START Line-By-Line Language Feature Testing

Niall Kelly 20461772

Adam Gray 20364103

START Web Application

Document finalised 01/02/2024

Introduction

This testing document will contain the tests carried out as part of the initial testing of our new line by line debugger developed using Java with ANTLR4 for Windows, as part of our START Web Application. This new version of the original START language developed in 2023 will be a key component of the web app as it will allow us to implement a full debugging system for the user on the front end of the app.

Test Type

Initial Feature Test followed by a Regression Test to fix any bugs initially present in the original implementation.

Test Strategy

The testing strategy will initially be an exploratory test to find edge cases we may have missed when creating the breakpoints in the language. We have already developed a set of test scripts that test all the main features of the language as well checking edge cases we predicted to be tricky to implement without a proper test cycle to validate if they have been covered.

If any bugs are found, once the changes have been made to facilitate the fixes needed, we will follow up with a regression test of the initially failed features to validate if they have been fixed.

Feature Identification

1. Variable Assignment
2. Addition
3. Subtraction
4. Multiplication
5. Division
6. Modulus
7. Powers
8. Not Expressions/Parenthesis
9. Comparisons
10. Print Statements
11. Arrays and Array Indexing
12. Boolean Operators
13. If Statements
14. While Loops
15. For Loops
16. Functions
17. Remove All
18. String Indexing
19. Comments
20. Line Stop Messages
21. Variables Outputted to File

Test Data Preparation

The test files used for this process can be found in the project under the following path:

`res/START-test-files/`

Each file will be named accordingly in the Test Data section for traceability of testing. The file of that name can be found under the above path.

Test Execution

Test Name: Variable Assignment

Test Description

Ensure a variable of any type can be assigned and used correctly.

Ensure the breakpoints work as expected.

Test Data

`var-assign.st`

Expected Result

All variables should be shown to be there originally assigned value on output.

There should be 9 points at which the program waits, and it should not wait on the final line.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it's breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

Actual Result

All variables were of correct value.

There was 9 stopping points in the program.

The program did not wait on the final line.

Evidence

```
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by-line> run start ../../../res/START-test-files/var-assign.st
intis10
current line: intis10
LINE: Press Enter to continue...Line: 1

write(int)
current line: write(int)
10LINE: Press Enter to continue...Line: 2

boolistrue
current line: boolistrue
LINE: Press Enter to continue...Line: 4

write(bool)
current line: write(bool)
trueLINE: Press Enter to continue...Line: 5

null_valueisnull
current line: null_valueisnull
LINE: Press Enter to continue...Line: 7

write(null_value)
current line: write(null_value)
nullLINE: Press Enter to continue...Line: 8

nameis"jon"
current line: nameis"jon"
LINE: Press Enter to continue...Line: 10

write(name)
current line: write(name)
jonLINE: Press Enter to continue...Line: 11

floatis10.5
current line: floatis10.5
LINE: Press Enter to continue...Line: 13

write(float)
current line: write(float)
10.5
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by-line>
```

```
int is 10
write(int) nl

bool is true
write (bool) nl

null_value is null
write(null_value) nl

name is "jon"
write(name) nl

float is 10.5
write(float) nl
```

Overall Result

Pass

Test Name: Addition

Test Description

Ensure assigned variables are able to be added together correctly as well as regular integers and floats within `write()` statements.

Ensure the breakpoints work as expected.

Test Data

`addition.st`

Expected Result

All additions should yield the correct total based on the original variable values provided to the addition operator.

There should be 9 points at which the program waits, and it should not wait on the final line.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it’s breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

Actual Result

All final outputted values were correct, meaning the addition feature is working.

There was 9 stopping points in the program.

The program did not wait on the final line.

Evidence

```
i is -1
j is 2
write(i + j) nl

k is -5 + 5
write(k) nl

l is i + j + -25
write(l) nl

m is 10
write( m + -100) nl

write(100 + -100)
```

```
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\sta
rt\line-by-line> run start ../../../res/START-test-files/addition.st
iis-1
current line: iis-1
LINE: Press Enter to continue...Line: 1

jis2
current line: jis2
LINE: Press Enter to continue...Line: 2

write(i+j)
current line: write(i+j)
1LINE: Press Enter to continue...Line: 3

kis-5+5
current line: kis-5+5
LINE: Press Enter to continue...Line: 5

write(k)
current line: write(k)
0LINE: Press Enter to continue...Line: 6

lisi+j+-25
current line: lisi+j+-25
LINE: Press Enter to continue...Line: 8

write(l)
current line: write(l)
-24LINE: Press Enter to continue...Line: 9

mis10
current line: mis10
LINE: Press Enter to continue...Line: 11

write(m+-100)
current line: write(m+-100)
-90LINE: Press Enter to continue...Line: 12

write(100+-100)
current line: write(100+-100)
0
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\sta
rt\line-by-line>
```

Overall Result

Pass

Test Name: Subtraction

Test Description

Ensure assigned variables are able to be subtracted from each other correctly as well as regular integers and floats within `write()` statements.

Ensure the breakpoints work as expected.

Test Data

`subtraction.st`

Expected Result

All subtractions should yield the correct total based on the original variable values provided to the subtraction operator.

There should be 9 points at which the program waits, and it should not wait on the final line.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it’s breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

Actual Result

All final outputted values were correct, meaning the subtraction feature is working.

There was 9 stopping points in the program.

The program did not wait on the final line.

Evidence

```
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by-line> run start ../../../res/START-test-files/subtraction.st
iis1
current line: iis1
LINE: Press Enter to continue...Line: 1

jis2
current line: jis2
LINE: Press Enter to continue...Line: 2

write(i-j)
current line: write(i-j)
-1LINE: Press Enter to continue...Line: 3

kis5-5
current line: kis5-5
LINE: Press Enter to continue...Line: 5

write(k)
current line: write(k)
0LINE: Press Enter to continue...Line: 6

lisi-j-25
current line: lisi-j-25
LINE: Press Enter to continue...Line: 8

write(l)
current line: write(l)
-26LINE: Press Enter to continue...Line: 9

mis10
current line: mis10
LINE: Press Enter to continue...Line: 11

write(m-100)
current line: write(m-100)
-90LINE: Press Enter to continue...Line: 12

write(100-100)
current line: write(100-100)
0
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by-line>
```

```
i is 1
j is 2
write(i - j) nl

k is 5 - 5
write(k) nl

l is i - j - 25
write(l) nl

m is 10
write( m - 100) nl

write(100 - 100)
```

Overall Result

Pass

Test Name: Multiplication

Test Description

Ensure assigned variables are able to be multiplied together correctly as well as regular integers and floats within `write()` statements.

Ensure the breakpoints work as expected.

Test Data

`multiplication.st`

Expected Result

All multiplications should yield the correct total based on the original variable values provided to the multiplication operator.

There should be 12 points at which the program waits, and it should not wait on the final line.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it’s breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

Actual Result

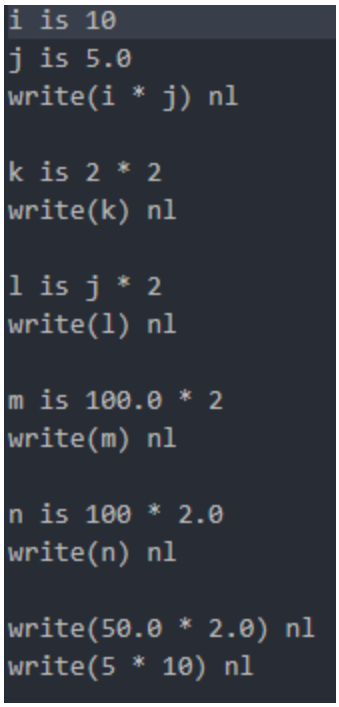
All final outputted values were correct, meaning the multiplication feature is working.

There was 12 stopping points in the program.

The program did not wait on the final line.

Evidence

```
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line> run start ../../../res/START-test-files/multiplication.st
iis10
current line: iis10
LINE: Press Enter to continue...Line: 1

jis5.0
current line: jis5.0
LINE: Press Enter to continue...Line: 2

write(i*j)
current line: write(i*j)
50.0LINE: Press Enter to continue...Line: 3

kis2*2
current line: kis2*2
LINE: Press Enter to continue...Line: 5

write(k)
current line: write(k)
4LINE: Press Enter to continue...Line: 6

lisj*2
current line: lisj*2
LINE: Press Enter to continue...Line: 8

write(l)
current line: write(l)
10.0LINE: Press Enter to continue...Line: 9

mis100.0*2
current line: mis100.0*2
LINE: Press Enter to continue...Line: 11

write(m)
current line: write(m)
200.0LINE: Press Enter to continue...Line: 12

nis100*2.0
current line: nis100*2.0
LINE: Press Enter to continue...Line: 14

write(n)
current line: write(n)
200.0LINE: Press Enter to continue...Line: 15

write(50.0*2.0)
current line: write(50.0*2.0)
100.0LINE: Press Enter to continue...Line: 17

write(5*10)
current line: write(5*10)
50
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line>
```



```
i is 10
j is 5.0
write(i * j) nl

k is 2 * 2
write(k) nl

l is j * 2
write(l) nl

m is 100.0 * 2
write(m) nl

n is 100 * 2.0
write(n) nl

write(50.0 * 2.0) nl
write(5 * 10) nl
```

Overall Result

Pass

Test Name: Division

Test Description

Ensure two assigned variables are divisible as well as regular integers and floats within `write()` statements.

Ensure the breakpoints work as expected.

Test Data

`division.st`

Expected Result

All sets of division should yield the correct total based on the original variable values provided to the division operator.

There should be 11 points at which the program waits, and it should not wait on the final line.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it's breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

Actual Result

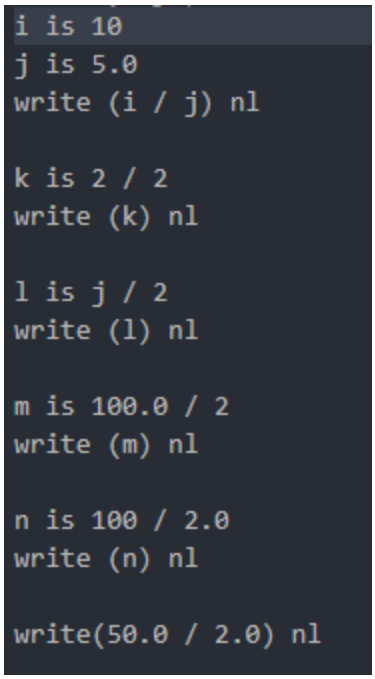
All final outputted values were correct, meaning the division feature is working.

There was 11 stopping points in the program.

The program did not wait on the final line.

Evidence

```
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line> run start ../../../res/START-test-files/division.st
iis10
current line: iis10
LINE: Press Enter to continue...Line: 1

jis5.0
current line: jis5.0
LINE: Press Enter to continue...Line: 2

write(i/j)
current line: write(i/j)
2.0LINE: Press Enter to continue...Line: 3

kis2/2
current line: kis2/2
LINE: Press Enter to continue...Line: 5

write(k)
current line: write(k)
1.0LINE: Press Enter to continue...Line: 6

lisj/2
current line: lisj/2
LINE: Press Enter to continue...Line: 8

write(l)
current line: write(l)
2.5LINE: Press Enter to continue...Line: 9

mis100.0/2
current line: mis100.0/2
LINE: Press Enter to continue...Line: 11

write(m)
current line: write(m)
50.0LINE: Press Enter to continue...Line: 12

nis100/2.0
current line: nis100/2.0
LINE: Press Enter to continue...Line: 14

write(n)
current line: write(n)
50.0LINE: Press Enter to continue...Line: 15

write(50.0/2.0)
current line: write(50.0/2.0)
25.0
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line>
```



Overall Result

Pass

Test Name: Modulus

Test Description

All modulus operations should yield the correct remainder after one variable is divided by another, as well as any integer or float within a `write()` statement.

Ensure the breakpoints work as expected.

Test Data

`modulus.st`

Expected Result

The correct remained after division is shown for each modulus operation.

There should be 8 points at which the program waits, and it should not wait on the final line.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it’s breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

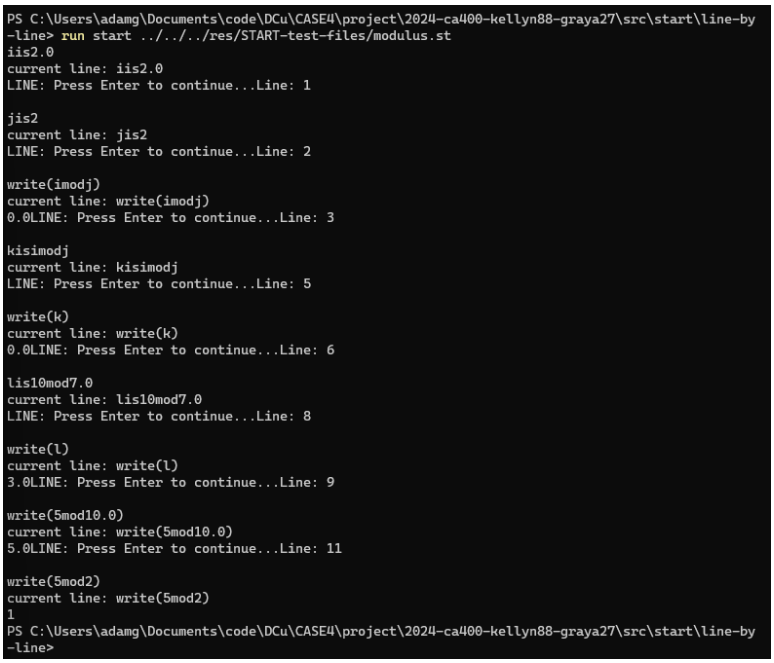
Actual Result

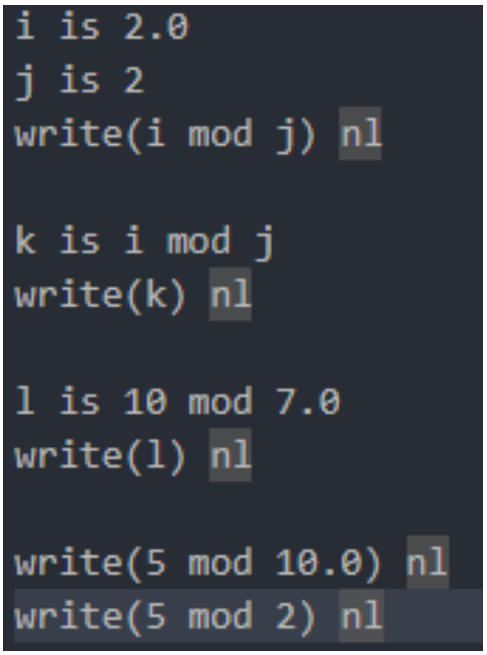
Each result is the correct remainder after the modulus operator was called.

There was 8 stopping points in the program.

The program did not wait on the final line.

Evidence





Overall Result

Pass

Test Name: Powers

Test Description

The correct result should be produced when one variable is put to the power of another, as well as two integers or floats within a `write()` statement.

Ensure the breakpoints work as expected.

Test Data

`powers.st`

Expected Result

All results are the expected value when a power operation is performed.

There should be 7 points at which the program waits, and it should not wait on the final line.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it’s breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

Actual Result

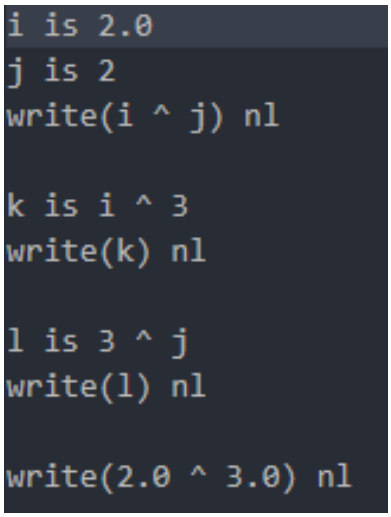
All values was correct after the power operator was invoked.

There was 7 stopping points in the program.

The program did not wait on the final line.

Evidence

```
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line> run start ../../../res/START-test-files/powers.st
iis2.0
current line: iis2.0
LINE: Press Enter to continue...Line: 1

jis2
current line: jis2
LINE: Press Enter to continue...Line: 2

write(i^j)
current line: write(i^j)
4.0LINE: Press Enter to continue...Line: 3

kisi^3
current line: kisi^3
LINE: Press Enter to continue...Line: 5

write(k)
current line: write(k)
8.0LINE: Press Enter to continue...Line: 6

lis3^j
current line: lis3^j
LINE: Press Enter to continue...Line: 8

write(l)
current line: write(l)
9.0LINE: Press Enter to continue...Line: 9

write(2.0^3.0)
current line: write(2.0^3.0)
8.0
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line>
```



Overall Result

Pass

Test Name: Not Expressions & Parenthesis

Test Description

The logical not operator should be able to swap the result of a boolean value that is calculated. As well as this we need to ensure the parenthesis take precedence in the operations.

Ensure the breakpoints work as expected.

Test Data

`not-paren.st`

Expected Result

The final values returned are the opposite of what the operation inside of the parenthesis states, hence the logical negation would work.

There should be 6 points at which the program waits, and it should not wait on the final line.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it's breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

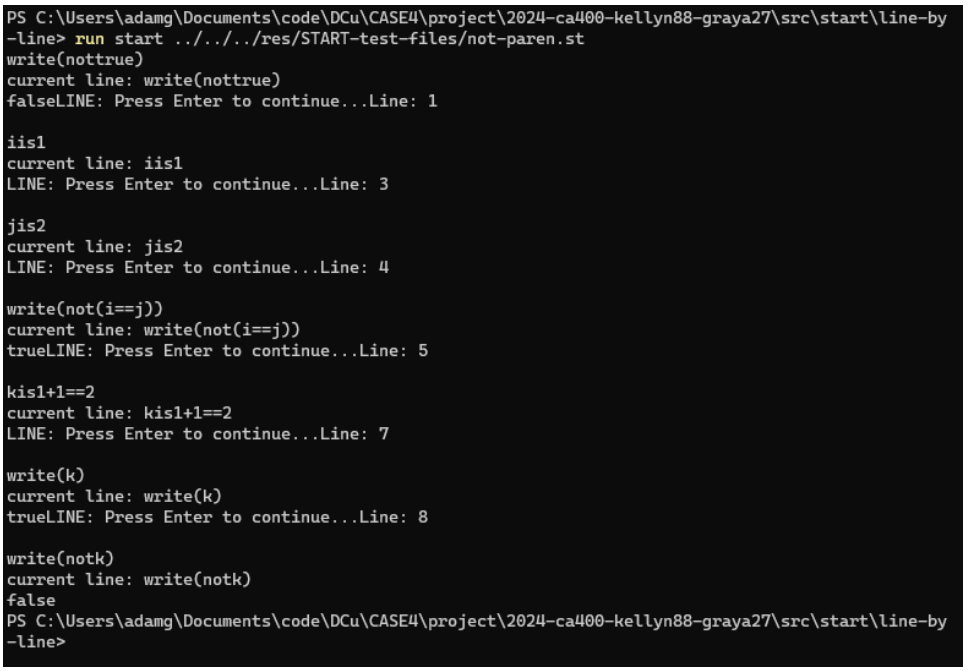
Actual Result

All final values were the logical negation of the original condition, hence the not condition is successful as well as the parenthesis taking precedence during the operation.

There was 7 stopping points in the program.

The program did not wait on the final line.

Evidence



```
write(not true) nl

i is 1
j is 2
write(not (i == j)) nl

k is 1 + 1 == 2
write(k) nl

write(not k) nl
```

Overall Result

Pass

Test Name: Comparisons

Test Description

All comparison operators should be able to check if the given condition evaluates to either true or false.

Ensure the breakpoints work as expected.

Test Data

`comp.st`

Expected Result

We expect to see all comparisons yield the correct boolean value once the comparison has taken place and has been written out.

There should be 11 points at which the program waits, and it should not wait on the final line.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it’s breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

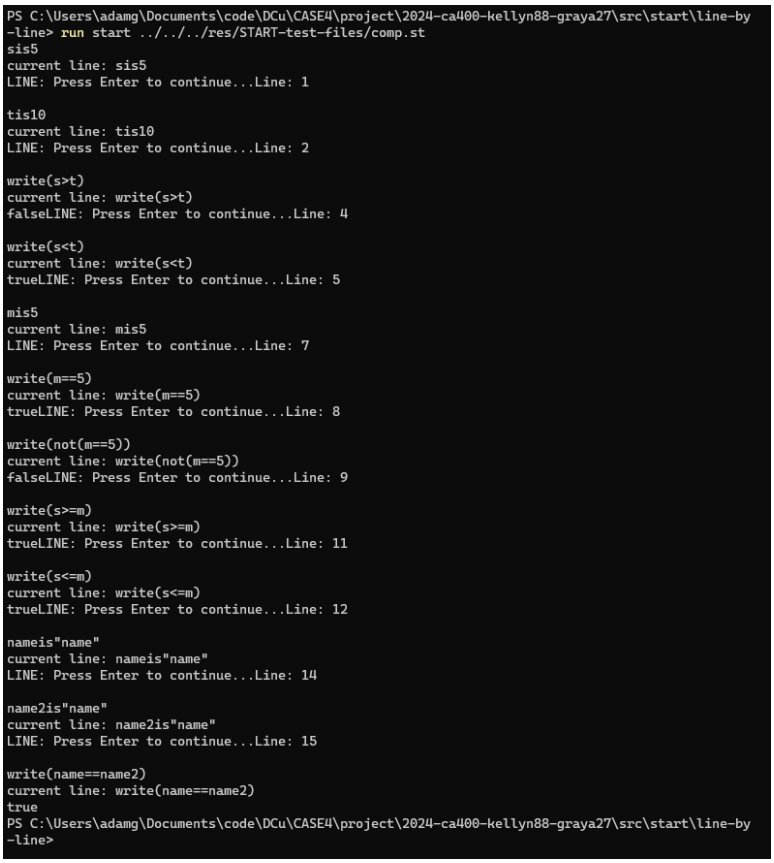
Actual Result

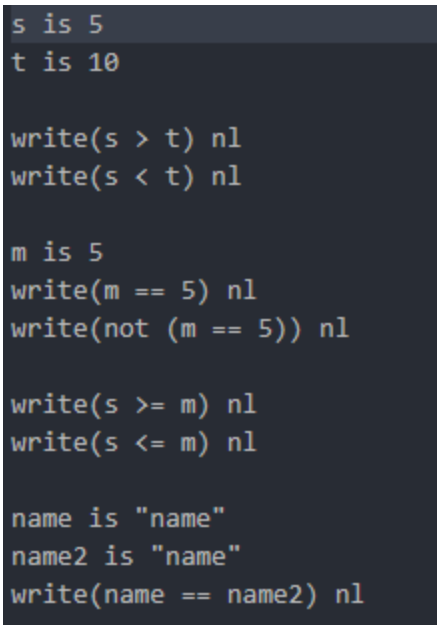
All final values were correct based on the comparisons given in the file.

There was 11 stopping points in the program.

The program did not wait on the final line.

Evidence





```
s is 5
t is 10

write(s > t) nl
write(s < t) nl

m is 5
write(m == 5) nl
write(not (m == 5)) nl

write(s >= m) nl
write(s <= m) nl

name is "name"
name2 is "name"
write(name == name2) nl
```

Overall Result

Pass

Test Name: Print Statements

Test Description

The built in `write()` function must be able to output to the console, both with and without a newline.

Ensure the breakpoints work as expected.

Test Data

`print.st`

Expected Result

We expect to see the output in the console and for it to be correct based on the test file.

There should be 3 points at which the program waits, and it should not wait on the final line.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it’s breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

Actual Result

All output is correct including the use of the `nl`.

There was 11 stopping points in the program.

The program did not wait on the final line.

Evidence

```
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line> run start ../../../res/START-test-files/print.st
write("my name is ")
current line: write("my name is ")
my name is LINE: Press Enter to continue...Line: 1

nameis"jon"
current line: nameis"jon"
LINE: Press Enter to continue...Line: 2

write(name)
current line: write(name)
jonLINE: Press Enter to continue...Line: 3

write("my name is not "+"dave")
current line: write("my name is not "+"dave")
my name is not dave
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line>
```

```
write("my name is ")
name is "jon"
write(name) nl

write("my name is not " + "dave")
```

Overall Result

Pass

Test Name: Arrays & Array Indexing

Test Description

Arrays must be able to support the following features:

- Printing
- Multi type content
- Indexing the array to print
- Changing values at an index
- Appending to the end of an array
- Concatenation of 2 arrays
- Getting the length of the array
- Removing from an array

Ensure the breakpoints work as expected.

Test Data

`arr-arrIndex.st`

`arr-append.st`

`arr-concat.st`

`arr-length.st`

`arr-remove.st`

Expected Result

- An array is printed
- Arrays contain multi content successfully
- Array index retrieved and changed
- 2 arrays are concatenated
- The length is found of an array
- Items are removed from an array

The correct number of breakpoints are present for each of the test files.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it’s breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

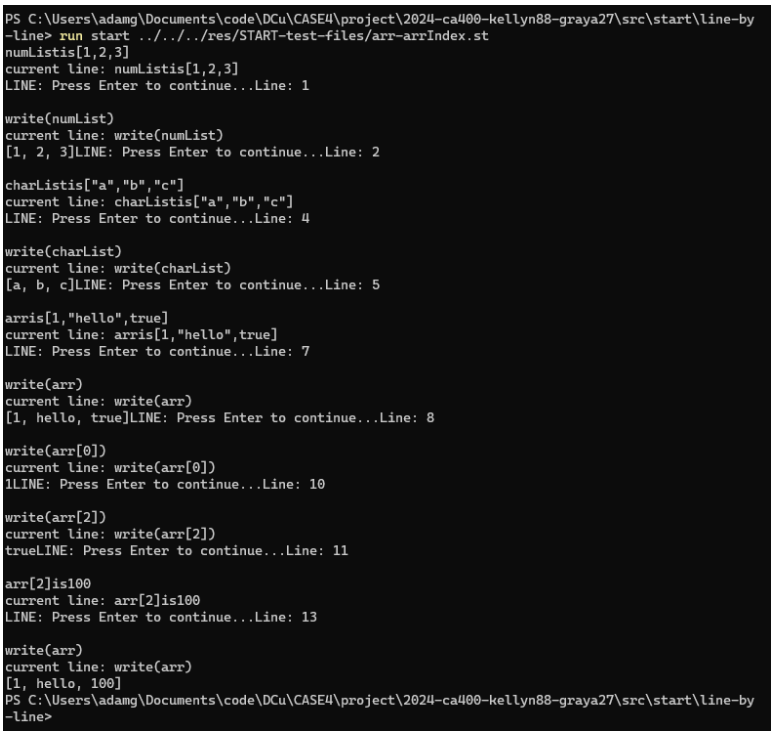
Actual Result

All cases were passed, and all expected results were found to be correct, meaning the arrays are fully functional.

The correct number of breakpoints where present per file.

The program did not wait on the final line of each file.

Evidence



```
numList is [1,2,3]
write(numList) nl

charList is ["a","b","c"]
write(charList) nl

arr is [1, "hello", true]
write(arr) nl

write(arr[0]) nl
write(arr[2]) nl

arr[2] is 100
write(arr) nl
```

```
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line> run start ../../../res/START-test-files/arr-append.st
ais[1,2,3]
current line: ais[1,2,3]
LINE: Press Enter to continue...Line: 1

write(a)
current line: write(a)
[1, 2, 3]LINE: Press Enter to continue...Line: 2

aisaadd100
current line: aisaadd100
LINE: Press Enter to continue...Line: 3

write(a)
current line: write(a)
[1, 2, 3, 100]
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line>
```

```
a is [1,2,3]
write(a) nl
a is a add 100
write(a) nl
```

```
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line> run start ../../../res/START-test-files/arr-concat.st
ais[1,2,3]
current line: ais[1,2,3]
LINE: Press Enter to continue...Line: 1

bis[4,5,6]
current line: bis[4,5,6]
LINE: Press Enter to continue...Line: 2

cisaconcatb
current line: cisaconcatb
LINE: Press Enter to continue...Line: 4

write(c)
current line: write(c)
[1, 2, 3, 4, 5, 6]
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line>
```

```
a is [1,2,3]
b is [4,5,6]

c is a concat b
write(c) nl
```

```
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line> run start ../../../res/START-test-files/arr-length.st
ais[1,2,3]
current line: ais[1,2,3]
LINE: Press Enter to continue...Line: 1

write(a)
current line: write(a)
[1, 2, 3]LINE: Press Enter to continue...Line: 2

write(length ofa)
current line: write(length ofa)
3LINE: Press Enter to continue...Line: 3

bislength ofa
current line: bislength ofa
LINE: Press Enter to continue...Line: 5

write(b)
current line: write(b)
3
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line>
```

```
a is [1,2,3]
write(a) nl
write(length of a) nl

b is length of a
write(b) nl
```

```
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line> run start ../../../res/START-test-files/arr-remove.st
ais[1,2,3]
current line: ais[1,2,3]
LINE: Press Enter to continue...Line: 1

remove2froma
current line: remove2froma
LINE: Press Enter to continue...Line: 2

write(a)
current line: write(a)
[1, 3]LINE: Press Enter to continue...Line: 3

stringArris["hello","world"]
current line: stringArris["hello","world"]
LINE: Press Enter to continue...Line: 5

remove"hello"fromstringArr
current line: remove"hello"fromstringArr
LINE: Press Enter to continue...Line: 6

write(stringArr)
current line: write(stringArr)
[world]
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line>
```

```
a is [1,2,3]
remove 2 from a
write(a) nl

stringArr is ["hello", "world"]
remove "hello" from stringArr
write(stringArr) nl
```

Overall Result

Pass

Test Name: If Statements & Boolean Operators

Test Description

If statements should be able to correctly follow the right path based on the comparison of each if, otherwise if and otherwise statement, and such we should see the correct messages.

Within these comparisons the Boolean Operators should be working properly meaning AND and OR should work as expected, with either both or one of the statements needing to be true, respectively.

Ensure the breakpoints work as expected.

Test Data

`if-statement.st`

Expected Result

We should see the messages:

```
15 is a multiple of 3 and 5
n is 15 and m is 11
a is greater than 1
```

We should stop at the conditions of the if statement, as well as not stop when the last line is within an if statement.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it's breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

Actual Result

We see the expected messages as the if statement followed the correct path and the Boolean Operators worked as expected.

However, we did not stop correctly on the conditions of the if statements.

Also stopped when the last line was within an if statement.

Evidence

```
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line> run start ../../../res/START-test-files/if-statement.st
nis15
LINE: Press Enter to continue...
ifnmod3equals0{ifnmod5equals0{write(n)write(" is a multiple of 3 and 5")nl}otherwise{write(n)wri
te(" is a multiple of 3")nl}}otherwise{write(n)write(" is not a multiple of 3")nl}
15BLOCK: Press Enter to continue...

 is a multiple of 3 and 5BLOCK: Press Enter to continue...


LINE: Press Enter to continue...
mis11
LINE: Press Enter to continue...
if(nequals15)and(mequals20){write("n is 15 and m is 20")nl}otherwiseif(nequals15)and(mequals11){
write("n is 15 and m is 11")nl}otherwise{write("neither are correct")nl}
n is 15 and m is 11BLOCK: Press Enter to continue...


LINE: Press Enter to continue...
ais2
ifagreater than1oraequals0{write("a is greater than 1")nl}
a is greater than 1BLOCK: Press Enter to continue...


PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line>
```

```
n is 15       You, 5 days ago • just need to ad

if n mod 3 equals 0 {
    if n mod 5 equals 0 {
        write(n)
        write(" is a multiple of 3 and 5") nl
    }
    otherwise {
        write(n)
        write(" is a multiple of 3") nl
    }
}
otherwise{
    write(n)
    write(" is not a multiple of 3") nl
}

m is 11
if (n equals 15) and (m equals 20) {
    write("n is 15 and m is 20") nl
}
otherwise if (n equals 15) and (m equals 11) {
    write("n is 15 and m is 11") nl
}
otherwise{
    write("neither are correct") nl
}

a is 2
if a greater than 1 or a equals 0{
    write("a is greater than 1") nl
}
```

Overall Result

Fail

Test Name: While Loops

Test Description

The loop should be able to run given a certain provided condition is true, and stop once the condition is broken.

Ensure the breakpoints work as expected.

Test Data

`while-loop.st`

Expected Result

We should see the current state of `n` while it is less than 3, and each time we should see the state of m, as well as seeing `m` reset for each iteration.

We should stop on every line of the while loop, as well as the condition.

All other non while breakpoints should be unaffected.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it's breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

Actual Result

The loop iterated 3 times and stopped once `n` became 3, as expected, and each time the value of `m` was reset successfully.

However, we did not stop on the condition, we needed to look at our logic and implement this.

Evidence

```
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by-line> run start ../../../res/START-test-files/while-loop.
st
nis0
loop whilen<3{write("current n is ")write(n)nlmis0loop whilem<3{write("current m is ")write(m)nlmismadd1}nisnadd1}
current n is BLOCK: Press Enter to continue...

0BLOCK: Press Enter to continue...


BLOCK: Press Enter to continue...

current m is BLOCK: Press Enter to continue...

0BLOCK: Press Enter to continue...


current m is BLOCK: Press Enter to continue...

1BLOCK: Press Enter to continue...


current m is BLOCK: Press Enter to continue...

2BLOCK: Press Enter to continue...


BLOCK: Press Enter to continue...

current n is BLOCK: Press Enter to continue...

1BLOCK: Press Enter to continue...


BLOCK: Press Enter to continue...

current m is BLOCK: Press Enter to continue...

0BLOCK: Press Enter to continue...


current m is BLOCK: Press Enter to continue...

1BLOCK: Press Enter to continue...


current m is BLOCK: Press Enter to continue...

2BLOCK: Press Enter to continue...


BLOCK: Press Enter to continue...

current n is BLOCK: Press Enter to continue...

2BLOCK: Press Enter to continue...


BLOCK: Press Enter to continue...

current m is BLOCK: Press Enter to continue...

0BLOCK: Press Enter to continue...


current m is BLOCK: Press Enter to continue...

1BLOCK: Press Enter to continue...


current m is BLOCK: Press Enter to continue...

2BLOCK: Press Enter to continue...


BLOCK: Press Enter to continue...
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by-line>
```

```
n is 0        You, 5 days ago • ju
loop while n < 3{
    write("current n is ")
    write(n) nl
    m is 0
    loop while m < 3{
        write("current m is ")
        write(m) nl
        m is m add 1
    }
    n is n add 1
}
```

Overall Result

Fail

Test Name: For Loops

Test Description

The loop should be able to move through each element of the array, as well as the sub elements of a matrix in this given case.

Ensure the breakpoints work as expected.

Test Data

`for-loop.st`

Expected Result

We expect to see the numbers 1 to 6 sequentially as the embedded loops work to extract the elements of the matrix.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it's breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

Actual Result

We see the numbers of the matrix printed out in order correctly.

However, we did not expect for the program to stop on the for loop lines. This added extra input for no reason and will need to be fixed.

Evidence

```
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line> run start ../../../res/START-test-files/for-loop.st
matrixis[[1,2],[3,4],[5,6]]
LINE: Press Enter to continue...
ais5
loop for eachrowinmatrix{loop for eachcolumninrow{write(column)nl}}
FOR LOOP: Press Enter to continue...

FOR LOOP: Press Enter to continue...

1
FOR LOOP: Press Enter to continue...

2
FOR LOOP: Press Enter to continue...

FOR LOOP: Press Enter to continue...

3
FOR LOOP: Press Enter to continue...

4
FOR LOOP: Press Enter to continue...

FOR LOOP: Press Enter to continue...

5
FOR LOOP: Press Enter to continue...

6
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line>
```

```
matrix is [[1,2], [3,4], [5,6]]
a is 5
loop for each row in matrix{
    loop for each column in row{
        write(column) nl
    }
}       You, 5 days ago • just nee
```

Overall Result

Fail

Test Name: Functions

Test Description

We should be able to define a function, as well as return from one, call a function within a function, and print from a function.

Ensure the breakpoints work as expected.

Test Data

`functions.st`

Expected Result

We expect to see `Hello World` printed when `main` is called correctly, and then see 7 when the `addup` function is called within main and the values passed are

returned added together.

There should be no points at which the program stops.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it's breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

Actual Result

The `main` function as well as the `addup` function are called correctly and we see both sets of output.

There was no stopping points in the program.

Evidence

```
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line> run start ../../../res/START-test-files/functions.st
main()
current line: main()
Hello World
7
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line>
```

```
function main(){
    write("Hello World") nl
    a is 5
    b is 2
    write(addup(a,b)) nl
}

function addup(a,b){
    return a add b
}

main()
```

Overall Result

Pass

Test Name: Remove All

Test Description

We should be able to remove all instances of a given number from a given array using a built in operation. No other indexes should be removed.

Ensure the breakpoints work as expected.

Test Data

`remove-all.st`

Expected Result

We expect to see the original array within any 1's within the array outputted.

There should be 2 points at which the program waits, and it should not wait on the final line.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it's breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

Actual Result

The final result contained only the numbers that were not 1, showing it worked as intended.

There was 2 stopping points in the program.

The program did not wait on the final line.

Evidence

```
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line> run start ../../../res/START-test-files/remove-all.st
arris[1,3,1,5,1,7,1,9]
current line: arris[1,3,1,5,1,7,1,9]
LINE: Press Enter to continue...Line: 1

removeall1fromarr
current line: removeall1fromarr
LINE: Press Enter to continue...Line: 2

write(arr)
current line: write(arr)
[3, 5, 7, 9]
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line>
```

```
arr is [1,3,1,5,1,7,1,9]
remove all 1 from arr
write(arr) nl
```

Overall Result

Pass

Test Name: String Indexing

Test Description

We expect that a character from a string can be stored in a variable by using its index.

Ensure the breakpoints work as expected.

Test Data

`string-dex.st`

Expected Result

We expect to see the correct index value printed out to the terminal.

There should be 7 points at which the program waits, and it should not wait on the final line.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it’s breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

Actual Result

All outputted values were of correct value.

There was 7 stopping points in the program.

The program did not wait on the final line.

Evidence

```
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line> run start ../../../res/START-test-files/string-dex.st
str1is"hello"
current line: str1is"hello"
LINE: Press Enter to continue...Line: 1

bisstr1[0]
current line: bisstr1[0]
LINE: Press Enter to continue...Line: 2

cisstr1[3]
current line: cisstr1[3]
LINE: Press Enter to continue...Line: 3

write("char at positionn 0 is: ",b)
current line: write("char at positionn 0 is: ",b)
char at positionn 0 is:  hLINE: Press Enter to continue...Line: 4

write("char at positionn 3 is: ",c)
current line: write("char at positionn 3 is: ",c)
char at positionn 3 is:  lLINE: Press Enter to continue...Line: 5

str2is"char char"
current line: str2is"char char"
LINE: Press Enter to continue...Line: 7

spaceisstr2[4]
current line: spaceisstr2[4]
LINE: Press Enter to continue...Line: 8

write("space is: ",space,"before here")
current line: write("space is: ",space,"before here")
space is:    before here
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line>
```

```
str1 is "hello"
b is str1[0]
c is str1[3]
write("char at positionn 0 is: ", b) nl
write("char at positionn 3 is: ", c) nl

str2 is "char char"
space is str2[4]
write("space is: ", space, "before here") nl
```

Overall Result

Pass

Test Name: Comments

Test Description

Comments should be ignored completely by the program.

Ensure the breakpoints work as expected.

Test Data

`comments.st`

Expected Result

- All single line comments should be ignored
- Multi line comments should have every line of the comment ignored
- In line comments should be ignored
- All other lines should work as expected.

There should be 1 point at which the program waits, and it should not wait on the final line.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it’s breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

Actual Result

The correct value for n was outputted showing only non comment lines were seen as valid code.

There was 1 stopping point in the program.

The program did not wait on the final line.

Evidence

```
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line> run start ../../../res/START-test-files/comments.st
nis5
current line: nis5
LINE: Press Enter to continue...Line: 8

write(n)
current line: write(n)
5
PS C:\Users\adamg\Documents\code\DCu\CASE4\project\2024-ca400-kellyn88-graya27\src\start\line-by
-line>
```

```
// start of program //

// another in line comment //

//m is 20//
//write(m) nl //

n is 5 //in line comment //

//
a
multi
line
comment
//

write (n) nl

// end of program //
```

Overall Result

Pass

Identified Defects

1. If statements did not stop on the condition being checked.
    - This issue was fixed with the following code:

```
Object val = visit(ctx.expression());
//wait for user input after condition is checked
System.out.println("AT CONDITION: Press Enter to continue...'
System.out.println("Line: " + ctx.start.getLine());
scanner.nextLine();
```

- We needed to simply add in a stop at the condition.
- Fixed by Adam Gray

2. If statements did not continue when the last valid line was inside of an if statement.
    - The issue was fixed with the following code:

```
if (ctx.line(i + 1).getText().equals("nl") && i == ctx.line(
    //we need to check if the next line in program exists
    //get parent
    var parent2 = ctx.getParent();
    //loop back until we find program
    while (parent2 != null){
        System.out.println("parent is: " + parent2.getClass(
        if(parent2 instanceof startParser.ProgramContext){
            //print child 0 of program
            System.out.println("child 0 of program is: " + pa
            //get the line that comes after that
            var line = (parent2.getChildCount() - 2);
            System.out.println("line is: " + line);
            //start at this node, while not null update to pa
            var parent3 = ctx.getParent();
            int numOfIfs = 0;
            while (parent3 != null){
                System.out.println("parent3 is: " + parent3.g
                if (parent3 instanceof startParser.If_stateme
                    numOfIfs++;
                }
                parent3 = parent3.getParent();
```

```
            }
            System.out.println("num of ifs is: " + numOfIfs)
            var parent4 = ctx.getParent();
            while (parent4 != null){
                System.out.println("parent4 is: " + parent4.g
                if (parent4 instanceof startParser.If_stateme
                    numOfIfs--;
                    if (numOfIfs == 0){
                        //get the text of this line, save it
                        String text = parent4.getText();
                        System.out.println("this text is: " +
                        String lineText = parent2.getChild(li
                        System.out.println("line text is: " +
                        //if the text is the same as text on
                        if (text.equals(lineText)){
                            break;
                        }
                        else{
                            System.out.println("BLOCK: Press
                            System.out.println("Line: " + ctx
                            scanner.nextLine();
                        }
                    }
                }
                parent4 = parent4.getParent();
            }

            break;
        }
        else{
            parent2 = parent2.getParent();
        }
    }
}
```

- We use the parent of the current node to see if we are at the original program state, if not keep updating the parent until we are.
- Once at program context, we find how many if statements we are within, to handle embedded if's.
- We then move outside of the if statements, and check if the line of the if is the same as the final line of the program.
- If it is, we don't have to wait, else, we wait for user input.
- Fixed by Adam Gray

3. While loops did not stop on the condition.
    - This issue was fixed with the following code:

```
Object val = visit(ctx.expression());
//wait for user input after condition is checked
System.out.println("AT CONDITION: Press Enter to continue...'
System.out.println("Line: " + ctx.start.getLine());
scanner.nextLine();
```

    - We needed to simply add in a stop at the condition.
    - Fixed by Adam Gray
4. For loops were stopping on the for loop line, they should not, adds extra inputs needed.
    - The issue was fixed with the following code

```
boolean isForLoop = false;
  for (int j = 0; j < ctx.line(i).parent.getChildCount(); j+
      if (ctx.line(i).parent.getChild(j).getText().equals("f
          isForLoop = true;
          break;
      }
  }
  //if the current line is a for loop, continue
  if (!isForLoop){
```

```
        //rest global variables
        loopedOver = null;
        currentCharInArray = null;
        lastNonSpecialChar = null;
    }
```

- By checking if we are in a for loop, it allowed us to reset the global variables we used later to determine if we need to stop or not, hence not stopping.

Regression Testing

Test Name: If Statements & Boolean Operators

Test Description

If statements should be able to correctly follow the right path based on the comparison of each if, otherwise if and otherwise statement, and such we should see the correct messages.

Within these comparisons the Boolean Operators should be working properly meaning AND and OR should work as expected, with either both or one of the statements needing to be true, respectively.

Ensure the breakpoints work as expected.

Test Data

`if-statement.st`

Expected Result

We should see the messages:

```
15 is a multiple of 3 and 5
n is 15 and m is 11
a is greater than 1
```

We should stop at the conditions of the if statement, as well as not stop when the last line is within an if statement.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it’s breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

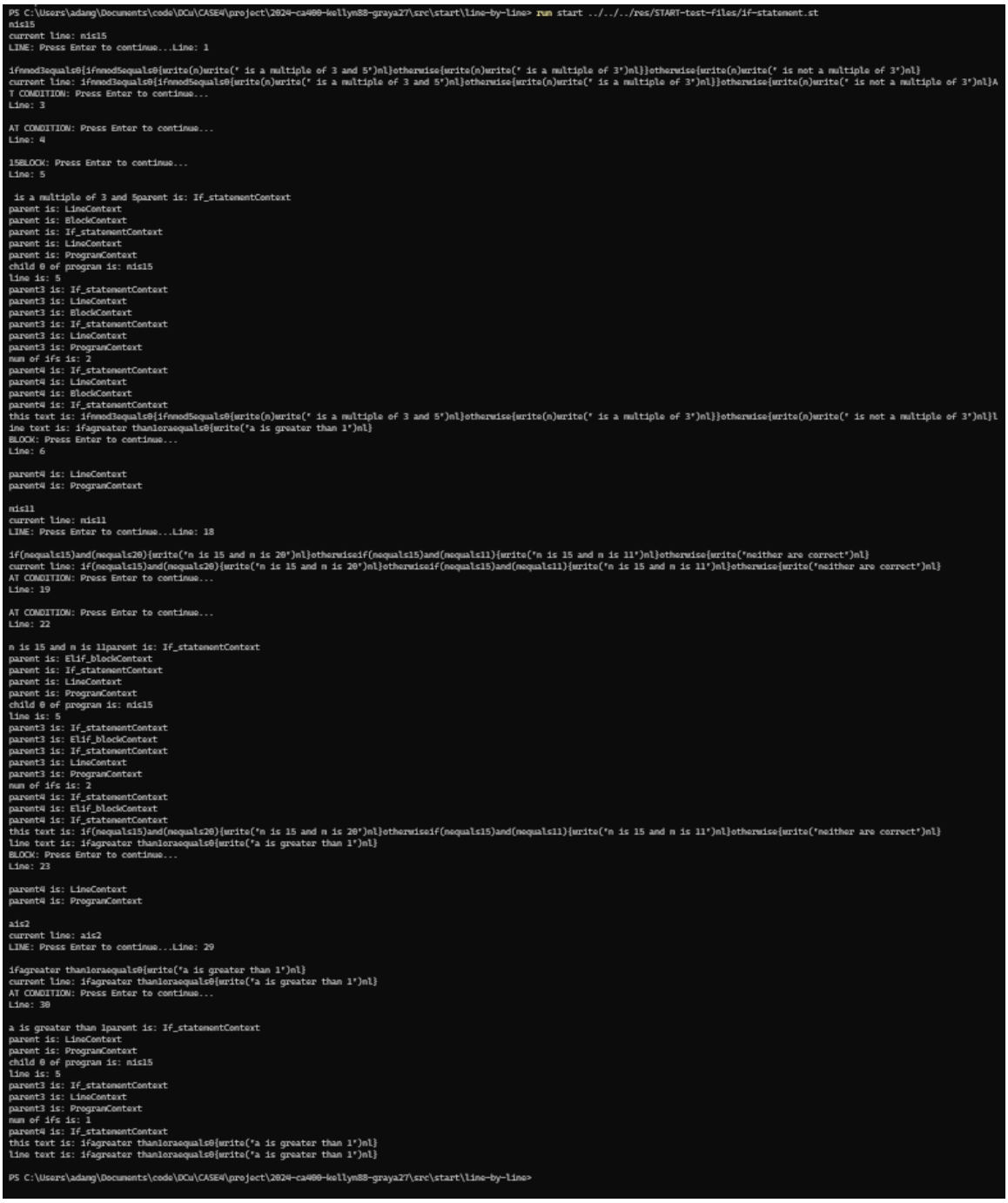
Actual Result

We see the expected messages as the if statement followed the correct path and the Boolean Operators worked as expected.

The program stopped at the conditions being checked in the if statements.

The last line was inside the if statement, and the program did not wait, as expected, so the bug is fixed.

Evidence


PS C:\Users\adang\Documents\code\DCu\CASE4\project\2024-ca400-kellym88-graya27\src\start\line-by-line> run start ../../../res/START-test-files/if-statement.st
mis15
current line: mis15
LINE: Press Enter to continue...Line: 1
ifnmod3equals0{ifnmod5equals0{write(n)write(" is a multiple of 3 and 5")nl}otherwise{write(n)write(" is a multiple of 3")nl}}otherwise{write(n)write(" is not a multiple of 3")nl}
current line: ifnmod3equals0{ifnmod5equals0{write(n)write(" is a multiple of 3 and 5")nl}otherwise{write(n)write(" is a multiple of 3")nl}}otherwise{write(n)write(" is not a multiple of 3")nl}A
T CONDITION: Press Enter to continue...
Line: 3
AT CONDITION: Press Enter to continue...
Line: 4
15BLOCK: Press Enter to continue...
Line: 5
is a multiple of 3 and 5parent is: If_statementContext
parent is: LineContext
parent is: BlockContext
parent is: If_statementContext
parent is: LineContext
parent is: ProgramContext
child 0 of program is: mis15
line is: 5
parent3 is: If_statementContext
parent3 is: LineContext
parent3 is: BlockContext
parent3 is: If_statementContext
parent3 is: LineContext
parent3 is: ProgramContext
num of ifs is: 2
parent4 is: If_statementContext
parent4 is: LineContext
parent4 is: BlockContext
parent4 is: If_statementContext
this text is: ifnmod3equals0{ifnmod5equals0{write(n)write(" is a multiple of 3 and 5")nl}otherwise{write(n)write(" is a multiple of 3")nl}}otherwise{write(n)write(" is not a multiple of 3")nl}l
ine text is: ifagreater than1oraequals0{write("a is greater than 1")nl}
BLOCK: Press Enter to continue...
Line: 6
parent4 is: LineContext
parent4 is: ProgramContext
mis11
current line: mis11
LINE: Press Enter to continue...Line: 18
if(nequals15)and(nequals20){write("n is 15 and n is 20")nl}otherwiseif(nequals15)and(nequals11){write("n is 15 and n is 11")nl}otherwise{write("neither are correct")nl}
current line: if(nequals15)and(nequals20){write("n is 15 and n is 20")nl}otherwiseif(nequals15)and(nequals11){write("n is 15 and n is 11")nl}otherwise{write("neither are correct")nl}
AT CONDITION: Press Enter to continue...
Line: 19
AT CONDITION: Press Enter to continue...
Line: 22
n is 15 and n is 11parent is: If_statementContext
parent is: Elif_blockContext
parent is: If_statementContext
parent is: LineContext
parent is: ProgramContext
child 0 of program is: mis15
line is: 5
parent3 is: If_statementContext
parent3 is: Elif_blockContext
parent3 is: If_statementContext
parent3 is: LineContext
parent3 is: ProgramContext
num of ifs is: 2
parent4 is: If_statementContext
parent4 is: Elif_blockContext
parent4 is: If_statementContext
this text is: if(nequals15)and(nequals20){write("n is 15 and n is 20")nl}otherwiseif(nequals15)and(nequals11){write("n is 15 and n is 11")nl}otherwise{write("neither are correct")nl}
line text is: ifagreater than1oraequals0{write("a is greater than 1")nl}
BLOCK: Press Enter to continue...
Line: 23
parent4 is: LineContext
parent4 is: ProgramContext
ais2
current line: ais2
LINE: Press Enter to continue...Line: 29
ifagreater than1oraequals0{write("a is greater than 1")nl}
current line: ifagreater than1oraequals0{write("a is greater than 1")nl}
AT CONDITION: Press Enter to continue...
Line: 38
a is greater than 1parent is: If_statementContext
parent is: LineContext
parent is: ProgramContext
child 0 of program is: mis15
line is: 5
parent3 is: If_statementContext
parent3 is: LineContext
parent3 is: ProgramContext
num of ifs is: 1
parent4 is: If_statementContext
this text is: ifagreater than1oraequals0{write("a is greater than 1")nl}
line text is: ifagreater than1oraequals0{write("a is greater than 1")nl}
PS C:\Users\adang\Documents\code\DCu\CASE4\project\2024-ca400-kellym88-graya27\src\start\line-by-line>

```
n is 15       You, 5 days ago • just need to ad

if n mod 3 equals 0 {
    if n mod 5 equals 0 {
        write(n)
        write(" is a multiple of 3 and 5") nl
    }
    otherwise {
        write(n)
        write(" is a multiple of 3") nl
    }
}
otherwise{
    write(n)
    write(" is not a multiple of 3") nl
}

m is 11
if (n equals 15) and (m equals 20) {
    write("n is 15 and m is 20") nl
}
otherwise if (n equals 15) and (m equals 11) {
    write("n is 15 and m is 11") nl
}
otherwise{
    write("neither are correct") nl
}

a is 2
if a greater than 1 or a equals 0{
    write("a is greater than 1") nl
}
```

Overall Result

Pass

Test Name: While Loops

Test Description

The loop should be able to run given a certain provided condition is true, and stop once the condition is broken.

Ensure the breakpoints work as expected.

Test Data

`while-loop.st`

Expected Result

We should see the current state of `n` while it is less than 3, and each time we should see the state of m, as well as seeing `m` reset for each iteration.

We should stop on every line of the while loop, as well as the condition.

All other non while breakpoints should be unaffected.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it's breakpoints.

Check the `output.txt` file to ensure variables are of correct value at the end of the program.

Actual Result

The loop iterated 3 times and stopped once `n` became 3, as expected, and each time the value of `m` was reset successfully.

While loops stopped on conditions, bug has been fixed.

Evidence

while-text.txt

```
n is 0        You, 5 days ago • ju
loop while n < 3{
    write("current n is ")
    write(n) nl
    m is 0
    loop while m < 3{
        write("current m is ")
        write(m) nl
        m is m add 1
    }
    n is n add 1
}
```

Overall Result

Pass

Test Name: For Loops

Test Description

The loop should be able to move through each element of the array, as well as the sub elements of a matrix in this given case.

Ensure the breakpoints work as expected.

Test Data

`for-loop.st`

Expected Result

We expect to see the numbers 1 to 6 sequentially as the embedded loops work to extract the elements of the matrix.

Execution Steps

File executed in Windows Powershell using START.

Enter Key used to advance the program past it’s breakpoints.

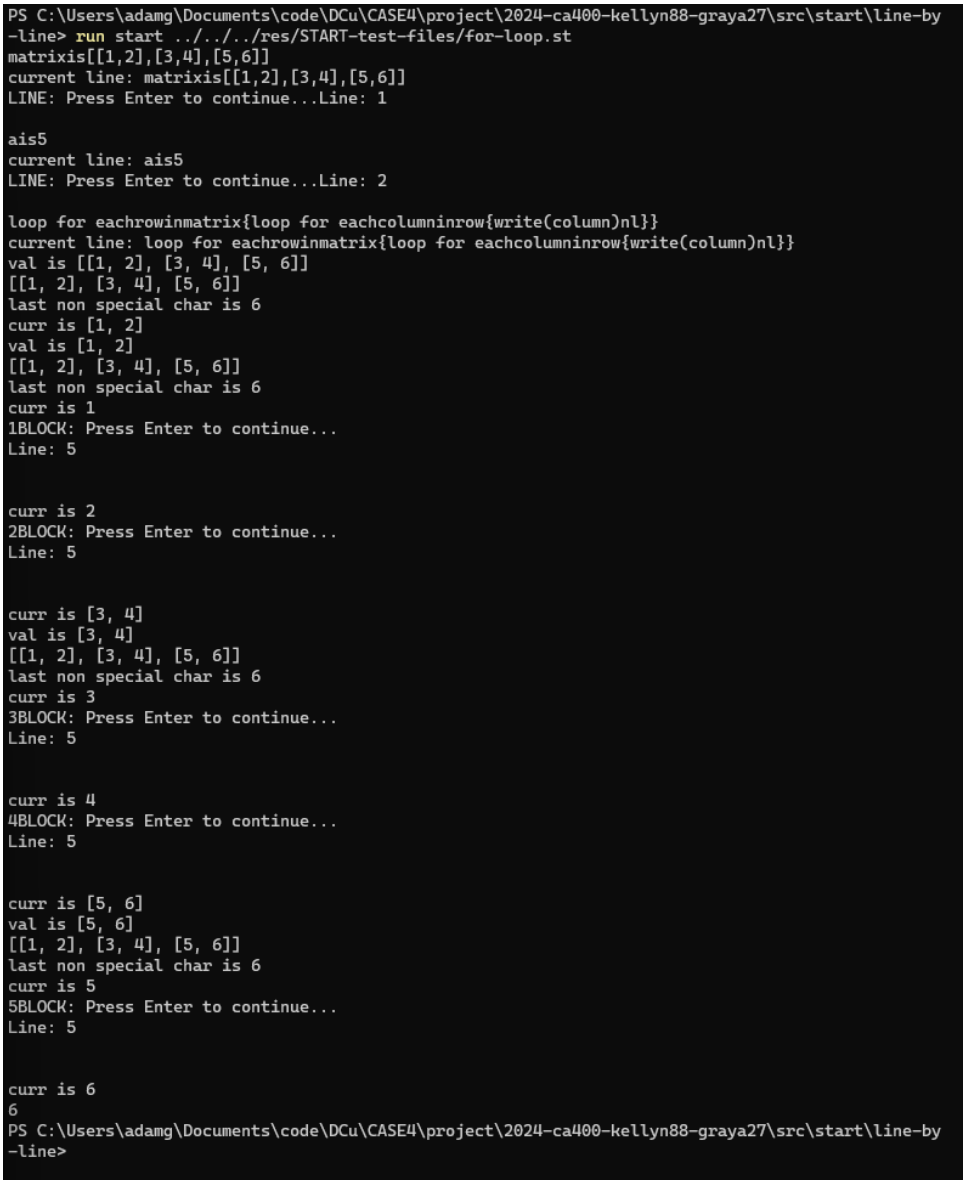
Check the `output.txt` file to ensure variables are of correct value at the end of the program.

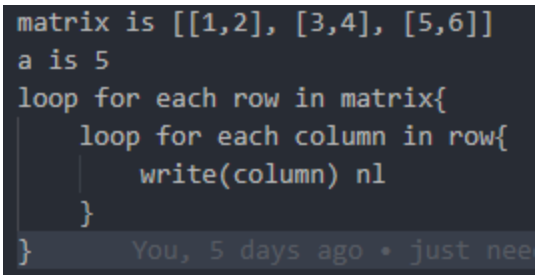
Actual Result

We see the numbers of the matrix printed out in order correctly.

The program no longer stops on for loop lines, hence the bug has been fixed.

Evidence





Overall Result

Pass

Document Review

In the initial feature tests, all but 3 of the features were found to be bug free. The 3 tests that contained issues that needed further development time were:

- If Statements
- For Loops
- While Loops

Adam took the found bugs and began to work on fixing the found issues. Once they had been implemented, we needed to then rerun the tests that failed as part of a regression test. All 3 of the features then passed, we attribute this to knowing exactly what the issues where due to our thoroughly thought out and developed test cases.

Author & Reviewer

Document Author and Tester: Adam Gray

Document and Test Reviewer: Niall Kelly

Date: 28/01/24