Universidade Estadual do Oeste do Paraná - UNIOESTE
Centro de Ciências Exatas e da Tecnológicas - CCET
Curso de Bacharelado de Ciência da Computação
Docente: Guilherme Galante

Analisador Léxico

Acadêmicos:
Augusto Barella Dal Prá
Gustavo Portela Rautenberg

1. Introdução:

A linguagem Ayrton é uma linguagem de programação estrutural em desenvolvimento, projetada para ser intuitiva e única. Inspirada pelo automobilismo, especialmente pela figura icônica de Ayrton Senna, a linguagem adota uma nomenclatura criativa que faz alusão a termos e conceitos do automobilismo, como por exemplo, seus arquivos são com a extensão ".senna" ou até mesmo sua função main sendo "race".

2. Características
    - Tipos de dados:
        - Inteiro = int;
        - Booleano = boolean;
        - Ponto flutuante = double;
        - Letra = char;
        - String = str;

    - Operadores Aritméticos:
        - Soma = "+";
        - Diferença = "-";
        - Divisão = "/";
        - Multiplicação = "*";
        - Potência = "^";

    - Operadores Lógicos:
        - And = &&;
        - OR = ||;
        - NOT = !;

    - Operadores Relacionais:
        - Menor = "<";
        - Maior = ">";
        - Igual = "=";
        - Diferente = "!=";
        - Menor igual = "<=";
        - Maior igual = ">=";

    - Regras para geração de ID:
        - ID deve começar por uma letra (a-Z);
        - Pode conter o caracter "_";
        - Pode conter número (0-9);

    - Comando Entrada e Saída:
        - Saída = prt("Hello World").
        - Entrada = scn(id_variável).

- Palavras Reservadas:
    - while = circuit(){}
    - for = laps(){}
    - if = pitEntry(){}
    - else = pitExit(){}
    - incremento = overtake;
    - decremento = brake;
    - comentário = @;
    - main = race()

- Estrutura Geral:

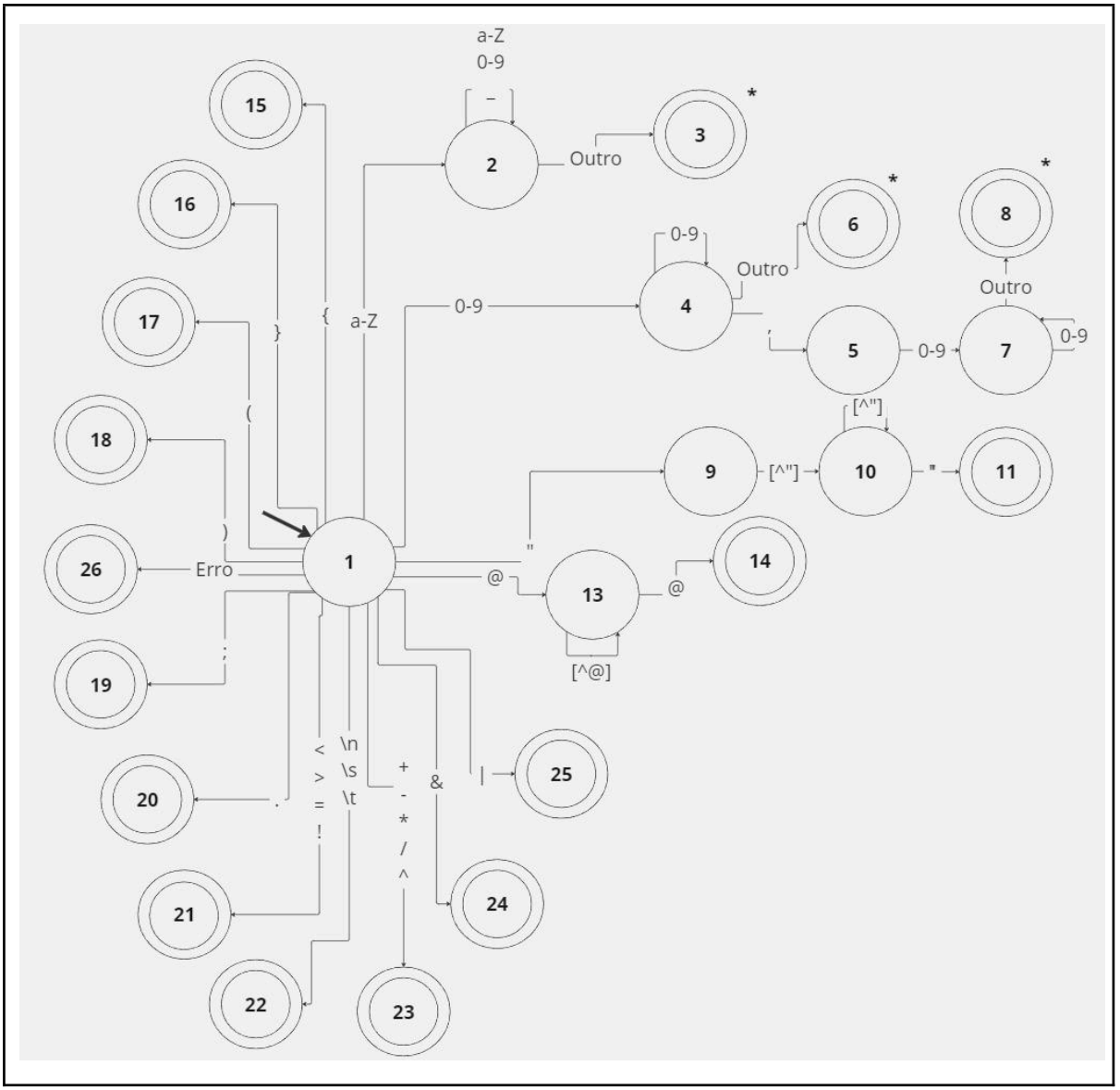
```
int race() {

int exemplo = 0.

laps(int i = 0; i<50; i = overtake) {

exemplo = i.

prt(exemplo).

}

return 0.

}
```

- ER:

```
regex integer("[0-9]+");
regex
double_regex("(\\+|-)?[0-9]+(\\,[0-9]+)?(E(\\+|-)?[0-9]+)?
");
regex char_regex("[a-zA-Z]");
regex digit("[0-9]");

regex symbol_id("[\\_]");
regex symbol_parameter_init("[(]");
regex symbol_parameter_end("[)]");

regex symbol_op_init("[{]");
regex symbol_op_end("[}]");
regex symbol_op_mid("[;]");

regex id("[a-zA-Z]([a-zA-Z0-9]|[_])*");
```

```
regex comma(",");

regex reserved_loops("(laps|circuit)");
regex reserved_condition("(pitEntry|pitExit)");
regex reserved_arit("(overtake|brake)");
regex reserved_main("(race)");
regex reserved_comment("\\@");
regex reserved_types("(int|double|char|boolean|str)");
regex reserved_prt("(prt)");
regex reserved_scn("(scn)");


regex op_log_e("&");
regex op_log_or("\\|");

regex end_line("\\.");

regex op_arit_sum("\\+");
regex op_arit_sub("-");
regex op_arit_mult("\\*");
regex op_arit_div("/");
regex op_arit_pow("\\^");

regex op_rel_minor("<");
regex op_rel_bigger(">");
regex op_rel_equal("=");
regex op_rel_not("\\!");

regex line_feed("(\r\n|\n)");
regex line_feed2("[\r\n]+");

regex spaces("[\\s\t\r\n]+");
regex portuguese("/^[A-Za-záàâãéèêíïóôõöúçñÁÀÂÃÉÈÍÏÓÔÕÖÚÇÑ
]+$/");
regex quotes(R"(\")");

regex all_except_at("[^@]");

regex all_except_quotes(R"([^"]+)");
```

- Autômato:


a-Z
0-9
_
15
2
Outro
3
*
16
6
*
8
*
0-9
Outro
Outro
17
{
a-Z
0-9
4
5
0-9
7
0-9
}
(
[^"]
18
9
[^"]
10
"
11
)
26
Erro
1
"
@
13
@
14
[^@]
;
19
\n
\s
\t
+
-
*
/
^
&
|
25
<
>
=
!
20
.
21
24
22
23