Um Ambiente de Desenvolvimento Personalizável para o Ensino de Programação

Willian L. S. Frantz[1], Herleson P. Pontes[1,2]

[1]Centro de Ciências Tecnológicas – Universidade de Fortaleza (UNIFOR)
Fortaleza – CE – Brasil

[2]Faculdades Nordeste (DeVry Brasil)
Fortaleza – CE – Brasil

willianluigii@hotmail.com, herleson@unifor.br

Abstract. *This paper presents a development environment for teaching computer programming in the initial stages of Computation and Engineering courses. This tool allows the implementation of algorithms totally written in Portuguese, lowering the common barriers presented in the programming learning process. As a differential, our environment allows the teacher defines the commands that will be prompted by students, smoothing the transition between this language and the programming languages used in production environments. After some tests for validate the impact of this tool in the teaching process, the results shows that the use of a localized and customizable environment catalyzes the learning process done by the teacher.*

Resumo. *Este artigo apresenta um ambiente de desenvolvimento voltado para o ensino de programação de computadores nas disciplinas iniciais dos currículos de Computação e Engenharias. Essa ferramenta permite a implementação de algoritmos na forma de códigos descritos totalmente na língua portuguesa, reduzindo as barreiras comuns presentes no processo de aprendizado em programação. Como diferencial, nosso ambiente possibilita ao professor definir todo o dicionário de comandos que serão disponibilizados aos alunos, suavizando a transição entre essa linguagem e as linguagens de programação utilizadas em ambientes de produção. Após a realização de testes com o objetivo de validar o impacto dessa ferramenta no ensino de programação, os resultados mostram que o uso de um ambiente no idioma nativo e personalizável catalisa o aprendizado do aluno, além de simplificar o processo de aprendizagem realizado pelo professor.*

1. Introdução

As disciplinas de programação possuem um papel crítico na formação dos profissionais na área de Computação, visto que seus princípios estimulam o uso do raciocínio lógico na descoberta, organização e implementação de soluções para os mais diversos problemas. Contudo, o conhecimento nessa área não limita-se ao domínio dos recursos e técnicas voltados para o desenvolvimento de programas. A resolução de uma vasta quantidade de problemas através da criação de programas possibilita, principalmente, o amadurecimento das habilidades de concepção e organização do pensamento, produzindo ideias factíveis como solução para as questões analisadas.

Apesar da relevância da programação na formação do estudante de Computação, a excelência no processo de aprendizagem nas disciplinas dessa área é um objetivo dificilmente tangível, tanto por alunos quanto por professores [Shabanah and Chen 2009].

Na visão do aluno, as disciplinas de programação apresentam várias barreiras para o seu aprendizado, como o uso de modelos para a descrição de conceitos complexos, a aplicação de conceitos provenientes da Matemática, a construção e manipulação de estruturas dinâmicas para os dados do programa, a abordagem de problemas desconhecidas, a falta de organização das ideias e dificuldades na língua inglesa. Esses empecilhos resultam na perda da motivação do estudante em relação ao seu curso e o bloqueio do desenvolvimento de suas habilidades em programação.

Na perspectiva do professor, o ensino de programação demanda um grande esforço em atividades como longas explanações dos conceitos, a montagem de diversas demonstrações e confecção de figuras e animações, na tentativa de maximizar a compreensão dos fundamentos apresentados. Essas atribuições o confere, consequentemente, a responsabilidade pelo fortalecimento do processo de descoberta de novos conhecimentos pelos seus estudantes [Henriques 2010], [Perrenoud 2010].

Motivados por esse cenário, apresentamos um ambiente para ensino de programação de computadores no qual a implementação de algoritmos é realizada através da construção de códigos escritos em português. Nosso ambiente suporta a definição, pelo professor, de todas as instruções que estarão à disposição dos alunos, reduzindo as barreiras presentes no processo de aprendizado em programação e aproximando a linguagem nativa da linguagem de programação utilizada nas demais disciplinas do curso.

2. Trabalhos Relacionados

A aprendizagem de programação é, sem dúvida, um dos mais importantes fundamentos presentes na formação de futuros profissionais nas carreiras de Computação e Engenharias, constituindo-se como um dos grandes desafios na estruturação do ensino de desenvolvimento de programas para uma nova formação universitária [Santiago and Dazzi 2004].

Nesse sentido, diversas pesquisas abordam metodologias e ferramentas voltadas para auxiliar o processo de aprendizagem desse fundamento. Alguns estudos, como [Santos and Costa 2005], propõem soluções que possibilitam o aprendizado de programação através de ambientes de desenvolvimento voltados para a fácil montagem e a visualização de programas. Esses sistemas auxiliam, adicionalmente, o amadurecimento dos conhecimentos no aluno nas Ciências Exatas, um dos pré-requisitos necessários para a construção do raciocínio lógico-matemático.

O correto processo de materialização das ideias em programas pelo aluno é outra preocupação inerente ao ensino dessa área. Trabalhos como os apresentados por [Rocha 1991] e [Santos and Costa 2006] objetivam o uso de aplicativos que possibilitam interações simples e intuitiva entre aluno e computador, visando facilitar o entendimento dos principais conceitos envolvidos no aprendizado de programação.

Por fim, estudos como o proposto por [Giraffa, Marczak and Prikladnicki 2005], discutem possíveis recursos que podem estar presentes em softwares educacionais no intuito de aumentar a eficiência do processo de aprendizado, como hipermídias, inteligência artificial, entre outros.

3. Aspectos Pedagógicos no Ensino da Programação

Pela sua importância, as disciplinas básicas que envolvem programação devem transmitir ao aluno todos os fundamentos da área computacional, permitindo ao aluno expandir o seu raciocínio, de forma a encaixar a lógica da computação nas mais diversas situações presentes no dia-a-dia de pessoas e empresas. Ao final dessas disciplinas, espera-se que esse estudante mantenha essas habilidades durante o resto de seu curso e no exercício de sua profissão [Falkembach *et al* 2003].

Nas aulas dessas disciplinas, no entanto, vários professores utilizam exemplos hipotéticos e dinâmicas associativas para demonstrar o funcionamento do computador e o descrever o processo de desenvolvimento de programas utilizando linguagens computacionais. Sem o ambiente correto e a supervisão do professor, o aprendizado do conhecimento transmitido nessa fase inicial acaba exigindo um esforço maior por parte do aluno, devido à ausência das competências inerentes a um profissional de Computação, como o raciocínio lógico e a organização das ideias [Pontes 2013].

Nesse sentido, é importante que o aluno disponha de ferramentas que possam dar suporte para as práticas de programação, onde ele possa desenvolver seu raciocínio sem a interferência de fatores como o conhecimento em inglês e a manipulação de ambientes complexos para desenvolvimento de programas. Uma boa ferramenta de suporte ao ensino de programação auxilia o aluno na representação da sua ideia em forma de códigos, além de possibilitar a descoberta e correção de possíveis erros. Assim, a resolução de um problema seria apresentada priorizando a lógica usada, e não o código, potencializando assim a construção do conhecimento pelo aluno.

Assim, no ensino de programação de computadores, o professor é responsável por oferecer uma série de recursos que abstraiam aspectos irrelevantes e foquem na transmissão do conhecimento ao aluno, esse responsável maior pelo próprio aprendizado [Jesus and Brito 2010].

Na representação dos passos que definem uma solução, cada aluno possui uma forma pessoal para representar a sua lógica, ainda que ele esteja utilizando uma linguagem de programação previamente conhecida. Graças a unicidade do raciocínio humano, é possível que dois ou mais alunos apresentem soluções diferentes para um mesmo problema.

Nesse sentido, é essencial que o aprendizado de algoritmos para resolver problemas enfatize o raciocínio do aluno. Nesse momento, contudo, o aluno ainda não possui familiaridade suficiente com o computador para compreender o que realmente está acontecendo. Essa dependência nos aspectos computacionais, ainda que apresentados pelo professor, acaba por dificultar o processo de aprendizagem, uma vez que o aluno é obrigado a implementar um algoritmo de acordo com a sintaxe e a semântica de uma determinada linguagem computacional, desviando sua atenção ao real objetivo: o amadurecimento do raciocínio lógico.

Portanto, devido aos pré-requisitos técnicos à complexidade do processo de construção e análise de algoritmos utilizando linguagens de programação, o ensino das disciplinas iniciais por diversos professores peca pelo desvio, ainda que involuntário, do foco no aprendizado da lógica e do algoritmo, sendo um aspecto negativo presente em diversas instituições de ensino em Computação.

4. Ambiente de Desenvolvimento Personalizável

No intuito de reduzir a preocupação dos alunos nos quesitos computacionais e aumentar o foco no uso da lógica para a solução de problemas, criamos um ambiente de desenvolvimento personalizável voltado para o ensino de programação. Nossa solução possui um conjunto de funcionalidades as quais possibilitam alunos e professores definirem como seus códigos serão construídos para resolver os problemas propostos, priorizando assim a lógica envolvida e ignorando o máximo de detalhes técnicos.

Baseando-se nas concepções presentes em uma *Integrated Development Environment (IDE)*, esse ambiente possibilita a implementação de algoritmos na forma de programas escritos em idioma nativo, concentrando assim a atenção do usuário na construção do algoritmo. A Figura 1 apresenta a janela principal da nossa ferramenta.

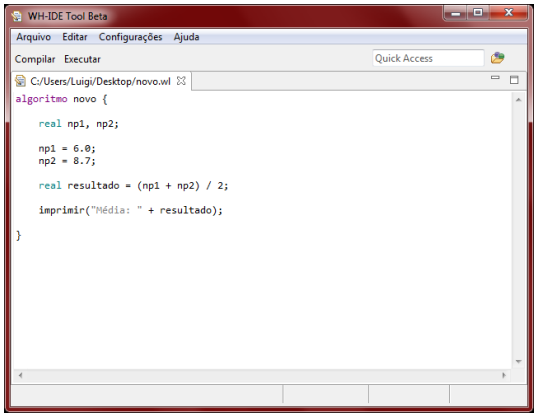
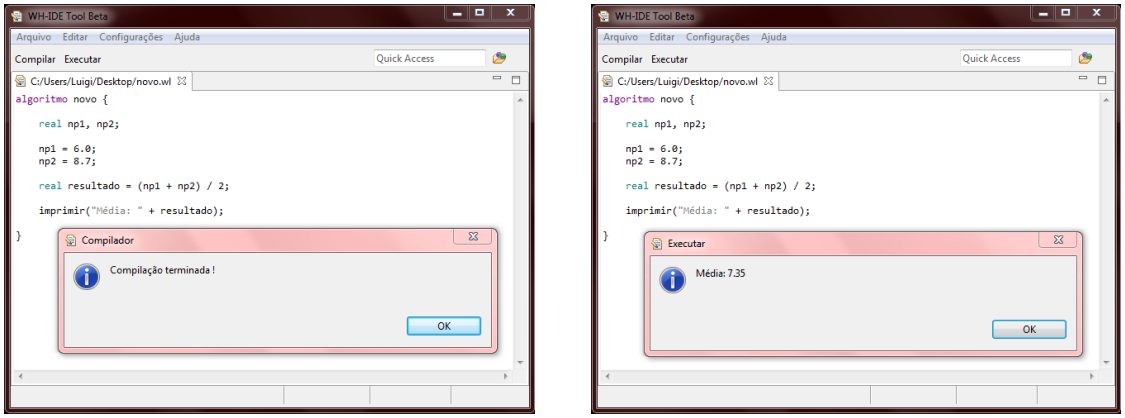


Figura 1 – Janela de edição do ambiente de desenvolvimento.

Para garantir o encapsulamento dos requisitos computacionais durante seu uso, nosso ambiente de desenvolvimento foi construído sobre a plataforma *Rich Client Plataform (RCP)* da *IDE Eclipse* [Eclipse Foundation 2014]. O *RCP* é uma plataforma voltada para o desenvolvimento de programas baseado na arquitetura do *Eclipse*, a qual oferece um conjunto de *plug-ins* para a construção de qualquer tipo de programa cliente. Entre os benefícios do *RCP*, destacam-se o reaproveitamento dos componentes, visual e funcionalidades já existentes na *IDE*, além de simplificar a portabilidade da solução entre os principais sistemas operacionais existentes no mercado.

Ainda visando a facilidade no seu uso, a instalação da nossa ferramenta resume-se a cópia dos arquivos necessários para uma pasta. A partir dessa pasta, o usuário aciona o arquivo executável e o programa é então carregado.

Além de oferecer um ambiente totalmente localizado para o usuário, nossa ferramenta apresenta alguns recursos importantes. Todos os comandos inseridos na janela de edição do programa são identificados através de cores, permitindo a identificação das palavras reservadas da linguagem utilizada e auxiliando a assimilação da estrutura lógica montada pelo usuário. Ademais, as saídas do programa são exibidas em uma janela no centro da tela, permitindo ao usuário focar nos resultados gerados pelo seu código, como mostra a Figura 2b.



(a) (b)

Figura 2 – Em (a), o processo de compilação. Em (b), saída de um algoritmo.

Outro importante recurso desse ambiente voltado para o aprendizado de programação é a separação das etapas de compilação e execução de um programa. Essa divisão permite ao aluno compreender o processo de tradução do seu código em linguagem de máquina, para então iniciar a sua execução. A Figura 2a apresenta a mensagem informando ao usuário que seu código foi compilado e está pronto para execução.

Embora as funcionalidades apresentadas auxiliem o processo de ensino-aprendizagem em programação, a linguagem é o ponto mais importante do nosso ambiente. Durante a definição da linguagem padrão utilizada pela ferramenta, considerou-se como critérios a sua simplicidade, a sua proximidade com a linguagem natural e o grau de liberdade na manipulação de suas palavras. Essa linguagem é definida através de um arquivo de inicialização editável, nomeado *lang.ini*, no qual todo o dicionário de palavras reservadas é armazenado, conforme a Figura 3. O suporte a edição possibilita ao professor alterar os comandos utilizados pela *IDE*, oferecendo um alto poder de adaptação do ambiente para as necessidades do docente. Ainda sobre o dicionário, o usuário não precisa definir todos os comandos da sua linguagem; apenas aqueles que serão diferentes dos existentes na linguagem padrão, a qual é aplicada para esses comandos não definidos. Assim, a preocupação do usuário será apenas com o formato das instruções que são importantes no seu ponto de vista.

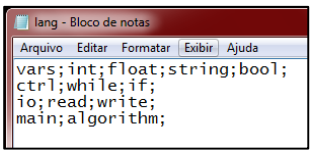


Figura 3 – Arquivo de configuração dos comandos da linguagem.

Independente da personalização realizada pelo usuário, a linguagem utilizada pela nossa *IDE* suporta a criação de variáveis com um dos 4 (quatro) tipos de dados apresentados na Tabela 1. Esses tipos foram concebidos a partir do agrupamento de tipos semelhantes existentes nas principais linguagens de programação. Por exemplo, o tipo de dado *literal* permite representar valores textuais, abstraindo o conceito de caractere e palavra. O resultado é um número reduzido de tipos de dados, o que reduz a complexidade do algoritmo criado pelo usuário.

Tabela 1 – Tipos de dados da linguagem suportada pela *IDE*.

inteiro	Armazena valores do tipo inteiro.
real	Representa números com casas decimais.
literal	Recebe um caractere ou uma cadeia de caracteres.
lógico	Descreve valores lógico (*Verdadeiro* ou *Falso*).

Para garantir a importância da identificação de um algoritmo e a delimitação do bloco de comandos que serão processados, todo programa desenvolvido na nossa *IDE* é delimitado por uma estrutura chamada *algoritmo*, a qual demarca o início e o final do código a ser compilado e executado. Essa estrutura também recebe o nome do algoritmo, incentivando o usuário a sempre nomear seus programas.

Seguindo a simplicidade da solução, foram definidas apenas 2(duas) estruturas de controle, como apresenta a Tabela 2. A estrutura de decisão é representada pela palavra reservada *se*. Já a estrutura de repetição é utilizada através da chamada ao comando *enquanto*. Ambas as estruturas utilizam os símbolos "{" e "}" para delimitar, respectivamente, o início e o final de cada estrutura, ainda que ela possua uma única linha. Esse recurso estimula o aluno a sempre trabalhar com blocos de comandos delimitados, suavizando o processo posterior de aprendizado de uma linguagem de programação.

Tabela 2 – Estruturas de controle suportadas pela *IDE*.

se	Representa as estruturas de decisão.
enquanto	Representa as estruturas de repetição.

A interação com o usuário final durante a execução de um código é realizada através de 2 (duas) instruções de *E/S*, conforme a Tabela 3. A primeira instrução *(ler)* é responsável por recebe dados inseridos pelo usuário através do teclado, enquanto a segunda *(imprimir)* exibe a saída na tela. Importante destacar que ambos os comandos abstraem qualquer necessidade de tratamento sobre os dados manipulados, evitando o uso de comandos voltados para a validação e conversão de dados.

Tabela 3 – Instruções de entrada e saída da *IDE*.

ler	Recebe dados do usuário.
imprimir	Envia dados para a saída.

Apesar de fornecer um ambiente básico para o desenvolvimento de programas através de pseudocódigo, o programa possui algumas limitações. Devido ao uso do compilador da linguagem *Java* para a geração do programa executável, o programa não captura possíveis erros de sintaxe existentes no código. Ademais, a ferramenta carece de um depurador visual e uma área para exibição das variáveis criadas em memória.

5. Estudo de Caso

No intuito de validar os benefícios da nossa *IDE* nos processos de aprendizado de programação de computadores, 10 (dez) alunos da disciplina de Algoritmos participaram

do desenvolvimento de um programa utilizando o ambiente proposto. Cada um dos participantes recebeu um treinamento de 5 (cinco) minutos na ferramenta e, em seguida, foram convidados a desenvolver um programa que retornasse a média de duas notas. Ao longo do desenvolvimento do programa, os alunos podiam realizar perguntas para o professor acerca da *IDE*.

Para determinar o possível impacto no aprendizado do aluno ao utilizar nossa *IDE*, cada participante respondeu a um questionário contendo 4 (quatro) perguntas sobre a utilização do programa e a sua percepção da ferramenta a partir da sua experiência.

6. Resultados e Discussão

No total, 10 estudantes com média de idade de 21.6 anos, participaram dessa avaliação. Desses, 8 são homens e 2 são mulheres. Dentro desse grupo, 6 utilizam o computador diariamente por períodos de, no mínimo, 4 horas. A Figura 4 sintetiza as respostas para as perguntas realizadas aos alunos.

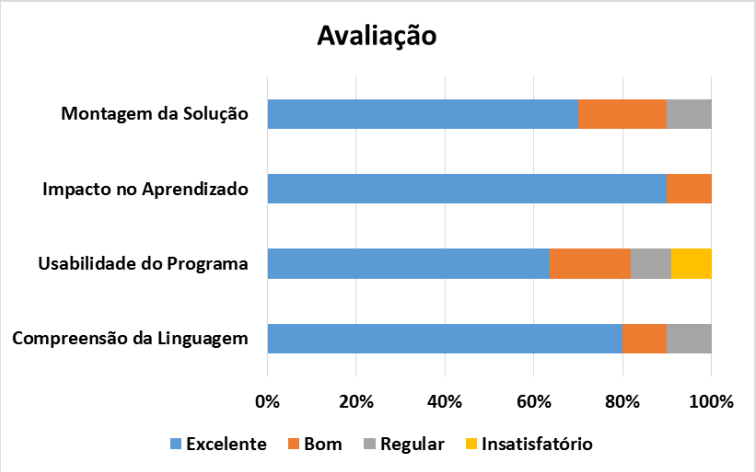


Figura 4 – Resultado do processo de compilação de um algoritmo.

Inicialmente, os participantes avaliaram sua experiência na montagem do programa que representava a solução para o problema aplicado. Do total de alunos, 8 consideraram a montagem bastante simples, sugerindo que a nossa *IDE* possibilita o simples desenvolvimento de programas.

Em seguida, os participantes foram questionados acerca da sua percepção do impacto desse ambiente no aprendizado de programação. Praticamente todos os alunos concordam que um ambiente na linguagem nativa auxilia no aprendizado de programação.

Também foi perguntado aos alunos sobre a usabilidade do programa, nesse quesito, um aluno queixou-se da falta do recurso de auto-completar para os comandos da nossa *IDE*. Esse resultado mostra que, embora simples, ainda faltam ao nosso ambiente aspectos voltados para a usabilidade do usuário.

Por fim, os alunos classificaram a sua compreensão da linguagem utilizada pela *IDE*. Oito alunos classificaram a linguagem como excelente, o que mostra o potencial dessa linguagem no ensino dos fundamentos de programação.

7. Conclusão

Este trabalho desenvolveu e avaliou com sucesso a utilização de um ambiente de desenvolvimento personalizável como ferramenta no aprendizado de fundamentos relacionados à área de programação. Ao avaliar os alunos, observou-se que o uso do idioma nativo facilita o aprendizado em programação de computadores. Dessa forma, acredita-se que esse tipo de solução possa ser assimilado nas matrizes curriculares dos cursos de Computação, tornando-se uma valiosa ferramenta na área educacional.

Como trabalhos futuros, pretende-se adicionar recursos para a depuração e otimização de códigos, assim como telas gráficas para a montagem de algoritmos através de fluxogramas. Também planeja-se estreitar as atividades da ferramenta com as principais linguagens de programação existentes no mercado, oferecendo ao usuário transições suaves entre os ambientes de ensino e de programação.

Referências

Eclipse Foundation. (2014) "Eclipse". Disponível em http://www.eclipse.org. Fevereiro.

Falkembach, G. A. M., Amoretti, M. S. M., Tarouco, L. R. and Vieiro, F. (2003) "Aprendizagem de algoritmos: Uso da estratégia ascendente de resolução de problemas". In: 8º Taller Internacional de Software Educativo.

Giraffa, L., Marczak, S. and Prikladnicki, R. (2005) "Em direção a um processo para desenvolvimento de Software Educacional". In: XI WIE - Workshop de Informática na Escola, São Leopoldo: SBC.

Henriques, A. C. (2010) "Aspectos da Teoria Piagetiana e Pedagogia". Instituto Piaget.

Jesus, A., Brito, G. S. (2010) "Concepção de Ensino-Aprendizagem de Algoritmos e Programação de Computadores: Prática Docente". Revista Varia Scientia. Vol. 09, No. 16, p. 149-158.

Perrenoud, P. (2010) "Dez competências para ensinar". Editora ArtMed.

Pontes, H. (2013) "Desenvolvimento de Jogos no Processo de Aprendizado em Algoritmos e Programação de Computadores". In: Proceedings of the XII Simpósio Brasileiro de Games e Entretenimento Digital (SBGames).

Rocha, H. V. (1991) "Representações Computacionais Auxiliares ao Entendimento de Conceitos de Programação". Campinas: UNICAMP.

Santiago, R. de., Dazzi, R. L. S. (2004) "Ferramenta de apoio ao ensino de Algoritmos".

Santos, R. P. and Costa, H. A. X. (2005) "TBC-AED: Um Software Gráfico para Apresentação de Algoritmos e Estruturas de Dados aos Iniciantes em Computação e Informática". In: I Congresso de Computação do Sul do Mato Grosso (COMPSULMT' 2005). Rondonópolis.

Santos, R. P. dos; Costa, H. A. X. (2006) "Análise de Metodologias e Ambientes de Ensino para Algoritmos, Estruturas de Dados e Programação aos Iniciantes em Computação e Informática". In: INFOCOMP, Vol. 5, No. 1.

Shabanah, S. and Chen J. X. (2009) "Simplifying algorithm learning using serious games". In: Proceedings of the 14th Western Canadian Conference on Computing Education. ACM, p. 34-41.