UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE UM SISTEMA ESPECIALISTA PARA AUXILIAR NO ENSINO DE ALGORITMOS

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

ANDRÉ IRALDO GUBLER

BLUMENAU, JUNHO/2002

2002/1-09

PROTÓTIPO DE UM SISTEMA ESPECIALISTA PARA AUXILIAR NO ENSINO DE ALGORITMOS

ANDRÉ IRALDO GUBLER

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Mauro Marcelo Mattos — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Mauro Marcelo Mattos

_____

Prof. Paulo Roberto Dias

_____

Prof. Jomi Fred Hübner

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

RESUMO

Este trabalho apresenta uma proposta de metodologia de suporte ao ensino do conteúdo de lógica de programação em turmas introdutórias, a qual vem sendo desenvolvida como projeto de pesquisa na FURB. O problema é contextualizado e são apresentadas extensões a um protótipo que fora anteriormente desenvolvido em CLIPS. A nova versão desenvolvida neste trabalho, foi convertida para Delphi-Pascal visando obter-se uma ferramenta com um melhor apelo visual para facilitar o emprego em sala-de-aula, e também de mais fácil manutenção

ABSTRACT

This work describes some extensions that has been proposed to an ougoing research project at Furb that aims to build a tool to help students to learn programming logics exercises. The problem is characterized and a new methodology is presented. An expert system tool that was developed in CLIPS to validate the work is presented and a new version converted to Delphi-Pascal is presented. This new version has a better user-interface and is easier to be fixed than the former one.

1 INTRODUÇÃO

De acordo com Forbellone (2000) a lógica é a arte de bem pensar, que é a ciência das formas do 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 administrativas, 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.

Segundo Forbellone (2000) a lógica de programação significa o uso correto das leis do pensamento, da ordem da razão e de processos de raciocínio e simbolização formais na programação de computadores, objetivando racionalidade e o desenvolvimento de técnicas que cooperam para a produção de soluções logicamente válidas e coerentes, que resolvam com qualidade os problemas que se deseja programar.

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 algoritmos (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 a disciplina de algoritmos verificou que a maioria deles não possuía conhecimentos abstratos de áreas cientificas (matemática, física, biologia, historia, geografia...), pois foram alunos de 2º grau. Quando apresentados à descrição textual dos enunciados dos problemas nesta disciplina introdutória, na maioria dos casos, os alunos 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 alunos da disciplina de algoritmos (lógica de programação).

Um acompanhamento realizado por Mattos (1999) sobre as turmas de algoritmos durante o período compreendido entre o primeiro semestre de 1996 e segundo semestre de 1998, 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. 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.

Foi apresentada em Mattos (1999) uma primeira proposta para introduzir o conceito de análise de requisitos já nas primeiras fases do ensino de computação, onde foi desenvolvida uma ferramenta didática baseada em sistemas especialistas.

Segundo Heinzle (1995), um sistema especialista é um sistema computacional projetado e desenvolvido para solucionar problemas que normalmente exigem um especialista humano com conhecimento na área de domínio da aplicação. Tal como um especialista o sistema deve ser capaz de emitir decisões justificadas acerca de um determinado assunto a partir de uma substancial base de conhecimento. Para tomar uma decisão o especialista busca em sua memória conhecimentos prévios, formula hipóteses, verifica os fatos que encontra e compara-os com as informações já conhecidas e então emite a decisão. Neste processo, o especialista realimenta a sua "base de conhecimentos" acerca do assunto.

Para implementar o protótipo desenvolvido em Mattos (1999) foi utilizado como ambiente de desenvolvimento o Clips (*C Language Integrated Production System*) o qual foi desenvolvido pela NASA/Johnson Space Center. Segundo Giarratano (1994), Clips é uma linguagem de programação multiparadigma a qual fornece suporte a programação tradicional, ou seja, baseada em procedimentos, programação orientada a objetos e baseado em regras.

Pode-se considerar que, apesar da implementação do protótipo atender os 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, a interface com o usuário foi implementada no modo caractere que nos dias de hoje gera alguma dificuldade para alunos

iniciantes no curso de ciências da computação que estão mais acostumados a utilizar softwares com uma interface gráfica, como exemplo o sistema operacional Windows. O. trabalho desenvolvido em Mattos (1999) também não implementa todas as estruturas de algoritmos, assim sendo a ferramenta não completa em relação ao assunto de algoritmos.

Como continuação do trabalho desenvolvido em Mattos (1999) desenvolveu-se 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 havia sido implementado.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é, portanto complementar o trabalho desenvolvido em Mattos (1999) agregando além de uma interface gráfica com usuário, um módulo para construção de estruturas de repetição.

Os objetivos específicos do trabalho são:

a) implementação de um protótipo desenvolvido na linguagem de programação Object-Pascal no ambiente de desenvolvimento Delphi, visando converter o protótipo desenvolvido por Mattos(1999) no ambiente de desenvolvimento Clips para uma interface gráfica.

b) incorporação do tratamento de estruturas de repetição no novo protótipo desenvolvido;

c) construção de uma ferramenta que contemple a especificação apresentada em Mattos (1999), a qual possa ser utilizada em sala de aula.

1.2 ESTRUTURA DO TRABALHO

O trabalho está organizado da seguinte forma: o segundo capítulo refere-se a uma introdução a algoritmos, as formas de representação dos algoritmos, as estruturas básicas dos algoritmos.

O terceiro capítulo apresenta alguns conceitos e paradigmas de informática na educação, os tipos de softwares educativos, os problemas no aprendizado de algoritmos e os trabalhos já propostos.

O quarto capítulo apresenta alguns conceitos, definições, características, componentes, histórico da sua evolução e ainda as principais formas de representação do conhecimento de sistemas especialistas.

O quinto capítulo enfoca as principais características do trabalho elaborado em Mattos (1999).

O sexto capítulo apresenta a especificação e o desenvolvimento do protótipo deste trabalho.

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

2 ALGORITMOS

Algoritmo com certeza é um assunto muito importante na computação, pois para implementar qualquer software, por mais simples que ele seja, é necessário construir um algoritmo utilizando alguma linguagem de programação. Portanto o conceito central da programação e da ciência da computação é o algoritmo.

2.1 INTRODUÇÃO A ALGORITMOS

De acordo com Forbellone (2000), quando estamos elaborando um algoritmo devemos especificar ações claras e precisas, que a partir de um estado inicial, após um período de tempo finito, produzem um estado final previsível e bem definido. Isto significa que o algoritmo fixa um padrão de comportamento a ser seguida, uma norma de execução a ser trilhada, com vistas a alcançar, como resultado final, a solução de um problema, garantindo que sempre que executado, sob as mesmas condições, produza o mesmo resultado.

Ao contrário do que se pode pensar, o conceito de algoritmos não foi criado para satisfazer às necessidades da computação. Pelo contrário, a programação de computadores é apenas um dos campos de aplicação dos algoritmos. Na verdade, há inúmeros casos que podem exemplificar o uso (involuntário ou não) de algoritmos para a padronização do exercício de tarefas rotineiras. Como exemplo temos uma receita de bolo, onde está descrita uma série de ingredientes necessários e uma seqüência de diversos passos que devem ser fielmente cumpridos para que se consiga fazer o bolo, conforme se esperava antes do início das atividades, ou seja, um objetivo bem definido.

Um algoritmo tem por objetivo representar mais fielmente o raciocínio envolvido na lógica de programação e, dessa forma, permite-nos abstrair de uma série de detalhes computacionais, que podem ser acrescentados mais tarde. Assim, podemos focalizar nossa atenção naquilo que é importante: a lógica da construção de algoritmos. Outra importância da construção dos algoritmos é que, uma vez concebida uma solução algorítmica para um problema, esta pode ser traduzida para uma da muitas linguagens de programação.

2.2 ESTRUTURAS BÁSICAS DE ALGORITMOS

A seguir são apresentadas as estruturas básicas para a implementação de algoritmos.

2.2.1 TIPOS PRIMITIVOS

De acordo com Forbellone (2000), deve-se observar um conceito muito importante em relação aos tipos primitivos, a informação. A informação é a matéria prima que faz com que seja necessária a existência dos computadores, pois eles são capazes de manipular e armazenar um grande volume de dados com alta performance, liberando o ser humano para outras tarefas nas quais seu conhecimento é indispensável.

Há basicamente quatro tipos primitivos:

a) **inteiro:** toda e qualquer informação numérica que pertença ao conjunto dos números inteiros relativos (negativa, nula e positiva);

b) **real:** toda e qualquer informação numérica que pertença ao conjunto dos números reais (negativa, nula ou positiva);

c) **caractere:** toda e qualquer informação composta por um conjunto de caracteres alfanuméricos: numéricos (0...9), alfabéticos (A-Z,a-z) e especiais (exemplo: #, ?, &, @, !).

d) **lógico:** toda e qualquer informação que pode assumir apenas dois valores (exemplo: verdadeiro e falso).

2.2.2 CONSTANTES

Entende-se com uma constante quando ela não sofre nenhuma variação no decorrer do tempo, ou seja, seu valor é constante desde o início até o fim da execução do algoritmo.

2.2.3 VARIÁVEL

Uma variável é quando o valor dela tem a possibilidade de ser alterado em algum instante no decorrer do tempo, ou seja, durante a execução do algoritmo em que é utilizado, o valor do dado sofre alteração ou o dado é dependente da execução em um certo momento ou circunstância. Como exemplo de uma variável onde fica guardado o valor da média de alunos.

2.2.4 EXPRESSÕES ARITMÉTICAS

É denominada expressão aritmética aquela cujos operadores são aritméticos e cujos operandos são constantes ou variáveis do tipo numérico (inteiro ou real).

2.2.5 OPERADORES ARITMÉTICOS

Chama-se de operadores aritméticos o conjunto de símbolo que representa operações básicas da matemática.

No quadro 1 são apresentados alguns operadores aritméticos:

Quadro 1 – Operadores aritméticos

Operador	Função	Exemplos
+	Adição	3 + 1, Z + Y
-	Subtração	10 – 5, X - Z
*	Multiplicação	2 * 1, M * N
/	Divisão	20 / 10, A / B
mod	Resto da divisão	9 mod 1 = 1
Div	Quociente da divisão	27 div 5 = 5

Fonte: Forbellone(2000)

O programador tem plena liberdade de introduzir novos operadores ou nomes de funções para adaptar a linguagem às necessidades específicas da sua área de aplicação, ou seu problema, sempre que eles estejam bem definidos, sem deixar margem a ambigüidades.

2.2.6 EXPRESSÕES LÓGICAS

É denominada expressão lógica aquela cujos operadores são lógicos ou relacionais e cujos operandos são relações ou variáveis ou constantes do tipo lógico.

2.2.7 OPERADORES RELACIONAIS

Os operadores relacionais são utilizados para realizar comparações entre dois valores de mesmo tipo primitivo, que são representados por constantes, variáveis ou expressões aritméticas. No quadro 2 são apresentados alguns operadores relacionais:

Quadro 2 – Operadores relacionais

Operador	Função	Exemplos
=	Igual a	2 = 2, X = Y
>	Maior que	10 > 5, M > N
<	Menor que	1 < 2, A < B
>=	Maior ou igual a	3 > 1, Z > X
<=	Menor ou igual a	5 < 6, Z < Y
<>	Diferente de	1 <> 2, X <> Y

Fonte: Forbellone(2000)

2.2.8 OPERADORES LÓGICOS

Utiliza-se para a formação de novas proposições lógicas compostas a partir de outras proposições lógicas simples três conectivos, que são apresentados abaixo no quadro 3.

Quadro 3 – Operadores lógicos

Operador	Lógica
não	Negação
e	Conjunção
ou	Disjunção

Fonte: Forbellone(2000)

2.2.9 ATRIBUIÇÃO

Um comando de atribuição permite-nos fornecer um valor a uma variável, em que o tipo dever ser compatível com o tipo da variável, isto é, somente podemos atribuir um valor lógico a uma variável capaz de comportá-lo, ou seja, uma variável declarada do tipo lógico.

No quadro 4 é apresentado um exemplo de representação em pseudocódigo de atribuição:

Quadro 4 – Representação de atribuição

```
{IDENTIFICADOR} + {←} + {EXPRESSÃO} + “;”
```

Fonte: Forbellone(2000)

2.2.10 BLOCOS

Segundo Guimarães (1994), um bloco pode ser definido como um conjunto de comandos com uma função bem definida. Ela serve também para definir os limites onde as

variáveis declaradas em seu interior são conhecidas. Um bloco é delimitado por um **Inicio** e por um **Fim**.

2.2.11 ENTRADA E SAÍDA

De acordo com Forbellone (2000), os algoritmos precisam ser alimentados com dados para efetuarem as operações e cálculos que são necessários a fim de alcançar o resultado desejado. Uma analogia desse processo com uma atividade que é corriqueira, que é fazer um bolo, onde também são seguidos os mandamentos da informática. Como entrada, se tem os ingredientes que serão processados segundo um algoritmo, a receita, e no final se tem como saída o bolo pronto.

2.2.11.1 ENTRADA DE DADOS

Para entrada de dados que o algoritmo necessitar, é utilizado o comando "*leia*", cuja finalidade é atribuir o dado a ser fornecido à variável identificada. No quadro 5 estão representados na forma de pseudocódigo alguns exemplos:

Ouadro 5 – Entrada de dados

```
Leia (Z);
Leia (A, B, Nota);
```

Fonte: Guimarães(1985)

2.2.11.2 SAIDA DE DADOS

Para que o algoritmo possa mostrar os dados que calculou, como resposta ao problema que resolveu, é adotado um comando de saída de dados denominado "e*screva"*, cuja finalidade é exibir o conteúdo da variável identificada. No quadro 6 estão representados na forma de pseudocódigo alguns exemplos:

Quadro 6 – Saída de dados

```
Escreva (media);
Escreva ("Média do aluno :", media);
```

Fonte: Guimarães (1985)

2.2.12 ESTRUTURA SEQÜENCIAL

De acordo com Forbellone (2000), uma estrutura seqüencial de um algoritmo corresponde ao fato de que o conjunto de ações primitivas será executado em uma seqüência

linear de cima para baixo e da esquerda para a direita, isto é, na mesma ordem em que foram escritas. As ações serão seguidas por um ponto-e-vírgula(;), que objetiva separar uma ação da outra e auxiliar a organização seqüencial das ações, pois após encontrar um ponto-e-vírgula(;) devera-se executar o próximo comando da seqüência.

2.2.13 ESTRUTURA DE CONDIÇÃO

Uma estrutura de condição permite a escolha de um grupo de ações (Bloco) a ser executado quando determinadas condições, representadas por expressões lógicas ou relacionais, são ou não satisfeitas.

2.2.13.1 ESTRUTURA DE CONDIÇÃO SIMPLES

Quando é preciso testar uma certa condição antes de executar uma ação, é utilizado uma seleção simples, que é representado em forma de pseudocódigo no quadro 7:

Quadro 7 – Condição simples

```
se <condição> então
inicio
     <seqüência de comandos>;
fim;
```

Fonte: Forbellone(2000)

A "*<condição>*" é uma expressão lógica que quando inspecionada, pode gerar um resultado falso ou verdadeiro. Se a condição for verdadeira, será executada a seqüência de comandos que se encontra dentro da estrutura de condição, e se for falsa será executado o comando abaixo da estrutura de condição se houver.

2.2.13.2 ESTRUTURA DE CONDIÇÃO COMPOSTA

Quando existem situações em que duas alternativas dependem de uma mesma condição, uma condição ser verdadeira e outra condição for falsa, é utilizado a estrutura de seleção composta. Supondo que um conjunto de ações dependa da avaliação verdadeiro e outro conjunto de ações dependa da avaliação falso da condição, é utilizado a estrutura representada na forma de pseudocódigo no quadro 8:

Quadro 8 – Condição composta

```
se <condição> então
inicio
     <seqüência de comandos>;
fim
senão inicio
     <seqüência de comandos>;
fim;
```

Fonte: Forbellone(2000)

Pode observar que a existência do bloco verdadeiro continua, sendo que este será executado caso a condição seja verdadeira, mas se a condição for falsa, tem-se a execução da seqüência de comandos do bloco senão da condição.

2.2.14 ESTRUTURA DE REPETIÇÃO

As estruturas de repetição consistem em utilizar uma seqüência de comandos novamente, onde a quantidade de vezes será determinada de acordo com uma condição especificada na estrutura de repetição. O trecho do algoritmo que serão repetidos tem como nome de laço de repetição. O número de repetições pode ser indeterminado, porém necessariamente finito.

2.2.14.1 REPETIÇÃO COM TESTE NO INÍCIO

Consiste em uma estrutura de controle do fluxo de execução que permite varias vezes repetir um mesmo trecho de algoritmo, porém sempre verificando a condição antes de cada execução. É utilizado à estrutura *"enquanto"* para realizar a repetição com teste no início, que permite que um bloco ou uma ação primitiva seja repetido enquanto uma determinada condição dor verdadeira. O modelo genérico deste tipo de repetição é representado na forma de pseudocódigo no quadro 9:

Quadro 9 – Repetição com teste no início

```
enquanto <condição> faça
inicio
     <seqüência de comandos>;
fim;
```

Fonte: Forbellone(2000)

Quando o resultado da condição for falso, o comando de repetição é abandonado. Se já da primeira vez o resultado é falso, os comandos não são executados nenhuma vez.

2.2.14.2 REPETIÇÃO COM TESTE NO FINAL

Para realizar a repetição com teste no final, é utilizado à estrutura *"repita"*, que permite que um bloco ou ação primitiva seja repetido até que uma determinada condição seja verdadeira. No quadro 10 é representado na forma de pseudocódigo um modelo genérico para este tipo de repetição:

Quadro 10 – Repetição com teste no final

```
repita
     <seqüência de comandos>;
até <condição>;
```

Fonte: Forbellone(2000)

Pela sintaxe desta estrutura, pode-se observar que a seqüência de comandos é executada pelo menos uma vez, independente da validade da condição. Isto ocorre porque a inspeção da condição é feita após a execução do bloco de comandos.

2.2.14.3 REPETIÇÃO COM VARIÁVEL DE CONTROLE

A estrutura de repetição com variável de controle é representada pela estrutura *"para"*, que repete a execução do bloco um número definido de vezes, pois possui limites fixos. O quadro 11 representa na forma de pseudocódigo um modelo genérico para a estrutura *"para"*.

Quadro 11 – Repetição com variável de controle

```
Para V de VI até VF passo P faça
inicio
     <seqüência de comandos>;
fim;
```

Em que:
V é a variável de controle
VI é o valor inicial da variável **V**
VF é o valor final da variável **V**, ou seja, até o valor que ela vai chegar.
P é o valor do incremento dado à variável **V**;

Fonte: Forbellone(2000)

2.3 FORMAS DE REPRESENTAÇÃO DE ALGORITMOS

Segundo Saliba (1993) um algoritmo pode ser representado por diversas formas, mas não há um consenso com relação à melhor delas. Algumas formas de representação tratam os problemas apenas em nível lógico, abstraindo-se de detalhes de implementação muitas vezes relacionadas com alguma linguagem de programação específica. Por outro lado, existem formas de representação de algoritmos que possuem uma maior riqueza de detalhes e muitas vezes acabam obscurecendo a idéia principal, o algoritmo.

Dentre as formas de representação de algoritmos mais conhecidas tem-se a descrição narrativa, o fluxograma convencional e o pseudocódigo, também conhecido como linguagem estruturada ou portugol.

2.3.1 DESCRIÇÃO NARRATIVA

Nesta forma de representação os algoritmos são expressos diretamente em linguagem natural. No quadro 12 é mostrado um exemplo de representação de um algoritmo em linguagem natural:

Quadro 12 – Exemplo de linguagem natural

```
Troca de pneu furado:
  1-afrouxar ligeiramente as porcas;
  2-suspender o carro;
  3-retirar as porcas e o pneu;
  4-colocar o pneu reserva;
  5-apertar as porcas;
  6-abaixar o carro;
  7-dar o aperto final nas porcas.
```

Fonte: Manzano(1996)

2.3.2 FLUXOGRAMA CONVENCIONAL

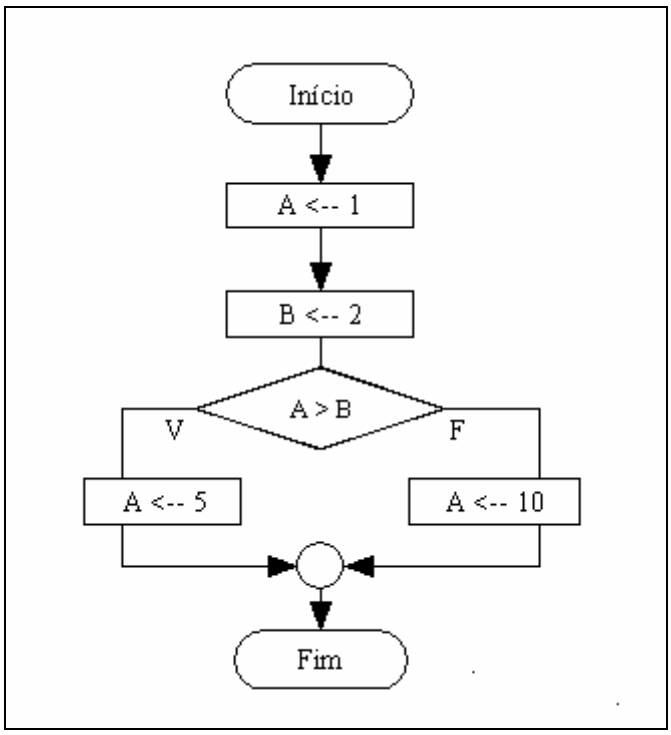
É uma representação gráfica de algoritmos onde formas geométricas diferentes implicam ações (instruções, comandos) distintos. Tal propriedade facilita o entendimento das idéias contidas nos algoritmos e justifica sua popularidade.

De modo geral, um fluxograma se resume a um único símbolo inicial, por onde a execução do algoritmo começa, e um ou mais símbolos finais, que são pontos onde a execução do algoritmo encerra. Partindo do símbolo inicial, há sempre um único caminho

orientado a ser seguido, representando a existência de uma única seqüência de execução das instruções. Isto pode ser mais bem visualizado pelo fato de que, apesar de vários caminhos poderem convergir para uma mesma figura do diagrama, há sempre um único caminho saindo desta. Exceções a esta regra são os símbolos finais, dos quais não há nenhum fluxo saindo, e os símbolos de decisão, de onde pode haver mais de um caminho de saída, representando uma bifurcação no fluxo.

Na fig. 1 é mostrado um exemplo de fluxograma convencional:

Figura 1 – Exemplo de fluxograma convencional



Fonte: Guimarães(1985)

2.3.3 PSEUDOCÓDIGO

Esta forma de representação de algoritmo é rica em detalhes, como a definição dos tipos das variáveis usadas no algoritmo e, por assemelhar-se bastante à forma que os programas são escritos, encontra muita aceitação.

Na verdade, esta representação é suficientemente geral para permitir a tradução de um algoritmo nela representado para uma linguagem de programação específica seja praticamente

direta. No quadro 13 é apresentado um exemplo de representação de algoritmos em pseudocódigo:

Quadro 13 – Exemplo de pseudocódigo

```
//Calculo da média de um aluno.
Algoritmo media;
Var N1, N2, Media : real;
Inicio
   Leia N1, N2;
   Media ← (N1 + N2) / 2;
   Se Media > 7 Entao
      Escreva (‘Aprovado’);
   Senao
      Escreva (‘Reprovado’);
Fim.
```

Fonte: Manzano(1996)

3 INFORMÁTICA NA EDUCAÇÃO

Com o crescente processo de transformação, que ocorre a cada instante na área da informática, é preciso atualizar os conhecimentos, para o desenvolvimento de novas técnicas de aprendizagem. Uma dessas técnicas é o desenvolvimento de softwares educativos.

Segundo Mattos (1999), todos têm os mesmos instrumentos para chegar ao conhecimento, mas não os utilizam com a mesma intensidade. Normalmente, os processos educacionais se baseiam, quase exclusivamente, no desenvolvimento da inteligência lingüística e lógico-matemática, deixando de lado as outras formas de acesso ao conhecimento.

O uso do computador na educação tem causado uma grande mudança no processo de ensino-aprendizagem. Por modismo ou por consciência, a informática tem adentrado as escolas.

De acordo com Silva (2000), a informática na educação no Brasil nasceu a partir do interesse de educadores de algumas universidades brasileiras, motivados pelo que já vinha acontecendo em outros países como nos Estados Unidos e na França. Embora o contexto mundial de uso do computador na educação sempre foi uma referencia para as decisões que foram tomadas no Brasil, sua caminhada é muito particular e difere daquilo que se faz em outros paises. Apesar das inúmeras diferenças, os avanços pedagógicos conseguidos no Brasil através da informática são quase os mesmos que em outros países.

3.1 PARADIGMAS DA INFORMÁTICA NA EDUCAÇÃO

“É importante mencionar que dependendo do paradigma utilizado em informática aplicada à educação, instrucionista ou construcionista, o profissional terá um papel mais ou menos relevante” (Valente, 1993).

Segundo Silva (2000), para o paradigma instrucionista o computador pode ser usado na educação como máquina de ensinar ou ferramenta. O uso do computador como máquina de ensinar na informatização dos métodos de ensino tradicionais. Alguém implementa no computador uma série de informações, que devem ser passadas ao aluno na forma de um tutorial, exercício-e-prática ou jogo. Entretanto, é muito comum encontrar-se essa abordagem

sendo usada como uma abordagem construtiva, ou seja, para propiciar a construção do conhecimento na “cabeça” do aluno.

De acordo com Silva (2000) o paradigma construcionista significa o uso do computador como meio para propiciar a construção do conhecimento pelo aluno, ou seja, o aluno interagindo com o computador na resolução de problemas, tema a chance de construir seu conhecimento. O conhecimento não é passado para o aluno, mas ele que é o construtor do seu próprio conhecimento. Portanto o paradigma construcionista enfatiza a aprendizagem ao invés de destacar o ensino; a construção e não a instrução.

3.2 TIPOS DE SOFTWARES EDUCATIVOS

Os softwares educativos devem ser considerados em relação ao aspecto pedagógico, ou seja, ao que ele propõe a ensinar e como isso é feito. Tem-se como pressuposto que a elaboração de um software educativo deva ter o acompanhamento de um professor qualificado para atuar na área de abrangência do software.

De acordo com Franciosi (1994), quanto á maneira como o ensino pela informática ocorre, o software educativo pode ser classificado em três grandes categorias:

- **instrução auxiliada por computador**: é um conceito do tipo livro eletrônico, onde o conteúdo é apresentado através de uma seqüência estruturada de telas;
- **aprendizagem por descoberta**: um sistema deste tipo pode ou não ser interativo. Pode-se fazer uma simulação onde o usuário não possa interferir e também fazer uma simulação com intervenção do usuário, ou seja, um jogo.
- **Ferramentas para alunos e professores**: consiste no uso de aplicativos de uso geral (processadores de texto, gerenciadores de banco de dados, planilhas eletrônicas, etc.) e de linguagem de programação orientada ao ensino.

3.3 PROBLEMAS NO APRENDIZADO DE ALGORITMOS

“É grande a dificuldade de aprendizado nesta disciplina em todas as faculdades brasileiras. O mesmo acontecendo e em grau ainda maior nos cursos técnicos de informática no nível de segundo grau” (Sucheuski, 1996), com isto nesta disciplina tem alto índice de reprovação e desistência dos alunos.

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 algoritmos (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 está disciplina verificou que a maioria deles não possuía conhecimentos abstratos de áreas cientificas (matemática, física, biologia, historia, geografia...), pois foram alunos de 2º grau. Quando apresentados à descrição textual dos enunciados dos problemas nesta disciplina introdutória, na maioria dos casos, os alunos encontravam dificuldades em extrair as informações necessárias para iniciar a solução destes problemas.

No trabalho desenvolvido por Mattos (1999) também salientado 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 alunos da disciplina de algoritmos (lógica de programação).

Estas grandes dificuldades têm inspirado alguns trabalhos para tentar resolver ou pelo menos minimizar os problemas encontrados no aprendizado de algoritmos.

3.4 TRABALHOS JÁ PROPOSTOS

Para auxiliar no ensino de algoritmo já foram elaborados alguns trabalhos, como os trabalhos realizados por Tagliari (1996) e Schmitt (1998), onde foram desenvolvidos sistemas que contém os conceitos teóricos de algoritmos, exercícios resolvidos e possibilidade de se fazer teste de mesa.

Por Mattos (1999) foi desenvolvido um sistema especialista, aonde por meio de perguntas que o usuário responde é gerado passos para elaborar um algoritmo de um determinado problema, e por meio destas perguntas o aluno vai aprendendo a lógica de programação e depois de um tempo não seria mais necessário o sistema, pois o aluno aprendeu a lógica de programação e assim conseguindo elaborar os algoritmos sem a ajuda do sistema especialista.

3.5 MOTIVAÇÃO

Para o desenvolvimento deste trabalho teve-se como motivação o sistema especialista desenvolvido em Mattos (1999), onde o mesmo apresenta uma proposta de metodologia de suporte ao ensino de lógica de programação, validada através de um protótipo de um sistema especialista implementado em CLIPS. Segundo Mattos (1999), os alunos da disciplina de algoritmos avaliaram está proposta muito útil, principalmente porque estabelecia uma “receita de bolo”, ou seja, um procedimento a ser seguido para gerar o algoritmo de determinado problema.

Outra grande motivação para o desenvolvimento de um trabalho para o auxiliar no aprendizado de lógica de programação ou algoritmos, está relacionada diretamente com a perspectiva de oferecer uma ferramenta de apoio para minimizar as desistências e reprovações desta disciplina, que é muito grande.

4 SISTEMAS ESPECIALISTAS

Sistemas especialistas são uma das muitas aplicações da inteligência artificial, onde são implementados programas de computador planejados para adquirir e disponibilizar o conhecimento operacional de um especialista humano. São tradicionalmente vistos como sistemas de suporte à decisão, pois são capazes de tomar decisões como especialistas em diversas áreas. Sua estrutura reflete a maneira como o especialista humano arranja e faz inferência sobre o seu conhecimento.

Segundo Heinzle (1995), os sistemas especialistas são sistemas computacionais projetados e desenvolvidos para solucionarem problemas que normalmente exigem especialistas humanos com conhecimento na área de domínio da aplicação. Tal como um especialista o sistema deve ser capaz de emitir decisões justificadas acerca de um determinado assunto a partir de uma substancial base de conhecimentos. Para tomar uma decisão o especialista busca em sua memória conhecimentos prévios, formula hipóteses, verifica os fatos que encontra e compara-os com as informações já conhecidas e então emite a decisão. Neste processo o especialista realimenta a sua base de conhecimento acerca do assunto.

Segundo Ribeiro (1987) um sistema especialista deve, além de inferir conclusões, ter capacidade de aprender novos conhecimentos e, desse modo melhorar o seu desempenho de raciocínio e a qualidade de suas decisões.

4.1 ABORDAGEM HISTÓRICA

Segundo Heinzle (1995) na década de 1960 começaram os primeiros trabalhos nos sistemas que hoje são chamados de especialistas. Inicialmente pretendia-se construir máquinas inteligentes com grande raciocínio e solução de problemas. Imaginava-se que a partir de um pequeno conjunto de normas ou regras de raciocínio introduzidas num poderoso computador criariam-se sistemas de capacidade superior à humana. Não tardou para que os pesquisadores observassem o engano e verificassem as reais dimensões do trabalho.

De acordo com Rabuske (1995) durante a década de 1970, vários sistemas especialistas apareceram. Entre os que mais se destacaram estão o Mycin, um sistema para detecção e diagnósticos de doenças infecciosas, e o Prospector, um sistema para dar suporte a geólogos

na exploração mineral. Os especialistas baseados em regras de produção dominaram amplamente as experiências efetuadas.

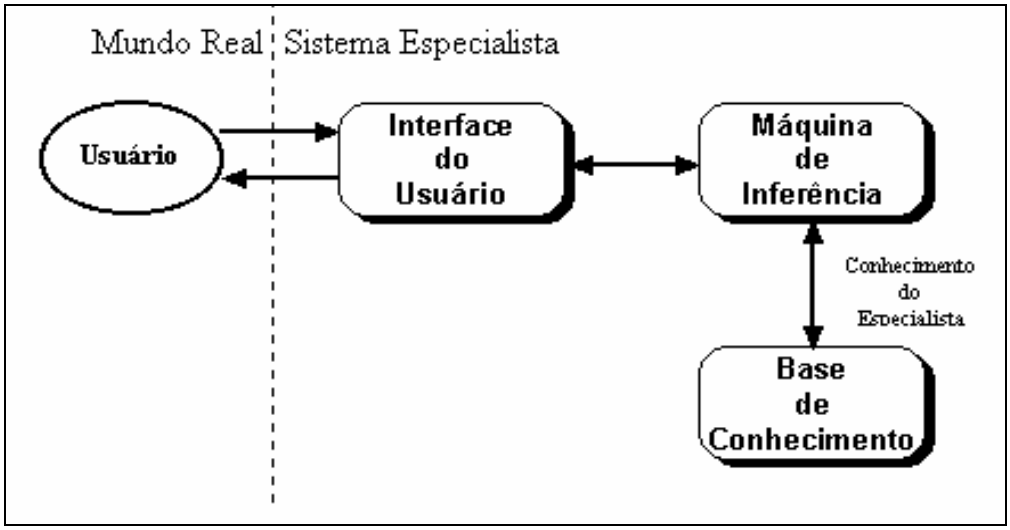
No Brasil, a Pontifícia Universidade Católica do Rio de Janeiro desenvolveu importantes trabalhos com sistemas especialistas. O principal resultado da universidade nesta área é um sistema chamado SAFO, cuja finalidade é a demonstração de teoremas matemáticos. Outra referencia histórica no Brasil, é o Instituto Militar de Engenharia que há alguns anos vem desenvolvendo sistemas de recuperação em grandes bases de conhecimento.

4.2 COMPONETES DE UM SISTEMA ESPECIALISTA

Um sistema especialista pode estar composto por vários elementos, que dependem de fatores com a generalidade pretendida, os objetivos do mesmo, a representação interna do conhecimento e as ferramentas usadas na implementação.

Um modelo básico da arquitetura dos sistemas especialistas pode ser apresentado como na fig. 2, com três componentes básicos: a base de conhecimento, a máquina de inferência, e a interface com usuário.

Figura 2 – Componentes de um sistema especialista



Fonte: Chaiben (1999)

4.3 INTERFACE COM O USUÁRIO

Segundo Chaiben (1999) à interface com o usuário tem a importância de facilitar a comunicação entre o sistema especialista e o usuário, onde permite a interação com o sistema,

através da entrada de fatos e dados e através da saída em forma de perguntas, conclusões e explicações.

Muitos princípios baseados nas teorias cognitivas têm sido propostos para projetos de interface, como resultado de pesquisas na área de interação homem-máquina. Uma das considerações principais no projeto de qual quer interface homem-máquina deve ser a facilidade de uso, reduzindo ao máximo a carga cognitiva sobre o usuário.

Portanto ter uma interface amigável com o usuário é muito importante para o usuário poder executar o sistema especialista sem qualquer dúvida, tendo assim uma interação com o sistema, e assim conseguindo resolver o seu problema que ele tiver em determinada área através do sistema especialista que vai auxiliar ele a resolver.

4.4 BASE DE CONHECIMENTO

Segundo Heinzle (1995) a base de conhecimento é o local onde se armazenam fatos, heurísticas, crenças, etc, ou seja, é um deposito de conhecimentos acerca de um determinado assunto. Este conhecimento é passado ao sistema pelo especialista e armazenado de uma forma própria que permite ao sistema fazer posteriormente o processamento ou inferência.

Um dos problemas mais sérios, e ao mesmo tempo muito comum, encontrado na implementação de sistemas especialistas, é que usualmente parece impossível fornecer um conhecimento completo sobre o qual o sistema vai operar. Portanto, o nível de desempenho de um sistema especialista está relacionado ao tamanho e a qualidade de sua base de conhecimento.

4.5 MECANISMO DE APRENDIZAGEM E AQUISISÃO DO CONHECIMENTO

A tarefa de extrair o conhecimento e utilizá-lo adequadamente é bastante complexa. Segundo Rabuske (1995) a aquisição do conhecimento tende a caracterizar áreas de pesquisa especificas nas universidades e nos centros de pesquisa, geralmente ligadas à engenharia do conhecimento. Obter o conhecimento é a parte mais critica da construção de um sistema especialista.

De acordo com Rich (1993) o processo convencional, ou seja, entrevistas com o especialista para formalização e introdução do conhecimento na base, é caro e lento e que vale a pena procurar maneiras mais automatizadas de construir bases de conhecimentos. Rich (1993) ainda afirma que embora já existam muitos programas úteis que interagem com os especialistas para extrair conhecimento especializado com eficiência, ainda não existe nenhum sistema de aquisição de conhecimento totalmente automatizado.

4.6 MOTOR OU MÁQUINA DE INFERÊNCIA

Para Rabuske (1995), a máquina de inferência não é normalmente um único módulo de programa. É, em geral, entendido como compreendendo o interpretador de regras e o escalonador das regras, quando o sistema especialista envolve regras de produção.

De acordo com Ribeiro (1987) o mecanismo de inferência depende de como se está representado o conhecimento. Nos sistema de avaliação de regras, o mecanismo de inferência busca regras na base de conhecimento e as avalia. Essa busca depende dos fatos e das hipóteses que existem e que se quer determinar a cada momento. Os objetivos a serem determinados pelo sistema de inferência devem se relacionados com uma determinada ordem. A busca de regras é feita de maneira automática para que uma meta seja atingida. Entretanto, existem casos em que a resposta pode ser obtida de maneira imediata e, nesses casos são estabelecidas estratégias de avaliação imediata, evitando todo o processo de inferência proceder antes à busca das novas regras que foram causadas pela necessidade de se atender a uma meta, e avaliar essas regras a serem pesquisadas. Como os atributos são encontrados em diversas regras, o valor de uma clausula já pode ter sido estabelecido. Esse valor, sozinho, permite determinar antecipadamente que a premissa da regra é falsa, e que não há razões para novas buscas. As estratégias de busca e avaliação de regras dependem do tipo de representação para o conhecimento e da arquitetura das próprias regras.

A máquina de inferência, de certo modo, tenta imitar os tipos de pensamento que o especialista humano emprega quando resolve um problema, ou seja, ele pode começar com uma conclusão e procurar uma evidência que a comprove, ou pode iniciar com uma evidência para chegar a uma conclusão. Em sistemas especialistas, estes dois métodos são chamados de *“backward chaining”* e *“forward chaining”* respectivamente. Nem todos os sistemas utilizam a mesma abordagem para a representação do seu conhecimento, portanto, a máquina de

inferência deve ser projetada para trabalhar com a representação de conhecimento específica utilizada.

4.7 REPRESENTAÇÃO DO CONHECIMENTO

Segundo Heinzle (1995), para que um sistema especialista possa resolver problemas é imprescindível que esteja associado a ele um razoável volume de conhecimentos relativos ao domínio do problema. Este conhecimento, por sua vez deve ser transformado em organizadas estruturas de dados que permitam a sua utilização pelo computador e ao mesmo tempo sejam facilmente administradas pelos especialistas e usuário do sistema.

4.7.1 QUADROS

De acordo com Rabuske (1995) está forma de representação descreve, tipicamente, classes de objetos. Os quadros, também conhecidos por *"frames"*, são estruturas de preenchimento, constituindo caso particular das redes semânticas. O quadro é constituído por um nome, um conjunto de atributos chamados *"Slots"* e um conjunto de métodos para sua utilização.

4.7.2 REDES SEMÂNTICAS

Segundo Rabuske (1995) uma rede semântica é representada por um grafo dirigido, com arestas e nós rotulados. Os nós (vértices) representam objetos, situações ou conceitos, sendo os elementos pertencentes à rede, enquanto que os arcos (arestas) exprimem as relações entre estes elementos. Esta descrição faz com que ela tenha algo em comum com a maioria das representações do conhecimento, as quais, em última análise, devem descrever objetos e suas interrelações.

As redes semânticas permitem qualquer tipo de ligação, contanto que elas tenham capacidade de transmitir o que significam. Os dois tipos de ligações mais comuns são: "é-parte-de" e "é-um". As ligações pode ser dividida em quatro tipos básicos:

- **ligação do tipo propriedade:** relaciona dois nós para demonstrar uma propriedade;
- **ligação do tipo subparte:** indica que um nó é subparte ou componente de outro;

- **ligação do tipo subclasse:** exprime está características entre os nós envolvidos, por exemplo teses pode ser consideradas subclasses em relação ao nó pesquisa;
- **ligação do tipo relacionamento:** indica que os nós envolvidos devem ser entendidos como, de alguma forma, relacionados. Por exemplo “livro” e “escrever”.

De acordo com Heinzle (1995) para se representar um grande volume de informações, uma rede semântica pode tornar-se bastante complexa e de difícil representação gráfica. Em geral se utiliza uma estrutura de dados adequada para representá-la além de uma linguagem de programação apropriada para utilizar esta forma de representação do conhecimento.

4.7.3 LÓGICA DAS PREPOSIÇÕES E DOS PREDICADOS

“Na lógica das proposições, será atribuído o valor lógico verdadeiro se as informações disponíveis permitem tirar esta conclusão a respeito de uma preposição, caso contrário é atribuído o valor falso. Para se trabalhar com várias preposições utilizam-se operadores de conexão para assim obter as chamadas preposições compostas e aumentar a capacidade de expressão. Estes operadores são: AND, NOT, OR, IMPLIES, EQUIVALENT” (Heinzle, 1995).

Pode se facilmente representar fatos do mundo real usando lógica proposicional, entretanto, isto ainda não é suficiente para que a lógica das preposições se torne um instrumento valioso à representação do conhecimento em sistemas especialistas, com isto acaba surgindo a lógica dos predicados, que apresenta uma capacidade bem ampliada neste sentido.

A lógica dos predicados, ou também referida por cálculo dos predicados introduz funções, quantificadores e predicados para dar-lhe maior capacidade de expressão. A lógica dos predicados também permite que as expressões tenham variáveis, e com elas consegue-se generalizar declarações sobre classes ou entidades. Na criação dos predicados que devem constituir o conhecimento do sistema, são usados identificadores para representar objetos ou relações que podem ser livremente criados.

De acordo com Heinzle (1995) está forma de representação do conhecimento é uma ferramenta poderosa, pois possuí uma notação simples capaz de traduzir em sentenças de sintaxe e semântica bem definidas as situações da vida cotidiana. Entre os aspectos negativos

da lógica dos predicados está a impossibilidade de representar os tempos de ocorrência dos fatos quando incrementos na representação precisam ser feitos.

4.7.4 REGRAS DE PRODUÇÃO

De acordo com Heinzle (1995) sua estrutura constitui-se basicamente de uma premissa, ou conjunto de premissas, e uma conclusão, ou conjunto de conclusões. As regras são armazenadas como uma coleção de declarações SE-ENTÃO (Se <premissa> ENTÃO <conclusões>). Onde a parte condicional consiste de uma expressão proposicional ou simplesmente um termo.

5 TRABALHO PROPOSTO POR MATTOS

Depois das pesquisas, Mattos (1999) apresentou uma primeira proposta para introduzir o conceito de análise de requisitos já nas primeiras fases do ensino de computação, onde foi desenvolvida uma ferramenta didática baseada em sistemas especialistas. Foi utilizado para implementar este protótipo como ambiente de desenvolvimento o Clips (*C Language Integrated Production System*) o qual foi desenvolvido pela NASA/Johnson Space Center.

5.1 ANÁLISE

Durante o processo de análise do conhecimento, verificou-se que havia questões que exigiam respostas do tipo sim/não, questões que exigiam respostas textuais, situações que era necessário uma orientação ao aluno no sentido de guiá-lo para o passo seguinte e finalmente situações onde se fazia necessário uma realimentação sobre as decisões tomadas anteriormente tendo em vista posicioná-lo no contexto da solução em andamento.

A partir destas informações, construiu-se uma árvore de decisões a qual possui 4 tipos de nodos:

- **Nodos de decisão:** exemplo: *Há alguma operação lógica ou aritmética?*
- **Nodos de ação:** exemplo: *Descreva a operação?*
- **Nodo de status:** exemplo: *Até o momento você identificou os seguintes passos:*
- **Nodo de ajuda:** exemplo: *Uma vez analisadas as pré-condições, podemos identificar a operação a ser realizada!*

5.2 IMPLEMENTAÇÃO

A estratégia adotada para orientar o aluno foi à filosofia 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.

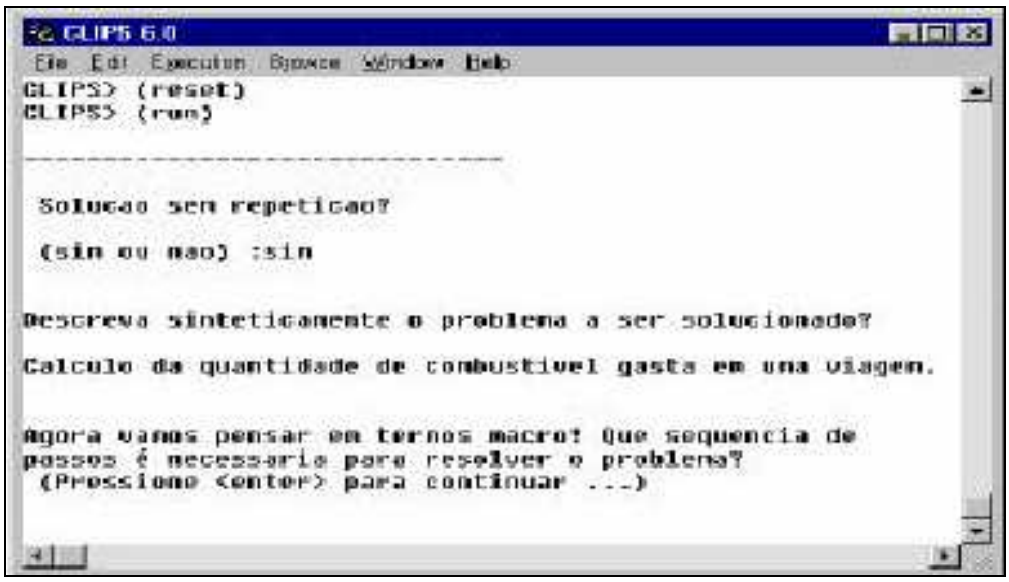
Para tanto, foram construídas regras, que permitem a navegação através dos nodos da árvore de decisões. Assim sendo, medida em que os nodos vão sendo repetidamente visitados, novos fatos vão sendo gerados na memória de trabalho. Estes novos fatos disparam regras que

uma vez executadas geram uma nova árvore auxiliar (denominada árvore de log), a qual registra as respostas do usuário e estabelece a seqüência em termos temporários em que as respostas vão sendo cadastradas.

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. Está estratégia permitiu que fosse adquirido o conhecimento tácito do aluno, a partir do conhecimento declarativo, ou seja, o aluno sabe responder sim e não as perguntas que vão sendo realizadas. No final do processo é possível inferir a solução macro do problema através da conjunção dos vários tipos de conhecimento envolvidos no processo.

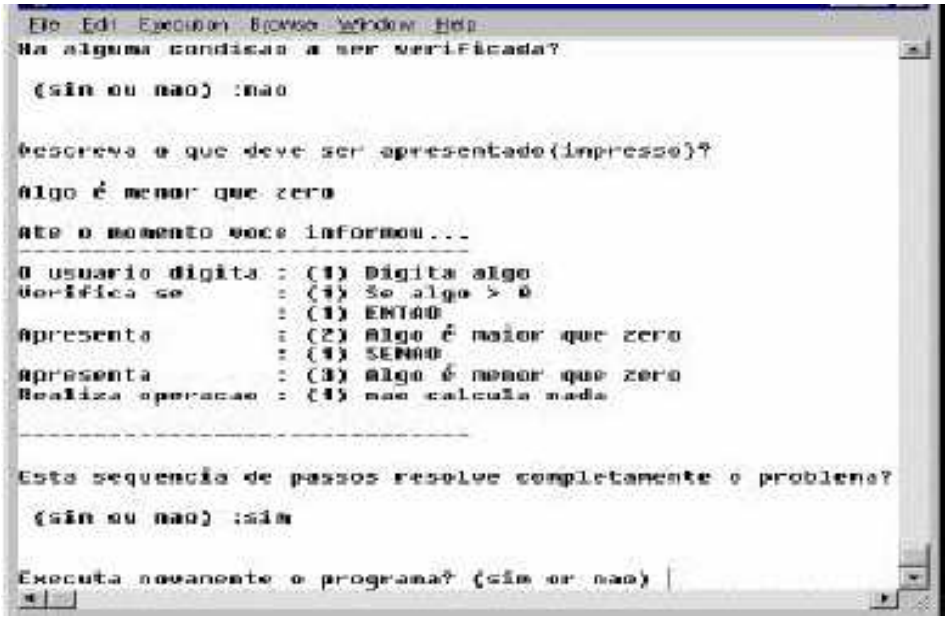
Na fig. 3 e fig. 4 são apresentados duas telas do protótipo desenvolvido em Clips por Mattos (1999):

Figura 3 – Tela 1 do sistema em Clips



Fonte: Mattos (1999)

Figura 4 – Tela 2 do sistema em Clips



Fonte: Mattos (1999)

5.3 CONSIDERAÇÕES

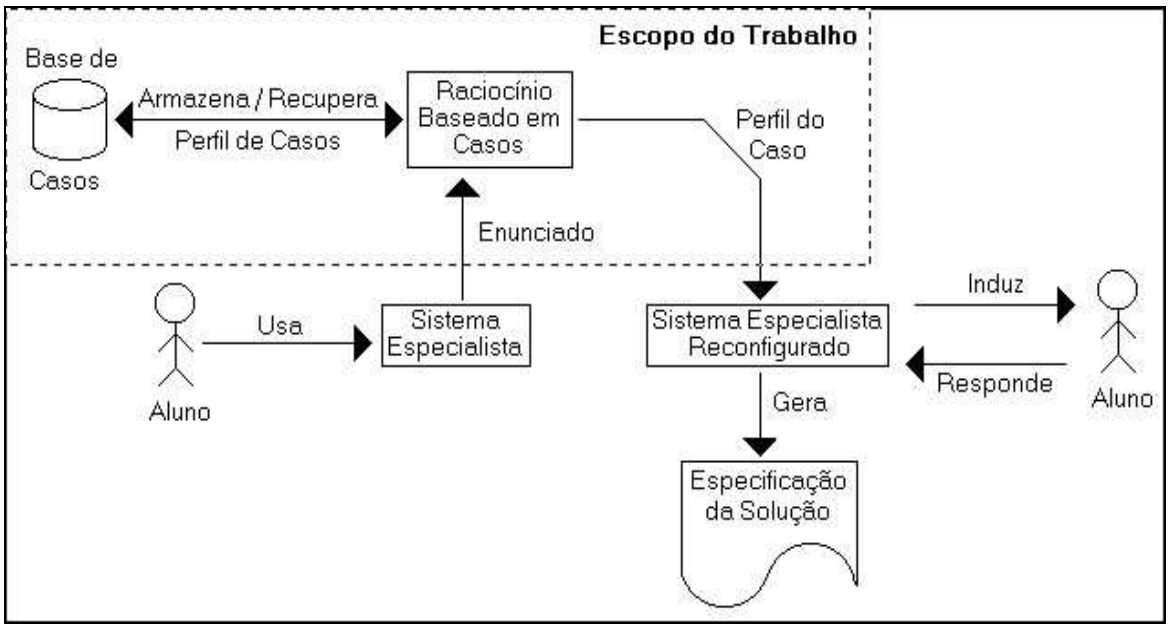
Pode-se considerar que, apesar da implementação do protótipo atender os objetivos iniciais que eram o desenvolvimento de uma aplicação em software que se constituísse em uma ferramenta de apoio ao aprendizado de lógica de programação, o protótipo além de estar implementado em modo caractere é de difícil manutenção, com isto sendo difícil implementar as estruturas de algoritmos que ainda faltam, como a estrutura de repetição e de matrizes e vetores.

Como continuação do trabalho acima mencionado, foi desenvolvido um protótipo validando a idéia descrita em Mattos (1999) para auxiliar no ensino de algoritmos desenvolvendo-se um protótipo de um sistema especialista através da utilização da ferramenta de programação Delphi, onde o protótipo tem uma interface gráfica com o usuário e também implementa a estrutura de suporte a problemas cuja solução envolva estruturas de repetição, que ainda não foram implementados no protótipo elaborado em Clips por Mattos (1999).

6 DESENVOLVIMENTO DO TRABALHO

Este 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. Na fig.5 pode-se ter uma visão deste projeto:

Figura 5 – Projeto principal



A seguir são apresentadas as fases de desenvolvimento percorridas durante a especificação e a implementação deste protótipo. Esta ferramenta pode ser classificada como um protótipo porque alguns aspectos referentes à construção de algoritmos ainda não foram implementados e além disto este protótipo faz parte de um outro projeto maior conforme é mostrado na fig. 5.

6.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Como já foi destacado anteriormente, o objetivo deste trabalho é validar a idéia descrita em Mattos (1999), desenvolvendo um protótipo utilizando a técnica de sistemas

especialistas capaz de auxiliar o aluno (usuário) a elaborar algoritmos corretamente, onde através do protótipo o aluno responde uma série de perguntas, inferindo suas respostas na base de conhecimento que por sua vez se encarrega-se de formular novas perguntas com base nas informações cedidas pelo aluno (usuário), e no final o protótipo informa ao aluno (usuário) os passos a serem seguidos para a elaboração do algoritmo.

6.2 AQUISIÇÃO DO CONHECIMENTO

O conhecimento do especialista para implementar este protótipo foi adquirido através do trabalho desenvolvido por Mattos (1999) onde foram realizados pesquisas e acompanhamentos em algumas turmas da disciplina de algoritmos da Universidade Regional de Blumenau.

Utilizou-se também para se adquirir o conhecimento, os livros de Guimarães (1985), Forbellone (2000) e Venâncio (1997), além dos meu conhecimentos adquiridos durante o curso de graduação em ciências da computação na Universidade Regional de Blumenau.

6.3 ESPECIFICAÇÃO

Neste capítulo será apresentada a especificação do protótipo desenvolvido, sendo descrito a representação do conhecimento do especialista e a representação do conhecimento adquirido.

O sistema desenvolvido terá a finalidade de adquirir o conhecimento do especialista no assunto de construção de algoritmos para auxiliar o aluno. Este conhecimento foi armazenado no protótipo através de várias perguntas, que o usuário responde até chegar à solução para um problema qualquer em algoritmo.

O protótipo desenvolvido está dividido em quatro units desenvolvidas no Delphi:

- **Unit principal:** unit principal do sistema, onde são apresentados um menu e botões para executar as tarefas do protótipo;
- **Unit de perguntas:** unit mais importante do protótipo, pois nela está armazenado o conhecimento do especialista, são gerados os passos para a resposta ao aluno e onde são apresentadas as perguntas que o usuário responde para chegar em um resultado final;

- **Unit da resposta:** unit do sistema onde são apresentados os passos gerados para resolver o problema em algoritmos através das perguntas, que o aluno estiver resolvendo;
- **Unit de sobre:** unit do sobre no sistema.

6.4 REPRESENTAÇÃO DO CONHECIMENTO DO ESPECIALISTA

Para construir um sistema especialista é necessário adquirir o conhecimento do especialista na área desejada, assim sendo para este trabalho foi necessário adquirir conhecimento de profissionais na área de construção de algoritmos.

No apêndice 1 é demonstrado o grafo das perguntas geradas através das pesquisas efetuadas que é a estrutura da base de conhecimento do protótipo. A base de conhecimento que é representada no grafo do apêndice 1 deste sistema especialista é baseada em redes semânticas.

Durante o processo de aquisição do conhecimento verificou-se que havia questões que exigiam resposta sim/não, questões que exigiam respostas textuais, situações em que era necessária uma orientação ao aluno no sentido de guiá-lo para o passo seguinte, situações onde se fazia necessário uma realimentação sobre decisões tomadas anteriormente tendo em vista posicioná-lo no contexto da solução em andamento, situações onde é necessário recomeçar a partir de uma determinada pergunta armazenando a pergunta atual para possibilitar determinar os passos dentro de estruturas de repetição e de condição e situações onde chega ao fim das perguntas, que possibilita voltar para a próxima pergunta depois de determinar os passos dentro de alguma estrutura de condição ou repetição que esteja em aberto ou determinar o fim das perguntas e mostrar os passos gerados pelo sistema ao usuário. A partir destas informações construiu-se um grafo, a qual possui sete tipos de nodos (“perguntas”):

- **nodo de inicio:** este nodo é o primeiro do sistema especialista, onde é descrito qual o problema a ser resolvido em algoritmos. Este nodo é determinado como root;
- **nodos de decisão:** estes nodos são os que têm como resposta sim ou não;

- **nodos de ação**: estes nodos são aqueles que serão utilizados para a geração dos passos para resolver o problema em algoritmo. Têm uma resposta descritiva;
- **nodo de status:** nodo onde é demonstrado os passos gerados até o momento pelo sistema para resolver o problema;
- **nodos de ajuda:** nodo onde é demonstrado um auxilio ao usuário do sistema especialista;
- **nodos de redirecionamento:** nodo onde é redirecionado para um determinada pergunta, e quando chegar ao fim retorna para este nodo e continua a responder a próxima pergunta. Este nodo é utilizado para determinar os passos necessários dentro das estruturas de repetição.
- **nodo de fim:** este nodo é o que determina o fim das perguntas, mas quando ainda existe alguma pergunta em aberto continua a fazer as perguntas a partir da mesma. Como exemplo temos quando chegamos ao nodo fim e estamos respondendo os passos de uma estrutura de repetição, o sistema especialista terá que continuar a fazer as perguntas para descobrir que outros passos são necessários depois da estrutura de repetição.

6.4.1 ESTRUTURA DE REPETIÇÃO

As perguntas para resolver problemas que utilizam estrutura de repetição foram conseguidas através da sua própria estrutura, onde podemos ter repetição com teste no inicio e repetição com variável de controle.

A repetição com teste no inicio é representada na forma de pseudocódigo através da descrição “enquanto”, onde as rotinas dentro da repetição são executadas enquanto a condição que lhe foi determinada for verdadeira. Assim sendo foram determinadas as perguntas para resolver este tipo de repetição conforme o quadro 14:

Quadro 14 – Perguntas da estrutura de repetição com teste no inicio

1 – (Condição)Tem repetição? 2 – (Condição)Se número não conhecido de repetição então? 3 – (Pergunta) Qual a condição da repetição? 4 – (Ajuda)Identificar os passos dentro da repetição!

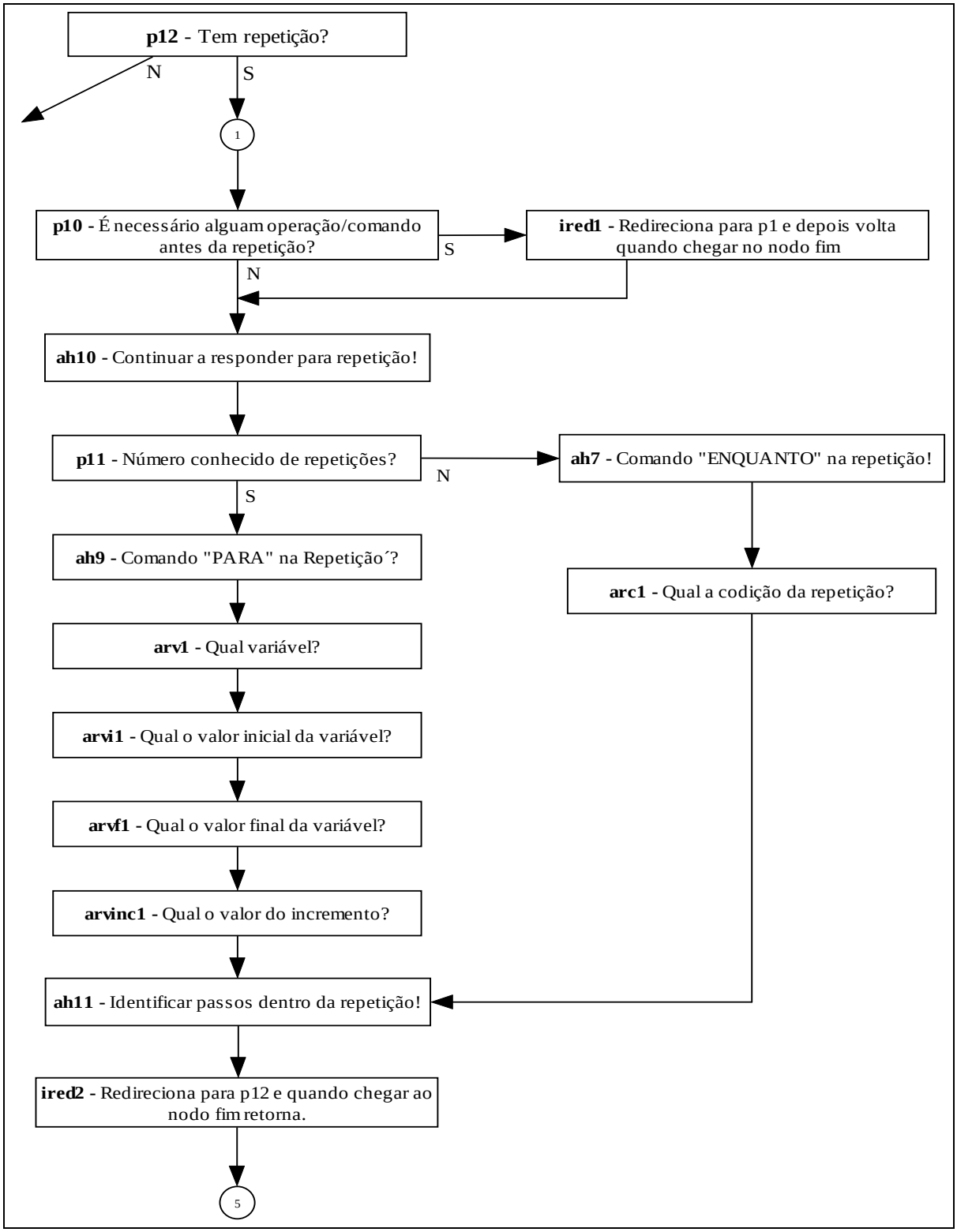
A repetição com variável de controle é representada na forma de pseudocódigo através da descrição *"para"*. Este tipo de estrutura de repetição é utilizado quando se sabe qual o número de vezes que determinada estrutura de algoritmos é executada. A estrutura *"para"* utiliza uma variável de controle que recebe um valor inicial, onde é executada até quando o valor dela for aquele que foi determinado com último valor. Está variável e sempre incrementada com o valor que é descrito na estrutura para. Assim sendo foram determinadas algumas perguntas para resolver este tipo de repetição conforme o quadro 15:

Quadro 15 – Perguntas da estrutura de repetição com variável de controle

1 – (Condição)Tem repetição? 2 – (Condição)Se número conhecido de repetição então? 3 – Qual a variável? 4 – Qual o valor inicial da variável? 5 – Qual o valor final da variável? 6 – Qual o valor de incremento da variável? 7 – (Ajuda)Identificar os passos dentro da repetição!

Na fig. 6 é apresentada uma parte do grafo, demonstrando a representação das perguntas relacionadas a implementação da funcionalidade de controle de repetição no grafo do sistema.

Figura 6 – Parte do grafo mostrando a estrutura de repetição



6.5 REPRESENTAÇÃO DO CONHECIMENTO ADQUIRIDO

Para o protótipo gerar os passos necessários para a construção do algoritmo é necessário ter as respostas do usuário em um determinado problema que ele precisa resolver. Estas respostas precisam estar corretamente informadas pelo usuário, senão o sistema especialista não demonstrará corretamente os passos na resposta final. Para adquirir são utilizadas todas as perguntas do tipo ação. Como exemplo temos no grafo que se encontra no apêndice 1 os nodos “ai1” que quando respondida armazena a condição de uma estrutura de condição.

Quando o usuário responde algumas destas perguntas o sistema especialista armazena estes conhecimentos em uma lista duplamente encadeada que é demonstrada a seguir no detalhamento da implementação.

6.6 IMPLEMENTAÇÃO

Abaixo serão apresentadas as características da implementação deste algoritmo, onde será detalhada à técnica e ferramenta utilizada para implementação. Também será apresentado um detalhamento da implementação bem como a operacionalidade do protótipo.

6.6.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

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 as 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 de baixo nível (assembler) e tem evoluído em busca do aperfeiçoamento para poder atender as exigências cada vez mais complexas dos softwares modernos. Para ajudar no desenvolvimento do trabalho em delphi foram utilizados os livros de Cantù (2002), Engo (1997), Sonnino (2000) e Cornell (1995).

6.6.2 DETALHAMENTO DA IMPLEMENTAÇÃO

A implementação propriamente dita passará a ser vista em maiores detalhes a partir de agora. O principal objetivo deste tópico é demonstrar como foi concebida a implementação, tendo-se como ponto de partida a especificação.

Para armazenar as perguntas ou o conhecimento foi implementada uma constante que é uma matriz do tipo *nodos* e este tipo é declarado como um registro. Cada registro tem um índice controlado pelo sistema especialista que é o nome que é declarado com um tipo chamado *tipos* conforme quadro 16 e ainda é informado qual o tipo de pergunta que temos em cada pergunta armazenada nesta matriz.

As perguntas podem ser dos tipos:

- **simnao**: nodo onde o usuário responde sim ou não;
- **desc_problema**: nodo onde é perguntado a descrição do problema a ser resolvido pelo usuário;
- **desc_digi**: nodo onde o usuário responde descritivamente, armazenando na lista duplamente encadeada de resposta para ser utilizado quando ser demonstrado a resposta ao usuário, onde será utilizado para ler algum dado através comando “*leia”*;
- **desc_if**: nodo onde o usuário responde descritivamente, armazenando na lista duplamente encadeada de resposta para ser utilizado quando ser demonstrado a resposta para o usuário, onde será utilizado como condição;
- **desc_imp**: nodo onde o usuário responde descritivamente, armazenando na lista duplamente encadeada de resposta para ser utilizado quando ser demonstrado a resposta para o usuário, onde será utilizado para ser impresso algum dado através do comando “*escrever*”;
- **desc_oper**: nodo onde o usuário responde descritivamente, armazenando na lista duplamente encadeada de resposta para ser utilizado quando ser demonstrado a resposta para o usuário, onde será utilizado como uma operação aritmética ou lógica;

- **var_rep_para**: nodo onde o usuário responde descritivamente, armazenando na lista duplamente encadeada de resposta para ser utilizado quando ser demonstrado a resposta para o usuário, onde será utilizado como a variável de uma repetição do comando “*para*”;
- **vlr_i_rep_para**: nodo onde o usuário responde descritivamente, armazenando na lista duplamente encadeada de resposta para ser utilizado quando ser demonstrado a resposta para o usuário, onde será utilizado como o valor inicial da variável de uma repetição do comando “*para*”;
- **vlr_f_rep_para**: nodo onde o usuário responde descritivamente, armazenando na lista duplamente encadeada de resposta para ser utilizado quando ser demonstrado a resposta para o usuário, onde será utilizado como o valor final da variável de uma repetição do comando “*para*”;
- **vlr_inc_rep_para**: nodo onde o usuário responde descritivamente, armazenando na lista duplamente encadeada de resposta para ser utilizado quando ser demonstrado a resposta para o usuário, onde será utilizado como o incremento da variável de uma repetição do comando “*para*”;
- **cond_enquanto**: nodo onde o usuário responde descritivamente, armazenando na lista duplamente encadeada de resposta para ser utilizado quando ser demonstrado a resposta para o usuário, onde será utilizado como condição de uma repetição do comando “*enquanto*”;
- **i_redirecio**: nodo onde apenas é feito um redirecionamento para uma outra pergunta e quando chegar ao nodo fim, retornará para a próxima pergunta depois do redirecionamento;
- **ajuda**: nodo onde o usuário recebe uma instrução ou ajuda para a próxima pergunta;
- **situacao**: nodo onde o usuário vai para a tela de resposta, mostrando uma situação dos passos gerados ou os passos completos;
- **fim**: nodo onde termina as perguntas. Se tiver algum nodo em aberto então irá continuar a partir deste, senão chegará ao fim do processo e serão mostrados os passos gerados para resolver o problema informado pelo usuário.

- **lentao**: nodo onde é redirecionado para a pergunta inicial, para determinar os passos dentro do lado positivo de uma condição e quando chegar ao nodo fim retornará para então continuar na próxima pergunta;
- **lsenao**: nodo onde é redirecionado para a pergunta inicial, para determinar os passos dentro do lado negativo de uma condição e quando chegar ao nodo fim retornará para então continuar na próxima pergunta;

No Quadro 16 está sendo mostrada a declaração na implementação de:

- **tipos**: declaração do nome de cada pergunta ou nodo na base de conhecimento do sistema especialista;
- **tipo_nome**: declaração do tipo de pergunta ou nodo que estão na base de conhecimento do sistema especialista;
- **nodos**: declaração do registro onde é armazenado as perguntas dos sistema especialista.

Quadro 16 – Declaração dos tipos, tipo de nome e do registro nodos

```
//declarações do nomes das perguntas
   tipos = (root, p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, ad1, ad2,
            ai1, ai2, ap1, ap2, ao1, arv1, arv2, arvi1, arvf1, arvinc1, arc1, ah1, ah2,
            ah3, ah4, ah5, ah6, ah7, ah8, ah9, ah10, ah11, ah12, as1, af1, vazio, nif ,
            ired1, ired2);
//declarações dos tipos de nome de cada pergunta
   tipo_nome = (simnao, desc_problema, desc_digi, desc_if, desc_imp, desc_oper,
                ajuda, situacao, fim, lentao, lsenao, var_rep_para, vlr_i_rep_para,
                vlr_f_rep_para, vlr_inc_rep_para, i_redirecio, cond_enquanto);
//declaração do registro nodos para guardar as perguntas
    nodos = record
        nome: tipos;
        quest: string;
        sim: tipos;
        nao: tipos;
        prox: tipos;
        LadPosIf: tipos;
        LadNegIf: tipos;
        ProxRed: tipos;
        tipo: tipo_nome;
    end;
```

Como já foi mencionado acima para armazenar as perguntas da base de conhecimento foi utilizada uma constante de uma matriz do tipo nodos que está declarado conforme o quadro 16. No quadro 17 é mostrado um exemplo de como foi implementado está constante:

Quadro 17 – Exemplo de declaração da constante do tipo nodos

```
//Declaração da constante onde estão guardados as perguntas do sistema
especialista
const tab : array [0..42] of nodos =
//root --> Inicio do grafo
  ((nome:root;quest:'Descreva sinteticamente o problema a ser solucionado?';
    sim:vazio;nao:vazio;prox:ah1;LadPosIf:vazio;LadNegIf:vazio;ProxRed:vazio;
    tipo:desc_problema),
//Ação de Perguntas
  (nome:p1;quest:'Ha alguma operação lógica ou aritmética a ser realizada?';
   sim:ah2;nao:p7;prox:vazio;LadPosIf:vazio;LadNegIf:vazio;ProxRed:vazio;
   tipo:simnao),
//Ação onde o usuario informa alguma coisa a ser digitado no algoritmo
  (nome:ad1;quest:'Que informação o usuário deve digitar?';sim:vazio;nao:vazio
   ;prox:p3;LadPosIf:vazio;LadNegIf:vazio;ProxRed:vazio;tipo:desc_digi),
//Ação onde o usuario informa qual a condição a ser colocada no algoritmo
  (nome:ai1;quest:'Descreva que condição deve ser verificada?';sim:vazio;
   nao:vazio;prox:p4;LadPosIf:ah5;LadNegIf:ah6;ProxRed:vazio;tipo:desc_if),
//Ação onde o usuario informa os dados a serem imprimidos no algoritmo
  (nome:ap1;quest:'Descreva qual a informação que deve ser apresentado(impresso)
   para o usuario?'; sim:vazio;nao:vazio;prox:ah3;LadPosIf:vazio;LadNegIf:vazio;
   ProxRed:vazio;tipo:desc_imp),
//Ação onde o usuario informa a operação no algoritmo
   (nome:ao1;quest:'Descreva que operação lógica ou aritmética deve ser
    realizada:';sim:vazio;nao:vazio;prox:as1;LadPosIf:vazio;LadNegIf:vazio;
    ProxRed:vazio;tipo:desc_oper),
//Ação de Ajuda nas perguntas a serem respondidas
   (nome:ah1;quest:'Agora vamos pensar em termos macro! Que sequencia de passos
    é necessaria para resolver o problema?';sim:vazio;nao:vazio;prox:p12;
    LadPosIf:vazio;LadNegIf:vazio;ProxRed:vazio;tipo:ajuda),
//Ação de Status do algoritmo
   (nome:as1;quest:'Até este momento voce identificou os seguintes passos para
    resolver o problema:';sim:vazio;nao:vazio;prox:p5;LadPosIf:vazio;
    LadNegIf:vazio;ProxRed:vazio;tipo:situacao),
//Ação De Fim de geração do algoritmo
   (nome:af1;quest:'Fim';sim:vazio;nao:vazio;prox:vazio;LadPosIf:vazio;
    LadNegIf:vazio;ProxRed:vazio;tipo:fim));
//Fim da declaração de constantes
```

Agora que foi mostrada a implementação do armazenamento de perguntas do sistema especialista no quadro vai ser mostrado como é armazenada a resposta do usuário que é utilizada para gerar uma série de passos para auxiliar o aluno a resolver determinado exercício em algoritmo.

Para armazenar as respostas do usuário foi implementada uma lista duplamente encadeada, onde são armazenadas as respostas das perguntas do tipo ação da base de conhecimento, que são necessárias para a geração da resposta apresentada no final para o usuário. No quadro 18 é apresentada a declaração desta lista duplamente encadeada:

Quadro 18 – Declaração da lista duplamente encadeada da resposta

```
ApontLista = ^lista_Resposta;
lista_Resposta = record
    Atual: tipos;
    desc_respo: string;
    tipo_atual: tipo_nome;
    proximo: ApontLista;
    anterior: ApontLista;
end;
```

Para conseguir descobrir os passos necessários, dentro das estruturas de repetição e de condição é necessário armazenar a próxima pergunta e redirecionar para a primeira pergunta, como podemos verificar no apêndice 1. A próxima pergunta é armazenada dentro de uma pilha que é declarada conforme o quadro 19:

Quadro 19 – Declaração da pilha de nodos em abertos

```
ApontPilha = ^nodos_pilha;
nodos_pilha = record
    nome_prox: tipos;
    tipo_prox: tipo_nome;
    anterior: ApontPilha;
end;
```

6.6.3 ESTUDO DE CASO

Para exemplificar o funcionamento do protótipo que foi desenvolvido neste trabalho será feito um estudo de caso para resolver o problema em algoritmos que se encontra no quadro 20:

Quadro 20 – Descrição do exercício de algoritmos

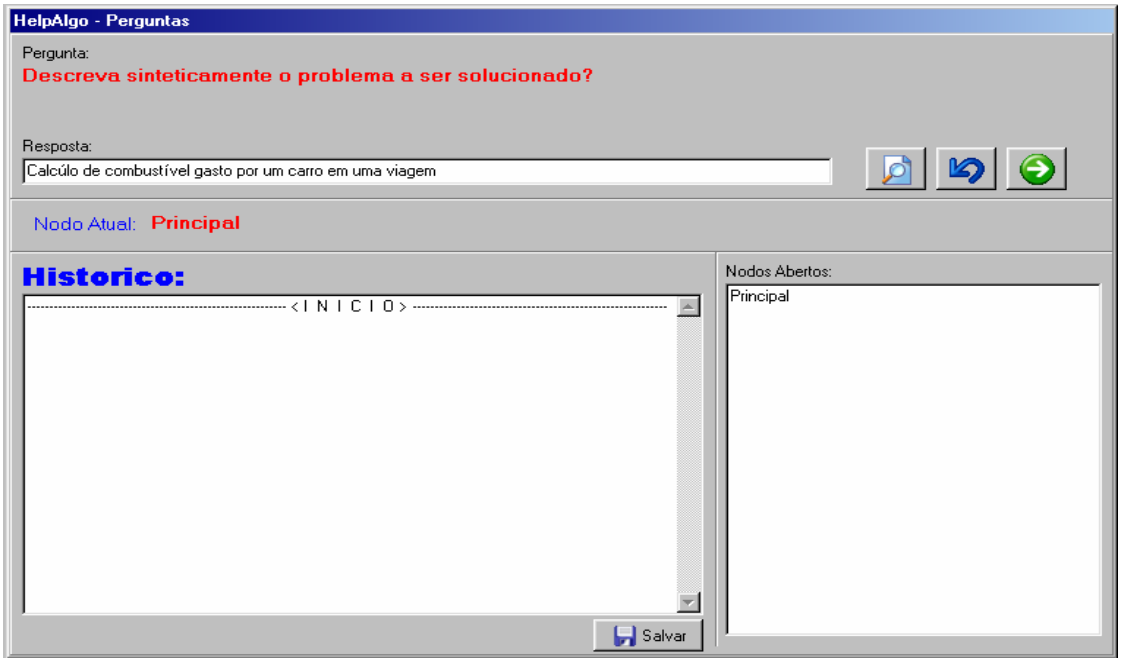
Calcular a quantidade de combustível gasto em uma viagem de um carro, onde o carro faz 12 kilometros por litro. O usuário vai informar qual a distância percorrida pelo carro na viagem, onde não poderá ser menor ou igual a zero.

Na fig. 7 está sendo mostrado a tela principal do protótipo desenvolvido neste trabalho. Para começar a executar o sistema é necessário clicar no botão novo, ou ir no menu arquivo e clicar em novo.

Figura 7 – Tela principal do protótipo

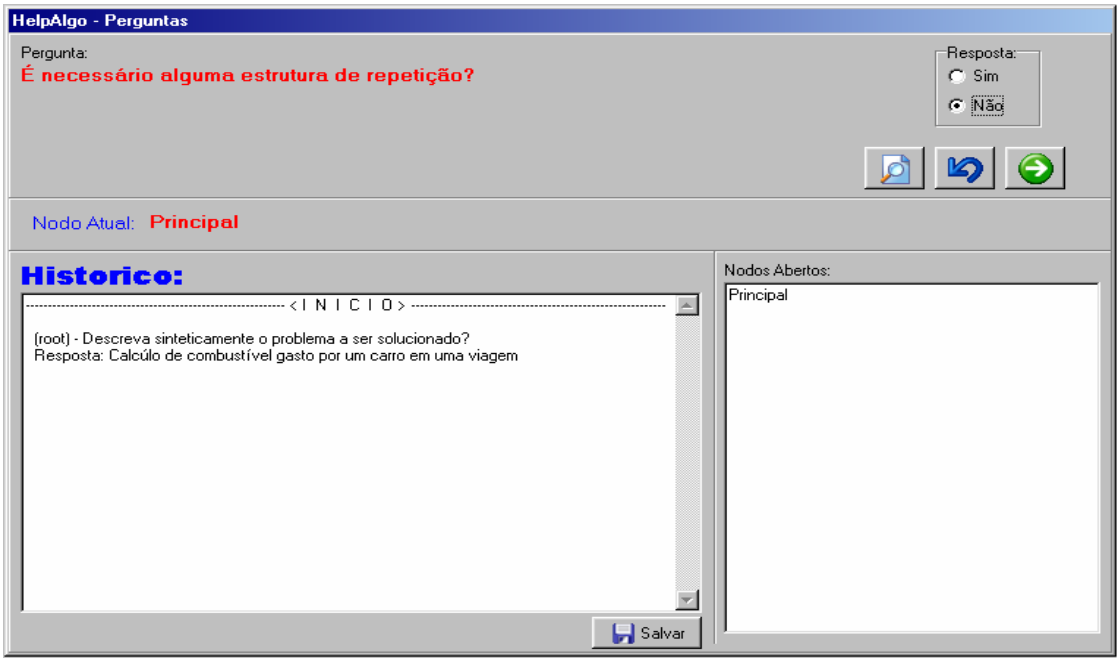
Na fig. 8 é apresentada a tela inicial com as perguntas para o usuário, onde está sendo informado a descrição do exercício que está sendo resolvido.

Figura 8 – Tela 1 do estudo de caso



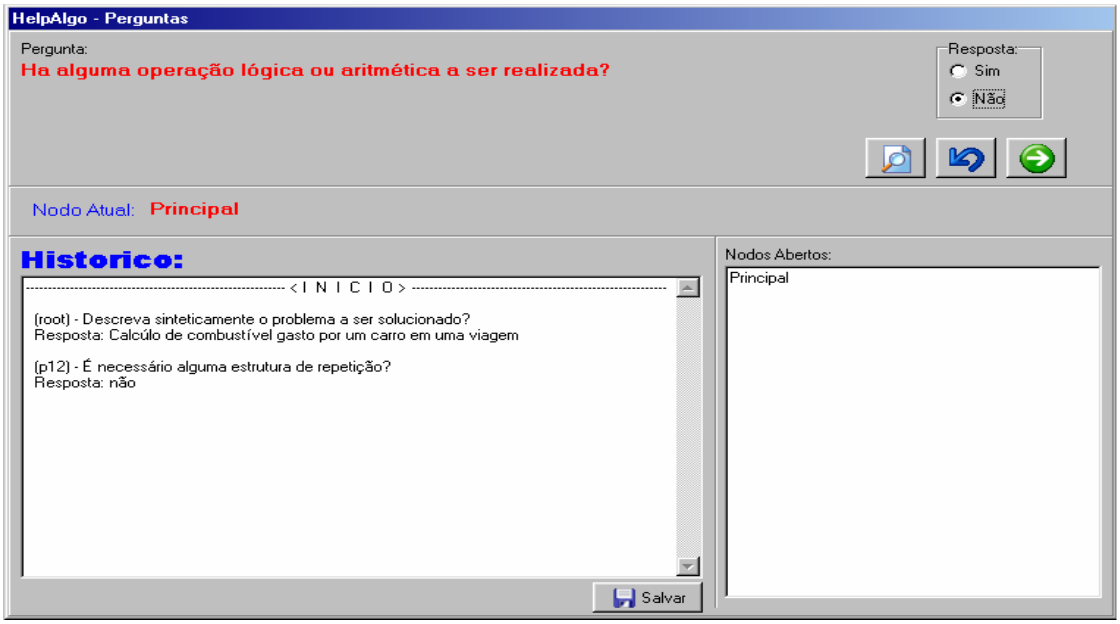
Na fig. 9 é apresentada a tela com a pergunta para verificar se existe repetição no algoritmo, que no caso deste exercício que está sendo resolvido não é necessário.

Figura 9 – Tela 2 do estudo de caso



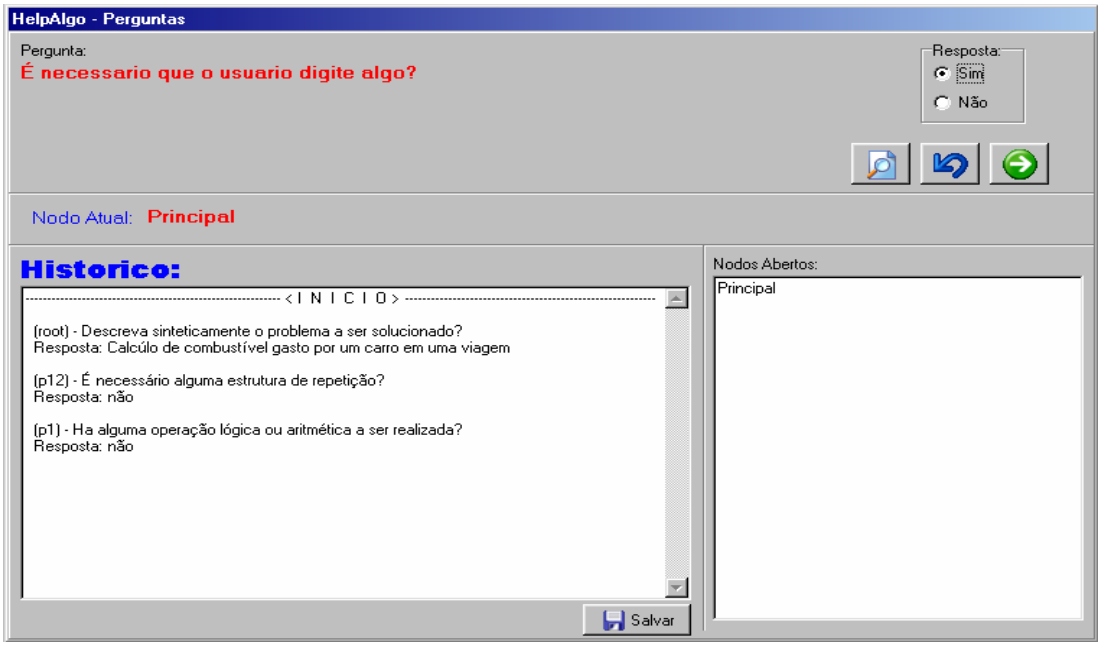
Na fig. 10 é mostrada a próxima tela, onde é perguntado ao usuário se existe operação aritmética ou lógica , que neste momento não é necessário, pois a primeira coisa necessária no algoritmo é o usuário digitar qual à distância percorrida.

Figura 10 – Tela 3 do estudo de caso



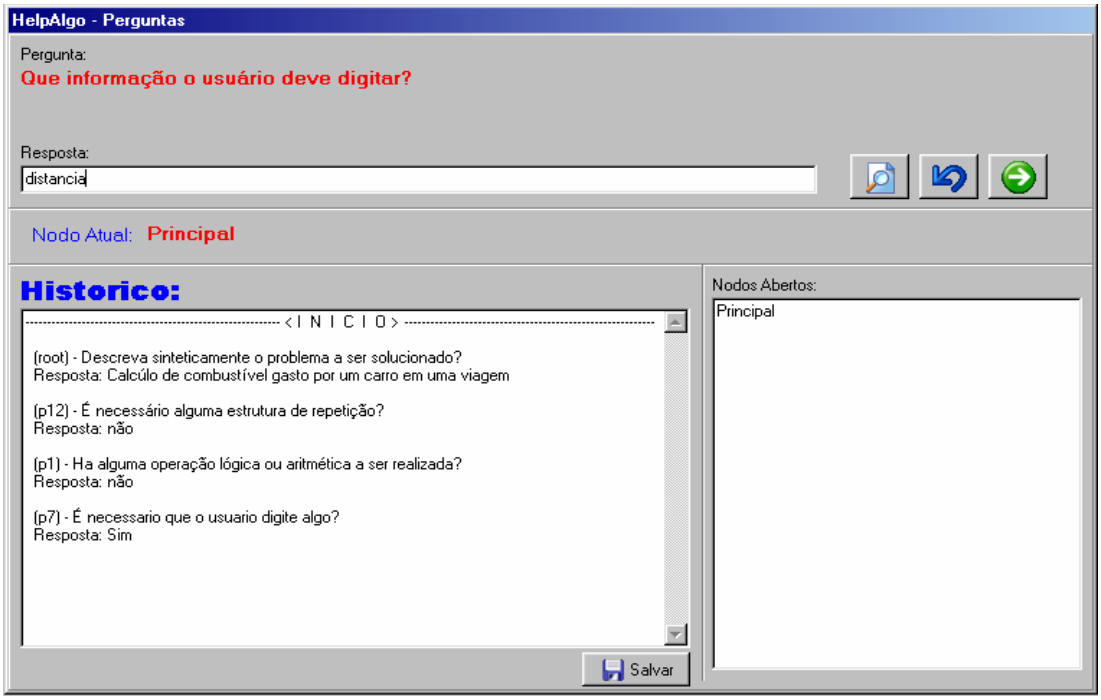
Na fig. 11 é mostrado uma pergunta para verificar se é necessário que o usuário informe algo, onde neste caso a resposta é sim porque o usuário precisa informar a distância percorrida pelo carro.

Figura 11 – Tela 4 do estudo de caso



Na fig. 12 é mostrada a tela onde o usuário vai descrever o que precisa ser informado no algoritmo, que no caso é informado à distância (“distancia”).

Figura 12 – Tela 5 do estudo de caso



Na fig. 13 é apresentado a pergunta para verificar se existe alguma condição no exercício, onde neste caso é sim, porque é necessário testar se à distância percorrida é igual a zero ou diferente de zero. Quando maior que zero calcula qual o combustível gasto na viagem e mostra na tela, senão mostra uma mensagem na tela informando que à "distância digitada deverá ser maior que zero". Na fig. 14 é apresentada a tela onde é informada está condição.

Figura 13 – Tela 6 do estudo de caso

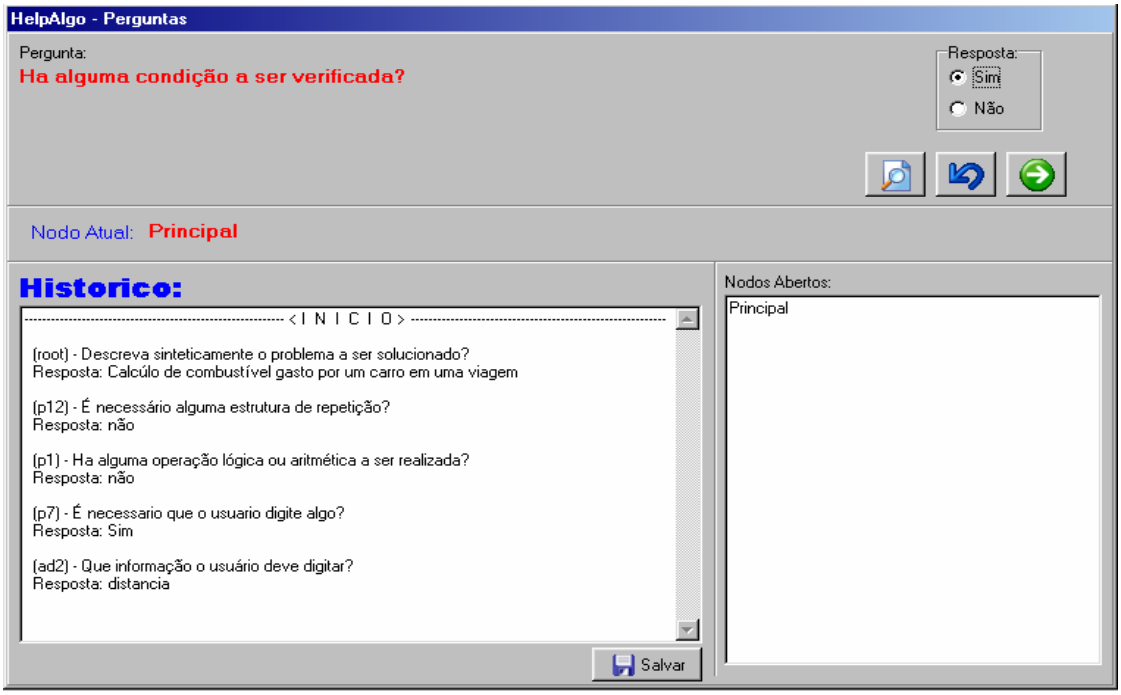
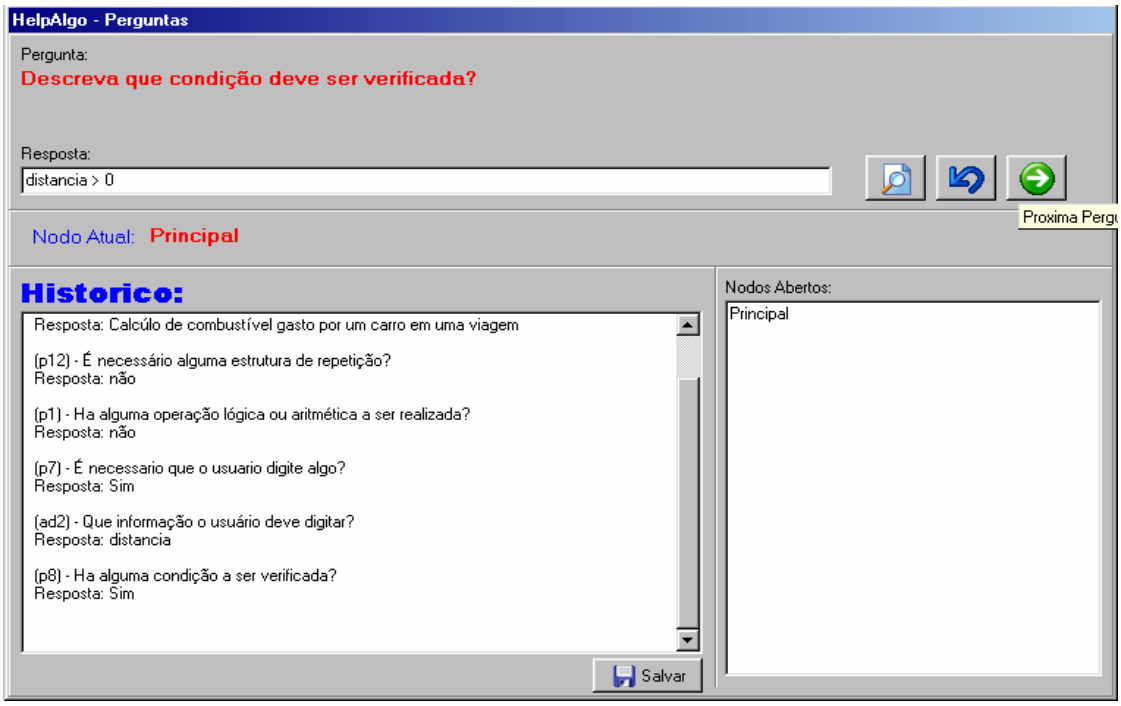
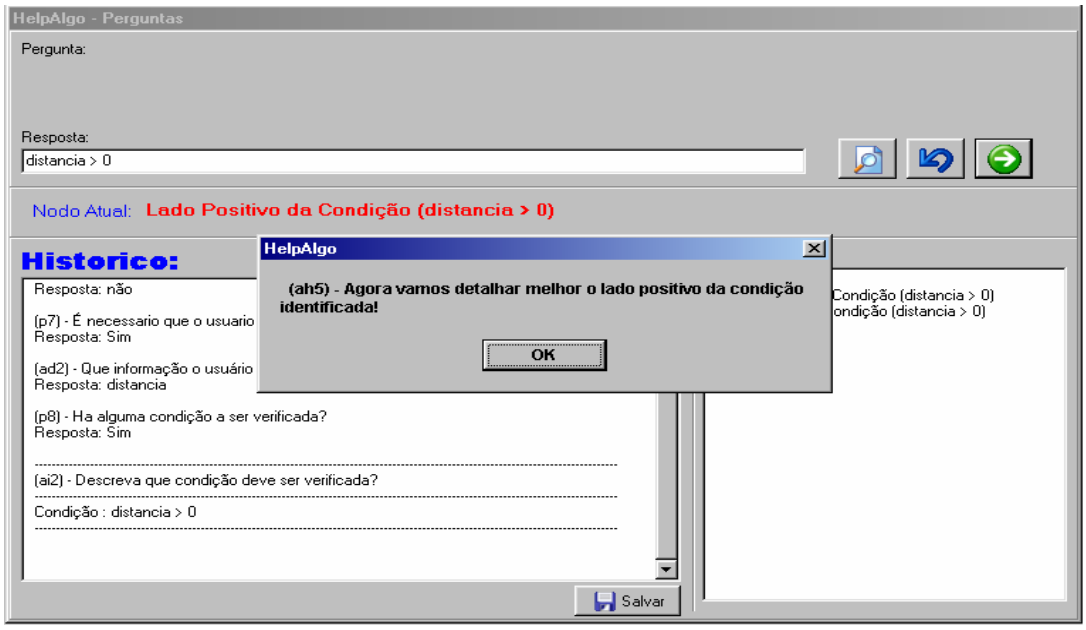


Figura 14 – Tela 7 do estudo de caso



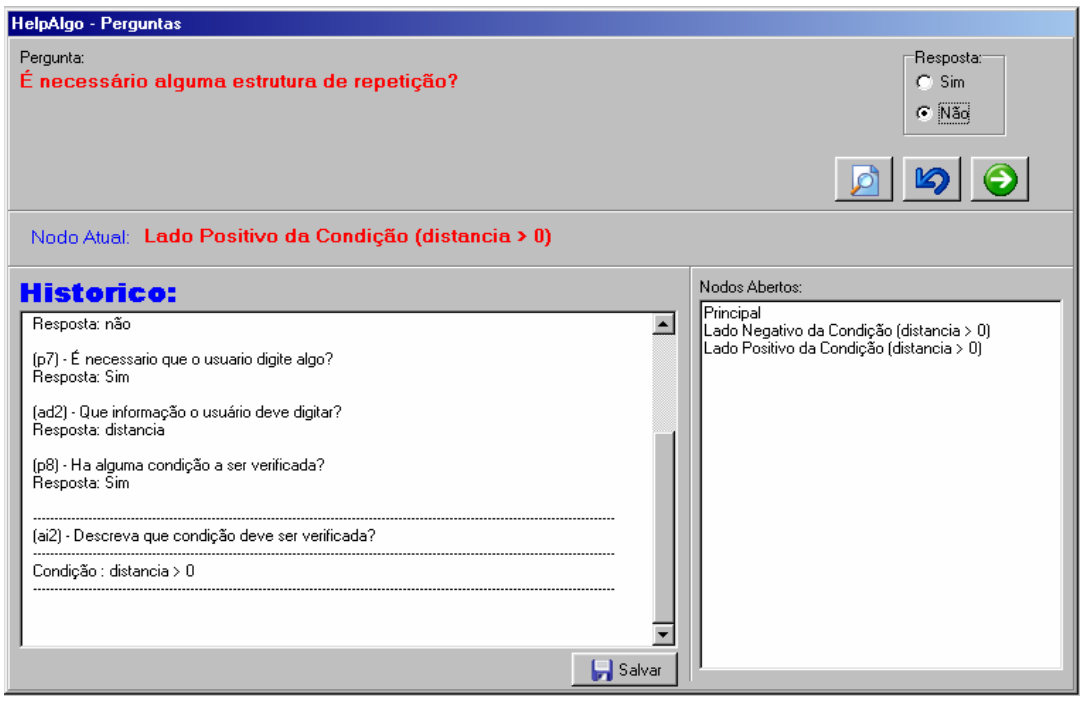
Na fig. 15 é mostrada uma mensagem de ajuda, para informar que a partir de agora as perguntas que serão apresentada vão determinar que passos são necessário dentro do lado positivo da condição, neste caso quando a distância ser maior que zero.

Figura 15 – Tela 8 do estudo de caso



Na fig. 16 é mostrada a tela com a pergunta para verificar se existe alguma repetição dentro da condição (distancia > 0), que neste caso não tem.

Figura 16 – Tela 9 do estudo de caso



Na fig. 17 está sendo apresentada a pergunta para verificar se existe alguma operação aritmética ou lógica a ser realizada, que neste caso é sim, pois é necessário calcular qual foi o combustível gasto. Algumas das próximas perguntas do sistema não estão sendo apresentadas porque as respostas são “não”, assim não sendo necessário apresentar elas. Na fig. 18 é apresentada a tela onde é informado qual a operação, que neste caso é para descobrir quanto combustível foi gasto (comb_gasto = distancia / 12).

Figura 17 – Tela 10 do estudo de caso

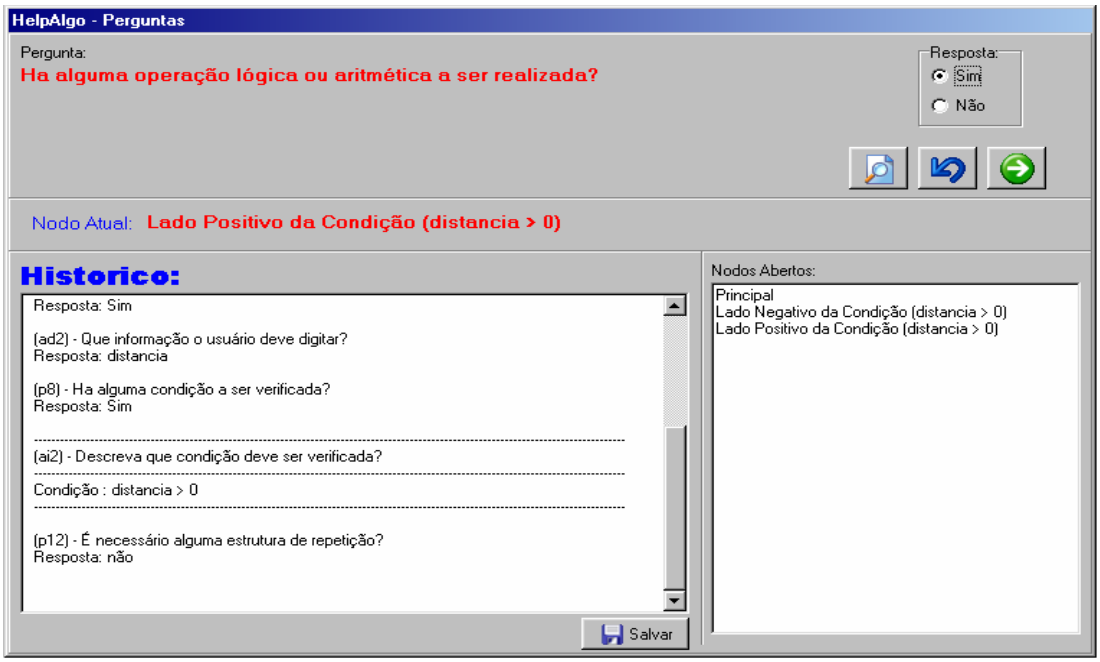
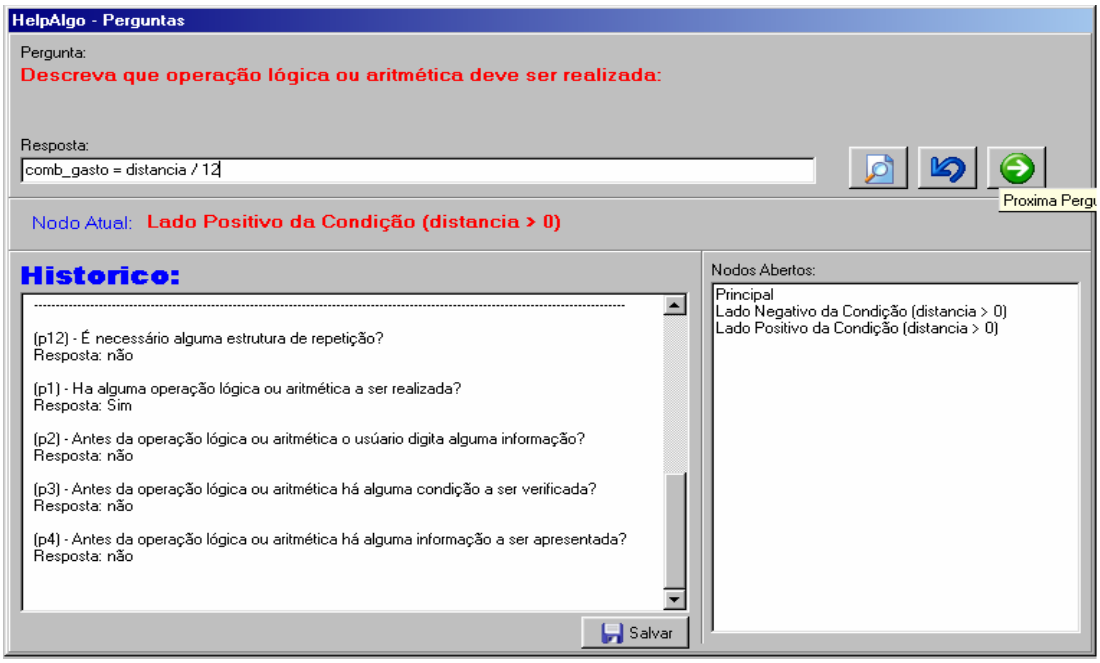
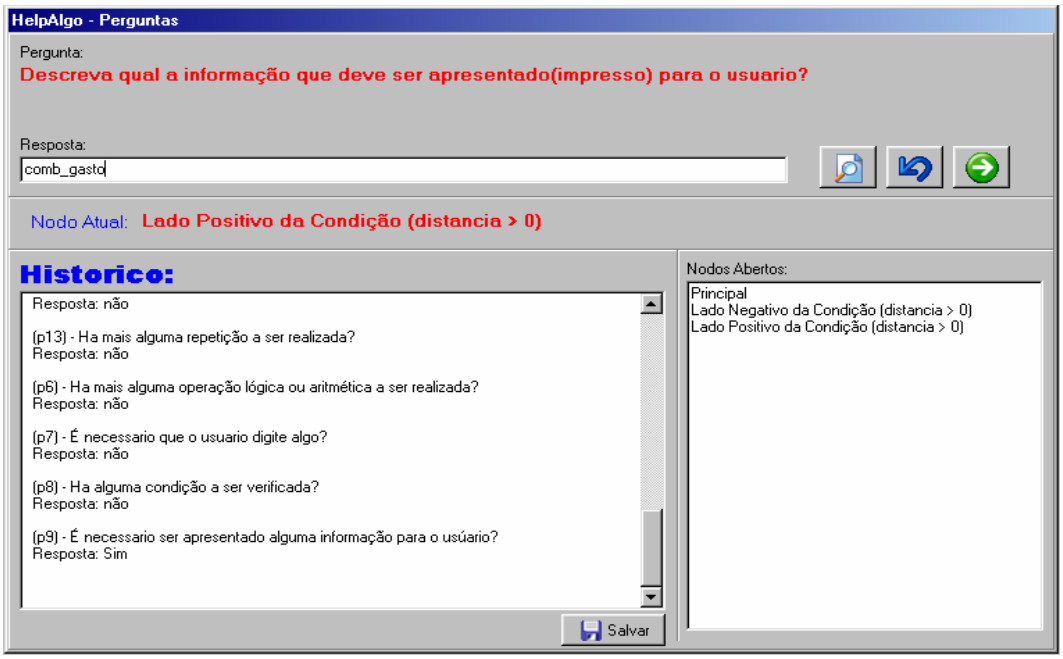


Figura 18 – Tela 11 do estudo de caso



Depois de informado qual a operação que é mostrado na tela final com os passos gerados até o momento, é feito uma pergunta verificando se existe mais algum passo dentro do nodo atual, que no caso é o lado positivo da condição (distancia > 0). Neste caso a resposta é “sim” porque ainda é necessário no algoritmo deste exercício mostrar quanto foi gasto de combustível. Logo após de responder “sim” nesta pergunta, é iniciado desde a primeira pergunta que pergunta se existe repetição. A partir de então serão informadas várias perguntas com “não” na resposta até chegar na pergunta para verificar se é necessário informar algo para o usuário, onde a resposta é sim e será perguntado após o que é necessário informar, conforme a fig. 19:

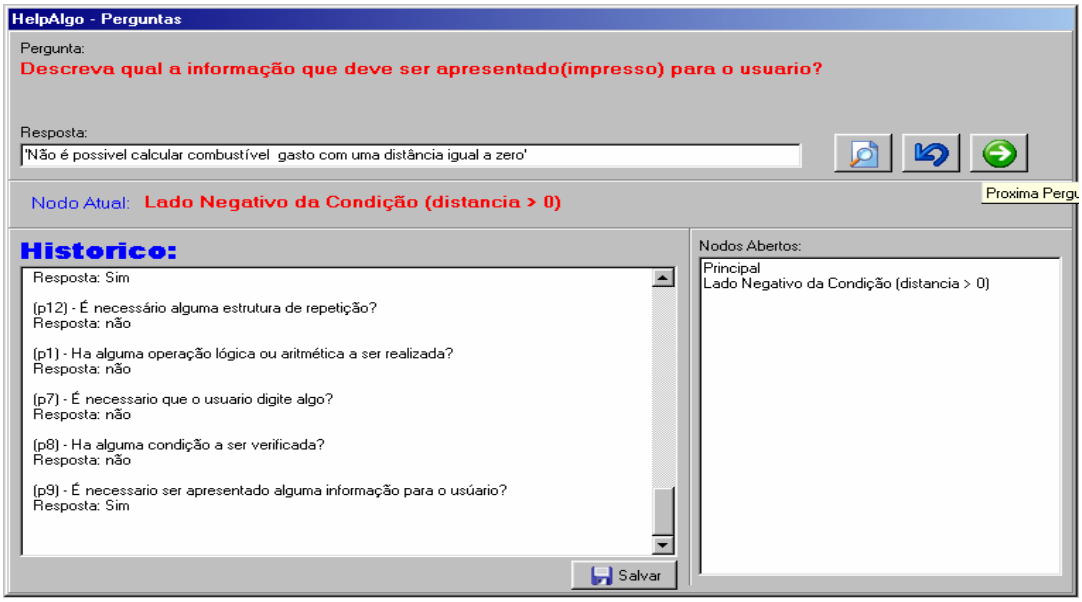
Figura 19 – Tela 12 do estudo de caso



Depois de informar o que vai ser apresentado para o usuário dentro do algoritmo do exercício, é mostrado novamente a tela final com os passos gerados até o momento, e logo após e feito novamente a 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 de responder está pergunta começa do

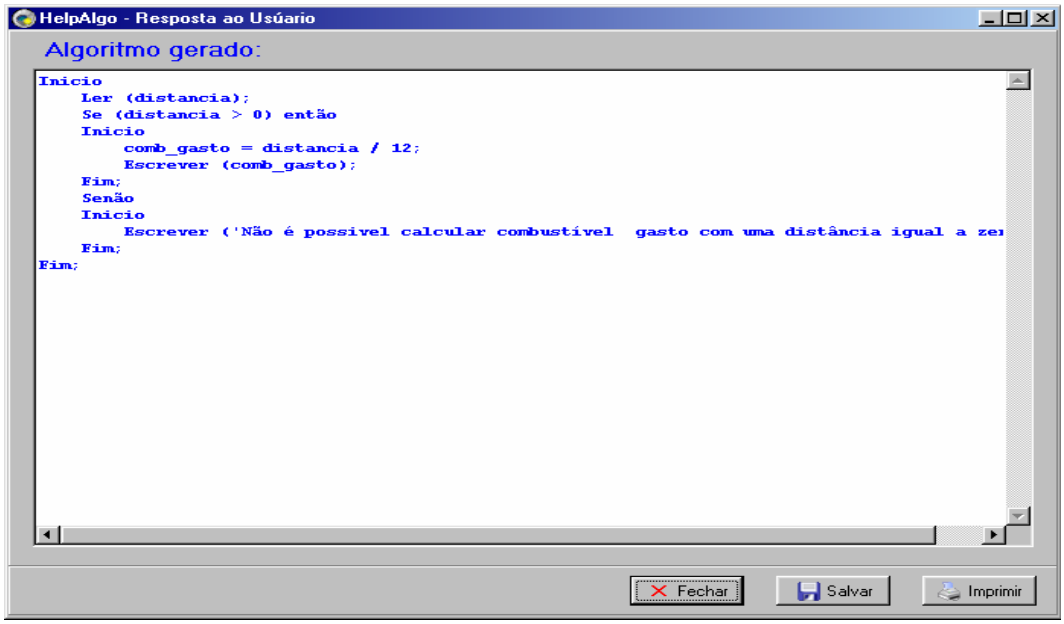
início novamente perguntando se tem repetição, onde a resposta será “não” e em todas as próximas perguntas também até chegar na pergunta que é descrito “É necessário informar algo para o usuário?”, onde será a resposta sim e então na próxima tela é informado o que vai ser apresentado, conforme a fig. 20:

Figura 20 – Tela 13 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. 21:

Figura 21 – Tela 14 do estudo de caso



6.7 RESULTADOS E DISCUSSÃO

Este trabalho que é uma extensão do trabalho desenvolvido em Mattos (1999) atingiu seus objetivos, conseguindo desenvolver utilizando a ferramenta de desenvolvimento Delphi 6 uma interface gráfica com o usuário, como pode ser verificado nas figuras que estão demonstrando o protótipo. Além da interface gráfica, foi atingido o objetivo de melhorar as perguntas da base de conhecimento desenvolvida no protótipo de Mattos (1999) e acrescentar as estruturas de repetição em algoritmos na mesma.

7 CONCLUSÕES

Pode-se considerar que a proposta apresentada em Mattos(1999) é interessante na medida em que caracteriza um esforço no sentido de instrumentalizar o processo de ensino-aprendizagem da disciplina de algoritmos – notadamente uma disciplina que apresenta um alto grau de desistências e reprovações.

O objetivo de implementar um protótipo em modo gráfico, visando converter o trabalho proposto por Mattos (1999) foi alcançado, pois foi como se pode verificar no trabalho, foi implementada uma interface gráfica com o usuário. Foi desenvolvido também um melhoramento nas estruturas de condição, onde existiam alguns problemas e em relação às estruturas de repetição foram formuladas com sucesso as perguntas e inseridas no grafo da base de conhecimento do sistema especialista.

7.1 LIMITAÇÕES

Cabe salientar que, esta proposta não conduz o aluno no sentido de gerar a solução correta para um determinado problema, pois o objetivo do protótipo não é este. Tanto isto é verdade que, caso o aluno não responda corretamente as questões, o protótipo apresentará um esboço de solução de acordo. Portanto, o aspecto mais importante que deve ser destacado refere-se ao fato que está solução permite que o aluno sistematicamente aprenda a pensar em termos de passos para solucionar problemas na disciplina de algoritmos.

7.2 EXTENSÕES

Como extensões para este trabalho sugere-se:

- A implementação das estruturas de vetores e matrizes nas perguntas do sistema especialista;
- Juntar este trabalho de sistemas especialistas com outro trabalho que utiliza raciocínio baseado em casos para melhorar as perguntas de acordo com os estudos de casos que estão armazenados na base do mesmo;
- Implementar um protótipo para demonstrar fluxogramas dos passos conseguidos das perguntas respondidas pelo usuário.

REFERÊNCIAS BIBLIOGRÁFICAS

CANTÙ, Marco. **Dominando o Delphi 6:** a bíblia. São Paulo: Makron Books, 2002.

CHAIBEN, Hamilton. **Inteligência Artificial na educação,** Curitiba, nov. [1999]. Disponível em: <http://www.cce.ufpr.br/~hamilton/iaed/iaed.htm>. Acesso em 12 mai. 2002.

CORNELL, Gary; STRAIN, Tony. **Delphi segredos e soluções**. São Paulo: Makron Books, 1995.

ENGO, Frank. **Como programar em Delphi 3**. São Paulo: Makron Books, 1997.

FORBELLONE, André Luiz Villar; EBERSPACHER, Henri Frederico. **Lógica de programação:** a construção de algoritmos e estrutura de dados. 2. ed. São Paulo: Makron Books, 2000.

FRANCIOSI, Beatriz Regina Tavares; MAÇADA, Débora Laurino. Informática na educação. In: Simpósio Brasileiro de informática na educação, 5., 1994, Porto Alegre. **Anais..**Porto Alegre, PUCRS, 1994.

GIARRATANO, J. C.; RILEY, G. **Expert system:** principies and programming. Boston: PWS, 1993.

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

HEINZLE, Roberto. **Protótipo de uma ferramenta para criação de sistemas especialistas baseados em regras de produção**. Florianópolis, 1995. 145 f. Dissertação (Mestrado em Engenharia de Produção) – Engenharia de Produção e Sistemas, Universidade Federal de Santa Catarina.

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: Congresso Ibero-americano de Educação Superior em Computação, 7., 1999, Florianópolis. **Anais...** Assunção, Universidad Autônoma de Assunción, 1999.

RABUSKE, Renato Antônio. **Inteligência artificial.** Florianópolis: Editora da UFSC, 1995.

RIBEIRO, Horácio da Cunha e Souza. **Introdução aos sistemas especialistas.** Rio de Janeiro: LTC, 1987.

RICH, Elaine; KNIGHT, Kevin. **Artificial Intelligence.** McGraw-Hill, 1993.

SALIBA, Walter Luiz Caram Saliba. **Técnicas de programação:** uma abordagem estruturada. São Paulo: Makron Books, 1993.

SCHMITT, Fátima Aparecida B. da S.. **Protótipo de um ambiente para o ensino de algoritmos**. Blumenau, 1998. 50 f. Monografia (Pós-graduação em Tecnologia de sistemas) – Tecnologia de sistemas, Universidade Regional de Blumenau.

SILVA, Carlos Alberto da. **Informática na Educação**. Canoinhas, 2000. 45 f. Monografia (Pós-graduação em tecnologias em desenvolvimento de sistemas) – Informática na educação, Universidade Regional de Blumenau.

SONNINO, Bruno. **Desenvolvendo aplicações com Delphi 5**. São Paulo: Makron Books, 2000.

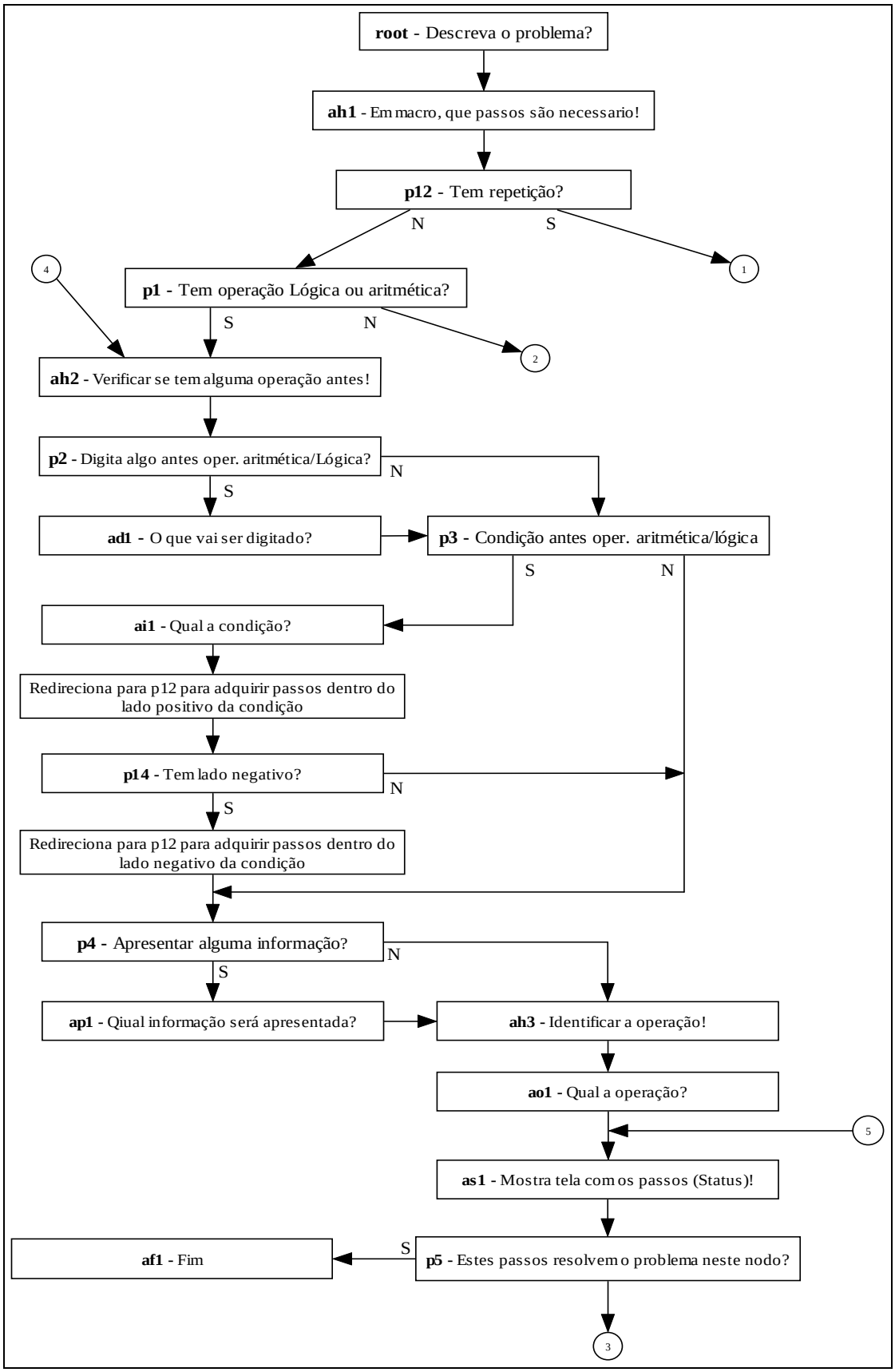
SUCHEUSKI, Maurício. **Desenvolvedor profissional:** algoritmos. Curitiba, 1996.

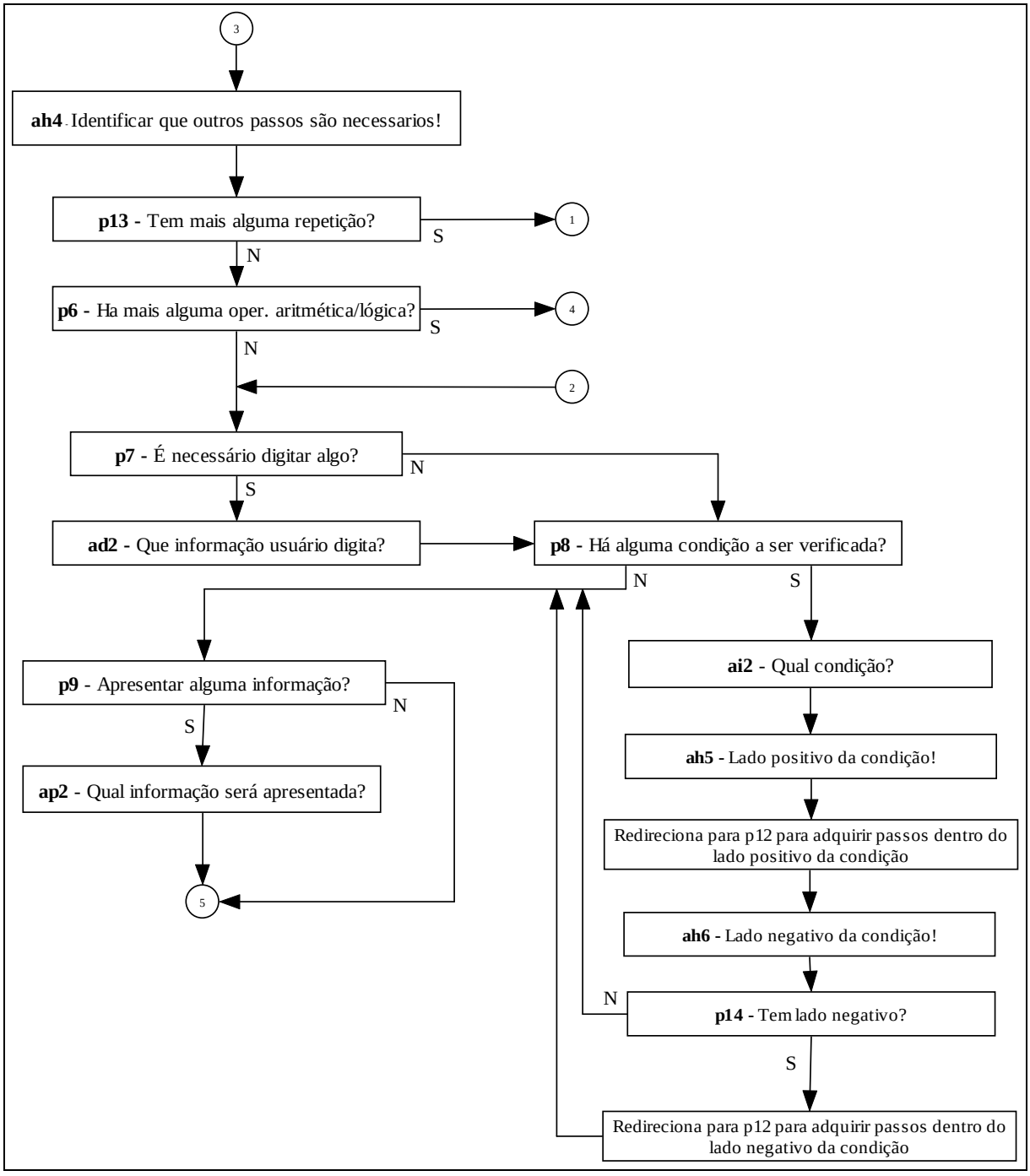
TAGLIARI, Alessandra. **Protótipo de um software para auxílio ao aprendizado de algoritmos.** 1996. (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

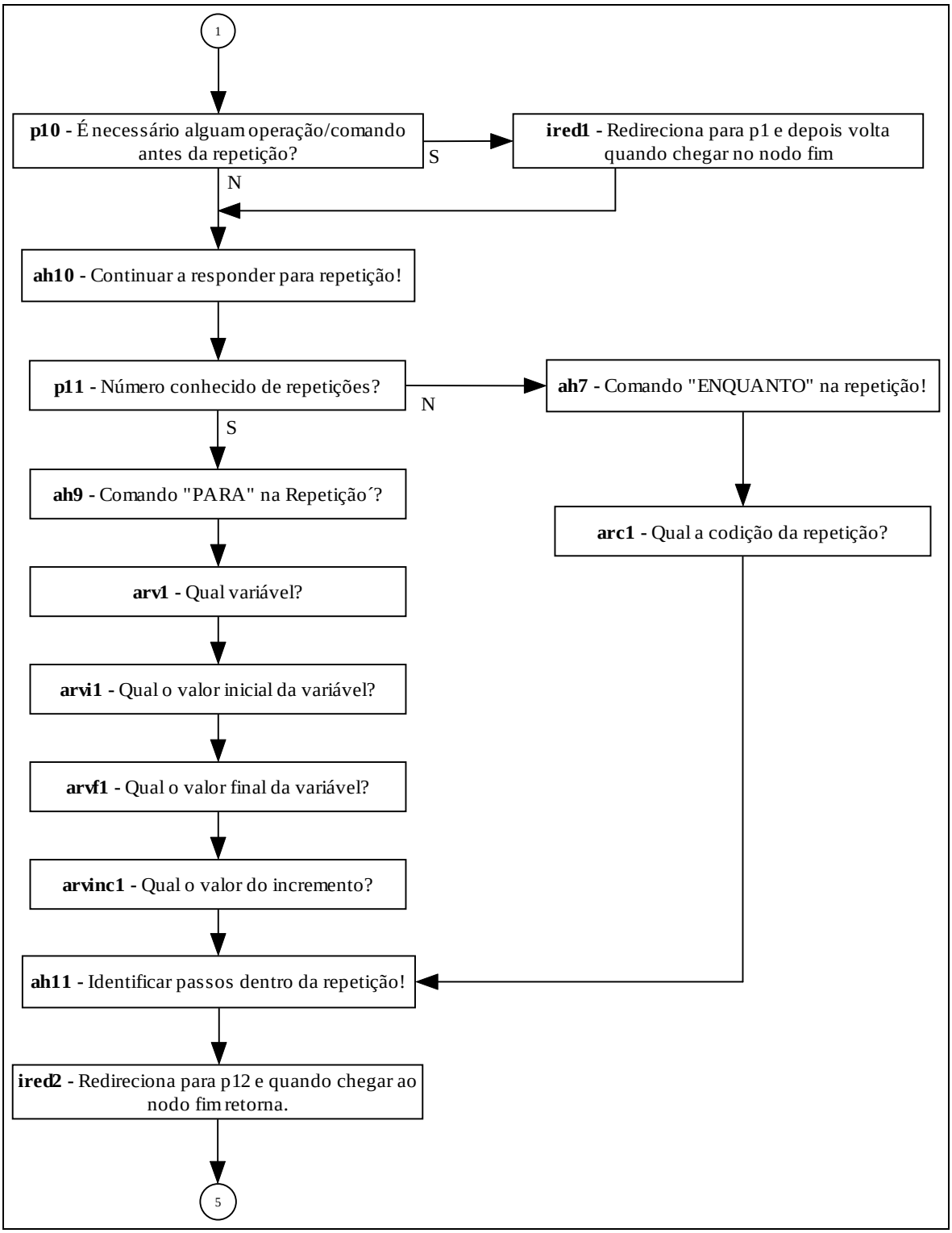
VALENTE, J. A. **Computadores e conhecimento:** repensando a educação. Campinas, Gráfica da Unicamp, 1993.

VENÂNCIO, Cláudio Ferreira. **Desenvolvimento de algoritmos:** Uma nova abordagem. São Paulo: Érica, 1997.

APÊNDICE 1 – GRAFO DE DECISÕES




3
ah4 - Identificar que outros passos são necessarios!
p13 - Tem mais alguma repetição?
S
1
N
p6 - Ha mais alguma oper. aritmética/lógica?
S
4
N
2
p7 - É necessário digitar algo?
N
S
ad2 - Que informação usuário digita?
p8 - Há alguma condição a ser verificada?
N
S
p9 - Apresentar alguma informação?
N
S
ap2 - Qual informação será apresentada?
5
ai2 - Qual condição?
ah5 - Lado positivo da condição!
Redireciona para p12 para adquirir passos dentro do lado positivo da condição
ah6 - Lado negativo da condição!
N
p14 - Tem lado negativo?
S
Redireciona para p12 para adquirir passos dentro do lado negativo da condição


1
p10 - É necessário alguam operação/comando antes da repetição?
S
ired1 - Redireciona para p1 e depois volta quando chegar no nodo fim
N
ah10 - Continuar a responder para repetição!
p11 - Número conhecido de repetições?
N
ah7 - Comando "ENQUANTO" na repetição!
S
ah9 - Comando "PARA" na Repetição´?
arc1 - Qual a codição da repetição?
arv1 - Qual variável?
arvi1 - Qual o valor inicial da variável?
arvf1 - Qual o valor final da variável?
arvinc1 - Qual o valor do incremento?
ah11 - Identificar passos dentro da repetição!
ired2 - Redireciona para p12 e quando chegar ao nodo fim retorna.
5