UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
COLEGIADO DE CIÊNCIA DA COMPUTAÇÃO

Compilador Parte 1 – Compiladores

Título: Requisitos para a especificação/implementação da LP
Aluno: David Antonio Brocardo
Gabriel Tadioto de Oliveira
Data: 31/07/2024

1.1 Requisitos para a especificação/implementação da LP

Nome : CapyScript

Especificar:

1. Tipos de dados suportados (pelo menos 3);
    a. Inteiro; Float; String; Boolean.
2. Operadores suportados (aritméticos, lógicos e relacionais);
    a. Aritméticos:
        i. Soma (“+”); Subtração (“ -”); Divisão (“ / ”); Multiplicação (“ * ”).
    b. Lógicos:
        i. OR; AND; NOT.
    c. Relacionais:
        i. Maior (“ > ”); Menor (“ < ”); Maior igual (“ >= ”); Menor igual (“ <= ”); Igual (“ = ”); Atribuição (“ := ”); Diferente (“ != ”).
3. Formação dos identificadores (regra para nomes de variáveis);
    a. Para declarar uma variável deve fazer da seguinte maneira, “tipo nome ;”
        i. Ex : int valor;
    b. Não poderá ser declarados variáveis com nome igual a palavras reservadas
    c. Não sei o'que mais colocar aqui
4. Comandos de entrada e saída (um para entrada e um para saída);
    a. Entrada
        i. input()
    b. Saída
        i. output(“Hello World” \n);
5. Palavras-reservadas;
    a. or, and, not, for, do, while, int, float, char, boolean, input, output, if, else, OR, AND, NOT, FOR, DO, WHILE, INT, FLOAT, CHAR, BOOLEAN, INPUT, OUTPUT, IF, ELSE

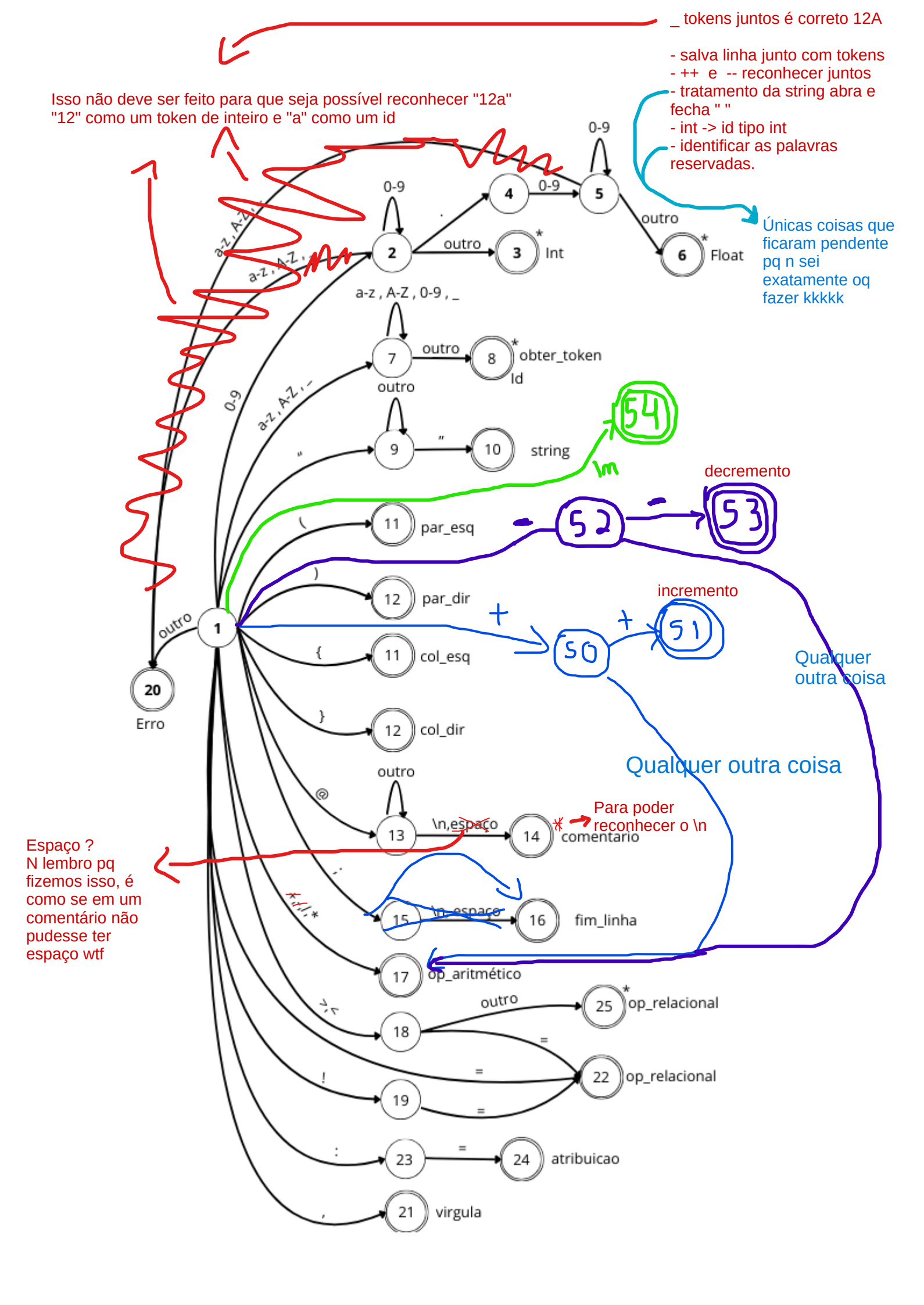
6. Estruturas suportadas pela linguagem (decisão, repetição, etc.) e seus respectivos comandos. Deve ser implementado pelo menos uma estrutura de decisão, um laço de repetição e laço contado (for);
    a. Estrutura de decisão : IF {} ELSE {}
        i. Ex : IF Valor1 > Valor2 {
                comandos
                }
        ii. Parênteses Opcional:
            IF (Valor1 > Valor2) {
                comandos
                }
    b. Laço de Repetição : DO {} While
        i. EX: DO {
                comandos
            } While valor != 0
    c. Laço Contado: FOR {}
        i. Ex: FOR i = 0, i < x , i ++ {
                comandos
            }
7. Estrutura geral do programa (sintaxe) com exemplo.

    Exemplo:

```
int x;
x := input();

int pot;
pot := input();

int mult;
mult := x;

for i := 0, i < pot, i++ {
        x := x * mult;
};

if x > 100 {
        output(“Numero maior que 100”);
} else {

        output(“Numero menor que 100”);
};

output(“Valor :”);
output(x);
```

(2) Especificar as Definições Regulares e os Automata de reconhecimento dos símbolos admitidos pela linguagem proposta (Analisador Léxico);

Token	Lexema	Padrão
id	input	string iniciadas com letra ou _
string	“Valor: %d”	Tudo que estiver entre aspas duplas
int	110	Números inteiros positivos
float	1.2	Números que tenham um ponto entre eles
par_esq	(	“(”
par_dir	)	“)”
virgula	,	“ ,”
ponto_virgula	;	“;”
col_esq	{	“{“
col_dir	}	“}”
atribuicao	:=	“:=”
comentario	@CapyScript	Tudo até encontrar um “\n”
op_aritmetico	+	Os seguintes caracteres: “+”, “-”, “*” e “/”
fim_linha	;	“;”
op_relacional	>=	As seguintes sequências de caracteres: “>=”, “>”, “<”, “<=”, “=” e “!=”


_ tokens juntos é correto 12A
- salva linha junto com tokens
- ++ e -- reconhecer juntos
- tratamento da string abra e fecha " "
- int -> id tipo int
- identificar as palavras reservadas.
Isso não deve ser feito para que seja possível reconhecer "12a" "12" como um token de inteiro e "a" como um id
Únicas coisas que ficaram pendente pq n sei exatamente oq fazer kkkkk
0-9
a-z , A-Z , _
outro
Int
Float
obter_token
Id
string
par_esq
par_dir
col_esq
col_dir
Erro
decremento
incremento
Qualquer outra coisa
Qualquer outra coisa
\n,espaço
comentario
Para poder reconhecer o \n
Espaço ? N lembro pq fizemos isso, é como se em um comentário não pudesse ter espaço wtf
fim_linha
op_aritmético
op_relacional
op_relacional
atribuicao
virgula

Expressões Regulares

int = [0-9]+

float = int (‘.’ int)

id = ([a-z] | [A-Z] | _) ([a-z] | [A-Z] | [0-9] | _)+

string = “(~”)”

par_esq = (

par_dir =)

col_esq = {

col_dir = }

comentario = @(~\n) + \n

fim_linha = ; (\n |’ ‘)

op_aritmetico = (+ | - | * | /)

op_relacional = (>= | > | < | <= | =)

atribuicao = :=

virgula = ,

erro = outro