ESPECIFICAÇÃO DA LINGUAGEM 2024.2

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

<programa>	::=	**make** <identificador do programa> <declaração de constantes e/ou variáveis> <lista de comandos> **end . #1**
<identificador do programa>	::=	*identificador* **#2** | ε
<declaração de constantes e/ou variáveis>	::=	<constantes e variáveis> | ε
<constantes e variáveis>	::=	<declaração de constantes> <variáveis''> |<declaração de variáveis> <constantes''>
<declaração de constantes>	::=	**const #3** <constantes> **end ;**
<constantes>	::=	<tipo> **:** <lista de identificadores> **#4 =** <valor> **#5 .** <constantes'>
<constantes'>	::=	<constantes> | ε
<constantes''>	::=	<declaração de constantes> | ε
<declaração de variáveis>	::=	**var #6** <variáveis> **end ;**
<variáveis>	::=	<tipo> **:** <lista de identificadores> **#4 .** <variáveis'>
<variáveis'>	::=	<variáveis> | ε
<variáveis''>	::=	<declaração de variáveis> | ε
<tipo>	::=	**int #7 | real #8 | char #9 | bool #10**
<lista de identificadores>	::=	*identificador* **#11**<lista de identificadores'>
<lista de identificadores'>	::=	**,** <lista de identificadores> | ε
<lista de comandos>	::=	<comando> **.** <lista de comandos'>
<lista de comandos'>	::=	<lista de comandos> | ε
<comando>	::=	<atribuição> | <entrada> | <saída> | <seleção> | <repetição>
<atribuição>	::=	<expressão> **->** *identificador* **#12**
<entrada>	::=	**get #13 (** <lista de identificadores> **)**
<saída>	::=	**put (** <lista de identificadores e/ou constantes> **)**
<lista de identificadores e/ou constantes>	::=	<item> **#14** <lista de identificadores e/ou constantes'>
<lista de identificadores e/ou constantes'>	::=	**,** <lista de identificadores e/ou constantes> | ε
<item>	::=	*identificador* **#15** | *constante inteira* **#16** | *constante real* **#17** | *constante literal* **#18** | **true #19** | **false #20**
<seleção>	::=	**if** <expressão> **#21 then** <lista de comandos> <senão> **end #22**
<senão>	::=	**else #23** <lista de comandos> | ε
<repetição>	::=	**while #24** <expressão> **#25 do** <lista de comandos> **end #26**
<expressão>	::=	<expressão aritmética ou lógica> <expressão'>
<expressão'>	::=	**=** <expressão aritmética ou lógica> **#27** | **<>** <expressão aritmética ou lógica> **#28** | **<** <expressão aritmética ou lógica> **#29** | **>** <expressão aritmética ou lógica> **#30** | **<=** <expressão aritmética ou lógica> **#31** | **>=** <expressão aritmética ou lógica> **#32** | **ε**
<expressão aritmética ou lógica>	::=	<termo2> <menor prioridade>
<menor prioridade>	::=	**+** <termo2> **#33** <menor prioridade> | **-** <termo2> **#34** <menor prioridade> | **|** <termo2> **#35** <menor prioridade> | **ε**
<termo2>	::=	<termo1> <media prioridade>
<media prioridade>	::=	***** <termo1> **#36** <media prioridade> | **/** <termo1> **#37** <media prioridade> | **%** <termo1> **#38** <media prioridade> | **%%** <termo1> **#39** <media prioridade> | **&** <termo1> **#40** <media prioridade> | **ε**
<termo1>	::=	<elemento> <maior prioridade>
<maior prioridade>	::=	****** <elemento> **#41** <maior prioridade> | **ε**

<elemento>	::=	*identificador* **#15** | *constante inteira* **#16** | *constante real* **#17** | *constante literal* **#18** | **true #19** | **false #20** | **(** <expressão> **)** | **!(** <expressão> **) #42**

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 = "*variavel*") ou em um comando de entrada de dados (contexto = "*entrada dados*");
- **VT** ← 0 : contador para número total de constantes ou variáveis;
- **VP** ← 0 : contador para número de constantes ou variáveis de um determinado tipo;
- **tipo** : indica um determinado tipo de constante ou variável, sendo 1 para variável do tipo **int**, 2 para variável do tipo **real**, 3 para variável do tipo **char**, 4 para variável do tipo **bool**, 5 para constante compatível com o tipo **int**, 6 para constante compatível com o tipo **real** e 7 para constante compatível com o tipo **char**;
- **pilha de desvios** : pilha de endereços para resolução de desvios com operandos inicialmente desconhecidos, quando da análise dos comandos de seleção e de repetição;
- **área de instruções**
- **ponteiro** ← 1 : indicador da posição onde será gerada a próxima instrução na área de instruções;
- **tabela de símbolos**

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* ***const***
contexto ← "*constante*"

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

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**
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* ***var***
contexto ← "*variavel*"

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

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

ação #11: *reconhecimento de identificador*
ESCOLHA **contexto**
"*constante*" ou "*variavel*" **:**
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

"*entrada dados*" **:**
SE (*identificador* existe na **tabela de símbolos**) ENTÃO
SE (*identificador* é identificador de variável) ENTÃO
recuperar da **tabela de símbolos** a "categoria" correspondente ao *identificador* reconhecido
gerar instrução: (**ponteiro**, REA, "categoria")
ponteiro ← **ponteiro** + 1
recuperar da **tabela de símbolos** o "atributo" correspondente ao *identificador* reconhecido
gerar instrução: (**ponteiro**, STR, "atributo")
ponteiro ← **ponteiro** + 1
SENÃO
erro: "*identificador de programa ou de constante*"
FIMSE
SENÃO
erro: "*identificador não declarado*"
FIMSE

ação #12: *reconhecimento de identificador em comando de atribuição*
SE (*identificador* existe na **tabela de símbolos**) ENTÃO
SE (*identificador* é identificador de variável) ENTÃO
recuperar da **tabela de símbolos** o "atributo correspondente ao *identificador* reconhecido
gerar instrução: (**ponteiro**, STR, "atributo")
ponteiro ← **ponteiro** + 1

SENÃO

erro: "*identificador de programa ou de constante*"
FIMSE
SENÃO
erro: "*identificador não declarado*"
FIMSE

ação #13: *reconhecimento da palavra reservada* ***get***
contexto ← "*entrada dados*"

ação #14: *reconhecimento de mensagem em comando de saída de dados*
gerar instrução: (**ponteiro**, WRT, 0)
ponteiro ← **ponteiro** + 1

ação #15: *reconhecimento de identificador em comando de saída ou em expressão*
SE (*identificador* existe na **tabela de símbolos**) ENTÃO
SE (*identificador* é identificador de constante ou de variável) ENTÃO
recuperar da **tabela de símbolos** o "atributo" correspondente ao *identificador* reconhecido
gerar instrução: (**ponteiro**, LDV, "atributo")
ponteiro ← **ponteiro** + 1
SENÃO
erro: "*identificador de programa*"
FIMSE
SENÃO
erro: "*identificador não declarado*"
FIMSE

ação #16: *reconhecimento de constante inteira em comando de saída ou em expressão*
gerar instrução: (**ponteiro**, LDI, *constante inteira*)
ponteiro ← **ponteiro** + 1

ação #17: *reconhecimento de constante real em comando de saída ou em expressão*
gerar instrução: (**ponteiro**, LDR, *constante real*)
ponteiro ← **ponteiro** + 1

ação #18: *reconhecimento de constante literal em comando de saída ou em expressão*
gerar instrução: (**ponteiro**, LDS, *constante literal*)
ponteiro ← **ponteiro** + 1

ação #19: *reconhecimento de constante lógica verdadeiro*
gerar instrução: (**ponteiro**, LDB, TRUE)
ponteiro ← **ponteiro** + 1

ação #20: *reconhecimento de constante lógica falso*
gerar instrução: (**ponteiro**, LDB, FALSE)
ponteiro ← **ponteiro** + 1

ação #21: *reconhecimento de expressão em comando de seleção*
gerar instrução: (**ponteiro**, JMF, ?), onde endereço = ?
ponteiro ← **ponteiro** + 1
empilhar (**ponteiro** - 1) na **pilha de desvios**, ou seja, o endereço da instrução JMF

ação #22: *reconhecimento do fim de comando de seleção*
desempilhar da **pilha de desvios** o endereço da instrução JMP (ou JMF) empilhado na **ação #23** (ou **ação #21**)
atualizar a instrução de desvio com: endereço ← **ponteiro**

ação #23: *reconhecimento da cláusula senão em comando de seleção*
desempilhar da **pilha de desvios** o endereço da instrução JMF empilhado na **ação #21**
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 #24: *reconhecimento da palavra reservada* ***while***
empilhar (**ponteiro**) na **pilha de desvios**, ou seja, o endereço onde inicia a expressão do comando *de repetição*

ação #25: *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 #26: *reconhecimento do fim do comando de repetição*
desempilhar da **pilha de desvios** o endereço da instrução de desvio empilhado na **ação #25**
atualizar a instrução de desvio com: endereço ← **ponteiro** + 1
desempilhar da **pilha de desvios** o endereço da instrução empilhado na **ação #24**
gerar instrução: (**ponteiro**, JMP, "endereço"), onde "endereço" é igual ao valor desempilhado
ponteiro ← **ponteiro** + 1

ação #27: *reconhecimento de operação relacional igual*
gerar instrução: (**ponteiro**, EQL, 0)
ponteiro ← **ponteiro** + 1

ação #28: *reconhecimento de operação relacional diferente*
gerar instrução: (**ponteiro**, DIF, 0)
ponteiro ← **ponteiro** + 1

ação #29 *reconhecimento de operação relacional menor*
gerar instrução: (**ponteiro**, SMR, 0)
ponteiro ← **ponteiro** + 1

ação #30: *reconhecimento de operação relacional maior*
gerar instrução: (**ponteiro**, BGR, 0)
ponteiro ← **ponteiro** + 1

ação #31: *reconhecimento de operação relacional menor igual*
especificar

ação #32: *reconhecimento de operação relacional maior igual*
especificar

ação #33: *reconhecimento de operação aritmética adição*
gerar instrução: (**ponteiro**, ADD, 0)
ponteiro ← **ponteiro** + 1

ação #34: *reconhecimento de operação aritmética subtração*
gerar instrução: (**ponteiro**, SUB, 0)
ponteiro ← **ponteiro** + 1

ação #35: *reconhecimento de operação lógica OU*
gerar instrução: (**ponteiro**, OR, 0)
ponteiro ← **ponteiro** + 1

ação #36: *reconhecimento de operação aritmética multiplicação*
gerar instrução: (**ponteiro**, MUL, 0)
ponteiro ← **ponteiro** + 1

ação #37: *reconhecimento de operação aritmética divisão real*
gerar instrução: (**ponteiro**, DIV, 0)
ponteiro ← **ponteiro** + 1

ação #38: *reconhecimento de operação aritmética divisão inteira*
especificar

ação #39: *reconhecimento de operação aritmética resto da divisão inteira*
especificar

ação #40: *reconhecimento de operação lógica E*
gerar instrução: (**ponteiro**, AND, 0)
ponteiro ← **ponteiro** + 1

ação #41: *reconhecimento de operação aritmética potenciação*
especificar

ação #42: *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
```