Unioeste - Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

Descrição da Linguagem: Merlin

Marlon F. Pereira, Ronaldo Drecksler
Docente: Guilherme Galante

CASCAVEL - PR
31 de Julho de 2024

1- Tipos:

Os tipos de dados suportados pela linguagem merlin, são **int**, **float**, **string** e **vetor** de int ou float;

- Int: o tipo inteiro será de 32 bits, equivalente ao tipo int da linguagem C;
- Float: terá 32 bits, equivalente a float;
- String: strings são imutáveis, uma vez atribuídas, não podem sofrer alterações;
- Vetor: suportam apenas os tipos int e float e devem ser criados de forma estática no início do programa.

2- Operações:

- Aritméticas: são suportados nativamente as quatro operações básicas, soma, subtração, multiplicação e divisão, os respectivos símbolos seguem abaixo:
  - +
  - -
  - *
  - /
- Lógicas: operações de OU, AND e NOT são suportadas, com os respectivos símbolos;
  - |
  - &
  - !
- Relacionais: igualdade, maior que e menor que;
  - =
  - >
  - <
- Atribuição:
  - :
- Indexação:
  - []

3- Formação de identificadores (regra para nomes de variáveis):

Deve começar com uma letra de a_Z e pode conter zero ou mais caracteres alfanuméricos, sendo case-sensitive;

4- Comando de entrada e saída:

- Entrada: summon()
- Saída: echo()

5- Palavras reservadas:

- Comentários se iniciam com o símbolo “_” e terminam com o new line (“\n”)
- Inicialização: **tome**
- Seção de definição de variáveis: **ingredients**
- As demais palavras reservadas são: **if**, **while**, **for**, **int**, **float**, **string**, **summon** e **echo**;

6- Estruturas de seleção:

- if (), estrutura condicional de divisão;
- while (), estrutura de repetição atrelado a uma condicional;
- for (), estrura de repetição contada, podendo executar 0…n vezes;

8- Exemplo de código na linguagem:

- Blocos: {}, usado para delimitar o código das estruturas de repetição e condicionais, assim a seção de definição de variáveis **ingredients**.
- Fim de linha: ;

```
tome tomename {
    ingredients {
        int numero : 0;
        float naoinicializado;
        int[3] vetorexemplo;
        string texto : "abc";
        int i;
    }

    _ Exemplo de como preencher um vetor
    for (i : 0; i < 3; 1){
        vetorexemplo[i] : 0;
    }

    summon(numero);
    naoinicializado : ~2.102 - numero;
    while(numero < 13 & numero > 0 ) {
        if (numero / 2 = 0) {
            numero : numero / 2;
        }
        if (numero / 2 > 0 | !naoinicializado) {
            numero : numero * 3 + 1;
        }
    }

    echo("numero = ");
    echo(numero);
}
```

9 - Expressões regulares

Tabela 1. Tipos de Tokens x Expressão Regular

Linha	**Token**	**Expressão Regular**
1	ID	[a-Z][a-Z0-9]*
2	INT	"~"?[0-9]$^{+}$
3	FLOAT	"~"?[0-9]$^{+}$"."[0-9]$^{+}$
4	STRING	""“"(~"”")*"”"
5	IGNORE	("_"(~"\n")*"\n")|(" "|"\n"|"\t")$^{+}$
6	TYPE_INT	int
7	TYPE_FLOAT	float
8	TYPE_STRING	string
9	IF	if
10	FOR	for
11	WHILE	while
12	SUMMON	summon
13	ECHO	echo
14	TOME	tome
15	INGREDIENTS	ingredients
16	ENDLINE	;
17	ADD	+
18	SUB	-
19	MUL	*
20	DIV	/
21	OR	|
22	AND	&
23	NOT	!
24	EQ	=

Linha	Token	Expressão Regular
25	LE	<
26	GR	>
27	ATR	:
28	PAR_ESQ	(
29	PAR_DIR	)
30	COL_ESQ	[
31	COL_DIR	]
32	CHV_ESQ	{
33	CHV_DIR	}

10 - Autômato de reconhecimento de tokens

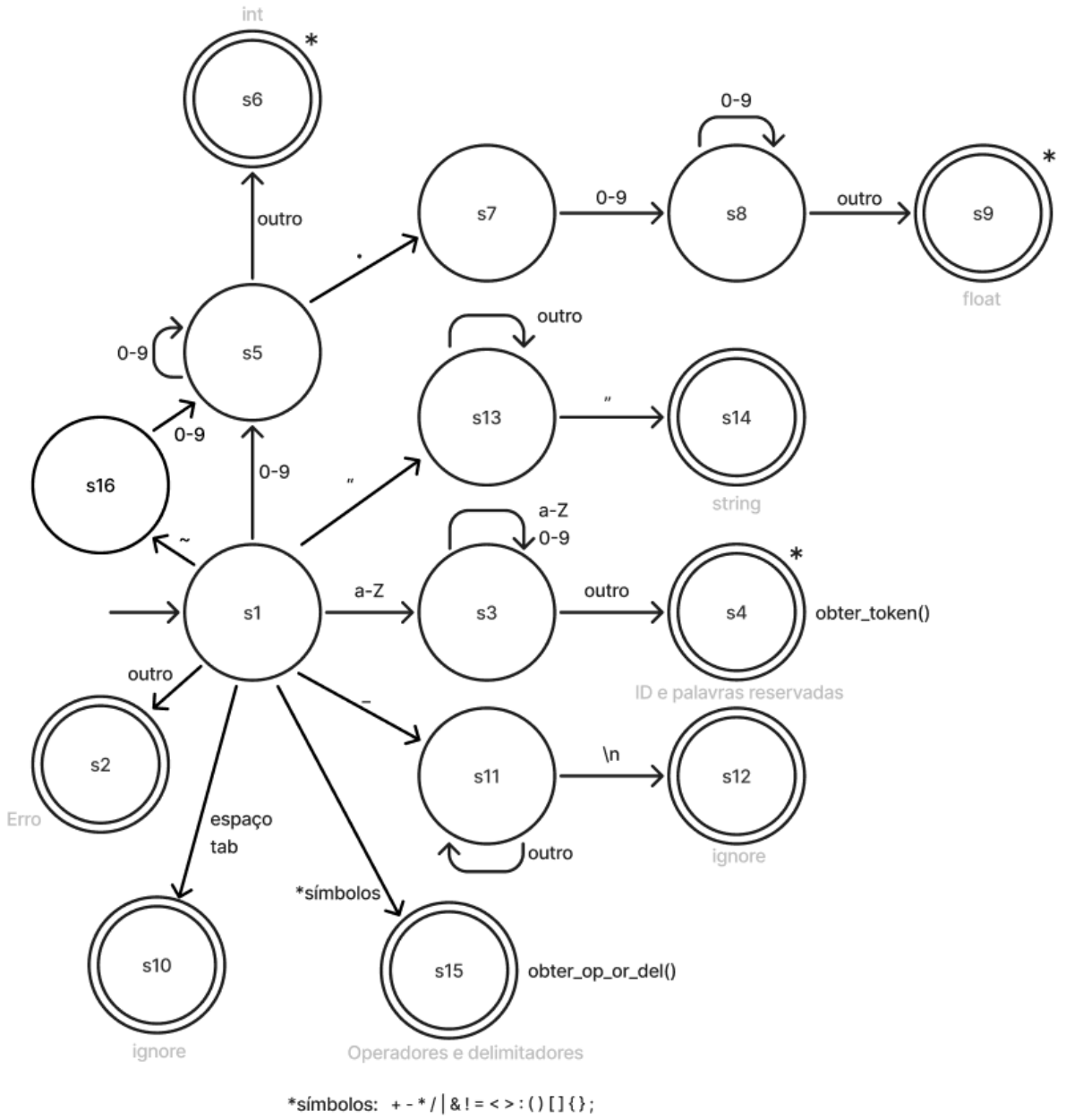

Figura 1. Autômato geral de reconhecimento de token da linguagem Merlin.

O autômato da Figura 1 utiliza duas funções, **obter_token**() e **obter_op_or_del**(), para simplificar sua construção geral. A função **obter_token**() compará cadeias de caracteres obtidas com as palavras reservadas listadas acima. Caso a cadeia de caracteres não corresponda a nenhuma dessas palavras reservadas, ela é declarada como um identificador de variável. Já a função **obter_op_or_del**() é responsável por fazer a ligação entre os símbolos operadores e delimitadores e seus respectivos tipos de tokens, como pode ser visto na Tabela 1, em resumo, ela agrupa a responsabilidade de vários estados finais, em um único.