Analisador Sintático – Uma Ferramenta Didática para Auxílio ao Ensino-Aprendizagem de Algoritimos

Andrezza R. Filizzola da Silva[1], Jacqueline S. Moura[2], Thiago Bezerra Lima e Silva[3]

[1] Professora MSc do curso de Bacharelado em Sistemas de Informação, UPIS–União Pioneira de Integração Social, 61- 445-6702
[2] Aluna de Bacharelado em Sistemas de Informação, UPIS – União Pioneira de Integração Social, 61-445-6702
[3] Aluno de Bacharelado em Sistemas de Informação, UPIS – União Pioneira de Integração Social, 61-445-6702

andrezza@upis.br, jacqueline31015@upis.br, jbls@brturbo.com

Resumo. *Este trabalho é resultado de um estudo sobre a utilização de ferramentas didáticas no processo de ensino-aprendizagem, onde se apresenta uma ferramenta para auxílio ao ensino-aprendizagem de algoritmos, implementada em linguagem C e object-pascal, cuja função corresponde à análise de algoritmos descritos em linguagem corrente com base em uma sintaxe própria, estabelecida com o objetivo de agregar maior consistência a linguagem portugol, definindo uma linguagem para a mesma, garantir a organização do algoritmo, corrigir erros relevantes para a clareza lógica do algoritmo do aluno. Por esta ferramenta ser desenvolvida com estas duas linguagens é possível visualizar o resultado da aplicação, tanto em ambiente DOS, como em padrão Windows.*
Palavras-chave - Ensino-aprendizagem, Lógica de programação, Algoritmo, Softwares didáticos, Ferramentas didáticas.

Abstract. *This work is the result of the studying about the use of didactics softwares in the teaching-learning process, which presents a software to help the students in the algorithmic construction task, where the student write the algorithmic using a current language and the software analyze the structure through its own syntax.*
Keywords - teaching-learning process, programming logic, algorithmic learning, didactics softwares.

1. Introdução

Quando um estudante está sendo iniciado na área de programação, é muito comum que haja dificuldades quando à construção de algoritmos, principalmente quando efetuado sem o apoio de um software que analise e informe ao estudante se seu algoritmo está correto ou não.

Diante de um software didático desse tipo, pode-se perceber vantagens tanto ao professor, que ganha um aliado ao Ensinar, quanto ao aluno, a quem cabe o interesse de aprender o assunto, que recebe o apoio da ferramenta como incentivo ao estudar, uma vez que aprender praticando é uma forma de adquirir o conhecimento e outras experiências para o desenvolvimento de determinada tarefa. [1] [2].

A prática dos assuntos em estudo, além de grande incentivador ao aluno, é de fundamental importância profissional, principalmente, na área de informática. Portanto, é relevante que o professor faça um planejamento de suas aulas antes de iniciá-las [3], prevendo como um dos métodos de ensino-aprendizagem a utilização de ferramentas de auxílio didático, que proporcionem a prática dos assuntos a serem ministrados, facilitando assim a abstração dos mesmo, conforme a proposta da ferramenta apresentada neste trabalho.

Quanto à abstração dos conceitos, em cursos de informática ou computação, o ensino-aprendizagem de algoritmos traz bastantes desafios, haja vista que o algoritmo trata-se de uma seqüência lógica de fatos, cada um deles denominado como uma proposição no estudo da lógica matemática, que pode assumir os valores falso ou verdadeiro e ser simples ou composta. Quando composta, usa conectivo, formando assim uma seqüência lógica com o grau de complexidade diretamente relacionado ao número de conectivos, que é armazenada pelo computador para que um algoritmo tenha uma forma organizada e não perca os dados que serão manipulados no decorrer de seu processamento [4].

2. Ensino-aprendizagem de Algoritmo

Com as linguagens de programação, os algoritmos, podem ser definidos pela descrição da aparência de seus programas, que é a sintaxe da linguagem e do que os mesmos significam, e pela semântica da linguagem. Para a construção de algoritmos são usados: comandos, entradas e saídas de dados, além de estruturas como as de seleção e de repetição, as quais podem ser encadeadas, heterogêneas ou homogêneas.

Essa mesma idéia pode ser aproveitada através da ferramenta deste trabalho, que permite a construção de um algoritmo, escrito em linguagem corrente estruturada, proporcionando, além da edição, a análise da estrutura do mesmo através da ferramenta. Tudo isso, tornou-se possível mediante a existência de regras que estabeleçam uma estruturação pré-definida de comandos a serem utilizados no algoritmo, com o objetivo de organizar e padronizar a escrita do mesmo dentro da ferramenta, implementada em linguagem C.

Assim, apresenta-se um software didático de apoio ao estudo de algoritmos, que consiste em analisar e apontar erros de um algoritmo descrito em qualquer arquivo texto ou na própria interface do Ambiente Didático Integrado criado a partir através da ferramenta Delphi e linguagem object-pascal, proposto para ser usado com o Analisador sintático, seguindo regras propostas para descrição do mesmo.

2.1. Apresentação do Analisador Sintático

O programa está dividido em seis módulos: principal, léxico, sintático, erros, exibição e tela de símbolos.

O módulo principal é responsável por exibir informações gerais relevantes do software e chamar os demais módulos e a rotina principal do analisador sintático. O módulo léxico reúne um conjunto de funções para reconhecer cadeias de caracteres, ao ser iniciado, ele abre o arquivo passado como parâmetro para o programa e armazena seu texto em um buffer, aguardando. Então, pela demanda dos tokens feita pelo módulo sintático, caso encontre um erro, ele informa ao módulo de erros.

A tarefa principal do módulo sintático, é validar o algoritmo pelas regras da gramática da ferramenta, faz-se um pedido para o módulo léxico para que retorne o próximo token do texto, a árvore gramatical é percorrida para que as comparações possam ser feitas a partir das funções contidas neste módulo.

O módulo de erros é responsável por armazenar todos os erros encontrados e o módulo de exibição, mostra em forma de texto, os erros cometidos pelo usuário. E finalmente o módulo da tabela de símbolos é responsável por manter informações referentes aos identificadores do

algoritmo como lexema, tipo, argumentos, caso seja um procedimento e tipo de retorno caso seja uma função.

Com essa estrutura, os módulos podem ser reutilizados para outras funções. Devido a centralização das tarefas, pode-se facilmente incorporar extensões como outros módulos de análise sintática para outras linguagens ou uma reportagem de erros mais precisa e explicativa.

2.1.1. Regras usadas ao estabelecimento da Sintaxe

A seguir, são apresentadas as definições sintáticas, isso é, a ordem e disposição dos símbolos terminais. As palavras que estão em negrito são palavras chave da ferramenta.

algoritmo = decl_algoritmo bloco_decl_var principal $

decl_algoritmo = **algorítmo** identificador **;**

bloco_decl_var = **variáveis** decl_var **fim-variáveis**

decl_var = (**constante**? identificador decl_var_mais_id **:** tipo_dado **;**)+

decl_var_mais_id = **,** identificador decl_var_mais_id
| Є

tipo_dado = **inteiro**
| **real**
| **caractere**
| **literal**
| **lógico**

principal = **início** enunciados **fim**

enunciados = atribuição_enun enunciados
| se_enun enunciados
| enquanto_enun enunciados
| para_enun enunciados
| Є

atribuição = identificador **:=** expressão **;**

se_enun = **se** expressão **então** enunciados se_opc **fim-se**

se_opc = **senão** enunciados
| Є

enquanto_enun = **enquanto** expressão **faça** enunciados **fim-enquanto**

para_enun = **para** identificador **de** expressão **até** expressão **faça** enunciados **fim-para**

expressão = expr_ou

expr_ou = expr_e expr_ou_ex

expr_ou_ex = **ou** expr_ou
| Є
expr_e = expr_bit_ou expr_e_ex

expr_e_ex = **e** expr_e
| Є

xpr_bit_ou = expr_bit_xou expr_bit_ou_ex

xpr_bit_ou_ex = | expr_bit_ou
| Є

xpr_bit_xou = expr_bit_e expr_bit_xou_ex

expr_bit_xou_ex = ^ expr_bit_xou
| Є

expr_bit_e = expr_igual expr_bit_e_ex

expr_bit_e_ex = **&** expr_bit_e
| Є

expr_igual : expr_relacional expr_igual_ex

expr_igual_ex = = expr_igual
| <> expr_igual
| Є

expr_relacional = expr_ad expr_relacional_ex

expr_relacional_ex = < expr_relacional
| <= expr_relacional
| > expr_relacional
| >= expr_relacional
| Є

expr_ad = expr_multip expr_ad_ex

expr_ad_ex = + expr_ad
| - expr_ad

expr_multip = expr_unário expr_multip_ex=

expr_multip_ex = / <expr_multip>
| * <expr_multip>
| Є

expr_unário = **-** expr_elemeno
| + expr_elemento
| **não** expr_elemento
| ~ expr_elemento
| expr_elemento

expr_elemento = identificador (**(** expressão? (**,** expressão)* **)**)?
| constante
| **(** expressão **)**

2.1.2. O Funcionamento do Analisador Sintático

Um grande benefício dessa ferramenta, é que ela não faz parte da interface gráfica, podendo ser usada por ela ou não, por isso é portável para outras plataformas, como o UNIX.

Para iniciar uma análise, executa-se o programa passando como parâmetro o nome do arquivo que contém o algoritmo. Na plataforma MS-Windows/DOS isso é feito na linha de comando:

c:\prototipo> pparser.exe teste.txt

Onde c:\prototipo é uma pasta de exemplo que deve conter o programa pparser.exe (ou tê-lo na variável de ambiente PATH) e o arquivo algoritmo.txt. A figura 1 mostra um algoritmo que foi escrito em arquivo texto.

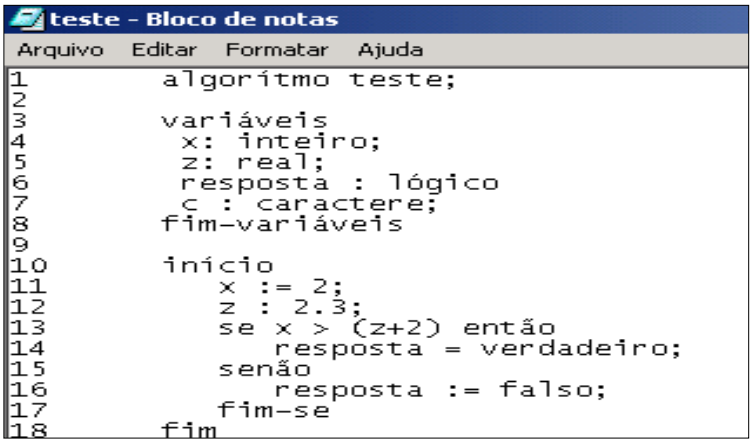

Fig.1. Algoritmo com erros

A figura 2 mostra as linhas onde os erros foram encontrados, como esta ferramenta analisa o algoritmo tendo como base a sintaxe e semântica estabelecidas para uma liguagem portugol, o erro estará na linha anterior a indicada no algoritmo, conforme explicado no item 2.1.1.

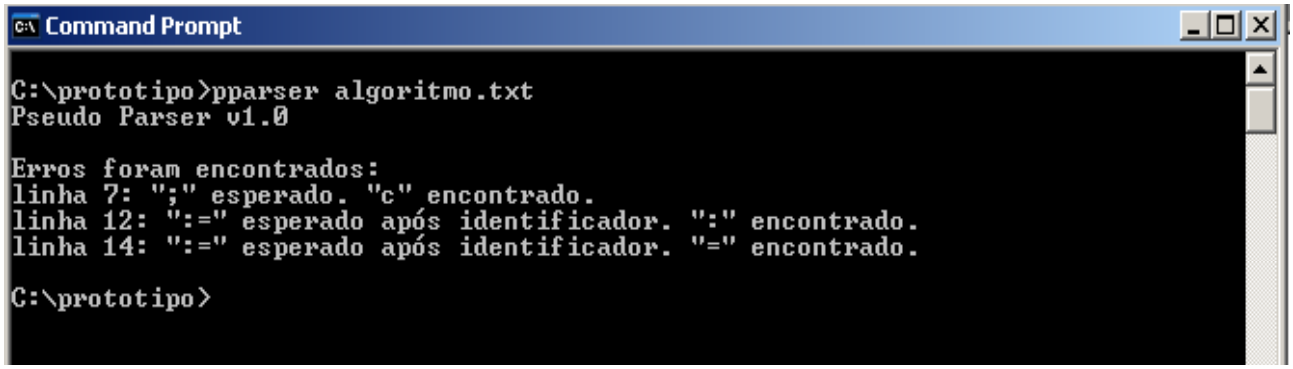

Fig. 2. Correção dos erros do algoritmo.

Para um melhor entendimento de como esta ferramenta funciona o que acontece na primeira linha de indicação de um erro é a sinalização da falta de ponto e vírgula na declaração da variável “resposta”. Isso é, foi encontrado “c” ao invés de “;”, uma vez que “c” é o próximo lexema após “lógico”.

linha 6 resposta : lógico
linha 7 c : caractere;

2.3.1 A Interface Gráfica em Conjunto com o Analisador Sintático

O *design* do ambiente ADI, foi influenciado pelos IDEs (Ambientes de desenvolvimento integrado) da Borland como o Delphi e C++ Builder, O programa é composto de uma janela principal pequena contendo o menu e botões de atalho que refletem o padrão de programas semelhantes no que diz respeito a edição de texto. Além desses botões, o mais importante é o botão de execução da análise sintática, que executa o Analisador sintático feito como processo filho em um *pipe,* representado pelo pequeno foguete. Isso é visualizado na figura a seguir:

Fig.3. Janela principal da Interface Gráfica

Para que o ADI suporte análise sintática, um recurso chamado *pipe* foi utilizado. Esse recurso permite que a saída de um programa seja passado como entrada para outro programa. Isso foi feito através de funções da biblioteca do sistema operacional MS Windows. Com esse recurso, é possível que outros softwares sejam integrados ao ADI, porém, de forma independente, isso é, os programas não compartilham do mesmo código e mesmo binário. A seguir é apresentado o algoritmo anterior mostrado nesta nova tela de visualização.

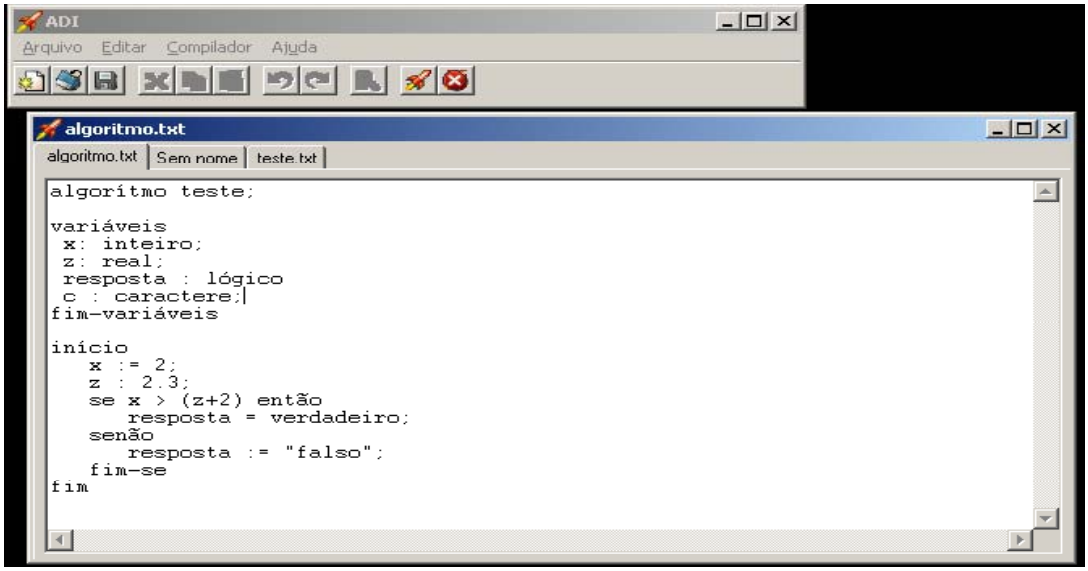

Fig.4. Editor de texto do programa ADI

Observa-se que nesta visualização é permitido que vários arquivos textos podem ser abertos.

Executando o algoritmo da figura, tem-se uma janela exibindo os erros encontrados. O texto dessa janela nada mais é que a saída do programa Analisador Sintático.

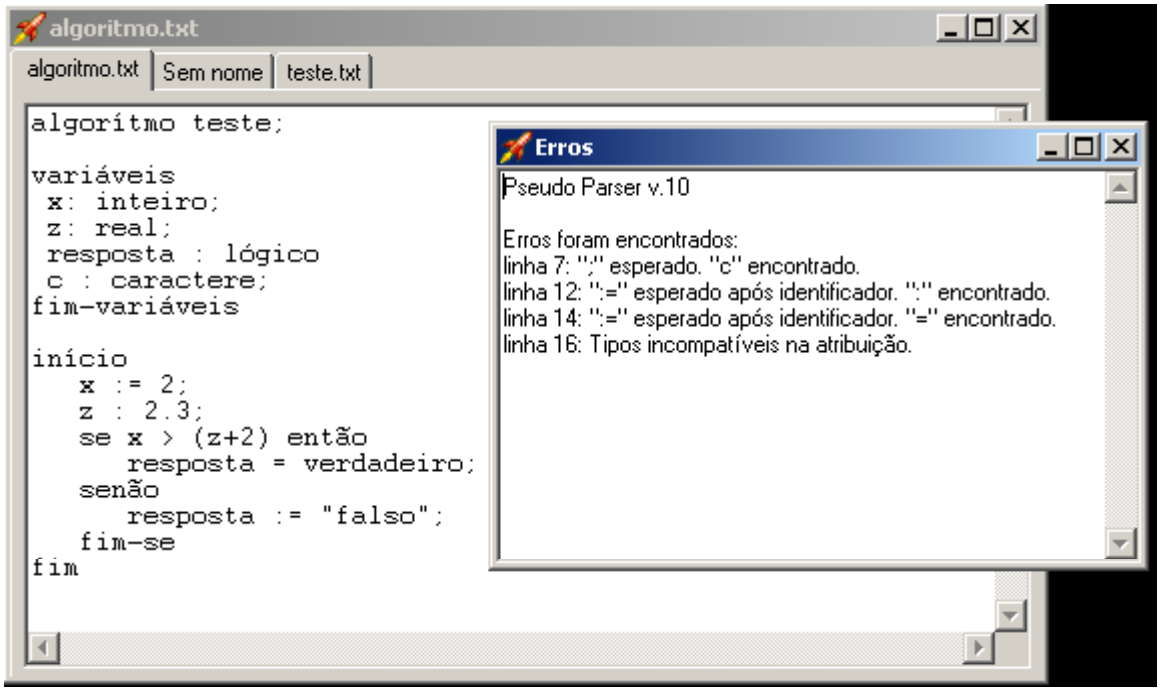

Fig.5. Exibição de erros do programa Pseudo Parser pelo ADI

Essa ferramenta agrega maior consistência não só a gramática da linguagem a ser analisada como sua estrutura interna. Com isso tem-se uma ferramenta mais robusta, modular, extensível e portável.

O ADI torna a prática de exercícios de algoritmo mais dinâmica, utilizando um mesmo aplicativo para desenvolvimento e correção de erros, além de permitir adição de novos recursos práticos para o uso pelos alunos. Tudo isso centralizado em apenas uma ferramenta didática.

Com essa Ferramenta, alunos e professores se beneficiam por seu uso, dentro e fora da sala de aula, tornando o professor independente de certos aspectos da aula, dinamizando as aulas, envolvendo assuntos mais relevantes que as horas de correções de algoritmos e trazendo maior produtividade no ensino em geral, devido ao as suas vantagens de retorno imediato de erros e acertos de sintaxe, maior aproveitamento e organização do tempo no ensino e na aprendizagem e melhor visualização do resultado final

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Bordenave, Juan Díaz; Pereira, Adair Martins. (1991) "Estratégias de Ensino-Aprendizagem", Editora Vozes, 12ª Edição. Petrópolis.
[2] Nunes, D.J.; Bichinho, G.l., Ii Curso de Qualidade de Cursos de Graduação da Área de Computação e Informática. Anais. Editora Universitária Champagnat. Curitiba, 2000.
[3] Shipley, Morton C. (1973) "Síntese de Métodos Didáticos", Editora globo, 1º edição, porto alegre.
[4] Forbellone, André Luiz Villar; Eberspächer, Frederico Henri. (1993) "Lógica de programação", Editora Mcgrawhill e Makron Books, São Paulo.

[5] Tenenbaum, Aaron M.(1995). “**Estrutura de dados usando C”**. Editora Makron Books, São Paulo.
[6]Sebasta, Robert W.(2002). **Concepts of programming languages**. Addison-Wesley, São Paulo.