ESPECIFICAÇÃO DA LINGUAGEM 2021.1

GRAMÁTICA LIVRE DE CONTEXTO
(para implementação do analisador semântico e do gerador de código)

<programa>	::=	<comentário> **program {** <declaração de constantes e variáveis> <corpo do programa> **}** <identificador do programa> **#1**
<comentário>	::=	**:-** *constante literal* | ε
<identificador do programa>	::=	*identificador* **#2** | ε
<declaração de constantes e variáveis>	::=	**define {** <constantes e variáveis> **}** | ε
<constantes e variáveis>	::=	<declaração de constantes> <variáveis''> | <declaração de variáveis> <constantes''>
<declaração de constantes>	::=	**not variable #3** <constantes>
<constantes>	::=	<tipo> **is** <lista de identificadores de constantes> **#4** <valor> **#5 .** <constantes'>
<constantes'>	::=	<constantes> | ε
<constantes''>	::=	<declaração de constantes> | ε
<declaração de variáveis>	::=	**variable #6** <variáveis>
<variáveis>	::=	<tipo> **is** <lista de identificadores de variáveis> **#4 .** <variáveis'>
<variáveis'>	::=	<variáveis> | ε
<variáveis''>	::=	<declaração de variáveis> | ε
<tipo>	::=	**natural #7 | real #8** | **char #9** | **boolean #10**
<lista de identificadores de constantes>	::=	*identificador* **#11** <lista de identificadores de constantes'>
<lista de identificadores de constantes'>	::=	**,** <lista de identificadores de constantes> | ε
<lista de identificadores de variáveis>	::=	*identificador* **#12** <índice> **#13** <lista de identificadores de variáveis'>
<lista de identificadores de variáveis'>	::=	**,** <lista de identificadores de variáveis> | ε
<índice>	::=	**[** *constante inteira* **#14]** | ε
<valor>	::=	*constante inteira* | *constante real* | *constante literal*
<corpo do programa>	::=	**execute {** <lista de comandos> **}**
<lista de comandos>	::=	<comando> **.** <lista de comandos'>
<lista de comandos'>	::=	<lista de comandos> | ε
<comando>	::=	<atribuição> | <entrada> | <saída> | <seleção> | <repetição>
<atribuição>	::=	**set #15** <expressão> **to** <lista de identificadores de variáveis> **#16**
<entrada>	::=	**get #17 {** <lista de identificadores de variáveis> **}**
<saída>	::=	**put {** <lista de identificadores e/ou constantes> **}**
<lista de identificadores e/ou constantes>	::=	<item> **#18** <lista de identificadores e/ou constantes'>
<lista de identificadores e/ou constantes'>	::=	**,** <lista de identificadores e/ou constantes> | ε
<item>	::=	*identificador* **#19** <índice> **#20** | *constante inteira* **#21**

		| *constante real* **#22**
		| *constante literal* **#23**
<seleção>	::=	**verify** <expressão> **is** <cláusulas> **#24**
<cláusulas>	::=	**true #25 {** <lista de comandos> **}** <falsa>
		| **false #26 {** <lista de comandos> **}** <verdadeira>
<verdadeira>	::=	**is true #27 {** <lista de comandos> **}** | ε
<falsa>	::=	**is false #27 {** <lista de comandos> **}** | ε
<repetição>	::=	**loop #28 {** <lista de comandos> **} while** <expressão> **#29 is true**
		| **while #30** <expressão> **#31 is true do {** <lista de comandos> **} #32**
<expressão>	::=	<expressão aritmética ou lógica> <expressão'>
<expressão'>	::=	**==** <expressão aritmética ou lógica> **#33**
		| **!=** <expressão aritmética ou lógica> **#34**
		| **<** <expressão aritmética ou lógica> **#35**
		| **>** <expressão aritmética ou lógica> **#36**
		| **<=** <expressão aritmética ou lógica> **#37**
		| **>=** <expressão aritmética ou lógica> **#38**
		| **ε**
<expressão aritmética ou lógica>	::=	<termo2> <menor prioridade>
<menor prioridade>	::=	**+** <termo2> <menor prioridade> **#39**
		| **-** <termo2> <menor prioridade> **#40**
		| | <termo2> <menor prioridade> **#41**
		| **ε**
<termo2>	::=	<termo1> <media prioridade>
<media prioridade>	::=	***** <termo1> <media prioridade> **#42**
		| **/** <termo1> <media prioridade> **#43**
		| **%** <termo1> <media prioridade> **#44**
		| **%%** <termo1> <media prioridade> **#45**
		| **&** <termo1> <media prioridade> **#46**
		| **ε**
<termo1>	::=	<elemento> <maior prioridade>
<maior prioridade>	::=	****** <elemento> <maior prioridade> **#47**
		| **ε**
<elemento>	::=	*identificador* **#19** <índice> **#20**
		| *constante inteira* **#21**
		| *constante real* **#22**
		| *constante literal* **#23**
		| **true #48**
		| **false #49**
		| **(** <expressão> **)**
		| **! (** <expressão> **) #50**

DESCRIÇÃO DAS AÇÕES SEMÂNTICAS

Para executar a análise semântica e a geração de código é necessário fazer uso de algumas variáveis, quais sejam:

- **contexto** : situação onde foi encontrada uma lista de identificadores, ou seja, na declaração de constantes (contexto = "*constante*"), na declaração de variáveis (contexto = "*variável*"), em um comando de atribuição (contexto = "*atribuição*") ou em um comando de entrada de dados (contexto = "*entrada dados*");
- **VT** ← 0 : contador para número total de constantes e de variáveis;
- **VP** ← 0 : contador para número de constantes ou variáveis de um determinado tipo;
- **VIT** ← 0 : contador para o tamanho das variáveis indexadas de um determinado tipo;
- **tipo** : indica um determinado tipo de constante ou variável, sendo 1 para variável do tipo **natural**, 2 para variável do tipo **real**, 3 para variável do tipo **char**, 4 para variável do tipo **boolean**, 5 para constante compatível com o tipo **natural**, 6 para constante compatível com o tipo **real** e 7 para constante compatível com o tipo **char**;
- **ponteiro** ← 1 : indicador da posição onde será gerada a próxima instrução na área de instruções;
- **variável indexada** : valor lógico que indica ou não a ocorrência de uma variável indexada;
- **pilha de desvios** : pilha de endereços para resolução de desvios com operandos inicialmente desconhecidos, quando da análise dos comandos de seleção e de repetição;
- **tabela de símbolos**
- **área de instruções**

ação #1: *reconhecimento de fim de programa*
gerar instrução: (**ponteiro**, STP, 0)

ação #2: *reconhecimento do identificador de programa*
inserir na **tabela de símbolos** a tupla (*identificador*, 0, **-**, **-**)

ação #3: *reconhecimento da palavra reservada not variable*
contexto ← "*constante*"
VIT ← 0

ação #4: *reconhecimento do término da declaração de constantes ou variáveis de um determinado tipo*

VP ← **VP** + **VIT**

ESCOLHA **tipo**
1, 5: gerar instrução: (**ponteiro**, ALI, **VP**)
ponteiro ← **ponteiro** + 1

2, 6: gerar instrução: (**ponteiro**, ALR, **VP**)
ponteiro ← **ponteiro** + 1

3, 7: gerar instrução: (**ponteiro**, ALS, **VP**)
ponteiro ← **ponteiro** + 1

4: gerar instrução: (**ponteiro**, ALB, **VP**)
ponteiro ← **ponteiro** + 1

SE **tipo** = 1, 2, 3 ou 4 ENTÃO
VP ← **0**
VIT ← **0**
FIMSE

ação #5: *reconhecimento de valor na declaração de constante*
ESCOLHA **tipo**
5: gerar instrução: (**ponteiro**, LDI, *valor*)
ponteiro ← **ponteiro** + 1

6: gerar instrução: (**ponteiro**, LDR, *valor*)
ponteiro ← **ponteiro** + 1

7: gerar instrução: (**ponteiro**, LDS, *valor*)
ponteiro ← **ponteiro** + 1

onde *valor* corresponde ao valor reconhecido

gerar instrução: (**ponteiro**, STC, **VP**)
ponteiro ← **ponteiro** + 1

VP ← **0**

ação #6: *reconhecimento da palavra reservada variable*
contexto ← "*variável*"

ação #7: *reconhecimento da palavra reservada natural*
SE **contexto** = "*variável*" ENTÃO
tipo ← **1** (variável do tipo inteiro)
SENÃO
tipo ← **5** (constante do tipo inteiro)
FIM SE

ação #8: *reconhecimento da palavra reservada real*
SE **contexto** = "*variável*" ENTÃO
tipo ← **2** (variável do tipo real)
SENÃO
tipo ← **6** (constante do tipo real)
FIM SE

ação #9: *reconhecimento da palavra reservada char*
SE **contexto** = "*variável*" ENTÃO
tipo ← **3** (variável do tipo literal)
SENÃO
tipo ← **7** (constante do tipo literal)
FIM SE

ação #10: *reconhecimento da palavra reservada boolean*
SE **contexto** = "*variável*" ENTÃO
tipo ← **4** (variável do tipo lógico)
SENÃO
erro: "*tipo inválido para constante*"
FIM SE

ação #11: *reconhecimento de identificador de constante*
SE (*identificador* existe na **tabela de símbolos**) ENTÃO
erro: "*identificador já declarado*"
SENÃO
VT ← **VT** + 1
VP ← **VP** + 1
inserir na **tabela de símbolos** a tupla: (*identificador*, **tipo**, **VT, -**)
FIMSE

ação #12: *reconhecimento de identificador de variável*
SE **contexto** = "*variável*" ENTÃO
SE (*identificador* existe na **tabela de símbolos**) ENTÃO
erro: "*identificador já declarado*"
SENÃO
variável indexada ← falso
armazenar o *identificador* reconhecido
FIMSE
SENÃO
variável indexada ← falso
armazenar o *identificador* reconhecido
FIMSE

ação #13: *reconhecimento de identificador de variável e tamanho da variável indexada*
ESCOLHA **contexto**

"*variável*" :
SE (**variável indexada** = falso) ENTÃO
VT ← **VT** + 1
VP ← **VP** + 1
inserir na **tabela de símbolos** a tupla: (*identificador*, **tipo**, **VT**, **-**), onde identificador é aquele armazenado na **ação #11**
SENÃO
VIT ← **VIT** + *constante inteira*, onde *constante inteira* é aquela armazenada na **ação #14**
inserir na **tabela de símbolos** a tupla: (*identificador*, **tipo**, **VT + 1**, *constante inteira*), onde identificador é aquele armazenado na **ação #12** e *constante inteira* é aquela armazenada na **ação #14**

VT ← **VT** + *constante inteira*, onde *constante inteira* é aquela armazenada na **ação #14**
FIMSE

"atribuição" :
para o *identificador* armazenado na **ação #12**
SE (*identificador* existe na **tabela de símbolos**) E (*identificador* é identificador de variável) ENTÃO
recuperar da **tabela de símbolos** os "atributo 1" e "atributo 2" correspondentes ao *identificador* reconhecido

SE ("atributo 2" = "**-**") ENTÃO
SE (**variável indexada** = falso) ENTÃO
armazenar o "atributo 1" em uma lista de atributos
SENÃO
erro: "*identificador de variável não indexada*"
FIMSE
SENÃO
SE (**variável indexada** = verdadeiro) ENTÃO
armazenar o "atributo 1" + *constante inteira* – 1 em uma lista de atributos, onde *constante inteira* é aquela armazenada na **ação #14**
SENÃO
erro: "*identificador de variável indexada exige índice*"
FIMSE
FIMSE
SENÃO
erro: "*identificador não declarado ou de constante*"
FIMSE

"*entrada dados*" :
para o *identificador* armazenado na **ação #12**
SE (*identificador* existe na **tabela de símbolos**) E (*identificador* é identificador de variável) ENTÃO
recuperar da **tabela de símbolos** os "atributo 1" e "atributo 2" correspondentes ao *identificador* reconhecido
SE ("atributo 2" = "**-**") ENTÃO
SE (**variável indexada** = falso) ENTÃO
recuperar da **tabela de símbolos** a "categoria" correspondente ao *identificador* reconhecido
gerar instrução: (**ponteiro**, REA, "categoria")
ponteiro ← **ponteiro** + 1
gerar instrução: (**ponteiro**, STR, "atributo 1")
ponteiro ← **ponteiro** + 1

SENÃO
erro: "*identificador de variável não indexada*"
FIMSE
SENÃO
SE (**variável indexada** = verdadeiro) ENTÃO
recuperar da **tabela de símbolos** a "categoria" correspondente ao *identificador* reconhecido
gerar instrução: (**ponteiro**, REA, "categoria")
ponteiro ← **ponteiro** + 1
gerar instrução: (**ponteiro**, STR, "atributo 1" + *constante inteira* – 1), onde *constante inteira* é aquela armazenada na **ação #14**
ponteiro ← **ponteiro** + 1
SENÃO
erro: "*identificador de variável indexada exige índice*"
FIMSE
FIMSE
SENÃO
erro: "*identificador não declarado ou de constante*"
FIMSE

ação #14: *reconhecimento de constante inteira como tamanho da variável indexada ou como índice*
armazenar a *constante inteira* reconhecida
variável indexada ← verdadeiro

ação #15: *reconhecimento do início do comando de atribuição*
contexto ← "*atribuição*"

ação #16: *reconhecimento do fim do comando de atribuição*
gerar instrução: (**ponteiro**, STR, "atributo"), para cada atributo armazenado na lista de atributos pela **ação #13**
ponteiro ← **ponteiro** + 1, para cada instrução STR gerada

(o valor do topo NÃO deverá ser decrementado para cada instrução STR gerada, exceto para a última)

ação #17: *reconhecimento do comando de entrada de dados*
contexto ← "*entrada dados*"

ação #18: *reconhecimento de mensagem em comando de saída de dados*
gerar instrução: (**ponteiro**, WRT, 0)
ponteiro ← **ponteiro** + 1

ação #19: *reconhecimento de identificador em comando de saída ou em expressão*
SE (*identificador* existe na **tabela de símbolos**) E (*identificador* é identificador de constante ou de variável)
ENTÃO
    variável indexada ← falso
    armazenar o *identificador* reconhecido
SENÃO
    erro: "*identificador não declarado*"
FIMSE

ação #20: *reconhecimento de índice de variável indexada em comando de saída*
recuperar da **tabela de símbolos** os "atributo 1" e "atributo 2" correspondentes ao *identificador* armazenado na **ação #19**

SE (**variável indexada** = falso) ENTÃO
    SE ("atributo 2" = "**-**") ENTÃO
        gerar instrução: (**ponteiro**, LDV, "atributo 1")
        ponteiro ← **ponteiro** + 1
    SENÃO
        erro: "*identificador de variável indexada exige índice*"
    FIMSE
SENÃO
    SE ("atributo 2" ≠ "**-**") ENTÃO
        gerar instrução: (**ponteiro**, LDV, "atributo 1" + *constante inteira* – 1), onde *constante inteira* é aquela armazenada na **ação #14**
        ponteiro ← **ponteiro** + 1
    SENÃO
        erro: "*identificador de constante ou de variável não indexada*"
    FIMSE
FIMSE

ação #21: *reconhecimento de constante inteira em comando de saída ou em expressão*
gerar instrução: (**ponteiro**, LDI, *constante inteira*)
ponteiro ← **ponteiro** + 1

ação #22: *reconhecimento de constante real em comando de saída ou em expressão*
gerar instrução: (**ponteiro**, LDR, *constante real*)
ponteiro ← **ponteiro** + 1

ação #23: *reconhecimento de constante literal em comando de saída ou em expressão*
gerar instrução: (**ponteiro**, LDS, *constante literal*)
ponteiro ← **ponteiro** + 1

ação #24: *reconhecimento de fim de comando de seleção*
desempilhar da **pilha de desvios** o endereço da instrução de desvio empilhado na **ação #25** (ou **#26** ou **#27**)
atualizar a instrução de desvio com: endereço ← **ponteiro**

ação #25: *reconhecimento da palavra reservada true*
gerar instrução: (**ponteiro**, JMF, ?), onde endereço = ?
ponteiro ← **ponteiro** + 1
empilhar (**ponteiro** - 1) na **pilha de desvios**, ou seja, o endereço da instrução JMF

ação #26: *reconhecimento da palavra reservada false*
gerar instrução: (**ponteiro**, JMT, ?), onde endereço = ?
ponteiro ← **ponteiro** + 1
empilhar (**ponteiro** - 1) na **pilha de desvios**, ou seja, o endereço da instrução JMT

ação #27: *reconhecimento da palavra reservada false (ou true)*
desempilhar da **pilha de desvios** o endereço da instrução de desvio empilhado na **ação #25** (ou **#26**)
atualizar a instrução de desvio com: endereço ← **ponteiro** + 1
gerar instrução: (**ponteiro**, JMP, ?), onde endereço = ?

ponteiro ← **ponteiro** + 1
empilhar (**ponteiro** - 1) em **pilha de desvios**, ou seja, o endereço da instrução JMP

ação #28: *reconhecimento do comando de repetição*
empilhar (**ponteiro**) na **pilha de desvios**, ou seja, o endereço onde inicia o comando *de repetição*

ação #29: *reconhecimento do fim do comando de repetição*
desempilhar da **pilha de desvios** o endereço da instrução empilhado na **ação #28**
gerar instrução: (**ponteiro**, JMT, “endereço”), onde "endereço" é igual ao valor desempilhado
ponteiro ← **ponteiro** + 1

ação #30: *reconhecimento do início de expressão em comando de repetição*
empilhar (**ponteiro**) na **pilha de desvios**, ou seja, o endereço onde inicia a expressão do comando *de repetição*

ação #31: *reconhecimento de expressão em comando de repetição*
gerar instrução: (**ponteiro**, JMF, ?), onde endereço = ?
ponteiro ← **ponteiro** + 1
empilhar (**ponteiro** - 1) na **pilha de desvios**, ou seja, o endereço da instrução JMF

ação #32: *reconhecimento do fim do comando de repetição*
desempilhar da **pilha de desvios** o endereço da instrução de desvio empilhado na **ação #31**
atualizar a instrução de desvio com: endereço ← **ponteiro** + 1
desempilhar da **pilha de desvios** o endereço da instrução empilhado na **ação #30**
gerar instrução: (**ponteiro**, JMP, “endereço”), onde "endereço" é igual ao valor desempilhado
ponteiro ← **ponteiro** + 1

ação #33: *reconhecimento de operação relacional igual*
gerar instrução: (**ponteiro**, EQL, 0)
ponteiro ← **ponteiro** + 1

ação #34: *reconhecimento de operação relacional diferente*
gerar instrução: (**ponteiro**, DIF, 0)
ponteiro ← **ponteiro** + 1

ação #35 *reconhecimento de operação relacional menor*
gerar instrução: (**ponteiro**, SMR, 0)
ponteiro ← **ponteiro** + 1

ação #36: *reconhecimento de operação relacional maior*
gerar instrução: (**ponteiro**, BGR, 0)
ponteiro ← **ponteiro** + 1

ação #37: *reconhecimento de operação relacional menor igual*
gerar instrução: (**ponteiro**, SME, 0)
ponteiro ← **ponteiro** + 1

ação #38: *reconhecimento de operação relacional maior igual*
gerar instrução: (**ponteiro**, BGE, 0)
ponteiro ← **ponteiro** + 1

ação #39: *reconhecimento de operação aritmética adição*
gerar instrução: (**ponteiro**, ADD, 0)
ponteiro ← **ponteiro** + 1

ação #40: *reconhecimento de operação aritmética subtração*
gerar instrução: (**ponteiro**, SUB, 0)
ponteiro ← **ponteiro** + 1

ação #41: *reconhecimento de operação lógica OU (|)*
gerar instrução: (**ponteiro**, OR, 0)
ponteiro ← **ponteiro** + 1

ação #42: *reconhecimento de operação aritmética multiplicação*
gerar instrução: (**ponteiro**, MUL, 0)
ponteiro ← **ponteiro** + 1

ação #43: *reconhecimento de operação aritmética divisão real*
gerar instrução: (**ponteiro**, DIV, 0)

ponteiro ← **ponteiro** + 1

ação #44: *reconhecimento de operação aritmética divisão inteira*
especificar

ação #45: *reconhecimento de operação aritmética resto da divisão inteira*
especificar

ação #46: *reconhecimento de operação lógica E (&)*
gerar instrução: (**ponteiro**, AND, 0)
ponteiro ← **ponteiro** + 1

ação #47: *reconhecimento de operação aritmética potenciação*
especificar

ação #48: *reconhecimento de constante lógica true*
gerar instrução: (**ponteiro**, LDB, TRUE)
ponteiro ← **ponteiro** + 1

ação #49: *reconhecimento de constante lógica false*
gerar instrução: (**ponteiro**, LDB, FALSE)
ponteiro ← **ponteiro** + 1

ação #50: *reconhecimento de operação lógica NÃO (!)*
gerar instrução: (**ponteiro**, NOT, 0)
ponteiro ← **ponteiro** + 1

---

instrução STC - armazenar o conteúdo do topo da pilha de dados na últimas (deslocamento) constantes alocadas

instrução: **STC, deslocamento**
PARA i DE (topo - deslocamento) ATÉ (topo - 1)
FAÇA pilha [i]:= pilha[topo]
FIMPARA
topo ← topo - 1
ponteiro ← ponteiro + 1

TABELA DE SÍMBOLOS

A estrutura da **tabela de símbolos** será

identificador	categoria	atributo 1	atributo 2

identificador → identificador do programa, de constante ou de variável
categoria → **0** para identificador de programa
1, **2**, **3** ou **4** para variáveis inteiras, reais, literais ou lógicas, respectivamente
5, **6** ou **7** para constantes inteiras, reais ou literais, respectivamente
atributo 1 → deslocamento na pilha de dados (pilha de execução)
atributo 2 → para variáveis indexadas será o tamanho (comprimento) da variável e para as demais variáveis ou constantes será "**-**" (hífen)