Exame - Lista 1 - CES41 Compiladores

14 de Julho de 2019

Prof. Dr. Fábio Carneiro Mokarzel

Igor Bragaia[1]

[1]Aluno de Graduação em Engenharia do Instituto Tecnológico de Aeronáutica (ITA).

E-mail: igor.bragaia@gmail.com

Questão 1 Implementar num só programa, escrito em Linguagem C ou em alguma linguagem de programação de sua preferência, toda a análise léxica, sintática e semântica do Mini Pascal, fundamentada em diagramas de transições, conforme desenvolvimento apresentado no Capítulo III dos Slides Teóricos de CES-41. Executar o programa e apresentar resultados suficientes para comprovar seu bom funcionamento. O referido desenvolvimento pode ser usado como base para a execução desta questão.

Resolução Dado que a implementação da análise léxica, sintática e semântica independe da linguagem, optei pela implementação em Python dada a simplificação da sintaxe em comparação ao C. O código completo ficou muito extenso e pode ser consultado no repositório do GitHub https://github.com/igorbragaia/CES-41/tree/master/list1/ex1 ou nos arquivos em anexo

- arquivo ex1.py contém a implementação da análise léxica, sintática e semântica
- arquivo ex1_constants.py contém as constantes utilizadas no projeto
- arquivo ex1_estruturas_de_dados_aux.py contém as estruturas de dados auxiliares utilizadas para a tabela de símbolos, como o hash e as diversas funções de manipulação da tabela
- pasta exemplos contém os resultados dos testes apresentados a seguir

A fim de se garantir o correto funcionamento do analisador léxico, observa-se a tabela, coerente em todos os exemplos apresentados a seguir, que classifica todas as strings presentes no programas. Para se garantir o correto funcionamento do analisador sintático, observa-se o programa de input pretty printed. Por fim, para se garantir o correto funcionamento do analisador semântico, observa-se os resultados dos testes semânticos.

Além disso, o log do analisador léxico, sintático e semântico são salvos em variáveis que são acessadas ao final da execução. Por fim, ao final da execução, o programa irá exibir como output os resultados dos analisadores léxico, sintático e semântico. Para demonstrar o correto funcionamento do código, foram testados diversos inputs, sendo que todos os exemplos testados podem ser acessados integralmente na pasta exemplos.

INPUT1 - programa sem nenhuma incoerência

```
PROGRAM fatorial;
VAR n, fat, i: INTEGER;
BEGIN
READ (n);
fat := 1;  i := 1;
WHILE i <= n DO BEGIN
fat := fat * i; i := i + 1
END;
WRITE (‘O fatorial de’, n, ‘ eh ’, fat)
END.
```

OUTPUT1

```
Log do analisador léxico

PROGRAM,    9,
     ID,   16,   fatorial
  PVIRG,   23,
    VAR,   13,
     ID,   16,   n
   VIRG,   26,
     ID,   16,   fat
   VIRG,   26,
     ID,   16,   i
 DPONTS,   25,
INTEGER,    8,
  PVIRG,   23,
  BEGIN,    1,
   READ,   10,
  ABPAR,   27,
     ID,   16,   n
   FPAR,   28,
  PVIRG,   23,
     ID,   16,   fat
  ATRIB,   29,
    CTE,   17,   1
  PVIRG,   23,
     ID,   16,   i
  ATRIB,   29,
    CTE,   17,   1
  PVIRG,   23,
  WHILE,   14,
     ID,   16,   i
  OPREL,   22,   10
     ID,   16,   n
     DO,    3,
  BEGIN,    1,
     ID,   16,   fat
  ATRIB,   29,
     ID,   16,   fat
 OPMULT,   20,   4
     ID,   16,   i
  PVIRG,   23,
     ID,   16,   i
  ATRIB,   29,
     ID,   16,   i
   OPAD,   19,   1
    CTE,   17,   1
    END,    5,
  PVIRG,   23,
  WRITE,   15,
  ABPAR,   27,
 CADEIA,   18,   O fatorial de
   VIRG,   26,
     ID,   16,   n
   VIRG,   26,
 CADEIA,   18,    eh
   VIRG,   26,
     ID,   16,   fat
   FPAR,   28,
    END,    5,
  PONTO,   24,
  PONTO,   24,
```

```
Log do analisador sintático (pretty printer)

PROGRAM fatorial;
VAR n,fat,i: INTEGER;
BEGIN
    READ (n);
    fat := 1;
    i := 1;
    WHILE i<= n DO BEGIN
        fat := fat * i;
        i := i + 1
    END;
    WRITE ("O fatorial de", n, " eh ", fat)
END.

Log do analisador semantico

['fatorial, IDPROG, , inic=False, ref=False',
 'n, IDVAR, INTEIRA, inic=True, ref=True',
 'fat, IDVAR, INTEIRA, inic=True, ref=True',
 'i, IDVAR, INTEIRA, inic=True, ref=True']
```

INPUT2 - programa em que a variável j é declarada mas não é referenciada

```
PROGRAM fatorial;
VAR n, fat, i, j: INTEGER;
BEGIN
READ (n);
fat := 1;  i := 1;
WHILE i <= n DO BEGIN
fat := fat * i; i := i + 1
END;
WRITE (‘O fatorial de’, n, ‘ eh ’, fat)
END.
```

OUTPUT2 - para simplificar, note apenas que o analisador semântico reconhece o símbolo não referenciado. Os resultados da análise sintática e léxica são semelhantes ao exemplo anterior

```
Log do analisador semantico

['fatorial, IDPROG, , inic=False, ref=False',
 'n, IDVAR, INTEIRA, inic=True, ref=True',
 'fat, IDVAR, INTEIRA, inic=True, ref=True',
 'i, IDVAR, INTEIRA, inic=True, ref=True',
 'j, IDVAR, INTEIRA, inic=False, ref=False',
 'Simbolo nao ref ou inic: j']
```

INPUT3 - programa em que a variável k é referenciada mas não foi declarada, além de possuir uma expressão não booleana no while

```
PROGRAM fatorial;
VAR n, fat, i, j: INTEGER;
BEGIN
READ (n);
fat := 1;  i := 1;
WHILE 'texto' DO BEGIN
fat := fat * i; i := i + 1; k := k / 3
END;
WRITE ('O fatorial de', n, ' eh ', fat)
END.
```

OUTPUT3 - para simplificar, note apenas que o analisador semântico reconhece o símbolo k não declarado (além de outros erros decorrentes da não declaração do símbolo k, como o operando incompatível devido à não declaração do k), a expressão não booleana no while e o símbolo não referenciado presente no OUTPUT2

```
Log do analisador semantico

['Expressao nao booleana no while',
 'Não declarado: k',
 'Não declarado: k',
 'Operando incompatível com operador aritmetico multiplicativo',
 'Lados esquerdo e direito incompatíveis num atribuição',
 'fatorial, IDPROG, , inic=False, ref=False',
 'n, IDVAR, INTEIRA, inic=True, ref=True',
 'fat, IDVAR, INTEIRA, inic=True, ref=True',
 'i, IDVAR, INTEIRA, inic=True, ref=True',
 'j, IDVAR, INTEIRA, inic=False, ref=False',
 'Simbolo nao ref ou inic: j']
```

Questão 2 Seja a gramática do exemplo da Seção 5.4.4 do Capítulo V dos Slides Teóricos de CES-41 e a tabela de produções de um analisador preditor não-recursivo, no mesmo exemplo. Fazer uma tabela de execução, no mesmo analisador, simulando a análise da seguinte sentença: **id * ((id + id) * (id * (id + id))) $**

Resolução O algoritmo descrito nos slides teóricos foi implementado em Python e pode ser consultado nos arquivos anexos ou no link https://github.com/igorbragaia/CES-41/tree/master/list1/ex2. O resultado do algoritmo para o exemplo pedido, por sua vez, pode ser consultado nos arquivos anexos ou no link https://github.com/igorbragaia/CES-41/blob/master/list1/ex2/ex2_exercise.csv e encontra-se também abaixo,

Pilha	Entrada	Acao	Producao
$ E	id * ((id + id) * (id * (id + id))) $	Expandir	E->TE'
$ E' T	id * ((id + id) * (id * (id + id))) $	Expandir	T->FT'
$ E' T' F	id * ((id + id) * (id * (id + id))) $	Expandir	F->id
$ E' T' id	id * ((id + id) * (id * (id + id))) $	Desempilhar e avancar	
$ E' T'	* ((id + id) * (id * (id + id))) $	Expandir	T'->*FT'
$ E' T' F *	* ((id + id) * (id * (id + id))) $	Desempilhar e avancar	
$ E' T' F	((id + id) * (id * (id + id))) $	Expandir	F->(E)
$ E' T') E (	((id + id) * (id * (id + id))) $	Desempilhar e avancar	
$ E' T') E	(id + id) * (id * (id + id))) $	Expandir	E->TE'
$ E' T') E' T	(id + id) * (id * (id + id))) $	Expandir	T->FT'
$ E' T') E' T' F	(id + id) * (id * (id + id))) $	Expandir	F->(E)
$ E' T') E' T') E (	(id + id) * (id * (id + id))) $	Desempilhar e avancar	
$ E' T') E' T') E	id + id) * (id * (id + id))) $	Expandir	E->TE'
$ E' T') E' T') E' T	id + id) * (id * (id + id))) $	Expandir	T->FT'
$ E' T') E' T') E' T' F	id + id) * (id * (id + id))) $	Expandir	F->id
$ E' T') E' T') E' T' id	id + id) * (id * (id + id))) $	Desempilhar e avancar	
$ E' T') E' T') E' T'	+ id) * (id * (id + id))) $	Expandir	T'->epsilon
$ E' T') E' T') E'	+ id) * (id * (id + id))) $	Expandir	E'->+TE'
$ E' T') E' T') E' T +	+ id) * (id * (id + id))) $	Desempilhar e avancar	
$ E' T') E' T') E' T	id) * (id * (id + id))) $	Expandir	T->FT'
$ E' T') E' T') E' T' F	id) * (id * (id + id))) $	Expandir	F->id
$ E' T') E' T') E' T' id	id) * (id * (id + id))) $	Desempilhar e avancar	
$ E' T') E' T') E' T'	) * (id * (id + id))) $	Expandir	T'->epsilon
$ E' T') E' T') E'	) * (id * (id + id))) $	Expandir	E'->epsilon

$ E' T') E' T')	) * (id * (id + id))) $	Desempilhar e avancar	
$ E' T') E' T'	* (id * (id + id))) $	Expandir	T'->*FT'
$ E' T') E' T' F *	* (id * (id + id))) $	Desempilhar e avancar	
$ E' T') E' T' F	(id * (id + id))) $	Expandir	F->(E)
$ E' T') E' T') E (	(id * (id + id))) $	Desempilhar e avancar	
$ E' T') E' T') E	id * (id + id))) $	Expandir	E->TE'
$ E' T') E' T') E' T	id * (id + id))) $	Expandir	T->FT'
$ E' T') E' T') E' T' F	id * (id + id))) $	Expandir	F->id
$ E' T') E' T') E' T' id	id * (id + id))) $	Desempilhar e avancar	
$ E' T') E' T') E' T'	* (id + id))) $	Expandir	T'->*FT'
$ E' T') E' T') E' T' F *	* (id + id))) $	Desempilhar e avancar	
$ E' T') E' T') E' T' F	(id + id))) $	Expandir	F->(E)
$ E' T') E' T') E' T') E (	(id + id))) $	Desempilhar e avancar	
$ E' T') E' T') E' T') E	id + id))) $	Expandir	E->TE'
$ E' T') E' T') E' T') E' T	id + id))) $	Expandir	T->FT'
$ E' T') E' T') E' T') E' T' F	id + id))) $	Expandir	F->id
$ E' T') E' T') E' T') E' T' id	id + id))) $	Desempilhar e avancar	
$ E' T') E' T') E' T') E' T'	+ id))) $	Expandir	T'->epsilon
$ E' T') E' T') E' T') E'	+ id))) $	Expandir	E'->+TE'
$ E' T') E' T') E' T') E' T +	+ id))) $	Desempilhar e avancar	
$ E' T') E' T') E' T') E' T	id))) $	Expandir	T->FT'
$ E' T') E' T') E' T') E' T' F	id))) $	Expandir	F->id
$ E' T') E' T') E' T') E' T' id	id))) $	Desempilhar e avancar	
$ E' T') E' T') E' T') E' T'	))) $	Expandir	T'->epsilon
$ E' T') E' T') E' T') E'	))) $	Expandir	E'->epsilon
$ E' T') E' T') E' T')	))) $	Desempilhar e avancar	
$ E' T') E' T') E' T'	)) $	Expandir	T'->epsilon
$ E' T') E' T') E'	)) $	Expandir	E'->epsilon
$ E' T') E' T')	)) $	Desempilhar e avancar	
$ E' T') E' T'	) $	Expandir	T'->epsilon
$ E' T') E'	) $	Expandir	E'->epsilon
$ E' T')	) $	Desempilhar e avancar	

$ E' T'	**$**	**Expandir**	**T'->epsilon**
$ E'	**$**	**Expandir**	**E'->epsilon**
$	**$**	**Desempilhar e avancar**	
		Encerrar com sucesso	

Questão 3 Seja a gramática descrita nas instruções da lista. Construir para ela uma tabela de produções de um analisador sintático preditor não-recursivo. Mostrar o primeiro de cada símbolo, o seguinte de cada não-terminal e o primeiro de cada lado direito das produções. Não é necessário implementar no computador, mas também não é proibido.

Resolução O algoritmo descrito nos slides teóricos foi implementado em Python e pode ser consultado nos arquivos em anexo ou no link https://github.com/igorbragaia/CES-41/blob/master/list1/ex3/ex3.py. Os resultados obtidos encontram-se abaixo

A->alfa	**Prim(alfa)**	**Prim(A)**	**Seg(A)**
Prog->DeclsCmdComp	int real	int real	$
Decls->DeclaracaoLDaux	int real	int real	{
LDaux->eps	eps	eps int real	{
LDaux->Decls	int real	eps int real	{
Declaracao->TipoListId;	int real	int real	int real {
Tipo->int	int	int real	ID
Tipo->real	real	int real	ID
ListId->IDLIaux	ID	ID	;
LIaux->eps	eps	eps ,	;
LIaux->,ListId	,	eps ,	;
CmdComp->{ListCmd}	{	{	$ { ID }
ListCmd->ComandoLCaux	{ ID	{ ID	}
LCaux->eps	eps	eps { ID	}
LCaux->ListCmd	{ ID	eps { ID	}
Comando->CmdComp	{	{ ID	{ ID }
Comando->CmdAtrib	ID	{ ID	{ ID }
CmdAtrib->ID=Expressao;	ID	ID	{ ID }
Expressao->TermoEaux	(ID CTE	(ID CTE	;)
Eaux->eps	eps	eps +	;)
Eaux->+Expressao	+	eps +	;)
Termo->(Expressao)	(	(ID CTE	+ ;)
Termo->ID	ID	(ID CTE	+ ;)
Termo->CTE	CTE	(ID CTE	+ ;)

Tabela de produções (para visualizar em maior resolução, consultar o pdf anexo ou o link https://github.com/igorbragaia/CES-41/blob/master/list1/ex3/tabeladeproducoes.pdf)

	ID	CTE	int	real	+	*	=	;	,	(	)	{	}	$
Prog			Prog->DeclsCmdComp	Prog->DeclsCmdComp										
Decls			Decls->DeclaracaoLDaux	Decls->DeclaracaoLDaux										
LDaux			LDaux->Decls	LDaux->Decls								LDaux->eps		
Declaracao			Declaracao->TipoListId;	Declaracao->TipoListId;										
Tipo			Tipo->int	Tipo->real										
ListId	ListId->IDLlaux													
Llaux								Llaux->eps	Llaux->,ListId					
CmdComp												CmdComp->{ListCmd}		
ListCmd	ListCmd->ComandoLCaux											ListCmd->ComandoLCaux		
LCaux	LCaux->ListCmd											LCaux->ListCmd	LCaux->eps	
Comando	Comando->CmdAtrib											Comando->CmdComp		
CmdAtrib	CmdAtrib->ID=Expressao;													
Expressao	Expressao->TermoEaux	Expressao->TermoEaux								Expressao->TermoEaux				
Eaux					Eaux->+Expressao			Eaux->eps			Eaux->eps			
Termo	Termo->ID	Termo->CTE								Termo->(Expressao)				