DANIEL BATISTA DIAS

DESENVOLVIMENTO DE FERRAMENTA EDUCATIVA PARA ENSINO DE PROGRAMAÇÃO BÁSICA NAS FACULDADES DOCTUM

FACULDADES UNIFICADAS DE TEÓFILO OTONI
TEÓFILO OTONI - MG
2016

DANIEL BATISTA DIAS

DESENVOLVIMENTO DE FERRAMENTA EDUCATIVA PARA ENSINO DE PROGRAMAÇÃO BÁSICA NAS FACULDADES DOCTUM

Monografia apresentada ao Curso de Sistemas de Informação das Faculdades Unificadas de Teófilo Otoni, como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação.
Área de concentração: Desenvolvimento.
Professor Orientador: Fabiano Souza Santos

FACULDADES UNIFICADAS DE TEÓFILO OTONI
TEÓFILO OTONI - MG
2016

Dedico esse projeto e o meu esforço durante esses 4 anos de estudos aos meus pais pelo incentivo de estudar algo que eu gosto, e por terem me apoiado em todos os momentos.

Dedico também a minha avó, que cuidou de mim como se eu fosse um filho, e que se foi pouco antes de me ver terminar a faculdade.

AGRADECIMENTOS

Agradeço a minha namorada por ser tão especial, estar sempre ao meu lado e por permitir que eu compartilhe com ela todos os meus momentos.

RESUMO

Com o principal objetivo de desenvolver uma ferramenta educativa no âmbito da disciplina de programação, tal monografia irá demonstrar a possibilidade de aumentar o interesse dos alunos em tópicos essenciais de desenvolvimento da lógica no curso de Sistemas de Informação da Faculdade Doctum de Teófilo Otoni. Desta forma dividiu-se a ferramenta em duas partes principais: a conceitual, onde irá reunir os conceitos de variáveis, condições, estruturas de repetição, e vetores, tendo as informações necessárias para que o aluno comece a desenvolver os códigos em uma linguagem de programação; e a gráfica, onde serão apresentadas as interações e execuções das instruções de códigos discutidas na parte conceitual. Assim, procurou-se demonstrar por meio desse trabalho de conclusão de curso que a técnica de programação, bastante utilizada nas disciplinas do curso de Sistemas de Informação nas quais os alunos demonstram maiores dificuldades, possa ser de mais fácil entendimento.

Palavras-chave: programação, lógica, instruções.

SUMÁRIO

INTRODUÇÃO

Como um dos maiores polos de atração do curso de S.I. (Sistemas de Informação), a Faculdade Doctum de Teófilo Otoni forma turma todos os anos. Todavia, um dos seus pontos fracos é a falta de interesse dos alunos com a maior parte das disciplinas. Mesmo fazendo parte de quase todos os períodos do curso, a disciplina de Programação é a que mais sofre com isso, abaixando a média das notas das turmas.

Esse seguinte trabalho de conclusão de curso tem como principal objetivo criar uma ferramenta para auxiliar os alunos da Faculdade Doctum de Teófilo Otoni que estejam estudando Programação. A ideia de desenvolver a ferramenta partiu da observação da dificuldade dos alunos com a disciplina, podendo abrangir os leigos em informática.

Dessa forma, por ser uma ferramenta de livre distribuição e sem fins lucrativos, poderá ser usada por aqueles com interesse de aprendizagem. Entretanto, o alvo desse desenvolvimento serão os alunos que estejam iniciando o curso de Sistemas de Informação.

A ferramenta de desenvolvimento utilizada para o desenvolvimento do projeto de pesquisa foi a linguagem de programação Java. E ao final, espera-se validar a viabilidade do uso de uma ferramenta que tentará impulsionar o aprendizado, aumentando o rendimento e entendimento dos alunos que usufruírem da ferramenta.

A era da informação nos trouxe facilidade na busca por informações e compartilhamento de conhecimentos, disponibilizados por meio da internet, ou em forma de alguma ferramenta ou software. A quantidade de softwares cresceu exponencialmente nos últimos anos, assim como as desenvolvedoras. Consequentemente, a busca por profissionais do ramo de informática cresceu junto.

Com a chegada da Linguagem de Programação Java, os softwares deixaram de ser exclusivo de computadores, e passaram a operar aviões, computadores de

bordo dos carros, assim como sistema de ABS (usado para evitar que as rodas do veículo travem) e câmbio automático, micro-ondas, geladeiras, smart tv's, etc.. Assim como os softwares se expandiram para novos dispositivos, eles também se expandiram para novas áreas.

Existem vários softwares voltados para fazer cálculos complexos, softwares para resolver problemas científicos, softwares voltados para usos gerenciais e controles de empresas, e também softwares educativos, que é o intuito dessa pesquisa. O surgimento e crescimento dos softwares educativos não aconteceram por acaso, eles surgiram para auxiliar alunos com dificuldades ou deficiência em uma determinada área ou matéria.

O desenvolvimento de uma ferramenta educativa, voltada para os alunos iniciantes do curso de Sistemas de Informação na faculdade Doctum de Teófilo Otoni, que ensine os conceitos de condições, loops, operadores lógicos e vetores, poderia auxiliar os que tem dificuldades em entender os conceitos mais básicos de programação. A partir disso, este trabalho se baseia na seguinte pergunta: um software educativo que ensine os conceitos básicos de programação conseguiria aumentar o interesse dos alunos na disciplina, e diminuir a evasão dos alunos que entram no curso de Sistemas de Informação na Faculdade Doctum ?

H0: a ferramenta irá acabar com a dificuldade de todos os alunos na disciplina de programação, pois trabalhará melhor a capacidade de cada aluno.

H1: o desenvolvimento da ferramenta servirá como material de apoio na construção do conhecimento de programação básica.

H2: seria viável a elaboração da ferramenta, uma vez que poderá ser usado em qualquer instituição de ensino superior, na disciplina de programação.

H3: o desenvolvimento da ferramenta se torna viável, pois demonstrará a execução de algoritmos de forma amigável e de fácil entendimento.

Diante do exposto, a ferramenta trará uma abordagem sobre os conceitos de: condições, loop, operadores lógicos e vetores, para servir como material de estudo e apoio. Para tanto foi necessário, compreender as maiores dificuldades encontradas por alunos nestas disciplinas, estudar as melhores práticas e técnicas no desenvolvimento de softwares educativos, e aprofundar um pouco mais na linguagem de programação Java, para adquirir conhecimento para o desenvolvimento do projeto.

1. REFERENCIAL TEÓRICO

1.1 ALGORITMOS

Um conceito fundamental para a Ciência da Computação é algoritmo. Para Salvetti e Barbosa (1998, p. 5), “Um algoritmo é uma sequência de instruções, bem definida e finita, cuja execução resulta na realização de uma determinada tarefa.” De maneira natural, alguns tipos de algoritmos estão presentes no nosso dia a dia, não necessariamente envolvendo aspectos computacionais. Uma receita de bolo, um manual de utilização de uma televisão, uma partitura musical são exemplos de algoritmos.

Segundo os autores Szwarcfiter, Markenzon (2010, p. 1):

> Um algoritmo computa uma saída, o resultado do problema, a partir de uma entrada, as informações inicialmente conhecidas e que permitem encontrar a solução do problema. Durante o processo de computação o algoritmo manipula os dados, gerados a partir de sua entrada. Quando os dados são dispostos e manipulados de uma forma homogênea, constituem um tipo abstrato de dados. Este é composto por um modelo matemático acompanhado por um conjunto de operações definido sobre esse modelo. Um algoritmo é projetado em termos de tipos abstratos de dados. Para implementá-los numa linguagem de programação é necessário encontrar uma forma de representá-los nessa linguagem, utilizando tipos e operações suportadas pelo computador. Na representação do modelo matemático emprega-se uma estrutura de dados.

1.1.1 V**ariáveis**

Para Schildt (1996, p.3) “variável é uma posição na memória, que é usada para guardar um valor que pode ser modificado pelo programa”.

Sebesta (2003, p. 183) classifica as variáveis como “uma abstração de célula ou de um conjunto de células de memória de computador. Os programadores frequentemente pensam nas variáveis como localizações de memória”.

1.2 BLOCOS DE INSTRUÇÕES

1.2.1 C**omandos de condição e operadores lógicos**

É muito comum encontrar situações em que a execução de uma ou mais instruções devem estar condicionadas ao fato de que uma condição seja satisfeita. Há situações, por exemplo, em que há a necessidade de se fazer uma escolha entre uma ou mais sequencias de instruções, se a verificação da condição for satisfeita, uma determinada sequência de comandos deve ser executada.

Sebesta (2003, p. 300) diz que “uma instrução de condição oferece os meios de escolher entre dois ou mais caminhos de execução em um programa”.

O quadro mostra um exemplo de um algoritmo condicional em Portugol (Quadro 1):

Quadro 1. Exemplo de uso do comando SE

```
SE (alunos != 15) ENTÃO // se a quantidade de alunos for diferente de 15
        ESCREVA “Todos os alunos estão presente.”
SENÃO // só será executada se a condição acima não for satisfeita
        ESCREVA “Nem todos os alunos estão presente”
```

Fonte: do próprio autor

Quando há a necessidade de trabalhar com duas ou mais condições ao mesmo tempo, são utilizados operadores lógicos, que são responsáveis pela formação de novas proposições lógicas. Sebesta diz que os operadores lógicos são utilizados para comparar mais de uma condição em uma mesma expressão, ou seja, pode-se fazer mais de uma comparação ao mesmo tempo. Os operadores lógicos são: não (not), e (and), ou (or).

Exemplo do uso de um operador lógico em Portugol (Quadro 2):

Quadro 2. Exemplo de uso do operador lógico E

```
LEIA numero
SE (numero > 5) E (numero < 20) ENTÃO
        ESCREVA “O número está entre 5 e 20”
SENÃO
        ESCREVA “O número não está entre 5 e 20”
```

Fonte: do próprio autor

1.2.2 C**omandos de Repetição**

Um comando de repetição é utilizado quando se precisa que um determinado conjunto de comandos ou instruções sejam executados um número definido ou indefinido de vezes, ou enquanto um determinado estado de coisas prevalecer ou até que seja alcançado.

1.2.3 C**omando ENQUANTO**

O comando ENQUANTO é uma estrutura de repetição usado para repetir a execução de uma determinada sequência de comandos um certo número de vezes. Exemplo de um algoritmo usando o comando enquanto em Portugol:

Quadro 3. Exemplo de uso do comando ENQUANTO

```
num  <- 0  // declarando uma variável e passando 0 como valor

ENQUANTO (num < 20) FAÇA  // enquanto num for menor que 20 faça...
        ESCREVA num  // imprime o valor de num na tela
        num <- num + 1  // num é incrementado em +1 a cada vez que o loop rodar

FIMENQUANTO  // se num for igual ou maior que 20, o loop encerra
```

Fonte: do próprio autor

No quadro acima (Quadro 3), é declarada uma variável e passado o valor 0 a ela. Na segunda linha, o laço (loop) ENQUANTO é iniciado, e compara se o valor da variável num é menor do que 20, se for, irá executar as duas linhas abaixo, e após num será incrementado em mais 1. Isso irá se repetir até que num for menor do que

20.

1.2.4 C**omando PARA**

O comando para, assim como o comando enquanto, também é usado para repetir uma sequência de comandos. Exemplo de um algoritmo usando o comando para em Portugol:

Quadro 4. Exemplo de uso do comando PARA

```
num <- 20

PARA (num > 0) FAÇA
        ESCREVA num
        num <- num – 1

FIMPARA
```

Fonte: do próprio autor

A variável num é declarada e passada o valor 20 para ela. Na segunda linha, o comando PARA é iniciado, e enquanto num for menor do que 20, as duas linhas abaixo serão executadas. Na primeira dessas linhas, o valor de num é exibido, e na linha seguinte, num é incrementado em mais 1.

1.3 PROGRAMAÇÃO DE COMPUTADORES

Computadores são máquinas muito rápidas, porém não muito inteligentes. Um computador por si só não consegue fazer nada. Cada funcionalidade de um computador foi previamente programada por um ser humano. Inicialmente, os programadores tinham que programar diretamente no computador, através de códigos binários (0 e 1). “Nos primeiros tempos da computação propriamente dita os programas eram escritos em código de máquina e colocados diretamente no computador por meio de cabos e fios” (FONSECA FILHO, 2007, p. 111).

1.4 LINGUAGENS DE PROGRAMAÇÃO

Linguagem de programação pode ser definida, segundo Linder, como:

> Uma linguagem de programação é um vocabulário e um conjunto de regras gramaticais usadas para escrever programas de computador. Esses programas instruem o computador a realizar determinadas tarefas específicas. Cada linguagem possui um conjunto único de palavras-chaves (palavras que ela reconhece) e uma sintaxe (regras) específica para organizar as instruções dos programas.

Um grande avanço ocorreu na computação quando se conseguiu desenvolver programas que traduzissem instruções escritas originariamente numa linguagem dos seres humanos para a linguagem de máquina. Com o passar do tempo, foram surgindo as linguagens de programação para facilitar o desenvolvimento das aplicações. Segundo Fonseca Filho (2007, p. 112):

> Percebeu-se claramente que os programas em código de máquina eram extremamente difíceis de editar e modificar, e quase impossíveis de se compreender. A comunidade computacional logo entendeu que era necessário inventar uma notação simbólica para tornar os programas mais fáceis de escrever.

1.5 PROGRAMAÇÃO ESTRUTURADA

As primeiras linguagens de programação foram surgindo, e com elas veio o conceito de linguagem estruturada, que segundo Schildt (1996, p. 18), as define assim:

> A característica especial de uma linguagem estruturada é a compartimentalização do código e dos dados. Trata-se da habilidades de uma linguagem seccionar e esconder do resto do programa todas as informações necessárias para se realizar uma tarefa específica. Uma das maneiras de conseguir essa compartimentalização é pelo uso de sub-rotinas que empregam variáveis locais (temporárias). Com o uso de variáveis locais é possível escrever sub-rotinas de forma que os eventos que ocorrem dentro delas não causem nenhum efeito inesperado nas outras partes do programa.

Para Dall'Oglio (2015, p. 9) “A abordagem mais usada durante muito tempo foi a programação procedural (ou estruturada). Esta ocorre quando o programa é construído com base em um conjunto de procedimentos (procedures). Um procedimento basicamente é uma unidade de código que pode ser reaproveitada em diversas situações”

Exemplo de código em uma linguagem estruturada:

Quadro 5. Exemplo de código na Linguagem de Programação C

```
int *p1;      // p1 é um ponteiro que aponta para um inteiro
int *p2;      // p2 é outro ponteiro que aponta para um inteiro
p1 = &a;    // o valor de p é o endereço de a
p2 = &b;    // p aponta para b
c = *p1 + *p2;
```

Fonte: do próprio autor

1.6 PROGRAMAÇÃO ORIENTADA A OBJETOS

A tecnologia veio evoluindo, hardwares cada vez melhores, softwares melhores, e isso também exigiu mais das linguagens de programação.

> Portanto, as linguagens que começaram com instruções de máquina e Assemblers deram lugar às estruturadas. As linguagens estruturadas evoluíram sempre focadas na legibilidade e elegância de programação. Também trouxeram facilidades de funções e sub-rotinas ou procedimentos e de utilização de bibliotecas para a modularização de sistemas. Em paralelo, a área de programação evoluía com novas técnicas e métodos para melhorar a qualidade dos sistemas produzidos. Esta necessidade de melhoria trouxe mais um conceito, que apesar de ter nascido na década de 1970, só tomou forma e cresceu na década de 1980, e até hoje domina o projeto de qualquer linguagem que seja criada com o intuito de concorrer com as outras: Orientação a Objetos (CLAYTON et al, 2010, p.13).

Claro e Sobral (2008, p. 8) definem que os objetos são como as pessoas, e que as classes podem ser os grupos nas quais elas pertencem (como grupo de estudos, pesquisas), os métodos são as características dos objetos, que podem se equiparar ao que as pessoas podem fazer, como andar, correr, pular…, e os atributos são as características dos objetos, que podem ser as características das pessoas, como cor da pele, cor do cabelo, tamanho, etc..

Dall'Oglio (2015, p. 9), traz também uma ótima explicação sobre os conceitos básicos de Orientação a Objetos. “Na orientação a objetos utilizamos uma ótica mais próxima do mundo real. Lidamos com objetos: estruturas que carregam dados e comportamento próprio, além de trocarem mensagens entre si com o objetivo de formar algo maior, um sistema.”

Outra grande característica da Linguagem Orientada a Objetos é poder separar os códigos em blocos, e estes podem ser chamados a qualquer momento

no código, e até mesmo em outras aplicações. Uma grande facilidade, pois evita-se erros assim, já que o código foi previamente testado em alguma outra aplicação, evita ficar reescrevendo código, e também deixa o código mais fácil de dar manutenção.

Com uma tecnologia que possibilitou desenvolver melhores softwares que se relacionam com o mundo real, desenvolver certos aplicativos que antes eram trabalhosos de desenvolver em linguagens estruturadas, se tornaram mais simples nas linguagens orientadas a objetos. Essa nova tecnologia dava maior poder e facilidade em certos tipos de projetos aos desenvolvedores. Segundo Sebesta (2003, p. 16), em seu livro sobre Conceitos de Linguagens de Programação:

> Acredita-se que a profundidade com a qual as pessoas podem pensar é influenciada pelo poder de expressividade da linguagem que elas usam para comunicar seus pensamentos. As pessoas que tem apenas um fraco entendimento da linguagem natural são limitadas na complexidade de seus pensamentos, particularmente na profundidade de abstração. Em outras palavras, é difícil para essas pessoas criar conceitos de estruturas que elas não podem descrever verbalmente ou expressar na escrita.

Exemplo de código em uma linguagem orientada a objetos:

Quadro 6. Exemplo de código em Java

```
public class Fornecedor {       // declarando uma classe
String nome, cod, cnpj, endereco;     // criando variáveis do tipo String

public Fornecedor(String nome, String cod, String cnpj, String endereco){
this.nome = nome;
this.cod = cod;
this.cnpj = cnpj;
this.endereco = endereco;
}
}
```

Fonte: do próprio autor

1.7 LINGUAGEM DE PROGRAMAÇÃO JAVA

Uma das linguagens de programação mais utilizadas atualmente, e que vem sendo utilizada em boa parte das faculdades, Java se tornou uma linguagem poderosa, e com uma comunidade ainda mais poderosa, com bibliotecas que podem fazer as mais variadas funções, e diversas API's.

O grande crescimento da linguagem Java não se deve somente ao fato dela ter sido inicialmente lançada como uma linguagem voltada para a web, quando esta ainda estava em seu início, mas também por causa da sua possibilidade em inúmeros dispositivos e hardwares, como em micro-ondas, computadores de bordo, geladeiras, etc..

> Java é agora utilizada para criar páginas da Web com conteúdo interativo e dinâmico, para desenvolver aplicativos corporativos de grande porte, para aprimorar a funcionalidade de servidores da World Wide Web (os computadores que fornecem o conteúdo que vemos em nossos navegadores da Web), fornecer aplicativos para dispositivos destinados ao consumidor final (como telefones celulares, pagers e assistentes pessoais digitais) e para muitas outras finalidades. (DEITEL, 2003, p. 59)

Ainda segundo Deitel (2003, p. 374), ele relata da seguinte maneira a diferença entre uma linguagem estruturada e orientada a objetos:

> Em C e em outras linguagens de programação procedurais, a programação tende ser orientada a ações. Em Java, a programação é orientada a objetos. Em C, a unidade de programação é a função (chamada de método em Java). Em Java, a unidade de programação é a classe a partir da qual os objetos são instanciados (isto é, criados) em algum momento. As funções não desapareceram em Java; em vez disso, elas estão encapsuladas como métodos, com os dados que elas processam, dentro das “paredes” das classes.

1.8 ENGENHARIA DE SOFTWARE

A engenharia de software é importante no desenvolvimento de qualquer software, pois através dela se estuda os requisitos, calcula-se os gastos, tempo para entrega, tempo para liberação de partes do software, e da ao cliente ideia de valores e prazos. Para Pressman (2011, p. 39), “Engenharia de software é o estabelecimento e o emprego de sólidos princípios de engenharia de modo a obter software de maneira econômica, que seja confiável e funcione de forma eficiente em máquinas reais”.

Os modelos de processos definem como são feitos os requisitos, como são

feitas as iterações (se houver), quem, quando e como vão se comunicar com o cliente para deixá-lo informado sobre o andamento do projeto e, dependendo do modelo, fazê-lo parte da equipe.

1.8.1 M**odelo de Processo Cascata**

"O modelo de processo cascata, algumas vezes chamado ciclo de vida clássico, sugere uma abordagem sequencial e sistemática para o desenvolvimento de software, começando com o levantamento de necessidades por parte do cliente, avançando pelas fases de planejamento, modelagem, construção, emprego e culminando no suporte contínuo do software concluído" (PRESSMAN, p. 59, 2011).

1.8.2 Processo XP

"A Extreme Programming (programação extrema) emprega uma abordagem orientada a objetos como seu paradigma de desenvolvimento preferido, e envolve um conjunto de regras e práticas constantes no contexto de quatro atividades metodológicas: planejamento, projeto, codificação e testes" (PRESSMAN, p. 87, 2011).

1.8.3 Processo Scrum

Pressman (2011, p. 95) define o modelo scrum da seguinte forma:

> Scrum (o nome provém de uma atividade que ocorre durante a partida de rugby) é um método de desenvolvimento ágil de software concebido por Jeff Sutherland e sua equipe de desenvolvimento no início dos anos 1990.
> Os princípios do Scrum são consistentes com o manifesto ágil e são usados ara orientar as atividades de desenvolvimento dentro de um processo que incorpora as seguintes atividades estruturais: requisitos, análise, projeto, evolução e entrega. Em cada atividade metodológica, ocorrem tarefas a realizar dentro de um padrão de processo chamado sprint. O trabalho realizado dentro de um sprint *o número de sprints necessários para cada atividade metodológica varia dependendo do tamanho e da complexidade do produto) é adaptado ao problema em questão e definido, e muitas vezes modificado em tempo real, pela equipe Scrum.

1.8.4 Engenharia de Usabilidade

Assim Pádua definiu a engenharia de usabilidade:

> A atividade de Avaliação de usabilidade pode ser considerada como uma atividade de garantia de qualidade dentre outras que são utilizadas no processo de desenvolvimento do software.
> Diferentes tipos de avaliação, com objetivos e características próprias, podem ser utilizados. As avaliações mais confiáveis, no entanto, envolvem experimentações com a participação de representantes dos usuários. Estas avaliações, normalmente fazem uso de roteiros de tarefas que são executadas por usuários com a utilização de protótipos ou com o produto final, dependendo do estágio de desenvolvimento do software.

1.9 SOFTWARES EDUCATIVOS

Para Sancho (1998, p. 169), “o software educativo é um conjunto de recursos informáticos projetados com a intenção de serem usados em contexto de ensino e aprendizagem”.

Bona diz que:

> Os software educativos podem ser um notável auxiliar para o aluno adquirir conceitos em determinadas áreas do conhecimento, pois o conjunto de situações, procedimentos e representações simbólicas oferecidas por essas ferramentas é muito amplo e com um potencial que atende boa parte dos conteúdos das disciplinas.

1.10 SOFTWARES COMO FERRAMENTA DE ENSINO

Talvez encontrar a maneira de como lecionar seja uma das tarefas mais difíceis para os profissionais que trabalham com ensino e educação. Encontrar um meio que chame a atenção dos alunos, e consiga passar o conhecimento não é tarefa fácil. Mais recentemente, os softwares educativos vem sendo utilizados dentro e fora de salas de aula como uma forma de auxílio. Para Fialho (2007, p. 16):

> A exploração do aspecto lúcido, pode se tornar uma técnica facilitadora na elaboração de conceitos, no reforço de conteúdos, na sociabilidade entre os alunos, na criatividade e no espírito de competição e cooperação, tornando esse processo transparente, ao ponto que o domínio sobre os objetivos propostos na obra seja assegurado.

Fazer uso da tecnologia como uma ferramenta ou metodologia de ensino, é

ter o auxílio de um poderoso aliado. Para Oliveira (2007, p. 16), um software educativo é mais do que uma ferramenta comum, assim ele define um software educativo: “O software educativo é uma dessas ferramentas privilegiadas que podem integrar favoravelmente o projeto pedagógico da escola, ampliando a efetividade do processo ensino-aprendizagem”.

Mercado (2002, p. 129) também mostra o seu ponto de vista sobre o uso de tecnologias dentro da sala de aula, afirmando que:

> O uso da informática pode contribuir para auxiliar os professores na sua tarefa de transmitir o conhecimento e adquirir uma nova maneira de ensinar cada vez mais criativa, dinâmica, auxiliando novas descobertas, investigações e levando sempre em conta o diálogo. E, para o aluno, pode contribuir para motivar a sua aprendizagem e aprender, passando assim, a ser mais um instrumento de apoio no processo de ensino-aprendizagem, abrindo possibilidade de novas relações entre os alunos, que estão inseridos numa sociedade diferente da dos seus pais.

“Um software educativo pode ter vários usos e pode também ser voltado para diferentes áreas, assim como podem ser utilizados de várias maneiras, dependendo da metodologia aplicada por cada professor.” SOFFA; ALCÂNTARA, assim definiram os softwares educativos de maneira geral:

> Um software educacional é um programa de computador utilizado pela escola de forma adequada, mas nem sempre produzido com o desígnio de emprego no sistema escola. O software educativo é engendrado com a finalidade de levar o aluno a construir um determinado conhecimento referente a um conteúdo didático. O objetivo de um software educativo é a de favorecer os processos de ensino-a-aprendizagem e sua característica principal é seu caráter didático.

1.11 ENSINO APRENDIZAGEM

Segundo Bona (2009) há uma grande variedade de softwares disponíveis que podem contribuir de forma significativa para o processo ensino-aprendizagem, além de se caracterizarem como alternativas enriquecedoras que auxiliam na didática do professor.

Bona ainda diz que:

> As novidades tecnológicas e a grande variedade de softwares educativos disponíveis na rede mundial de computadores podem contribuir de forma expressiva para facilitar o processo ensino-aprendizagem e oferecer para o professor diferentes e enriquecedoras alternativas didáticas auxiliares. Muitos softwares educacionais estão se tornando uma solução reveladora e interessante, à medida que são empregados nas mais variadas situações.

2. MATERIAIS

Este capítulo tem como objetivo apresentar as ferramentas e tecnologias envolvidas no desenvolvimento deste trabalho. Apresenta com detalhes os softwares utilizados bem como onde obtê-los, algumas bibliotecas usadas no aplicativo criado e a descrição dos hardwares utilizados.

2.1 COMPUTADOR UTILIZADO

O computador utilizado para desenvolvimento e testes do software tem a seguinte configuração: placa mãe M5A78L-M LX/BR, processador Intel i5 4460, 3.20 GHZ, 2 pentes memória RAM DDR4, 8 GB, 2133 MHZ Kingston cada, placa de vídeo Nvidiia GT 240 1GB, HD 1 TB 7200 rpm SATA III Wester Digital.

2.2 IDE NETBEANS

Para o desenvolvimento desse trabalho, foi utilizado o NetBeans IDE versão 8.1. Esta IDE pode ser obtida gratuitamente no seguinte endereço: https://netbeans.org/downloads/

A linguagem de programação Java, versão 8, foi utilizada para o desenvolvimento da ferramenta. A linguagem Java e o JDK (Kit de Desenvolvimento Java, utilizado para o desenvolvimento das aplicações), podem ser baixados gratuitamente no site: https://www.java.com/pt_BR/download/

2.2.1 Biblioteca ActionListener

O ActionListener é usado para que o software fique “escutando”, para que caso um determinado evento ocorra, ele irá executar uma determinada ação. Por exemplo: se houver algum botão na interface, quando esse botão é clicado, e se

houver um ActionListener dentro daquela classe, ela irá executar alguma ação, caso tenho sido programado para isso.

2.2.2 Biblioteca ActionPerformed

A biblioteca ActionPerformed serve para tratar e executar determinado evento no software definido pelo próprio programador. Para que o ActionPerformed seja executado, ele precisa ser chamado pelo ActionListener.

2.2.3 Biblioteca Graphics2D

A biblioteca Graphics2D contém os objetos aritméticos, traços, retângulos e demais objetos aritméticos que aparecem no projeto. Essa biblioteca também é a responsável por exibir os textos presentes no software, assim como as cores deles.

“A classe Graphics2D herda da classe Graphics, que provê controles mais sofisticados sobre figuras geométricas, coordenadas, gerenciamento de cores e layout de textos.” Informação disponível em (<https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics2D.html>)

3. FERRAMENTA EDUCATIVA

Este capítulo é destinado ao desenvolvimento da ferramenta educativa. Descreve toda a arquitetura de construção exibindo desde os requisitos funcionais até a demonstração de alguns códigos utilizados nas principais telas do sistema.

3.1 PRINCIPAIS PROBLEMAS

Foi observado pelo próprio autor, durante o curso Técnico em Informática no Centro Educacional Tecnológico (2012 – 2013), e durante o curso de Sistemas de Informação, na faculdade Doctum de Teófilo Otoni, que os alunos possuem dificuldades na disciplina de programação. Há uma grande dificuldade em todas as matérias, mas talvez a que os professores mais notem, seja na disciplina de programação.

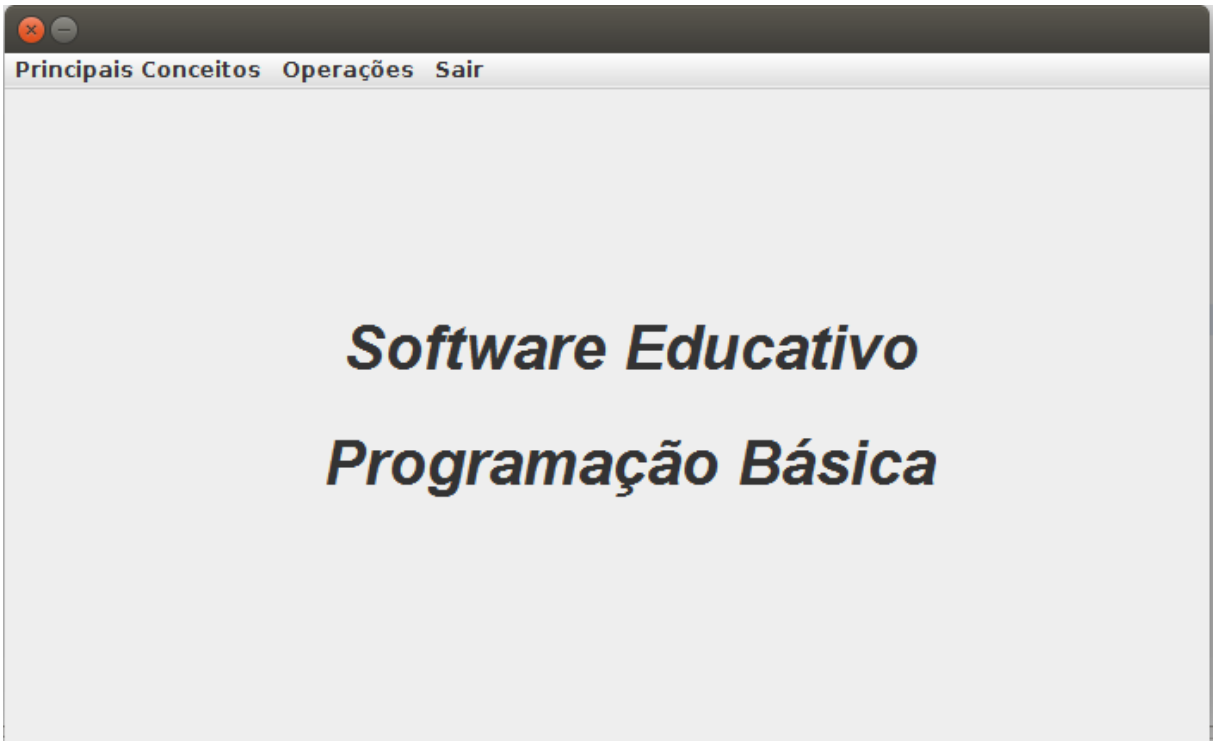
Desenvolver uma ferramenta educativa não é uma tarefa fácil, pois a ferramenta nem sempre irá conseguir ajudar a todos, e também porque nem todos terão a curiosidade de experimentar.

O objetivo é atingir e conseguir dar bons resultados ao máximo de alunos possível.

3.2 CODIFICAÇÃO DAS INTERFACES DO SISTEMA

A primeira tela que o usuário encontra ao executar o software, é o menu, onde ele poderá escolher qual conceito de programação ver. A tela de menu, assim como todas as outras telas, são chamados através do método jFrame.

Figura 1: tela de menu



Fonte: projeto desenvolvido pelo próprio autor.

Dentro de cada jButton que aparece no menu Operações, há um pedaço de código para chamar a janela, e a classe referente ao jButton que foi clicado.

Figura 2: demonstração de código que chama os menus

```
383         Variaveis var = new Variaveis();
384         JFrame fVar = new JFrame();
385         fVar.add(var);
386         fVar.setSize(800, 600);
387         fVar.setTitle("Variáveis");
388         fVar.setVisible(true);
389         fVar.setLocationRelativeTo(null);
390         fVar.setResizable(false);
391         fVar.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

Fonte: projeto desenvolvido pelo próprio autor.

Na linha 383, a classe Variaveis é declarada, e a responsável por chamá-la será a variável var. Na linha 384, é criada uma variável do tipo jFrame, que é a classe usada para a criação das telas. Essa variável se chama fVar. Na tlinha 385, a variável var é adicionada a classe jFrame para que seja possível exibi-la dentro da janela que será iniciada e é inicializada. Quando o software chega nessa parte, uma

janela é exibida na tela, e na linha abaixo (linha 386), são passados dois parâmetros, width e height, para que a tela tenha o tamanho 800 de largura e 600 de altura. A linha 387 serve para dar título a tela, e a linha3897 serve para que a tela apareça no centro do monitor. A linha 390 não permite que a tela seja redimensionada (maximizada) e a linha 391 fecha a janela quando o usuário clica no “X” que fica no topo da tela.

As outras telas são chamadas de maneira bem parecida, mudando apenas o nome das variáveis da classe que se refere a cada jButton.

3.2.1 Menu de Conceitos

As janelas desse menu são parecidas, todas contém jTextField (campo para inserção de título) mostrando o nome do menu, e abaixo uma jTextArea (campo para inserção de frases e/ou textos) falando sobre aquele determinado conceito.

São 5 menus conceituais, o primeiro falando sobre variáveis, o segundo strings, o terceiro sobre comandos de condição, o quarto sobre loops, e o quinto sobre vetores. Em cada um desses menus, exceto no menu strings, há um submenu, no qual abordam, conceitos, como usar, exercícios e respostas. No menu strings são abordados conceitos e funções de strings em linguagem C.

Os submenus conceituais explicam sobre o termo abordado. Os submenus “Como usar”, explicam e dão exemplos de como usar, declarar e passar parâmetros para o termo abordado. Nos submenus de exercícios, há alguns exercícios propostos de acordo com as explicações dos dois submenus anteriores, e nos submenus “Respostas”, há as respostas para os exercícios propostos.

3.2.2 Menu de Operações

3.2.2.1 Janela de Variáveis

A janela variáveis, tem uma variável de controle, a variável “i”. Durante toda a execução, a variável “i” é incrementada. Quando a variável alcança determinados valores, ela executa uma determinada ação.

Figura 3: exemplo de código para adicionar botão e jlabel na tela

```
54                  add(inserir_valor);
55                  add(campo_valor);
56                  add(botao3);
```

Fonte: projeto desenvolvido pelo próprio autor.

A linha 54 exibe na tela a frase para que o usuário insira um número, e a linha 55 adiciona na tela o campo para que o usuário insira os valores. A linha 56 exibe na tela o botão “Enviar”, para que o programa leia o valor digitado.

A pausa da execução ocorre da seguinte forma: uma variável booleana chamada “pause” é declarada e passada o valor false. Se o usuário clicar no botão, e o valor de pause for false, esse valor passará a ser true, e então ocorre a pausa no software. Se o usuário clicar, e o valor de pause for true, ele passará a ser false, fazendo com o que a variável i volte a incrementar e assim o software volta a executar. Todas as janelas de execuções contém o botão pause, e da mesma forma ele é declarado e tratado em todas elas.

Figura 4: demonstração do código que pausa a aplicação

```
207             if(e.getSource() == botao2){
208                 if(!pause)
209                     pause = true;
210                 else
211                     pause = false;
```

Fonte: projeto desenvolvido pelo próprio autor.

Abaixo são mostrados trechos do código de quando a variável i chega a determinado valor:

Figura 5: demonstração de algumas linhas do código da aplicação

```
80              if(i >= 3)
81                  g2.drawString(""+auxX, 105, 100);
82              if(i >= 4){
83                  g2.drawString("x", 570, 100);
84                  if(i >= 13 && i <= 15)
85                      g2.setColor(Color.red);
86                  g2.drawString(" "+auxX, 560, 130);
87                  g2.setColor(Color.black);
```

Fonte: projeto desenvolvido pelo próprio autor.

Na linha 80, é feita uma comparação e se “i” for maior ou igual a 3, ele irá exibir na tela o valor que o usuário informou para a variável “x”, na posição horizontal 60 e vertical 100.

Na linha 82, compara se i é maior ou igual a 4, e se for, o caractere x é exibido na tela, na posição horizontal 570 e vertical 100.

Na linha 84 compara se i é maior ou igual a 13 e se i é menor ou igual a 15, se for, todos os comandos que estiverem nas linhas abaixo irão jogar na tela os caracteres na cor vermelha.

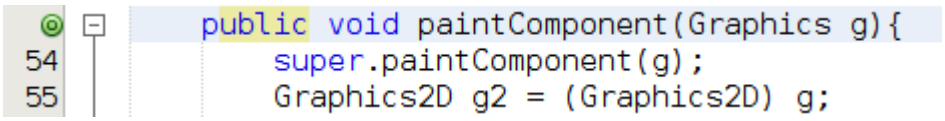
Na linha 86, é exibido na tela o valor de auxX na posição horizontal 560 e vertical 130.

Na linha 87, o comando indica que todos os comandos abaixo que imprimirem algum caractere, irão aparecer na cor preta.

3.2.2.2 Janela de Condição

Para que os elementos gráficos sejam exibidos, é necessário chamar um determinado método da classe Graphics. Essa biblioteca também disponibiliza texto e cores.

Figura 6: demonstração de algumas linhas do código da aplicação



Fonte: projeto desenvolvido pelo próprio autor.

Na linha 53, o método paintComponent (Graphics g) possibilita usar os elementos gráficos da biblioteca Graphics2D, passando como parâmetro uma variável do tipo classe chamada g. Na linha 54, a classe g chama o método paintCOmponent() da classe pai (a classe Graphics). Na linha 55, é declarada um objeto da classe Graphics2D chamado g2.

Figura 7: demonstração de algumas linhas do código da aplicação

```
49          Font f = new Font("Dialog", Font.PLAIN, 20);
50          g2.setFont(f);
51          g2.drawString("int x, y, z", 50, 50);
```

Fonte: projeto desenvolvido pelo próprio autor.Na linha 49, a variável f, do tipo fonte, é declarado, entre parênteses é passado o nome Dialog, e logo após, é escolhida a fonte PLAIN, tamanho 20. Na linha 50, é informado que tudo que vier nas linhas abaixo irá usar a fonte estabelecida por “f”, e na linha 51 algumas informações são exibidas na tela.

O software vai avançando a medida que a variável i vai crescendo. Quando i atinge um determinado valor, determinada ação é executada, como por exemplo:

Figura 8: demonstração de algumas linhas do código da aplicação

```
76          if(i >= 2)
77              g2.drawString("x = ", 60, 100);
78          if(i >= 3)
79              g2.drawString(""+auxX, 105, 100);
80          if(i >= 13 && i <= 15)
81              g2.setColor(Color.red);
82          g2.drawString(" "+auxX, 560, 130);
83          g2.setColor(Color.black);
```

Fonte: projeto desenvolvido pelo próprio autor.

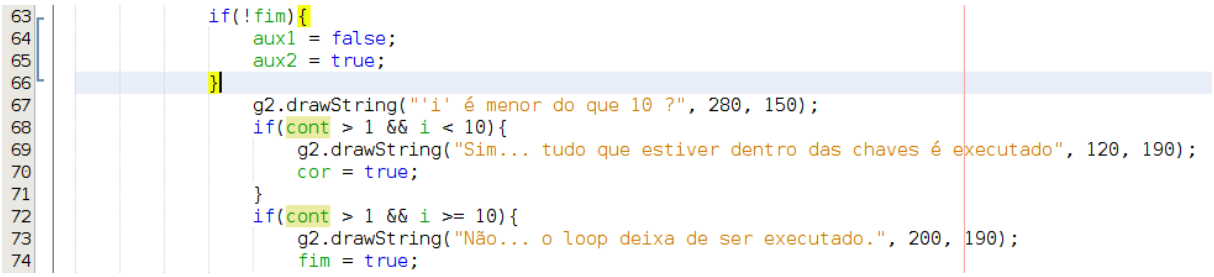
Na linha 76 é feita uma comparação, e se i for maior ou igual a 2, a linha 77 é executada e a variável g2 (uma classe do tipo Graphics2D) é chamada, e também é chamado o método drawString (para exibir um ou mais caracteres na tela). Dentro dos parênteses precisam ser passados 3 parâmetros, a string a ser exibida, a posição horizontal e a posição vertical em que a string deve aparecer. Na linha 81, o método setColor da variável g2 é chamado. Esse método serve para mudar a cor das próximas strings que serão inseridas na tela. Apenas um comando precisa ser passado, que é a cor (em inglês) escolhida.

3.2.2.3 Janela de Loop

A janela de loop possui uma variável de controle, a variável “cont”, que começa valendo 0. A medida que “cont” é incrementado, o software vai avançando.

Inicialmente, é jogada uma string na ela, que irá ficar até o final, alterando apenas um valor quando a variável “i” mudar o seu valor. Essa variável é usada para dar continuidade, ou parar a execução do software, dependendo do valor de “i”. As variáveis aux1, aux2 e aux3 ajudam a destacar em vermelho as letras. Por exemplo, quando aux1 for verdadeiro, os outros dois serão falsos, e o primeiro parâmetro do loop ficará em vermelho. Quando aux2 for verdadeiro, os outros dois serão falsos e o segundo parâmetro do loop ficará vermelho, e quando aux3 for verdadeiro, os outros dois aux serão falsos e o terceiro parâmetro do loop ficará vermelho.

Figura 9: demonstração de algumas linhas do código da aplicação



```
63              if(!fim){
64                  aux1 = false;
65                  aux2 = true;
66              }
67                  g2.drawString("'i' é menor do que 10 ?", 280, 150);
68                  if(cont > 1 && i < 10){
69                      g2.drawString("Sim... tudo que estiver dentro das chaves é executado", 120, 190);
70                      cor = true;
71                  }
72                  if(cont > 1 && i >= 10){
73                      g2.drawString("Não... o loop deixa de ser executado.", 200, 190);
74                      fim = true;
```

Fonte: projeto desenvolvido pelo próprio autor.

A variável “fim” fica sendo verdadeiro quando i for maior ou igual a 10.

Quando i for igual ou maior que 10, a variável fim passar a ser true, evitando assim que os comandos sejam executados novamente.

3.2.2.4 Janela de Vetor

A janela de vetor possui uma variável de controle, a variável “i”. A medida que “i” vai avançando, o software também avança. Nessa janela, um retângulo é jogado na tela para destacar alguma posição do vetor, por exemplo:

Figura 10: demonstração de algumas linhas do código da aplicação

```
234            if(i >= 29){
235                switch(j){
236                    case 0:
237                        xRect = 260;
238                        j++;
239                    break;
240                    case 1:
241                        xRect = 320;
242                        j++;
243                    break;
244                    case 2:
245                        xRect = 380;
246                        j++;
```

Fonte: projeto desenvolvido pelo próprio autor.

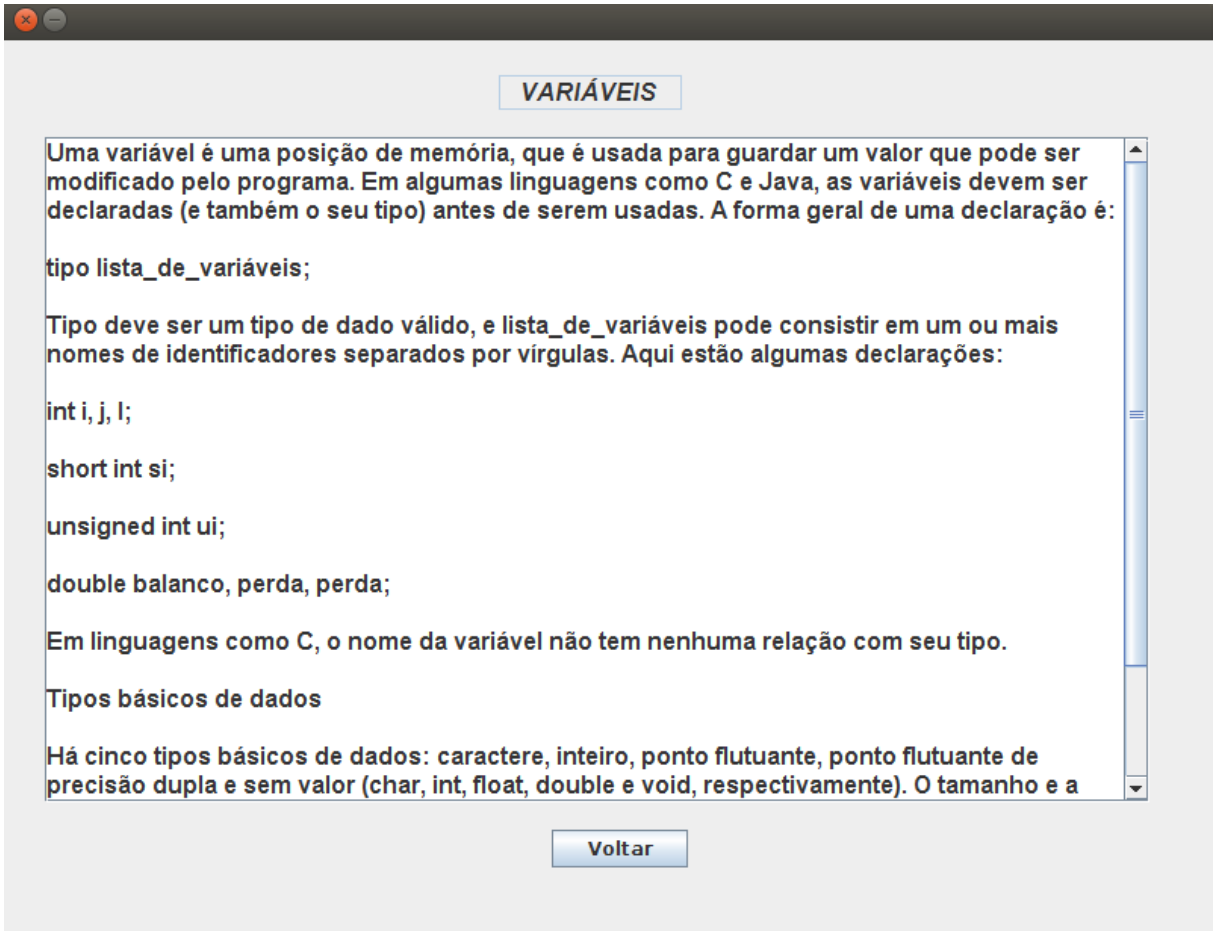
Quando i tem o valor 29 ou mais, o switch passa a ser executado. A cada vez que o switch cai em um caso, o j é incrementado, para na próxima vez ele cair no próximo case, e xRect (variável que define a posição horizontal do retângulo que aparece nessa janela) recebe um novo valor. A variável i é incrementada quando o método ActionPerformed (que executa os eventos que ocorrem) chega a última linha.

4. APRESENTAÇÃO DO SOFTWARE

O software contém 4 ícones básicos na sua tela principal, cada um ensinando conceitos básicos de programação e estão ordenados de modo que será melhor aproveitado (caso o aluno ainda esteja aprendendo) se usados de cima para baixo, pois os dois primeiros conceitos precisam ser entendidos para que facilite o entendimento dos dois últimos conceitos.

A primeira tela mostra o menu com 4 conceitos iniciais de programação, são eles: variáveis, condição, loop e vetores. Figura 1 (página 23). Cada um desses menus contém submenus, no qual abordam: conceitos, como usar, exercícios e respostas sobre o conceito selecionado.

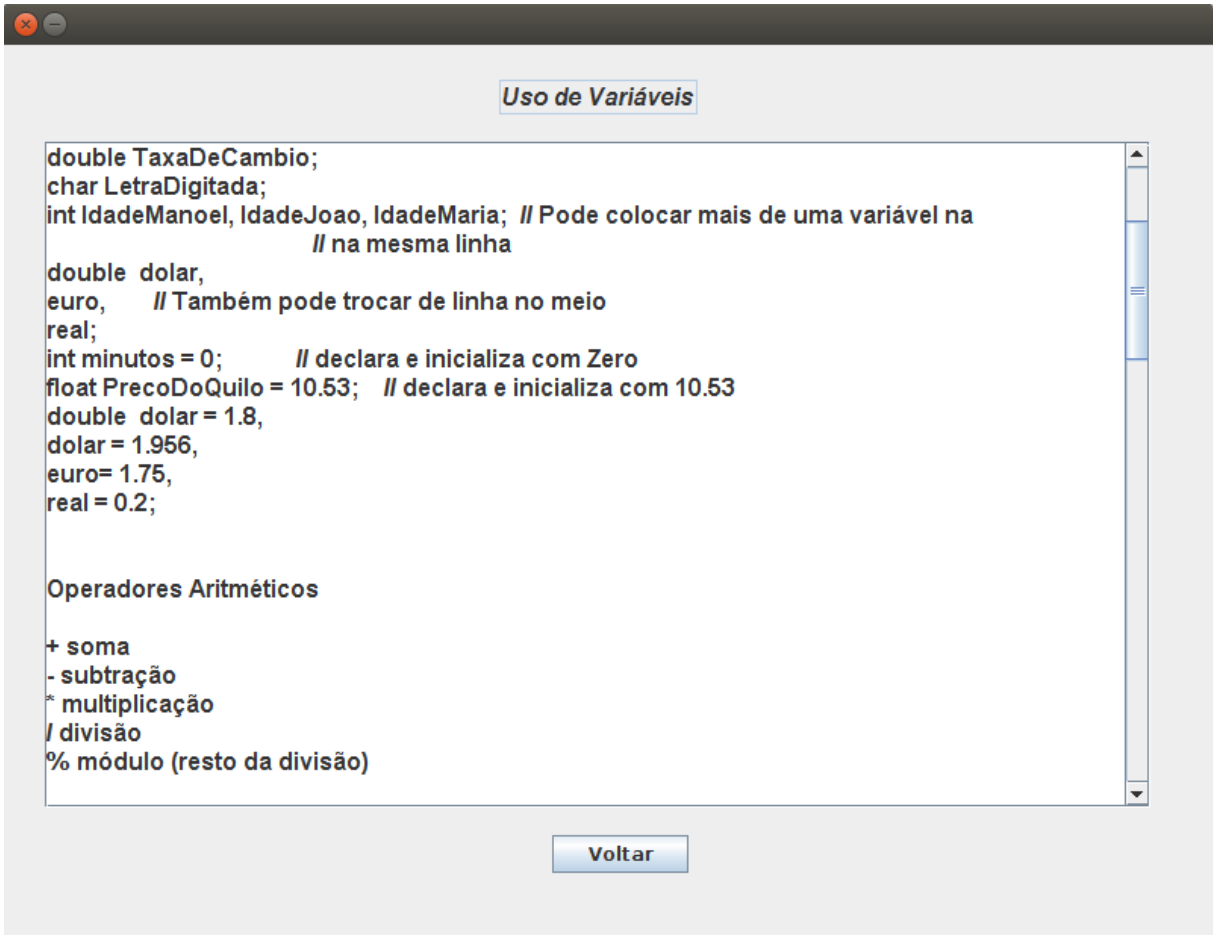
Figura 11: tela conceitual de variáveis



Fonte: projeto desenvolvido pelo próprio autor.

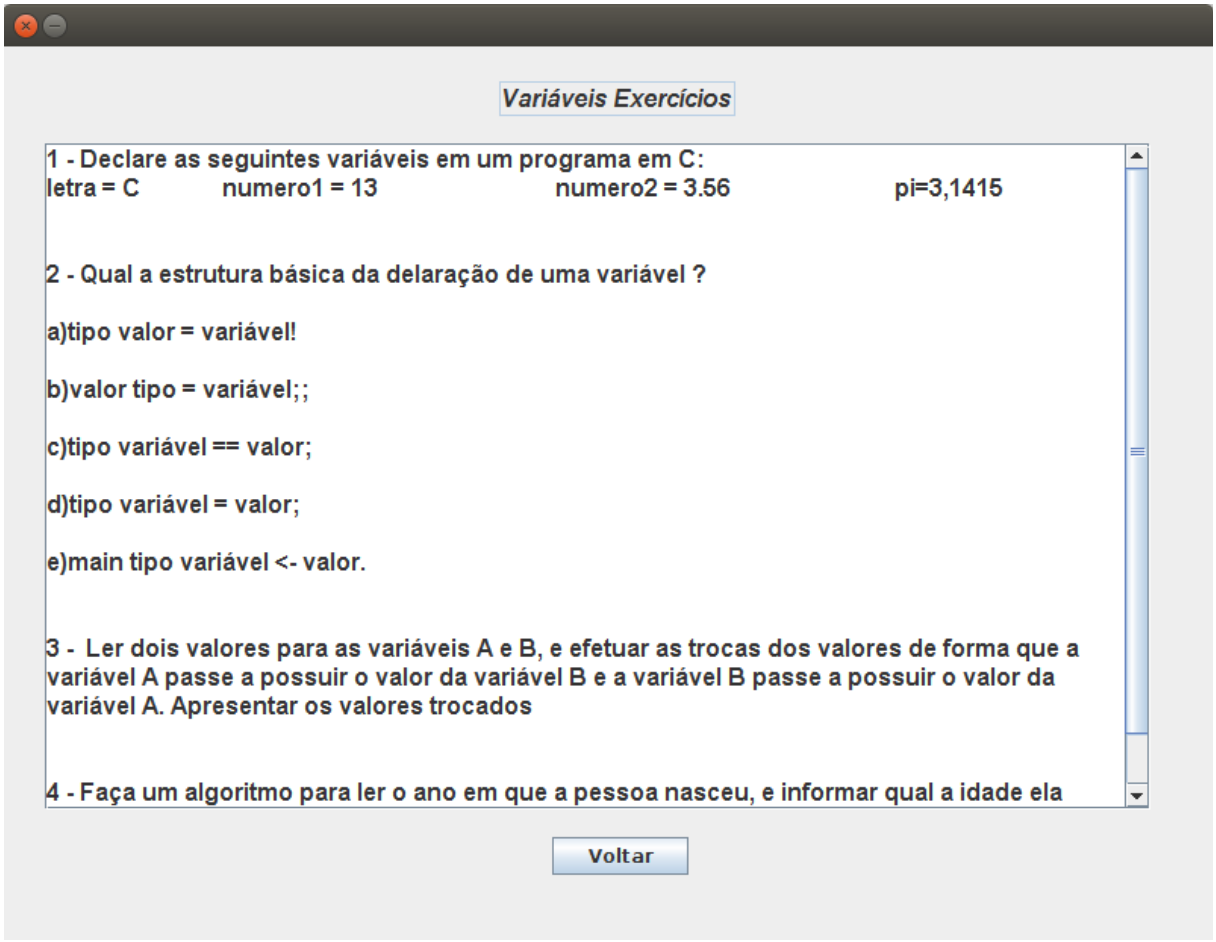
O primeiro ícone do menu “Principais Conceitos” se chama Variáveis. Nele é explicado o que são variáveis, como declará-las e como usá-las. As informações contidas nesta tela foram retiradas do livro C completo e total (Deitel, 2003).

Figura 12: tela de “Como usar” do conceito variáveis



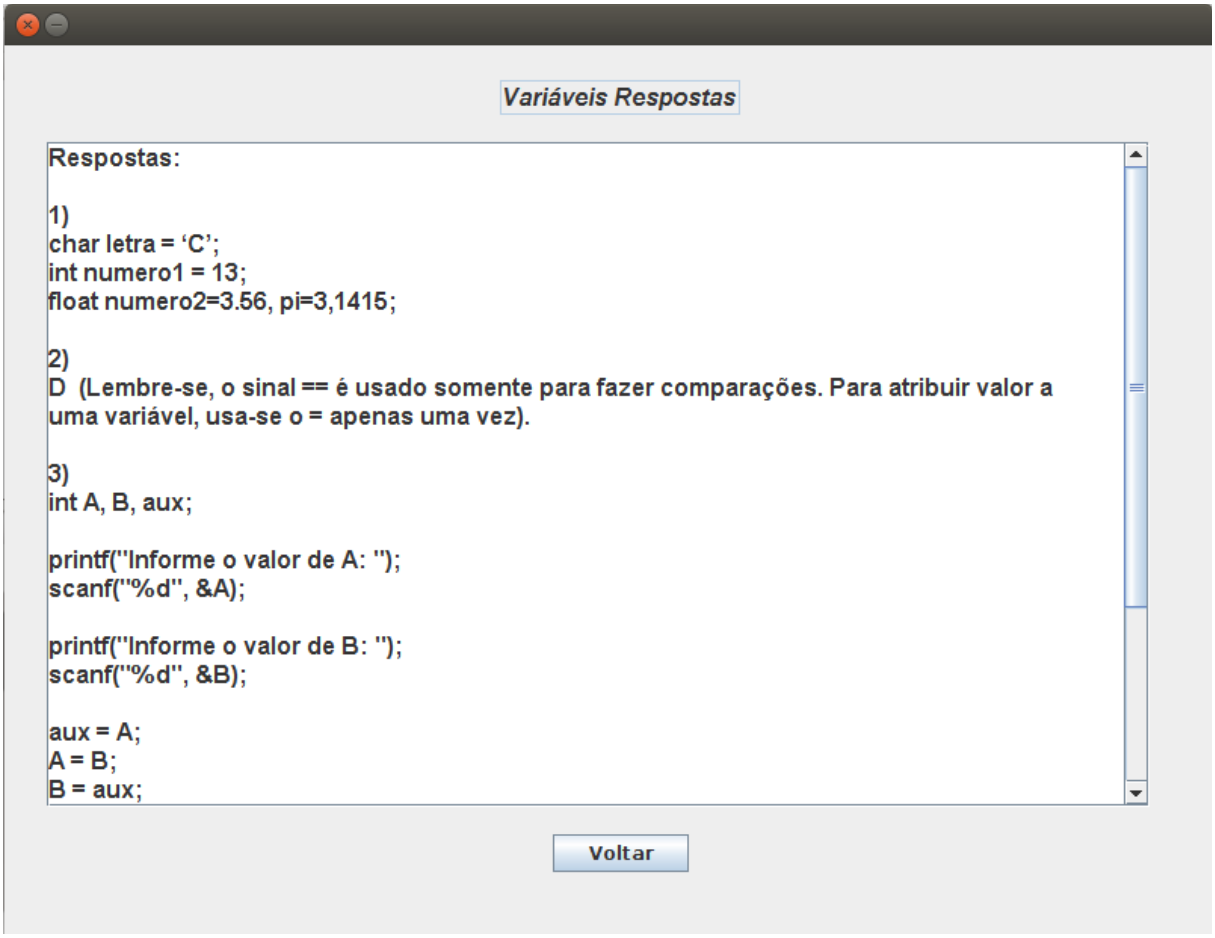
Fonte: projeto desenvolvido pelo próprio autor.

Figura 13: tela de “Exercícios” do conceito variáveis



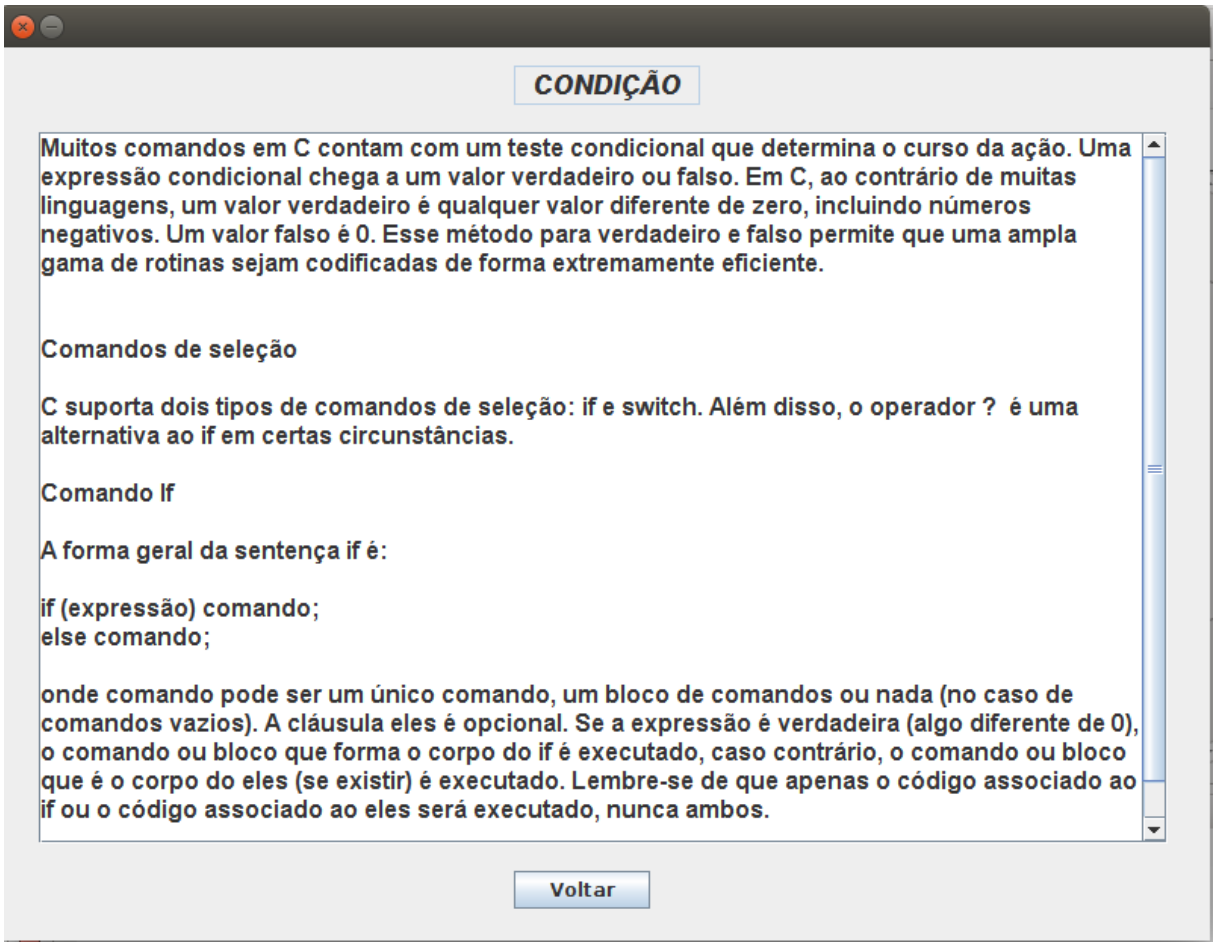
Fonte: projeto desenvolvido pelo próprio autor.

Figura 14: tela de “Respostas” do conceito variáveis



Fonte: projeto desenvolvido pelo próprio autor.

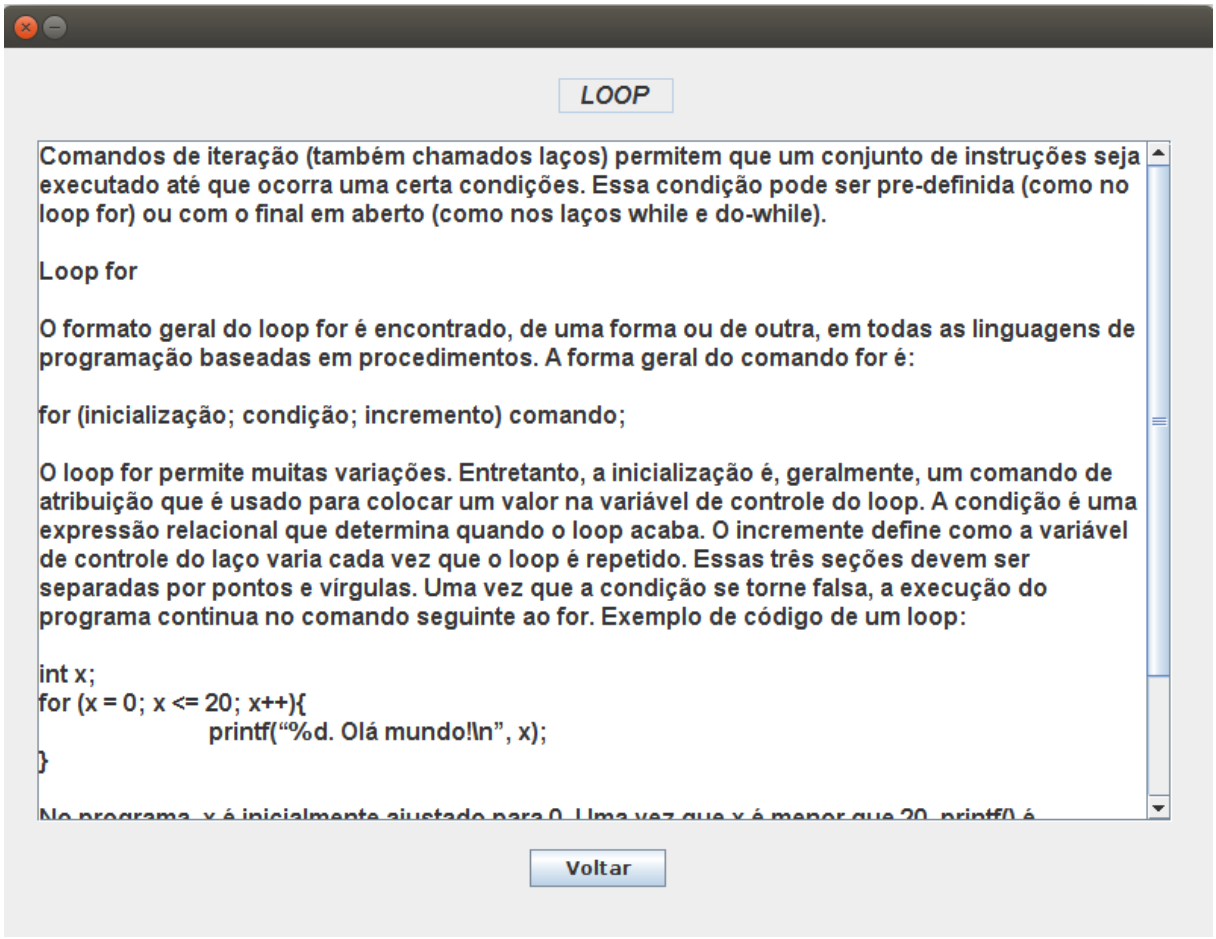
Figura 15: tela conceitual de condição



Fonte: projeto desenvolvido pelo próprio autor.

O segundo ícone do menu “Principais Conceitos” se chama Comando Condicional. Nele é explicado o que é uma condição, como declará-la e quais os seus parâmetros. As informações contidas nesta tela foram retiradas do livro C completo e total (Deitel, 2003).

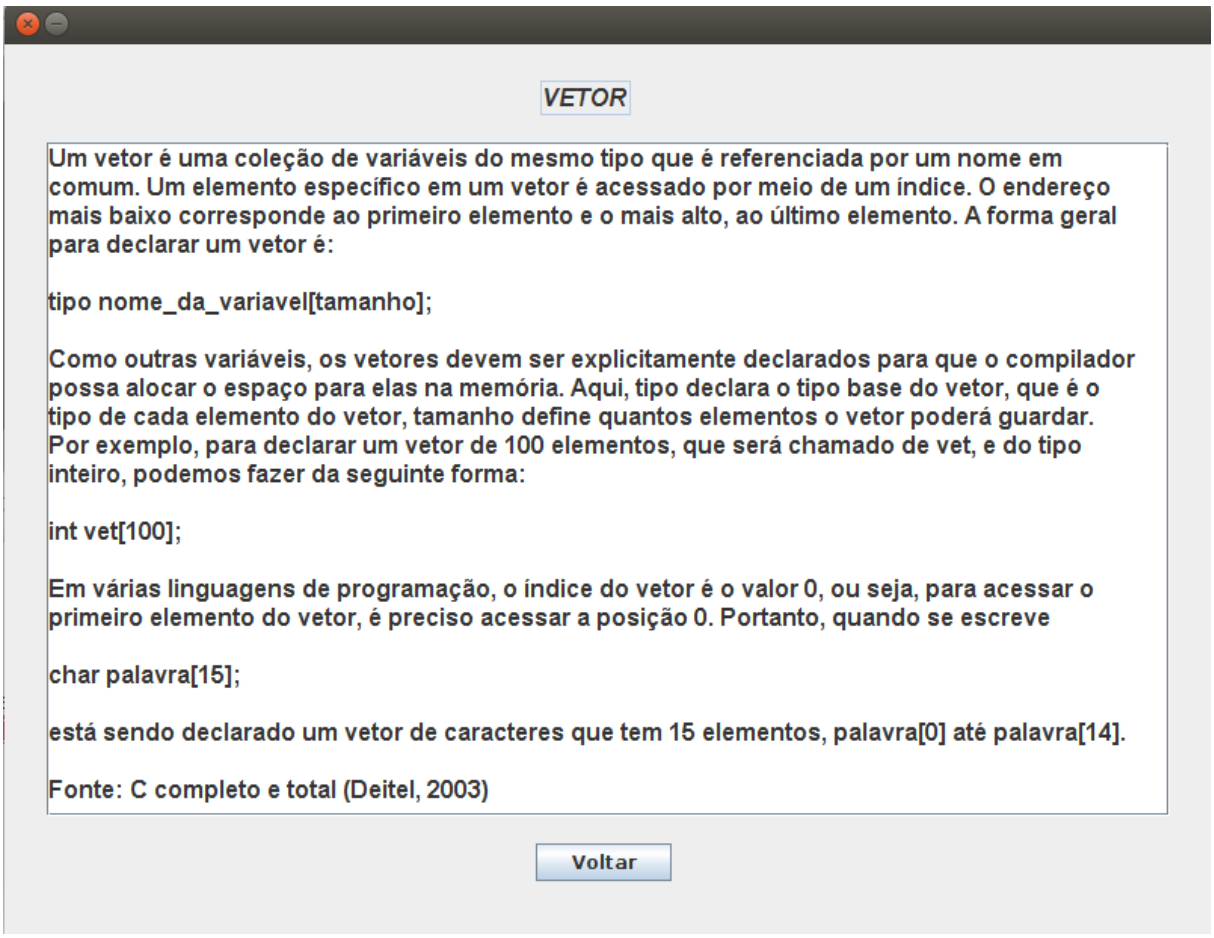
Figura 16: tela conceitual de loop



Fonte: projeto desenvolvido pelo próprio autor.

O terceiro ícone do menu “Principais Conceitos" se chama Comando de Repetição. Nele é explicado o que é um loop, como declará-lo e quais os seus parâmetros. As informações contidas nesta tela foram retiradas do livro C completo e total (Deitel, 2003).

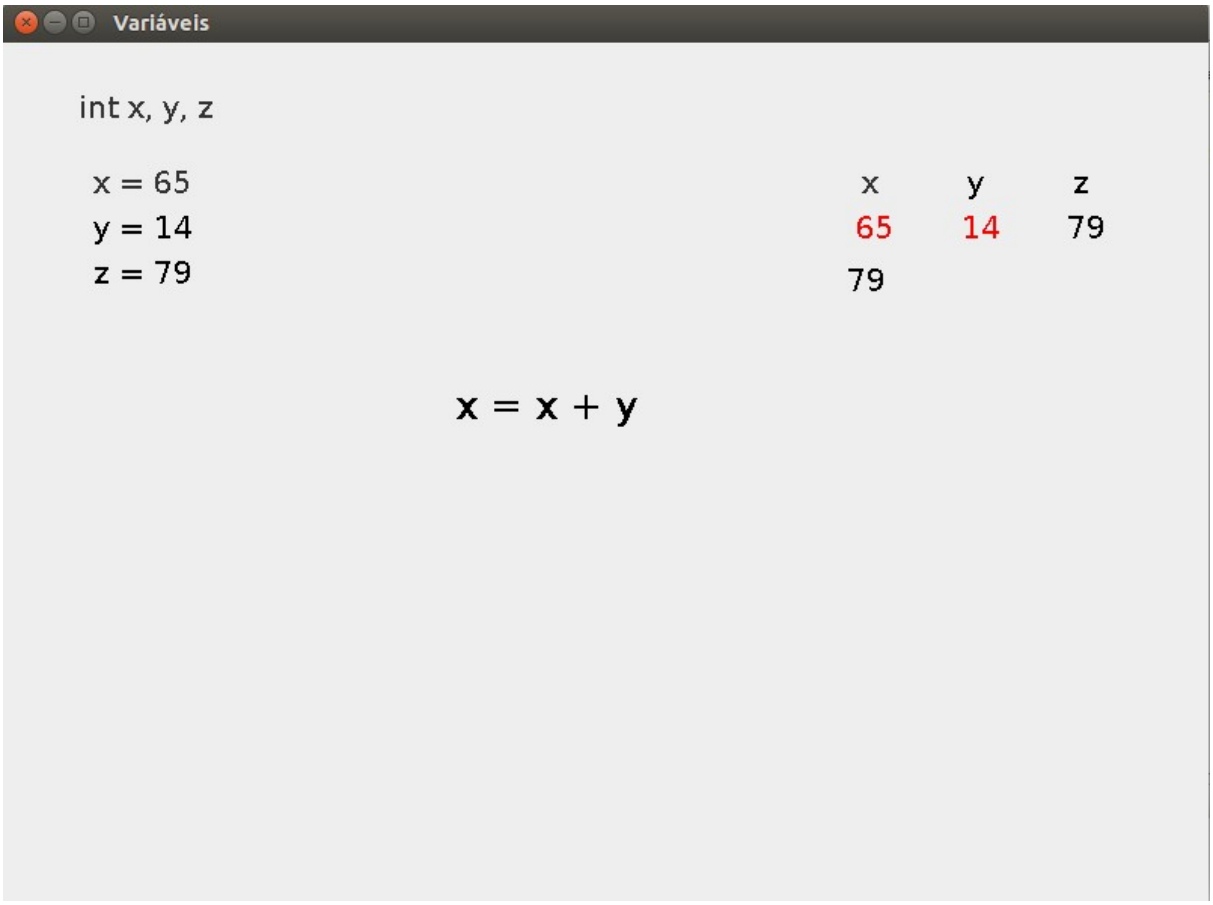
Figura 17: tela de demonstração de vetor



Fonte: projeto desenvolvido pelo próprio autor.

O quarto e último ícone do menu “Principais Conceitos” se chama Vetores. Nele é explicado o que é um vetor, como declará-lo e como acessar as posições do vetor.. As informações contidas nesta tela foram retiradas do livro C completo e total (Deitel, 2003).

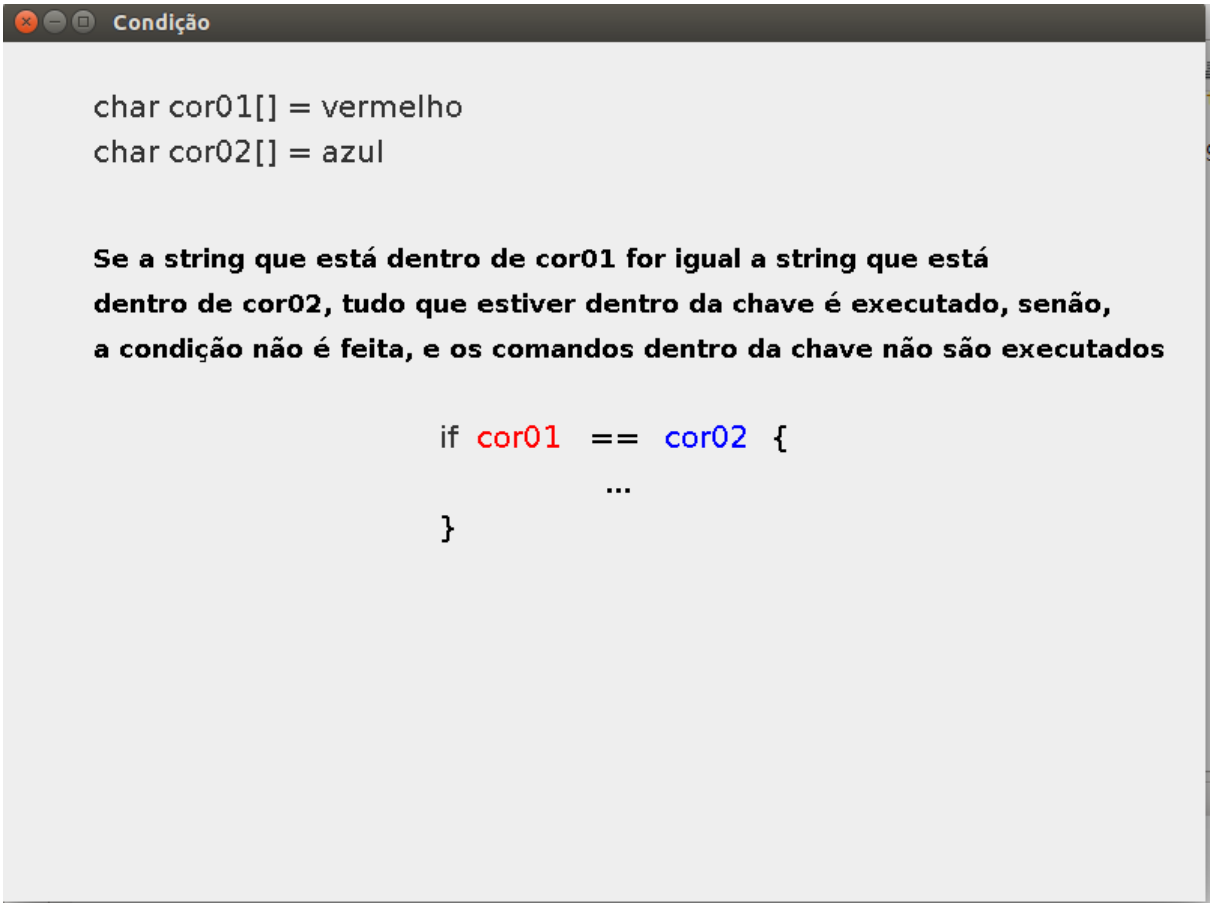
Figura 18: tela de demonstração de variáveis



Fonte: projeto desenvolvido pelo próprio autor.

O primeiro ícone do menu Operações se chama Atribuição de Variáveis. Ao clicar no ícone, uma nova janela é aberta, e nela é mostrada algumas variáveis recebendo valores, e fazendo cálculos aleatórios, podendo esses cálculos ser adição, subtração ou multiplicação. Divisão não aparece nos cálculos.

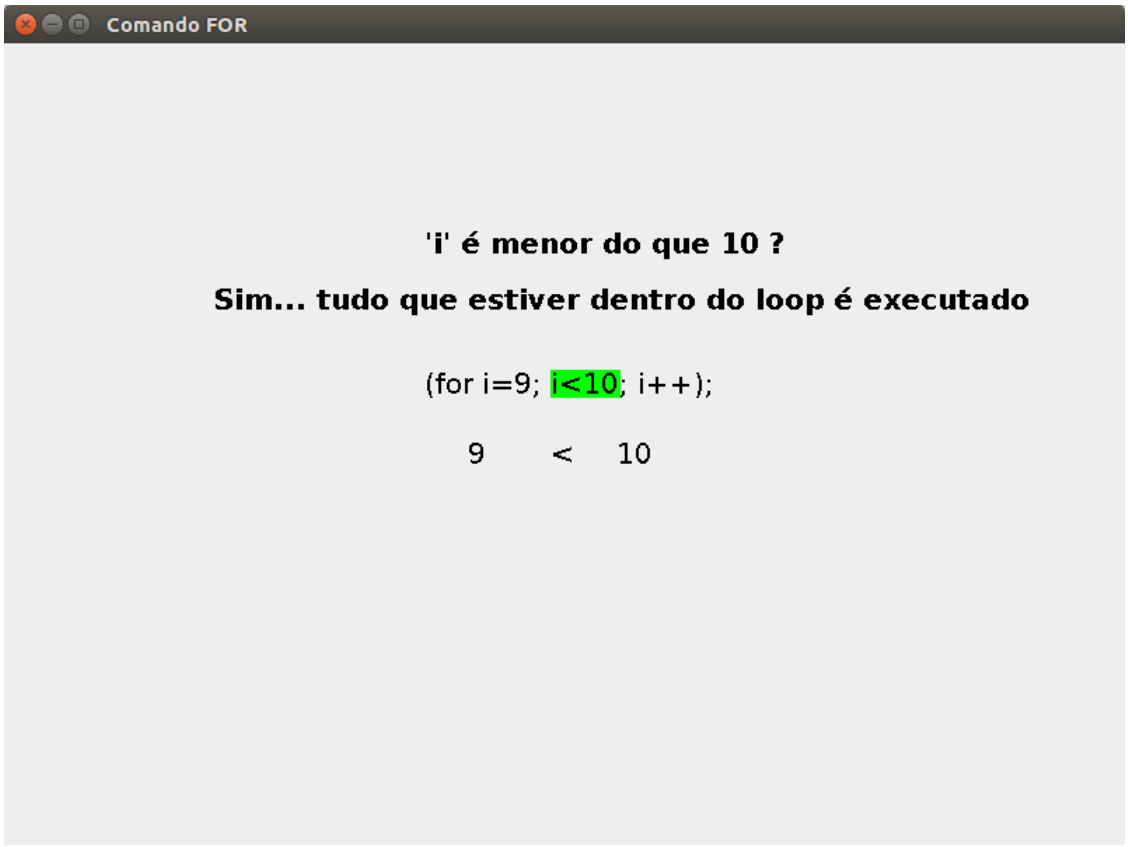
Figura 19: tela de demonstração de condição



Fonte: projeto desenvolvido pelo próprio autor.

No segundo ícone do menu “Operações”, chamado Uso Condicional, é ensinado como acontece a condição (comparação) entre duas strings primeiramente, e após é mostrado a comparação entre dois valores .

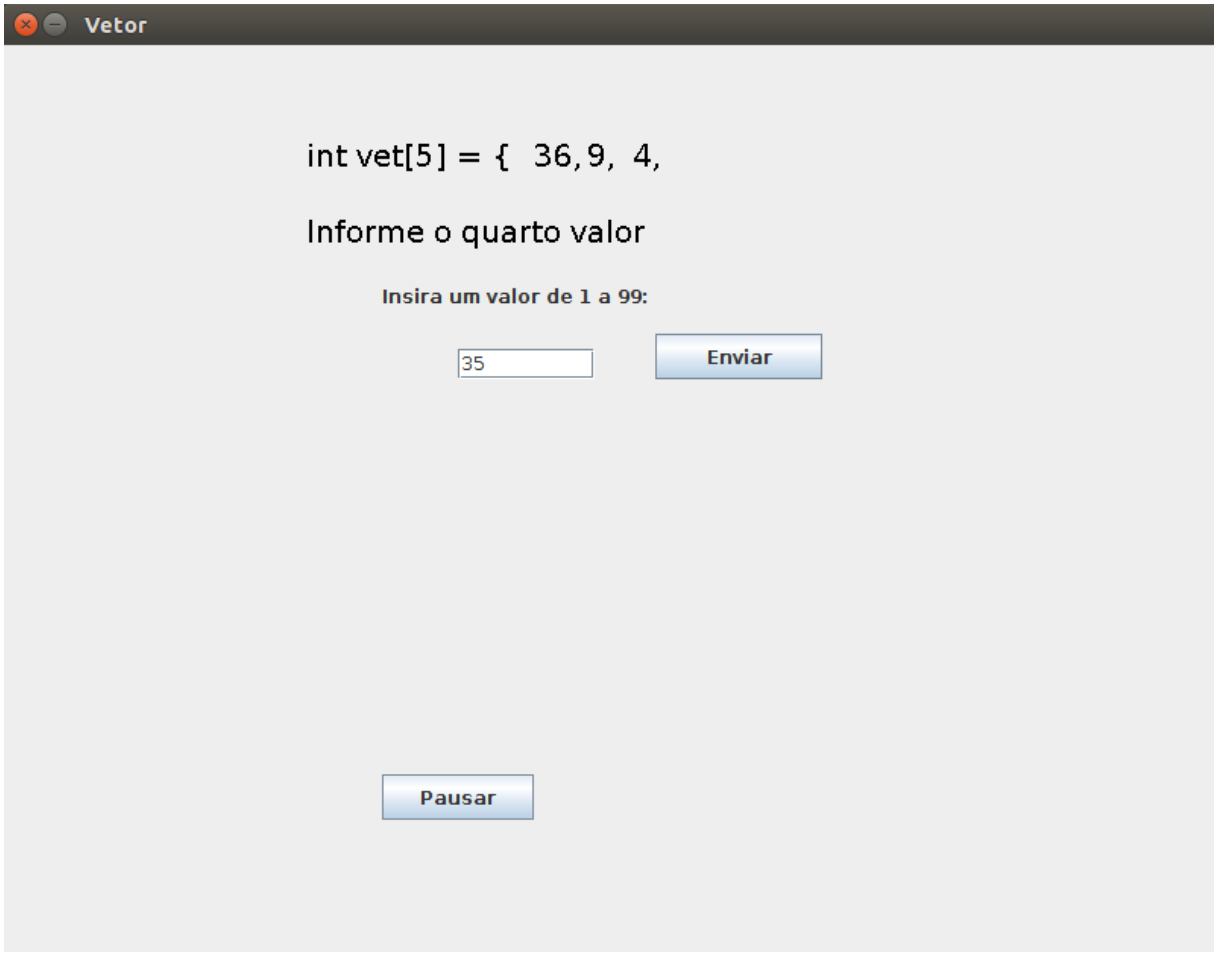
Figura 20: tela de demonstração de condição



Fonte: projeto desenvolvido pelo próprio autor.

No terceiro ícone do menu Operações, chamado Comando de Repetição, é ensinado como acontece um loop. Essa funcionalidade inicia com uma variável “i” recebendo o valor 0, e um outro valor, que recebeu o valor “10”. O “i” movimenta até bem próximo do valor “10”, e compara se o valor que está em “i” é menor do que 10. Se for, o valor que está dentro de “i” se movimenta até sumir, e aquele valor é incrementado em + 1. Esse procedimento se repete até que o valor de “i” não mais seja menor do que 10.

Figura 21: tela de demonstração de condição



Fonte: projeto desenvolvido pelo próprio autor.

No quarto e último ícone, chamado Vetores, é mostrado como são acessados os valores que ficam dentro de um vetor. Primeiramente são gerados 3 números aleatórios quando a opção “Vetores” é escolhida, depois esses números são jogados nas 5 posições do vetor. O retângulo quando fica verde, corresponde a posição do vetor que está sendo acessada.

5 ANÁLISE E RESULTADOS

5.1 ANÁLISE DAS OPERAÇÕES

A ferramenta apresenta os conceitos iniciais de programação, dividido em duas partes principais: os conceito e como são utilizados na prática.

No menu prático sobre variáveis, é demonstrado como elas são declaradas, e como fazer operações lógicas com o valor dela. O usuário insere os valores a sua escolha, dando uma interatividade com a ferramenta, e ajudando com que ele consiga entender melhor a partir do seu próprio raciocínio.

O comando de condição é demonstrado como comparar o conteúdo das variáveis, e o que acontece quando a comparação é verdadeira ou falsa. Nesse menu, o usuário também entra com os valores, podendo assim ter uma melhor compreensão e interatividade com a ferramenta.

O comando de repetição é demonstrado como ocorre a comparação, e em que momento são aplicados os parâmetros do loop for.Para o vetor, é demonstrado como acessar as posições de um vetor. O vetor tem 5 posições, no qual o usuário informa o valor para cada uma delas.

Os itens do menu conceitual, apresentam explicações e exemplos sobre cada conceito, explicando cada um dos conceitos abordados, como declará-los, usá-los, como passar valores, e há também uma lista de exercícios e respostas para cada conceito abordado.

5.2 APLICAÇÃO DE QUESTÕES

A ferramenta foi apresentada aos alunos do 2° período de Sistemas de Informação da faculdade Doctum de Teófilo Otoni, durante o horário de aula no laboratório da própria instituição. Durante o tempo de teste por parte dos alunos, que durou aproximadamente 30 minutos, eles utilizaram as funcionalidades da

ferramenta desenvolvida, e depois responderam a um questionário, que continha as seguintes perguntas:

1) Grau de Satisfação em relação ao uso da ferramenta
( ) 1 – Péssimo 2 - ( ) Ruim 3 - ( ) Regular 4 - ( ) Bom 5 - ( ) Excelente

2) Em relação a sua expectativa ao que o sistema propôs fazer
( ) 1 – Péssimo 2 - ( ) Ruim 3 - ( ) Regular 4 - ( ) Bom 5 - ( ) Excelente

3) Os layouts das telas foram úteis ?
( ) 1 – Péssimo 2 - ( ) Ruim 3 - ( ) Regular 4 - ( ) Bom 5 - ( ) Excelente

4) Quantidade de informação mostrada na tela foi suficiente ?
( ) 1 – Péssimo 2 - ( ) Ruim 3 - ( ) Regular 4 - ( ) Bom 5 - ( ) Excelente

5) As informações foram bem organizadas ?
( ) 1 – Péssimo 2 - ( ) Ruim 3 - ( ) Regular 4 - ( ) Bom 5 - ( ) Excelente

6) A sequencia das Telas facilitam o uso do sistema ?
( ) 1 – Péssimo 2 - ( ) Ruim 3 - ( ) Regular 4 - ( ) Bom 5 - ( ) Excelente

7) As instruções para comandos no software são adequadas ?
( ) 1 – Péssimo 2 - ( ) Ruim 3 - ( ) Regular 4 - ( ) Bom 5 - ( ) Excelente

8) O software manteve você informado sobre o que ele estava fazendo ?
( ) 1 – Péssimo 2 - ( ) Ruim 3 - ( ) Regular 4 - ( ) Bom 5 - ( ) Excelente

9) Houve aprendizado para operar o sistema ?
( ) 1 – Péssimo 2 - ( ) Ruim 3 - ( ) Regular 4 - ( ) Bom 5 - ( ) Excelente

10) A ferramenta apresentada auxilia aprender os conceitos demonstrados ?
( ) 1 – Péssimo 2 - ( ) Ruim 3 - ( ) Regular 4 - ( ) Bom 5 - ( ) Excelente

11) Os conceitos apresentados pela ferramenta foram claros e de fácil entendimento ?
( ) 1 – Péssimo 2 - ( ) Ruim 3 - ( ) Regular 4 - ( ) Bom 5 - ( ) Excelente

12) Quanto você acha que o software poderá ser útil para os próximos alunos que entrarem no curso de Sistemas de Informação ?
( ) 1 – Péssimo 2 - ( ) Ruim 3 - ( ) Regular 4 - ( ) Bom 5 - ( ) Excelente

13) Você considera que as informações demonstradas no software são suficientes para que um aluno consiga começar a programar ?

( ) 1 – Péssimo 2 - ( ) Ruim 3 - ( ) Regular 4 - ( ) Bom 5 - ( ) Excelente

14) Quanto você avalia as informações demonstradas na parte conceitual ?
( ) 1 – Péssimo 2 - ( ) Ruim 3 - ( ) Regular 4 - ( ) Bom 5 - ( ) Excelente

15) Quanto você avalia as informações demonstradas na parte teórica ?
( ) 1 – Péssimo 2 - ( ) Ruim 3 - ( ) Regular 4 - ( ) Bom 5 - ( ) Excelente

16) No geral, que nota você daria para o software ?
1-( ) 2-( ) 3-( ) 4-( ) 5-( ) 6-( ) 7-( ) 8-( ) 9-( ) 10-( )

Uma quantidade de 9 alunos responderam as dezesseis questões do questionário.

RESULTADO DO QUESTIONÁRIO USABILIDADE					
QUESTÃO	EXCELENTE	BOM	REGULAR	RUIM	MUITO RUIM
1	11,11%	77,78%	11,11%	0,00%	0,00%
2	11,11%	88,89%	0,00%	0,00%	0,00%
3	22,22%	22,22%	55,56%	0,00%	0,00%
4	66,67%	22,22%	11,11%	0,00%	0,00%
5	22,22%	66,67%	11,11%	0,00%	0,00%
6	33,33%	55,56%	11,11%	0,00%	0,00%
7	22,22%	77,78%	0,00%	0,00%	0,00%
8	0,00%	77,78%	22,22%	0,00%	0,00%
9	0,00%	77,78%	22,22%	0,00%	0,00%
10	55,56%	44,44%	0,00%	0,00%	0,00%
11	11,11%	88,89%	0,00%	0,00%	0,00%
12	11,11%	77,78%	11,11%	0,00%	0,00%
13	22,22%	33,33%	44,44%	0,00%	0,00%
14	22,22%	66,67%	11,11%	0,00%	0,00%
15	11,11%	66,67%	22,22%	0,00%	0,00%

A análise obteve um bom resultado no geral do questionário respondido pelos alunos do 2º período de S.I.. A maioria das questões obtiveram “Bom” acima da média. A questão 3 que obteve “Regular” com 55,56% das respostas, em que pergunta se “os layouts foram úteis”, e na questão 13, “Regular” obteve 44,44% das respostas, ficando acima de excelente, com 22,22% e bom com 33,33%.

As questões 2 e 11, obtiveram o maior percentual de respostas marcadas como “Bom”, com 88,89% cada. A questão 2 pergunta se “a ferramenta atendeu as expectativas”, e a questão 11, pergunta se “os conceitos apresentados foram claros

e de fácil entendimento”, sendo essa ultima uma das perguntas mais importantes para ajudar a validar as hipóteses levantadas no início desse trabalho. Na questão 16, foram obtidas 4 respostas no item 9, 4 respostas no item 8, e 1 respostas no item 7, dando uma média de 8,3 pontos para a ferramenta, e fazendo com que ela atingindo um nível satisfatório no geral entre os alunos que a analisaram.

A engenharia de usabilidade foi utilizada para demonstrar a ferramenta desenvolvida aos alunos, pois, estes foram os representantes dos usuários que serão o alvo no qual este trabalho foi desenvolvido para auxiliar.

CONCLUSÃO

Esse projeto veio da observação ~~ideia~~, que o autor deste trabalho percebeu nas disciplinas que utilizam programação em suas grades. Assim procurou-se tentar construir algo que ajudasse os alunos que possuem dificuldades em programação. Espera-se que com esse software, os alunos consigam aprender de forma mais rápida os conceitos básicos de programação, e alavancar os seus estudos.

H0**:** a ferramenta irá acabar com a dificuldade de todos os alunos na disciplina de programação, pois trabalhará melhor a capacidade de cada aluno. INVALIDADA

A hipótese H0 não pôde ser validada, pois, mesmo que a ferramenta consiga ajudar uma boa parte dos alunos que a utilizarem, dificilmente irá conseguir ajudar a todos.

H1: o desenvolvimento da ferramenta servirá como material de apoio na construção do conhecimento de programação básica. VALIDADA.

A hipótese H1 foi validada pelas respostas obtidas no questionário. Para essa hipótese, foram avaliadas todas as respostas, porém, as questões 4 e 12 tiveram grande peso, obtendo 66,67% excelente a questão 4, e 77,78% bom a questão 12. Essas duas questões questionam os usuários sobre o quanto ela é útil para ajudar a construir conhecimento. VALIDADA.

H2: seria viável a elaboração da ferramenta, uma vez que poderá ser usado em qualquer instituição de ensino superior. VALIDADA.

A H2 se torna válida, pois, por ser um projeto sem fins lucrativos, pode ser distribuído e utilizado por qualquer pessoa e qualquer instituição que tenha interesse. VALIDADA.

H3: o desenvolvimento da ferramenta se torna viável, pois demonstrará a execução de algoritmos de forma amigável e de fácil entendimento. VALIDADA.

A hipótese H3 foi validada. Todas as questões foram analisadas para validar essa hipótese, porém, as questão 6, 7 e 11 tiveram grande peso, pois, nelas, são questionadas sobre a forma como a ferramenta apresenta as informações, e se o

usuário achou que estão claras e fáceis de entender. A questão 6 obteve 33,33% excelente, 55,56% bom e 11,11% regular. A questão 7 obteve 22,22% excelente e 77,78% bom. A questão 11 recebeu 11,11% excelente e 88,89% bom.

O ganho pessoal com esse projeto foi a oportunidade de estudar e aprender mais de uma das linguagens de programação mais utilizadas atualmente no mercado, e deixar um pouco do conhecimento adquirido nesses anos de estudo para auxiliar os próximos alunos que entrarem no curso de Sistemas de Informação.

A ferramenta obteve respostas e uma nota bem satisfatória para o autor da pesquisa e desenvolvedor da ferramenta. Agora, espera-se que nos próximos anos os próximos alunos que estudarem programação utilizem a ferramenta, e que consigam ter muitos bons resultados em programação.

Essa ferramenta pode se tornar ainda mais útil, e muito mais rico em termos de conteúdo. Para isso, deixo abaixo algumas dicas e sugestões para trabalhos futuros:

- Avançar a ferramenta para conceitos de ponteiros, listas, filas, etc..
- Desenvolver um jogo para esta finalidade baseado nas ementas.

REFERÊNCIAS

ASCENCIO, Ana F.; CAMPOS, Edilene A.. *Fundamentos da Programação de Computadores*. São Paulo: Prentice Hall, 2002.

BARROS, A.; LEHFELD, N.. Fundamentos de metodologia: Um guia para a iniciação científica. 2ª ed. São Paulo: Makron Books, 2000.

BONA, Berenice de Oliveira. *Análise de Software educativos para o ensino de Matemática nos anos iniciais do Ensino Fundamental*. Universidade Luterana do Brasil. Carazinho, RS. Disponível em <http://www.if.ufrgs.br/eenci/artigos/Artigo_ID71/v4_n1_a2009.pdf>. Acesso em 23 de Nov. de 2016.

CLARO, Daniela B.; SOBRAL, João B.. *Programação em Java*. Florianópolis: Pearson Education, 2008.

CHAGAS, Clayton et al. *Java Básico e Orientação a Objeto*. Rio de Janeiro: Fundação CECIERJ, 2010.

Dall'Oglio, Pablo. *PHP Programando com Orientação a Objetos 3ª ed.*. São Paulo: Novatec Editora Ltda, 2015.

DEITEL, Harvey M. *Java Como Programar*. Porto Alegre: Bookman, 2003.

DONATELLI, Luiza. ***O mundo virtual*.** Veja,São Paulo, Editora Abril, 2016.

EVARISTO, Jaime. *Aprendendo a Programar Programando na Linguagem C Para Iniciantes*. 3ª ed. Rio de Janeiro: Book Express, 2006.

FONSECA FILHO, Cléuzio. *História da Computação O caminho do pensamento e da tecnologia*. Porto Alegre: EDIPUCRS, 2007.

LINDER, Marcelo. *Linguagens de Programação*. Disponível em <http://www.univasf.edu.br/~marcelo.linder/arquivos_pc/aulas/aula5.pdf>. Acesso em 09 de Jun. de 2016.

MERCADO, Luís Paulo. *Novas tecnologias na educação: reflexão sobre a prática*. Maceió: EDUFAL, 2002.

OLIVEIRA, Celina et al. *Ambientes informatizados de aprendizagem produção e avaliação de software educativo.* Campinas: Papirus, 2001.

PÁDUA, Clarindo. *Engenharia de Usabilidade*. Belo Horizonte: UFMG. 2012.

PRESSMAN, Roger S. Engenharia de software: uma abordagem profissional. 7 ª ed. Porto Alegre: AMGH, 2011.

SALEN, Zimmerman; KATTIE, Eric. *Rules of Play Game Design Fundamentals*. London: The MIT Press, 2012.

SALVETTI, Dirceu; BARBOSA, Lisbete. *Algoritmos*. São Paulo: Makron Books, 1998.

SANTOMAURO, Beatriz. *Jogos: quando, como e por que usar*. Diponível em <http://revistaescola.abril.com.br/formacao/jogos-quando-como-usar-741268.shtml#ad-image-0> Acesso em 19 de Mai. de 2016.

SCHILDT, Herbert. *C Completo e Total*. 3ª ed. São Paulo: MAKRON Books Ltda, 1996.

SEBESTA, Robert W. *Conceitos de Linguagem de Programação.* 5ª ed. São Paulo: ARTMED, 2003.

SOFFA, Marílice; ALCÂNTARA, Paulo. O uso do software educativo: reflexões da

prática docente na sala informatizada. Disponível em <http://www.pucpr.br/eventos/educere/educere2008/anais/pdf/335_357.pdf>. Acesso em 29 de Ago. 2016.

SZWARCFITER, Markenzon; JAYME, Lilian. *Estruturas de Dados e Seus Algoritmos*. 3ª ed. Rio de Janeiro: Livros Técnicos e Científicos Editora Ltda, 2010.

TIEGHI, Ana Luiza. *Tecnologia pode ser aliada da saúde*. Disponível em: <http://www.usp.br/espacoaberto/?materia=tecnologia-pode-ser-aliada-da-saude> Acesso em 09 de Jun. de 2016.

Class Graphics2D disponível em <https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics2D.html>. Acesso em 30 de Nov. de 2016.