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

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: move the head to the left, move the head to the right, and write a symbol to the cell. Therefore, the following methods are necessary: (1) concatenate—concatenate two strings (2) substring—returns the substring of requested length, starting at a given position as a string (3) length—returns the length of the string It is easy to see that these methods are sufficient to implement Turing machine commands. Therefore, all computable algorithms, including string-related algorithms, can be described through strings. Because standard libraries include regular expressions, function regex—split string into groups according to regular expression pattern.

2.2 Populating the Set

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

(1) Functions char-at, regex, and concatenate are added to the set
(2) The function that is (a) Can not be implemented in three or fewer instructions using functions already in set (b) Can be implemented in the least amount of instructions among all remaining functions 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

Author's address:

2.3 Analysis of the final selected functions

After executing the algorithm, following functions were selected: (1) concatenate—concatenates two strings (2) substring—returns substring specified by arguments (3) compare—compares two strings (4) capitalize—converts the first character to the uppercase, and all other to the lowercase (5) format—returns formatted string (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) to-float—converts string to float (12) to-int—converts string to integer

2.4 StringBuilder implementation

After discussing the details of the implementation of the functions, we pointed out that the focus was not on the efficiency of the functions, but purely on their availability and usability. Therefore it was decided at this stage of development to remove the implementation of the StringBuilder class from the general task list and to implement functions without efficient development. In the future, it may be possible to refine these functions with the addition of effective use by implementing the StringBuilder.

2.5 GitHub analysis

The team analyzed the libraries of several languages: Ruby, C++, Java, and Python. The data was collected from GitHub with an advanced search among all the code. The result was a list of the most popular functions for each language involved. Some of the most popular functions have already been implemented, as well as functions related to regular expressions. 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.

3 CONCLUSION

Therefore, after analyzing the selection of functions, we chose several: concatenate, substring, compare, capitalize, format, hash, is-alphabetic, is-lowercase, is-uppercase, to-float, swap-case, and to-int. In 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.