Um Interpretador Gráfico de Estruturas de Dados como ferramenta de ensino para Computação

Tiago Davi N. de Sousa, Andrei A. Formiga

Centro de Informática – Universidade Federal da Paraíba
(CI-UFPB) João Pessoa, PB – Brasil

tiagodvneves@gmail.com, andrei@ci.ufpb.br

Abstract. *In this paper it is presented a command interpreter for the IGED (Graphical Interpreter of Data Structures) teaching tool. Thus, the characteristics and requirements of the command interpreter are presented in this paper with the aim of be useful to the teaching and learning for disciplines of Programming, Algorithms and Data Structures.*

Resumo. *Neste artigo é apresentado o componente Interpretador de Comandos para a ferramenta de ensino IGED (Interpretador Gráfico de Estruturas de Dados). As características e requisitos do Interpretador de Comandos, apresentados neste artigo, foram elaborados com vista à utilidade da ferramenta para o ensino e aprendizagem de disciplinas de Programação, Algoritmos e Estruturas de Dados.*

1. Introdução

Nos cursos superiores da área da computação, uma das dificuldades é o ensino de programação. Um aspecto desta dificuldade é o ensino dos algoritmos e estruturas de dados, principalmente pela capacidade de abstração requerida do aprendiz (como observado em experiências com a disciplina de Estruturas de Dados [Netto, 2010]). No conteúdo didático das disciplinas de Programação, Algoritmos e Estruturas de Dados, geralmente os conceitos são apresentados de forma estática, mas a execução de um algoritmo é essencialmente uma tarefa dinâmica.

Prover uma forma a partir da qual os alunos possam executar códigos de manipulação de Estruturas de Dados e visualizar os resultados dessas manipulações de forma gráfica e dinâmica teria uma grande utilidade no ensino e aprendizagem dessas disciplinas, já que os alunos poderiam lidar com essas abstrações de forma mais fácil e os professores teriam uma ferramenta que os auxiliassem no processo didático e pedagógico.

Para isso, em Netto et al (2011) foi proposto o IGED (Interpretador Gráfico de Estruturas de Dados), que consiste em um ambiente voltado para o ensino de Programação, Algoritmos e Estruturas de Dados e se baseia na visualização gráfica e dinâmica de como as estruturas de dados são manipuladas por um programa. Além disso, o IGED possui um tutor hipermídia baseado no modelo de autoria NCM para a utilização de materiais didáticos que sejam úteis no ensino e aprendizagem de determinado assunto, como exercícios, slides e códigos de exemplo [Filho et al 2012].

Neste trabalho é apresentado um interpretador de comandos para o IGED, para possibilitar que os códigos informados por um usuário sejam executados e que as animações sejam geradas a partir dele de forma automática e transparente, sem que seja

necessário um segundo usuário para programar as animações, como ocorre em alguns trabalhos relacionados apresentados na Seção 2. O trabalho está organizado da seguinte forma: a Seção 2 apresenta trabalhos relacionados e como o IGED se diferencia destes. Em seguida, a Seção 3 apresenta a arquitetura geral do IGED. Na Seção 4 é apresentado o interpretador de comandos proposto, e a Seção 5 conclui o trabalho com as considerações finais.

2. Trabalhos Relacionados

Nesta seção, são apresentados alguns trabalhos relacionados que consistem em ferramentas de ensino que permitem a visualização das manipulações sobre estruturas de dados.

O Balsa [Brown e Sedgewick 1984] é uma ferramenta de visualização de algoritmos que possui três tipos de usuários: um *scriptwriter*, que prepara material didático para outros usuários; o designer de algoritmos, que implementa os algoritmos a serem animados; e o animador, que projeta e implementa os programas que irão exibir graficamente as saídas dos algoritmos. No seu sucessor, o Balsa II [Brown 1988], foram incluídas melhorias na interface com o usuário e geradores de entradas que proveem dados a serem manipulados pelo algoritmo.

O Zeus [Brown 1991], além de animar algoritmos, possui um editor para construção de interfaces gráficas, mas é necessário um usuário que programe as animações, além do que programe o código a ser executado. O Tango [Stasko 1990], possui um *framework* para visualização de algoritmos. A ferramenta Astral [Garcia, Rezende e Calheiros 1997], além de possibilitar visualizar os algoritmos dos usuários, fornece um conjunto de aplicativos de exemplo que ilustram o funcionamento de algoritmos e interfaces com rotinas de visualização. O AnimA [Amorim e Rezende 1993] permite a reusabilidade de componentes de animação, que ao serem implementados, ficam disponíveis em uma biblioteca do sistema, e possibilita a execução concorrente de animações.

No jGRASP [Cross et al 2009], há três formas de interação e visualização: *debugger*, onde o modo de execução é através de *breakpoints* e *stepping*; *workbench*: objetos de estruturas de dados podem ser criados e seus métodos invocados por meio de diagramas UML; na terceira forma de execução, um usuário informa um texto, contendo um código, e o interpretador da ferramenta o executa. O PCIL [Malone et al 2009] também possui um interpretador que gera visualização das estruturas de dados e executa código em uma linguagem de alto nível da ferramenta. O jLab [Papadimitriou 2007] pode executar tanto arquivos *.class* de *bytecodes* Java, como também *scripts* em uma linguagem própria da ferramenta por meio de um interpretador.

Outras ferramentas, como o JHAVÉ [Naps 2005] e o JIVE [Cattaneo et al 2004], proveem apenas um *framework*, ou seja, uma plataforma a partir das quais é posssível a construção de aplicações de visualização de algoritmos. Já o CIFluxProg [Santiago e Dazzi 2004] permite a construção e execução de algoritmos tanto na forma de Portugol, como na forma de Fluxogramas.

Com o Interpretador desenvolvido neste trabalho, só seria necessário um usuário para animar um algoritmo no IGED, o qual seria multiplataforma e não seria um *framework* como em algumas ferramentas, ou seja, o próprio IGED já seria uma aplicação de visualização de algoritmos. Até o momento, o Interpretador permite apenas

uma forma de execução, mas o modo de execução por *stepping*, com o uso de *breakpoints* pode ser adicionado à ferramenta. Além dessas características, vale ressaltar que com o Interpretador desenvolvido neste trabalho, o IGED teria características e funcionalidades adicionais relacionadas à proposta pedagógica da ferramenta, como descritas em Filho et al (2012), não encontradas em nenhum desses trabalhos.

3. Arquitetura do IGED

Em Netto et al (2011) foi definida uma arquitetura para o IGED, que posteriormente foi modificada em Filho et al (2012) com a adição da Camada Pedagógica. A arquitetura do IGED é formada por quatro camadas, conforme ilustrado na Figura 1: Camada Gráfica, Camada Pedagógica, Interpretador de Comandos e Camada Avaliadora.

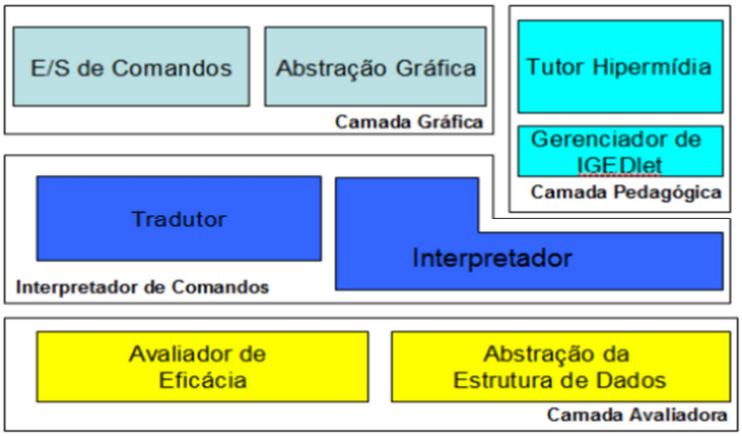

Figura 1. Arquitetura do IGED [Filho et al 2012]

A Camada Gráfica é a responsável pela interface com o usuário, atuando na E/S de comandos e na apresentação gráfica das Estruturas de Dados. A Camada Pedagógica é a responsável por efetivar a proposta pedagógica do IGED, conforme definida em Filho et al (2012), por meio da qual é possível a inserção de objetos hipermídia de acordo com o modelo NCM, que auxiliem na resolução de exercícios e no aprendizado de uma forma geral. O Interpretador de Comandos executa o código informado pelo usuário em uma linguagem de alto nível, onde o Tradutor o traduz em um código numa linguagem de baixo nível para que seja executado por um Interpretador. Na Camada Avaliadora, é feita a determinação se um código informado pelo aluno para a resolução de um problema está correto ou não, ao comparar o resultado da execução desse código com o resultado da execução do código considerado correto pelo professor para um problema.

4. Interpretador do IGED

O Interpretador que foi desenvolvido e que é proposto para o IGED executa código traduzido pelo Tradutor do IGED em linguagem Oolong, uma linguagem de montagem de *bytecodes* Java, definida em Engel (1999), e gerará saídas com operações de atualização para a Camada Gráfica e para a Abstração das Estruturas de Dados. É um Interpretador de pilha, ou seja, executa operações básicas com o uso de uma pilha [Parr 2010].

Ele possui as mesmas estruturas básicas que uma implementação da JVM possui, conforme apresentadas em Engel (1999): uma pilha de *stackframes* para a execução de métodos, um *heap* para conter os objetos em execução e uma área de classes que contém definições de classes e dados relacionados às variáveis e métodos dessas classes.

O Interpretador possui um Montador que efetua uma análise léxica, sintática e semântica do código recebido pelo Tradutor em Oolong. Após a análise sintática é gerada uma árvore sintática e na análise semântica são feitas duas passagens sobre essa árvore. Na primeira, são definidos os símbolos em uma Tabela de Símbolos. Os símbolos são um conjunto formado por símbolos de classe e seus componentes: símbolos de atributos, métodos e variáveis. Na segunda passagem, são feitas as referências dos símbolos definidos no código, onde para cada símbolo é gerado um identificador numérico. Essas duas passagens permitem a ocorrência de *forward references*, isto é, é possível fazer uma referência a um símbolo no código antes dele ter sido definido [Parr 2010].

O Montador também é responsável por gerar e colocar os códigos de máquina em seus respectivos métodos em um símbolo de classe. Dessa forma, os símbolos de classe irão formar a Área de Classes do Interpretador. Um Carregador de Classes cria os objetos a partir das informações dos símbolos de classe na Área de Classes, colocando-os no *heap*.

Cada símbolo de método tem sua memória de código, que é organizada em células de 1 byte e armazena código em linguagem de máquina. Uma instrução na linguagem de máquina do Interpretador é formada por um opcode, com um 1 byte de comprimento, seguido de zero, um ou dois operandos. Cada operando ocupa 4 bytes na memória de código. Na figura 2, é ilustrado como uma estrutura do tipo Lista fica organizada na máquina virtual do interpretador, com as variáveis do *stackframe* de um método referenciando seus objetos no *heap*. O Interpretador refletirá as alterações desses objetos ao longo da execução de um programa em seus objetos correspondentes da Camada Gráfica, que exibirá de forma gráfica para o usuário essas alterações.

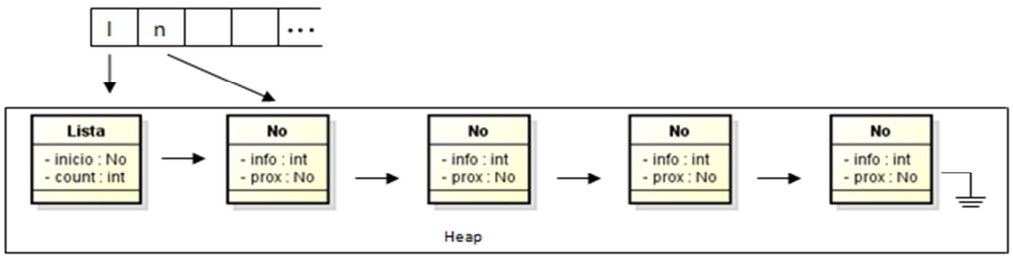

Figura 2. Ilustração de como uma estrutura do tipo Lista fica disposta na VM do Interpretador durante sua execução

Com isso, esse interpretador dá suporte à orientação a objetos. Por meio dele, é possível executar código em que aparecem relacionamentos entre classes, como herança, composição e polimorfismo. Uma classe também pode possuir atributos, variáveis e métodos estáticos e não estáticos. Além de operações de criação e manipulação de objetos, o Interpretador executa instruções de desvio condicional e

incondicional, invocação de métodos estáticos e não estáticos com chamadas recursivas e operações aritméticas.

Dessa forma, o Interpretador pode lidar com várias estruturas de dados pelo fato de seus modelos de execução serem os mesmos. Isso porque o Interpretador pode manipular várias estruturas como vetores, listas, árvores e grafos com os algoritmos definidos pelo usuário, já que ambas as estruturas e seus respectivos nós podem ser definidos como classes e objetos a serem representados e executados pelo Interpretador em sua máquina virtual.

5. Conclusões e Trabalhos Futuros

O Interpretador desenvolvido neste trabalho pode fornecer à ferramenta IGED características essenciais na efetivação da sua abordagem construtiva, porque com ele, seria possível a execução dos códigos traduzidos pelo componente Tradutor e a geração de saídas de atualização gráficas quando os objetos das estruturas sofrerem alteração no *heap* ao longo da execução de um programa.

Pelas características que ele possui, ele seria um componente central e muito importante para o IGED, visto que com ele, seria possível a execução de códigos de programação de alto nível traduzidos pelo Tradutor com as principais características encontradas em linguagens de programação estruturadas e orientadas a objetos.

Outras funcionalidades úteis para as disciplinas de Algoritmos, Programação e Estruturas de Dados, podem ser adicionadas, como a execução por *stepping* com o uso de *breakpoints*. O Interpretador também poderia efetuar uma contagem dos passos básicos executados para um algoritmo, gerando gráficos que poderiam fornecer aos alunos noções das complexidades de seus algoritmos. Enfim, outras características úteis que essa ferramenta poderia fornecer ao ensino poderiam ser exploradas, as possibilidades são muitas.

Referências

Amorim, R. C. e Rezende, P. J. (1993). Compreensão de Algoritmos através de Ambientes Dedicados a Animação. In: XX SEMISH, p. 32.

Brown, M. H. (1991). Zeus: A System for Algorithm Animation and Multi-View Editing. IEEE Workshop on Visual Languagens.

Brown, M. H. Exploring Algorithms Using Balsa II. IEEE Computer, n. 5.

Brown, M. H. e Sedgewick, R. (1984). A System for Algorithm Animation. ACM SIGGRAPH Computer Graphics, n. 3.

Cattaneo, G., Faruolo, P., Petrillo, U. F e Italiano, G. F. (2004). JIVE: Java Interactive software Visualization Environment. In: IEEE Symposium on Visual Languages and Human Centric Computing, p. 41–43.

Cross, J. H., Hendrix, T. D., Umphress, D. A., Barowski, L. A., Jain, J. e Montgomery, L. N. (2009). Robust Generation of Dynamic Data Structure Visualizations with Multiple Interaction Approaches. ACM Transactions on Computing Education, n. 2.

Engel, J. (1999). Programming for the Java Virtual Machine. Addison Wesley, 512 p.

Filho, G. F. S., Netto, D. P. S., Procopio, L. D. P., Formiga, A. A. e Brito, A. V. (2012). Tutor hipermídia baseado no modelo de autoria NCM para o Interpretador Gráfico de Estrutura de Dados. In: Anais do XX Workshop sobre Educação em Computação.

Garcia, I. C., Rezende, P. J. e Calheiros, F. C. (1997). Astral: Um Ambiente para Ensino de Estruturas de Dados através de Animações de Algoritmos. Revista Brasileira de Informática na Educação, n. 1.

Malone, B., Atkison, T., Kosa, M. e Hadlock, F. (2009). Pedagogically Effective Effortless Algorithm Visualization with a PCIL. In: Frontiers in Education Conference, 2009. FIE ’09. 39th IEEE, p. 1-6.

Naps, T. L. (2005). JHAVÉ: Supporting Algorithm Visualization. IEEE Computer Graphics and Applications, n. 5.

Netto, Dorgival. Análise dos dados do questionário da disciplina Estrutura de Dados. Relatório Técnico. Departamento de Ciências Exatas, UFPB, 2010.

Netto, D. P. S., Oliveira, T. J., Sousa, T. D. N., Filho, G. F. S., Formiga, A. A., Brito, A. V. (2011). Desenvolvimento de um Interpretador de Comandos e Avaliador Gráfico para o Ensino de Estrutura de Dados. In: Anais do XIX Workshop sobre Educação em Computação.

Papadimitriou, S. (2007). Scientific programming with Java classes supported with a scripting interpreter. IET Software, n. 2.

Parr, T. (2010). Language Implementation Patterns. United States of America: The Pragmatic Bookshelf, 369 p.

Santiago, R. e Dazzi, R. L. S. Ferramenta de apoio ao ensino de algoritmos. In: Anais do Seminário de Computação - SEMINCO.

Stasko, J. T. (1990). Tango: A Framework and System for Algorithm Animation. IEEE Computer, n. 9.