START Base Language Feature Testing

Niall Kelly 20461772

Adam Gray 20364103

START Web Application

Document finalised 22/01/2024

Introduction

This document will contain the the tests carried out as part of a regression test of the START language developed in 2023 by ourselves. This is needed as we are using our language as part of our web application.

Test Type

Regression Test.

Test Strategy

The testing strategy for this will be a manual process of running test scripts we created which will be discussed below. Each set of tests for each feature will be documented below.

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

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.

Test Data

`var-assign.st`

Expected Result

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

Execution Steps

File executed in Windows Powershell using START.

Actual Result

All variables were of correct value.

Evidence

```
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
```

```
PROBLEMS   OUTPUT   TERMINAL   PORTS   GITLENS   SQL CONS

PS C:\...\START-test-files>start1 .\var-assign.st
10
true
null
jon
10.5
PS C:\...\START-test-files>
```

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.

Test Data

`addition.st`

Expected Result

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

Execution Steps

File executed in Windows Powershell using START.

Actual Result

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

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)
```

```
PROBLEMS    OUTPUT    TERMINAL    PORTS    GITLENS

PS C:\...\START-test-files>start1 .\addition.st
1
0
-24
-90
0
PS C:\...\START-test-files>
```

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.

Test Data

`subtraction.st`

Expected Result

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

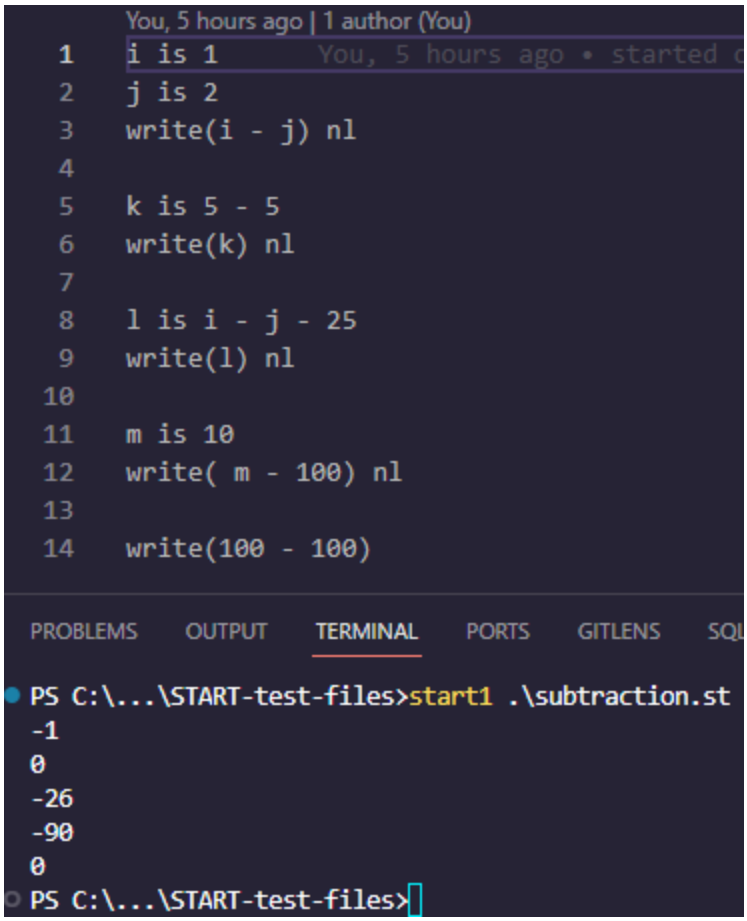
Execution Steps

File executed in Windows Powershell using START.

Actual Result

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

Evidence



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.

Test Data

`multiplication.st`

Expected Result

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

Execution Steps

File executed in Windows Powershell using START.

Actual Result

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

Evidence

```
You, 4 hours ago | 1 author (You)
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        You, 4 hours ago
write(5 * 10) nl
```

```
PROBLEMS   OUTPUT   TERMINAL   PORTS   GITLENS   ...

PS C:\...\START-test-files>start1 .\multiplication.st
50.0
4
10.0
200.0
200.0
100.0
50
PS C:\...\START-test-files>
```

Overall Result

Pass

Test Name: Division

Test Description

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

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.

Execution Steps

File executed in Windows Powershell using START.

Actual Result

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

Evidence

```
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
```

PROBLEMS OUTPUT TERMINAL PORTS GITLENS

```
PS C:\...\START-test-files>start1 .\division.st
2.0
1.0
2.5
50.0
50.0
25.0
PS C:\...\START-test-files>
```

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.

Test Data

`modulus.st`

Expected Result

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

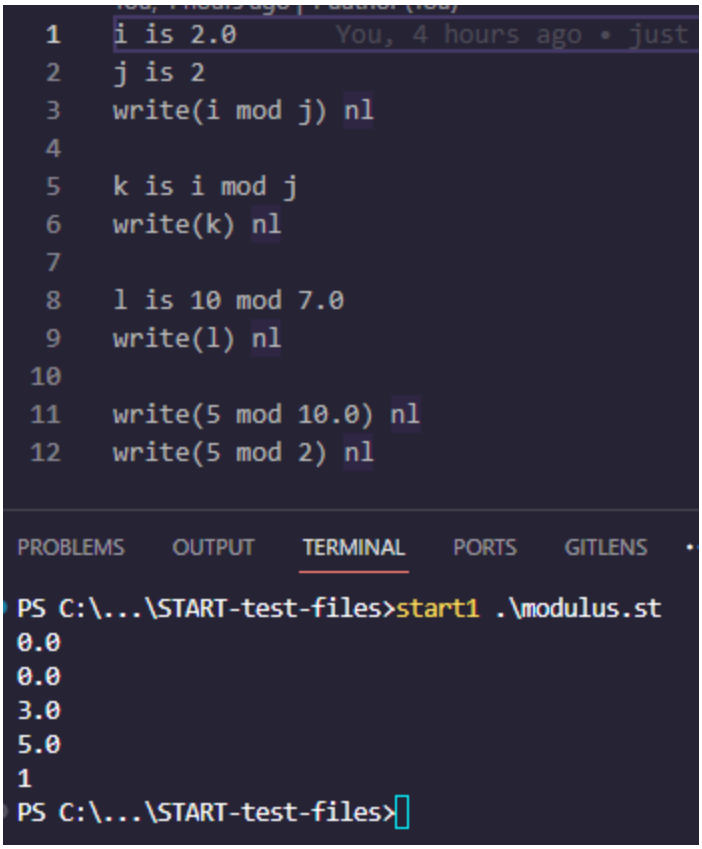
Execution Steps

File executed in Windows Powershell using START.

Actual Result

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

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.

Test Data

`powers.st`

Expected Result

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

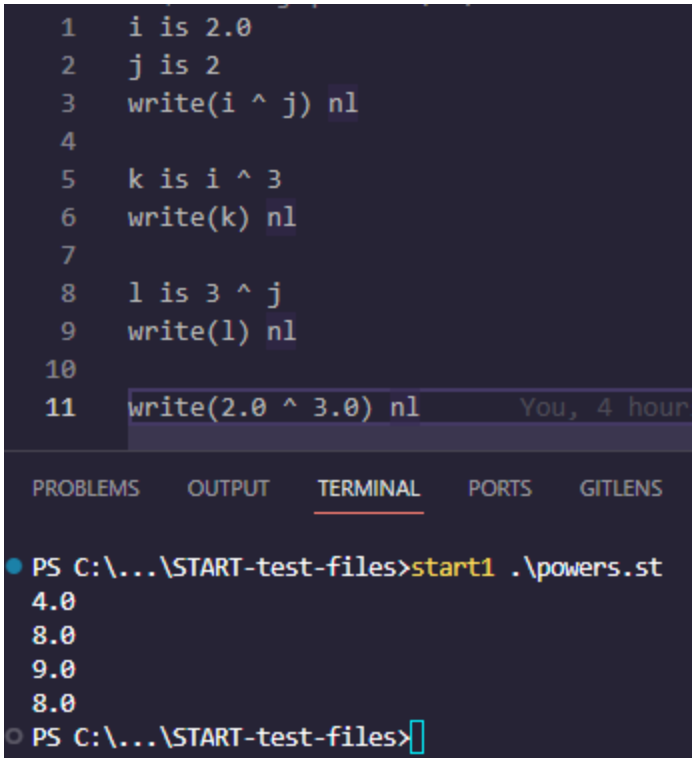
Execution Steps

File executed in Windows Powershell using START.

Actual Result

All values was correct after the power operator was invoked.

Evidence



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.

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.

Execution Steps

File executed in Windows Powershell using START.

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.

Evidence

```
You, 4 hours ago | 1 author (You)
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
```

```
PROBLEMS   OUTPUT   TERMINAL   PORTS   GITLENS

PS C:\...\START-test-files>start1 .\not-paren
false
true
true
false
PS C:\...\START-test-files>
```

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.

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.

Execution Steps

File executed in Windows Powershell using START.

Actual Result

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

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
```

```
PROBLEMS    OUTPUT    TERMINAL    PORTS    GITLENS

PS C:\...\START-test-files>start1 .\comp.st
false
true
true
false
true
true
true
PS C:\...\START-test-files>
```

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.

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.

Execution Steps

File executed in Windows Powershell using START.

Actual Result

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

Evidence

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

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

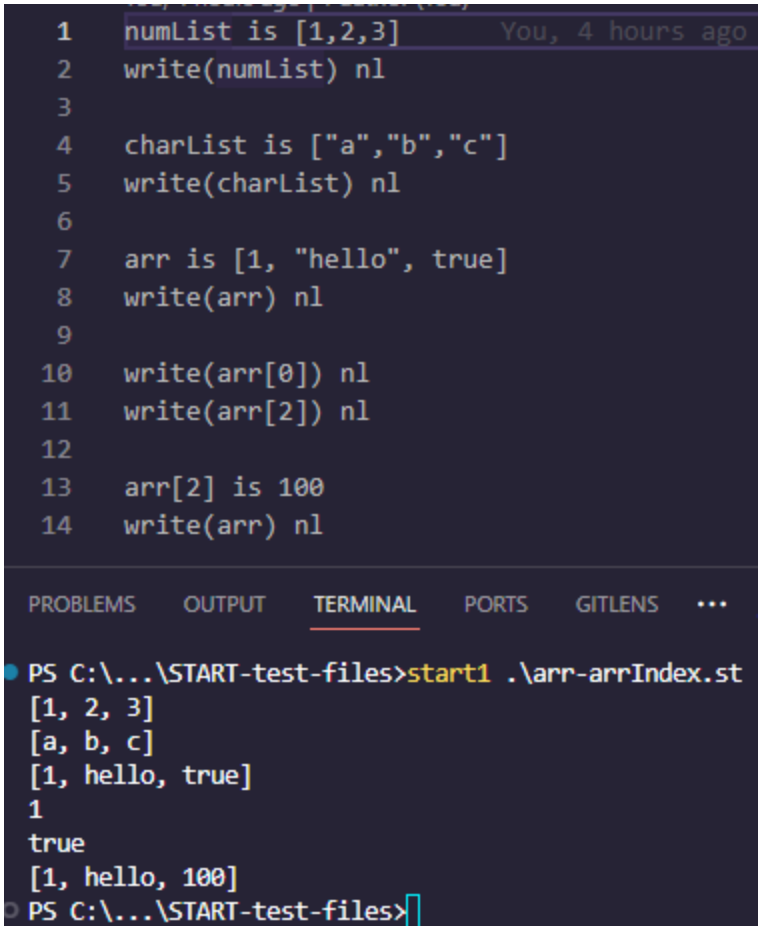
Execution Steps

File executed in Windows Powershell using START.

Actual Result

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

Evidence



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

PROBLEMS OUTPUT TERMINAL PORTS GITLENS ...

```
PS C:\...\START-test-files>start1 .\arr-append.st
[1, 2, 3]
[1, 2, 3, 100]
PS C:\...\START-test-files>
```

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

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

PROBLEMS OUTPUT TERMINAL PORTS GITLENS ...

```
PS C:\...\START-test-files>start1 .\arr-concat.st
[1, 2, 3, 4, 5, 6]
PS C:\...\START-test-files>
```

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

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

PROBLEMS OUTPUT TERMINAL PORTS GITLENS ...

```
PS C:\...\START-test-files>start1 .\arr-length.st
[1, 2, 3]
3
3
PS C:\...\START-test-files>
```

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

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

```
PROBLEMS   OUTPUT   TERMINAL   PORTS   GITLENS   ...

PS C:\...\START-test-files>start1 .\arr-remove.st
[1, 3]
[world]
PS C:\...\START-test-files>
```

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, otherwsie 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.

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
```

Execution Steps

File executed in Windows Powershell using START.

Actual Result

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

Evidence

```
n is 15

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 11") 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
}
```

PROBLEMS OUTPUT TERMINAL PORTS GITLENS ··· powe

```
PS C:\...\START-test-files>start1 .\if-statement.st
15 is a multiple of 3 and 5
n is 15 and m is 11
a is greater than 1
PS C:\...\START-test-files>
```

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.

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.

Execution Steps

File executed in Windows Powershell using START.

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.

Evidence

```
n is 0
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
}
```

```
PROBLEMS    OUTPUT    TERMINAL    PORTS    GITLENS

PS C:\...\START-test-files>start1 .\while-loop.st
current n is 0
current m is 0
current m is 1
current m is 2
current n is 1
current m is 0
current m is 1
current m is 2
current n is 2
current m is 0
current m is 1
current m is 2
PS C:\...\START-test-files>
```

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.

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.

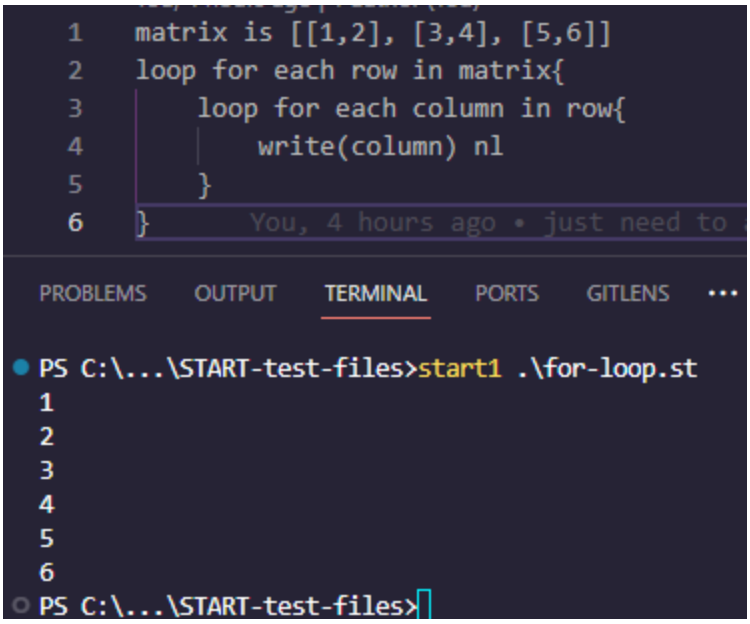
Execution Steps

File executed in Windows Powershell using START.

Actual Result

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

Evidence



Overall Result

Pass

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.

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.

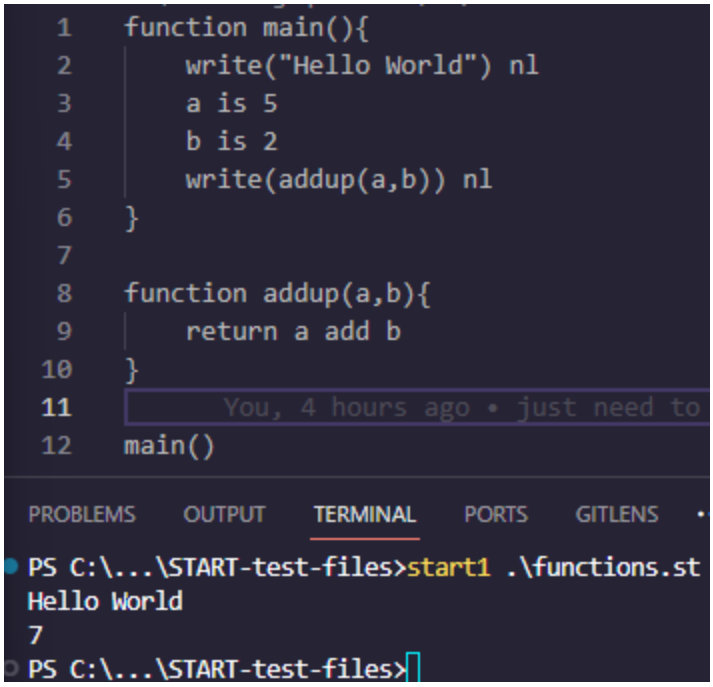
Execution Steps

File executed in Windows Powershell using START.

Actual Result

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

Evidence



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.

Test Data

`remove-all.st`

Expected Result

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

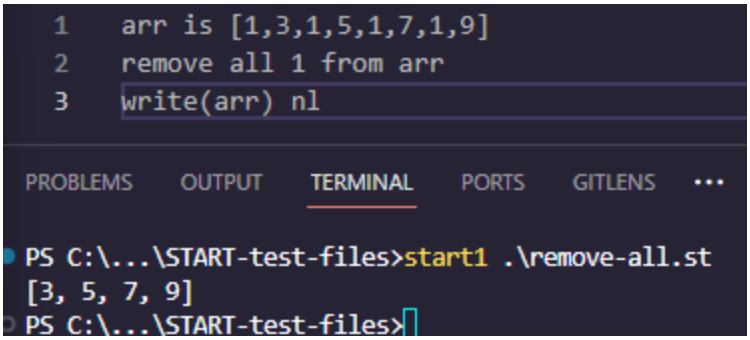
Execution Steps

File executed in Windows Powershell using START.

Actual Result

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

Evidence



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.

Test Data

`string-dex.st`

Expected Result

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

Execution Steps

File executed in Windows Powershell using START.

Actual Result

All outputted values were of correct value.

Evidence

```
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")
```

PROBLEMS OUTPUT TERMINAL PORTS GITLENS ...

```
PS C:\...\START-test-files>start1 .\string-dex.st
char at positionn 0 is:  h
char at positionn 3 is:  l
space is:    before here
PS C:\...\START-test-files>
```

Overall Result

Pass

Test Name: Comments

Test Description

Comments should be ignored completely by the program.

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.

Execution Steps

File executed in Windows Powershell using START.

Actual Result

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

Evidence

```
// 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 //
```

```
PROBLEMS    OUTPUT    TERMINAL    PORTS    GITLENS

PS C:\...\START-test-files>start1 .\comments.st
5
PS C:\...\START-test-files>
```

Overall Result

Pass

Document Review

The regression test of the language showed no flaws with the implementation previously designed as we had expected. This allows us to now move onto expanding the language into different versions to be used by the debugger.

Author & Reviewer

Document Author and Tester: Adam Gray

Document and Test Reviewer: Niall Kelly

Date: 22/01/2024