```cpp
#ifndef LEXER_H
#define LEXER_H

#include <sstream>
#include <vector>
#include <iostream>
#include <unordered_set>
#include <iomanip>

class Lexer
{
public:
  enum State
  {
    NS = 0, // NULL STATE
    S01,    // ACCEPTABLE ID
    S02,    // ACCEPTABLE ID
    S03,
    S04, // ACCEPTABLE INT
    S05,
    S06, // ACCEPTABLE REAL
    S07,
    S08,
    S09,
    S10,
    S11, // ACCEPTABLE '$$'
    S12,
    S13,
    S14,
    TRM // TERMINATING
  };

  enum TransitionType
  {
    IDENTIFIER = 0,
    INTEGER,
    REAL,
    CARROT,
    EQUALS,
    GREATERTHAN,
    LESSTHAN,
    PLUS,
    MINUS,
    MULTIPLY,
    DIVIDE,
    SEPARATOR,
    FUNC_SEPARATOR,
    REJECT
  };

  // State table
  int stateTable[18][14] = {{S01, S04, TRM, S10, S12, S14, S14, S14, S14, S14, S14, S07, S08, TRM}, // INITIAL STATE
                            {S02, S03, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM}, // ACCEPTABLE ID
                            {S02, S03, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM}, // ACCEPTABLE ID
                            {S02, S03, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM},
                            {TRM, S04, S05, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM}, // ACCEPTABLE INT
                            {TRM, S06, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM},
                            {TRM, S06, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM}, // ACCEPTABLE REAL
                            {TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM}, // ACCEPTABLE SEPARATOR
                            {TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, S09, TRM},
                            {TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM
, TRM, TRM, TRM},  // ACCEPTABLE '$$'
                            {TRM, TRM, TRM, TRM, S11, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM},
                            {TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM}, // ACCEPTABLE "^="
                            {TRM, TRM, TRM, TRM, S13, S13, S13, TRM, TRM, TRM, TRM, TRM, TRM, TRM}, // ACCEPTABLE "="
                            {TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM}, // ACCEPTABLE DOUBLE OP
                            {TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM}, // ACCEPTABLE SINGLE OF
                            {TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM, TRM}}; // TERMINATING

  std::unordered_set<std::string> keywords = {"while", "whileend", "int", "function", "if", "ifend", "return", "get", "put", "true", "false", "boolean", "real", "else"};
  std::unordered_set<char> separators = {'(', ')', '{', '}', ',', ':', ';'};

  struct Token
  {
    Token(std::string token, std::string lexeme, int lineNumber)
    {
      this->token = token;
      this->lexeme = lexeme;
      this->lineNumber = lineNumber;
    }

    std::string token;
    std::string lexeme;
    int lineNumber;
  };

  // Constructor
  Lexer();

  // Destructor
  ~Lexer();

  std::vector<Token> lex(std::stringstream &buffer, int lineNumber);

private:
  bool comment;

  int getTransition(char tokenChar) const;

  std::string stateToString(int state) const;

  bool isValidSeparator(char c) const;

  bool isKeyword(std::string token) const;
};

#endif // LEXER_H
```