Alunos: Luis Alan, Luis Vinicius, Marlon Vitor, Matheus Moutinho

Easy-Script

Salvador, BA

2024

Alunos: Luis Alan, Luis Vinicius, Marlon Vitor, Matheus Moutinho

Easy-Script

Trabalho apresentado na Universidade Salvador, orientado pelo professor Cleber Brito Santos.

Salvador, BA

2024

1. **Código fonte**

```
import fs from 'fs';
import chalk from 'chalk';
import vm from 'vm';
import { lexicalAnalyzer } from './analysis/lexical/index.js';
import { syntaxAnalyzer } from './analysis/syntatic/index.js';
import { semanticAnalyzer } from './analysis/semantic/index.js';
import { generateJavaScript } from './generation/index.js';

// Read the code from code.txt
fs.readFile('./src/code.txt', 'utf8', (err, code) => {
  if (err) {
    console.error('Error reading file:', err);
    return;
  }

  // Print the source code
  console.log(`${chalk.magenta('Source Code Read:')}\n${code}\n`);

  // Run the lexical analysis and print the tokens
  const tokens = lexicalAnalyzer(code);
  console.log(`${chalk.magenta('Tokens Generated:')}\n`, tokens);

  // Run the syntactic analysis and print the syntactic tree
  const tree = syntaxAnalyzer(tokens);
  console.log(`${chalk.magenta('\n\nSyntactic Tree:')}`);
  printAST(tree);

  // Run the semantic analysis and print the variable scope table
  const variableScope = semanticAnalyzer(tree);
```

```
  console.log(`${chalk.magenta('\n\nVariable Scope Table:')}\n`, variableScope);

  // Run the JavaScript generation and print the code
  const javaScript = generateJavaScript(tree);
  console.log(
    `${chalk.magenta('\n\nGenerated JavaScript Code:')}\n`,
    javaScript,
  );

  // Run JavaScript code
  const jsCode = javaScript;
  console.log(`${chalk.magenta('\n\nOutput:')}`);
  vm.runInNewContext(jsCode, { console });
});

// Function to print the AST
const printAST = (node: any, depth: number = 0): void => {
  const indent = '│   '.repeat(depth);
  const nodeLabel = node.value ? `${node.type}: ${node.value}` : `${node.type}`;
  console.log(`${indent}├── ${nodeLabel}`);

  if (node.children && node.children.length > 0) {
    node.children.forEach((child: any) => printAST(child, depth + 1));
  }
};
```

2. Aplicação executável (GitHub)

https://github.com/luisvinicius403/EasyScript

3. Códigos teste

```
variable string nome = "João";
variable integer idade = 20;
variable integer contador = 0;

while (nome != "João") {
  write("Usuário não cadastrado: " + nome);
};

if (idade >= 18) {
  write("João é maior de idade.");
} else {
  write("João é menor de idade.");
};

do {
  write("Tentativa número: " + contador);
  contador = contador + 1;
} while (contador < 3);
```

4. Backus-Naur form (Easy-Script)

<program> ::= <declarations> (<command> ";")*

<declarations> ::= <variable_declaration> ";" | <variable_declaration> <declarations>

<variable_declaration> ::= "variable" <type> <identifier> "=" <value>

<type> ::= "integer" | "decimal" | "string"

<identifier> ::= <letter> (<letter> | <number>)*

<letter> ::= "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"

<value> ::= <integer_value> | <decimal_value> | <string_value>

<integer_value> ::= <digit>+

<decimal_value> ::= <digit>+ "." <digit>+

<string_value> ::= """ <character>* """

<digit> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

<character> ::= <letter> | <digit> | " " | "." | "," | "!" | "?" | ...

<conditional_declaration> ::= "if" "(" <condition> ")" "{" (<command> ";")* "}" ";"
| "if" "(" <condition> ")" "{" (<command> ";")* "}" <else_option> ";"

<else_option> ::= "else" "{" (<command> ";")* "}" ";"

<loop_declaration> ::= <while_loop> | <do_while_loop>

<while_loop> ::= "while" "(" <condition> ")" "{" (<command> ";")* "}" ";"

<do_while_loop> ::= "do" "{" (<command> ";")* "}" "while" "(" <condition> ")" ";"

<condition> ::= <expression> <relational_operator> <expression>

<expression> ::= <term> {("+" | "-") <term>}

<term> ::= <factor> {("*" | "/") <factor>}

<factor> ::= <identifier> | <integer_value> | <decimal_value> | "(" <expression> ")"

<relational_operator> ::= "==" | "!=" | "<" | ">" | "<=" | ">="

<command> ::= <variable_declaration>
| <conditional_declaration>
| <loop_declaration>
| <assignment>
| <read>
| <write>

<assignment> ::= <identifier> "=" <expression>

<read> ::= "read" "(" <identifier> ")"

<write> ::= "write" "(" (<value> | <expression>) ")"

5. Analises Léxica, Sintática e Semântica

.src/analysis/lexical/index.js
.src/analysis/syntatic /index.js
.src/analysis/semantic/index.js

6. Equivalências

```
<variable_declaration> ::= "variable" <type> <identifier> "=" <value>
variable string nome = "João";
variable integer idade = 20;
variable integer contador = 0;
```

```
<while_loop> ::= "while" "(" <condition> ")" "{" (<command> ";")* "}" ";"
while (nome != "João") {
  write("Usuário não cadastrado: " + nome);
};
```

```
<conditional_declaration> ::= "if" "(" <condition> ")" "{" (<command> ";")* "}" ";"
                            | "if" "(" <condition> ")" "{" (<command> ";")* "}" <else_option> ";"
if (idade >= 18) {
write("João é maior de idade.");
```

```
<else_option> ::= "else" "{" (<command> ";")* "}" ";"
} else {
write("João é menor de idade.");
```

```
<do_while_loop> ::= "do" "{" (<command> ";")* "}" "while" "(" <condition> ")" ";"
do {
  write("Tentativa número: " + contador);
  contador = contador + 1;
} while (contador < 3);
```