BCI433 Lab 3B (updated Fall 2021)

**Writing an interactive RPGLE screen program**

**Lab objectives:**

* **Use Case structure to solve an application problem**
* **Use the Debug option to check a program’s logic**

**Lab Requirements:**

**Show compiler listing for AUTOINSRPG (with Lab3B logic including Case structure)**

**Successfully run AUTOINSRPG (with Lab3B logic including Case structure)**

Start an RDi session

Start a ‘Green Screen’ (emulator) Session.

**Using** **Rational Developer for i (RDi):**

**Part A**

**Objectives:**

Case structure is a good choice to solve application logic problems. Here is an example of that:

SELECT;

When Day = 1;

DayName = ‘Monday’;

When Day = 2;

DayName = ‘Tuesday’;

When Day = 3;

DayName = ‘Wednesday’;

When Day = 4;

DayName = ‘Thursday’;

……

ENDSL;

If Day is a 1, then the first test would be done, the DayName would be set to ‘Monday’ and control would go to ENDSL. The test for Day = 2 would not be performed. If Day is a 7 then seven tests would be done and a DayName = ‘Sunday’ statement would be executed.

If an 8 or 9 was entered in Day you could include an Other clause in place of a When clause at the bottom of this structure before the ENDSL. **Other** is a catch all.

Other

DayName = ‘Invalid Day Number’

ENDSL

Tests done with the select When statement can include <=, >=, <> (not equal to), AND, OR

In class exercise:

You need to figure out how to convert a number to a grade.

The mid term test is a mark out of 100 and is worth 35% of the final grade. The labs are entered as a mark out of 30 and are 30% of the final mark. The exam is a mark out of 100 and is worth 35%. If a student fails either the final exam or the test, they will fail the course. The final grade is stored in a field called NUMGRADE. After this grade has been determined, it needs to be converted to a letter grade LETGRADE.

The following information is used to do this.

A+ = 90 - 100 A = 80 – 89 B+ = 75 - 79 B = 70 – 74

C+ = 65 - 69 C = 60 – 64 D+ = 55 - 59 D = 50 - 54

F = 0 – 49

Solution:

Add one additional validation to your logic:



Processing for Lab3B

You have an interactive RPGLE program running from lab 3a with no logic to determine the annual insurance cost and the age of the vehicle. Apply the information and features below to determine the correct output for completion of lab 3b.

A rate is determined but is not shown on any screens. Search on how to define a three decimal free format standalone field. Do a google search for “**rpgle standalone field free format**” Don’t waste a lot of time on how to do this – if you need help, ask your instructor. Refer to this in your program as RATE.

With the first screen example the **rate** was determined to be .048. That rate is applied to the replacement value of the vehicle to determine the annual cost.  45,000 \* .048 = 2,160.

Graphical user interface, text, application, email

Description automatically generated

In your main routine use two subroutines to determine the rate.

**KilometerRate** – will determine the rate according to the number of kilometers driven in a year. Use Case Structure here.

**DrivingRecord** – will adjust the rate according to the person’s driving record. Use Case Structure here.

Employ a subroutine to Reset all the fields called CLEAR. You can call the subroutine, but do not have to code it.

 Assume all the fields are reset to zeroes and the VYear is reset to **D’0001-01-01’**

Indicators are not reset in the CLEAR subroutine, reset all indicators after they have served their purpose.

Insurance Rate % of Replacement Value

|  |  |
| --- | --- |
| Kilometers Driven | Insurance Rate |
| 60,000 or higher | 3% |
| 40,000 – 59,999 | 2.6% |
| 20,000 – 39,999 | 2.3% |
| 10,000 – 19,999 | 2% |
| 1 – 9,999 | 1.3% |

If Demerit points are 13 or higher, you need to add 3% to the insurance rate. If Demerit points are 10 – 12 inclusive you need to add 2.8%. When Demerit points are 5 – 9 inclusive the rate is adjusted up by 2.5% and for people with 1 – 4 Demerit points there is an extra 2%.

Drivers with one accident get a free pass, but if they have had two accidents their rate goes up by 4%. If they have had 3 accidents their rate goes up by 5%. The company does not want people with more than three accidents, so the rate goes up by 30% and a comment appears that indicates the company does not want that customer.

The final driving record adjustment takes into account the number of traffic ticket convictions.

|  |  |
| --- | --- |
| Traffic Tickets | Rate adjustment up |
| 1,2 | .5% |
| 3 | 1% |
| 4 or more | 30% and We don’t want your business comment |

Graphical user interface, text, application, email

Description automatically generated

This next part requires BCI433LIB to be on your RDi library list when compiling and on your green or white screen library list when running your program.

Your program will use a display file called BCI433LIB/EXITDSP to do the following.

When the F3 key is pressed to exit the program the following screen record should show. It is available from the compiled display file named above. It is called EXITRECORD.

Text

Description automatically generated

The user must use a mouse to click on one of the choices and then press the enter key.

If the first option is selected, you will show a screen record called INSUREINFO and then the program will end. INSUREINFO is part of the EXITDSP display file.

Just above the \*INLR = \*ON statement include the following logic.

SELECT;

WHEN OPTION = 1;

?

WHEN OPTION = 2;

?

//(Option is defined in EXITDSP)

The first option uses a INSUREINFO screen record.

Graphical user interface, text

Description automatically generated

The second option uses a COMPINFO screen record.

Graphical user interface, text, application

Description automatically generated

The best way to check the behaviour of the program is to run the instructor version.

ChgCurLib BCI433LIB

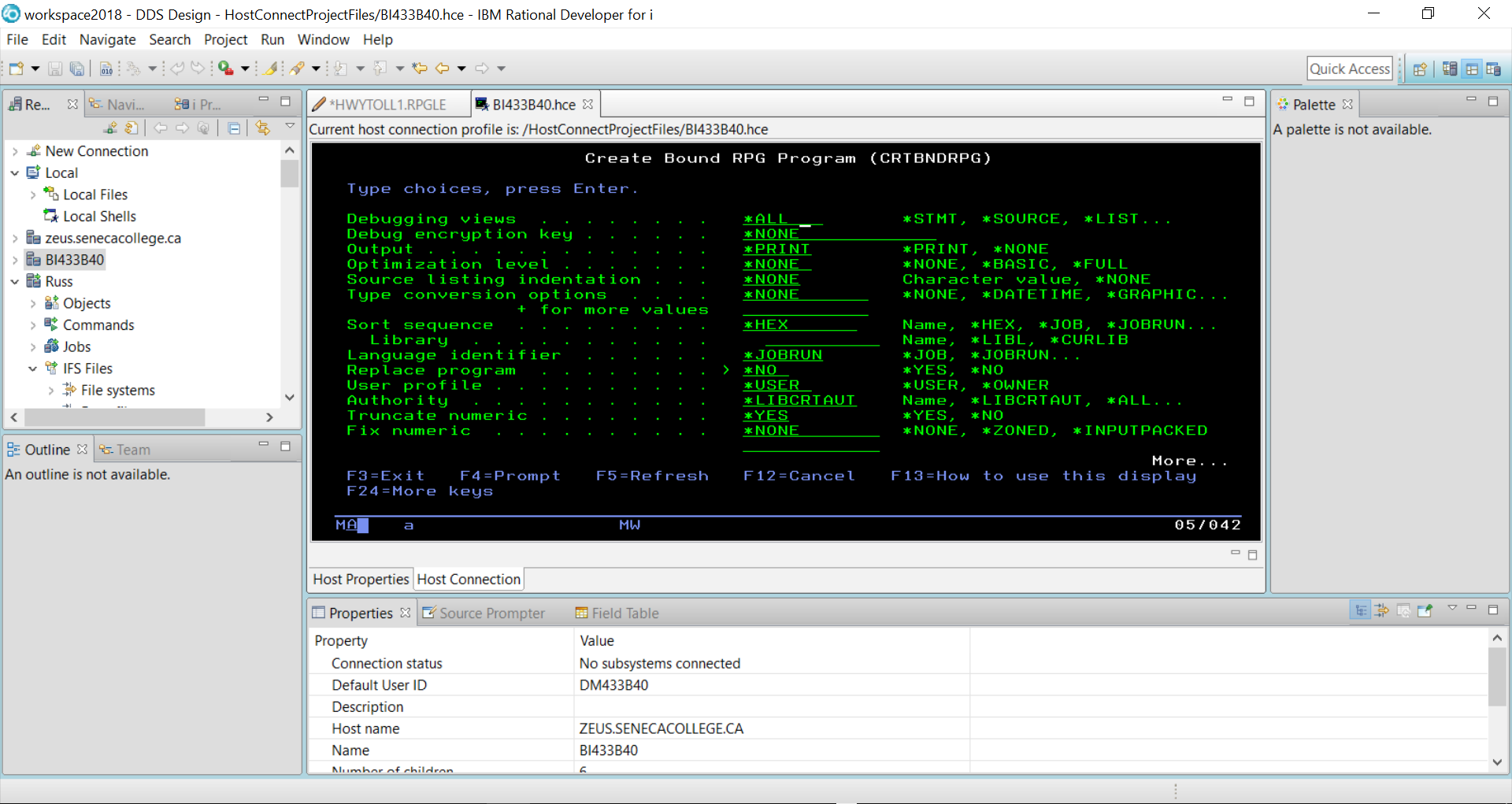
Call AUTOINS3B

ChgCurLib (Back to your library)

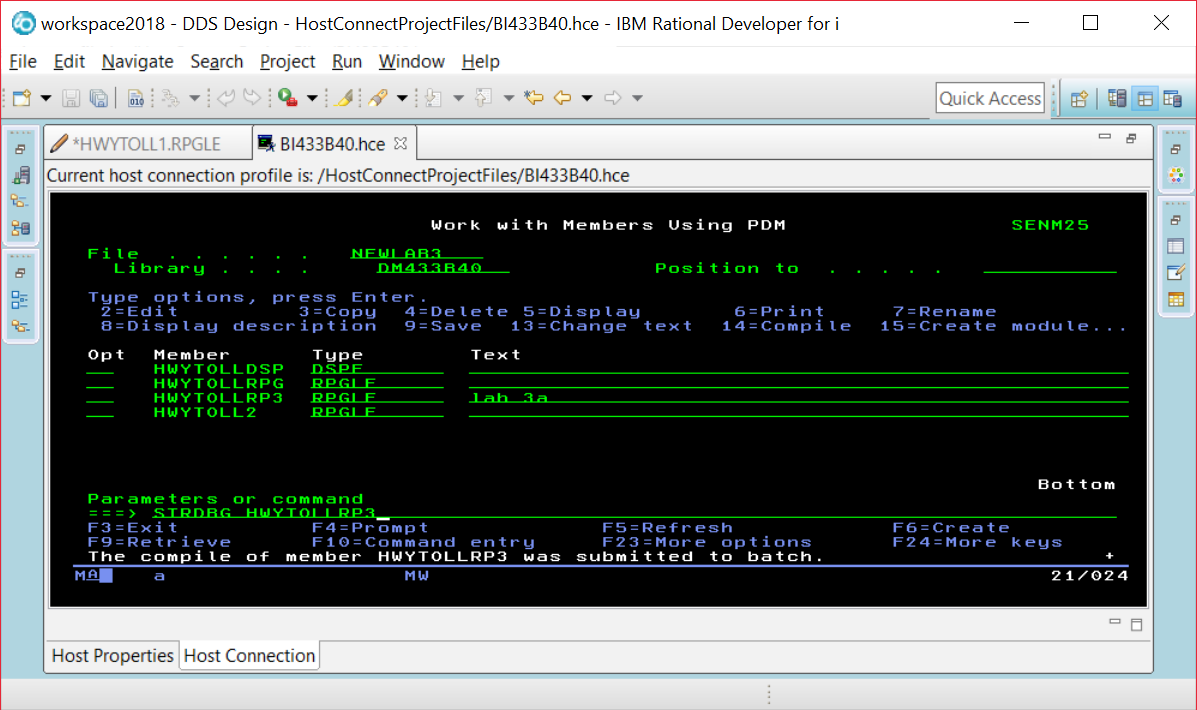
Your program should get similar results. In order to test your program, your professor will try a few scenarios, scan your PDF listing to see if there are any obvious errors. Your listing can be printed by a PC printer and should have utilized the CALL STRJOB command before compilation so your id and name appear at the bottom of each page.

Using Debug With an Interactive RPGLE Program

Green screen compile requires an option to be changed. Debugging views should be set to all. RDi compile does this by default.

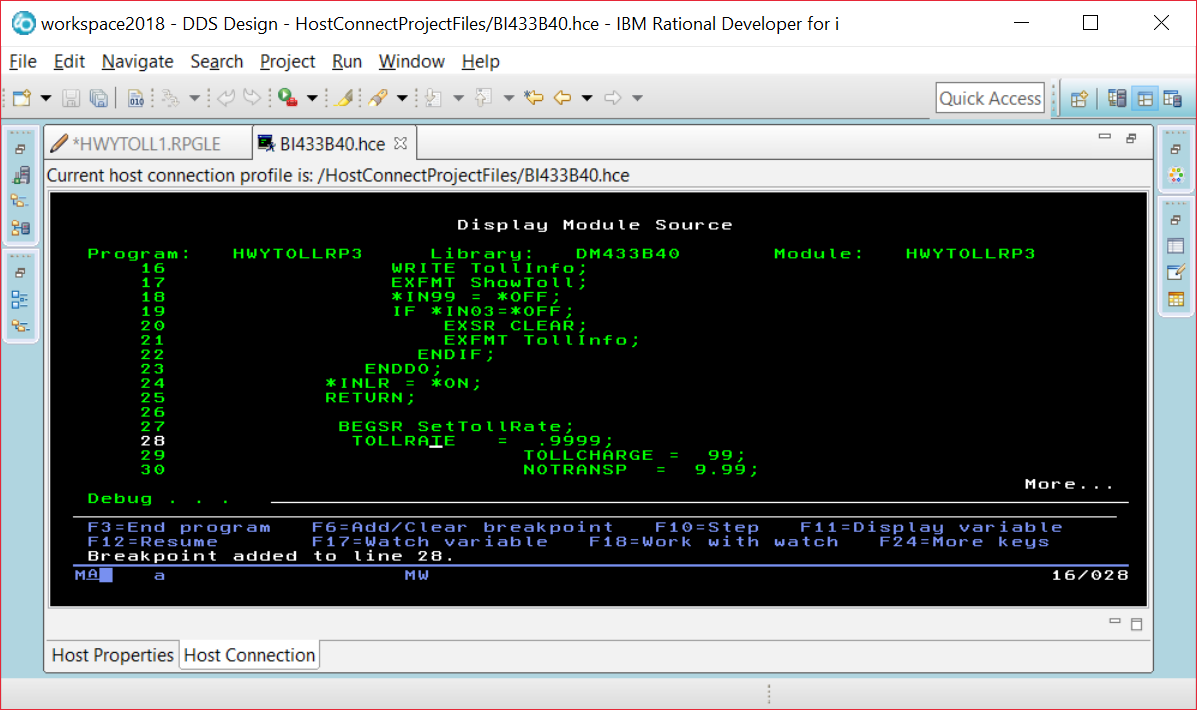


The STRDBG command allows you to set a breakpoint in the program when it is running.

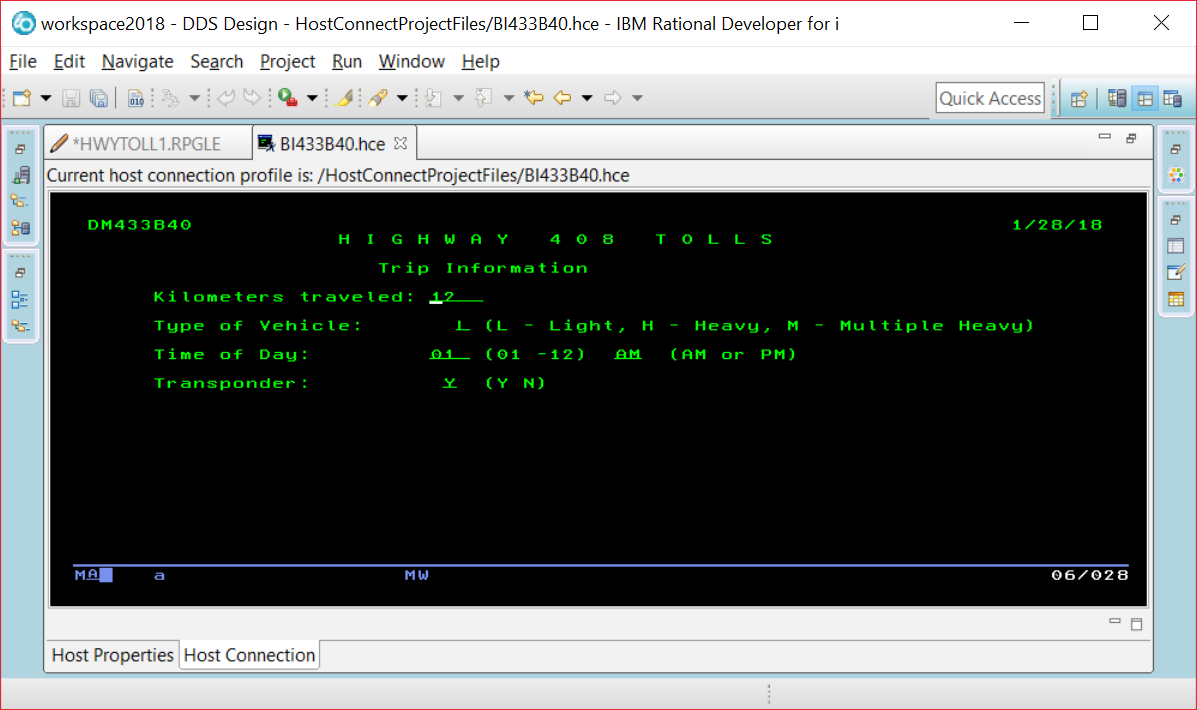


If you are already in DEBUG mode, this command will not work. Just enter the ENDDBG command to exit from DEBUG mode and then enter the STRDBG command with your program name.

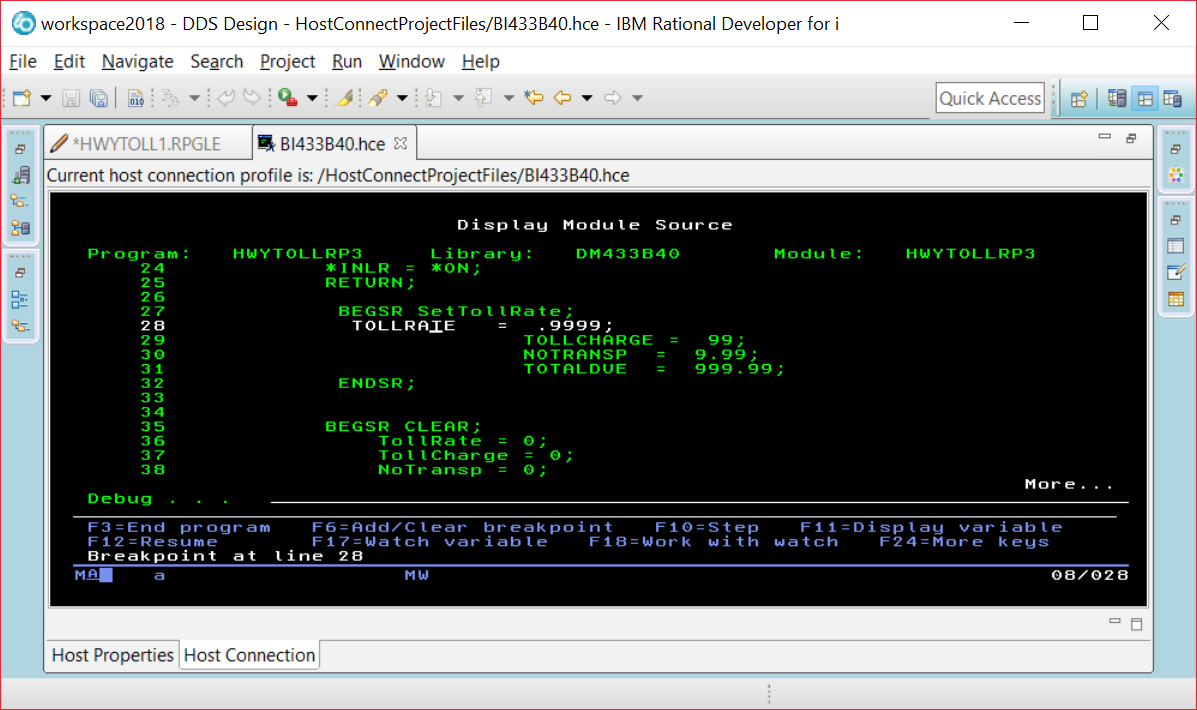
Page down and put your cursor on the line where you want a breakpoint, and press F6.



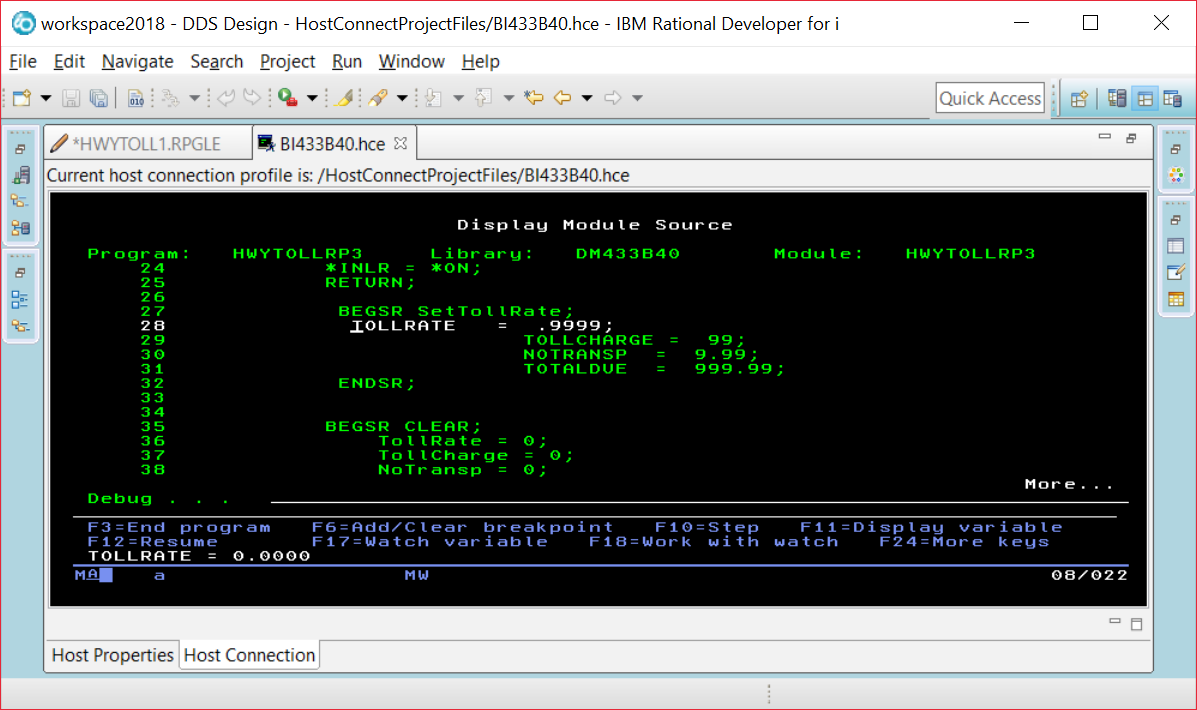
When you run the program you will see screens the program displays before the breakpoint. Our breakpoint was after the EXFMT line, so we see this screen and can enter data into the fields.



The breakpoint is reached and this line has not been executed yet. So TOLLRATE would still be set at 0.

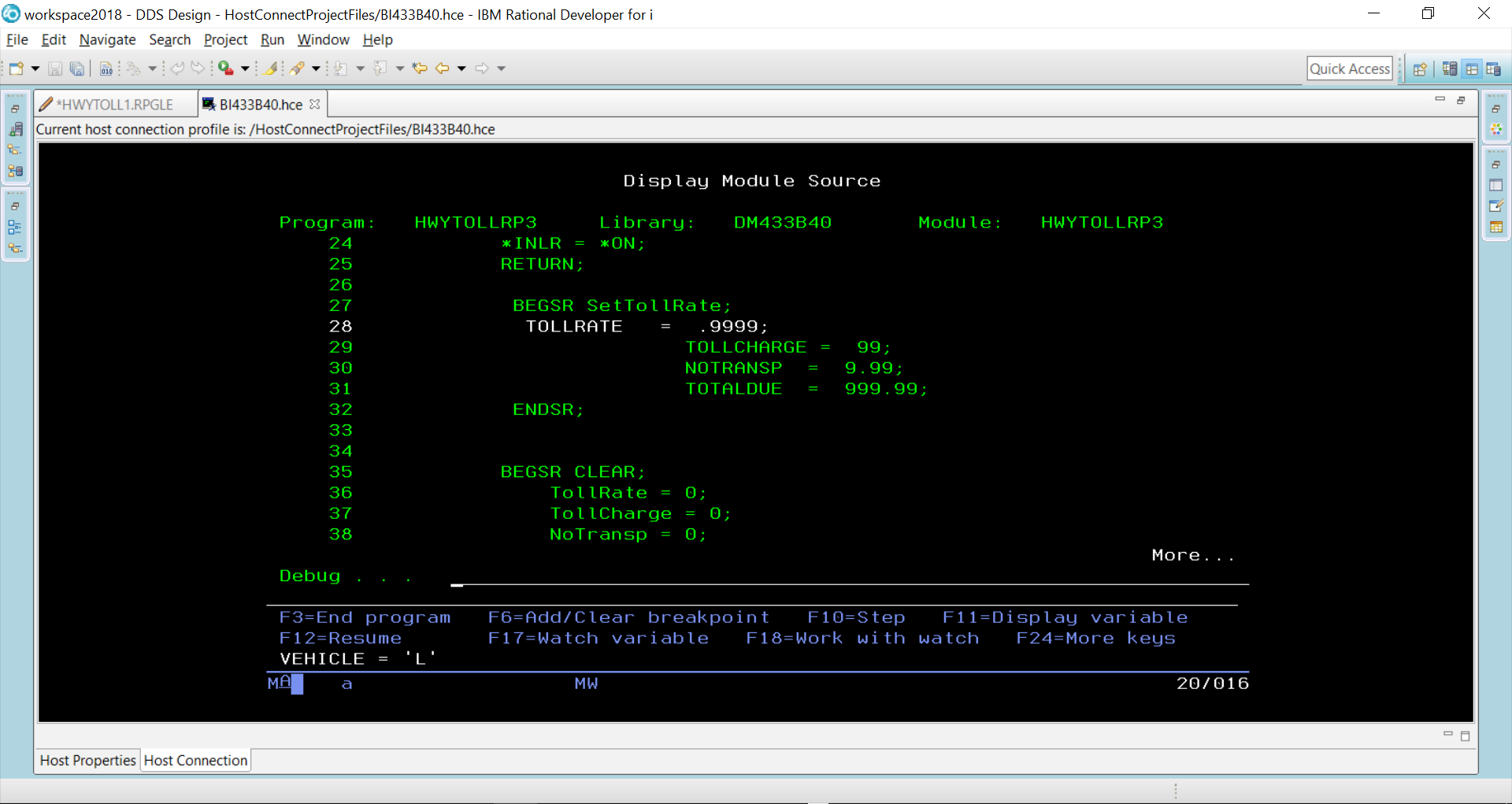


If you put your cursor on the TOLLRATE field, you will see what is currently stored in the field.



Instead of using F11, you run the command EVAL VEHICLE command at the prompt and be able to find out what is in a field that is not currently being displayed in a line of code on your screen.

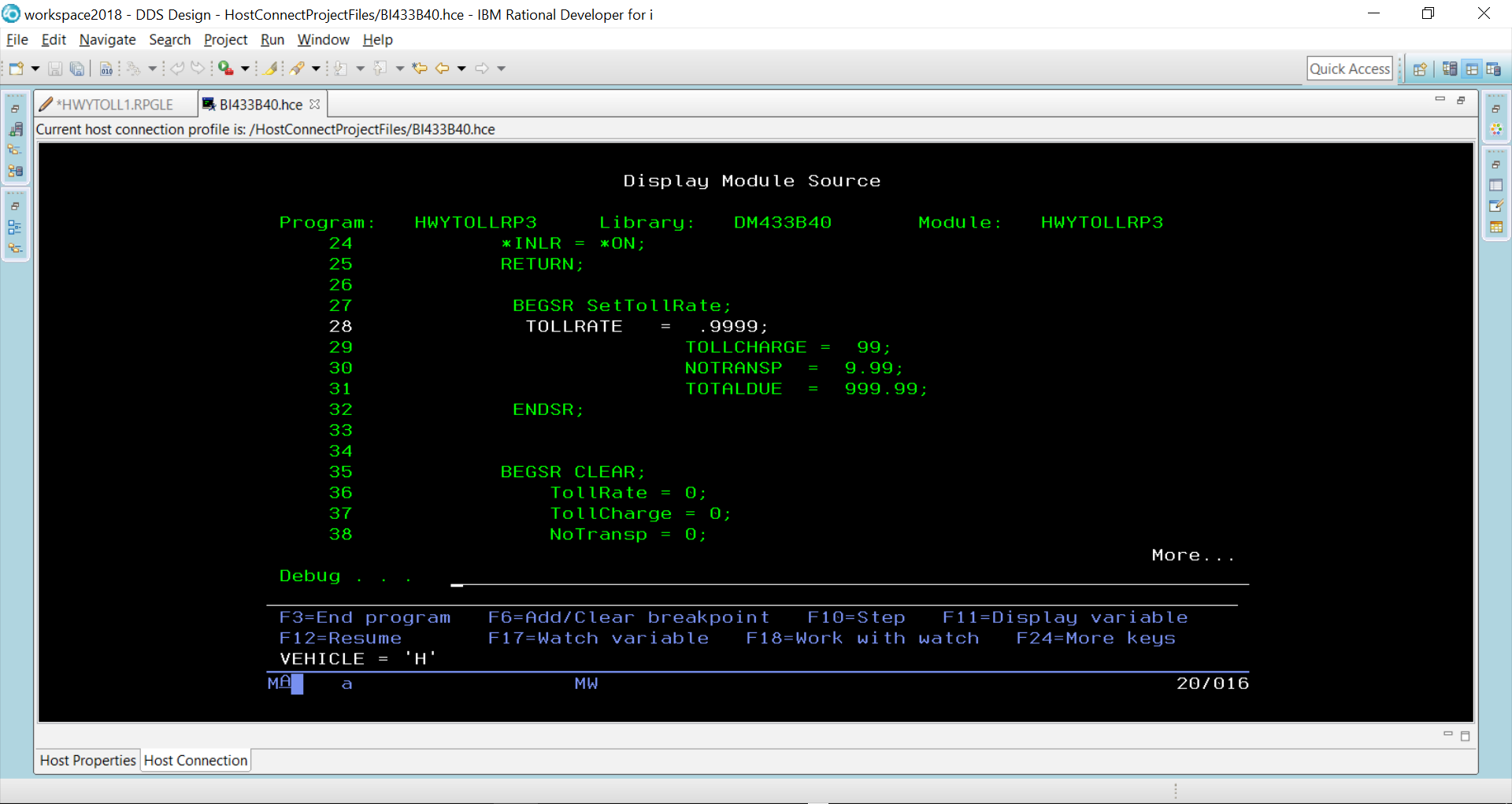
EVAL VEHICLE



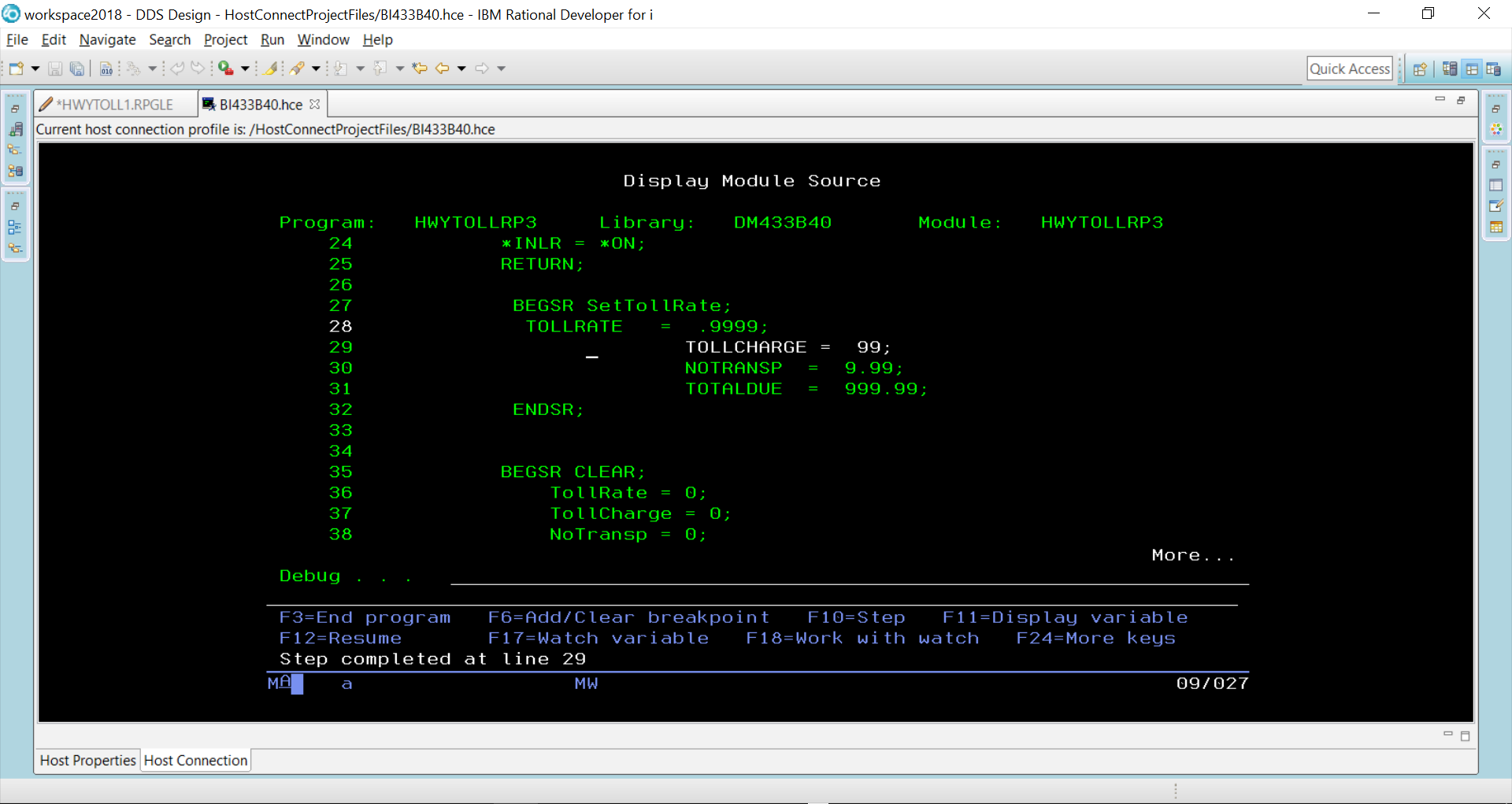
You can alter the contents of a field while your program is running.

EVAL VEHICLE = ‘H’

EVAL VEHICLE



F10 is pressed to step through each line of code



TollRate would now be set at 9999 because line 28 has executed.

When F12 pressed – the rest of the code executes until we get to where the two screens are made available and the program is paused.

