COBGDB - The GnuCOBOL TUI DEBUGGER / ANIMATOR

HOW TO USE COBGDB

Table of Contents

1. Introduction 3

1.1. Installing the debugger COBGDB on Windows 4
1.2. Compile and Debug GnuCOBOL programs 5
1.3. Main Commands 6

2. Tutorial - Sample Debugging Session 7

2.1. Help Command 9
2.2. Run Command 10
2.3. Step Command 12
2.4. Go Command 14
2.5. Display Variables 15
2.6. Show Command 16
2.6.1. Edit subCommand 17
2.7. Variable Command 18
2.7.1. Enter subCommand 18
2.7.2. Edit subCommand 20
2.7.3. Return subCommand 21
2.8. Step Command 22
2.9. Focus Command 24
2.10. Pop-up Variable windows 26
2.11. File Command 27
2.12. Run Command 30
2.13. Window Size command 31
2.14. Quit Command 33
2.15. Attach Command 34

3. Other Line Commands 35

3.1. Debugging a pre-compiled Program 35
3.2. COBGDB Version 35

4. Document Change Log 36

1. Introduction

COBGDB is a TUI (Text User Interface) application, programmed in C, designed to assist in animate and debugging GnuCOBOL programs using **GDB** (the GNU Debugger at https://www.gnu.org/software/gdb/).

The COBGDB project is hosted at ***https://github.com/marcsosduma/cobgdb***

Very important: you don't need to know how to use the GDB product and its many commands (https://www.sourceware.org/gdb/).
COBGDB has its own user interface (described in this document) that is very simple to use and is responsible for interfacing the underlying GDB which is the real debug and animate engine but operates practically in a transparent way to the GnuCOBOL developer.

The COBGDB application is based on the extension for Visual Studio Code (VSCode) created by Oleg Kunitsyn, which can be found on GitHub: *https://github.com/OlegKunitsyn/gnucobol-debug.*

At https://github.com/marcsosduma/cobgdb in the Windows subdirectory, the executable program cobgdb.exe for this operating system is available and ready to use.

To compile COBGDB from C source code on Windows, you can use MinGW.
The Makefile is configured to generate the program **`cobgdb.exe`** for both Windows and Linux.

1.1. Installing the debugger COBGDB on Windows

On Windows, just download **`cobgdb.exe`** from following folder:
https://github.com/marcsosduma/cobgdb/tree/main/windows .

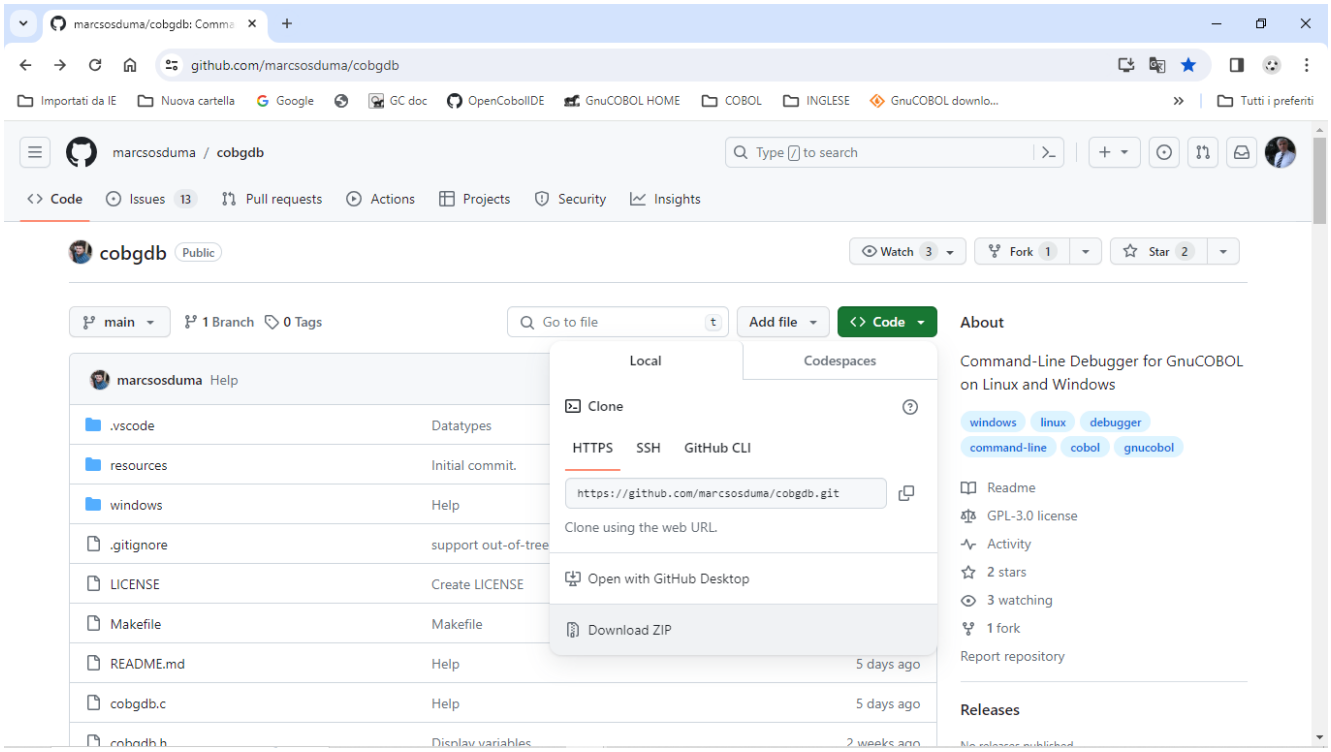
As an example you can put **`cobgdb.exe`** into the "**`bin`**" folder of your GnuCOBOL installation (the same folder where the GnuCOBOL compiler **`cobc.exe`** is located)

or

first install MinGW (Minimalist GNU for Windows).
Then execute the make ('mingw32-make' for Windows) command to compile the code from C source.

Note: if you have security problems downloading an .exe file then you can try download the entire repository with the following github button <> Code v **-->** *Download ZIP*

Then unzip the file and copy cobgdb.exe to the "bin" folder of your GnuCOBOL installation (the same folder where the GnuCOBOL compiler cobc.exe is located)



1.2. Compile and Debug GnuCOBOL programs

Compile and run a debugging session of the sample program using the following command:

```
cobgdb customer.cob -x -lpdcurses
```

Source code of customer.cob used also for following tutorial is at:
https://github.com/marcsosduma/cobgdb/tree/main/windows

Note: '**`-lpdcurses`**' is an instance of an argument that can be indirectly passed to '**`cobc`**' by '**`cobgdb`**,' even if it is not used by '**`cobgdb`**' itself.

or, other example for **`cobc`** parameters , use : **`cobgdb customer.cob -x -Tcustomer.txt`** .(-T creates a compilation list output into **`customer.txt`** file)

COBGDB takes one or more programs with COB or CBL extension as parameters and runs the GnuCOBOL compiler with the following format:

```
cobc -g -fsource-location -ftraceall -v -free -O0 -x prog.cob prog2.cob ...
```

To debug multiple programs, use COBGDB with the following syntax:

```
cobgdb prog.cob subprog1.cob subprog2.cob . . .
```

This will create a single prog.exe executable.

You can run GDB/GDBSERVER remotely using the "A" (Attach) key.
COBGDB will prompt you to provide the server and port in the format server: port or the PID of the application.
Example:

- localhost: 5555
- 9112

1.3. Main Commands

Cmd		Description	
?	Help	Show the HELP window	
R	Run	This is the first command to always use to start a debugging session. Start and Run the program until a breakpoint is encountered, also when, by default, the first Breakpoint is at first program statement.	
B	Breakpoint	Set or Unset a Breakpoint at a specific line of the Procedure Division code.	
C	Cursor	Runs the program until it reaches the selected line at Cursor location.	
N	Next	Runs the program until the next line but does not enter a subroutine executed by CALL or PERFORM.	
S	Step	Runs the program until the next line. If needed it goes into a subroutine executed by CALL or PERFORM.	
G	Go	Continues the program execution until it encounters a stopping point: a breakpoint, the end of the program, or the return from a subroutine (PERFORM / CALL).	
J	Jump	Ask for a line number and Runs the program until it reaches that line.	
V	Variables	Displays a window with a list of all variables and their content for the running program. From this window you can also change the content of variables.	
H	sHow	Displays a window with a list of variables and its content from the cursor selected line. From this window you can also edit / change the content of variables from the selected line. Right-click on a row is same as command "H".	
D	Display	Automatic Display of variables of current and previous statement in execution during a debugging / animation session is settled to OFF or ON. At program start is OFF.	
O	fOcus	When program encounters an ACCEPT verb, this command switch the focus from debugging / animation screen to the application screen to accept an user input and then automatic go back to the debugging / animation screen.	
F	File	When COBGDB is executed with more than one program, allows selecting one of those source file to manage debugging commands.	
A	Attach	Attach to GDBSERVER or to an Application PID.	
W	Window	Switch between two window size: 24 rows x 80 cols and 34 rows x 132 cols.	
Q	Quit	Quits (ends) the debugging / animation session and the program (or programs).	

2. Tutorial - Sample Debugging Session

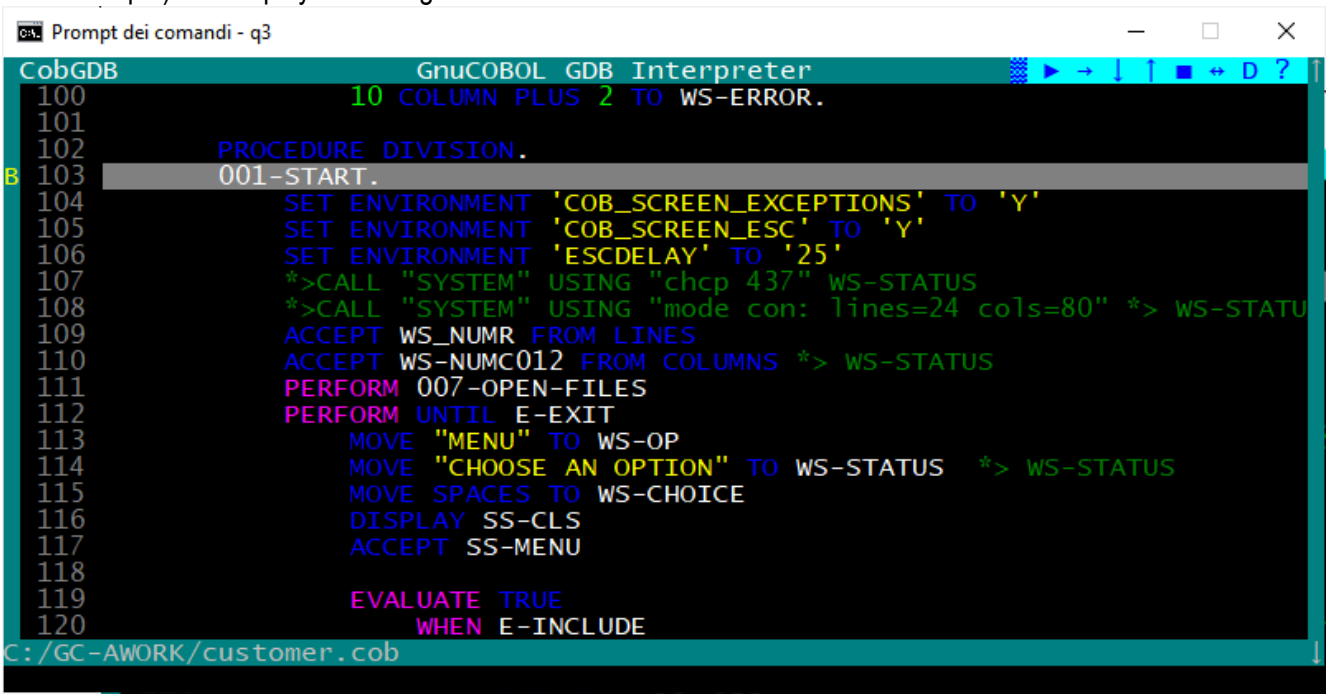
Following tutorial is on a Windows 10 platform using follwing version of GnuCOBOL:

```
cobc (GnuCOBOL) 3.2.0
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
Written by Keisuke Nishida, Roger While, Ron Norman, Simon Sobisch, Edward Hart
Built Jul 28 2023 16:07:38
Packaged Jul 28 2023 16:58:47 UTC
C version (MinGW) "13.1.0"
```

Downloaded from *https://www.arnoldtrembley.com/GnuCOBOL.htm*

====== Version 3.2 ======
GnuCOBOL 3.2 (28Jul2023) MSYS2 64-bit GC32M-BDB-x64.7z -- MSYS2 64-bit GnuCOBOL 3.2 Final release **with full debugging support**. (95.4 Megabytes).
GnuCOBOL 3.2 (28Jul2023) MSYS2 32-bit GC32M-BDB-x32.7z -- MSYS2 32-bit GnuCOBOL 3.2 Final release **with full debugging support**. (96.1 Megabytes).

After executing **cobgdb customer.cob -x -lpdcurses** the application automatically insert a Breakpoint at first executable program statement of PROCEDURE DIVISON (see the B symbol at left of line 103 in this sample) and displays following screen:



You can scroll the source code window with cursor keys UP and DOWN, PG UP and PG DOWN or with mouse wheel or with mouse left click on the right scroll bar. Use cursor RIGHT and cursor LEFT to scroll horizontally,

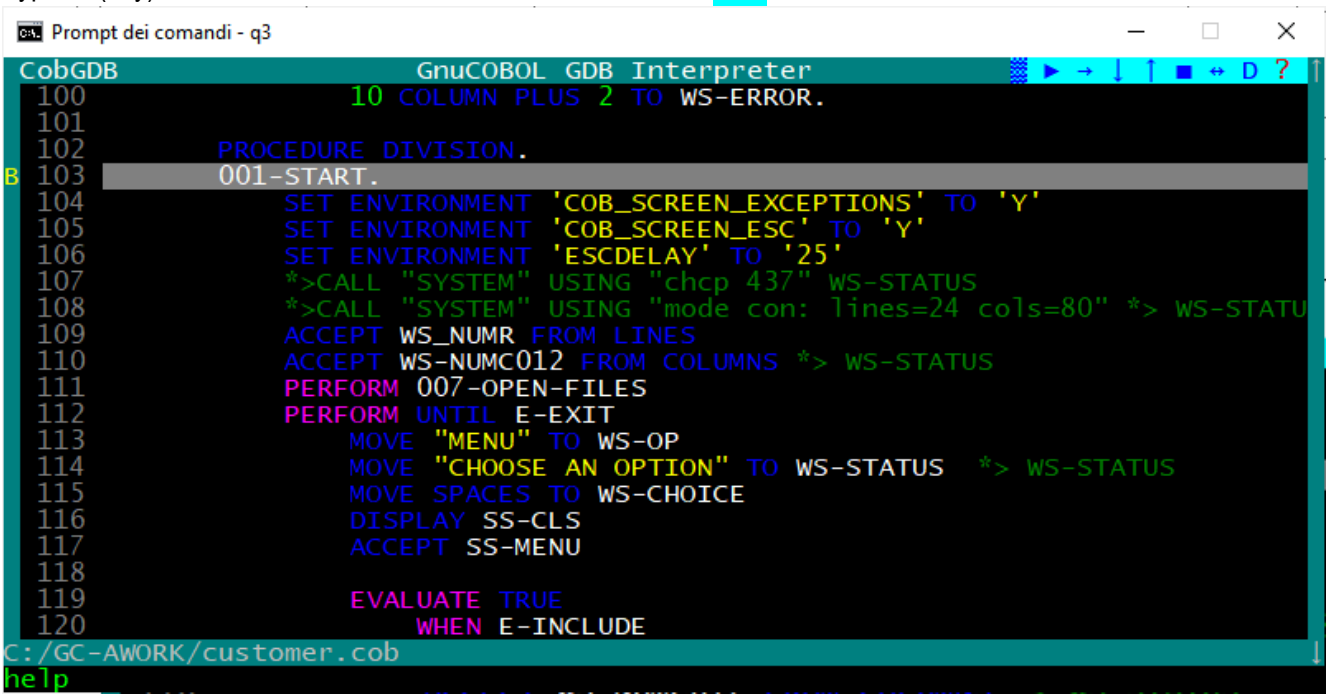
In the upper right window corner there is a "button bar" where you can find some buttons (symbols):

>	=	**Run** command
→	=	**Next** command
↓	=	**Step** command
↑	=	**Go** command
■	=	**Quit** command
<->	=	**Focus** command
D	=	**Display** command
?	=	**Help** command

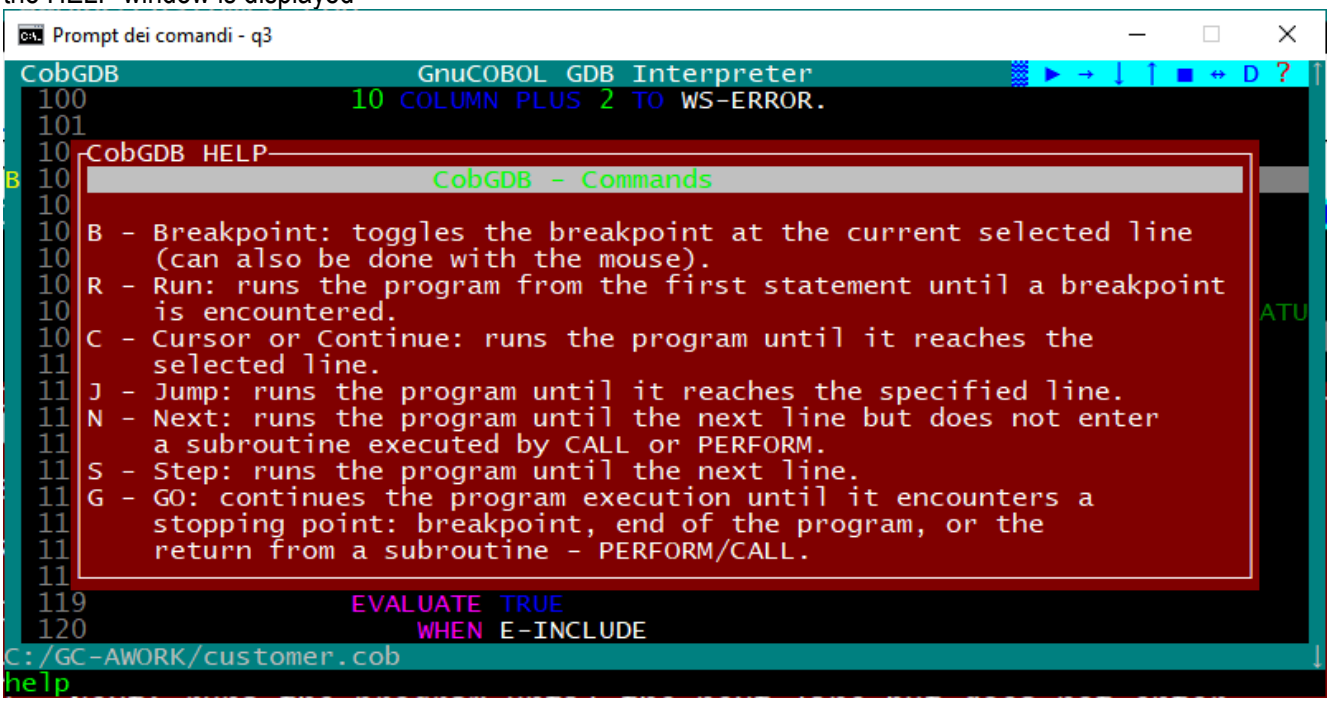
when you hover over one of these symbol, you get the corresponding command description (like a tooltip) displayed at the bottom left of the screen.

2.1. Help Command

Type **?** (key) HELP command or left click with mouse on the **?** button:



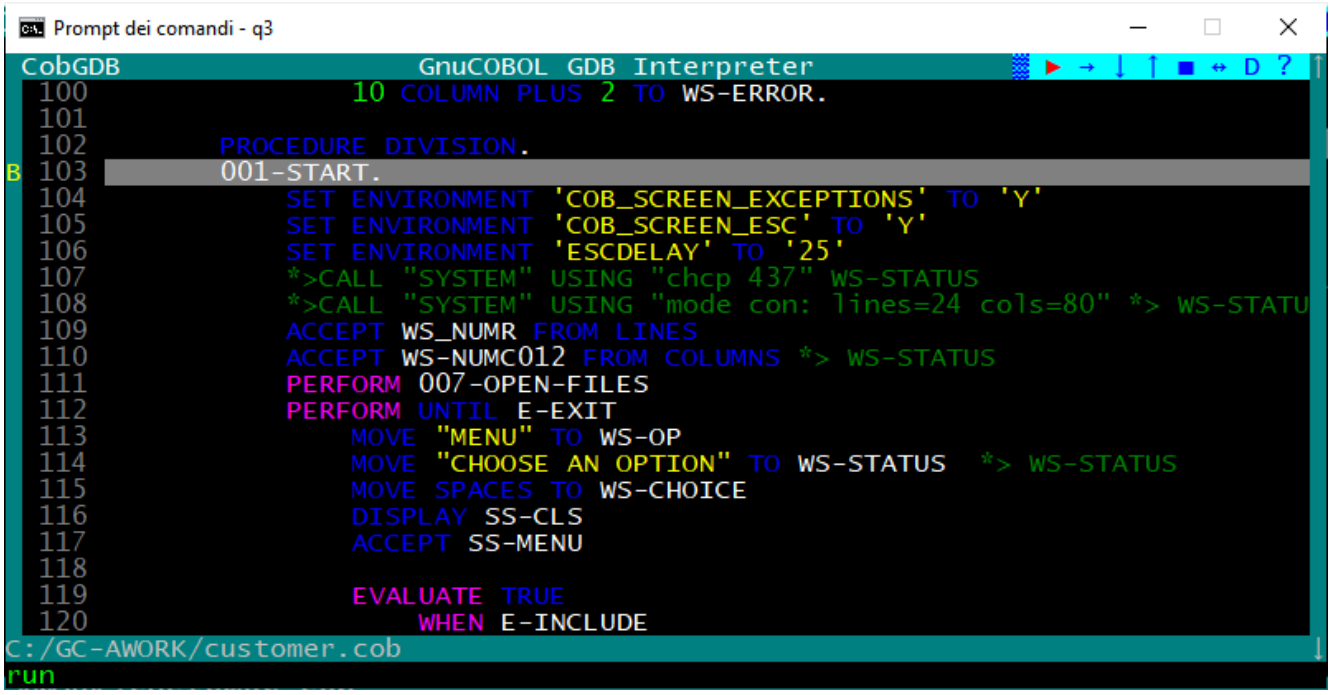
the HELP window is displayed



scroll the Help window with cursor keys UP and DOWN or mouse wheel.
Use ESC or Enter or left click to exit from this HELP window and return to debugging session.

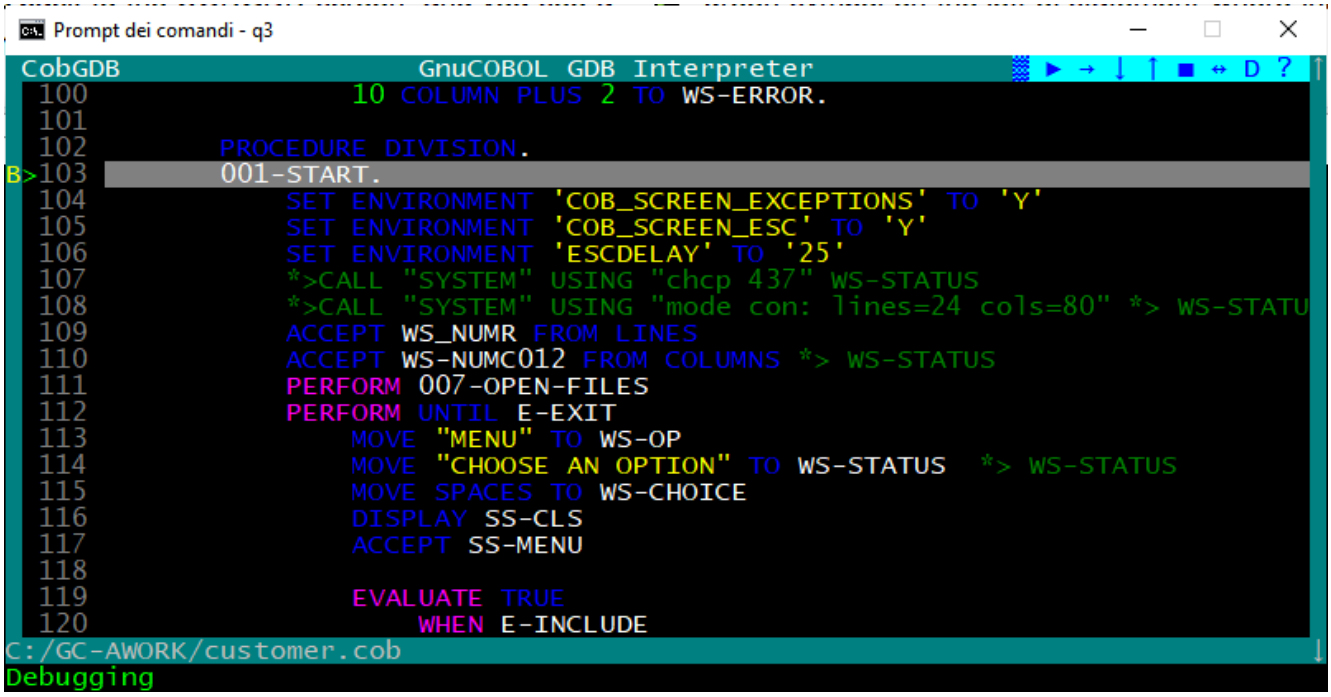
2.2. Run Command

To start executing the program and the debugging session from first program statement you always must use the "**R**" command (key) or left click with mouse on the **>** **Run** button



cobgdb opens the program terminal window (the application will run in this separate window.).

go back to the COBGDB screen, and you see a **>** green symbol on the left of statement where initial B Breakpoint is present (in our example is at line 103 B>103):



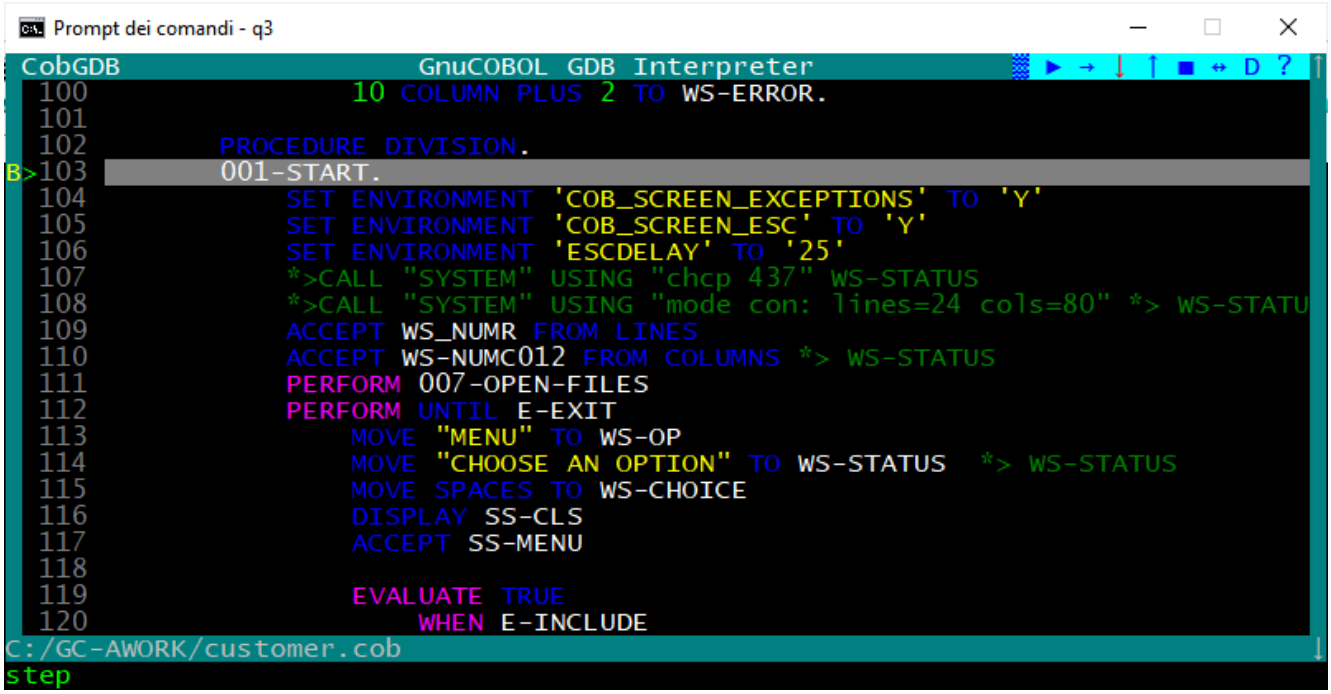
From that moment on, you can use all the commands (keys) or corresponding buttons to "animate" and debug the application, example: "**S**" (Step), "**N**" (Next), "**G**" (Go) and so on.

During the source code animation, the debugger can automatically shows some pop-up windows with variables content from the line in execution.

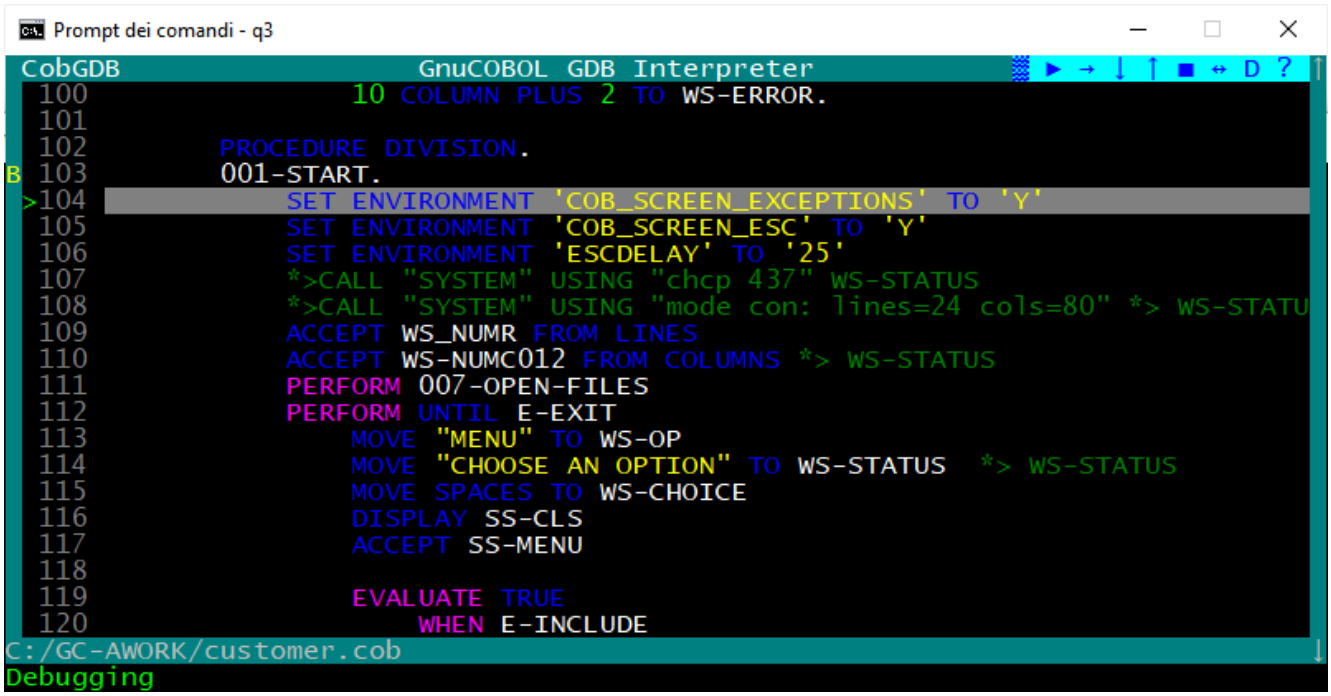
In order not to slow down the debugging / animation of the program unnecessarily, the automatic display of all variables of the statement in execution is disabled by default at program startup.
To activate the Automatic Display of all variables of the statement in execution use the "D" command.

2.3. Step Command

Proceed with **S** (Step) command or left click with mouse the ↓ button:.



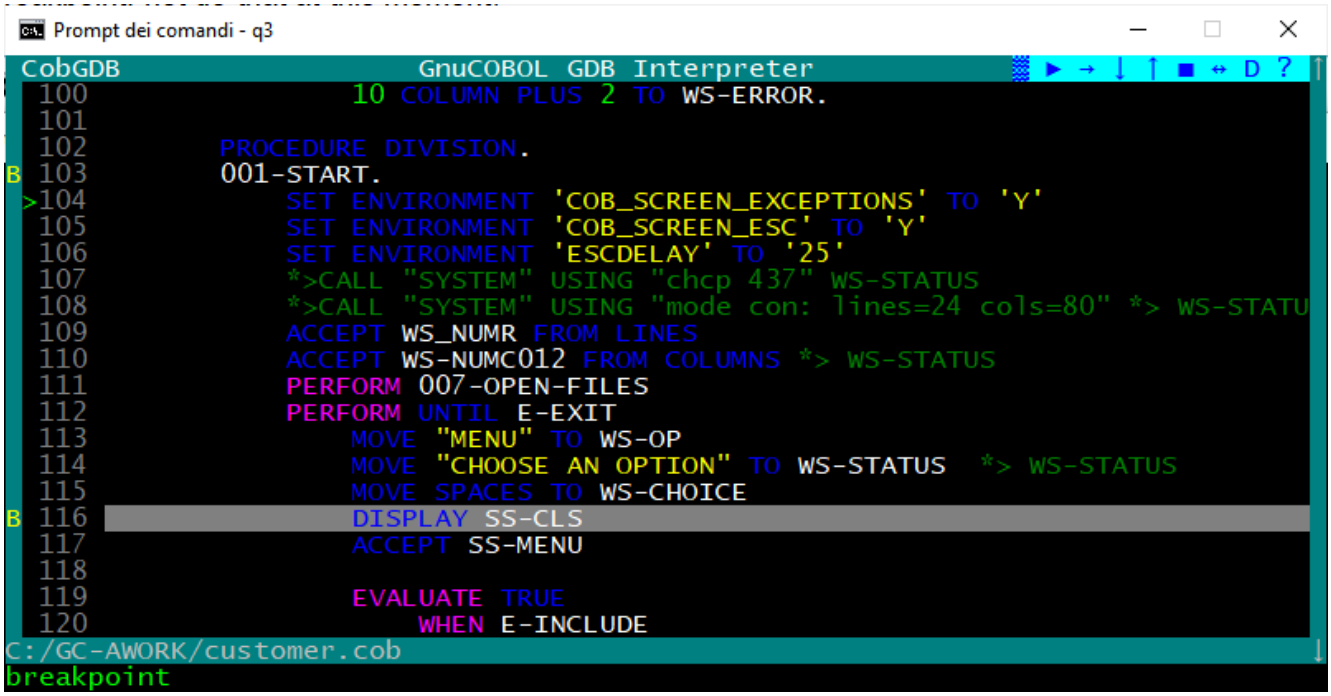
the **>104** green symbol now is on the following line 104:



Now you can proceed with S command or N command or as an example:

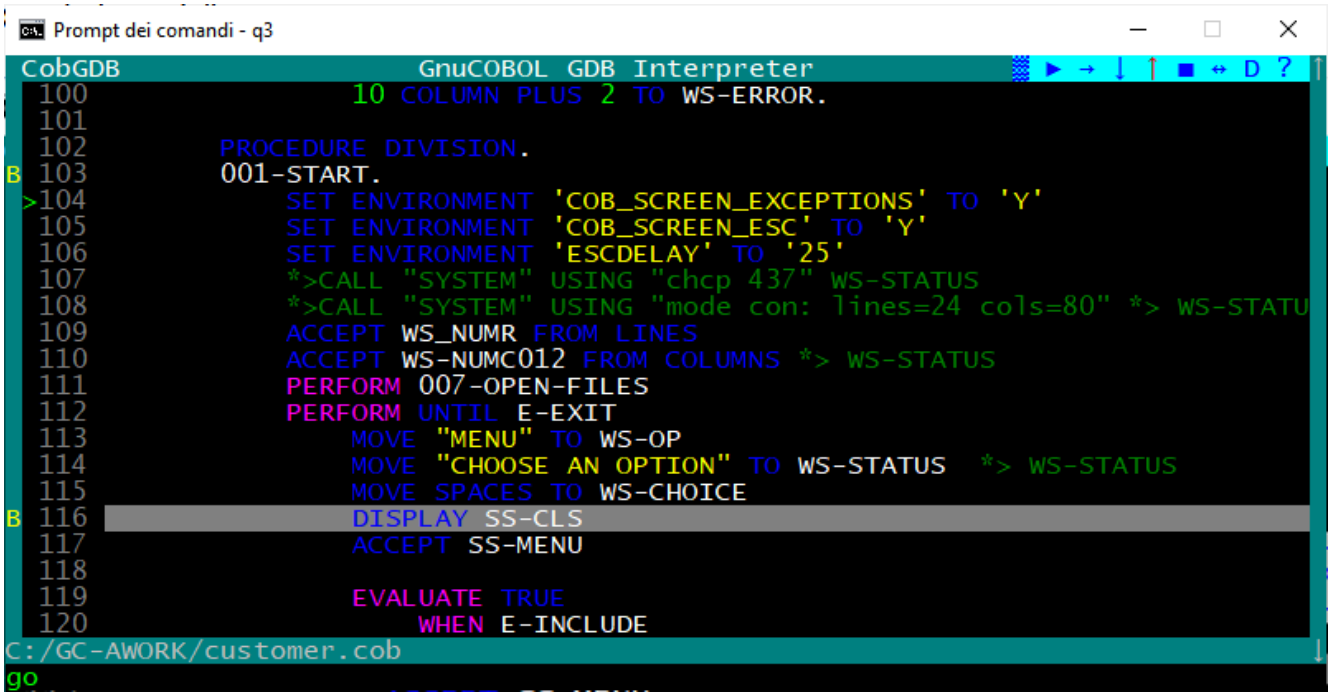
- Scroll with cursor down to select line 116 of Procedure Division and type "B" (to set a Breakpoint), (you also can simply click with mouse left button on the 116 row number)

- The application displays a "**B**" on the left of the line (type B again - or re-click - when you want to delete the Breakpoint, not do that at this moment)

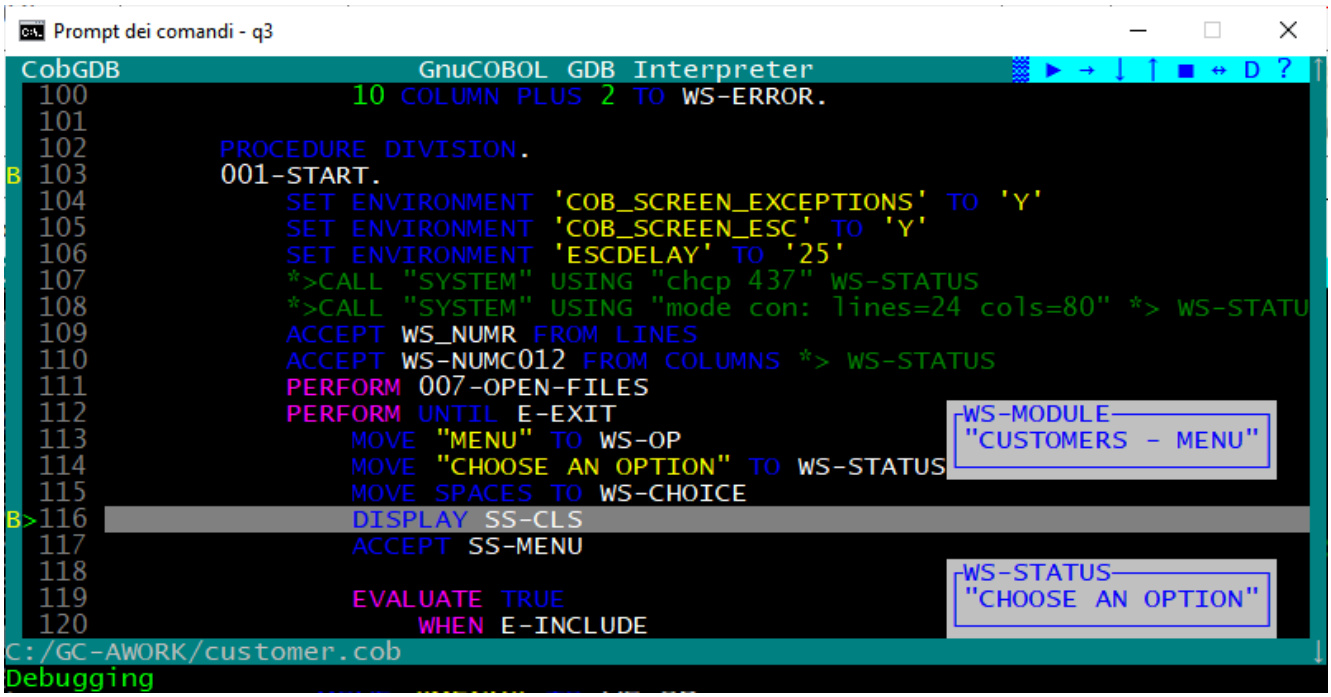


2.4. Go Command

Type **G** (Go) or left click with mouse the ↑ button to execute the program until a B Breakpoint is detected:

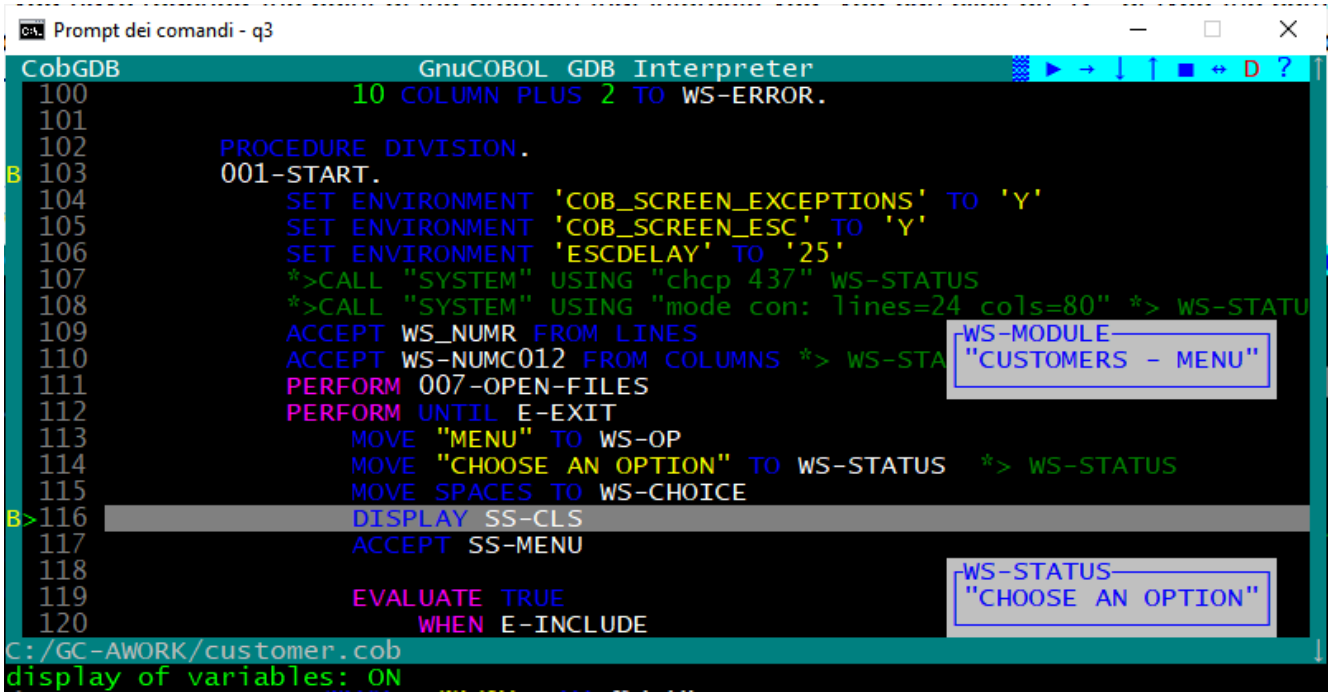


the system reach the second breakpoint at line 116 and displays a green **>** symbol to the left of the line to be executed, see following screen:



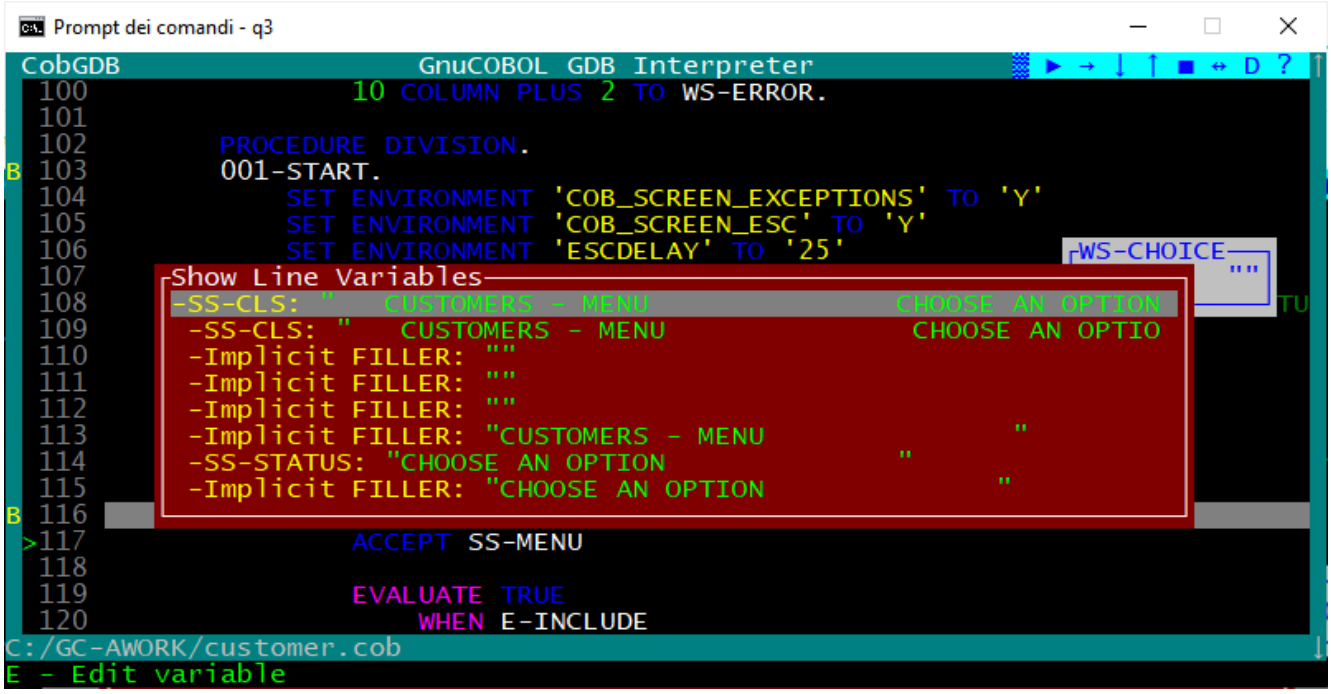
2.5. Display Variables

The debugger / animator can do an automatic display of all variables content from the statement executed and from the previous statement.
This can slow down the animation / execution and it may not be useful to have it activated everywhere.
Type **D** (Display Variables) command to enable or disable this display at your convenience.
Alternatively, click on the "D" symbol in the top right bar of the screen.
By moving the mouse over the "D" symbol in the bar you will see a message with the current status of this option displayed at the bottom left of screen: Display of Variables: ON or Display of Variables: OFF.
At the beginning of the program this option is always set to OFF to speed up the animation.
When you have reached the point of the program that interests you, you can click on "D", or type the command "D", to activate the automatic display of all the variables of the statement in execution (black pop-up windows) and of the previous statement (blue pop-up windows).

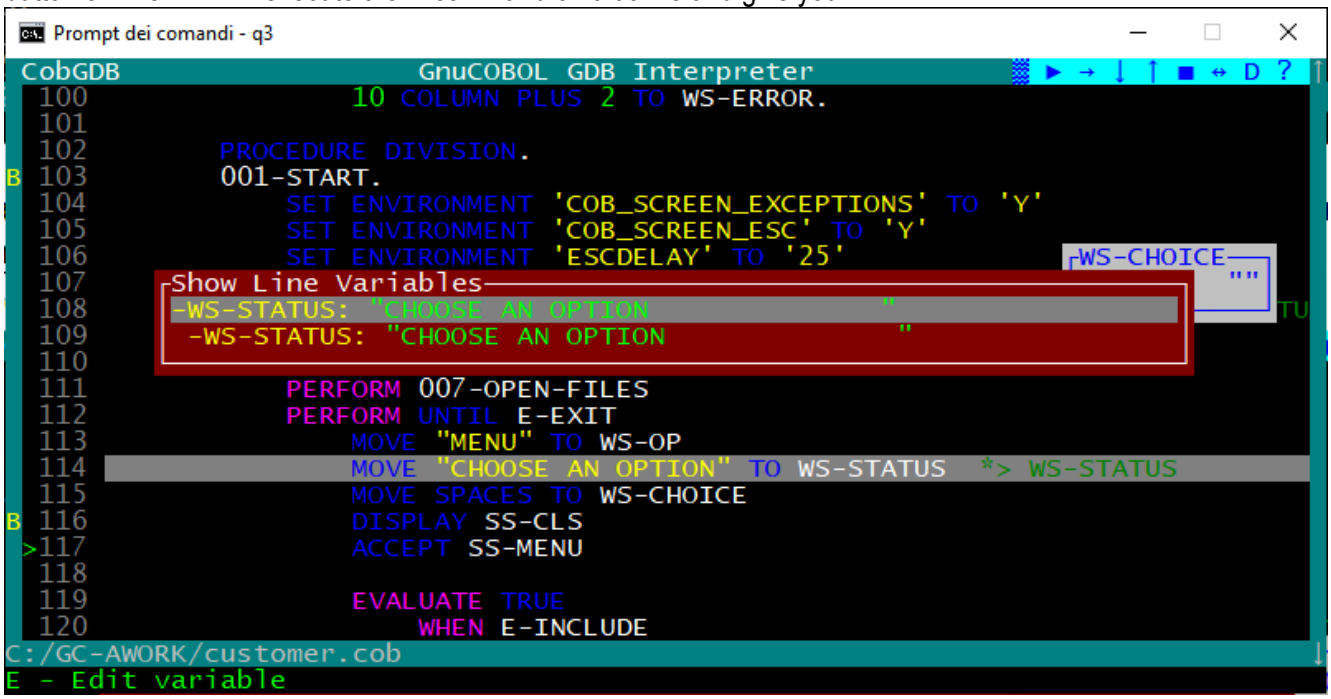


2.6. Show Command

typing the '**H**' command (key) allows you to view the variables on the cursor selected (highlighted) line.



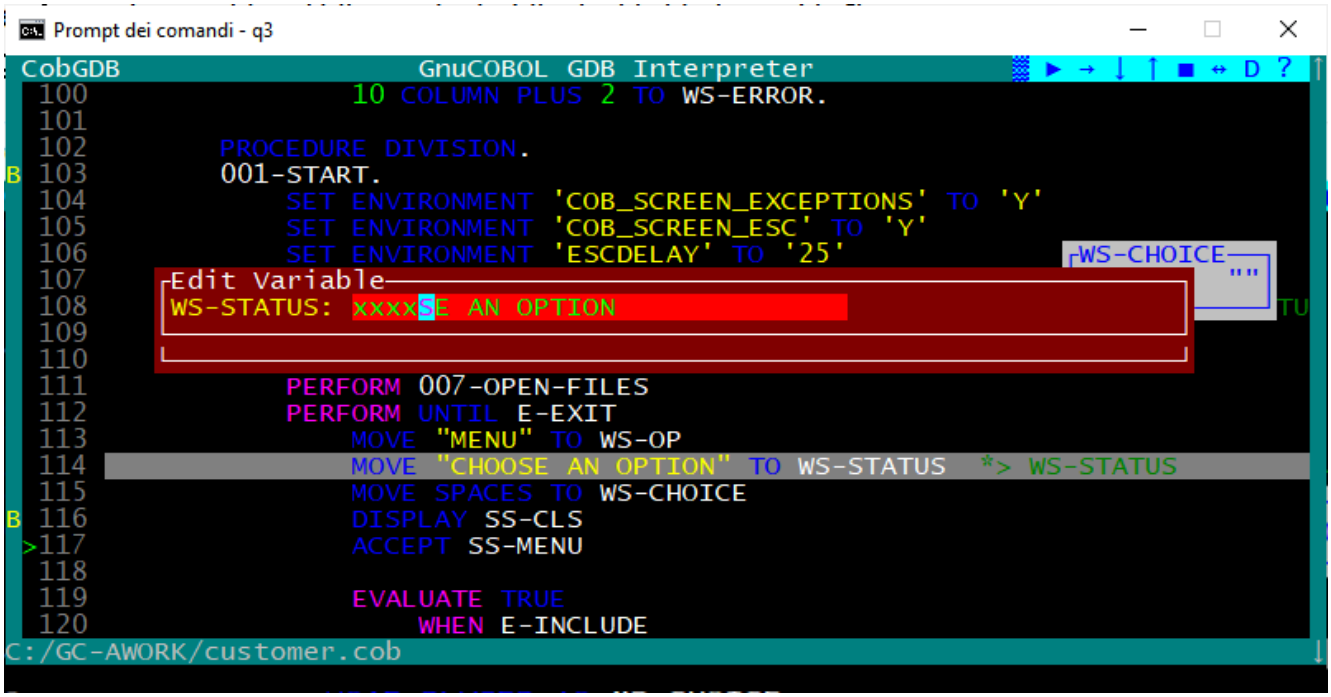
Display the content of variables also clicking right mouse button on a source line, example click right mouse button on line 114 will execute the H command on that line and give you:



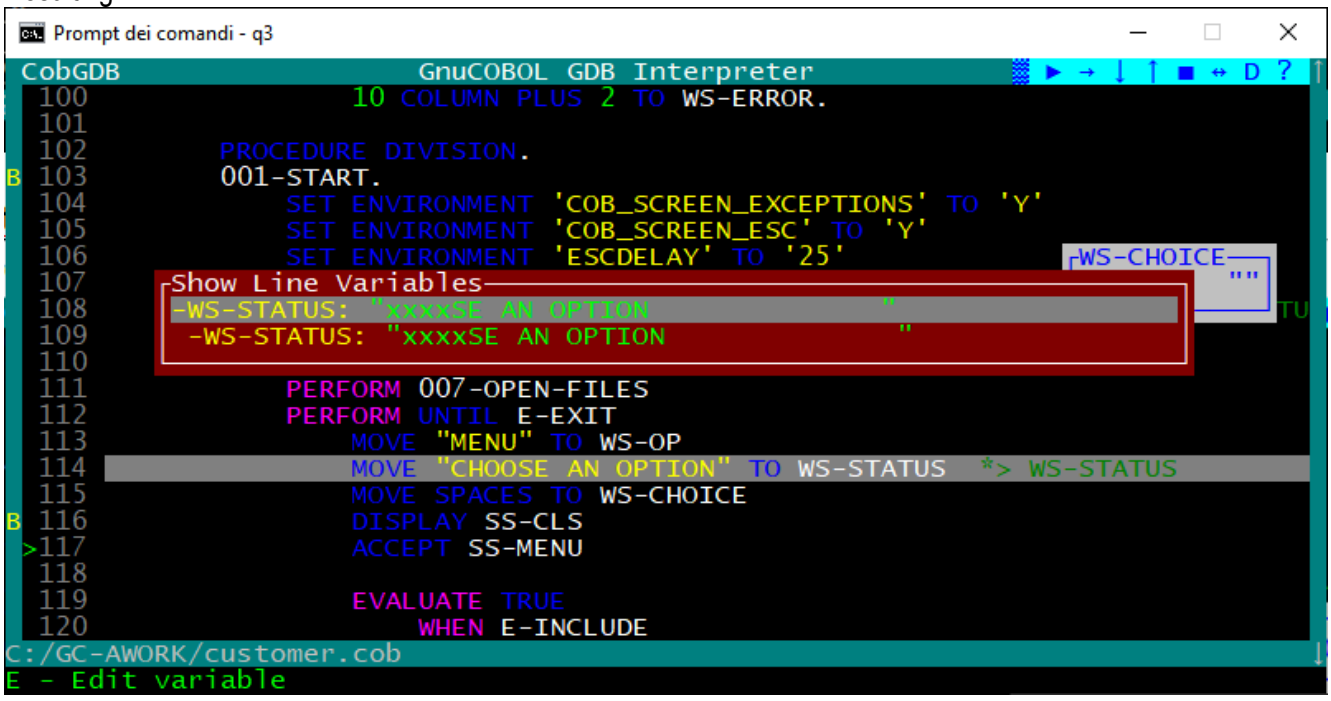
Now at lower left line on screen is a message: E = Edit (change) the variable value.

2.6.1. Edit subCommand

Scroll with cursor key UP and DOWN (or with mouse wheel). to select one of the variable.
The key **E** can be used to edit the content of the highlighted variable.
Change the value and type Enter to confirm canges or use ESC to exit without changes :

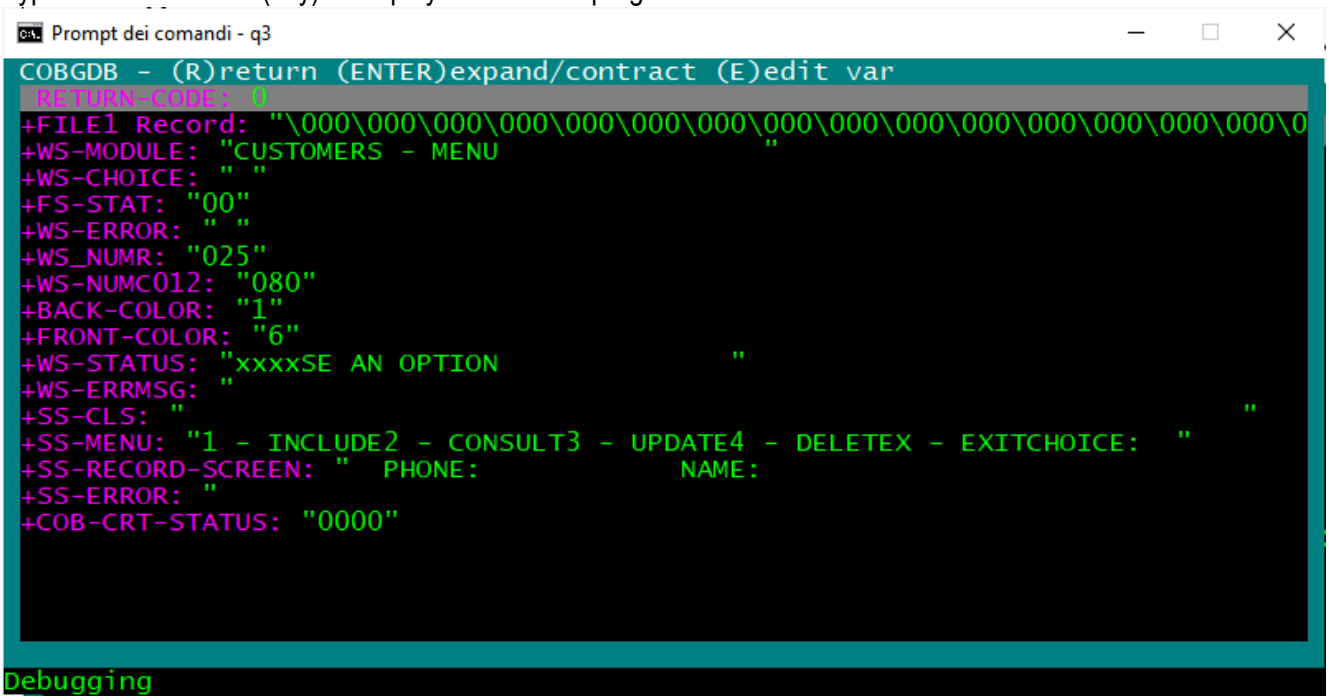


Resulting:



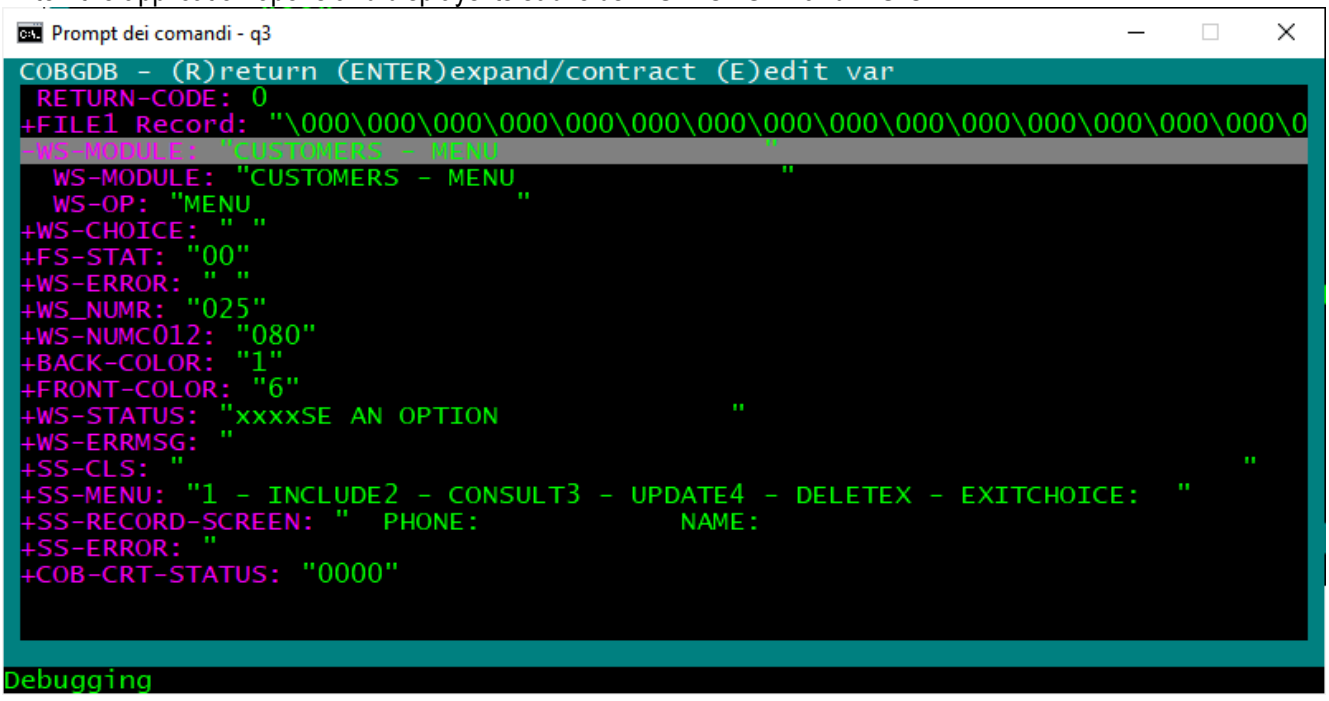
2.7. Variable Command

Type the **V** command (key) to display the list of all program variables:

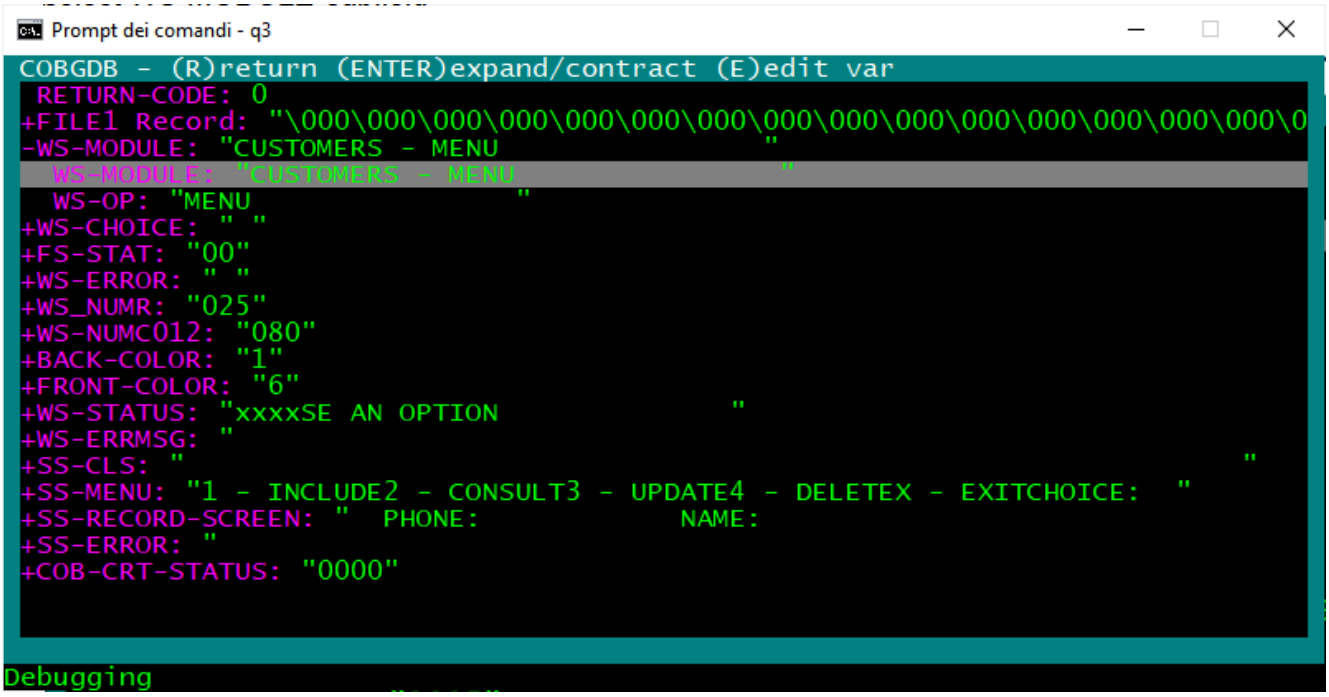


2.7.1. Enter subCommand

Scroll with cursor key UP and DOWN (or with mouse wheel) in this list to the variable WS-MODULE and type Enter: the application opens and displays its subfields WS-MODULE and WS-OP

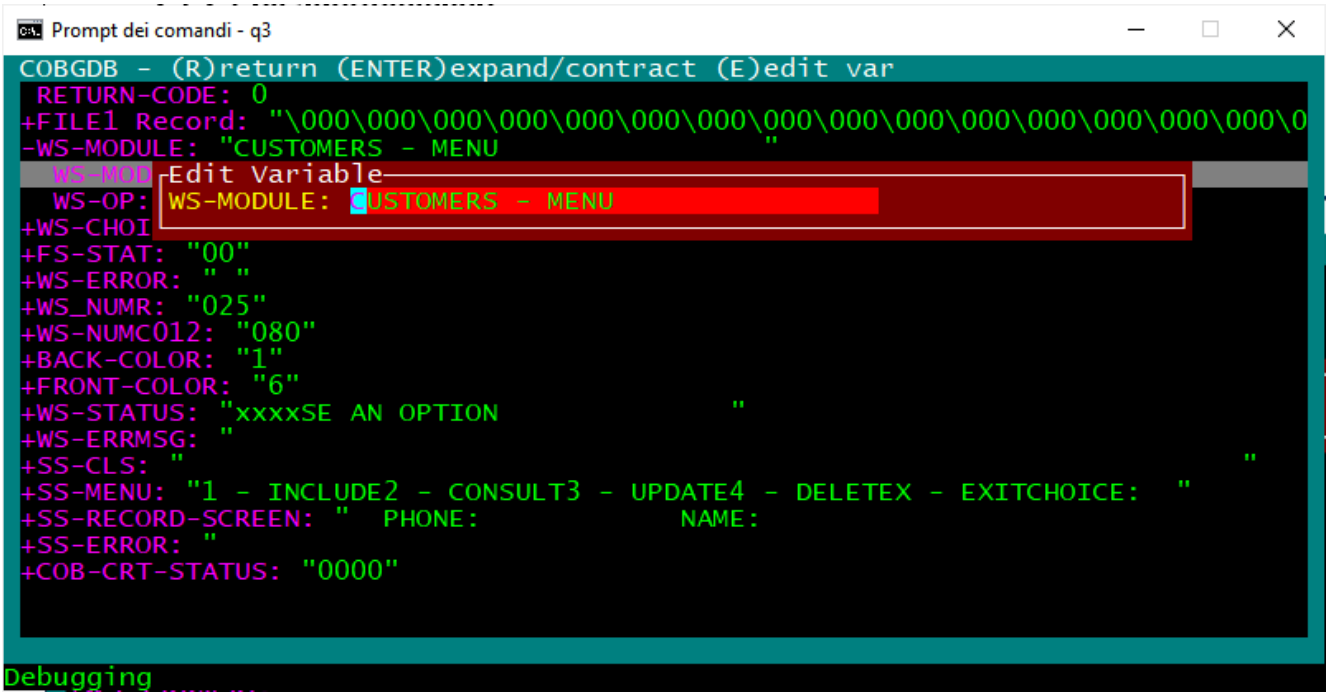


select WS-MODULE subfield

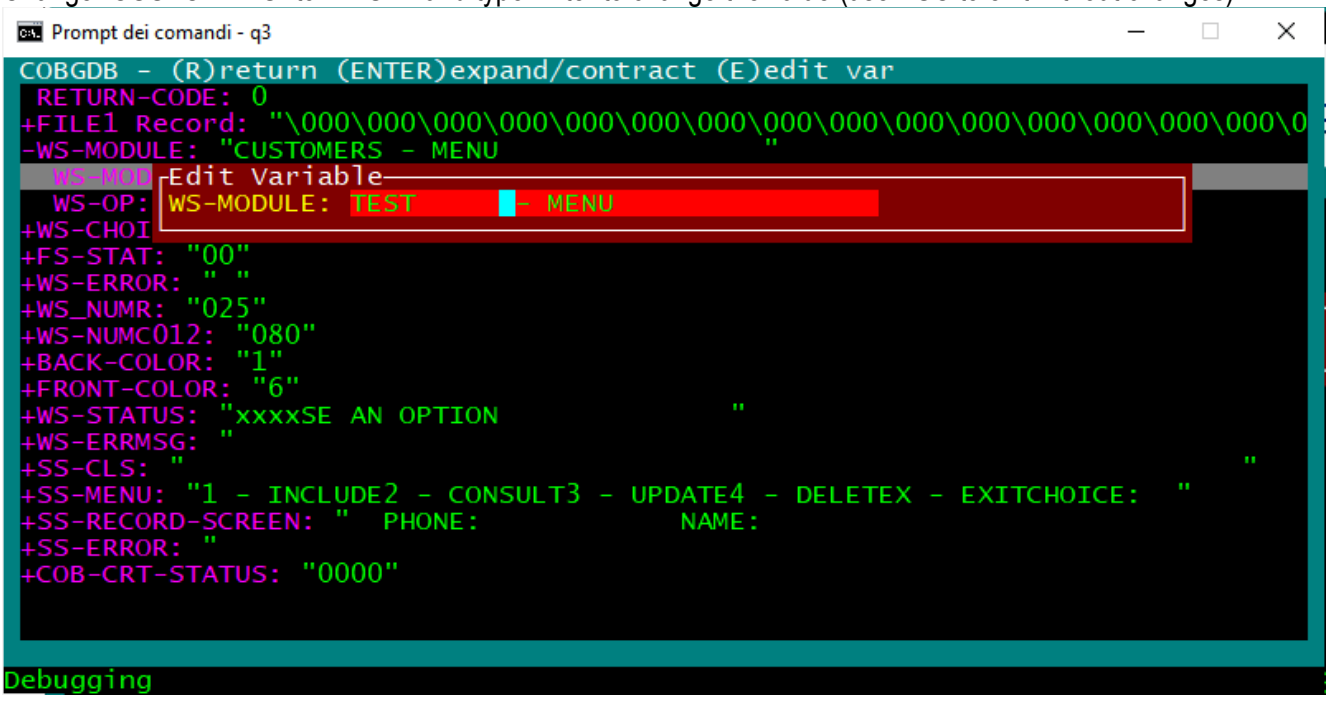


2.7.2. Edit subCommand

Now you can select WS-MODULE subfield whit cursor DOWN or mouse wheel and type "**E**" (Edit)
COBGDB shows a Edit Variable window:



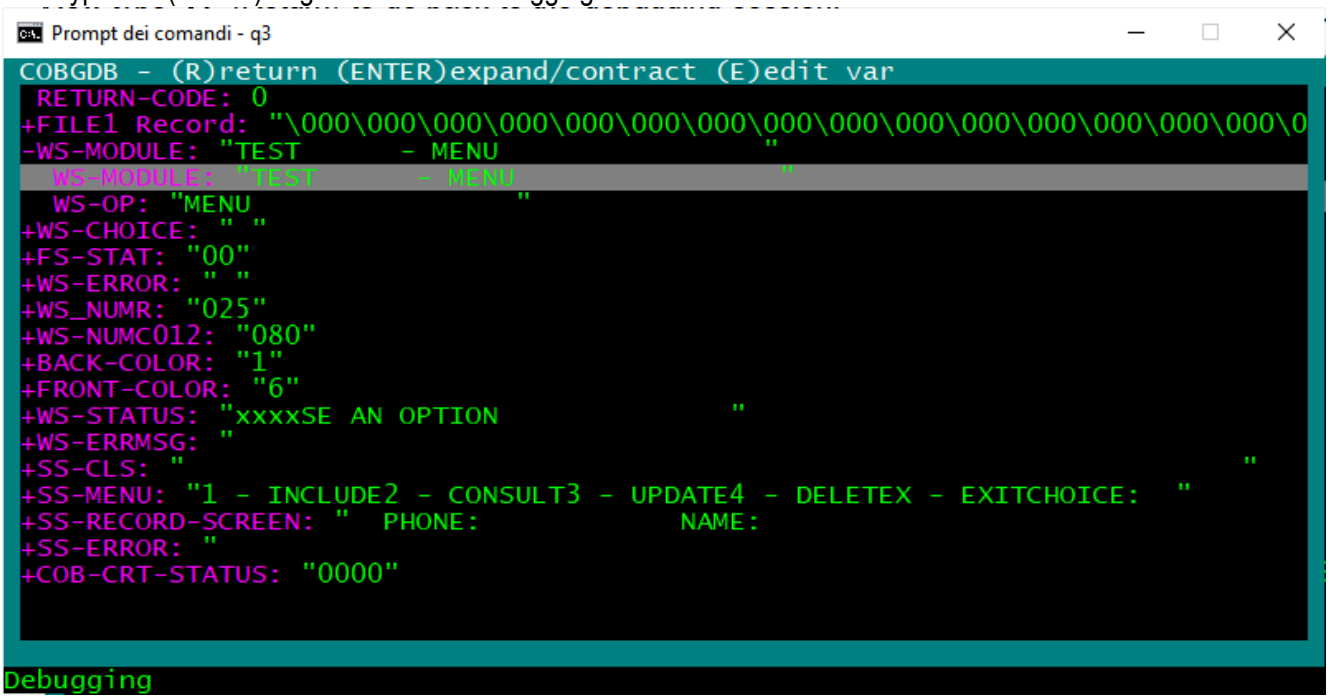
Change "CUSTOMERS" to "TEST" and type Enter to change the value (use ESC to exit without changes) :



2.7.3. Return subCommand

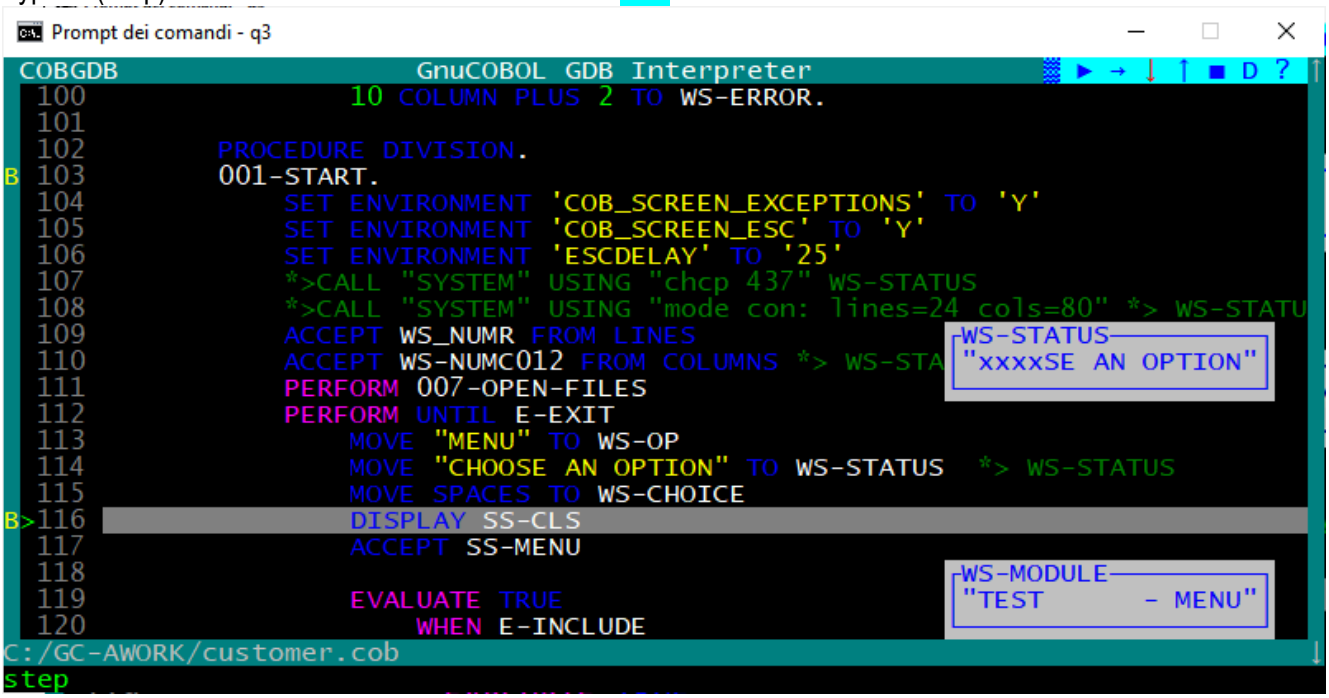
WS-Module has new value.

Now type "**R**" (Return) to go back to the debugging session:



2.8. Step Command

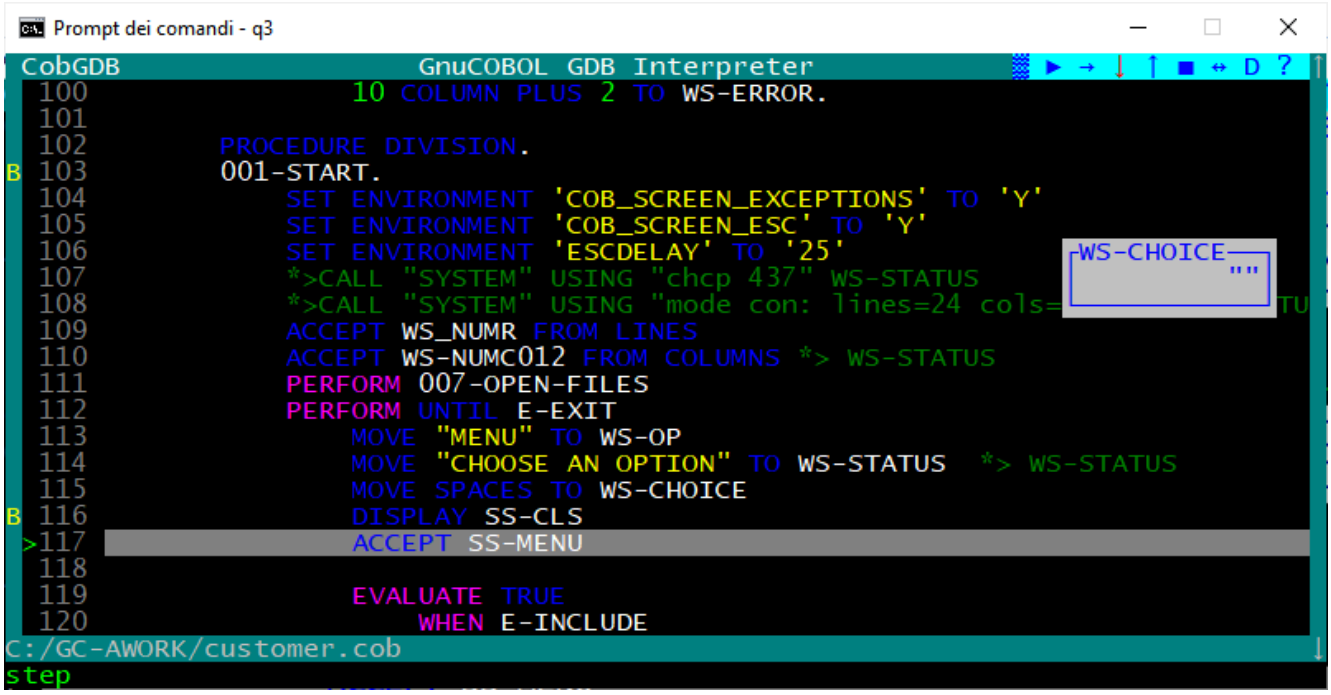
now you are back in the main debugging window:
Type **S** (Step) command or left click with mouse the ↓ button to execute the DISPLAY statement at line 116



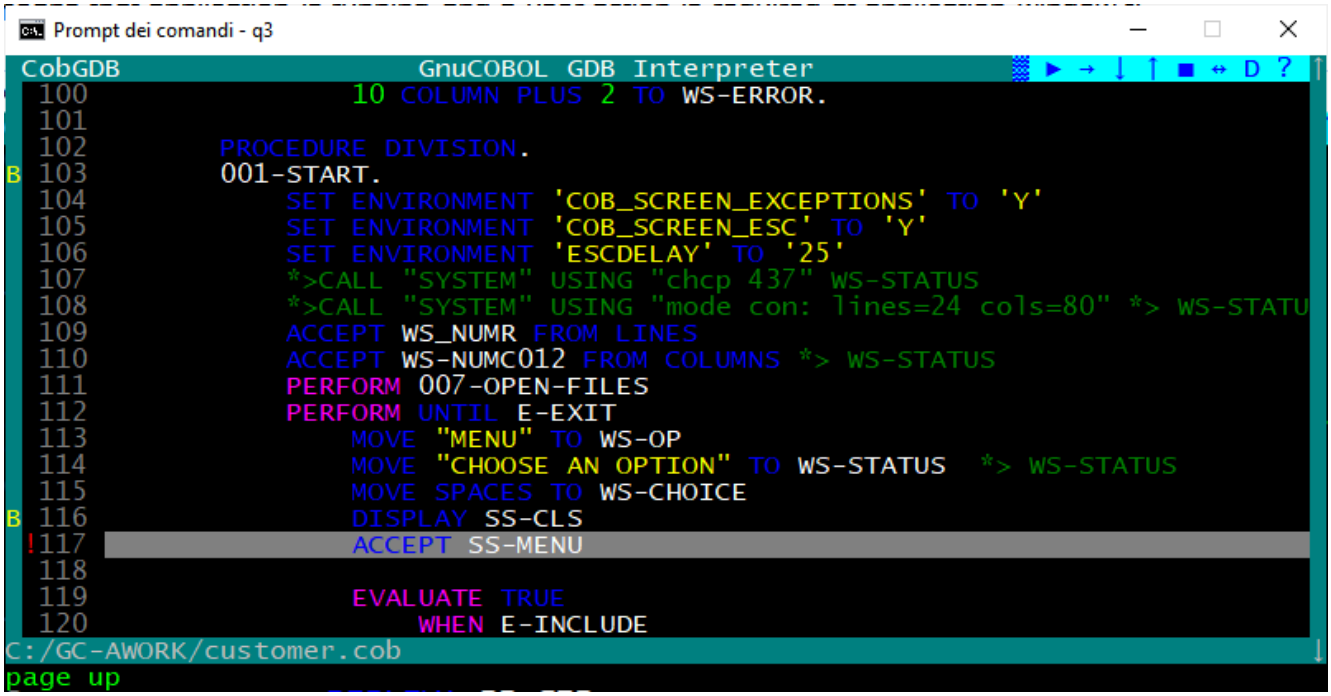
in the other application window you can see the result of DISPLAY statement

go back to debugging window and now the ACCEPT statement will be executed with S command

Type **S** (Step) command or leftclick with mouse the ↓ button again to execute ACCEPT statement at line 117:

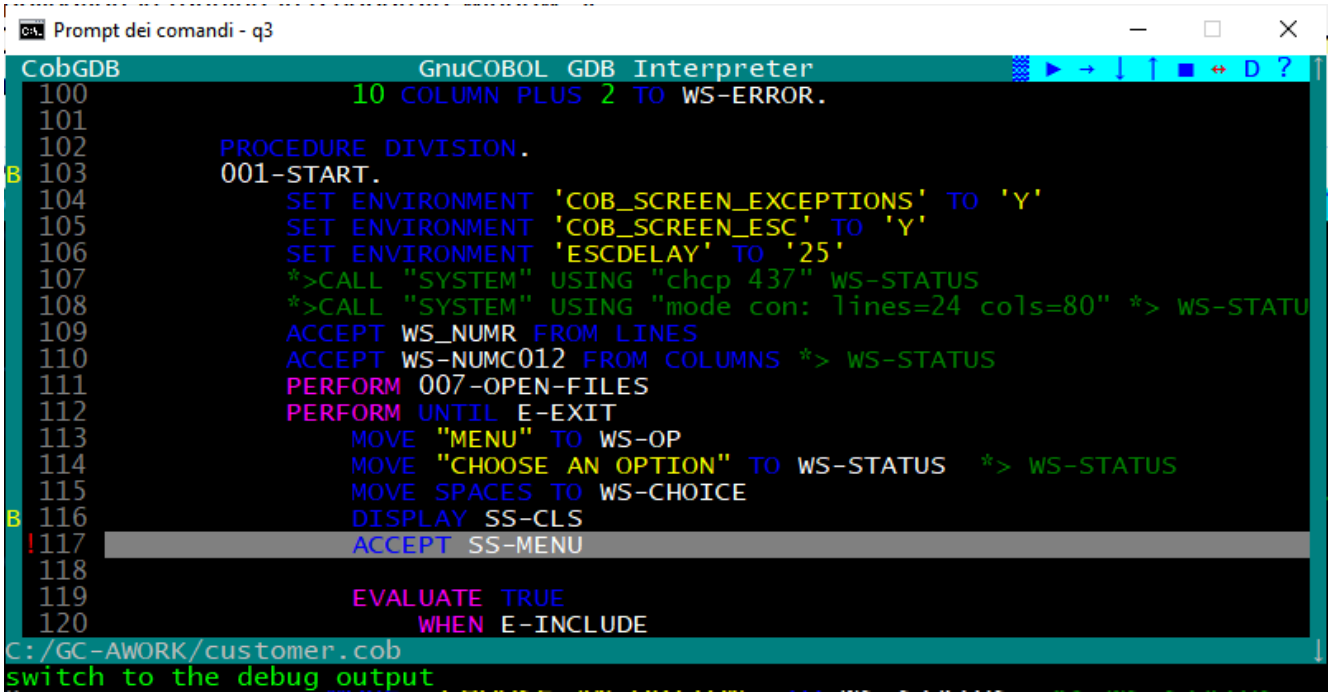


A red ! exclamation mark appears on the line !117 .
This means that the application is running and a user action is required at application window.

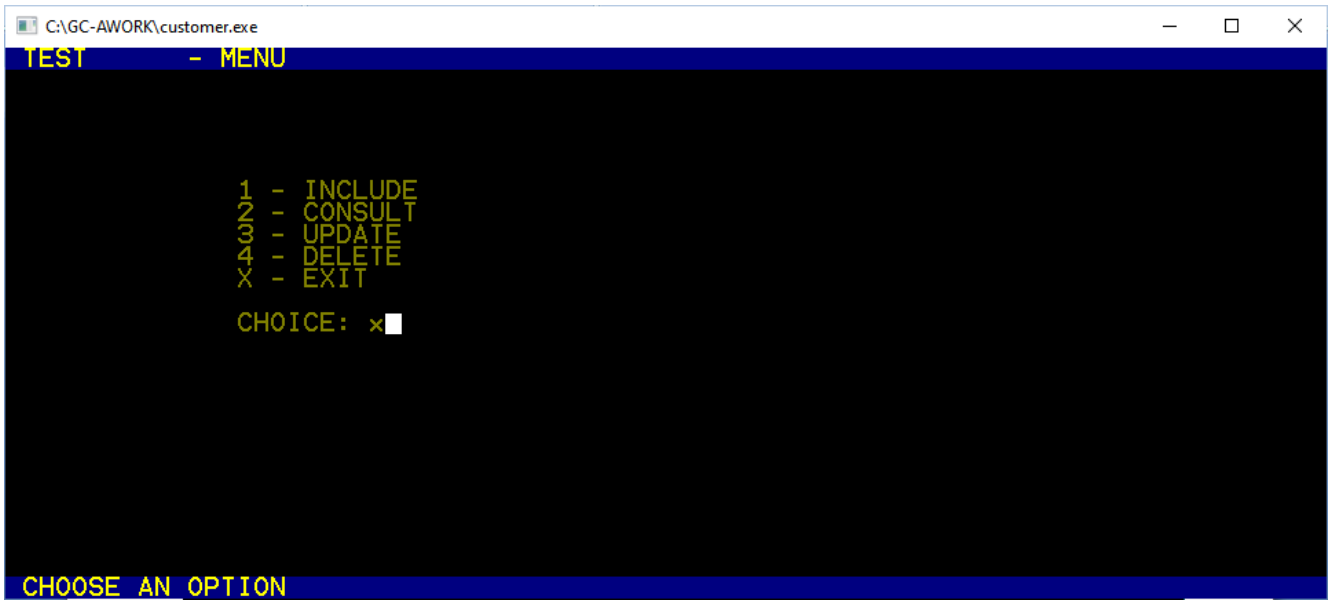


2.9. Focus Command

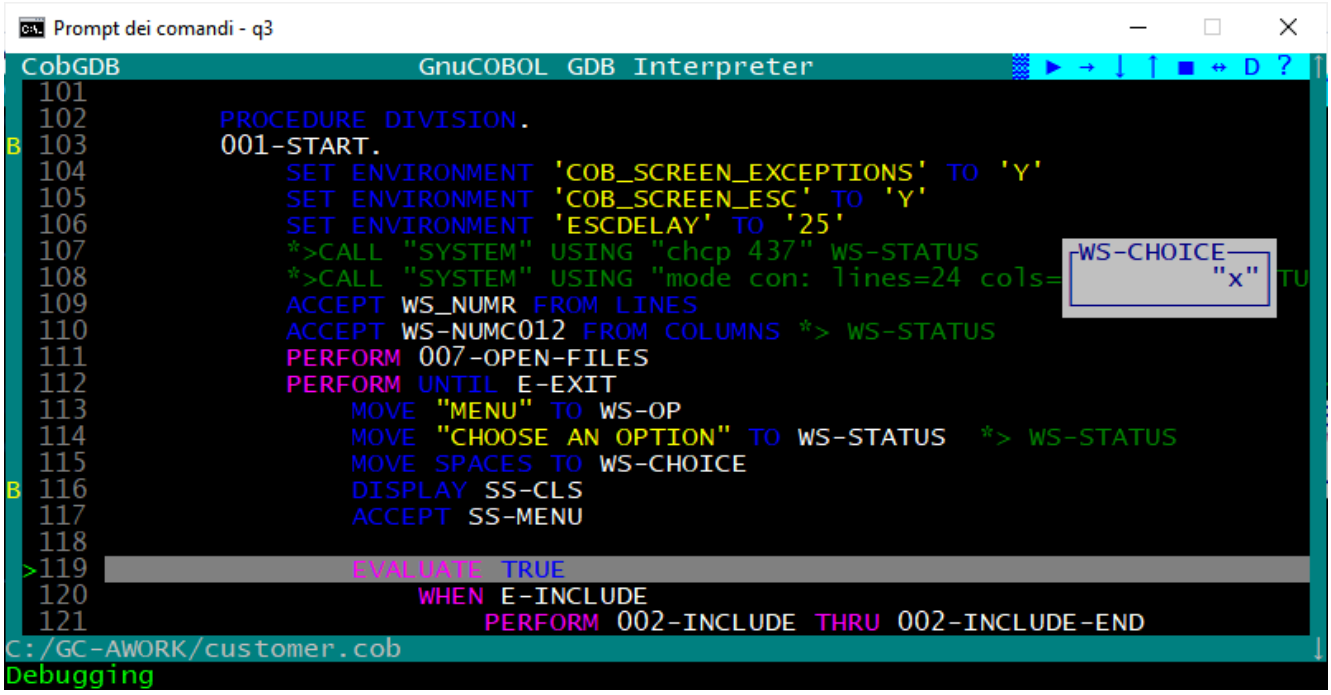
The application is running in a separate window.
The 'O = Focus' command switches the focus from debugging / animation screen to application window.
Instead of using the "O" command you can left-click on the <-> icon.



The focus now is on the application screen where a user action is needed.
In our sample, we type the "X" choice and Enter:



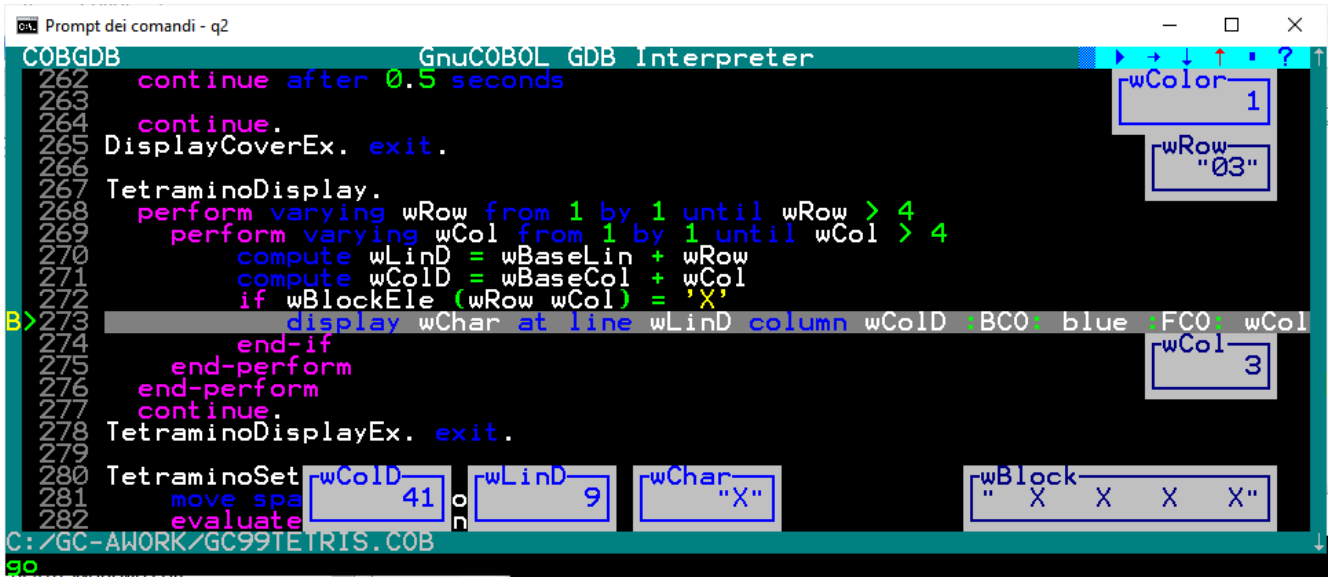
Now the system automatic switch back to the debugger / animation screen to continue debugging.
You can see that the ACCEPT statement has been executed, now you are on line >119 :



2.10. Pop-up Variable windows

During a debugging session COBGDB shows variable content.
Blue frame and values : variables of executing cobol statement
Black frame and values : variables of last executed cobol statement.

Sample:



2.11. File Command

To show this command we use following sample:

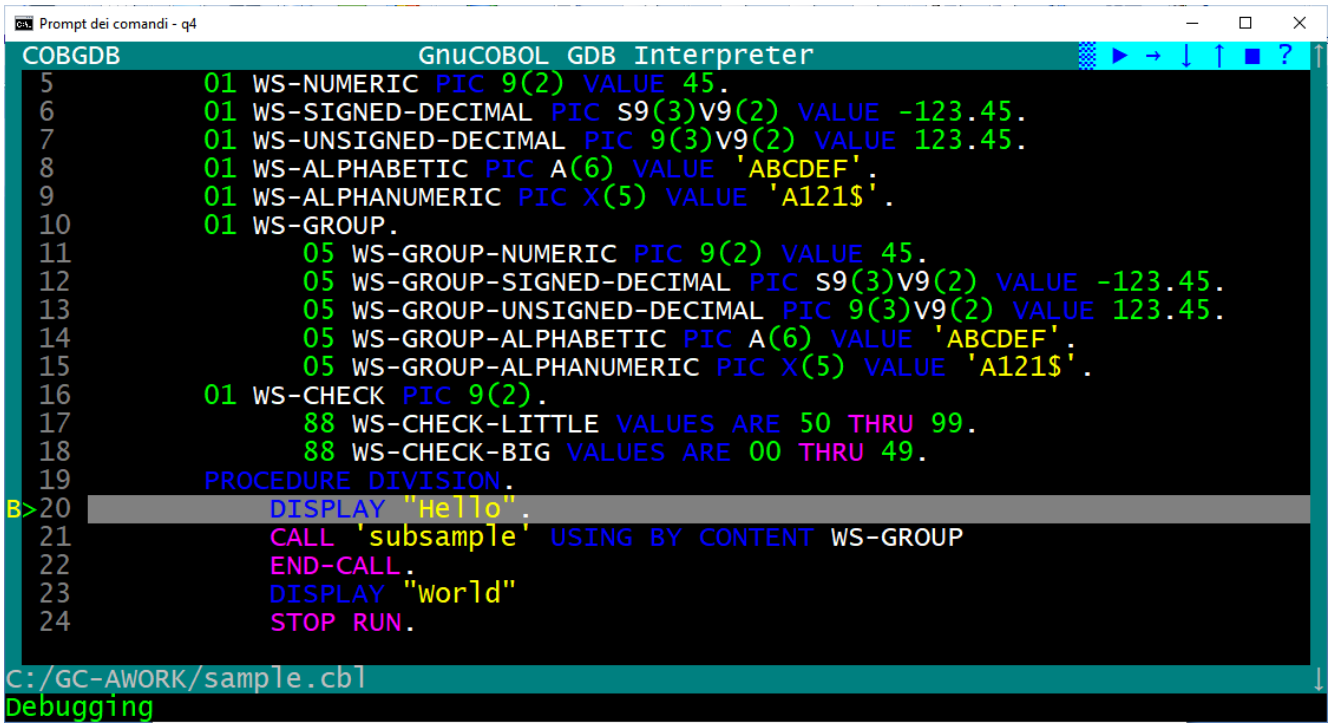
```
cobgdb sample.cbl subsample.cbl subsubsample.cbl -x -lpdcurses
```

where sample.cbl is the main program; it calls
--> subsample.cbl; it calls
--> subsubsample.cbl

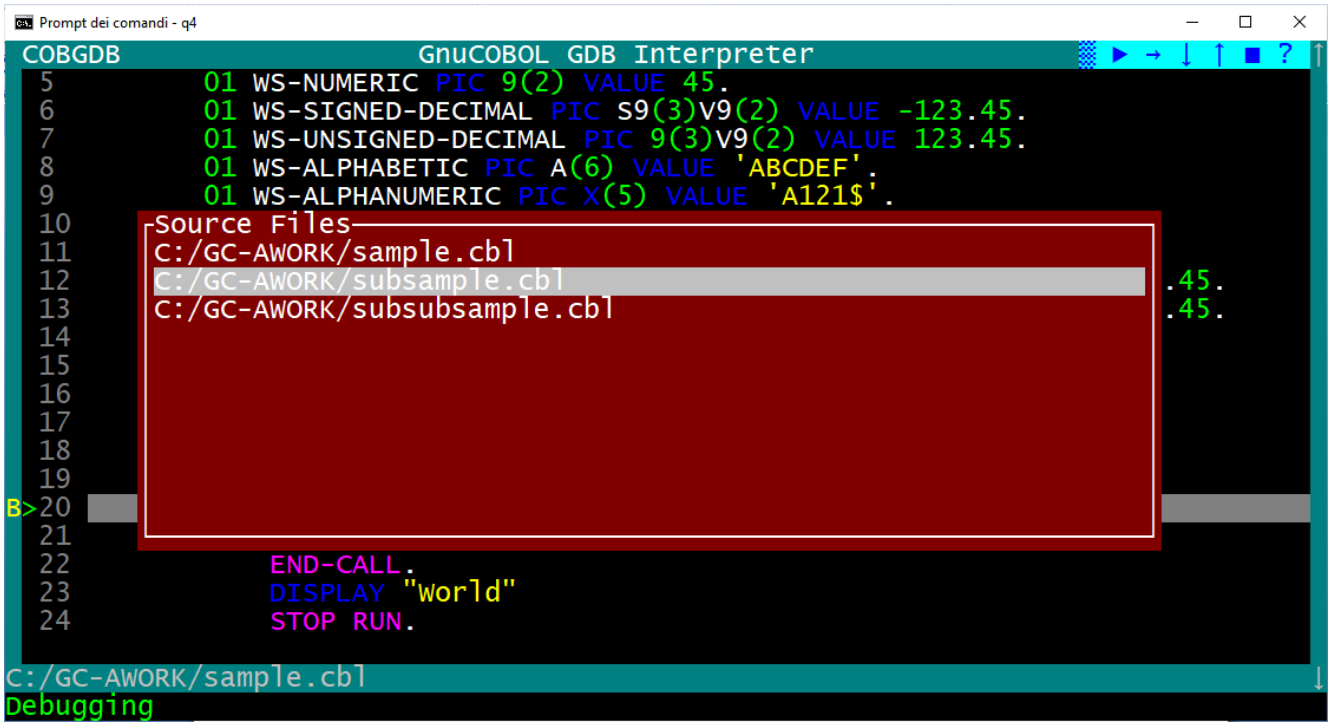
Source code is at https://github.com/marcsosduma/cobgdb/tree/main/resources.
This will create a single sample.exe executable.

This example shows that when you need to debug only subsample.cbl or only subsubsample.cbl you need to execute COBGDB with all three programs.

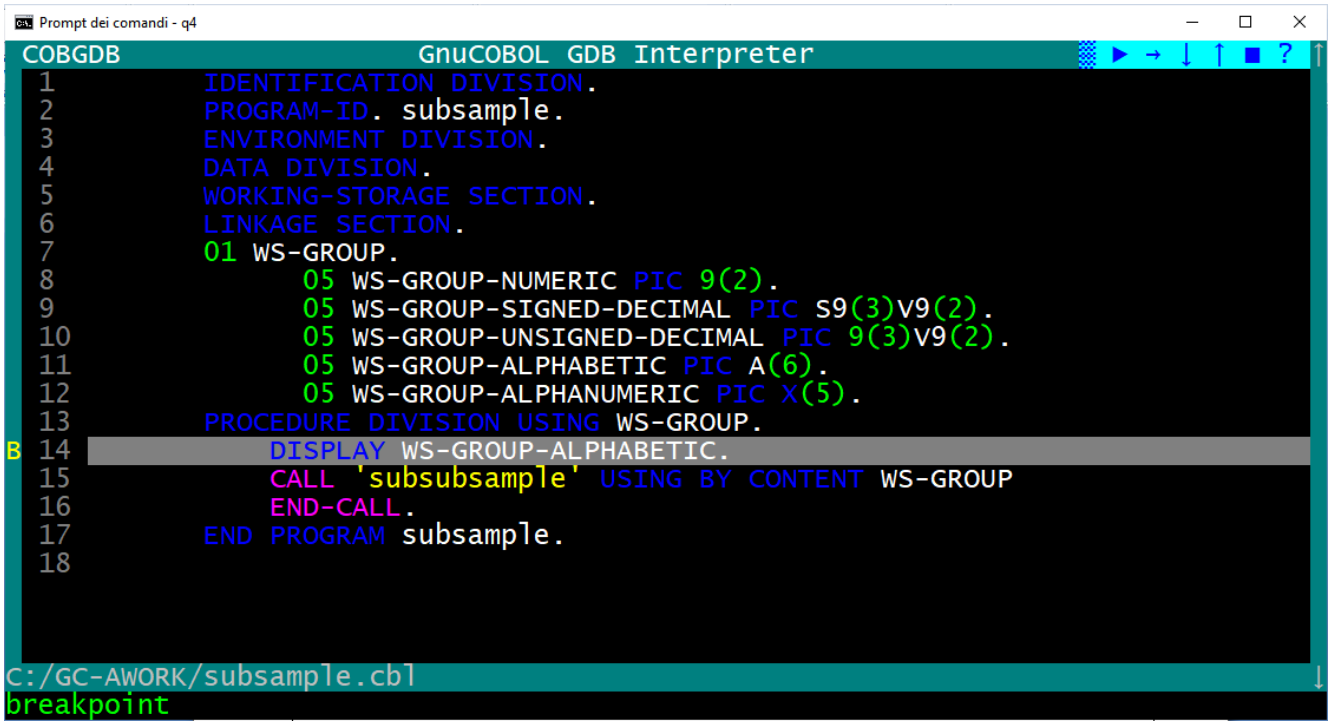
COBGDB sets the B breakpoint at first executable statement of first program "sample.cbl".
here use the R Run command to start the debugging session.



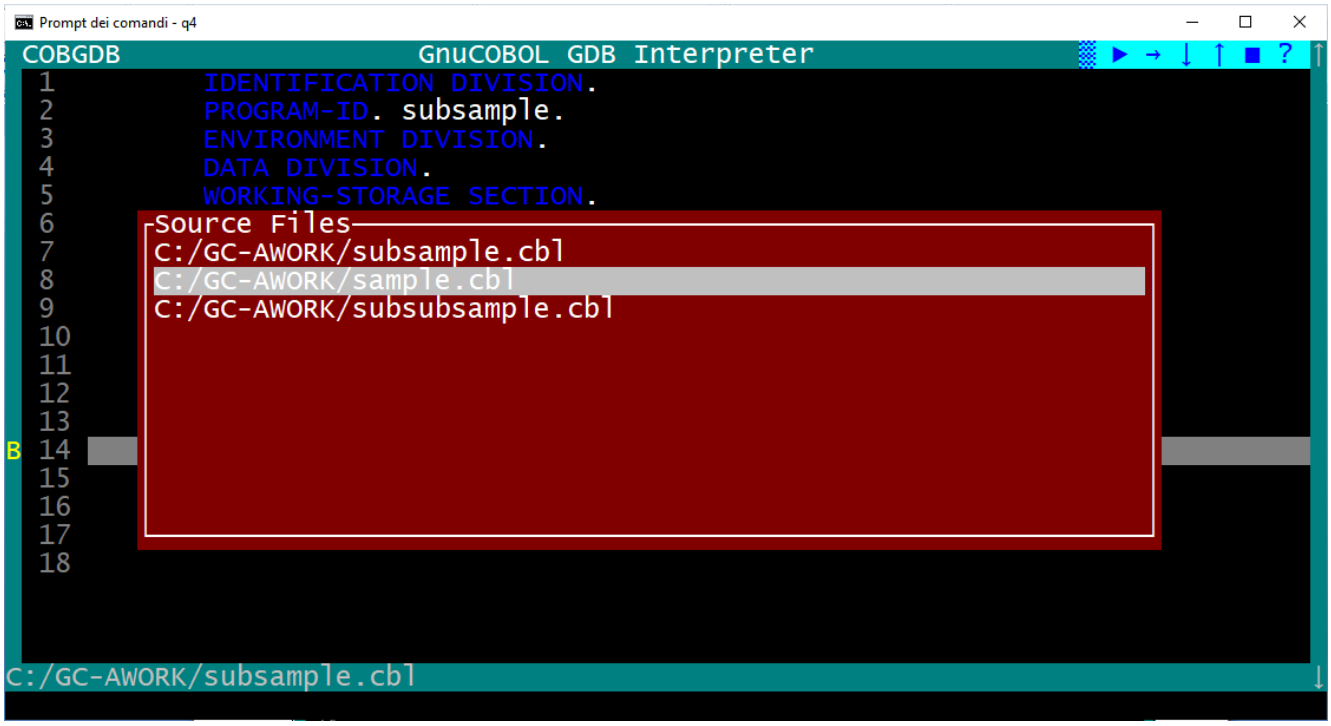
Now you can type the **F File** command and you will have the "Source Files" window.
In this sample we select the second program in the list (subsample.cbl) and type Enter.



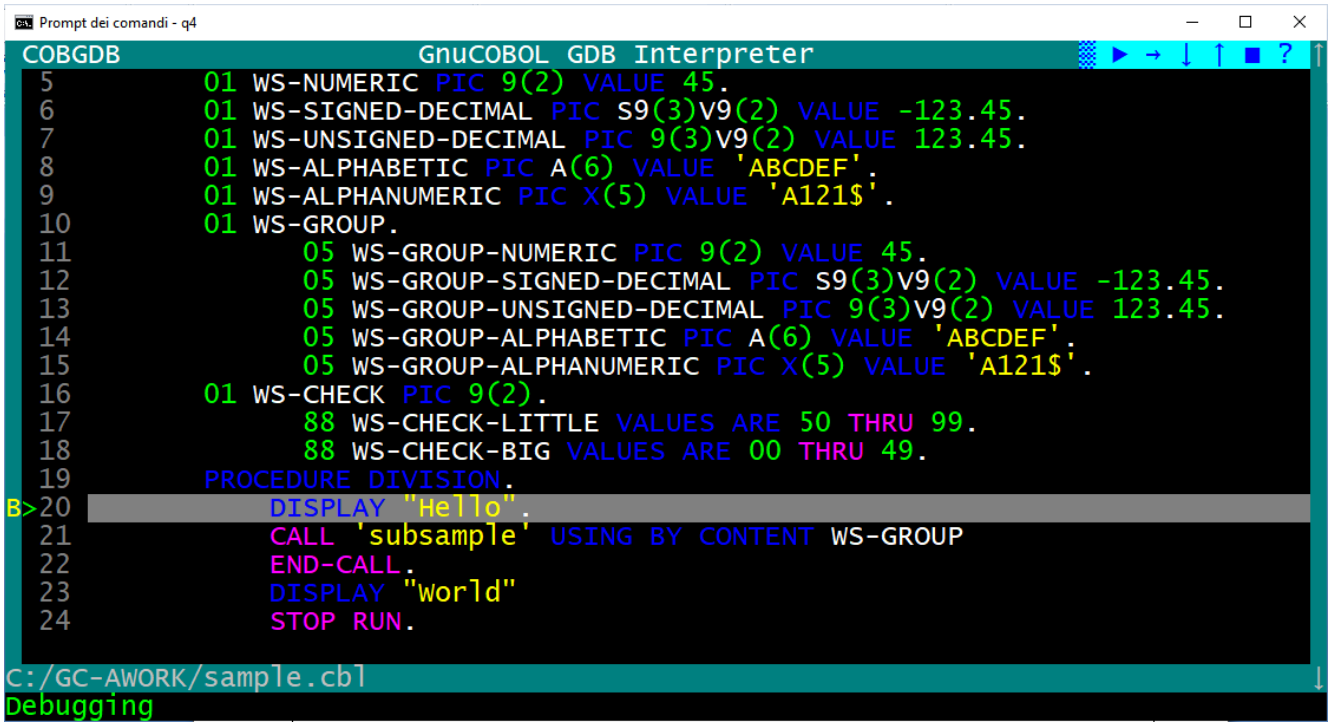
COBGDB shows the selected program source code where in this sample we type a B command at line 14.



now we type the F command again, then select the "sample.cbl" program and press Enter

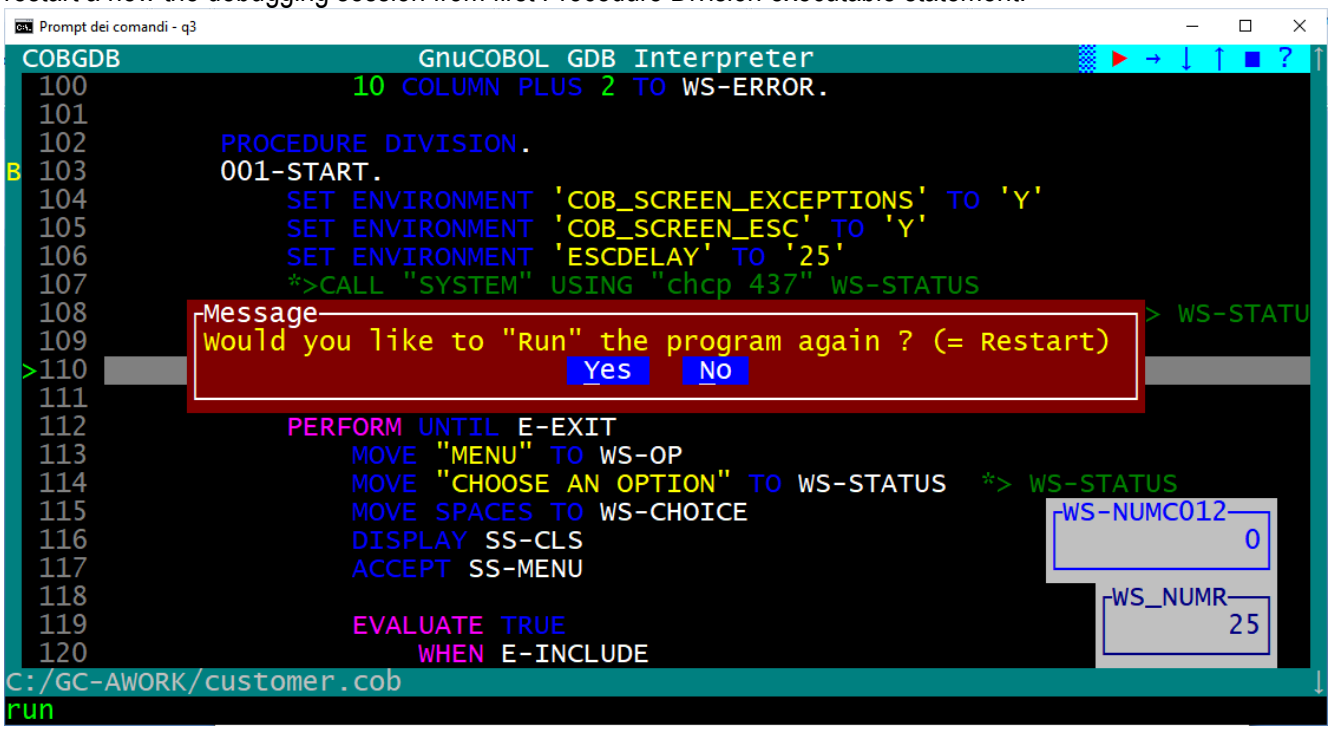


now we are back to the sample.cbl program to continue the debugging session as we need.

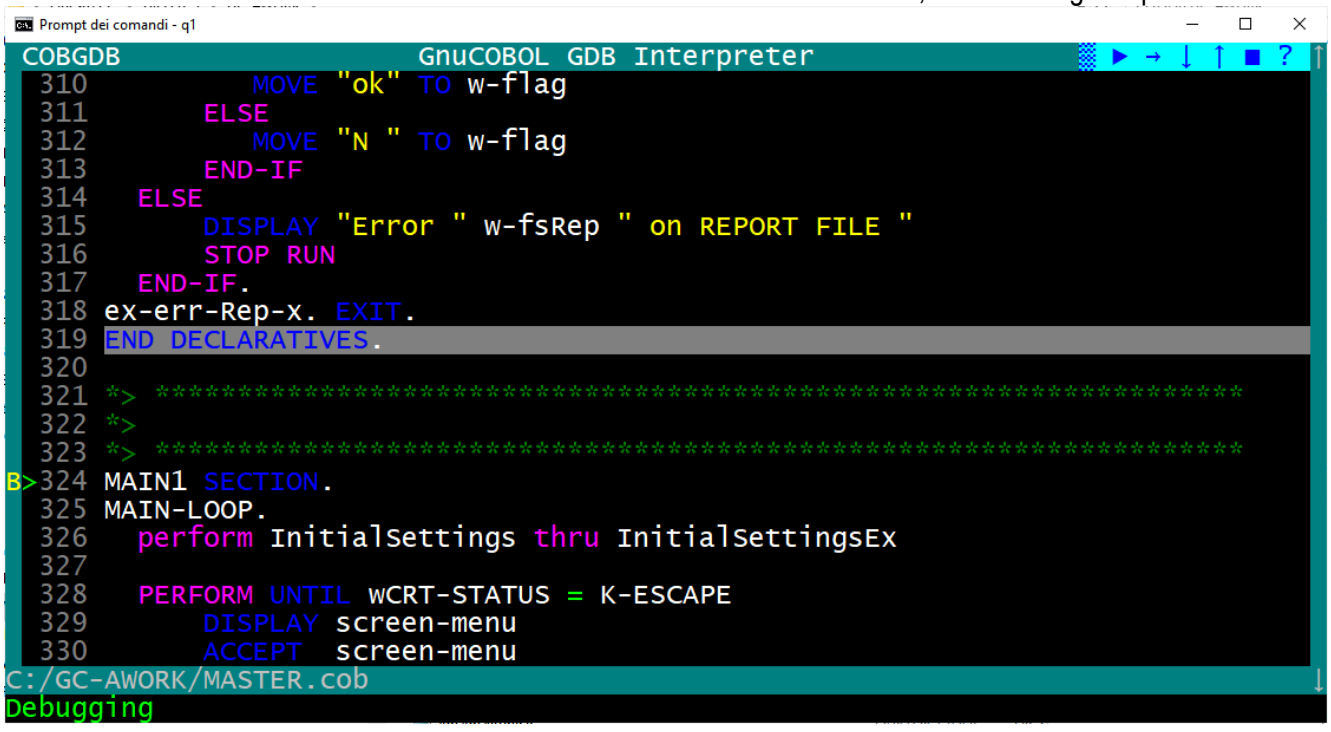


2.12. Run Command

If you click the Run command during a debug session you will receive a confirmation request, because Yes will restart a new the debugging session from first Procedure Division executable statement:

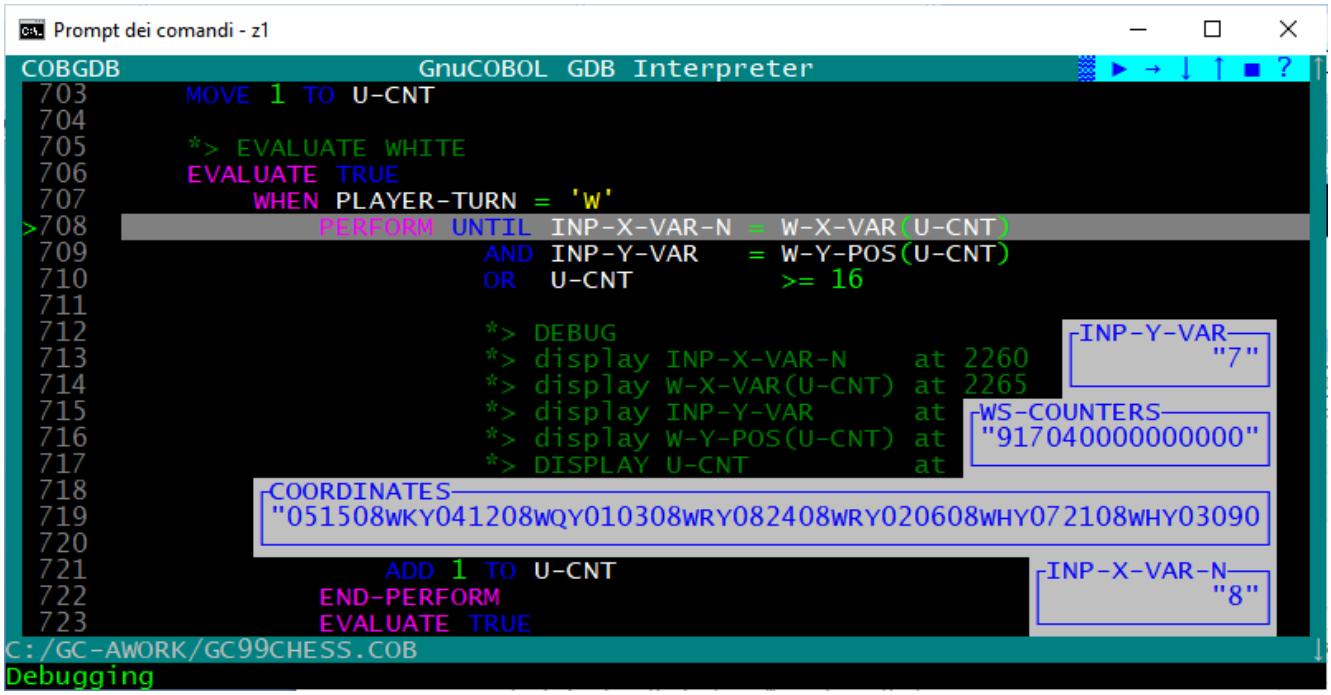


Note: if program has DECLARATIVES then the first automatic B Breakpoint will be settled at first executable PROCEDURE DIVISION statement that is the one after END DECLARATIVES, see following sample:

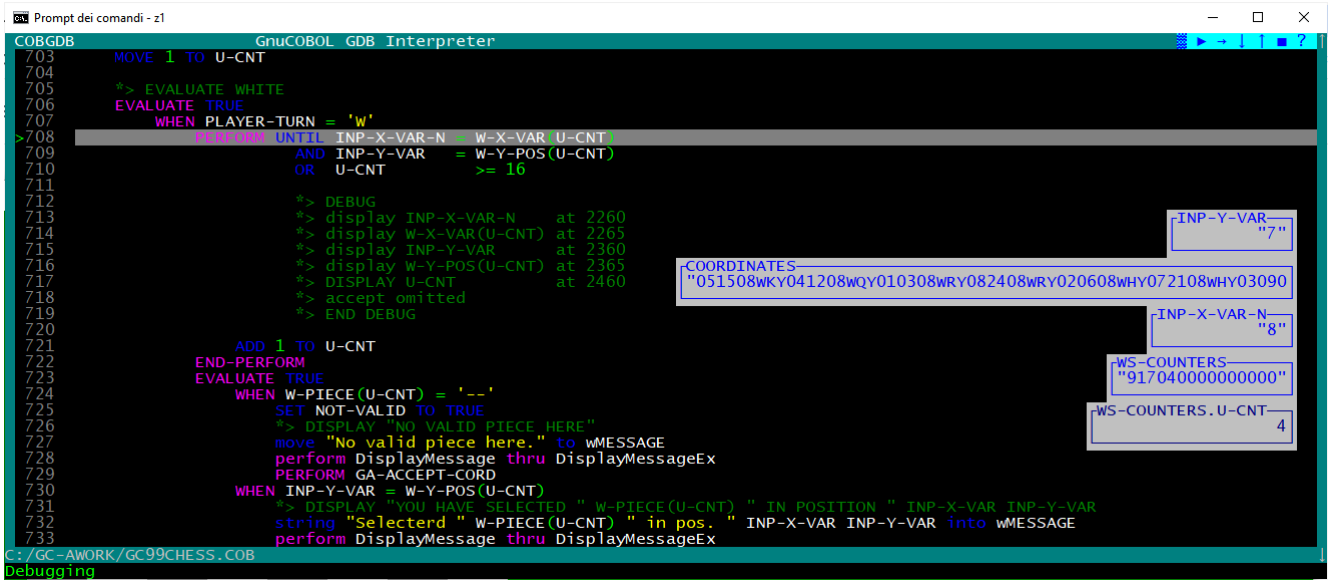


2.13. Window Size command

When you start a debugging session, the screen size is 24 x 80 columns.

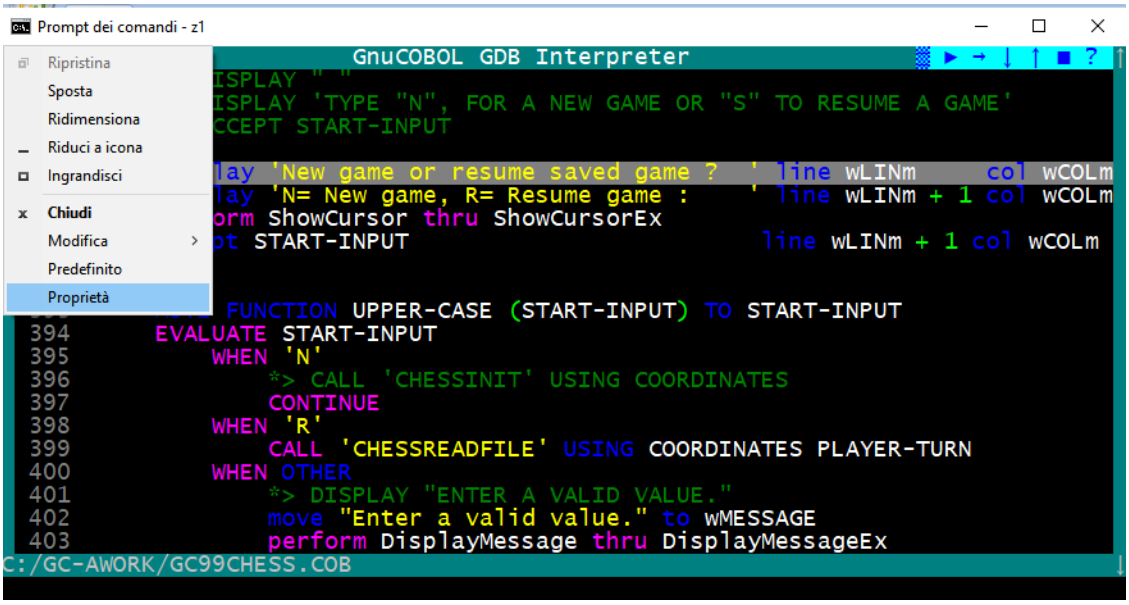


Type **W** (Window Size) command to switch between two window size of the debugger : 24 x 80 or 34 x 132.

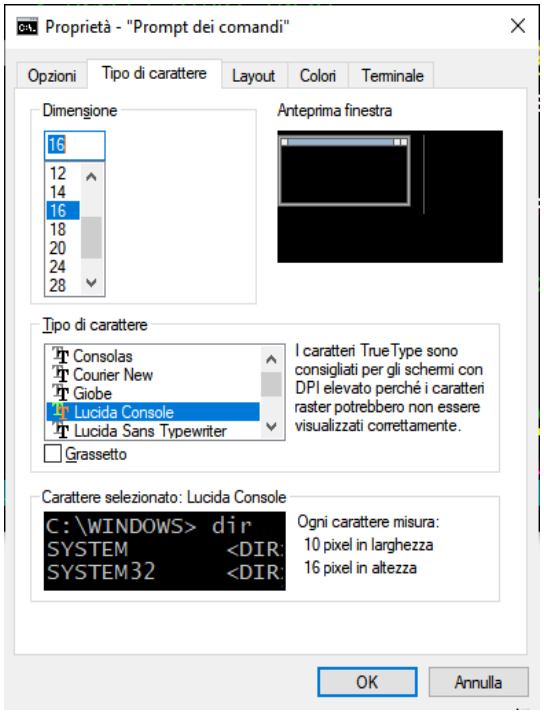


This can be very useful when you write GnuCOBOL code with the GnuCOBOL FREE FORMAT source option where each line of code can be longer than the classic 80 characters of GnuCOBOL FIXED FORMAT.

Warning.
Make sure the system font size is not too large otherwise the W command cannot display the 132 columns (your screen will "blink"). For a Windows environment use "Property" menu item to chek or change the Font size:

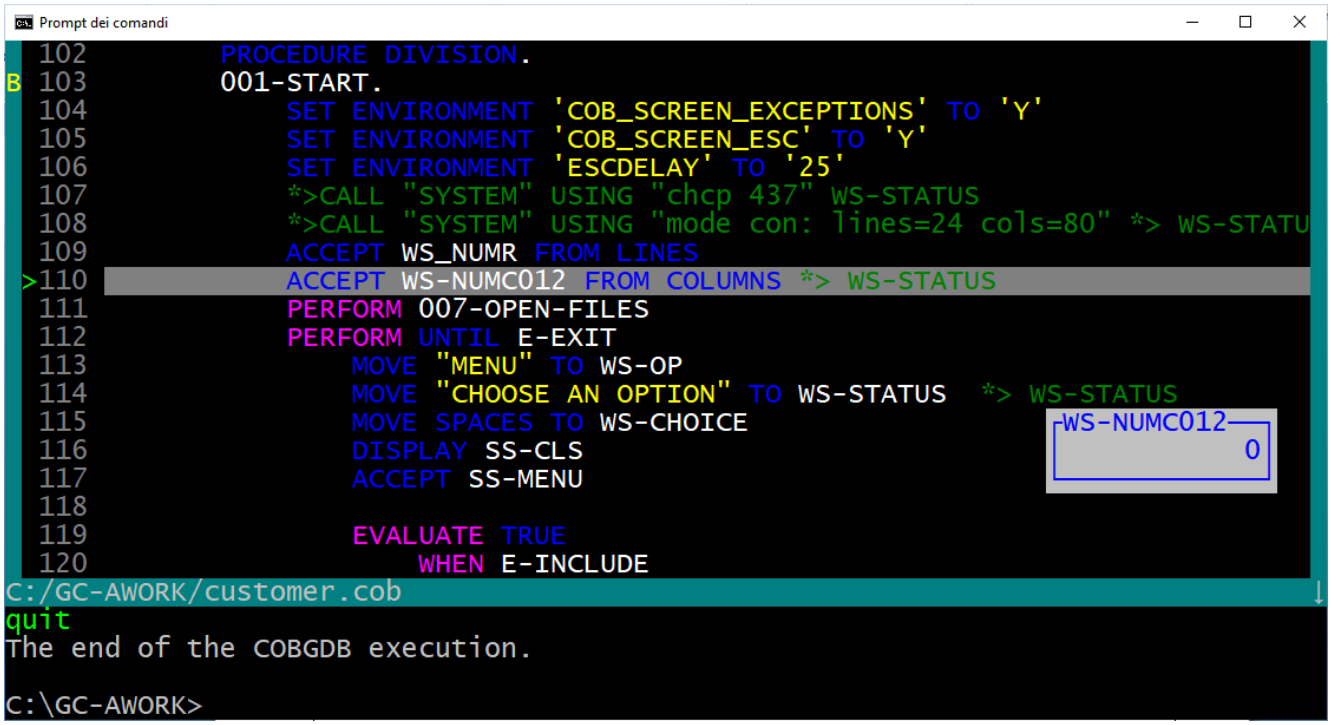


and then select a suitable font, for example Lucida Console of 16 might be adequate.



2.14. Quit Command

To close the debug session use the **Q** Quit command or left click with mouse the ■ button



2.15. Attach Command

Prerequiste: your programs are all compiled for debug (use the -g, a cobc compiler option), because you just compile everything for debug because that's no big overhead and you can attach and use system corefiles at any time. COBGDB needs the .cob, .c, .c.h, and .c.l.h files in order to perform debugging.

Scenario 1. Start a program from the command line.
You are "somewhere in the application" and find the current state to be strange or can reproduce the bug by something you do. Now you just open the debugger from a different terminal, **Attach** to the program, debug normally, then either kill or detach.

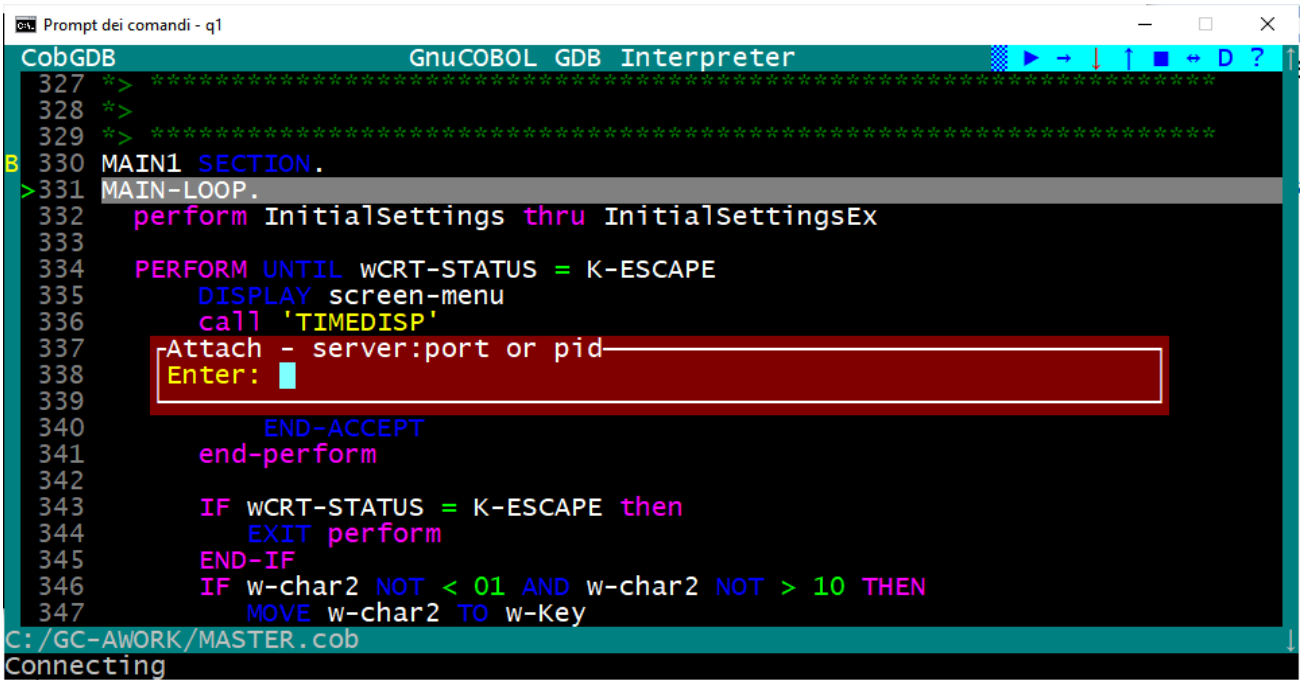
Scenario 2. The program has no terminal at all (like with [web]services, Java/C/HTML frontends, runs in the background, is started by other COBOL programs, ...) it is just "somehow started". Now you just open the debugger from a different terminal, **Attach** to the program, debug normally, then either kill or detach.

You can run GDB/GDBSERVER remotely using the A key. COBGDB will prompt you to provide the server and port in the format server:port or the PID of the application.
Example:

- `server:port = localhost:5555`
- `PID = 9112`

Command line:

- `cobgdb --connect localhost:5555 prog.cob`



3. Other Line Commands

3.1. Debugging a pre-compiled Program

You can use COBGDB to debug a previously generated executable file ex. **`prog.exe`**.
To do this, you must first compile the GnuCOBOL program **`prog.cob`** with these options:

```
cobc -g -fsource-location -ftraceall -v -O0 -x prog.cob prog2.cob ...
```

To start debugging without recompile the program, run cobgdb using the **`--exe`** directive as follows:

Windows:

```
cobgdb --exe prog.exe
```

Linux:

```
cobgdb --exe prog
```

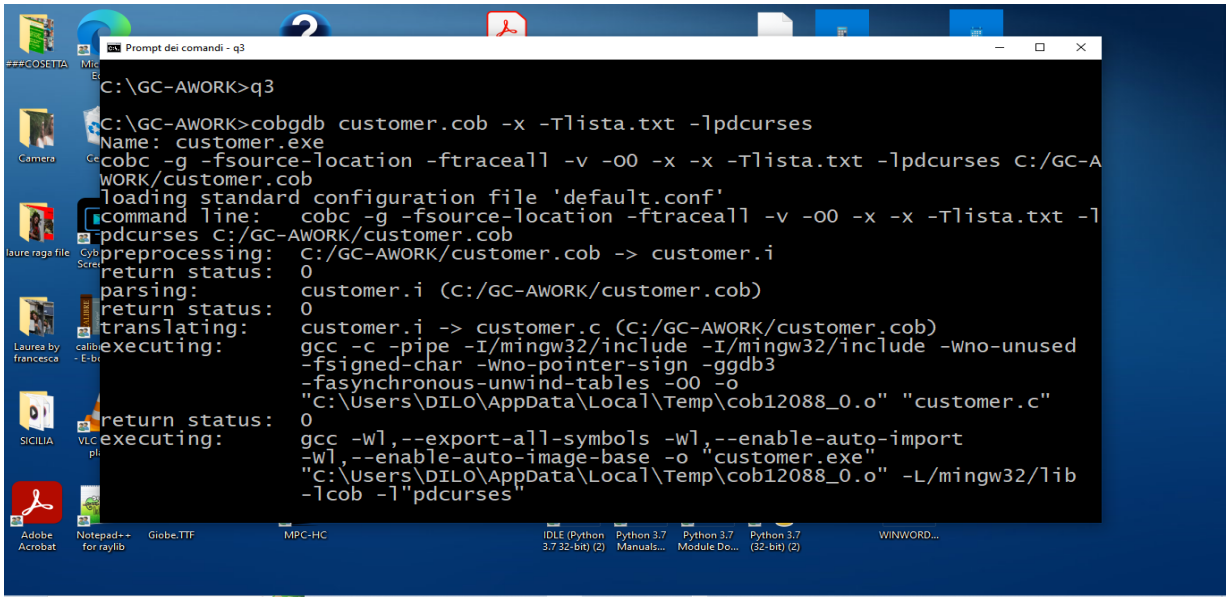
3.2. COBGDB Version

Use command option: **`cobgdb --version`**

to display CobGDB version informations as follows:

```
CobGDB - GnuCobol GDB Interpreter - version 1.4.4
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
This CobGDB was configured as "MinGW32".
For bug reporting instructions, please see:
<https://github.com/marcsosduma/cobgdb>.
The end of the CobGDB execution.
```

4. Document Change Log

CHANGE LOG
Version 1 of 2023.12.12. *First release* . ***Version 2 of 2023.12.23.*** *Step by Step sample of use is added* *Some minor changes* . ***Version 3 of 2024.02.18.*** *Restructured showing new cobgdb screens and features* . ***Version 4 of 2024.04.01 and 20240403.*** *Added EDIT subcommand at H Show Command when viewing the variable from a line of code* *Added cobgdb --version option* . ***Version 5 of 2024.05.01.*** *Added the W Window size command to change User interface screen size* . ***Version 6 of 2024.05.05.*** *Added the --exe option to debug precompiled program* . ***Version 7 of 2025.04.24.*** *Added the new D command and icon useful to enable or disable automatic display variables during debugging / animation* . ***Version 8 of 2025.06.16.*** *Added the new O (Focus) command and icon useful to switch the focus to application screen from debugging / animation screen* . ***Version 9 of 2025.10.20.*** *Added more documentation for the A (Attach) command.*

Technical info