Relatório Trabalho I Analisador Léxico

Matheus Doretto Compri 7151885
Thaís Emanuele dos Santos 6453087
Vinícius Alvarenga Lovato 7696455

Introdução:

Neste primeiro trabalho prático foi modelado e implementado o analisador léxico da linguagem LALG (Pascal simplificado), com tratamentos de erro.

Especificação do analisador Léxico

Para o tratamento de erro foram usadas as seguintes expressões regulares:

1. [a-zA-Z][_a-zA-Z0-9]{30,} no qual identifica o erro ERROR_LONG_ID de identificador muito longo
2. .[1-9][0-9]{10,} no qual identifica o erro ERROR_LONG_NUMBER de número muito longo
3. [a-zA-Z]+([^a-zA-Z0-9_\d\n\t\{\}\:\;\,\.\+\-*\\/\(\)\=\+\-\"\'\>\<]+[a-zA-Z0-9_]*)* no qual identifica o erro ERROR_INVALID_IDENTIFIER de identificador que possui erro na sua formação ou caracter inválido
4. [0-9]+?(".")[0-9]*([^0-9\t\{\}\:\;\,\.\+\-*\\/\(\)\=\+\-\"\'\>\<]+[0-9]*)* no qual identifica o erro ERROR_INVALID_NUMBER de número mal formatado
5. [^a-zA-Z0-9\d\n\t\{\}\:\;\,\.\+\-*\\/\(\)\=\+\-\"\'\>\<]+ no qual identifica o erro ERROR_NON_ALPHANUMERIC de caracter não válido

Os *tokens* decididos para o analisador léxico foram os seguintes:

```
1.  [a-zA-Z][_a-zA-Z0-9]*  return IDENTIFIER;
2.  [0-9]*                 return INTEGER;
3.  [0-9]*"."[0-9]+        return REAL;
4.  \'.\'                  return CHAR;
5.  \".*\"                 return STRING;
6.  [\ \t\n]               ;
7.  ":"                    return COLON;
8.  ";"                    return SEMICOLON;
9.  ","                    return COMMA;
10. "."                    return PERIOD;
11. "+"                    return ADD;
12. "-"                    return SUB;
13. "*"                    return MUL;
14. "/"                    return DIV;
15. "="                    return EQUAL;
16. "<>"                   return N_EQUAL;
17. ">"                    return G_THAN;
```

```
18.  "<"          return L_THAN;
19.  ">="         return GE_THAN;
20.  "<="         return LE_THAN;
21.  ":="         return ASSIGNMENT;
22.  "("          return LEFT_P;
23.  ")"          return RIGHT_P;
24.  "{".*"}"     ;
```

Decisões de Projeto

O analisador léxico é a parte do compilador que leva mais tempo para ser executada, portanto procuramos desenvolver o trabalho otimizando da melhor forma possível, levando em conta memória e processamento.

Para atingir tal objetivo, optamos por não inserir no arquivo de descrição do analisador sintático (scanner.l) as *keywords*, uma vez que seriam necessárias várias máquinas de estado para analisá-las, diminuindo a performance. Ao invés disso, criamos uma tabela *hash*, que contém todas as *keywords*, e analisamos cada identificador para tentar encontrá-los na tabela *hash*. Caso esse identificador seja encontrado, então ele passa a ser classificado como *keyword* ao invés de identificador. A tabela hash foi construída considerando o valor ASCII de cada caractere menos 97 somados, e tomou-se o resto da divisão por 40.

Passo a passo de compilação

Na pasta do projeto, encontra-se um arquivo chamado *make.sh*, no qual contém os comando necessários para a compilação. Para compila-lo, basta digitar *./make.sh* no diretório do projeto.

Para executar o trabalho é necessário redirecionar para a entrada do programa os arquivos de teste, como exemplo: *./scanner < teste*