OBTAIN

```
OBTAIN operand1 ...
```

This chapter covers the following topics:

- Function
- Restriction
- Syntax Description
- Examples

Function

The `OBTAIN` statement is used in reporting mode to cause one or more fields to be read from a file. The `OBTAIN` statement does not generate any executable code in the Natural object program. It is primarily used to read a range of values of a multiple-value field or a range of occurrences of a periodic group so that portions of these ranges may be subsequently referenced in the program.

An `OBTAIN` statement is *not* required for each database field to be referenced in the program since Natural automatically reads each database field referenced in a subsequent statement (for example, a `DISPLAY` or `COMPUTE` statement).

When multiple-value or periodic-group fields in the form of an array are referenced, the array must be defined with an `OBTAIN` statement to ensure that it is built for all occurrences of the fields. If individual multiple-value or periodic-group fields are referenced before the array is defined, the fields will not be placed within the array and will exist independent of the array. The fields will contain the same value as the corresponding occurrence within the array.

Individual occurrences of multiple-value or periodic-group fields or subarrays can be held within a previously defined array if the array dimensions of the second individual occurrence or array are contained within the initial array.

References to multiple-value or periodic-group fields with unique variable index cannot be contained in an array of values. If indvidual occurrences of an array are to be processed with a variable index, the index expression must be prefixed with the unique variable index to denote the individual array.

Restriction

The `OBTAIN` statement is for reporting mode only.

Syntax Description

Operand Definition Table:

Operand	Possible Structure					Possible Formats													Referencing Permitted	Dynamic Definition
operand1		S	A	G		A	U	N	P	I	F	B	D	T	L				yes	yes

Syntax Element Description:

operand1	With *operand1* you specify the field(s) to be made available as a result of the `OBTAIN` statement.

Examples:

```
READ FINANCE OBTAIN CREDIT-CARD (1-10)
DISPLAY CREDIT-CARD (3-5) CREDIT-CARD (6-8)
SKIP 1 END
```

The above example results in the first 10 occurrences of the field `CREDIT-CARD` (which is contained in a periodic group) being read and occurrences 3-5 and 6-8 being displayed where the subsequent subarrays will reside in the initial array (1-10).

```
READ FINANCE
MOVE 'ONE' TO CREDIT-CARD (1)
DISPLAY CREDIT-CARD (1) CREDIT-CARD (1-5)
```

Output:

```
    CREDIT-CARD      CREDIT-CARD
------------------ ----------------


ONE                DINERS CLUB
                   AMERICAN EXPRESS




ONE                AVIS
                   AMERICAN EXPRESS



ONE                HERTZ
                   AMERICAN EXPRESS



ONE                UNITED AIR TRAVEL
```

The first reference to `CREDIT-CARD (1)` is not contained within the array. The array which is defined after the reference to the unique occurrence (1) cannot retroactively include a unique occurrence or an array which is shorter than the one being defined.

```
READ FINANCE
OBTAIN CREDIT-CARD (1-5)
MOVE 'ONE' TO CREDIT-CARD (1)
DISPLAY CREDIT-CARD (1) CREDIT-CARD (1-5)
```

Output:

```
    CREDIT-CARD      CREDIT-CARD
------------------ ----------------


ONE                ONE
                   AMERICAN EXPRESS



ONE                ONE
                   AMERICAN EXPRESS



ONE                ONE
                   AMERICAN EXPRESS



ONE                ONE
```

The individual reference to `CREDIT-CARD (1)` is contained within the array defined in the `OBTAIN` statement.

```
MOVE (1) TO INDEX
READ FINANCE
DISPLAY CREDIT-CARD (1-5) CREDIT-CARD (INDEX)
```

Output:

```
    CREDIT-CARD      CREDIT-CARD
------------------- ----------------


DINERS CLUB         DINERS CLUB
AMERICAN EXPRESS



AVIS                AVIS
AMERICAN EXPRESS



HERTZ              HERTZ
AMERICAN EXPRESS



UNITED AIR TRAVEL  UNITED AIR TRAVEL
```

The reference to `CREDIT-CARD` using the variable index notation is not contained within the array.

```
RESET A(A20) B(A20) C(A20)
MOVE 2 TO I (N3)
MOVE 3 TO J (N3)
READ FINANCE
 OBTAIN CREDIT-CARD (1:3) CREDIT-CARD (I:I+2) CREDIT-CARD (J:J+2)
 FOR K (N3) = 1 TO 3
```

```
  MOVE CREDIT-CARD (1.K) TO A
  MOVE CREDIT-CARD (I.K) TO B
  MOVE CREDIT-CARD (J.K) TO C
  DISPLAY A B C
 LOOP /* FOR
LOOP / * READ
END
```

Output:

```
         A                   B                   C
------------------- ------------------- -------------------
CARD 01             CARD 02             CARD 03
CARD 02             CARD 03             CARD 04
CARD 03             CARD 04             CARD 05
```

The three arrays may be accessed individually by using the unique base index as qualifier for the index expression.

Invalid Example 1

```
READ FINANCE
OBTAIN CREDIT-CARD (1-10)
FOR I 1 10
MOVE CREDIT-CARD (I) TO A(A20)
WRITE A
END
```

The above example will produce error message NAT1006 (value for variable index = 0) because, at the time the record is read (READ), the index `I` still contains the value 0.

In any case, the above example would not have printed the first 10 occurrences of `CREDIT-CARD` because the individual occurrence with the variable index cannot be contained in the array and the variable index (`I`) is only evaluated when the next record is read.

The following is the correct method of performing the above:

```
READ FINANCE
OBTAIN CREDIT-CARD (1-10)
FOR I 1 10
MOVE CREDIT-CARD (1.I) TO A (A20)
WRITE A
END
```

Invalid Example 2

```
READ FINANCE
FOR I 1 10
WRITE CREDIT-CARD (I)
END
```

The above example will produce error message NAT1006 because the index `I` is zero when the record is read in the `READ` statement.

The following is the correct method of performing the above:

```
READ FINANCE
FOR I 1 10
GET SAME
WRITE CREDIT-CARD (0030/I)
END
```

The `GET SAME` statement is necessary to reread the record after the variable index has been updated in the `FOR` loop.

Examples

- Example 1 - OBTAIN Statement
- Example 2 - OBTAIN Statement with Multiple Ranges

Example 1 - OBTAIN Statement

```
** Example 'OBTEX1': OBTAIN
************************************************************************
RESET #INDEX (I1)
*
LIMIT 5
READ EMPLOYEES BY CITY
  OBTAIN SALARY (1:4)
  /*
  IF SALARY (4) GT 0 DO
    WRITE '=' NAME / 'SALARIES (1:4):' SALARY (1:4)
    FOR #INDEX 1 TO 4
      WRITE 'SALARY' #INDEX  SALARY (1.#INDEX)
    LOOP
    SKIP 1
  DOEND
LOOP
*
END
```

Output of Program OBTEX1:

```
Page      1                                                  05-02-08  13:37:48

NAME: SENKO
SALARIES (1:4):      31500      29900      28100      26600
SALARY    1      31500
SALARY    2      29900
SALARY    3      28100
SALARY    4      26600

NAME: HAMMOND
SALARIES (1:4):      22000      20200      18700      17500
SALARY    1      22000
SALARY    2      20200
SALARY    3      18700
SALARY    4      17500
```

Example 2 - OBTAIN Statement with Multiple Ranges

```
** Example 'OBTEX2': OBTAIN (with multiple ranges)
************************************************************************
RESET #INDEX (I1) #K (I1)
*
#INDEX  := 2
#K      := 3
*
LIMIT 2
*
READ EMPLOYEES BY CITY
  OBTAIN SALARY (1:5)
         SALARY (#INDEX:#INDEX+3)
  /*
  IF SALARY (5) GT 0 DO
    WRITE '=' NAME
    WRITE 'SALARIES (1-5):' SALARY (1:5) /
    WRITE 'SALARIES (2-5):' SALARY (#INDEX:#INDEX+3)
    WRITE 'SALARIES (2-5):' SALARY (#INDEX.1:4) /
    WRITE 'SALARY 3:' SALARY (3)
    WRITE 'SALARY 3:' SALARY (#K)
    WRITE 'SALARY 4:' SALARY (#INDEX.#K)
  DOEND
LOOP
```

Output of Program OBTEX2:

```
Page      1                                                  05-02-08  13:38:31

NAME: SENKO
SALARIES (1-5):      31500      29900      28100      26600      25200

SALARIES (2-5):      29900      28100      26600      25200
SALARIES (2-5):      29900      28100      26600      25200

SALARY 3:      28100
SALARY 3:      28100
SALARY 4:      26600
```

For further examples of using the `OBTAIN` statement, see

Referencing a Database Array (in the *Programming Guide*).