Laboratory 2

GitHub link: https://github.com/andrei-dragan/FLCD

```
#pragma once
#include <string>

class SymbolTable {
private:

  /*
    A node inside our BST is going to have:
      - a node pointer to the left child (NULL if it doesn't have one)
      - a node pointer to the right child (NULL if it doesn't have one)
      - the key of the node which represents the identifier / constant given as a string
      - the id of a certain key, which represents the unique identifier associated to the identifier / constant
  */

  struct Node {
    Node* left;
    Node* right;
    std::string key;
    int id;

    // here we just have a constructer for our node structure
    Node(std::string p_key, int p_id) : key(p_key), id(p_id) {
      left = NULL;
      right = NULL;
    };
  };


  int uniqueId; // this keeps count of the next available unique id to be associated within a class to the next added identifier / constant
  Node* root; // we also keep the root of our BST

  // this function allocates on the heap a new node with the given key (identifier / constant) and maps it to the next available unique ID
  Node* createNewNode(std::string key);

public:
  // constructer
  SymbolTable();

  /*
    We want to add a key (identifier / constant) to the BST.

    We start from the root and we go from node to node in the following way:
      - if the key we want to add is strictly smaller than the key of the current node we go to the left child of the current node
      - if the key we want to add is strictly greater than the key of the current node we go to the right child of the current node
      - if the key we want to add is equal with the key of the current node, then it means the key was already added so we just stop and do
        because we don't want to add the same key again

      - if at some point we want to go to the left child / right child but we can't since it is NULL, we create a new node with the given k
        link it inside our BST


    In the end we return true if we succesffuly managed to add the new node (or to find it) and false otherwise if for some reason the abov
  */
  bool add(std::string key);

  /*
    We want to search a key (identifier / constant).

    We start from the root and we go from node to node in the following way:
      - if the key we want to add is strictly smaller than the key of the current node we go to the left child of the current node
      - if the key we want to add is strictly greater than the key of the current node we go to the right child of the current node
      - if the key we want to add is equal with the key of the current node, then we found it and return true

    If at some point we want to go to the left child / right child but we can't since it is NULL, it means our searched key is not inside t
  */
  bool search(std::string key);
```

```
  /*
    We want to get the ID associated to the key (identifier / constant).

    We start from the root and we go from node to node in the following way:
      - if the key we want to add is strictly smaller than the key of the current node we go to the left child of the current node
      - if the key we want to add is strictly greater than the key of the current node we go to the right child of the current node
      - if the key we want to add is equal with the key of the current node, then we found it and return the associated ID

    If at some point we want to go to the left child / right child but we can't since it is NULL, it means our searched key is not inside t
  */
  int getId(std::string key);
};
```