Clipper Programming Language

From Wikipedia, the free encyclopedia.(*)

Clipper (or **CA-Clipper)** is a compiler 16 bits of language xBase environment for DOS . It was created in 1984 with the purpose of being a compiler for Ashton-Tate dBase , a database manager very popular in his time.

This is a derivation of the Clipper Summer and after being acquired by Computer Associates reached version 5.3b implemented by a graphical interface compatible with the MS-Windows 3.11, and for a subset of supported languages C and assembly , which made possible a prototype of Object Orientation .

When Computer Associates stopped support this language, it was intended for application development platforms for MS-DOS and offered libraries for network support.

Why make use of a standard language originally developed by Ashton Tate and operating systems from existing CP/M , has a very different syntax of the languages most current. In his day, was considered an elegant and intuitive language, using small verbs and abbreviations, symbols and structuring.

Their compilers generate executable which in 2008 would be considered tiny, extremely fast and, for most users, with a little friendly interface. Again, in his time, was one of the most versatile and enabled the creation of fully integrated systems with images, sounds and video, and has already used the concepts of hyperlink (via standard RTF), context help and instantiating objects (although primitives, via Code Blocks), which, except for the C language, has only been achieved years later by the competition.

Another important feature was the inclusion of Rushmore, now owned by Microsoft, indexing tables for your data, making it one of the languages better performances in this area.

But original systems created with this language require constant adjustments to become usable in most modern operating systems. And as there is no official support for it, groups of users and developers solve problems that arise with the constant evolution of information for themselves by OpenSource libraries, patches, and other creative solutions portings.

Some more features of the language:

- Preprocessor source code;
- Compilers for high-performance;
- Interactive debugger;
- Graphical IDE (optional, requiring the MS-Windows ® installed);

- Support VGA graphics modes (with the appropriate drivers);
- Mouse support (driver with the manufacturer) integrated libraries will input data;
- Generation of executable that used the real or protected mode memory (picking up one of these compilers for specific characteristics);
- Generating Overlay modules (roughly equivalent to the dynamic link libraries), decreasing the size of your executables and memory usage;
- Two real objects to MS-DOS (and Get Tbrowse) to develop screens with masses of data and data inputs respectively;
- Accelerator keys (equivalent to the shortcut keys);

The whole object is dispensable to Get developer interface items such as check-box, list-box, radio buttons, radio buttons, scroll bars, menu bars and menu items, among others. These items were visible in MS-DOS through semi-graphic characters from the ASCII table extended.

The version 5.03 update has undergone a radical, having been developed using the C programming language from Microsoft and thereby making possible the use of arithmetic processors, if they were present on the computer. Thus, applications were performed with up to 30% more performance will, without any change in the source code.

Index

[secondary]

History

Legend tells that two friends were having lunch at a restaurant called Seafood Nantucket Lighthouse, discussing how frustrating was the fact that Ashton-Tate refuses to create a compiler for your main product. The low processing speed of dBase compared to compiled applications was striking. They began to discuss the idea of creating a compiler and start a company to market it. Clipper The name came from a picture on the wall of the restaurant that showed one of these stylish and fast merchant ships. The company name was borrowed from a choice restaurant name.

Upon its establishment, the Nantucket Clipper is basically proposed to be the best compiler for dBase existed. In the Summer '87 version had the words "dBase III ® Compiler." The versions were:

Nantucket Corporation, on behalf of seasons, sold as "dBase compilers"

- Nantucket Clipper Winter '84 - released May 25 1985
- Nantucket Clipper Summer '85 - released 1985
- Nantucket Clipper Winter '85 - released January 29 1986
- Nantucket Clipper Autumn '86 - released October 31 1986
- Nantucket Clipper Summer '87 - released December 21, 1987

Nantucket Corporation, Clipper 5

- Nantucket Clipper 5.00 - released 1990
- Nantucket Clipper 5.01 - released April 15, 1991
- Nantucket Clipper 5:01 Rev.129 - released March 31 1992

Computer Associates

- CA-Clipper 5.01a -
- CA-Clipper 5.20 - released February 15, 1993
- CA-Clipper 5.2a - released March 15 1993
- CA-Clipper 5.2b - released June 25 1993
- CA-Clipper 5.2c - released August 6 1993
- CA-Clipper 5.2d - released March 25 1994
- CA-Clipper 5.2e - released February 7 1995
- CA-Clipper 5.30 - released June 26 1995
- CA-Clipper 5.3a - released May 20 1996
- CA-Clipper 5.3b - released May 20 1997

With version 5, Clipper began the process of untying the dBase , becoming a programming language with its own line of evolution. Version 5 added features missing in dBase, including a visual debugger, structured exceptions, RDD (a type of prehistoric ODBC), new types, new functions for managing arrays and a preprocessor that allowed the language could be extended form nearly limitless in 16-bit environment.

Prior to Computer Associates purchase Nantucket, Nantucket German office had begun a project informally known as "ASPEN", though internally it was simply called Clipper for Windows. The project represented a break with previous versions of Clipper as it introduces the concept of object-orientation **(**OO**)** and supports the graphical environment of Microsoft without worrying about backward compatibility. Among the new features, performance was comparable to the C + + language with a much more affordable and that had a huge base of potential programmers coming dBase, Clipper and other environments xBase**.**

The CA decided that he would enter the fray in earnest by a sizable chunk of the nascent market for Windows and programming acquired Nantucket because of VO (Visual Objects), betting big on what was one of the greatest failures in the area of technology. The lack of compatibility with previous versions led many developers to migrate to the new tool Borland, called Delphi.

The Clipper, which was renamed after acquiring CA-Clipper, still saw an edition that used the memory management of Windows interface while maintaining the character, but it was officially retired in favor of the new product, VO.

Although today is considered an obsolete language since stopped evolving after version 5.3 and treating the VO from a separate product and stillborn, Clipper still has a reasonable basis for programmers known by a nickname cşippeiros". Projects open-source as Harbour Project continue to support the standard while xBase are oriented modern graphical environments, although no official support of CA, which holds the rights to the Clipper.

In summary, the Clipper allows dynamic applications with data files, making them easier and faster than those developed in a traditional programming language such as Cobol, Basic or Pascal. With a simple, modern and efficient programming language, allows the orderly and logical linking of his commands quickly enabling the definition of programs with a high degree of complexity and sophistication, even allowing interactions with other languages like "C" and Assembly, which gives the flexibility needed for professional use.

SUMMER 85 - in this version, the CLIPPER was fully compatible with the 1.0 version of dBASE III, and became very close to him, however, with some additional features like:

- Increased ability to manipulate files and variables; - Construction of "HELP" to the user; - Multiple relationships between files; - Creation-of-function user (UDF's) - New commands and functions that did not exist in DBASE III.

WINTER 85 - still remained compatibility with dBASE III appeared however some implementations, among which the main ones are:

- Indexed variables: vectors; - Rise of the command @ ... PROMPT (menu bar) - New functions for manipulating MEMO fields. Soon after the release WINTER 85 CLIPPER was released on the counterattack, the DBASE III PLUS which included the various commands and functions that already had CLIPPER only with the main novelty, it was able to work in local network environment. To accompany the release of DBASE, has released version:

AUTUMN 86 - in this version, the CLIPPER also went to work in LAN environment, gained new commands and functions, but many of the new features of DBASE III PLUS were implemented on a provisional basis, through auxiliary routines written in C language and Assembly . The compatibility with dBASE III PLUS still existed.

SUMMER 87 - this version takes place two important factors for the development stage of the CLIPPER, they are:

- Change the C compiler, through which it was built. - Decision to separate once the DBASE. Thus the CLIPPER really became a tool for the construction of systems professionals. Besides a considerable amount of new commands and functions, the architecture of CLIPPER was practically open. It became possible to write any function using Microsoft C and linkedit it directly with libraries and object modules generated by CLIPPER.

VERSION 5.0 - a trend already observed in version 87 SUMMER confirmed. The compatibility with the language DBASE, despite maintained just became one historical circumstance. The new structure CLIPPER programming and sophisticated new features based on the structure of the programming language C and trends in object-oriented programming, indicate a definite departure from the standard DBASE.

NEWS FROM VERSION 5.0 - Among the many new features that version 5.0 brought us CLIPPER, we highlight a few below:

- Access to the compiler preprocessor (policies) - New compiler with more optimized features and options; - New linker (RTlink), which enables the creation of dynamic overlays; - Help "online" for programmer (Norton Guide) , - Newcomers; - Definitions of functions for user (UDF's) - more efficient debugger; - New types and classes of variables; - Multidimensional Arrays, etc..

Use Descending

Both **dBase** as the **Clipper** are products of a time when personal computers were disconnected, and the database was a set of disk files accessed by only one user. Both programs operate in practice as a library linked to the final program, monolithic, which directly accesses the files containing the data, without intermediation (as in the case of DBMS).

With the advent of computer networks, it has become possible to use shared disks to access these files directly, but doing what the programmer had to manage and resolve various issues related to access shared files and records.

Currently, although many programs still use these languages, the use of a DBMS is recommended, which leads gradually to the abandonment of this technology.

For Clipper RDDs have been developed that allow the use of DBMS, such as ADS Advantage Database Server, RaSQL / b for Btrieve / Pervasive and UltiRoute, which allows access to any DBMS via ODBC.

File Extensions Handled By Clipper

The CLIPPER like any other programming language has its own files and extensions to be easily recognized by a programmer. Just below, are broken down the various files handled by CLIPPER.

- .PRG : source program files
- .CH : header files or include files
- .OBJ : object files
- .LIB : library files
- .TMP : Temporary Files
- .PPO : preprocessor file
- .EXE : executable files
- .DBF : data files
- .DBT : memo field files
- .NTX : index files
- .MEM : memory variables files
- .LBL : label definition files
- .FRM : reports definition files
- .FMT : format files
- .CLP : script files
- .LNK : link files
- .PLL and PLT : pre-linked library files
- .OVL overlay files
- .MAP memory allocation files

Additional Information

With the clipper is possible:

• Create, organize, sort, copy, select and relate sets of files that make up the database;

• Add, change, delete, and list view or selectively global information contained in the data files;

• Generate standard reports, place automatically sums, aggregations, counts and arithmetic operations on the data values stored in the archives;

• Format data entry screens in video and generate reports, tables and listings complex in the printer, according to user needs;

• Produce Information Systems complete and integrated resources and sophistication only found in more modern software that is currently fighting the fabulous marketing microcomputer.

In summary, the Clipper allows dynamic applications with data files, making them easier and faster than those developed in a traditional programming language such as Cobol, Basic or Pascal. With a simple and efficient programming language, allows the orderly and logical linking of his commands quickly enabling the definition of programs with a high degree of complexity and sophistication, even allowing interactions with other languages such as C and assembly, which gives it the flexibility to professional use.

Procedural Paradigm: Clipper is owned by the procedural paradigm (as Pascal, C, Ada, COBOL, FORTRAN and Clipper). Procedural languages are those in which our code is divided into subroutines (procedures) or function (functions). A procedure is a function of no return (can be seen as a function that returns nothing, void). If you really want to modularize your program, the best you can do is split it into several subroutines / functions, and a higher level in various libraries. Prior to structured programming, was used in the GOTO feature codes, which made most programs unreadable; the famous "spaghetti code".

System Types Used by Language

Clipper has a dynamic typing, but surprisingly forte. Following code snippet goes by the compiler, but gives error on execution:

```
Local a, b

a = 1

b = 2

b = "x"

? a + b
```

The Clipper 5 introduced the policy "local", which allows you to declare variables at the beginning of the function or procedure. It is a step toward static typing, the plan was to make the Nantucket Clipper more like C throughout its evolution. The comparison with dynamically typed languages, that resolve names at run-time, is truly cowardly, but for completeness:

```
a = table → field     // Clipper

$ A = $ row → field   // PHP

a = # # table.field   // Python
```

The Clipper natively supports the above syntax for DBF tables, but not allow me to define a new type with this syntax. Since PHP and Python types with attributes allow you to create dynamic, user-friendly.

Example Compiler

The Harbour is a free software compiler for the Clipper language (the language that is implemented by the compiler CA-Clipper). The Harbour is a multi-platform compiler and know that compiles and runs on MS-DOS, MS-Windows, OS / 2 and GNU / Linux. The main difference to other compilers Harbour dBase is that it is free software.

Examples of syntax

- **'NUMERIC:** Accepts only numbers with or without decimal places (decimals). The decimal point of the account size (width).

Example: 999.99 (width 6, decimals 2)

- **CHARACTER:**

Accepts any character. (Maximum size of 256 characters (?).)

- **DATE:** Accepts only dates.
- **LOGICAL:** Accepts only True (. T.) or False (. F.)
- **MEMO:** Accepts letters and numbers, is like a field type Character with size up to 64Kb. Usually used to store "Note" or other information that need to be detailed. By adding a memo field on your DBF is a mirror created it with the extension. DBT. (If the RDD used for the native Clipper DBFNTX). You can only edit a memo field with MEMOEDIT function ().

Example: The DBF "clients" has a memo field observations to the customers, then there will be:

```
CLIENTES.DBF

CLIENTES.DBT
```

- **NUMERIC:** Accepts only numbers with or without decimal places (decimals). The decimal point of the account size (width).

Example:

```
nTotal = 0

nCAMPO = 125
```

- **FEATURES:** Accepts letters and numbers. Maximum size of 256 characters.

Example:

```
cName = SPACE (35)

cSName = "ANDERSON"
```

- **DATE:** Accepts only dates. Consider using the SET DATE BRITISH to set the default date equal to the Brazilian dd/mm/yy. Also consider using the SET EPOCH TO 1980 to resolve the "Millennium Bug", see more on Know-how - Y2K Bug of the Year 2000.

Example:

```
dDate = CTOD ("")

dToday = DATE ()

dDTFINAL = CTOD ("06/08/2005")
```

- **LOGICAL:** Accepts only True (. T.) or False (. F.)

Example:

```
lCompleted = .T.

lError = .F.
```

- **MEMO:** Memo type variable does not exist, it would be a character variable with more than 256 bytes that only fit into a Memo field.

Example:

```
Mobs = MEMOREAD ("NOTAS.TXT")
```

- **ARRAY:** Also called a matrix or vector. Are grouped in various fields as a list. It is a data structure that contains a series of ordered data, called "elements". The elements are referenced by ordinal number, the first element is 1, 2 ... The second element may be of any type, character, numeric, etc. date. See details in ALL ABOUT ARRAYS.
- **Codeblock:** It is a special kind of variable that stores a piece of compiled code.

Example:

```
@ 10.10 SAY "CODE:" GET WCOD PICT "999999" VALID;
 EVAL ({| | WCOD := STRZERO (WCOD, 6). T.})
```

Examples of Basic Commands Repetition and Loops

@. . . SAY. . . GET

Purpose: Create and run a new object GET (data entry), putting it on screen display.

Syntax : @ <nRow>, <nCol>

```
        [SAY <exp>
           [PICTURE <cSayPicture>]
           [COLOR <cColorString>]]
        GET <idVar>
           [PICTURE <cGetPicture>]
           [COLOR <cColorString>]
           [WHEN <lPreExpression>]
           [RANGE* <dnLower>, <dnUpper>] |
           [VALID <lPostExpression>] Example
```

```
@ 15.10 SAY 'FATEC'
xcod :=  0
18.10 @ GET xcod
READ
```

STR ()

Purpose: Convert a numeric expression to a character expression.

Syntax: STR (<nValue >, <length>, <decimais>).

Example

```
SALARY :=  3020.29
? STR (SALARY, 4)    //  result: 3020
? STR (SALARY, 8.3)  //  result: 3020,290
```

VAL ()

Purpose: Convert a character expression to a numeric value.

Syntax: VAL (<string>).

Example

```
SALARY :=  "2929.20"
```

```
? VAL (WAGE)  //  result: 2929.20

TEST :=  "COMPUTER"

? VAL (TEST)  //  result: 0
```

FOR. . . NEXT

Purpose: Performs a control structure, a certain number of times.

Syntax

FOR <counter> := <start> TO <stop> STEP <increametnt>

..... <instructions>

[EXIT]

..... <instructions>

[LOOP]

NEXT

Example

```
1) FOR I := 1 TO 100

      @ 15.10 SAY 'COUNTER' + STR (I, 3)

   NEXT

2) FOR J := 100 TO 500 STEP 10

     @ 18.05 SAY 'VALUE OF J's' STRZERO + (J, 3)

   NEXT
```

DO WHILE ... ENDDO

Purpose: Performs a control structure while a condition is true.

Syntax

```
DO WHILE <condition>

<instructions>

[EXIT]

[LOOP]

ENDDO
```

Examples

1)

```
DO WHILE. T.

xnumero :=  0

11.10 @ say 'Enter a number'

@ 11.20 get xnumero

READ

if empty (xnumero)

exit

endif

13.10 @ say 'the number entered was "+ strzero (xnumero, 3)

ENDDO
```

2)

```
xresp :=  'S'

XRESP # DO WHILE 'N'  //  # or <> symbols of different

xnome := space (40)

11.10 @ say 'Name'

@ 11.25 get xnome

READ

if lastkey() == 27  //  == exactly

exit

endi

15.18 @ say 'The name was typed:' + xnome

xresp :=  space (01)

20.10 @ say 'Are you sure?

@ 20.30 xresp get picture '!'

READ
```

```
ENDDO
```

Examples of code

```
?  "Hello World!"

USE Customer NEW SHARED

CLEAR SCREEN

DO WHILE LASTKEY()!= 27

   @ 01, 0 SAY "Code" GET client->code ;
            PICT "999999" VALID client->code > 0
   @ 03, 0 SAY "Name" GET client->name VALID !empty(client->name)
   @ 04, 0 SAY "Address" GET customer->address

   READ

ENDDO
```

Note: The above code violates a basic rule of using tables in shared mode.

@... CLEAR

Purpose: Delete (clear) only a specific area of the screen.

Syntax: `@ <Row_Home>, <Col_Home> CLEAR [TO <Row_End>, <Col_End> ]`

Example:

```
SET COLOR TO B+/W          //  changes the color
CLS                        //  equivalent to CLEAR, ie clean the entire screen
SET COLOR TO W+/N          //  sets a new standard color
@ 10, 10 CLEAR TO 20, 20   //  clear a region of the screen
@ 10,10 TO 20, 20 DOUBLE   //  draw a frame (frame)
```

@...PROMPT

Purpose: Build a menu of selectable options on the screen.

Syntax: @ <nRow>, <nCol> PROMPT <cMenuItem> [MESSAGE <cExpression>]

Example:

```
 LOCAL nOption :=  1
 SET WRAP ON            //  enable the scroll between the extremes of menu
 SET MESSAGE TO CENTER 23 // output message of the screen line 23
 DO WHILE. T.
    CLEAR  //  CLEAN SCREEN
    //  define variables to facilitate the coordinated menu
   nRow :=  8
   nCol :=  32
   //  Set the screen
   @ 01,01 TO 24,79 DOUBLE
   @ 02,02 TO 04,78
   @ 03,60 SAY DATE ()
   @ 03,70 SAY TIME ()
   // Menu bar
   @ nRow,    nCol PROMPT "INCLUSION"  MESSAGE "inclusion of data"
   @ nRow +1, nCol PROMPT "change"     MESSAGE "data change"
   @ nRow +2, nCol PROMPT "QUERY"      MESSAGE "QUERY DATA"
   @ nRow +3, nCol PROMPT "BYPASS"     MESSAGE "EXCLUSION OF DATA"
   @ nRow +4, nCol PROMPT "REPORTS"    MESSAGE "SYSTEM REPORTS"
   @ nRow +5, nCol PROMPT "UTILITIES"  MESSAGE "SYSTEM UTILITIES"
   @ nRow +6, nCol PROMPT "END"        MESSAGE "THE RETURN OS"

   // Execute the menu and toolbar controls
   MENU TO nOption
```

```
   DO CASE  //  do cases
      CASE nOption 1
         DO PROG1
      CASE nOption = 2
         DO PROG2
      CASE nOption 3
         DO PROG3
      CASE nOption = 4
         DO PROG4
      CASE nOption = 5
         DO PROG5
      CASE nOption = 6
         DO PROG6
      CASE nOption = 7
         CANCEL  //  cancels the program execution
   ENDCASE
   INKEY(0)  //  waits a key
ENDDO
```

See also

- xBase
- dBase
- Visual FoxPro
- Harbour

External links

- CA-Clipper Website Language Clipper, forum, downloads, social networking etc. - FREE
- Forum Clipper On Line Sharing information about xBase - Free
- FlagShip Compiler compatible Clipper for Linux, Unix and Windows - Commercial
- Alaska xBase + + Compiler for Windows compatible Clipper - Commercial
- Harbour Design Compiler compatible Clipper for Linux, Unix and Windows - Open Source
- xHarbour Compiler compatible Clipper for Linux, Unix and Windows - Open Source or Commercial
- [1] - Harbour Project
- [2] - error codes Clipper
- Clip Compiler of a Russian company Clipper compatible for Linux - Open Source
- ADVPL [3] - Language itself originates from Clipper

(*) http://pt.wikipedia.org/wiki/CA-Clipper (Translated by Google)