Vowel

The Way for Wordsmiths

Name	UNI
Coby Simler	zys2102
Aidai Beishekeeva	ab5248
Lex Mengenhauser	am4958
Vikram Rajan	vjr2123

1. Motivation

Certain tasks require programmatic processing of large amounts of textual data. Such tasks often involve analyzing, operating on, and comparing large bodies of structured and unstructured streams of text. Iterating over and manipulating these streams with existing languages means relying on generic control flow and syntax, or otherwise importing third-party libraries. Vowel aspires to enable programmers to quickly analyze, manipulate, and map functions onto such large text streams.

2. Description

Vowel is a high-level, imperative, statically-typed programming language intended to be used to iterate over, operate on, and manipulate large text streams. To do this, Vowel introduces the 'stream' data type and supports a novel control flow syntax tailored for text processing. Specifically, the language will have Python-style control flow and function definition syntax with Java-like typing. This will enable potential optimizations due to known types at compile time. In addition, the language will use Java-live parentheses, curly braces, and semicolons.

3. Language Features

3.1 Stream data type

Key to Vowel is the native `stream` data type, which is intended to hold text data. The underlying storage of a `stream` object in memory will be no different than that of a string, though Vowel-implemented operators (e.g. +, -, *, /) will perform distinct manipulations on stream parameters. Additionally, strings in this language are immutable whereas streams are not. The following table describes the `stream` data type and operators.

3.2 Operators for Stream data type

Operator	Data Types	Description
+	`stream A + stream B = list C`	Produces a list of words that appear in the union of both streams

-	`stream A - stream B = list C`	Produces a list of words that appear in `stream A` and not in `stream B`
%	`stream A % int x = stream A`	Modifies `stream A` to contain the first `x` sentences in the `stream A`
*	`stream A * int x = stream A`	Modifies `stream A` to contain the text of `stream A` repeated `x` number of times.
/	`stream A / stream B = list C`	Produces a new `list C` that contains the intersection of `streams A` and `B`
[]	`stream A[language unit int x : int y] = stream A`	Modifies `stream A` to contain language unit `x - (y-1)`

3.3 Control Flow Syntax

In addition to providing Python-like `while` loops and `for` loops, Vowel introduces a new loop syntax and keywords intended to simplify the process of iterating over and manipulating stream data types. Programmers can use the 'replace' keyword and syntax to more easily express stream manipulations in fewer lines. For instance, a program in Vowel to apply the .upper() method to every third word in a string might look like the following:

```
stream sentence = "The quick brown fox jumps over the lazy dog";
replace word 3 in sentence { word.upper(); }
```

4. Syntax

4.1 Reserved Words

```
while, for, in, if, else, def, class, return, true, false, int, float,
char, bool, string, stream, len, letter, word, paragraph
```

4.2 Primitive Data Types

	Description	Example
Int	Numeric type. Positive or negative integer number.	int year = 2021
Float	Numeric type. Positive or negative decimal number.	float pi = 3.14

Bool	Boolean value: true or false.	bool coffeeDay = true
String	Text type. Immutable sequences of Unicode code points.	string name = “Vowel”
Stream	Text type. Mutable sequences of Unicode code points. Provides more functionality than String data type.	stream a = “Some large text here”

4.3 Basic Operators

Operation	Data types	Description	Example
+	Int, Float, String	Addition Int + Int = Int Int + float = float Str + str = str	 4 + 4 = 8 4 + 5.0 = 9.0 “Vo” + “wel” = “Vowel”
-	Int, Float	Subtraction Int - Int = Int Int - Float = Float	 16 - 8 = 9 13 - 7.0 = 6.0
*, /	Int, Float	Multiplication, Division Int * Int = Int Int * Float = Float Float * Float = Float	 4 * 3 = 12 4 * 1.2 = 4.8 1.2 * 1.2 = 1.44

4.4 Assignment, Comparison, and Logical Operators

Operator	Description	Example
=	Assignment operator	string color = “grey”
+=, -=	Adds/Subtracts value from existing variable and assign it new value	int a = 5 a += 3
==	Compares values of two expressions. Returns True of False	“Vowel” == “VowEl” ⇒ False
>, >=, <, <=	Checks if a value is larger/smaller (or equal).	4 >= 4.0 ⇒ True
!=	Check if values of two expressions are not equal.	int a = 5 Int b = 8 a != b ⇒ True

and	Returns True if all of the expressions are True, otherwise False	9 > 5 and 8 < 13 ⇒ True
or	Returns True if one of the expression is True, otherwise False	7 > 9 or 4 < 19 ⇒ True
not	Inverses the result of False to True, True to False	not True ⇒ False

4.5 Stream Operators

Member Variables

- stream.**word_count**
  - Returns total number of words in a stream
- stream.**paragraph_count**
  - Returns total number of paragraphs in a stream
- stream.**page_count**(int size)
  - Returns page count based off size of each page (words per page)

Operators

- stream.**cite**(int year, string author, int date_published)
  - Returns a dictionary mapping the input stream with a MLA citation
- stream.**concatenate**(stream s)
  - Adds the contents of the stream passed in to the stream object called on
- stream.**toString()**
  - Converts a stream to type string
- stream.**replace**(string source, string new_value)
  - Returns the stream with replaced values
- stream.**freq**(string s)
  - Returns an integer frequency of the word's appearance in the stream
- stream.**split_p**()
  - Returns a list where the values are paragraphs, delimited by "\t"
- stream.**split_s**()
  - Returns a list where the values are sentences, delimited by "., !, ?"
- stream.**split_w**()
  - Returns a list where the values are words, delimited by " " and with ending punctuation removed
- stream.**contains**(string s)
  - Returns true if string s is contained inside the stream
- string.**toStream()**
  - converts a string to type stream

5. Sample Code

The following code finds every third occurance of the word "very" in a stream and replaces it with a synonym for it. This is an example of something that could be useful for preventing the overuse of common words in writing.

```
for (3 word in stream) {
      if word == "very" { word = "extremely"; }
}
```

Make each word uppercase:

```
for (word in stream) { word = word.upper(); }
```

Returns the intersection of words:

```
def stream sharedWithConstitution (stream StreamA) {
    stream const = "We the People of the United States, in Order to
                    form a more perfect Union...";
    return StreamA / const;
}
```

To compare common words between a stream A and stream B:

```
def list compareStreams(stream streamA, stream streamB) {
   stream A = "My fun comparison sentence";

   stream B = "This is a super long multi paragraph page of text. I'm
   gonna go through it sentence by sentence. Using the function
   below. You'll see!";
   list my_union_lists = A*B;

   return my_union_lists;
}
```

Modifies stream to contain the first x(number) sentences in the stream:

```
def stream removeAfter(stream StreamA, int number) {
   return StreamA % number;
}
```

References

https://docs.oracle.com/javase/7/docs/api/
https://docs.python.org/3/c-api/index.html