DOCUMENTATION

Github source code

Symbol table:

The symbol table is implemented as a binary search tree.

- **`public <T> void add(T key)`**

The function to add a new value in the table is implemented recursively. Starting from the root, it compares the node to be added with the value of the root and calls the same function either for the left child tree, if the added value is smaller than the root, or for the right child tree, if the added value is greater than the root.

The stop condition is when the root of the tree is null and it adds the value as the root of the current tree. This way, the new value is always added as a leaf in the tree on the correct position.

A global variable is maintained for the current value of the code and it increments with each added node.

- **`<T> boolean search(T key)`**

The function to search for a value in the table is implemented recursively. Starting from the root, it compares the searched value with the value of the root and calls the same function either for the left child tree, if the searched value is smaller than the root, or for the right child tree, if the searched value is greater than the root.

The stop condition is when the searched value is equal to the root of the current tree, returning true, or when the root of the current tree is null, meaning that the algorithm reached the bottom of the tree and did not find the value on the path that it was supposed to be on, returning false.

Program Internal Form:

The program internal form is implemented as a list of pairs between a String and a Pair of Integer and Integer.

- **`List<Pair<String, Pair<Integer, Integer>>>`**

The outer key is the name of the token and the value is another Pair, which has the code of the token from the tokens.in file and the position of the token inside the symbol table.