BCI433 Lab 3B (updated Winter 2020)

**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:**

**Hand in the signed compiler listing for INCTAXRPG (with taxrate logic)**

**Successfully run INCTAXRPG (with taxrate logic)**

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 chart 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:

SELECT;

WHEN NUMGRADE >= 90;

LETGRADE = ‘A+’;

When(numgrade > = 80);

Letgrade = ‘A+’;

When(numgrade > = 75);

Letgrade = ‘B+’;

When(numgrade > = 70);

Letgrade = ‘B+’;

When(numgrade > = 65);

Letgrade = ‘C+’;

When(numgrade > = 60);

Letgrade = ‘C’;

When(numgrade > = 55);

Letgrade = ‘D’;

Other;

Letgrade = ‘F’;

ENDSL;

Do not test both the upper and lower limits of a range – you only need to test one or the other depending on your approach. Testing both limits is wasting CPU time. Not a big problem with this application, but may be a factor when processing millions of records.

You have an interactive Income tax RPGLE program running from lab 3a with no logic to determine the correct income tax rates. Apply the information and features below to determine the correct output for completion of lab 3b.

Processing.

The following table is used by your program to determine the tax rates:

Income Tax Rates

|  |  |  |
| --- | --- | --- |
| Income Level Single | Tax rate | Income Level Married |
| 0 – 9,525 | 10% | 0 – 19,050 |
| 9,526 – 38,700 | 12% | 19,051 – 77,400 |
| 38,701 – 82,500 | 22% | 77,401 – 165,000 |
| 82,501 – 157,500 | 24% | 165001 – 315,000 |
| 157,501 – 200,000 | 32% | 315,001 – 400,000 |
| 200,001 – 500,000 | 35% | 400,001 – 600,000 |
| 500,001 + | 37% | 600,001 + |

Income tax is due on April 30. So whenever your program is run this year, you should be able to report on how many days until taxes are due. If your program was run on January 12, 2020 your program would report back “Your Taxes Are Due in 109 Days”. If the program is run after the tax due date, a negative value would show for days if you use an edit code for the DaysToPay field that supports showing the negative sign.

Some screen shots are provided to highlight aspects of the program. The best way to check the behaviour of the program is to run the instructor version.

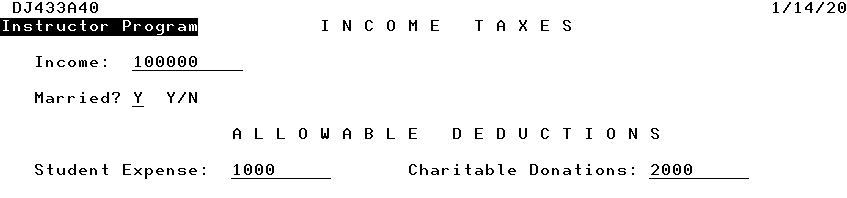
ChgCurLib BCI433LIB

Call INCTAXRPGB

ChgCurLib (Back to your library)

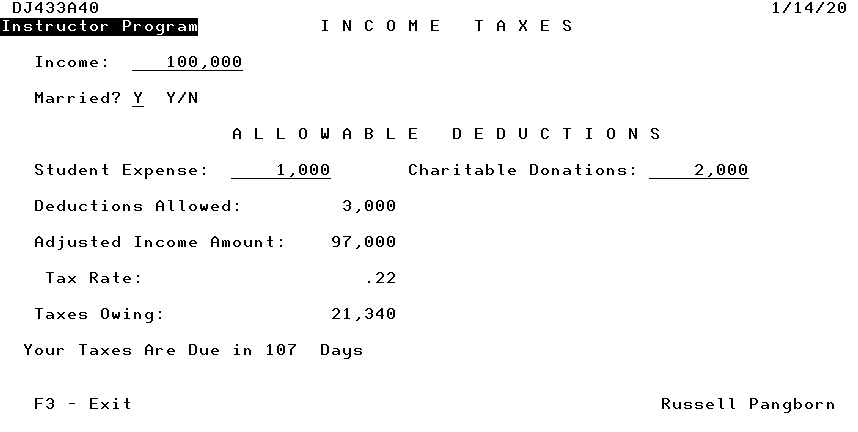
Your program should get similar results. In order to test your program, your professor will try a few scenarios, scan your signed paper listing to see if there are any obvious errors and keep the listing. Your signature indicates you did not electronically copy this code from another student in this semester or a previous semester. Your listing is 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.

**First screen:**

****

You will have to determine the income tax rate based on marital status and net income after deductions. based on the tax charts.

**First screen with a second overlaid screen:**



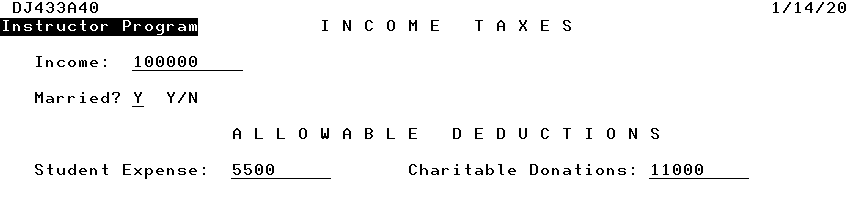
In the first program you calculated the allowable deductions without any limits. The maximum allowed for student expenses is 4,000 and the maximum allowed for charitable donations is 10,000.

IF StdExpense > 4000;

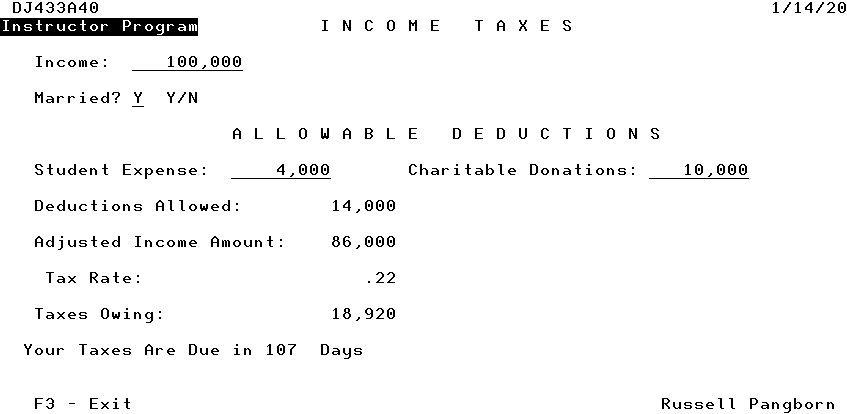
StdExpense = 4000;

ENDIF;

The following indicates how that is handled.



5,500 and 11,000 are readjusted to the maximum allowed amounts.



**Date and Time Arithmetic**

Num\_Days = %DIFF (Today: Due\_Date: \*DAYS);

If Today was set as 2019-09-12 and Due\_Date was set as 2019-09-15

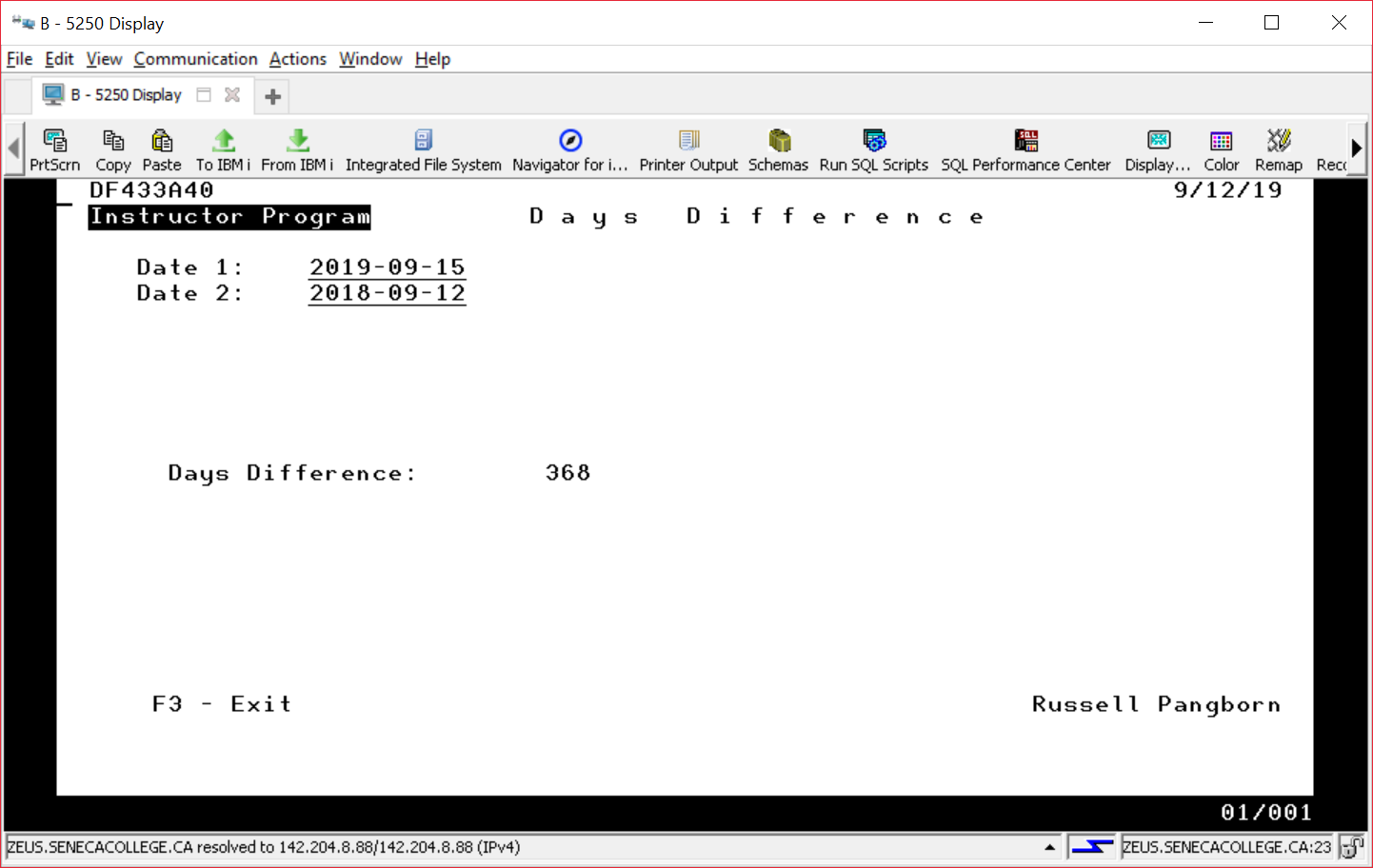
Num\_Days would be equal to –3

If Today was set as 2019-09-15 and Due\_Date was set as 2019-09-12

Num\_Days would be equal to 3

You can also use \*YEARS and \*MONTHS

For times and timestamps you can find a difference in \*SECONDS, \*MINUTES AND \*HOURS

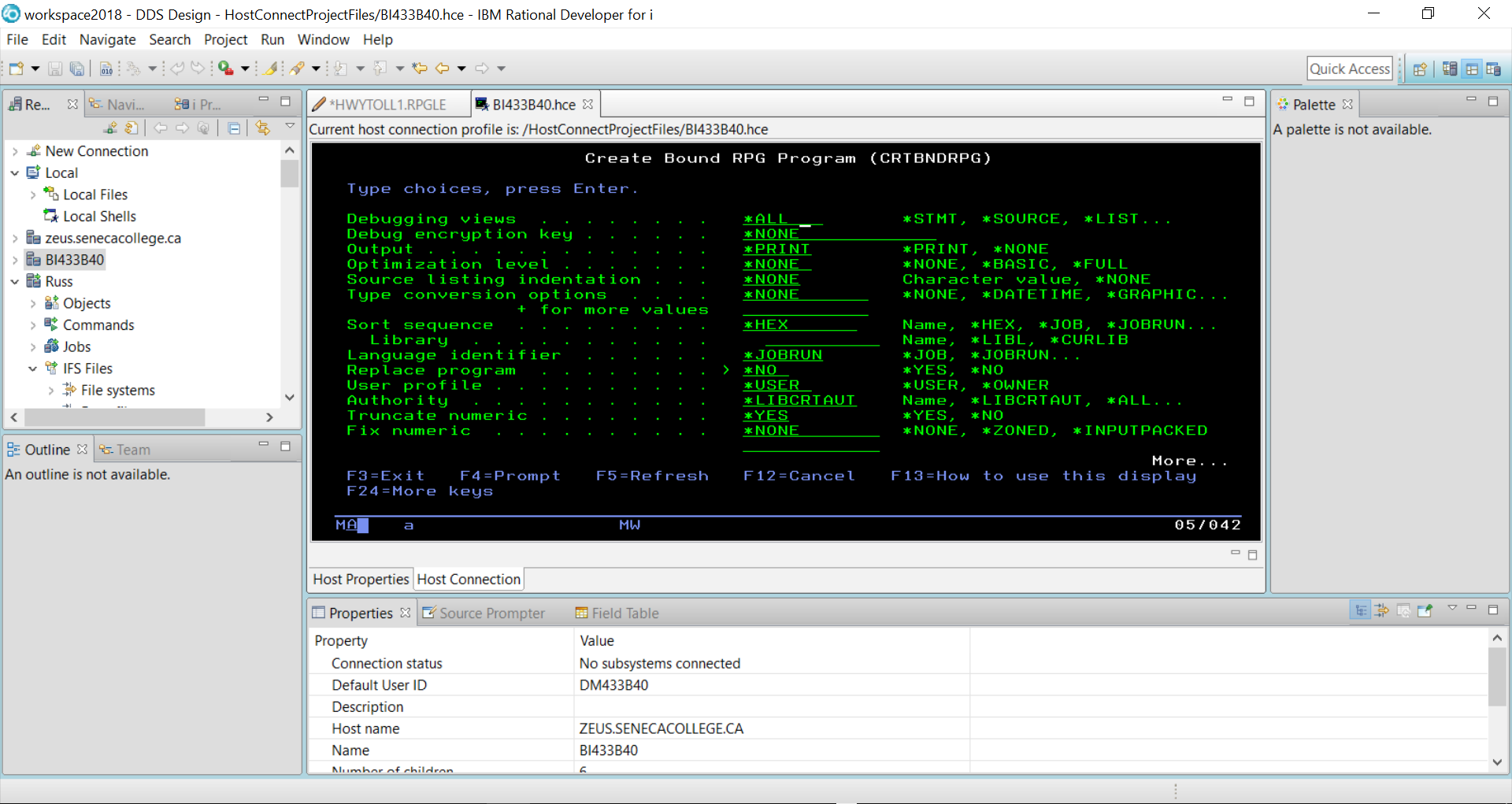


**What was the statement to determine Num\_Days in the above screenshot using the fields DATE1 and DATE2?**

**\_\_\_\_\_\_\_\_DaysToPay = %Diff(d’2021-04-30’: %Date(): \*D);\_\_\_\_\_\_\_**

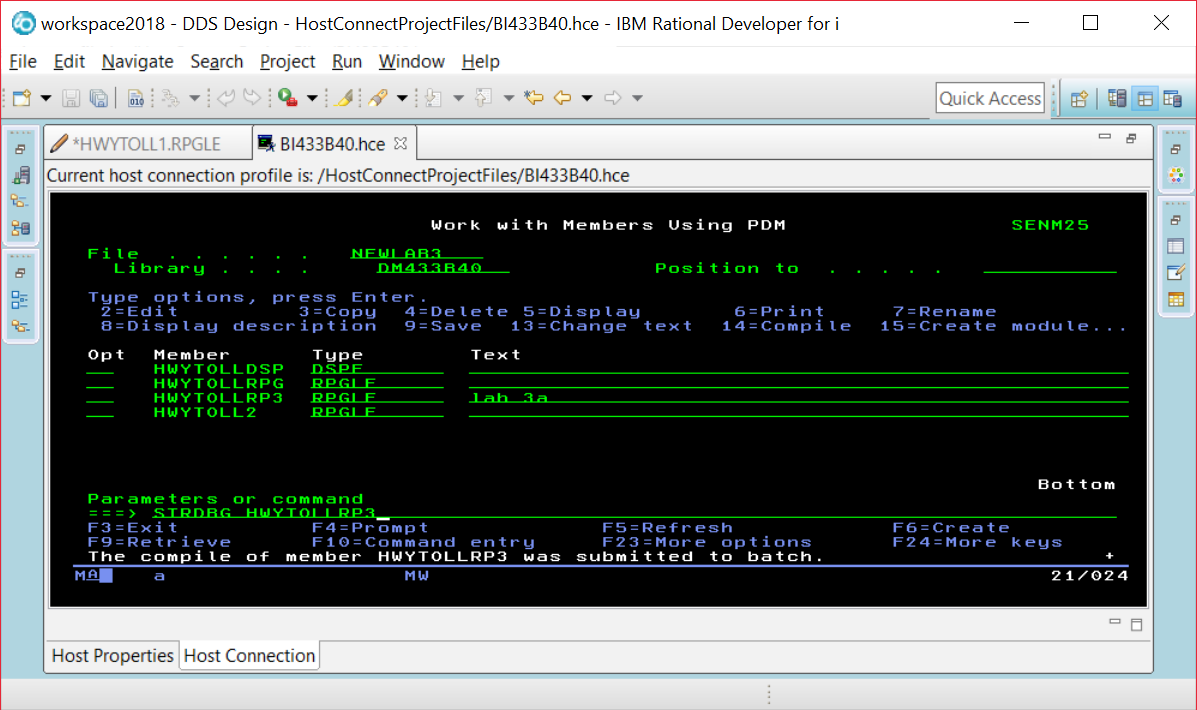
Using Debug With in Interactive RPGLE Program

Green screen compile requires an option to be changed. Debugging views should be set to all. RDi compile does this by default.1.in Rdi: right click INCTAXRPGA.rpgle > select “Compile(prompt)” > CRTBNDRPG ..> set “Debugging views” to ALL > click OK



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

2.Start DEBUG mode: In green screen, issue command=> STRDBG INCTAXRPGA

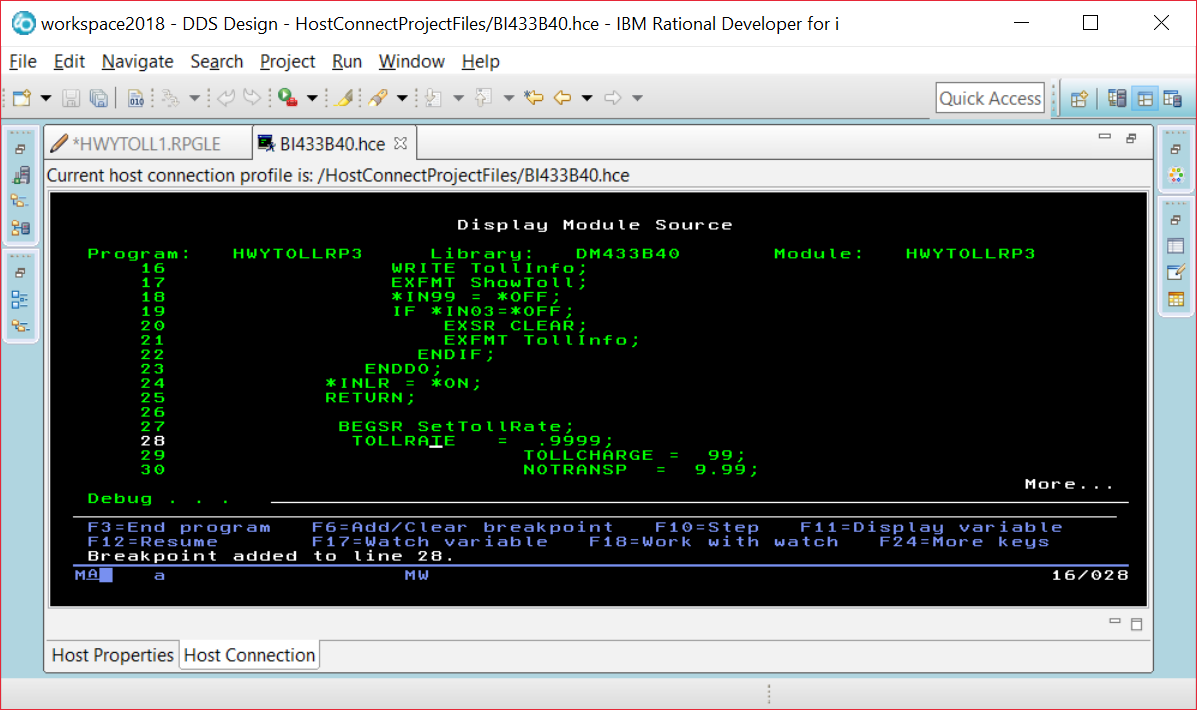


If you are already in DEBUG mode, this command will not work. Just enter the ENDDBG command to exit from DEBUG mode and then enter the STRDBG command with your program name.

3.Set up a breakpoint: move cursor to a line, and press F6

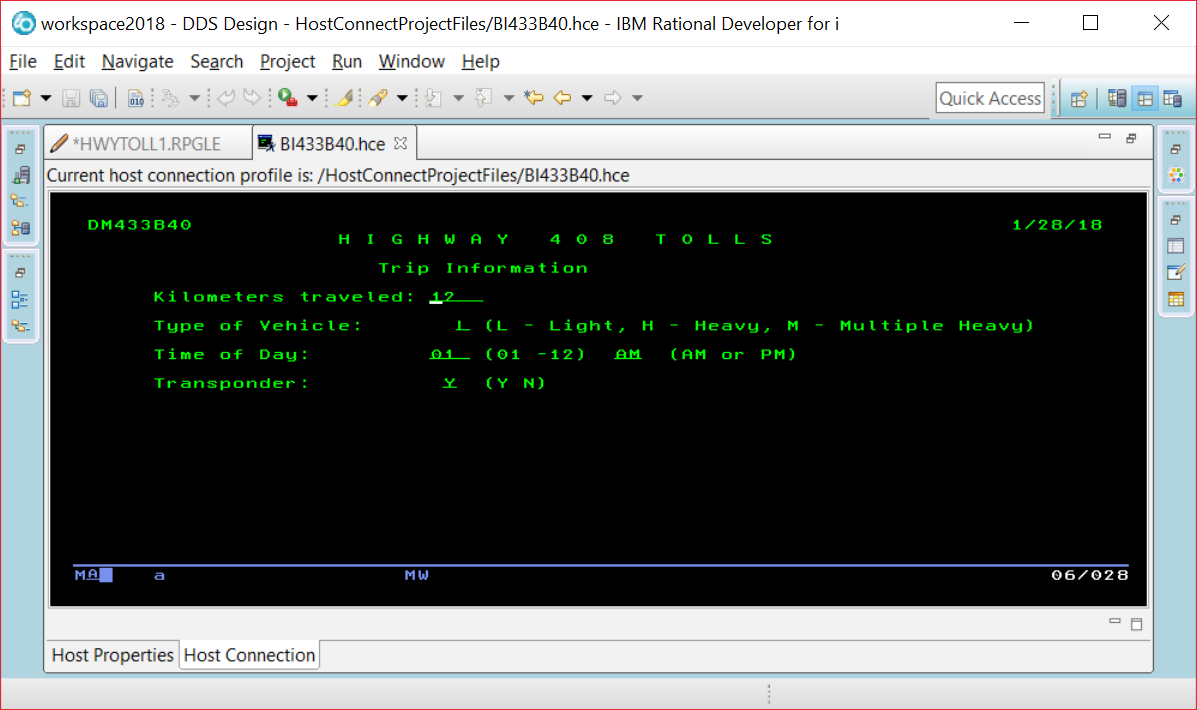
Page down and put your cursor on the line where you want a breakpoint, and press F6.

4.Then press F10 to exit from this screen so you can start your program



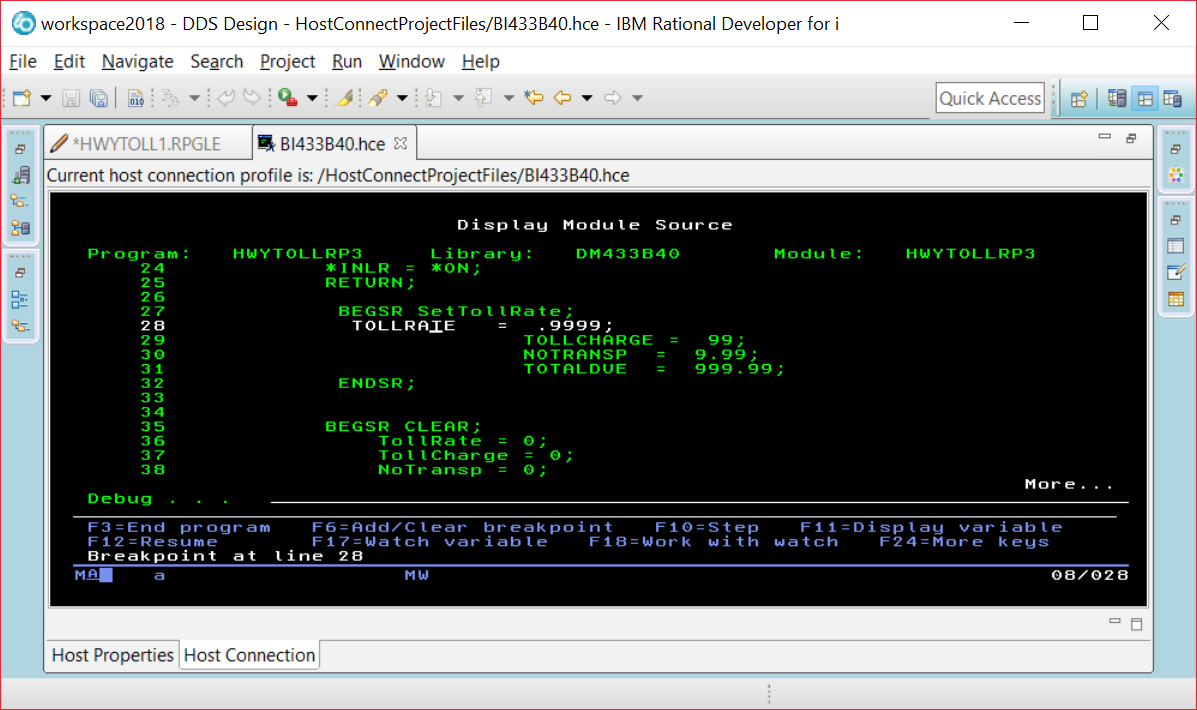
5.Run the program 🡺 CALL INCTAXRPGA

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.

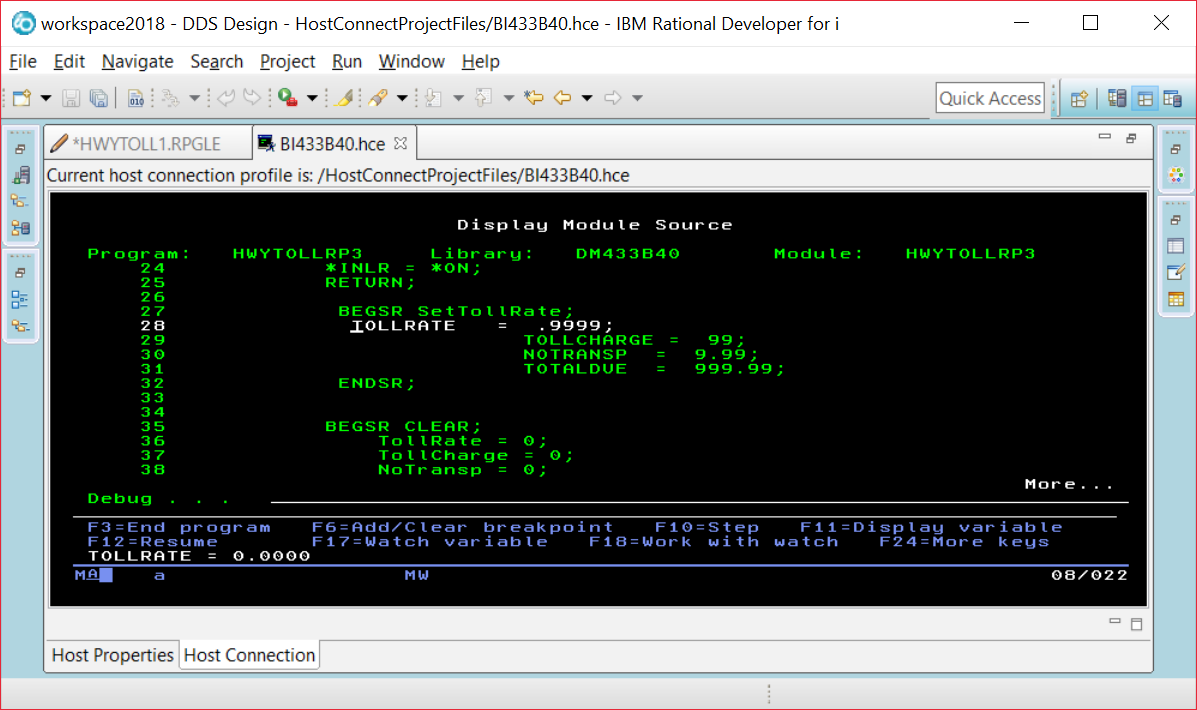


6.Press F10 to step to next line or press F12 to resume/start

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



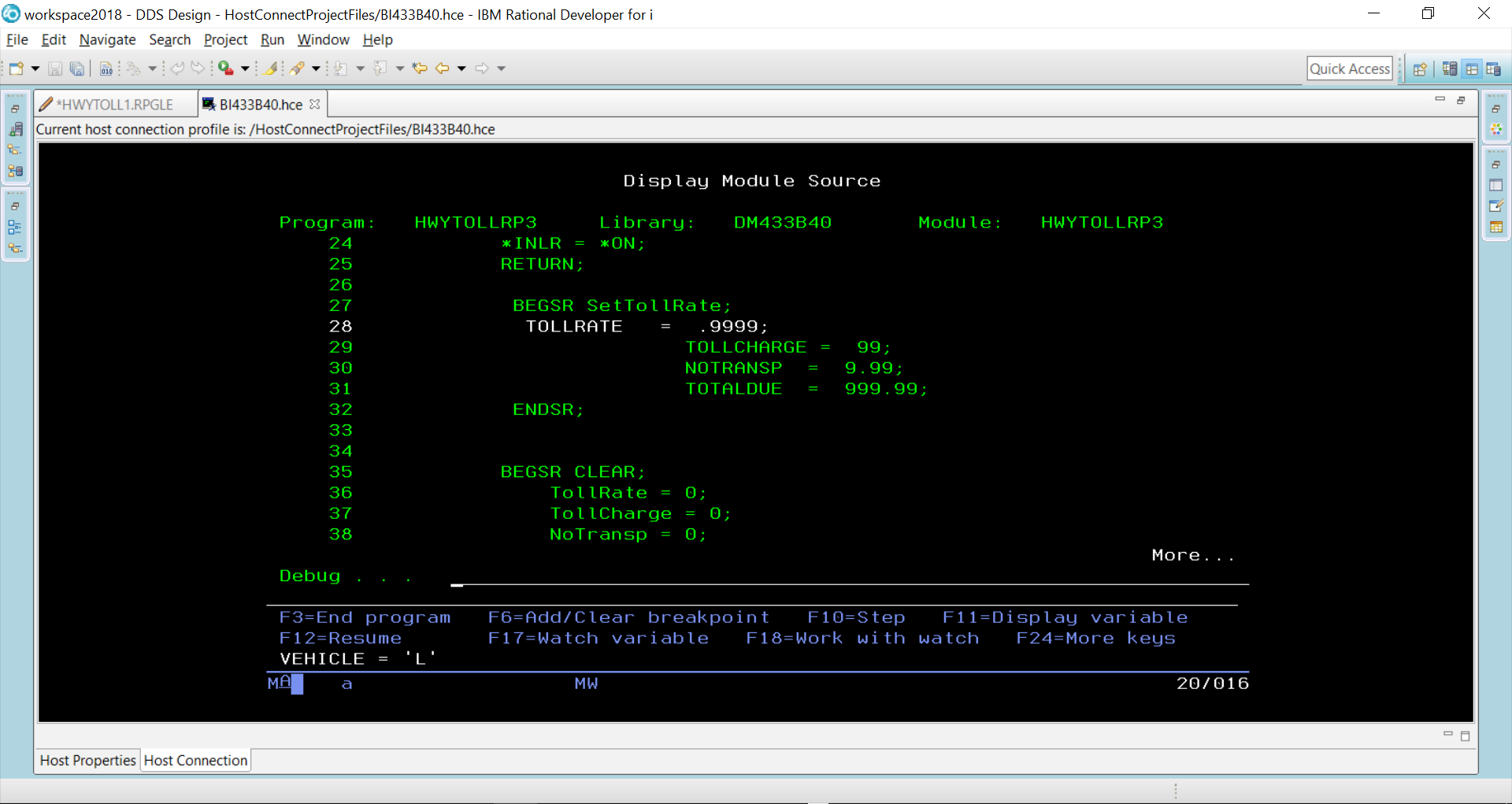
If you put your cursor on the TOLLRATE field, you will see what is currently stored in the field.



7.At the breaking point, Move cursor to a variable on screen and press F11 to check its value, OR: use command EVAL VarName at bottom line to find the variable value

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.

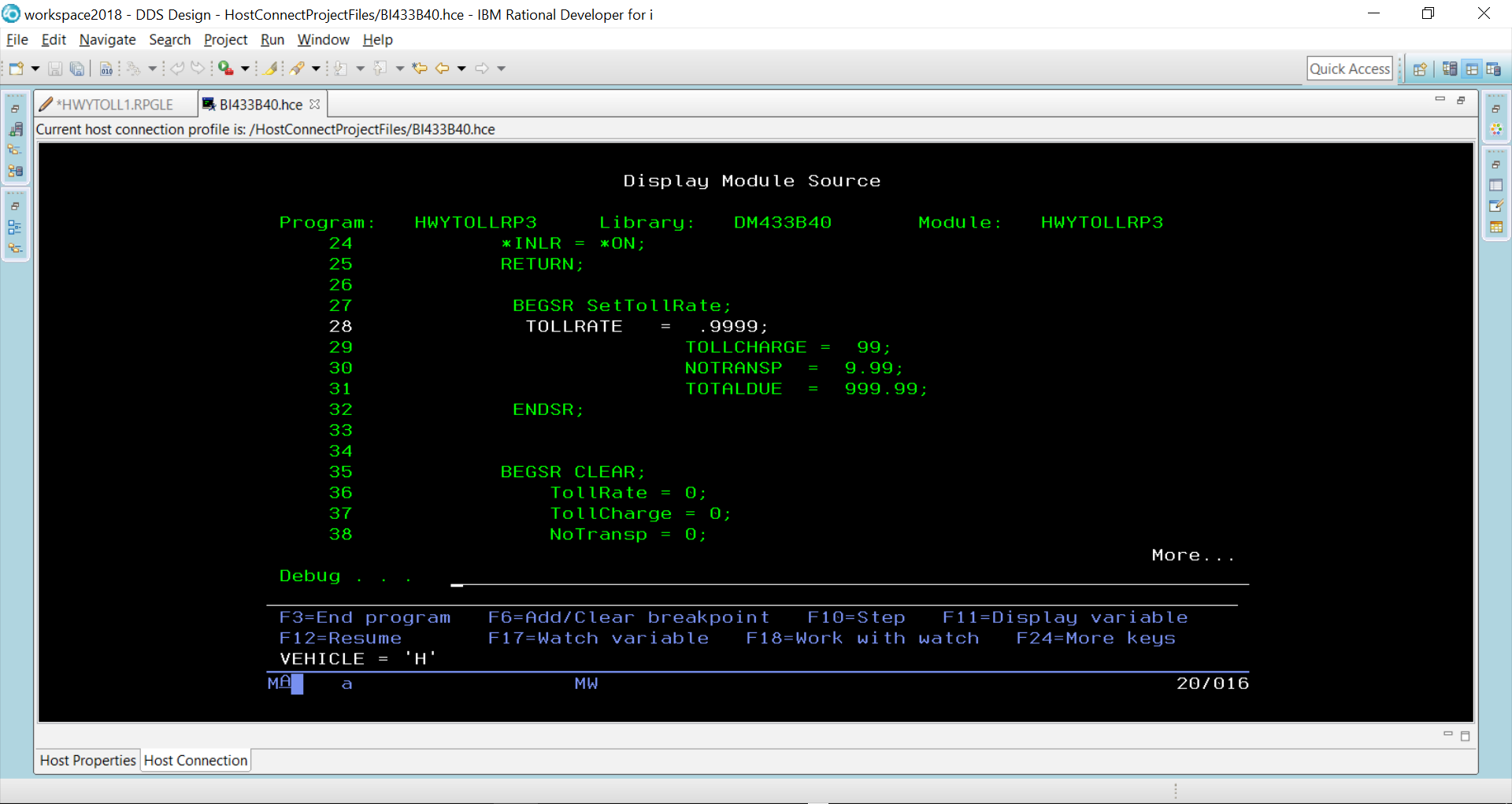
EVAL VEHICLE



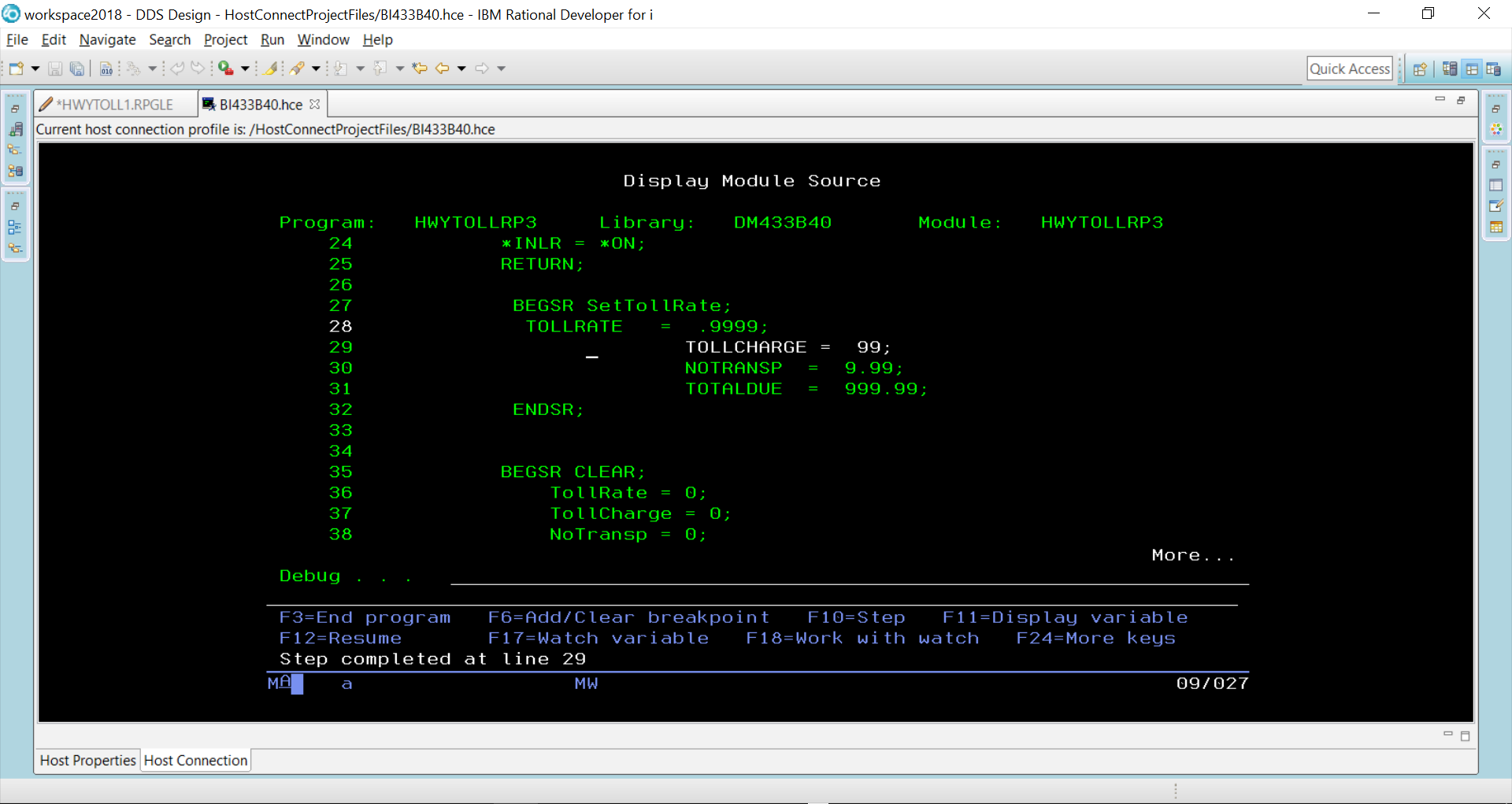
You can alter the contents of a field while your program is running.

EVAL VEHICLE = ‘H’

EVAL VEHICLE

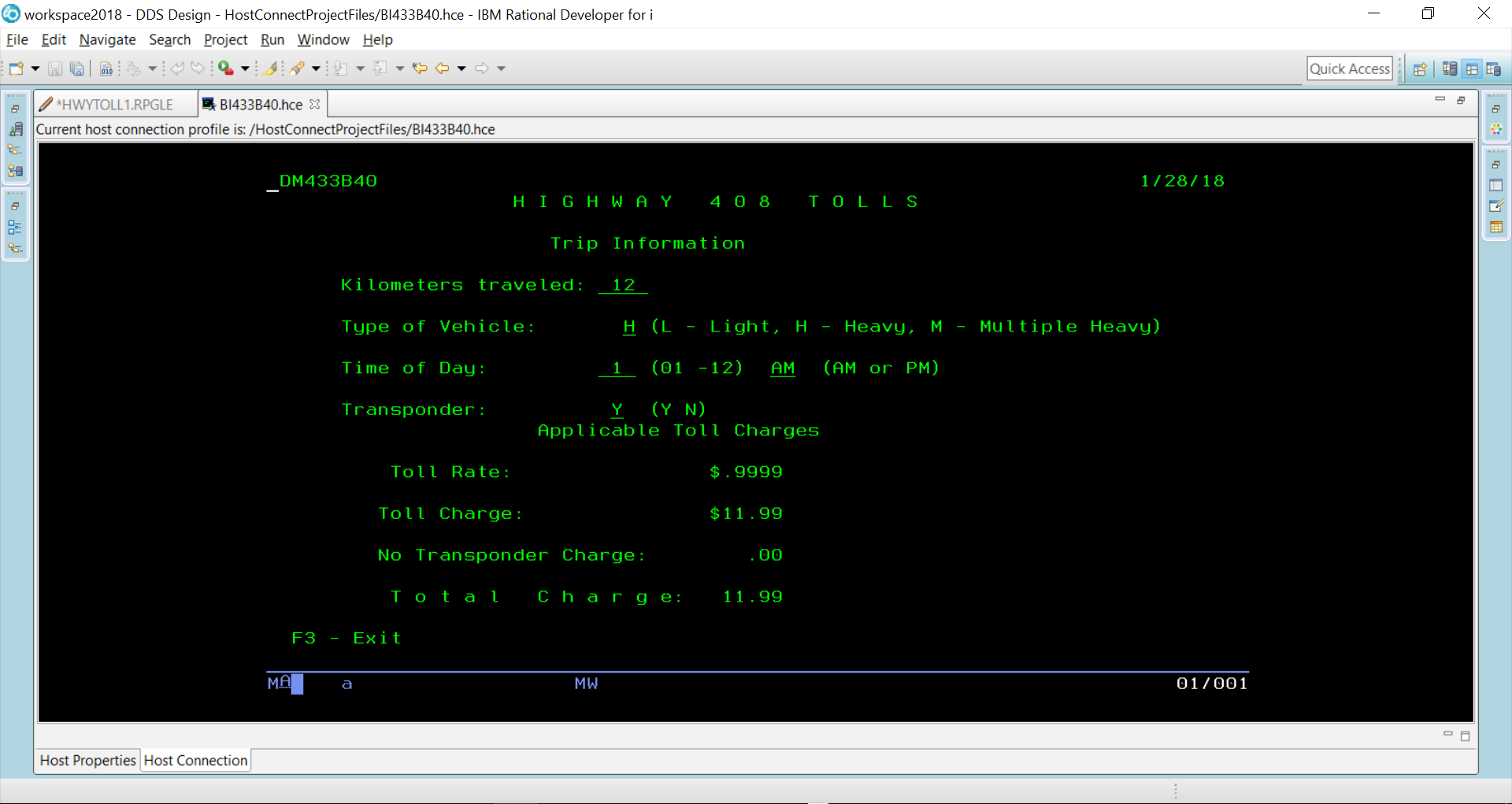


F10 is pressed to step through each line of code



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.



**End the debug mode 🡺ENDDBG**