nanoBase 1997

An old, but free xBase for Dos.[1]

What is it

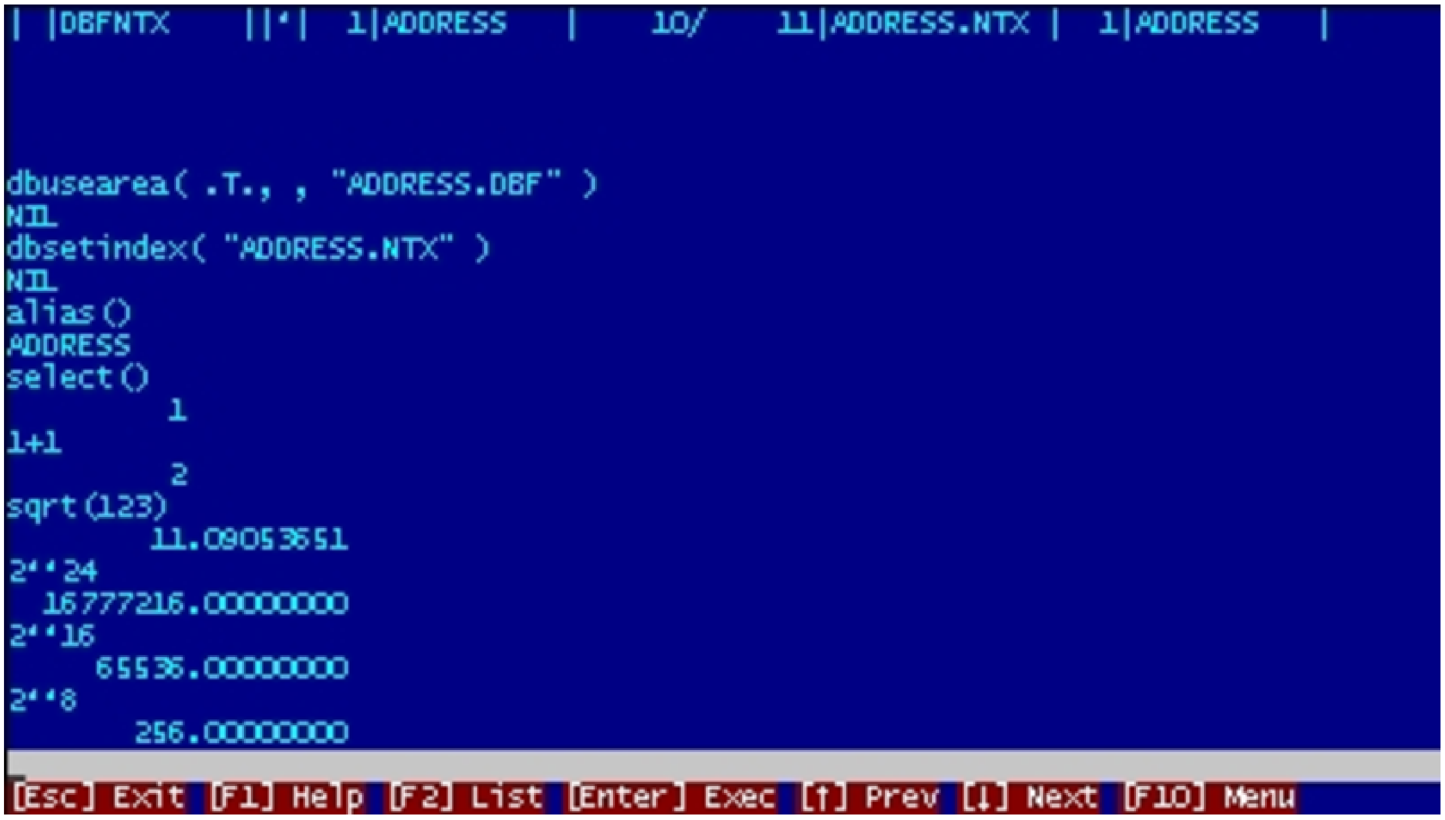
nanoBase [2] is a Dos program that works essentially as:

- a dot command line xBase,
- a menu driven xBase,
- a xBase program interpreter.

nanoBase 1997 is compiled in two versions: a small one to be used with old computers (x86-16 with 640 Kibyte RAM), and a second one to be used with better computers, at least i286 (or better) with 2 Mibyte RAM.

The dot command line

Figure u136.1. The dot line.



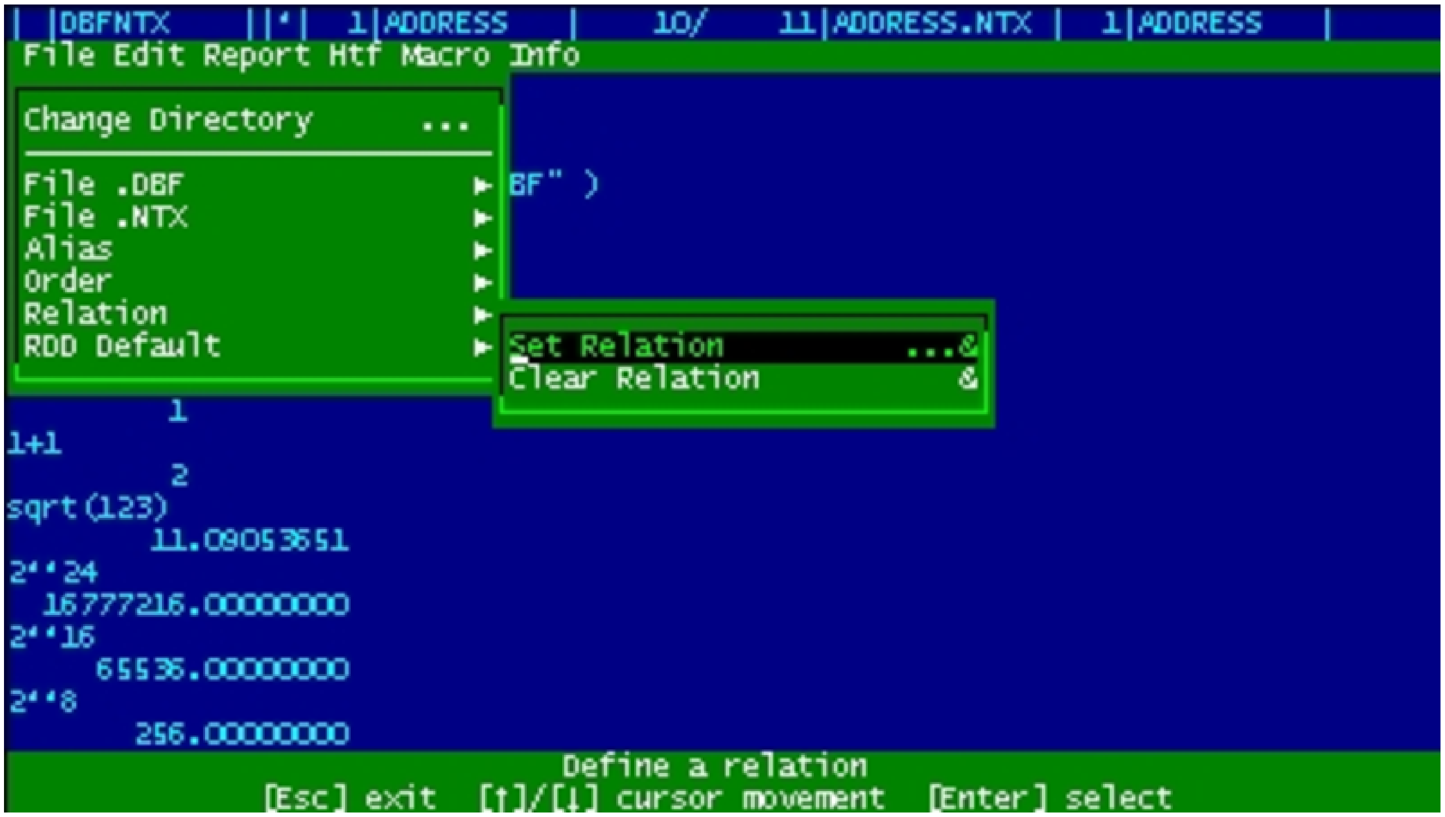
```
| |DBFNTX     ||*|  1|ADDRESS    |     10/     11|ADDRESS.NTX |  1|ADDRESS    |



dbusearea( .T., , "ADDRESS.DBF" )
NIL
dbsetindex( "ADDRESS.NTX" )
NIL
alias()
ADDRESS
select()
         1
1+1
         2
sqrt(123)
        11.08053651
2**24
  16777216.00000000
2**16
     65536.00000000
2**8
       256.00000000

[Esc] Exit [F1] Help [F2] List [Enter] Exec [↑] Prev [↓] Next [F10] Menu
```

The dot command line is the first face of nanoBase, the one that appears starting the program normally. It recalls the dot line command of the old xBases.

Please note that **nanoBase recognise only expressions** (that is: no commands).

The menu

Figure u136.2. The file menu.



```
| |DBFNTX    ||*|  1|ADDRESS   |    10/    11|ADDRESS.NTX |  1|ADDRESS   |
 File Edit Report Htf Macro Info
 Change Directory       ...
 ─────────────────────────
 File .DBF              ►BF" )
 File .NTX              ►
 Alias                  ►
 Order                  ►
 Relation               ►
 RDD Default            ► Set Relation        ...&
                          Clear Relation         &
          1
1+1
          2
sqrt(123)
         11.09053651
2**24
   16777216.00000000
2**16
      65536.00000000
2**8
        256.00000000
                              Define a relation
                [Esc] exit  [↑]/[↓] cursor movement  [Enter] select
```

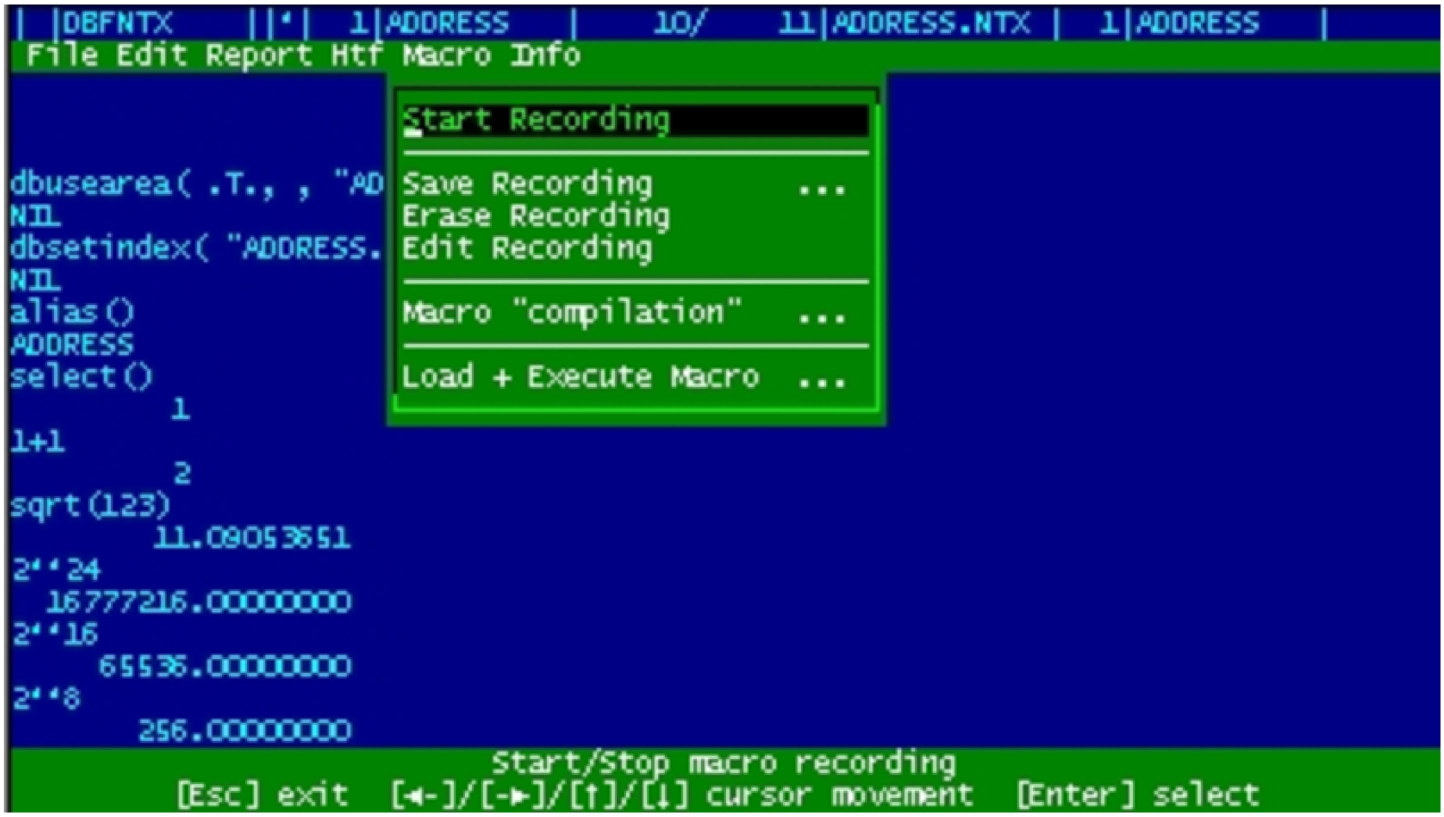
Pressing [*F10*] the nanoBase menu appears.

From this menu the operations are easier than writing all commands on a prompt line, but it is always possible to come back to the dot line to do an operation not available from the menu.

The macro recording, compiling and execution

«

Figure u136.3. The macro menu.



```
| |DBFNTX    ||•|  1|ADDRESS    |     10/     11|ADDRESS.NTX |  1|ADDRESS    |
 File Edit Report Htf Macro Info
                     Start Recording
                     ─────────────────────────
dbusearea( .T., , "AD Save Recording         ...
NIL                  Erase Recording
dbsetindex( "ADDRESS. Edit Recording
NIL                  ─────────────────────────
alias()              Macro "compilation"     ...
ADDRESS              ─────────────────────────
select()             Load + Execute Macro    ...
         1
1+1
         2
sqrt(123)
        11.09053651
2**24
  16777216.00000000
2**16
     65536.00000000
2**8
       256.00000000
                      Start/Stop macro recording
        [Esc] exit  [←-]/[-→]/[↑]/[↓] cursor movement  [Enter] select
```

nanoBase is able to record some actions made with the menu and all what is correctly typed from the dot prompt. This may be the begin for a little program (called macro inside nanoBase) that can be executed as it is (ASCII), or compiled into another format, faster to execute.

Macros for nanoBase are made with a reduced set of the Clipper syntax. The statements recognised from nanoBase are:

```
PROCEDURE procedure_name
    statements...
    [RETURN]
    statements...
ENDPROCEDURE
```

```
DO PROCEDURE procedure_name
BEGIN SEQUENCE
    statements...
    [BREAK]
    statements...
END
```

```
DO CASE
CASE lCondition1
    statements...
[CASE lCondition2]
    statements...
[OTHERWISE]
    statements...
END
```

```
WHILE lCondition
    statements...
    [EXIT]
    statements...
    [LOOP]
    statements...
END
```

```
IF lCondition1
    statements...
    [ELSE]
    statements...
END
```

- the '**FOR**' loop is not available (too difficult to implement),
- there may be no user defined functions (code blocks may be created instead),
- procedure calls cannot transfer variables,
- there are only public (global) variables.

Beside these limitations, there are many added functions to the standard language that make the programming easier.

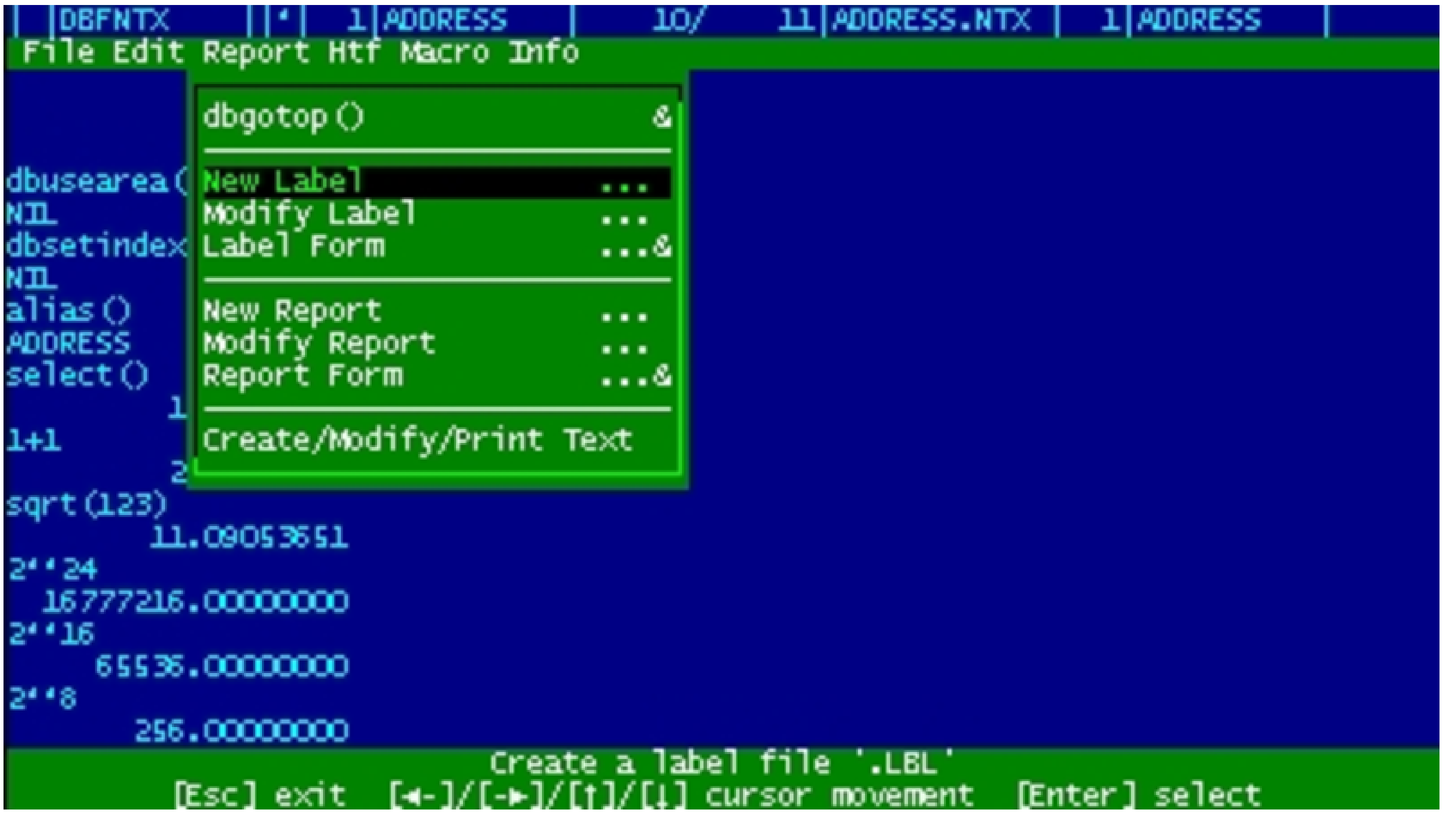
All you need is inside '**NB.EXE**':

- the utility to handle manually the data,
- the macro compiler,
- the macro executor.

The report system

«

Figure u136.4. The report menu.



```
| |DBFNTX ||*| 1|ADDRESS |  10/  11|ADDRESS.NTX |  1|ADDRESS |
dbusearea(
NIL
dbsetindex(
NIL
alias()
ADDRESS
select()
         1
1+1
         2
sqrt(123)
       11.09053651
2**24
   16777216.00000000
2**16
      65536.00000000
2**8
        256.00000000
                    Create a label file '.LBL'
      [Esc] exit  [←]/[→]/[↑]/[↓] cursor movement  [Enter] select
```

nanoBase can handle label ('.LBL') and form ('.FRM') files in the dBaseIII format. Labels and forms may be created and edited inside nanoBase. Beside these old report system there is another way to make a little bit complicated reports without making a complex macro: it is called RPT.

A RPT file is a ASCII file with text mixed with code. The text may contain variables (usually a field or an expression containing fields).

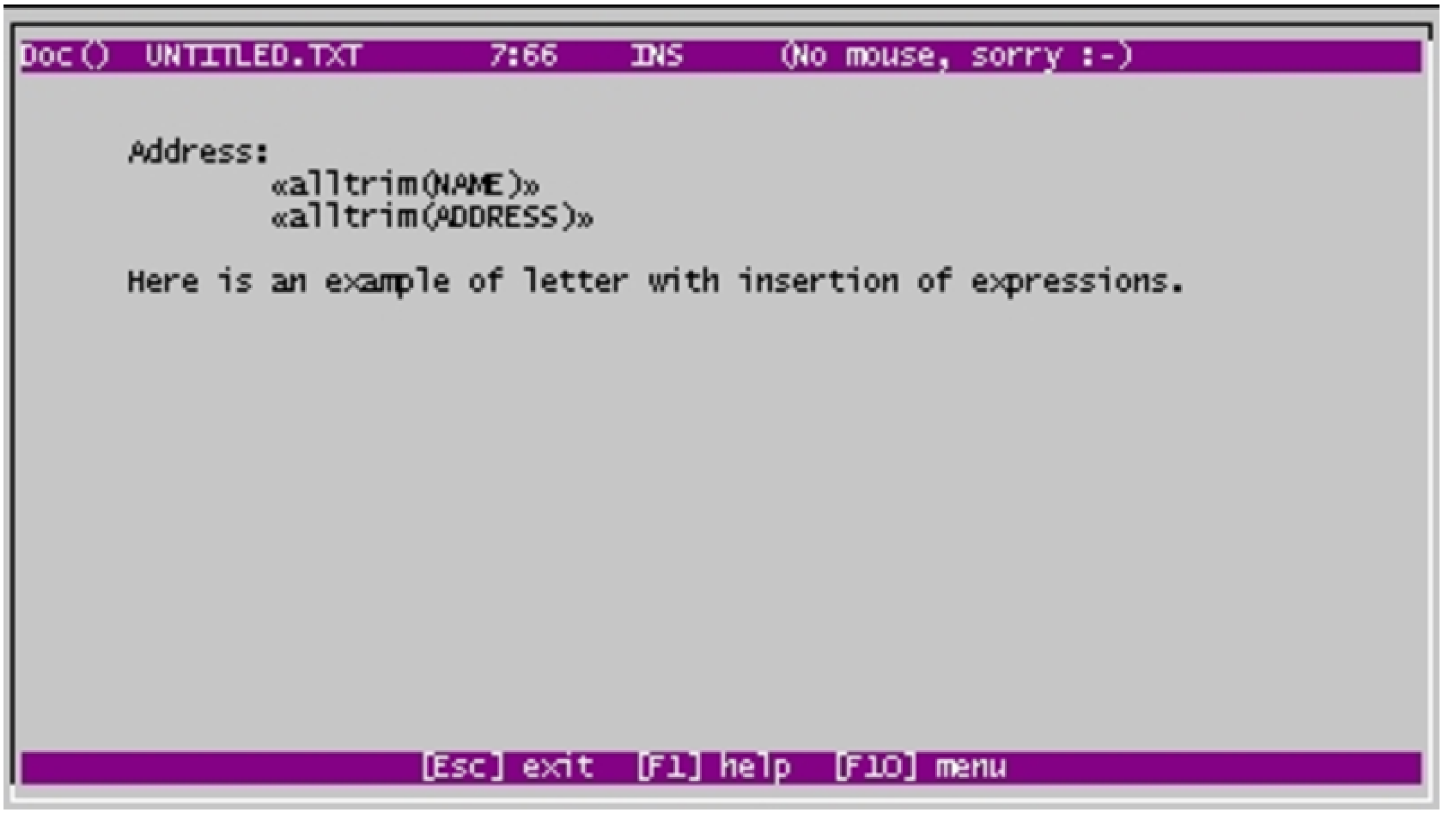
To make a complex report some work is needed, but surely less than the time needed to make a report program.

The main purpose of it was to be able to print text with variables (typically names and addresses) for every record of a particular '.DBF' file. Now the RPT system makes something more.

The integrated text editor

«

Figure u136.5. The integrated text editor.



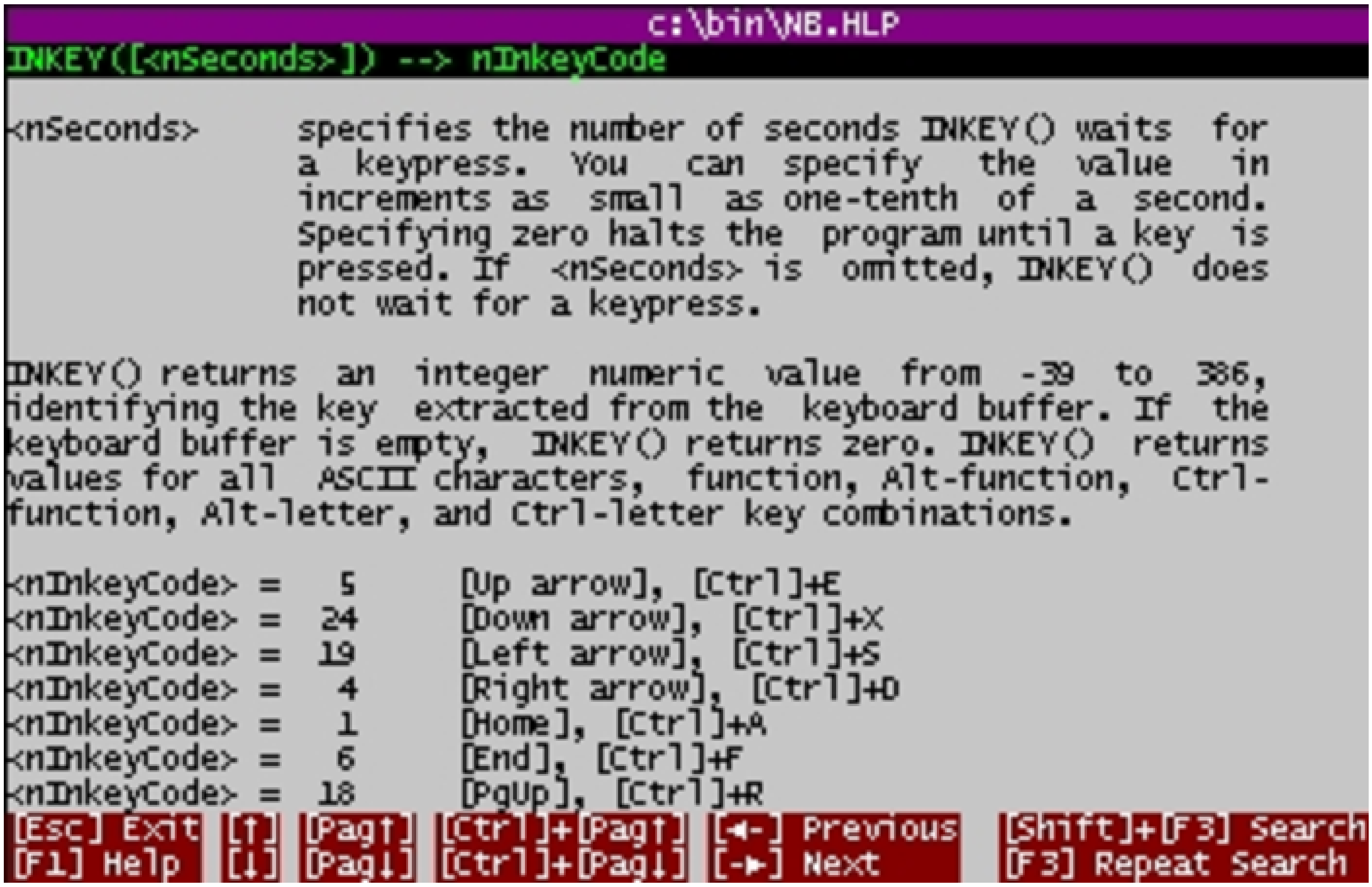
```
Doc()  UNTITLED.TXT        7:66    INS      (No mouse, sorry :-)

      Address:
              «alltrim(NAME)»
              «alltrim(ADDRESS)»

      Here is an example of letter with insertion of expressions.

                        [Esc] exit  [F1] help  [F10] menu
```

nanoBase contains an integrated text editor not particularly good, but very usefull for RPT files (as the expression insertion is very easy with the use of the [*F2*] key) and whenever there isn't any other editor there.

The internal documentation

«

Figure u136.6. The internal documentation.



```
                              c:\bin\NB.HLP
INKEY([<nSeconds>]) --> nInkeyCode

<nSeconds>     specifies the number of seconds INKEY() waits  for
               a  keypress.  You   can  specify  the  value   in
               increments as  small  as one-tenth  of  a  second.
               Specifying zero halts the  program until a key  is
               pressed. If  <nSeconds> is  omitted, INKEY()  does
               not wait for a keypress.

INKEY() returns  an  integer  numeric  value  from  -39  to  386,
identifying the key  extracted from the  keyboard buffer. If  the
keyboard buffer is empty,  INKEY() returns zero. INKEY()  returns
values for all  ASCII characters,  function, Alt-function,  Ctrl-
function, Alt-letter, and Ctrl-letter key combinations.

<nInkeyCode> =   5      [Up arrow], [Ctrl]+E
<nInkeyCode> =  24      [Down arrow], [Ctrl]+X
<nInkeyCode> =  19      [Left arrow], [Ctrl]+S
<nInkeyCode> =   4      [Right arrow], [Ctrl]+D
<nInkeyCode> =   1      [Home], [Ctrl]+A
<nInkeyCode> =   6      [End], [Ctrl]+F
<nInkeyCode> =  18      [PgUp], [Ctrl]+R
[Esc] Exit  [↑] [Pag↑] [Ctrl]+[Pag↑] [◄-] Previous  [Shift]+[F3] Search
[F1] Help   [↓] [Pag↓] [Ctrl]+[Pag↓] [-►] Next      [F3] Repeat Search
```

nanoBase's documentation si translated also inside the HTF format: '`NB.HLP`'. Pressing [*F1*], normally, a contextual piece of the manual appears.

Some standard functions have its own internal help, contained inside the '`.EXE`' file. This was made to help programming with nanoBase.

Download it

Here is the 1997 edition of nanoBase.

- EXE for small computers.
  ftp://ftp.simtel.net/pub/simtelnet/msdos/database/nbase7a1.zip
  http://www.google.com/search?q=nbase7a1.zip

- EXE for i286 with more than 2 Mibyte.
  ftp://ftp.simtel.net/pub/simtelnet/msdos/database/nbase7a2.zip
  http://www.google.com/search?q=nbase7a2.zip
- Runtime for small computers.
  ftp://ftp.simtel.net/pub/simtelnet/msdos/database/nbase7a3.zip
  http://www.google.com/search?q=nbase7a3.zip
- Documentation in many different formats.
  ftp://ftp.simtel.net/pub/simtelnet/msdos/database/nbase7a4.zip
  http://www.google.com/search?q=nbase7a4.zip
- Macro programming examples.
  ftp://ftp.simtel.net/pub/simtelnet/msdos/database/nbase7a5.zip
  http://www.google.com/search?q=nbase7a5.zip
- Source for version 96.06.16, without mouse support (1996).
  ftp://ftp.simtel.net/pub/simtelnet/msdos/database/nbase7a6.zip
  http://www.google.com/search?q=nbase7a6.zip
- Source for version 1997.
  ftp://ftp.simtel.net/pub/simtelnet/msdos/database/nbase7a7.zip
  http://www.google.com/search?q=nbase7a7.zip

Bugs and known problems

«

Here is the list of known bugs and problems.

- Comparison with floating point numbers may fail. It is better to convert numbers into string before comparing them.

- Macros may be contained inside ASCII files or a ‘‘compiled’’ ‘`.DBF`’ file. In the second case, when nanoBase executes the macro, a work area (the last available one) is used, so it should not be closed or the macro execution will be stopped. A ‘**`dbcloseall()`**’ will stop execution of the macro. In substitution of ‘**`dbcloseall()`**’, ‘**`DBCLOSE()`**’ should be used.
- To simplify the macro interpretation, lines such as this:

  ```
  qqout( "You can't do that // you can't do that!" )
  ```

  will generate an error as the interpreter will read only:

  ```
  qqout( "You can't do that
  ```

- nanoBase works good also if you have a screen configuration that permits you to show more than the usual 80 columns and 25 lines, but the library used to handle the mouse is not able to work outside the 80×25 area.

[1] This material appeared originally at ‘`http://www.geocities.com/SiliconValley/7737/nanobase.html`’, in 1997.

[2] **nanoBase** GNU GPL