Lucas Santos RA: 11045514
Guilherme Dias Belarmino RA: 11099814
Murilo Bolzan Dionisio RA: 11107414

IMPLEMENTAÇÃO DE UM ANALISADOR SINTÁTICO

Atividade II - Compiladores - Prof. Dr. Thiago Covões - UFABC

1. ABORDAGEM

A análise sintática está baseada na análise descendente (ASD), ou seja, parte-se do símbolo inicial da gramática e tenta-se chegar às folhas. Foi utilizado a estratégia ASD preditiva recursiva. Para isso, a gramática utilizada para o analisador precisa estar na forma LL(1) (*Left to right, letftmost derivation*), onde se é utilizado um símbolo a frente para se determinar qual regra utilizar.

2. PRINCIPAIS FUNÇÕES IMPLEMENTADAS

A implementação do analisador sintático foi realizada utilizando a linguagem C. A estrutura dos tokens estão armazenadas em uma estrutura chamada *TokenStruct*, onde são armazenados o lexema, o próprio token, a linha e a coluna correspondente.

As funções implementadas estão baseadas na gramática, isto é, nos seus tokens e suas respectivas transições. Então, o token *"IDENTIFICADOR"* possui uma função relacionada a esse estado tratando as possíveis transições.

Algumas funções auxiliares foram utilizadas para realizar as transições, são elas: *createTokenFromCurrentLine*, que atribui os campos do próximo token da leitura do arquivo que foram gerados pela função *nextToken*. Existe uma função para imprimir os erros relacionados aos tokens, que é chamada de *printError*.

A leitura do arquivo é realizada através da função *fopen*, as linhas do arquivo são obtidas pelo *getline* e a separação pelo delimitador "@" é feito pela função *strtok*.

3. GRAMÁTICA LL(1)

A gramática construída é LL(1) onde não são recursivas à esquerda e para cada não terminal, não existem regras cujo lado direito iniciem com o mesmo terminal.

```
T_FUNC      -> PR_VOID id(PARAM) { INSTRUCAO }
T_FUNC      -> T_NUM id(PARAM) { INSTRUCAO }
T_FUNC      -> PR_BOOL id(PARAM) { INSTRUCAO }
T_FUNC      -> ''
PARAM       -> (PR_INT | PR_DOUBLE | PR_BOOL) id PARAM2
PARAM       -> ''
PARAM2      -> , PARAM3
PARAM2      -> ''
PARAM3      -> (PR_INT | PR_DOUBLE | PR_BOOL) id PARAM2
INSTRUCAO   -> SENTENCA
INSTRUCAO   -> INSTRUCAO2
INSTRUCAO2  -> INSTRUCAO INSTRUCAO
INSTRUCAO2  -> '';
SENTENCA    -> PR_IF (COND) { INSTRUCAO }
SENTENCA    -> PR_IF (COND) { INSTRUCAO } PR_ELSE { INSTRUCAO }
SENTENCA    -> PR_WHILE (COND) { INSTRUCAO }
SENTENCA    -> (PR_INT | PR_DOUBLE | PR_BOOL) id;
SENTENCA    -> PR_INT id OP_IGUAL INT;
SENTENCA    -> PR_DOUBLE id OP_IGUAL FLOAT;
SENTENCA    -> PR_BOOL id OP_IGUAL T_BOOL;
SENTENCA    -> ATTR;
SENTENCA    -> INCREMENT;
SENTENCA    -> PR_RETURN (COND | EXP_MATH | INCREMENT);
COND        -> id
COND        -> INT
COND        -> BOOL
COND        -> ATTR
COND        -> id OP_BOOL id COND2
COND2       -> OP_BOOL COND
COND2       -> ''
ATTR        -> id OP_IGUAL (id | INT) ATTR2
ATTR        -> INCREMENT;
ATTR2       -> OP_NUM (id | INT) ATTR3
ATTR2       -> '';
ATTR3       -> ATTR2
INCREMENT   -> INCR_ID id
INCREMENT   -> id INCR_ID
```