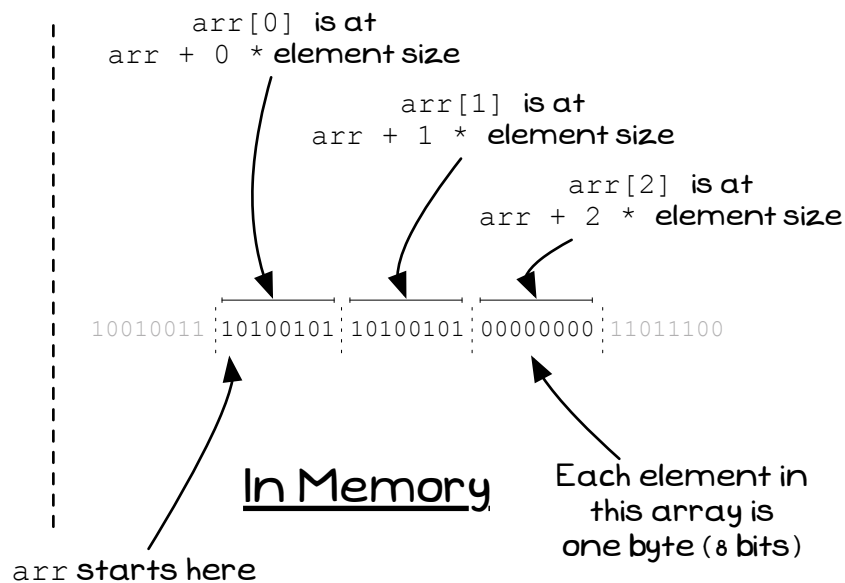
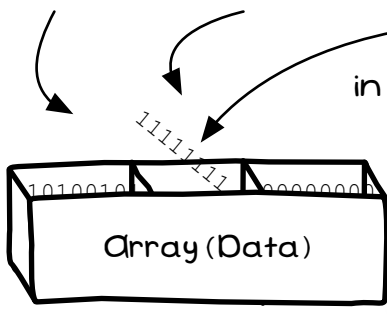


Conceptually



In this array [at index 1]



and stored
in an element in
the array

Value is
Calculated

Expression
The Value is
an Expression

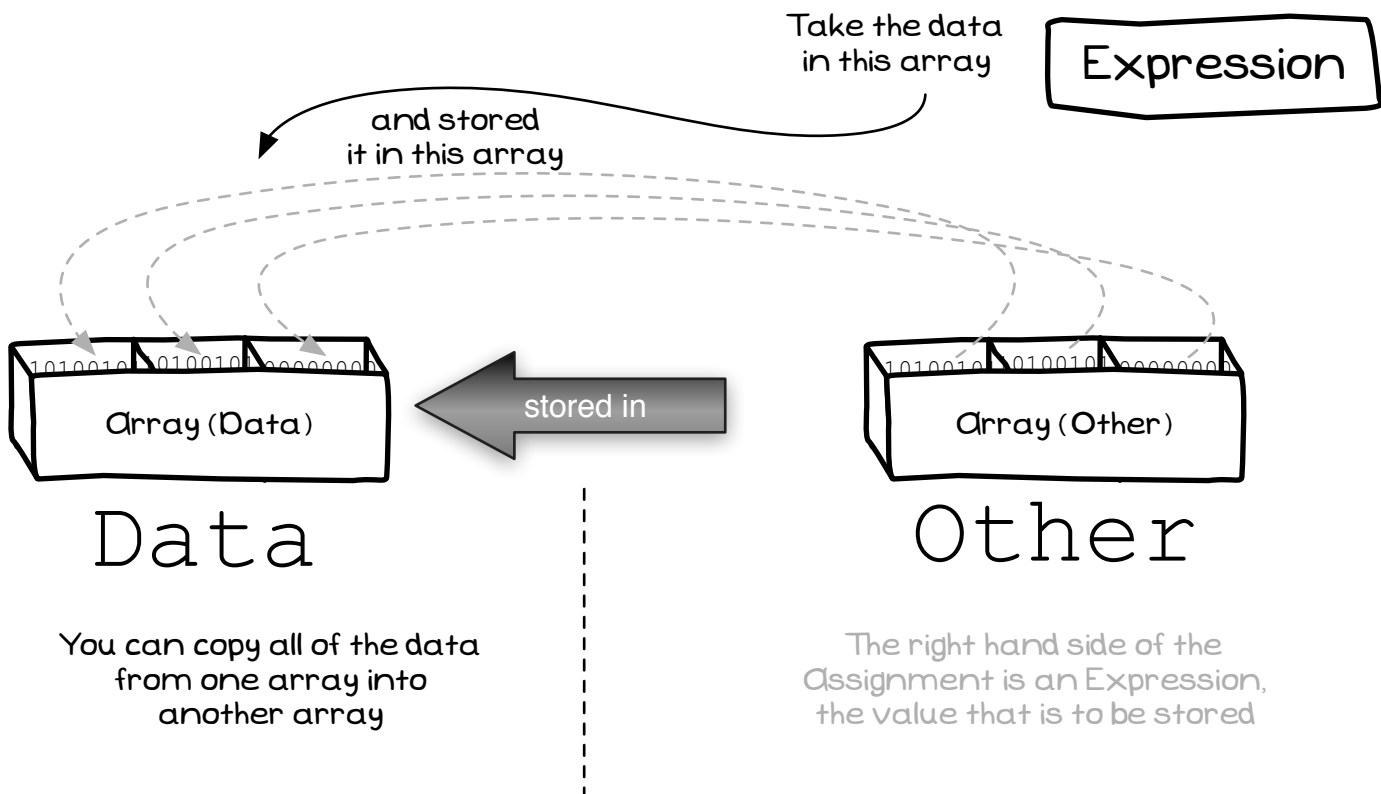
Data [1]



127.5 * 2

When storing a value in an
array, you specify both the
array and the index you
want the value stored at

The right hand side of the
Assignment is an Expression,
the value that is to be stored

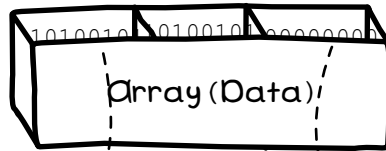


Is a value.

Expression

Values are read
from array elements

Part or all of the
value can be read
from array(s)



$3 * \text{Data}[0] + \text{Data}[i+1]$

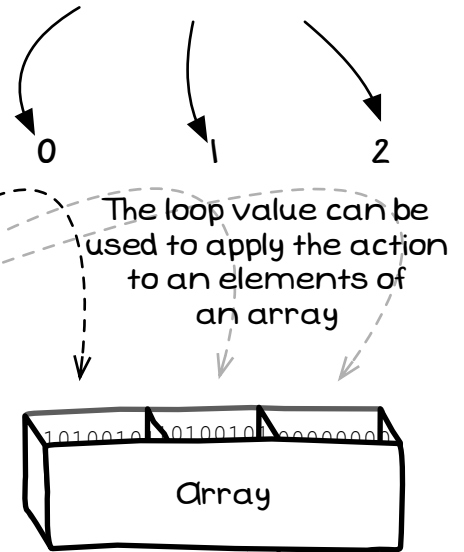
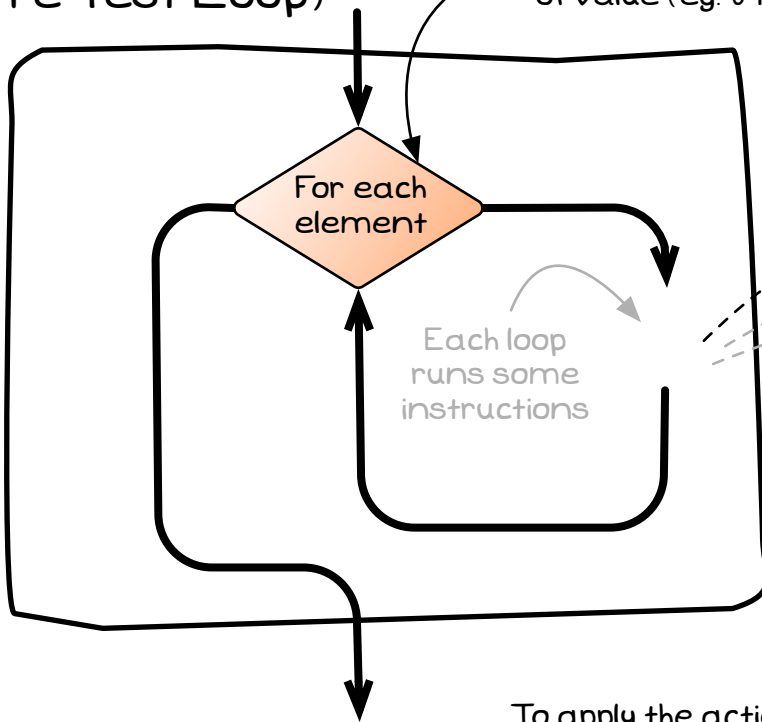
The array index is
an Expression

Array index values can be calculated.
If the value in i is 1, then this has the value
2, indicating the 3rd element of the array

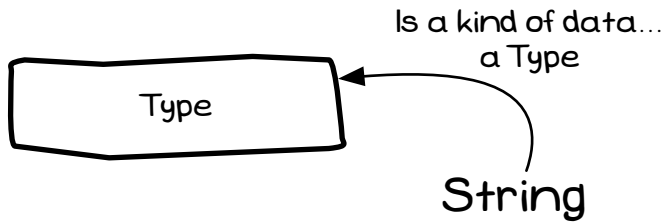
For Loop (Pre-Test Loop)

Can be thought of
as a counting loop,
looping over a range
of value (eg. 0 to 2)

Iterates over a
range of values



To apply the actions to all elements in an array the for loop is used to count from the array's lowest index to its highest index.

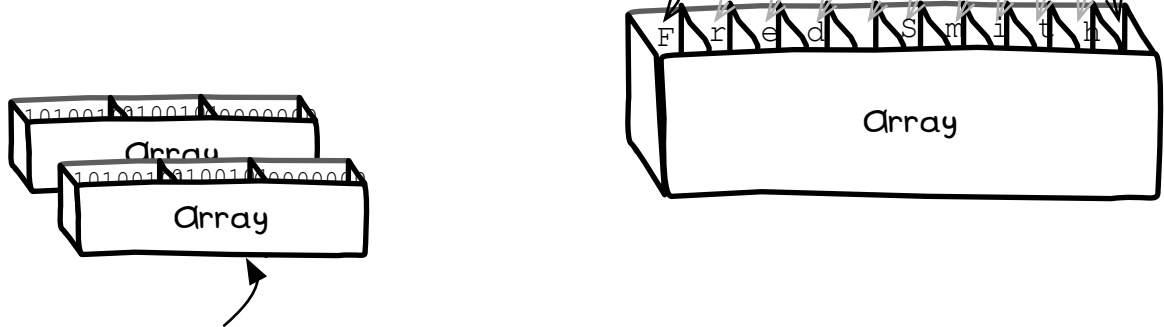


A String is text data,
a sequence (string) of characters

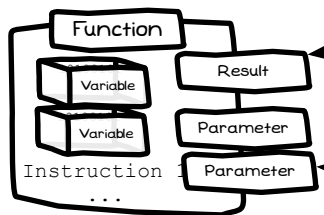
'Fred Smith'

Strings are stored
as arrays
of characters

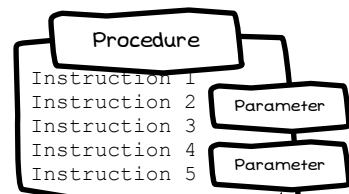
One byte
overhead



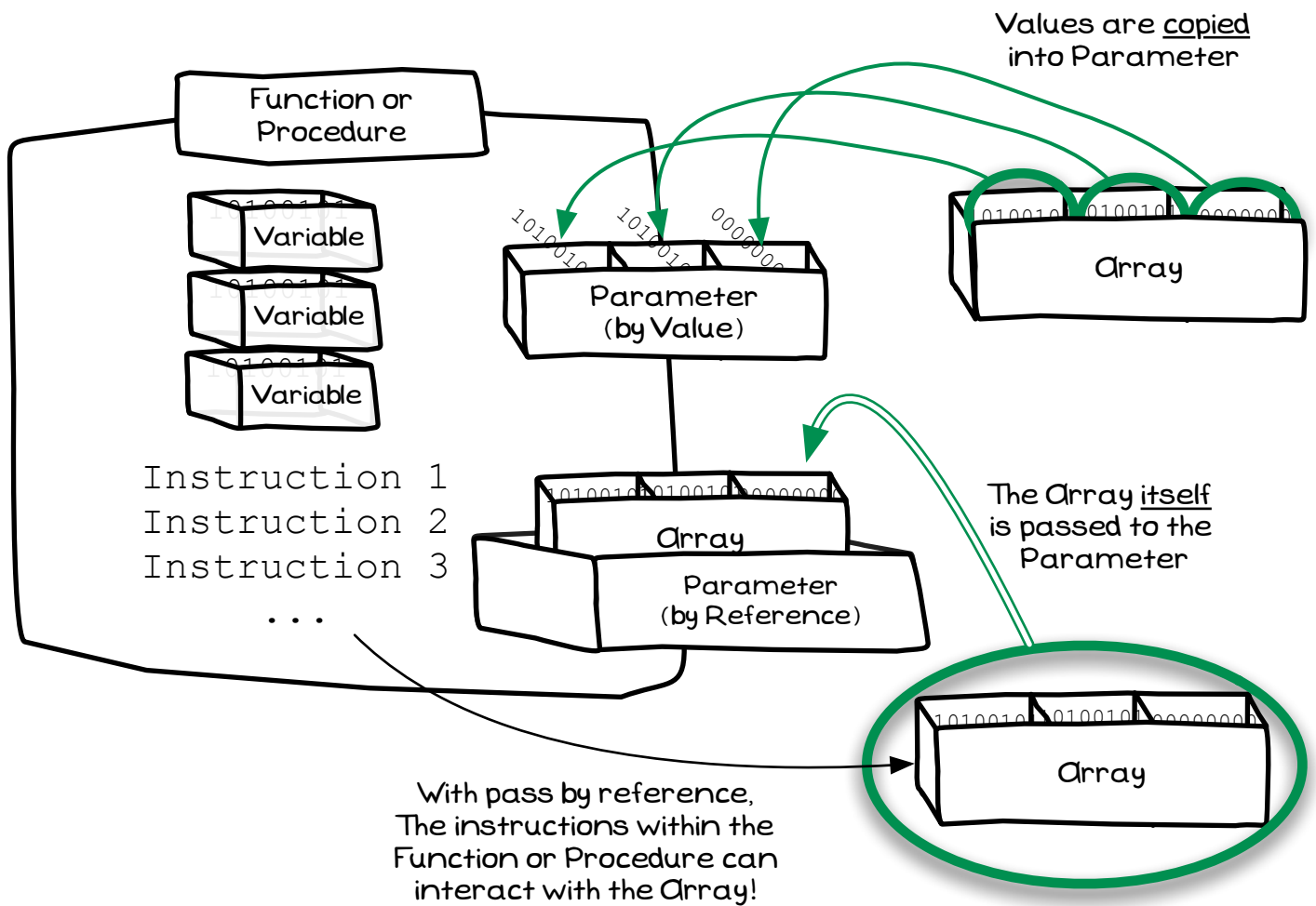
Variables can store
String values

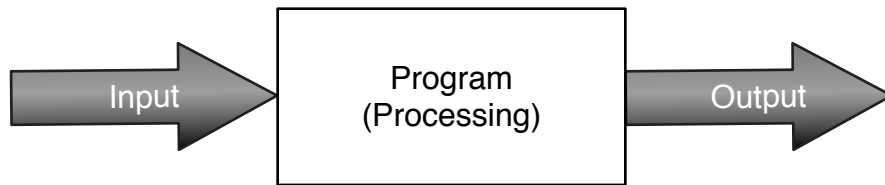


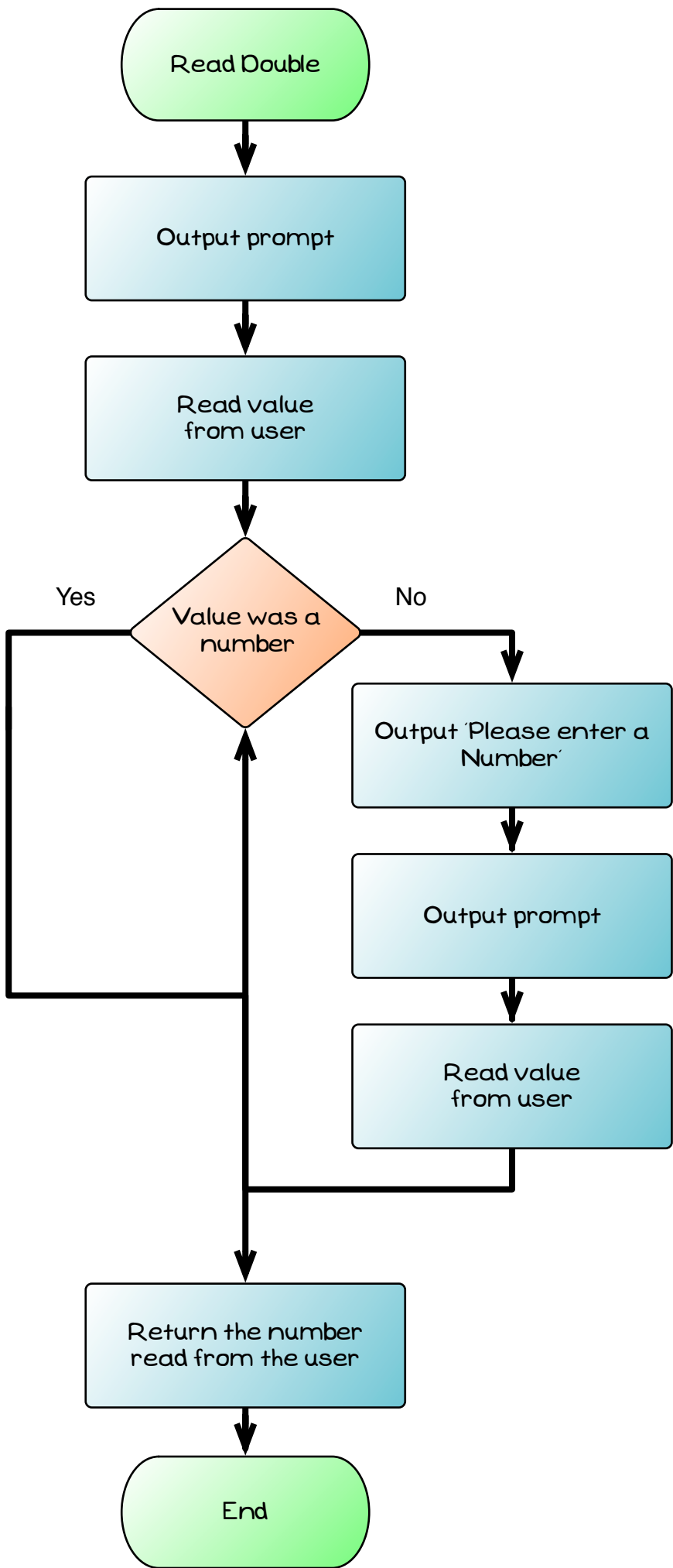
A Functions result may
be a String value

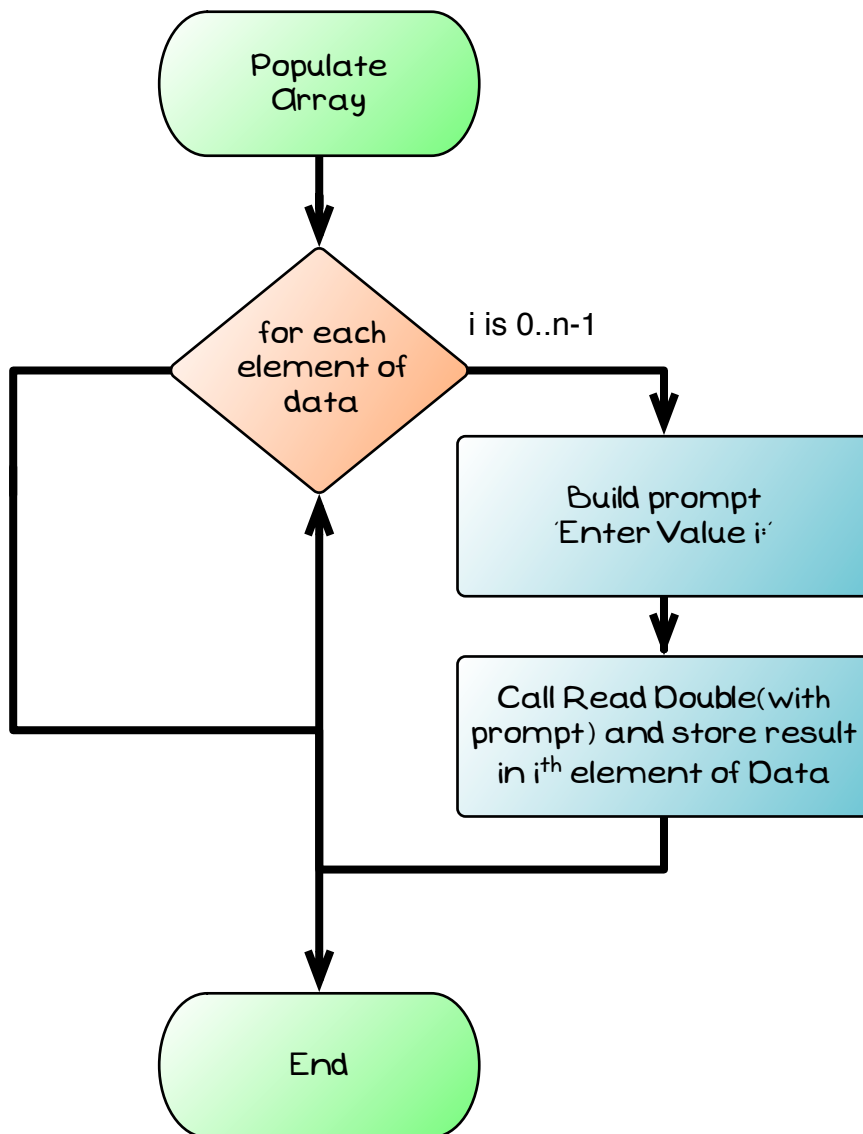


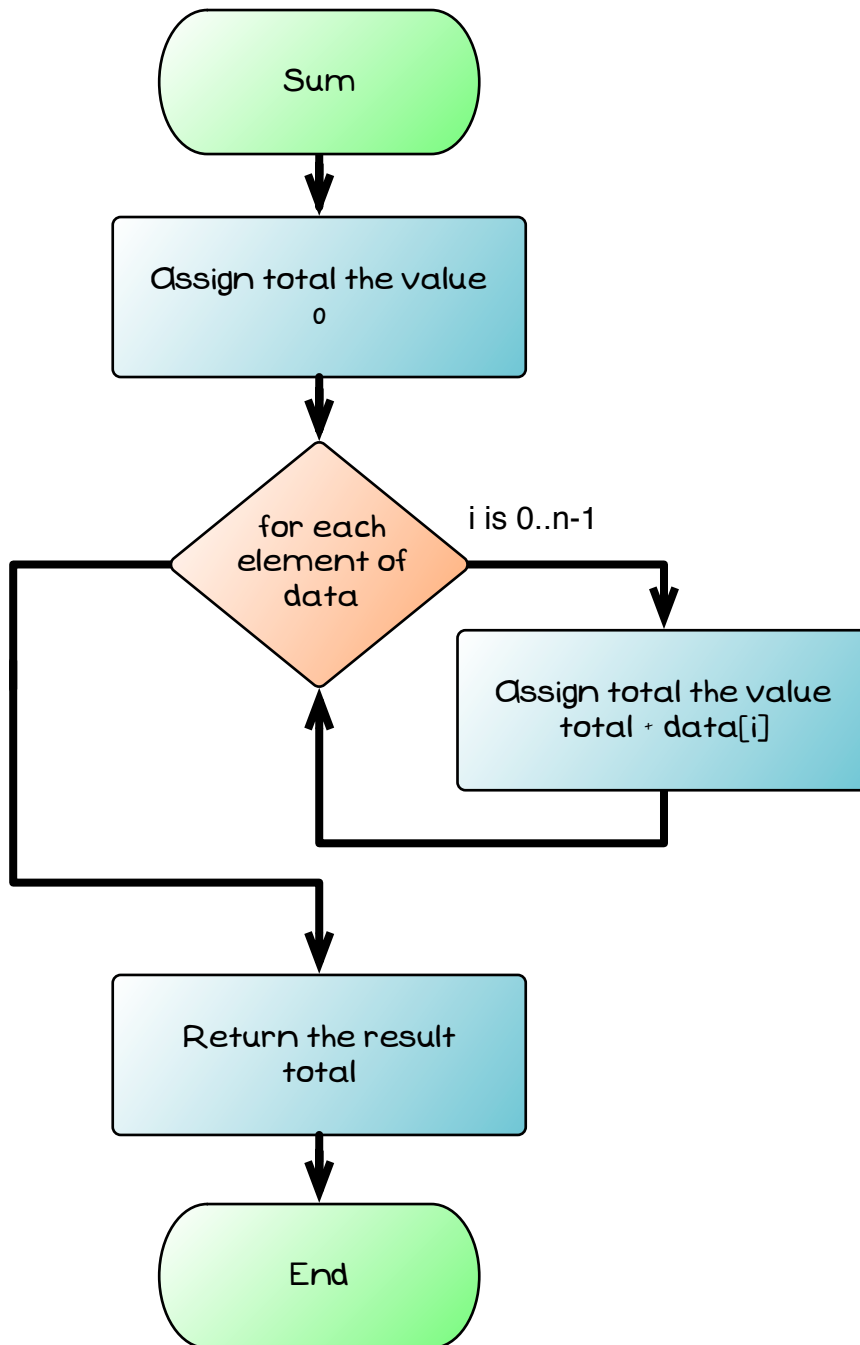
Parameters can accept
String values

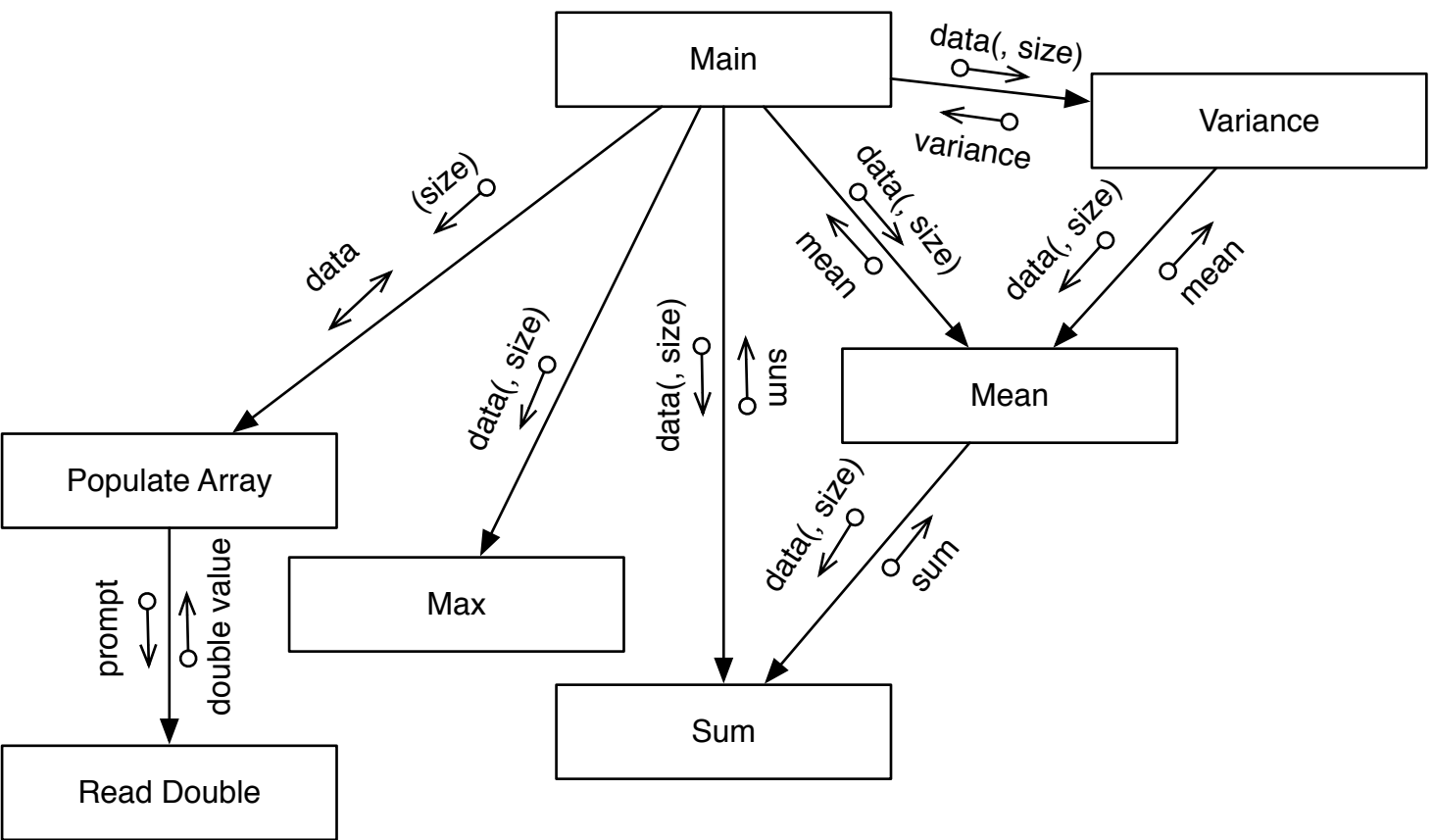


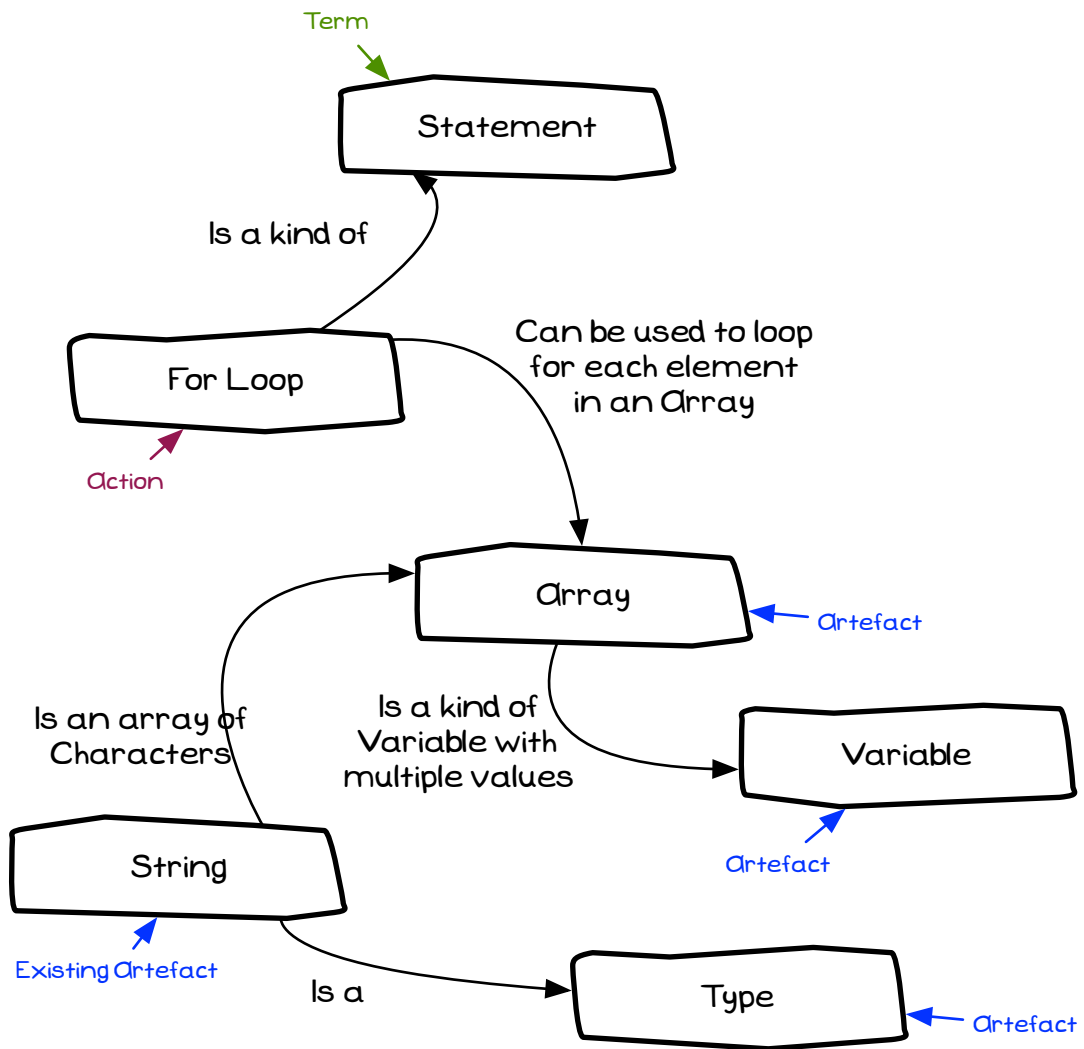


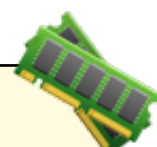












Procedure: Populate Array

Parameters:

- 1: data (by ref, array of Double) - the list of values to sum
- 2: size (Integer) - the number of elements in data (C only)

Local Variables:

- *: i (Integer) - index of the current element in the array
- *: prompt (String) - the prompt to be shown to the user

Steps:

- 1: For i, loops from lowest to highest index of data
- 2: Assign prompt, 'Enter value ' + (i + 1) + ': '
- 3: Assign data[i], the result of calling Read Double(prompt)

Procedure: Main

Local Variables:

- *: my_data (array containing 3 Double values) - data array

Steps:

- 1: Call Populate Array (my_data, 3)
- 2: ...



With the value returned, the function ends

The result is returned to the expression where the function was called

3

2

Function: Sum

Returns: Double - The sum of the numbers from the data array
Parameters:

1: data (by const ref, array of Double) - values to sum
2: size (Integer) - the number of elements in data (C only)

Local Variables:

*: i (Integer) - index of the current element in the array
*: total (Double) - running total

Steps:

1: total is assigned 0

2: For i, starts at 0 and loops to the highest index of data

3: total is assigned total + data[i]

4: Return the result, total

Procedure: Main

Local Variables:

*: my_data (array containing 3 Double values) - data array

Steps:

1: Call Populate Array (my_data, 3)

2: Output 'Sum is ', and Sum (my_data

3: ...

1

Sum	
data[]:	buff1
size:	3
i:	3
total:	22.21
result:	22.21
Instruction:	Step 2

Main	
my_data[]:	10.0
	-5
	17.21
Instruction:	Step 2

Total becomes the value returned



Enter value 1: Ten
Please enter a number.
Enter value 1: 10.0
Enter value 2: -5
Enter value 3: 17.21



Before	Call	After
<div>prompt:<div>ashe28&2kk;)j2;0J</div></div> <div>buffer:<div>#dZ</div></div>	<div>strncpy(prompt, "Enter value ", 13);</div>	<div>prompt:<div>Enter value 2;0J</div></div> <div>buffer:<div>#dZ</div></div>
<div>prompt:<div>Enter value 2;0J</div></div> <div>buffer:<div>#dZ</div></div>	<div>sprintf(buffer, "%d", (i + 1) % 100);</div>	<div>prompt:<div>Enter value 2;0J</div></div> <div>buffer:<div>1Z</div></div>
<div>prompt:<div>Enter value 2;0J</div></div> <div>buffer:<div>1Z</div></div>	<div>strncat(prompt, buffer, 2);</div>	<div>prompt:<div>Enter value 1Z;0J</div></div> <div>buffer:<div>1Z</div></div>
<div>prompt:<div>Enter value 1Z;0J</div></div> <div>buffer:<div>1Z</div></div>	<div>strncat(prompt, ": ", 2);</div>	<div>prompt:<div>Enter value 1:Z</div></div> <div>buffer:<div>1Z</div></div>