Methods Necessary for String Classes in Object-Oriented Programming Languages

1 INTRODUCTION

In programming, a string is a primitive type designed to provide an interface for an ordered set of characters. However, standard library implementations of this interface vary in different languages. Thus, it is clear that the programming community has not decided on the optimal implementation of a string class. Therefore, this paper aims to determine a minimalistic set of string functions sufficient for a string class.

2 SET BUILDING

2.1 Necessary and Sufficient Functions

One of the ways to select necessary and sufficient set of methods is to select methods that are sufficient and necessary to build a Turing machine. The Turing machine is the automaton with the highest computational capability, thus, any program that can be implemented can be implemented through selected methods. The memory tape of the Turing machine can be represented as three strings: parts of the memory tape right and left from the pointer and the memory cell with the pointer. Three operations are necessary for a Turing machine: (1) move the head to the left, (2) move the head to the right, and (3) write a symbol to the cell. Therefore, assuming that assignment operations and user interactions are handled by other aspects of programming language, the following methods are necessary:

(1) `concatenate`—concatenate two strings
(2) `slice`—returns the substring of requested length, starting at a given position as a string
(3) `length`—returns the length of the string

If `left_side` is part of the tape left to the head, `right_side` is part of the tape right to the head, `cell` is the cell tape with the head on it. Turing machine operations can be implemented as follows:

(1) Move the head to the left:

```
right_side := cell.concatenate(right_part);
cell := left_side.slice(left_side.length() - 1, 1);
left_size := left_side.slice(0, left_side.length() - 1);
```

Author's address:

(2) Move the head to the right:

```
left := left_side.concatenate(cell);
cell := right_side.slice(0, 1);
right_side := right_side.slice(1, right_side.length() - 1);
```

(3) Write a symbol to the cell:

```
cell := symbol;
```

Therefore, all computable algorithms, including string-related algorithms, can be described through strings using these three methods.

2.2 Populating the Set

However, such an approach is not practical, and other methods need to be included. For this purpose, the following criteria for selection and algorithm for set construction are proposed:

2.2.1 Criteria.

(1) Can not be implemented in three or fewer instructions using functions already in set
(2) Can be implemented in the least amount of instructions among all remaining functions

2.2.2 Algorithm.

(1) Functions `slice`, `length`, and `concatenate` are added to the set
(2) The function that falls under all criteria in 2.2.1 is added to the set
(3) The second step is repeated until no fitting functions remain
(4) Then, the set is hedged by deleting functions that do not fit requirements outlined in step two

2.3 Analysis of the final selected functions

After executing the algorithm, following functions were selected:

(1) `concatenate`—concatenates two strings
(2) `length`—returns the length of the string
(3) `slice`—returns substring specified by arguments
(4) `compare`—compares two strings
(5) `capitalize`—converts the first character to the uppercase, and all other to the lowercase
(6) `hash`—returns the hash of the string
(7) `is-alphabetic`—checks if the string contains only letters
(8) `is-lowercase`—checks if the string contains only lowercase letters
(9) `is-uppercase`—checks if the string contains only uppercase letters
(10) `swap-case`—converts all lowercase characters to uppercase, and vise versa

(11) `sscanf`—scans the string according to format strings and returns resulting values
(12) `sprintf`—given the format string and a set of values, returns formetted output

2.4 Functions in Other Languages

The team analyzed the libraries of several languages: Ruby, C++, Java, and Python. The data was collected from GitHub using advanced search among all the repositories and compiled into a list of the most popular functions for each language. In C++, the selected functions cover 76% of the uses of functions with strings in general. In Python this number is 44%, Ruby - 33%, Java - 38%. The small percentage in the last three languages can be explained by a large number of redundant functions in that language. The most complete string libraries are the Java and Python ones. The C++ library is the most minimalistic one. This could be due to the age of C++, as the language appeared at 1985, almost 10 years earlier than other three. It lacks some common functions, but has a few unapplicable to high-level languages, mostly memory-related. For example, capacity - this function for strings is not available in Java, Python or Ruby. Ruby has the most balanced library, with many features but almost no duplicate functions as in Java. The collected data can be used to estimate the importance of a function from the frequency of its use by other programmers. This estimate may help improve the ease of use of the set of functions selected in 2.2. The analysis of a few of the functions, not included in the set can be found in *Table 1*

3 CONCLUSION

After analyzing the selection of functions, several were selected: `concatenate`, `slice`, `compare`, `capitalize`, `format`, `hash`, `is-alphabetic`, `is-lowercase`, `is-uppercase`, `to-float`, `swap-case`, and `to-int`. While selecting the functions, we were guided by features such as sufficient quantity for the Turing machine, the algorithm in the "Populating the Set" section, and statistics on GitHub.

Function name	Libraries	Reasons of excluding
capacity	C++	Presented only in one library, has limited functionality, used only in 0,02% of general string functions usage
fill-letft	Ruby	Presented only in one library, can be implemented in three or fewer instructions, used only in 0,002% of general string functions usage
fill-right	Ruby	Presented only in one library, can be implemented in three or fewer instructions, used only in 0,002% of general string functions usage
is-title	Python	Presented only in one library, can be implemented in three or fewer instructions, used only in 0,0001% of general string functions usage
repeat	Java, Python	Presented only in 2 libraries, can be implemented in three or fewer instructions, used only in 0,002% of general string functions usage
patrition	Python, Ruby	Presented only in 2 libraries, used only in 0,003% of general string functions usage
shrink-to-fit	C++	Presented only in one library, has limited functionality, used only in 0,003% of general string functions usage
strip	Java, Python, Ruby	Can be implemented in three or fewer instructions, used only in 0,015% of general string functions usage
starts-with	Java, Python, Ruby	Can be implemented in three or fewer instructions, used only in 0,01% of general string functions usage

Table 1. Analysis of the sample of the functions not included in the set.