UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CUROS DE CIÊNCIAS DA COMPUTAÇÃO BACHARELADO

PROTÓTIPO DE UM SISTEMA DE GERAÇÃO E ANIMAÇÃO DE FLUXOGRAMAS

GILBERTO FREITAS

BLUMENAU
2003

2003/2-18

GILBERTO FREITAS

PROTÓTIPO DE UM SISTEMA DE GERAÇÃO E ANIMAÇÃO DE FLUXOGRAMAS

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso do curso de Ciência da Computação — Bacharelado.

Prof. Mauro Marcelo Mattos – Orientador

BLUMENAU
2003

2003/2-18

PROTÓTIPO DE UM SISTEMA DE GERAÇÃO E ANIMAÇÃO DE FLUXOGRAMAS

Por

GILBERTO FREITAS

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso, pela banca examinadora formada por:

Presidente:	Prof. Mauro Marcelo Mattos, Dr. – Orientador, FURB
Membro:	Prof. Luiz Bianchi, FURB
Membro:	Prof. Luis Heinzen, FURB

Blumenau, 24 de Novembro de 2003

Dedico este trabalho ao meu cunhado Claudionor Silveira, que com todo seu carinho e humildade, me apoiou no decorrer do curso.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

Aos meus pais que sempre me deram todo apoio para que eu continuasse e não parasse de estudar.

A minha amada esposa Simone e a minha filha Carolina que, com paciência, entenderam a importância da conclusão deste curso.

As minhas duas irmãs Janete e Jaqueline.

A meu chefe Antonio Rodrigues de Oliveira que através da empresa A. Angeloni & Cia Ltda cedeu alguns recursos que me ajudaram na conclusão do curso.

Ao meu orientador, Mauro Marcelo Mattos, por ter me orientado e acreditado na conclusão deste trabalho.

RESUMO

O presente trabalho de conclusão de curso está inserido no contexto do projeto Tutor de algoritmos que vem sendo desenvolvido na FURB desde 1999 o qual se caracteriza como uma ferramenta para o apoio ao ensino de lógica de programação. Neste trabalho é descrita a versão atual do sistema, e apresentada a especificação e implementação do módulo de geração de fluxograma e execução passo-a-passo do algoritmo produzido pela versão anterior da ferramenta. O funcionamento da ferramenta é caracterizado através de um estudo de caso.

Palavras chaves: Ferramenta de Ensino; Fluxograma; Teste de mesa.

Excluído:

ABSTRACT

This work is inserted in the context of the project Tutor de Algoritmos that is being developed at FURB since 1999. The project is characterized as a tool for support the programming logic teaching. In this work the last version of the system is described and presented the specification and implementation of the flowcharting generation and step-by-step testing modules. A case study is used to demonstrate the functioning of the new version.

Key-words: Teaching Tool; Flowchart, Step-by-step testing

LISTA DE ILUSTRAÇÕES

Excluído: FIGURA 1 – Tela CLIPS contextualização do problema . 7¶
FIGURA 2 – Tela CLIPS perguntas do sistema especialista . 7¶
FIGURA 3 – Tela CLIPS feedback do contexto . 8¶
QUADRO 1 – Descrição do exercício de algoritmos . 10¶
FIGURA 4 – Tela principal do protótipo HelpAlgo . 10¶
FIGURA 5 – Tela 1 do estudo de caso . 11¶
FIGURA 6 – Tela perguntas do estudo de caso . 12¶
FIGURA 7 – Tela algoritmo gerado . 12¶
FIGURA 8 – Tela principal do ambiente do ATMUF . 15¶
FIGURA 9 – Tela exemplo do Editor de Fluxogramas . 16¶
FIGURA 10 – Tela Construtor na execução passo-a-passo . 19¶
FIGURA 11 - Tela narrativa do sistema . 22¶
FIGURA 12 –Apresentação do Diagrama de Casos de Uso . 23¶
QUADRO 2 - Documentação sobre os casos de uso . 24¶
FIGURA 13 – Diagrama de seqüência . 25¶
FIGURA 14 – Fluxograma edição algoritmo . 27¶
FIGURA 15 – Gerar Fluxograma . 29¶
FIGURA 16 – Redesenho na execução passo-a-passo . 32¶
FIGURA 17 – Fluxograma passo-a-passo cálculo das expressões . 34¶
FIGURA 18 – Exemplo do término da execução passo-a-passo . 36¶
QUADRO 3 – Descrição do estudo de caso . 37¶
FIGURA 19 – Demonstração da abertura de um arquivo . 37¶
FIGURA 20 – Algoritmo aberto e pronto para ser executado . 38¶
FIGURA 21 – Desenho da geração do fluxograma . 38¶
FIGURA 22 – Exemplo do andamento de uma execução passo-a-passo . 39¶
FIGURA 23 – Criação da variável ainda não usado no algoritmo . 40¶
FIGURA 24 – Conteúdo das variáveis e saída impressão na tela . 41¶

LISTA DE SIGLAS

ASA – Animação e Simulação de Algoritmos

ATMUF – Ambiente para Teste de Mesa Utilizando Fluxogramas

BCC – Curso de Ciências da Computação – Bacharelado

CLIPS – C Language Integrated Production System

RBC – Raciocínio Baseado em Casos

SE – Sistema Especialista

SENAC – Serviço Nacional de Aprendizagem Comercial

TCC – Trabalho de Conclusão de Curso

SUMÁRIO

Excluído: 1 . INTRODUÇÃO . 1¶
1.1 . OBJETIVOS DO TRABALHO . 2¶
1.2 . ESTRUTURA DO TRABALHO . 3¶
2 . CONTEXTO DO PROJETO . 4¶
2.1 . TRABALHO DE MATTOS . 4¶
2.1.1 . Motivação . 4¶
2.1.2 . Ambiente de desenvolvimento . 5¶
2.1.3 . Desenvolvimento do sistema . 5¶
2.1.4 . Implementação . 6¶
2.1.5 . Considerações . 8¶
2.2 . TRABALHO DESENVOLVIDO POR GUBLER . 9¶
2.2.1 . Análise . 9¶
2.2.2 . Implementação . 9¶
2.2.3 . Funcionamento . 10¶
2.3 . TRABALHO DE HEINZEN . 13¶
2.4 . OUTROS TRABALHOS CORRELATOS . 14¶
2.4.1 . Trabalho de Cares (2002) . 14¶
2.4.1.1 . Funcionamento do sistema . 14¶
2.4.1.2 . Considerações da ferramenta . 15¶
2.4.2 . Trabalho de Souza (1999) . 15¶
2.4.2.1 . Funcionamento da ferramenta . 15¶
2.4.2.2 . Considerações . 16¶
2.4.3 . Software Construtor . 16¶
2.4.3.1 . Livro Construção de algoritmos . 17¶
2.4.3.2 . Apresentação do Software Construtor . 17¶
2.4.3.3 . Execução do Construtor . 18¶
2.4.4 . ASA . 19¶
3 . DESENVOLVIMENTO DO TRABALHO . 21¶
3.1 . REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO . 21¶
3.2 . ESPECIFICAÇÃO . 21¶
3.2.1 . Levantamento de requisitos . 22¶
3.2.2 . Diagrama de Casos de Uso . 23¶
3.2.3 . Diagrama de Seqüência . 25¶
3.3 . IMPLEMENTAÇÃO . 26¶
3.3.1 . Edição e Contagem de "SE(s)" . 26¶
3.3.2 . Geração do desenho do fluxograma e armazenamento das expressões . 29¶
3.3.3 . Execução Passo-a-passo . 31¶
3.3.4 . Operacionalidade da Implementação . 35¶
3.4 . RESULTADOS E DISCUSSÃO . 41¶
4 . CONCLUSÕES . 43¶
4.1 . EXTENSÕES . 43¶
REFERÊNCIAS BIBLIOGRÁFICAS . 45¶
ANEXO 1 – Representação do formato de sintaxe analisad ... [1]

Excluído: 1¶

1 INTRODUÇÃO

De acordo com Forbellone (2000) a lógica é a arte de bem pensar, que é a ciência das formas de pensamento. Visto que a forma mais complexa do pensamento é o raciocínio, a lógica estuda a correção do raciocínio. Pode-se ainda dizer que a lógica tem em vista a ordem da razão, isto dá a entender que a nossa razão pode funcionar desordenadamente. Por isso a lógica estuda e ensina a colocar ordem no pensamento.

"Usar a lógica é um fator a ser considerado por todos, principalmente pelos profissionais de informática (programadores, analistas de sistemas e suporte), pois seu dia-a-dia dentro das organizações é solucionar problemas e atingir os objetivos apresentados por seus usuários com eficiência e eficácia, utilizando recursos computacionais. Saber lidar com problemas de ordem administrativa, de controle, de planejamento e de estratégia requer atenção e boa performance de conhecimento de nosso raciocínio" (MANZANO, 1996), portanto o uso da lógica para aprender algoritmos (lógica de programação) é muito importante para os desenvolvedores de softwares.

De acordo com Venâncio (1997), algoritmo é um conjunto de regras que permite a resolução de um problema ou a execução de um trabalho, através de um número finito de operações.

De acordo com Mattos (1999), os estudantes que iniciam o curso de graduação em informática, normalmente encontram uma primeira dificuldade relacionada com a disciplina de introdução à programação (lógica de programação), cujo principal objetivo é o de introduzir os conceitos básicos de lógica de programação. O autor, analisando o perfil dos alunos que fazem esta disciplina verificou que a maioria deles possuía conhecimentos abstratos de áreas cientificas (matemática, física, biologia), pois são alunos de 2º grau. Quando apresentado à descrição textual dos enunciados dos problemas nesta disciplina introdutória, na maioria dos casos encontravam dificuldades em extrair as informações necessárias para iniciar a solução destes problemas.

Foi constatado por Mattos (1999) que muitos alunos não possuem experiência prática em áreas comerciais e/ou industriais, a partir das quais vários exercícios são elaborados. Outros, apesar de entenderem os problemas propostos, nem sempre conseguem facilmente descrevê los em pequenos passos para os demais colegas.

Excluído: 1¶

Um acompanhamento realizado por Mattos (1999) sobre as turmas de introdução a programação durante o período compreendido entre os semestres 96/1 e 98/2, permitiu a constatação de que os alunos poderiam ser separados claramente em dois grupos: aqueles que haviam entendido o "como fazer" e superado as dificuldades iniciais e aqueles que não conseguiam superá-las, ou seja, não haviam se apropriado do conhecimento. Observou-se também que, quando induzidos a pensar sobre o problema através de perguntas direcionadas, em sua grande maioria, os alunos do segundo grupo conseguiam descrever a solução intuitivamente, ou seja, sem o formalismo necessário à área de computação.

Uma primeira proposta foi apresentada por Mattos (1999) no VII Congresso Ibero-americano de Educação Superior em Computação – CIESC99. Neste trabalho, descreve-se uma proposta de ferramenta didática baseada em sistemas especialistas, a qual tem por objetivo introduzir o conceito de análise de requisitos já nas primeiras fases do ensino de computação. Para implementar está ferramenta utilizou-se como ambiente de desenvolvimento o software *C Language Integrated Production System* (CLIPS).

Como continuação do trabalho desenvolvido por Mattos (1999), o trabalho de Gubler (2002) desenvolveu um protótipo de um sistema especialista com uma interface gráfica para o usuário, utilizando a ferramenta de programação Delphi, bem como a implementação da estrutura de suporte a problemas cuja solução envolva estruturas de repetição, que ainda não haviam sido implementadas.

Como continuação dos trabalhos acima relacionados a essa temática, propõe-se agregar uma interface gráfica através de fluxogramas animados, mostrando passo-a-passo a execução do algoritmo gerado no sistema especialista desenvolvido por Gubler (2002).

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho, portanto, é continuar o trabalho desenvolvido por Mattos (1999) e Gubler (2002), agregando a criação de fluxograma e execução passo-a-passo do algoritmo gerado pela ferramenta de Gubler (2002), facilitando assim o aprendizado do aluno.

Os objetivos específicos do trabalho são:

a) ler o algoritmo gerado na ferramenta HelpAlgo proposto por Gubler (2002);

Formatados: Marcadores e numeração

Excluído: ¶

Excluído: 1¶

b) criar a interface gráfica do fluxograma;

Excluído: ¶

c) executar o algoritmo passo-a-passo permitindo a interação com o usuário.

Excluído: c)

1.2 ESTRUTURA DO TRABALHO

O trabalho está organizado da seguinte forma:

O primeiro capítulo apresenta uma introdução sobre o assunto e objetivos do trabalho, a fim de dar ao leitor as informações necessárias ao entendimento do assunto abordado.

O segundo capítulo apresenta alguns trabalhos que fizeram parte da fundamentação teórica do trabalho proposto.

O terceiro capítulo descreve a especificação e o desenvolvimento deste trabalho.

As conclusões do trabalho e algumas sugestões para futuros trabalhos encontram-se no quarto capítulo.

2 CONTEXTO DO PROJETO

Neste capítulo serão apresentados alguns trabalhos que estão relacionados ao projeto iniciado em Mattos (1999), da criação de ferramentas didáticas que auxiliem os alunos com dificuldades de aprendizado na disciplina de lógica de programação.

2.1 TRABALHO DE MATTOS

Como primeira proposta, já validada, Mattos (2000), descreve uma proposta de ferramenta didática baseada em sistemas especialistas, a qual tem por objetivo introduzir o conceito de análise de requisitos já nas primeiras fases do ensino de computação, de tal forma que o aluno acostume-se a realizar uma análise mais detalhada dos problemas que, apesar de serem simples, já requerem algum grau de formalização.

Em função das dificuldades relacionadas com a disciplina de Algoritmos (ou com nome similar), cujo principal objetivo é o de introduzir os conceitos básicos de lógica de programação, Mattos (2000) complementa que esta dificuldade é na maioria das vezes, decorrente da falta de experiência com os aspectos relacionados a ambientes industriais e/ou comerciais, pois é a partir destes ambientes que são caracterizados os exercícios propostos.

2.1.1 Motivação

A partir da constatação do problema, observou-se que, quando induzidos a pensar sobre o problema através de perguntas direcionadas, em sua grande maioria, os alunos conseguem descrever a solução “intuitivamente”, ou seja, sem o formalismo necessário à área de computação.

Em função disto, optou-se pelo desenvolvimento de um sistema especialista que a partir do conhecimento informal sobre o processo de indução, permitisse a construção de uma ferramenta de software que possibilitasse ao aluno, na ausência do professor, desenvolver o processo de aprendizado dos conceitos básicos da disciplina.

Outro aspecto interessante dos sistemas especialistas é que, normalmente encontram-se referências à aplicação deste tipo de tecnologia na solução de problemas “complexos” tais como: diagnóstico médico, análise de mercado , análise de crédito e assim por diante.

Excluído: 1¶

Observou se, contudo que, o emprego de sistemas especialistas no aprendizado, na área de computação não são relatados com tanta ênfase.

2.1.2 Ambiente de desenvolvimento

Como ambiente de desenvolvimento Mattos (2000), utilizou o software CLIPS, o qual foi desenvolvido pela NASA/Johnson Space Center. Em sua origem, tinha como finalidade gerar soluções que apresentassem alta portabilidade, baixo custo e fácil integração com sistemas externos.

Segundo Giarratano (1994), CLIPS é uma linguagem de programação multiparadigma que fornece suporte à programação tradicional, ou seja, baseada em procedimentos, à programação orientada a objetos e baseada em regras.

2.1.3 Desenvolvimento do sistema

Em uma das fases do desenvolvimento do sistema, Mattos (2000), baseou-se em uma turma na disciplina de Lógica de Programação, na Universidade Regional de Blumenau (FURB) e verificou que, o processo de indução apresentava resultados. Procurou identificar quais as perguntas e em que seqüência elas eram apresentadas no sentido de conduzir a turma em direção a uma possível solução de alguns dos problemas propostos.

Como primeira proposta procurava fazer com que o aluno respondesse às seguintes questões:

a) Quais as "variáveis" conhecidas?;
b) O que precisa ser calculado (ou executado)?;
c) Quais as variáveis desconhecidas?;
d) O que precisa ser informado (digitado)?;
e) O que precisa ser apresentado (impresso)?;
f) realizar um esboço da solução;
g) construir um Fluxograma;
h) construir um teste-de-mesa.

Excluído: a)
Excluído: ¶
Formatados: Marcadores e numeração
Excluído: b)
Excluído: c)
Excluído: d)
Excluído: e)
Excluído: f)
Excluído: g) C
Excluído: ¶
Excluído: h)
Excluído: ;

Embora a metodologia descrita acima facilitasse mais o encaminhamento da solução, ainda carecia de um detalhamento melhor, pois havia obviamente muitos passos ocultos entre a passagem das fases de (a) a (e) para a fase (f). Para detalhar o aspecto "obscuro", refinou-se

Excluído: r que
Excluído: m

Excluído: 1¶

o conhecimento a ponto de obter-se uma planilha com 28 colunas. Nesta planilha, além de identificada a seqüência de perguntas, identificou-se também grupos de perguntas relacionadas, de tal forma que, dependendo das respostas obtidas, somente um subconjunto das mesmas faz-se necessário responder para atingirem-se os objetivos desejados.

Excluído: c

2.1.4 Implementação

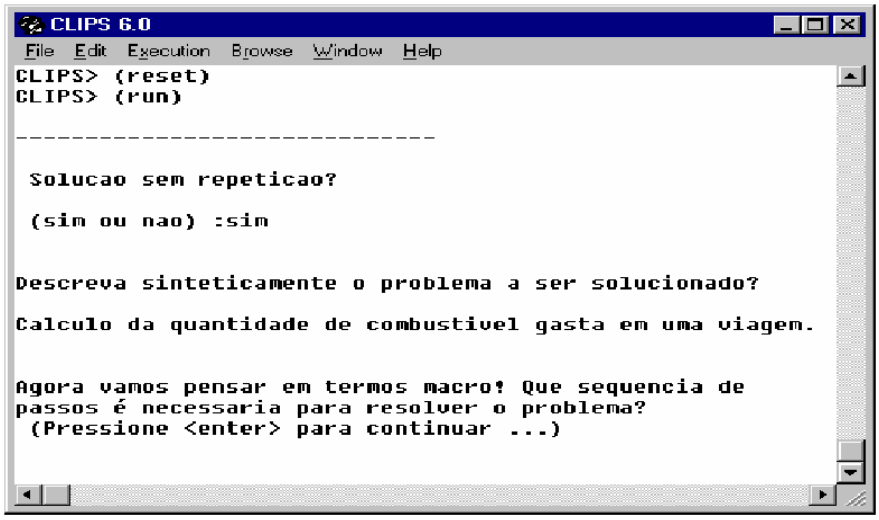
A estratégia adotada para orientar o aluno foi a filosofia de desenvolvimento top-down, onde o desenvolvimento da aplicação dá-se por refinamentos sucessivos. Sempre que houver necessidade, um novo passo de refinamento vai sendo realizado até a obtenção do nível desejado de especificação que efetivamente solucione o problema.

Para tanto, foram construídas regras, que permitem a navegação através dos nodos das árvores de decisões. Assim sendo, na medida em que os nodos vão sendo repetidamente visitados, novos fatos vão sendo gerados na memória de trabalho. Estes novos fatos disparam regras que, uma vez executadas, geram uma nova árvore auxiliar, que registra as respostas do usuário e estabelece a seqüência em termos temporais em que as respostas vão sendo cadastradas.

Excluído: à medida em que

A fig. 1 apresenta uma tela onde é solicitado ao aluno que descreva sinteticamente o problema a ser resolvido.

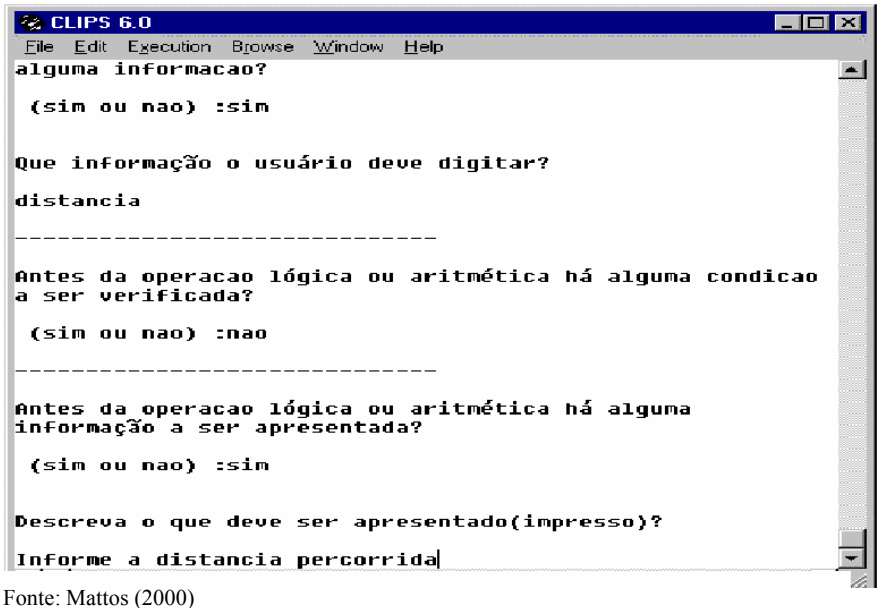
Excluído: 1¶



Fonte: Mattos (2000)

FIGURA 1 – Tela CLIPS contextualização do problema

A figura abaixo mostra ainda a continuação das perguntas que servem de condução a um melhor entendimento do contexto do problema a ser solucionado.



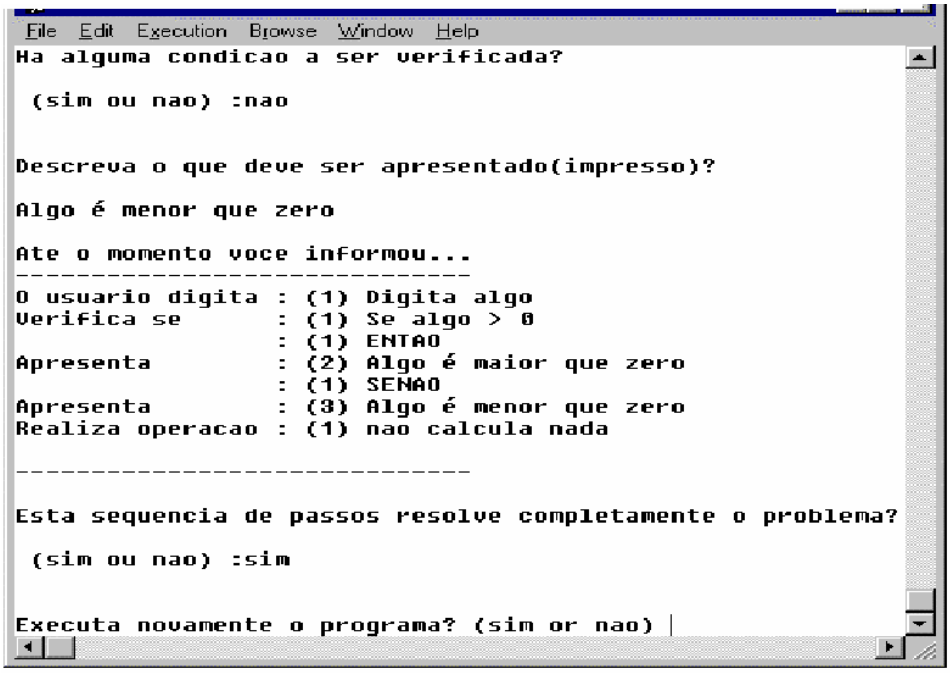
Fonte: Mattos (2000)

FIGURA 2 – Tela CLIPS perguntas do sistema especialista

Excluído: 1¶

Após cada rodada na árvore de decisões, apresenta-se ao aluno um feedback do contexto delineado pelo mesmo até o momento, e dependendo do nível de refinamento necessário, automaticamente o sistema inicia um novo passo de refinamento de alguma estrutura que ainda requeira informações complementares (fig. 3).

Excluído: ura



Fonte: Mattos (2000)

FIGURA 3 – Tela CLIPS feedback do contexto

2.1.5 Considerações

Segundo Mattos (1999), pode-se considerar que, apesar da interface com o usuário ser bastante primitiva, a implementação do protótipo atendeu aos objetivos iniciais que eram o desenvolvimento de uma aplicação em software que se constituísse em uma ferramenta de apoio ao aprendizado de lógica de programação.

Porém como continuação deste trabalho, Mattos (2000) sugeriu desenvolver uma interface mais amigável com o usuário, e também a implementação de estruturas de suporte a problemas cuja solução envolva estruturas de repetição e manipulação de vetores e matrizes.

2.2 TRABALHO DESENVOLVIDO POR GUBLER

Validando a idéia descrita em Mattos (1999), foi desenvolvido por Gubler (2002) um protótipo para auxiliar no ensino de algoritmos.

O protótipo foi desenvolvido usando a ferramenta de programação Delphi. A ferramenta possui uma interface gráfica mais amigável com o usuário e também implementa a estrutura de suporte a problemas cuja solução envolva estruturas de repetição que ainda segundo Gubler (2002), não havia sido implementada no protótipo elaborado em CLIPS por Mattos (1999).

2.2.1 Análise

Segundo Gubler, o protótipo faz parte de um projeto onde se pretende construir um software para auxiliar no ensino de algoritmos. Neste software o aluno, através de um sistema especialista é induzido a produzir respostas para questões de algoritmos. As perguntas do sistema especialista são sempre atualizadas e melhoradas através dos casos que são sempre armazenados na base de casos do raciocínio baseado em casos.

Esta ferramenta pode ser classificada como um protótipo porque alguns aspectos referentes à construção de algoritmos ainda não foram implementados.

2.2.2 Implementação

Para a implementação do protótipo, foi utilizada a técnica de sistemas especialistas, pela qual através do conhecimento do especialista foram armazenadas regras (perguntas) na base de conhecimento. O usuário responde às perguntas e no final recebe a resposta que serão os principais passos para conseguir resolver um determinado exercício de algoritmo.

O protótipo utilizou como ferramenta de desenvolvimento o Delphi 6 por ser uma ferramenta de fácil implementação para gerar softwares em modo gráfico para o usuário. O Delphi permite criar softwares de alto desempenho sem necessariamente recorrer à linguagem

Excluído: 1¶

de baixo nível (Assembler) e tem evoluído em busca do aperfeiçoamento para poder atender as exigências cada vez mais complexas dos softwares modernos.

2.2.3 Funcionamento

Para exemplificar o funcionamento do protótipo desenvolvido neste trabalho é feito um estudo de caso para resolver o problema em algoritmos que se encontra no quadro 1:

Calcular a quantidade de combustível gasto em uma viagem de um carro, onde o carro faz 12 kilometros por litro. O usuário vai informar qual a distância percorrida pelo carro na viagem, onde não poderá ser menor ou igual a zero.

Fonte: Gubler (2002)

QUADRO 1 – Descrição do exercício de algoritmos

Na fig. 4 está sendo mostrada a tela principal do protótipo desenvolvido por Gubler (2002). Para começar de um novo estudo de caso basta clicar no *botão novo* conforme circulado na figura abaixo.

FIGURA 4 – Tela principal do protótipo HelpAlgo

Excluído: 1¶

Na fig. 5 é apresentada a tela inicial com as perguntas para o usuário, onde está sendo informada a descrição do exercício que se está querendo resolver.

Excluído: ura

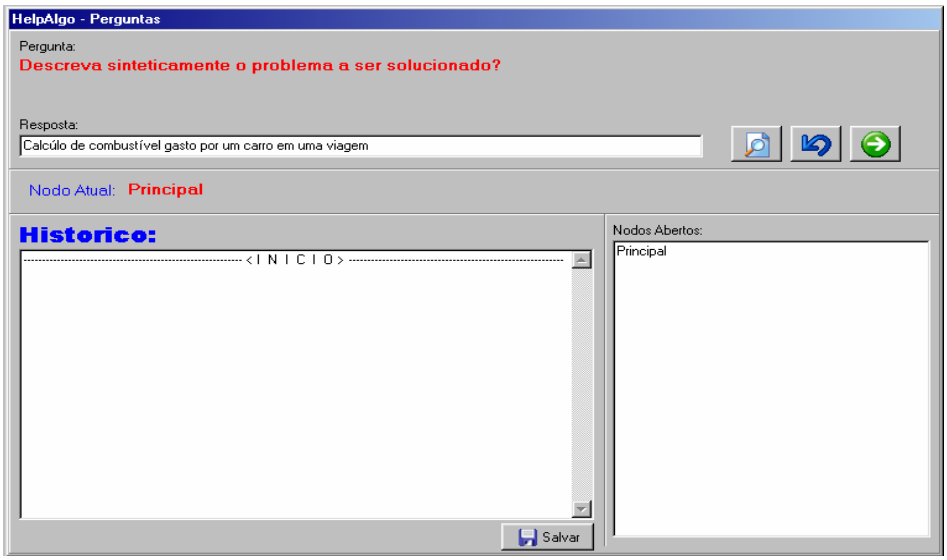


FIGURA 5 – Tela 1 do estudo de caso

Depois de informar o que vai ser apresentado para o usuário dentro do algoritmo do exercício, é mostrada novamente a tela final com os passos gerados até o momento, e logo após é feita novamente uma pergunta para verificar se existe mais algum passo dentro do nodo atual, que no caso é o lado positivo da condição (distancia > 0). Mas a resposta agora vai ser "não" porque não é mais necessário nenhum passo dentro deste nodo. Então a seguir será apresentada uma pergunta para verificar se existe algum passo dentro do lado negativo desta condição, onde a resposta é "sim" porque é necessário mostrar uma mensagem informando que a *distância digitada foi igual a zero*. Logo, após responder está pergunta, o sistema começa do início novamente perguntando se tem repetição, onde a resposta será "não" e em todas as próximas perguntas também até chegar na pergunta que é descrito "É necessário informar algo para o usuário?", onde a resposta é sim e então na próxima tela é informado o que vai ser apresentado, conforme a fig. 6.

Excluído: de

Excluído: :

Excluído: 1¶

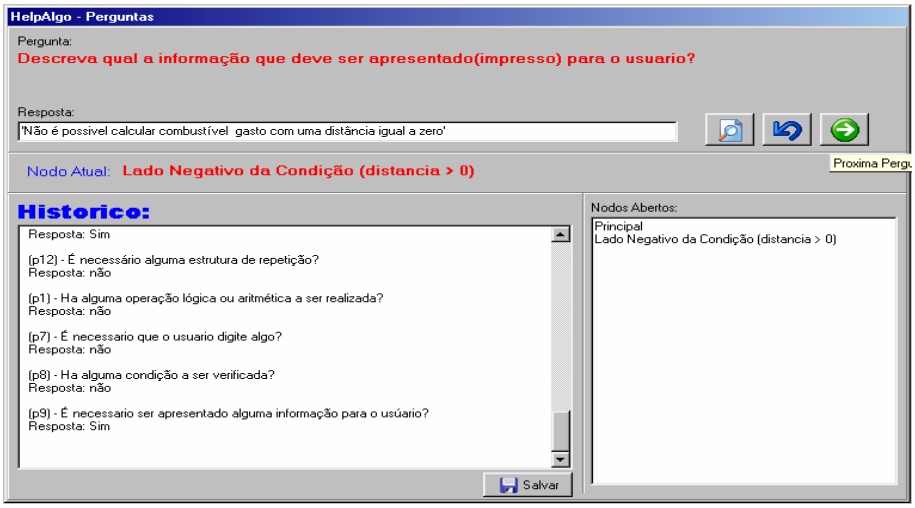


FIGURA 6 – Tela perguntas do estudo de caso

A partir de agora não é necessário mais descobrir nenhum passo, pois o objetivo do exercício foi conseguido, que foi o de gerar os passos para calcular a quantidade de combustível gasto em uma viagem. Assim sendo, foram gerados os passos que estão sendo mostrados na fig. 7.

Excluído: :

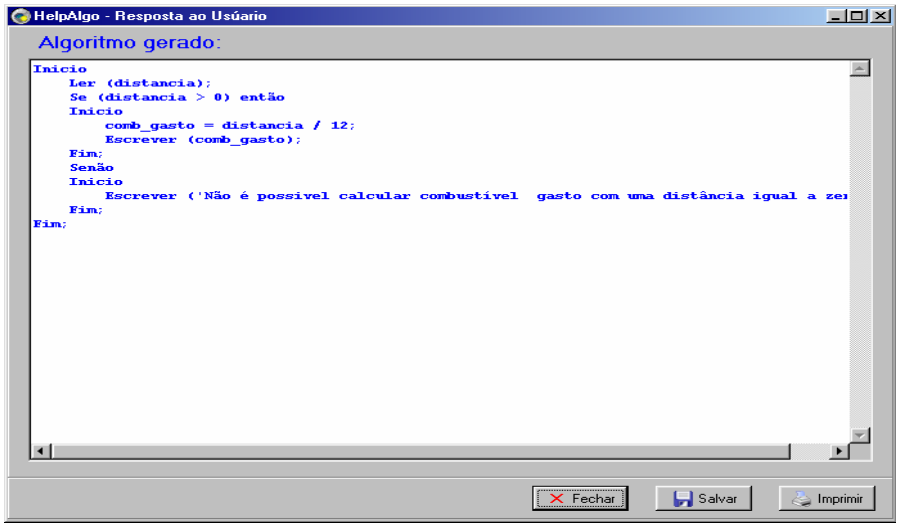


FIGURA 7 – Tela algoritmo gerado

Excluído: 1¶

2.3 TRABALHO DE HEINZEN

Este trabalho teve como objetivo complementar o trabalho de Mattos (2000), transformando em um módulo de Raciocínio Baseado em Casos (RBC) operacional que possa ser usado no dia a dia pelos alunos.

Este módulo permite que o usuário, o professor, utilize o sistema para cadastrar novos casos na Base de Casos. Como exemplo de caso de entrada, o enunciado de um problema de algoritmos contendo o Enunciado, chave Básica, palavras-chave presentes no caso e a solução para aquele enunciado.

O módulo é incorporado a um sistema especialista que por meio do enunciado, e através de perguntas acessa o RBC e retorna ao Sistema Especialista já reconfigurado, a fim de trazer a solução do problema.

Como complemento ao trabalho de Gubler (2002), este módulo acoplado dentro do sistema HelpAlgo, que foi desenvolvido paralelamente no mesmo semestre.

A implementação do RBC foi feita em Delphi, e tem como detalhes mais importantes em sua implementação o analisador de textos que é responsável pela fase de identificação das características do sistema no reconhecimento do enunciado, a base dos casos que são guardados em arquivos textos e o cálculo da similaridade dos casos, que fazem com que o programa consiga calcular de maneira satisfatória a similaridade entre o caso de entrada e os casos que se encontram na biblioteca de casos.

A integração dos módulos RBC e do módulo de Sistema Especialista, que resulta em uma ferramenta de apoio ao ensino de Lógica de Programação, através do apresentado aqui, em Mattos (1999,2000), é uma ferramenta promissora, e faz com que as dificuldades apresentadas pelos alunos sejam mais facilmente resolvidas.

Excluído:) e Mattos (2000

Excluído: 1¶

2.4 OUTROS TRABALHOS CORRELATOS

Existem vários trabalhos desenvolvidos que podem ser categorizados como parte das ferramentas de apoio ao ensino, as quais poder ser consideradas como trabalhos correlatos a esse.

2.4.1 Trabalho de Cares (2002)

Neste trabalho é desenvolvido usando a linguagem de programação Delphi 5, um ambiente de teste de mesa utilizando fluxograma, que segundo Cares (2002), explora a inteligência espacial-visual visando melhorar e facilitar a assimilação das técnicas de programação.

Esta ferramenta trabalha apenas com os tipos básicos de dado (inteiro, real, caractere e lógico) e tem uma interface de fácil manipulação e entendimento, voltado a pessoas que já estejam estudando a lógica de programação e que possam, a partir da ferramenta, aplicar o conteúdo visto sem necessitar saber previamente uma linguagem de programação.

2.4.1.1 Funcionamento do sistema

Na hora em que o usuário abre o programa encontra-se com uma interface que mostra um pequeno menu, uma área com alguns símbolos gráficos de fluxograma, um painel que irá conter o desenho e um espaço para o teste de mesa. Isto é visualizado na Fig. 8.

Para usar a ferramenta, o aluno usuário precisa primeiramente montar o fluxograma selecionando o símbolo desejado e clicando na área destinada à montagem do fluxograma. Quando o símbolo é criado, é aberta uma caixa de texto para escrever as instruções, e assim vai montando símbolo após símbolo, até o fluxograma ficar completo.

Quando estiver completo o fluxograma, pode-se fazer a compilação do mesmo, clicando em um botão específico. Caso não tenha erro é apresentada uma mensagem dizendo que não tem erros, caso contrário indica onde está o erro.

Excluído: esta

Assim estando tudo certo com a compilação, pode-se fazer a execução do fluxograma e visualizar o teste de mesa.

Excluído: 1¶

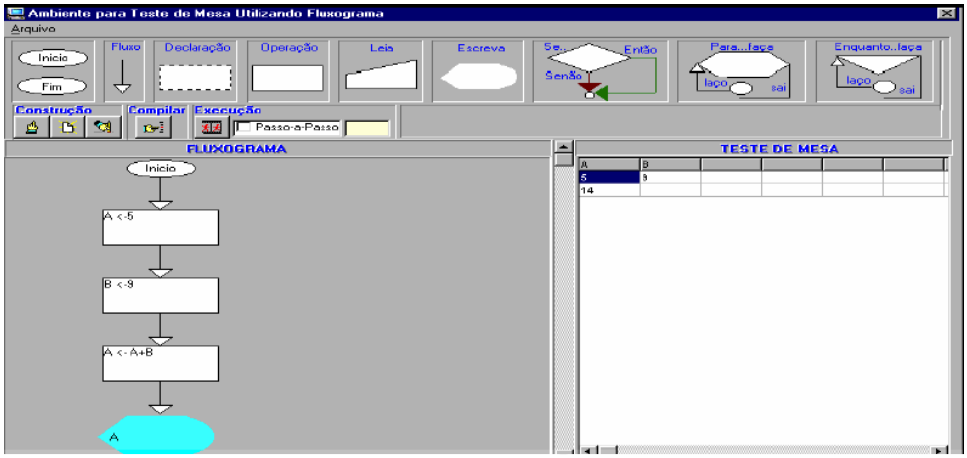


FIGURA 8 – Tela principal do ambiente do ATMUF

2.4.1.2 Considerações da ferramenta

Esta ferramenta é limitada faltando algumas estruturas como de repetição, não permite salvar os fluxogramas, mas a autora afirma que esta ferramenta pode ser aprimorada para atender soluções mais complexas.

2.4.2 Trabalho de Souza (1999)

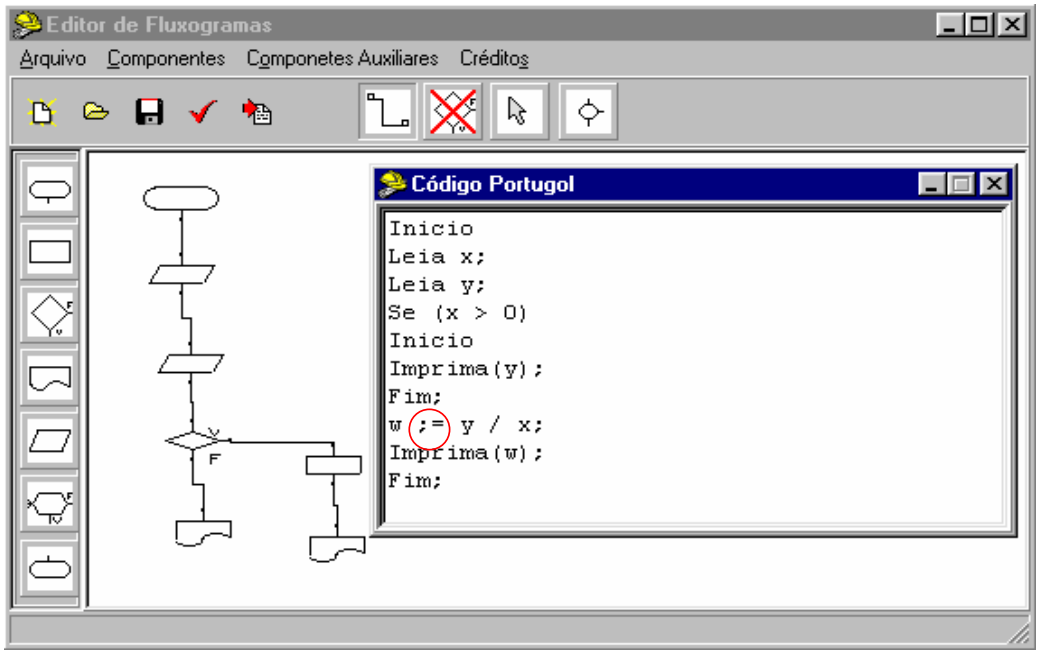
Este trabalho apresenta um editor gráfico de fluxogramas para representação de comandos da linguagem portugol, faz parte da mesma idéia de auxiliar o aluno de algoritmos.

2.4.2.1 Funcionamento da ferramenta

Como quase todos os editores, ele permite que se selecione um objeto e coloque em um ponto qualquer na área de desenho, no caso o símbolo do fluxograma. Clicando em cima de cada símbolo do fluxograma após ser colocado no painel de desenho é exibida uma pequena tela de edição para entrar com dados, no intuito de gerar o código posteriormente. Assim vai se formando todos os fluxos respeitando suas conexões até o fluxograma estar completo.

Após isto, através de um botão "*Gerar Código Portugol*" é apresentada uma janela com o código *Portugol* gerado. A fig. 9 apresenta uma amostra de uma das telas do sistema.

Excluído: 1¶



[1]FIGURA 9 – Tela exemplo do Editor de Fluxogramas

2.4.2.2 Considerações

O trabalho citado, além de ser relevante no que diz respeito à idéia do ensino de Lógica de Programação, tem também uma grande influência no protótipo proposto neste TCC que reutiliza alguns códigos desta ferramenta para gerar criação dos desenhos.

2.4.3 Software Construtor

O CONSTRUTOR elaborado pelo Centro de Informática Educacional Aplicada/DMM, tem como objetivo executar algoritmos, isto é, colocar em prática os conhecimentos adquiridos na teoria de Lógica de Programação. Mas é importante ressaltar que junto a está ferramenta existe todo um embasamento teórico em um livro trazendo inclusive este software, que se chama Construção de Algoritmos.

A ferramenta CONSTRUTOR e o livro são pontos de referência para o desenvolvimento do trabalho proposto. Por esse motivo serão apresentadas abaixo características começando primeiro pelo livro e depois avançando para o software.

Excluído: Através de consultas feitas à

Excluído: a

Excluído: ,

Excluído: é dado como

[1] Cabe destacar que o erro de sintaxe é uma falha do programa.

Excluído: 1¶

2.4.3.1 Livro Construção de algoritmos

Publicado pelo SENAC tem como objetivo de superar dificuldades no problema da formação de programadores. Na falta de embasamento teórico para o raciocínio lógico, ele é fundamental para preparar profissionais nesse setor de atividades.

O livro foi dividido em 4 capítulos e um anexo com o guia de utilização do Construtor, que abaixo é descrito.

Excluído: :

No primeiro capítulo, com o título *Algoritmo contexto genérico*, o livro conceitua algoritmo sob um contexto genérico, depois apresenta as estruturas básicas de controle que compõem um algoritmo, mostra a importância da pontuação na construção de algoritmo e como este pode ser representado, e mostra as principais formas de representar as estruturas de controle.

No segundo capítulo intitulado *Algoritmo contexto computacional*, ele mostra como se armazenam informações na memória principal do computador, como se criam variáveis necessárias à construção de um algoritmo e como se apresentam as instruções primitivas de leitura, de atribuição e de impressão em um algoritmo.

No terceiro capítulo, *Manipulação de tipos de dados*, aborda como se declaram os tipos de dados utilizados nos algoritmos e mostra como eles são usados em expressões.

No quarto capítulo, *Desenvolvimento de algoritmos*, mostra uma série de exercícios propostos e exercícios resolvidos para a prática da lógica básica de programação.

Excluído: Em

Excluído:

No Anexo 1, é mostrado o guia de utilização do software que acompanha o livro e que irá ser descrito a seguir:

2.4.3.2 Apresentação do Software Construtor

O objetivo do Software é gerar algoritmos e como características relevantes do Construtor podem-se citar:

a) uso de uma linguagem gráfica (fluxograma) que facilita a visualização dos programas;
b) visualização de algoritmos em várias linguagens (Pascal C, Clipper e Pseudocódigo);

Excluído: 1¶

c) interface amigável que utiliza ícones para a manipulação dos fluxogramas;
d) diversos modos de execução;
e) armazenamento de algoritmos em meio secundário.

2.4.3.3 Execução do Construtor

Quando se entra no Construtor, ele apresenta os seguintes elementos da interface:

a) a barra de menu com quatro opções: Arquivo, Visão, Outros e Sobre;
b) nove botões para construção do algoritmo: Declarar, Atribuir, Ler, Imprimir, Se, Enquanto, Para, Editar e Lixo;
c) quatro botões para controle da execução do algoritmo: Executar, Passo, Testar e Encerrar.

Excluído: N

Excluído: Quatro

Selecionando o menu *Arquivo/abrir*, seleciona-se um arquivo com estrutura pré-definida. O ponto de partida para usar o software é excluir as instruções deixando como fluxograma principal as estruturas Variáveis, Comandos e Fim. A partir daí cria-se um novo arquivo, conforme exemplo fig. 10.

Depois de criado, pode-se escolher em clicar no botão Executar para executar todo o fluxograma em um só momento, botão *Passo* para passo-a-passo ou *Encerrar* para encerrar a execução.

O exemplo que é mostrado é o final de uma execução passo-a-passo, onde o que está pintado de verde na figura abaixo, é a instrução que está sendo executada no momento. Pode-se observar que mais à esquerda na fig. 10 estão os resultados da Saída de dados e Variáveis.

Excluído: demonstra

Excluído: esta

Excluído: está

Excluído: 1¶

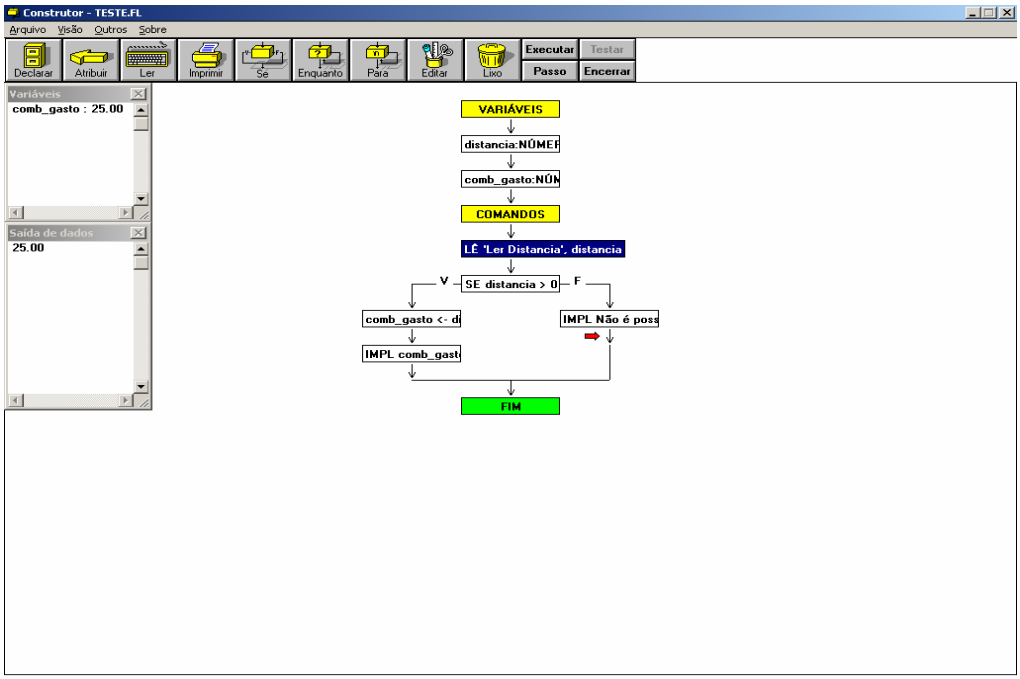


FIGURA 10 – Tela Construtor na execução passo-a-passo

O uso do software é simples, mesmo assim existe uma pequena dificuldade ao criar um novo fluxograma e também em outros pontos como no tratamento dos nomes das variáveis e na geração dos códigos em Pascal, C, Clipper, Pseudocódigo.

2.4.4 Animação e Simulação de Algoritmos (ASA)

Segundo Cares (2002), *apud* SENAC, ASA é uma ferramenta que possui um conteúdo teórico completo da disciplina de lógica sobre sistemas numéricos, memória, variáveis, comandos simples, operações lógicas, seleção SE, repetição Para, representações, vetores, ordenação e matrizes, e apresenta vários exemplos explicativos em relação a esse conteúdo.

Excluído: conforme autor

Excluído: Cares

A interface do ASA não é de fácil manipulação, faltam explicações nos ícones a respeito de suas funções, (o que nos faz entrar em cada opção para saber qual o propósito da ferramenta) e como utilizar seus recursos.

O ASA possibilita a apresentação da estrutura lógica em fluxogramas, contudo a estrutura apresentada no ASA para o fluxograma não segue a representação real das estruturas de fluxograma; a lógica pode também ser apresentada por um diagrama estruturado e por um pseudocódigo.

Excluído: 1¶

Esta ferramenta permite ao usuário construir e testar passo a passo as suas soluções. Porém esta utiliza um formato que engloba portugol com os fluxogramas em uma só solução, o que por um lado pode ser interessante, pois apresenta as instruções escritas (tipo um portugol) e setas direcionais interligando essas, para demonstrar a direção do fluxo de execução do algoritmo. Por outro lado, não utiliza o recurso gráfico para facilitar a assimilação do significado das estruturas de controle.

Excluído: 1¶

3 DESENVOLVIMENTO DO TRABALHO

O presente capítulo descreve a especificação do modelo de geração de fluxograma e teste de mesa do projeto descrito em Mattos (1999).

Na primeira seção estão descritos os requisitos principais do problema a ser trabalhado.

A segunda seção mostra a especificação do trabalho proposto.

A terceira e última seção abrange as técnicas e ferramentas utilizadas juntamente com a operacionalidade da implementação.

Excluído: Técnicas

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Como já foi destacado anteriormente, o objetivo deste trabalho é validar a idéia descrita em Mattos (1999) e Gubler (2002), desenvolvendo um protótipo para ler um algoritmo com uma sintaxe definida, gerar o fluxograma e executá-lo colorindo a cada passo a fim de orientar o aluno através de figuras geométricas, numa forma mais estruturada de elaborar algoritmos corretamente, onde ele pode ver cada criação de variáveis, atribuições, cálculos, saídas, decisões e repetições.

Excluído: que leia

3.2 ESPECIFICAÇÃO

Nesta parte do trabalho, é apresentada a especificação do Protótipo de animação de algoritmos, sendo apresentados os levantamentos de requisitos, Diagrama de Casos de Uso e Diagrama de seqüência.

A fig. 11 apresenta as duas possibilidades de uso do sistema:

Excluído: ura

Excluído:

a) utilizando o programa HelpAlgo, que conforme descrito anteriormente, é gerado um algoritmo que pode ser salvo. Este arquivo texto pode ser aberto e interpretado a partir do protótipo desenvolvido;
b) outra forma é através do protótipo que permite a criação de um algoritmo novo que, para ter sucesso na geração e execução do Fluxograma precisa obedecer algumas regras de sintaxe descritas em anexo.

Excluído: ;

O resultado da execução do protótipo é a constatação que a lógica está ou não esta correta.

Excluído: 1¶

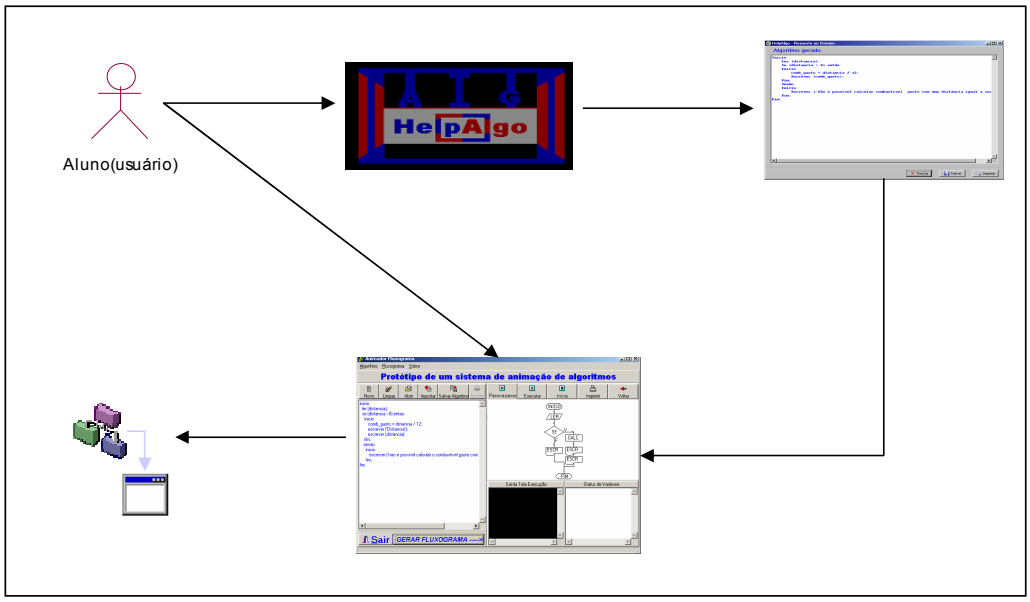


FIGURA 11 - Tela narrativa do sistema

3.2.1 Levantamento de requisitos

Segundo Bezerra (2002), a atividade de levantamento de requisito corresponde à etapa de compreensão do problema aplicada ao desenvolvimento de software. Os requisitos podem ser de três tipos: funcionais, não-funcionais ou de restrições.

Os requisitos funcionais definem as funcionalidades do sistema. Para essa ferramenta, pode-se definir como requisitos funcionais os seguintes itens:

a) o sistema deve permitir que o usuário possa abrir qualquer arquivo texto;
b) o sistema deve gerar o desenho gráfico do fluxograma desde que o algoritmo esteja sintaticamente correto com o padrão gerado pelo Protótipo HelpAlgo (o anexo 1 descreve a sintaxe utilizada);
c) o sistema deve permitir que execute passo-a-passo o fluxograma gerado, pintando da cor vermelha o que está sendo executado no momento;
d) o sistema deverá criar as variáveis dinamicamente sem ser preciso declará-las;
e) o sistema deverá mostrar em uma área específica os conteúdos das variáveis a cada passo e também resolver cálculos aritméticos e lógicos;
f) o sistema deve, na execução, pedir entrada de dados e mostrar suas saídas em uma área especifica.

Excluído: 1¶

Os requisitos não-funcionais declaram as características de qualidade que o sistema deve possuir e que estão relacionadas às suas funcionalidades. Para esse software pode-se considerar que:

a) deve ser implementado em Delphi tendo em vista adequar-se ao framework já desenvolvido;
b) deve ser compatível com a linguagem portugol gerada pelo trabalho de Gluber (2002).

3.2.2 Diagrama de Casos de Uso

A fig. 12 mostra o caso de uso do sistema, visto do ponto de vista do aluno (usuário) que irá usar o sistema.

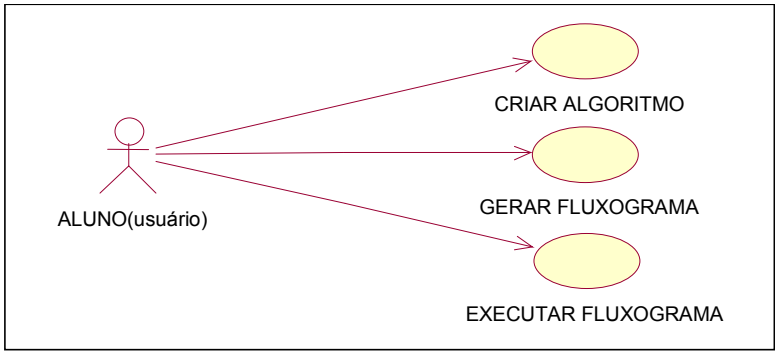


FIGURA 12 –Apresentação do Diagrama de Casos de Uso

CRIAR ALGORITMO

Sumário: O aluno (usuário) usa a ferramenta HelpAlgo para resolver um problema de estudo de caso retornando o algoritmo ou pode criar um novo.

Ator Principal: Aluno (usuário)

Precondições: O algoritmo terá que obedecer a uma sintaxe definida.

Fluxo Principal:
a) O aluno Abre ou cria um algoritmo.
b) O algoritmo editado pode ser salvo, alterado e impresso.

GERAR FLUXOGRAMA
Sumário: O aluno, com o algoritmo editado, gera o fluxograma desde que sua sintaxe esteja correta. **Ator Principal:** aluno (usuário) **Precondições:** Deve ter algum algoritmo aberto na ferramenta.
EXECUTAR O FLUXOGRAMA
Sumário: O aluno inicia com o processo de gerar o fluxograma, ficando disponível a opção de execução passo-a-passo. **Ator Principal:** aluno (usuário) **Precondições:** Deve estar com o fluxograma gerado completamente sem sua estrutura corrompida. **Fluxo Principal:** a) O aluno executa cada passo observando a mudança de cor no fluxo. b) O aluno observa seus pedidos de entrada, saída e variáveis. c) É aberta uma caixa onde é permitida a entrada dos dados solicitados no algoritmo.

QUADRO 2 - Documentação sobre os casos de uso

3.2.3 Diagrama de Seqüência

Neste Diagrama de Seqüência é apresentado o terceiro Caso de Uso (Executar o fluxograma).

O diagrama de seqüência para o caso de uso *Executar Passo-a-Passo* é formado por seis métodos, conforme Fig. 13.

O método CRIAR_NOVO_ALGORITMO dá início ao processo de geração do fluxograma.

Excluído: 1¶

Os métodos ABRIR_ALGORITMO e SALVAR_ALGORITMO seleciona um arquivo texto a ser aberto ou salvo em um diretório que vem por definição.

O método LER_ESTRUTURA é o processo de leitura do algoritmo que é executado depois que o usuário solicita a geração do fluxograma.

O método GERAR_FLUXOGRAMA é o método que ativa o LER_ESTRUTURA, e desenha paralelamente o Fluxograma.

Excluído: o método

O método EXEC_PASSO-A-PASSO é aquele em que é percorrido cada símbolo do fluxograma mudando sua cor e fazendo o teste de mesa.

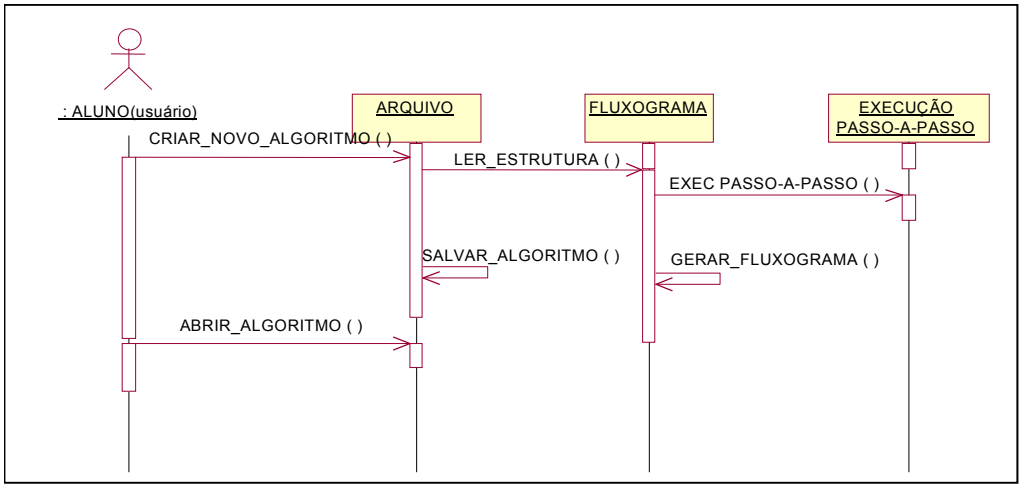


FIGURA 13 – Diagrama de seqüência

3.3 IMPLEMENTAÇÃO

Nesta seção é detalhada a implementação do protótipo, mostrando através de fluxogramas como foi estruturado o software em relação as suas *funções* e *procedimentos*.

Excluído: c

Excluído: procedimento

Para melhor compreensão a ferramenta foi dividida em três partes:

a) a primeira fala sobre a edição e contagem da quantia das estruturas "SE";

b) a segunda parte descreve a montagem do fluxograma juntamente com o armazenamento dos comandos no *array*.

Formatados: Marcadores e numeração

Excluído: A

Excluído: A

Excluído: 1¶

c) a terceira, mostra como é feito o processo da execução passo-a-passo.

Excluído: A

3.3.1 Edição e Contagem de “SE(s)”

A fig. 14 apresenta a estrutura lógica de funcionamento que permite a edição de um algoritmo.

Excluído: ura

Excluído: Abaixo

Excluído: 1¶

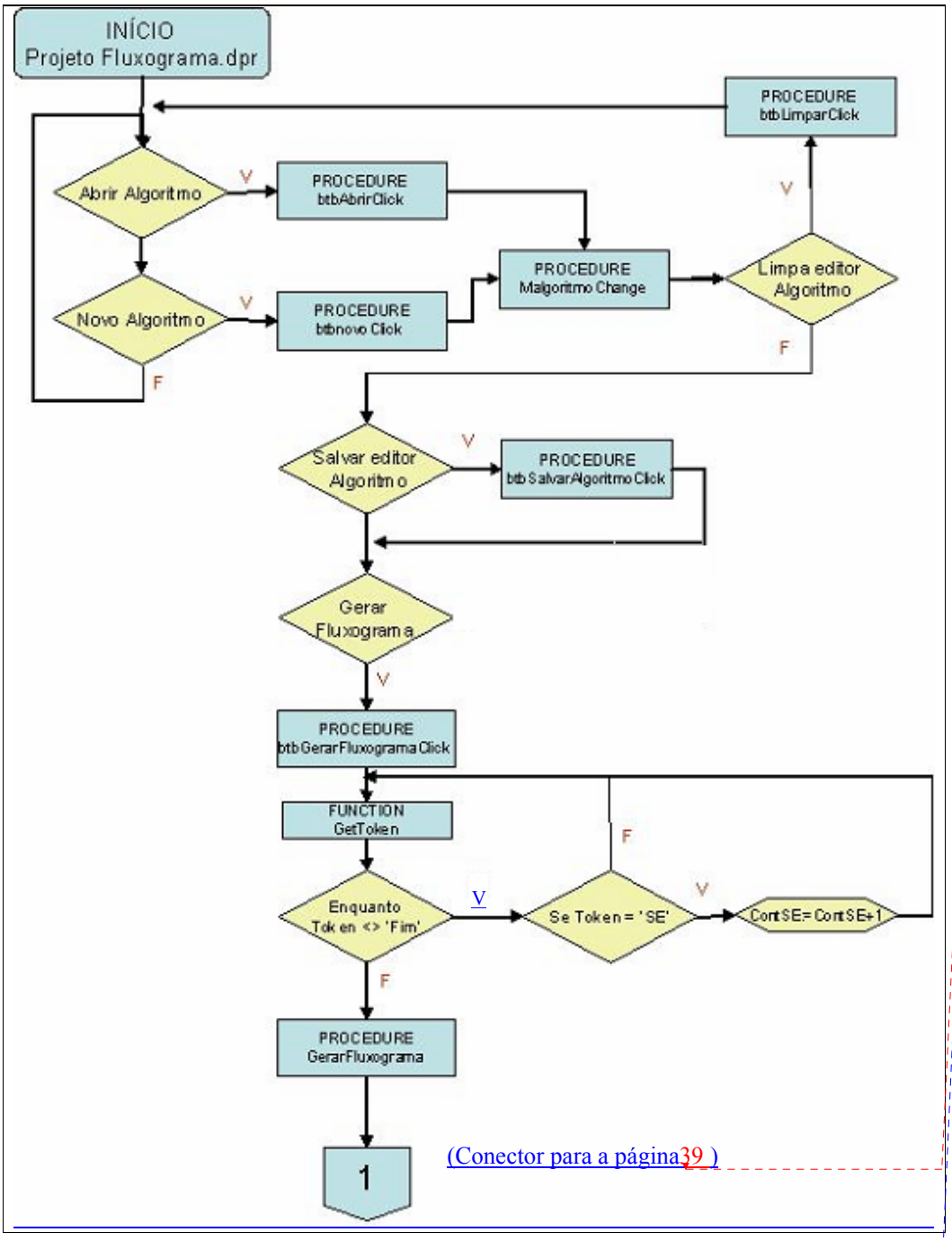
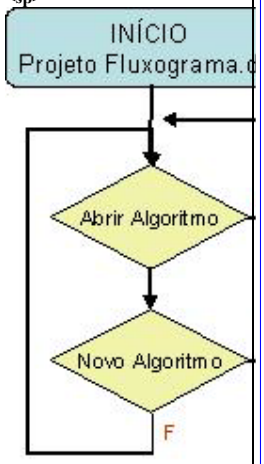


FIGURA 14 – Fluxograma de edição de algoritmo

Excluído: 29

Excluído: <sp><sp><sp><sp>



Excluído: 1¶

Quando executado, o protótipo fica esperando ser editado um algoritmo no campo *memo* do Delphi 7. Assim como é mostrado no fluxograma a partir do símbolo Início, segue a explicação a seguir.

Excluído: i

Excluído: do

Quando é pressionado o Botão ABRIR, é chamada a procedure btbAbrirClick, que por sua vez, trás a janela abrir do Windows. Já pressionando o botão NOVO é chamado a procedure btbNovoClick. A procedure mAlgoritmoChange administra o algoritmo editado.

Excluído: que

O algoritmo no campo *memo* pode ser apagado pressionando-se o botão LIMPAR, que dispara a procedure btbLimparClick que limpa todas as linhas do campo *memo* deixando pronto para editar.

Se clicar em SALVAR, chama-se a procedure BtbSalvarAlgoritmoClick, que pega o conteúdo do campo Memo e transforma em um arquivos texto, que pode ser salvo no diretório desejado.

À medida que se interage com o campo memo, é executado um comando para habilitar ou não o botão GERAR FLUXOGRAMA. Clicando-se neste Botão é ativada a procedure btbGerarFluxogramaClick que faz todas as inicializações da tela que vai ser exibida no fluxograma e também faz a verificação da quantia de "SE(s)" que o algoritmo possui.

Como motivo principal para contagem dos "SE(s)" é que na geração do desenho dos Fluxos dos "Se(s)", há uma grande probabilidade de cair o desenho em cima do outro. Na hora em que for executado o desenho do fluxo, conforme a quantia ser maior que 1, a linha da parte verdadeira começa de um maior tamanho vindo para um menor.

Para determinar a quantia de SE(s) é utilizado uma Função chamada de Gettoken que varre o algoritmo, trazendo como resposta um conjunto de caracteres e sua classificação (Símbolo, Literal, Número, Pontuação e Palavra qualquer, data, fim.).

É verificado palavra por palavra no algoritmo, enquanto o Token, que é o retorno, for diferente de fim e a cada encontro a variável ContSe é incrementada.

Excluído: 1¶

3.3.2 Geração do desenho do fluxograma e armazenamento das expressões

O fluxograma desenhado na fig. 15 apresenta a lógica utilizada na varredura dos tokens no algoritmo gerando de forma seqüencial do desenho e armazenamento das instruções.

Excluído: abaixo

Excluído:

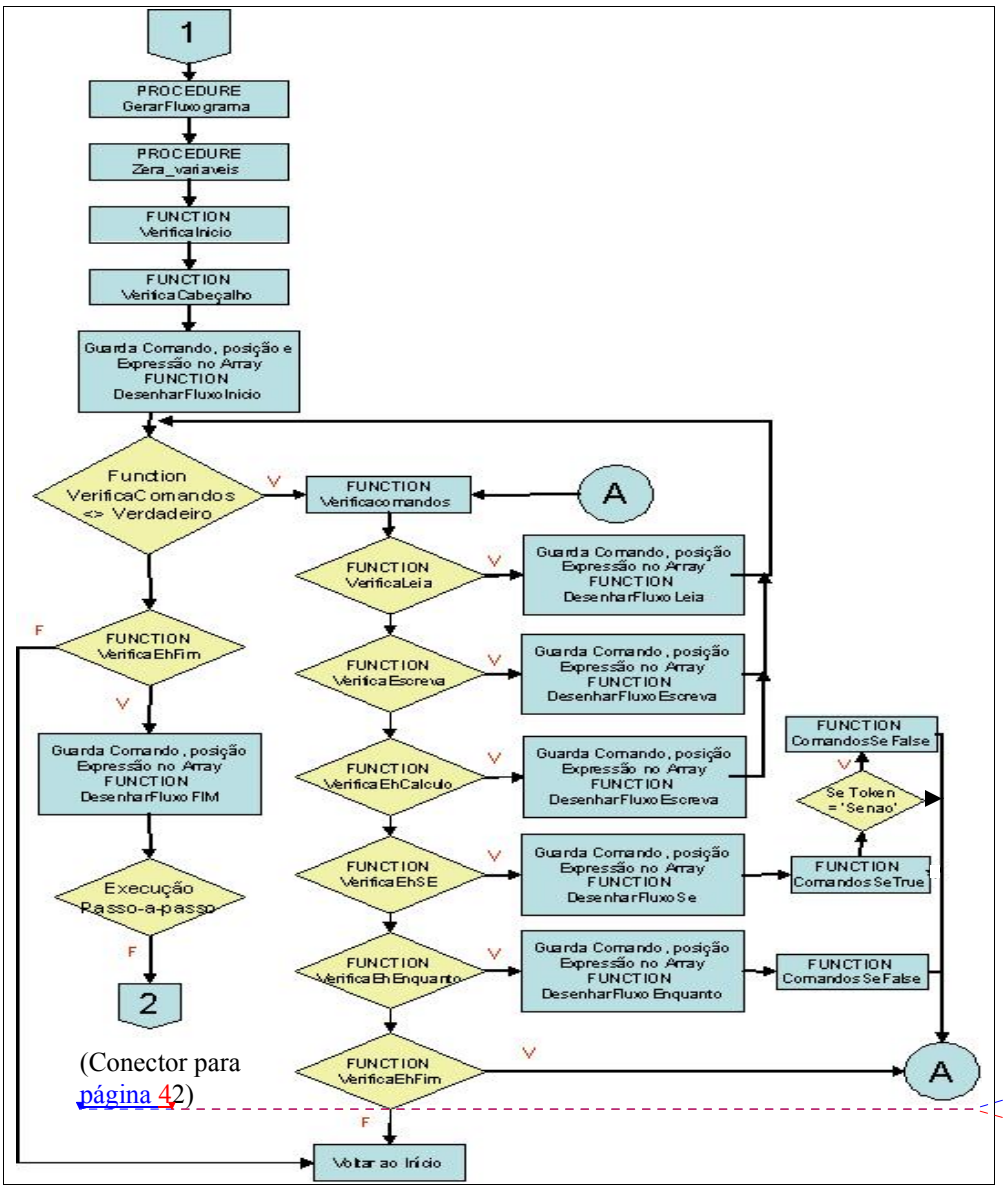


FIGURA 15 – Gerar Fluxograma

Excluído: pagina

Excluído: 3

Excluído: 1¶

Após ser feita a contagem dos “SE(s)”, é posicionado o ponto de leitura do Token na sua primeira linha e coluna do algoritmo fazendo inicializações ainda no btbGerarFluxograma.

Chama-se a procedure GerarFluxograma responsável por todo o processo do desenho e armazenamento de todas as características no array.

O primeiro passo de Gerar Fluxograma é zerar todos os espaços que são cedidos para as criações das variáveis no algoritmo chamando a procedure Zerar_Variaveis.

O segundo passo é verificar o inicio do algoritmo através da procedure VerificaInicio, que testa se o algoritmo tem cabeçalho.

Excluído: Exemplo: “**Programa CalcCombustivel"**¶

Caso não tenha cabeçalho ele verifica se a próxima palavra é “inicio” chamando a procedure respectiva, sendo que verdadeira guarda a palavra Inicio, uma posição X e Y onde irá ser o ponto início do desenho do fluxo e é chamada a Function *Desenhar_inicio* levando esta posição inicial e trazendo uma posição final.

Excluído: inicio

Excluído:

Excluído: *início*

Como terceiro passo, reconhecido e desenhado o fluxo Inicio, é hora de desenhar o corpo do algoritmo até ser encontrado o fim. Para isto é criado a Function VerificaComandos que enquanto ela for verdadeira, isto é, enquanto tiver comandos irá desenhando os símbolos.

Os fluxos Ler, Escrever e Cálculo são um pouco mais complicados, pois da mesma forma que o início é executado eles também são executados, somente com o diferencial que é de guardar no array *passo-a-passo* uma expressão para ser usada posteriormente na execução passo-a-passo.

Excluído: i

O Fluxo SE inicialmente desenha o Losango pegando uma posição final X e Y gerada do fluxo anterior e resultando em duas saídas X e Y que representa a saída do lado verdadeiro e X2, Y2 que representa a saída do lado falso.

Verifica-se os comandos verdadeiro executando uma função ComandosSeTrue, que retorna verdadeiro se todos os comandos daquele laço forem executados. Isto é recursivo sendo que internamente é chamada a function VerificaComandos novamente.

Depois da execução da Função ComandosSeTrue, verifica se tem a palavra SENAO se tiver executa a Função ComandosSeFalse. Lembrando que ao encontrar a palavra fim, é

desenhado as conexões de encontro para resultar a continuação do desenho partindo de um só ponto X e Y.

O fluxo ENQUANTO trabalha semelhantemente ao “Se”, porém na hora de desenhar o bloco de comandos usa somente a Function ComandosSeFalse enquanto condição for verdadeira até encontrar a palavra “fim”.

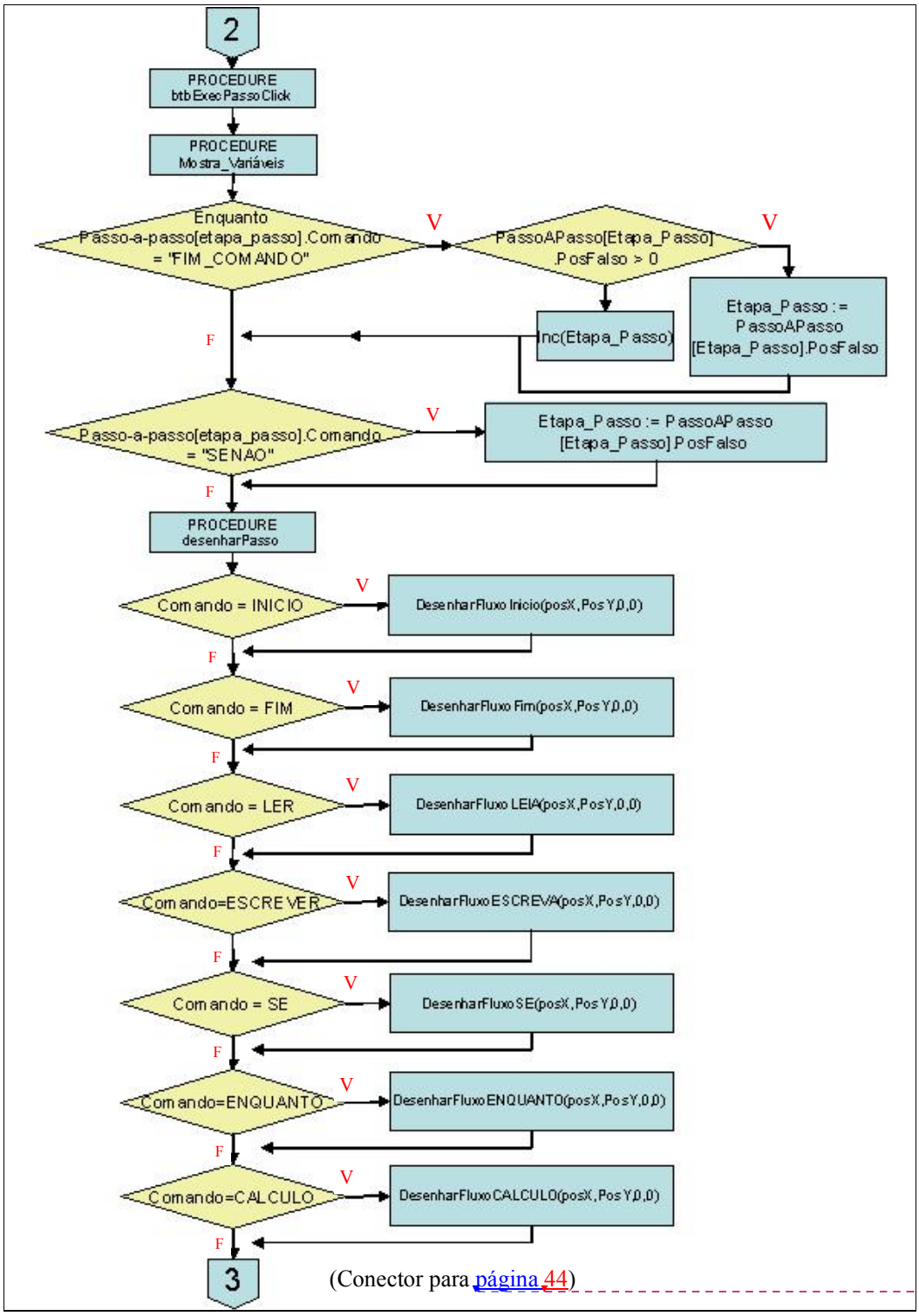
Quando todos os comandos do fluxograma forem executados, é encontrada a palavra fim, e a function VerificaComandos será verdadeira indo para a parte final do algoritmo que é verificar se termina com “fim.”.

Com tudo isto o fluxograma já está gerado, desde que na leitura de cada token não tenha encontrado nenhum erro na escrita. A imagem é gerada em um componente TImage do Delphi, utilizando a biblioteca Canvas para desenho.

3.3.3 Execução Passo-a-passo

Os dois fluxogramas desenhados (fig.16 e fig.17) mostram o funcionamento da execução passo-a-passo.

Excluído: 1¶



(Conector para página 44)

Excluído: pagina

Excluído: 34

FIGURA 16 – Redesenho na execução passo-a-passo

Excluído: 1¶

Conforme o fluxograma da Fig. 16, toda vez que é clicado no Botão PASSO-A-PASSO, é mostrado em um campo *memo* (mVariaveis), o conteúdo das variáveis trabalhadas através da Procedure Mostra_variaveis.

É indicado através da decisão FIM_COMANDO e no SENAO o ponto de saída para a execução dos laços verdadeiro e falso para que, conforme a decisão ele pinta e executa somente o laço que for verdadeiro conforme a expressão passada.

As outras decisões como LER, ESCREVER e COMANDOS servem para pintar todo o fluxograma novamente de Preto. Isto serve para atualizar o desenho, pois no próximo fluxograma ilustrado na fig. 17, deixa marcado de vermelho não sendo apagado depois que executou o comando.

Depois que é executado o bloco acima, é hora de realmente executar o fluxograma.

Primeiramente é atribuído 2 para a cor, que significa que na hora em que chamar uma procedure que desenha os símbolos, dentro de cada uma delas irá perguntar se a Cor é igual a 2, e assim desenha-se em vermelho.

Na decisão dos comandos INICIO e FIM, só é redesenhado em vermelho sem nenhum teste relevante.

Já nas outras decisões é trabalhado com as variáveis e expressões.

Se for a instrução LER, é procurado através de uma função, se já existe uma variável com o mesmo nome, se não existir cria-se automaticamente no array variáveis.

Na instrução ESCREVER, é analisado se o que vem escrito após a palavra *escrever* é uma literal ou uma variável. Se não começar com "aspas simples ou duplas", é verificado no array *variáveis* se existe uma com o mesmo nome, caso contrário é impresso na tela uma mensagem que a variável não existe.

Excluído: é imprimido

Na decisão do SE e do ENQUANTO, analisa-se se a expressão é verdadeira ou falsa. Para analisar foi utilizada uma *unit Evaluate*, que é mandado a expressão em string e é retornado o resultado. Se a expressão for falsa, no array que guarda todos os comandos, é desviado para a execução do laço correspondente.

Excluído: 1¶

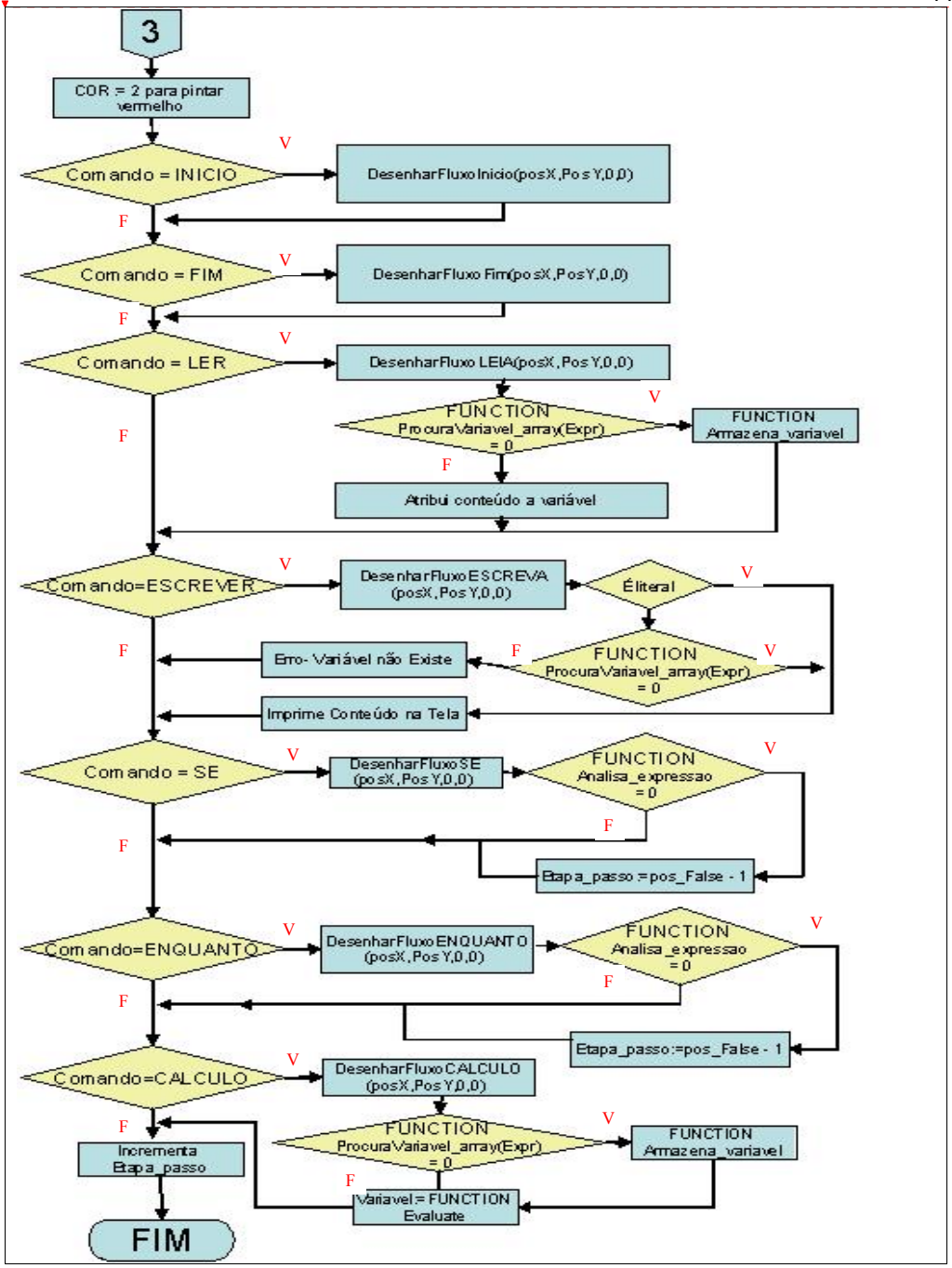


FIGURA 17 – Fluxograma passo-a-passo cálculo das expressões

No comando CALCULO, é lida toda a linha da expressão, retira-se a palavra antes do sinal da igualdade e a compara para ver se já existe na lista do array variáveis, se não existir é

Excluído: 1¶

criada. Esta variável é guardada em uma variável chamada de VCalc, onde irá ser recebido a atribuição.

Excluído: ,

O cálculo propriamente dito é visto depois do sinal de igualdade. A expressão é convertida analisando se no meio do cálculo se está trabalhando com nomes de variáveis. Depois de convertida é chamada a função *Evaluate* que leva esta expressão, faz os cálculos e retorna o seu resultado, atribuindo assim à variável "Vcalc" .

Para finalizar este processo é incrementada a variável Etapa_passo, posicionando o array PassoApasso no próximo comando e assim repete-se sucessivamente até encontrar o FIM COMANDO.

3.3.4 Operacionalidade da Implementação

Excluído: ,

O protótipo animador de fluxograma apresenta duas janelas principais, uma é utilizada para a edição do algoritmo, e a outra é criada depois de pressionado o botão GERAR FLUXOGRAMA. A figura 18 mostra o andamento de uma execução que contém estas duas janelas.

Excluído: 1¶

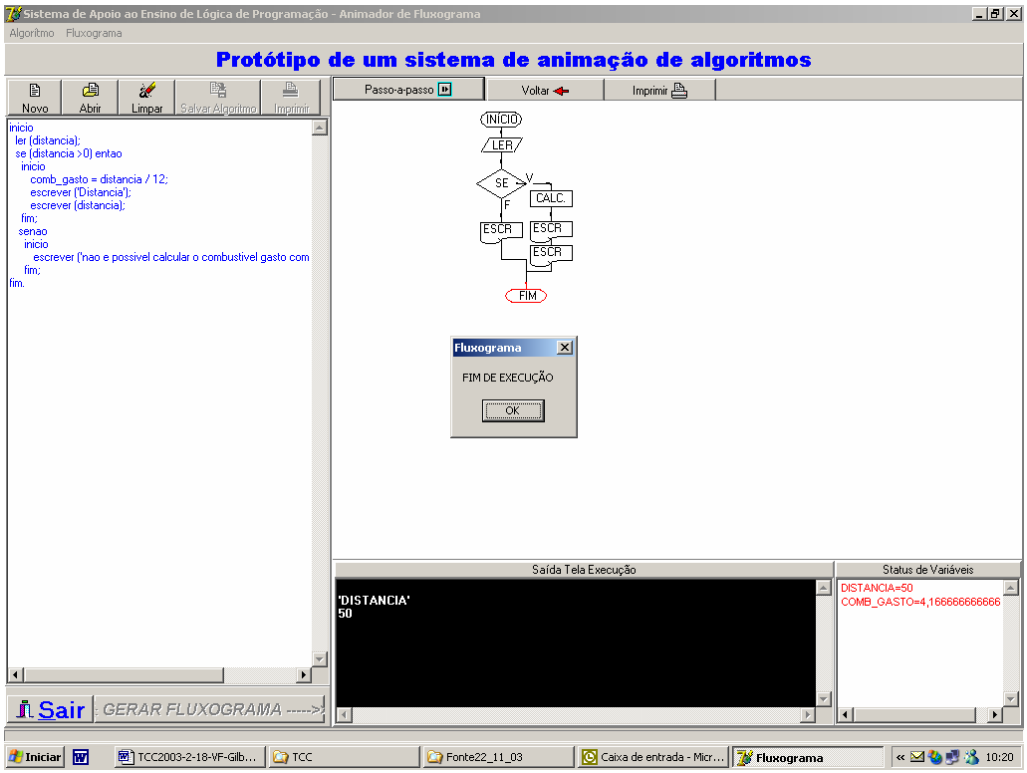


FIGURA 18 – Exemplo do término da execução passo-a-passo

Para exemplificar o funcionamento do protótipo, irá ser pego um algoritmo gerado no protótipo HelpAlgo para atender um dos objetivos da ferramenta. O quadro abaixo apresenta o estudo de caso que Gubler (2002) utilizou e o algoritmo gerado em sua ferramenta.

A partir deste algoritmo vamos demonstrar passo-a-passo do inicio até o final da execução do fluxograma.

Excluído: 1¶

Estudo de Caso

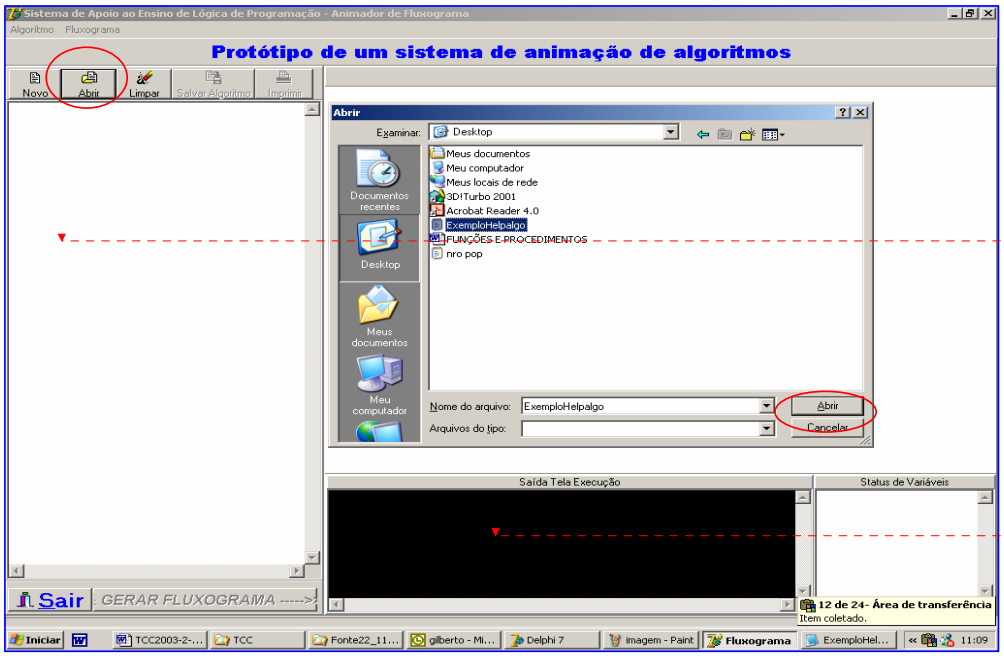
Calcular a quantidade de combustível gasto em uma viagem de um carro, onde o carro faz 12 kilometros por litro. O usuário vai informar qual a distância percorrida pelo carro na viagem, onde não poderá ser menor ou igual a zero.

Algoritmo Gerado

```
Inicio
      Ler (distancia);
      Se (distancia > 0) entao
      Inicio
            comb_gasto =   distancia / 12;
             Escrever (comb_gasto);
      Fim;
      Senao
       Inicio
            Escrever (' Não é possível calcular combustível gasto com uma distância igual a zero');
       Fim;
Fim.
```

QUADRO 3 – Descrição do estudo de caso

Na fig. 19 está sendo mostrada a tela principal do protótipo desenvolvido neste trabalho. Para começar a executar o sistema é necessário clicar no botão *Novo* ou *Abrir*. No exemplo abaixo abre-se um já existente.



Excluído: <sp>

Excluído: <sp>

FIGURA 19 – Demonstração da abertura de um arquivo

Excluído: 1¶

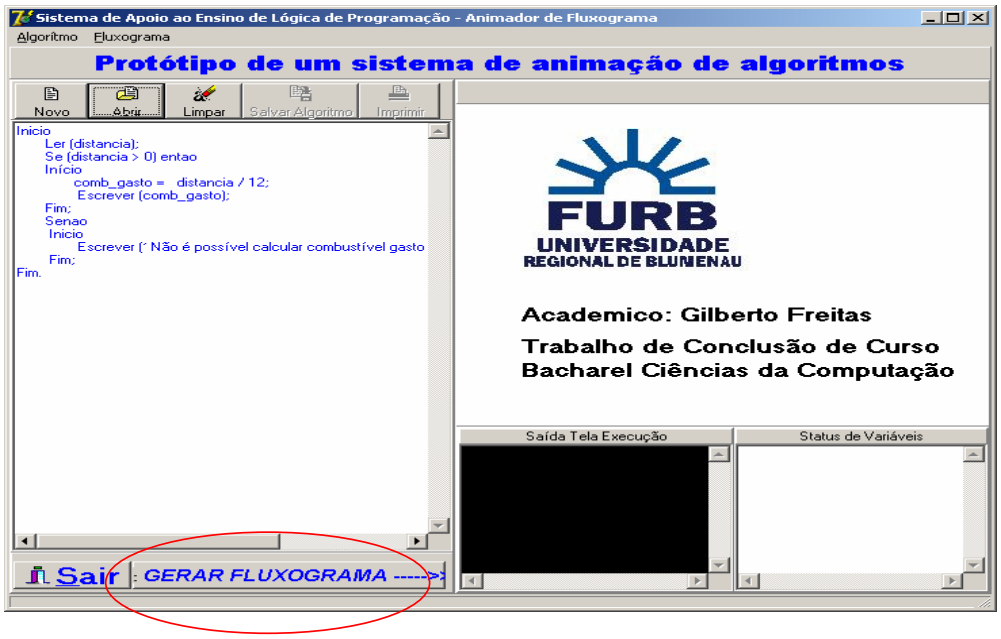


FIGURA 20 – Algoritmo aberto e pronto para ser executado

Com o arquivo carregado no campo memo, ao pressionar-se o botão GERAR FLUXOGRAMA, será executado o desenho do fluxograma. A fig. 21 mostra destacado o fluxograma gerado neste exemplo.

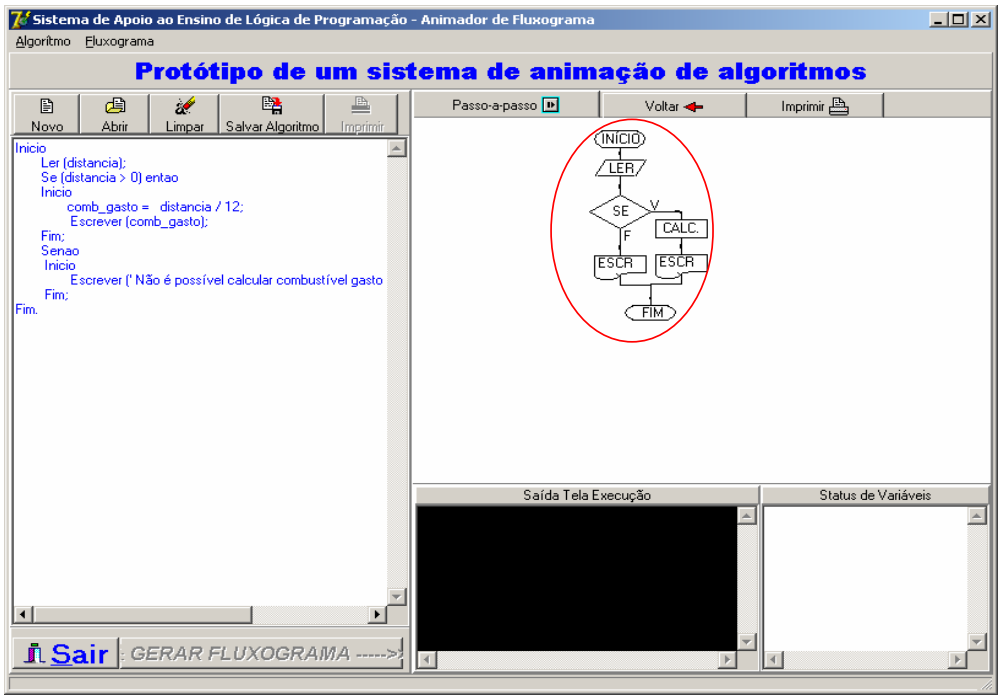


FIGURA 21 – Desenho da geração do fluxograma

Excluído: 1¶

A partir da geração do fluxograma selecionado em vermelho na fig. 21, será feito seu teste de mesa. Para isto, clica-se no botão PASSO-A-PASSO e como resposta a instrução que está sendo executada é pintada sua borda de vermelho até o componente FIM do fluxograma.

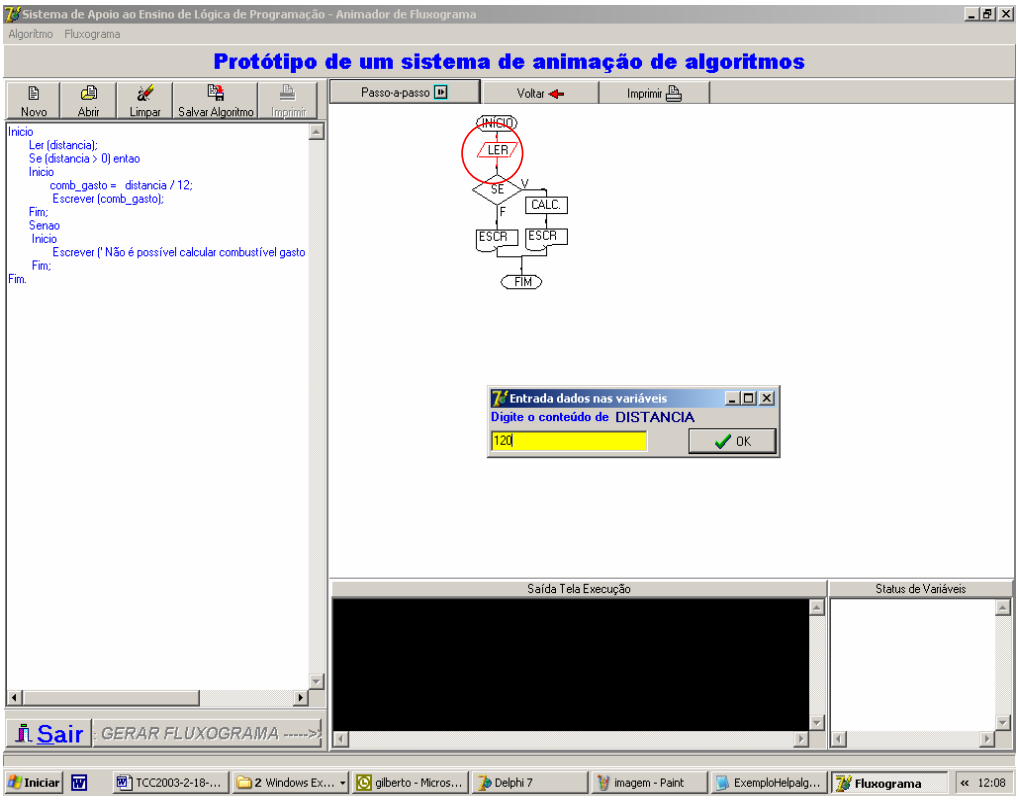


FIGURA 22 – Exemplo do andamento de uma execução passo-a-passo

Na hora em que o sistema encontra um pedido de leitura é aberta um caixa de diálogo para entrada com o valor da variável. Se a variável não tinha sido declarada anteriormente, ela é criada dinamicamente pelo sistema apresentando uma mensagem na tela.

Excluído: 1¶

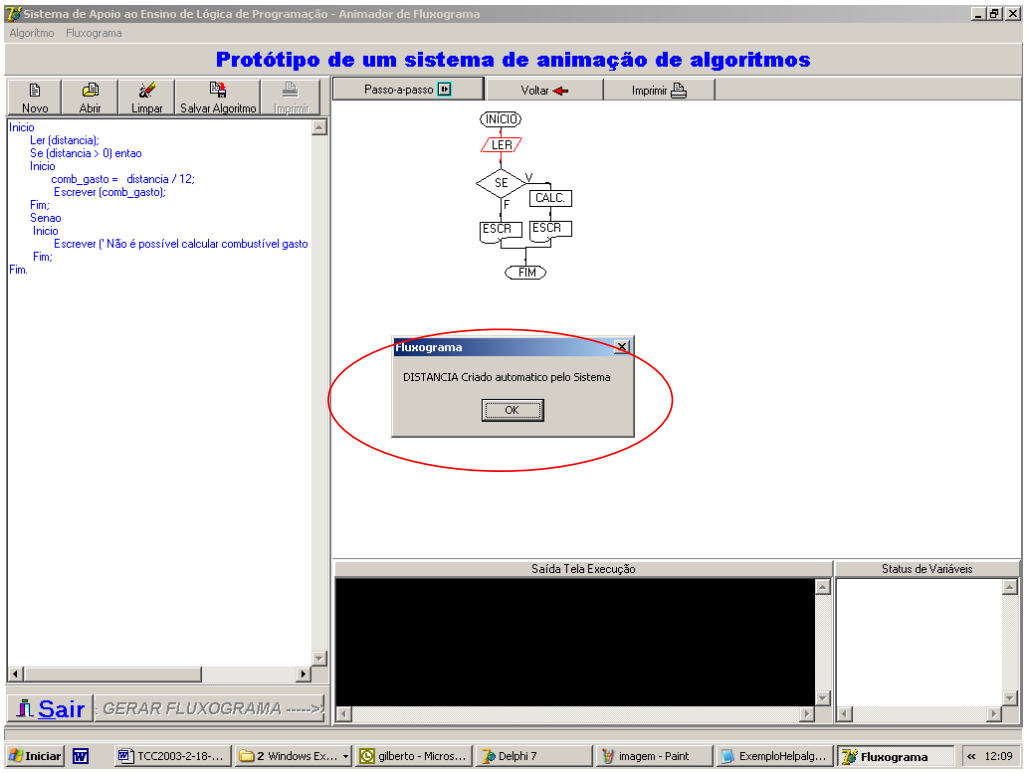


FIGURA 23 – Criação da variável ainda não usado no algoritmo

Excluído: ao

Nos cálculos também é feito da mesma forma, se a variável do lado esquerdo do sinal de igualdade ainda não tenha sido declarada, o sistema cria a variável e atribui o resultado da expressão calculada. Esta expressão é calculada somente quando as variáveis dentro dela estiverem dentro do *array variáveis*.

Excluído: já

Excluído: tiverem sido usadas anteriormente

Na fig. 23, é apresentada a execução depois de um cálculo, observa-se que existem dois campos *memo*; um tem a saída dos dados que são apresentados na tela (Saída Tela Execução) e outro é onde mostra sempre as variáveis existentes e seus conteúdos atualizados.

Excluído: *o*

Excluído: /;

Excluído:

No andamento da execução do algoritmo observa-se que:

a) na instrução “Ler (distância);” foi criada a variável distância e atribuído o valor de 120;

b) na instrução “Se (distancia > 0) então” entra-se no laço verdadeiro pois a distância é 120;

c) no “comb_gasto = distancia / 12;” é criada a variável comb_gasto e feito o cálculo

Excluído: 1¶

da expressão;

d) já no "escrever (comb_gasto);" é apresentado no *memo* "Saída Tela Execução" o valor da variável na tela.

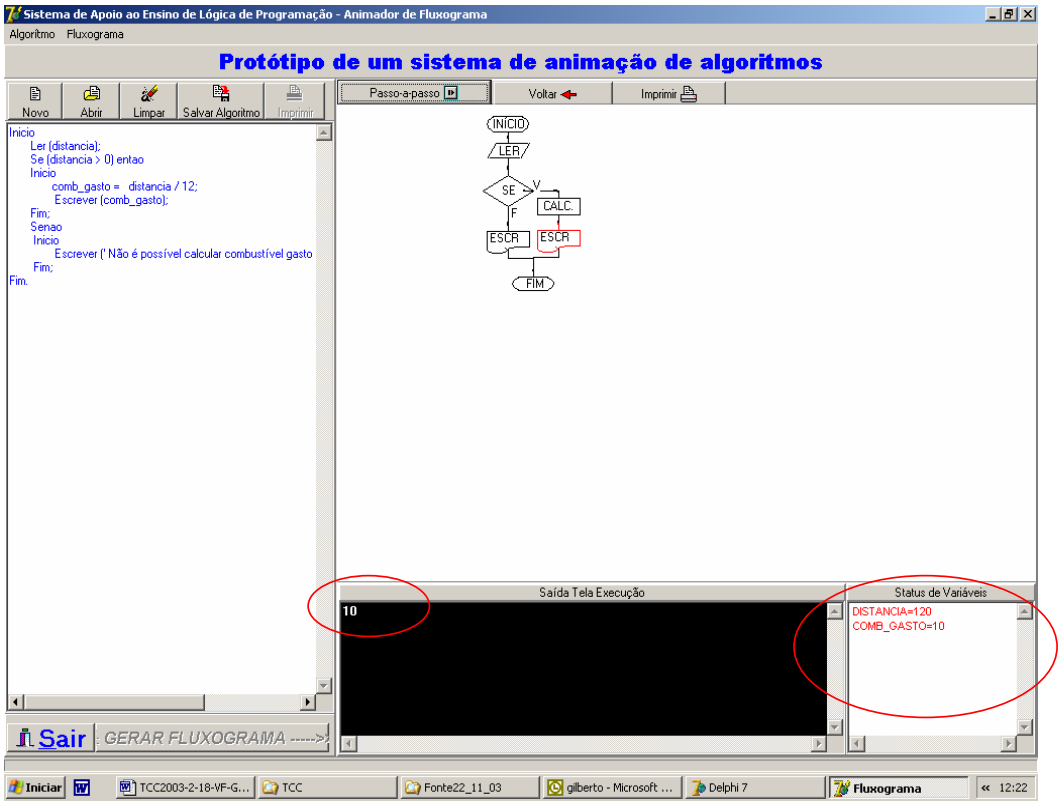


FIGURA 24 – Conteúdo das variáveis e saída impressão na tela

Como último passo é pintado o fluxo FIM, sendo apresentada a mensagem "FIM DA EXECUÇÃO". Para voltar a trabalhar com a edição do algoritmo clica-se no botão "VOLTAR".

3.4 RESULTADOS E DISCUSSÃO

Este trabalho, que é primeiramente uma extensão do trabalho desenvolvido em Mattos (1999), atingiu seus objetivos, conseguindo-se desenvolver um protótipo utilizando a ferramenta de desenvolvimento Delphi 7, a geração e execução de algoritmos baseando-se como sintaxe padrão algoritmo gerado no Helpalgo.

Excluído:

Excluído:

Excluído: 1¶

Com base no trabalho de Gubler (2002), na seção de extensões é aceita a sugestão de implementar um protótipo para demonstrar fluxogramas dos passos conseguidos das perguntas respondidas pelo usuário.

4 CONCLUSÕES

A partir do estudo de caso apresentado, constata-se que a ferramenta introduz uma nova forma alternativa de ensinar algoritmos através da geração automática de um fluxograma a partir do código fonte escrito em portugol. Esta facilidade associada ao sistema especialista que conduz o aluno no processo de desenvolvimento de algoritmos caracteriza-se como mais um recurso que o aluno pode ter na ausência do professor.

A partir dos trabalhos precedentes que compõem o conjunto de ferramentas, que fazem parte da idéia de criar-se uma ferramenta de aprendizado em lógica de programação, podem ser analisado os seguintes aspectos:

a) Mattos (1999), verificou que o índice de reprovação nesta disciplina é muito alto e imaginou que poderia usar sistemas especialistas não só para as outras áreas profissionais, mas também para o próprio aprendizado em computação. Como pioneiro, seu trabalho foi validado e vem sendo melhorado gradativamente;

b) Gubler (2002) e Heinzen (2002) continuaram o trabalho de Mattos (2000) a fim de desenvolver uma interface mais amigável e introduzir um módulo de raciocínio baseado em casos.

Neste contexto, o presente trabalho é relevante tendo em vista que complementa a ferramenta com a introdução de um módulo de resposta gráfico que permite ao aluno visualizar a solução lógica, bem como testá-la antes da geração final do código.

4.1 EXTENSÕES

Como extensões para este trabalho sugere-se:

a) introduzir um recurso para salvar também os fluxogramas;

b) inserir dentro do símbolo o comando executado;

c) implementar o suporte a Matrizes;

d) complementar o analisador léxico para que entenda outros tipos de palavras como, por exemplo: no lugar de LER aceitar Leia, no lugar de FIM dentro de um "SE" aceitar "FIM SE" ou "FIM ENTAO", entre outros;

e) integrar todos os módulos desenvolvidos até o momento em um único programa que inicie no sistema especialista e, após o teste de mesa, faça a geração de código-fonte e a compilação do programa produzido.

Excluído: 1¶

REFERÊNCIAS BIBLIOGRÁFICAS

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. Rio de Janeiro: Campus, 2002.

CARES, Paula Lorena Lovera. **Ambiente para teste de mesa utilizando fluxograma.** 2002. 92 f. Trabalho de Conclusão de Curso (Ciências da Computação) – Centro de Educação Superior de Ciências Tecnológicas, da Terra e do Mar, Universidade o Vale do Itajaí, Itajaí.

FERNANDES, Antonio Luiz B.; BOTINI, Joana. **Construção de algoritmos**. Rio de Janeiro: Senac Nacional, 1998.

FOWLER, Martin; SCOTT, Kendall. **UML essencial:** um breve guia para linguagem padrão de objetos. Porto alegre: Bookman, 2000.

FORBELLONE, André Luiz Villar; EBERSPACHER, Henri Frederico. **Lógica de programação:** a construção de algoritmos e estrutura de dados. São Paulo: Makron Books, 2000.

GENARO, Sergio. **Sistemas especialistas:** o conhecimento artificial. Rio de Janeiro: LTC, 1986.

GIARRATANO,J.C. & RILEY,G. **Expert System:** principles and programming. PWS Publishing Co, Boston, 1994.

GUBLER, André Iraldo. **Protótipo de um sistema especialista para auxiliar no ensino de algoritmos**. 2002. 55 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

GUIMARÃES, Ângelo de Moura; LAGES, Newton Alberto de Castilho. **Algoritmos e estrutura de dados**. Rio de Janeiro: LTC, 1985.

HARMON, Paul; KING, David. **Sistemas especialistas**. Rio de Janeiro: Campus , 1988.

HEINZEN, Luiz Ângelo. **Módulo de raciocínio baseado em casos em uma ferramenta de apoio ao ensino de lógica de programação.** 2002. 62f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

KELLER, Roberto. **Tecnologia de sistemas especialistas**: desenvolvimento e aplicação. São Paulo: McGraw-Hill, 1991.

MANZANO, José Augusto N. G.; OLIVEIRA, Jayr Figueiredo. **Algoritmos:** lógica para desenvolvimento e programação. São Paulo: Erica, 1996.

MATTOS, Mauro Marcelo; FERNANDES, Andrino; LÓPEZ, Oscar Ciro. **Sistema especialista para o apoio ao aprendizado de lógica de programação**. In: VII Congresso Iberoamericano de Educação Superior em Computação – CIESC99. 1999, Assunção, Paraguay.

MATTOS, Mauro Marcelo. **Construção de abstrações em lógica de programação** Anais, II Simpósio de Informática do Planalto Médio – SIPM'2000, Universidade de Passo Fundo, Passo Fundo,2000.

POSSAMAI, Roque Cesar. **Ferramenta de análise de código fonte em Delphi**. 2000. 71 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

PRESSMAN, Roger S. **Engenharia de Software.** Rio de Janeiro: McGraw-Hill, 2002.

SOUZA, Luiz Carlos S. **Protótipo de editor gráfico de fluxograma para representação de comandos de linguagem portugol.** 1999. 48 f. Trabalho de conclusão de curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

STEVE, Teixeira. **Delphi 6,** o guia do desenvolvedor**.** Rio de Janeiro: Campus, 2002.

VENÂNCIO, Cláudio Ferreira. **Desenvolvimento de algoritmos:** uma nova abordagem. São Paulo: Érico, 1997.

Excluído: 1¶

ANEXO 1 – Representação do formato de sintaxe analisado pelo protótipo

Tabela 1 – Modelos de sintaxe padrão do algoritmo portugol

TIPO DO COMANDO	SINTAXE	EXEMPLO
Cabeçalho	Programa [nome_do_programa];	Programa CalcComb;
Programa Principal	Inicio Bloco de Comandos; Fim.	Inicio Ler (distancia); Escreva (distancia); Fim.
Entrada	LER ([variável]);	Ler (distancia);
Saída	ESCREVER ([variável]); ESCREVER (‘literal’);	Escrever(distancia) ; Escrever (‘Não é possível calcular’);
Atribuição	[variável] =[variável2] [variável]=[variável + valor]; [variável] = [calculo expressão];	Calculo = Valor; Cont_valor=Cont_valor+1; Media = (N1 + N2)/2;
Condição	Se (condição_Verdadeira) entao Inicio Bloco de comandos verdadeiro; Fim; Senao Inicio Bloco de comandos Falso; Fim;	Se (distância > 0) então Inicio Comb_gasto = distancia/12; Fim; Senao Inicio Escrever (‘Não é possivel calcular’); Fim;
Enquanto	Enquanto (condição Verdadeira) faca Inicio Bloco de Comandos V; Fim;	Enquanto (distancia<500) faca Inicio Escrever (distancia); Fim;

Excluído: ¶
ANEXO 1 – Representação do formato de sintaxe analisado pelo protótipo¶
Tabela 1 – Modelos de sintaxe padrão do algoritmo portugol¶
TIPO DO COMANDO ... [2]

Excluído: ¶

Página 10: [1] Excluído Gilberto Freitas 03/02/2004 20:15:00

Página 56: [2] Excluído **Gilberto Freitas** **03/02/2004 21:17:00**

ANEXO 1 – Representação do formato de sintaxe analisado pelo protótipo

Tabela 1 – Modelos de sintaxe padrão do algoritmo portugol

TIPO DO COMANDO	SINTAXE	EXEMPLO
Cabeçalho	Programa [nome_do_programa];	Programa CalcComb;
Programa Principal	Inicio Bloco de Comandos; Fim.	Inicio Ler (distancia); Escreva (distancia); Fim.
Entrada	LER ([variável]);	Ler (distancia);
Saída	ESCREVER ([variável]); ESCREVER ('literal');	Escrever (distancia); Escrever ('Não é possível calcular');
Atribuição	[variável] =[variável2]; [variável]=[variável + valor]; [variável] = [calculo expressão];	Calculo = Valor; Cont_valor=Cont_valor+1; Media = (N1 + N2)/2;

Condição	Se (condição_Verdadeira) entao Inicio Bloco de comandos verdadeiro; Fim; Senao Inicio Bloco de comandos Falso; Fim;	Se (distância > 0) então Inicio Comb_gasto = distancia/12; Fim; Senao Inicio Escrever (‘Não é possivel calcular’); Fim;
Enquanto	Enquanto (condição Verdadeira) faca Inicio Bloco de Comandos V; Fim;	Enquanto (distancia<500) faca Inicio Escrever (distancia); Fim;