PROJETO E ANÁLISE SINTÁTICA DE LINGUAGEM DE PROGRAMAÇÃO EDUCACIONAL

Área de conhecimento (Tabela CNPq): 1.03.01.02-0 Linguagens Formais e Autômatos

Apresentado no
10° Congresso de Inovação, Ciência e Tecnologia do IFSP ou no 4º Congresso de Pós-Graduação do IFSP
27 e 28 de novembro de 2019 — Sorocaba, SP, Brasil

RESUMO: Este projeto tem como propósito projetar e implementar o analisador sintático de uma linguagem de programação educacional aplicando os conceitos de gramáticas regulares, livres-de-contexto, análise léxica e sintática. Tem-se como propósito fundamentar o desenvolvimento de uma linguagem de fácil compreensão lógica, com o intuito de auxiliar no ensino da lógica de programação para indivíduos que ainda não possuem contato com seus conceitos básicos.

PALAVRAS-CHAVE: linguagem de programação; lógica de programação; compiladores; análise sintática;

SPECIFICATION AND SYNTATIC ANALYSIS FOR AN EDUCATIONAL PROGRAMMING LANGUAGE

ABSTRACT: This project aims to specify and implement the parser for an educational programming language, applying the concepts of regular and context-free grammars, lexical and syntactic analisys. Its purpose is to support the development of a language that is easy to understand, in order to assist the teaching of programming logic for individuals who do not yet have contact with its basic concepts.

KEYWORDS: programming language; programming logic; compilers; syntax analisys;

INTRODUÇÃO

Este trabalho em andamento objetiva estabelecer a sintaxe de uma linguagem de programação voltada ao ensino de lógica de programação, sem necessariamente seguir o padrão da linguagem informal PORTUGOL, caso corriqueiro no meio acadêmico. Também objetiva-se construir seu analisador sintático aplicando-se a técnica de Análise Descendente Recursiva (WIRTH, 2017).

O uso atual de linguagens completas e já consolidadas para o ensino da lógica de programação tende a criar dificuldades para o discente iniciante na programação de computadores, uma vez que, faltando-lhe familiaridade com os conceitos fundamentais da lógica de programação—atribuições, operadores, estruturas de desvio de fluxo *etc.*—este é conduzido ao seu aprendizado por meio de linguagens que não são específicas para esta finalidade e cujas propriedades e características de uso são voltadas para o desempenho do aplicativo final e não necessariamente para seu aprendizado. Este fator tende a dificultar tanto a explicação dos conceitos fundamentais de lógica de programação, quanto seu aprendizado (BINDER, 1999).

MATERIAIS E MÉTODOS

Os materiais utilizados no desenvolvimento da pesquisa foram livros, periódicos e publicações de teses para realizar a pesquisa e a definição formal da gramática da linguagem proposta. A partir deste estudo foi possível definir que a linguagem inicialmente será composta pelas seguintes estruturas: declarações de variáveis e constantes, atribuições, estruturas condicionais, estruturas de

laço, entrada e saída de dados. As definições formais dos lexemas e do Formalismo de Backus-Naur Estendido (*Extended Backus-Naur Form* ou EBNF) foram estabelecidas seguindo um padrão inicial previamente acordado entre os envolvidos na pesquisa. As palavras reservadas foram estabelecidas seguindo termos comuns da língua portuguesa, visando reduzir e simplificar a escrita do código-fonte. A tabela 1 mostra algumas das palavras reservadas em suas respectivas categorias. Os operadores seguiram o padrão da linguagem C, porém podem vir a mudar com as necessidades de adaptação da sintaxe final.

Tabela 1 — Algumas das palavras reservadas da linguagem. Não estão sendo apresentadas, palavras para estruturas de desvio de fluxo, conectores *etc.*

Contexto	**Palavras reservadas**			
Definições	`rotina`	`tipo`	`registro`	`enumeração`
Entrada e saída	`leia`	`escreva`		
Outros	`novo`			

Também foram estabelecidos exemplos genéricos de como a sintaxe poderia ser escrita, comparando possibilidades de escrita tanto sintática, quanto lexicalmente. A tabela 2 mostra apenas alguns dos exemplos genéricos criados. A equipe envolvida testou diversas combinações: uso de declaração no modelo Pascal com blocos delimitados com chaves (modelo C), pontuação equivalente ao da língua portuguesa com declaração no modelo C *etc.*

Tabela 2 — Exemplos genéricos da sintaxe proposta.

Declaração no modelo C, pontuação em português	**Declaração no modelo Pascal, pontuação em português**
Pontuação em inglês	Pontuação em inglês
`real i = 0,0`	`i: real = 0,0`
`real i = 0.0`	`i: real = 0.0`

`rotina olá(texto nome)`
`  escreva "Olá "; nome`
`  retorne`
 	
`olá("Alvaro")`
	`rotina olá(nome: texto) {`
	`  escreva "Olá "; nome`
	`}`
	
	`olá("Alvaro")`

`rotina olá(texto nome)`
`  escreva "Olá ", nome`
`  retorne`
 	
`olá("Alvaro")`
	`rotina olá(nome: texto) {`
	`  escreva "Olá ", nome`
	`}`
	
	`olá("Alvaro")`

Por fim, definiu-se que o padrão aplicado de forma preliminar seria a declaração em modelo C como uso de palavra reservada única para a delimitação de blocos.

RESULTADOS E DISCUSSÃO

A partir do levantamento citado na seção anterior, as regras sintáticas em EBNF foram propostas. A implementação seguirá o padrão de Análise Descendente Recursiva a partir das regras estabelecidas, ocorrendo um passo prévio de ajuste para evitar problemas insolúveis com o padrão de análise escolhido (recursão à esquerda, ambiguidade *etc.*). Os lexemas básicos da linguagem foram estabelecidos usando padrão IEEE POSIX 1003.2 (POSIX.2) estendido (IEEE, 1994) para expressões regulares e foram aplicados diretamente nas regras sintáticas. Algumas das regras resultantes encontram-se na listagem 1.

Listagem 1 — Algumas das regras sintáticas em EBNF. Foram suprimidas as definições diversas—índices para vetores, membros de registros *etc.*—bem como as definições para tipos de expressões.

```
// Símbolo não-terminal inicial.
PROGRAMA = BLOCO .

// Cerne do código-fonte.
BLOCO      = {DECLARACAO | COMANDO | ROTULO} .
DECLARACAO = VARIAVEL | ROTINA | REGISTRO | TIPO .
COMANDO    = ATRIBUICAO | CHAMADA | CONDICIONAL | ENQUANTO | PARA | ENTRADA | SAIDA .
ROTULO     = id ":" nl .

// Declarações.
VARIAVEL  = BASE_TIPO VARIAVEL_ITEM {";" VARIAVEL_ITEM} nl .
ROTINA    = "rotina" id [PARAMETROS] [":" BASE_TIPO] nl BLOCO "retorne" [EXPR] nl .
REGISTRO  = "registro" id ["(" LISTA_ID ")"] nl {VARIAVEL} "siga" nl .
TIPO      = "tipo" id "define" (BASE_TIPO | ASSINATURA) nl .

// Comandos.
ATRIBUICAO  = id {INDICES | MEMBRO} "=" EXPR nl .
CHAMADA     = id [ARGUMENTOS] nl .
CONDICIONAL = "se" EXPR nl BLOCO {"caso" EXPR nl BLOCO} ["senão" nl BLOCO] SALTO nl .
ENQUANTO    = "enquanto" EXPR nl BLOCO "volte" nl .
PARA        = "para" id "=" EXPR_ARIT ".." EXPR_ARIT nl BLOCO "volte" nl .
ENTRADA     = "leia" [LISTA_ID] nl .
SAIDA       = "escreva" EXPR {";" EXPR} nl .
```

CONCLUSÕES

Com a definição formal da gramática da linguagem, obtendo como resultado principal as regras sintáticas da linguagem de programação em EBNF, concluímos o projeto inicial da linguagem que será utilizado para implementação da análise sintática por meio de um Analisador Descendente Recursivo. Este trabalho ainda encontra-se em desenvolvimento e será finalizado até o final do ano corrente.

REFERÊNCIAS

BINDER, Fabio Vinicius. **CONCEITOS E FERRAMENTAS PARA APOIAR O ENSINO DE LÓGICA DE PROGRAMAÇÃO IMPERATIVA** . (Pós graduação em informática)- Universidade Federal do Paraná, Curitiba, 1999. Disponível em: <https://acervodigital.ufpr.br/bitstream/handle/1884/24713/D%20-%20BINDER,%20FABIO%20VINICIUS.pdf?sequence=1&isAllowed=y>. Acesso em: 01 maio 2018.

WIRTH, Niklaus. **COMPILER CONSTRUCTION**. Zurich, 2017. Disponível em: <http://www.ethoberon.ethz.ch/WirthPubl/CBEAll.pdf>. Acesso em: 06 março 2019

IEEE Computer Society. **STANDARDS INTERPRETATIONS FOR IEEE STD 1003.2**. Institute Of Electrical And Electronics Engineers. New York, 1994. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6880751>. Acesso em 18 junho 2019.