UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE EDITOR GRÁFICO DE FLUXOGRAMAS PARA REPRESENTAÇÃO DE COMANDOS DA LINGUAGEM PORTUGOL

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

LUÍS CARLOS SEIFERT DE SOUZA

BLUMENAU, JUNHO/1999

1999/1-37

PROTÓTIPO DE EDITOR GRÁFICO DE FLUXOGRAMAS PARA REPRESENTAÇÃO DE COMANDOS DA LINGUAGEM PORTUGOL

LUÍS CARLOS SEIFERT DE SOUZA

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. Wilson Pedro Carli — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Wilson Pedro Carli

_____

Prof. Maurício Capobianco Lopes

_____

Prof. Paulo Roberto Dias

DEDICATÓRIA

No decorrer deste curso sempre estive acompanhado de pessoas muito especiais que me apoiaram e incentivaram-me a chegar até aqui. Este trabalho é dedicado aos meus irmãos, minha namorada que sempre esteve ao meu lado, meu pai Aristides Lima de Souza que me deu todo apoio e condições para cursar a faculdade, e principalmente para minha mãe Eni Terezinha Seifert de Souza, que me apoiava e incentivava em todos os momentos de minha vida, mas que por escolha de Deus, partiu desta vida no decorrer deste semestre, restando a mim e minha família somente muita saudade e boas lembranças da pessoa que mais amávamos na nossa vida.

AGRADECIMENTOS

Agradeço à todos que de uma maneira ou de outra contribuíram para que chegasse até aqui, dando apoio e força para que continuasse este trabalho até o fim, apesar das dificuldades que tive no decorrer deste semestre.

Em especial agradeço a toda a minha família e amigos da minha terra, que sempre estiveram ao meu lado no decorrer da minha vida.

Ao meu orientador Wilson Pedro Carli e ao coordenador do TCC José Roque Voltolini pela compreensão, apoio e incentivo que me deram no decorrer deste semestre e na realização deste trabalho.

Agradeço ainda às pessoas da Senior Sistemas, que me auxiliaram em tudo que precisei para que este trabalho chegasse ao fim, em especial ao Adilson.

Por último, agradeço a todas as pessoas que direta ou indiretamente cruzaram a minha vida, tornando a minha vida digna de ser vivida, e dizer Valeu a Pena!!!

Obrigado à todos.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

RESUMO

Este trabalho descreve a construção de um protótipo que auxilia o estudo e a compreensão de lógica de programação através da construção ou edição de um fluxograma, e da geração do código Portugol que descreve as ações do fluxograma editado. Na elaboração deste protótipo, foi necessário o estudo dos principais comandos da pseudolinguagem de programação Portugol, estudo dos principais símbolos utilizados pela linguagem gráfica de diagrama de fluxo ou fluxograma, além do estudo do conceito de orientação a eventos e orientação a objetos, para auxiliar no desenvolvimento das ações que os componentes gráficos representam neste protótipo. Este trabalho teve como resultado, a construção de um protótipo de uma ferramenta que possibilita facilmente a criação ou edição de um fluxograma, além da geração do código em portugol referente ao fluxograma editado.

ABSTRACT

This work describes the construction of a prototype that aids the study and the understanding of programming logic through the construction or edition of a fluxograma, and of the generation of the code Portugol that describes the actions described of the fluxograma. In the elaboration of this prototype, it was necessary the study of the main commands of the pseudolinguagem of programming Portugol, study of the main symbols used by the language graphic of the flow diagram or fluxograma, besides the study of the orientation concept to events and orientation to objects, to aid in the development of the actions that the graphic components represent in this prototype. This work had as result, the construction of a prototype of a tool that facilitates the creation or edition of a fluxograma easily, besides the generation of the code in portugol regarding the published fluxograma.

1 INTRODUÇÃO

O aprendizado de lógica de programação atualmente é obtido através de exercícios práticos, através de exemplos da pseudolinguagem Portugol ou através da edição de fluxogramas, para uma melhor fixação e entendimento da lógica de programação.

Atualmente, existem diversas ferramentas de edição de fluxogramas, voltadas para as diversas áreas de desenvolvimento, sendo que algumas geram códigos fontes, relatórios, telas, tudo para facilitar o desenvolvimento de sistemas, mas, no entanto, nenhuma tem como objetivo o ensino da lógica de programação.

Desta forma, este trabalho tem por finalidade o entendimento de algoritmos, permitindo ao aluno editar um fluxograma sem se preocupar com a sintaxe e a estruturação de comandos, pois o protótipo gerará o código fonte, tornando o entendimento facilitado da lógica de programação, bem como da estruturação sintática da linguagem portugol.

1.1 MOTIVAÇÃO

No primeiro semestre do curso de Ciências da Computação em 1993, teve-se uma disciplina chamada Algoritmos, a qual ensina a lógica de programação. Nesta disciplina percebe-se muita dificuldade no entendimento de lógica de programação, pois na época não tinha-se tantas salas de laboratórios como tem-se hoje, ou melhor tinha-se apenas um laboratório com cerca de 25 computadores, que eram utilizados por todos os cursos da universidade e por esse motivo, o aprendizado da lógica era prejudicado, pois os exercícios eram feitos somente em sala, utilizando-se somente o quadro negro para a resolução de exercícios e a representação de algoritmos e fluxogramas.

Com o propósito de auxiliar os alunos que estão entrando na faculdade, este trabalho tem por objetivo, utilizar os recursos de um ambiente visual, permitindo a edição de fluxogramas para um melhor entendimento da lógica de programação, além de gerar o código na pseudolinguagem Portugol, auxiliando em muito o aluno no entendimento da sintaxe e da estruturação de comandos, representando muito bem as tarefas e ações que a aplicação deverá realizar em sua execução.

1.2 OBJETIVOS

O objetivo principal deste trabalho de conclusão de curso é a especificação e implementação de editor gráfico de fluxogramas para representação da pseudolinguagem Portugol, tornando um facilitador para o ensino das disciplinas que ensinam a lógica de programação, visto que a representação gráfica através de fluxogramas se torna mais legível e mais compreensível que a representação textual.

1.3 ORGANIZAÇÃO DO TEXTO

No capítulo 1, encontra-se a introdução juntamente com a motivação e os objetivos deste trabalho.

No capítulo 2, encontra-se a definição da pseudolinguagem Portugol e seus comandos.

No capítulo 3, encontra-se a definição da linguagem gráfica diagrama de fluxo com seus símbolos.

No capítulo 4, encontra-se a definição das ferramentas, que foram utilizadas para a definição e implementação deste trabalho.

No capítulo 5, encontra-se a especificação do protótipo e sua implementação.

No capítulo 6, está a apresentação do protótipo.

No capítulo 7 encontra-se a conclusão deste trabalho.

2 PORTUGOL

Uma linguagem de programação é, segundo [GUI95] "uma técnica de notação para programar, com a intenção de expressão de um raciocínio (Algoritmo) e a execução automática de um algoritmo no computador", e é "uma pseudolinguagem de programação (simbiose de Português com o *ALGOL* e *PASCAL*)". A idéia desta linguagem é dar liberdade ao programador ou projetista para que possa pensar no problema e não na linguagem que este vai desenvolver, e por outro lado, que não fique distante desta mesma linguagem. A pseudolinguagem Portugol pode ser utilizada na definição, criação, desenvolvimento e na documentação de um programa.

Em toda linguagem, a construção de seu código envolve a sintaxe e a semântica. A sintaxe segundo [FRA86] é :

a) "Conjunto de relações entre caracteres ou grupo de caracteres, independentemente de seus significados ou de seu modo de interpretação e uso";

b) "Estrutura das expressões de uma linguagem ou as regras que a determinam. Por ex., um universo de símbolos que recebeu uma atribuição semântica (significado) tem sua composição de frases portadoras de informação regida pela sintaxe".

A semântica é:

a) "Ciência do significado";

b) "Parte da semiótica que trata da definição do significado que é prescrito a uma sentença, em uma linguagem, por seu originador (a pessoa que fala ou que escreve)".

2.1 TIPOS BÁSICOS DE DECLARAÇÃO DE VARIÁVEIS

No Portugol, tem-se quatro tipos básicos que podem ser utilizados:

a) Inteiro: qualquer número inteiro, negativo, nulo ou positivo. Ex: -15, 0, 534;

b) Real: qualquer número real, negativo, nulo ou positivo. Ex: -5.54, 30, 0.01;

c) Caractere: qualquer conjunto de caracteres alfanuméricos. Ex: "Hoje", "Tarde";

d) Lógico: conjunto de valores FALSO ou VERDADEIRO em notações lógicas.

2.2 DEFINIÇÃO DE VARIÁVEIS

A definição de variáveis é uma associação de um tipo a um nome de variável. Exemplo:

Inteiro : Valor01;

Real : Valor02, Valor03;

Caractere: Frase, Nome;

Lógico : Tem;

Segundo [GUI95], a semântica de uma declaração de variáveis corresponde a criação de locais na memória rotulada com o nome da variável (identificador) e marcada com o tipo de valores que esta variável pode conter e operações.

Assim, Valor01 é um nome de um local na memória que só pode conter valores do tipo *inteiro*, do mesmo modo, Valor02 e Valor03 são locais de memória que só podem conter valores *reais*, e a variável Tem só pode conter *Falso* ou V*erdadeiro*, e as variáveis Frase e Nome só podem conter conjuntos de *caracteres*.

2.3 COMANDOS BÁSICOS

O Portugol possui uma sintaxe que representa as ações. Para representar estas ações na pseudolinguagem, ou a sintaxe de construção dos programas, utiliza-se de comandos de atribuição, operadores aritméticos, operadores lógicos, operadores relacionais, blocos e comandos básicos de controle.

Para os comandos de atribuição de valores a uma variável, é utilizado o símbolo de atribuição ←.

A sintaxe do comando é: **identificador ← expressão;**

Nos operadores aritméticos são utilizados os quatro símbolos das operações básicas “+”, “-“, “*”,”/”, além do símbolo de exponenciação “**”. O nome de algumas funções matemáticas também são utilizados, como: sen(x), cos(x), tg(x), arctg(x), exp(x), abs(x), etc.

É utilizado também como nome de operador:

mod, exemplo: *xval mod x* (resto da divisão de *xval* por *x*).

div ou “÷”, exemplo *x div xval* (quociente da divisão inteira de *x* por *xval)*;

Os operadores lógicos são utilizados como conectivos lógicos, que são:

a) *e* – utilizado para conjunção;

b) *ou* – utilizado para disjunção (não exclusivo);

c) *não* – utilizado para a negação.

Para os operadores relacionais são utilizados os conectivos “=”, “<>(≠)”, “>”, “>=”, “<”, “<=”.

Dentre os comandos básicos de controle, existem os blocos, que são um conjunto de comandos com uma função bem definida. Servem para a definição dos limites onde as variáveis declaradas em seu interior são conhecidas.

Um Exemplo de bloco de comando está na figura 1.

```
Início
    <declaração de variáveis>
    <comandos>
Fim
```

Figura 1 – Bloco de Comandos.

A seqüência simples é um conjunto de comandos, separados por ponto e vírgula (;), que são executados seqüencialmente, de cima para baixo está representado na figura 2.

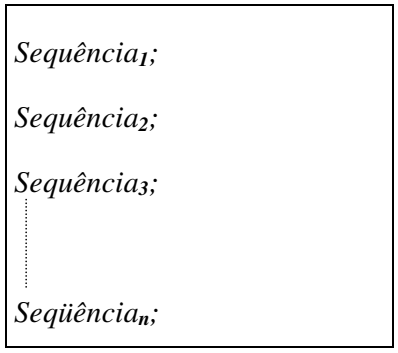


Figura 2: Seqüência de Comandos.

Os comandos de alternativa são usados em uma comparação, onde uma ação a ser executada, depende de um teste para verificar se a alternativa está correta.

Um exemplo de alternativa simples está representado na figura 3.

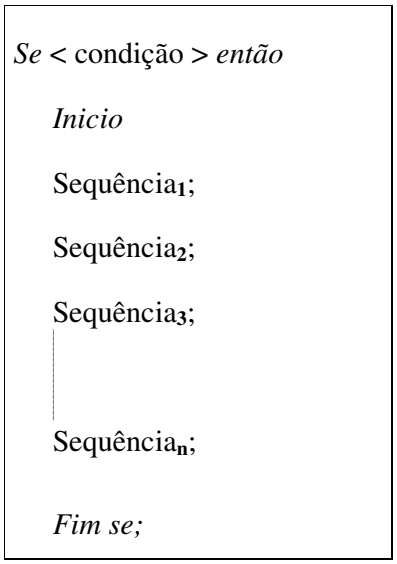
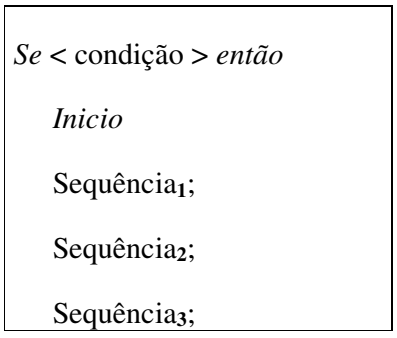


Figura 3: Alternativa Simples.

Um exemplo de alternativa composta está apresentado na figura 4.



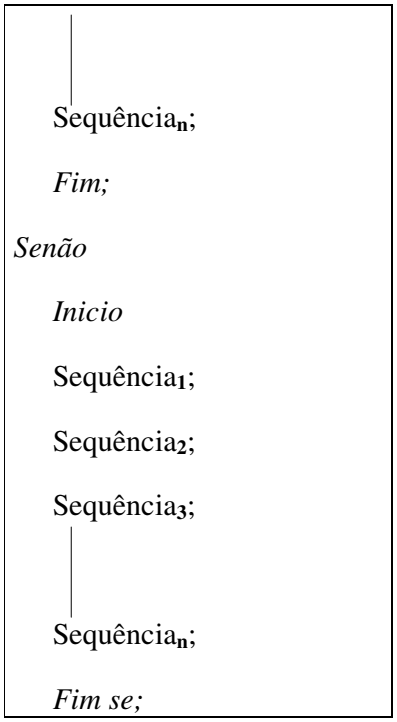


Figura 4 – Alternativa Composta.

O comando de repetição enquanto-faça é utilizado para fazer um *loop* de comandos, ou seja, fazer com que um bloco de comandos seja executado determinado número de vezes, até que a < condição > seja falsa, o exemplo deste comando pode ser visto na figura 5.

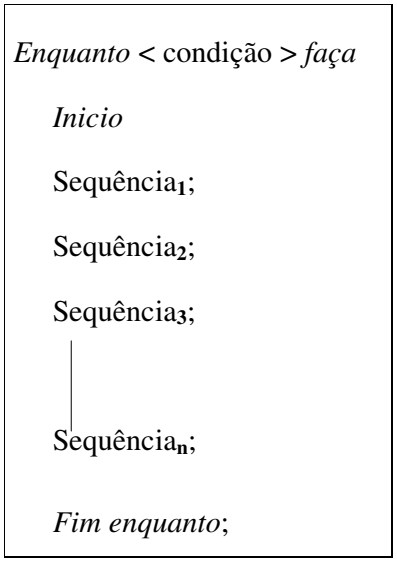


Figura 5 – Repetição.

Enquanto o valor da < condição > for verdadeiro, as ações dos comandos serão executadas, caso seja falso estes comandos não serão executados.

Os comandos de entrada representam a obtenção de dados para a execução do problema, e a saída representa os resultados gerados por esta execução.

Alguns exemplos de comandos de entrada e saída:

a) impressão dos dados na tela;

b) impressão dos dados na impressora;

c) leitura de dados vindos do teclado;

d) leitura de dados vindos de um arquivo.

A sintaxe utilizada para a leitura e gravação de dados é:

a) *Leia (Val1,Val2,Val3,... Valn), ou*

b) *Imprima (Val1,Val2).*

3 DIAGRAMAS DE FLUXO

Para uma melhor compreensão de etapas necessárias a execução de um algoritmo, foi desenvolvido uma linguagem gráfica que recebeu o nome de Diagrama de Fluxo ou Fluxograma, sendo utilizados para representar graficamente as diversas ações que o computador deverá cumprir para executar uma tarefa proposta pelo programador.

A simbologia utilizada neste trabalho está baseada em [GUS77], e na simbologia tradicional descrita por [GUI95].

3.1 SÍMBOLOS

Nos dias de hoje existem diversos símbolos dedicados a diversas formas de representação de fluxogramas como, por exemplo, análise de sistemas e organização e métodos. Para este trabalho, apenas serão usados símbolos utilizados nos fluxogramas dedicados exclusivamente à programação.

3.1.1 INÍCIO E FIM

O símbolo utilizado para inicializar um programa é apresentado na figura 6. Este símbolo também serve para finalizar ou terminar a execução de um programa.

Figura 6 – Símbolo de Início e Fim.

3.1.2 PROCESSAMENTO OU FUNÇÕES

O símbolo de processamento ou função é utilizado para representar as ações que o programa deve executar (funções), para a elaboração do fluxo correto de informações que o fluxograma deve realizar no decorrer de sua execução. O símbolo de processamento ou funções está descrito na figura 7.

Figura 7 - Símbolo de Processamento, funções;

Um exemplo do uso da simbologia de inicio e fim e processamento está apresentado na figura 8.

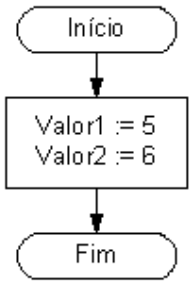


Figura 8 – Fluxo de representação do símbolo de início, fim e de processamento ou função;

O símbolo de decisão é utilizado para representar ações ou situações mais complexas, onde é necessário para representar uma operação de decisão, que define um caminho sempre que houver mais de um. O símbolo de decisão está descrito na figura a 9.

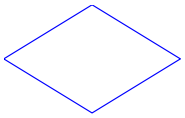


Figura 9 - Símbolo de Decisão.

Um exemplo do uso da simbologia de decisão está representado na figura 10.

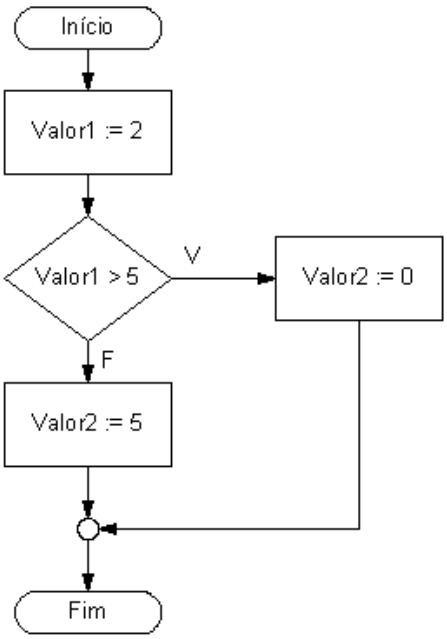


Figura 10 – Fluxograma para representar o símbolo de decisão;

No fluxo apresentado na figura 10 é feito a leitura de um valor (valor1), e após é feito um teste para ver se o valor1 lido é maior que 5. Se este valor for maior que 5 a variável Valor2 receberá zero, e se for menor ou igual, será atribuido 5 a variável Valor2.

3.1.3 IMPRESSÃO

O símbolo de impressão é utilizado para a saída de dados, e representa os relatórios impressos do sistema. O símbolo de impressão está representado na figura 11.

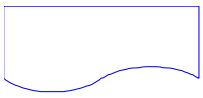


Figura 11 - Símbolo de Impressão.

Um exemplo do uso da simbologia de impressão está representado na figura 12.

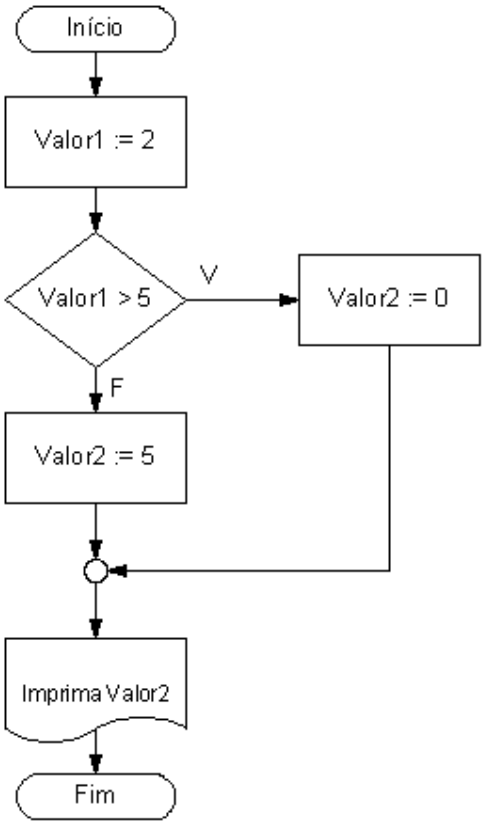


Figura 12 – Fluxograma para representar o símbolo de Impressão;

No fluxo apresentado na figura 12 é atribuído 2 a variável Valor1, e após é feito um teste para ver se o valor1 lido é maior que 5. Se este valor for maior que 5 a variável Valor2 receberá zero, e se for falso será atribuido 5 a variável Valor2, que após esta atribuição o conteúdo da variável Valor2 será impresso e meio físico.

3.1.4 LEITURA

O símbolo de leitura é utilizado para a entrada de valores no sistema, podendo ser através do teclado, mouse ou entrada de dados. Esta simbologia faz parte da representação tradicional de Fluxogramas descrita por [GUI95], representada na figura 13.

Figura 13 - Símbolo de Leitura.

Um exemplo da simbologia de leitura está representado na figura 14.

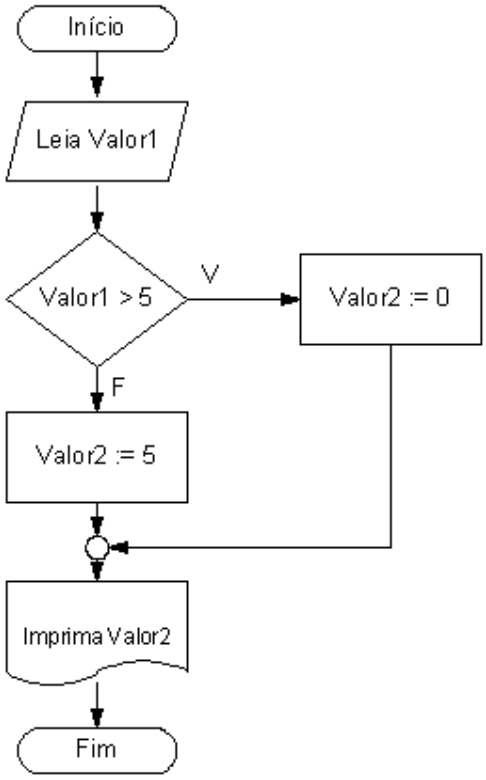


Figura 14 – Fluxograma para representa o símbolo de Leitura;

No fluxo apresentado na figura 14 é feito a leitura de um valor na variável Valor1, e após é feito um teste para ver se o valor1 lido é maior que 5. Se este valor for maior que 5 a variável Valor2 receberá zero, e se for falso será atribuido 5 a variável Valor2, que em seguida, este valor será impresso em meio físico.

3.1.5 REPETIÇÃO

O símbolo de repetição é utilizado para representar a execução de um bloco de comandos em um determinado número de vezes, ou seja, executar um bloco de comandos até que a condição imposta pelo desenvolvedor seja falsa. Esta simbologia utilizada neste item faz parte da simbologia tradicional de fluxogramas, descrita por [GUI95], representada na figura 15.

Figura 15 - Símbolo de Repetição.

Um exemplo do uso da simbologia de repetição está representado na figura 16.

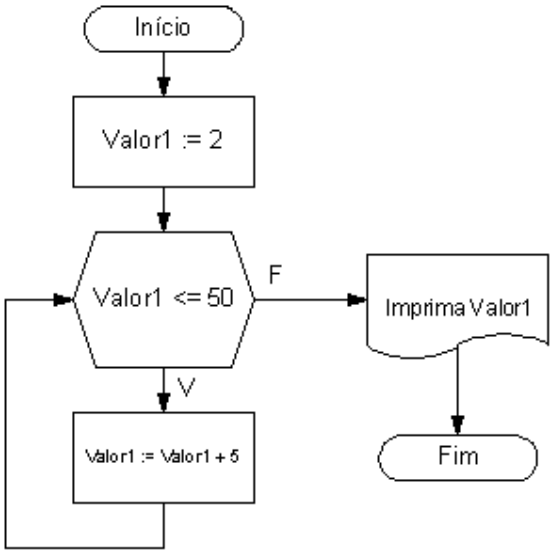


Figura 16 – Fluxograma para representar o símbolo de Repetição**.**

No fluxo apresentado na figura 16 é atribuído o valor 2 à variável Valor1, após esta atribuição, a execução entra em *looping*, ou seja, será acumulado 5 em Valor1 enquanto for verdadeira a condição (Valor1 <= 50). Caso a condição seja falsa, será impresso o conteúdo da variável Valor1.

3.1.6 VISUALIZAÇÃO

O símbolo de visualização é utilizado para representar a saída de dados, ou seja, visualizar de alguma forma o resultado ou ação do sistema. Os símbolos de visualização adotados por [GUS77] e [GUI95] diferem-se um do outro, tanto no formato como nas ações que cada um representa, o símbolo de visualização adotado por [GUS77] representa a visualização ou a impressão dos dados em uma impressora, já o símbolo adotado por [GUI95] representa a visualização dos dados em vídeo.

A simbologia de visualização adotada por [GUS77] representa a saída em vídeo de um resultado do sistema, que está representada na figura 17.

Figura 17 - Símbolo de Visualização segundo [GUS77].

Um exemplo do uso da simbologia de visualização está representado na figura 18.

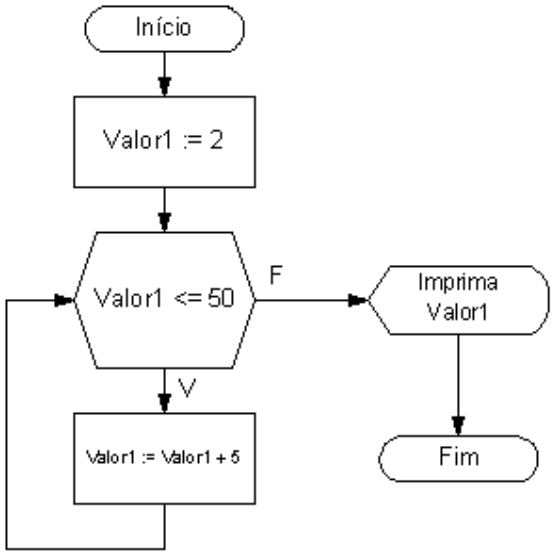


Figura 18 – Fluxograma para representar o símbolo de Visualização**.**

No fluxo apresentado na figura 18 é atribuído o valor 2 à variável Valor1, após esta atribuição, a execução entra em *looping*, ou seja, será acumulado 5 em Valor1 enquanto for verdadeira a condição (Valor1 <= 50). Caso a condição seja falsa, será mostrado no vídeo o conteúdo da variável Valor1.

Na simbologia tradicional adotada por [GUI95], a representação da visualização de uma ação é feita pelo mesmo símbolo utilizado pela leitura, representando não somente a visualização em vídeo mais também a visualização através de outros meios como impressoras. A simbologia de visualização está descrita na figura 19.

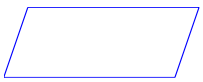


Figura 19 - Símbolo de visualização segundo [GUI95].

Um exemplo da simbologia de Visualização segundo [GUI95] está descrito na figura 20.

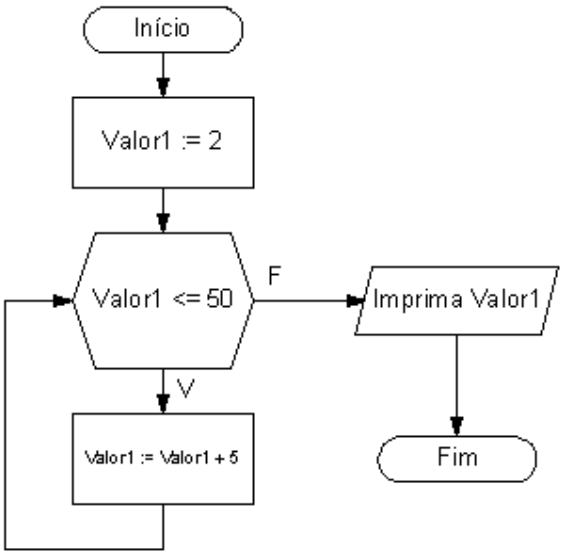


Figura 20 – Fluxograma para representar o símbolo de visualização segundo [GUI95].

No fluxo apresentado na figura 18 é atribuído o valor 2 à variável Valor1, após esta atribuição, a execução entra em *looping*, ou seja, será acumulado 5 em Valor1 enquanto for verdadeira a condição (Valor1 <= 50). Caso a condição seja falsa, o conteúdo da variável Valor1 poderá ser impresso tanto em vídeo como em impressora.

4 FERRAMENTAS

O ambiente de edição dos fluxogramas deste trabalho está baseado no Protótipo de [SOU98], como por exemplo: a disposição dos botões auxiliares (exclusão, seleção e ligações entre controles), botões de operações lógicas, composição do menu (todas as operações executadas pelos botões estão no menu), e por último o conceito de geração da expressão lógica, que gera a expressão conforme a edição do circuito, no caso deste protótipo será gerado o código em portugol ao contrário da expressão lógica.

Um outro trabalho correlato está descrito em [FER98], que descreve o software Construtor, no qual pode-se editar um fluxograma, e a partir deste fluxo pode-se gerar código em Pascal, C, Clipper e Pseudocódigo, além de permitir a impressão de resultados e acompanhar os valores das variáveis.

O ambiente de programação Delphi 4, que pode ser observado na figura 21, foi utilizado para o desenvolvimento deste protótipo, pois seu uso adapta-se perfeitamente às características do protótipo, além de ser uma ferramenta amigável para o desenvolvimento.

Segundo [DAO95] e [SWA96] o ambiente de desenvolvimento do Delphi 4 está de acordo com o padrão Windows 95, e em sua tela principal (figura 21), pode-se controlar várias janelas como por exemplo a *Object Inspector, Form* e *Code Editor*. As características do ambiente de programação Delphi segundo [DAO95] e [SWA96] são:

a) linguagem descendente do Turbo Pascal;
b) possui programação orientada a objetos e em eventos;
c) linguagem compilada e não interpretada;
d) padrão SQL em banco de dados;
e) conectividade através de ODBC.

Este ambiente também disponibiliza alguns recursos como:

a) ambiente personalizado para o desenvolvimento;
b) programas compilados;
c) reutilização de componentes;
d) permite criar bibliotecas de funções;
e) assistente para criação de formulários;
f) suporte a OCX;

g) recursos para acesso a banco de dados.

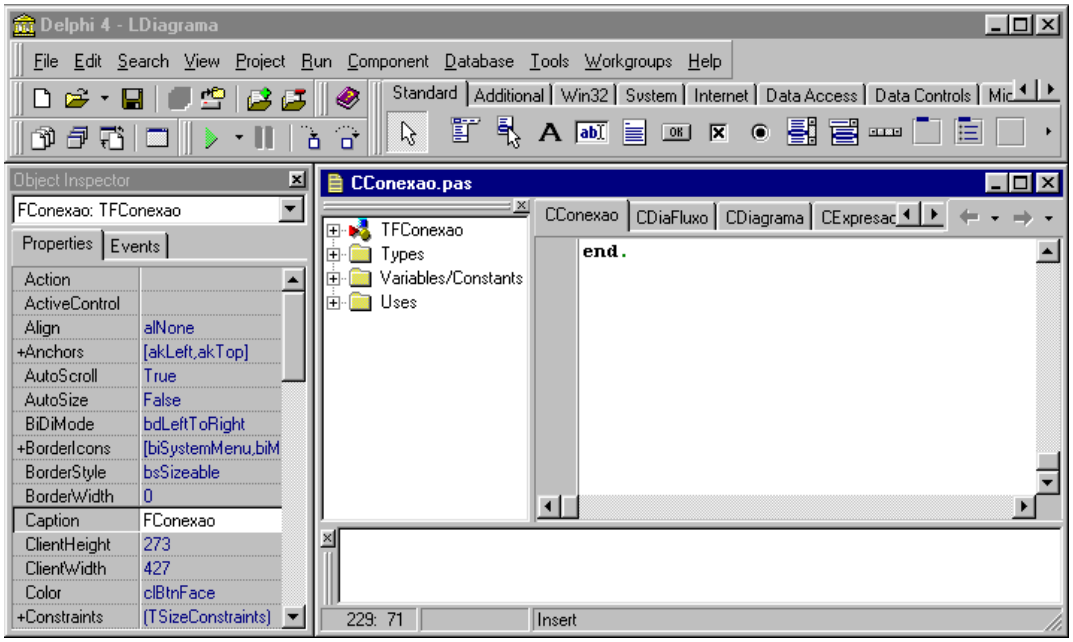


Figura 21 - Tela principal do Ambiente Delphi 4.

5 ESPECIFICAÇÃO DO PROTÓTIPO

A definição do sistema proposto foi definido segundo a análise orientada a objeto, baseada na metodologia utilizada por [OLI96] e [WIN93]. Nesta metodologia os objetos são representados por um retângulo dividido em três partes, onde a primeira parte identifica o objeto pelo seu nome, a segunda parte mostra as propriedades do objeto, e a terceira parte representa os métodos do objeto (figura 22).

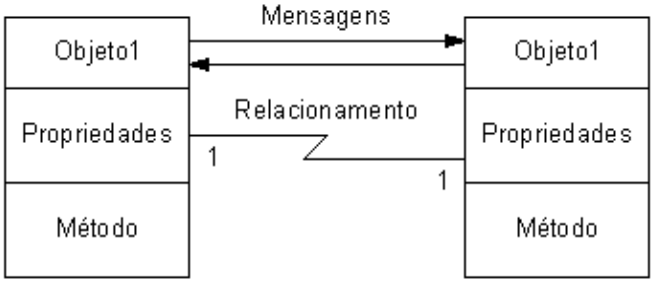


Figura 22– Simbologia utilizada por [OLI96].

A simbologia apresentada na figura 22 está baseada na metodologia de Coad & Yordon que descrevem 5 passos para a realização da análise de um sistema orientado a objetos:

a) obtenção das classes e objetos;

b) representação das estruturas de hierarquia de classes;

c) identificação dos assuntos;

d) definição dos atributos;

e) definição dos serviços.

Na orientação a objeto existem as mensagens (comunicação entre processos), que está sendo representada por duas setas na figura acima.

O modo de relacionamento entre os objetos também pode ser visualizado na figura 22, indicando que o relacionamento entre os objetos pode ser somente de 1(um) para 1(um).

Na figura 23 está o diagrama que representa os relacionamentos dos objetos do protótipo.

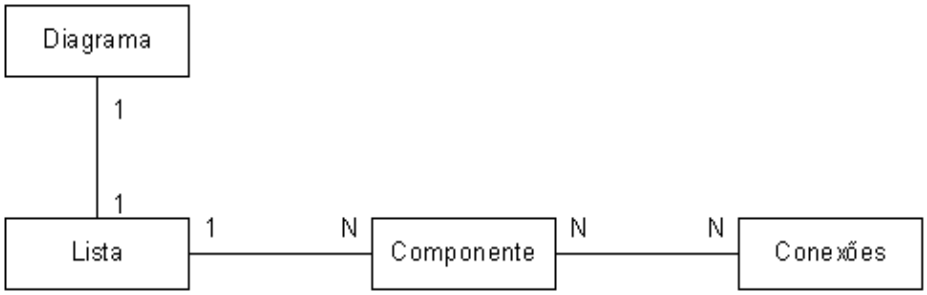


Figura 23 - Diagrama de relacionamento dos objetos.

No diagrama demonstrado na figura 23, o objeto Diagrama (representa o objeto base da implementação) possui uma relação de um para um com o objeto lista (responsável por armazenar os componentes do diagrama), ou seja, o objeto Diagrama possui uma lista de componentes. O objeto lista pode conter vários objetos Componentes (objeto que representa as ações que os símbolos do diagrama representam), e cada objeto componente pode conter vários objetos chamados conexões (objetos que fazem a conexão entre os componentes que compõem o diagrama editado).

5.1 DESCRIÇÃO DOS OBJETOS

Neste item serão descritos todas as propriedades e os métodos dos objetos que foram apresentados no diagrama de relacionamento dos objetos.

5.1.1 OBJETO DIAGRAMA

Este objeto considera-se como objeto base, pois faz a união de todos os componentes utilizados no protótipo. É através deste objeto representado na figura 24, que será definido todo diagrama, além da obtenção ou geração do código em portugol.

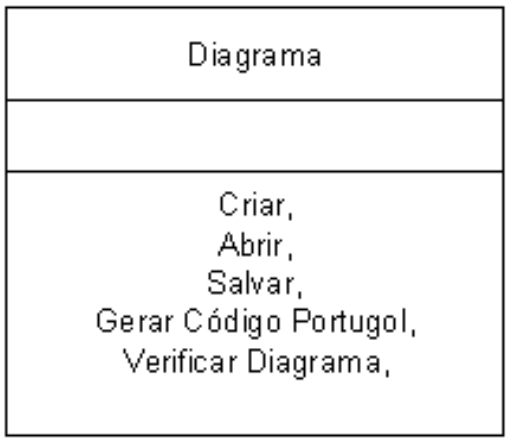


Figura 24 – Objeto Diagrama

Os métodos do objeto diagrama são:

a) criar: cria um novo diagrama em branco;
b) abrir: cria um diagrama a partir de um arquivo gravado em disco;
c) salvar: grava em um arquivo em disco as informações referentes ao diagrama;
d) gerar código portugol: gera ou monta o código portugol com base no diagrama montado;
e) verificar diagrama: verifica através da lista de componentes, se os mesmos estão conectados, e se estas conexões estão entre um início e fim.

5.1.2 OBJETO LISTA

Este objeto é responsável por armazenar os componentes do diagrama, sendo utilizado principalmente para percorrer todos os componentes do diagrama (figura 25).

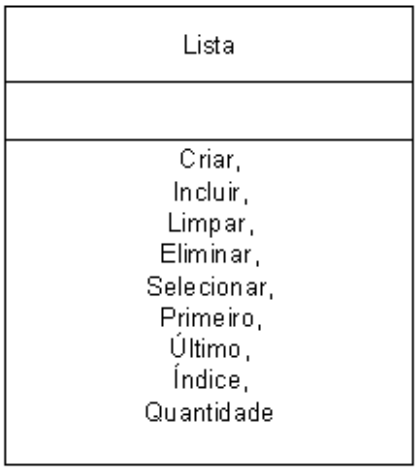


Figura 25 – Objeto lista

Os métodos do objeto lista são:

a) criar: cria uma lista de componentes do diagrama;
b) incluir: inclui um componente do diagrama na lista de componentes;
c) limpar: exclui todos os componentes da lista de componentes;
d) eliminar: exclui um elemento determinado da lista de componentes;
e) selecionar: retorna um determinado elemento, na lista de componentes;
f) primeiro: retorna o primeiro elemento da lista de componentes;
g) último: retorna o último elemento da lista de componentes;
h) índice: retorna um índice de um elemento da lista de componentes;
i) quantidade: retorna a quantidade total de elementos da lista.

5.1.3 OBJETO COMPONENTE

Este objeto apresentado na figura 26, representa as ações que os símbolos do diagrama representam.

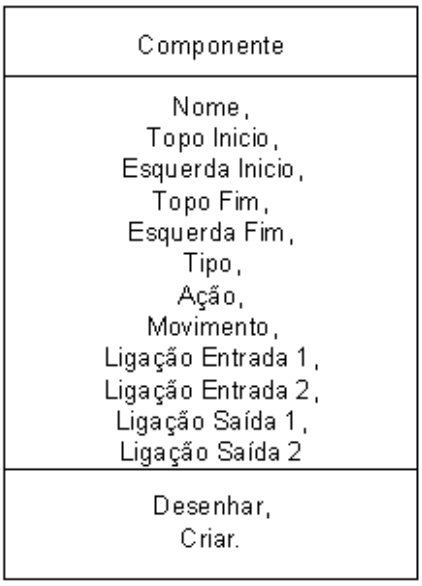


Figura 26 – Objeto componente.

As propriedades do objeto componente são:

a) nome: é o identificador do componente. Identifica as ações que serão executadas exemplo (Se, Enquanto, Atribuição, etc);
b) topo início: indica o valor do ponto base horizontal (Y) onde começará a ser desenhado o componente. Este ponto é indicado em relação ao canvas (área de desenho dos componentes gráficos) do fluxograma;
c) esquerda início: indica o valor do ponto base vertical (X) onde começará a ser desenhado o componente. Este ponto é indicado em relação ao canvas do fluxograma;
d) topo fim: indica a quantidade de pontos de deslocamento, em relação ao ponto de início do desenho do componente (topo início), que é contado de cima para baixo;
e) esquerda fim: indica a quantidade de pontos de deslocamento, em relação ao ponto de início do desenho do componente (esquerda início), que é sempre contado da esquerda para a direita;
f) tipo: indica qual o tipo de componente, representando a ação que o componente representa, além de seu desenho;
g) ação: representa a ação que o componente executará. Esta ação é informada pelo usuário;
h) movimento: indica se o componente está sendo movimentado;
i) ligação entrada 1: indica os parâmetros da primeira entrada do componente;

j) ligação entrada 2: indica os parâmetros da segunda entrada 2 do componente, para aqueles componentes que possuem mais de uma entrada;
k) ligação saída 1: indica o primeiro parâmetro da saída do componente.
l) ligação saída 2: indica o segundo parâmetro da saída, para os componentes que possuem mais de uma saída.

Os métodos do objeto componente são:

a) desenhar: desenha o símbolo do componente, baseado no tipo do mesmo;
b) criar: cria um novo componente.

5.1.4 OBJETO CONEXÕES

O objeto conexões apresentado na figura 27 faz a ligação entre os componentes do diagrama, dando sentido lógico e seqüencial as ações que os componentes representam.

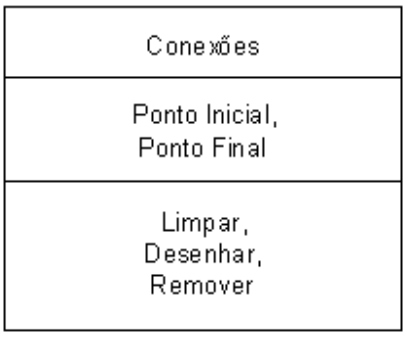


Figura 27– Objeto conexões

As propriedades do objeto conexões são:

a) ponto inicial: armazena o ponto inicial da conexão;
b) ponto final: armazena o ponto final da conexão.

Os métodos do objeto conexão são:

a) limpar conexão: zera ou limpa os pontos da conexão;
b) desenhar: desenha as linhas que fazem a conexão entre os componentes;
c) remover: apaga as linhas que fazem a conexão entre os componentes.

6 IMPLEMENTAÇÃO DO PROTÓTIPO

Este protótipo teve sua implementação no ambiente *Delphi* 4, onde foi dividido em quatro módulos básicos na sua implementação, sendo um módulo para cada objeto. Cada módulo foi desenvolvido segundo as especificações mostradas anteriormente. A seguir tem-se uma visão de cada módulo deste protótipo.

6.1 MÓDULO DIAGRAMA

Este é o principal módulo do protótipo, correspondendo ao objeto diagrama. É neste módulo que está a tela de edição do fluxograma com seus componentes de edição, sendo este módulo responsável por todos os controles do protótipo.

Neste módulo, como nos demais, todas as ações são realizadas através de eventos (escolha de uma opção no menu, clique do mouse), sendo o principal evento o clique do mouse sobre a área de edição do fluxograma, o qual pode executar ações como criar, excluir, mover ou conectar componentes de acordo com a opção desejada pelo usuário do sistema. Tem-se no quadro 1 uma parte do algoritmo utilizado quando for feito a criação de um componente.

```
If      Criar then
        Begin
        inc(Quantidade);
        Fdiagrama.Novo;
        Case     Tipo     of
              Pomove     :  Fdiagrama.DiagramaMove;
              Poauxiliar  :  Fdiagrama.DiagramaAuxiliar;
              Poinicio    :  Fdiagrama.DiagramaInicio;
              ...
              Porepeticao :  Fdiagrama.DiagramaRepeticao;
              Pofim       :  Fdiagrama.DiagramaFim;
            end;
      FDiagrama.Desenhar(X,Y);
      end;
```

Quadro 1 - Algoritmo de criação de componentes.

6.2 MÓDULO LISTA

Este módulo é responsável pelo armazenamento dos componentes incluídos pelo usuário no fluxograma, em uma lista de componentes, isto é, armazenar todas as propriedades do componente incluído em um registro.

No caso do uso de algumas ações realizadas pelo usuário no fluxograma, como por exemplo: excluir, mover ou incluir componentes. As ações incluir e excluir, exigem uma inclusão ou exclusão do componente da lista de componentes, e a ação mover, exige uma alteração das propriedades do componente que foi movido pelo usuário. No quadro 2 pode-se ver o algoritmo para incluir e excluir um componente da lista de componentes.

```
Procedure TFLista.Eliminar(Indice : Byte);
Begin
        Lista[Indice].Ativo := false;
End;

Procedure TFLista.Incluir(Item : Componente) : Byte;
Begin
        Inc(Quantidade);
        Lista[Quantidade] := Item;
End;
```

Quadro 2 - Algoritmos de inclusão e exclusão da lista.

6.3 MÓDULO COMPONENTE

Este módulo trata de todas as ações relacionadas a cada componente incluído na tela de edição do protótipo, sendo este módulo do protótipo, o responsável pela edição ou desenho dos componentes na tela de edição do fluxograma.

No quadro 3 pode ser visto uma parte do algoritmo que desenha o componente correspondente a ação selecionada.

```
.....
cofuncao  : begin
              LocalDesenho.Canvas.Rectangle(Esquerda + 14,Topo + 3,Esquerda + 16,Topo + 5);
              LocalDesenho.Canvas.MoveTo(Esquerda + 15,Topo + 5);
              LocalDesenho.Canvas.LineTo(Esquerda + 15,Topo + 10);
              LocalDesenho.Canvas.MoveTo(Esquerda + 0,Topo + 10);
              LocalDesenho.Canvas.LineTo(Esquerda + 30,Topo + 10);
              LocalDesenho.Canvas.MoveTo(Esquerda + 0,Topo + 20);
              LocalDesenho.Canvas.LineTo(Esquerda + 30,Topo + 20);
              LocalDesenho.Canvas.MoveTo(Esquerda + 15,Topo + 20);
              LocalDesenho.Canvas.LineTo(Esquerda + 15,Topo + 25);
              LocalDesenho.Canvas.Rectangle(Esquerda + 14,Topo + 25,Esquerda + 16,Topo + 27);
              LocalDesenho.Canvas.MoveTo(Esquerda + 0,Topo + 10);
              LocalDesenho.Canvas.LineTo(Esquerda + 0,Topo + 20);
              LocalDesenho.Canvas.MoveTo(Esquerda + 30,Topo + 10);
```

```
            LocalDesenho.Canvas.LineTo(Esquerda + 30,Topo + 20);
            End;
.....
```

Quadro 3 - Algoritmo de desenho do componente função.

6.4 MÓDULO CONEXÕES

Este módulo é responsável pela conexão entre os componentes que estão no fluxograma, ou seja, é responsável pelas ações desenhar e apagar uma conexão entre dois componentes. Esta conexão é realizada através de um ponto inicial e um ponto final. A conexão é tratada como um componente, e este componente como visto anteriormente, também está armazenado na lista de componentes.

No quadro 4 pode ser visto o algoritmo que faz o desenho de uma conexão. Ao desenhar a conexão é verificado se os dois componentes que serão conectados são o mesmo, pois não é permitido conectar um componente com ele próprio. Isto é feito através das variáveis elemento 1 e elemento 2 que terão os nomes dos dois componentes a serem conectados.

```
if      Elemento1 <> Elemento2 then
        begin
        X1 := Conexao.Ponto1.PontoX;
        X2 := Conexao.Ponto2.PontoX;
        Y1 := Conexao.Ponto1.PontoY;
        Y2 := Conexao.Ponto2.PontoY;
        If      Y1 > Y2 then
                Begin
                PontoDivisao := ((Y1 + Y2) div 2);
                LocalDesenho.Canvas.MoveTo(X2,Y2);
                LocalDesenho.Canvas.LineTo(X2,PontoDivisao);
                LocalDesenho.Canvas.LineTo(X1,PontoDivisao);
                LocalDesenho.Canvas.LineTo(X1,Y1);
                End
        Else
                Begin
                PontoDivisao := ((Y1 + Y2) div 2);
                LocalDesenho.Canvas.MoveTo(X2,Y2);
                LocalDesenho.Canvas.LineTo(X2,PontoDivisao);
                LocalDesenho.Canvas.LineTo(X1,PontoDivisao);
                LocalDesenho.Canvas.LineTo(X1,Y1);
                end;
        end
        else
                ShowMessage('Conexão Inválida!');
```

Quadro 4 - Algoritmo de desenho da conexão.

6.5 APRESENTAÇÃO DO PROTÓTIPO

Este protótipo foi desenvolvido em ambiente Delphi 4, onde foram utilizados as ferramentas e os recursos necessários ao desenvolvimento do mesmo. A tela de edição ou construção de fluxogramas do protótipo pode ser vista na figura 28.

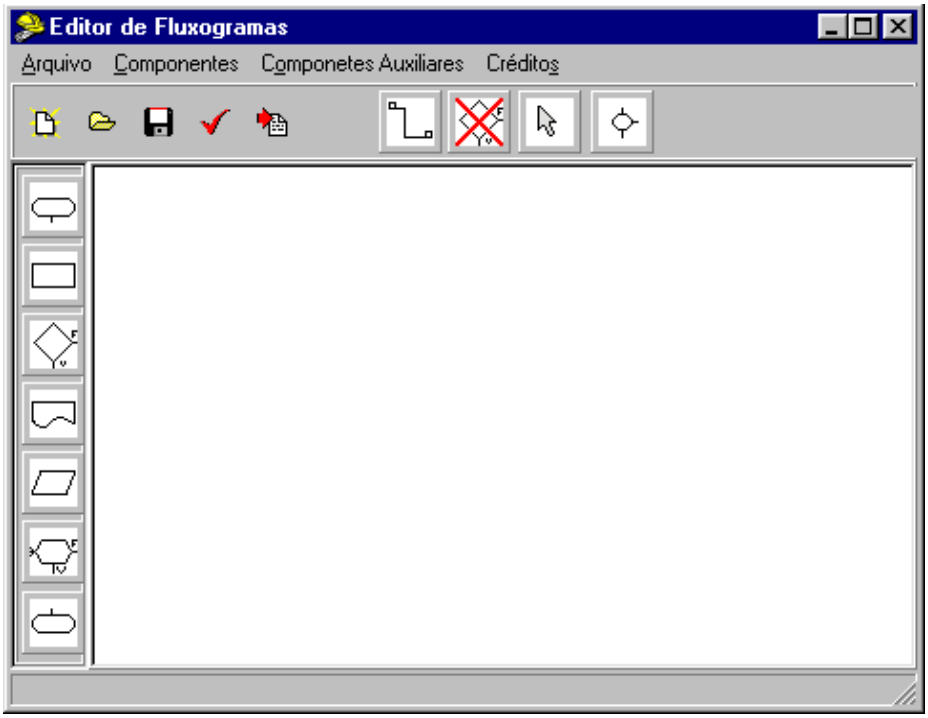


Figura 28 – Editor de Fluxogramas.

A apresentação deste protótipo se dará em seis partes, onde, nas cinco primeiras partes será comentado sobre o ambiente de edição, seguido do item que explica como se deve usar ou operacionalizar o protótipo

6.6 TELA DE EDIÇÃO

Esta tela ocupa a maior parte da área de trabalho do sistema, e é nesta tela que o usuário do sistema vai montar ou construir o fluxograma, desenhando os componentes de acordo com sua necessidade.

6.7 FERRAMENTAS

As ferramentas utilizadas pelo protótipo estão situadas na parte superior esquerda da tela, visualizados em cinco botões, que consistem na execução de procedimentos gerais do editor de fluxogramas, que estão descritos abaixo:

a) Botão Novo: este botão limpa a área de edição do editor de diagrama de fluxo, criando um novo fluxograma. Se tiver um diagrama ativo, será dado a opção para o usuário salvar o diagrama ativo;

b) Botão Abrir: este botão abre um diagrama de fluxo já existente, gravado em um arquivo. Caso tenha um diagrama em edição;

c) Botão Salvar: este botão permite salvar o diagrama de fluxo que está em edição, gravando em um arquivo para uma posterior edição;

d) Botão Verificar Diagrama: este botão varre todo o diagrama em edição, verificando se todos os componentes estão conectados e se o diagrama tem um componente de início e um componente de fim, sendo este o requisito para a geração posterior do código em portugol;

e) Botão Gerar Código Portugol: este botão mostra uma tela (figura 29), que gera o código em portugol, em relação ao diagrama criado na tela de edição do editor de fluxogramas.

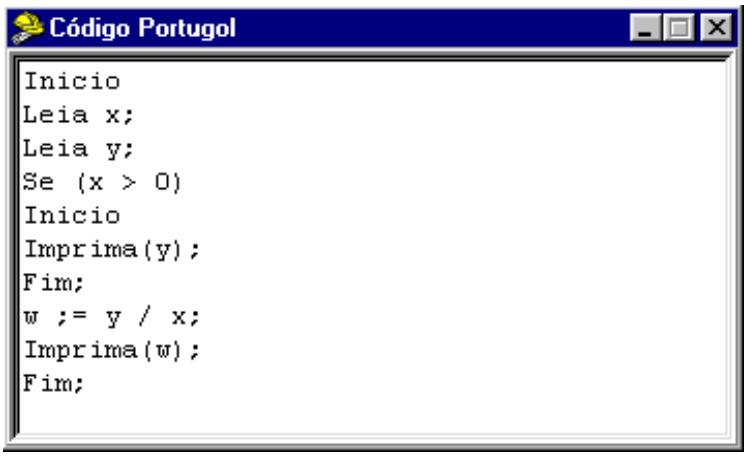


Figura 29 – Código Portugol.

6.8 COMPONENTES DE OPERAÇÃO

Estes botões estão situados ao lado esquerdo da tela de edição do editor de fluxogramas, onde cada botão representa uma ação que o sistema vai executar. Somente um destes botões pode ser acionado por vez, e somente será desativado quando outro botão de operação for ativado ou um dos botões auxiliares for ativado, os quais serão descritos na figura 28.

Os botões de componentes de operação são:

a) Início: este botão indica que a ação desenhada no editor de fluxograma representa o início do fluxograma;

b) Funções: este botão indica que a ação a ser desenhada no editor de fluxogramas representa uma função, por exemplo uma atribuição de valores a certas variáveis;

c) Decisão: este botão indica que a ação a ser desenhada no editor de fluxogramas representa uma operação de decisão, onde pode definir um caminho sempre quando houver mais de um;

d) Impressão: este botão indica que a ação a ser desenhado no editor de fluxogramas representa dados de saída (relatórios), que são impressos pelo sistema;

e) Leitura e Escrita: este botão indica que a ação a ser desenhado no editor de fluxogramas representa a entrada e saída de valores no sistema, por exemplo a leitura do valor de uma variável que é utilizada no fluxograma;

f) Repetição: este botão indica que o desenho a ser desenhado no editor de fluxogramas representa a execução de um bloco de comandos um determinado número de vezes;

g) Fim: este botão indica que a ação desenhada no editor de fluxograma representa o do fluxo.

6.9 COMPONENTES AUXILIARES

Os botões de componentes auxiliares estão situados ao lado direito dos botões de ferramentas, e são utilizados para criar ou mover componentes no fluxograma em edição e

fazer a conexão entre os mesmos. Estes botões funcionam juntamente com os botões de componentes de operação, onde somente um deles pode estar ativo por vez.

a) Conectar Componente: este botão ativa a edição da conexão entre os componentes do diagrama. Para conectar um componente à outro é necessário clicar com o mouse na saída ou na entrada de um componente, e em seguida clicar na entrada ou na saída de outro componente;

b) Excluir Componente: este botão tem por função excluir um componente da tela de edição do diagrama de fluxo. Para excluir um componente é necessário clicar neste botão, e em seguida no componente a ser excluído do diagrama;

c) Mover Componente: este botão indica que nenhum componente será excluído ou criado, e sim, permite mover os componentes já existentes no diagrama de fluxo. Para mover um componente na tela de edição é necessário selecionar este botão e após clicar com o mouse sobre o componente a ser movido, permanecendo com o botão pressionado, e arrastar o componente para o local que se deseja soltando o botão do mouse. Somente poderá mover um componente para uma área limpa, ou seja, mover para uma área onde não tenha nenhum componente desenhado;

d) Conexão Auxiliar: este botão indica uma conexão dupla, utilizado principalmente após ou antes de um comando de seleção.

6.10 MENU

O menu fica na parte superior esquerda da tela, disponibilizando todas as opções que os botões do protótipo executam. Para executar a partir do menu, basta clicar por sobre o item principal do menu, o qual abrirá um submenu com vários itens, onde um segundo clique por sobre um submenu executará a ação correspondente ao item do menu. O menu pode ser visualizado na figura abaixo, e está dividido em quatro partes, as quais serão descritas na figura 30.

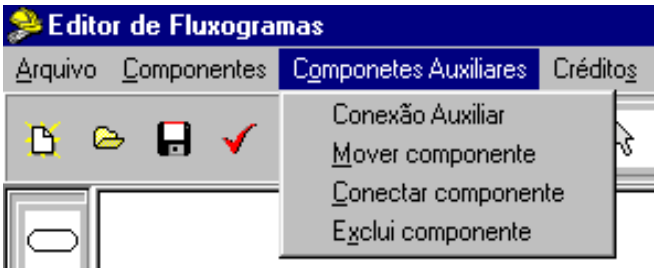


Figura 30 – Menu do Editor de Fluxogramas.

a) Menu Arquivo: neste menu se executa operações referentes aos arquivos do protótipo, opções como novo, salvar e abrir arquivos, opções estas que executam as mesmas operações executadas pelos botões que possuem o mesmo nome. Este menu possui ainda um último submenu que é sair que finaliza a execução do editor de fluxogramas. Caso exista um fluxograma em edição, este item permitirá que o diagrama em execução seja salvo em arquivo;

b) Menu Componentes: este menu, possibilita executar todas as opções dos botões de componentes de operação, ou seja, executa a mesma ação que o acionamento dos componentes de operações que estão situados ao lado esquerdo da tela de edição do protótipo;

c) Menu Componentes Auxiliares: este menu, possibilita executar todas as opções dos botões de componentes auxiliares, ou seja, executa a mesma ação que o acionamento dos componentes auxiliares que estão situados na parte superior direita da tela de edição do protótipo;

d) Menu Créditos: neste menu estão as informações sobre o protótipo, conforme figura 31.

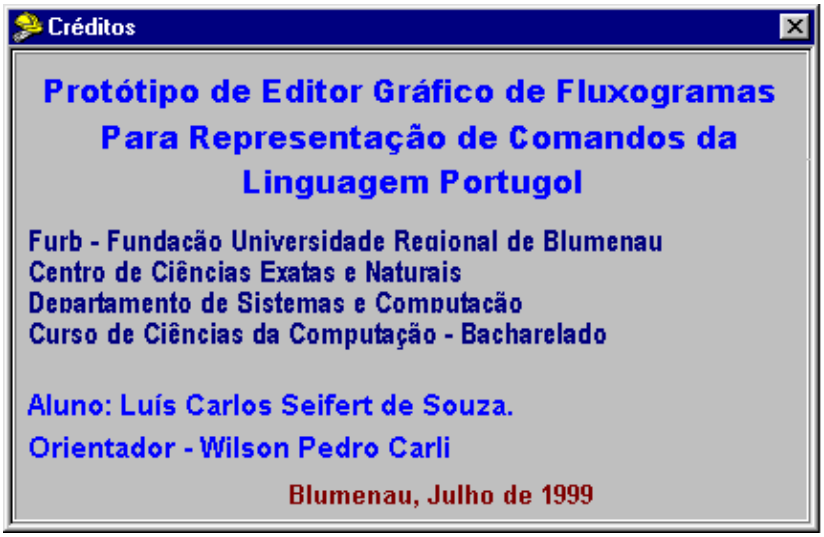


Figura 31 – Créditos do Protótipo.

6.11 OPERACIONALIZAÇÃO DO PROTÓTIPO

Este protótipo tem como principal objetivo a construção de fluxogramas e a geração do código em portugol em relação ao fluxograma ativo. A maioria das funções realizadas pelo protótipo foram vistas anteriormente, quando foi apresentado o protótipo, com as descrições dos componentes que compõem o protótipo.

Para se criar um fluxograma, deve-se ir no menu Arquivo/Novo, ou clicar por sobre o botão novo. Esta opção deve ser utilizada quando já estiver um fluxograma ativo, pois ao inicializar o sistema, um fluxograma novo é criado automaticamente.

Para incluir componentes de edição ao fluxograma, deve-se selecionar um dos botões de componentes desejados, ou pelos itens componentes e componentes auxiliares no menu, e em seguida clicar na tela de edição, onde será desenhado o componente no local indicado pelo cursor do mouse. Após a inclusão dos componentes, tem-se a opção de movimentar os componentes através da seleção do botão mover componente, ou pelo menu Componentes Auxiliares/Mover Componente, clicando após por sobre o desenho desejado para a movimentação, mantendo o botão pressionado, e arrastar o desenho até o local desejado pelo usuário.

A atribuição da expressão aos componentes de atribuição se dá através do clique no botão do componente desejado, onde através deste abrirá uma tela (figura 32), onde permitirá

a edição da expressão que esta ação executará no fluxograma desenvolvido. É através desta expressão que será gerado posteriormente o código portugol deste fluxograma.

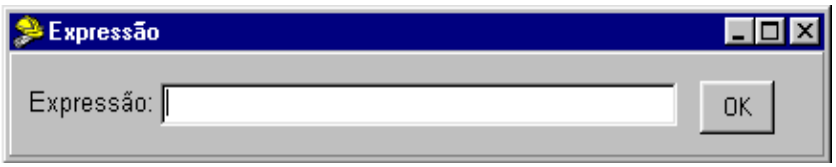


Figura 32 – Expressão.

A conexão ou ligação dos componentes do fluxograma pode ser feita pelo menu Componentes Auxiliares/Conectar Componente, ou pelo botão Conectar Componente, em seguida clicar sobre o primeiro ponto de seleção do primeiro componente, e em seguida por sobre o ponto de seleção do segundo componente. Este protótipo não permite uma conexão com o mesmo componente, ou seja não permite que seja conectado a saída de um componente com a entrada do mesmo componente.

Após o término da edição do diagrama de fluxo, deve haver uma validação do fluxograma editado, através do botão Verificar Diagrama, validação esta necessária para a geração do código portugol.

Para gerar o código portugol do diagrama de fluxo editado, deve-se clicar por sobre o botão Gerar Código Portugol, o qual somente gerará o código portugol, após uma validação do diagrama de fluxo editado.

Um exemplo da geração do código portugol de um fluxo editado pode ser visto na figura 33.

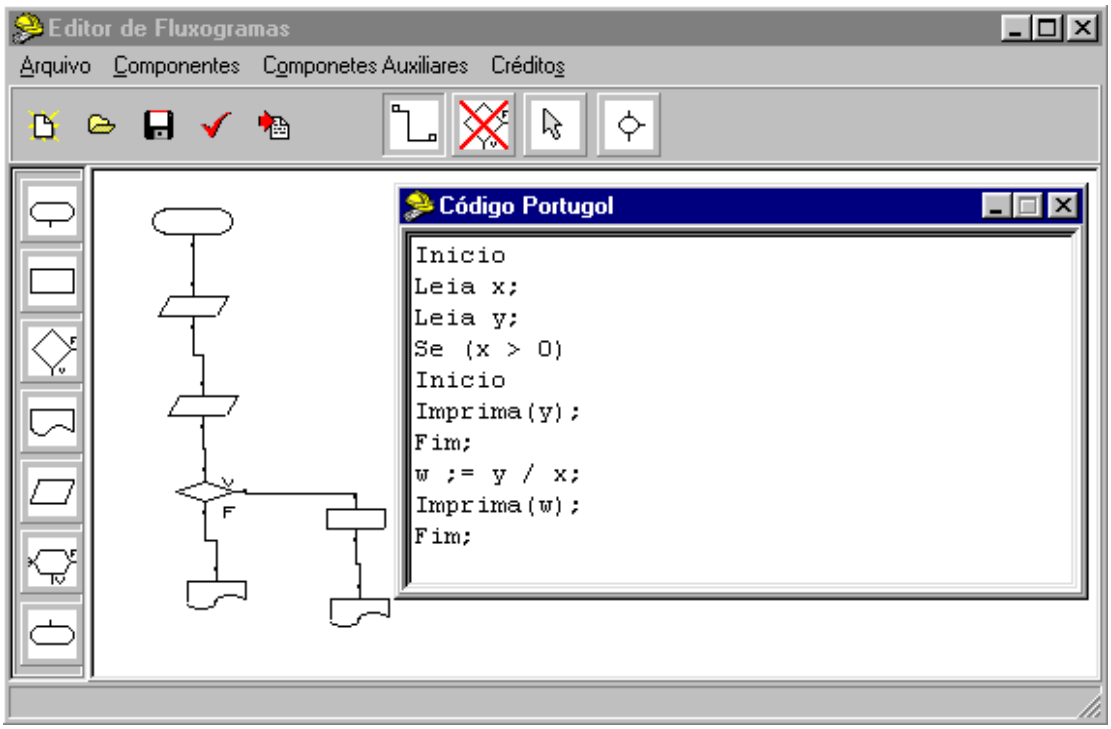


Figura 33 – Tela da execução do protótipo.

7 CONCLUSÃO

Atualmente o ensino da disciplina de Algoritmos na Universidade Regional de Blumenau, ensina a lógica de programação através de exemplos da representação de fluxogramas e de códigos na pseudolinguagem Portugol, geralmente descritos no quadro negro ou no papel.

Através do estudo efetuado da pseudolinguagem Portugol e da linguagem gráfica diagrama de fluxo ou fluxograma, constatou-se que estas duas representações tem como principal função, a visualização do funcionamento e da estruturação do sistema a ser desenvolvido, abstraindo os comandos e estruturas das linguagens de programação utilizadas para o desenvolvimento do sistema. É neste sentido que a linguagem portugol e diagrama de fluxo trabalham; elas abstraem os comandos e as estruturas das linguagens utilizadas para o desenvolvimento, facilitando desta forma a construção e o entendimento do sistema a ser implementado. Estes são os fatores que tornam estas duas representações imprescindíveis ao ensino de lógica de programação.

A maior dificuldade observada neste trabalho está na implementação do protótipo, em virtude de pouca prática ou nenhuma em desenvolvimento utilizando uma metodologia orientada a objetos.

Com relação aos objetivos gerais, estes foram concluídos, tendo como sugestão para a continuação deste trabalho os seguintes itens:

a) criação de uma ferramenta para mostrar o fluxo da execução do fluxograma, através da mudança de cor dos componentes editados, além de um melhoramento na edição do fluxograma;
a) criar um método para a geração de fluxogramas, ou seja, através da edição de um algorítimo (linguagem portugol), gerar um fluxograma, fazendo o inverso do que este protótipo implementa;
b) gerar a impressão do código portugol e do fluxograma.

REFERÊNCIAS BIBLIOGRÁFICAS

[DAO95] DAMASCENO, Américo Jr.. **Aprendendo Delphi Avançado.** São Paulo : Érica, 1995.

[FRA86] FRAGOMENI, Ana Helena. **Dicionário Enciclopédico de Informática**. Rio de Janeiro : Campos, 1986.

[GUI95] GUIMARÃES, Ângelo de Moura; LAGES, Newton Alberto de Castilho. **Algoritmos e estruturas de dados**. Rio de Janeiro : LCT – Livros Técnicos e Científicos, 1995.

[GUS77] GUSMAN, Gilza; VASCONCELLOS, Augusto de. **Fluxogramas e programação COBOL**. Rio de Janeiro : LTC/LTD - Livros Técnicos e Científicos, 1977.

[FER98] FERNANDES, Antonio Luiz B.; BOTINI, Joana. **Construção de Algoritmos**. Rio de Janeiro : Editora Senac Nacional, 1998.

[OLI96] OLIVEIRA, Adelize Generini de. **Análise, projeto e programação orientados a objetos.** Florianópolis : Bookstore Livraria, 1996.

[SOU98] SOUZA, Clovis Fabiano de. **Protótipo de Ambiente Gráfico para a Edição de Circuítos Lógicos**. Blumenau, 1998. Monografia (Bacharelado em Ciências da Computação) Centro de Ciências Exatas e Naturais, FURB.

[SWA96] SWAN, Tom. **Delphi: bíblia do programador**. São Paulo : Berkeley Brasil, 1996.

[WIN93] WINBLAD, Ann L., EDWARDS, Samuel D., KING, David R. **Software orientado ao objeto.** São Paulo : McGraw-Hill, 1993.