UNIVERSIDADE FEDERAL DE LAVRAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Orientador: Prof. Heitor Augustus Xavier Costa

Desenvolvimento de um *Software* Educacional para o Ensino de Algoritmos, Estruturas de Dados e Programação

Bolsista: Rodrigo Pereira dos Santos

Relatório Final apresentado à Universidade Federal de Lavras, como parte das exigências do PBIICT/FAPEMIG, referente ao período de março/2004 a fevereiro/2005.

_____ _____

Assinatura do bolsista **Assinatura do orientador**

Rodrigo Pereira dos Santos
Heitor Augustus Xavier Costa

Desenvolvimento de um *Software* Educacional para o Ensino de Algoritmos, Estruturas de Dados e Programação

Resumo

Este trabalho tem o objetivo de apresentar o **TBC-AED**, um produto de software que contém vários temas relacionados aos conteúdos lecionados aos iniciantes em Computação, procurando deixar uma contribuição à literatura e aos docentes da área. Com isso, o **TBC-AED** busca contribuir para a melhoria da qualidade de ensino de programação, algoritmos e estruturas de dados nos cursos de graduação em Computação e Informática.

Palavras-chaves: algoritmos e estruturas de dados, programação, informática na educação.

Abstract

This work has the objective to present **TBC-AED**, a software that contains a lot of themes related to the taught contents to the new students in Computation, trying to contribute to literature and to teachers of the field. So, **TBC-AED** tries to help the improvement of the teaching quality of programming, algorithms and data structures in the graduation courses in Computing.

Key-words: algorithms and data structures, programming, computing in education.

Sumário

Lista de Figuras

Capítulo 1 – Introdução

A busca por melhores métodos educacionais para cursos que se embasam em Computação e Informática é uma preocupação constante de professores e coordenadores de curso na área. A cada dia novas tecnologias são desenvolvidas e novos conceitos precisam ser incorporados à formação universitária, para que sejam formados bons profissionais para o mercado.

Um grande desafio que está vinculado à essa formação é a maneira de estruturar o ensino de **programação e desenvolvimento de algoritmos**, em quaisquer dos cursos escolhidos de Computação e Informática.

Segundo [ASCENCIO *apud* CAMPOS *et al* 2003], algoritmo é a descrição de uma seqüência de passos que deve ser seguida para a realização de uma tarefa, e para [FARRER *apud* CAMPOS *et al* 2003], ação é um acontecimento que, a partir de um estado inicial, após um período de tempo finito, produz um estado final previsível e bem definido. Assim, um algoritmo é a descrição de um conjunto de comandos que, obedecidos, resultam em uma sucessão finita de ações.

Atualmente, o ensino de algoritmos busca nas Ciências Exatas seu pilar de sustentação, pois disciplinas nessa área despertam o raciocínio matemático-lógico para resolução de problemas. Mas, isso deve ser mais uma das ferramentas para o estudante e não a sua única fonte, pois a sua formação futura será muito ampla.

O conceito de programação, segundo [WIRTH 1989], é a arte ou a técnica de construir e formular algoritmos de uma forma sistemática. Esse conceito induz que o ingresso tem um dom (arte) de programar ou ele pode aprender as técnicas necessárias a aprimorar seu conhecimento. Logo, ao ingressar em um curso superior na área, ele deve ter sua base bem sedimentada, para que fique apto a prosseguir de maneira positiva durante os seus estudos. Com isso, professores de disciplinas relacionadas à programação e coordenadores de curso sentem a grande responsabilidade de estar buscando e aperfeiçoando sua maneira de conduzir a estrutura disciplinar e curricular dos graduandos em seus estágios iniciais da universidade.

Este trabalho faz um breve apanhado sobre os problemas encontrados no ensino de Computação e como ferramentas da área poderiam contribuir para a melhora deste quadro. Além disso, pretende apresentar uma ferramenta computacional didática, o **TBC-AED** *(Treinamento Baseado em Computador para Algoritmos e Estruturas de Dados)*, fruto de um estudo sobre algoritmos, estruturas de dados e programação. Esta visa tornar o ensino deste conteúdo – básico para o estudo em Computação – mais prático e abrangente, de forma a despertar o interesse do aluno, o seu espírito de pesquisa e a busca de informações que possam torná-lo um profissional crítico e de opinião sólida. Assim, seu rendimento ao longo do curso aumentaria, melhorando o seu desempenho em disciplinas mais específicas.

1.1 Motivação

É importante salientar alguns pontos especiais acerca do intuito da pesquisa, visto que a aula de uma disciplina que relaciona programação, algoritmos e estruturas de dados deve ser realizada em laboratórios. Isso para que os alunos possam entender as abstrações apresentadas. Deve ser composta de uma parte teórica, onde conceitos são transmitidos, e uma parte prática, onde vão entender comandos e tópicos ministrados.

Nesse sentido, o **TBC-AED**, o qual enfatiza animação gráfica, é de extrema importância como facilitador do processo de aprendizagem, uma vez que a apresentação de conceitos abstratos se torna mais viável e didática, melhorando a qualidade do material das aulas. Normalmente, um aluno se interessa por aulas diferenciadas e isso não só prende sua atenção, como também influenciam positivamente nas avaliações.

Além disso, uma economia de tempo pode ser conseguida e esta seria direcionada para maiores explicações e resoluções de exercícios, uma vez que o material didático seria virtual, evitando uma explanação cansativa através do quadro-negro.

Um dos grandes pontos positivos é que o aluno teria acesso livre ao material virtual para que pudesse estudar em casa. Além disso, ele possuirá um valioso projeto, para consultas futuras, em caso de necessidade – a tendência em abstrair detalhes, com o passar do tempo, sempre leva a dúvidas e estas a novas pesquisas em estruturas básicas ora aprendidas.

A transição de educadores também seria facilitada, uma vez que existiria uma base pronta para ser apreciada, mantendo a qualidade de ensino e aprendizagem do conteúdo.

Assim sendo, o uso do **TBC-AED** é um subsídio valioso para a formação de alunos, ajudando-os a concluírem o seu curso de Computação, pois são disciplinas que estão no início e são pré-requisitos necessários para as disciplinas mais específicas de períodos avançados. Quando feita uma boa base, o rendimento e o desempenho aumentam, proporcionando melhores resultados, melhores currículos e, conseqüentemente, melhores profissionais para o mercado.

1.2 Objetivos

Os objetivos para a realização deste trabalho de pesquisa são:

- propiciar ao orientado um estudo dirigido da linguagem de programação Java que é amplamente utilizada no mercado e na pesquisa para o desenvolvimento de produtos de *software*;
- ampliar conhecimentos do orientado na área de programação, algoritmos e estruturas de dados, nos seguintes tópicos:
    - Busca em vetor (busca binária);
    - Métodos de ordenação;
    - Tipos abstratos de dados: lista, pilha e fila (alocação estática e dinâmica);
    - Conceitos, propriedades e implementação de árvores binárias.
- proporcionar ao orientado contato com áreas específicas de seu curso, como Linguagens de Programação e Engenharia de *Software*;
- levar à pesquisa de métodos de ensino de programação mais didáticos para os ingressos no curso de graduação, com devidas considerações;
- acarretar o desenvolvimento de um produto de *software* – o **TBC-AED** –, ambiente gráfico para estudo da disciplina de programação, a fim de tornar as aulas mais dinâmicas e didáticas. Isso será disponibilizado aos alunos e demais interessados, como uma ferramenta poderosa para programadores iniciantes;

- facilitar e melhorar o acesso de alunos de outros cursos que não os fundamentados em Computação e Informática às disciplinas de programação, embasando o processo de transição curricular. Isso porque muitos dos cursos apresentam disciplinas de programação, algoritmos e estruturas de dados como suas disciplinas eletivas;
- estimular mudanças na metodologia de ensino de cursos de Computação, com o desafio de não manter na vanguarda uma área de constante atualização;
- construir um suporte para o orientado desenvolver projetos futuros de forma bem consistente e engajada – uma vez que a programação e os algoritmos formam a base de grande parte dos estudos de Computação;
- manter a estabilidade de ensino de disciplinas da área, mesmo quando da alteração no quadro docente.

1.3 Metodologia de Desenvolvimento

Este projeto atende aos seus objetos, utilizando-se alguns procedimentos.

Foi realizado um levantamento bibliográfico, na *Internet* e em bibliotecas, de textos, artigos e informações a respeito dos tópicos relacionados à programação, algoritmos e estruturas de dados, em especial capítulos específicos de [WIRTH 1989]. Também foram verificadas novas metodologias de ensino na área de Computação.

Foram identificadas também as dificuldades de alunos das disciplinas da área, para que pudessem contribuir para a qualidade do projeto. Isso foi feito através de reuniões entre o orientador (professor da disciplina diretamente envolvida há três semestres consecutivos) e o orientado (ex-aluno da disciplina que atingiu média máxima – 100 pontos).

Com o material apropriado, partiu-se para o estudo orientado da linguagem de programação *Java*, com a finalidade de implementar o produto de *software*. Isso foi feito principalmente com a leitura do livro [DEITEL 2003] em capítulos mais relacionados à pesquisa.

Em seguida, foi analisada a forma de ministrar a disciplina de COM155 – Algoritmos e Estruturas de Dados II, oferecida pelo Departamento de Ciência da

Computação da Universidade Federal de Lavras (DCC/UFLA). Esta é peça fundamental para a formação de programadores e bons estudantes para o curso. Por isso, com o término deste trabalho, serão preparados seminários para discussão sobre o tema e o engrandecimento de pesquisas futuras.

Foram disponibilizados no *site* do orientado e do orientador *links* que contém atualmente um resumo do projeto, avanços e informações para os interessados e os alunos diretamente ligados a disciplinas da área.

A preocupação de construir códigos-fonte manuteníveis do produto de *software* educacional foi constante. O paradigma de programação utilizado foi a orientação a objetos e a linguagem de programação foi *Java* devido a alguns aspectos intrínsecos considerados importantes como a portabilidade.

A implementação bruta do produto de *software* já foi concluída. Foram desenvolvidos os temas Busca em Vetor (**busca binária**), Métodos de Ordenação (***select sort***, ***insert sort***, ***boubble sort***, ***merge sort*** e ***quick sort***), Tipos Abstratos de Dados (TAD) **lista**, **pilha** e **fila** implementados utilizando alocação estática – uso de vetor (*array*) – e dinâmica – uso de apontadores (ponteiros) –, e Árvores (**árvore binária**), de maior complexidade. Para evitar quaisquer problemas, uma vez que a disciplina é de caráter básico para programação e importante para o futuro dos graduandos, optou-se por iniciar a etapa laboratorial após a conclusão do produto de *software*. Isso porque se trata uma experiência bastante interessante e inovadora, mas que precisa ser realizada com orientação inicial bem elaborada e estruturada. Logo, pensou-se em cuidar para que quaisquer pontos negativos imprevisíveis pudessem alterar o resultado final. Assim, a avaliação da eficiência, da validade, da qualidade e da didática esperadas pelo projeto poderão ser feitas de maneira formal e engajada na segunda fase do projeto. Além disso, ambos, orientado e orientador, poderão discutir e montar um programa de ensino melhor e mais completo para a disciplina a partir de 2005.

Aconteceram reuniões entre orientado e orientador para discussão a respeito do produto de *software*, coleta constante de artigos e periódicos na área de programação, linguagem de programação *Java* e metodologias de ensino de Computação. Assim, pelo fato do orientador lecionar a disciplina envolvida, a pesquisa se tornou menos propícia a

erros. Isso torna transparente e mais astuta a posição tomada por ambos de conduzir a disciplina com recursos gerados pela mesma, a partir de 2005. No entanto, se acontecer alguma eventualidade durante o decorrer do curso, o orientado será acionado e estará apto para verificar o que ocorreu juntamente com o orientador.

Devido à imersão em um acervo interessante de informações gerada pela pesquisa, foram pensados artigos, que estão sendo elaborados para a divulgação através da submissão a eventos e a periódicos científicos relacionados ao tema. Além disso, o produto de *software* é uma realidade esperada como resultado final do projeto.

Um seminário direcionado à primeira turma a utilizar o **TBC-AED** (1º semestre de 2005) será realizado, para apresentação dos objetivos do produto de *software* e avaliação de perspectivas dos alunos para o futuro. Uma apresentação final dos resultados e das metodologias de ensino para o engrandecimento da pesquisa, do órgão que possibilitou esta realidade, e da equipe técnica (orientado e orientador) será realizado. Esta terá como público alvo alunos, professores e coordenadores de curso. Dessa forma, contribuirá para a melhoria de profissionais das áreas de Informática e Computação.

1.4 Estrutura do Trabalho

O Capítulo 2 discorre sobre a análise de metodologias de ensino em Computação, alguns problemas enfrentados e possíveis soluções. O capítulo 3 apresenta a linguagem de programação *Java* e o Capítulo 4, conceitos de Informática na Educação, alvos de estudos durante a pesquisa. No Capítulo 5, tem-se a apresentação do **TBC-AED**, com a descrição de seu desenvolvimento, ferramentas utilizadas, funcionalidade e conceitos envolvidos. Por fim, no Capítulo 6, serão apresentadas as considerações finais sobre a pesquisa.

Capítulo 2 – Metodologias de Ensino de Computação

2.1 Considerações Iniciais

Neste capítulo, são apresentados alguns dos pontos críticos relacionados às metodologias de ensino de Computação, enfatizando a forma de resolução de problemas e suas implicações. Na seção 2, é apresentada uma breve análise de problemas e desafios enfrentados no ensino desta ciência, considerando algumas soluções. Na seção 3, são retratados os problemas de estudo de algoritmos, estruturas de dados e programação em decorrência da forma de ensino. Na seção 4, são listados alguns produtos de *software* utilizados para o ensino desses tópicos. Por fim, são apresentadas as considerações finais do capítulo.

2.2 Análise de problemas e desafios no ensino de Computação

Dentro da esfera de discussão dos métodos educacionais como ferramentas para a formação de profissionais de qualidade para o mercado, a presente pesquisa se propõe a dar sua contribuição. Entretanto, ela analisa um tema mais abrangente e complexo: as áreas de Computação e Informática, que apresentam inovações constantes e grande transformação de preceitos existentes. Sabe-se que é complicado promover mudanças na educação básica e, ao se tratar de cursos de graduação baseados na produção de tecnologia, o problema é ainda maior. O que se pode perceber é que esses cursos exercem grande atração sobre os futuros ingressos em uma faculdade, o que se mostra pela concorrência elevada em processos seletivos. No entanto, vencido o obstáculo de entrada, vêm o início da fase universitária e um dos maiores dilemas: o contato com disciplinas básicas que vão preparar o ingresso para atuar em áreas específicas. Esse contato, sobretudo em cursos de Computação, pode gerar tanto afinidade quanto repulsa, o que acontece com freqüência expressiva, conforme [RODRIGUES 2002, SCHULTZ 2003, CHAVES DE CASTRO *et al* 2003, DELGADO *et al* 2004 *apud* JÚNIOR e RAPKIEWICZ 2004].

Dessa forma, a falta de compreensão do raciocínio lógico pode ser a principal razão pelo alto índice de reprovação nas disciplinas de **Algoritmos e Programação** e,

em alguns casos, pela desistência de um curso. Em parte, isso é decorrência da dificuldade encontrada pelos professores para acompanharem efetivamente as atividades laboratoriais de programação, dado o grande número de estudantes geralmente sob sua supervisão [TOBAR *et al* 2001]. Dentro desse paradigma, pode ser destacada também a forma como os alunos estudam para a disciplina, geralmente memorizando, e a falta de pré-requisitos em conteúdos relacionados [JÚNIOR e RAPKIEWICZ 2004].

[RODRIGUES 2002] relata outros problemas associados a este processo:

- a falta de motivação do aluno criada pelo despreparo, citado anteriormente, e o desânimo quando há, principalmente, a crença de que a disciplina constitui um obstáculo extremamente difícil de ser superado;
- o processo tradicional de avaliação pode deixar o aluno tenso prejudicando o aprendizado;
- o relacionamento entre professor e aluno pode ser um problema quando o primeiro preocupa-se em mostrar o que sabe desconsiderando um ambiente de aprendizagem descontraído e colaborativo;
- a didática ou a falta de metodologia de ensino dificulta o aprendizado dos novos e diversos conceitos. Inclui-se aqui a grave falta de comunicação entre os professores das várias disciplinas que permitiria identificar conteúdos afins ou superposição de tópicos, tornando o trabalho realizado mais integrado e auxiliando o processo de ensino e aprendizagem [GIRAFFA, MARCZAK e ALMEIDA 2003; PIMENTEL, FRANÇA e OMAR 2003].

[RODRIGUES 2002] ainda afirma que o professor deve ser capaz de fazer o aluno compreender a abstração envolvida com toda a simbologia utilizada. Para isto, deve utilizar sua criatividade e tentar resolver cada problema baseando-se em situações do cotidiano e, assim, o aluno começa a ter raciocínio lógico e ordem de pensamento.

No entanto, para [BUZIN 2001], o iniciante do curso superior traz uma bagagem cultural que não apenas não serve para o apropriado desenvolvimento de um estudo acadêmico, mas até atrapalha o desenvolvimento do estudo. Ele vem de uma experiência de estudante de disciplinas desenvolvidas em torno do paradigma de apresentação de resposta e soluções, usualmente da velha e surrada didática diretiva, ao

invés de focar no processo de questionamento. Com certeza, sem uma radical mudança cultural de atitude e comportamento, este indivíduo não poderá ser um profissional de computação adequado, visto que é preciso desenvolver a capacidade de buscar a resposta através de novas perguntas que levem à resposta da questão original. A formação do profissional em computação deve incitar o desenvolvimento de raciocínio crítico, a solução de problemas, a aplicação de métodos de pesquisa e o desenvolvimento profissional contínuo. Assim, mais uma das vertentes do problema educacional na área de Computação encontra-se na sua formação básica.

Mas, considerando o campo universitário, o nível de maturidade do aluno é importante neste ponto para que o professor possa adequar o processo pedagógico no sentido de acompanhar o amadurecimento do aluno, permitindo aplicar técnicas de ensino e aprendizagem [JÚNIOR e RAPKIEWICZ 2004].

O acompanhamento contínuo de cada aprendiz e o tratamento personalizado a cada estudante, principalmente em turma com um número elevado de alunos, só serão possíveis se forem apoiados por sistemas inteligentes, ou seja, auxiliados por computador [PIMENTEL, FRANÇA e OMAR 2003]. Logo, a idéia de usar produtos de *software* para o ensino de programação é de grande valia e ainda poderá transformar o ensino de disciplinas avançadas em algo mais primoroso e tecnológico.

A partir de todo esse problemático desafio, mudanças passaram a ser consideradas por graduados, mestres e doutores que trabalham com o ensino de algoritmos e programação. A análise da literatura feita por [JÚNIOR e RAPKIEWICZ 2004], independente de ser nacional ou internacional, mostrou que há três vertentes na busca de soluções para os problemas apontados:

- **Ferramentas**, que incluem trabalhos que apresentam ferramentas computacionais visando facilitar o processo de ensino e aprendizagem;
- **Estratégias**, que incluem trabalhos que discutem estratégias de ensino e/ou de avaliação de competências;
- **Ferramentas e Estratégias**, que são os trabalhos que discutem alguma estratégia suportada por ferramentas computacionais.

Assim, a linha da presente pesquisa vem ao encontro com o terceiro tópico. Além do desenvolvimento do produto de *software* (ferramenta computacional), ocorre

também uma discussão a respeito de estratégias para a melhoria do ensino de programação, algoritmos e estruturas de dados, um dos pontos fortes para a formação de futuros profissionais nas áreas tecnológicas de Computação e Informática.

Para [GARCIA, REZENDE e CALHEIROS 1996], diversos sistemas para implementação de animações de algoritmos e de estruturas de dados foram produzidos. Vários destes sistemas exploram muito bem a potencialidade do uso de visualizações gráficas das operações realizadas nas estruturas de dados como ferramenta de ensino. O uso de movimento em tempo real, cores e sons enriquecem ainda mais o poder de comunicação. De fato, algumas experiências positivas foram relatadas na literatura, principalmente com o uso de sistemas que utilizam a abordagem ativa, na qual o aprendiz interage com as animações na criação de instâncias testes. Por outro lado, a ênfase na qualidade da animação gráfica para enriquecer a abordagem ativa da maioria dos sistemas levou-os a um grau de complexidade que impede seu uso para fins de criação de animações pelos próprios aprendizes. Deste modo, sua utilização se limita à interação, ainda que ativa, com as animações pré-implementadas, representando as mais novas tendências de trabalhos para o ensino de Computação e Informática.

[GARCIA, REZENDE e CALHEIROS 1996] ainda afirmam que, do mesmo modo que a experimentação durante a operação das animações enriquece o aprendizado mais do que a mera observação passiva delas, é de se esperar que, com um sistema onde a própria implementação das animações gráficas é facilitada a ponto de poder ser realizada pelo estudante, a absorção do funcionamento dos algoritmos seja ainda mais intensa.

Apesar desta nova tendência dos produtos de *software* para educação em Computação, ainda é muito utilizada a abordagem ativa descrita anteriormente por [GARCIA, REZENDE e CALHEIROS 1996]. Isso porque configura uma forma de controle do aprendizado pelo docente, uma vez que quaisquer informações erroneamente assimiladas na maioria das vezes causam um impacto negativo na evolução do aluno durante o curso. Dessa forma, o produto de *software* proveniente desta pesquisa utiliza estratégias passo-a-passo na simulação de estruturas de dados conhecidas e bem sedimentadas. Assim, o aluno realiza uma série de operações ao utilizar o programa, assimilando conceitos básicos sem correr o risco de implementar seus próprios algoritmos de forma incorreta.

2.3 Programação e Algoritmos: a base da Computação

Segundo a estrutura curricular abstrata constante nas Diretrizes Curriculares do MEC, a matéria de Programação faz parte da área de formação básica em Ciência da Computação, juntamente com as matérias Computação e Algoritmos e Arquitetura de Computadores. [AZEREDO 2000] afirma que seu conteúdo abrange, além do ensino de linguagens de programação propriamente ditas, os conceitos, os princípios e os modelos de programação e o estudo de estruturas de dados e de métodos de classificação e pesquisa de dados.

Ainda de acordo com [AZEREDO 2000], conforme consta na caracterização da matéria, "A programação, entendida como programação de computadores, é uma atividade voltada à solução de problemas. Nesse sentido, ela está relacionada com uma variada gama de outras atividades como especificação, projeto, validação, modelagem e estruturação de programas e dados, utilizando-se das linguagens de programação propriamente ditas, como ferramentas.

Ao contrário do que se apregoava há alguns anos, a atividade de programação deixou de ser uma 'arte' para se tornar uma ciência, envolvendo um conjunto de princípios, técnicas e formalismos que visam a produção de *software* bem estruturado e confiável. Cite-se, dentre estes, os princípios da abstração, do encapsulamento e as técnicas de modularização e de programação estruturada.

Portanto, o estudo de programação não se restringe ao estudo de linguagens de programação. As linguagens de programação constituem-se em uma ferramenta de concretização de *software*, que representa o resultado da aplicação de uma série de conhecimentos que transformam a especificação da solução de um problema em um programa de computador que efetivamente resolve aquele problema.

No estudo de linguagens de programação, deve ser dada ênfase aos aspectos funcionais e estruturais das linguagens de programação, em detrimento aos detalhes de sintaxe. Conceitos como o significado de associação, avaliação, atribuição, chamada de procedimento, envio de mensagens, passagem de parâmetros, herança, polimorfismo, encapsulamento, etc devem ser enfatizados. O estudo de linguagens deve ser precedido do estudo dos principais paradigmas de programação, notadamente a programação imperativa, a funcional, a baseada em lógica e a orientada a objetos.

O desenvolvimento de algoritmos, juntamente com o estudo de estruturas de dados, deve receber especial atenção na abordagem do tema programação. Igualmente, deve ser dada ênfase ao estudo das técnicas de especificação, projeto e validação de programas. Um excelente campo para o exercício da programação é constituído pelo estudo de pesquisa em tabelas e de técnicas de ordenação." (Deve-se acrescentar o uso de estruturas de dados para o exercício da programação).

Dessa forma, deve-se entender que ensinar programação não é simplesmente ensinar uma linguagem de programação. Este ensino envolve sobretudo entender problemas e descrever formas de resolução, de maneira imparcial, para que então sejam codificadas em uma linguagem. Ou seja, somente após o aprendizado dos conceitos de algoritmo e fundamentos de lógica é que o estudante pode travar contato com uma linguagem de programação concreta (executável), para experimentar aqueles conceitos.

Além disso, o estudo de programação deve passar efetivamente pelo estudo de estruturas de dados. Ou seja, quando se projeta um programa, deve-se pensar tanto nos algoritmos que manipulam os dados como também na forma de estruturar estes dados. Desta maneira, as duas coisas se complementam.

De acordo com [AZEREDO 2000], dentro do ensino de estruturas de dados, deve ser dado destaque a parte conceitual e comportamental das estruturas, antes de pensar em implementação. Por exemplo, no ensino de estruturas lineares, deve-se transmitir ao aluno inicialmente o conceito desta estrutura, através das operações aplicáveis, como criar, acrescentar, consultar, remover, etc objetos da lista. Em um curso de Ciência da Computação, esta parte pode ser apresentada através de uma especificação algébrica, por exemplo. Após, pode-se prosseguir mostrando disciplinas de uso, como FIFO, LIFO, etc, caracterizando assim casos particulares de listas lineares (fila, pilha, etc). Finalmente, discutem-se formas de implementações ou representações físicas das listas (contigüidade, encadeamento, circulares, duplamente encadeada, com/sem descritor, etc). Deve ser dada ênfase especial no estudo de estruturas encadeadas. Este tópico é onde normalmente o aluno apresenta maior dificuldade de assimilação do conceito, uma vez que o encadeamento não é um conceito muito freqüente na vida real, dificultando sua compreensão. Dessa forma, é recomendável que as disciplinas de estruturas de dados possuam significativa atividade prática em laboratório.

Quanto ao sistema de avaliação aplicado atualmente, é o mesmo adotado tradicionalmente na maioria dos cursos de natureza teórico-prática. Além de provas periódicas, são aplicados alguns trabalhos de programação para fixação. Uma sugestão que vem ao encontro do uso de um produto de *software* durante as aulas é a redução do peso em provas teóricas, normalmente elaboradas sobre um conteúdo denso.

Com relação aos cursos na área de Computação e Informática, os cursos de licenciatura visam a formação de educadores para o ensino médio. Os de Ciência da Computação e Engenharia de Computação são voltados para a formação de recursos humanos para o desenvolvimento científico e tecnológico da computação. Embora conhecedores da matéria, não necessariamente devem sair dos cursos com o perfil de "programadores". Os cursos de Sistemas de Informação incluem a formação de programadores entre seus objetivos. Nesse sentido, em um curso de Licenciatura em Computação, o aluno poderá ter uma carga de prática de programação menor do que a exigida nos demais cursos, sem prejuízo da parte conceitual da matéria. Também deve haver uma diferenciação entre os casos de cursos de atividade fim e atividade meio. Para este último, recomenda-se uma ênfase maior no conhecimento de linguagens de programação e atividades práticas.

Assim, a programação é, pois, um dos pontos chaves em um curso de Computação, pois é a atividade que supre o computador com meios para servir ao usuário. Seu uso adequado, juntamente com o entendimento de conceitos, princípios, teorias e tecnologias, pode conduzir à produção de *software* de qualidade, objetivo final, tanto quando se tem a computação como um meio como quando a computação é tida como um fim.

Com esse intuito, [SETUBAL 2000] apresenta algumas recomendações genéricas a serem aplicadas no ensino de disciplinas relacionadas a algoritmos e computação:

- **coerência com os objetivos fundamentais.** Por coerência, entende-se que o professor deve expressar claramente as idéias, os conceitos e as técnicas perante os alunos (se o professor coloca algoritmos confusos na lousa ou em transparências, ele não pode esperar algoritmos claros nas respostas dos alunos); deve destacar a importância dos resultados teóricos e mostrar rigor formal toda vez que isto se fizer necessário; e deve procurar valorizar o uso

de técnicas na resolução de problemas. Esta última coerência pode ser alcançada em particular usando a técnica de "descobrir" a solução de um problema junto com os alunos, ao invés de simplesmente apresentar e explicar soluções "prontas";

- **ênfase no pensamento crítico.** Este é papel de toda educação, porém nesta matéria um cuidado especial deve ser necessário, dada sua natureza teórica com forte componente matemático. Em outras palavras, os alunos que têm pouca maturidade matemática tendem a acreditar em qualquer demonstração que se lhes põe na frente. Tal comportamento deve ser desestimulado! É essencial que os alunos duvidem daquilo que lhes é apresentado e é com dúvidas saudáveis e sua resolução que a percepção da importância do resultado teórico poderá ser consolidada. Nesse sentido, considera-se um recurso valioso os exercícios que pedem para os alunos identificarem falhas de argumentação (por exemplo em uma demonstração errada), erros em algoritmos ou erros em notícias da imprensa;
- **a teoria na prática.** A experiência com o ensino dessas disciplinas de Computação mostra que os alunos em geral não se sentem atraídos por elas, por considerarem-na muito abstrata. Por esse motivo, crê-se ser importante usar como recurso didático sempre que possível um grande número de exemplos da "vida real". A inclusão de projetos de implementação, seja dentro das disciplinas teóricas, seja dentro de uma disciplina específica, também visa a tornar a matéria menos abstrata. De resto, é importante salientar para os alunos o grande impacto que os resultados teóricos têm alcançado na prática.

2.4 Projetos de software *relacionados ao ensino de Algoritmos e Estruturas de Dados em Computação*

Como considerado relevante para o enriquecimento da pesquisa, serão listados alguns projetos envolvidos com algoritmos e estruturas de dados, segundo [GARCIA, REZENDE e CALHEIROS 1996]. Entretanto, não há pretensão alguma aqui de fazer uma resenha exaustiva de todos os trabalhos relacionados. São eles:

a) **Balsa** (*Brown ALgorithm Simulator and Animator*) foi desenvolvido na *Brown University* no início dos anos 80 e seu intuito era o de servir como um laboratório de experimentação com representações dinâmicas de algoritmos em tempo real. Seu sucessor, **Balsa-II**, introduziu uma interface mais amigável e uma maneira homogênea para a execução e interações com animações, trabalhando sobre a plataforma Macintosh e com a linguagem Pascal;

b) No início dos anos 90, surgiu o projeto **Zeus** <http://www.research.digital.com/SRC/zeus/home.html> de M. Brown cujas contribuições incluíram uma proposta de separação entre algoritmo e visualizadores, colocando interfaces bem definidas entre eles; o suporte à construção de programas de grande porte, orientados a objetos e a utilização da linguagem Modula 3. Em Zeus, múltiplos algoritmos podem ser executados concorrentemente, facilitando a análise de diferenças e similaridades entre vários algoritmos;

c) O projeto **Zada** <http://ls4-www.informatik.unidortmund.de/RVS/zada.html> desenvolvido na *University of Dortmund* implementou algoritmos distribuídos utilizando o ambiente Zeus;

d) **Xtango** <http://www.cc.gatech.edu/gvu/softviz/algoanim/xtango.html> é um sistema de animação de algoritmos de propósito genérico que permite a construção de animações em tempo real. O objetivo principal é prover facilidades para os usuários que irão construir as suas animações, fornecendo uma interface de alto nível. Utiliza a linguagem C e opera sobre plataformas Unix e X11 Window System;

e) **Polka** <http://www.cc.gatech.edu/gvu/softviz/parviz/polka.html> é sucessor de Xtango e foi projetado para dar suporte ao desenvolvimento de animações concorrentes, de maneira a facilitar a exibição de algoritmos paralelos;

f) O projeto **AnimA** foi desenvolvido na Unicamp por R. Amorim e P. de Rezende e seu principal objetivo é a produção sistemática de animações em C++ dentro de um ambiente de estações de trabalho Unix.

2.5 Considerações Finais

Apesar de existirem várias frentes de estudos interessadas em contribuir para a melhoria e enriquecimento do ensino de Computação, muito trabalho ainda terá que ser feito. Uma vez que a maior parte das disciplinas dos cursos da área tem enfoque teórico, acabam por deixar as aplicações e as tarefas práticas a cargo dos alunos, o que quase sempre acaba por deixá-los desestimulados e pensativos com relação à sua escolha profissional. Logo, deve ser uma preocupação constante de coordenadores de cursos da área tentar manter condições físicas apropriadas ao desenvolvimento dos cursos, aumentando a produtividade e qualidade dos trabalhos de conclusão de curso.

Capítulo 3 – Informática na Educação

3.1 Considerações Iniciais

Neste capítulo, é apresentada uma das áreas de atuação em Computação e Informática, a Informática na Educação. Na seção 2, é feito um breve apanhado do conceito estudado, da situação relativa ao surgimento e da consolidação do uso de computadores e produtos de *software* como ferramentas para o ensino. Por fim, são apresentadas as considerações finais do capítulo.

3.2 Incorporação da Informática no ensino: análise de causas, conseqüências e implicações

Informática em Educação é uma das áreas de Computação e Informática, a qual trata da utilização de computadores para o ensino, como forma de ampliar a difusão de conhecimentos e facilitar a assimilação de informações.

Entretanto, conforme [ZAMBALDE e ALVES 2002], a incorporação da informática nos ambientes educacionais, sem nenhuma dúvida, provoca mudanças. Essas mudanças representam impactos dos mais variados tipos. Nesse sentido, deve-se sempre questioná-las e, se necessário, nem sempre adotá-las. O objetivo da introdução da informática no ambiente educacional não deve ser o modismo ou a atualização com relação às inovações tecnológicas, ao mercado e à globalização! O questionamento é imprescindível, uma vez que as alterações podem alcançar estrutura física, equipamentos, cultura, economia, forma de produção, forma de aprendizado, modos de comunicação, enfim, processos e atividades dos mais variados tipos.

Para [ALMEIDA e NOGUEIRA 2001], a utilização de computadores na escola pode contribuir muito para um ensino de melhor qualidade, além de facilitar a aprendizagem. Para isso, é necessário que os professores sintam-se estimulados e preparados para enfrentar esta nova realidade. É preciso oferecer subsídios, recursos e capacitação. Com isto, poderão sentir-se capazes de criar seus próprios produtos de *software*, suas próprias *home-pages*, conforme sua metodologia e disciplina, visto que atualmente é difícil encontrar alunos, principalmente de escolas particulares, que não

utilizam o computador para realizar tarefas escolares, pesquisa e/ou lazer. Cabe então à escola preparar seus docentes para enfrentar esta nova realidade e buscar estas ferramentas para o complemento do seu trabalho pedagógico.

De fato, sabe-se que a informática está sendo inserida na educação pela necessidade de transpor barreiras do educar convencional, pois tudo se modernizou na educação. Até o advento da informática se tornou convencional, frente a esta nova forma pedagógica de educação, oportunizando às escolas uma renovação de trabalhar os conteúdos programáticos, propiciando ao aluno eficiência na construção do conhecimento, convertendo a aula em um espaço real de interação, de troca de resultados, adaptando os dados à realidade do educando. [KOSLOWSKI e NUNES 2001]

Além disso, [KOSLOWSKI e NUNES 2001] ainda afirmam que a introdução dos computadores no ensino de 1º e 2º graus não é conseqüência de um modismo. A resolução do governo de aplicar a informática no processo educacional brasileiro resulta da necessidade de minimizar alguns dos problemas do sistema de ensino. O computador surge como um meio auxiliar alternativo de ensino, um recurso a mais para a diminuição das carências, em especial no 1º grau, quanto à repetência e à evasão escolar.

Segundo [VALENTE *apud* KOSLOWSKI e NUNES 2001], o uso da informática em educação não significa a soma da informática e da educação, mas a integração destas duas áreas. Para haver integração, é necessário o domínio dos assuntos que estão sendo integrados. Seguindo esta linha de pensamento, pensar em computadores na educação não significa pensar na máquina, mas sim na educação. Educação e Informática devem ser consideradas como um todo, visando o benefício da sociedade.

Apesar dos aspectos positivos da utilização da informática na escola, percebe-se que a maioria das escolas ainda está a passos lentos e tem dúvidas sobre a real contribuição dos computadores para o ensino, especialmente quando se trata de séries iniciais.

No entanto, a mesma forma aditiva pela qual tem sido pensada a introdução de computadores na educação também vem se aplicando ao processo de preparação de professores. Freqüentemente, tal preparação realiza-se através de cursos ou

treinamentos de pequena duração, para exploração de determinados produtos de *software*. Resta ao professor desenvolver atividades com essa nova ferramenta junto aos alunos, mesmo sem ter a oportunidade de analisar as dificuldades e potencialidades de seu uso na prática pedagógica e, muito menos, de realizar reflexões e depurações dessa nova prática. Os alunos, por crescerem em uma sociedade permeada de recursos tecnológicos, são hábeis manipuladores da tecnologia e a dominam com maior rapidez e desenvoltura que seus professores. Mesmo os alunos pertencentes a camadas menos favorecidas têm contato com recursos tecnológicos na rua, na televisão, etc, e sua percepção sobre tais recursos é diferente da percepção de uma pessoa que cresceu em uma época em que o convívio com a tecnologia era muito restrito [ALMEIDA 2000].

Mas deve-se levar em consideração ainda que os professores treinados apenas para o uso de certos recursos computacionais são rapidamente ultrapassados por seus alunos, que têm condições de explorar o computador de forma mais criativa, e isso provoca diversas indagações quanto ao papel do professor e da educação. O educador preparado para usar o computador através de um produto de *software* pergunta qual será o seu papel e o futuro de sua profissão, em uma sociedade em que afloram outros espaços de conhecimento e de aprendizagem, fora do *lócus* escolar.

Nesse contexto, surge o instrumento computacional a ser utilizado para educar pessoas. Tal recurso é o *software* educacional. Consoante [ZAMBALDE e ALVES 2002], o *software* educacional é um programa desenvolvido para atender aos objetivos educacionais previamente estabelecidos e, para que ele seja efetivo e esteja à altura das necessidades pedagógicas, é necessário que seu desenvolvimento conte com especialistas das áreas de Educação e Informática. Também, como qualquer produto de *software*, os "educacionais" possuem pontos fortes e limitações. É preciso saber quando um produto de *software* é adequado para os objetivos curriculares e pode, por isso, ser integrado ao contexto educacional. O professor deve adotar o produto de *software* que instigue as habilidades cognitivas de seus alunos e, acima de tudo, lhes ofereça situações para que possam transferir seus conhecimentos para a solução de novos problemas [LUCENA 1994 *apud* ZAMBALDE e ALVES 2002].

Entende-se que várias são as ferramentas de autoria encontradas para facilitar a criação de aplicações educacionais nas mais diversas áreas do ensino, mas em geral todas buscam a criação de *software* multimídia. Sabe-se que o uso desta técnica aplicada

a educação teve um elevado crescimento nos últimos anos, mas ressalta-se porém que, o *software* multimídia, apesar da riqueza visual e sonora que comporta, faz com que o aprendiz interaja com o ambiente como se estivesse manuseando um livro eletrônico. Infelizmente, na maioria dos casos, é pouca ou quase nula a possibilidade do aprendiz intervir nesse ambiente para criar ou interagir de forma ativa sobre o objeto de conhecimento. Quando essa interação é permitida, ela consiste apenas na capacidade de adicionar novos textos e imagens ao sistema existente. Diante disso, buscou-se um paradigma alternativo, onde o aprendiz interage ativamente na construção de conhecimento, em ambientes tais como o de jogos, nos quais os participantes são personagens de uma história que se desenrola, e suas ações vão modificar esta história à medida que ela acontece. Os participantes vêem-se frente a situações que exigem criatividade e conhecimento da estrutura do mundo no qual a história se desenrola, para poderem ser resolvidas [SOUZA e WAZLAWICK 1997].

Dessa forma, existem diferentes formas de utilizar o computador na educação e, infelizmente, é mais comum encontrar a sua utilização como objeto de estudo, como facilitador na realização de determinadas tarefas ou como máquina de ensinar que repetem os métodos tradicionais de ensino, conhecidos como instrucionistas. Está-se interessado no uso do computador como objeto que contribua no processo de aprendizagem. Isso ocorre por vários motivos. Em primeiro lugar, existe uma exigência muito grande da sociedade para que os indivíduos saibam manipular o computador, pois ele está presente em quase todas as nossas atividades diárias e, portanto, uma pessoa que não consiga conviver com esse objeto ficará inevitavelmente à margem de um mundo cada vez mais dominado por recursos tecnológicos. Esse argumento tem a sua verdade, mas por outro lado, os educadores não podem se preocupar apenas com esse lado da questão, bastante limitada, voltada apenas para o treinamento de indivíduos na utilização de produtos de *software*. Em segundo lugar, cresce cada vez mais o interesse em conseguir novos mercados de consumidores de produtos de *software* educativos e esses aparecem com os mais variados atrativos gráficos, porém pobres do ponto de vista pedagógico. Diante da falta de conhecimento por parte da população, essa indústria tem encontrado um campo bastante receptivo a esse tipo de produto [MENEZES e VALLI 1997].

Ainda assim, não se deve considerar como digno de uso no ambiente escolar apenas os produtos de *software* construídos especificamente para fins pedagógicos. Além disso, às vezes, os ditos *softwares* educacionais nem mesmo possuem uma proposta pedagógica clara. Dentro da visão que se tem de informática educativa, onde o computador deve ser usado com ferramenta cognitiva, entende-se que deve selecionar produtos de *software* (ditos educacionais ou não) que viabilizem esta forma de uso.

Mas, para usufruir todo o aparato tecnológico disponível, deve-se considerar que uma inovação imposta, decidida ou planejada por organismos externos à instituição é incompatível com a concepção dialética de inovação, pois tende a produzir rejeição ou a adicionar quantitativamente o novo instrumento ao arsenal disponível. As práticas impostas visam à otimização do ensino e não deixam espaço para o desenvolvimento de processos criativos. Isso foi observado com a introdução dos recursos audiovisuais nas escolas e hoje muitas instituições adotam esse procedimento em relação aos microcomputadores [ALMEIDA 2000]. Isso leva à necessidade de preparação de pessoal que saiba lidar com a iniciação à tecnologia, sobretudo professores secundários e universitários. Logo, a preocupação com a formação de professores é uma constante nas universidades que desenvolvem trabalhos relacionados ao uso do computador em educação. O desenvolvimento de atividades de formação se faz por caminhos diferentes, que variam desde a criação de disciplinas específicas que tentam integrar informática e educação até a realização de cursos de pós-graduação.

Além disso, deve considerar que a formação não se encerra com a conclusão de cursos, oficinas ou outros eventos, pois deve ter o caráter de continuidade, que se concretiza por meio de reuniões periódicas, seminários e debates através de redes telemáticas, encontros presenciais e oficinas. Quaisquer que sejam as modalidades de formação escolhidas, sua concretização deve ser coerente com as necessidades do grupo em formação e prever espaço para o estabelecimento de conexões entre teoria, prática e domínio de recursos computacionais. Isso promove uma reorganização e uma transformação da prática pedagógica, segundo a reflexão favorecida pelo ciclo descrição-execução-reflexão-depuração.

3.3 Considerações Finais

Ao englobar conceitos e análise de impactos causados pela presença da informática na educação, percebe-se que a aderência a novas tecnologias deve ser permeada por uma postura crítica e sábia. Isso porque uma vez que o uso de computadores acarreta em elevação de custo, tanto na sua aquisição, quanto na sua manutenção, deve-se preparar o ambiente onde novas tecnologias serão inseridas, para que o retorno esperado seja alcançado. Dessa forma, não haverá prejuízo da aprendizagem e sim novas possibilidades de assimilação de informações e conhecimentos, superando barreiras e limitações antes causadas pela resistência de recursos humanos.

Capítulo 4 – Linguagem de Programação Java

4.1 Considerações Iniciais

Neste capítulo, é apresentada a linguagem de programação Java, ferramenta sobre a qual foi implementado o produto de *software*. Na seção 2, é apresentado um breve histórico da linguagem, causas de seu surgimento e perspectivas oferecidas para a programação. Na seção 3, é definido Java, com tecnologias suportadas e suas implicações. Na seção 4, são listadas as principais características do Java. Na seção 5, é definida a programação orientada a objetos, cuja linguagem Java dá suporte. Por fim, são apresentadas as considerações finais do capítulo.

4.2 Pequeno histórico

Segundo [SILVA 2003], tudo começou em 1991, com um pequeno grupo de projeto da *Sun* Microsystems denominado *Green* que pretendia criar uma nova geração de computadores portáteis inteligentes, capazes de se comunicar de muitas formas, ampliando suas potencialidades de uso. Para tanto, decidiu-se criar também uma nova plataforma para o desenvolvimento destes equipamentos de forma que seu produto de *software* pudesse ser portado para os mais diferentes tipos de equipamentos. A primeira escolha de uma linguagem de programação para tal desenvolvimento foi C++, aproveitando suas características e a experiência dos integrantes do grupo no desenvolvimento de produtos de *hardware* e *software*. Mas, mesmo o C++ não permitia realizar com facilidade tudo aquilo que o grupo visionava.

James Gosling, coordenador do projeto, decidiu então pela criação de uma nova linguagem de programação que pudesse conter tudo aquilo que era considerado importante e que ainda assim fosse simples, portável e fácil de programar. Surgiu a linguagem interpretada *Oak* (carvalho em inglês) batizada assim dada à existência de uma destas árvores em frente ao escritório de Gosling. Para dar suporte a linguagem, também surgiu o *Green OS* e uma interface gráfica padronizada.

Após dois anos de trabalho o grupo finaliza o *Star7* (ou *7), um avançado PDA (*Personal Digital Assistant*) e em 1993 surge a primeira grande oportunidade de aplicação desta solução da *Sun* em uma concorrência pública da *Time-Warner* para desenvolvimento de uma tecnologia para TV a cabo interativa, injustamente vencida pela SGI (*Silicon Graphics Inc.*).

O *Oak*, então rebatizado Java devido a problemas de *copyright*, continua sem uso definido até 1994, quando estimulados pelo grande crescimento da *Internet*, Jonathan Payne e Patrick Naughton desenvolveram o programa navegador *WebRunner*, capaz de efetuar o *download* e a execução de código Java via *Internet*. Apresentado formalmente pela *Sun* como o navegador *HotJava* e a linguagem Java no *SunWorld'95*, o interesse pela solução se mostrou explosivo. Poucos meses depois, a *Netscape Corp.* lança uma nova versão de seu navegador *Navigator* também capaz de efetuar o *download* e a execução de pequenas aplicações Java então chamadas *applets*. Assim se inicia a história de sucesso de Java.

Em uma iniciativa também inédita, a *Sun* decide disponibilizar Java gratuitamente para a comunidade de desenvolvimento de produtos de *software*, embora detenha todos os direitos relativos à linguagem e as ferramentas de sua autoria. Surge assim o *Java Developer's Kit* 1.0 (JDK 1.0). As plataformas inicialmente atendidas foram: *Sun Solaris* e *Microsoft Windows 95/NT*. Progressivamente, foram disponibilizados *kits* para outras plataformas tais como *IBM OS/2*, *Linux* e *Applet Macintosh*.

Em 1997, surge o JDK 1.1 que incorpora grandes melhorias para o desenvolvimento de aplicações gráficas e distribuídas e, no início de 1999, é lançado o JDK 1.2, contendo muitas outras melhorias, de forma que também seja conhecido como Java 2.

Atualmente a *Sun* vem liberando novas versões ou correções a cada nove meses bem como novas API (*Application Program Interface*) para desenvolvimento de aplicações específicas.

Segundo [DOEDERLEIN 2004], por sorte, a evolução de Java não está parada. Como se sabe, o próximo *release* (JDK 1.5, codinome "Tiger") promete muitas melhorias na facilidade de programação. Mas as vantagens não param por aí: existem diversas frentes adicionais ao mesmo sentido, tocadas por outras JSRs, além de algumas

soluções já disponíveis, que mais desenvolvedores poderiam aproveitar para simplificar muito seu desenvolvimento com Java. Muitas melhorias relacionadas à facilidade de uso servirão para evitar a perda de tempo com a parte mais chata e rotineira do desenvolvimento, liberando seus talentos para coisas mais interessantes e produtivas.

Além disso, devido à abordagem conservadora do *Tiger*, que torna muitas das melhorias compatíveis com JVMs 1.4, será possível tirar proveito parcial das novidades muito antes que os *runtimes* 1.5 se tornem amplamente disponíveis e confiáveis. O J2SE 1.5 também inclui novas APIs e muitos aperfeiçoamentos de infra-estrutura.

4.3 O que é Java?

Java é uma nova e poderosa linguagem de programação, desenvolvida pela *Sun Microsystems*. Originalmente concebida para ser usada com a televisão interativa, Java tornou-se uma linguagem de grande interesse para a comunidade *Internet* quando a *Sun* lançou o *HotJava*, um *browser Web* que podia executar pequenos programas Java embutidos, chamados *applets*, dentro das páginas *Web*. Logo depois, o suporte para *applets* Java foi acrescentado ao *Netscape Navigator* 2.0, sem dúvida o mais conhecido *browser Web* que existe. Agora, as *applets* Java embutidas estão se tornando obrigatórias nos *sites Web* mais sofisticados. [THOMAS, PATEL, HUDSON e BALL 1997]

De acordo com [SILVA 2003], Java é resultado de uma busca por uma linguagem de programação que pudesse fornecer uma ligação com C++, mas com segurança. Os primeiros objetivos alcançados com desenvolvimento desta nova linguagem foram:

- criação de uma linguagem orienta a objetos;
- fornecimento de um ambiente de desenvolvimento por dois motivos: velocidade no desenvolvimento – eliminando o ciclo de compilar-linkar-carregar-testar; e portabilidade do código – com um interpretador que especifica a forma do nível do sistema operacional (pode rodar em qualquer tipo de sistema operacional);
- não tem acesso a ponteiros do sistema operacional;
- fornece dinamismo durante a manutenção de programas;

Java é notável não apenas porque *applets* podem ser executadas dentro de páginas *Web*, mas também porque ela é uma linguagem orientada a objetos poderosa e fácil de usar. A linguagem Java lida com muitos dos problemas comuns, porém complexos, que os programadores geralmente encontram ao desenvolver aplicações robustas. Java suporta multiprocessamento com suas classes *thread* (linhas de execução) e executa automaticamente a coleta de lixo, liberando memória que não está mais sendo usada em segundo plano. A *Java Application Programming Interface* (API), incluída no *Java Developers Kit* fornecido pela *Sun*, fornece aos programadores acesso independente de plataforma para ferramentas essenciais na programação de aplicações *Internet* complexas, como soquetes de rede e um sistema de janelas gráficas. Poucas linguagens de programação geram o tipo de interesse que Java tem gerado, mas também poucas linguagens redefinem, como ela, o que são programas e o que os programadores podem fazer. Enquanto os programas de outras linguagens estão confinados a uma determinada plataforma, Java é independente de plataforma [THOMAS, PATEL, HUDSON e BALL 1997].

4.4 Características importantes da linguagem Java

Como forte critério de decisão pela implementação do produto de *software* em linguagem de programação Java, estão seu conjunto de características que a tornam singular perante outras linguagens de programação disponíveis na área de Computação [SILVA 2003]:

- **Orientada a Objetos.** Java é uma linguagem puramente orientada a objetos, pois, com exceção de seus tipos primitivos de dados, tudo em Java são classes ou instância de uma classe. Java atende todos os requisitos necessários para uma linguagem ser considerada orientada a objetos que resumidamente são oferecer mecanismos de abstração, encapsulamento e herança;
- **Independente de Plataforma.** Java é uma linguagem independente de plataforma, pois os programas escritos em Java são compilados para uma forma intermediária de código denominada ***bytecodes*** que utiliza instruções e tipos primitivos de tamanho fixo, ordenação *big-endian* e uma

biblioteca de classes padronizada. Os *bytecodes* são como uma linguagem de máquina destinada a uma única plataforma, a máquina virtual Java (JVM – *Java Virtual Machine*), um interpretador de *bytecodes*. Pode-se implementar uma JVM para qualquer plataforma, assim, um mesmo programa escrito em Java pode ser executado em qualquer arquitetura que disponha de uma JVM;

- **Performance.** Java foi projetada para ser compacta, independente de plataforma e para utilização em rede, o que levou a decisão de ser interpretada através do esquema de *bytecodes*. Como uma linguagem interpretada, a performance é razoável, não podendo ser comparada a velocidade de execução de código nativo. Para superar esta limitação, várias JVM dispõe de compiladores *just in time* (JIT) que compilam os *bytecodes* para código nativo durante a execução otimizando a execução, que nesses casos melhora significativamente a performance de programas escritos em Java;
- **Segurança.** Considerando a possibilidade de aplicações obtidas através de uma rede, a linguagem Java possui mecanismos de segurança que podem, no caso de *applets*, evitar qualquer operação no sistema de arquivos da máquina-alvo, minimizando problemas de segurança. Tal mecanismo é flexível o suficiente para determinar se uma *applet* é considerada segura especificando nesta situação diferentes níveis de acesso ao sistema alvo;
- **Permite Multithreading.** Java oferece recursos para o desenvolvimento de aplicações capazes de executar múltiplas rotinas concorrentemente bem como dispõe de elementos para a sincronização destas várias rotinas. Cada um destes fluxos de execução é denominado *thread*, um importante recurso de programação de aplicações mais sofisticadas. Além disso, Java é uma linguagem bastante robusta, oferece tipos inteiros e ponto flutuante compatíveis com as especificações IEEE, tem suporte para caracteres UNICODE, é extensível dinamicamente além de ser naturalmente voltada para o desenvolvimento de aplicações em rede ou aplicações distribuídas. Tudo isto torna Java uma linguagem de programação única;

- **Garbage Collection.** Java não segura áreas de memória que não estão sendo utilizadas, isto porque ele tem uma alocação dinâmica de memória em tempo de execução. Em C e C++ (e em outras linguagens), o programa desenvolvido é responsável pela alocação e desalocação da memória. Durante o ciclo de execução do programa, Java verifica se os objetos instanciados estão sendo utilizados. Caso não estejam, Java libera automaticamente esta área que não esta sendo utilizada.

Além disso, conforme [SILVA 2003], deve-se salientar também as diferenças da linguagem de programação Java que a tornaram vantajosa em detrimento da linguagem C++:

- Java não implementa ponteiro explicitamente;
- não tem gabarito;
- inexistência de aritmética de ponteiros (ponteiros são apenas referências);
- *arrays* são objetos;
- *Strings* são objetos;
- não há a necessidade de desalocação explícita de memória;
- os tamanhos (em *bits*) dos tipos primitivos são os mesmos em todas as plataformas;
- não há sobrecarga de operadores;
- não há métodos com lista de argumentos de tamanho variável;
- não existem diretivas de pré-compilação;
- não existem Modelos (*Templates*);
- não existe herança múltipla com classe, mas apenas com interfaces;
- não existem funções, mas apenas métodos de classes.

4.5 Programação Orientada a Objetos

Entende-se que Java é parcialmente bom porque ela é nova. Uma das vantagens mais importantes de começar do zero é que Java foi criada para ser uma linguagem de programação orientada a objetos. Como a orientação a objetos talvez seja

a tendência mais conhecida no desenvolvimento moderno de produtos de *software*, pode-se colocar uma marca na coluna de palavras da moda. A orientação a objetos fornece um nível de abstração sobre as ações dos programas, permitindo solucionar problemas definindo objetos [THOMAS, PATEL, HUDSON e BALL 1997].

Segundo [SILVA 2003], a programação orientada a objetos é baseada na escrita de programas em termos de objetos (coisas) que compõe um sistema. Um objeto representa uma entidade do mundo real (coisa) que pode armazenar dados (atributos) e possui um conjunto específico de operações (métodos) que são realizadas nele.

A programação orientada a objetos é uma técnica nova para a concepção e implementação de produtos de *software* e centraliza-se nos seguintes principais conceitos: tipos de dados abstratos, classes, hierarquias de tipos (subclasses), herança e polimorfismo. Segundo [SILVA 2003], o ponto marcante da programação orientada a objetos é a possibilidade de "reutilizar código", ou seja, um objeto pode usar as implementações de um outro objeto e estendê-las para alcançar um objetivo específico.

Quando se trabalha com orientação a objetos, todos os objetos são organizados em classes. São definidas classes bases e estas são estendidas de forma a criar novas classes, o que torna o produto de *software* orientado a objeto muito mais fácil de ser documentado e estendido [SILVA 2003].

4.6 Considerações Finais

Com todo esse aparato oferecido pela linguagem de programação Java, ela acabou por se tornar a linguagem preferida para implementar aplicativos baseados em *Intranet* e *Internet* e produtos de *software* para dispositivos que se comunicam através de uma rede. Não é de surpreender se um novo estéreo e outros dispositivos domésticos estiverem conectados em rede utilizando tecnologia Java! Além disso, Java é particularmente atraente como primeira linguagem de programação. No evento JavaOne^{TM}, em junho de 2001, foi anunciado que Java é parte obrigatória do currículo de linguagens de programação de 56% das faculdades e universidades dos EUA. Ademais, 87% das faculdades e universidades americanas oferecem cursos sobre Java. Java também é atraente para as escolas de segundo grau. Em 2003, o *College Board* estabeleceu Java como padrão para cursos de Ciência da Computação de colocação

antecipada [DEITEL 2003]. Logo, diante de todos esses fatos, a experiência de implementação de produtos de *software* em linguagem Java se torna muito importante, tanto para usuários quanto para programadores, como forma de inserção no atual universo tecnológico.

Capítulo 5 – TBC-AED: Desenvolvimento do Produto de *Software* e seus Reflexos

5.1 Considerações Iniciais

Neste capítulo, é apresentado o produto de *software* **TBC-AED – Treinamento Baseado em Computador para Algoritmos e Estruturas de Dados**, fruto da pesquisa realizada durante o ano de 2004 na área de Informática na Educação e desenvolvimento de produtos de *software* em ambiente Java. Na seção 2, é realizada uma breve análise do desenvolvimento do **TBC-AED**, considerando obstáculos enfrentados e vantagens a serem alcançadas. Na seção 3, são apresentadas a estrutura e a forma de organização do produto de *software*, incluindo recursos utilizados para sua construção e a forma de estudar algoritmos e estruturas de dados pelo programa. Na seção 4, são descritos os temas abordados pelo programa, apresentando conceitos e seus modelos de apresentação gráfica. Por fim, são apresentadas as considerações finais do capítulo.

5.2 Análise metodológica do desenvolvimento do TBC-AED

Assimilar o conceito e o desenvolvimento de algoritmos é alvo de muitas dificuldades enfrentadas pelos alunos de cursos da área de Computação. Alunos que apresentam deficiências advindas dos ensinos fundamental e médio encontram grandes dificuldades em aprender conceitos simples de programação de computadores [ALVES e COSTA 2004].

Assim sendo, o **TBC-AED** é o produto de *software* que vem ao encontro dos objetivos discutidos no capítulo 1, buscando corrigir este problema da educação básica. Ele procura analisar tópicos básicos de programação, englobando um conteúdo teórico sintético e direto, acompanhado de processo gráfico passo-a-passo. Isso facilita a visualização e o entendimento das informações ora apresentadas e disponibiliza o tempo que seria gasto em transcrever explicações do quadro-negro para o caderno para a resolução de exercícios de aplicação e fixação. Além disso, pode-se alcançar maior

interação entre o professor e seus alunos, no sentido de aumentar o espaço para questionamentos.

Foi observado que o **TBC-AED**, mais que uma simples ferramenta, é um verdadeiro material de coleta de informações a respeito de programação, algoritmos e estruturas de dados. Ele representa uma experiência profissional, uma vez que é um produto desenvolvido para fins de consumo que, neste caso, é gratuito e direcionado aos alunos da área de Computação e Informática. Além disso, ele visa atender aos alunos que estudam algoritmos e estruturas de dados básicas, mais especialmente os da disciplina Algoritmos e Estruturas de Dados II oferecida pelo DCC/UFLA. Isso é importante para tornar maior a interação graduando-computador, o que propicia estímulos de curiosidade, atenção e eficácia no desenvolvimento de trabalhos relacionados e do raciocínio lógico.

Durante o processo de embasamento teórico-prático, vários exemplos foram vistos e estudados para que o **TBC-AED** pudesse ser elaborado com cuidado especial, o que, *a priori*, não despertava qualquer tipo de receio; além disso, o seu conteúdo era de conhecimento. No entanto, por se tratar de um processo complexo e inovador e devido ao fato do orientado estar cursando estágios iniciais da faculdade, gerou dificuldades iniciais expressivas. Estas estavam relacionadas ao paradigma de programação orientado a objetos e à linguagem de programação Java, que tiveram que ser acompanhadas e foram aos poucos resolvidas. Isso levou a uma aquisição relativamente positiva de novos conhecimentos e de experiências na resolução de problemas – que são os principais desafios dos ingressos em cursos superiores na área.

Com a introdução e a sedimentação de informações, o **TBC-AED** começou a ser desenvolvido e passou por várias etapas de melhorias, aplicadas tanto à forma de apresentação, quanto ao processamento dos algoritmos propriamente dito.

Dentre as vantagens da abordagem construtiva, estão mecanismos para facilitar o processo de abstração, o fato de a animação refletir a interação com o aprendiz e as várias facilidades para a detecção visual de erros. Com isso, incentiva-se o processo de compreensão e autocorreção [GARCIA, REZENDE e CALHEIROS 1996]. Isso pode ser facilmente percebido em processos recursivos, os quais são de difícil explicação teórica, mas que podem ser vistos facilmente através da animação gráfica

disponível em ambientes que utilizam algum algoritmo desse tipo (como *merge sort* e *quick sort*).

Com isso, verifica-se que a organização do **TBC-AED** é amplamente didática e esse fato serve de grande utilidade para o ensino de disciplinas que apresentam como ementa os tópicos relacionados. Além disso, é uma experiência desafiadora de graduandos, mestres e doutores a tornar o ensino de Computação mais dinâmico, ao despertar o seu interesse a terem a mesma iniciativa, se possível com igual dedicação e empenho, para melhorar e aprimorar a formação de recursos humanos para a tecnologia.

5.3 Organização e estrutura do TBC-AED

Quanto à organização, o **TBC-AED** apresenta uma tela principal, contendo uma barra de *menu* separando os temas, que direcionam o usuário para o assunto de seu interesse (Figura 1). Ao selecionar um desses temas, um *submenu* é apresentado, contendo um item explicativo do tema (Figura 2) e outros itens de representação gráfica do tema. A Figura 3 mostra uma janela relativa à representação gráfica do tópico **Busca Binária**, o qual apresenta uma breve introdução a respeito do assunto pesquisado (na parte superior), o algoritmo em *Portugol*[1] (à esquerda), o painel de animação, onde o algoritmo será executado graficamente (à direita), contendo uma legenda, e um conjunto de botões (na parte inferior). Exceto este, os demais temas contém uma janela de mensagem interna usada para o efeito de passo-a-passo na parte inferior à direita (Figura 4). A única tela que se diferencia dos padrões descritos é relativa ao tema **Árvore Binária de Busca** que, devido à sua peculiaridade de representação perante aos demais temas, demandou maior espaço para o processo gráfico (Figura 5). As mudanças ocorreram no posicionamento interno das informações: o algoritmo em *Portugol* possui uma área menor contendo na parte inferior uma janela de mensagem interna, usada para o efeito de passo-a-passo, e o painel de animação é exibido em uma área maior, contendo na parte inferior um espaço para a exibição dos elementos quando o usuário requisitar uma das formas de se imprimir uma árvore binária (pré-ordem, in-ordem ou pós-ordem) e uma legenda.

[1] **Portugol** ou **pseudocódigo** consiste em analisar o enunciado do problema e escrever, por meio de regras predefinidas, os passos a serem seguidos para a resolução do problema [CAMPOS *et al* 2003].

Figura 1. Tela principal do TBC-AED

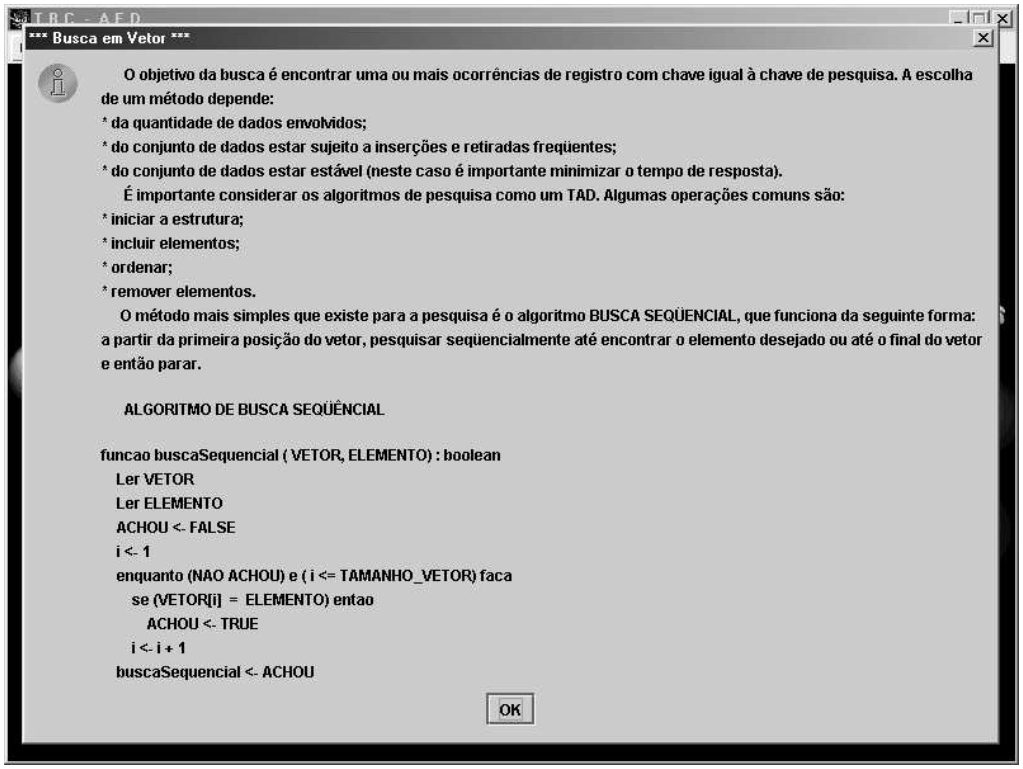

Figura 2. Janela de mensagem exibida contendo uma introdução a respeito do tópico relacionado

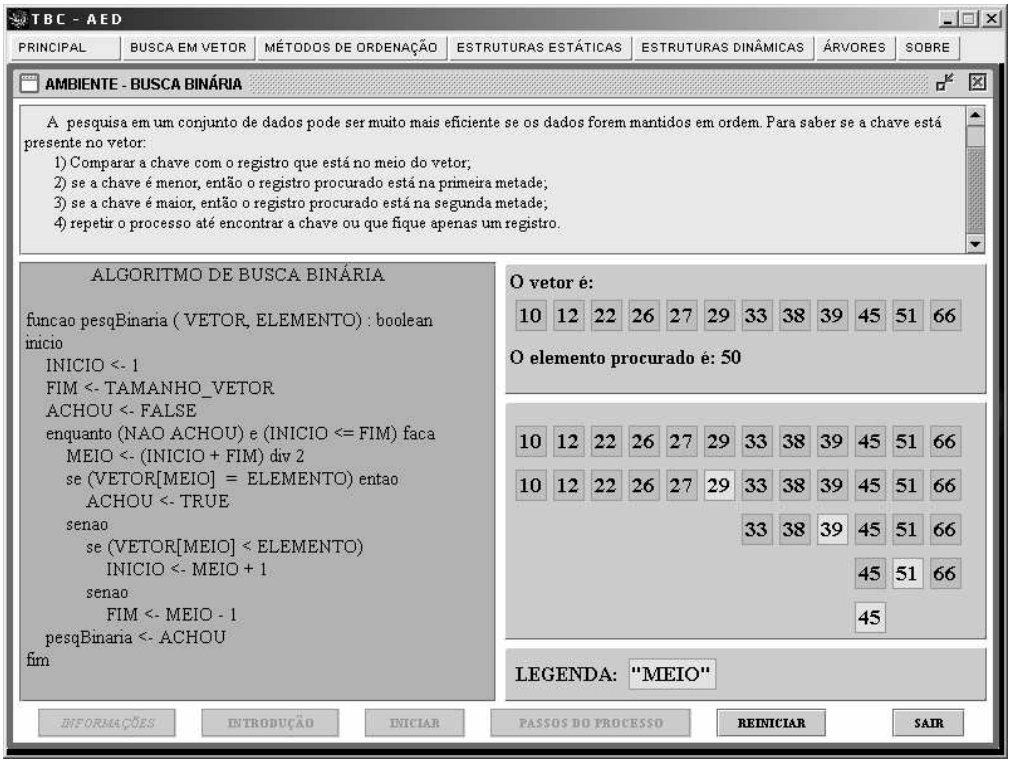

Figura 3. Tela Busca Binária

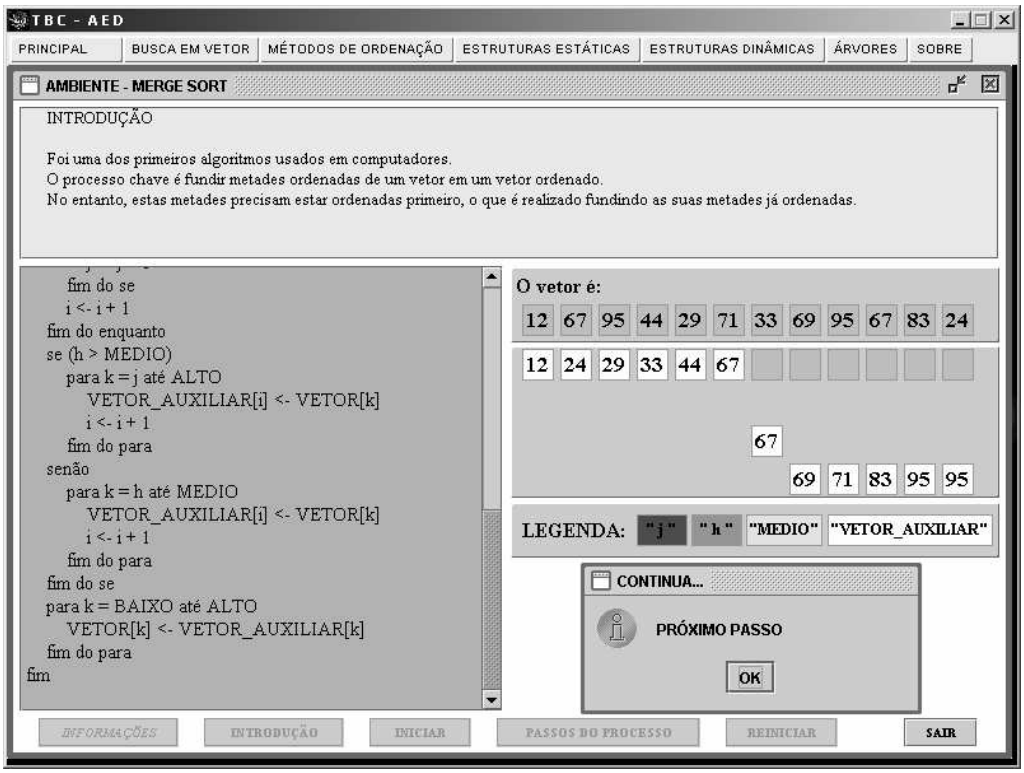

Figura 4. Tela *Merge Sort*, com janela de mensagem interna usada para o efeito de passo-a-passo

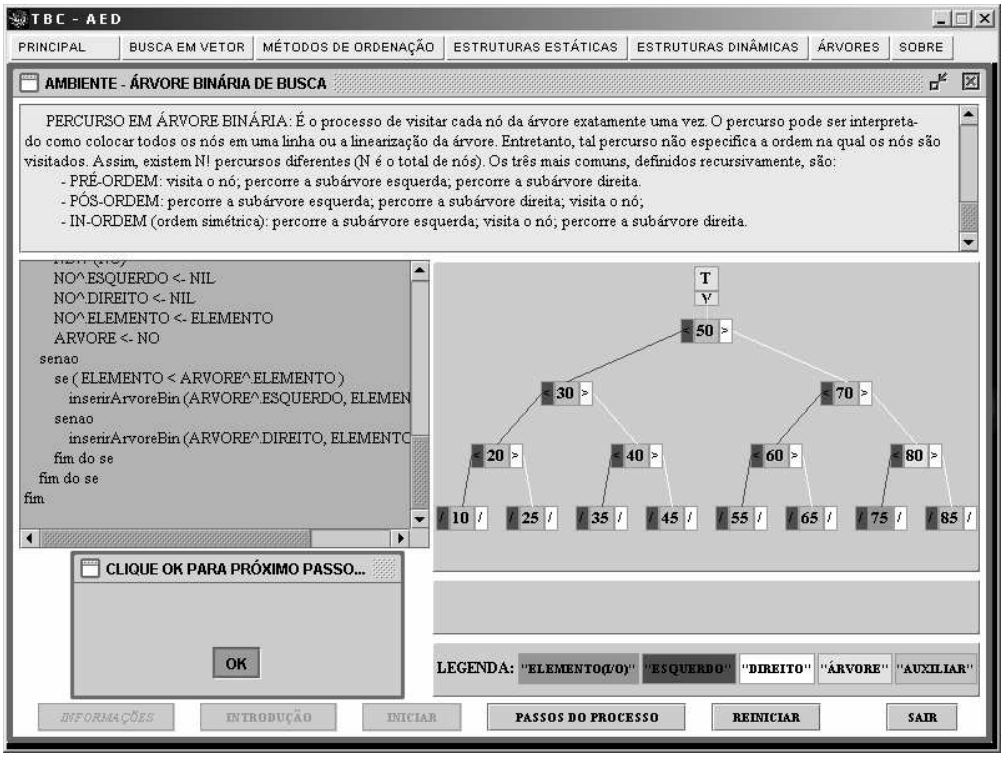

Figura 5. Tela Árvore Binária, com suas peculiaridades

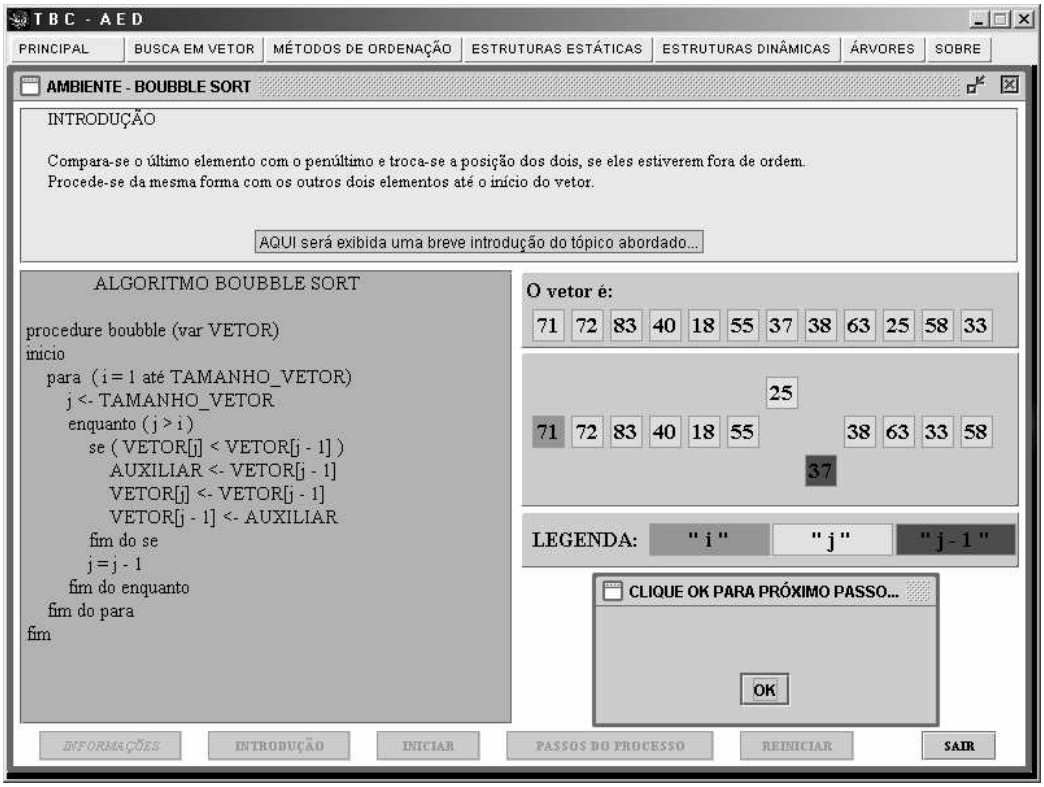

Figura 6. O *mouse* sobre a região da introdução exibe uma breve mensagem sobre ela

Uma característica importante do **TBC-AED** é ser auto-explicativo, bastando deslizar com o *mouse* sobre as partes da tela para ver uma breve mensagem sobre a região posicionada (Figura 6). Além disso, as janelas nas quais os ambientes gráficos passo-a-passo são apresentados exibem apenas os botões os quais são de utilidade para o usuário, dada a parte do processo em que ele se encontra estudando. Isso deixa o usuário mais à vontade e sem preocupações quanto a quaisquer peculiaridades, também por se tratar de um produto de *software* direcionado para alunos iniciantes aos cursos de Computação.

Quanto à estrutura, o **TBC-AED** foi elaborado segundo um *JFrame*[2] principal, que se apresentará como a sua tela principal, contendo o nome e o logotipo. Esta tela contém uma barra de menu, onde estarão os itens de menu e seus sub-itens, separando os temas, que direcionarão o usuário para o assunto de seu interesse. Os itens de menu representam os temas lecionados na disciplina (Figura 7) e os seus sub-itens, um deles como direcionador para uma janela na mensagem explicativa acerca do tema (Figura 8) e os demais como direcionadores para tópicos de representação gráfica do tema a ser estudado (Figura 9). Quando o sub-item que direciona para a janela de mensagem é acionado, uma caixa de diálogo de mensagem é exibida (método *JOptionPane.showMessageDialog*[3]), contendo uma introdução a respeito do tópico relacionado (como mostrado na Figura 2). Quando um sub-item que direciona para uma tela é acionado, um *JInternalFrame*[4] é exibido (como mostrado na Figura 3).

[2] **JFrame** é "uma janela com uma barra de título e uma borda. A classe **JFrame** é uma subclasse de **Java.awt.Frame** (que é uma subclasse de **Java.awt.Window**)" [DEITEL 2003].
[3] O método **JOptionPane.showMessageDialog** é um método especial da classe **JOptionPane** do pacote **javax.swing**. Esse método exibe uma caixa de diálogo para informar o usuário sobre algum evento [DEITEL 2003].
[4] **JInternalFrame** é uma janela-filha que pode ser alternada com outras de mesmo tipo ou que pode ser fechada, gerenciadas por uma janela-pai e que não alteram-na [DEITEL 2003].

Figura 7. Barra de *menu* do TBC-AED

Figura 8. Item explicativo do tema Métodos de Ordenação

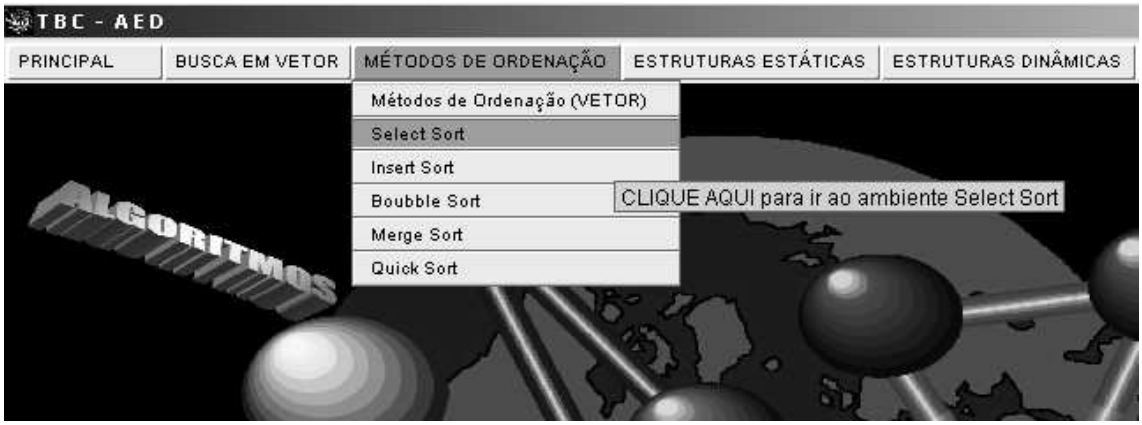

Figura 9. Item de representação gráfica do tópico *Select Sort*, um dos métodos de ordenação disponíveis

5.4 Temas abordados pelo TBC-AED: conceitos e aplicação

O **TBC-AED** visa abordar temas envolvidos a um tópico em Ciência da Computação denominado **Tipo Abstrato de Dados**, construído a partir dos **Tipos de Dados** existentes. Um tipo de dados caracteriza o conjunto de valores a que uma constante pertence ou que pode ser assumido por uma variável ou expressão, ou que pode ser gerado por uma função. Por exemplo, variável do tipo *Boolean*, variável do

tipo inteiro, variável do tipo real. Um tipo abstrato de dados (TAD) pode ser visto como um modelo matemático acompanhado das operações sobre o modelo. Por exemplo, o conjunto de inteiros e suas operações (soma, subtração, etc), lista de inteiros e suas operações (fazer lista vazia, obter o 1º elemento, inserir elemento, remover elemento, etc). Essas informações estão contidas no *submenu* "TAD" do *menu* "Principal" do **TBC-AED**.

5.4.1 Busca em Vetor

Freqüentemente, o programador trabalhará com grandes quantidades de dados armazenados em vetores (*arrays*), análogos em matemática a uma matriz-linha, definidos cada qual como um novo tipo de dado cujos valores são agregados homogêneos de um tamanho definido construído sobre um tipo primitivo. Conforme [DEITEL 2003], pode ser necessário determinar se o *array* contém um valor que corresponde a um determinado valor-chave. O processo de localizar o valor de um elemento particular em um *array* chama-se pesquisa. Dentre as técnicas de pesquisa, tem-se a técnica simples da pesquisa linear (busca seqüencial) – que consiste em percorrer todo o *array* comparando cada elemento do vetor ao elemento procurado – e a técnica mais eficiente da pesquisa binária (busca binária).

O algoritmo de pesquisa binária pressupõe que os elementos no *array* estão ordenados, por exemplo em ordem crescente, e elimina metade dos elementos no *array* que está sendo pesquisado a cada comparação. O algoritmo localiza o elemento do meio do *array* e o compara com a chave de pesquisa. Se forem iguais, a chave de pesquisa foi localizada e a pesquisa binária retorna o valor desse elemento. Caso contrário, a pesquisa binária reduz o problema, pesquisando metade do *array*. Se a chave de pesquisa foi menor que o elemento do meio do *array*, a primeira metade do *array* será pesquisada; caso contrário, a segunda metade do *array* será pesquisada. Se a chave de pesquisa não for o elemento do meio do *subarray* (um pedaço do *array* original) especificado, o algoritmo será repetido para um quarto do *array* original. A pesquisa continua até que a chave de pesquisa seja igual ao elemento do meio de um *subarray* ou até que o *subarray* consista em apenas um elemento que não é igual à chave de pesquisa (isto é, a chave de pesquisa não foi localizada). Essa é uma enorme melhora no

desempenho em relação à pesquisa linear que exigia comparação da chave de pesquisa com todos os elementos existentes em posições subseqüentes, da primeira até a posição em que se encontra o elemento, ou até a última posição, encontrando-o ou não, no pior caso [DEITEL 2003].

Baseado nestas definições, para agilizar e melhorar a apresentação do tema, o **TBC-AED** apresenta os conceitos e um ambiente gráfico relacionados à busca em vetor. Analisando o funcionamento do ambiente gráfico:

- ao clicar no item de menu "Busca em Vetor", sub-item "Busca", uma janela de mensagem é exibida, apresentando os conceitos relacionados ao tema e o algoritmo em *Portugol* do método de busca seqüencial, conforme visto na Figura 2;
- ao clicar no item de menu "Busca em Vetor", sub-item "Busca Binária", uma janela interna é exibida, contendo o ambiente gráfico passo-a-passo, mostrado na Figura 3. A partir de então, o usuário pode começar a estudar e entender esse algoritmo;
- devido a apenas os botões necessários em dado instante estarem habilitados, ao iniciar, o usuário deve clicar no botão Informações para que uma janela de mensagem seja exibida, contendo algumas considerações sobre o funcionamento do programa (Figura 10);
- ao clicar em *OK*, o botão Introdução estará livre para que o usuário possa prosseguir com seus estudos. Ao clicar nele, uma breve introdução é exibida na área superior do ambiente gráfico, além de uma janela de mensagem direcionando o usuário em sua forma de prosseguir no programa (Figura 11);
- feito isso, o botão Iniciar estará disponível para que o usuário possa clicar nele para exibir o algoritmo de busca binária e uma janela de diálogo a fim de que o usuário entre com o número de elementos do vetor a ser gerado sobre o qual deseja que a operação seja realizada (Figura 12);
- ao fornecer o tamanho do vetor, o painel superior à direita exibe a estrutura com no máximo 12 números de dois algarismos gerados aleatoriamente. Uma nova janela de mensagem é exibida para que o usuário possa fornecer

com o valor a ser procurado na estrutura, usando o processo de busca binária (Figura 13);

- a partir desse momento, uma janela de diálogo contendo uma dica de como prosseguir durante o processo gráfico é exibida (Figura 14) e o botão Passos do Processo é habilitado para que o usuário possa acompanhar o processo passo-a-passo. Além disso, uma legenda é exibida na parte inferior à direita (Figura 15). Ao ir pressionando o botão mencionado, alguns sub-vetores são exibidos, mostrando as partes do vetor original sobre as quais está sendo processada a busca binária e o elemento mediano de cada parte (Figura 16);
- ao fim da busca, uma janela de mensagem é exibida informando se o elemento foi ou não encontrado (Figura 17).

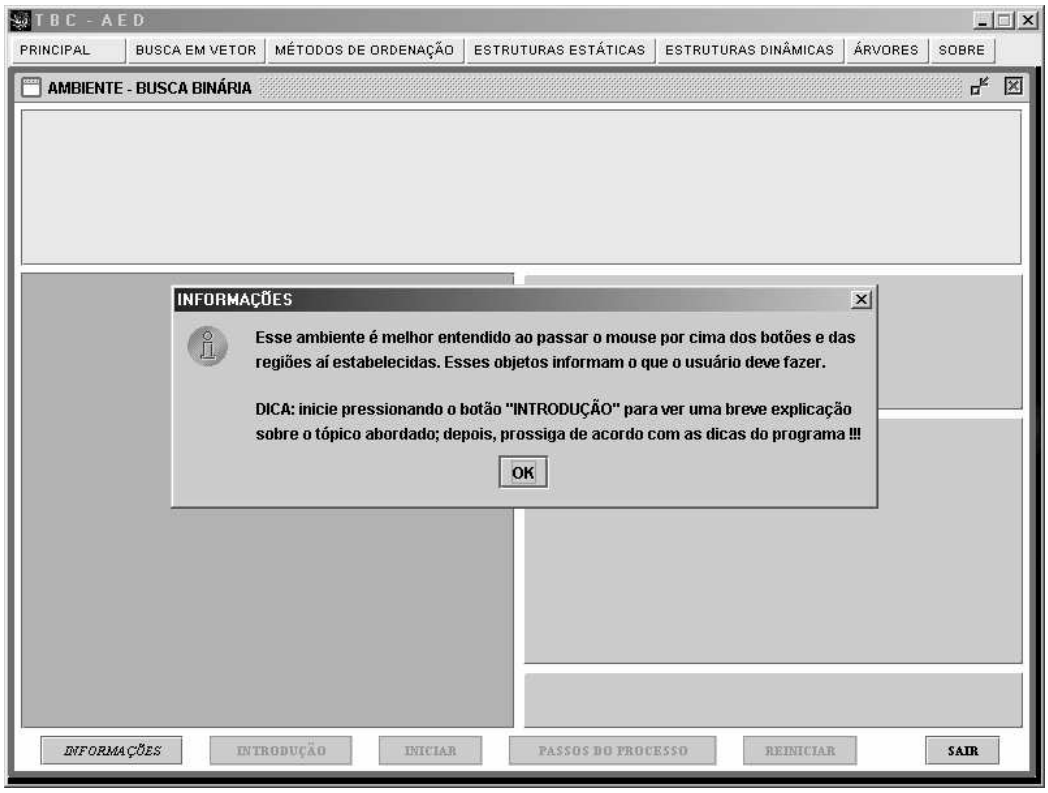

Figura 10. Janela de mensagem exibida ao clicar no botão Informações

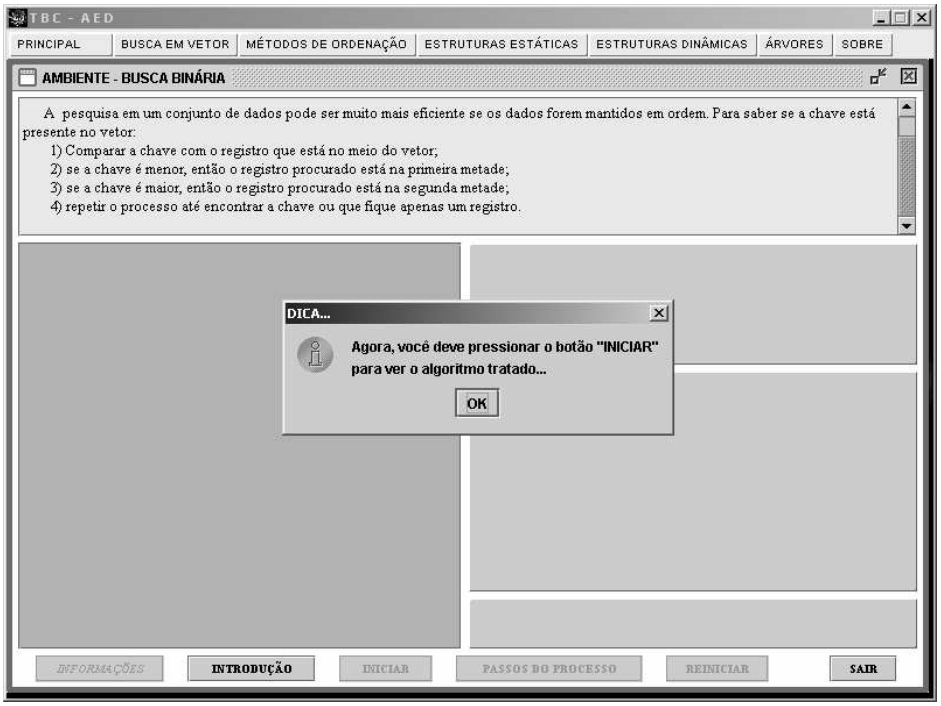

Figura 11. Ao clicar no botão Introdução, a área superior é preenchida com uma breve introdução e uma janela de mensagem é exibida

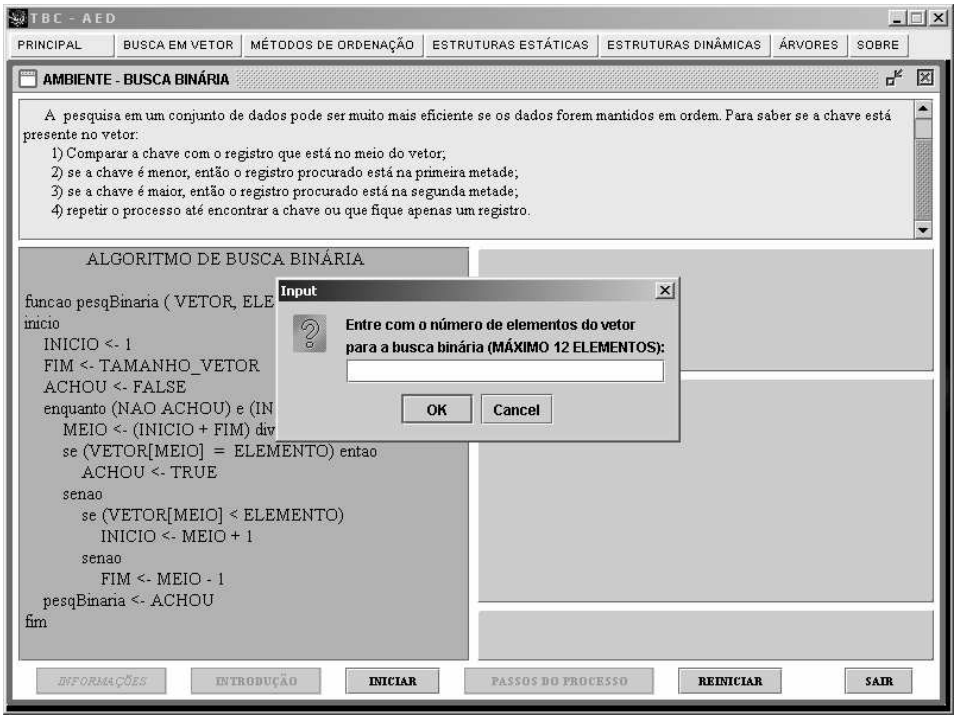

Figura 12. Uma janela de mensagem é exibida para que o usuário entre com o número de elementos

Figura 13. Uma janela de mensagem é exibida para que o usuário possa entrar com o elemento a ser procurado

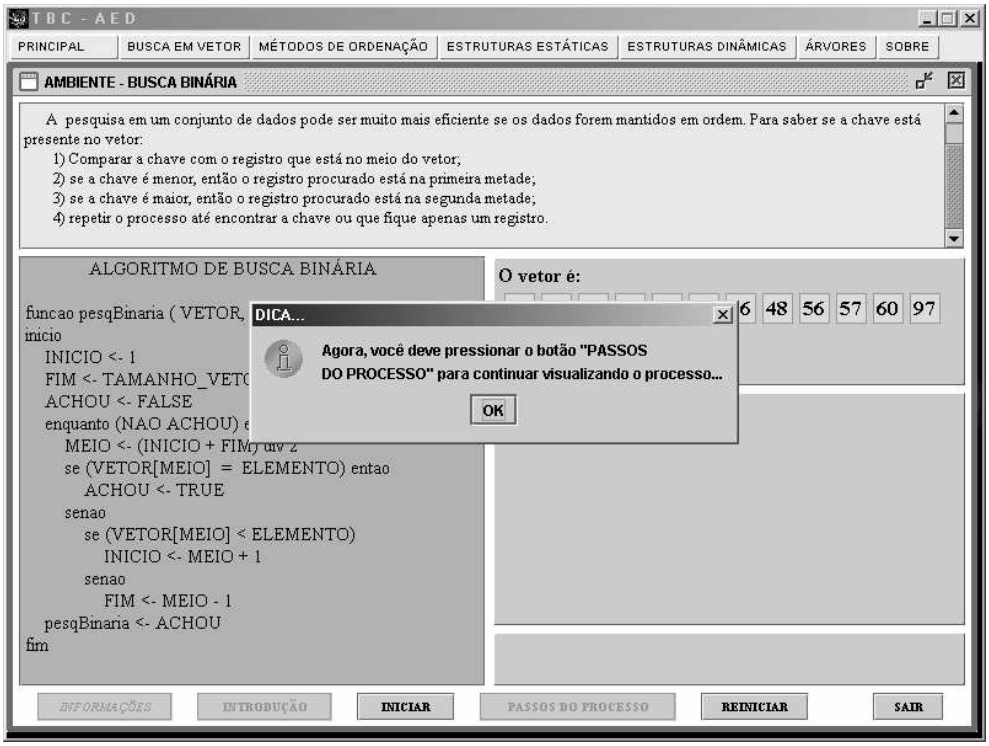

Figura 14. Janela de mensagem exibida quando do advento do processo gráfico

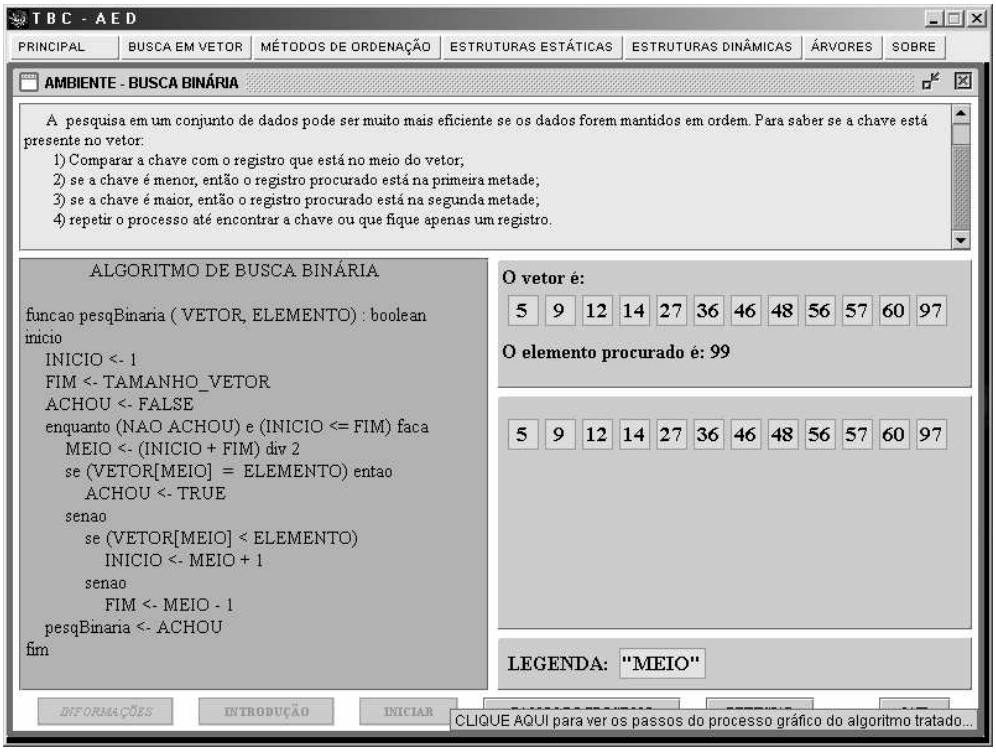

Figura 15. O botão Passos do Processo é habilitado para iniciar o processo gráfico

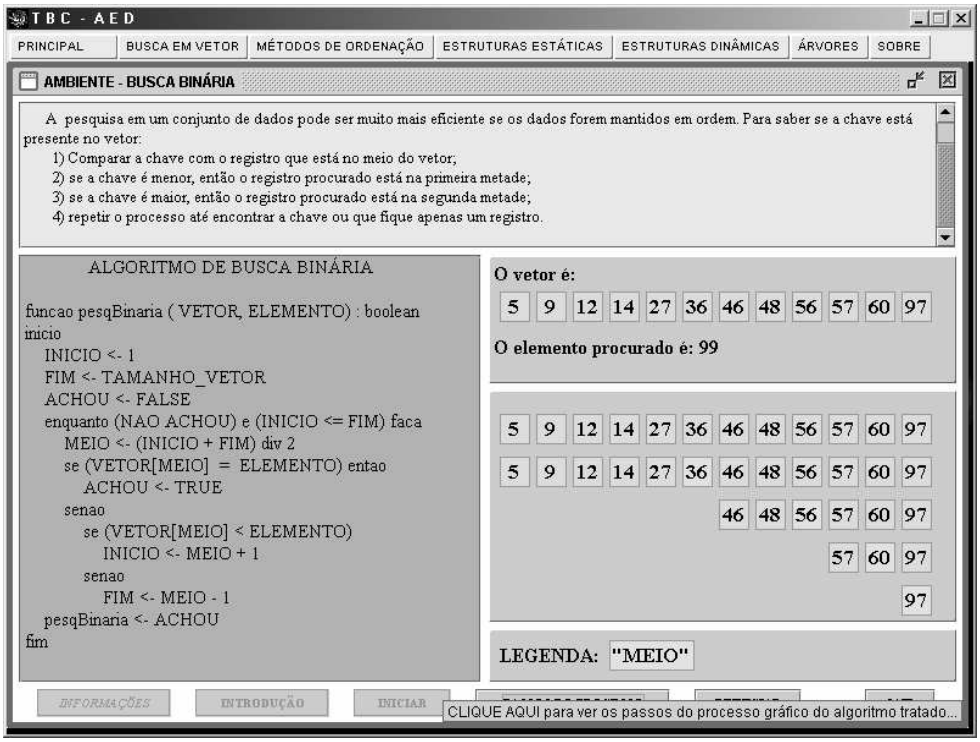

Figura 16. Cada um dos sub-vetores é exibido a cada clique sobre o botão Passos do Processo

Figura 17. Janela de mensagem contendo o resultado da pesquisa

Analisando o decorrer deste tema do **TBC-AED**, percebe-se que há uma facilidade maior em apresentar o conteúdo e considerações acerca do problema a ser discutido. Sugere-se como trabalho aos discentes implementar outros tipos de pesquisa em vetor, por exemplo a busca ternária, para que o aluno possa assimilar o conteúdo e consolidar seu conhecimento sobre o assunto.

5.4.2 Métodos de Ordenação

Os algoritmos de ordenação constituem bons exemplos de como resolver problemas utilizando computadores. As técnicas de ordenação permitem apresentar um conjunto amplo de algoritmos distintos para resolver uma mesma tarefa. Dependendo da aplicação, cada algoritmo considerado possui uma vantagem particular sobre os outros. Além disso, os algoritmos ilustram muitas regras básicas para a manipulação de estruturas de dados [ZIVIANI 1992].

Ordenar corresponde ao processo de rearranjar um conjunto de objetos em uma ordem ascendente ou descendente. O objetivo principal da ordenação é facilitar a recuperação posterior de itens do conjunto ordenado. Imagine como seria difícil utilizar

um catálogo telefônico se os nomes das pessoas não estivessem listados em ordem alfabética. A atividade de colocar as coisas em ordem está presente na maioria das aplicações onde os objetos armazenados têm que ser pesquisados e recuperados, tais como dicionários, índices de livros, tabelas e arquivos [ZIVIANI 1992].

No **TBC-AED**, foram implementados os seguintes métodos de ordenação: *Select Sort, Insert Sort, Boubble Sort, Merge Sort* e *Quick Sort*. Seus conceitos são apresentados nas seções subseqüentes.

5.4.2.1 Select Sort

Segundo [ZIVIANI 1992], um dos algoritmos mais simples de ordenação é o método *Select Sort*, cujo princípio de funcionamento é : selecione o menor item do vetor e troque-o com o item que está na primeira posição do vetor. Repita estas duas operações com os $n - 1$ itens restantes, depois com os $n - 2$ itens, até que reste apenas um elemento.

Apesar de ser um método de fácil entendimento, sua apresentação através de um ambiente gráfico como o **TBC-AED** é de grande utilidade, pois permite ao usuário verificar seus conhecimentos através da movimentação de elementos com cores diferentes, o que realça os passos do algoritmo. Analogamente, o tópico *Select Sort* possui os mesmos procedimentos iniciais do tema Busca Binária, exceto a partir do funcionamento do processo gráfico. Agora, o usuário terá que clicar em *OK* dentro de uma janela de mensagem exibida na parte inferior da área de processo gráfico. Isso pode ser verificado na Figura 18.

5.4.2.2 Insert Sort

Este é o método preferido dos jogadores de cartas, conforme [ZIVIANI 1992]. Em cada passo, a partir de i = 2 (onde i é a posição correspondente a um elemento no vetor a ser ordenado), o *i*-ésimo item da seqüência fonte é pego e transferido para a seqüência destino, sendo inserido no lugar apropriado. Como mostrado na Figura 19, este tópico em nada difere do descrito anteriormente no que se refere ao funcionamento e organização, apesar de ainda ser um dos métodos de ordenação mais simples.

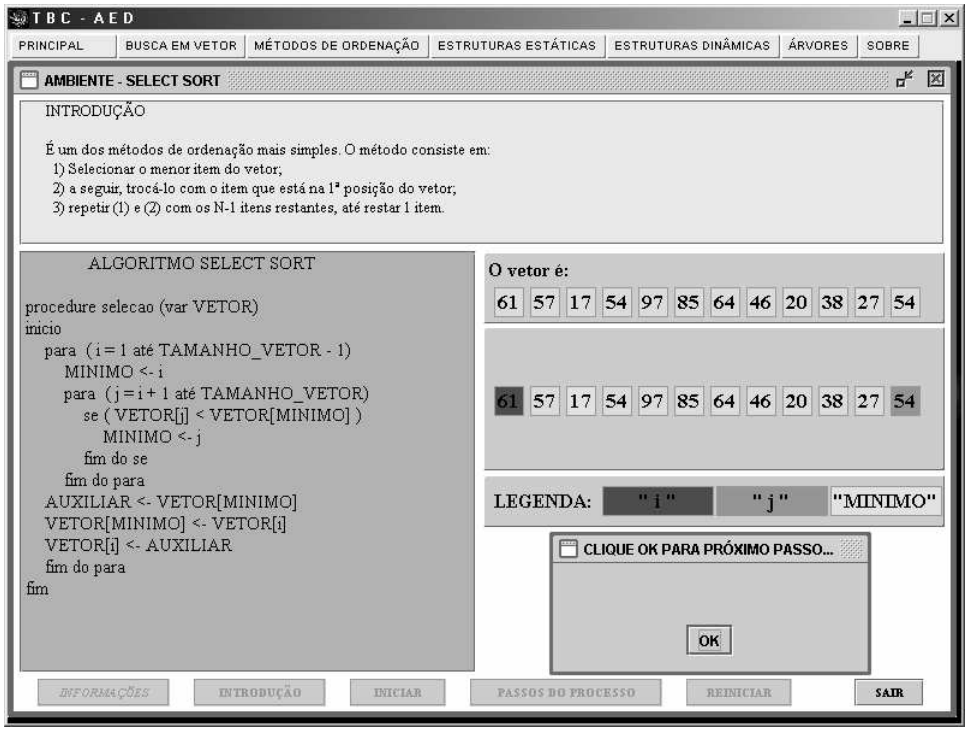

Figura 18. Tópico *Select Sort* do tema Métodos de Ordenação

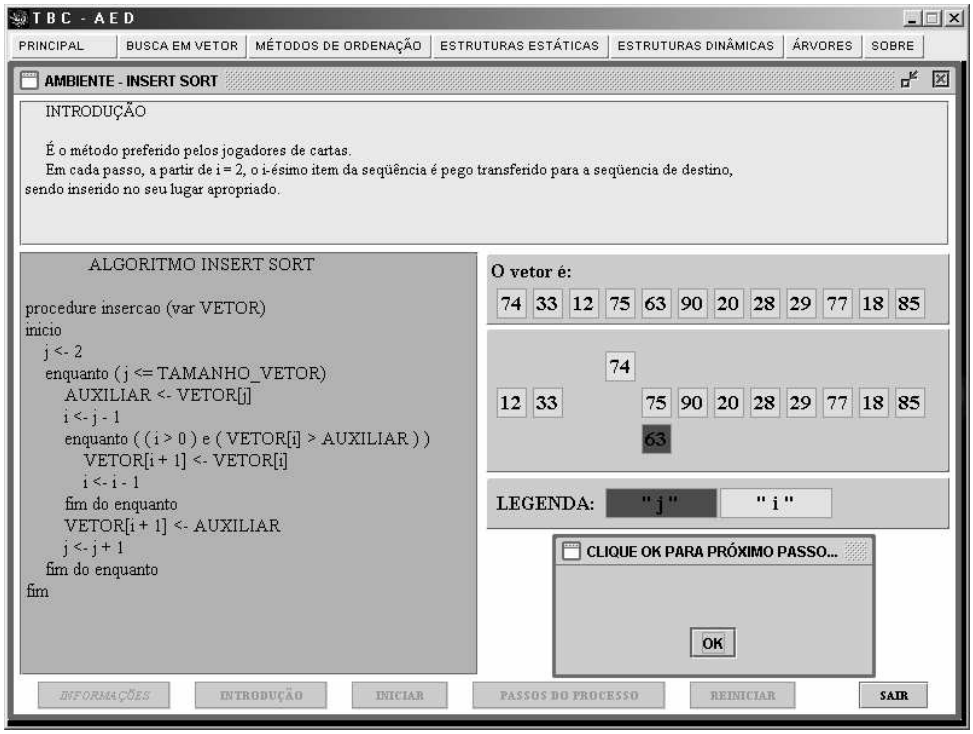

Figura 19. Tópico *Insert Sort* do tema Métodos de Ordenação

5.4.2.3 Boubble Sort

No *Boubble Sort*, compara-se o último elemento com o penúltimo e troca-se a posição dos dois, se eles estiverem fora de ordem. Procede-se da mesma maneira com os outros números até o início da lista, isto é, compara-se o penúltimo com o antepenúltimo e assim sucessivamente. Se os elementos forem testados desde o último até o primeiro, sem ser feita nenhuma troca é porque já estão ordenados. Caso contrário, isto é, toda vez que houver pelo menos uma troca, volta-se ao fim do vetor e repete-se o mesmo procedimento. Deve-se, no entanto, considerar que no primeiro retorno é suficiente fazer as comparações, desde o último elemento até o segundo, pois, neste caso, o menor de todos já borbulhou (método da bolha), isto é, o menor elemento ficou arrumado na primeira posição do vetor (Figura 20).

5.4.2.4 Merge Sort

Foi um dos primeiros algoritmos usados em computadores. O processo chave é fundir metades ordenadas em um vetor ordenado. No entanto, estas metades precisam estar ordenadas primeiro, o que é realizado fundindo as suas metades ordenadas. Isso pode ser feito de forma recursiva, o que torna mais complexo o entendimento deste algoritmo. Devido a esse fato, sua implementação demandou um certo tempo. No entanto, isso foi superado e a tela *Merge Sort* foi elaborada e apresentada da mesma forma que os demais métodos de ordenação (Figura 21).

5.4.2.5 Quick Sort

Quick Sort é o algoritmo de ordenação mais rápido que se conhece para uma ampla variedade de situações, sendo provavelmente mais utilizado do que qualquer outro algoritmo. Segundo [ZIVIANI 1992], a idéia básica é partir do problema de ordenar um conjunto com *n* itens em dois problemas menores. A seguir, os problemas menores são ordenados independentemente e depois os resultados são combinados para produzir a solução do problema maior. A parte mais delicada deste método é relativa ao procedimento partição, o qual tem que rearranjar o vetor *A*[1..n] através da escolha arbitrária de um item do vetor chamado pivô, de tal forma que as três condições abaixo sejam simultaneamente satisfeitas:

1. o pivô x é colocado no seu lugar definitivo *A*[k], para algum índice k do vetor;
2. todos os itens em *A*[1], *A*[2], ..., *A*[k - 1] são menores ou iguais a *A*[k];
3. todos os itens em *A*[k + 1], *A*[k + 2], ..., *A*[n] são maiores ou iguais a *A*[k].

Este comportamento pode ser descrito pelo seguinte algoritmo: primeiro, escolha arbitrariamente um item do vetor e coloque-o em um variável x (no **TBC-AED**, foi escolhido o primeiro elemento do vetor). A seguir, percorra o vetor a partir da esquerda até que um item $A[i] \geq x$ é encontrado; da mesma forma percorra o vetor a partir da direita até que um item $A[j] \leq x$ é encontrado. Como os dois itens *A*[i] e *A*[j] estão fora de lugar no vetor final então troque-os de lugar. Continue este processo até que os apontadores *i* e *j* se cruzem em algum ponto do vetor. Ao final o vetor *A* está particionado em uma parte esquerda com chaves menores ou iguais a x e uma parte direita com chaves maiores ou iguais a x.

Este algoritmo também demandou certo trabalho, pois se trata de outro método de ordenação que utiliza a recursão como meio facilitador de ser executado. Além disso, o **TBC-AED** trata esse tópico, do ponto da apresentação, como os demais, o que torna mais fácil sua apresentação gráfica, dada a complexidade de visualização de algoritmos recursivos (Figura 22).

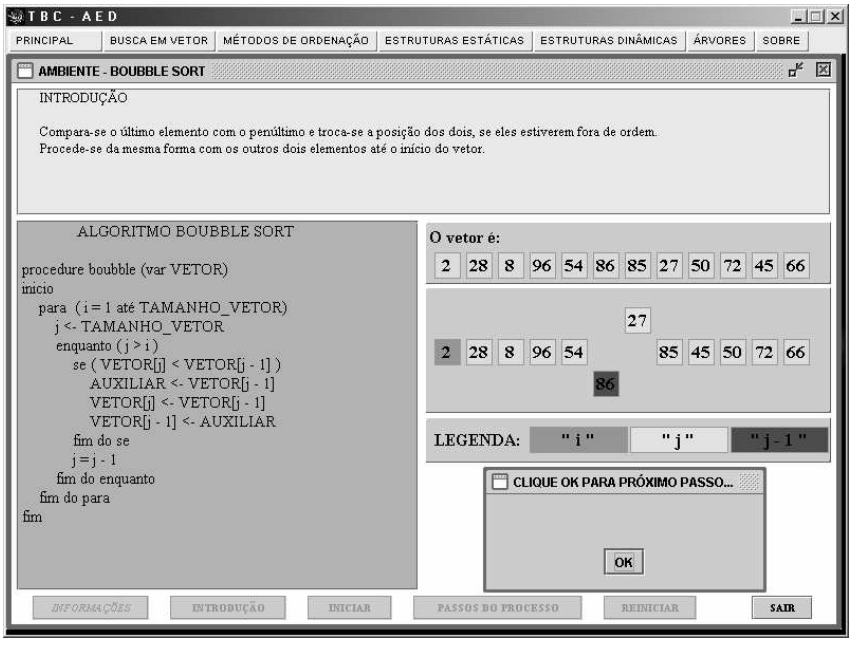

Figura 20. Tópico *Boubble Sort* do tema Métodos de Ordenação

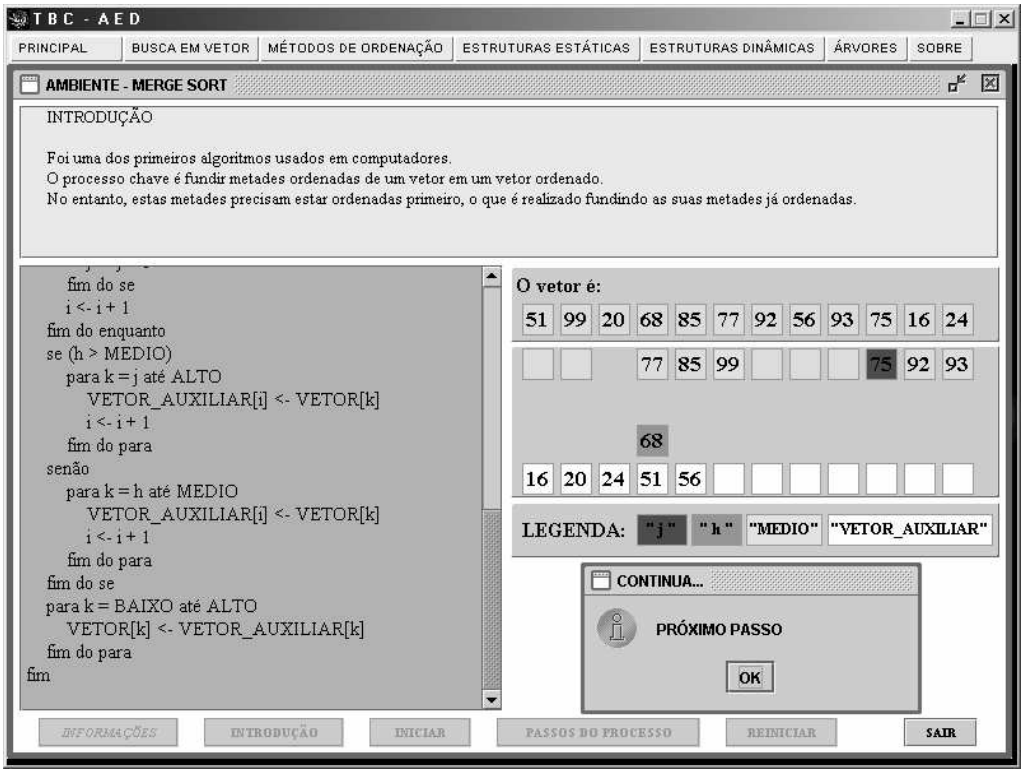

Figura 21. Tópico *Merge Sort* do tema Métodos de Ordenação

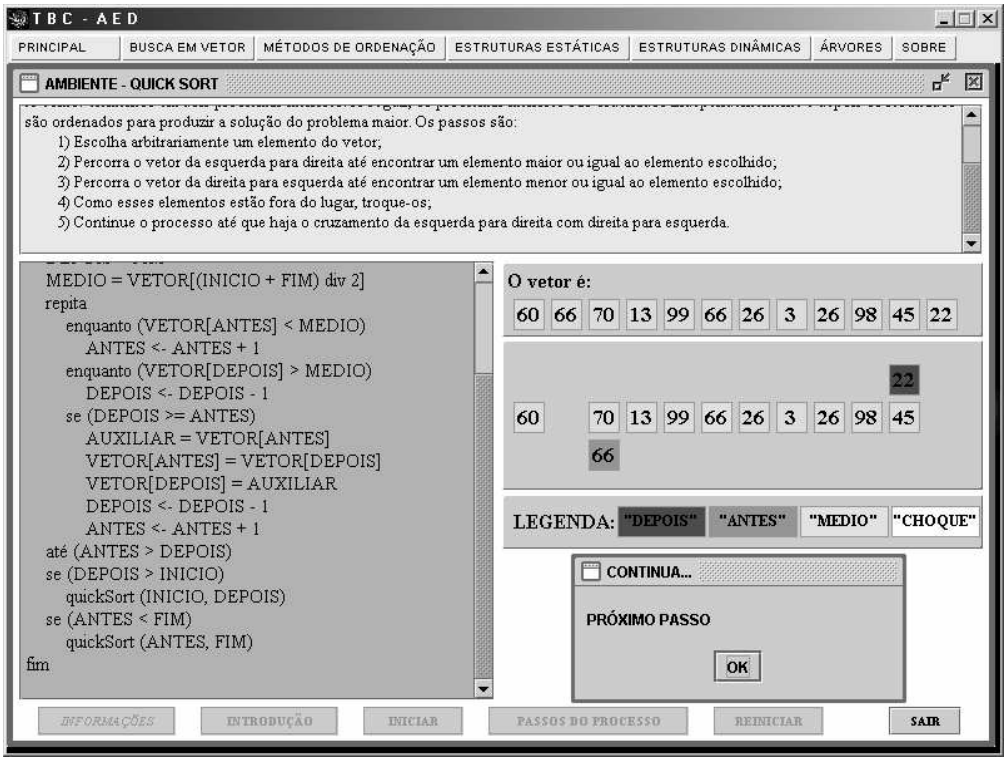

Figura 22. Tópico *Quick Sort* do tema Métodos de Ordenação

5.4.3 Estruturas de Dados de Alocação Estática e Dinâmica de Memória

Tipos abstratos de dados que utilizam alocação estática de memória são aqueles nos quais o comprimento é definido *a priori*, através do uso de um vetor. Isso representa uma limitação em tais tipos. No entanto, a contigüidade de elementos pode representar uma vantagem, que torna o acesso aos dados mais fácil e rápido. Os tipos abstratos de dados apresentados neste trabalho e que podem utilizar alocação estática de memória são:

- lista linear;
- fila;
- pilha.

Os mesmos tipos abstratos de dados podem utilizar alocação dinâmica de memória, que consiste em alocar espaço em memória à medida que for precisando. Isso representa uma vantagem em tais tipos. No entanto, não há a contigüidade de elementos em memória, o que pode representar uma desvantagem, devido a riscos advindos de má implementação (acesso e operações sobre posições errôneas de memória).

5.4.3.1 Listas

Conforme [ZIVIANI 1992], uma das formas mais simples de interligar os elementos de um conjunto é através de uma lista. Lista é uma estrutura onde as operações inserir, retirar e localizar são definidas. Uma **lista linear** é uma seqüência de um ou mais x_1, x_2, ..., x_n, onde x_i é de um determinado tipo e *n* representa o tamanho da lista linear. Sua principal propriedade estrutural envolve as posições relativas dos itens em uma dimensão.

Para criar um tipo abstrato de dados lista, é necessário definir um conjunto de operações sobre os objetos do tipo lista. O conjunto de operações a ser definido depende de cada aplicação, não existindo um conjunto de operações que seja adequado a todas as aplicações.

Quanto à implementação feita deste tópico para o **TBC-AED**, uma diferença relativa é notada. O usuário terá a oportunidade de interagir mais com o produto de *software*, uma vez que executará as operações sobre o TAD lista e observará suas

ocorrências passo-a-passo. A escolha da tarefa a ser executada é feita através de uma janela de diálogo exibida quando o usuário clicar no botão Passos do Processo (Figura 23).

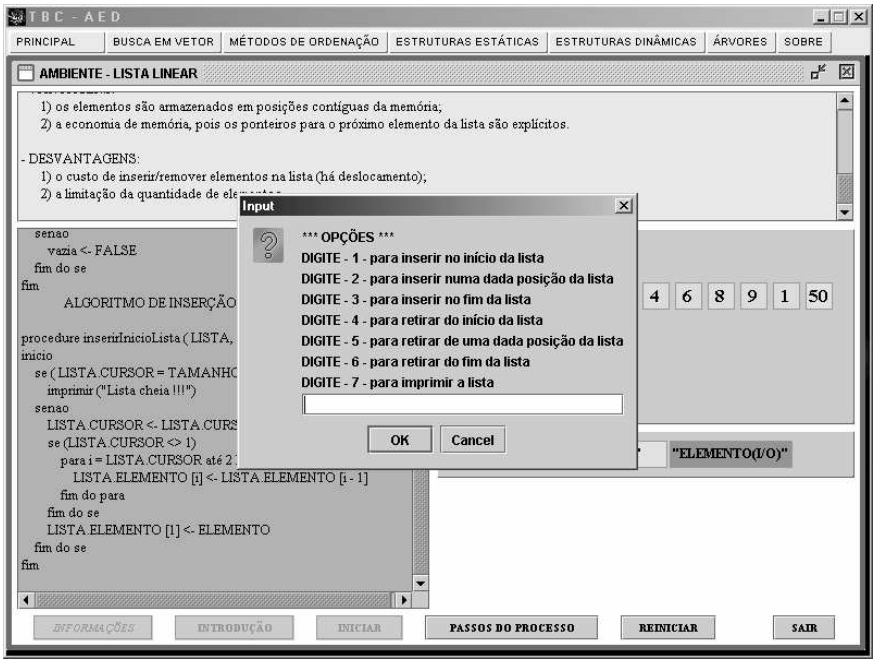

Figura 23. Tópico Lista Linear do tema Estruturas Estáticas

No entanto, segundo [ZIVIANI 1992], em uma implementação de listas através de apontadores, cada item da lista é encadeado com o seguinte através de uma variável do tipo apontador. Este tipo de implementação permite utilizar posições não contíguas de memória, sendo possível inserir e retirar elementos sem haver necessidade de deslocar os itens seguintes da lista. A lista possui um apontador indicando o início da lista.

A lista é constituída de nós, onde cada nó contém um item da lista (informação) e um apontador para o nó seguinte. A implementação através de apontadores permite inserir ou retirar itens do meio da lista a um custo constante, aspecto importante quando a lista tem que ser mantida em ordem. Em aplicações em que não existe previsão sobre o crescimento da lista, é conveniente usar listas encadeadas por apontadores, porque, neste caso, o tamanho máximo da lista não precisa ser definido *a priori*. A maior desvantagem deste tipo de implementação é a utilização de memória extra para armazenar os apontadores.

No **TBC-AED**, foram feitas duas implementações de listas com alocação de memória dinâmica: lista simplesmente encadeada e lista duplamente encadeada. A diferença entre elas é que na primeira existe apenas um ponteiro, que indica qual o próximo elemento da lista, e na segunda, existe a possibilidade de "caminhar" na lista em direção aos elementos anteriores e posteriores. Quanto à forma de apresentação da tela, não há eventuais detalhes em relação às estruturas de dados de alocação estática, salvo os ponteiros existentes e um ponteiro, que aponta para o primeiro elemento (Figura 24 e Figura 25).

O tópico lista circular não foi implementado. No **TBC-AED**, há apenas uma referência em forma de janela de mensagem, contendo a definição e algumas considerações. Segundo [SZWARCFITER e MARKENZON 1994], uma pequena modificação na estrutura física da lista pode ser de grande auxílio: obrigar o último nó da lista a apontar para o nó-cabeça, criando assim uma lista circular encadeada. Com isso, tem-se mais dois tipos derivados de listas: lista circular simplesmente encadeada e lista circular duplamente encadeada, que seguem a definição mencionada, somada à forma circular da estrutura de dados (Figura 26). Como trabalho, sugere-se que o docente peça aos alunos que implementem esta estrutura de dados, para fixarem os conceitos adquiridos nas aulas.

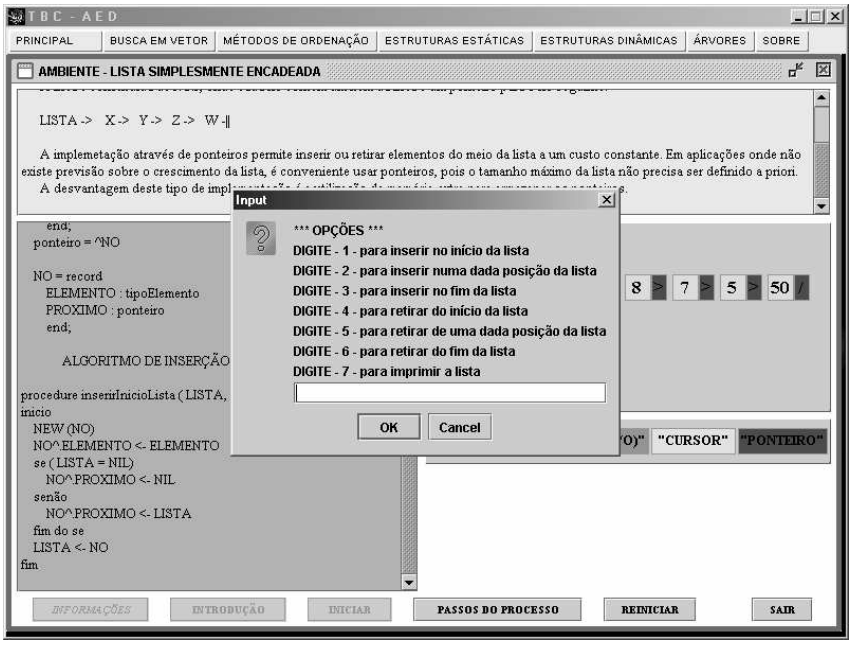

Figura 24. Tópico Lista Simplesmente Encadeada do tema Estruturas Dinâmicas

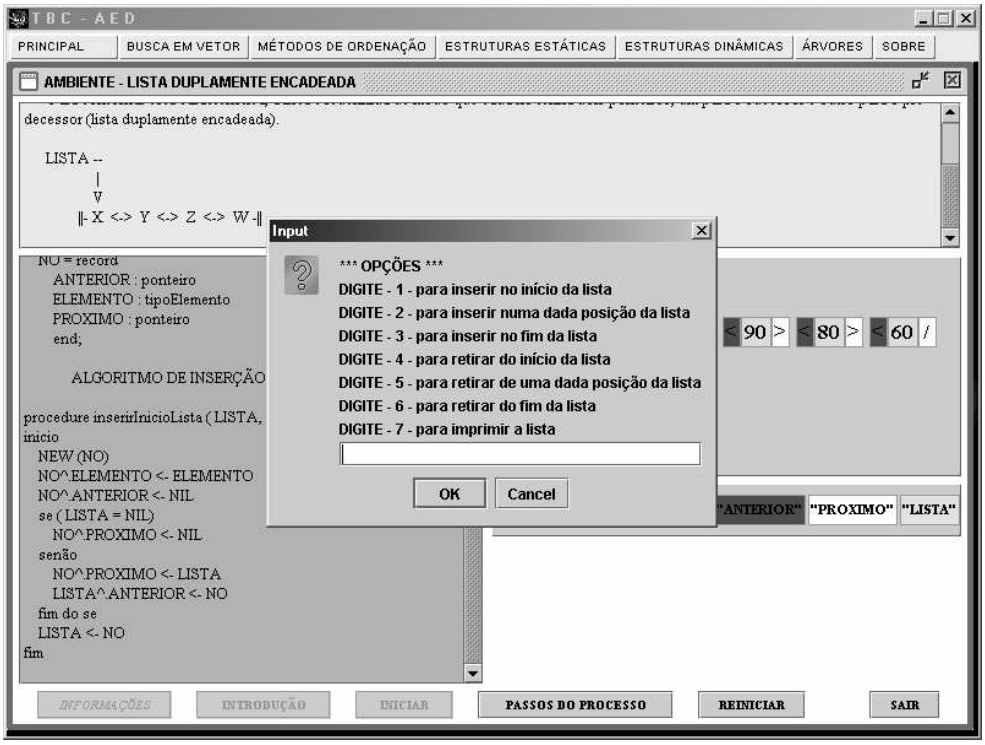

Figura 25. Tópico Lista Duplamente Encadeada do tema Estruturas Dinâmicas

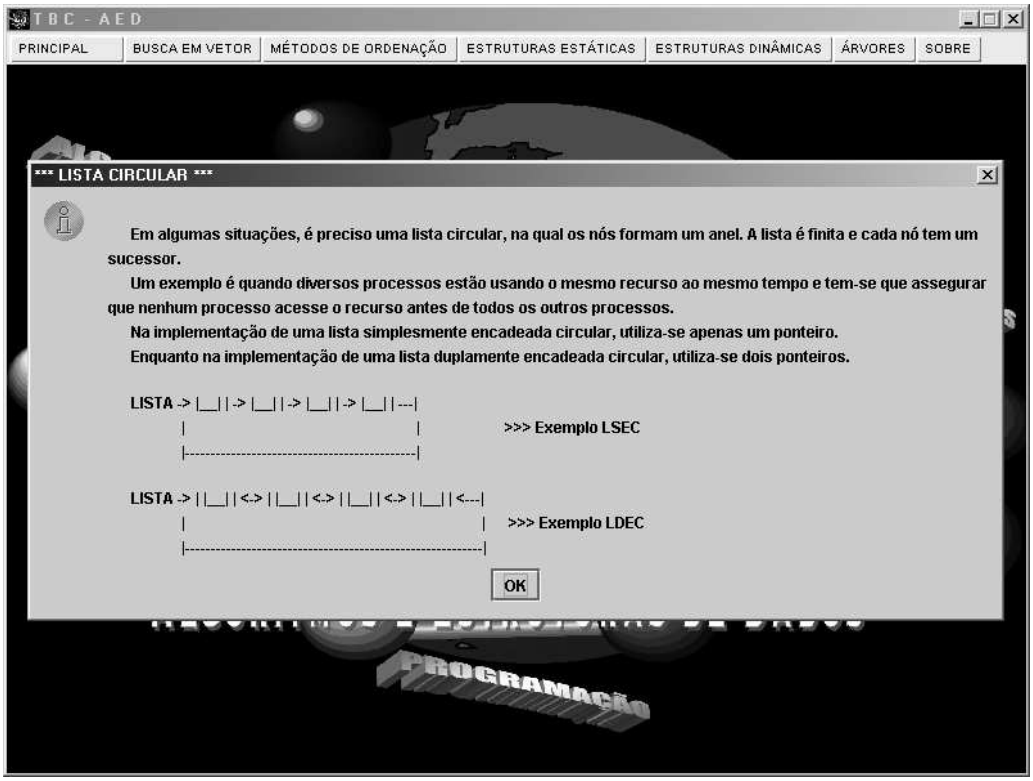

Figura 26. Tópico Lista Circular do tema Estruturas Dinâmicas, que não foi implementado

5.4.3.2 Filas

Uma fila é uma lista linear em que todas as inserções são realizadas em um extremo da lista e todas as retiradas e os acessos são realizados no outro extremo da lista. O modelo intuitivo de uma fila é o de uma fila de espera em que as pessoas no início da fila são servidas primeiro e as pessoas que chegam, entram no fim da fila. Por esta razão, as filas são chamadas de listas **fifo**, termo formado a partir de *first-in, first-out*. Existe uma ordem linear para filas que é a "ordem de chegada". Filas são utilizadas quando se deseja processar itens de acordo com a ordem "primeiro que chega, primeiro atendido". Sistemas operacionais utilizam filas para regular a ordem na qual tarefas devem receber processamento e recursos devem ser alocados a processos [ZIVIANI 1992].

A sua implementação pelo **TBC-AED** segue os mesmos passos da tópico *lista linear*, descrito anteriormente, com a participação ativa do usuário durante as operações realizadas sobre o tipo abstrato de dados (Figura 27).

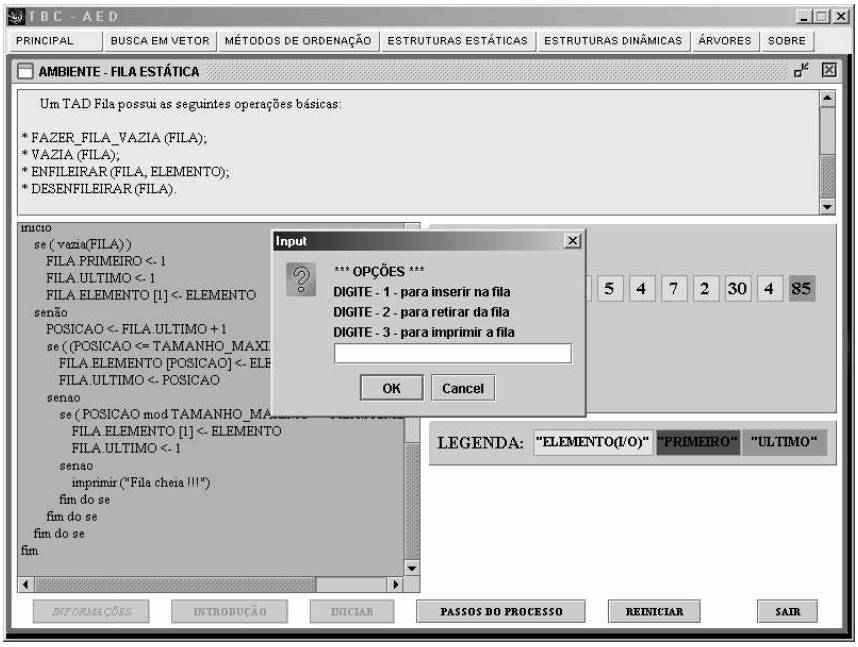

Figura 27. Tópico Fila do tema Estruturas Estáticas

A fila, implementada com uso de alocação dinâmica de memória, possui dois apontadores indicando o início e o final da fila. Quando a fila está vazia, os apontadores

"frente" e "trás" são "aterrados". Para enfileirar um novo item, basta criar um nó, ligá-lo após o último nó na fila, colocar nele o novo item e fazer o apontador "trás" apontar para ele. Para desenfileirar um item, basta fazer o apontador "frente" apontar para o segundo nó e desligar o nó a ser retirado do segundo nó. A fila é implementada através de nós, onde cada nó contém um item (informação) da fila e um apontador para outro nó.

No **TBC-AED**, a implementação deste tópico não difere da estrutura fila de alocação estática de memória, quanto à funcionalidade, mas apresenta ponteiros que a classificam como estruturas de alocação dinâmica de memória (Figura 28).

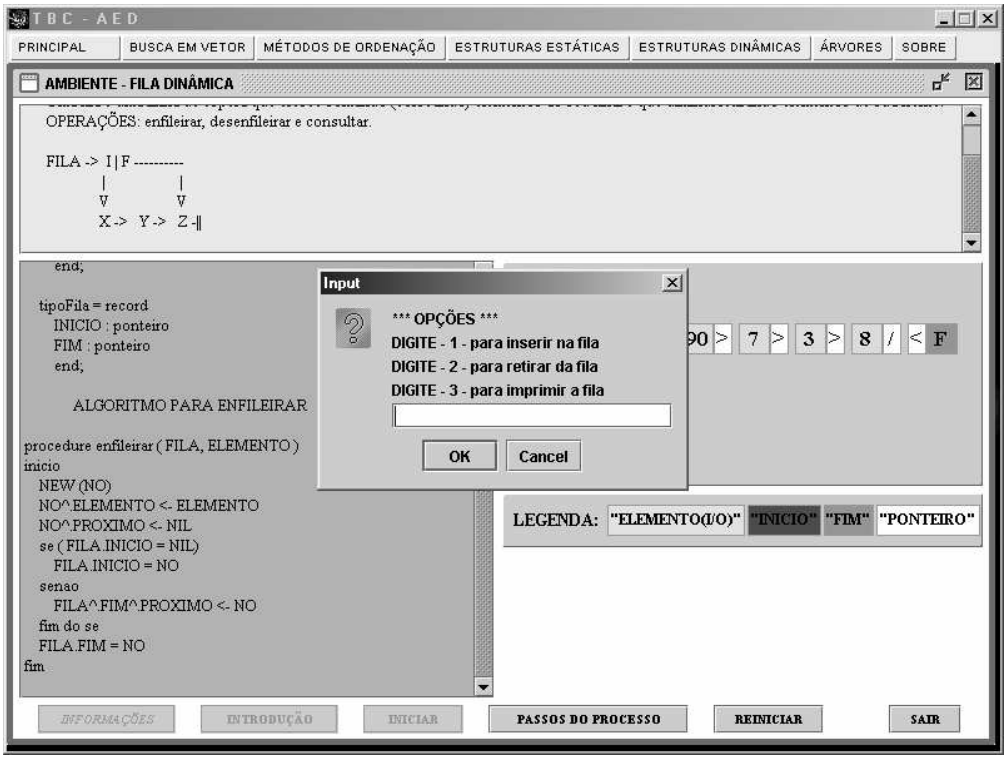

Figura 28. Tópico Fila do tema Estruturas Dinâmicas

5.4.3.3 Pilhas

Uma pilha é uma lista linear em que todas as inserções, retiradas e, geralmente, todos os acessos são feitos em apenas um extremo da lista.

Os itens em uma pilha estão colocados um sobre o outro, com o item inserido mais recentemente no topo e o item inserido menos recentemente no fundo. O modelo

intuitivo de uma pilha é o de um monte de pratos em uma prateleira, sendo conveniente retirar pratos ou adicionar novos pratos na parte superior. Esta imagem está freqüentemente associada com a teoria de autômatos, onde o topo de uma pilha é considerado como o receptáculo de uma cabeça de leitura/gravação que pode empilhar e desempilhar itens da pilha [HOPCROFT e ULLMAN 1969 *apud* ZIVIANI 1992].

As pilhas possuem a seguinte propriedade: o último item inserido é o primeiro item que pode ser retirado da lista. Por esta razão as pilhas são chamadas de listas **lifo**, termo formado a partir de *last-in, first-out*. Existe uma ordem linear para pilhas, que é a ordem do "mais recente para o menos recente". Esta propriedade torna a pilha uma ferramenta ideal para processamento de estruturas aninhadas de profundidade imprevisível. A qualquer instante uma pilha contém uma seqüência de obrigações adiadas, cuja ordem de remoção da pilha garante as estruturas mais internas serão processadas antes das estruturas mais externas. As pilhas ocorrem também em conexão com algoritmos recursivos e estruturas de natureza recursiva, tais como as árvores.

No caso do tópico implementado pelo **TBC-AED**, a metodologia de apresentação é a mesma das demais estruturas de alocação estática de memória, conforme apresentado na Figura 29.

Figura 29. Tópico Pilha do tema Estruturas Estáticas

A pilha, implementada com uso de alocação dinâmica de memória, possui um apontador indicando o topo da pilha. Para desempilhar o item x_n, basta o apontador do topo da pilha apontar para o nó seguinte. Para empilhar um novo item, basta fazer a operação contrária, criando um novo nó, colocando o novo item no novo nó e fazendo-o como o primeiro nó da pilha (o apontador do topo da pilha aponta para o novo nó). Cada nó de uma pilha contém um item (informação) da pilha e um apontador para outro nó.

Dessa forma, a implementação deste tópico também segue o padrão das estruturas de alocação estática de memória (Figura 30).

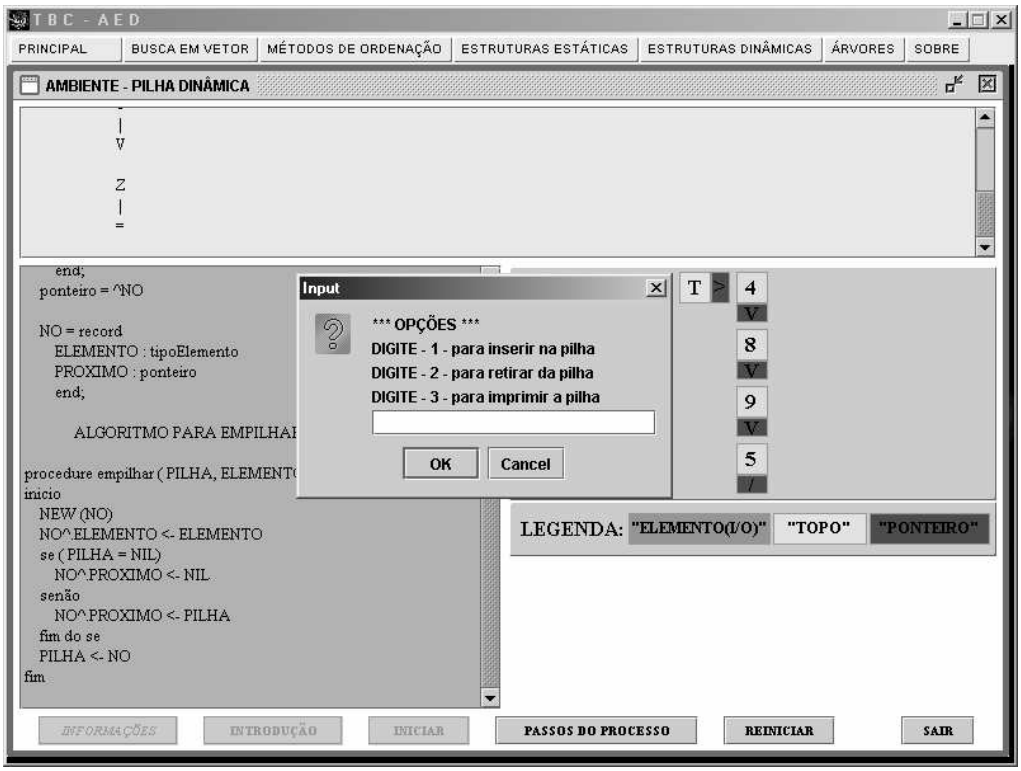

Figura 30. Tópico Pilha do tema Estruturas Dinâmicas

5.4.5 Árvores

As listas ligadas geralmente fornecem maior flexibilidade que as matrizes, mas são estruturas lineares e é difícil usá-las para organizar uma representação hierárquica. Embora as pilhas e as filas possuam alguma hierarquia, são limitadas a uma dimensão. Para superar essa limitação, criou-se um novo tipo abstrato de dados chamado árvore. A

definição de uma árvore não impõe qualquer condição sobre o número de filhos de um nó. Esse número pode variar de zero a qualquer inteiro.

Segundo [SZWARCFITER e MARKENZON 1994], em diversas aplicações, necessita-se de estruturas mais complexas do que puramente seqüenciais, as quais foram examinadas. Entre essas, destacam-se as árvores, por existirem inúmeros problemas práticos que podem ser modelados através delas. Além disso, as árvores, em geral, admitem um tratamento computacional simples e eficiente.

Da mesma forma que as listas lineares, árvores são estruturas de dados que caracterizam uma relação entre os dados que a compõem. Neste caso, a relação existente entre os dados (denominados nós) é uma relação de hierarquia ou de composição, onde um conjunto de dados é hierarquicamente subordinado a outro. [VELOSO *et al* 1986]

Segundo [VELOSO *et al* 1986], árvores binárias são estruturas do tipo árvore, onde o grau (quantidade de filhos) de cada nó é menor ou igual a dois. No caso de árvores binárias, distinguem-se as subárvores de um nó entre subárvore da esquerda e subárvore da direita. Assim, se o grau de um nó for igual a 1, deve ser especificado se a sua subárvore é a da esquerda ou a da direita. Uma árvore binária também pode ser vazia, isto é, não possuir nenhum nó. Entretanto, deve ser considerado que uma árvore binária não é um caso especial de árvore [SZWARCFITER e MARKENZON 1994].

A estrutura de árvore é utilizada em casos onde os dados ou objetos a serem representados possuem relações hierárquicas entre si. Considere, por exemplo, a estrutura de uma **universidade**, composta de **centros**. Cada **centro** composto por um certo número de **departamentos**. Cada **departamento** oferece um conjunto de **disciplinas** nas quais são matriculados os **alunos**. Outra aplicação é a representação de expressões aritméticas, que podem ser representadas sob a forma de árvores binárias, de tal forma que a hierarquia (prioridade) dos operadores fique clara (bem definida). Pode-se ainda utilizar árvores binárias para armazenar dados de maneira tal que, ao final da operação de armazenamento, os dados estejam ordenados, segundo um dado critério, independente da ordem de armazenamento. [VELOSO *et al* 1986]

A estrutura do tópico árvore binária implementada pelo **TBC-AED** não segue a risca o padrão dos demais tópicos abordados. Como já foi mencionado na seção 5.3, ocorrem algumas mudanças na apresentação, além das estratégias utilizadas para a

exibição de deste tipo abstrato de dados, as quais demandaram cálculos matemáticos envolvidos com a representação geométrica. Entretanto, com relação à funcionalidade, nada de especial está agregado ao tópico (Figura 31).

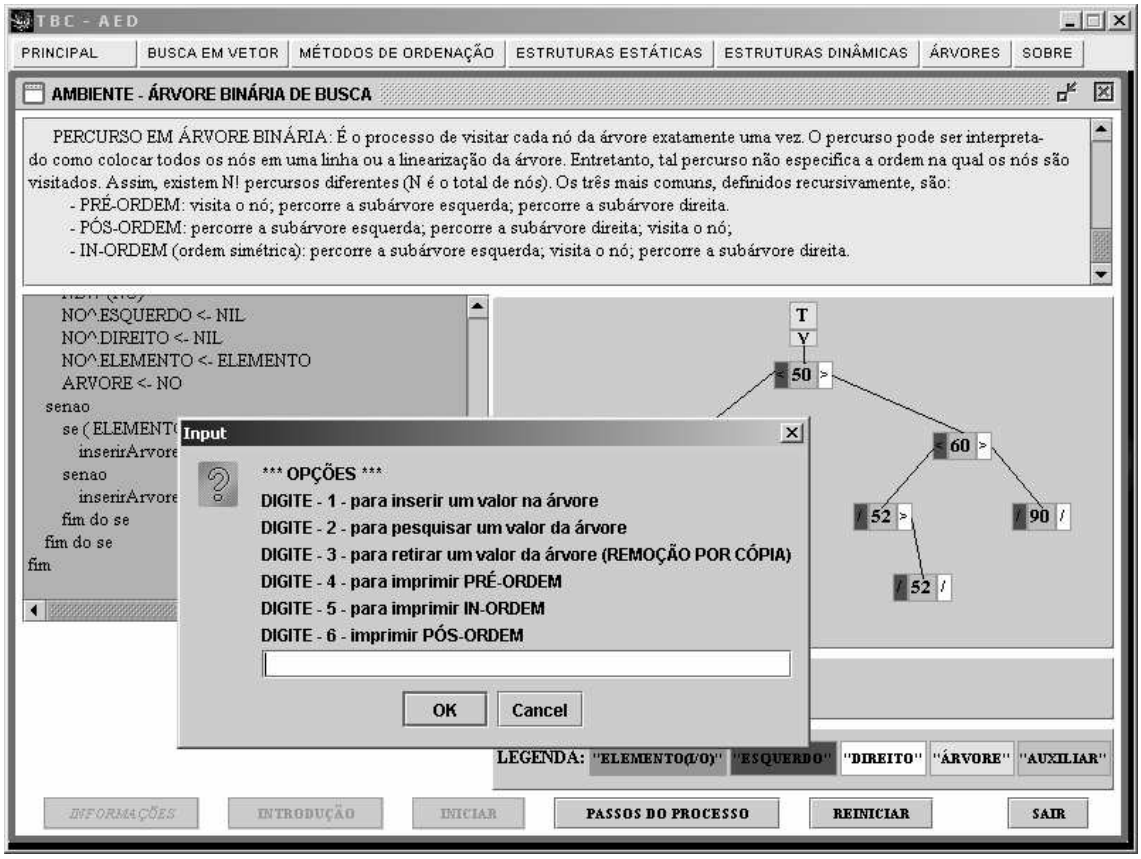

Figura 31. Tópico Árvore Binária de Busca do tema Árvores

5.5 Considerações Finais

Apesar de todas as vantagens alcançadas pelo **TBC-AED**, é importante ressaltar que alguns temas também envolvidos com algoritmos e estruturas de dados não foram abordados, como balanceamento de árvores binárias e outros métodos de ordenação. Isso é deixado aos docentes para sugerir aos alunos como pesquisa, para aprofundar conhecimentos e permitir maior discussão durante as aulas. Isso considerando que haverá uma economia de tempo a ser utilizada para resolução de aplicações reais e esclarecimento de dúvidas.

Capítulo 6 – Considerações Finais

Neste capítulo, são apresentados alguns aspectos resultantes da pesquisa. Na seção 1, é apresentada uma breve conclusão da relevância do trabalho. Na seção 2, são citadas algumas contribuições que a pesquisa gerou em seu desenvolvimento. Por fim, na seção 3, é feita uma síntese dos trabalhos a serem desenvolvidos com base nesta pesquisa.

6.1 Conclusões

Sob todo conjunto de informações obtidas até o presente momento, pode-se perceber que existe uma gama de recursos que possibilitam o desenvolvimento de ferramentas computacionais para o auxílio da educação superior em Computação, com destaque para a linguagem de programação *Java*, que conta com a portabilidade e forte suporte para técnicas adequadas de Engenharia de *Software* [DEITEL 2003]. Isso faz com que as novas tecnologias atinjam mais rapidamente seu público alvo.

Além disso, observou-se que o uso de produtos de *software* para o ensino de programação e algoritmos é uma idéia inteligente, se for amadurecida e desenvolvida de forma cuidadosa e estruturada. Isso fornece novas experiências para professores e alunos, que se proponham a trabalhar em cima desse tema. Além disso, eles terão sua qualidade de ensino elevada e poderão ser capazes de avançar mais rapidamente no campo do conhecimento, através da agilidade de processos didáticos e da participação ativa do docente da área.

Conclui-se também que uma boa elaboração de produtos tecnológicos que facilitem a transmissão de conhecimentos em Computação deve ser acompanhada de expressiva pesquisa no campo de novas metodologias de ensino. A principal finalidade disso é cada vez mais haver contribuições que incorram no aprimoramento do ensino superior e na formação de profissionais melhor qualificados para o mercado. Isso reflete principalmente sobre futuros professores, uma vez que aqueles que têm passado por esse tipo de experiência durante a graduação terão grande interesse em executar o mesmo processo quando estiverem lecionando.

Dessa forma, compreende-se que a educação precisa ser reavaliada, pois representa a base da atual sociedade mundial e o único instrumento que pode desenvolver o ser humano e torná-lo digno de constituir sua vida e seu ambiente. Isso deve ser acompanhado por uma atitude crítica da comunidade científica formada, que é a maior interessada em manter e prosseguir com avanços que contribuam para a melhoria da qualidade de vida.

6.2 Contribuições

Os problemas apresentados na compreensão de conceitos abstratos de programação, conforme [BUZIN 2001], encontra sua raiz ainda durante a formação básica, o que caracteriza uma importante linha de pesquisa na metodologia educacional. Com certeza, sem uma radical mudança cultural de atitude e comportamento, um indivíduo dificilmente poderá ser um profissional de Computação adequado, visto que é preciso desenvolver a capacidade de buscar respostas através de novas perguntas que levem à resposta da questão original. A formação do profissional em Computação deve incitar o desenvolvimento de raciocínio crítico, a solução de problemas, a aplicação de métodos de pesquisa e o desenvolvimento profissional contínuo.

Uma outra linha de pesquisa realizada é aquela direcionada para a tendência de utilizar ferramentas computacionais como ambientes de estudo, pois segundo [SOARES *et al apud* SOARES, CORDEIRO, STEFANI e TIRELO 2004], "a partir de observações dentro de disciplinas de graduação, percebe-se também um melhor resultado no aprendizado por meio de atividades práticas de desenvolvimento de simuladores e ferramentas visuais didáticas de representação de conceitos abstratos". Além disso, acompanhando esse desenvolvimento, a linha acopla contribuições dos pesquisadores para novas estratégias de transmissão de conhecimentos, promovendo a ampliação do acervo existente e a melhor organização de currículos para o mercado de trabalho, e acesso facilitado de tais recursos inovadores através de suporte via *Web*.

Dessa forma, a pesquisa deixa como contribuição um produto de *software*, o **TBC-AED**, que engloba temas relacionados a algoritmos e estruturas de dados. Além disso, faz uma revisão da literatura envolvendo análise de metodologias de ensino em Computação e tópicos de Informática na Educação e Linguagem de Programação Java.

Isso tem como objetivo estimular mudanças na educação de cursos que formam recursos humanos para a área tecnológica e incentivar docentes a melhorarem sua forma de pesquisa e ensino envolvendo Computação e Informática.

6.3 Trabalhos Futuros

Finalmente, como trabalhos futuros, pretende-se estudar uma maneira de disponibilizar o **TBC-AED** via ***Web***, para que possa ter maior acessibilidade, além de possibilitar a troca de idéias, através da interatividade com usuários diversos. Esse é um dos tópicos da continuação da pesquisa, uma vez que segundo [THOMAS, PATEL, HUDSON e BALL 1997], hoje em dia na *Internet* muito se fala sobre Java. Grande parte dessa agitação foi gerada pelos *applets* – pequenos programas que podem ser embutidos em páginas *Web*. Mas, Java não é apenas um programa que torna uma página *Web* interessante. Antes de mais nada, ela é uma linguagem de programação poderosa, geral e independente de plataforma, o que torna seu estudo interessante e capaz de levar à resolução de problemas e desafios da área tecnológica da Computação

A partir disso, também é interessante atacar outro assunto de grande importância, a **teoria dos grafos**, a fim de ampliar o conhecimento e contribuir para a melhoria do ensino na área. Um dos grandes ramos iniciais de estudo do graduando da área se encontra nesse assunto que, segundo [SANTOS, MELLO e MURADI 1995], tem origem relativamente recente (século XVIII) e apresenta extensiva utilização em matemática aplicada, pois demonstrou ser uma poderosa ferramenta para a modelagem de diversas situações reais em física, química, biologia, engenharia elétrica e pesquisa operacional. Entretanto, seu estudo e os algoritmos relacionados muitas vezes são de difícil imaginação, o que gera uma antítese. Portanto, torna-se um grande desafio para professores e coordenadores de cursos da área promover mudanças que possam impulsionar as capacidades intelectuais para que estudantes possam contribuir mais e mais para a evolução global.

Dentro dessa esfera, percebe-se que cada ramo profissional deve ser embasado de acordo com sua própria definição, através da idéia de meta-educação. E ao encontro

disso que um novo trabalho a ser realizado em 2005 propõe um estudo de *applets*[5] para disponibilizar via *Web* o **TBC-AED**. Além disso, propõe também um estudo da **teoria de grafos**, para o desenvolvimento de uma nova ferramenta computacional didática (*software* educacional), dentro da mesma linha de pesquisa seguida. Essa pesquisa visa tornar o ensino deste conteúdo – básico para o estudo em Computação avançada, que promove a resolução de problemas de otimização – mais prático e abrangente. Além disso, também propõe o acompanhamento da primeira turma a utilizar o **TBC-AED** para ensino de algoritmos e estruturas de dados. Dentro da possibilidade de tempo, de forma a não fugir do escopo do projeto, pretende-se deixar uma contribuição a respeito dos problemas que a educação básica gera sobre a formação de futuros profissionais nas áreas de Computação e Informática.

[5] *Applets* são programas Java que podem ser embutidos em documentos HTML. Quando um navegador carrega uma página da Web que contém um applet, o applet é baixado para o navegador e começa a ser executado [DEITEL, 2003].

Referências Bibliográficas

Almeida, M. E. (2000) "Informática e Formação de Professores", Série de Estudos – Educação à Distância (ProInfo). Ministério da Educação. Secretária de Educação à Distância. v. 2

Almeida, M. S. e Nogueira, C. R. D. (2001) "A *Internet* no Trabalho Pedagógico dos Professores", Informática na Educação – Artigos Científicos. Universidade Regional Integrada do Alto Uruguai e das Missões – URI – RS.

Alves, A. A. e Costa, M. R. (2004) "*algol*PUC: Uma Ferramenta de Apoio à Programação para Iniciantes", III Workshop de Educação em Computação e Informática do estado de Minas Gerais (WEIMIG' 2004). Belo Horizonte, MG, Brasil.

Azeredo, P. A. (2000) "Uma proposta de Plano Pedagógico para a Matéria de Programação", Anais do II Curso: Qualidade de Cursos de Graduação da Área de Computação e Informática (WEI 2000). Editora Universitária Champagnat.

Buzin, P. F. W. K. (2001) "A epistemologia da Ciência da Computação: Desafio do Ensino dessa Ciência", Revista de Educação, Ciência e Cultura, v. 6, nº 2. Centro Universitário La Salle. Canoas, RS, Brasil.

Campos, E. A. V. e Ascencio, A. F. G. (2003) "Fundamentos da Programação de Computadores". Editora Prentice Hall.

Deitel, H. M. e Deitel, P. J. (2003) "Java Como Programar". Bookman Companhia ed Informática.

Doederlein, O. P. (2004) "Tiger: A Evolução de Java", Java Magazine. Edição 12. Ano II. Neofício Editora.

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 (RBIE'1997) nº 01, http://www.inf.ufsc.br/sbc-ie/revista/nr1/garcia.htm, 28 abr 2004.

Giraffa, L., Marczak, S. e Almeida, G. (2003) "O Ensino de Algoritmos e Programação mediado por uma Ambiente *Web*", Congresso Nacional da Sociedade Brasileira de Computação (SBC'2003). Campinas, SP, Brasil.

Grillo, M. C. A. (1986) "Programação Estruturada com Fortran e Watfiv". Editora LTC. 2ª edição.

Júnior, J. C. R. P. e Rapkiewicz, C. E. (2004) "O Processo de Ensino e Aprendizagem de Algoritmos e Programação: Uma Visão Crítica da Literatura", III Workshop de Educação em Computação e Informática do estado de Minas Gerais (WEIMIG' 2004). Belo Horizonte, MG, Brasil.

Koslowski, S. R. e Nunes, M. A. S. N. (2001) "O Computador como Ferramenta Pedagógica nas Séries Iniciais", Informática na Educação – Artigos Científicos. Universidade Regional Integrada do Alto Uruguai e das Missões – URI – RS.

Menezes, C. S. e Valli, M. C. P. (1997) "O Uso da Planilha Eletrônica como Instrumento de Apoio à Construção do Conhecimento". Anais do VIII Simpósio Brasileiro de Informática na Educação. Instituto Tecnológico de Aeronáutica (ITA). v. 1

Pimentel, E. P., França, V. F. e Omar, N. (2003) "A Caminho de um Ambiente de Avaliação e Acompanhamento Contínuo de Aprendizagem em Programação de Computadores", II Workshop de Educação em Computação e Informática do Estado de Minas Gerais (WEIMIG'2003). Poços de Caldas, MG, Brasil.

Rodrigues, M. C. "Como Ensinar Programação?". Informática – Boletim Informativo Ano I nº 01, ULBRA. Canoas, RS, Brasil.

Santos, J. P. O., Mello, M. P., Muradi, I. T. C. (1995) "Introdução à Análise Combinatória". Editora da UNICAMP.

Setubal, J. C. (2000) "Uma proposta de Plano Pedagógico para a Matéria de Computação e Algoritmos", Anais do II Curso: Qualidade de Cursos de Graduação da Área de Computação e Informática (WEI 2000). Editora Universitária Champagnat.

Silva, W. (2003) "Apostila do Curso de Extensão – Linguagem de Programação Java e Orientação a Objetos", DCC – Departamento de Ciência da Computação – UFLA.

Soares, T. C. A. P., Cordeiro E. S., Stefani Í. G. A., Tirelo, F. (2004) "Uma Proposta Metodológica para o Aprendizado de Algoritmos em Grafos Via Animação Não-Intrusiva de Algoritmos", III Workshop de Educação em Computação e Informática do Estado de Minas Gerais (WEIMIG' 2004). Belo Horizonte, MG, Brasil.

Souza, P. C. e Wazlawick, R. S. (1997) "Ferramenta de Autoria para a Criação de Ambientes Construtivistas em Realidade Virtual". Anais do VIII Simpósio Brasileiro de Informática na Educação. Instituto Tecnológico de Aeronáutica (ITA). v. 1

Szwarcfiter, J. L., Markenzon, L. (1994) "Estruturas de Dados e seus Algoritmos". Editora LTC.

Thomas, M. D., Patel, P. R., Hudson, A. D., Ball, D. A. Jr. (1997) "Programando em Java para *Internet*". Editora Markron Books.

Tobar, C. M., Rosa, J. L. G., Coelho, J. M. A. e Pannain, R. (2001) "Uma Arquitetura de Ambiente Colaborativo para o Aprendizado de Programação", XII Simpósio Brasileiro de Informática na Educação (SBIE'2001). Vitória, ES, Brasil.

Veloso, P., Santos, C. dos, Azeredo P., Furtudo, A. (1986) "Estruturas de Dados". Editora Campus. 4ª edição.

Wirth, N. (1989) "Algoritmos e Estruturas de Dados". LTC Informática-Programação.

Zambalde, A. L. e Alves, R. M. (2002) "Introdução à Informática Educativa", Textos Acadêmicos para o Curso de Pós-Graduação '*Lato Sensu*' (Especialização) à Distância – Informática em Educação. UFLA/FAEPE.

Ziviani, N. (1992) "Projeto de Algoritmos com Implementações em Pascal e C". Editora Pioneira Thompson.