LINGUAGENS FORMAIS E COMPILADORES – PROJETO COMPILADOR – FASE 01

Estudantes: Mateus Ferro Antunes de Oliveira e Tasi Guilhen Pasin

Bloco de declaração:

Fornecidas:	**Adequadas/Adicionadas:**
program -> block	program -> block
block -> {decls stmts}	block -> { decls stmts }
decls -> decls decl | null	decls -> decls decl | **null**
decl -> type **id** ;	decl -> type id | type id = const
type -> type [num] | basic	type -> type [num] | basic | **void**
stmts -> stmts stmt | null	stmts -> stmts stmt | **null**
	const -> num | factor
	basic -> **int** | **float** | **boolean**

Lexema Basic:

Fornecidas:	**Adequadas/Adicionadas:**
stmt -> loc = bool;	stmt -> loc = bool
| if (bool) stmt	| id (args) { stmt }
| if (bool) stmt else stmt	| id
| while (bool) stmt	| **if** (bool) { stmt }
| do stmt while (bool)	| **if** (bool) { stmt } **elseif** (bool) { stmt } **else** { stmt }
| break	| **if** (bool) { stmt } **else** { stmt }
| block	| **if** (bool) { stmt } **elseif** (bool) { stmt }
loc -> loc [bool] | id	| **while** (bool) { stmt }
	| **for** (loc = num ; bool ; loc = loc +/- num) { stmt }
	| **switch** (id) { cases }
	| **return** id
	| **break**
	| call
	loc -> loc [bool] | id
	id -> letter | id letter | id num
	call -> id (args)
	args -> arglist | null
	arglist -> arglist , bool | bool
	| arglist , num | num
	| arglist , id | id
	cases -> cases caseopt | cases defaultopt | **null**
	caseopt -> **case** bool: stmts
	defaultopt -> **default** : stmts

Regras de Produção:

Fornecidas:	Adequadas/Adicionadas:
bool -> bool || join | join	bool -> bool OR join | join
join -> join && equality | equality	join -> join AND equality | equality
equality -> equality == rel | equality != rel | rel	equality -> equality == rel | equality != rel | rel
rel -> expr < expr | expr <= expr | expr >= expr | expr > expr | expr	rel -> expr < expr | expr <= expr | expr > expr | expr >= expr | expr
expr -> expr + term | expr - term | term	expr -> expr + term | expr - term | term
term -> term * unary | term / unary | unary	term -> term * unary | term / unary | term % unary | unary
unary -> ! unary | - unary | factor	unary -> ! unary | - unary | ++loc | loc++ | --loc | loc-- | factor
factor -> (bool) | loc | num | real | true | false	factor -> bool | loc | num | real | **TRUE** | **FALSE**

Tratamento de números e letras:

Fornecidas:	Adequadas/Adicionadas:
	num -> real
	real -> DIGITS DOT DIGITSF
	DIGITSF -> DIGITS
	| DIGITSF DIGITS
	| e FEXP
	FEXP -> PLUS DIGITSF
	| MINUS DIGITSF
	PLUS -> **+**
	MINUS -> **-**
	e -> e | **null**
	letter -> **[a-zA-Z]**
	DIGITS -> **-?[0-9]+**
	DOT -> **.**

Comunicação direta com hardware:

Fornecidas:	Adequadas/Adicionadas:
	io_input_digital(args)
	io_output_digital(args)
	io_input_analog(args)
	io_output_analog(args)

Exemplos com código:

1:

```
{
	float a123 = 7.2800
	boolean blink = FALSE
	int port = 3

	for (int i = 0; i < a123; i = i + 1) {
		blink = !blink
		io_output_digital(port, blink)
	}
}
```

2:

```
{
	float oper(a, b) {
		float resultado = 0
		if (a > b) {
			resultado = a + b
		} else {
			resultado = a / b
		}
		return resultado
	}

	int a = 10
	int b = 5

	float result = oper(b,a)
}
```

3:

```
{
        float num = 0
        boolean foot = TRUE

        while (1 == 1) {
                switch (foot) {
                        case TRUE:
                                num = 10.0
                        case FALSE:
                                num = 0.0
                        default:
                                num = 5.0
                }

                num = 0
                while (num == 0) {
                        foot = FALSE
                        break
                }
        }
}
```