mit der Systemvariablen *COUNTER wird die Anzahl der Datensätze geprüft, die in der READ-Schleife gefunden werden. Wenn der Inhalt gleich 0 ist (d.h. wenn ein Mitarbeiter mit dem angegebenen Namen nicht gefunden wurde), wird die Meldung, die mit dem REINPUT-Statement definiert wurde, unten in der Map angezeigt.

Um die READ-Schleife zu bezeichnen, geben Sie ihr ein Label (in diesem Fall ist es RD1.). Da ein komplexes Datenbankzugriffsprogramm viele Schleifen enthalten kann, müssen Sie angeben, auf welche Schleife Sie sich beziehen.

Ihr Programm sollte nun folgendermaßen aussehen:

```
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 Führen Sie das Programm mit RUN aus.

4 Geben Sie in der daraufhin erscheinenden Map einen Startnamen ein, der nicht in der Demodatenbank definiert ist (zum Beispiel "XYZ") und drücken Sie EINGABE.

Ihre Meldung sollte jetzt in der Map angezeigt werden.

5 Geben Sie einen Punkt (.) in dem Feld für den Startnamen ein und drücken Sie EINGABE, um die Map zu verlassen. Vergessen Sie nicht, die restlichen Zeichen des Namens zu löschen, der noch in diesem Feld angezeigt wird.

6 Speichern Sie das Programm mit `STOW`.

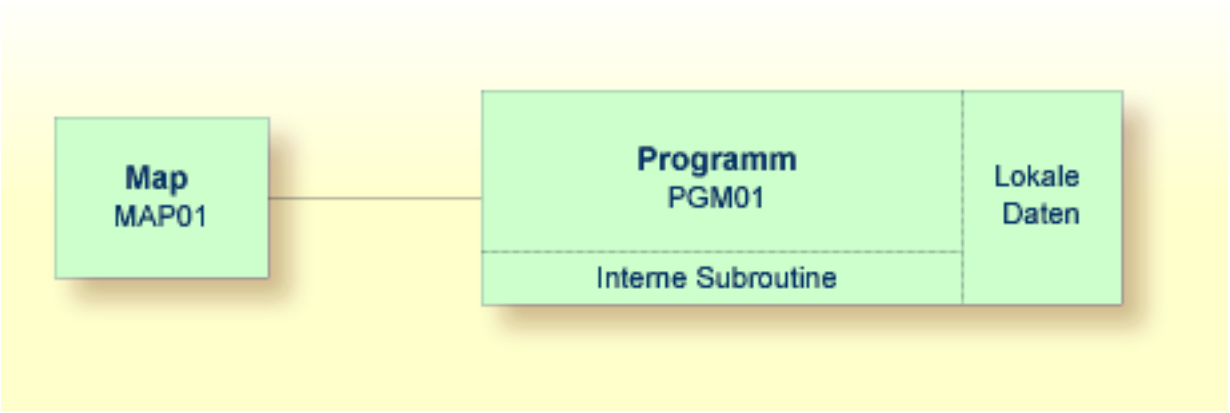
Sie können nun mit den nächsten Übungen fortfahren: ***Interne Subroutinen***.

8 Interne Subroutinen

Natural unterscheidet zwei Arten von Subroutinen: interne Subroutinen, die direkt im Programm definiert werden, und externe Subroutinen, die als separate Objekte außerhalb des Programms gespeichert werden (dies wird später in diesem Tutorial beschrieben).

Sie werden jetzt eine interne Subroutine in Ihrem Programm definieren, mit der ein Stern (*) in die neue Benutzervariable #MARK geschrieben wird. Diese Subroutine wird aufgerufen, wenn ein Mitarbeiter 20 oder mehr Urlaubstage hat.

Wenn Sie mit den Übungen in diesem Kapitel fertig sind, wird Ihre Beispielanwendung folgendermaßen strukturiert sein:



Interne Subroutine definieren

Sie werden jetzt die interne Subroutine in Ihr Programm einfügen.

Interne Subroutine definieren

1 Geben Sie Folgendes unter der Benutzervariablen #NAME-END ein:

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

Diese Variable wird von der Subroutine benutzt. Daher muss Sie zuerst definiert werden.

2 Um die Subroutine zu definieren, geben Sie Folgendes vor dem END-Statement ein:

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

Wenn diese Subroutine ausgeführt wird, schreibt sie einen Stern (*) in die Benutzervariable #MARK.

Anmerkung: Statt des Statements `MOVE '*' TO #MARK` können Sie auch die folgende Variante des `ASSIGN`- oder `COMPUTE`-Statements benutzen: `#MARK := '*'`.

3 Ändern Sie das `DISPLAY`-Statement wie folgt:

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

Hiermit wird eine neue Spalte in der Ausgabe angezeigt. Die Überschrift dieser Spalte ist ">=20". Die Spalte enthält einen Stern (*), wenn der betreffende Mitarbeiter 20 oder mehr Urlaubstage hat.

Interne Subroutine ausführen

Jetzt, nachdem Sie die interne Subroutine definiert haben, können Sie den entsprechenden Code zum Aufruf dieser Subroutine eingeben.

Interne Subroutine ausführen

1 Geben Sie Folgendes vor dem `DISPLAY`-Statement ein:

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

Wenn ein Mitarbeiter gefunden wird, der 20 oder mehr Urlaubstage (`LEAVE-DUE`) hat, wird die neue Subroutine mit dem Namen `MARK-SPECIAL-EMPLOYEES` ausgeführt. Wenn ein Mitarbeiter weniger als 20 Urlaubstage hat, wird der Inhalt von #MARK zurückgesetzt.

Ihr Programm sollte nun folgendermaßen aussehen:

```
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 Führen Sie das Programm mit `RUN` aus.

3 Geben Sie in der daraufhin erscheinenden Map "JONES" ein und drücken Sie EINGABE.

Die Liste der Mitarbeiter sollte jetzt eine zusätzliche Spalte enthalten.

4 Geben Sie `EDIT` in der `MORE`-Zeile ein, um zum Programmeditor zurückzukehren.

5 Speichern Sie das Programm mit `STOW`.

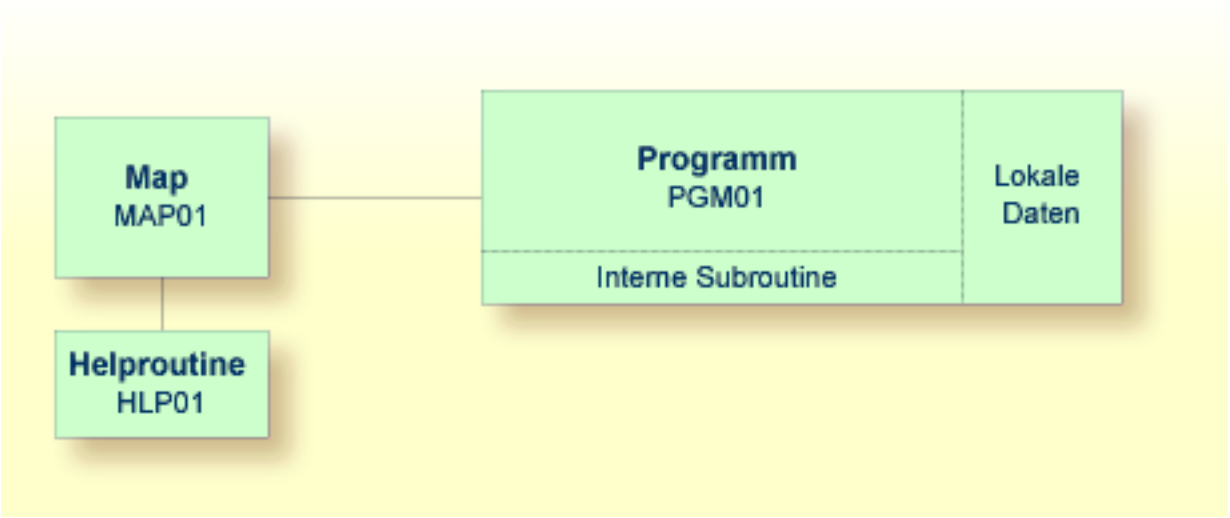
6 Geben Sie einen Punkt (.) in der Kommandozeile ein, um zum Menü **Development Functions** zurückzukehren.

Sie können nun mit den nächsten Übungen fortfahren: ***Verarbeitungsregeln und Helproutinen***.

9 Verarbeitungsregeln und Helproutinen

Verarbeitungsregeln und Helproutinen werden für die Felder in einer Map definiert.

Wenn Sie mit den Übungen in diesem Kapitel fertig sind, wird Ihre Beispielanwendung aus den folgenden Modulen bestehen (eine Verarbeitungsregel kann nicht als separates Modul definiert werden; sie ist immer Teil einer Map):



Verarbeitungsregel definieren

Sie werden jetzt die Meldung definieren, die angezeigt werden soll, wenn der Benutzer EINGABE drückt ohne vorher einen Startnamen anzugeben.

Verarbeitungsregel definieren

1 Kehren Sie zum Map-Editor zurück, indem Sie Folgendes im Menü **Development Functions** eingeben.

```
    Code .. E    Type .. _   Name .. MAP01_____________________________



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

2 Geben Sie Folgendes ab der ersten Position des Datenfeldes für den Startnamen ein:

```
.P
```

Beispiel:

```
                          Start .PXXXXXXXXXXXXXXXXXX

                            End :XXXXXXXXXXXXXXXXXXX

↩
```

3 Drücken Sie EINGABE

Für das markierte Feld wird der folgende Bildschirm angezeigt:

```
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 Geben Sie die folgende Verarbeitungsregel ein:

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

Das Kaufmanns-Und (&) in der Verarbeitungsregel wird bei der Ausführung des Programms dynamisch mit dem Namen des Feldes ersetzt. In diesem Fall wird es mit #NAME-START ersetzt. Wenn #NAME-START leer ist, wird die Meldung angezeigt, die mit dem REINPUT-Statement definiert wurde.

MARK ist eine Option des REINPUT-Statements. Die Syntax ist MARK *fieldname*.

MARK definiert das Feld, in das der Cursor gestellt werden soll, wenn das REINPUT-Statement ausgeführt wird. In diesem Fall wird der Cursor in das Feld #NAME-START gestellt.

5 Geben Sie im Feld **Rank** (Rang) den Wert "1" ein.

```
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
```

Mit dem Rang wird die Reihenfolge definiert, in der die Regeln für die verschiedenen Felder verarbeitet werden. Alle Regeln mit Rang 1 werden zuerst verarbeitet, gefolgt von denen mit Rang 2, usw.

6 Drücken Sie EINGABE, um Ihre Eingabe zu speichern. Drücken Sie dann PF3, um zur Map zurückzukehren.

Anmerkung: Wenn Sie Ihre Verarbeitungsregel noch einmal anzeigen möchten, müssen Sie eines der folgenden Kommandos benutzen: .P1 (zeigt alle Regeln mit Rang 1) oder .P* (zeigt eine Liste mit allen Regeln für dieses Feld).

7 Testen Sie die Map.

8 Geben Sie in der daraufhin erscheinenden Ausgabe einen beliebigen Startnamen ein und drücken Sie EINGABE.

Der Ausgabebildschirm wird geschlossen.

9 Testen Sie die Map noch einmal. Geben Sie dieses Mal keinen Namen ein und drücken Sie EINGABE.

Die Meldung, die mit der Verarbeitungsregel definiert wurde, sollte jetzt in der Map erscheinen.

10 Geben Sie einen Punkt (.) in dem Feld für den Startnamen ein und drücken Sie EINGABE, um den Ausgabebildschirm zu verlassen.

11 Speichern Sie die Map mit `STOW` (drücken Sie PF3, um zum Menü **Edit Map** zurückzukehren und geben Sie "W" im Feld **Code** ein).

Helproutine definieren

Eine Helproutine wird ausgeführt, wenn der Benutzer die Hilfetaste drückt, während der Cursor in dem Eingabefeld für den Startnamen steht.

Sie werden zuerst die Helproutine definieren und sie dann mit einem bestimmten Feld verknüpfen.

Helproutine definieren

1 Drücken Sie PF3 im Menü **Edit Map**, um zum Menü **Development Functions** zurückzukehren.

2 Geben Sie unten im Menü **Development Functions** die folgenden Informationen ein und drücken Sie EINGABE:

```
    Code .. C    Type .. H   Name .. HLP01_____________________________



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

"C" steht für die Funktion **Create Object** (Objekt erstellen), "H" steht für den Objekttyp Helproutine, und "HLP01" ist der Name für Ihre neue Helproutine.

Ein leerer Editor erscheint.

3 Geben Sie Folgendes ein:

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

4 Speichern Sie die Helproutine mit `STOW`.

Helproutine mit einem Feld in der Map verknüpfen

1 Kehren Sie zurück zum Map-Editor, indem Sie Folgendes in der Kommandozeile des Bildschirms eingeben, in dem Sie eben die Helproutine eingegeben haben.

```
E MAP01
```

2 Geben Sie Folgendes ab der ersten Position des Datenfeldes für den Startnamen ein und drücken Sie EINGABE:

```
.E
```

Oder:

Stellen Sie den Cursor an eine beliebige Stelle im Datenfeld und drücken Sie PF5.

Erweiterte Feldinformationen werden für das Feld angezeigt.

3 Geben Sie im Feld **HE** den Namen "'HLP01'" ein (einschließlich der einfachen Anführungszeichen).

Dies ist der Name, unter dem Sie die Helproutine gespeichert haben.

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

4 Drücken Sie PF3, um die erweiterten Feldinformationen zu verlassen.

5 Testen Sie die Map.

6 Geben Sie in der daraufhin erscheinenden Ausgabe ein Fragezeichen (?) im Eingabefeld für den Startnamen ein und drücken Sie EINGABE.

Der von Ihnen definierte Hilfetext wird jetzt angezeigt.

7 Drücken Sie EINGABE, um zur Map zurückzukehren.

8 Geben Sie einen Punkt (.) in dem Feld für den Startnamen ein und drücken Sie EINGABE, um die Map zu verlassen.

9 Speichern Sie die Map mit `STOW` (drücken Sie PF3, um zum Menü **Edit Map** zurückzukehren und geben Sie "W" im Feld **Code** ein).

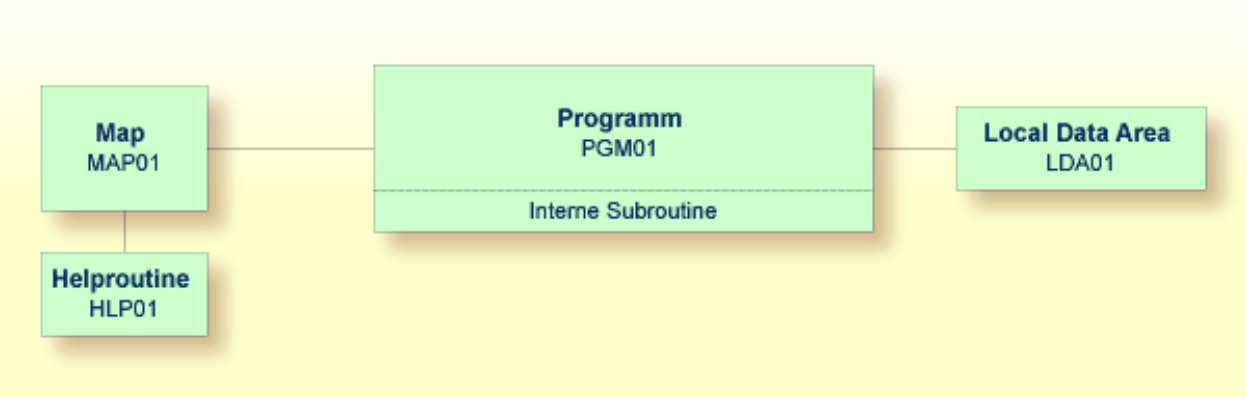
10 Drücken Sie PF3, um zum Menü **Development Functions** zurückzukehren.

Sie können nun mit den nächsten Übungen fortfahren: ***Local Data Areas***.

10 Local Data Areas

Zurzeit werden die Felder, die in Ihrem Programm benutzt werden, mit `DEFINE DATA` im Programm selbst definiert. Es ist jedoch auch möglich, die Felddefinitionen in einer Local Data Area (LDA) außerhalb des Programms zu definieren und diese Local Data Area im `DEFINE DATA`-Statement des Programms lediglich namentlich zu referenzieren. Für die Wiederverwendbarkeit und zu Gunsten einer klaren Anwendungsstruktur ist es in der Regel besser, die Felder außerhalb eines Programms in einer Data Area zu definieren.

Sie werden jetzt die Informationen aus dem `DEFINE DATA`-Statement in eine Local Data Area verlagern. Wenn Sie mit den Übungen in diesem Kapitel fertig sind, wird Ihre Beispielanwendung aus den folgenden Modulen bestehen:



Local Data Area erstellen

Sie werden jetzt den Data-Area-Editor aufrufen, in dem Sie die erforderlichen Felder definieren.

Data-Area-Editor aufrufen

- Geben Sie die folgenden Informationen unten im Menü **Development Functions** ein und drücken Sie EINGABE:

```
    Code .. C    Type .. L   Name .. LDA01_____________________________



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

"C" steht für die Funktion **Create Object** (Objekt erstellen), "L" steht für den Objekttyp Local Data Area, und "LDA01" ist der Name für Ihre neue Local Data Area.

Der Data-Area-Editor erscheint. Der Objekttyp wurde auf "Local" gesetzt. Dies wird oben links im Bildschirm angezeigt.

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

Datenfelder definieren

Sie werden jetzt die folgenden Felder definieren:

Level (Spalte L)	Name	Format (Spalte F)	Länge
1	#NAME-START	A	20
1	#NAME-END	A	20
1	#MARK	A	1

Dies sind die Benutzervariablen, die Sie zuvor im `DEFINE DATA`-Statement definiert haben.

Datenfelder definieren

1 Geben Sie das folgende Kommando ein, um Ihr Programm und den Data-Area-Editor im selben Bildschirm im Split-Screen-Modus anzuzeigen:

```
SPLIT P PGM01
```

Der Bildschirm ist jetzt in zwei Hälften geteilt. Ihr Programm wird in der unteren Hälfte des Bildschirms angezeigt. In diesem Modus kann es nicht geändert werden. Sie können das Programm aber als Referenz verwenden, um die Definitionen der Benutzervariablen im Data-Area-Editor einzufügen. Mit den Kommandos `SPLIT +` und `SPLIT -` können Sie im Programm vor- und zurückblättern.

2 Geben Sie alle erforderlichen Informationen so ein, wie sie in der Tabelle oben angegeben sind.

Die Local Data Area sollte nun folgendermaßen aussehen:

```
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 Geben Sie das folgende Kommando ein, um den Split-Screen-Modus zu beenden:

```
SPLIT .
```

Datenfelder aus einem DDM importieren

Sie werden jetzt dieselben Datenfelder importieren, die Sie zuvor im `DEFINE DATA`-Statement des Programms definiert hatten. Die Felder werden direkt aus einem Natural-Daten-View in den Data-Area-Editor eingelesen. Ein Daten-View referenziert die Datenbankfelder, die in einem DDM (Data Definition Module) definiert sind.

Datenfelder aus einem DDM importieren

1 Geben Sie Folgendes in der Zeile unter den Variablen ein, die Sie bisher definiert haben - beginnend in der Spalte **T**:

```
.V(EMPLOYEES)
```

Beispiel:

```
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 Drücken Sie EINGABE.

Der `EMPLOYEES`-View erscheint.

```
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 Markieren Sie die folgenden Felder indem Sie ein beliebiges Zeichen in der Spalte **I** eingeben:

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

Nicht alle diese Felder werden auf der ersten Seite des Views angezeigt. Drücken Sie EINGABE, um vorwärts zu blättern.

Anmerkung: Das Feld `PERSONNEL-ID` wird später benutzt, wenn Sie das Subprogramm erstellen.

4 Nachdem Sie alle erforderlichen Felder markiert haben, drücken Sie solange EINGABE bis der Data-Area-Editor wieder angezeigt wird.

Die Local Data Area sollte nun folgendermaßen aussehen:

```
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 ↩
 ↩
```

"-VIEW" wurde automatisch an den Namen des Views angehängt. Das ist derselbe Name, den Sie bereits in Ihrem Programm benutzt haben.

Die Spalte **T** gibt den Variablentyp an. Der View ist mit "V" gekennzeichnet und jede Gruppe ist mit "G" gekennzeichnet.

5 Speichern Sie die Local Data Area mit `STOW`.

Local Data Area aus dem Programm aufrufen

Sobald eine Local Data Area mit `STOW` gespeichert wurde, kann sie aus einem Natural-Programm heraus aufgerufen werden.

Sie werden jetzt das `DEFINE DATA`-Statement in Ihrem Programm so ändern, dass die eben von Ihnen definierte Local Data Area benutzt wird.

Local Data Area in Ihrem Programm benutzen

1 Kehren Sie zum Programmeditor zurück, indem Sie Folgendes in der Kommandozeile des Data-Area-Editors eingeben.

```
E PGM01
```

2 Löschen Sie im DEFINE DATA-Statement alle Variablen zwischen LOCAL und END-DEFINE (benutzen Sie hierzu das Zeilenkommando .D).

3 Referenzieren Sie die Local Data Area, indem Sie die LOCAL-Zeile folgendermaßen ändern:

```
LOCAL USING LDA01
```

Ihr Programm sollte nun folgendermaßen aussehen:

```
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 Führen Sie das Programm mit `RUN` aus.

5 Um zu überprüfen, ob das Ergebnis immer noch dasselbe ist wie vorher (als noch keine Local Data Area mit `DEFINE DATA` referenziert wurde), geben Sie "JONES" als Startname ein und drücken Sie EINGABE.

6 Geben Sie `EDIT` in der `MORE`-Zeile ein, um zum Programmeditor zurückzukehren.

7 Speichern Sie das Programm mit `STOW`.

Sie können nun mit den nächsten Übungen fortfahren: ***Global Data Areas***.

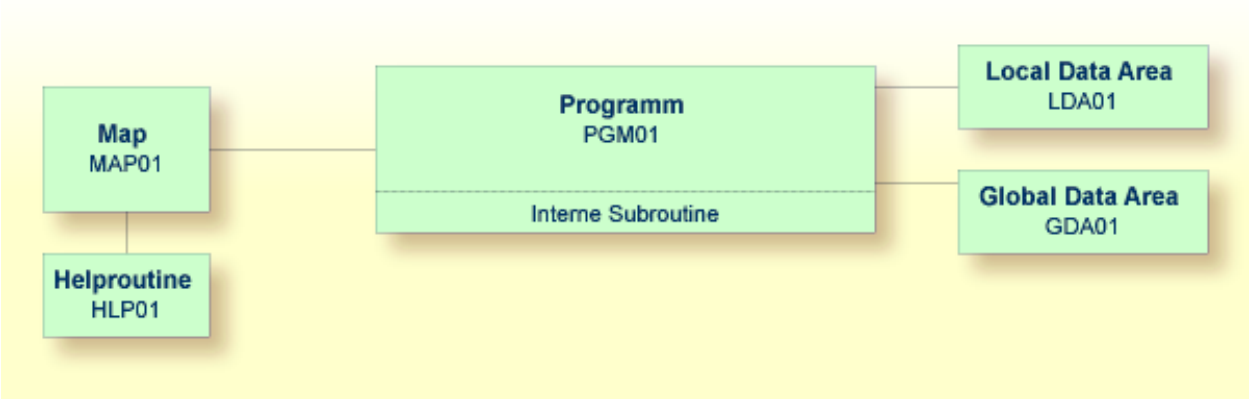
11 Global Data Areas

Die Daten in einer Global Data Area (GDA) können von mehreren Programmen, externen Subroutinen und Helproutinen benutzt werden.

Jede Änderung eines Datenelements in einer Global Data Area hat Auswirkungen auf alle Natural-Objekte, die diese Global Data Area referenzieren. Aus diesem Grund müssen Sie nach der Änderung einer Global Data Area alle zuvor erstellten Natural-Objekte, die diese Global Data Area referenzieren, noch einmal mit `STOW` speichern. Die Reihenfolge, in der die Objekte mit `STOW` gespeichert werden, ist wichtig. Sie müssen zuerst die Global Data Area mit `STOW` speichern und danach das Programm. Wenn Sie das Programm zuerst mit `STOW` speichern wollen und dann erst die Global Data Area, dann kann das Programm nicht gespeichert werden, weil die neuen Elemente in der Global Data Area nicht gefunden werden.

Sie werden jetzt eine Global Data Area erstellen, die von Ihrem Programm benutzt wird und auch von einer externen Subroutine, die Sie später erstellen werden. Als Basis für Ihre Global Data Area übernehmen Sie einige Informationen aus der Local Data Area, die Sie soeben erstellt haben.

Wenn Sie mit den Übungen in diesem Kapitel fertig sind, wird Ihre Beispielanwendung aus den folgenden Modulen bestehen:



Global Data Area mit Hilfe einer bestehenden Local Data Area erstellen

Sie können eine neue Data Area erstellen, indem Sie eine bestehende Data Area editieren und sie dann unter einem anderen Namen und einem anderen Typ abspeichern. Die ursprüngliche Data Area bleibt hierbei unverändert und die neue Data Area kann sofort editiert werden. Da die Felder `#NAME-START` und `#NAME-END` nicht in der Global Data Area benötigt werden, werden Sie sie entfernen.

Global Data Area erstellen

1 Kehren Sie zur Local Data Area zurück, indem Sie Folgendes in der Kommandozeile des Programmeditors eingeben.

```
E LDA01
```

2 Geben Sie Folgendes in der Kommandozeile des Data-Area-Editors ein, um die Data Area unter einem neuen Namen zu speichern:

```
SA GDA01
```

Die aktuelle Data Area wird unter dem neuen Namen `GDA01` gespeichert. Die Local Data Area mit dem Namen `LDA01` wird noch immer im Data-Area-Editor angezeigt.

3 Laden Sie `GDA01` mit dem folgenden Kommando in dem Data-Area-Editor:

```
E GDA01
```

4 Geben Sie das folgende Kommando ein, um die Local Data Area in eine Global Data Area abzuändern:

```
SET TYPE G
```

wobei "G" für Global Data Area steht.

Der Objekttyp ändert sich in "Global". Dies wird oben links im Bildschirm angezeigt.

5 Benutzen Sie das Zeilenkommando `.D`, um die folgenden Felder zu löschen:

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

Sie geben das Zeilenkommando beginnend in der Spalte **T** der Zeile ein, die das zu löschende Feld enthält. Da die oben genannten Felder in zwei aufeinander folgenden Zeilen stehen, können Sie auch das Zeilenkommando `.D(2)` eingeben, um beide Felder gleichzeitig zu löschen.

6 Drücken Sie EINGABE.

Die Global Data Area sollte nun folgendermaßen aussehen:

```
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 Speichern Sie die Global Data Area mit `STOW`.

Local Data Area anpassen

Die Felder, die jetzt in der Global Data Area enthalten sind, werden nicht mehr in der Local Data Area benötigt. Deshalb werden Sie jetzt alle Felder außer `#NAME-START` und `#NAME-END` aus der Local Data Area entfernen.

Felder entfernen

1 Kehren Sie zu Ihrer Local Data Area zurück, indem Sie Folgendes in der Kommandozeile des Data-Area-Editors eingeben:

```
E LDA01
```

2 Benutzen Sie das Zeilenkommando `.D`, um alle Felder außer `#NAME-START` und `#NAME-END` zu löschen.

Wenn Sie den obersten Eintrag für einen View löschen (gekennzeichnet durch ein "V" vor dem View-Namen), werden alle Felder, die zu diesem View gehören, automatisch gelöscht.

3 Speichern Sie die geänderte Local Data Area mit `STOW`.

Die Local Data Area sollte nun folgendermaßen aussehen:

```
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
```

Global Data Area aus dem Programm aufrufen

Sobald eine Global Data Area mit `STOW` gespeichert wurde, kann sie aus einem Natural-Programm heraus aufgerufen werden.

Sie werden jetzt das `DEFINE DATA`-Statement in Ihrem Programm so ändern, dass die eben von Ihnen definierte Global Data Area benutzt wird.

Global Data Area in Ihrem Programm benutzen

1 Kehren Sie zum Programmeditor zurück, indem Sie Folgendes in der Kommandozeile des Data-Area-Editors eingeben.

```
E PGM01
```

2 Geben Sie Folgendes in der Zeile über `LOCAL USING LDA01` ein:

```
GLOBAL USING GDA01
```

Eine Global Data Area muss immer vor einer Local Data Area angegeben werden. Andernfalls tritt ein Fehler auf.

Ihr Programm sollte nun folgendermaßen aussehen:

```
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 Führen Sie das Programm mit `RUN` aus.

4 Um zu überprüfen, ob das Ergebnis immer noch dasselbe ist wie vorher (als noch keine Global Data Area mit `DEFINE DATA` referenziert wurde), geben Sie "JONES" als Startname ein und drücken Sie EINGABE.

5 Geben Sie `EDIT` in der `MORE`-Zeile ein, um zum Programmeditor zurückzukehren.

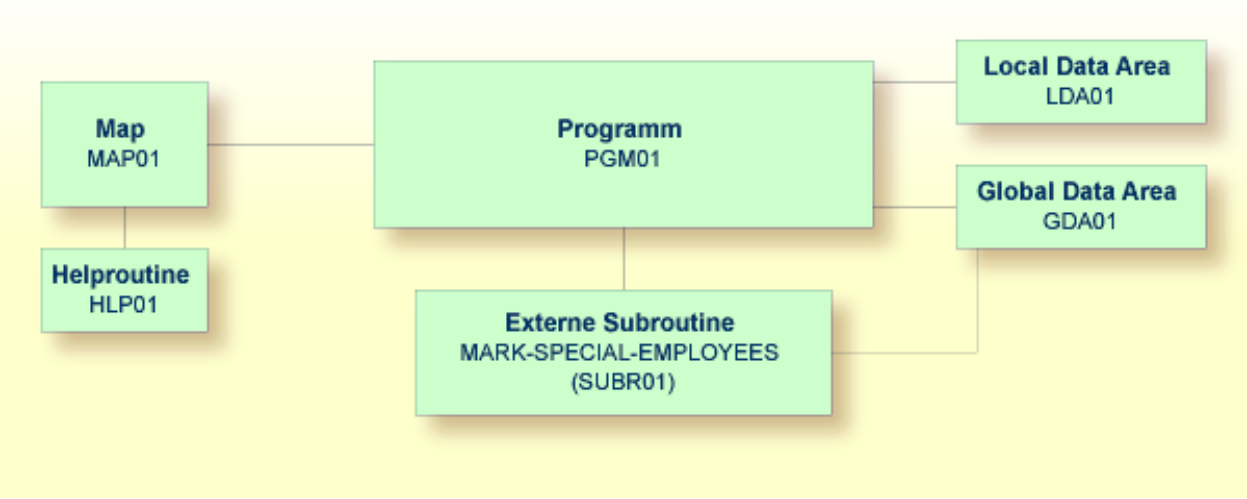
6 Speichern Sie das Programm mit `STOW`.

Sie können nun mit den nächsten Übungen fortfahren: ***Externe Subroutinen***.

12 Externe Subroutinen

Bis jetzt ist die Subroutine `MARK-SPECIAL-EMPLOYEES` mit einem `DEFINE SUBROUTINE`-Statement im Programm selbst definiert. Sie werden diese Subroutine jetzt als separates Objekt definieren, das sich außerhalb des Programms befindet.

Wenn Sie mit den Übungen in diesem Kapitel fertig sind, wird Ihre Beispielanwendung aus den folgenden Modulen bestehen:



Externe Subroutine erstellen

Da der Code, der zurzeit im Programm definiert ist, in der externen Subroutine wiederverwendet werden soll, werden Sie das Programm jetzt unter einem anderen Namen speichern, den Typ in Subroutine ändern und alle nicht benötigten Zeilen löschen.

Das `DEFINE SUBROUTINE`-Statement wird in der externen Subroutine genauso kodiert wie in der internen Subroutine des Programms.

Externe Subroutine erstellen

1 Geben Sie Folgendes in der Kommandozeile des Programmeditors ein:

```
SA SUBR01
```

Das aktuelle Programm wird unter dem neuen Namen `SUBR01` gespeichert. Das Programm wird noch immer im Editor angezeigt.

2 Laden Sie das neu erstellte Objekt mit dem folgenden Kommando in den Editor:

```
E SUBR01
```

Der Objekttyp ist noch immer Programm.

3 Geben Sie das folgende Kommando ein, um das Programm in eine externe Subroutine zu ändern:

```
SET TYPE S
```

wobei "S" für Subroutine steht.

Der im Bildschirm angezeigte Objekttyp ändert sich in "Subroutine".

4 Benutzen Sie das Zeilenkommando `.D`, um alle Zeilen außer den Folgenden zu löschen:

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

Sie können auch einen kompletten Textblock löschen. Hierzu gehen Sie folgendermaßen vor:

1. Geben am Anfang des Textblocks das Zeilenkommando `.X` ein.
2. Geben Sie am Anfang der letzten Zeile des Textblocks das Zeilenkommando `.Y` ein.
3. Drücken Sie EINGABE.

   Der zu löschende Textblock ist jetzt mit "X" und "Y" markiert. (Wenn Sie diese Markierungen entfernen möchten, geben Sie `RESET` in der Kommandozeile ein).
4. Geben Sie `DX-Y` in der Kommandozeile ein, um den markierten Textblock zu löschen.

5 Speichern Sie die Subroutine mit `STOW`.

Externe Subroutine aus dem Programm aufrufen

Mit dem `PERFORM`-Statement kann man sowohl interne als auch externe Subroutinen aufrufen. Wenn im Programm keine interne Subroutine gefunden wird, versucht Natural automatisch, eine externe Subroutine mit demselben Namen auszuführen. Natural sucht hierbei nach dem Namen, der im Subroutinencode definiert wurde (d.h. dem Subroutinennamen). Natural sucht nicht nach dem Namen, den Sie beim Speichern der Subroutine angegeben haben (d.h. dem Objektnamen).

Jetzt, nachdem Sie eine externe Subroutine erstellt haben, müssen Sie die interne Subroutine (die denselben Namen hat wie die externe Subroutine) aus Ihrem Programm entfernen.

Externe Subroutine in Ihrem Programm benutzen

1 Kehren Sie zum Programmeditor zurück, indem Sie Folgendes in der Kommandozeile des Editors eingeben, in dem die Subroutine zurzeit angezeigt wird.

```
E PGM01
```

2 Entfernen Sie die folgenden Zeilen:

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

Ihr Programm sollte nun folgendermaßen aussehen:

```
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 Führen Sie das Programm mit RUN aus.

4 Geben Sie "JONES" als Startname ein und drücken Sie EINGABE.

Die daraufhin erscheinende Liste sollte immer noch einen Stern bei jedem Mitarbeiter anzeigen, der 20 oder mehr Urlaubstage hat.

5 Geben Sie EDIT in der MORE-Zeile ein, um zum Programmeditor zurückzukehren.

6 Speichern Sie das Programm mit STOW.

Identische Subroutinennamen auflisten

1 Geben Sie eines der folgenden Kommandos in der Kommandozeile des Programmeditors ein:

```
LIST EXTENDED SUBROUTINE *
```

```
L EXT S *
```

Der folgende Bildschirm erscheint. In ihm werden alle externen Subroutinenobjekte (Member) mit den entsprechenden Langnamen aufgelistet, die in der aktuellen Natural-Library und Systemdatei enthalten sind.

```
12:21:09               ***** NATURAL LIST COMMAND *****             2012-07-17
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    2009-06-30  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
```

Wenn Sie nur Subroutinennamen mit bestimmten Anfangsbuchstaben anzeigen möchten, können Sie den entsprechenden Suchwert gefolgt von einem Stern (*) in dem Feld unter der Überschrift **Subroutine/Class Name** oder unter der Überschrift **Member** angeben. Beispiel:

```
Cmd Subroutine/Class Name            Type  S/C Member    Cat Date    Cat Time
--- MARK*____________________________ S____ --- *_______ *_________ *________
__  MARK-SPECIAL-EMPLOYEES           Subro S/C SUBR01    2009-06-30  12:11:56 ↩
 ↩
```

2 Drücken Sie PF3, um zum Programmeditor zurückzukehren.

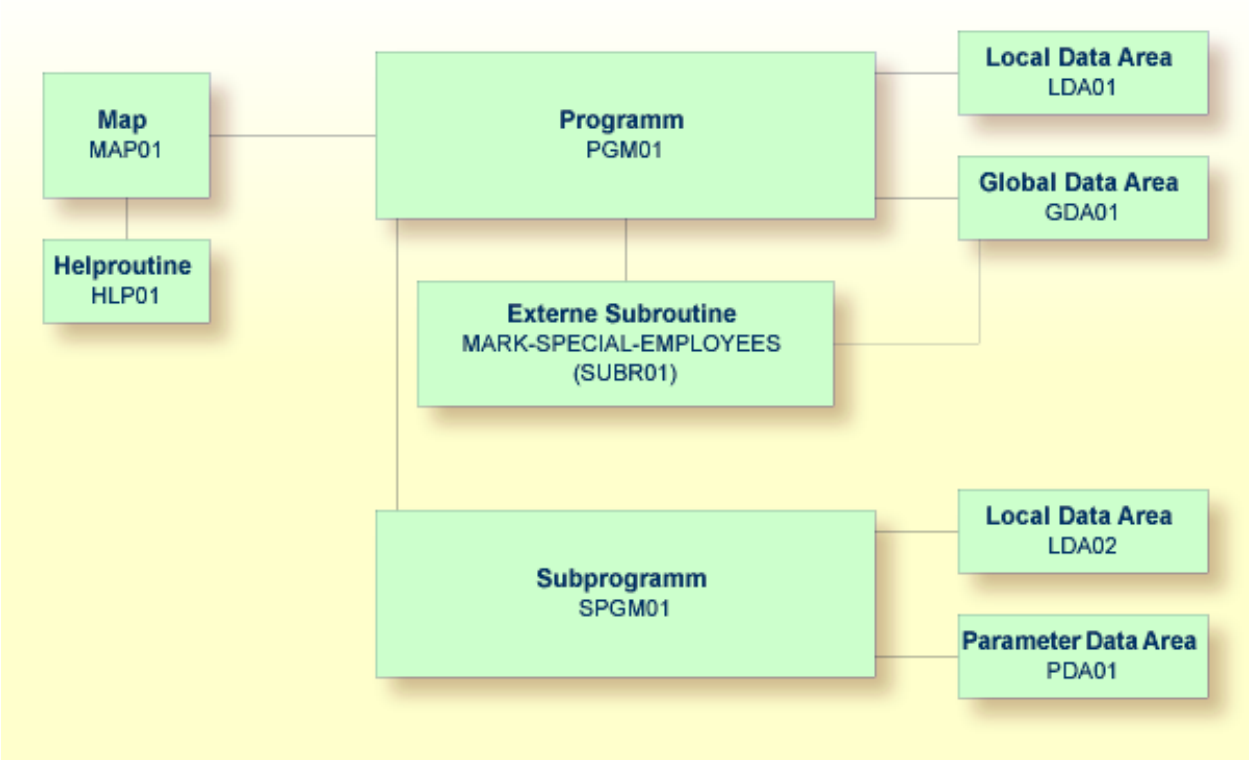
Sie können nun mit den nächsten Übungen fortfahren: ***Subprogramme***.

13 Subprogramme

Sie werden Ihr Programm jetzt durch ein `CALLNAT`-Statement erweitern, mit dem ein Subprogramm aufgerufen wird. In dem Subprogramm bilden die Mitarbeiter, die vom Hauptprogramm gefunden wurden, die Grundlage für eine `FIND`-Anfrage in der `VEHICLES`-Datei. Diese Datei ist ebenfalls Bestandteil der Demodatenbank. In der Ausgabe werden dann Fahrzeuginformationen aus dem Subprogramm und Mitarbeiterinformationen aus dem Hauptprogramm zu sehen sein.

Das neue Subprogramm benötigt eine weitere Local Data Area und eine Parameter Data Area.

Wenn Sie mit den Übungen in diesem Kapitel fertig sind, wird Ihre Beispielanwendung aus den folgenden Modulen bestehen:



Local Data Area ändern

Sie werden jetzt weitere Felder in die Local Data Area einfügen, die Sie zuvor erstellt haben. Diese Felder werden von dem Subprogramm benutzt, das Sie später erstellen werden.

Weitere Felder in die Local Data Area einfügen

1 Kehren Sie zur Local Data Area zurück.

```
E LDA01
```

2 Definieren Sie die folgenden Felder unter #NAME-END:

Level (Spalte L)	Name	Format (Spalte F)	Länge
1	#PERS-ID	A	8
1	#MAKE	A	20
1	#MODEL	A	20

Die Local Data Area sollte nun folgendermaßen aussehen:

```
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 Speichern Sie die Local Data Area mit STOW.

Parameter Data Area mit Hilfe einer bestehenden Local Data Area erstellen

Eine Parameter Data Area (PDA) wird zur Angabe der Datenparameter benutzt, die zwischen Ihrem Natural-Programm und dem Subprogramm (das Sie später noch erstellen werden) ausgetauscht werden sollen. Die Parameter Data Area wird im Subprogramm referenziert.

Mit kleinen Änderungen kann Ihre Local Data Area zur Erstellung der Parameter Data Area benutzt werden: Sie werden zwei Datenfelder in der Local Data Area löschen und die so veränderte Local Data Area als Parameter Data Area speichern. Die ursprüngliche Local Data Area wird hierdurch nicht verändert.

Parameter Data Area erstellen

1 Löschen Sie die Felder `#NAME-START` und `#NAME-END` in der Local Data Area.

2 Geben Sie Folgendes in der Kommandozeile des Data-Area-Editors ein.

```
SA PDA01
```

Die aktuelle Data Area wird unter dem neuen Namen `PDA01` gespeichert. Die bestehende Local Data Area wird noch immer im Editor angezeigt.

3 Laden Sie die neu erstellte Data Area mit dem folgenden Kommando in den Editor:

```
E PDA01
```

4 Geben Sie das folgende Kommando ein, um die Local Data Area in eine Parameter Data Area zu ändern:

```
SET TYPE A
```

wobei "A" für Parameter Data Area steht.

Der Objekttyp ändert sich in "Parameter". Dies wird oben links im Bildschirm angezeigt. Die Parameter Data Area sollte nun folgendermaßen aussehen:

```
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 Speichern Sie die Parameter Data Area mit `STOW`.

Eine zweite Local Data Area mit einem anderen View erstellen

Sie werden jetzt eine zweite Local Data Area erstellen und Felder aus dem DDM für die Datenbankdatei `VEHICLES` importieren.

Diese Local Data Area wird dann in Ihrem Subprogramm referenziert.

Local Data Area erstellen

1 Geben Sie das folgende Kommando in der Kommandozeile des Data-Area-Editors ein.

```
CLEAR
```

Der Data-Area-Editor ist jetzt leer.

2 Geben Sie das Folgendes in der Kommandozeile ein, um den Typ der Data Area zu ändern:

```
SET TYPE L
```

wobei "L" für Local Data Area steht.

3 Geben Sie Folgendes in der ersten Zeile des Editierbereiches ein, beginnend in der Spalte **T**:

```
.V(VEHICLES)
```

4 Drücken Sie EINGABE.

Der `VEHICLES`-View erscheint.

```
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 Markieren Sie die folgenden Felder, indem Sie ein beliebiges Zeichen in der Spalte **I** eingeben:

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

6 Nachdem Sie alle erforderlichen Felder markiert haben, drücken Sie EINGABE, um zum Data-Area-Editor zurückzukehren.

Die Local Data Area sollte nun folgendermaßen aussehen:

```
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 Geben Sie Folgendes in der Kommandozeile ein, um die neue Local Data Area zu speichern:

```
SA LDA02
```

8 Speichern Sie die neue Local Data Area mit `STOW`.

Subprogramm erstellen

Sie werden jetzt ein Subprogramm erstellen, das mit Hilfe einer Parameter Data Area und einer Local Data Area Informationen aus der `VEHICLES`-Datei abruft. Das Programm `PGM01` übergibt das Personalkennzeichen (`PERSONNEL-ID`) an das Subprogramm. Das Subprogramm benutzt dieses Kennzeichen für die Suche in der `VEHICLES`-Datei.

Subprogramm erstellen

1 Geben Sie das folgende Kommando in der Kommandozeile des Data-Area-Editors ein.

```
E N
```

wobei "N" für Subprogramm steht.

Ein leerer Programmeditor wird aufgerufen. Der Objekttyp ist auf Subprogramm gesetzt.

2 Geben Sie Folgendes ein:

```
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
```

Dieses Subprogramm gibt die folgenden Informationen an ein bestimmtes Personalkennzeichen zurück: die Marke und das Modell des Firmenfahrzeuges eines Mitarbeiters.

Auf Grund des Suchkriteriums `#PERS-ID` wählt das `FIND`-Statement eine Reihe von Datensätzen (hier: einen Datensatz) aus der Datenbank aus.

Das Feld `#PERS-ID` des Subprogramms erhält den Wert von `PERSONNEL-ID`, der vom Programm `PGM01` übergeben wurde. Das Subprogramm benutzt diesen Wert für die Suche in der `VEHICLES`-Datei.

3 Speichern Sie das Subprogramm mit `STOW`.

```
STOW SPGM01
```

Subprogramm aus dem Programm aufrufen

Ein Subprogramm wird mit einem `CALLNAT`-Statement aus dem Hauptprogramm aufgerufen. Ein Subprogramm kann nur mit einem `CALLNAT`-Statement aufgerufen werden; es kann selbst nicht ausgeführt werden. Ein Subprogramm kann nicht auf die Global Data Area zugreifen, die von dem aufrufenden Objekt benutzt wird.

Die Daten werden vom Hauptprogramm an das Subprogramm mit Hilfe von Parametern übergeben, die im Subprogramm mit einem `DEFINE DATA PARAMETER`-Statement referenziert werden.

Die Variablen, die in der Parameter Data Area des Subprogramms definiert sind, müssen nicht unbedingt dieselben Namen haben wie die Variablen im `CALLNAT`-Statement. Sie müssen nur in der Reihenfolge, im Format und in der Länge übereinstimmen.

Sie werden jetzt Ihr Hauptprogramm ändern, damit es das eben von Ihnen erstellte Subprogramm benutzen kann.

Subprogramm in Ihrem Hauptprogramm benutzen

1 Kehren Sie zum Programmeditor zurück, indem Sie Folgendes in der Kommandozeile eingeben.

```
E PGM01
```

2 Geben Sie Folgendes direkt über dem `DISPLAY`-Statement ein:

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

Das `RESET`-Statement setzt die Werte für `#MAKE` und `#MODEL` auf Nullwerte.

3 Löschen Sie die Zeile, die das `DISPLAY`-Statement enthält, und ersetzen Sie sie mit Folgendem:

```
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
```

Der mit dem `WRITE TITLE`-Statement definierte Text wird bei der Ausgabe oben auf jeder Seite angezeigt. Das `WRITE TITLE`-Statement setzt den Standardseitentitel außer Kraft: die Informationen, die bisher oben auf jeder Seite angezeigt wurden (Seitenzahl, Datum und Uhrzeit) werden nicht mehr angezeigt. Jeder Schrägstrich (/) sorgt dafür, dass die nachfolgenden Informationen in einer neuen Zeile angezeigt werden.

Da das Subprogramm jetzt zusätzliche Fahrzeuginformationen ausgibt, muss die Größe der Spalten in der Ausgabe angepasst werden. Die Spalten erhalten kürzere Überschriften. Die Spalte, in der der Stern angezeigt wird (`#MARK`), bekommt überhaupt keine Überschrift. Zwischen den einzelnen Spalten wird je ein Leerzeichen eingefügt (`1X`). Jeder Schrägstrich in einer Spaltenüberschrift lässt die nachfolgenden Informationen in derselben Spalte in einer neuen Zeile erscheinen.

Ihr Programm sollte nun folgendermaßen aussehen:

```
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 Führen Sie das Programm mit `RUN` aus.

5 Geben Sie "JONES" als Startname ein und drücken Sie EINGABE.

Die daraufhin erscheinende Liste sollte der folgenden Ausgabe ähneln:

```
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 Geben Sie `EDIT` in der `MORE`-Zeile ein, um zum Programmeditor zurückzukehren.

7 Speichern Sie das Programm mit `STOW`.

Sie haben das Tutorial jetzt erfolgreich abgeschlossen.

Stichwortverzeichnis