Natural Programming Modes

This chapter describes the two programming modes offered by Natural. The following topics are covered:

- Purpose of Programming Modes
- Setting/Changing the Programming Mode
- Functional Differences

Purpose of Programming Modes

Natural offers two ways of programming:

- Reporting Mode
- Structured Mode

Note:
Generally, it is recommended to use structured mode exclusively, because it provides for more clearly structured applications.

Reporting Mode

Reporting mode is only useful for the creation of adhoc reports and small programs which do not involve complex data and/or programming constructs. (If you decide to write a program in reporting mode, be aware that small programs may easily become larger and more complex.)

Please note that certain Natural statements are available only in reporting mode, whereas others have a specific structure when used in reporting mode. For an overview of the statements that can be used in reporting mode, see *Reporting Mode Statements* in the *Statements* documentation.

Structured Mode

Structured mode is intended for the implementation of complex applications with a clear and well-defined program structure. The major benefits of structured mode are:

- The programs have to be written in a more structured way and are therefore easier to read and consequently easier to maintain.
- As all fields to be used in a program have to be defined in one central location (instead of being scattered all over the program, as is possible in reporting mode), overall control of the data used is much easier.

With structured mode, you also have to make more detail planning before the actual programs can be coded, thereby avoiding many programming errors and inefficiencies.

For an overview of the statements that can be used in structured mode, see *Statements Grouped by Functions* in the *Statements* documentation.

Setting/Changing the Programming Mode

The default programming mode is set by the Natural administrator with the profile parameter `SM`. You can change the mode by using the Natural system command `GLOBALS` and the session parameter `SM`:

Structured Mode:	`GLOBALS SM=ON`
Reporting Mode:	`GLOBALS SM=OFF`

For further information on the Natural profile and session parameter `SM`, see *SM - Programming in Structured Mode* in the *Parameter Reference*.

For information on how to change the programming mode, see *SM - Programming in Structured Mode* in the *Parameter Reference*.

Functional Differences

The following major functional differences exist between reporting mode and structured mode:

- Syntax Related to Closing Loops and Functional Blocks
- Closing a Processing Loop in Reporting Mode
- Closing a Processing Loop in Structured Mode
- Location of Data Elements in a Program
- Database Reference

Note:
For detailed information on functional differences that exist between the two modes, see the *Statements* documentation. It provides separate syntax diagrams and syntax element descriptions for each mode-sensitive statement. For a functional overview of the statements that can be used in reporting mode, see *Reporting Mode Statements* in the *Statements* documentation.

Syntax Related to Closing Loops and Functional Blocks

Reporting Mode:	`(CLOSE) LOOP` and `DO ... DOEND` statements are used for this purpose. `END-...` statements (except `END-DEFINE`, `END-DECIDE` and `END-SUBROUTINE`) cannot be used.
Structured Mode:	Every loop or logical construct must be explicitly closed with a corresponding `END-...` statement. Thus, it becomes immediately clear, which loop/logical constructs ends where. `LOOP` and `DO/DOEND` statements cannot be used.

The two examples below illustrate the differences between the two modes in constructing processing loops and logical conditions.

Reporting Mode Example:

The reporting mode example uses the statements DO and DOEND to mark the beginning and end of the statement block that is based on the AT END OF DATA condition. The END statement closes all active processing loops.

```
READ EMPLOYEES BY PERSONNEL-ID
DISPLAY NAME BIRTH
AT END OF DATA
   DO
      SKIP 2
      WRITE / 'LAST SELECTED:' OLD(NAME)
   DOEND
END
```

Structured Mode Example:

The structured mode example uses an END-ENDDATA statement to close the AT END OF DATA condition, and an END-READ statement to close the READ loop. The result is a more clearly structured program in which you can see immediately where each construct begins and ends:

```
DEFINE DATA LOCAL
1 MYVIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
  2 BIRTH

END-DEFINE
READ MYVIEW BY PERSONNEL-ID
   DISPLAY NAME BIRTH
   AT END OF DATA
      SKIP 2
      WRITE / 'LAST SELECTED:' OLD(NAME)
   END-ENDDATA
END-READ
END
```

Closing a Processing Loop in Reporting Mode

The statements END, LOOP (or CLOSE LOOP) or SORT may be used to close a processing loop.

The LOOP statement can be used to close more than one loop, and the END statement can be used to close all active loops. These possibilities of closing several loops with a single statement constitute a basic difference to structured mode.

A SORT statement closes all processing loops and initiates another processing loop.

Example 1 - LOOP:

```
FIND ...
   FIND ...
   ...
   ...
   LOOP      /* closes inner FIND loop
LOOP         /* closes outer FIND loop
...
...
```

Example 2 - END:

```
FIND ...
   FIND ...
   ...
   ...
END             /* closes all loops and ends processing
```

Example 3 - SORT:

```
FIND ...
   FIND ...
   ...
   ...
SORT ...        /* closes all loops, initiates loop
...
END             /* closes SORT loop and ends processing
```

Closing a Processing Loop in Structured Mode

Structured mode uses a specific loop-closing statement for each processing loop. Also, the `END` statement does not close any processing loop. The `SORT` statement must be preceded by an `END-ALL` statement, and the `SORT` loop must be closed with an `END-SORT` statement.

Example 1 - FIND:

```
FIND ...
   FIND ...
   ...
   ...
   END-FIND       /* closes inner FIND loop
END-FIND          /* closes outer FIND loop
...
```

Example 2 - READ:

```
READ ...
   AT END OF DATA
   ...
   END-ENDDATA
   ...
END-READ           /* closes READ loop
...
...
END
```

Example 3 - SORT:

```
READ ...
   FIND ...
   ...
   ...
END-ALL         /* closes all loops
SORT            /* opens loop
```

```
...
...
END-SORT        /* closes SORT loop
END
```

Location of Data Elements in a Program

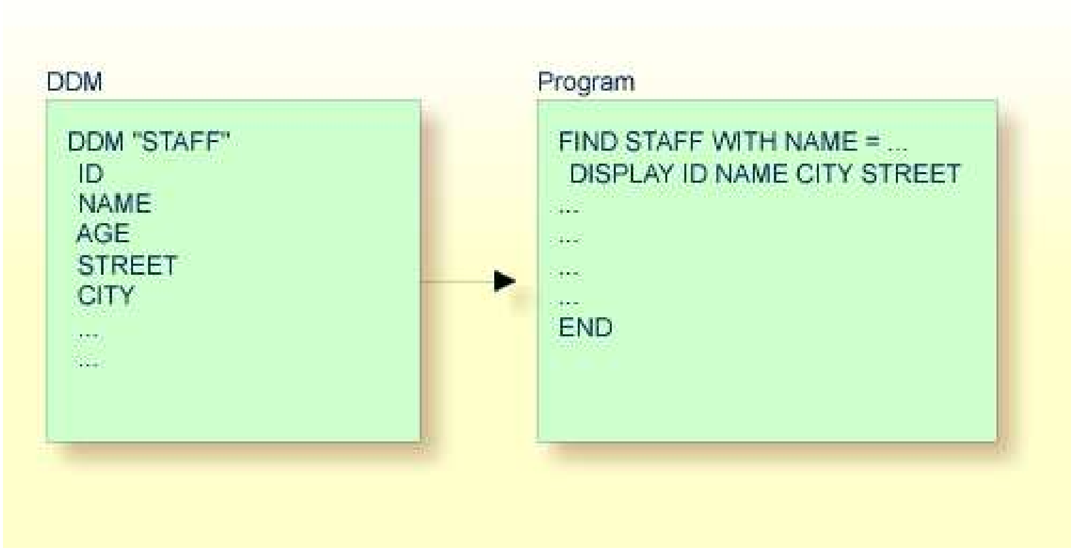
In reporting mode, you can use database fields without having to define them in a `DEFINE DATA` statement; also, you can define user-defined variables anywhere in a program, which means that they can be scattered all over the program.

In structured mode, *all* data elements to be used have to be defined in one central location (either in the `DEFINE DATA` statement at the beginning of the program, or in a data area outside the program).

Database Reference

Reporting Mode:

In reporting mode, database fields and data definition modules (DDMs) may be referenced without having been defined in a data area.

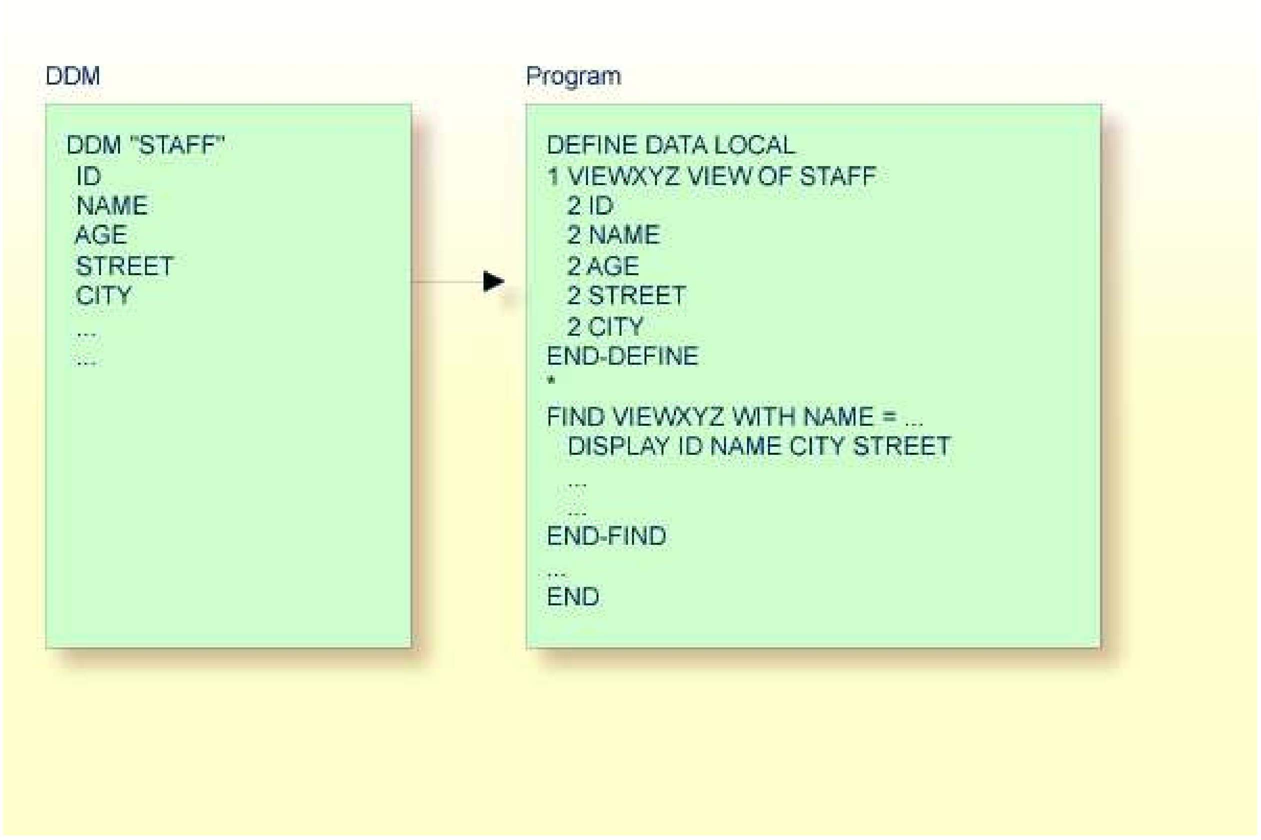


```
DDM "STAFF"
 ID
 NAME
 AGE
 STREET
 CITY
 ...
 ...
```

```
FIND STAFF WITH NAME = ...
 DISPLAY ID NAME CITY STREET
...
...
...
...
END
```

Structured Mode:

In structured mode, each database field to be used must be specified in a `DEFINE DATA` statement as described in *Defining Fields* and *Accessing Data in an Adabas Database*.



```
DDM "STAFF"
 ID
 NAME
 AGE
 STREET
 CITY
 ...
 ...
```

```
DEFINE DATA LOCAL
1 VIEWXYZ VIEW OF STAFF
  2 ID
  2 NAME
  2 AGE
  2 STREET
  2 CITY
END-DEFINE
*
FIND VIEWXYZ WITH NAME = ...
  DISPLAY ID NAME CITY STREET
  ...
  ...
END-FIND
...
END
```