TRABALHO DE CONCLUSÃO DE CURSO

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE CIÊNCIAS FÍSICAS E MATEMÁTICAS
DEPARTAMENTO DE MATEMÁTICA

MATRIZES: TEORIA E APLICAÇÃO – UMA ABORDAGEM COM O USO DO AMBIENTE *DELPHI*

ANA LÚCIA FRITZ BUENO

Florianópolis – SC
2004

ANA LÚCIA FRITZ BUENO

MATRIZES: TEORIA E APLICAÇÃO – UMA ABORDAGEM COM O USO DO AMBIENTE *DELPHI*

Monografia apresentada ao curso de Matemática – Habilitação Licenciatura, como requisito para a obtenção do grau de Licenciada em Matemática.

FLORIANÓPOLIS – SC

2004

Esta monografia foi julgada adequada como **TRABALHO DE CONCLUSÃO DE CURSO** no Curso de Matemática - Habilitação Licenciatura, e aprovada em sua forma final pela Banca Examinadora designada pela Portaria nº 60/SCG/04

_____

Prof.ª Carmem Suzane Comitre Gimenez
Professora da disciplina

Banca Examinadora:

_____

Sonia Palomino Bean
Orientadora

_____

Dale William Bean - Membro

_____

Rosimary Pereira - Membro

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus por ter me dado força para aguentar até o final, apesar de todos os contratempos que surgiram ao longo do percursso. Agradeço, em especial, a minha família por todo apoio. Agradeço também, a minha orientadora, por oferecer a oportunidade de trabalhar no projeto e adquirir mais conhecimento e prática; aos membros da banca examinadora, por terem aceitado avaliar o meu trabalho; ao GEIAAM, por ceder espaço para desenvolver meu trabalho; aos alunos e professores dos colégios participantes das apresentações do projeto. Finalmente, agradeço aos meus amigos, professores e secretárias do curso.

SUMÁRIO

LISTA DE FIGURAS

INTRODUÇÃO

No ano 2003 iniciou o projeto de extensão "*Explorando a Interdisciplinaridade dos Conteúdos de Álgebra Linear e Geometria Analítica*" (E.I.C.A.L.G.A.)[1], que teve como principal objetivo promover a interdisciplinaridade de conteúdos abordados no Ensino Médio e no curso de Graduação de Matemática.

A proposta original desse projeto foi a de desenvolver situações didáticas relativas a utilização de diferentes materiais didáticos e computacionais com demonstrações aos alunos de colégios públicos e/ou particulares e alunos do curso de Licenciatura em Matemática. Realizar seções organizadas para demonstrar a interdisciplinaridade existente nos conteúdos da *Geometria Analítica* e *Álgebra Linear*, assim como, apoiar o ensino e o aprendizado de conteúdos tais como *retas*, *planos*, *cônicas*, *determinantes* e *sistemas de equações*, estabelecendo conexão entre esses conteúdos. Mostrar aplicações práticas do uso de *matrizes* em sistemas de segurança (*Criptografia*), introduzir os conceitos de *Geometria Fractal* e mostrar exemplos dessas interessantes figuras que podem ser vistos na natureza (PALOMINO, 2004).

Para a realização desse projeto foram necessárias várias pesquisas tanto no campo tecnológico – através da verificação dos tipos de *softwares* existentes, as linguagens de programação que poderiam fornecer um ambiente dinâmico para a aprendizagem - quanto na área da Educação, onde buscou-se conhecer a importância e a validade de se utilizar a tecnologia no processo ensino/aprendizagem.

A partir das pesquisa feitas no campo tecnológico, foram escolhidos os *softwares Matlab, Derive, Oficina de Funções e Cabri Géomètre II* para o desenvolvimento das atividades que envolviam os conteúdos *retas*, *planos*, *matrizes, cônicas*, *determinantes* e *sistemas de equações*.

Para o desenvolvimento da parte de aplicação de matrizes em sistemas de segurança, foram utilizados o *software Matlab*, o qual foi desconsiderado por não estar de acordo com o que foi idealizado para essa atividade, e o

[1] Assim designado pois será mencionado várias vezes nesse trabalho.

ambiente de programação *Delphi*, que possibilitou a criação de programas interativos e amigáveis, adequados aos propósitos do projeto.

O envolvimento com a criação desses programas e com as atividades realizadas com o Projeto E.I.C.A.L.G.A. fez surgir a idéia de criar um outro aplicativo que pudesse fornecer ao usuário, além da parte de aplicação, a parte teórica, incluindo exemplos e exercícios, com uma abordagem diferente e agradável, que contribuisse no processo de ensino/aprendizagem. Para por em prática essa idéia, desenvolveu-se, também em ambiente *Delphi*, o programa ***Matrizes: teoria e aplicação***, o qual foi escolhido como tema central desse trabalho.

Baseado, então, no Projeto E.I.C.A.L.G.A., nas pesquisas e atividades realizadas originou-se este Trabalho de Conclusão de Curso, no qual serão abordados os seguintes conteúdos:

- *O uso das tecnologias no processo de ensino/aprendizagem*: onde é mostrado de maneira geral o que é tecnologia, tipos de recursos computacionais, o papel do professor com a introdução das tecnologias nesse processo e como fica o aprendizado dos alunos.
- *Linguagens de programação*: onde são apresentados os tipos de linguagem, como são gerados os programas, tipos de programação, etc.
- *Conhecendo o Delphi*: onde é apresentado o ambiente em que foram gerados os programas para o Projeto E.I.C.A.L.G.A.
- *Matrizes*: referencial teórico do conteúdo escolhido para a desenvolvimento do programa ***Matrizes: teoria e aplicação***.
- *Criptografia*: onde são apresentados alguns conceitos básicos da teoria, uma descrição de um método simples para codificar e decodificar mensagens. Apresenta também os programas criados com o *software Matlab* e no ambiente *Delphi*.
- *O programa* ***Matrizes: teoria e aplicação***: onde são apresentadas as partes essenciais do programa, dicas, exemplos, mensagens de erro, entre outras. Apresenta também o resultado obtido na avaliação feita por alunos e usuários desse programa.

CAPÍTULO 1 - O USO DAS TECNOLOGIAS NO PROCESSO DE ENSINO/APRENDIZAGEM

Em primeiro lugar, é importante ressaltar que a utilização de tecnologias na área de educação não é mais um assunto desconhecido. Vários trabalhos foram e estão sendo realizados no sentido de divulgar esse meio importante para se ensinar e aprender. Tão importante que conseguiu espaço nos Parâmetros Curriculares Nacionais (P.C.N.). Segundo eles, "*a formação do aluno deve ter como alvo principal a aquisição de conhecimentos básicos, a preparação científica e a capacidade de utilizar diferentes tecnologias* (...)" (P.C.N., 1999).

Entende-se por tecnologia, todos os recursos que auxiliam nos meios de comunicação. Entre eles está o computador, que se diferencia dos demais tipos de tecnologia por sua capacidade de programação, isto é, pode-se criar programas, jogos, entre outros, através de códigos específicos que podem ser lidos e interpretados pelo computador. Esses códigos são as chamadas Linguagens de Programação. A programação tornou possível a realização de muitas atividades ou tarefas essenciais como o controle de contas bancárias, arquivamento de dados, realização de cálculos matemáticos complexos, entre outras. Também através da programação, as atividades de entretenimento, em especial, os jogos, tiveram um grande desenvolvimento. Para programar é necessário conhecer e entender pelo menos uma linguagem de programação. No capítulo 3 será feita uma abordagem mais detalhada sobre essas linguagens.

Existem muitos outros recursos computacionais como os *Softwares*, a *Multimídia* e a *Internet*, que estão cada vez mais presentes no cotidiano. Esses recursos podem ser vistos como *mediadores culturais,* instrumentos criados pela espécie humana e que permeiam significantemente nossa relação com as coisas (VYGOTSKY *apud* VIERA, 2004). Todos esses recursos podem ser, e já estão sendo, utilizados na educação.

A utilização da tecnologia na educação vai de encontro aos objetivos propostos para a formação do aluno que são: "*o desenvolvimento de capacidades de pesquisar, buscar informações, analisá-las e selecioná-las; a capacidade de aprender, criar, formular, ao invés do simples exercício de*

memorização." (P.C.N, 1999). E ainda : "*Privilegiar a aplicação da teoria na prática e enriquecer a vivência da ciência na tecnologia e destas no social* (...)." (P.C.N., 1999). Além disso, a utilização de recursos computacionais em sala de aula, serve como um grande apoio pedagógico, tanto para o professor como para o aluno. O papel do professor passa a ser o de "*coordenador das informações, relacionando e contextualizando para auxiliar o aluno no processo de aprendizagem.*" (LACERDA, 2001). O aluno tem a oportunidade de aprender de uma forma diferente e dinâmica, passa a ter acesso a várias informações, imagens, que a forma tradicional de ensinar não conseguiria fornecer. Mas deve ficar claro que o uso desses recursos não deve substituir o professor. Como já foi mencionado, deve servir apenas como um apoio no processo de ensino/aprendizagem. Esse apoio pode vir com utilização da Internet, onde o professor e o aluno podem pesquisar mais sobre os conteúdos que estão sendo vistos em sala de aula, informações que estejam relacionadas com esses conteúdos como, por exemplo, onde são aplicados, etc. O professor pode buscar esse apoio também na utilização de *softwares* educacionais. O contato do professor com esse material pode ser feito através da própria Internet, onde já se tornou fácil encontrar materiais diversificados e interessantes sobre isso.

1.1. *Softwares* no ensino de Matemática

"*A aprendizagem matemática através de softwares deve ser baseada em situações-problema que considerem: os processos cognitivos, o raciocínio, as estratégias adotadas durante o processo de resolução, os estágios de desenvolvimento relativos às habilidades envolvidas e caracterização dos diversos problemas e seu nível de complexidade*. " (GOMES *et al*., 2002)

Ao utilizar *softwares* no ensino de Matemática deve-se ter em mente:

- qual é o público alvo que se deseja atingir, ou seja, com que séries podem ser trabalhados;
- qual o nível de conhecimento necessário para que possa haver uma interação maior com o *software* escolhido;
- quanto o professor deve saber para que possa auxiliar os alunos;

- quais os objetivos que se espera alcançar, por exemplo, ensinar aos alunos a arte de programar, ou simplesmente como utilizar o ambiente para resolver exercícios;
- como esse recurso pode auxiliar no processo de ensino/aprendizagem de forma positiva e produtiva.

Outro fator muito importante que se deve levar em conta é a escolha do *software*. Existem alguns critérios de avaliação (GOMES *et al.*, 2002) que podem auxiliar no processo de escolha dos *softwares* como, por exemplo:

- grau de compreensão sem a presença de um instrutor;
- clareza das alternativas possíveis de comando;
- coesão de linguagem e gramática;
- clareza na exposição das informações;
- clareza da transição entre partes dos programas e/ou lições;
- quanto à indicação de pré-requisitos, tais como: faixa etária ou nível de instrução, exercícios que devem anteceder ao programa.

De acordo com esses critérios, o professor tem garantia de que está utilizando com seus alunos um *software* adequado para auxiliar no processo de ensino/aprendizagem dos conteúdos matemáticos.

Atualmente é possível encontrar vários tipos de *softwares* voltados ao ensino de Matemática. Muitos deles abordando conteúdos como Geometria (*Cabri-Géomètre II*), Álgebra Linear e Geometria Analítica (*Matlab*, *Derive*, *Maple*), entre outros.

Para verificar a validade do uso de *softwares* como auxiliadores no processo de ensino/aprendizagem o Projeto E.I.C.A.L.G.A. selecionou alguns desses *softwares*, o *Cabri-Géomètre* II, o *Matlab* e o *Derive,* para serem utilizados nas atividades que envolviam os conteúdos *cônicas*, *matrizes* e *sistemas lineares*. Foram obtidos resultados interessantes que forneceram conteúdo suficiente para ser desenvolvido em um trabalho de conclusão de curso (ALVES, 2004). Mas o projeto não se ateve apenas à esses *softwares*.

Como um dos objetivos do projeto era mostrar aplicações de alguns conteúdos no cotidiano, como por exemplo, onde a teoria de matrizes pode ser aplicada e, com isso também introduzir conceitos novos como a criptografia, foram desenvolvidos programas utilizando-se o ambiente de programação *Delphi*. Para utilizar e entender um pouco sobre esse ambiente é necessário

conhecer alguns conceitos de programação, entre eles, as linguagens, como são gerados os programas e os tipos de programação.

CAPÍTULO 2 - LINGUAGENS DE PROGRAMAÇÃO*

Inicialmente, é necessário entender o que é um programa de computador. Um programa é um conjunto de instruções, ou códigos, que informa ao computador quais tarefas deverão ser executadas e de que forma serão realizadas. Ele pode ser escrito utilizando-se vários tipos de linguagens.

Os comandos e instruções são traduzidos para uma linguagem própria (linguagem de máquina), ou seja uma sequência de códigos que pode ser executada pelo microprocessador. Esses códigos são números em formato binário (dígitos 0 e 1), cuja combinação determina uma ação ou conjunto de ações a serem executadas pelo processador.

Os primeiros computadores exigiam a programação diretamente em código de máquina, o que tornava difícil o trabalho dos programadores. Com o objetivo de facilitar, de certa forma, o trabalho deles, foram criadas, então, outros tipos de linguagens de programação. Elas podem ser classificadas como linguagens de baixo nível e linguagens de alto nível.

As linguagens de baixo nível aproximam–se mais do código de máquina, como, por exemplo, a linguagem Assembly, que permite o controle total da máquina. Uma das dificuldades em se trabalhar com esse tipo de linguagem é que ela não é estruturada e modular. Os programas que são escrito com esse tipo de linguagem não são portáveis, ou seja, podem não funcionar em outro computador.

As linguagens de alto nível são mais produtivas, pois possuem comandos prontos e de fácil assimilação. A vantagem em se utilizar esse tipo de linguagem é que elas não dependem da máquina, ou seja, os comandos e instruções são os mesmos para diversos tipos de computadores.

Alguns exemplos de linguagem de alto nível são:

- *COBOL*, voltada para sistemas comerciais de grande porte;
- *LOGO*, dirigida a microcomputadores com apelo educacional;
- *BASIC*, *PASCAL*, *C/C++*, *FORTRAN*, utilizadas em programação de sistemas operacionais, compiladores, editores de texto, planilhas eletrônicas, bancos de dados e jogos.

---

* Grande parte dessas informações foram retiradas do livro de William Pereira Alves, 1997, com sua autorização.

Existem ainda as linguagens direcionadas a um uso específico, que não permitem acesso a características internas da máquina, mas fornecem os recursos necessários à sua utilização. Entre elas estão o *dBASE*, o *CLIPPER* e o *ACCESS*, dirigidos à manipulação de banco de dados em microcomputadores.

2.1. Gerando programas*

Existem duas maneiras para se gerar programas em computadores: a interpretação e a compilação. Na interpretação, os programas são traduzidos à medida que vão sendo executados na memória, através de um programa chamado interpretador.

O interpretador trabalha diretamente com o código-fonte do programa, convertendo cada instrução, feita em liguagem de alto nível, para a linguagem de máquina, antes da execução. A vantagem de se utilizar esse tipo de programa é a facilidade de uso, pois pode-se testar o programa durante a fase de desenvolvimento.

As desvantagens são:

- a necessidade de se ter o interpretador na memória, o que reduz a quantidade de espaço livre;
- o código-fonte deve ser fornecido para execução, o que implica que o programa pode ser alterado, pois não está protegido;
- a execução pode ser muito lenta;
- pode gerar vícios de programação, como desenvolver programa sem planejamento prévio (tentativa e erro).

Um exemplo de linguagem que utiliza o interpretador é o BASIC.

Na compilação, utiliza-se um programa chamado compilador, que converte a linguagem de alto nível em código de máquina. Ele se diferencia do interpretador por traduzir o programa inteiro de uma única vez.

As vantagens em utilizar linguagens compiladas são:

- execução mais rápida;

* Grande parte dessas informações foram retiradas do livro de William Pereira Alves, 1997, com sua autorização.

- código-fonte protegido, o arquivo executável não pode ser lido facilmente;
- programa executável de menor tamanho;
- exige menos memória.

Os compiladores podem ser separados em duas categorias:

- compiladores de pseudocódigo, que traduzem todo código-fonte e geram um arquivo intermediário, executado por um interpretador;
- compiladores de código nativo, que geram um arquivo-objeto em linguagem de máquina. Após serem ligados às bibliotecas de funções, não necessitam de um interpretador para serem executados.

2.2. Tipos de programação*

Existem três categorias de programação: programação procedimental, programação orientada a eventos e programação orientada a objetos.

Na programação procedimental, baseada em chamadas de procedimentos, a execução do aplicativo[2] inicia na primeira linha do código e segue o fluxo determinado pelo programa. Ou seja, o usuário fica preso à aplicação.

Na programação orientada a eventos, as ações do usuário, denominadas eventos[3], determinam o fluxo de execução do programa. Nesse caso, a aplicação depende do usuário. O *Windows* é um exemplo de sistema operacional orientado a eventos. Cada ação executada gera um evento diferente, como por exemplo, pressionar um botão com o *mouse*.

Na programação orientada a objetos[4], os aplicativos criados utilizam os objetos[5] como unidade básica de dados e procedimentos. Existem três elementos-chave na programação orientada a objetos, são eles:

- *classe*: tipo de dado definido pelo usuário;

---

* Grande parte dessas informações foram retiradas do livro de William Pereira Alves, 1997, com sua autorização.

2 "*Qualquer software que use os recursos básicos da máquina para realizar uma tarefa específica como por exemplo editar textos montar gráficos*". (dicionário Michaelis)

3 Ações reconhecidas por um objeto presente em um formulário ou em um relatório.

4 Surgiu no final dos anos 70 e início dos anos 80, quando pesquisadores do Centro de Pesquisas da Xerox, em Palo Alto, Califórnia, desenvolveram a linguagem de programação *SmalTalk*, que utilizava pela primeira vez o conceito de classes e objetos e possuia uma interface gráfica baseada em janelas e ícones.

- *herança*: os métodos empregados em um classe podem ser herdados em outra classe, evitando a repetição de código de programa;
- *polimorfismo*: uma mesma variável pode fazer referência a vários objetos de classes diferentes.

Dentre os ambientes, ou linguagens de programação, baseados em programação orientada a objetos encontra-se o *Delphi*, ambiente escolhido para criação dos programas apresentados no Projeto E.I.C.A.L.G.A. , os quais serão abordados mais adiante.

2.3. Conhecendo o *Delphi*[*]

O *Delphi* foi lançado em 1995 pela *Borland* (empresa de ferramentas de desenvolvimento). É uma ferramenta de desenvolvimento visual que utiliza uma linguagem de programação totalmente orientada a objeto, o *Object Pascal*, que possibilita a criação de novos componentes a partir dos existentes na biblioteca do *Delphi*, criação de novas classes e objetos, a transmissão das propriedades e métodos de uma classe existente, etc.

Por ser uma ferramenta de desenvolvimento visual, o *Delphi* automatiza o trabalho de criação de aplicativos, pois auxilia a construção manual de todas as partes do aplicativo, ou seja, através de um clique do *mouse* sobre os componentes que serão adicionados aos formulários. Isso gera uma redução do tempo de desenvolvimento e manutenção dos programas e, além disso, um aumento na sua eficiência.

O *Delphi* compila os programas em código nativo, gerando um arquivo executável (*.EXE*) independente de outros arquivos e de bibliotecas de vínculo dinâmico (arquivos *.DLL*). Em outras palavras, o programador, ou projetista, pode fornecer aos usuários um único arquivo *.EXE*, e escrever programas de instalação simples. (MANZANO *et al*., 1998).

O *Delphi* permite, ainda, a criação de arquivos *.DLL*, possibilitando, assim, a criação de rotinas que podem ser utilizadas em projetos de aplicações desenvolvidas em *C++, Visual Basic*, etc.

---

[5] Tudo que pode ser manipulado ou visualizado de alguma forma.

[*] Grande parte dessas informações foram retiradas do livro de William Pereira Alves, 1997, com sua autorização.

Recursos como os chamados *Templates* e *Experts*[6], que auxiliam na criação de formulários ou aplicações inteiras, incluindo aplicações de banco de dados, são oferecidos pelo *Delphi*. Ele oferece também um conjunto completo de componentes direcionados para criação de aplicações que acessam servidores *Web* (*Internet* ou *Intranet*) nos padrões *Microsoft* e *Netscape*.

O *Delphi* possui várias versões[7], mas mantém uma característica importante que é a total compatibilidade em nível de código-fonte da versão mais atual com as versões anteriores.

2.3.1. O ambiente de programação *Delphi*

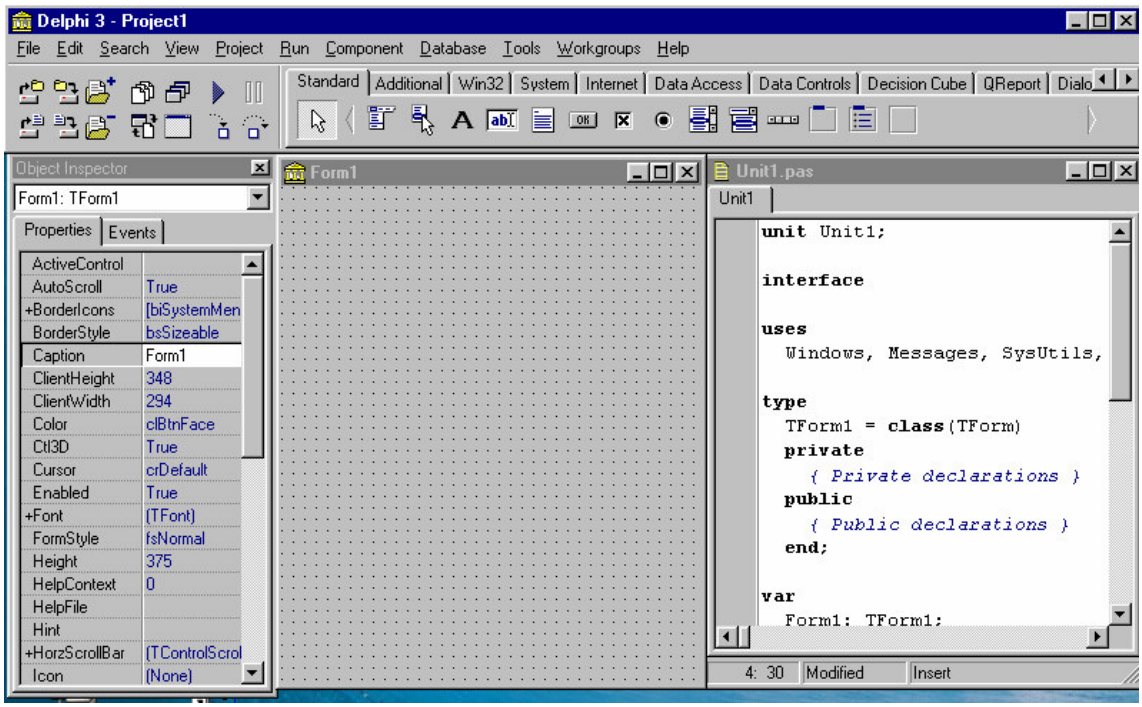


fig. 1 – o ambiente *Delphi*

O ambiente de programação *Delphi* (fig.1) integra todos os recursos de um compilador, um editor de códigos e um editor de componentes (MANZANO *et al*, 1998). Nesse ambiente são apresentadas quatro janelas: *Project1*, *Object Inspector*, *Form1* e *Unit1.pas*.

Form1 (fig.2) é um formulário onde adicionamos os componentes que farão parte da interface da aplicação como, por exemplo, os botões de

[6] Esses recursos diferenciam o Delphi de outras ferramentas de programação
[7] A versão atual é 7.0. Mas a versão utilizada para esse trabalho é 3.0.

comando (*Button*, *Bitmapbutton*), as imagens (*Image*), caixas de entrada de texto (*Edit*, *Label*), entre outras.

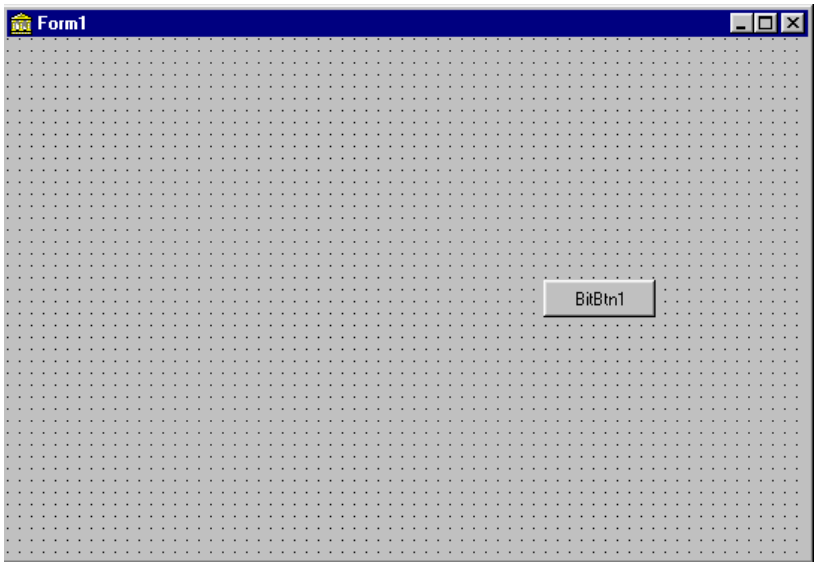


fig. 2 – o formulário com um componente adicionado

Para adicionar um componente no formulário (*form*) basta clicar em uma das opções da paleta de componentes (fig. 3) e em seguida clicar no formulário. Uma outra maneira de se adicionar componentes é dar um "clique duplo" no componente escolhido, ele será adicionado diretamente ao formulário.

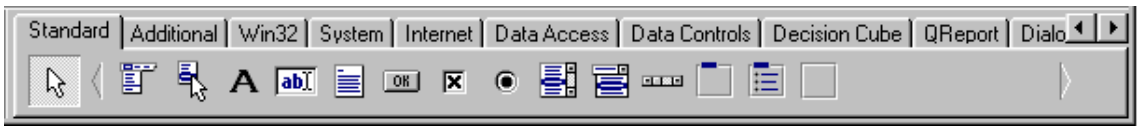


fig. 3 – a paleta de componentes

Unit1.pas é a janela de edição onde se faz a programação em geral. É interessante mencionar que o *Delphi* gera automaticamente parte dos códigos que são utilizados durante a programação (fig. 4), como por exemplo, a variável *Form1: Tform1*, que será utilizada em toda a aplicação, a *procedure*[8] do componente que foi adicionado no formulário, onde se estabele os comandos e parâmetros para esse componente, etc.

[8] A *procedure* é um "*pequeno trecho de programa de computador que é usado frequentemente e pode ser chamado por um programa principal*" (dicionário Michaelis).

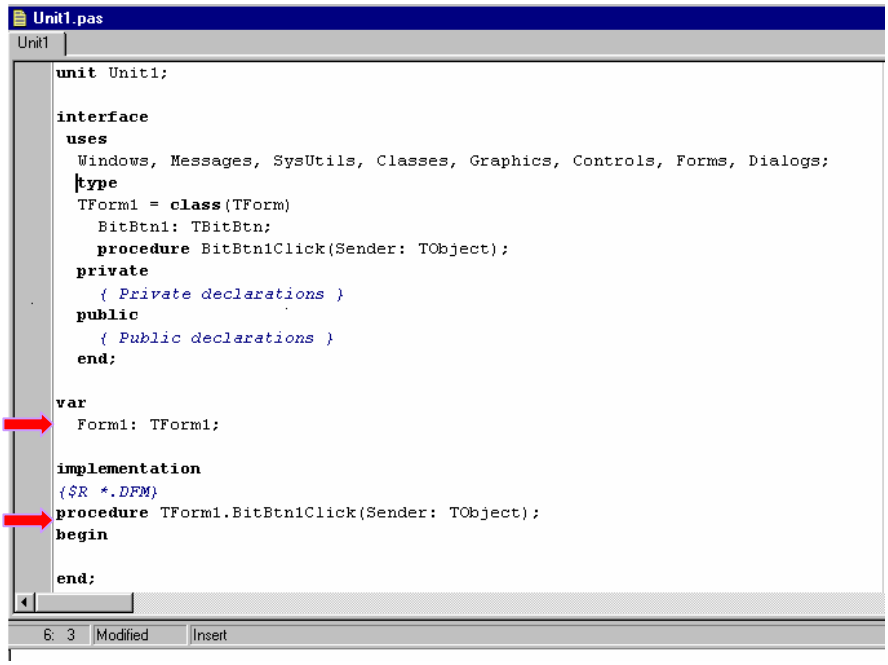


fig. 4 – a janela de edição (*unit1.pas*) – códigos gerados pelo *Delphi*

Object Inspector (fig. 5) é a janela na qual pode-se fazer os ajustes nas propriedades de um objeto (*Properties*) como, por exemplo, alterar o nome (*Name*) e a fonte (+*Font*) e nos eventos relacionados a esse objeto (*Events*) como, acionar através de um "clique" (*OnClick*).

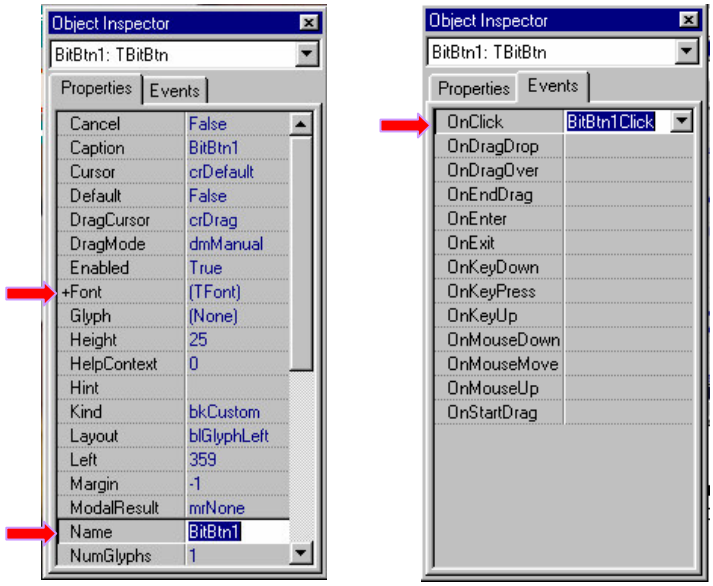


fig. 5 – a janela das propriedades e eventos

A figura 6 mostra o formulário da figura 2 modificado através do *Object Inspector*:

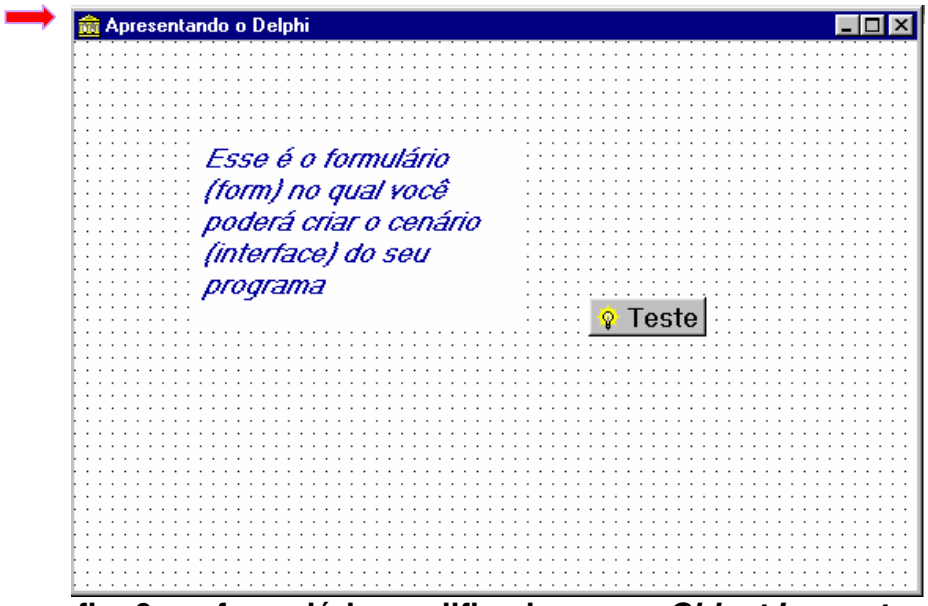


fig. 6 – o formulário modificado com o *Object Inspector*

Foram feitas as seguintes modificações no formulário:

- o componente *Memo* foi adicionado;
- utilizou-se a propriedade *Lines* para criar o texto "*Esse é ....*", e logo em seguida a propriedade +*Font* para mudar o tipo, o estilo e a cor da fonte;
- o componente BitmapButton também foi adicionado;
- utilizou-se a propriedade *Glyph* para inserir a imagem *bitmap* (lâmpada) e a propriedade +*Font* para aumentar o tamanho a fonte;
- utilizou-se a propriedade *Color* para mudar a cor do formulário;
- utilizou-se a propriedade *Caption* para mudar o nome usual do formulário de *Form1* para *Apresentando o Delphi*.

Várias outras modificações poderiam ser feitas no formulário sem a necessidade de se estar programando. Isso porque a parte de programação inicia no momento em que o programador começa a estabeler comandos e parâmetros para os componentes que foram adicionados ao formulário. Para isso, é necessário que o programador conheça alguns conceitos básicos desse ambiente.

2.3.2. Trabalhando com o *Delphi*

Como qualquer outra linguagem de programação, o *Delphi* também utiliza conceitos básicos como constantes e variáveis. As constantes, como o próprio nome sugere, não podem ser alteradas. São utilizadas para especificar um valor que poderá ser usado várias vezes durante a execução de um programa.

Por exemplo, se em um programa fosse necessário utilizar o valor do número irracional *Pi* (π) com apenas duas casas decimais, bastaria fazer a seguinte especificação:

```
Const
    Pi = 3.14;
```

Sempre que o valor de *Pi* for solicitado, o programa irá utilizar **3.14**.

As variáveis diferem das constantes, pois são utilizadas para especificar valores que poderão ser alterados durante a execução de um programa. Elas são declaradas da seguinte forma:

```
var
    x: string;
    y: integer;
    z: real;
```

onde *string* (sequência de caracteres alfanuméricos ou palavras consecutivas que são manipuladas e tratadas como uma unidade), *integer* (inteiro) e *real* são alguns dos tipos de váriaveis que se pode utilizar em uma programação.

"*O Delphi, por ser um ambiente de programação orientado a objeto, exige que os programas nele escritos sigam o conceito de programação estruturada. Assim sendo, o ambiente é baseado em sub-rotinas do tipo procedures e functions."* (MANZANO *et al.*, 1998)

Procedure é um "*pequeno trecho de programa de computador que é usado frequentemente e pode ser chamado por um programa principal*" (dicionário Michaelis). Em outras palavras, é um agrupamento lógico de instruções de programa em um único bloco. Esse bloco pode ser ativado pela execução de uma chamada de *procedure*. Por exemplo, a figura 7 mostra um programa simples de adição de dois números inteiros:

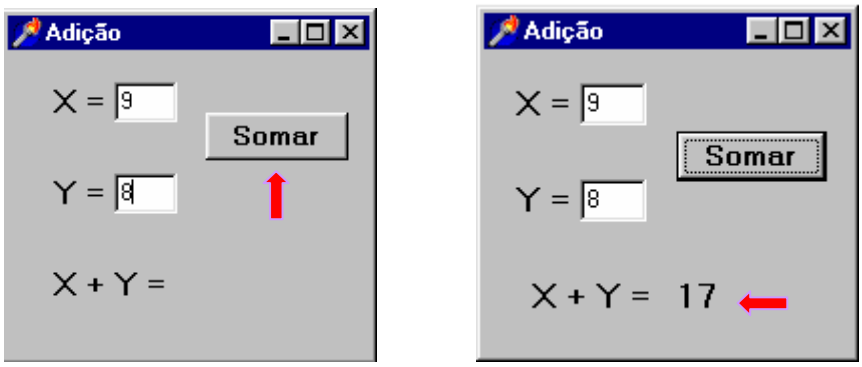


fig. 7 – programa *Adição*

Após escolher dois valores (x e y – inteiros) o usuário aciona o botão *Somar*, o programa chama a *procedure Tform1.Button1Click* (fig. 8), e faz o cálculo solicitado, ou seja, calcula o valor de x + y.

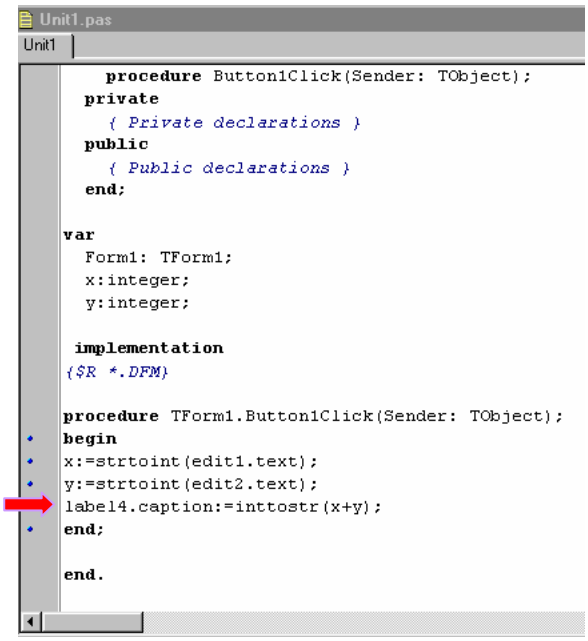


fig. 8 – a janela de edição do programa *Adição*

As funções (*functions*) são *procedures* que retornam um valor único. Seu código deve ser inserido antes da *procedure* usual. As funções, são usadas geralmente, quando se trabalha com expressões.

Considere, como exemplo, o programa para calcular fatoriais (fig. 9):

fig. 9 – programa *Fatorial*

Nesse exemplo, a função é chamada na linha (fig. 10):

Label1.caption:= inttostr(fatorial (N));

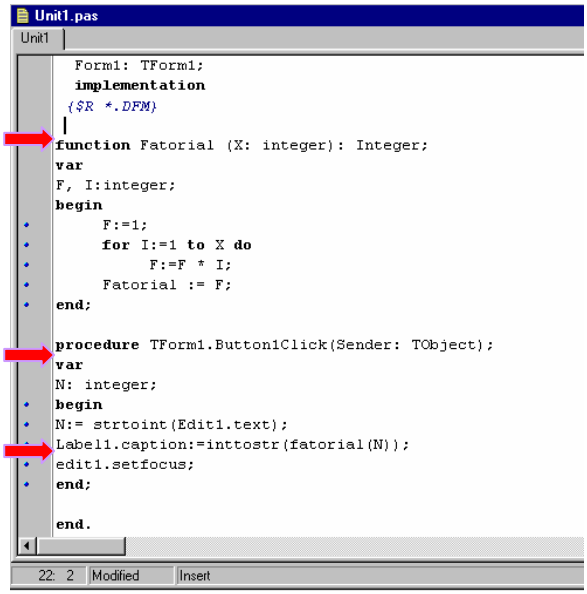


fig. 10 – janela de edição do programa *Fatorial*, utilizando função (*function*)

As unidades (*units*) são os módulos onde são agrupadas as funções e *procedures* relacionadas (fig. 10). Sempre que é criado um novo projeto, o *Delphi* gera um módulo de programa e uma unidade que define o formulário principal. Sempre que um novo formulário é adicionado a um programa, o *Delphi* adiciona uma nova unidade. As unidades são compiladas separadamente e ligadas ao programa principal. São criados então os arquivos de unidade com extensão *.PAS*, e os arquivos de programas com extensão *.DPR* (CANTU, 1997).

É necessário mencionar que o objetivo desse trabalho não é ensinar como programar no ambiente *Delphi*. Foi feito, até o momento, apenas uma

abordagem geral sobre o ambiente, para que se possa fazer uma apresentação dos programas que foram desenvolvidos através dele.

Mas, antes de fazer essa apresentação é necessário falar um pouco sobre os conteúdos que impulsionaram o desenvolvimento dos mesmos: *Matrizes* e a *Criptografia*, os quais serão apresentados no próximos capítulos.

CAPÍTULO 3 - MATRIZES

Na tabela abaixo estão relacionadas as médias dos alunos nas disciplinas Química, Inglês e Matemática:

	Quimica	Inglês	Matemática
Ana	7	10	9
Alex	6	8	10
Sonia	8	9	9

Os valores dessa tabela podem ser dispostos da seguinte forma:

$$\begin{bmatrix} 7 & 10 & 9 \\ 6 & 8 & 10 \\ 8 & 9 & 9 \end{bmatrix}$$

Essa maneira de dispor os elementos é conhecida como **matriz**.

Os primeiros registros que se tem sobre matrizes datam de 250 A.C. na antiga China, exatamente sob a forma de tabelas. Elas apareceram na obra Chui-Chang Suan-Shu (*Nove capítulos sobre a arte Matemática*)[9], onde são apresentados problemas sobre a mensuração de terras, impostos, equações, entre outros. O *Capítulo VIII - Tabelas rectangulares* (*Fang cheng*) contém 18 problemas que conduzem a sistemas de equações lineares. O processo de resolução destes sistemas é semelhante à resolução de sistemas por matrizes.

O termo matriz foi criado por Joseph Sylvester[10] (1814-1897), em 1850. O significado coloquial desse termo é: local onde algo é gerado ou criado. Sylvester as considerava como "...*um bloco retangular de termos... o que não representa um determinante*[11]*, mas é como se fosse uma MATRIZ a partir da*

---

[9] "*Os Nove Capítulos influenciaram toda a matemática chinesa, tendo sido utilizado como manual de ensino, não apenas na China, mas também nos países e regiões circundantes, até a ciência Ocidental ter se introduzido no Oriente, por volta de 1600. O livro é de autor desconhecido, como era comum na antiga China. Contém 246 problemas e está dividido em 9 capítulos. Para cada problema é dada uma resposta sem que seja fornecido o método utilizado para o resolver. A maior parte deles são problemas práticos do dia-a-dia. Alguns destes problemas datam da Dinastia Qin (221 - 201 a.C.), mas foram, ao que tudo indica, recompilados por Zhang Cang (256? - 152 a.C)*". (LAGARTO, 2004)

[10] James Joseph Sylvester (1814 - 1897): matemático inglês, um dos descobridores da teoria dos invariantes algébricos.

[11] Surgiu pela primeira vez em 1683, num trabalho do japonês Seki Kowa. O uso de determinantes no Ocidente começou dez anos depois num trabalho de Leibniz. O termo

qual podemos formar varios sistemas de determinantes, ao fixar um número p e escolher à vontade p linhas e p colunas..." (SYLVESTER, 1850 *apud* WEISSTEIN, 2003). Até então, as matrizes estavam sempre relacionadas aos determinantes. Somente com Arthur Cayley[12] (1821-1895) é que elas passam a ter seu espaço, e chegam a tornar-se, até mesmo, mais importantes que os determinantes.

Em 1857, Cayley introduziu as matrizes na Matemática. Ele propôs a definição da adição de matrizes, da multiplicação de matrizes e da multiplicação de matrizes por um número. Além disso apresentou a matriz identidade como elemento neutro do produto matricial e a matriz nula como elemento neutro da adição de matrizes. A partir dessas definições, as operações com matrizes passaram a ser pensadas como formação de uma Álgebra Matricial, acarretando um enorme desenvolvimento da teoria das matrizes.

Atualmente, as matrizes são utilizadas em várias áreas de conhecimento. Suas aplicações se dão na Matemática, na Física, na Biologia, na Economia, na Engenharia, na Computação, entre outras. Por exemplo, só na área de Matemática vários conteúdos como álgebra, equações diferenciais, probabilidade, estatística, teoria de grafos, utilizam as matrizes no seu desenvolvimento.

Mas afinal, para que servem as matrizes?

Desde sua primeira aparição na antiga China, as matrizes tem se mantido como uma importante ferramenta matemática. Elas são utilizadas não somente para resolver sistemas de equações lineares (SMOLLER, 2004), mas também para:

- o desenvolvimento de programas *3D*, em computação gráfica (FELDMAN, 1997);
- analisar relacionamentos (McWORTER, 2004);
- para esboçar complicados passos de dança (PETERSON, 1997);
- analisar as informações obtidas com o medidor de vibrações, em eletrônica;

---

determinante, com o sentido atual, surgiu em 1812 num trabalho de Cauchy sobre o assunto (DOMINGUES, 2004).

[12] Arthur Cayley (1821-1895): matemático inglês, um dos primeiros a trabalhar com Álgebra.

- codificar e/ou decodificar mensagens, em sistemas de segurança.

A aplicação de matrizes em sistemas de segurança foi um dos assuntos abordados no Projeto E.I.C.A.L.G.A. através de programas desenvolvidos com o *software Matlab* e, em especial, no ambiente *Delphi*. Ainda, antes de fazer a apresentação desses programas, é necessário fazer uma contextualização geral da teoria envolvida nessa aplicação, a Criptografia.

CAPÍTULO 4 - CRIPTOGRAFIA

A Criptografia foi criada pela necessidade de se ter ferramentas capazes de proteger a informação e de prover segurança aos dados que são armazenados e transmitidos por alguém, ou por um grupo.

Criptografia - do grego *Kryptos* (escondido, oculto) + *grapho* (grafia, escrita) - é o processo de conversão, também conhecido como **ciframento** ou **codificação**, de uma mensagem (texto comum) em código secreto (texto cifrado), o qual pode ser feito através de uma série de procedimentos.

O processo de recuperação da mensagem, novamente em forma de texto comum, é conhecido como **decriptografia**[13], podendo ser denominado também como **deciframento** ou **decodificação**.

A evolução da Criptografia aconteceu em três fases distintas:

- a criptografia manual, como exemplo pode-se citar o cifrário de César[14];
- a criptografia por máquinas, cujo exemplo mais conhecido é o Código Morse [15];
- a criptografia em rede, responsável pelo comércio eletrônico, como exemplo tem-se o DES (*Data Encryption Standard*) da IBM e o RSA (Rivest, Shamir, Adleman) [16].

A criptografia pode ser classificada quanto ao número de chaves que são utilizadas:

- Criptografia simétrica: utiliza mesma chave para criptografar e decriptografar;
- Criptografia assimétrica: utiliza chaves diferentes.

---

[13] Em alguns trabalhos esse termo aparece como descriptografia.

[14] No Império Romano, o imperador Júlio César utilizava textos cifrados para enviar mensagens secretas a generais que estavam no front de batalha, impedindo que o inimigo quando interceptasse suas mensagens, descobrisse o que estava sendo mandado.

[15] O *Código Morse* representa os caracteres através de "pontos" e "traços" correspondendo estes a impulsos elétricos e resultando daí sinais acústicos ou luminosos de uma certa duração.

[16] O DES utiliza uma chave de 56 bits e opera em blocos de 64 bits. Foi projetado inicialmente para ser utilizado em componentes de hardware. Nos dias atuais, ele é usado na Internet em conexões Web segura. O RSA é o mais conhecido dos métodos de criptografia de chave pública. Este código foi inventado em 1978 por Ronal Rivest, Adi Shamir e Leonard Adleman. É, atualmente, o mais usado em aplicações comerciais [Coutinho (1997), p.3]. Ele possui um algoritmo que usa grandes números primos para gerar um texto cifrado seguro.

A criptografia simétrica conhecida como “criptografia convencional”, utiliza uma única chave secreta. É necessário que ambas as partes interessadas (emissor e receptor) compartilhem suas chaves.[17]

A criptografia assimétrica utiliza uma chave pública e uma chave privada. A idéia é fazer a criptografia de uma mensagem com a chave pública e a decriptografia com a chave privada, ou vice-versa.

A forma pela qual as chaves são utilizadas para criptografar ou decriptografar uma mensagem, é que garante a autenticação, a confiabilidade e a integridade dessa mensagem.

A Criptografia, em geral, é muito utilizada em transações bancárias, ou em quaisquer outras operações que envolvam senhas e documentações importantes, principalmente aquelas realizadas através da Internet.

Por sua enorme importância ela é muito estudada por matemáticos, cientistas e engenheiros. Muitos livros e artigos são publicados sobre esse tema. Até mesmo em obras literárias como no conto “A aventura dos Dançarinos” de Arthur Conan Doyle, onde o autor “*envolve seu famoso personagem Sherlock Holmes numa trama criptográfica*” (PEIXOTO, 2000).

Na Universidade Federal de Santa Catarina (UFSC) também é possível encontrar várias dissertações de mestrado que abordam, de maneira diversificada, o tema, como por exemplo em “*Um estudo sobre a aplicabilidade de redes neurais em* ***criptografia***”, onde “*inspirado no perfeito processo da comunicação entre as células do sistema nervoso, implementam-se três propostas de cifradores, utilizando redes neurais artificias”* (CARNEIRO, 2001). Além disso, muitos encontros e congressos são organizados dedicando-se também à Criptografia.

Mas, qual é a relação existente entre a Criptografia e a Matemática?

As técnicas utilizadas para criptografar são, em sua grande parte, desenvolvidas por matemáticos, e envolvem conteúdos relacionados a Matemática. Quanto mais difíceis e trabalhosas forem essas técnicas mais difícil e trabalhoso se tornará o processo de deciframento.

[17] O emissor possui uma chave e o receptor possui outra. Para haver comunicação é necessário que ambos troquem suas chaves e armazenem de maneira segura. Um dos problemas encontrados é que essa segurança não pode ser garantida.

Como o objetivo desse capítulo é fazer apenas uma aborgem geral e voltada principalmente para o Ensino Médio, o modelo utilizado para exemplificar essas técnicas será a *Cifra de Hill*[18]. A *Cifra de Hill* é uma classe de sistema poligráfico[19] baseada em transformações matriciais e arimética modular (anexo 2).

Para criptografar uma mensagem utilizando *Cifra de Hill*, segue-se os seguintes passos:

1) Constrói-se uma tabela de conversão de letras em números:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0

2) Escolhe-se uma matriz de ordem n. Essa matriz será a chave do ciframento e pode ser denominada *matriz codificadora*. É necessário, no entanto, que ela possua inversa (o porquê será verificado mais adiante).

Como exemplo, será utilizada a seguinte matriz de ordem 2:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix}$$

3) Escolhe-se uma mensagem, por exemplo: **AS MATRIZES**. Em seguida, agrupa-se o texto em pares de letras[20], e faz-se a conversão de acordo com a tabela:

A	S	M	A	T	R	I	Z	E	S
1	19	13	1	20	18	9	0	5	19

4) Cada par de números deve ser convertido em um vetor coluna[21]. Cada vetor coluna criado será multiplicado pela matriz codificadora, da seguinte forma:

$$\begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 19 \end{bmatrix} = \begin{bmatrix} 39 \\ 57 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix} \cdot \begin{bmatrix} 13 \\ 1 \end{bmatrix} = \begin{bmatrix} 15 \\ 3 \end{bmatrix}$$

[18] O nome é referência a Lester S. Hill que introduziu esse sistema em dois artigos.
[19] Sistema de criptografia no qual o texto comum é dividido em conjuntos de **n** letras, cada um dos quais é substituído por um conjunto de **n** letras cifradas.
[20] Se o número de letras for ímpar, adiciona-se uma letra fictícia ou um símbolo ao final do texto.
[21] O vetor coluna é uma matriz de ordem n x 1.

$$\begin{bmatrix}1 & 2\\0 & 3\end{bmatrix}\cdot\begin{bmatrix}20\\18\end{bmatrix}=\begin{bmatrix}56\\54\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\0 & 3\end{bmatrix}\cdot\begin{bmatrix}9\\0\end{bmatrix}=\begin{bmatrix}9\\0\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\0 & 3\end{bmatrix}\cdot\begin{bmatrix}5\\19\end{bmatrix}=\begin{bmatrix}43\\57\end{bmatrix}$$

5) Faz-se a conversão dos valores encontrados de acordo com a tabela. Como pode se observar no exemplo acima, existem alguns valores que não possuem correspondente na tabela.

Nesse caso, e sempre que isso ocorrer, é necessário aplicar a aritmética modular (anexo 2). Quando o resultado obtido for maior que 25, ele será substituído pelo resto da divisão desse número por 26. É por esse motivo que a letra Z assume valor 0 na tabela. Assim:

$$\begin{bmatrix}1 & 2\\0 & 3\end{bmatrix}\cdot\begin{bmatrix}1\\19\end{bmatrix}=\begin{bmatrix}39\\57\end{bmatrix}=\begin{bmatrix}13\\5\end{bmatrix}(\mathrm{mod}\,26)$$

$$\begin{bmatrix}1 & 2\\0 & 3\end{bmatrix}\cdot\begin{bmatrix}13\\1\end{bmatrix}=\begin{bmatrix}15\\3\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\0 & 3\end{bmatrix}\cdot\begin{bmatrix}20\\18\end{bmatrix}=\begin{bmatrix}56\\54\end{bmatrix}=\begin{bmatrix}4\\2\end{bmatrix}(\mathrm{mod}\,26)$$

$$\begin{bmatrix}1 & 2\\0 & 3\end{bmatrix}\cdot\begin{bmatrix}9\\0\end{bmatrix}=\begin{bmatrix}9\\0\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\0 & 3\end{bmatrix}\cdot\begin{bmatrix}5\\19\end{bmatrix}=\begin{bmatrix}43\\57\end{bmatrix}=\begin{bmatrix}17\\5\end{bmatrix}(\mathrm{mod}\,26)$$

Fazendo a conversão:

13	**5**	**15**	**3**	**4**	**2**	**9**	**0**	**17**	**5**
M	**E**	**O**	**C**	**D**	**B**	**I**	**Z**	**Q**	**E**

obtém-se a seguinte mensagem: **MEOCDBIZQE**

Para decifrar a mensagem obtida (texto cifrado, ou codificado), segue-se os seguintes passos:

1) Obtem-se a inversa (anexo 1) da matriz codificadora, que será a chave para decifrar o texto cifrado, também denominada *matriz decoficadora*.

Utilizando-se o metódo da matriz adjunta (anexo 2):

$$\mathbf{A^{-1} = (3)^{-1} \cdot \begin{bmatrix} 3 & -2 \\ 0 & 1 \end{bmatrix} (mod\ 26)}$$

$$\mathbf{A^{-1} = 9 \cdot \begin{bmatrix} 3 & -2 \\ 0 & 1 \end{bmatrix}}$$

$$\mathbf{A^{-1} = \begin{bmatrix} 27 & -18 \\ 0 & 9 \end{bmatrix}}$$

$$\mathbf{A^{-1} = \begin{bmatrix} 1 & 8 \\ 0 & 9 \end{bmatrix} (mod\ 26)}$$

Portanto a matriz decodificadora é:

$$D = \begin{bmatrix} 1 & 8 \\ 0 & 9 \end{bmatrix}$$

2) Em seguida, agrupa-se o texto cifrado em pares de letras, e faz-se a conversão de acordo com a tabela:

M	**E**	**O**	**C**	**D**	**B**	**I**	**Z**	**Q**	**E**
13	**5**	**15**	**3**	**4**	**2**	**9**	**0**	**17**	**5**

3) Cada par de números deve ser convertido em um vetor coluna. Cada vetor coluna criado será multiplicado pela matriz decodificadora, e quando valor obtido for maior do que 25, aplica-se a aritmética módulo 26:

$$\begin{bmatrix} 1 & 8 \\ 0 & 9 \end{bmatrix} \cdot \begin{bmatrix} 13 \\ 5 \end{bmatrix} = \begin{bmatrix} 53 \\ 45 \end{bmatrix} = \begin{bmatrix} 1 \\ 19 \end{bmatrix} (\text{mod } 26)$$

$$\begin{bmatrix} 1 & 8 \\ 0 & 9 \end{bmatrix} \cdot \begin{bmatrix} 15 \\ 3 \end{bmatrix} = \begin{bmatrix} 39 \\ 27 \end{bmatrix} = \begin{bmatrix} 13 \\ 1 \end{bmatrix} (\text{mod } 26)$$

$$\begin{bmatrix} 1 & 8 \\ 0 & 9 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 2 \end{bmatrix} = \begin{bmatrix} 20 \\ 18 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 8 \\ 0 & 9 \end{bmatrix} \cdot \begin{bmatrix} 9 \\ 0 \end{bmatrix} = \begin{bmatrix} 9 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 8 \\ 0 & 9 \end{bmatrix} \cdot \begin{bmatrix} 17 \\ 5 \end{bmatrix} = \begin{bmatrix} 57 \\ 45 \end{bmatrix} = \begin{bmatrix} 5 \\ 19 \end{bmatrix} (\text{mod } 26)$$

5) Faz-se a conversão dos valores encontrados de acordo com a tabela:

1	**19**	**13**	**1**	**20**	**18**	**9**	**0**	**5**	**19**
A	**S**	**M**	**A**	**T**	**R**	**I**	**Z**	**E**	**S**

Retornou-se, então, a mensagem original: **AS MATRIZES**.

Foi baseado nesse tipo de ciframento que desenvolveu-se com o *software Matlab* dois programas para o Projeto E.I.C.A.L.G.A.: o **Criptografia** e o **Decriptografia**, cujo objetivo era mostrar aos alunos do Ensino Médio e de Graduação em Matemática uma aplicação de matrizes em sistemas de segurança.

4.1. Programas desenvolvidos com o *software Matlab* e no ambiente *Delphi*

No *Matlab* para se ter acesso a um programa[22] é necessário chamá-lo no diretório em que está gravado. Por exemplo, se o programa estiver no disquete, dentro da pasta ***matlab*** e ainda dentro da pasta ***prog***, deve-se usar o seguinte comando: **cd a:/matlab/prog**. Em seguida chama-se o programa (fig. 11):

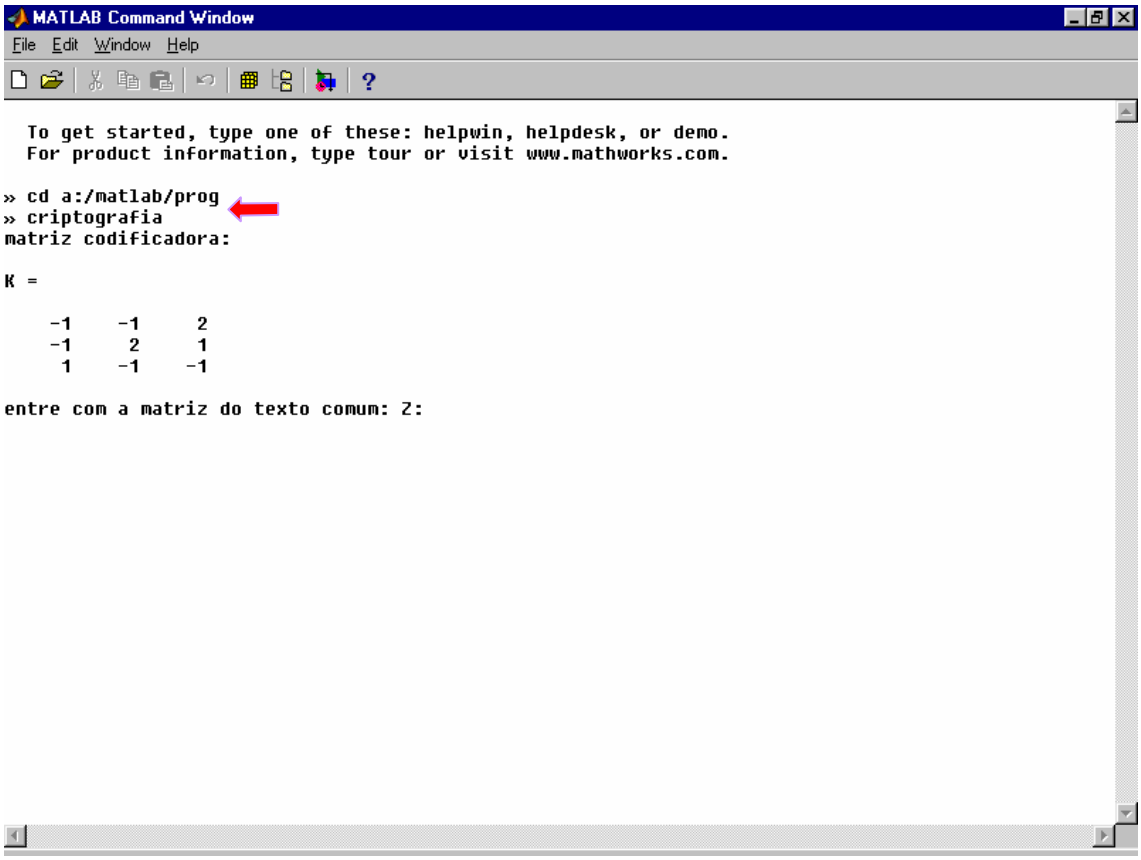


fig. 11 – o programa *Criptografia* (*MATLAB*)

[22] Esse programa é editado em um arquivo com extensão *.m*.

O programa **Criptografia** inicia mostrando a matriz codificadora e solicitando a matriz do texto comum, escolhida pelo usuário. Essa matriz deve ser de ordem 3. O usuário deve utilizar uma tabela de conversão, como a utilizada na *cifra de Hill*, para formar essa matriz.

Considere como exemplo a palavra **MATRIZES**. Como essa palavra só possui 8 letras, acrescenta-se no final mais uma letra[23], por exemplo, o **S**. Fazendo a conversão, obtem-se:

M	**A**	**T**	**R**	**I**	**Z**	**E**	**S**	**S**
13	**1**	**20**	**18**	**9**	**0**	**5**	**19**	**19**

Após a conversão, deve-se seguir os seguintes passos:

- separa-se as letras em grupos, nesse caso, em três grupos de três letras consecutivas;
- escreve-se a matriz onde o primeiro grupo de letras será a primeira coluna, o segundo será a segunda coluna e o terceiro a terceira coluna:

$$\begin{bmatrix} 13 & 18 & 5 \\ 1 & 9 & 19 \\ 20 & 0 & 19 \end{bmatrix}$$

- escreve-se essa matriz no programa da seguinte forma:

[13 18 5 ; 1 9 19 ; 20 0 19]

Em seguida, pressiona-se a tecla ***enter*** e o programa executa o processo de ciframento, como pode ser visto nas figuras 12, 13 e 14:

[23] Nesse caso, não pode ser usado um outro símbolo, é necessário que seja uma letra.

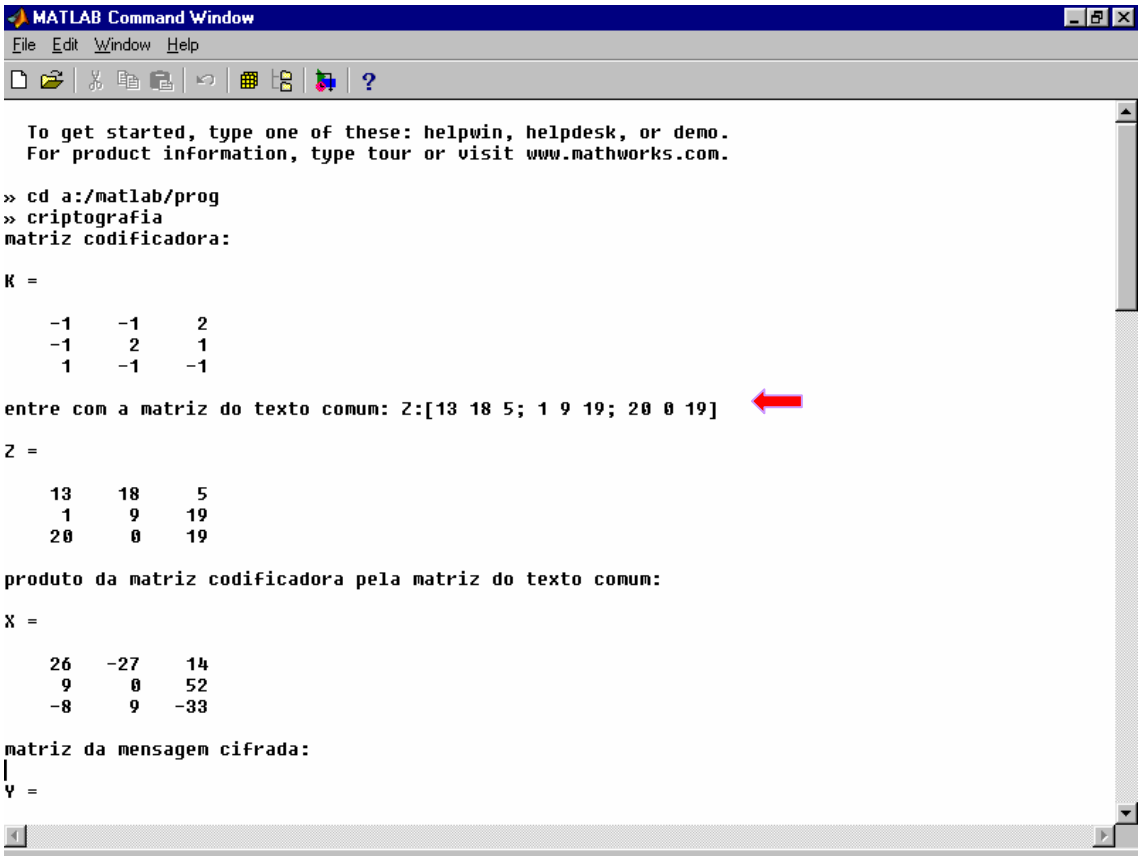


fig. 12 – o processo de ciframento

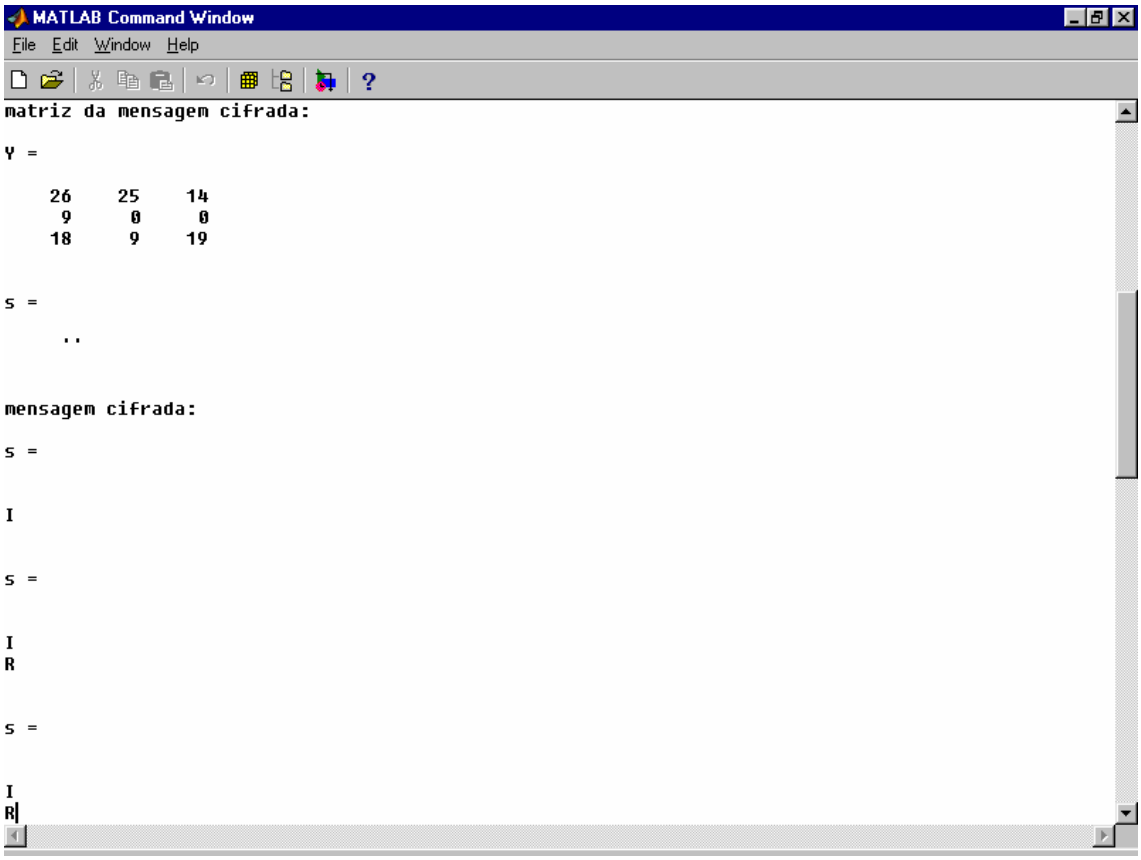


fig. 13 – o processo de ciframento (cont.)

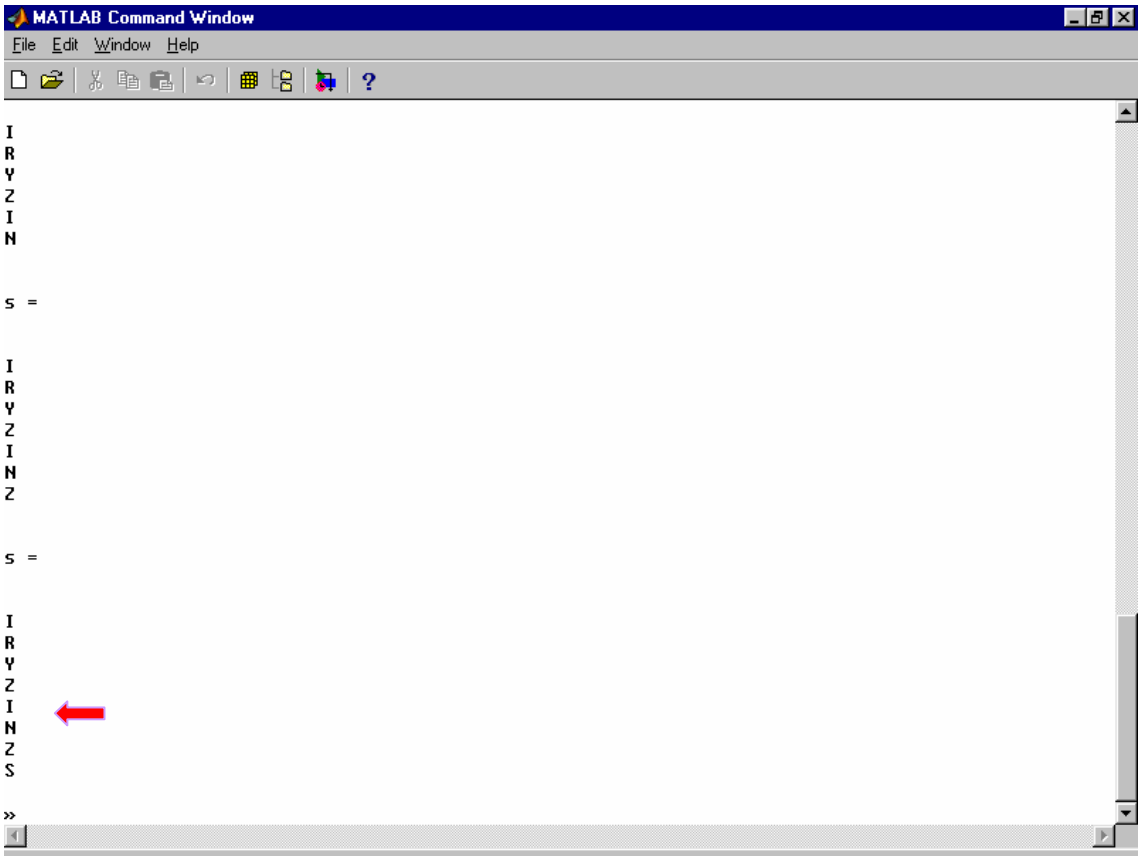


fig. 14 – finalização do processo de ciframento

A mensagem codificada é: **IRYZINZS**.

Para decifrar essa mensagem utiliza-se o programa **Decriptografia** (fig. 15).

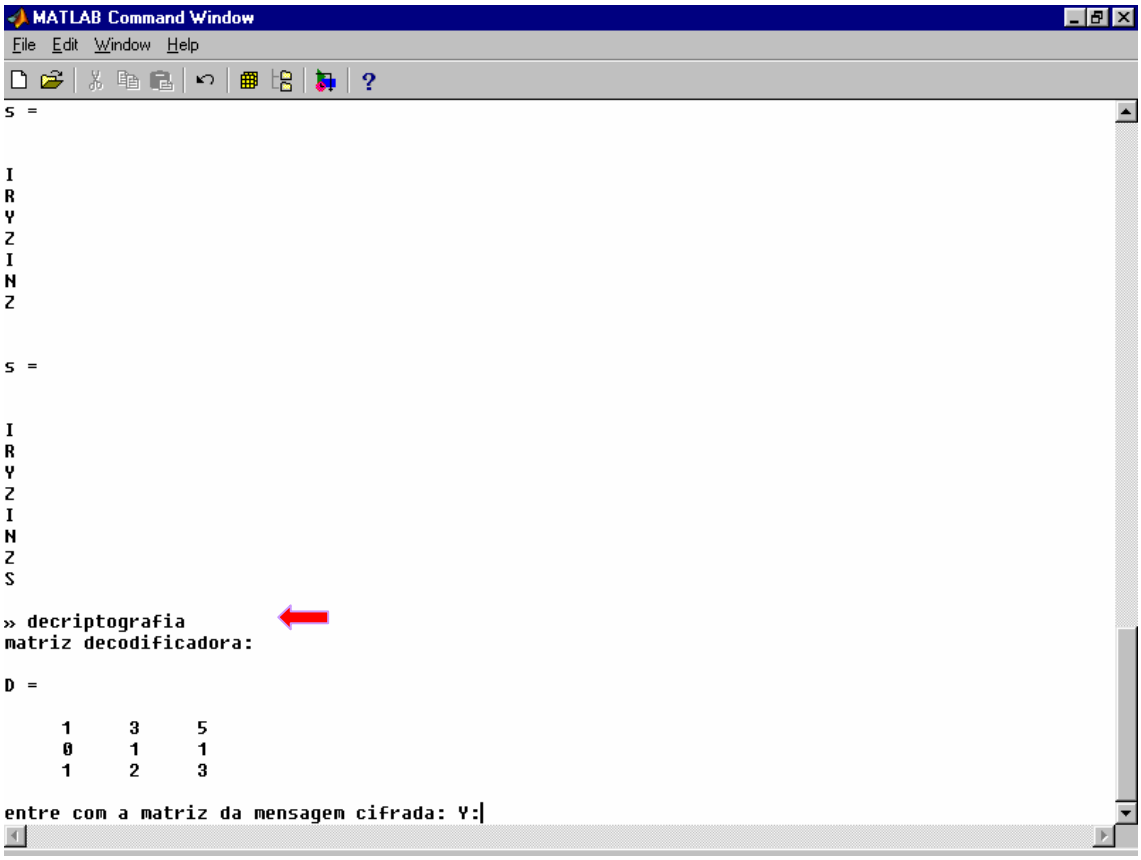


fig. 15 – o programa *Decriptografia* (*Matlab*)

Assim como o **Criptografia**, esse programa inicia mostrando a matriz decodificadora e solicitando a matriz da mensagem cifrada (codificada). Não é necessário, nesse caso fazer a conversão, pois o programa **Criptografia** fornece a matriz dessa mensagem (fig. 16).

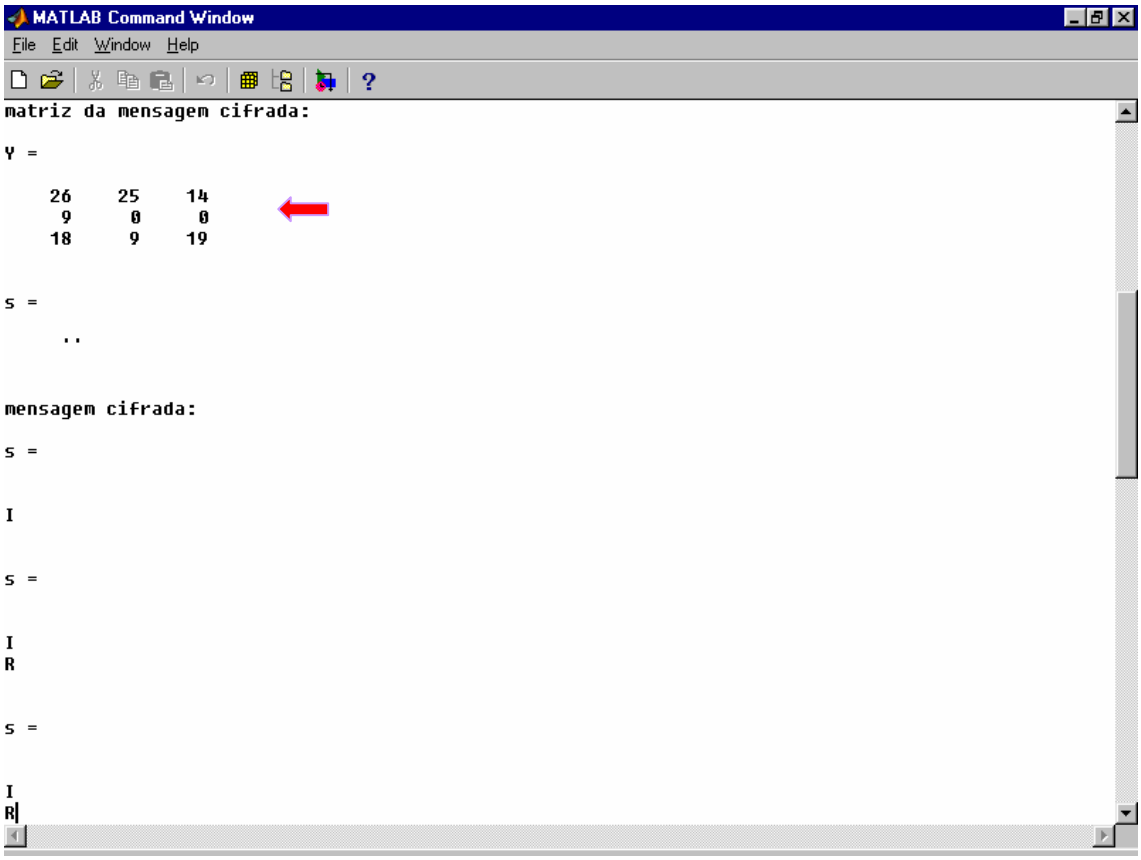


fig. 16 – a matriz da mensagem cifrada

Basta, então, escrever a matriz como mostra abaixo:

$$[26 \quad 25 \quad 14 \quad ; \quad 9 \quad 0 \quad 0 \quad ; \quad 18 \quad 9 \quad 19]$$

Pressionando a tecla ***enter***, obtem-se o processo de deciframento (decodificação) como mostram as figuras 17 e 18:

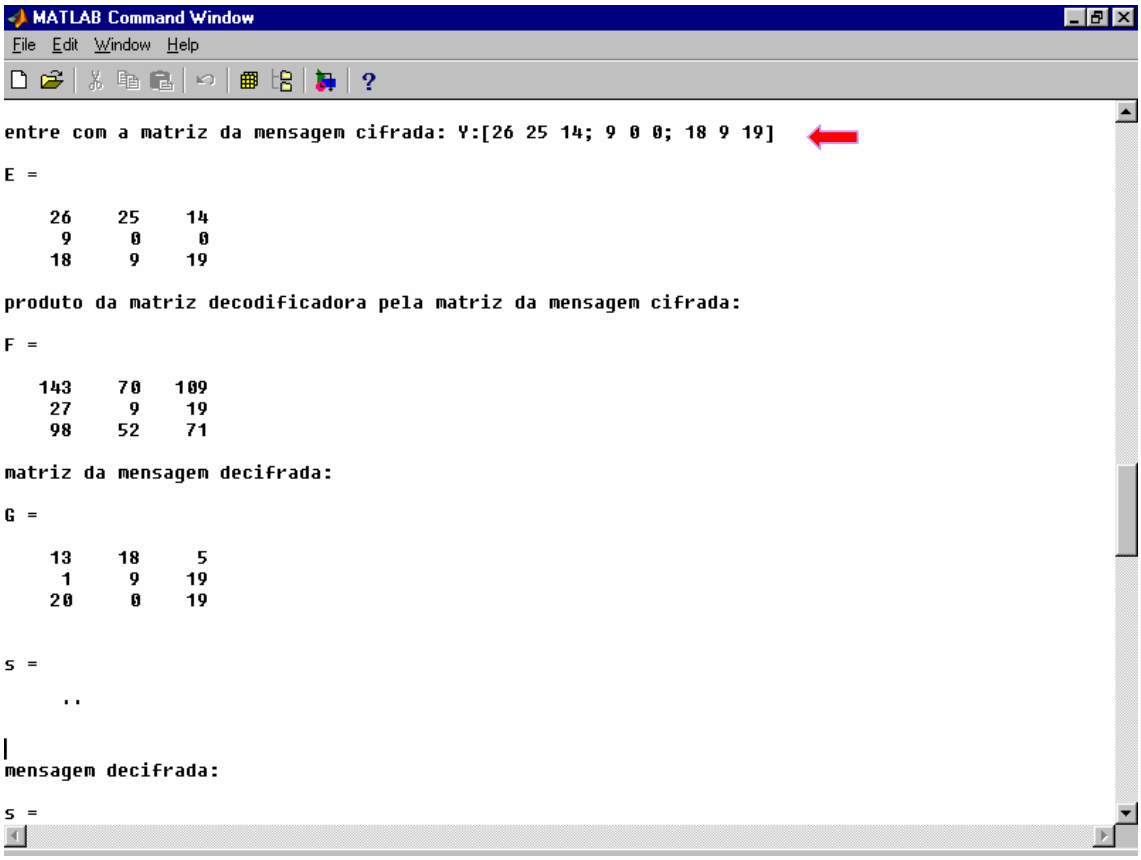


fig. 17 – o processo de deciframento

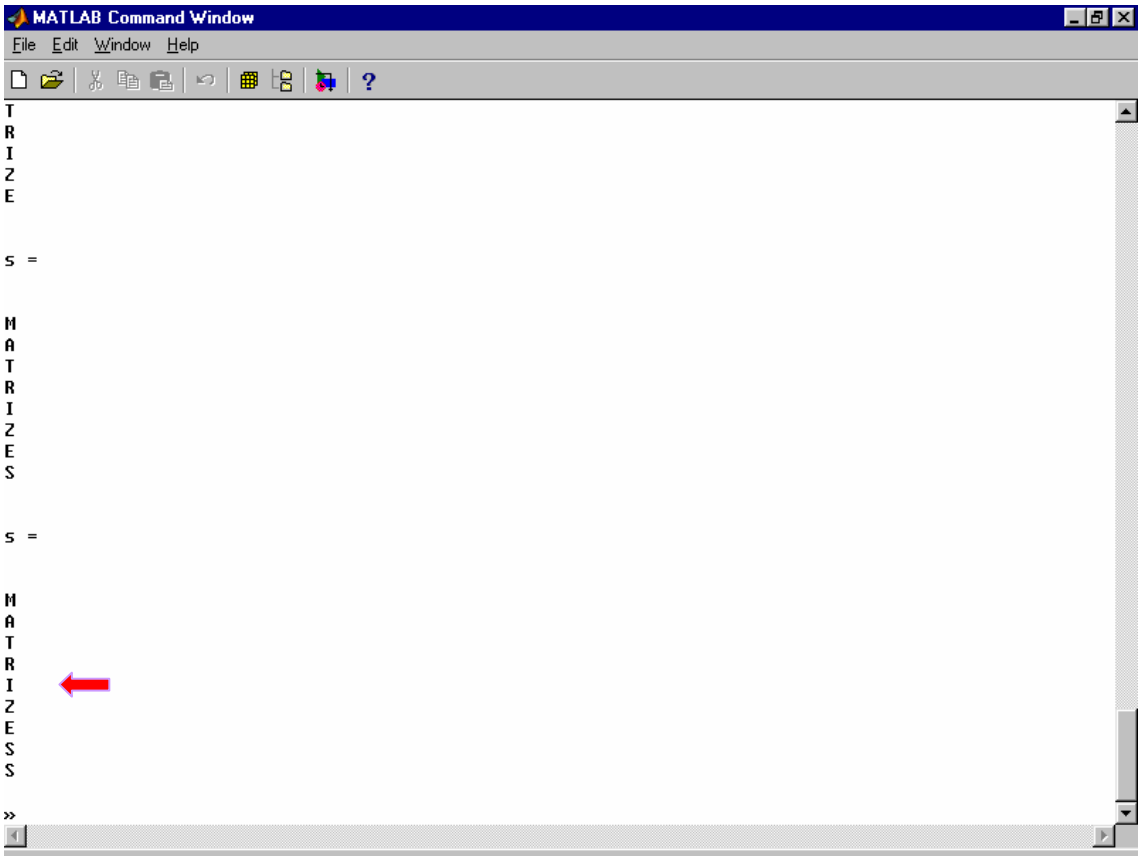


fig. 18 – finalização do processo de deciframento

Retorna–se então à mensagem inicial: **MATRIZES**.

Os códigos dos programas podem ser visualizados no anexo 5. Para os dois programas, assim como no exemplo feito para *Cifra de Hill*, foi utilizado uma tabela de conversão com 26 caracteres, sendo Z = 0. As matrizes codificadora e decodificadora são de ordem 3, sendo as duas pré-estabelecidas no código interno dos programas, ficando omitido então o processo de inversão da matriz codificadora.

Esses dois programas foram testados no Projeto apenas pelos alunos da Graduação de Matemática. Alguns fatores os tornaram não aplicáveis às demais apresentações do Projeto E.I.C.A.L.G.A.:

- O *Matlab* não é um *software* gratuito, o que restringe o acesso a ele;
- A *interface* não é muito interativa para o tipo de programa que se estava trabalhando. Para se visualizar o processo todo de ciframento e deciframento é necessário utilizar "a barra de rolagem" (no canto direito da tela);
- A disposição das mensagens não ficaram claras;
- A maneira de escrever as matrizes poderia confundir o usuário, especialmente se esse fosse um aluno de Ensino Médio;

Como o objetivo principal do Projeto E.I.A.L.G.A. era trabalhar, em especial, com alunos do Ensino Médio, fez-se necessário buscar um outro ambiente para desenvolver essa atividade. Optou-se, então, pela utilização do ambiente *Delphi*. Com ele desenvolveu-se um novo programa, também denominado **Criptografia**[24] (fig. 19):

[24] O fato de se ter colocado o mesmo nome em dois programas, nesse caso, não causou transtorno, pois o programa feito em Matlab, pelos motivos citados no texto, não foi mais utilizado na prática. Para evitar confusão durante a leitura desse trabalho, o Criptografia e Decriptografia serão mencionados, no decorrer desse trabalho, como "*os programas feitos em Matlab*".

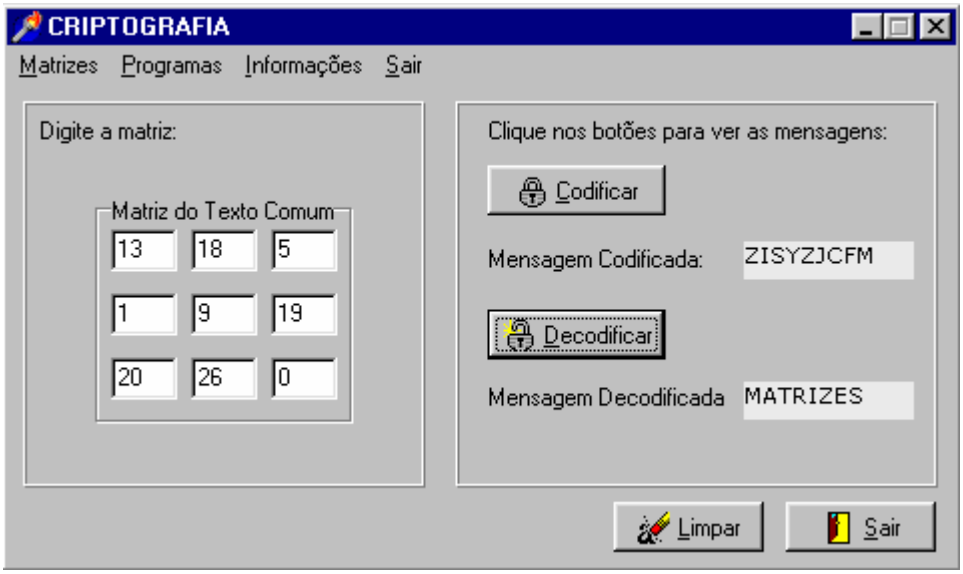


fig. 19 – o programa *Criptografia* (*Delphi*)

O programa **Criptografia** difere do exemplo trabalhado para *Cifra de Hill* e dos programas feitos em *Matlab* nos seguintes pontos:

- A mensagem não precisa ser separada em vetores colunas, basta fazer a conversão das letras em números e inserir na *Matriz do Texto Comum* (fig. 19);
- O programa trabalha com 27 caracteres, ou seja, aritmética de módulo 27, pois foi incluido o espaço em branco, representado na tabela pelo símbolo "_":

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	0

- A mensagem pode conter espaço entre as palavras, como por exemplo, a mensagem **BOA_TARDE** (fig. 20):

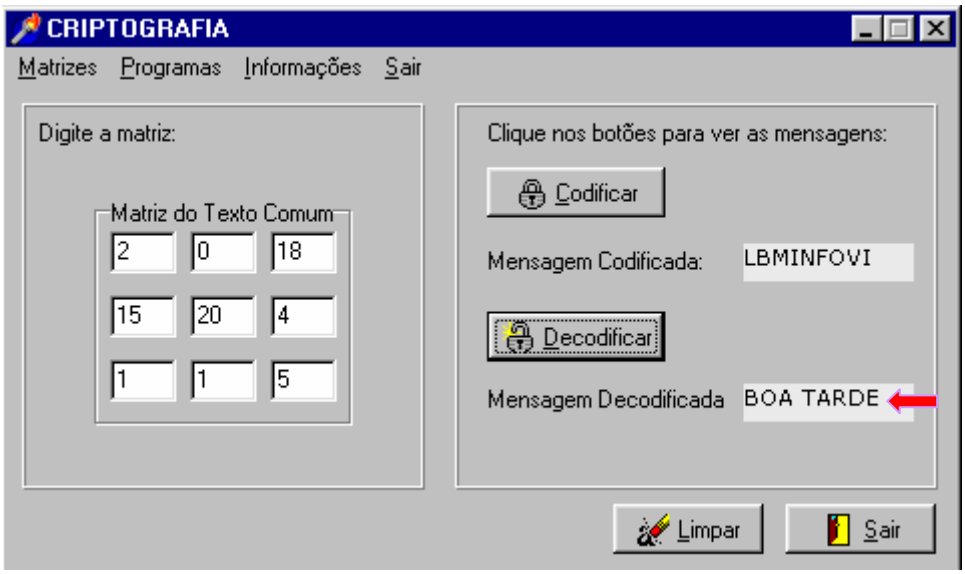


fig. 20 – mensagem contendo espaço

- A decodificação é feita sem que o usuário precise realizar o processo de conversão de letras em número e a transposição da mensagem em matriz;
- O processo de codificação e decodificação é feito internamente pelo programa. O usuário, caso tenha interesse, pode visualizar o processo acessando o menu *Matrizes* (fig. 21), onde pode optar por conhecer as matrizes *Codificadora* e *Decodificadora*, as matrizes *Produto* e as matrizes *Mensagem* (fig. 22a, 22b, 22c, 22d, 22e, 22f):

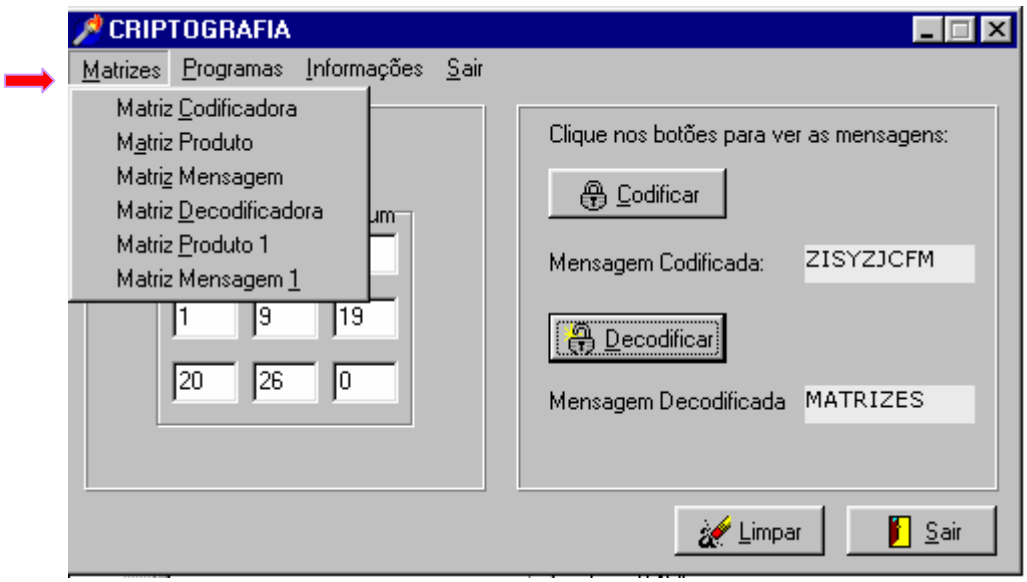


fig. 21 – o menu *Matrizes*

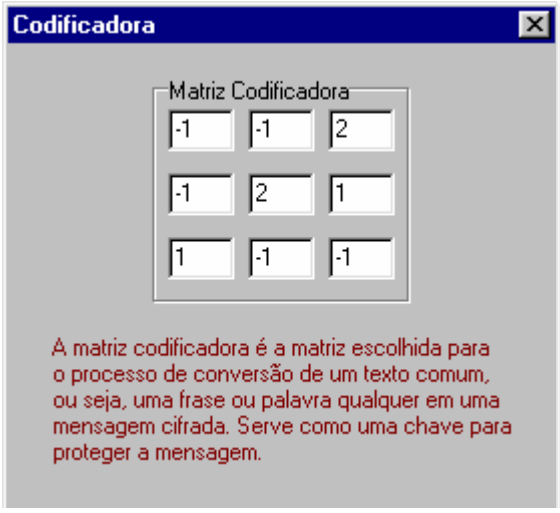


fig. 22a – matriz codificadora

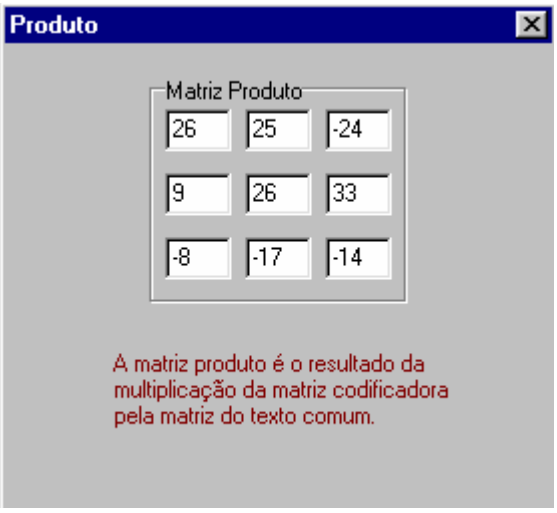


fig. 22b – matriz produto

fig. 22c – matriz mensagem

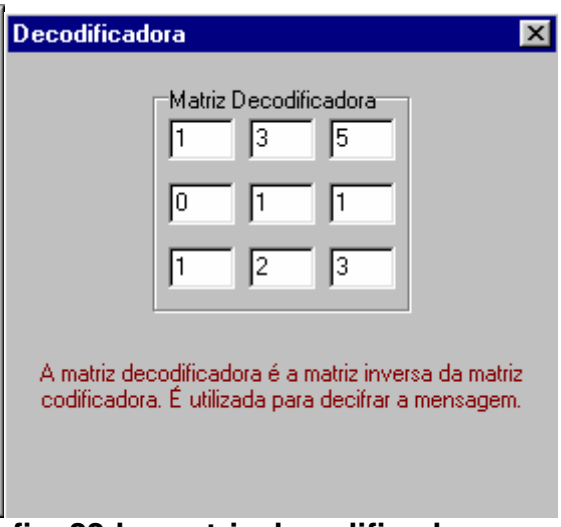


fig. 22d – matriz decodificadora

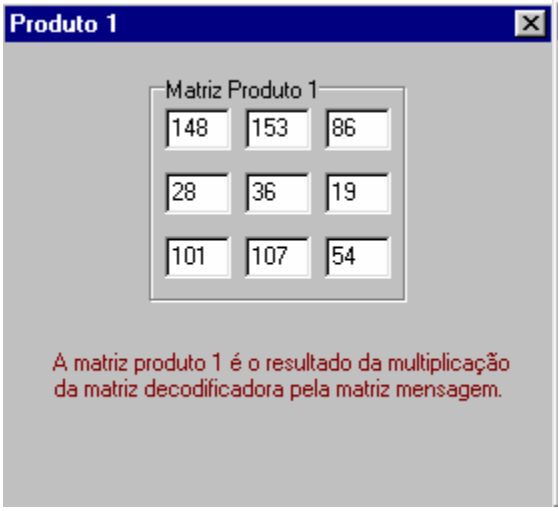


fig. 22e – matriz produto 1

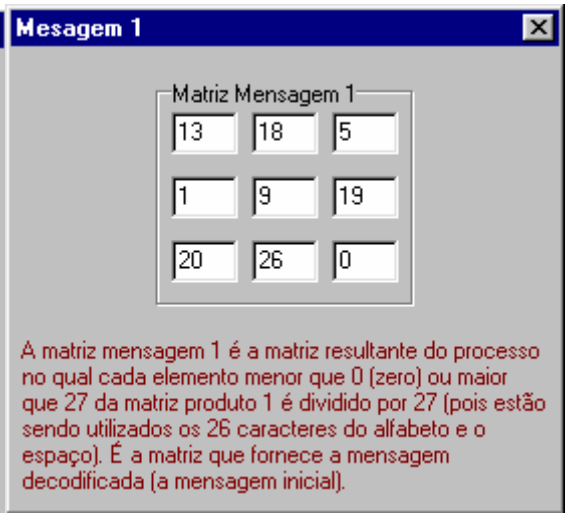


fig. 22f – matriz mensagem 1

Assim como nos programas feitos no *Matlab*, as matrizes *codificadora* (fig. 22a) e *decodificadora* (fig. 22d) são de ordem 3, e também foram pré-estabelecidas. A mensagem de texto pode conter apenas 9 caracteres, pois a matriz de texto comum também é de ordem 3.

Para alertar o usuário quanto ao uso indevido de valores maiores que 27 ou menores que 0 (zero), pois a aritmética utilizada é módulo 27, o programa emite a mensagem abaixo (fig. 23) assim que o usuário escreve o valor:

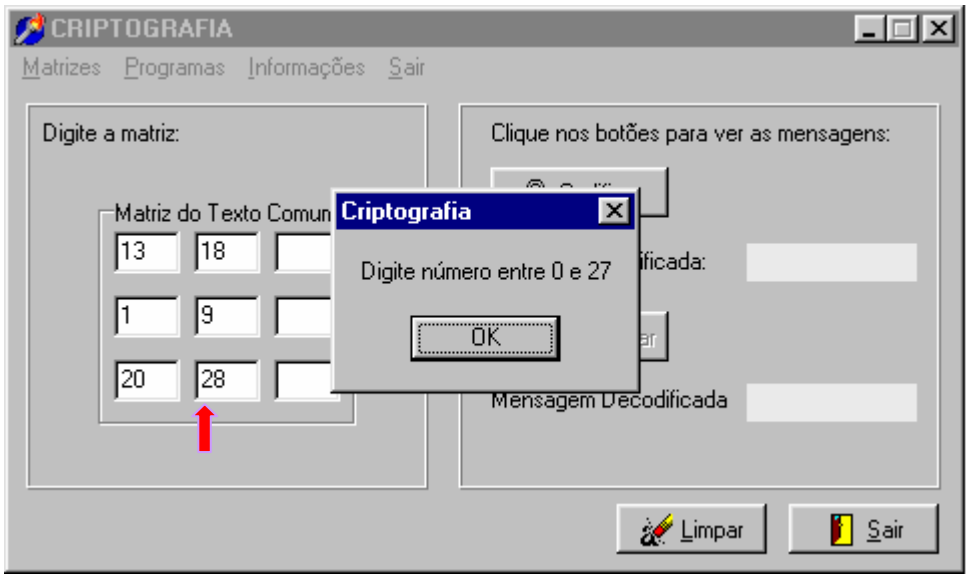


fig. 23 – número maior que 27

O programa emite também uma mensagem quando é deixado um espaço em branco na matriz (fig. 24) ou um espaço após o dígito (fig. 24a) :

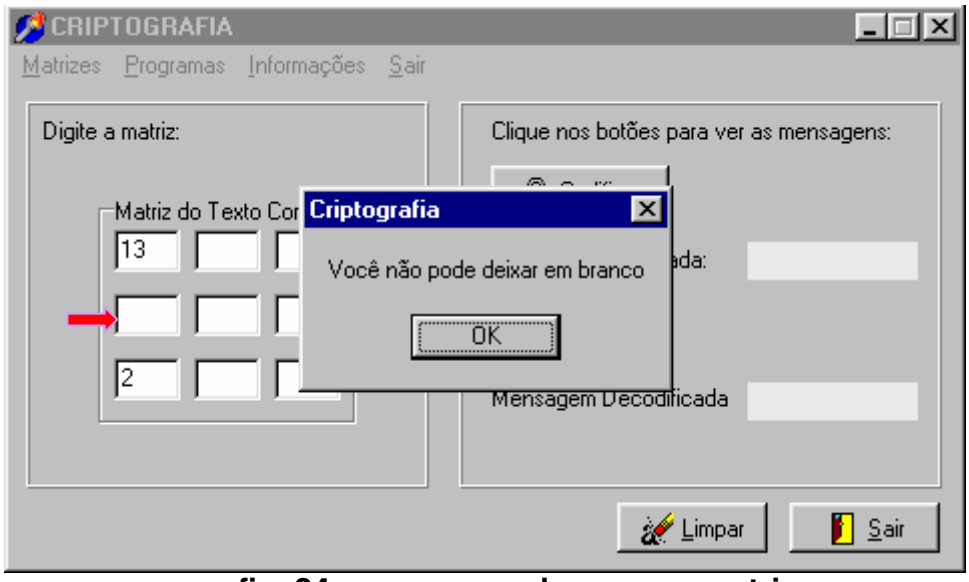


fig. 24 – espaço em branco na matriz

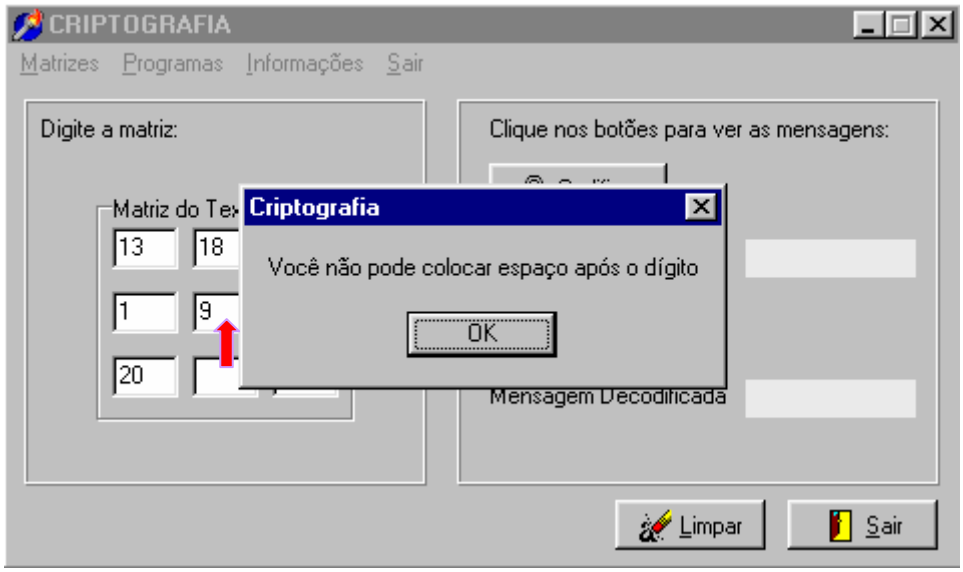


fig. 24a – espaço após o dígito

Um fator muito importante a se mencionar ainda é que o programa **Criptografia**, bem como qualquer outro programa desenvolvido no ambiente *Delphi*, pode ser instalado em qualquer computador, não sendo necessário ter instalado o ambiente e os arquivos de edição. Garante-se, com isso, segurança, pois o programa não pode ser alterado por qualquer usuário sem autorização e redução no espaço necessário para armazenar o programa. Isso por que o ambiente *Delphi* possibilita a criação de programas executáveis[25], com uma interface que dinamiza a interação usuário/computador, isto é, fornece um ambiente amigável, muito útil na elaboração de jogos e atividades.

Ainda com o objetivo de criar uma atividade mais interessante para ser apresentada aos alunos do Ensino Médio, fez-se necessária a ampliação da quantidade de caracteres para compor uma mensagem e a integração da matriz do texto codificado, para fixar a importância dessa parte do processo.

Foram desenvolvidos, então, os programas **Codificação** (fig. 25) e **Decodificação** (fig. 26), os quais possibilitam a criação de mensagens de até 36 caracteres, incluindo espaços em branco.

[25] Que não necessitam ter os arquivos de edição instalados no computador.

fig. 25 – o programa *Codificação*

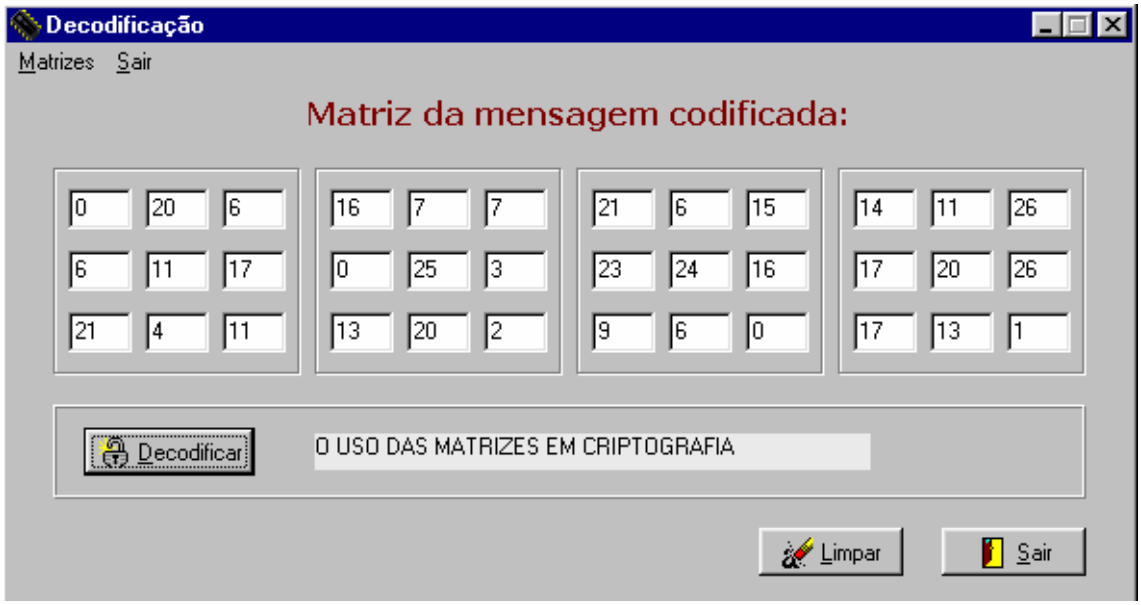


fig. 26 - o programa *Decodificação*

Eles possuem os mesmos princípios do programa **Criptografia**, ou seja, a mesma matriz codificadora, a mesma matriz decodificadora, as mensagens de aviso quanto espaço em branco, espaço após um dígito, uso de números maiores que 27 e menores que 0 (zero). O menu *Matrizes*, também mostra o processo interno dos programas, sendo a única diferença em relação ao programa anterior o tamanho da matriz mensagem (fig. 27) e da matriz produto, que é equivalente ao tamanho da matriz do texto comum e da matriz da mensagem codificada.

fig. 27 – o tamanho da matriz

Eles podem ser executados a partir do item *Programas* (fig. 28) desse programa[26]:

fig. 28 – acessando os programas *Codificação* e *Decodificação*

É importante mencionar que todos os programas foram feitos durante um tempo determinado para que pudessem ser apresentados junto com as atividades do Projeto E.I.C.A.L.G.A. Portanto, foram encontradas, algumas dificuldades durante a utilização desses programas. Entre elas, o fato de a tabela de conversão, essencial para execução dos mesmos, ser fornecida apenas na folha de atividade do projeto. Ou seja, o usuário não teria acesso a ela caso não tivesse participado da atividade. Outra dificuldade foi em trabalhar

[26] Como os três programas seriam apresentados no projeto E.I.C.A.L.G.A., para facilitar o acesso a eles foi criado esse menu.

com duas janelas (fig. 25 e 26) para realizar o processo de codificação e decodificação. O momento de transpor a mensagem codificada para que pudesse ser decodificada tornou-se um pouco confuso.

Com o intuito de melhorar o ambiente e proporcionar, assim uma melhor interação usuário/programa, fez-se uma versão, também em *Delphi,* que une os programas **Codificação** e **Decodificação** e fornece a tabela de conversão, o programa **Criptografando** (fig. 29). Apesar das modificações feitas, não houve alteração nos códigos dos programas.

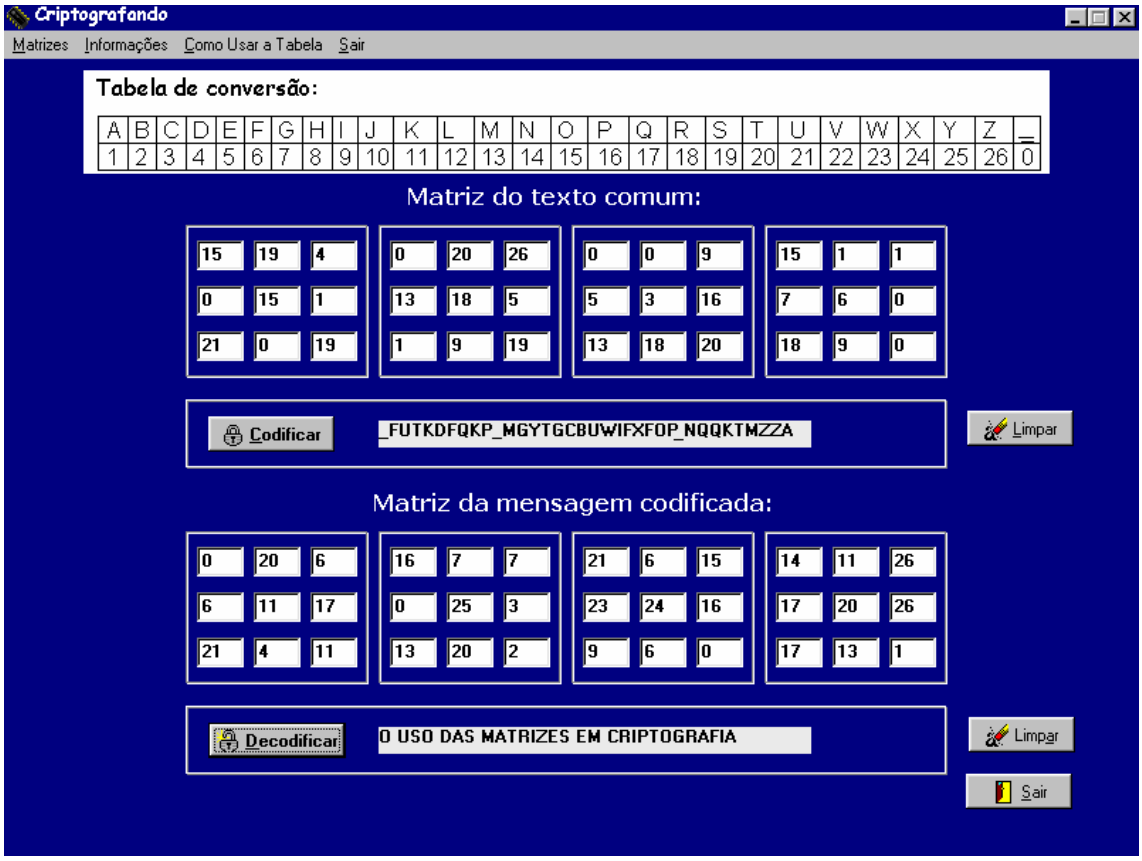


fig. 29 – o programa *Criptografando*

O menu *Matrizes,* assim como nos programas anteriores, traz todas as informações sobre o processo realizado internamente.

Adicionou-se também a essa versão o menu *Como usar a tabela* (fig. 30), que ainda trás um exemplo de como escrever os elementos na matriz do texto comum, sendo que o mesmo vale para a matriz da mensagem codificada.

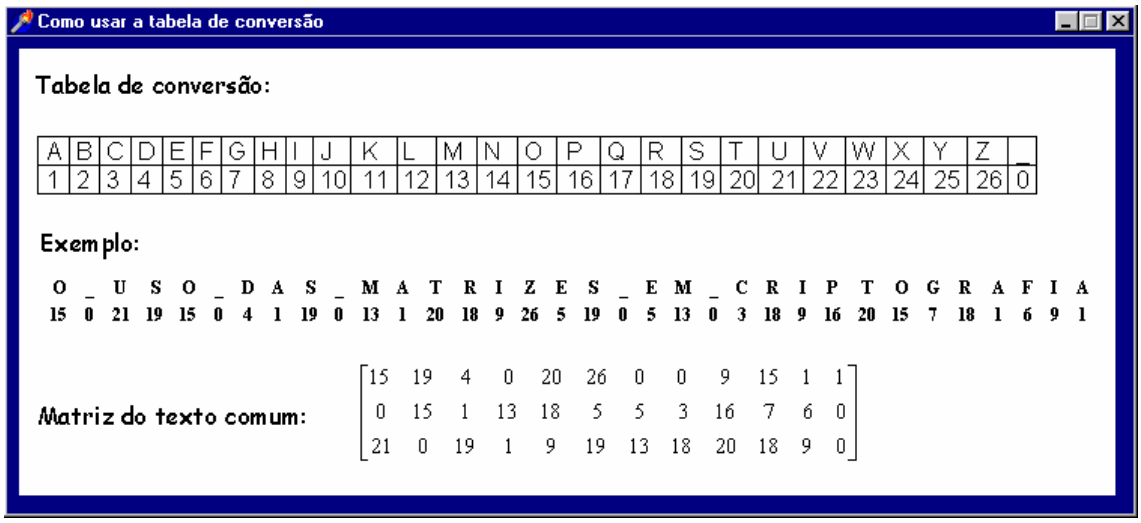


fig. 30 – como usar a tabela de conversão

É necessário mencionar que, embora seja possível, em nenhum dos programas desenvolvidos pôde ser utilizado acento, “ç” ou qualquer outro caractere que não conste nas tabelas de conversão apresentadas, porque isso tornaria um pouco mais trabalhosa a programação dos mesmos.

As atividades propostas aos alunos do Ensino Médio e de Graduação para utilização desses programas estão anexadas ao final do trabalho (anexo 3). O objetivo principal dessas atividades foi, através de um jogo interativo, mostrar uma aplicação do conteúdo matrizes, no caso, a aplicação de matrizes no sistema de segurança e com isso a importância da aprendizagem desse conteúdo.

CAPÍTULO 5 - O PROGRAMA *MATRIZES: TEORIA E APLICAÇÃO*

O envolvimento com os programas vistos no capítulo 5 e com as atividades feitas com o Projeto E.I.C.A.L.G.A. fez surgir a idéia de criar um outro aplicativo que pudesse fornecer ao usuário, além da parte de aplicação, a parte teórica, incluindo exemplos e exercícios, com uma abordagem diferente e agradável, que contribuisse no processo de ensino/aprendizagem.

Pondo em prática essa idéia, desenvolveu-se o programa **Matrizes: teoria e aplicação** (fig. 31).

fig. 31 – o programa *Matrizes: teoria e aplicação*

A *interface*, como pode ser vista na figura 31, é simples, facilitando, assim, a interação do usuário com o programa. O conteúdo foi disposto de forma que qualquer usuário pudesse aprender com ele, independente de ter conhecimento ou não desse contéudo.

O programa está subdivido em:

- *Matrizes – teoria*: essa parte inicia com uma introdução onde é mencionada a importância das matrizes e suas aplicações em diversas áreas. São abordados todos os conceitos básicos da teoria de matrizes, como a notação dos

elementos, representação algébrica da matriz, ordem das matrizes, lei de formação dos elementos, tipos de matrizes, diagonal principal e diagonal secundária, distribuídos pelas 16 páginas que compõem essa parte do programa. A figura 32 mostra como modelo a página 4, a qual apresenta dois exemplos que envolvem ordem da matriz e lei de formação:

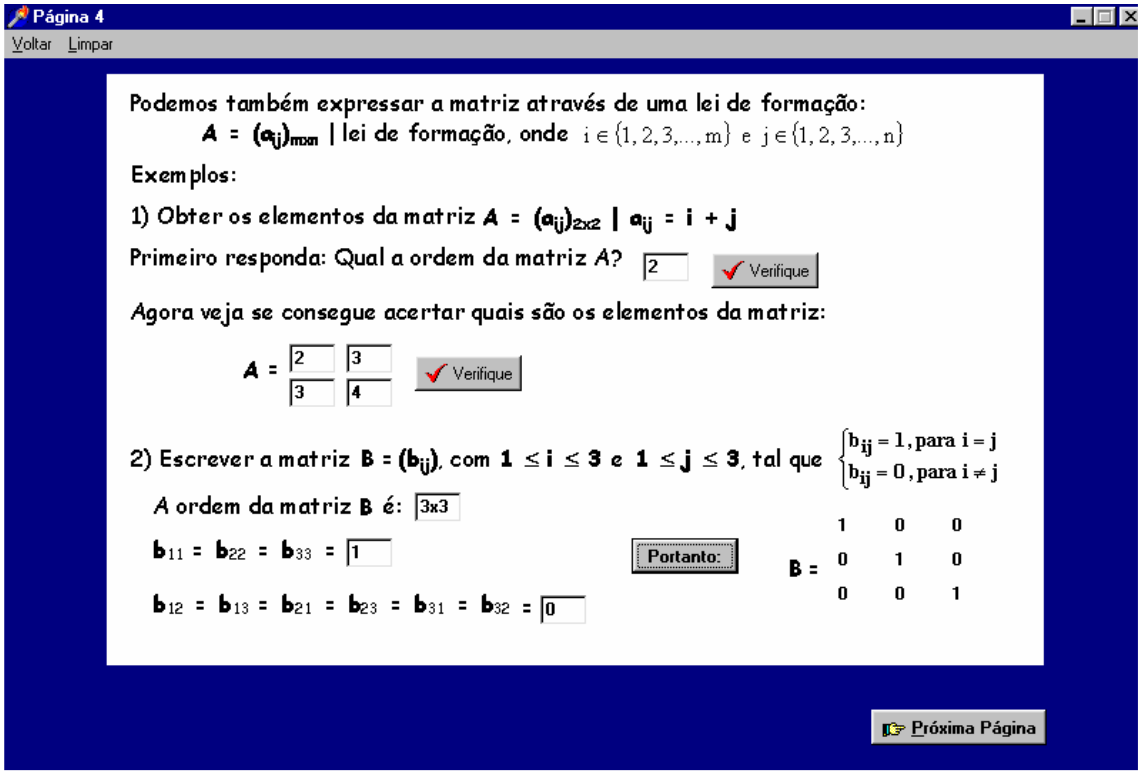


fig. 32 – exemplicando a parte *Matrizes – teoria*

- *Adição/Subtração* (fig. 33): nessa parte é feita a definição usual de cada uma dessas operações.

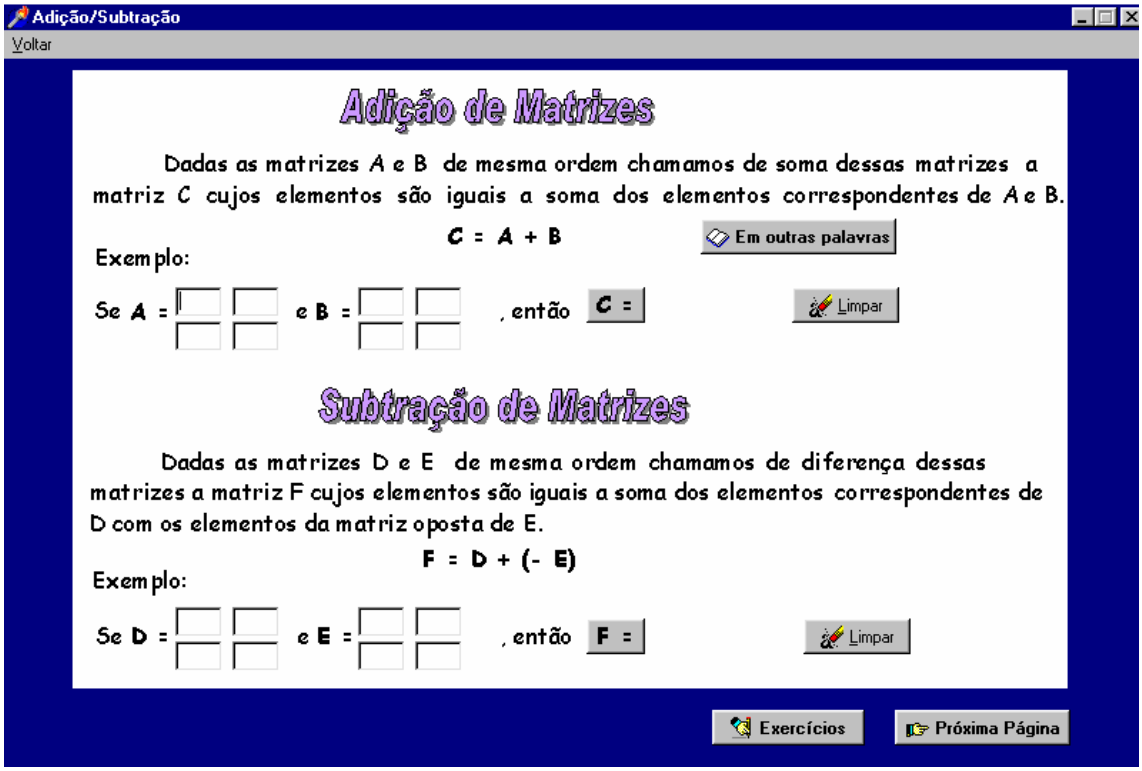


fig. 33 – a parte *Adição/Subtração*

- *Multiplicação* (fig. 34): nessa parte é feita a definição da multiplicação de um número real por uma matriz, e, logo em seguida, a definição da multiplicação de matrizes.

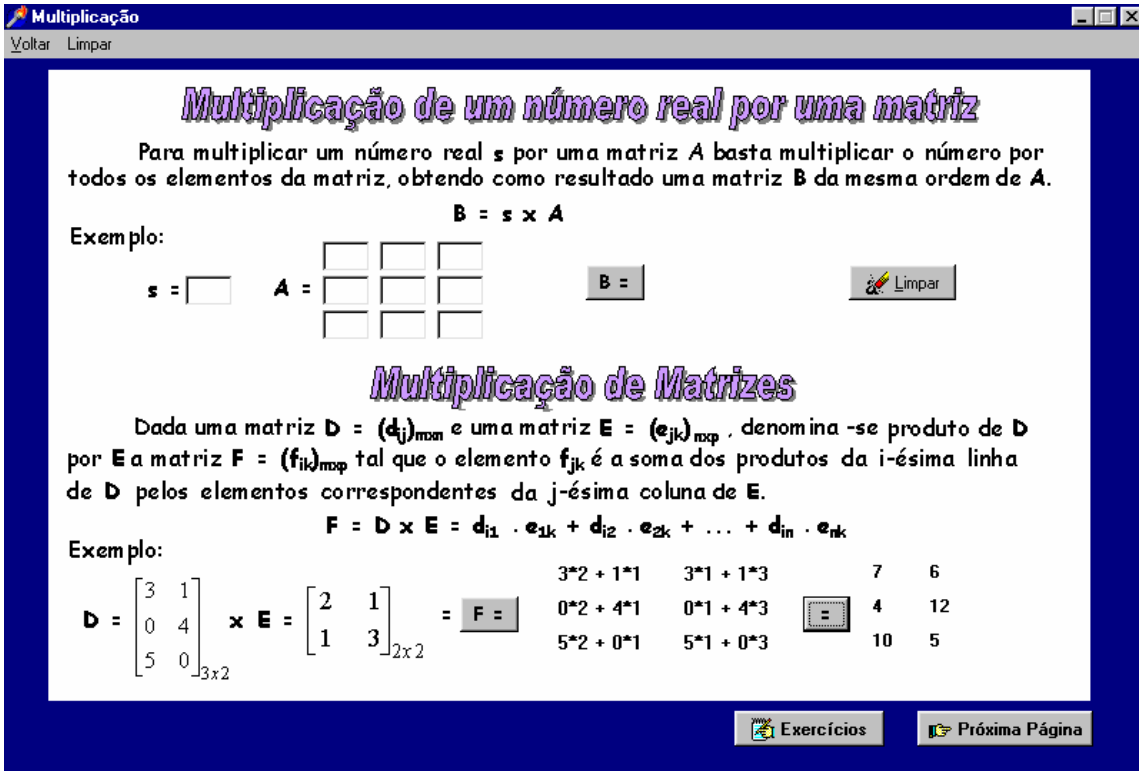


fig. 34 – a parte *Multiplicação*

- *Matriz Inversa* (fig. 35): nessa parte é feita a definição de matriz inversa.

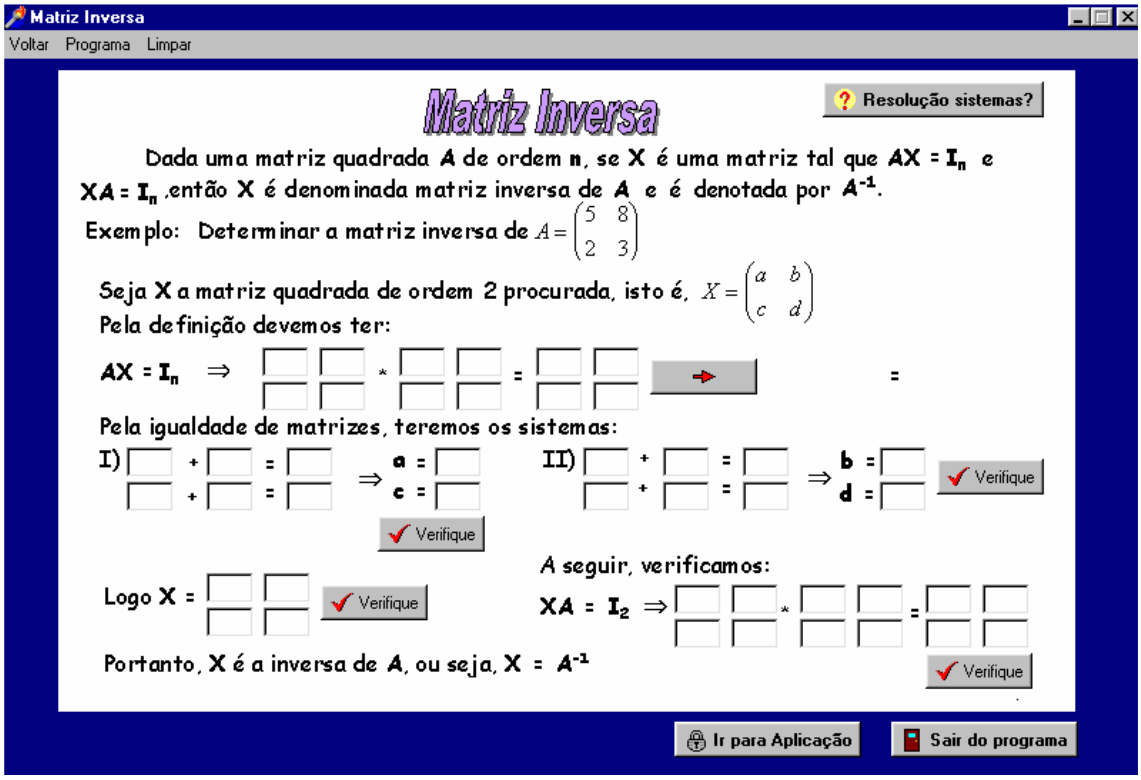


fig. 35 – a parte *Matriz Inversa*

- *Aplicação*: O programa **Criptografando**, mencionado no capítulo anterior, foi incluido nessa parte como um ótimo exemplo de como a teoria de matrizes pode ser utilizada.

Os exemplos que aparecem no decorrer do programa são em sua maioria feitos pelo usuário.

Na figura 36, os exemplos de matriz diagonal e matriz identidade, inclusive a lei de formação dos elementos dessa última, foram deixados a cargo do usuário.

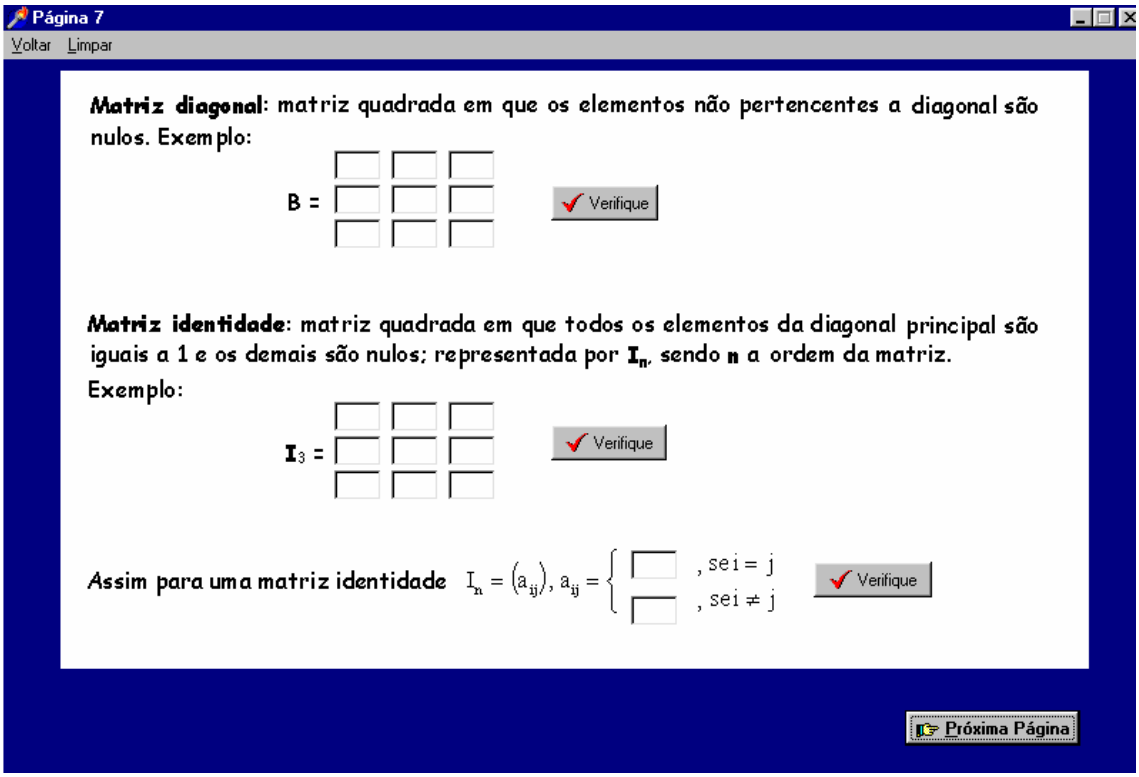


fig. 36 – exemplos

Boa parte dos exercícios foram retirados de livros didáticos. O usuário tem acesso a essas referências (fig. 37) no próprio programa, através do menu *Referências* da página principal (fig. 31).

fig. 37 – referências utilizadas no programa

Os exercícios são diversificados, e o grau de dificuldade é variado.

Na figura 38, tem-se três exercícios de nível fácil, considerados assim por se estar assumindo que o usuário compreenda bem como se dá a lei de formação de elementos das matrizes e a ordem de uma matriz.

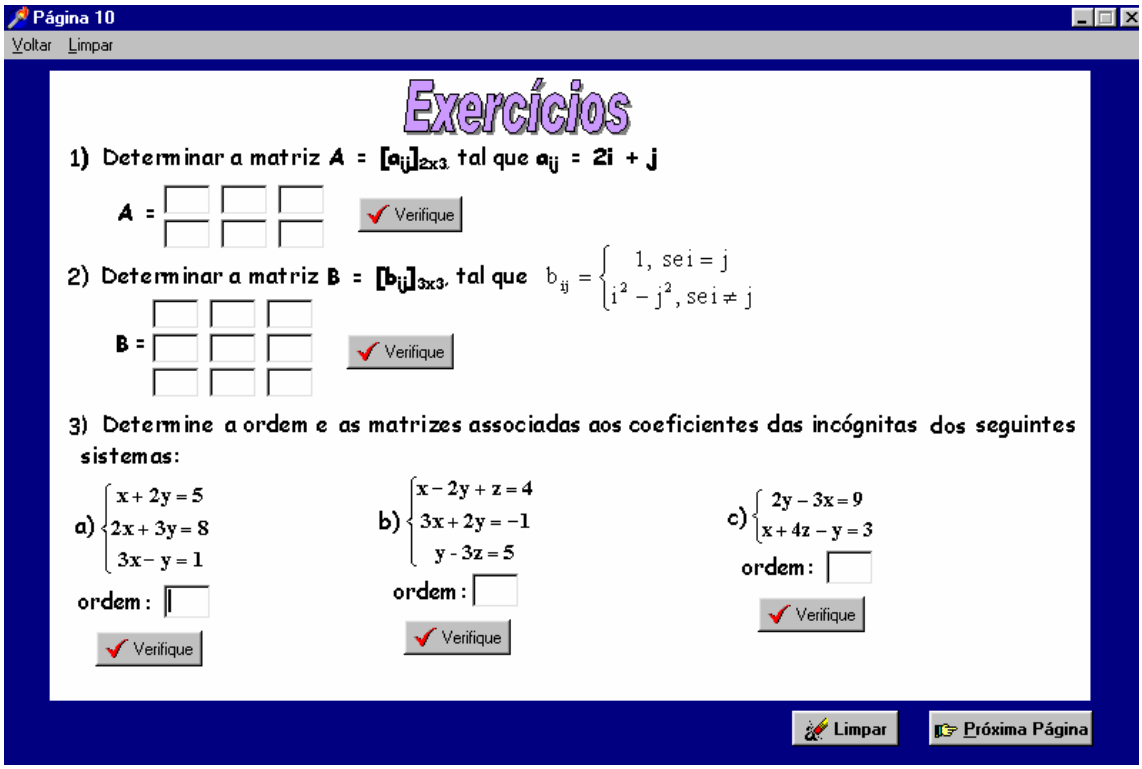


fig. 38 – exercícios fáceis

No exercício 3 (fig. 38) o usuário só consegue ter acesso a matriz das incógnitas se souber de que ordem ela é, e, então, pode escrever os elementos dessa matriz, como mostram as figuras 39 e 40:

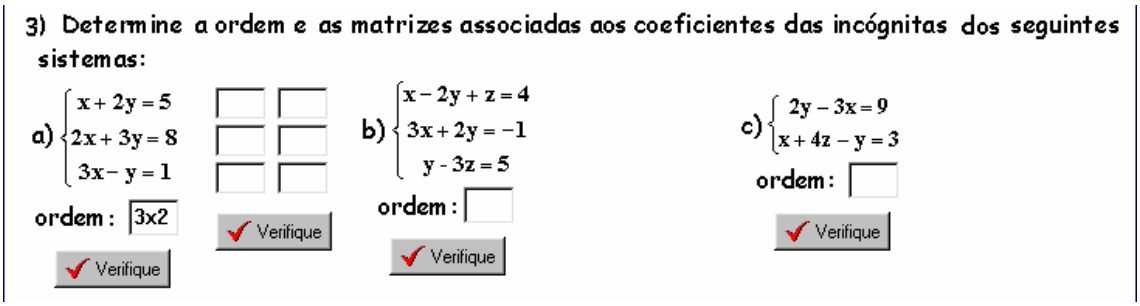


fig. 39 – acertando a ordem

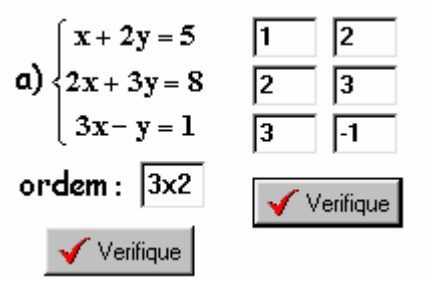


fig. 40 – a matriz das incógnitas

Antes de mostrar exemplos de exercícios de nível médio, é interessante fazer menção de como é feita a verificação de erros e acertos e além disso, falar um pouco das dicas que aparecem durante o programa.

Em todos os exemplos e exercícios o botão 

, quando acionado, emite uma mensagem. Essa mensagem pode ser indicativa de erro, ou indicativa de acerto. Em geral, no programa, quando o usuário comete algum erro aparece a seguinte mensagem:

fig. 41 - erro

Em alguns casos, as mensagens especificam qual o erro que está sendo cometido, como por exemplo nos casos de erro na ordem da matriz, uso de números decimais[27] e espaço em branco deixado pelo usuário:

fig. 41a – erro na ordem

fig. 41b – número decimal

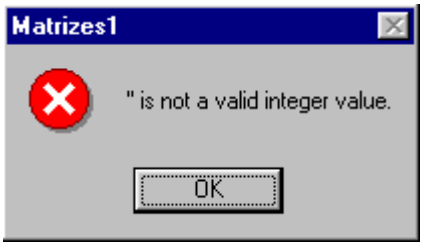


fig. 41c – espaço em branco

Para indicar que o usuário acertou, em geral aparece a seguinte mensagem:

[27] O programa só trabalha com números inteiros, esse fato deu-se devido a alguns problemas ocorridos durante a programação.

fig. 42 – acerto

Alguns exemplos e exercícios exigem que o usuário acerte a ordem da matriz para poder continuar avançando, nesse caso, ao acertar aparece a seguinte mensagem:

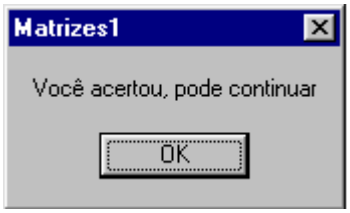


fig. 43 – avançando

Para que o usuário possa entender melhor como determinar a ordem de uma matriz e assim, poder avançar mais tranquilamente nos exemplos e exercícios, na página 3 do programa o botão Ordem 3x3 dá acesso a seguinte janela (fig. 44):

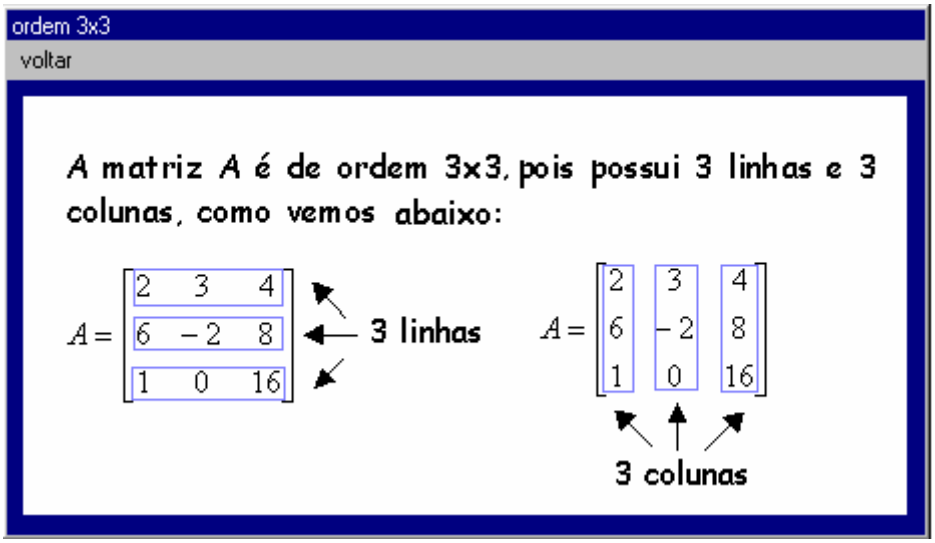


fig. 44 – ordem 3x3

Na página 4 do programa (fig. 32), o exemplo 2 requer que o usuário tenha compreendido não só como determinar a ordem da matriz, mas, também, a lei de formação de seus elementos. No caso em que o usuário tenha errado

a ordem ou qualquer um dos elementos, ao acionar o botão Portanto:, aparece a seguinte mensagem (fig. 45):

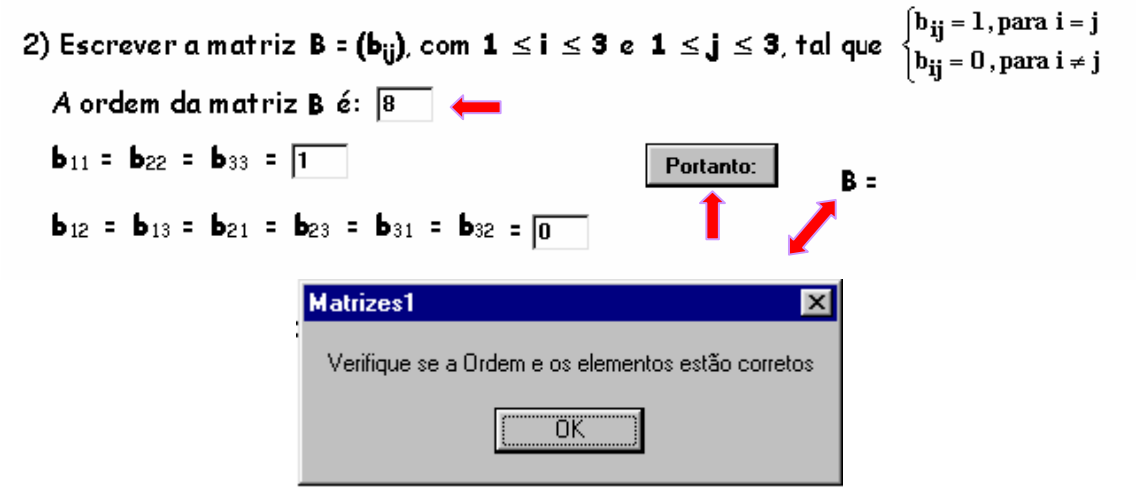


fig. 45 – erro no exemplo 2

Como existe um erro a matriz B não é identificada. Após identificar e corrigir o erro, acionando o botão aparecem os elementos da matriz B (fig. 46).

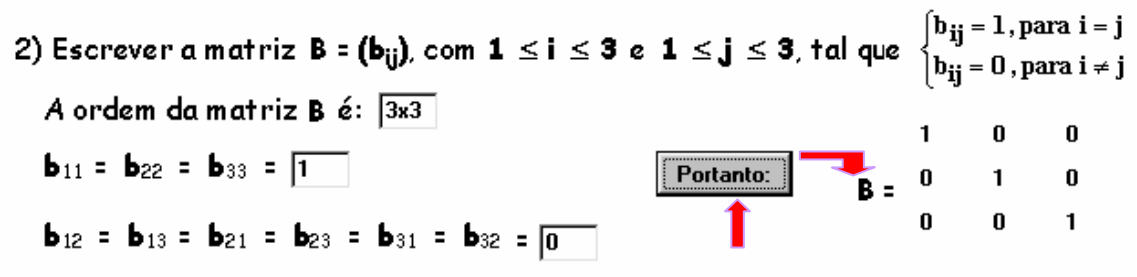


fig. 46 – elementos da matriz B

Na página 6 do programa, para fixar bem quais são os elementos que fazem parte da diagonal principal, o usuário tem a opção de acionar o botão I = J que dá acesso a janela abaixo (fig. 47):

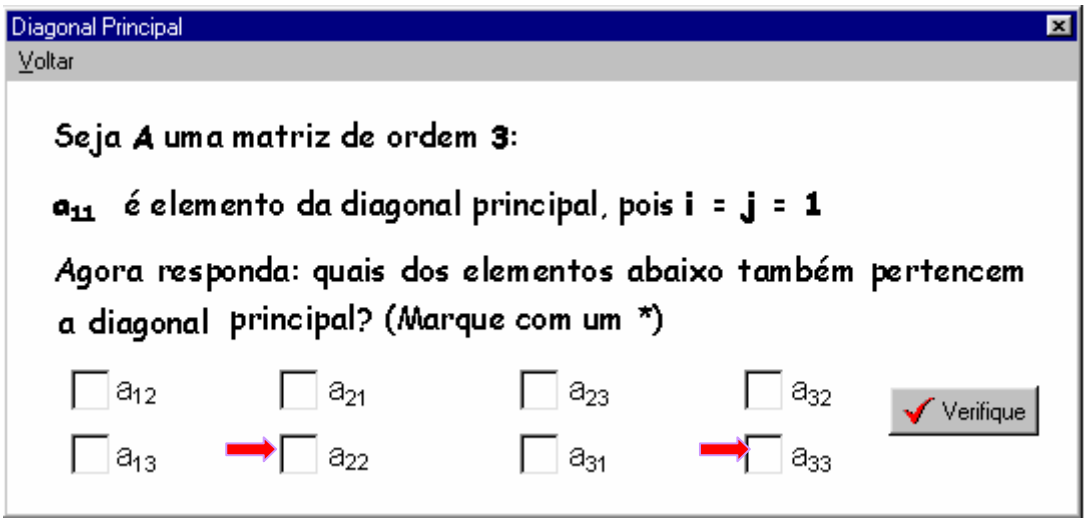


fig. 47 – diagonal principal

O usuário precisa apenas indicar com um asterisco (*) quais são os outros dois elementos que pertencem a diagonal principal. Para a diagonal secundária também foi feita essa opção.

Na página 7 do programa (fig. 36), se o usuário cometer algum erro aparecem as seguintes mensagens:

- quando erro cometido for na matriz diagonal:

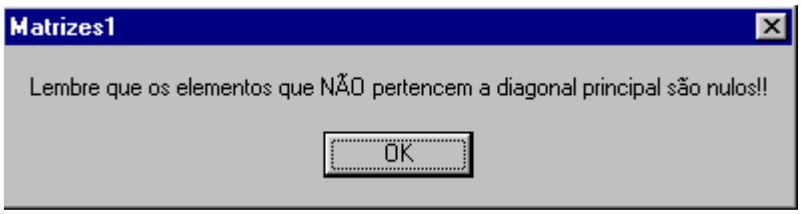


fig. 48 – erro na matriz diagonal

- quando erro cometido for na matriz identidade:

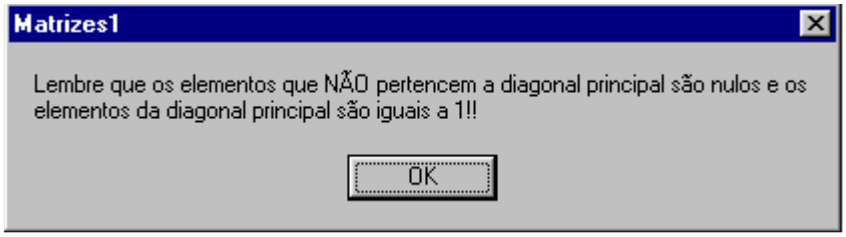


fig. 49 – erro na matriz identidade

Após ler essas mensagens o usuário consegue identificar onde errou.

Na página 8 do programa estão definidas as matrizes oposta e transposta. Para cada uma delas, o usuário escolhe elementos para a matriz e

em seguida escreve a matriz oposta (ou a transposta) correspondente. Caso algum elemento não esteja correto, ao fazer a verificação aparece a seguinte mensagem:

- para a matriz oposta:

fig. 50 – erro na matriz oposta

- para a matriz transposta:

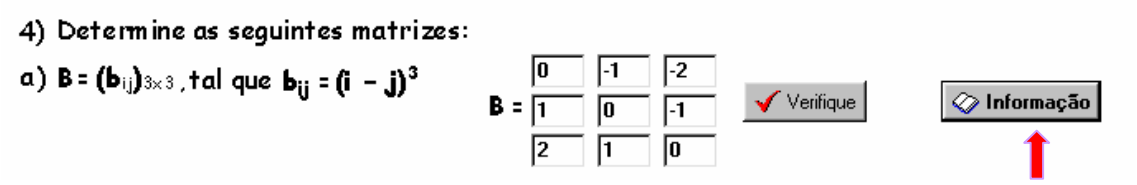
fig. 51 – erro na matriz transposta

As duas mensagens pedem apenas para verificar as matrizes oposta e transposta, pois o erro não pode estar na matriz escolhida pelo usuário.

O exercício 4, letra a) da página11 do programa, explora um conceito que não foi abordado nas páginas anteriores do mesmo. Achou-se interessante, então, fornecer essa informação aos usuários que tiverem interesse em aprender um pouco mais.

Acionando o botão 

o usuário encontra a definição de matriz anti-simétrica (fig. 52).



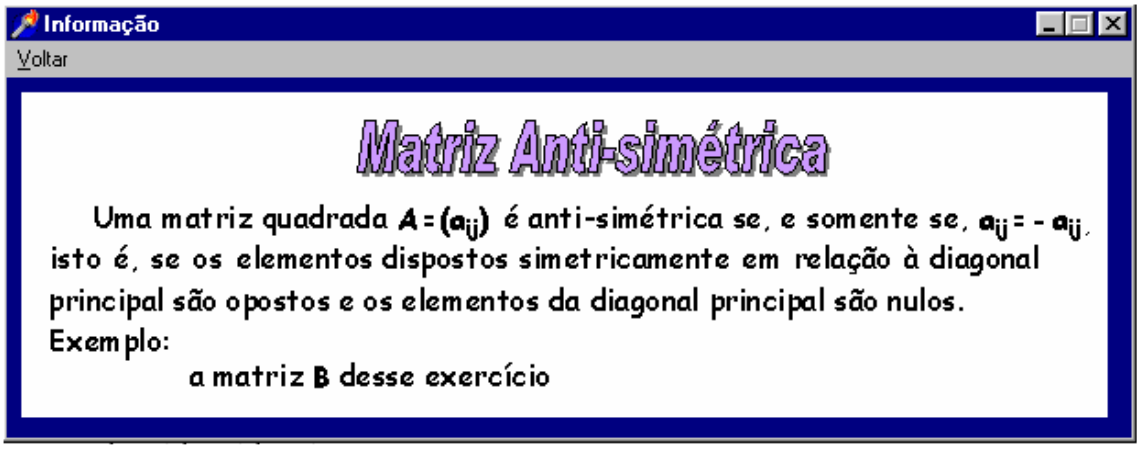


fig. 52 – matriz anti-simétrica

Na página 13 do programa, o exercício 9 deve ser resolvido de acordo com o modelo (fig. 53). Isso significa que, para responder as questões, o usuário deve utilizar apenas letras minúsculas. Por se tratar de programação, tornou-se inviável possibilitar o uso de letras maiúsculas, ou qualquer outra forma de escrever a frase. Caso o usuário responda de uma forma diferente da que foi estabelecida no programa, na verificação é mostrada a maneira que deve ser escrita, não significando, necessariamente, que o usuário tenha errado. Por exemplo, na figura 54, o erro cometido foi apenas na introdução da letra *m* maiúscula:

fig. 53 – siga o modelo

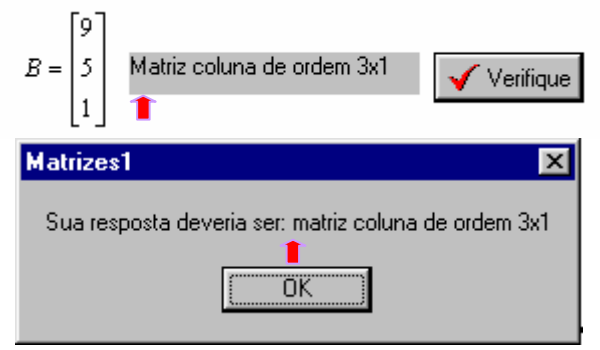


fig. 54 – como escrever a frase

Na página 14 do programa (fig. 55) encontram-se os exercícios considerados de nível médio, um deles envolvendo o conceito de logaritmo e alguns deles sistemas lineares.

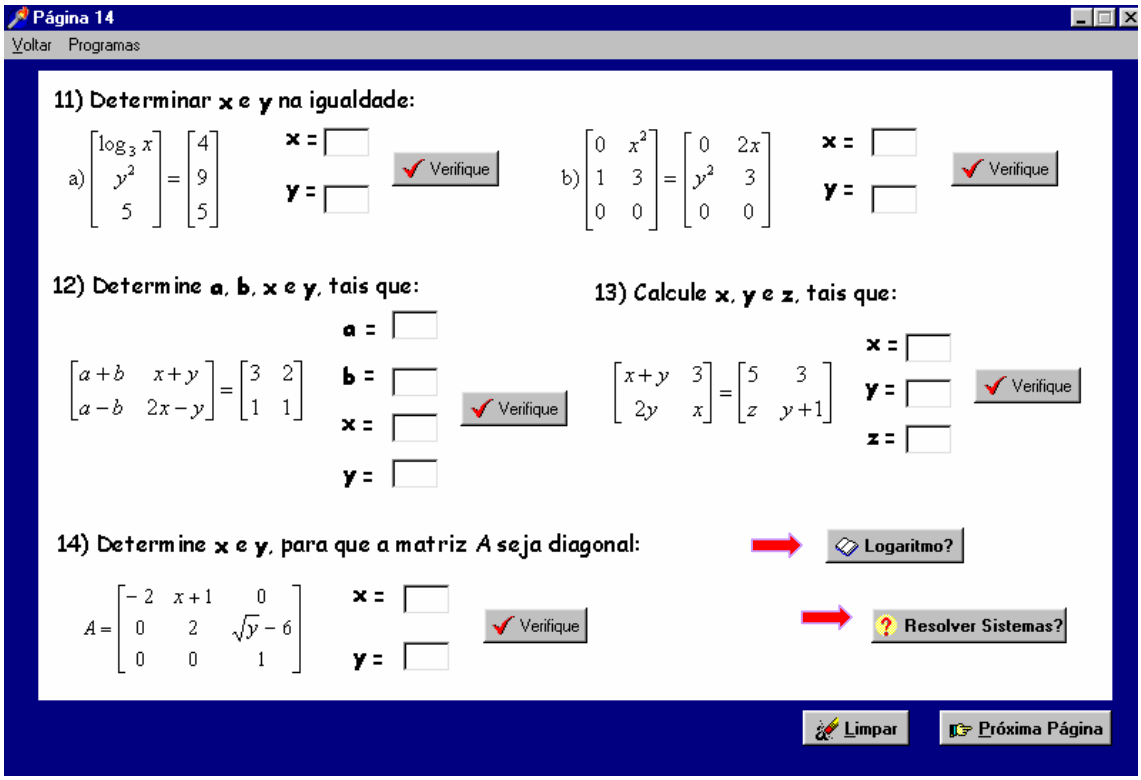


fig. 55 – exercícios de nível médio

Considerando o fato de que alguns usuários não têm conhecimento e outros possam ter esquecido da definição de logaritmo, nessa mesma página (fig.55) o botão Logaritmo? dá acesso a essa informação (fig. 56):

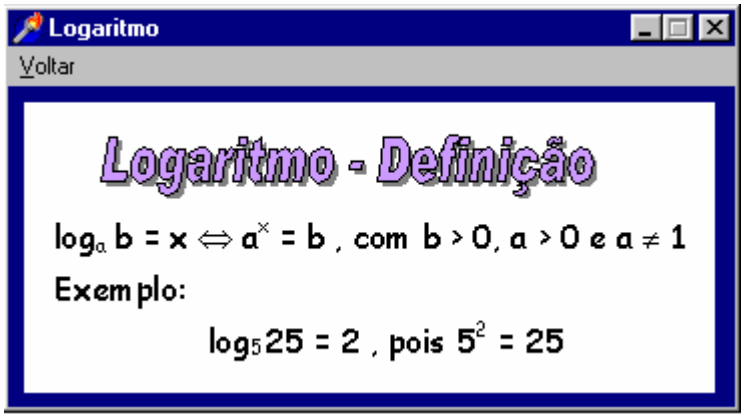


fig. 56 – definição de logaritmo

Ainda na página 14 do programa (fig. 55), o botão Resolver Sistemas? fornece ao usuário duas opções (fig. 57):

fig. 57 – ajuda

- *Sistema 2x2*: o usuário deve escolher essa opção, no caso em que o sistema obtido no exercício seja de ordem 2. Ao fazer essa opção, o programa **Solução de Sistemas Lineares**[28] (fig. 58) é executado, e então, o usuário pode utilizá-lo para resolver o sistema.

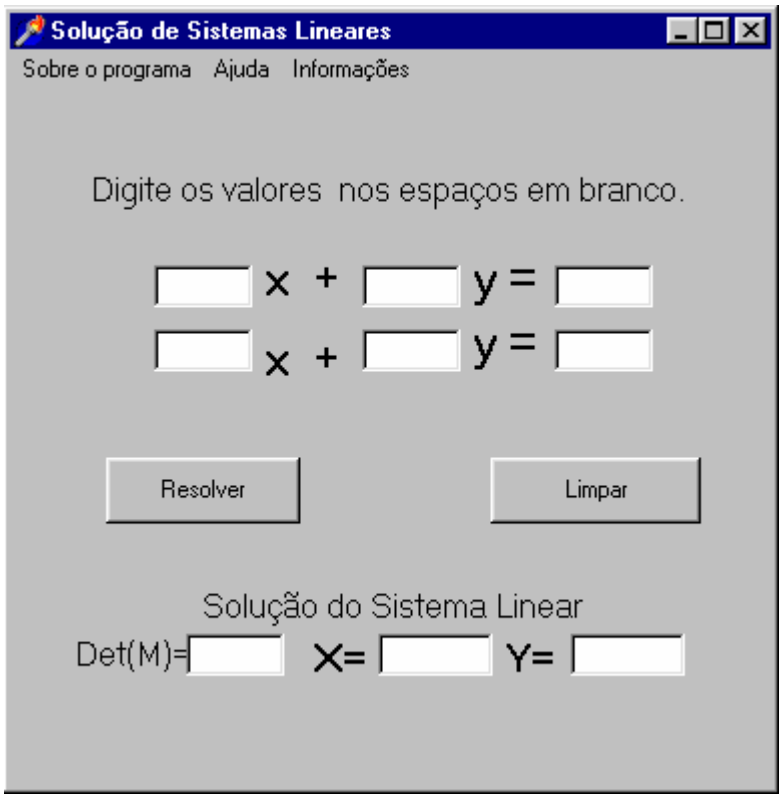


fig. 58 – Sistema 2x2

- *Sistema 3x3*: o usuário deve escolher essa opção, no caso em que o sistema obtido no exercício seja de ordem 3. Ao fazer essa opção, o programa **Solução de Sistemas Lineares 3x3** (fig. 59) é executado, e então, o usuário pode utilizá-lo para resolver o sistema.

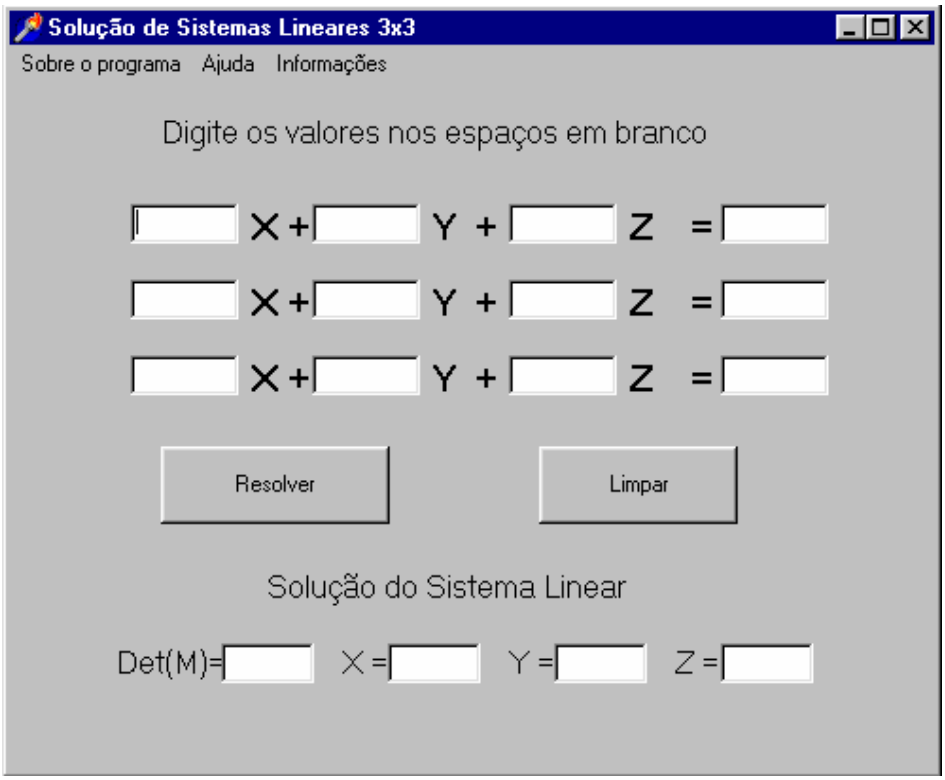


fig. 59 – Sistema 3x3

Observando os exercícios 15 e 17 da página 15 do programa (fig. 60), sobre matrizes transpostas, o usuário pode chegar a conclusão de que a transposta da transposta de uma matriz é a própria matriz ($(A^t)^t = A$).

[28] Os programas de resolução de Sistemas Lineares (fig. 58 e 59) foram desenvolvidos por Alex Deni Alves, integrante do Projeto E.I.C.A.L.G.A. e também foram apresentados nas atividades. A parte *Det(M)* não é utilizada no programa *Matrizes: teoria e aplicação*.

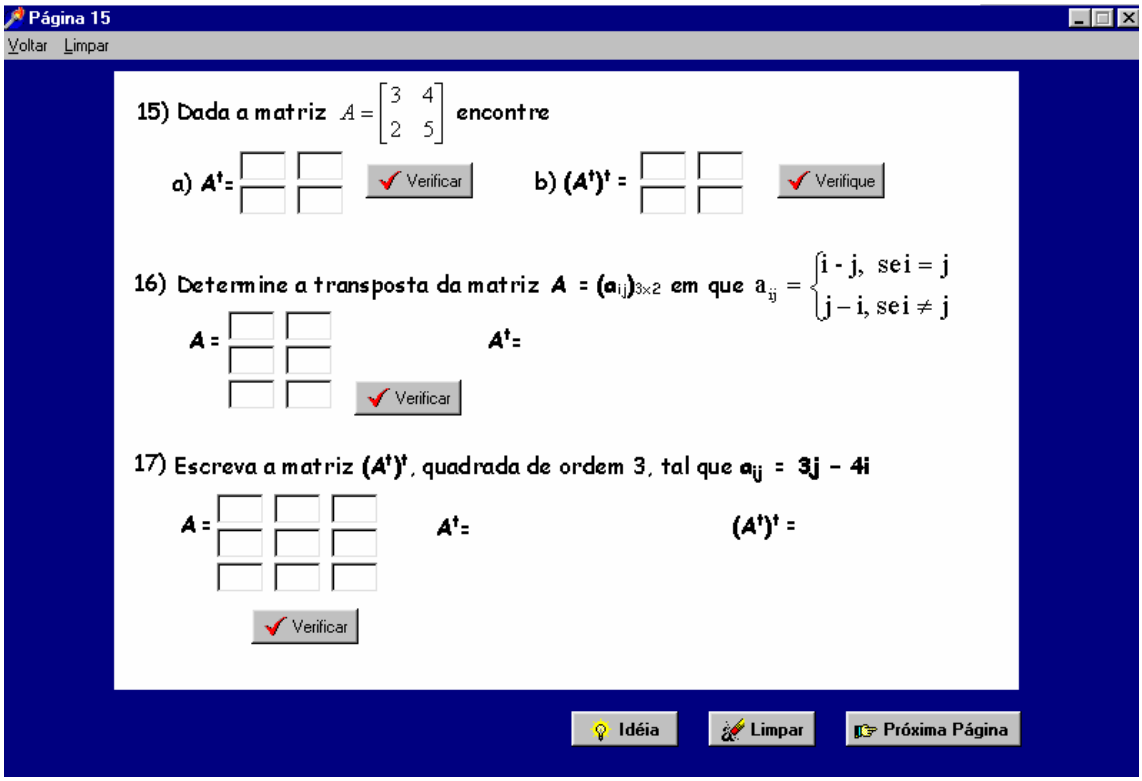


fig. 60 – página 15 do programa

Para não deixar que esse fato não fosse observado, nessa mesma página, o botão Idéia ajuda o usuário a chegar a essa conclusão (fig. 61):

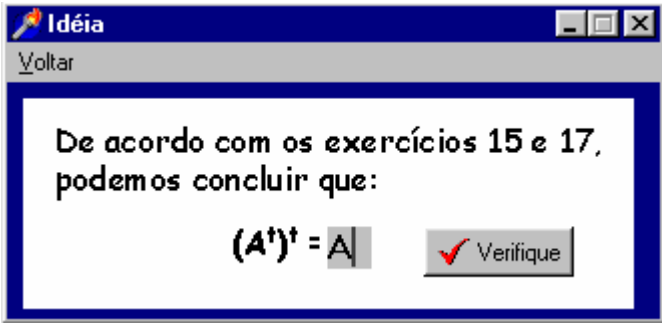


fig. 61 - conclusão

Em grande parte dos exemplos e exercícios, como pode se observar no programa, as matrizes utilizadas são de ordem inferior a 4. Para diversificar um pouco, na página 16, o exercício 18 utiliza uma matriz de ordem 4 (fig. 62):

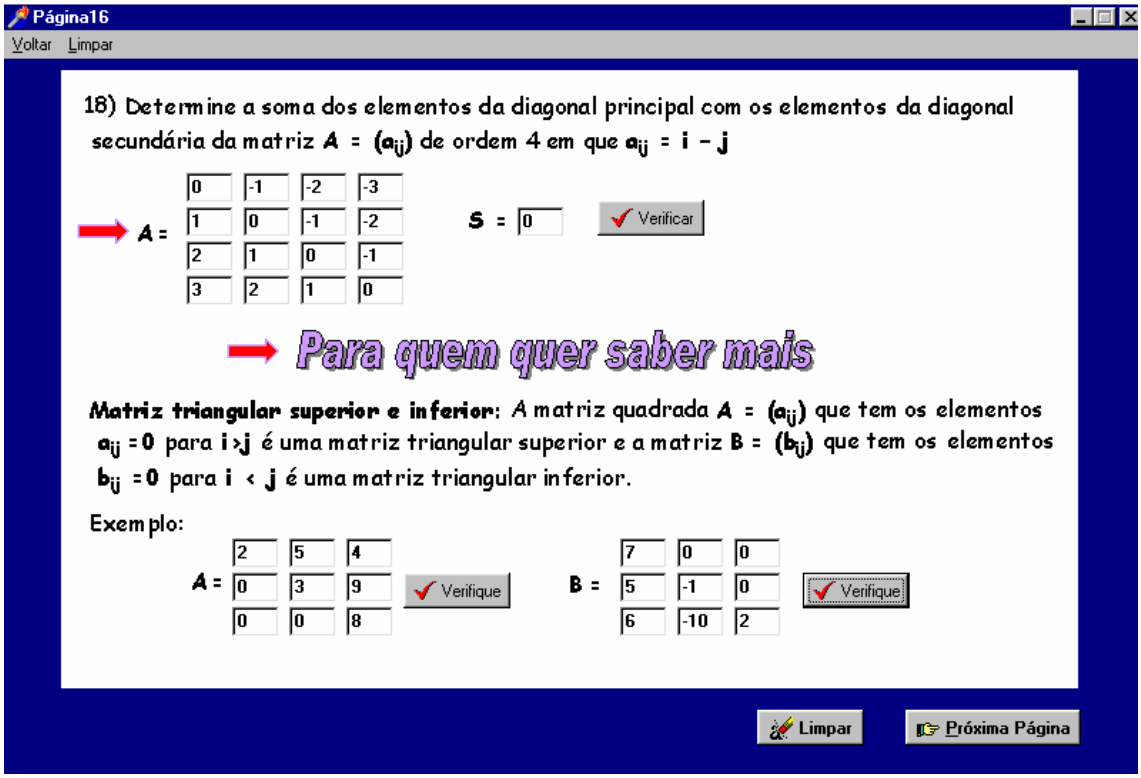


fig. 62 – matriz de ordem 4

Ainda nessa página (fig. 62), "*Para quem quer saber mais*" apresenta o conceito de matriz triangular superior e matriz triangular inferior, que não é abordado, em geral, no Ensino Médio. O objetivo de se adicionar esse conceito é ampliar o conhecimento do usuário, não sendo indispensável a sua leitura e resolução.

Em *Adição/Subtração* (fig. 33), o conceito de adição de matrizes pode ser visto, além da forma que se encontra na própria página, de uma maneira um pouco mais formal. Para visualizar essa outra maneira, basta acionar o botão Em outras palavras que abre a seguinte janela (fig. 63):

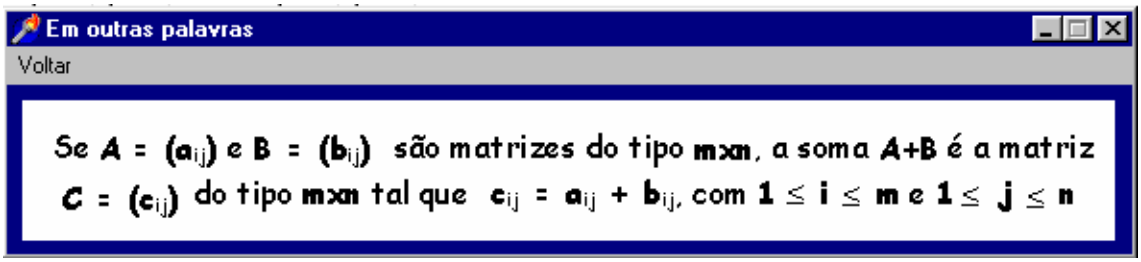


fig. 63 – o conceito formal de adição de matrizes

A idéia é fazer com que o usuário tenha acesso também a linguagem utilizada por grande parte dos livros.

Na página *Exercícios (cont.)*, o exercício 2 explora o conceito de determinantes. Após resolver a questão, o usuário pode acionar o botão Para saber mais e verificar que o que acabou de fazer foi exatamente calcular o determinante da matriz (fig. 64).

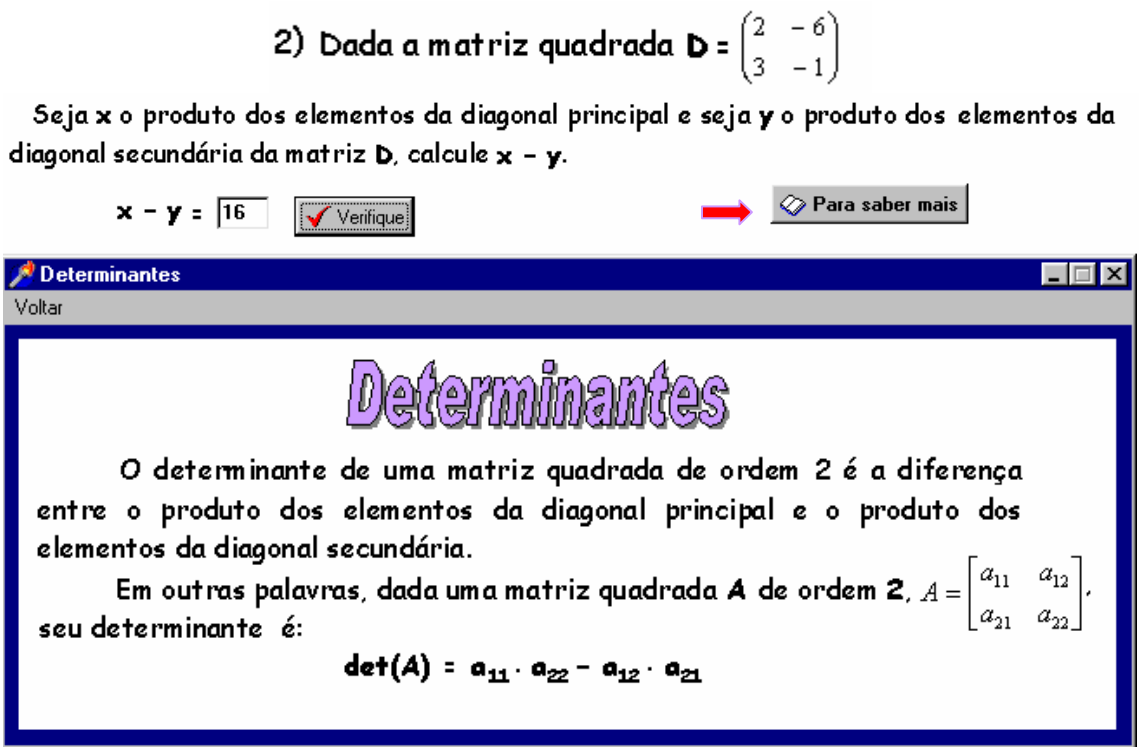


fig. 64 – determinante

Em *Exercícios (continuação*)* (fig. 65), o usuário precisa acertar a ordem da matriz produto e responder corretamente a pergunta para poder avançar no exercício 4.

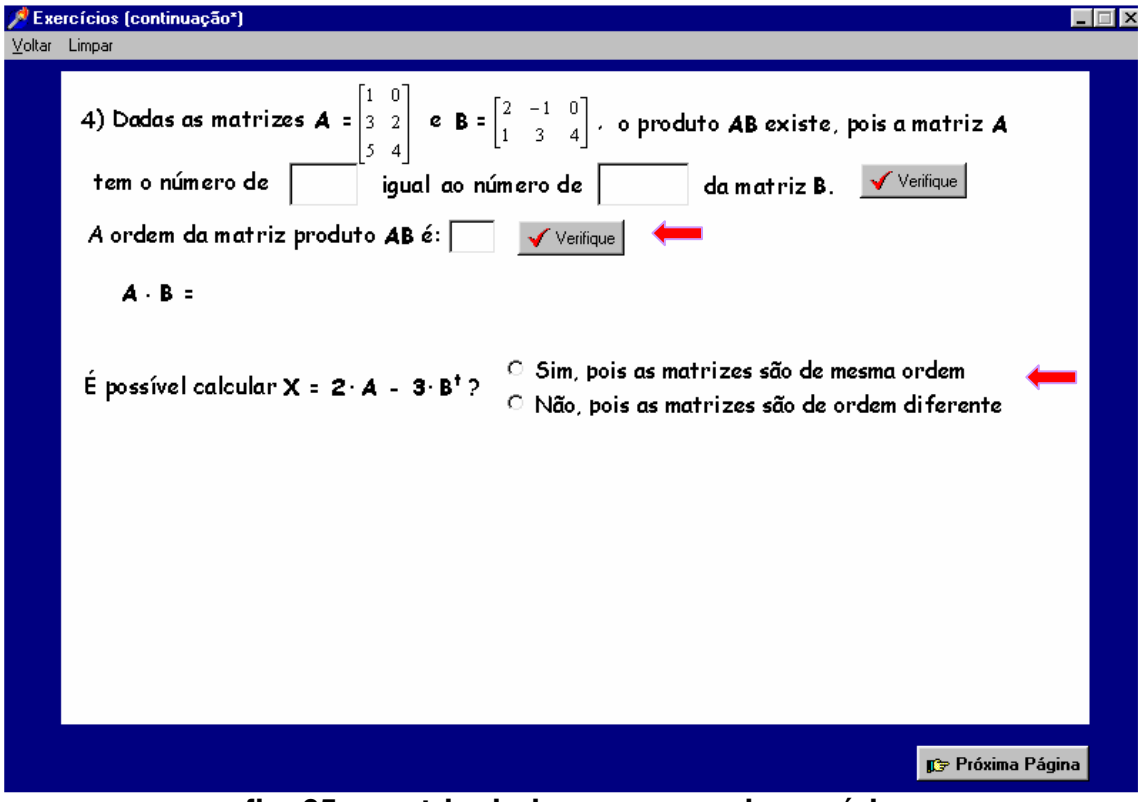


fig. 65 – restringindo os passos do usuário

Se o usuário escolher a opção "*Não, pois as matrizes são de ordem diferente*", o programa emite a seguinte mensagem (fig. 66):

fig. 66 – mensagem para opção "Não"

A mensagem solicita que o usuário verifique as ordens das matrizes, não informando quais são elas. O usuário deve identificar na questão quais são essas matrizes e a ordem delas.

Escolhendo a opção "*Sim, pois as matrizes são de mesma ordem*" o usuário pode continuar resolvendo a questão (fig. 67).

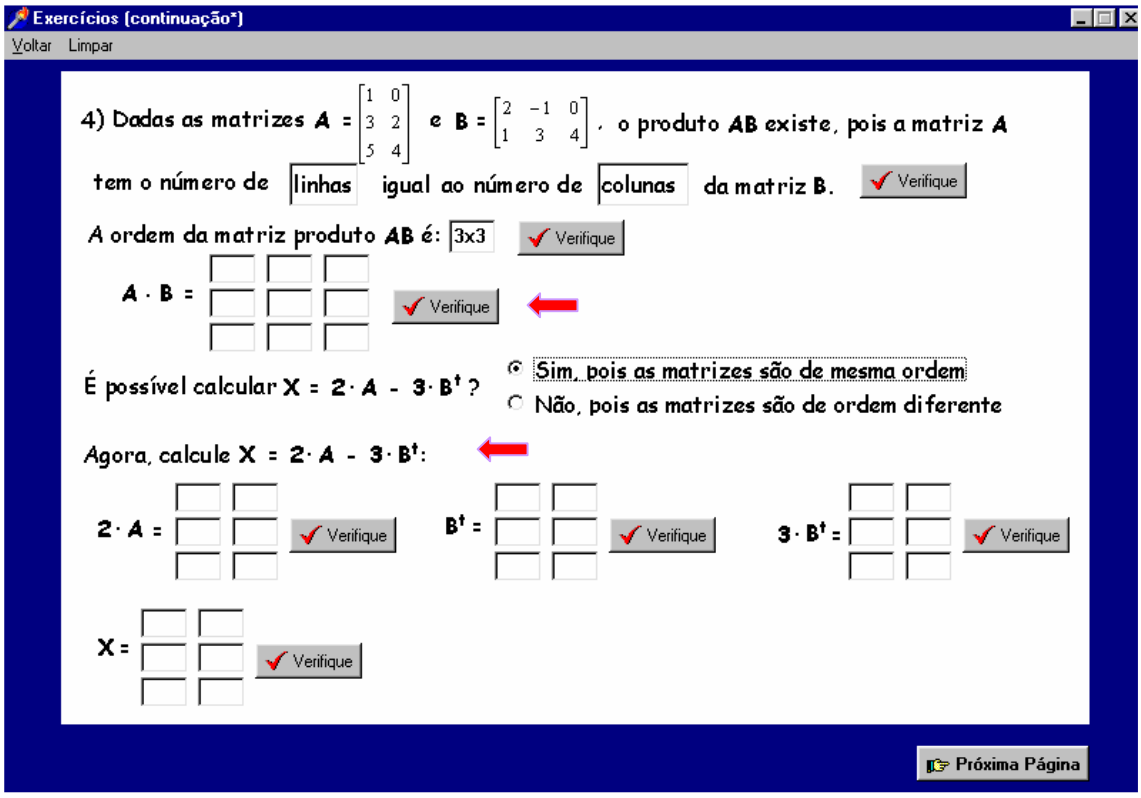


fig. 67 – habilitando a questão

É importante mencionar que várias páginas do programa foram omitidas durante essa apresentação. O motivo para fazer essa omição é que o programa é auto-explicativo, ou seja, as informações necessárias para utilizá-lo estão presentes em sua própria estrutura.

Após o desenvolvimento de um programa como esse, fez-se necessária uma avaliação, por parte dos usuários, principalmente aqueles aos quais o programa foi direcionado, nesse caso, alunos do Ensino Médio, para saber o resultado que esse trabalho pode proporcionar ao processo de ensino/aprendizagem. Como foi feita essa avaliação será comentado a seguir.

5.1. Testando o Programa

Para verificar a validade do programa e a existência de detalhes que precisassem ser modificados, foi feito um teste do programa **Matrizes: teoria e aplicação**. Esse teste foi realizado por alunos do Colégio Autonomia e por outros dois usuários[29]. Após a execução do programa, os alunos faziam a sua

[29] sendo um deles estudante da UFSC, cujo curso não tem relação imediata com a Matemática, e o outro que não está estudando no momento.

avaliação através de críticas ou opinões deixadas numa folha de comentários. As opiniões (críticas) feitas pelos usuários estão anexadas ao final do trabalho. Além desse teste, foi feita uma apresentação rápida do programa para alunos da Graduação e alguns professores da UFSC.

5.1.1. Análise das opiniões dos usuários

Quanto a validade, em geral, o programa foi considerado útil, interessante e criativo. De acordo com as opiniões, poderia ser utilizado nas escolas, tanto como um reforço escolar como até mesmo para avaliações. O fato do programa fazer a verificação de imediato dos erros e/ou acertos passou confiança aos usuários. Um outro ponto mencionado, considerado importante, foi o estimulo ao cálculo mental. Para eles, o programa apresenta uma abordagem diferente, mais dinâmica do conteúdo de Matemática.

Os usuários identificaram alguns detalhes que poderiam ser modificados para melhorar o programa, entre elas estão:

- quanto a ordem dos elementos da diagonal secundária: o programa só aceitava uma ordem pré-determinada, diferente da que é utilizada nas escolas;

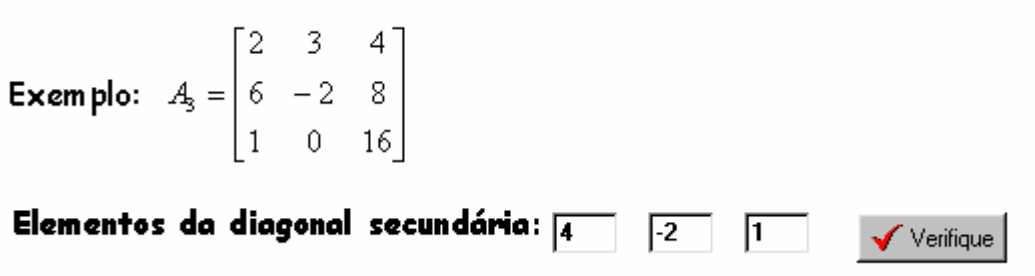


fig. 68 - ordem pré-determinada pelo programa

fig. 69 – ordem utilizada nas escolas

- quanto ao símbolo de multiplicação: preferência por “x” no lugar de “·”;
- especificação da ordem das incógnitas nos sistemas para saber como escrever a matriz: o que vem primeiro x, y ou z?

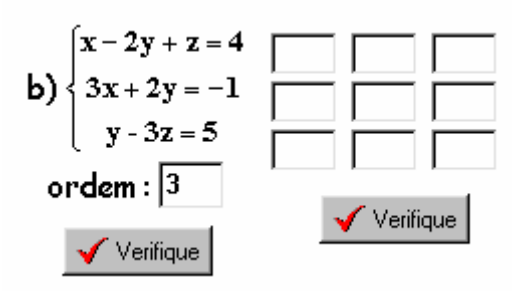


fig. 70 – ordem da incógnitas

- especificação da matriz que seria utilizada em *Matriz Inversa* (fig. 71) (a seta vermelha indica onde foi feita a alteração):

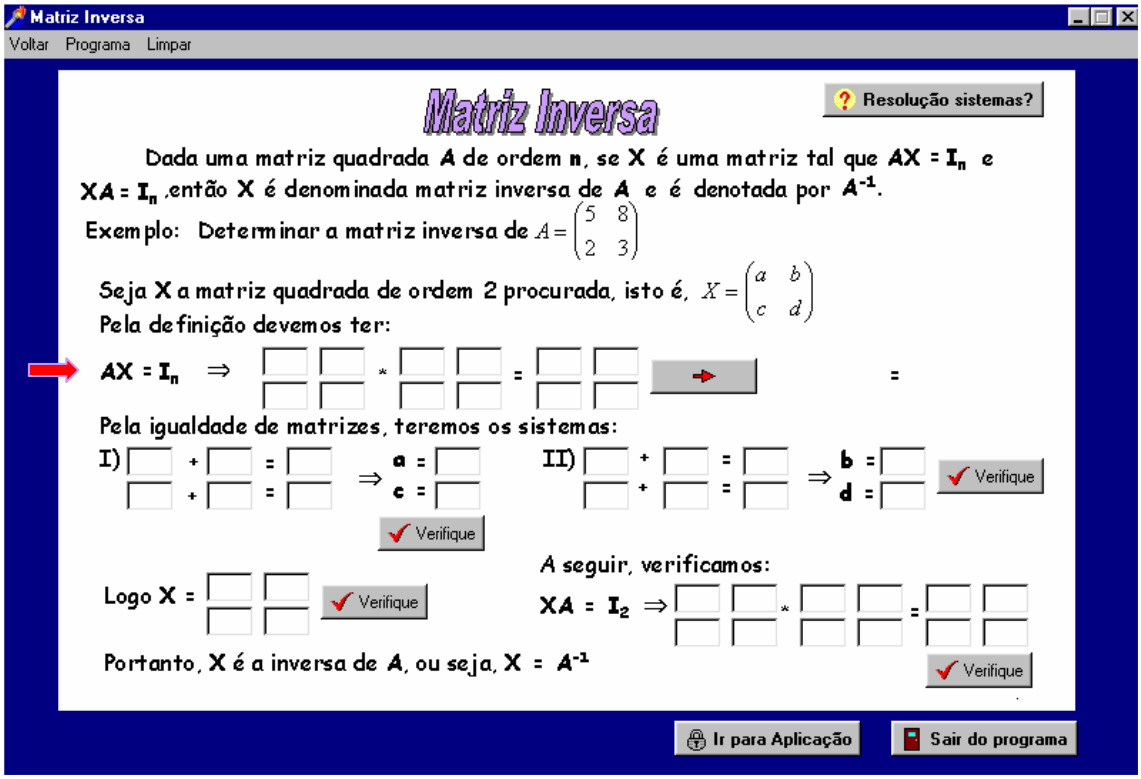


fig. 71 – especificando as matrizes

- representação da ordem da matriz quadrada: por exemplo “2” e “2x2”, o programa estava aceitando apenas uma delas;

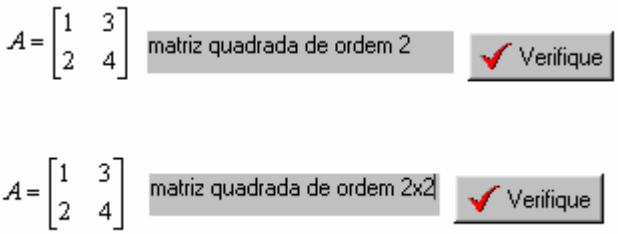


fig. 72 – ordem da matriz quadrada

Algumas alterações foram feitas de acordo com o que foi indicado pelos usuários. No caso do símbolo "·" e da ordem das incógnitas. O símbolo "·" impede que em alguns casos haja confusão, por exemplo, se na figura 61 fosse utilizado o símbolo "x" a notação ficaria um pouco confusa.

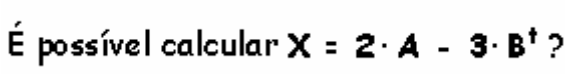


fig. 73 – o símbolo " · "

Quanto a ordem das incógnitas, um dos propósitos do programa é estimular o raciocínio lógico-dedutivo. Fornecer a ordem em que aparecem as incógnitas não serviria para esse propósito.

Ainda com relação ao programa, foi feito um comentário sobre o fato de não haver limitação para passar de uma página para a próxima, ou seja, o programa não impede que o usuário passe para outra parte sem ter respondido a anterior. Esse comentário foi feito em caráter de admiração, ou seja, o usuário ficou surpreso por não haver esse tipo de limitação, mas não foi considerado como um fator ruim na apresentação do programa.

Respondendo a esse comentário, o motivo de não haver limitação no programa, exceto em algumas partes que já foram mencionadas, onde a ordem da matriz influência nos exercícios, é deixar o usuário livre na decisão do que quer ou não quer fazer. Livre para escolher o que acha necessário aprender ou não. O usuário que não tiver conhecimento, saberá da necessidade de resolver e entender a teoria anterior ao sentir dificuldade em algum momento. O usuário que possui um bom conhecimento do conteúdo, pode optar por fazer somente os exemplos e exercícios que julgar mais interessante, aqueles que acrescentarão algo ao seu aprendizado.

CONSIDERAÇÕES FINAIS

A tecnologia tem uma enorme importância para educação. Utilizar recursos computacionais como auxiliadores no processo de ensino/aprendizagem, como os programas apresentados nesse trabalho, facilitam e tornam muitas vezes mais agradáveis o aprendizado de alguns conteúdos como matrizes, por exemplo.

Saber onde pode ser aplicado o contéudo que se está aprendendo é fundamental para despertar o interesse dos alunos.

Tentou-se reunir tudo isso em um único programa: ***Matrizes: teoria e aplicação***, desenvolvido para auxiliar no processo de ensino/aprendizagem do conteúdo matrizes e mostrar uma de suas aplicações através da Criptografia, uma forma interessante e ao mesmo tempo divertida de se trabalhar com matrizes.

Isso é um exemplo de como a tecnologia está proporcionando à sociedade educacional uma grande oportunidade de expandir todo o seu conhecimento. Oportunidade de criar novas maneiras de ensinar e com isso novas manerias de aprender. Por que não aproveitá-las?

Talvez o receio de enfrentar coisas novas, de mudar comportamentos tradicionais, faz com que muitos ainda deixem-na escapar. No entanto, muitos outros já aproveitaram e continuam utilizando os recursos que as tecnologias estão oferecendo para a Educação.

Participar de um grupo que quis avançar nessa “viagem” e conhecer pessoas que apoiam esse desenvolvimento foi muito importante.

Além disso, o contato com maneiras novas de ensinar e aprender e conseguir enxergar nos alunos a felicidade de estar aprendendo de uma maneira diferente o que costuma ser passado a eles de forma tradicional, estimulam ainda mais a vontade de estar crecendo em atitude e conhecimento.

O Projeto Explorando a Interdisciplinaridade dos Conteúdos de Álgebra Linear e Geometria Analítica funcionou assim, e abriu caminhos para que se possa dar continuidade a esse trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

ALVES, Alex Deni. *A utilização de recursos computacionais para a elaboração de atividades de matematica.* 2004. Trabalho de Conclusão de Curso (Curso de Graduação em Licenciatura Matemática),Universidade Federal de Santa Catarina, Florianópolis, 2004

ALVES, William Pereira. *Curso Prático de Delphi 3.0.* Editora Érica Ltda., São Paulo, 1997, p. 3 – 11.

ANTON, H; RORRES, I. *Álgebra Linear com Aplicações.* 8ª Edição, Artmed Editora, Porto Alegre, 2001.

CANTU, Marco. *Dominando o Delphi 3*. Makron Books, São Paulo, 1997.

CARNEIRO, Diogenes Lemos. *Um estudo sobre a aplicabilidade de redes neurais em criptografia*. Florianópolis, 2001. 74 f. Dissertação (Mestrado) - Universidade Federal de Santa Catarina.

COUTINHO, S. C. *Números inteiros e criptografia RSA*, Série de Computação e Matemática, IMPA, Rio de Janeiro, 1997.

DOMINGUES, Hygino H. *Origem dos Sistemas Lineares e Determinantes*. Disponível em <http://www.somatematica.com.br/historia/sistemas.php> Acesso em 18 nov. 2004

EDDINGS, Joshua. *Como Funciona a Internet.* Editora Quark. 3ª Edição.

FELDMAN, Mark. This article is part of *The Win95 Game Programmer's Encyclopedia.* Disponível em <http://www.geocities.com/SiliconValley/2151/matrices.html> Acesso em: 18 nov 2004

FERNANDEZ, Vicente Paz; YOUSSEF, Antonio Nicolau. *Matemática para 2º grau.*2ª edição, Editora Scipione, São Paulo, 1991.

GENTIL, Nelson; SANTOS, Carlos Alberto Marcondes dos; GRECO, Antônio Carlos; FILHO, Antônio Bellotto; GRECO, Sérgio Emílio. *Matemática para o 2º grau*. Vol. 2, 8ª edição. Editora Ática, São Paulo, 1996.

GIOVANNI, José Ruy; BONJORNO, José Roberto. *Matemática 2º grau.* Vol. 2, Editora FTD S.A., São Paulo.

GIOVANNI, José Ruy; DANTE, Luiz Roberto. *Matemática – teoria, exercícios e aplicações*. 2º grau, Vol. 2, Editora FTD S.A., São Paulo.

GOMES, Alex Sandro; CASTRO FILHO, José Aires; GITIRANA, Verônica; SPINILLO, Alina; ALVES, Mirella; MELO, Milena; XIMENES, Julie. *Avaliação de software educativo para o ensino de matemática.* WIE'2002, Florianópolis (SC).

JÚNIOR, Waldyr Dias Benits. *Sistemas criptográficos baseados em identidades pessoais*. Disponível em <http://www.ime.usp.br/dcc/posgrad/teses/benits.pdf> Acesso em: 08 nov. 2004.

KALINKE, Marco Aurélio. *Matrizes.* Disponível em <http://www.expoente.com.br/professores/kalinke/estudo/matrizes.htm> Acesso em: 08 nov. 2004.

LACERDA, Avâner Conceição de. *A história da tecnologia na educação: Do quadro de giz à realidade virtual*. Dissertação de Mestrado. Florianópolis, 2001.

LAGARTO, Maria João. *Nove Capítulos da Arte Matemática.* Jiuzhang suànshù *ou* Chu Chang Suan Shu. Disponível em <http://www.malhatlantica.pt/mathis/China/china1.htm> Acesso em: 18 nov. 2004

LEMOS, M. "*Criptografia, números primos e algoritmos*", 17º Colóquio Brasileiro de Matemática, IMPA\CNPq, 1989.

MANZANO, José Augusto N.G.; MENDES, Sandro Santa Vicca. *Estudo Dirigido – Delphi 3.0*. Editora Érica Ltda., 2ª Edição, São Paulo, 1998.

MCWORTER Jr., William A. *Matrices Help Relationships.* Disponível em <http://www.cut-the-knot.org/blue/relation.shtml> Acesso em: 18 nov. 2004

OSIER, Dan; BATSON, Steve, GROBMAN, Steve. *Aprenda em 14 dias Delphi 3.* Editora Campus, Rio de Janeiro, 1998.

Parâmetros Curriculares Nacionais: Ensino Médio. Ministério da Educação – Secretaria de Educação Média e Tecnológia, Brasília, 1999.

PALOMINO, Sonia. *Novas Tecnologias e Interdisciplinaridades no Ensino-aprendizado dos conteúdos da disciplina de Álgebra Linear com Aplicações.* FUNGRAD 2003, Universidade Federal de Santa Catarina. Departamento de Matemática.

PEIXOTO, Fábio. *O Segredo da Criptografia*. Revista SuperInteressante, número 5, Maio - 2000, páginas 51 a 53.

PETERSON, Ivars. *Contra Dancing and Matrices.* June 14, 1997. Disponível em <http://www.sciencenews.org/pages/sn_arc97/6_14_97/mathland.htm> Acesso 18 nov. 2004

SANTOS, Carlos Alberto Marcondes dos; GENTIL, Nelson; GRECO, Sérgio Emílio. *Matemática.* Volume único, Editora Ática, São Paulo, 2003.

SILVA, Wellerson Lopes da; CHAVES, Lucas Monteiro. *A Criptografia RSA e o Algoritmo Chinês do Resto*. Disponível em <http://www.dcc.ufla.br/infocomp/artigos/a2v1/criptografiaRSA.pdf> Acesso em 12 agosto 2003.

SMOLLER, Laura. Disponível em <http://www.ualr.edu/~lasmoller/matrices.html> Acesso em: 18 nov. 2004

SYLVESTER, J. J. "Additions to the Articles 'On a New Class of Theorems' and 'On Pascal's Theorem.'" *Philos. Mag.*, 363-370, 1850. Reprinted in *J. J. Sylvester's Mathematical Papers, Vol. 1.* Cambridge, England: At the University Press, pp. 145-151, 1904.

TKOTZ, Viktoria. Vários textos. Disponível em <http://www.numaboa.com/informatica/criptologia/> Acesso em: 08 nov. 2004.

VIERA, Luciana Salles. *Uso Da Informática Na Criação De Ambientes Integrados De Aprendizagem*. Disponível em <*http://www.c5.cl/ieinvestiga/actas/ribie98/242.html*> Acesso em: 14 março 2004.

WEISSTEIN, Eric W. "Matrix." From *MathWorld*--A Wolfram Web Resource. <http://mathworld.wolfram.com/Matrix.html> Acesso em: 20 out. 2004.

WEISSTEIN, Eric W. *Cayley, Arthur (1821-1895)*. Disponível em <http://scienceworld.wolfram.com/biography/Cayley.html> Acesso em 18 nov. 2004

Anexo 1

Matriz Inversa

Definição: Dizemos que uma matriz quadrada **A** é *inversível* se existe uma matriz **B** tal que $\mathbf{AB} = \mathbf{BA} = \mathbf{I}$, onde **I** é a matriz identidade. Essa matriz **B** é única e é chamada de matriz inversa de **A** e denotamos por A^{-1}.

Existência: Existe a inversa de **A** se, e somente se, **det A ≠ 0**. Neste caso diz-se que **A** é inversível ou **A** é *não singular*.

Se **det A = 0**, então **A** é não inversível ou **A** é *singular*.

Propriedades da Matriz Inversa:

a) $\mathbf{(A^{-1})^{-1} = A}$

b) $\mathbf{I = I^{-1}}$

c) $\mathbf{(\lambda A)^{-1}} = \frac{1}{\lambda}\mathbf{A^{-1}}$, onde $\lambda \in \Re$.

Se **A** e **B** são matrizes quadradas, de mesma ordem, tem-se que:

d) $\mathbf{(A + B)^{-1} = A^{-1} + B^{-1}}$

e) $\mathbf{(AB)^{-1} = B^{-1}\, A^{-1}}$

A matriz inversa pode ser determinada através da definição, ou ainda pelo método da matriz adjunta. Como será visto no anexo 2.

Definição: A matriz adjunta de **A**, representada por **Adj(A)** ou $\overline{\mathbf{A}}$, é obtida determinando-se a matriz transposta da matriz de cofatores.

$$\overline{\mathbf{A}} = [\mathbf{A_{ij}}]^{\mathbf{t}}$$

Propriedade: $\mathbf{A \cdot Adj(A) = Adj(A) \cdot A = det(A) \cdot I}$

Se **det(A) ≠ 0** então $\mathbf{A} \cdot \frac{\mathbf{Adj(A)}}{\mathbf{det(A)}} = \frac{\mathbf{Adj(A)}}{\mathbf{det(A)}} \cdot \mathbf{A} = \mathbf{I}$. Portanto: $\mathbf{A^{-1}} = \frac{\mathbf{Adj(A)}}{\mathbf{det(A)}}$

Definição: O cofator do elemento $\mathbf{a_{ij}}$ da matriz **A** é o número dado por

$$\mathbf{cof(A_{ij}) = (-1)^{i+j} \cdot det(A_{ij})}$$

Determinante da Inversa: Pela definição $\mathbf{A \cdot A^{-1} = I}$

$$\mathbf{\det (A \cdot A^{-1}) = \det (I)}$$

$$\mathbf{\det A \cdot \det A^{-1} = \det (I)} \Rightarrow \text{Lei de Binet}$$

$$\mathbf{\det A \cdot \det A^{-1} = 1}$$

$$\boxed{\mathbf{\det A^{-1}} = \frac{1}{\det A}\mathbf{, \text{ com } \det A \neq 0}}$$

Anexo 2

Aritmética Modular

***Definição**:* Dados um número inteiro positivo *m* e dois inteiros *a* e *b* quaisquer, dizemos que *a* é equivalente a *b* módulo *m*, se *a – b* é um múltiplo inteiro de *m*, e denotamos: $a = b \pmod{m}$.

Exemplos: $7 = 2 \pmod{5}$

$-1 = 25 \pmod{26}$

Observação: $a = a \pmod{m}$,

$a = b \pmod{m} \Leftrightarrow b = a \pmod{m}$,

$a = b \pmod{m}$, $b = c \pmod{m} \Rightarrow a = c \pmod{m}$

Proposição:

Se $a = b \pmod{m}$ e $c = d \pmod{m}$ então $a + c = b + d \pmod{m}$ e $a \cdot c = b \cdot d \pmod{m}$.

Demonstração:

$m \mid (b - a)$, $m \mid (d - c) \Rightarrow m \mid (b + d) - (a + c) \Rightarrow (a + c) = (b + d) \pmod{m}$, e

$b \cdot d - a \cdot c = b \cdot (d - c) + (b - a) \Rightarrow m \mid (b \cdot d - a \cdot c) \Rightarrow b \cdot d = a \cdot c \pmod{m}$ **.**

Definição: Dado um número a em Z_m[30], dizemos que um número a^{-1} em Z_m é um recíproco de a módulo m se $a \cdot a^{-1} = a^{-1} \cdot a = 1 \pmod{m}$.

Se a e m não têm fatores primos comuns, então a tem um único recíproco módulo m. Mas, se a e m têm fatores primos comuns, então a não tem recíproco módulo m.

Por exemplo, o número *3* tem recíproco módulo *26*, pois *3* e *26*, não tem fatores primos comuns. Para obter esse recíproco, basta encontrar o número x em Z_{26} que satisfaça a equação modular:

$$3 \cdot x = 1 \pmod{26}$$

Como se está trabalhando com módulo *26*, é possível chegar a uma solução para essa equação utilizando o método de tentativa e erro. O valor encontrado para x é *9*.

---

[30] Z_m é o conjunto dos números inteiros {0, 1, 2, ..., m - 1} equivalentes a a módulo m.

Verificando: 3 · 9 = 27 = 1 (mod 26). Portanto o recíproco de *3* é *9*.

Cálculo da matriz inversa pelo método da adjunta, usando aritmética modular[31]

Para calcular a matriz inversa pelo método da adjunta, usando aritmética modular, deve-se proceder da seguinte forma:

Verificar se a matriz **A** possui inversa, ou seja, calcular o determinante da matriz. Se **det A ≠ 0**, então pela definição de matriz inversa, existe $\mathbf{A}^{-1}$. Pelo método da adjunta $\mathbf{A^{-1}} = \frac{\mathbf{1}}{\mathbf{detA}} \cdot \begin{bmatrix} \mathbf{d} & \mathbf{-b} \\ \mathbf{-c} & \mathbf{a} \end{bmatrix} \mathbf{(mod\ 26)}$. O módulo 26 refere-se apenas a restrição feita pela quantidade de caracteres da tabela de conversão utilizada na *Cifra de hill* e nos programas feitos no *Matlab*.

Seja $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix}$, então $\mathbf{det(A) = 1 \cdot 3 - 2 \cdot 0 = 3}$. Tem – se:

$$\mathbf{A^{-1}} = \mathbf{(3)^{-1}} \cdot \begin{bmatrix} \mathbf{3} & \mathbf{-2} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \mathbf{(mod\ 26)}$$

Como o recíproco de *3* é *9* (anexo 2), tem – se:

$$\mathbf{A^{-1}} = \mathbf{9} \cdot \begin{bmatrix} \mathbf{3} & \mathbf{-2} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

$$\mathbf{A^{-1}} = \begin{bmatrix} \mathbf{27} & \mathbf{-18} \\ \mathbf{0} & \mathbf{9} \end{bmatrix}$$

$$\mathbf{A^{-1}} = \begin{bmatrix} \mathbf{1} & \mathbf{8} \\ \mathbf{0} & \mathbf{9} \end{bmatrix} \mathbf{(mod\ 26)}$$

[31] Para obter mais informações ver ANTON, 2001.

Anexo 3

Sequência didática

Objetivo: mostrar uma aplicação da teoria de matrizes através da Criptografia.

*Atividade proposta aos alunos, utilizando o programa **Criptografia**:*

Nessa atividade foi fornecida uma folha contendo as seguintes informações:

Tabela

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	0

Exemplo:

B	O	A	_	T	A	R	D	E
2	15	1	0	20	1	18	4	5

Matriz texto comum: $\begin{bmatrix} 2 & 0 & 18 \\ 15 & 20 & 4 \\ 1 & 1 & 5 \end{bmatrix}$

No programa digite a matriz, codifique e decofique a mensagem.

Agora, usando a tabela, faça o mesmo com:

a) GEOMETRIA
b) LOGARITMO
c) O_GRAFICO
d) CODIFICAR
e) BOLA

Crie mensagens, codifique e envie para os seus colegas.

O exemplo da folha foi desenvolvido com os alunos para ao mesmo tempo ser explicado como funcionava o programa. Em seguida, eles

codificaram e decodificaram as cinco mensagens (itens *a* até *e*). Para finalizar a apresentação foi deixado um tempo para se divertirem criando mensagens e enviando para os colegas.

Atividade proposta aos alunos, utilizando os programas ***Codificação*** *e* ***Decodificação****:*

Essa atividade foi apresentada utilizando-se o *PowerPoint*. A medida que as mensagens codificadas apareciam na tela, os alunos utilizavam o programa **Decodificação**, para descobrir o que estava escrito.

A primeira mensagem codificada dava boas vindas ao alunos, de acordo com o colégio que estava participando da apresentação:

WADMYOSUKWZWLIVJIRVQDTWWHZPBPZDOLTHT

SEJAM BEM VINDOS ALUNOS DO RODA PIAO

WADMYOSUKWZWLIVJIRVQDTWWHZPOKCKRRPGM

SEJAM BEM VINDOS ALUNOS DO INSTITUTO

A mensagem codificada a seguir, foi dividida em grupos de alunos. Cada grupo tinha que decifrar a parte que recebeu e tentar adivinhar o restante da mensagem. Por exemplo, um dos grupos ficou com a mensagem:

QHS_WEFTUZNNVJVKKJSJEDYCZQJAGIRWXCDP

E decifrou, obtendo:

A IMAGINACAO É MUITO MAIS IMPORTANTE

O outro ficou com:

ZCRLCLJNATE_OVRWNF_ _ _RYNQXUMSMGQBN_I

E decifrou, obtendo:

QUE O CONHECIMENTO ALBERT EINSTEIN

Em seguida, bastava os dois grupos juntarem as mensagens, ou cada um deles adivinhar o restante da sua e encontravam a famosa frase:

"A IMAGINAÇÃO É MUITO MAIS IMPORTANTE QUE O CONHECIMENTO."
(ALBERT EINSTEIN)

As mensagens codificadas seguintes, também foram divididas em grupo. Mas, além de decifrar a sua parte e adivinhar o restante, era necessário responder a questão que aparecia:

QAZQQSHZPXRNWFHSJEWIREWXFMRXDIKVEQQS
RZVV_TMXCNYUVFVFBCIXDWXHNTVD_IGJI_ _ _

O PAI DE MEU NETO É O NETO DE MEU PAI.
QUANTAS PESSOAS ESTÃO ENVOLVIDAS?

OKUGFIFISGQJRA_ULC_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

RESPOSTA: QUATRO

SBYAPPBLOAPPDE_HZPTIGWGURPCPKPQGNYVI
DYCMXCMGEKWDVJJTWWHZPZCRNTWZDWSWG_ _ _

O QUE É QUE SENDO APENAS SEU É USADO
MAIS PELOS OUTROS DO QUE POR VOCÊ?

OKUGFIFISKSHAJKXBMVVE_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

RESPOSTA: SEU NOME

A resposta era fornecida, em forma cifrada, apenas quando todos tinham chegado a uma conclusão, ou no caso em que não conseguiam responder de maneira alguma. Decifrando a resposta verificavam se haviam acertado ou não.

De acordo como o tempo disponível, era deixado um espaço para que os alunos criassem suas mensagens, utilizando para isso, o programa **Codificação**, e mandassem para seus colegas tentarem decifrar.

Para concluir a apresentação, os alunos decifravam uma mensagem de agradecimento, como a seguinte:

ADCELSRTLTWWPXXTWTMKHLTJWNNVVSPLOLLO
FLCFRPUCXJUJBNBKBMHVXFEMJIFYAPE_ _RTZ
CESVMENZOQTHCDARUGGCFBGTEEGVYCGRDYSM

AGRADECEMOS SUA PRESENÇA, ALUNOS DO COLÉGIO RODA PIÃO. ESPERAMOS QUE ESSA EXPERIÊNCIA TENHA SIDO VÁLIDA. VALEU!!!!

Anexo 4

Comentários dos alunos do Colégio Autonomia:

"*Achei muito interessante o programa criado, pois facilita a resolução de exercícios e ajuda no estudo de matrizes. O legal é que podemos estudar este assunto saindo da monotonia das aulas*."

"*Gostei muito do programa desenvolvido para aprender matrizes. Se todo aluno tivesse oportunidade de usar o programa com certeza iria ter muito mais facilidade de aprender e exercitar a matéria. Seria ótimo como um reforço escolar, pois é fácil de usar o programa*."

"*Na minha opinião foi por enquanto a melhor aula de matemática que já tivemos. Com a ajuda dos softwares, pudemos entender melhor o conteúdo de matrizes e aprender um pouco de matriz inversa (...). O programa também poderia ser aplicado nos computadores dos colégios*."

"(...) *A maneira de resolver matrizes através de um software é muito boa, torna o conteúdo mais prático e tendo a correção dos exercícios o torna mais seguro. Gostei bastante do exemplo dado mostrando onde as matrizes são aplicadas.* (...)"

"*O programa desenvolvido (...) apresenta uma abordagem diferente, mais dinâmica e divertida do conteúdo de Matemática (...)*"

"*O programa todo está bem desenvolvido, ensinando o conteúdo de forma bem didática. (...)*"

"*(...) A parte de matrizes funciona bem como exercício e revisão do conteúdo, ainda estimula o cálculo mental. Pode até ser usado para avaliações. (...)*"

"*Eu achei interessante e uma criativa forma de aprender o conteúdo de matrizes (...). Como me interesso muito por informática, pra mim é uma forma mais prática, divertida e fácil de aprender (...)*"

"*(...) trabalhar os conteúdos vistos em sala de aula de outra forma é sempre bom. Gostei de conhecer aplicações (...) como na cifragem, códigos. (...)*"

Anexo 5

Código dos programas Criptografia e Decriptografia desenvolvidos em Matlab

Criptografia

```
disp 'matriz codificadora:'
K=[-1 -1 2; -1 2 1; 1 -1 -1]
Z=input('entre com a matriz do
texto comum: Z:')
disp 'produto da matriz
codificadora pela matriz do
texto comum:'
X=K*Z
if 0<X(1,1)<26;
   Y(1,1)=X(1,1);
end
if X(1,1)<0
   q0=X(1,1)/26;
   w0=floor(q0);
   r0=X(1,1)-(w0*26);
   Y(1,1)=r0;
   end
if X(1,1)>26;
   q=X(1,1)/26;
   w=floor(q);
   r=X(1,1)-(w*26);
   Y(1,1)=r;
   end
   if 0<X(1,2)<26;
      Y(1,2)=X(1,2);
   end
   if X(1,2)<0
   q9=X(1,2)/26;
   w9=floor(q9);
   r9=X(1,2)-(w9*26);
   Y(1,2)=r9;
   end
if X(1,2)>26;
   q1=X(1,2)/26;
   w1=floor(q1);
   r1=X(1,2)-(w1*26);
   Y(1,2)=r1;
end
if 0<X(1,3)<26;
   Y(1,3)=X(1,3);
end
if X(1,3)<0
   q10=X(1,3)/26;
   w10=floor(q10);
   r10=X(1,3)-(w10*26);
   Y(1,3)=r10;
   end
if X(1,3)>26;
   q2=X(1,3)/26;
   w2=floor(q2);
   r2=X(1,3)-(w2*26);
   Y(1,3)=r2;
end
if 0<X(2,1)<26;
   Y(2,1)=X(2,1);
end
if X(2,1)<0
   q11=X(2,1)/26;
   w11=floor(q11);
   r11=X(2,1)-(w11*26);
   Y(2,1)=r11;
   end
if X(2,1)>26;
   q3=X(2,1)/26;
   w3=floor(q3);
   r3=X(2,1)-(w3*26);
   Y(2,1)=r3;
end
```

```
if 0<X(2,2)<26;
   Y(2,2)=X(2,2);
end
if X(2,2)<0
   q12=X(2,2)/26;
   w12=floor(q12);
   r12=X(2,2)-(w12*26);
   Y(2,2)=r12;
   end
if X(2,2)>26;
   q4=X(2,2)/26;
   w4=floor(q4);
   r4=X(2,2)-(w4*26);
   Y(2,2)=r4;
end
if 0<X(2,3)<26;
   Y(2,3)=X(2,3);
end
if X(2,3)<0
   q13=X(2,3)/26;
   w13=floor(q13);
   r13=X(2,3)-(w13*26);
   Y(2,3)=r13;
   end
if X(2,3)>26;
   q5=X(2,3)/26;
   w5=floor(q5);
   r5=X(2,3)-(w5*26);
   Y(2,3)=r5;
end
if 0<X(3,1)<26;
   Y(3,1)=X(3,1);
end
if X(3,1)<0
   q14=X(3,1)/26;
   w14=floor(q14);
   r14=X(3,1)-(w14*26);
   Y(3,1)=r14;
   end
if X(3,1)>26;
   q6=X(3,1)/26;
   w6=floor(q6);
   r6=X(3,1)-(w6*26);
   Y(3,1)=r6;
end
if 0<X(3,2)<26;
   Y(3,2)=X(3,2);
end
if X(3,2)<0
   q15=X(3,2)/26;
   w15=floor(q15);
   r15=X(3,2)-(w15*26);
   Y(3,2)=r15;
   end
if X(3,2)>26;
   q7=X(3,2)/26;
   w7=floor(q7);
   r7=X(3,2)-(w7*26);
   Y(3,2)=r7;
end
if 0<X(3,3)<26;
   Y(3,3)=X(3,3);
end
if X(3,3)<0
   q16=X(3,3)/26;
   w16=floor(q16);
   r16=X(3,3)-(w16*26);
   Y(3,3)=r16;
   end
if X(3,3)>26;
   q8=X(3,3)/26;
   w8=floor(q8);
   r8=X(3,3)-(w8*26);
   Y(3,3)=r8;
end
   disp 'matriz da mensagem
cifrada:'
   eval Y;

s=str2mat('')
disp 'mensagem cifrada:'
```

```
for j=1:3
for i=1:3
if Y(i,j) == 1
   s=str2mat(s,'A')
   end
if Y(i,j) == 2
 s=str2mat(s,'B')
 end
if Y(i,j) == 3
 s=str2mat(s,'C')
 end
if Y(i,j) == 4
 s=str2mat(s,'D')
 end
if Y(i,j) == 5
 s=str2mat(s,'E')
 end
if Y(i,j) == 6
 s=str2mat(s,'F')
 end
if Y(i,j) == 7
 s=str2mat(s,'G')
 end
if Y(i,j) == 8
 s=str2mat(s,'H')
 end
if Y(i,j) == 9
 s=str2mat(s,'I')
 end
 if Y(i,j) == 10
 s=str2mat(s,'J')
 end
 if Y(i,j) == 11
 s=str2mat(s,'K')
 end
 if Y(i,j) == 12
 s=str2mat(s,'L')
 end
 if Y(i,j) == 13
 s=str2mat(s,'M')
 end
 if Y(i,j) == 14
 s=str2mat(s,'N')
 end
 if Y(i,j) == 15
 s=str2mat(s,'O')
 end
 if Y(i,j) == 16
 s=str2mat(s,'P')
 end
 if Y(i,j) == 17
    s=str2mat(s,'Q')
 end
 if Y(i,j) == 18
 s=str2mat(s,'R')
 end
if Y(i,j) == 19
 s=str2mat(s,'S')
 end
 if Y(i,j) == 20
 s=str2mat(s,'T')
 end
 if Y(i,j) == 21
 s=str2mat(s,'U')
 end
 if Y(i,j) == 22
 s=str2mat(s,'V')
 end
 if Y(i,j) == 23
 s=str2mat(s,'W')
 end
 if Y(i,j) == 24
 s=str2mat(s,'X')
 end
if Y(i,j) == 25
 s=str2mat(s,'Y')
 end
  if Y(i,j) == 0
     s=str2mat(s,'Z')
      end
   end
end
```

Decriptografia

```
disp 'matriz decodificadora:'
D=[1 3 5;0 1 1; 1 2 3]
E=input('entre com a matriz da
mensagem cifrada: Y:')
disp 'produto da matriz
decodificadora pela matriz da
mensagem cifrada:'
F=D*E
if 0<F(1,1)<26;
   G(1,1)=F(1,1);
end
if F(1,1)<0;
   q0=F(1,1)/26;
   w0=floor(q0);
   r0=F(1,1)-(w0*26);
   G(1,1)=r0;
   end
if F(1,1)>26;
   q=F(1,1)/26;
   w=floor(q);
   r=F(1,1)-(w*26);
   G(1,1)=r;
   end
if 0<F(1,2)<26;
   G(1,2)=F(1,2);
end
if F(1,2)<0;
   q9=F(1,2)/26;
   w9=floor(q9);
   r9=F(1,2)-(w9*26);
   G(1,2)=r9;
   end
if F(1,2)>26;
   q1=F(1,2)/26;
   w1=floor(q1);
   r1=F(1,2)-(w1*26);
   G(1,2)=r1;
end
if 0<F(1,3)<26;
   G(1,3)=F(1,3);
end
if F(1,3)<0;
   q10=F(1,3)/26;
   w10=floor(q10);
   r10=F(1,3)-(w10*26);
   G(1,3)=r10;
   end
if F(1,3)>26;
   q2=F(1,3)/26;
   w2=floor(q2);
   r2=F(1,3)-(w2*26);
   G(1,3)=r2;
   end
if 0<F(2,1)<26;
   G(2,1)=F(2,1);
end
if F(2,1)<0;
   q11=F(2,1)/26;
   w11=floor(q11);
   r11=F(2,1)-(w11*26);
   G(2,1)=r11;
   end
if F(2,1)>26;
   q3=F(2,1)/26;
   w3=floor(q3);
   r3=F(2,1)-(w3*26);
   G(2,1)=r3;
end
if 0<F(2,2)<26;
   G(2,2)=F(2,2);
end
if F(2,2)<0;
   q12=F(2,2)/26;
   w12=floor(q12);
   r12=F(2,2)-(w12*26);
   G(2,2)=r12;
   end
if F(2,2)>26;
   q4=F(2,2)/26;
   w4=floor(q4);
   r4=F(2,2)-(w4*26);
```

```
   G(2,2)=r4;
   end
if 0<F(2,3)<26;
   G(2,3)=F(2,3);
end
if F(2,3)<0;
   q13=F(2,3)/26;
   w13=floor(q13);
   r13=F(2,3)-(w13*26);
   G(2,3)=r13;
   end
if F(2,3)>26;
   q5=F(2,3)/26;
   w5=floor(q5);
   r5=F(2,3)-(w5*26);
   G(2,3)=r5;
end
if 0<F(3,1)<26;
   G(3,1)=F(3,1);
end
if F(3,1)<0;
   q14=F(3,1)/26;
   w14=floor(q14);
   r14=F(3,1)-(w14*26);
   G(3,1)=r14;
   end
if F(3,1)>26;
   q6=F(3,1)/26;
   w6=floor(q6);
   r6=F(3,1)-(w6*26);
   G(3,1)=r6;
end
if 0<F(3,2)<26;
   G(3,2)=F(3,2);
end
if F(3,2)<0;
   q15=F(3,2)/26;
   w15=floor(q15);
   r15=F(3,2)-(w15*26);
   G(3,2)=r15;
   end
if F(3,2)>26;
   q7=F(3,2)/26;
   w7=floor(q7);
   r7=F(3,2)-(w7*26);
   G(3,2)=r7;
end
if 0<F(3,3)<26;
   G(3,3)=F(3,3);
end
if F(3,3)<0;
   q16=F(3,3)/26;
   w16=floor(q16);
   r16=F(3,3)-(w16*26);
   G(3,3)=r16;
   end
if F(3,3)>26;
   q8=F(3,3)/26;
   w8=floor(q8);
   r8=F(3,3)-(w8*26);
   G(3,3)=r8;
end
disp 'matriz da mensagem
decifrada:'
eval G;
s=str2mat('')
disp 'mensagem decifrada:'
for j=1:3
for i=1:3
if G(i,j) == 1
   s=str2mat(s,'A')
   end
if G(i,j) == 2
 s=str2mat(s,'B')
 end
if G(i,j) == 3
 s=str2mat(s,'C')
 end
if G(i,j) == 4
 s=str2mat(s,'D')
 end
if G(i,j) == 5
```

```
 s=str2mat(s,'E')
 end
if G(i,j) == 6
 s=str2mat(s,'F')
 end
if G(i,j) == 7
 s=str2mat(s,'G')
 end
if G(i,j) == 8
 s=str2mat(s,'H')
 end
if G(i,j) == 9
 s=str2mat(s,'I')
 end
 if G(i,j) == 10
 s=str2mat(s,'J')
 end
 if G(i,j) == 11
 s=str2mat(s,'K')
 end
 if G(i,j) == 12
 s=str2mat(s,'L')
 end
 if G(i,j) == 13
 s=str2mat(s,'M')
 end
 if G(i,j) == 14
 s=str2mat(s,'N')
 end
 if G(i,j) == 15
 s=str2mat(s,'O')
 end
 if G(i,j) == 16
 s=str2mat(s,'P')
 end
 if G(i,j) == 17
    s=str2mat(s,'Q')
 end
 if G(i,j) == 18
 s=str2mat(s,'R')
 end
if G(i,j) == 19
 s=str2mat(s,'S')
 end
 if G(i,j) == 20
 s=str2mat(s,'T')
 end
 if G(i,j) == 21
 s=str2mat(s,'U')
 end
 if G(i,j) == 22
 s=str2mat(s,'V')
 end
 if G(i,j) == 23
 s=str2mat(s,'W')
 end
 if G(i,j) == 24
 s=str2mat(s,'X')
 end
if G(i,j) == 25
 s=str2mat(s,'Y')
 end
  if G(i,j) == 0
     s=str2mat(s,'Z')
      end
   end
end
```