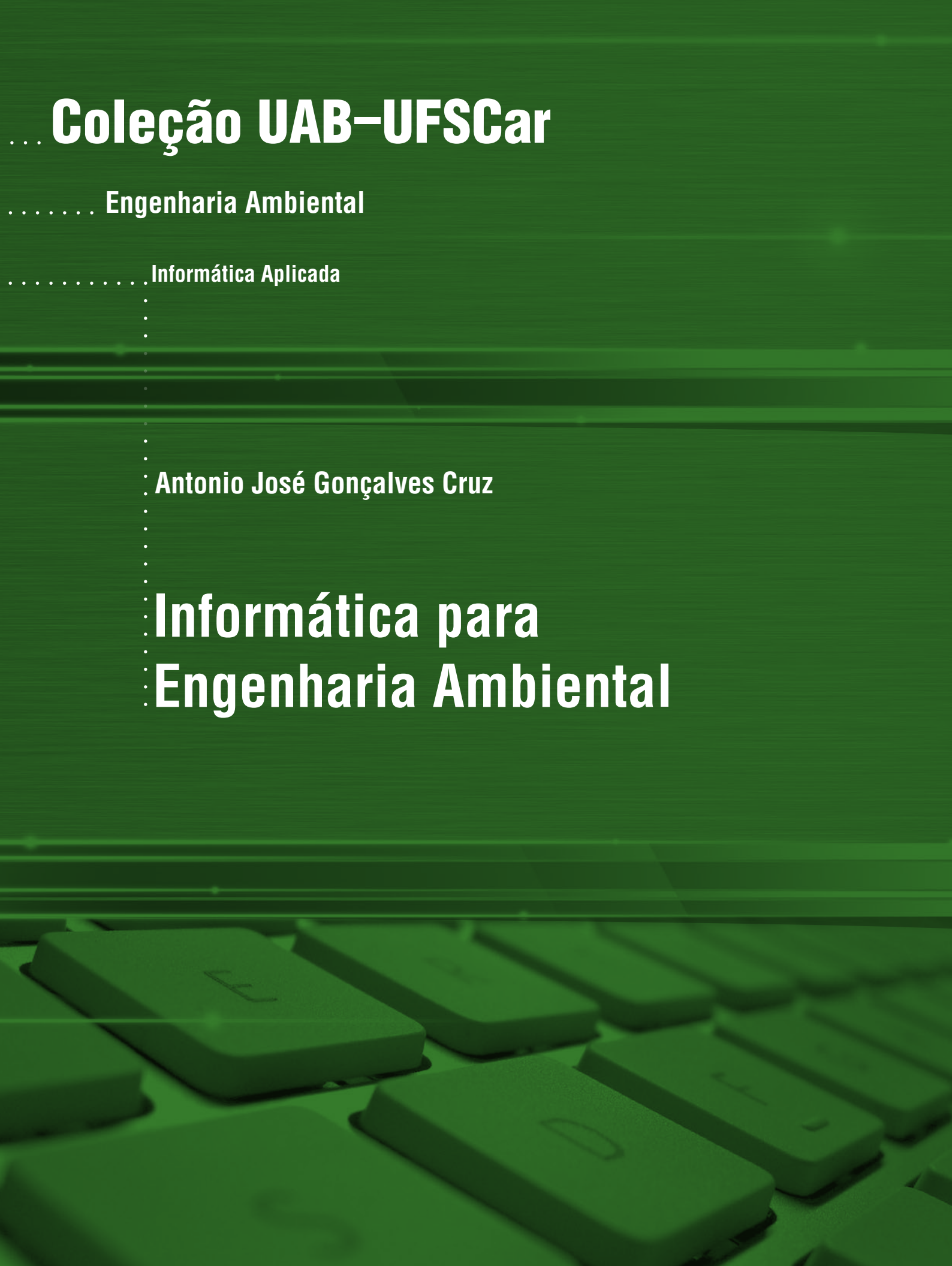
Coleção UAB–UFSCar

Engenharia Ambiental

Informática Aplicada

Antonio José Gonçalves Cruz

Informática para Engenharia Ambiental



Informática para Engenharia Ambiental

UAB-UFSCar
Universidade Federal de São Carlos
Rodovia Washington Luís, km 235
13565-905 - São Carlos, SP, Brasil
Telefax (16) 3351-8420
www.uab.ufscar.br
uab@ufscar.br

Antonio José Gonçalves Cruz

Informática para Engenharia Ambiental

São Carlos
2013

Concepção Pedagógica
Daniel Mill

Supervisão
Douglas Henrique Perez Pino

Equipe de Revisão Linguística
Clarissa Galvão Bengtson
Daniel William Ferreira de Camargo
Gabriela Aniceto
Letícia Moreira Clares
Sara Naime Vidal Vital

Equipe de Editoração Eletrônica
Izis Cavalcanti

Equipe de Ilustração
Maria Julia Barbieri Mantoanelli

Capa e Projeto Gráfico
Luís Gustavo Sousa Sguissardi

SUMÁRIO

APRESENTAÇÃO

O livro é organizado em quatro unidades. Na primeira é apresentado um breve histórico da evolução dos computadores adotando como marco inicial o surgimento do primeiro computador moderno no ano de 1946: o ENIAC (em inglês “Eletronic Numerical Integrator and Computer”, traduzido como “Integrador Numérico Eletrônico e Computador”). Apresentam-se os componentes básicos de um computador, sua arquitetura e funcionamento e a forma como a informação é armazenada. Definem-se os termos bits, byte e palavra. Citam-se alguns exemplos de sistemas operacionais.

Na segunda unidade aborda-se o tema algoritmo e suas formas de representação. O objetivo é mostrar que a solução de um problema no computador passa por uma seqüência de etapas. A primeira delas é a abstração, ou seja, a descrição do problema de forma clara e precisa. Objetiva-se buscar uma seqüencia de passos que conduzam até a solução do problema. A próxima etapa é a representação destes passos por meio de estruturas apropriadas para termos a solução do problema na forma de um fluxograma. O próximo passo consistirá em utilizar um programa computacional para resolução dos problemas, o que é apresentado nas próximas duas unidades.

Na terceira unidade apresenta-se o programa Scilab (Consortium Scilab: INRIA, ENPC). Este programa foi escolhido pelo fato de ser distribuído livremente. O programa Scilab é uma linguagem interpretada voltada para resolução de problemas numéricos. Sua linguagem de programação simples e de fácil aprendizado apresenta similaridade com outros programas comerciais. O conhecimento deste programa fornece a base para resolução de problemas típicos de um curso de engenharia. Aprofundando nesta ferramenta, o usuário poderá utilizá-lo para solução de problemas mais complexos.

Na quarta unidade apresenta-se a planilha de cálculo Calc (pacote computacional BrOffice.org). A escolha por este aplicativo tem a mesma justificativa já apresentada: ser distribuído livremente, e apresentar similaridades com a planilha Excel (pacote computacional Office). O conhecimento básico deste programa fornecerá os subsídios necessários para resolução de problemas típicos de um curso de engenharia. O domínio desta ferramenta permitirá que o usuário possa utilizá-lo na solução de problemas mais complexos.

UNIDADE 1

O Computador

1.1 Primeiras palavras

Caro aluno(a): vamos iniciar nossa jornada nesta disciplina com uma pergunta: O que é o computador ?

Imagino que você tenha clara essa definição. Contudo, busquei a definição desta palavra em dicionários da língua portuguesa. Apresento-a:

"[Do latim computadore.]"

"S.m. 1. O que computa: calculador, calculista. 2. Inf. Máquina destinada ao processamento de dados; dispositivo capaz de obedecer a instruções que visam produzir certas transformações nos dados com o objetivo de alcançar um fim determinado."

"Computador digital. Inf. Computador que opera com dados discretos ou descontínuos, efetuando com eles processos aritméticos ou lógicos."

"Um computador eletrônico digital é basicamente integrado por: unidade central de processamento (CPU), unidades de memória, canais de comunicação e dispositivos de entrada e saída."

"Computador eletrônico. Processador de dados com capacidade de aceitar informações, efetuar com elas operações programadas, fornecer resultados para resolução de problemas." (Ferreira, 1986; Houaiss et al., 2001).

"*S.m. (computar*[1]*+dor*[2]*).* 1 O que faz cômputos (pessoa ou máquina). 2 Calculista."

"C. analógico, Inform.: computador que processa dados na forma analógica, ou seja, dados representados por um sinal que varia continuamente. *(...)"*

"C. de colo, Inform.: V laptop."

"*C. desktop*, *Inform.*: computador, compatível com o PC, que pode ser colocado na mesa de um usuário. *Cf laptop. C. digital, Inform.*: computador que processa dados na forma digital."

"*C. eletrônico*: máquina de calcular automática, para solução de problemas matemáticos complexos, que utiliza dispositivos eletrônicos."

"C. pessoal, Inform.: microcomputador de baixo custo planejado para utilização doméstica ou em escritórios. Sigla: PC." (Dicionário Michaelis, 2008).

A origem da palavra computador deriva do ato de calcular, fazer cômputos. Por esta definição a história do computador tem início há muitos anos atrás, com a invenção pelo Homem do primeiro instrumento para auxiliar na elaboração de contas: o ábaco. Mas este equipamento não funciona por si só. É preciso de uma pessoa que domine um conjunto de regras necessárias para sua correta

operação. Com os computadores atuais isto não será diferente. Em nosso dia a dia o computador está cada vez mais presente, embora em muitas das vezes nem percebemos sua presença. O simples ato de acessar informações da conta bancária (saldo, extrato, pagamento, saque de dinheiro) em terminais eletrônicos (caixas eletrônico) tem por trás o acesso a um computador.

Chamados de computadores pessoais, o computador de mesa ou de escritório (*desktop*) ou o computador de colo (*notebook),* são ambos considerados ícones da era da informação.

1.2 Problematizando o tema

Nesta unidade apresentaremos uma breve história dos computadores, seus componentes principais e seu funcionamento. Nos dias atuais esta ferramenta tornou-se um elemento imprescindível em muitas áreas do conhecimento e, particularmente, como ferramenta de apoio ao ensino das engenharias.

1.3 Breve histórico dos computadores

O ábaco foi o primeiro instrumento criado pelo Homem para realizar contas. Pode ser considerado como o primeiro computador da história. O ábaco provavelmente surgiu como conseqüência do ato natural de se contar nos dedos. Não se conhece a data precisa em que este instrumento surgiu. O seu surgimento remonta há mais de 5500 anos, surgindo de forma independente em diferentes civilizações, como na China e na Mesopotâmia. Esse instrumento é formado por um quadro, composto por cordas ou arames transversais nas quais se inserem os elementos de contagem (bolas perfuradas). Os valores são representados em função da posição dos elementos de contagem em suas respectivas posições no ábaco. A Figura 1.1 apresenta a ilustração de um ábaco.

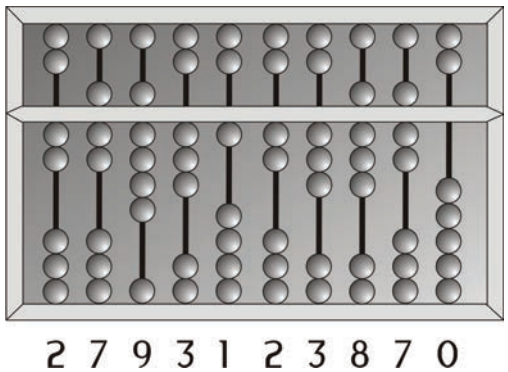


Figura 1.1 Ábaco chinês (*suan pan*) representando o número 2793123870.

Mas não é exatamente este tipo de “máquina” que desejamos abordar e que iremos utilizar no curso. Apresentar um histórico da evolução das máquinas de calcular até chegar ao que conhecemos hoje por computador seria um pouco tedioso e fugiria do tema desta unidade. Não pretendemos realizar um completo levantamento histórico, mas sim citar alguns fatos marcantes na história do desenvolvimento dos computadores.

1.3.1 Na linha do tempo

Foi durante a Segunda Guerra Mundial que surgiram os primeiros computadores com concepção similar aos computadores atuais. O primeiro computador do Mundo, o ENIAC (em inglês “Eletronic Numerical Integrator and Computer”, traduzido como “Integrador Numérico Eletrônico e Computador”) foi inventado pelos americanos John Mauchly e J. Presper Eckert no ano de 1946. Esse computador era composto por 18.000 válvulas, pesava algo em torno de 28 toneladas, consumia 178 kwh de energia e ocupava uma área de aproximadamente 170 m^2. O objetivo desta primeira máquina era auxiliar nos cálculos de balística para o exército americano durante a segunda guerra mundial. O ENIAC pode ser considerado um marco no desenvolvimento dos computadores, pois a partir de sua invenção, deu-se início a um processo de evolução sem precedentes. Citamos, a seguir, o ano e o nome de alguns computadores ou fatos que marcaram época:

- 1946: ENIAC (em inglês “Eletronic Numerical Integrator and Computer”, traduzido como “Integrador Numérico Eletrônico e Computador”).
- 1946: John Von Neumann, matemático húngaro, elabora proposta contendo as três características que formam a base para os computadores programáveis modernos:

  1. Codificar as instruções para serem armazenadas no computador;
  2. Armazenar na memória do computador as informações necessárias para a execução de determinada tarefa. Foi Von Neumann quem sugeriu que a informação poderia ser codificada empregando zeros e uns.
  3. Durante o processamento do programa, a busca das instruções deve ser realizada na memória do computador.

  Von Neumann propôs que um programa (conjunto de instruções a serem executadas pela máquina) deveria ser armazenado no computador da mesma forma que os dados. Surge o conceito de programa armazenado. Essa proposta foi chamada de “Arquitetura de Von Neumann” e é a base dos computadores atuais.

- 1946: John Tukey emprega pela primeira vez o termo ***bit*** com o significado de “digito binário”.
- 1947: EDVAC (do inglês “Electronic Discrete Variable Automatic Computer”, traduzido como “Computador Automático de Variáveis Discretas Eletrônicas”). Foi uma versão melhorada do computador ENIAC.
- 1948: Patenteado o primeiro transistor pelos pesquisadores William Bradford Shockley, John Bardeen e Walter H. Brattain da Bell Labs.
- 1949: EDSAC (do inglês “Electronic Delay Storage Automatic Calculator”, traduzido como “Calculador Automático de Armazenamento Eletrônico com Atraso”) desenvolvido por Maurice V. Wilker, na Universidade de Cambridge.

  Surge o conceito de subprogramas ou subrotinas, que eram pequenos programas armazenados em cartões perfurados de papel.
- 1950: Surge a linguagem Assembly na Universidade de Cambridge (Maurice V. Wilkes). A linguagem foi utilizada no computado EDSAC.
- 1951: Surgimento dos computadores empregando transistores. Redução drástica no peso, tamanho e consumo de energia em relação aos seus predecessores. O desempenho evolui muito.
- 1951: UNIVAC I (do inglês “UNIVersal Automatic Computer”, traduzido como “Computador Automático Universal”). Foi o primeiro computador produzido comercialmente pela empresa americana Remington Rand.
- 1952-1953: A empresa americana IBM (“International Business Machines”) introduz o computador denominado de 701, primeiro computador com programa armazenado.
- 1954-1957: Desenvolvimento do primeiro compilador da linguagem de programação científica FORTRAN (“Formula Translation”).
- 1955: Os laboratórios da ATT& Bell anunciam a construção do primeiro computador totalmente transistorizado. Nova redução no consumo de energia. Inicia-se a segunda geração dos computadores.
- 1958: Jack Kilby e a empresa “Texas Instrument” solicitaram o pedido de patente do circuito integrado, também chamado de *chip*.
- 1963: Neste ano é inventado o mouse, patenteado com o título “X-Y Position Indicator For A Display System”. Contudo, este dispositivo não encontrou utilidade na época, pois os computadores apenas manipulavam textos sem cursores na tela.
- 1966: A empresa americana HP (Hewlett-Packard) inicia a produção do computador HP-2115 para uso geral. Este computador possuía um alto

poder de processamento para época e permitia o uso de linguagens de programação como Basic, Algol e FORTRAN.

- 1967: a empresa americana construiu o primeiro disco para armazenamento de dados (floppy disk).
- 1969: Dois programadores dos laboratórios AT&T Bell, Ken Thmpson e Denis Richie, desenvolveram o sistema operacional denominado UNIX. Foi o primeiro sistema operacional que poderia ser instalado em qualquer computador.

  Neste mesmo ano o compilador PASCAL foi criado por Niklaus Wirth.
- 1971: A empresa americana Intel lança o primeiro microprocessador, o chip 4004.
- 1972: A mesma empresa Intel lança o chip 8008, processador de 8 bits que acessava 16 kbytes de memória.

  Bill Gates e Paul Allen fundaram a empresa “Traf-O-Data”, que viria no ano de 1975 ser renomeada para “MicroSoft”. O hífen seria retirado anos depois.
- 1975: A empresa americana IBM lança o IBM 5100, primeiro computador produzido em grande quantidade.

  Primeira linguagem de programação voltada para um computador pessoal é apresentada: o BASIC. Foi desenvolvida por Bill Gates e Paul Allen.

  Neste ano foi criada a empresa “Apple Corporation” que criaria os computadores Apple e Macintosh.
- 1976: Em 1º de abril de 1976, Steven Paul Jobs, Stephen Gary Wozniak e Ronald Gerald Wayne fundaram a Apple Computer. A empresa lança no mercado um kit para montagem de computador do tipo “faça você mesmo”, o Apple I.
- 1977: A empresa Apple Computer apresenta o computador pessoal Apple II, que permitia produzir gráficos a cores. O pc tinha memória de 4 KB, precisava ser conectado à uma TV e não tinha mouse, mas tinha circuito impresso em sua placa-mãe, fonte de alimentação, teclado e cartuchos para jogos.
- 1979: Surge o primeiro programa de planilha eletrônica (chamado VisiCalc). Surge também o processador de textos Wordstar para computadores pessoais.

- 1980: Neste ano estavam em uso nos Estados Unidos mais de 1 milhão de computadores.
- 1981: A empresa americana IBM lança no mercado o PC (do inglês, “Personal Computer”). A empresa Microsoft desenvolve o sistema operacional denominado de MS-DOS (“Microsoft Disk Operating System”).
- 1983: A empresa americana IBM lançou o computador XT, que teve sua arquitetura copiada em todo mundo.

  A partir deste ano os computadores passaram a ser conhecidos pelos processadores que utilizam (AT 286, AT 386, AT 486, Pentium, etc). A evolução na velocidade dos processadores foi enorme.
- 1984: A empresa Apple Computer (do empresário Steven P. Jobs), apresentou ao mercado o Apple Macintosh (CPU Motorola 68000 de 8 Mhz, 32 bits, vídeo incorporado monocromático de 9 polegadas, gráficos 512x342, unidade de disco flexível de 3,5 polegadas, com capacidade de 400 Kb, 128 Kb RAM e mouse). Rapidamente a venda do Apple Macintosh alcançou o número de 50.000 unidades vendidas. Tal sucesso de vendas pode ser atribuído às seguintes características do computador: uso de sistema operacional baseado em interface gráfica de usuário – GUI (Graphic User Interface), muito amigável; utilizar microprocessador de 32 bits e utilizar mouse.

  Richard Stallman, pesquisador do MIT (Massachussets Institute of Technology) idealizou o movimento pelo software livre.
- 1985: Sistema operacional denominado Windows foi lançado pela empresa Microsoft. Esse sistema operacional viria a ser utilizado em grande parte dos computadores pessoais. Versões do Microsoft Windows: Windows 2.0 (1987), Windows 3.0 (1990), Windows 3.1 (1992), Windows 3.11 (1993), Windows NT 3.1 (1993), Windows 95 (1995), Windows NT 4.0 (1996), Windows 98 e NT 5.0 (1998), Windows 2000 (2000), Windows XP (2001) e Windows Vista (2007).
- 1986: Ano em que foi descoberto o primeiro vírus de computador.
- 1988: Mais de 100 milhões de computadores estavam em uso no mundo.
- 1989: Projeto conhecido por “World Wide Web” (proposto por investigadores do Centro Europeu para a Investigação Nuclear (CERN) de Genebra, Tim Berners-Lee e Robert Cailliau, deu início ao que ficaria conhecido como a rede mundial de computadores.

- 1991: Surge o sistema operacional Linux, desenvolvido por Linus Torvalds, estudante de Ciência da Computação da Universidade de Helsinki, Finlândia.
- 1994: O site de buscas Yahoo! foi criado.
- 1998: Mais de 150 milhões de usuários conectavam-se à internet no mundo.
- 2001: a empresa Apple lançou um novo sistema operacional, de nome Mac OS X, e apresenta também o revolucionário iPod, com o serviço musical on-line “Tunes Music Store” associado.

Como pôde ser observado, em pouco mais de cinqüenta anos os computadores tiveram um grande desenvolvimento. Iniciaram-se na década de quarenta empregando válvulas, ocupando grande espaço físico e consumindo muita energia (ENIAC continha 18000 válvulas, pesava 28 toneladas e ocupava uma área de 167 m^2, consumia 178 kwh de energia) para os computadores atuais como os notebooks que pesam menos de 2,0 kg, ocupam alguns centímetros quadrados de espaço e consomem alguns watts de energia.

Alguns autores classificam os computadores em gerações. Os computadores da primeira geração (1930-1958) baseavam-se na utilização de válvulas e relés. Com a invenção do transistor (1948) os computadores entraram em sua segunda geração (1955-1965). A terceira geração (1965-1980) de computadores passou a empregar os circuitos eletrônicos (chips) em substituição aos transistores. Houve considerável aumento na velocidade de processamento destes computadores. Por fim, surgiram os computadores de quarta geração (1980-atual) devido ao grande desenvolvimento dos circuitos integrados.

1.4 Componentes de um computador

Independente da marca e modelo, seja um *desktop* (computador de mesa) ou um *notebook*, o computador (hardware) é composto dos seguintes componentes básicos[1] (Figura 1.2):

- monitor de vídeo;
- placa mãe (do inglês motherboard);
- processador;
- memória RAM;

1 Pode haver mais componentes; menciona-se aqui apenas o conjunto principal de componentes de um computador.

- placas de rede, som, vídeo, fax;
- fonte de alimentação (energia);
- leitor/gravador de CDs e/ou DVDs;
- memória secundária (disco rígido ou HD, abreviação do inglês de Hard Disk);
- teclado;
- mouse;

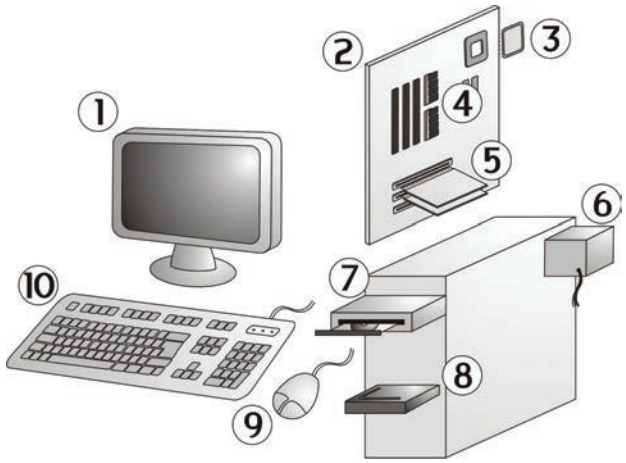


Figura 1.2 Elementos básicos de um computador pessoal.

Legenda: 1) Monitor de vídeo; 2) Placa-Mãe; 3) Processador; 4) Memória RAM; 5) Placas de Rede, Som, Vídeo, Fax; 6) Fonte de Energia; 7) Leitor/gravador de CDs e/ou DVDs; 8) Disco Rígido (HD, Hard Disk); 9) Mouse; 10) Teclado.

As funções básicas de cada um destes componentes são:

- Monitor de vídeo: dispositivo de saída de informações. Possui os mais variados tamanhos, sendo as dimensões de 14 e 15 polegadas as mais usuais.
- Placa mãe: placa de circuito impresso que contém conjunto de componentes eletrônicos de um computador (*chipset*: conjunto de chips eletrônicos que se constitui no cérebro da placa mãe; controladores de disco rígido; barramentos de alta velocidade como o AGP para vídeo e a PCI Express (em versões mais atuais); portas USB, paralela, PS/2, serial e barramentos PCI e ISA, sendo este último não mais usado em placas-mãe modernas.
- Processador: circuito integrado que contém a CPU (em inglês “Central Processing Unit” traduzido como Unidade Central de Processamento). É a parte do computador que interpreta e leva as instruções contidas

nos programas (software). Na maioria das CPUs, essa tarefa é dividida entre uma unidade de controle que dirige o fluxo do programa e uma ou mais unidades de execução que executam operações em dados. Quase sempre, uma coleção de registros é incluída para manter os operadores e intermediar os resultados. Quando cada parte de uma CPU está fisicamente em um único chip de circuito integrado, ela é chamada de microprocessador. Praticamente todas as CPUs fabricadas hoje são microprocessadores.

- Memória: Memória RAM (em inglês Random Access Memory), ou memória de acesso aleatório. É um tipo de memória que permite a leitura e a escrita, utilizada como memória primária em sistemas eletrônicos digitais. O termo acesso aleatório identifica a capacidade de acesso a qualquer posição em qualquer momento, por oposição ao acesso seqüencial, imposto por alguns dispositivos de armazenamento, como fitas magnéticas. O nome da Memória RAM não é verdadeiramente apropriado, já que outros tipos de memória (ROM, etc...) também permitem o acesso aleatório a seu conteúdo. O nome mais apropriado seria Memória de Leitura e Escrita. Apesar do conceito de memória de acesso aleatório ser bastante amplo, atualmente o termo é usado apenas para definir um dispositivo eletrônico que o implementa, basicamente um tipo específico de chip. Nesse caso, também fica implícito que é uma memória volátil, isto é, todo o seu conteúdo é perdido quando a alimentação (fornecimento de energia) é desligada. Algumas memórias RAM necessitam que os seus dados sejam freqüentemente atualizados, podendo então ser designadas por DRAM (Dynamic RAM) ou RAM Dinâmica. Por oposição, aquelas que não necessitam de refrescamento são normalmente designadas por SRAM (Static RAM) ou RAM Estática. Do ponto de vista da sua forma física, uma memória RAM pode ser constituída por um circuito integrado DIP ou por um módulo SIMM, DIMM, SO-DIMM, etc. Para os computadores pessoais elas são normalmente adquiridas em pentes de memória, que são placas de circuito impresso que já contêm várias memórias montadas e configuradas de acordo com a arquitetura usada na máquina. A capacidade de uma memória é medida em bytes, kilobytes (1 kB = 1024 ou 2^{10} bytes), megabytes (1 MB = 1024 kB ou 2^{20} bytes) ou gigabytes (1 GB = 1024 MB ou 2^{30} bytes). A velocidade de funcionamento de uma memória é medida em Hz ou MHz. Este valor está relacionado com a quantidade de blocos de dados que podem ser transferidos durante um segundo. Existem, no entanto, algumas memórias RAM que podem efetuar duas transferências de dados no mesmo ciclo de relógio, duplicando a taxa de transferência de informação para a mesma freqüência de trabalho. Além disso,

a colocação das memórias em paralelo (propriedade da arquitetura de certos sistemas) permite multiplicar a velocidade aparente da memória.

- Placas de rede, fax, som, vídeo: placas de circuito impresso que contêm dispositivos eletrônicos que possibilitam o envio de dados, de áudio, de imagens (placa de vídeo) para o monitor, respectivamente.
- Fonte de alimentação: dispositivo responsável por fornecer a energia necessária para o funcionamento do computador (todos os componentes físicos).
- Memória secundária: a memória secundária ou também denominada memória de massa é empregada para gravar grande quantidade de dados. Estes não são perdidos com o desligamento do computador. Exemplos deste tipo de memória são: HD (do inglês Hard Disk, traduzido como disco rígido), as mídias removíveis como o CD-ROM, o DVD, o disquete (1.44 Mb) e o pen drive.
- Teclado: dispositivo que contém teclas (letras, números, símbolos e outras funções), similar ao modelo de teclado das antigas máquinas de escrever. Permite o envio de informações para os dispositivos de processamento de um computador. As teclas são ligadas a um *chip* dentro do teclado, o qual identifica a tecla pressionada e manda para o PC as informações. O meio de transporte dessas informações entre o teclado e o computador pode ser sem fio (ou Wireless) ou a cabo (PS/2 e USB). O teclado vem se adaptando com a tecnologia e é um dos poucos periféricos que mais se destacam na computação.
- Mouse: dispositivo de entrada para terminal de computador, cuja forma encaixa na palma da mão do operador e é rolado sobre uma superfície lisa, movendo, correspondentemente, o cursor na tela, enviando sinais por um fio de conexão. O seu nome vem do inglês devido à semelhança com o rato.

1.5 Arquitetura básica de um computador

O computador é uma máquina que processa informações. É formado por um conjunto de componentes físicos (dispositivos mecânicos, magnéticos, elétricos ou eletrônicos, ópticos) cujo nome genérico é “hardware”. Este conjunto básico de elementos foi apresentado no item 1.2.

Para o funcionamento da parte física (hardware) são necessários métodos e procedimentos, regras e documentação, que se denominam genericamente de programas ou “softwares”.

Pelo fato dos computadores digitais variarem bastante em organização e características operacionais, seus componentes básicos podem ser representados em termos de suas unidades funcionais, ilustradas na Figura 1.3.

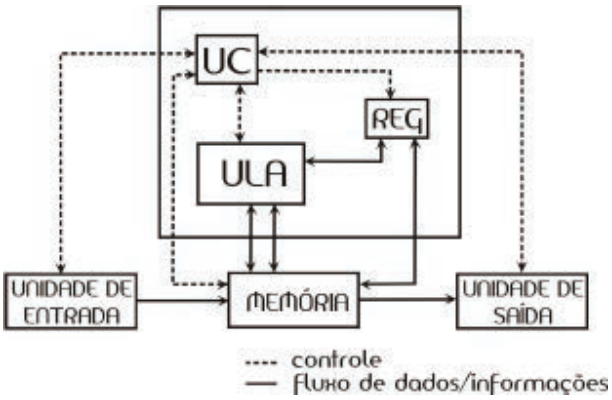


Figura 1.3 Arquitetura básica de um computador.

(I) Unidades de Entrada / Saída (E/S)

As unidades de E/S fornecem os meios de comunicação entre o computador e o meio externo (o usuário, por exemplo).

(II) Memória (unidade de armazenamento)

Uma unidade de armazenamento tem dois propósitos: (a) conter os programas ou conjunto de instruções que controlam as operações do computador; (b) armazenar os dados, que serão manipulados pelo mesmo. Estes dados podem ser de entrada, de saída ou resultados parciais de certo processamento.

As unidades de armazenamento são normalmente classificadas em uma das seguintes categorias:

- Memória Principal: formada por unidades do mesmo tamanho chamadas palavras, sendo que cada uma delas tem um endereço único e distinto. A capacidade de armazenamento de uma memória é medida em Quilo bytes (kb), em que kb significa $2^{10} = 1024$ (devido ao avanço tecnológico, que vem possibilitando cada vez mais o aumento da capacidade de memória, atualmente ela é quantificada em Mega bytes (Mb), em que 1 Mb significa 10^6 kb).
- Memória Secundária: unidades que permitem uma capacidade para aquisição e recuperação de dados a uma velocidade não tão elevada quanto à memória principal, mas capaz de armazenar muito mais informações.

Unidades de memória típicas são as de fita magnética, que armazenam informações gravadas magneticamente em uma fita e as de disco magnético (conhecidas como HD's, capazes de armazenar Giga bytes (Gb) de informações).

(III) Unidade Central de Processamento (UCP) / Control Processing Unity (CPU)

Realiza o controle e a execução de todo o processamento: faz cálculos, toma decisões, controla as E/S, etc. É composta de:

- ULA (Unidade Lógica Aritmética): dados são levados a esta unidade e são manipulados aritmeticamente por processos tais como: adição, subtração, multiplicação e divisão. Operações lógicas são também processadas nesta unidade, tais como teste de certas condições, comparação de diferentes partes de um dado, etc.
- UC (Unidade de Controle): controla todas as ações realizadas pela máquina. Opera sob o controle das instruções do programa armazenado na memória principal.
- Registradores (REG): são registros capazes de armazenar dados ou instruções úteis ao controle do processamento e às operações da ULA.

1.5.1 A informação armazenada no computador

Os computadores modernos utilizam o sistema de contagem binário. Isso significa que os computadores operam com sinais de dois níveis, ou seja, "ligado" ou "desligado". Como já foi mencionado, os computadores modernos baseiam-se na utilização de *chips* (unidades que contêm os transistores). Esses elementos, os transistores, possuem a capacidade de trabalhar como uma chave do tipo "liga-desliga", de forma a poder representar dois estados, representados pelos dígitos 0 e 1. Mas podemos questionar: porque os computadores não utilizam o sistema decimal (o qual estamos habituados a empregar em nosso dia a dia) em seus circuitos eletrônicos? A resposta a esta pergunta está no fato de que a solução empregando a lógica binária é simples e de baixo custo.

A Tabela 1.1 apresenta os algarismos dos sistemas decimal e binário.

Tabela 1.1 Algarismos dos sistemas decimal e binário.

Decimal		Binário	
Zero	0	Zero	0
Um	1	Um	1
Dois	2		
Três	3		
Quatro	4		
Cinco	5		
Seis	6		
Sete	7		
Oito	8		
Nove	9		

Nos dois sistemas, decimal e binário, a combinação de dois dígitos representa um número que é maior que o maior número formado por um dígito. O valor numérico da combinação dos dígitos um e zero (10) é importante. Essa combinação tem o valor dez (10) no sistema decimal (dá nome ao sistema) e tem o valor dois (2) no sistema binário (também dando nome ao sistema). Esses valores, 10 para o decimal e 2 para o binário, são chamados de base do sistema numérico.

A base de números, ou seja, o valor associado com a combinação dos dígitos 1 e 0 é interpretada para os números decimais e binários, da seguinte forma:

$$\begin{array}{cc} 1 & 0 \\ \uparrow & \uparrow \end{array}$$

$$1\times \text{base} \; + \; 0\times \text{base}^0$$

$$\text{Sistema decimal (base 10)}: 1\times 1c^1 + c\times 10^0 = 10$$

$$\text{Sistema binário (base 2)}: 1\times 2^1 + 0\times 2^0 = 2$$

Em qualquer sistema numérico (existem outros sistemas além do binário e decimal) o maior número que podemos representar empregando N dígitos corresponderá a $(\text{base}^N - 1)$. Ou seja, no sistema binário (emprega dois dígitos) o maior número a ser representado será o número 3 $(2^2 - 1 = 3)$. Já no sistema decimal, o maior número a ser representado será o número 99 $(10^2 - 1 = 99)$, Tabela 1.2.

Tabela 1.2 Representação de números no sistema binário e decimal utilizando dois dígitos (N = 2).

Binário (base = 2)		**Decimal (base = 10)**	
Maior número: $base^N - 1 = 2^2 - 1 = 3$		Maior número: $base^N - 1 = 10^2 - 1 = 99$	
00	Zero	0	Zero
01	Um	1	Um
10	Dois	...	...
11	Três	10	Dez
		11	Onze
		...	...
		99	Noventa e nove

Depreende-se da Tabela 1.2 que para representar a mesma quantidade o sistema binário irá precisar de um número maior de dígitos. Por exemplo, no caso exemplificado, para poder representar o número 99 o sistema binário precisaria de 7 dígitos. Nesse caso o maior número representado seria o número 127 ($2^7 - 1$). Com 6 dígitos ainda não seria possível representar o número 99 (verifique!).

1.5.2 Os termos: bit, byte e palavra (word)

A palavra bit (em inglês binary digit, traduzido como dígito binário) representa o menor item de informação binária. O bit é um dígito que pode assumir um dos dois valores ou estados diferentes 0 ou 1 (tal como um dígito decimal pode assumir um dos dez valores 0, 1, 2, 3, 4, 5, 6, 7, 8, 9). A convenção estipula que um bit “ligado” ou “ativado” é representado por 1 e um bit “desligado” ou “desativado” por 0. O bit representa um estado (“ligado” ou “desligado”) nos dispositivos eletrônicos digitais como as memórias e circuitos de um computador.

Podemos dizer que o bit é a unidade fundamental de informação de um computador. Contudo, ele por si só não teria muita utilidade. Dessa forma, nos circuitos eletrônicos dos computadores os bits são agrupados de modo a possibilitar a representação das informações. Surge então outro termo: o byte. O byte representa um conjunto de bits. O número de bits que formam um byte não é fixo e dependerá da arquitetura do computador. Usualmente 1 byte tem 8 bits. Na prática, hoje em dia se trabalha com os múltiplos do byte (Tabela 1.3).

Tabela 1.3: O byte e seus múltiplos (com base na IEC)*.

Nome	Valor
Quilobyte	$2^{10} = 1.024$ bytes
Megabyte	$2^{20} = 1.048.576$ bytes
Gigabyte	$2^{30} = 1073.741.824$ bytes
Terabyte	$2^{40} = 1.099.511.627.776$ bytes

IEC – International Electrotechnical Commission ou Comissão eletrotécnico internacional (organização internacional de padronização de tecnologias elétricas, eletrônicas e relacionadas).

O termo palavra (word) refere-se a quantidade de bits que pode ser manipulada em conjunto, ou seja, acessada de uma só vez. Esse número pode variar dependendo da arquitetura do computador. A maioria dos computadores pessoais utiliza em sua arquitetura 32 bits.

Quando uma palavra é empregada para representar um caractere (por exemplo, um número), o esquema de codificação binária é normalmente empregado. Cada bit é visto individualmente dentro da palavra e o valor decimal representado é obtido a partir da regra de conversão binário para decimal. Por exemplo, o número decimal 67 seria representado por 00100011 em uma palavra de 8 bits. Verificando:

$$0 \cdot 2^8 + 0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^{10} = 67$$

Uma palavra pode representar um número ou algum caractere de informação não numérica. Existem diferentes sistemas de codificação (binária, BCD, ASCII, EBCDIC, por exemplo).

Os dois sistemas de codificação mais empregados atualmente são o EBCDIC (Extended Binary Coded Decimal Interchange Code) e o ASCII (American Standard Code for Information Interchange). As tabelas com os códigos EBCDIC e ASCII podem ser encontradas no endereço eletrônico: <http://www.asciitable.com/>.

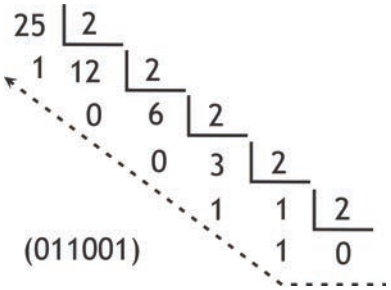
Conversão de números: binário x decimal e decimal x binário

Aprenderemos como converter um número decimal (base 10) para binário (base 2). Tomemos como exemplo o número 25,375.

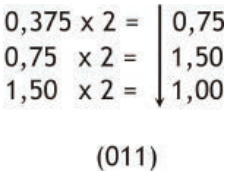
$$(25{,}375)_{10} \leftarrow (\,?\,)_2$$

Solução: A conversão é realizada em duas etapas.

- Primeira etapa: realiza-se a conversão da parte inteira do número:



- Segunda etapa: realiza-se a conversão da parte fracionária do número:



Observação: as multiplicações sucessivas continuam até que um produto igual a 1 seja obtido. Contudo, em muitas vezes o resultado 1 não será atingido finitamente. Nesses casos o processo pára quando um número conveniente de casas do número fracionário binário é alcançado.

Resposta: $(25{,}375)_{10} \leftarrow (\ 011001{,}011\ )_2$

Regra: Transformação de um número da base 10 (decimal) para uma base b qualquer.

Conforme ilustrado no exemplo, faremos uma generalização para conversão de base. Dado um número escrito na base 10 (decimal), para convertê-lo para base b qualquer a parte inteira do número deve ser dividida pela base b tantas vezes quantas forem necessárias até que o quociente de divisão seja menor que a base b. O último quociente e os restos das divisões sucessivas tomados na ordem inversa que foram obtidos correspondem ao respectivo número na nova base b. A parte fracionária é convertida multiplicando-se seu valor pela base até a obtenção do valor exato da base (muitas vezes isso não é conseguido e o processo pára quando um número conveniente de casas do número fracionário binário é alcançado).

Vejamos agora como converter um número da base 2 (binária) para a base 10. Tomemos como exemplo o número decimal representado com 6 bits: 001010,101.

$(001010,101)_2 \leftarrow (\ ?\ )_{10}$

Solução: A conversão é realizada em duas etapas.

- Primeira etapa: realiza-se a conversão da parte inteira do número. Para isso multiplica-se cada algarismo por b^n (n = 0, 1, 2, 5):

$$001010 \rightarrow 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 10$$

- Segunda etapa: realiza-se a conversão da parte fracionária do número:

$$101 \leftarrow 1 \cdot \frac{1}{2^1} + 0 \cdot \frac{1}{2^2} + 1 \cdot \frac{1}{2^3} = 0{,}625$$

Resposta: $(001010,101)_2 \leftarrow (10{,}625)_{10}$

Regra: Transformação de um número em qualquer base para base 10 (decimal).

Conforme foi ilustrado no exemplo, faremos uma generalização para conversão de base. Dado um número M escrito em qualquer base ($M = x_n\ x_{n-1} \ldots x_0, y_1\ y_2 \ldots y_m$) sendo:

- $x_n\ x_{n-1} \ldots x_0$: algarismos da parte inteira do número;
- $y_1\ y_2 \ldots y_m$: algarismos da parte fracionária do número.

Para encontrar o equivalente no sistema decimal basta multiplicar cada algarismo da parte inteira, x_i (i = n, n – 1, ...1,0) por b^n.

Para a parte fracionária y_j (j = 1, 2, ..., m) por $\frac{1}{b^m}$ e somar os produtos obtidos.

Os sistemas de numeração mais conhecidos, além do binário e decimal, são os sistemas octal (emprega os algarismo de 0 a 7) e o sistema hexadecimal (base 16,

emprega 16 símbolos para representar seus dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F).

1.6 Linguagem de máquina, de montagem, de alto nível e sistema operacional

Os computadores só podem executar diretamente os algoritmos expressos em linguagem de máquina, ou seja, por um conjunto de instruções capazes de ativar diretamente os dispositivos eletrônicos do computador.

Desvantagem desta linguagem:

- diferente para cada tipo de computador, pois depende da arquitetura da máquina;
- difícil de programar nesta linguagem;
- totalmente expressa em forma numérica (binária ou hexadecimal), o que a torna pouco expressiva (difícil de interpretar).

Para facilitar a programação, foi introduzida a possibilidade de programar em uma linguagem em que as instruções da máquina são representadas por meio de mnemônicos – linguagem "Assembly" ou de montagem. A tradução desta linguagem para linguagem de máquina é realizada por programas tradutores (escritos em linguagem de máquina) denominados "Assemblers" ou montadores. Em geral, um mnemônico corresponde a uma única instrução de máquina.

O sucesso da linguagem de montagem motivou os pesquisadores a desenvolverem linguagens mais poderosas – em que um único comando poderia corresponder a mais de uma instrução de máquina – em que a programação é feita por meio de uma notação matemática e de algumas palavras da linguagem falada – linguagem de alto nível.

A tradução de um programa escrito em linguagem de alto nível para a linguagem de máquina é realizada por um programa tradutor chamado compilador.

A linguagem FORTRAN foi a primeira linguagem de programação de alto nível a ser proposta (1956). Foi sugerida visando à resolução de problemas da área científica. Seu nome vem da composição da palavra FORmula TRANslation. Esta linguagem tornou-se muito utilizada na área científica e ainda hoje muitos programas empregam rotinas programadas em FORTRAN.

Além da grande facilidade, uma imensa vantagem de se escrever os programas em linguagem de alto nível, é a sua quase total independência da má-

quina a ser usada. Um programa escrito em linguagem de alto nível, geralmente com pouca alteração, é aceito por qualquer computador.

Como o FORTRAN se mostrou mais adequado à programação de natureza técnica e científica, logo surgiu a idéia de se criar uma linguagem mais voltada para problemas de natureza comercial e administrativa.

Em 1959, surgiu o COBOL (Common Business Oriented Language). A linguagem COBOL facilita que um programa seja escrito de uma forma mais próxima das linguagens naturais (no caso, inglês), facilitando seu entendimento por pessoas e permitindo uma documentação mais clara.

Com o FORTRAN na área técnica e científica e o COBOL na área comercial e administrativa, surgiu por volta de 1963, a idéia de se criar uma linguagem única que fosse apropriada para todas as áreas de aplicação – surgiu o PL/1 (Programming Language One). PL/1: linguagem extremamente vasta com numerosos recursos. O PL/1 exige um compilador muito complexo e seu aprendizado completo é longo e trabalhoso.

Com o objetivo de permitir aos não especialistas a utilização de computadores, foi criada por volta de 1964 a linguagem BASIC (Beginner's All-Purpose Symbolic Instructions Code).

Para evitar muitas das limitações da linguagem FORTRAN e para permitir uma melhor expressão dos algoritmos, foi criada em 1960 a linguagem ALGOL (Algorithm Language), bastante difundida na Europa. A linguagem ALGOL é muito ampla e o compilador correspondente só era disponível para poucos computadores.

A partir de 1968, N. Winth, em Zurique, desenvolveu uma nova linguagem, a qual chamou de PASCAL. Foi criada para facilitar o ensino de informática e sob muitos aspectos é semelhante ao ALGOL. Sua simplicidade proposital, aliada a uma adequada perfeição lógica, tornou-a bastante difundida.

Existem muitas outras linguagens de programação. Citamos, a título de exemplo algumas: RPG (Report Program Generation); FORTH; C; C++, APL (A Programming Language) ADA; LOGO; ProLog; LISP, Basic, Visual Basic, etc.

A escolha da linguagem de programação depende, antes de tudo, da existência de um programa (que traduza o algoritmo escrito na linguagem escolhida, para a linguagem de máquina – compilador) ou de um programa interpretador (que interprete cada comando do programa e execute uma série de instruções que a ele correspondem).

Existem compiladores ou interpretadores para diversas linguagens. A escolha pode ser em função da preferência do programador, ou em função do tipo de aplicação que se deseja fazer.

Para se resolver um problema em um computador, mais importante que a escolha da linguagem de programação, é o desenvolvimento de um algoritmo adequado.

O algoritmo deve ser desenvolvido objetivando-se, sobretudo, a clareza, e permitindo que os erros cometidos sejam detectados o quanto antes; evitando excessivas revisões; visando facilitar futuras modificações. Este será o tema de nossa próxima unidade.

O sistema operacional (ou do inglês Operating System) é um programa ou um conjunto de programas utilizado como interface entre um computador e seus recursos computacionais (*softwares ou hardwares*) e o usuário. Ele tem o papel de tornar o uso do computador mais amigável para o usuário. O sistema operacional é um gerenciador destes recursos no computador.

O sistema operacional é um intermediário entre o aplicativo (programa voltado para o usuário) e os componentes físicos do computador (*hardware*). Ele é um gerenciador de recursos, na medida em que controla quais aplicações (processos) podem ser executadas e que recursos (memória, disco, periféricos) podem ser utilizados.

Nos computadores de primeira geração (1945 a 1955), os quais eram construídos empregando válvulas e relés, os sistemas operacionais não existiam. A operação destes computadores era realizada por pessoas (operadores), que o controlavam por meio de chaves, fios e luzes de aviso.

Na geração seguinte de computadores (1955 a 1965) surgiram os primeiros sistemas operacionais, denominados de *batch systems* (do inglês, sistemas em lote). Como os computadores eram raros (devido ao seu altíssimo custo), o seu uso era compartilhado por vários pesquisadores. O sistema em lote foi desenvolvido com o objetivo de otimizar os recursos computacionais entre os vários diferentes usuários. Na prática os usuários ficavam afastados do computador, cabendo a eles a tarefa de fornecer ao operador do computador o programa em cartões perfurados. Estes eram carregados, juntamente com o compilador no computador, que por sua vez empregava uma linguagem chamada JCL (Job Control Language).

Citamos alguns exemplos de sistemas operacionais: DOS, UNIX, NETWARE, WINDOWS, LINUX.

Na prática, um computador não teria grande utilidade sem um sistema operacional. Os aplicativos nele instalados (programas instalados no computador como: navegador para internet, pacotes gráficos, leitores de e-mail, editores de textos) precisam de um sistema operacional para funcionar.

1.7 Considerações finais

Nesta unidade apresentamos um breve histórico da evolução dos computadores. A história é recente. Apenas seis décadas se passaram desde a invenção do ENIAC, considerado por muitos o primeiro computador da era moderna. E a evolução foi muita rápida. Ficaram para trás válvulas, relés e transistores. Desde o ano de 1970 a velocidade de processamento dos computadores vem dobrando a cada dois anos. Estamos na era dos chips de silício, e talvez evoluindo para os biochips. Dados recentes disponibilizados por empresa na internet afirmam que o número de computadores pessoais em uso no mundo já ultrapassou um bilhão (http://jc.uol.com.br/2008/06/23/not_172211.php).

Iniciamos no universo dos bits e bytes da informática. Aprendemos que um computador opera no sistema binário (empregando os dígitos zeros e uns para representar os estados “ligado” e “desligado”) e que precisa de um sistema operacional para funcionar.

1.8 Referências bibliográficas

CEREDA, R. L. D., MALDONADO, J. C. *Introdução ao FORTRAN 77 para microcomputadores*. Ed. McGraw-Hill, São Paulo, 1987.

SOUZA, M. A. F., GOMES, M. M., SOARES, M. V., CONCILIO, R. *Algoritmos e Lógica de Programação*. Ed. Pioneira Thomson Learning, São Paulo, 2005.

História-do-computador-e-da-internet, <http://www.algosobre.com.br/informatica/historia-do-computador-e-da-internet.html> [Última consulta em 25 de julho de 2008]

Wikipedia – The Free Encyclopedia, <http://en.wikipedia.org/wiki/ENIAC> [Última consulta em 25 de julho de 2008]

O Computador, <http://www.widesoft.com.br/users/virtual/mcomp.htm> [Última consulta em 25 de julho de 2008]

FERREIRA, A. B. H. *Novo dicionário da língua portuguesa*. 2 ed. Rio de Janeiro: Nova Fronteira, 1986. 1838 p.

Houaiss, ANTONIO; VILLAR, M. S.; FRANCO, F. M. M. *Dicionário Houaiss da língua portuguesa*. Rio de Janeiro: Objetiva, 2001. 2922 p.

Dicionário Michaelis. <http://michaelis.uol.com.br/> [Última consulta em 30 de julho de 2008].

UNIDADE 2

Algoritmos e programação

2.1 Primeiras palavras

Caro aluno(a), nesta unidade abordaremos os tópicos algoritmo e programação. Antes de programar é preciso aprender como se elabora um algoritmo. E por falar em algoritmo e programação, embora tenhamos intuitivamente o conceito destas duas palavras, busquemos as definições em dicionários da língua portuguesa.

Algoritmo: "[Do lat. Med. *algorismos*, *algorithmos*, "algarismo", por infl. do gr. *arithmós*, "número".] S. m. **1.** *Mat.* Processo de cálculo, ou de resolução de um grupo de problemas semelhantes, em que se estipulam, com generalidade e sem restrições, regras formais para a obtenção do resultado, ou da solução do problema. **2.** *Proc. Dados*. Conjunto predeterminado e bem definido de regras e processos destinados à solução de um problema, com um número finito de etapas. (...)."(Ferreira, 1986)

"S.m. (Do Latim algorithmus). Conjunto de regras formais cuja aplicação permite resolver um cálculo ou um problema enunciado por meio de um número finito de operações." (Souza et al., 2004)

Programação: "S. f. **1.** Ato de programar, de estabelecer um programa. (...) **4.** *Proc. Dados.* Elaboração de um programa para um computador. (...)."(Ferreira, 1986)

Programa: "[Do gr. *prográmma*, pelo lat. *programma*.]. S.m. (...) **9.** *Proc. Dados.* Seqüência de etapas que devem ser executadas pelo computador para resolver um problema determinado. (...) Programa de Computador**.** *Proc. Dados.* Seqüência de instruções ou declarações expressas em uma linguagem de programação, com o objetivo de obter um resultado específico. (...)."].(Ferreira, 1986)

Objetivaremos abordar de forma clara e objetiva os procedimentos necessários para resolução de problemas científicos e de engenharia com auxílio do computador. O primeiro passo é a identificação do problema. Em seguida, precisaremos propor uma seqüência de etapas que deverão ser seguidas para resolução do problema em questão. Estas etapas serão o conjunto de instruções do algoritmo. Por fim, precisaremos eleger uma linguagem de programação para implementar o algoritmo desenvolvido.

2.2 Problematizando o tema

A automatização de tarefas é uma característica presente na sociedade moderna. O avanço no desenvolvimento dos computadores em muito contribuiu para esta realidade. As ciências e as engenharias tiveram grande progresso

em razão disto. Hoje, o computador está presente em nosso dia a dia nas mais variadas tarefas, de forma direta ou indireta e muitas vezes nem nos damos conta disso. Quando vamos ao banco e acessamos o caixa eletrônico um computador está presente. Também é possível acessar informações de nossa conta bancária diretamente de um computador pessoal via internet (se o cliente dispuser de permissão para o acesso). Quando conversamos ao telefone celular ou vamos ao supermercado e realizamos uma consulta de preço em terminal apropriado um computador também é utilizado. Em todas essas situações algoritmos complexos foram desenvolvidos por experientes programadores e implementados em alguma linguagem de programação para resolução destas tarefas.

Nesta unidade abordaremos um ponto muito importante: a elaboração de algoritmos e suas formas de representação. A concepção de algoritmos para resolução de problemas com auxílio do computador é um ponto chave nesta disciplina.

2.3 Algoritmos

Os computadores estão sendo utilizados para resolver problemas cada vez de maior porte e complexidade. A utilização do computador para resolver problemas é precedida pela necessidade de se desenvolver um algoritmo.

Um algoritmo pode ser definido como sendo um conjunto de instruções (não ambíguas e finitas) que são executadas até que determinado objetivo seja atingido. A Figura 2.1 ilustra de forma simplificada a estrutura de um algoritmo.

Figura 2.1 Estrutura simplificada de um algoritmo.

Um exemplo de um algoritmo seria os passos descritos em uma receita culinária para elaboração de uma torta ou bolo. Vejamos como ficaria a Figura 2.1 no caso do preparo de um bolo. As entradas seriam os ingredientes necessários, por exemplo, farinha, açúcar, sal, leite, ovos, fermento, manteiga. As instruções são as etapas descritas na receita, ou seja, o modo de preparo: (1) bata a manteiga até ficar cremosa; (2) adicione o açúcar e as gemas; (3) bata até formar um creme; (4) adicione a farinha, o fermento, o sal e o leite; (5) misture

bem; transfira para uma assadeira; (6) leve ao forno por 45 minutos. A saída será o bolo pronto.

A correta execução dos passos contidos na receita, contudo, podem não garantir a obtenção do resultado esperado (o fermento utilizado pode não estar bom, as quantidades adicionadas dos componentes não foram corretas, por exemplo).

Além do mais, um algoritmo corretamente executado não irá solucionar um problema se estiver implementado de forma incorreta ou ainda se não for apropriado ao problema. No caso do preparo do bolo, as instruções precisam ser seguidas em uma ordem pré-definida. Não há como levar a mistura ao forno sem antes prepará-la. Em alguns casos, a inversão na ordem de algumas instruções poderá dificultar a solução do problema, ou mesmo levar a algum resultado não esperado (incorreto ou inconsistente).

Vejamos mais um exemplo simples. Imagine um algoritmo para uma pessoa se vestir. As entradas poderiam ser: meias, calça, camisa e sapato. As instruções seriam: (a) vestir meias; (b) vestir sapato; (c) vestir calça; (d) vestir camisa. O objetivo final é a pessoa vestida. Nesse caso, independente da ordem das instruções, o objetivo sempre será alcançado. Contudo, a depender da seqüência das instruções o resultado será diferente em termos da dificuldade para realização e do tempo de solução. Compare estas diferentes seqüências de instruções:

1. (a) vestir meias; (b) vestir sapato; (c) vestir calça; (d) vestir camisa;
2. (d) vestir camisa; (c) vestir calça; (a) vestir meias; (b) vestir sapato;
3. (a) vestir meias; (d) vestir camisa; (c) vestir calça; (b) vestir sapato.

Podemos chegar à conclusão que as três propostas apresentam diferentes graus de dificuldade (vestir a calça tendo o sapato já calçado é mais difícil que vestir a calça sem ter ainda os sapatos calçados). Contudo, o resultado final será o mesmo.

Todo algoritmo deveria apresentar cinco características importantes:

- *Entradas*: o algoritmo precisa possuir entradas, ou seja, receber as informações que são necessárias para sua execução.
- *Saídas*: o algoritmo deve ter uma ou mais saídas, que é (são) o(s) objetivo(s) atingido(s).
- *Finitude*: o algoritmo deve terminar após um número finito de passos.
- *Instruções elementares*: cada instrução deve ser claramente definida (sem ambigüidades).

- *Efetividade*: as instruções devem, em princípio, ser executadas com precisão empregando papel e lápis.

2.3.1 Como representar os algoritmos?

Existem várias formas de representação para os algoritmos. Vejamos alguns exemplos:

- *Linguagem natural ou algoritmo informal*: os algoritmos são expressos diretamente na linguagem escrita. Vejamos o exemplo de algoritmo para substituição de uma lâmpada queimada.
- *Entradas*: escada, lâmpada

  Instruções:

  1. posicionar a escada embaixo da lâmpada;
  2. subir na escada até alcançar a lâmpada;
  3. remover a lâmpada queimada do bocal;
  4. rosquear a nova lâmpada no bocal;
  5. descer da escada;
- *Saída (objetivo)*: lâmpada substituída, iluminação reestabelecida.

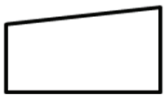
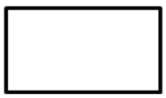
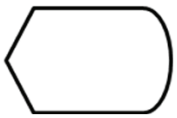
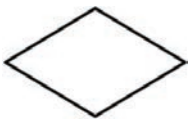
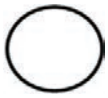
- *Fluxograma convencional* ou *Fluxograma*: os algoritmos são representados empregando formas geométricas padronizadas para indicar as diferentes instruções e decisões que devem ser executadas para resolver o problema. O fluxograma foi uma das primeiras ferramentas utilizadas para representar algoritmos. Veremos exemplo mais adiante.
- *Pseudo-código*: esta forma de representação emprega uma linguagem intermediária entre a linguagem natural e a linguagem de programação para representar os algoritmos. São constituídas por um vocabulário de uma linguagem natural (português, inglês, etc) e pela sintaxe de uma linguagem de programação. Quando os termos empregados estão em Português alguns autores os chamam de **Portugol**.

Existem ainda outras ferramentas utilizadas para representar os algoritmos. Dentre elas podemos citar os diagramas de Nassi-Shneiderman (cartas N-S), as tabelas de decisão, os diagramas de ação, os diagramas de Warnier, os diagramas de Jackson, por exemplo.

2.3.2 Fluxogramas

A idéia básica desta forma de representação dos algoritmos é empregar figuras geométricas na representação de cada instrução (ou passo) que compõe o algoritmo. Existe um conjunto específico de símbolos empregados na elaboração dos fluxogramas. Estes símbolos são definidos pela norma ISO 5807 (ISO, International Organization for Standardization). Apresentam-se na Tabela 2.1 os mais importantes.

Tabela 2.1 Simbologia gráfica empregada na elaboração de fluxogramas. Fonte: Adaptado da norma ISO 5807

Símbolo	Nome	Finalidade
	Terminal	Utilizado para representar o início e o fim do fluxo lógico de um programa. Empregado também na definição de sub-rotinas de procedimento ou função.
	Entrada manual	Utilizado para representar a entrada manual de dados, via de regra pelo teclado do computador.
	Processamento	Utilizado para representar a execução de uma operação ou grupo de instruções que estabelecem o resultado de uma operação lógica ou matemática.
	Exibição	Utilizado para representar a execução da operação de saída visual de dados em um monitor de vídeo conectado ao computador.
	Documento	Utilizado para representar a execução da operação de saída de dados em um documento emitido por uma impressora na forma de relatório.
	Decisão	Utilizado para representar o uso de desvios condicionais para outros pontos do programa de acordo com situações variáveis.
	Conector	Utilizado para representar a entrada ou saída em outra parte do diagrama de blocos. Pode ser usado na definição de quebras de linha e na continuação da execução de decisões.
	Processo pré-definido	Utilizado para representar um grupo de operações estabelecidas como uma sub-rotina de processamento anexa ao diagrama de blocos (referência a um subprograma externo).
	Linha (com seta)	Utilizado para representar o vínculo existente entre os vários símbolos de um diagrama de blocos. Indica o sentido de fluxo de execução.

2.3.3 Exemplos de fluxogramas

Apresentaremos alguns exemplos de fluxogramas implementados utilizando a norma ISO 5807.

Exemplo 1: Elaborar algoritmo para calcular a área de um triângulo (area) empregando a fórmula de Herão (ou de Heron). São fornecidos os comprimentos dos três lados do triângulo (a, b, c).

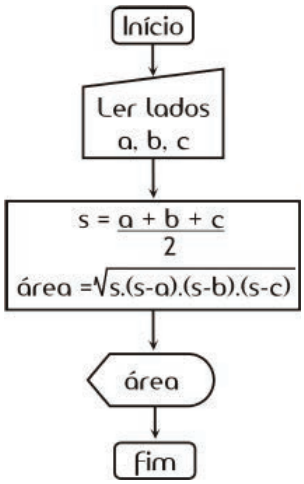
Fórmula para cálculo da área: $\text{area} = \sqrt{s \cdot (s-a) \cdot (s-b) \cdot (s-c)}$

em que s é o semi-perímetro do triângulo, calculado por:

$$s = \frac{(a+b+c)}{2}$$

Observe que a ordem de cálculo é muito importante para resolução deste problema. Primeiro é preciso calcular o perímetro (s) e em seguida a área (area).

Solução:



Exemplo 2: Elaborar algoritmo para o cálculo da média final (MF) dos alunos em uma disciplina do curso de Engenharia Ambiental. Nesta disciplina os alunos realizaram duas provas (P1 e P2) e um trabalho (T), e o seguinte critério de avaliação foi estabelecido pelo professor:

P1 – nota da primeira avaliação

P2 – nota da segunda avaliação

T – nota do trabalho realizado em equipe

MF – média final

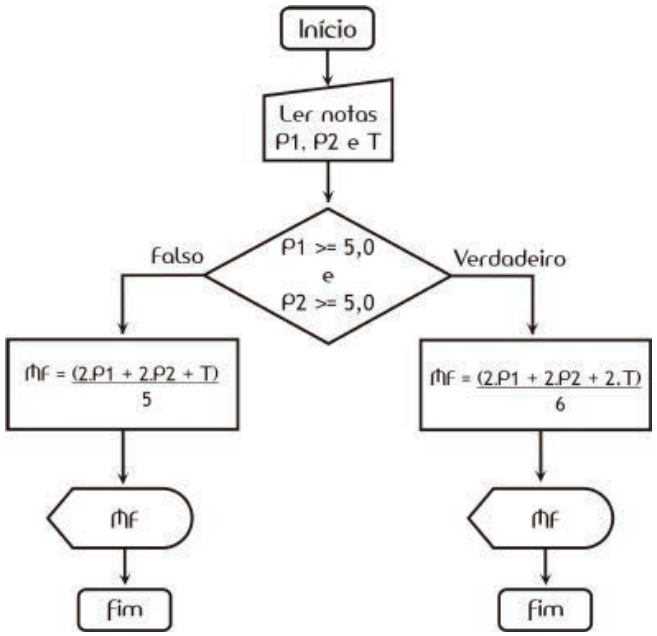
Se P1 ≥ 5,0 e P2 ≥ 5,0 a média final é calculada por:

$$MF = \frac{(2 \cdot P1 + 2 \cdot P2 + 2 \cdot T)}{6}$$

Caso contrário:

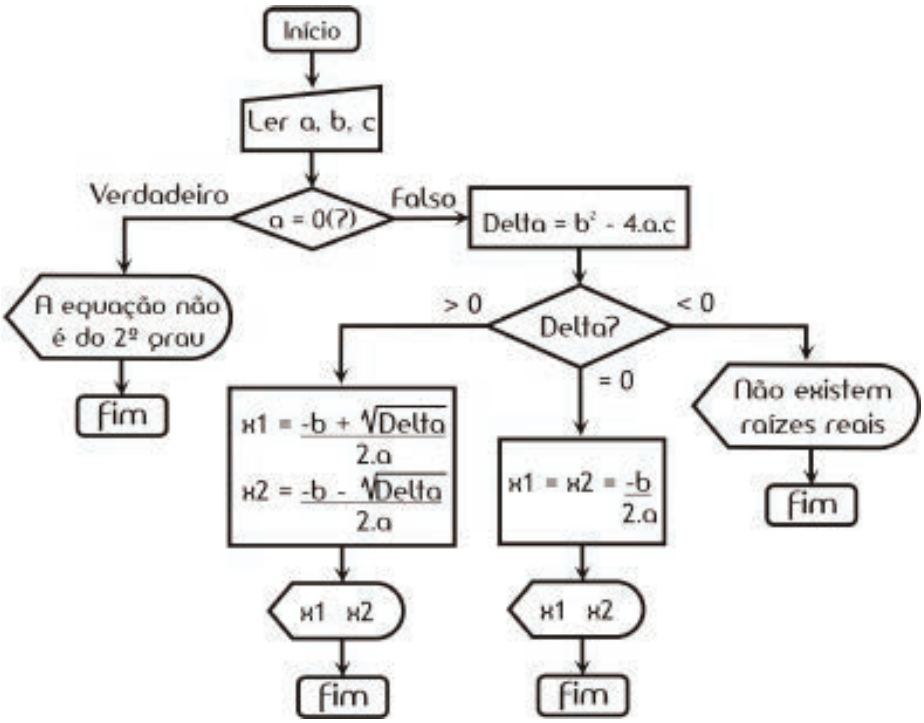
$$MF = \frac{(2 \cdot P1 + 2 \cdot P2 + T)}{5}$$

Solução:



Exemplo 3: Formular algoritmo para calcular as raízes (x1 e x2) de uma equação do segundo grau $(a \cdot x_2 + b \cdot x + c)$ a partir dos valores das constantes a, b e c fornecidas pelo usuário. O algoritmo deve utilizar a fórmula de Báskara.

Solução:



Observação: Este algoritmo poderia ser implementado para prever o cálculo das raízes complexas da equação.

Os fluxogramas são elaborados tendo como base três tipos de estruturas de programação. São elas: estruturas seqüenciais, estruturas de decisão e estruturas de repetição. Nos exemplos até aqui apresentados foram utilizados apenas dois tipos destas estruturas: seqüenciais e de decisão.

Estruturas seqüenciais: este tipo de estrutura é representado pela conexão de dois ou mais símbolos de processamento conectados por uma linha (ou seta), ou seja, indicando o sentido do fluxo de execução. A Figura 2.2 ilustra um exemplo deste tipo de estrutura.

Figura 2.2 Representação de uma estrutura seqüencial.

Estruturas de seleção: este tipo de estrutura é representado pelo uso combinado dos símbolos de decisão, de processamento e de indicação do sentido do fluxo de execução. A Figura 2.3 ilustra um exemplo deste tipo de estrutura.

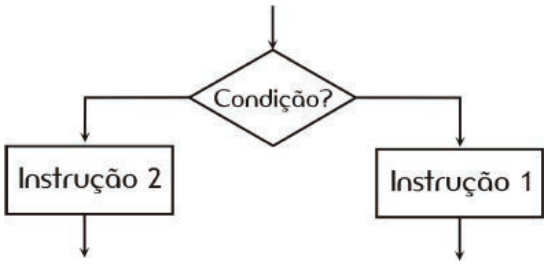


Figura 2.3 Representação de uma estrutura de seleção.

Estruturas de iteração: este tipo de estrutura também é representado pelo uso combinado dos símbolos de decisão, de processamento e de indicação do sentido do fluxo de execução. A Figura 2.4 ilustra um exemplo deste tipo de estrutura.

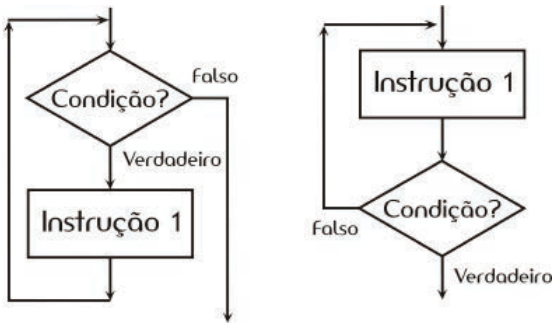


Figura 2.4 Representação de uma estrutura de iteração.

2.3.4 Exemplos de pseudo-códigos

Neste item apresentaremos os exemplos já implementados na forma de pseudo-código.

Exemplo 4: Implementar na forma de pseudo-código algoritmo para calcular a área de um triângulo (area) empregando a fórmula de Herão (ou de Heron). São fornecidos os comprimentos dos três lados do triângulo (a, b, c).

Fórmula para cálculo da área:

$$\text{área} = \sqrt{s \cdot (s-a) \cdot (s-b) \cdot (s-c)}$$

em que s é o semi-perímetro do triângulo, calculado por:

$$s = \frac{(a+b+c)}{2}$$

*Observe que a ordem de cálculo é muito importante para resolução deste problema. Primeiro é preciso calcular o perímetro (s) e em seguida a área (area).

Solução:

Pseudo-código do algoritmo
Início
Ler (a, b, c)
$s \leftarrow (a+b+c)/2$
area $\leftarrow \sqrt{s\cdot(s-a)\cdot(s-b)\cdot(s-c)}$
Exibir (area)
Fim

Exemplo 5: Implementar na forma de pseudo-código algoritmo para o cálculo da média final (MF) dos alunos em uma disciplina do curso de Engenharia Ambiental. Nesta disciplina os alunos realizaram duas provas (P1 e P2) e um trabalho (T), e o seguinte critério de avaliação foi estabelecido pelo professor:

P1 – nota da primeira avaliação

P2 – nota da segunda avaliação

T – nota do trabalho realizado em equipe

MF – média final

Se P1 $\geq$ 5,0 e P2 $\geq$ 5,0 a média final é calculada por:

$$MF = \frac{(2\cdot P1 + 2\cdot P2 + 2\cdot T)}{6}$$

Caso contrário:

$$MF = \frac{(2\cdot P1 + 2\cdot P2 + T)}{5}$$

Solução:

Pseudo-código do algoritmo
Início
Ler (P1, P2, T)
Se P1 ≥ 5,0 e P2 ≥ 5,0 **Então**
MF ← $(2 \cdot P1 + 2 \cdot P2 + 2 \cdot T)/6$
Senão
MF ← $(2 \cdot P1 + 2 \cdot P2 + T)/5$
Fim Se
Exibir (MF)
Fim

Exemplo 6: Formular algoritmo para calcular as raízes (x1 e x2) de uma equação do segundo grau ($ax^2 + bx + c$) a partir dos valores das constantes a, b e c fornecidas pelo usuário. O algoritmo deve utilizar a fórmula de Báskara.

Solução:

Pseudo-código do algoritmo
Início
Ler (a, b, c)
Se a = 0 **Então**
Exibir (“A equação não é do 2º grau”)
Fim
Fim Se
Delta ← $b^2 - 4 \cdot a \cdot c$
Se Delta < 0 **Então**
Exibir (“Não existem raízes reais”)
Fim
Fim Se

Se Delta = 0 **Então**

$$x1 \leftarrow \frac{-b}{2 \cdot a}$$

$x2 \leftarrow x1$

Senão

$$x1 \leftarrow \frac{-b + \sqrt{Delta}}{2 \cdot a}$$

$$x2 \leftarrow \frac{-b - \sqrt{Delta}}{2 \cdot a}$$

Fim Se

Exibir (x1, x2)

Fim

Observação: A estrutura **Se** é finalizada com o termo **Fim Se** para evitar confusões.

2.4 Estruturas de programação

A maioria das linguagens de programação possuem um conjunto de estruturas padrão, o que torna mais fácil implementação de um programa.

Já vimos que os fluxogramas são elaborados com base em três tipos de estruturas, que são:

- Estruturas seqüenciais;
- Estruturas de decisão ou seleção;
- Estruturas de repetição ou iteração.

Estas estruturas recebem o nome de estruturas de controle da programação. Podemos dizer que qualquer programa de computador pode ser representado combinando-se estas três estruturas de programação.

2.4.1 Estruturas sequenciais

Este tipo de estrutura representa os comandos que são executados pelo computador de forma seqüencial, ou seja, na ordem em que são apresentados. A leitura de dados, atribuições de valores a variáveis, cálculos, chamadas de funções, saída de resultados são exemplos dessas estruturas. Na Figura 2.5 apresentamos, na forma de fluxograma, algoritmo desenvolvido para o cálculo do peso (P) de uma determinada massa (m) sujeita a ação da força gravitacional (g) dado o valor da massa. Neste tipo de estrutura as ações são executadas de forma sucessiva da primeira a última instrução.

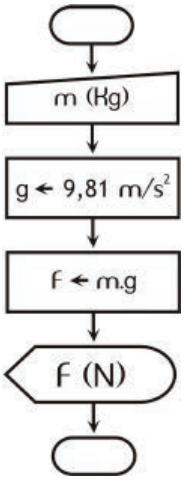


Figura 2.5 Fluxograma que implementa algoritmo para cálculo da força exercida por uma massa sujeita a ação de força gravitacional. Apenas estruturas seqüenciais são utilizadas.

2.4.2 Estruturas de decisão ou seleção

Existem três tipos de estruturas de seleção. As estruturas de decisão do tipo SE-ENTÃO e SE-ENTÃO-SENÃO, as estruturas de derivação ou escolha entre um ou mais caminhos possíveis e as estruturas do tipo CASO.

O primeiro tipo de estrutura (SE-ENTÃO) avalia uma expressão lógica resultando em uma resposta que pode ser verdadeira ou falsa. Sendo a resposta a resposta verdadeira, uma instrução ou conjunto de instruções é executado. Caso a resposta seja falsa é executado um desvio sem nenhuma instrução. A Figura 2.6 apresenta a implementação desta estrutura na forma de fluxograma.

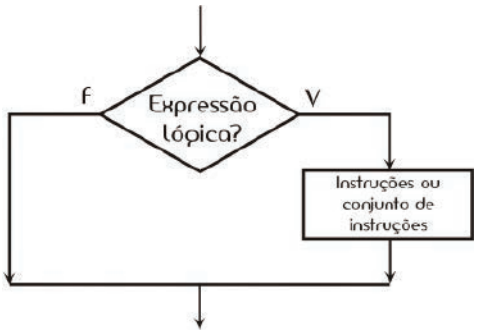


Figura 2.6 Fluxograma ilustrando a estrutura SE-ENTÃO.

Na generalidade das linguagens de programação as estruturas de programação são escritas em inglês. No caso do programa que utilizaremos na disciplina, essa estrutura é representada pela forma:

```
if (<condição>) then
   <instrução 1>
   <instrução 2>
   ...
   <instrução n>
end
```

A segunda estrutura de decisão é do tipo SE-ENTÃO-SENÃO. Nesta estrutura se o resultado da expressão lógica for verdadeiro uma instrução ou conjunto de instruções é executado. Caso a expressão lógica assuma valor falso, conjunto diferente de instruções é executado. A Figura 2.7 ilustra este tipo de estrutura.

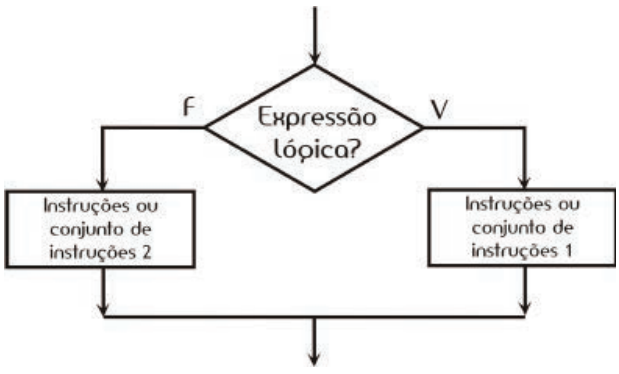


Figura 2.7 Fluxograma ilustrando a estrutura SE-ENTÃO-SENÃO.

Para representar em linguagem de programação a estrutura SE-ENTÃO-SENÃO ilustrada na Figura 2.7 teremos:

```
if (<condição>) then
   <instrução 1>
   <instrução 2>
   ...
   <instrução n>
else
   <instrução 1>
   <instrução 2>
   ...
   <instrução n>
end
```

As estruturas do tipo CASO permitem que se possa realizar a escolha entre dois ou mais caminhos a serem seguidos na programação. Neste tipo de estrutura o caminho do fluxo da informação dependerá do valor atribuído à expressão. Vejamos o exemplo ilustrado na Figura 2.8.

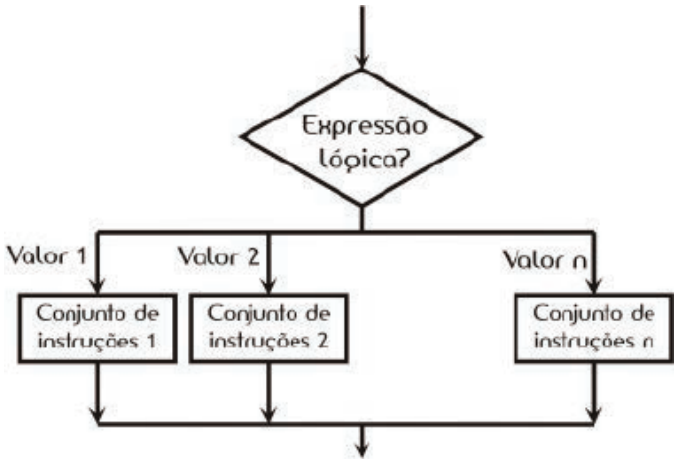


Figura 2.8 Fluxograma ilustrando a estrutura CASO.

No programa Scilab a sintaxe para este comando é da seguinte forma:

```
select <expressão>,
   case <valor_1> then,
      <conjunto de instruções 1>
   case <valor_2> then,
      <conjunto de instruções 2>
...
   case <valor_n> then,
      <conjunto de instruções n>
end
```

2.4.2.1 Expressões lógicas

São expressões que resultam nos valores lógicos, verdadeiro ou falso. A Tabela 2.2 apresenta os operadores condicionais empregados nas expressões lógicas.

Tabela 2.2 Operadores condicionais.

Operador	Símbolo
igualdade	= =
diferença	<>
maior do que	>
maior ou igual a	>=
menor do que	<
menor ou igual a	<=

2.4.3 Estruturas de repetição ou iteração

Como o próprio nome diz, este tipo de estrutura permite que se façam repetições controladas de alguma instrução ou conjunto de instruções.

Apresentaremos dois tipos desta estrutura, ambos disponíveis na linguagem de programação do Scilab. A primeira é a estrutura de repetição ENQUANTO-FAÇA. A Figura 2.9 apresenta a representação desta estrutura em forma de fluxograma

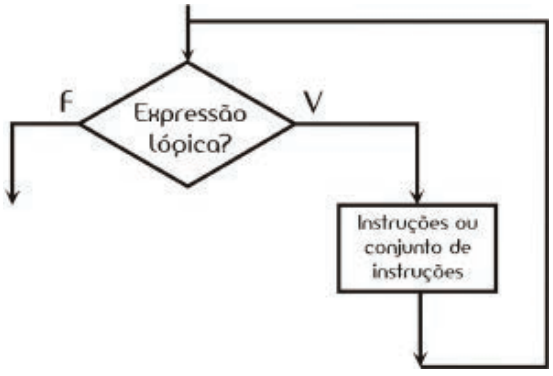


Figura 2.9 Fluxograma representando a estrutura ENQUANTO-FAÇA.

No Scilab esta estrutura é representada pelo comando **while** sendo implementada da seguinte maneira:

```
while <expressão>
  <instrução 1>
  <instrução 2>
  ...
  <instrução 3>
end
```

A segunda estrutura pode ser considerada um caso particular da estrutura ENQUANTO-FAÇA. Esta estrutura gera um ciclo de repetição controlado por uma variável (do tipo inteira) que assumirá todos os valores entre um valor inicial e um valor final, incrementando-se ou decrementando-se a si própria de um determinado valor que pode ser definido pelo programador (estrutura PARA-ATÉ-FAÇA). O caso padrão (default) é que esta variável sofra incrementos unitários. Vejamos sua representação na forma de fluxograma, Figura 2.10.

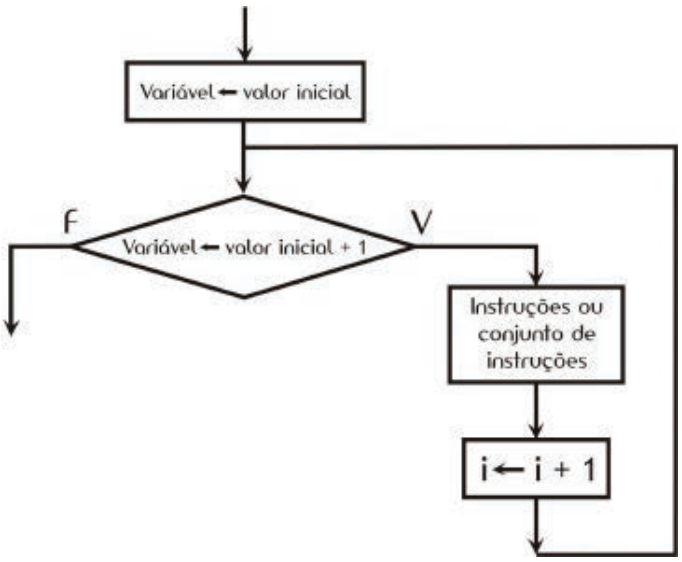


Figura 2.10 Fluxograma representando um caso particular da estrutura ENQUANTO-FAÇA.

No programa Scilab esta estrutura é representada pelo comando **for**, tendo a seguinte sintaxe.

```
for variável = expressão (vetor linha)
  <instrução 1>
  <instrução 2>
  ...
  <instrução n>
end
```

2.5 Considerações finais

Esta unidade abordou o tema algoritmo e suas formas de representação. O objetivo foi mostrar que a solução de um problema no computador passa por uma seqüência de etapas. A primeira delas é a abstração, ou seja, a descrição do problema de forma clara e precisa. Objetiva-se buscar uma seqüência de passos que conduzam até a solução do problema. A próxima etapa é a representação destes passos por meio de estruturas apropriadas para termos a solução do problema na forma de um fluxograma. O próximo passo consistirá em utilizar um programa computacional para resolução dos problemas. É o que faremos empregando o programa Scilab (unidade 3) e a planilha de cálculo Calc do pacote computacional BrOffice.org (unidade 4).

2.6 Referências bibliográficas

CEREDA, R. L. D., MALDONADO, J. C. *Introdução ao FORTRAN 77 para microcomputadores*. São Paulo: Ed. McGraw-Hill, 1987.

FERREIRA, A. B. H. *Novo dicionário da língua portuguesa*. 2 ed. Rio de Janeiro: Nova Fronteira, 1986. 1838 p.

RODRIGUES, D.; NUNO, F.; RAGGIOTTI, N. *Dicionário Larousse ilustrado da língua portuguesa*. São Paulo: Larousse do Brasil, 2004. 977 p.

SOUZA, M. A. F., GOMES, M. M., SOARES, M. V., CONCILIO, R. *Algoritmos e Lógica de Programação*. São Paulo, Ed. Pioneira Thomson Learning, 2005.

UNIDADE 3

Linguagem de programação

3.1 Primeiras palavras

Caro aluno(a): nesta unidade utilizaremos o programa computacional chamado Scilab (do inglês Scientific Laboratory, em português Laboratório Científico) para iniciarmos a programação de algoritmos. Optou-se por trabalhar com este programa pelo fato do mesmo ser distribuído gratuitamente. O programa Scilab é uma linguagem interpretada, voltada para resolução de problemas numéricos. Sua linguagem de programação é simples e de fácil aprendizado. Apresenta similaridade com outros programas comerciais. Como veremos, o programa tem um sistema de ajuda ao usuário (help) bastante prático (disponível em inglês ou francês), permite a elaboração de gráficos bi e tridimensionais, trabalha com matrizes, polinômios e sistemas lineares, possui várias funções pré-definidas. No Scilab o usuário pode definir novas funções de forma simples. Enfim, o conhecimento básico deste programa fornece a base para resolução de problemas típicos de um curso de engenharia. Aprofundando nesta ferramenta, o usuário poderá utilizá-lo para solução de problemas mais complexos.

3.2 Problematizando o tema

Nesta unidade será apresentado o programa Scilab. O objetivo será aprendermos a trabalhar com este aplicativo, quer seja em seu ambiente de trabalho ou elaborando programas (na forma scripts e funções) salvos em arquivos e que podem ser executados posteriormente no ambiente de trabalho.

Iremos implementar nesta linguagem de programação alguns dos exemplos já apresentados em unidades anteriores.

3.3 Introdução ao programa Scilab

O programa Scilab foi criado por pesquisadores pertencentes ao Institut National de Recherche en Informatique et an Automatique (INRIA) da École Nationale des Ponts et Chaussées (ENPC), no ano de 1990. Desde o ano de 1994 é distribuído com código fonte aberto (open source software): <http://www.scilab.org/>. Atualmente é produzido no INRIA por um consórcio, o Scilab Consortium, estabelecido em 2003 por iniciativa do próprio INRIA. É usado em diversos ambientes industriais e educacionais pelo mundo.

O Scilab é um pacote computacional científico para computação numérica semelhante a programas comerciais. É uma ferramenta computacional poderosa, de código fonte aberto e empregada para aplicações científicas e em engenharia.

O programa possui versões para uso na maioria dos sistemas operacionais Unix (incluindo GNU/Linux) e Windows (9X/2000/XP).

De acordo com os desenvolvedores/mantenedores do aplicativo Scilab, o tipo de licença adotado não é compatível com a licença GPL (do inlgês General Public License, em português Licença Pública Geral), a licença para programas da Free Software Foundation (em português Fundação para o Software Livre). Contudo, o tipo de licença adotado permite:

- livre uso e distribuição do aplicativo para fins não comerciais;
- uso para fins comerciais desde que o programa não sofra alterações (modificações na versão original) ou como programa composto (ou seja, incluindo-o em outro programa).

3.4 Ambiente de trabalho

Após a instalação do aplicativo, o usuário deverá iniciá-lo. A janela inicial será aberta, conforme ilustra a Figura 3.1.

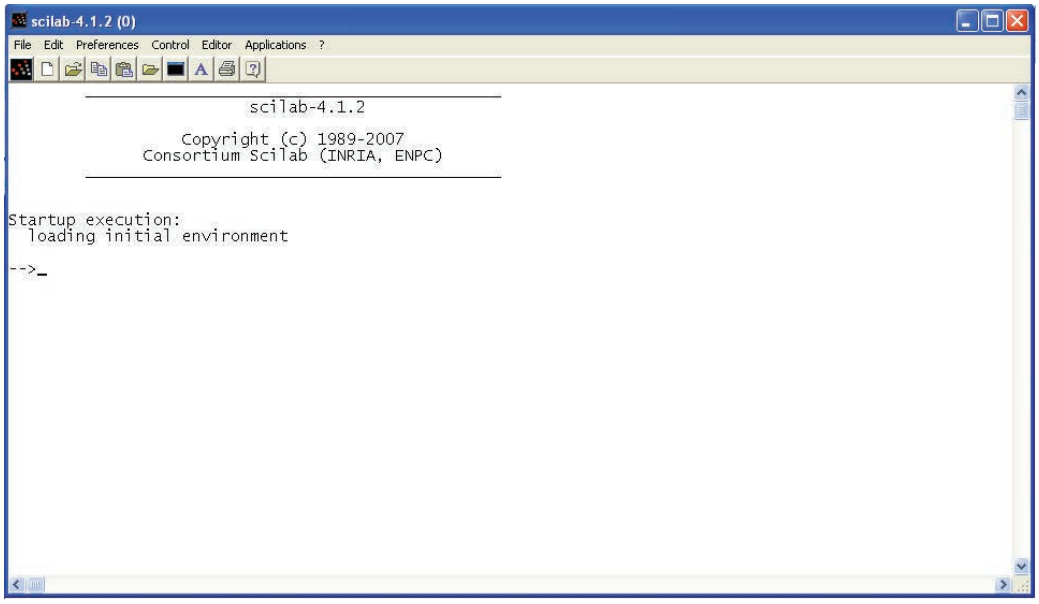


Figura 3.1 A janela do programa Scilab (versão 4.1.2).

Na Figura 3.1, observa-se que o cursor do Scilab (prompt) é representado por uma seta (-->), e que o mesmo permanece piscando. Os comandos devem ser inseridos após o cursor, seguido da tecla enter (←) para execução dos mesmos.

Na parte superior da janela têm-se os seguintes menus:

- File
- Edit
- Preferences
- Control
- Editor
- Applications
- ? (Help)

Cada um desses menus pode ser expandido em sub-menus ao clicar sobre eles. Também na parte superior da janela encontram-se teclas de acesso rápido, simbolizadas por figuras (ícones), conforme ilustra a Figura 3.2.

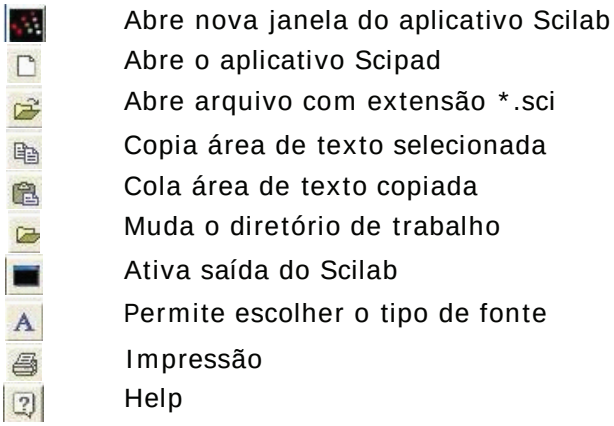


Figura 3.2 Teclas de acesso rápido do programa Scilab.

É possível a qualquer momento obter ajuda, clicando sobre o comando ***help***, por meio da opção na barra de tarefas, da tecla de atalho, ou simplesmente digitando na linha de comando do Scilab o comando ***help()*** ou ***help***. É aberta a janela, ilustrada pela Figura 3.3 (Scilab Browse Help), onde o usuário pode navegar pelos tópicos disponíveis.

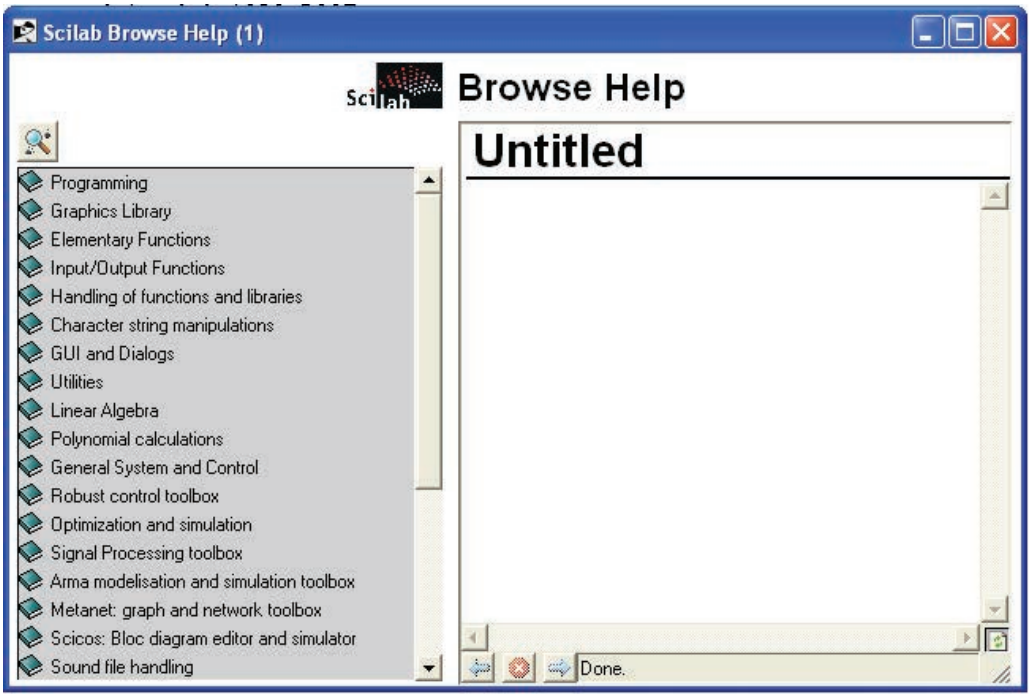


Figura 3.3 A janela de ajuda do programa Scilab.

3.5 Primeiros passos

Iniciaremos os primeiros passos no aplicativo Scilab abrindo a janela principal, ilustrada na Figura 3.1. Aprenderemos inicialmente a realizar comandos diretamente no ambiente de trabalho.

3.5.1 Inserindo comentários

É sempre muito conveniente o uso de comentários para explicar o que se está programando. Para inserir um comentário emprega-se o seguinte comando: **//**.

Exemplo:

```
—> // Inserindo um comentario
```

Os caracteres inseridos à direita das duas barras invertidas não são interpretados pelo Scilab. A utilização de comentários é um recurso importante na documentação de programas. Recomenda-se não utilizar no ambiente de programação palavras grafadas com acentos para evitar possíveis erros de interpretação no aplicativo.

3.5.2 Variáveis

No programa Scilab existem algumas variáveis que possuem valores pré-definidos. Estas variáveis são protegidas e não podem ser apagadas. Para saber quais são as variáveis pré-definidas digite o comando who na linha de comando, em seguida pressione a tecla enter (↵) para execução. A Figura 3.4 apresenta a lista com estas variáveis.

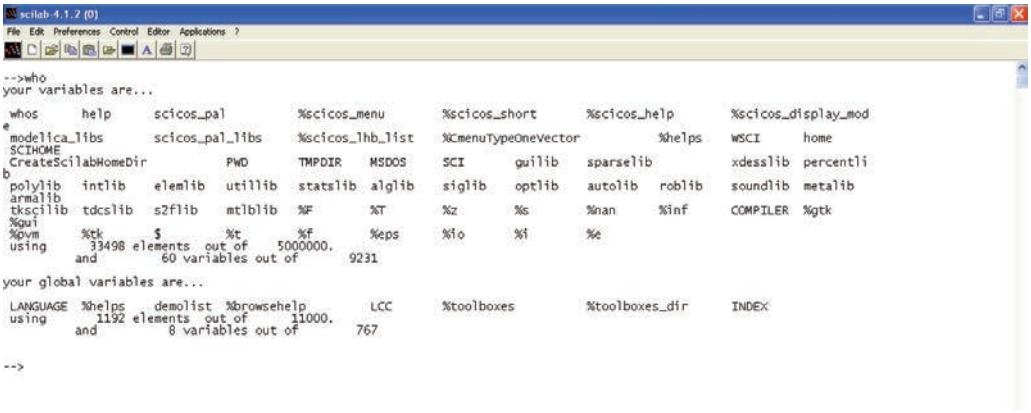


Figura 3.4 Lista de variáveis pré-definidas no programa Scilab.

Por exemplo, a variável **%e** representa o número de Euler, **%pi** representa o número pi (quociente entre o perímetro de uma circunferência e o seu diâmetro). Digitando na linha de comando **%e** [enter], e em seguida **%pi** [enter], teremos os valores destas variáveis, respectivamente, 2.7182818 e 3.1415927 (representados por 7 casas decimais). No aplicativo Scilab o separador entre a parte inteira e decimal de um número é representado pelo ponto (sistema americano).

Exemplo:

```
–> %e          [enter]
%e   =
             2.7182818
–> %pi         [enter]
%pi  =
             3.1415927
```

Podemos atribuir valores às variáveis. Vejamos o exemplo ilustrado na Figura 3.5 onde a variável a assume o valor 2.5 e a b o valor 7.5.

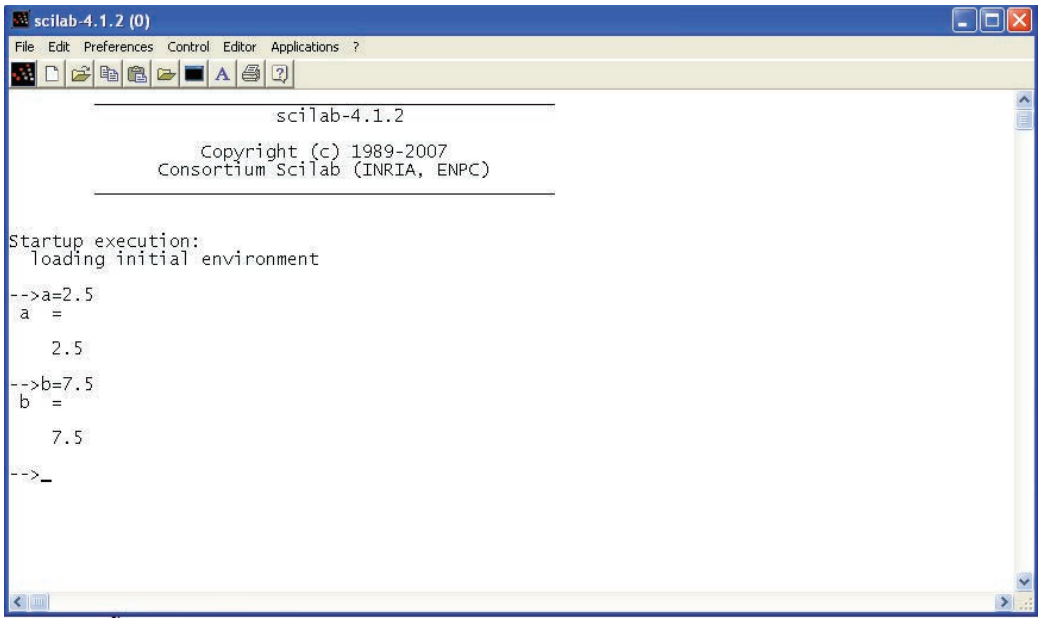


Figura 3.5 Atribuindo os valores 2.5 e 7.5 às variáveis a e b, respectivamente no ambiente de trabalho do programa Scilab.

É importante observar que o programa Scilab diferencia letras maiúsculas de minúsculas. Dessa forma, experimente atribuir os valores 4.5 e 1.4 às variáveis x (letra minúscula) e X (letra maiúscula), respectivamente, e veja o resultado. Você verá que o programa Scilab diferencia as letras maiúsculas das minúsculas (Figura 3.6). Voltaremos a abordar este tópico.

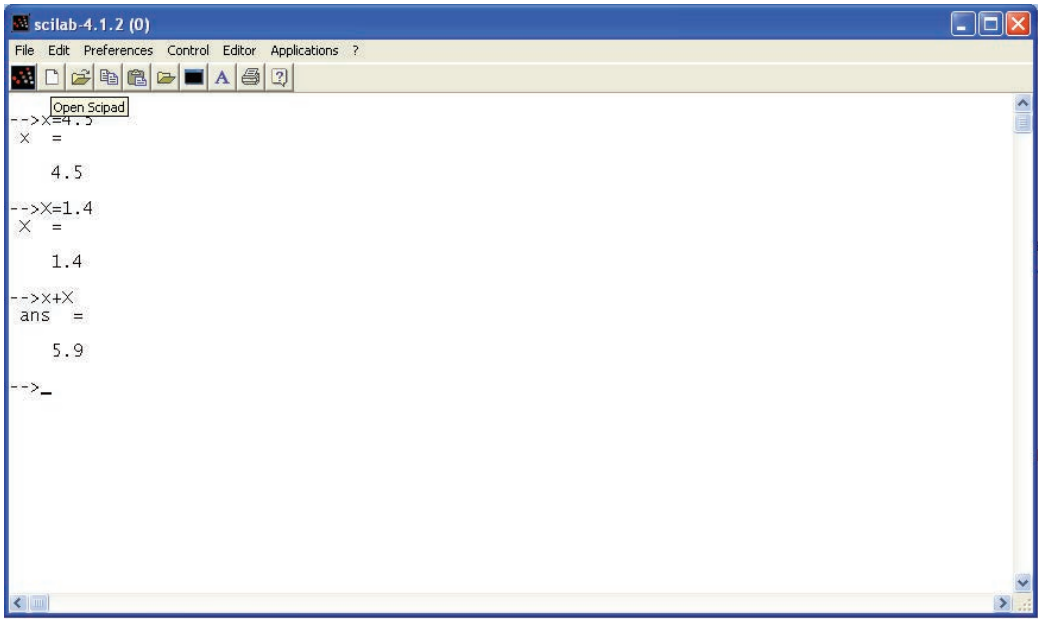


Figura 3.6 Atribuindo valores diferentes as variáveis x (minúscula) e X (maiúscula) no ambiente de trabalho do programa Scilab. A soma x+X produz o resultado 5.9.

3.5.3 Operações básicas

É possível realizar operações básicas com as variáveis declaradas. A soma, subtração, multiplicação e divisão são representadas pelos respectivos símbolos: +, -, *, /.

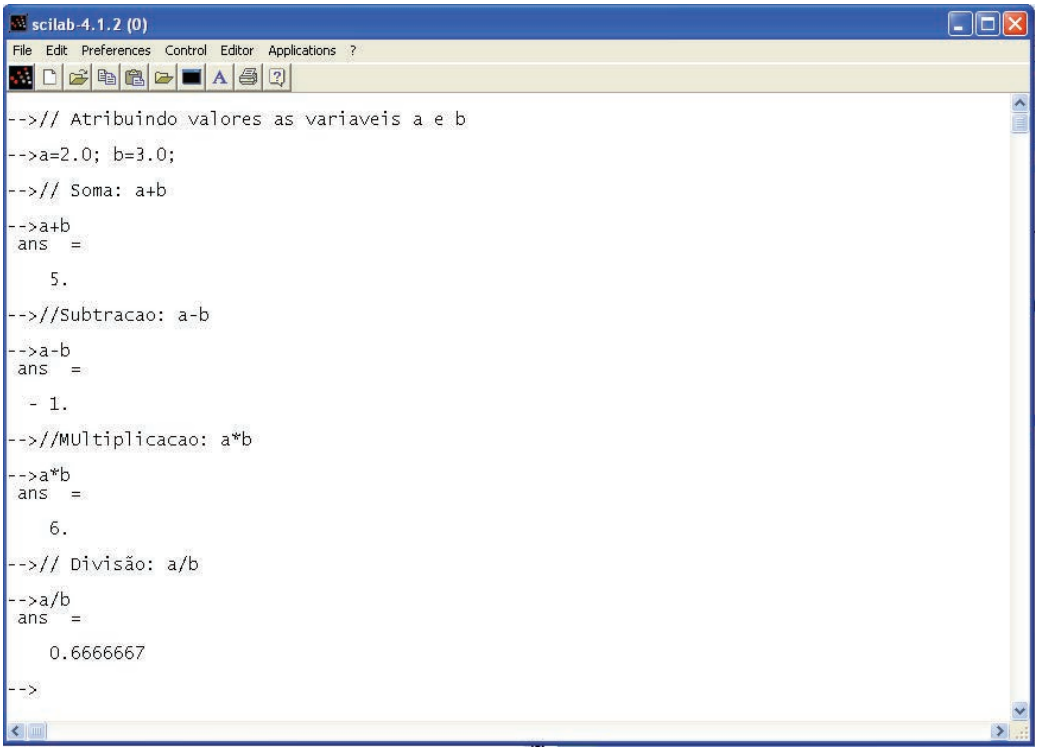
A exponenciação de uma variável é representada pelo símbolo ^ ou **.

A função raiz quadrada de uma variável é obtida por meio do comando sqrt( ); variável que se deseja extrair a raiz deve estar inserida dentro do parênteses.

A função exponencial é obtida por meio do comando exp( ), com a variável que se deseja aplicar a função inserida dentro do parênteses.

A função logaritmo neperiano de uma variável é calculado com o comando log( ). O logaritmo na base 10 é calculado com o comando log10( ).

A Figura 3.7 apresenta exemplos destas operações básicas.



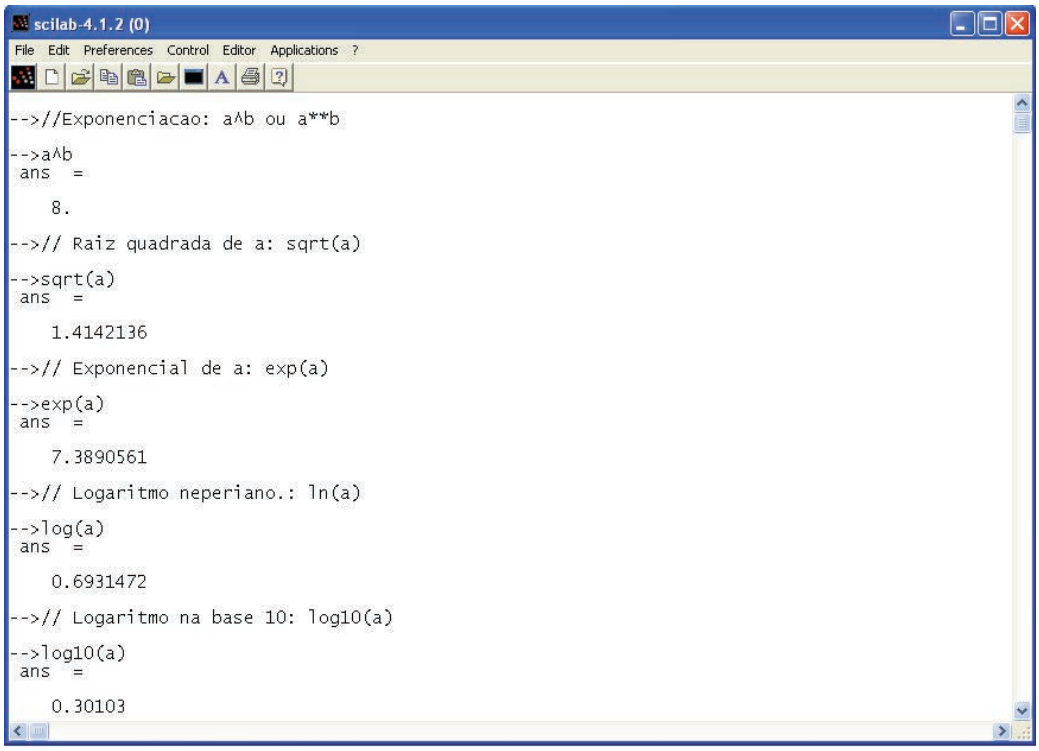


Figura 3.7 Operações básicas com variáveis no programa Scilab.

Os operadores algébricos possuem uma ordem para sua execução. A Figura 3.8 apresenta a ordem de execução.

——————— (menor) Hierarquia (maior) >			
+ e -	* e /	^ ou **	( )

Figura 3.8 Ordem de execução dos operadores algébricos.

3.5.4 Apagando variáveis declaradas

Para apagar uma variável criada pelo usuário utiliza-se o comando ***clear*** seguido do nome da variável que se deseja apagar.

Exemplo:

```
—> clear a      [enter]
```

Pode-se também optar por apagar todas as variáveis declaradas pelo usuário que se encontram na memória do programa. Para isso utiliza-se o comando ***clear***. As variáveis pré-definidas no Scilab não são apagadas com este comando.

Exemplo:

```
–> clear          [enter]
```

3.5.5 Limpando o ambiente de trabalho

O comando ***clc*** é utilizado para limpar a área de trabalho no programa Scilab. Lembre-se que o comando ***clc*** apenas limpa a tela (ambiente de trabalho), e o comando ***clear*** é que apaga as variáveis da memória.

3.5.6 Funções trigonométricas

De forma similar a outros pacotes computacionais, o programa Scilab também possui um conjunto de funções pré-definidas e que podem ser utilizadas pelo usuário, a exemplo das funções **exp( )** e **sqrt( )** vistas anteriormente.

A Tabela 3.1 apresenta as funções trigonométricas definidas no programa Scilab. O argumento utilizado está em radianos.

Tabela 3.1 Funções trigonométricas definidas no Scilab.

Função	Comando	Exemplo
seno	*sin( )*	*sin(%pi/2.0)*
arcoseno	*asin( )*	*asin(%pi/2.0)*
coseno	*cos( )*	*cos(%pi/2.0)*
arcoseno	*acos( )*	*acos(%pi/2.0)*
tangente	*tan( )*	*tan(%pi/4.0)*
arcotangente	*atan( )*	*atan(%pi/4.0)*
cotangente	*cotg( )*	*cotg(%pi/4.0)*

3.5.7 Constantes especiais

O Scilab possui algumas constantes denominadas especiais e que se iniciam com o símbolo porcentagem (%).

São elas:

- **%i**: representa o valor da raiz quadrada de –1.0.
- **%pi**: representa o valor da variável pi (3.1415927, representado com sete casas decimais).
- **%eps**: representa o menor número armazenado na precisão utilizada pelo programa no computador, tal que 1.0+%eps = 1.0. É a precisão da máquina na precisão empregada.
- **%inf**: representa um número muito grande.
- A divisão de um número por %inf resulta sempre em zero.
- **%Nan**: representa "not a number" (não é um número).
- **%t** e **%f**: representam as constantes booleanas verdadeiro e falso, respectivamente.

3.6 Alguns exemplos

Apresentamos alguns exercícios que devem ser executados no programa Scilab.

Atribuir os valores 4.0 e 8.0 a duas variáveis x e y. Realize as operações de soma x + y , subtração x – y , divisão x / y e multiplicação x * y .

Realize agora a seguinte operação: x + Y , escrevendo a variável y em maiúscula. O que acontece?

O programa irá exibir a seguinte mensagem:

```
!--error 4
undefined variable : Y
```

Tente agora: y / X , com a variável x em maiúsculo. O que acontece?

O programa avisa que ocorreu um erro; a variável X (em maiúscula) não foi definida pelo usuário e, portanto, a operação não pôde ser realizada. Caso

a variável X tivesse sido definida e a ela atribuído o valor zero (0), a seguinte mensagem seria mostrada na tela:

```
!–error 27
division by zero...
```

O programa apresenta um erro; foi realizada uma divisão por zero.

Você deve ter observado que o resultado das operações até agora são apresentados da seguinte forma:

```
–> a = 2.0       [enter]
     a = 2.
–> b = 5.0       [enter]
     b = 5.
–> a + b          [enter]
–ans = 7.
```

O valor da soma das variáveis a+b é armazenado na variável **ans** (abreviação de answer em inglês).

Experimente o seguinte comando (na seqüência dos anteriores já digitados):

```
–> c = ans
   c = 7.
```

Tente agora colocar um ponto e vírgula após cada comando.

```
–> a = 2.0;       [enter]
–> b = 5.0;       [enter]
–> a + b;         [enter]
–> ans            [enter] (sem ponto e vírgula)
   ans = 7.
```

O ponto e vírgula faz com que o resultado da operação não seja mostrado na linha de comando imediatamente após sua execução. Contudo, a operação foi realizada e o resultado encontra-se armazenado na variável **ans**.

Tente uma operação combinando operadores, para verificar a hierarquia dos operadores. Por exemplo:

```
-> a = 2.0 ; b = 4.0 ; c = 6.0 ;
-> x1 = 5.0*a + b - c^2.0
x1 = -22.
-> x2 = 5.0*(a + b) - c^2.0
x2 = -6.
-> x3 = 5.0*(a + b - c) ^2.0
x3 = 0.
```

O programa Scilab também trabalha com valores de variáveis complexas. Vejamos o exemplo:

```
-> a = 2.0 + 4.0 * %i
     2. + 4.i
-> b = 1.0 + 2.0 * %i
     1. + 2.i
```

Experimente realizar as operações de soma, subtração, divisão e multiplicação.

3.7 Trabalhando com polinômios, vetores e matrizes

O programa Scilab trabalha com polinômios, vetores e matrizes de forma similar a vários pacotes computacionais disponíveis no mercado.

3.7.1 Polinômios

No programa Scilab existe uma sintaxe para definição de polinômios. Experimente o comando ***help poly*** na linha de comando. Veja o que a janela de ajuda traz de informações sobre a função ***poly***.

Uma forma de criar um polinômio no Scilab é a partir de suas raízes. Vejamos o exemplo:

```
-> p = poly ([1 2], 'x', 'roots')
   p = 2 - 3x + x^2
```

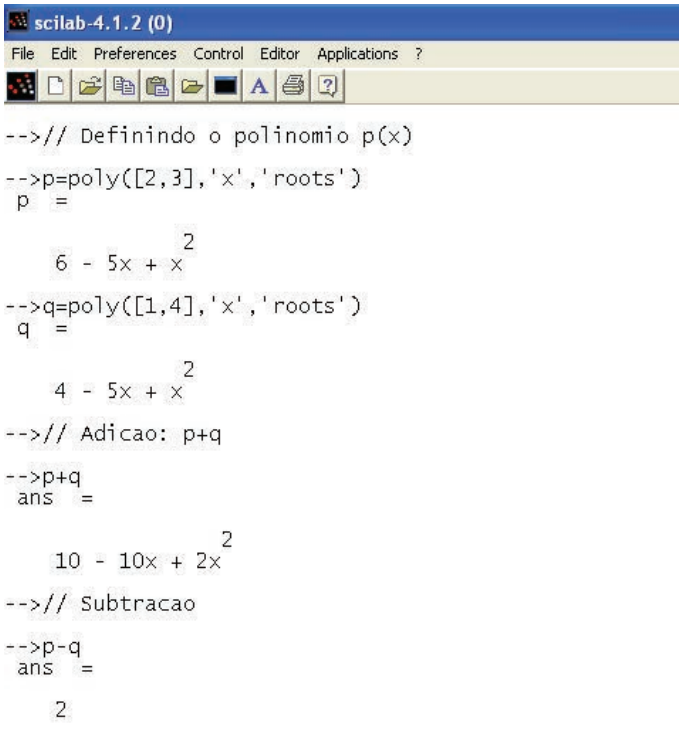
Com este comando definimos o polinômio cujas raízes são dadas pelos valores 1 e 2. Para confirmar, utilizemos a função ***roots***. Esta função é utilizada para calcular as raízes de um polinômio inserido no argumento.

```
-> roots(p)
    ans  =
              1.
              2.
```

Outra maneira de definirmos o mesmo polinômio é por meio do seguinte comando:

```
-> x=poly(0,'x');
-> p=2-3*x+x^2
   p = 2 - 3x + x^2
```

As operações de soma, subtração, multiplicação e divisão de polinômios são realizadas conforme ilustra a Figura 3.9.



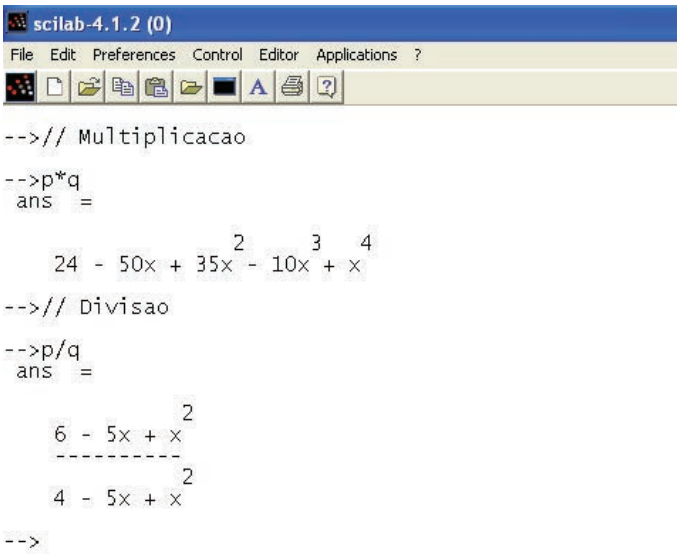


Figura 3.9 Operações de soma, subtração, multiplicação e divisão com os polinômios p(x) e q(x).

Para obter o quociente e o resto da divisão de p(x)/q(x) utilize a função ***pdiv(p,q)***.

```
-> [r,q] = pdiv(p,q)
   q  =   1
   r  =   2.
```

Se desejarmos saber o valor do polinômio p(x) em algum ponto (por exemplo, em x=3.0) utilize a função ***horner(p,3.0)***.

```
-> horner(p,3.0)
   ans  =   0.
```

3.7.2 Vetores e matrizes

O programa Scilab trabalha com vetores e matrizes de forma similar a vários pacotes computacionais disponíveis no mercado.

Os vetores são criados colocando seus componentes entre colchetes, [ ]. Contudo, podemos criar vetores linha ou vetores coluna. A diferença está no elemento separador utilizado para separar os elementos do vetor. Fica mais fácil entender com um exemplo. Suponhamos que desejamos criar um vetor coluna **x** (com três linhas e uma coluna) contendo os seguintes elementos reais:

$$y = [2.0 \quad 4.0 \quad 6.0]_{1x3}$$

No Scilab utilizaremos o seguinte comando:

```
-> x=[1.0; 2.0; 3.0]
   x = 1.
       2.
       3.
```

Como observamos, o separador dos elementos na criação de um vetor coluna é o ponto e vírgula. Para criarmos um vetor linha **y** empregamos como separador o espaço (espaço em branco) ou a vírgula. Representaremos o seguinte vetor linha no Scilab:

$$y = [2.0 \quad 4.0 \quad 6.0]_{1x3}$$

No Scilab utilizaremos o seguinte comando:

```
-> y=[2.0  4.0  6.0]
   y =  2.   4.   6.
```

Para transformarmos um vetor coluna em um vetor linha, ou vice-versa, podemos realizar a operação de transposição de vetores. Para transpor um vetor no Scilab é empregado o símbolo ‘ (apóstrofo). Vejamos o exemplo, transpondo o vetor y:

```
-> y = y'
   y = 2.
       4.
       6.
```

A dimensão de um vetor (linha ou coluna) pode ser determinada no Scilab empregando o comando **size**. Vejamos o exemplo:

```
-> y=[2.0  4.0  6.0];
-> size (y)
     ans
             1.   3.
```

A resposta deste comando nos diz que o vetor **y** é composto por uma linha e três colunas. Verifique no Scilab.

Os vetores podem ser multiplicados ou divididos por grandezas escalares. Vetores de mesma dimensão também podem ser somados ou subtraídos. A operação produto escalar (ou produto interno) é realizada entre dois vetores de mesma dimensão por meio da expressão vetorial:

$$x = \begin{bmatrix} 1.0 \\ 2.0 \\ 3.0 \end{bmatrix}_{3x1} \qquad y = \begin{bmatrix} 2.0 \\ 4.0 \\ 6.0 \end{bmatrix}_{3x1}$$

$$z = x^T \cdot y$$

O resultado será um número escalar. No Scilab fazemos:

```
-> x=[2.0;  4.0;  6.0];
-> y=[2.0;  4.0;  6.0];
-> z=x'*y
   z = 28.
```

Se dois vetores possuem dimensões diferentes, por exemplo, x_{1x3} e y_{1x4}, o produto vetorial (ou produto externo) é realizado por meio da expressão vetorial:

$$x = \begin{bmatrix} 1.0 \\ 2.0 \\ 3.0 \end{bmatrix}_{3x1} \qquad y = \begin{bmatrix} 1.0 \\ 2.0 \\ 3.0 \\ 4.0 \end{bmatrix}_{4x1}$$

$$z = x \cdot y^T$$

```
-> x=[1.0;  2.0;  3.0];
-> y=[1.0;  2.0;  3.0;  4.0];
-> z=x*y
   z  =    1.    2.    3.    4.
           2.    4.    6.    8.
           3.    6.    9.    12.
```

A composição de uma matriz no ambiente Scilab é realizada de forma similar a dos vetores, separando-se cada linha por um ponto e vírgula (os elementos da mesma linha podem ser separados por espaço ou por vírgula). Suponhamos que desejamos criar uma matriz **A** (com três linhas e três colunas) contendo os seguintes elementos reais (* no texto será empregado letras minúsculas para representar os vetores e maiúsculas para representação de matrizes, ambas em negrito):

$$A = \begin{bmatrix} 1.0 & 2.0 & 3.0 \\ 4.0 & 3.0 & 5.0 \\ 0.0 & 2.0 & 4.0 \end{bmatrix}_{3x3}$$

No Scilab utilizaremos o seguinte comando:

```
-> A=[1.0  2.0  3.0 ; 4.0  5.0  6.0;  7.0  8.0  9.0]
   A =   1.    2.    3.
         4.    5.    6.
         7.    8.    9.
```

Uma matriz é transposta pelo comando ‘ (apóstrofo). A Tabela 3.2 apresenta uma lista de operações que podem ser realizadas com matrizes.

Tabela 3.2: Operações com matrizes (estruturais* e matriciais).

Operação	Comando Scilab	Comentário
Soma estrutural	A+B	A soma (estrutural) de matrizes é idêntica à matricial. Obs: as duas matrizes precisam ter o mesmo número de linhas e colunas.
Subtração estrutural	A-B	Subtração de matrizes estrutural é idêntica à matricial. Obs: as duas matrizes precisam ter o mesmo número de linhas e colunas..
Multiplicação estrutural	A.*B	Multiplicação elemento a elemento de **A** e B. Obs: As duas matrizes precisam ter a mesmo número de linhas e colunas, ou uma delas ser um escalar.
Multiplicação matricial	A*B	Multiplicação das matrizes **A** e **B**. O número de colunas da matriz **A** precisa ser igual ao número de linhas da matriz **B**.
Divisão estrutural à direita	A./B	Divisão elemento a elemento de **A** e **B**: a(i,j)/b(i,j). Obs: As duas matrizes precisam ter a mesmo número de linhas e colunas, ou uma delas ser um escalar.
Divisão estrutural à esquerda	A.\B	Divisão elemento a elemento de **A** e **B**: b(i,j)/a(i,j) Obs: As duas matrizes precisam ter a mesmo número de linhas e colunas, ou uma delas ser um escalar.
Divisão matricial à direita	A/B	Divisão matricial definida por **A** * inv(**B**), onde inv(**B**) é a inversa da matriz **B**.
Divisão matricial à esquerda	A\B	Divisão matricial definida por inv(**A**)***B**, onde inv(**A**) é a inversa da matriz **A**.
Expoente estrutural	A.^B	Expoente elemento a elemento de **A** e **B**: a(i,j)^b(i,j). Obs: As duas matrizes precisam ter a mesmo número de linhas e colunas, ou uma delas ser um escalar.

* Operações estruturais: são operações entre matrizes executadas elemento a elemento; a operação é executada sobre os elementos correspondentes nas matrizes.

No Scilab é possível criar um vetor ou matriz nula, um vetor ou matriz unitária ou uma matriz identidade de forma simples empregando os seguintes comandos:

- **zeros**(nl, nc), **ones**(nl, nc) e **eye**(nl, nc), em que nl é o número de linhas e nc o número de colunas (se nl ou nc for igual a 1 teremos um vetor linha ou coluna). A Figura 3.10 ilustra esses comandos.

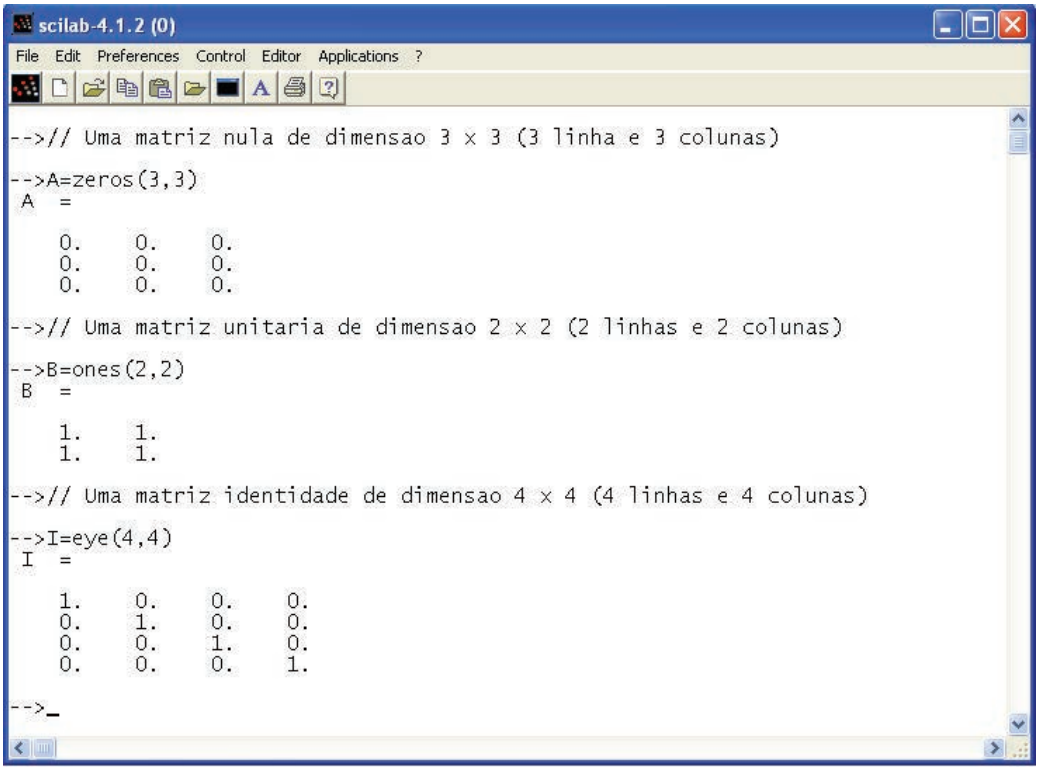


Figura 3.10 Criando matriz de zeros, uns e identidade no Scilab.

No programa Scilab o valor do determinante de uma matriz **A** (**A** deve ser uma matriz quadrada) é facilmente calculado empregando a função **det**( ), em que o argumento desta função é a matriz que se deseja obter o determinante, no caso det(**A**).

Também a inversa de uma matriz **A** (**A** deve ser uma matriz quadrada) é calculada empregando a função **inv**( ), em que o argumento desta função é a matriz que se deseja obter o determinante, no caso **inv**(**A**). Atenção para as propriedades do cálculo matricial. Vejamos o cálculo do determinante e da inversa de uma matriz $\mathbf{A}_{3x3}$, Figura 3.11.

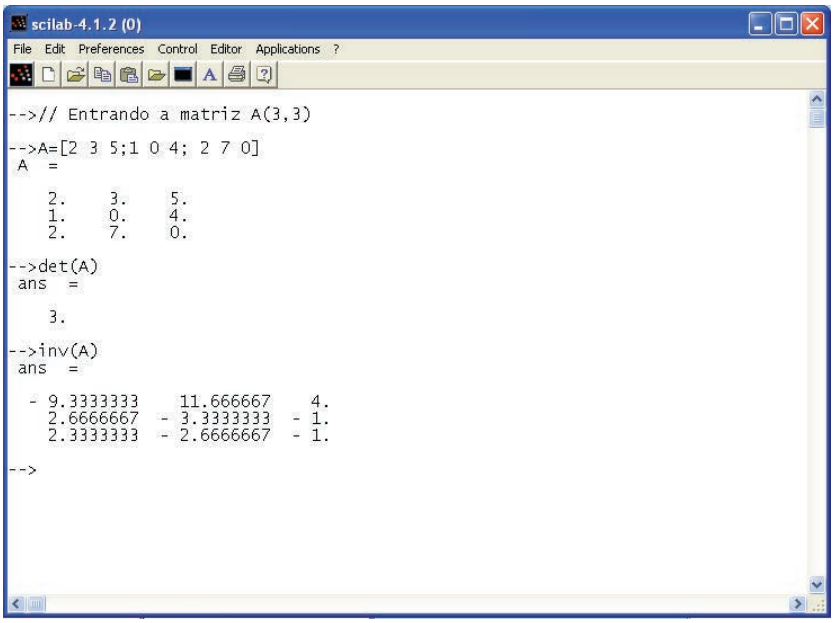


Figura 3.11 Avaliando o valor do determinante e a inversa de uma matriz $\mathbf{A}_{3x3}$.

3.8 Construindo gráficos

Outra ferramenta importante no ambiente de programação do Scilab é a possibilidade de trabalharmos gráficos. Apresentaremos alguns comandos usados para a construção de gráficos bidimensionais e tridimensionais dentro do ambiente do Scilab. As saídas funções gráficas do Scilab são sempre apresentadas em uma janela gráfica como a ilustrada na Figura 3.12.

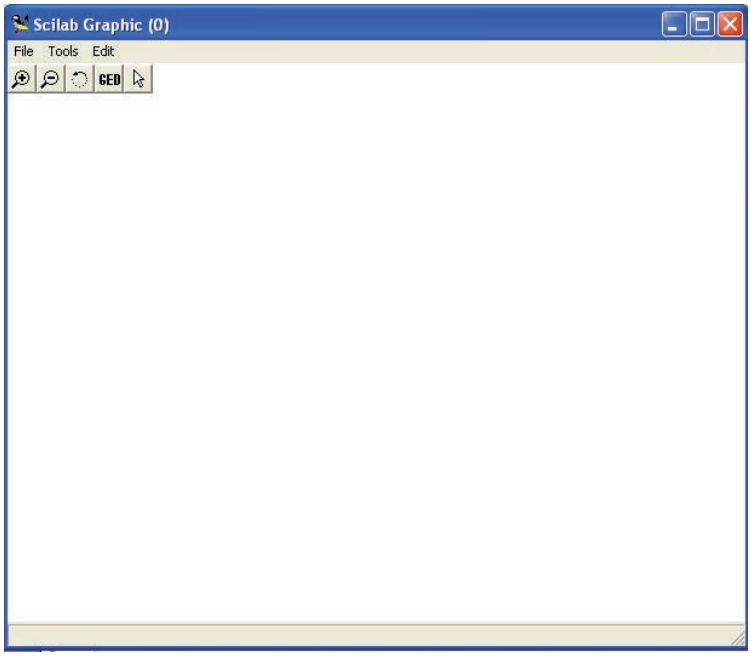
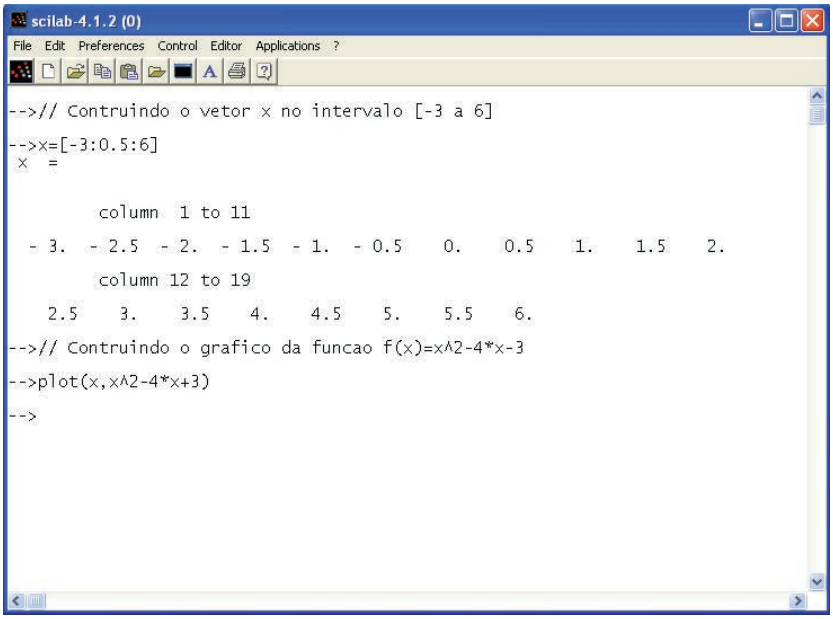
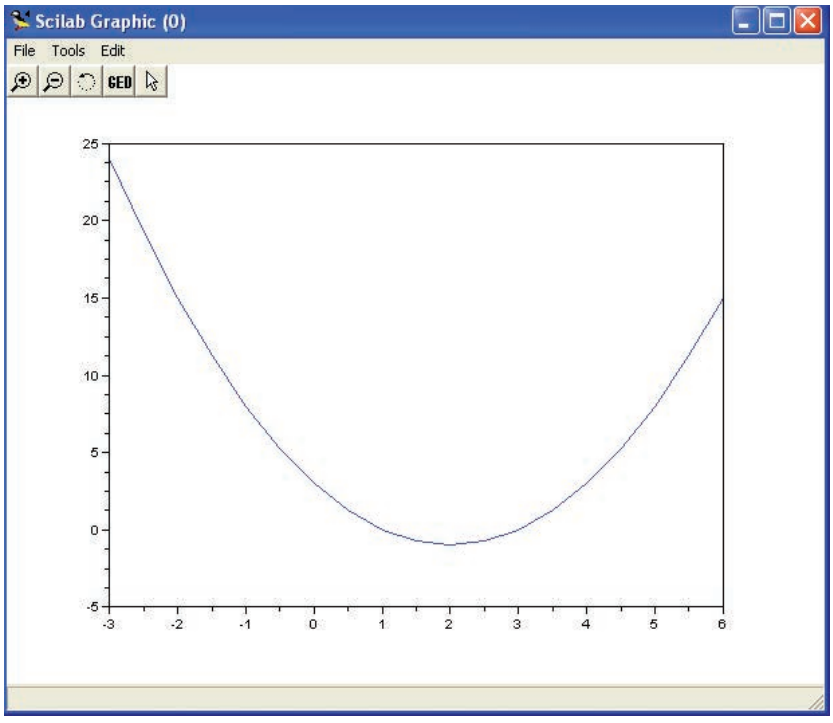


Figura 3.12 Janela gráfica do programa Scilab.

Para utilizarmos o comando **plot**( ) precisamos definir dois vetores (de mesma dimensão), **x** e **y**, por exemplo. Esses vetores serão empregados como argumentos do comando **plot**(**x**, **y**). Vejamos o exemplo ilustrado na Figura 3.13.



(a)



(b)

Figura 3.13 (a) Comandos utilizados para gerar o gráfico; (b) Gráfico da função f(x) = $x^2 - 4 \cdot x + 3$ no intervalo –3 ″ x ″ 6.

Como pode ser observado na Figura 3.13 o gráfico da função f(x) foi apresentado na janela gráfica por meio de uma linha sólida no intervalo correspondente (–3 ″ x ″ 6). Caso o usuário queira visualizar os pontos utilizados para gerar o gráfico (valores correspondentes a cada posição x, f(x), bastaria executar o comando **plot** com a seguinte sintaxe:

```
–> plot(x, x^2-4*x-3,'x')
```

O resultado deste comando é apresentado na Figura 3.14.

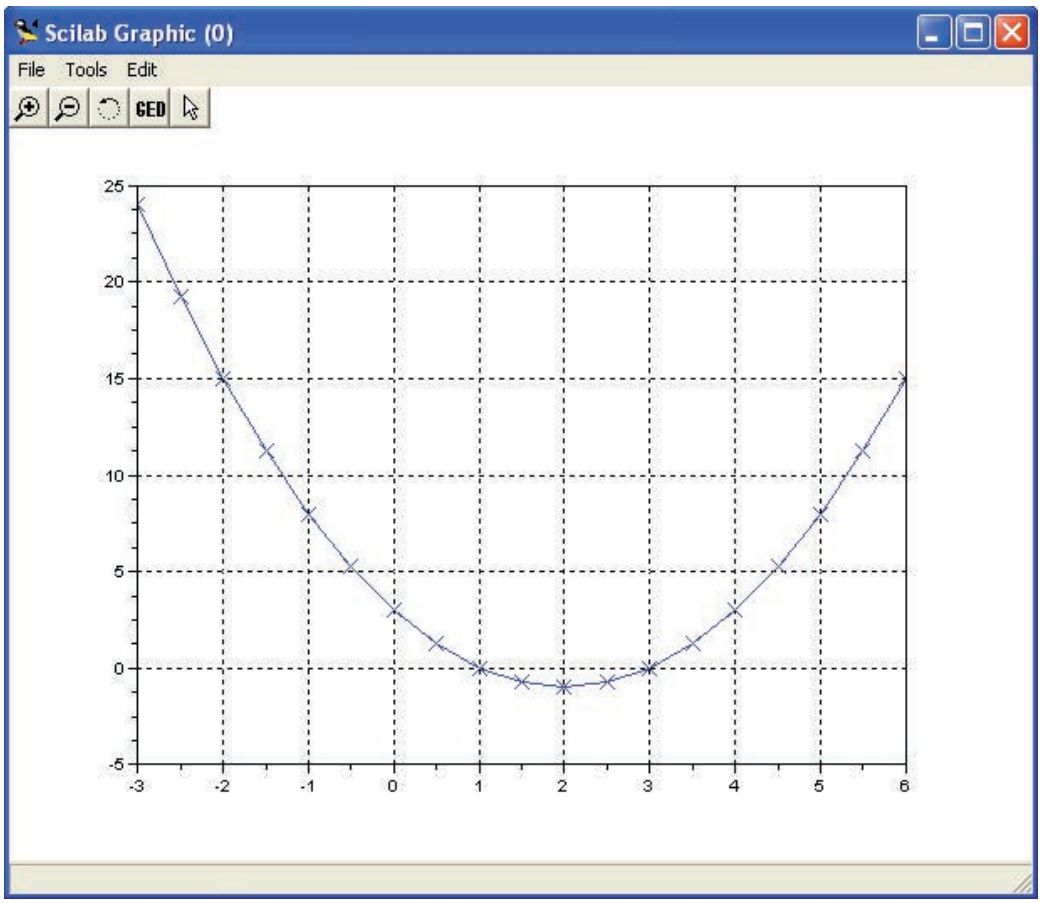


Figura 3.14 Gráfico da função $f(x) = x^2 – 4 \cdot x – 3$ ilustrando com um x os pontos x, f(x) no intervalo –3 ″ x ″ 6.

Nas duas figuras apresentadas, 3.13 e 3.14, observe que para a janela gráfica foi atribuído o valor 0 (por default). Se desejarmos atribuir números às janelas gráficas, caso desejemos construir mais de um gráfico, utilizamos o comando **scf**( ), especificando em seu argumento o número da janela gráfica, por exemplo, **scf**(1). O Scilab permite que mais de uma janela gráfica seja utilizada.

O comando **clf**() é usado para apagar o conteúdo de uma janela gráfica. Para eliminar uma janela gráfica é utilizado o comando **xdel**(). Em ambos os comandos uma janela específica pode ser acessada indicando seu índice entre os parênteses.

As Tabelas 3.3, 3.4 e 3.5 apresentam uma lista das propriedades empregadas com o comando **plot**.

Gráficos mais elaborados podem ser construídos com as funções plot2d, plot2d2 (função degrau), plot2d3 (gráficos 2D com barras verticais) e plot2d4 (gráficos 2D com setas). Além destes comandos para gráficos bi-dimensionais, o comando plot3d é indicado para a representação de dados em três dimensões. Vejamos o exemplo ilustrado na Figura 3.15.

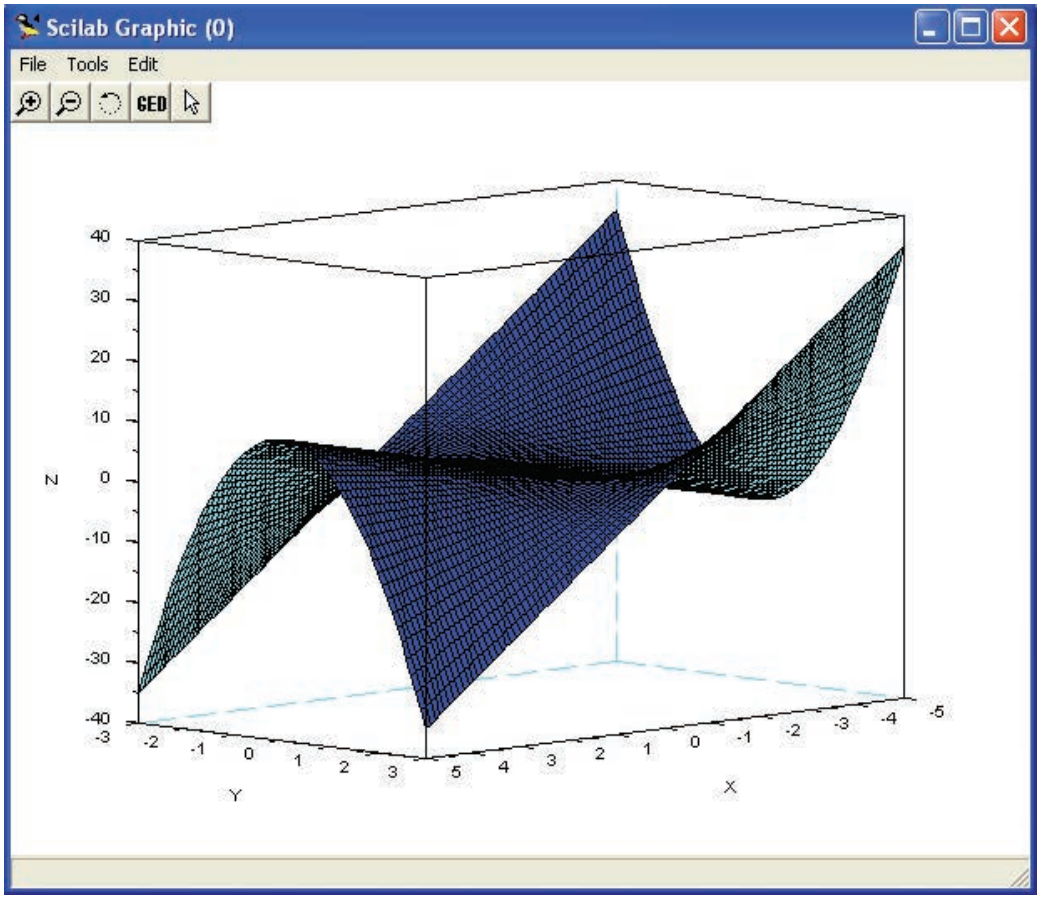


Figura 3.15 Exemplo de um gráfico implementado com a função plot3d. Comandos utilizados no Scilab para elaboração do gráfico:

```
—>y=-5:0.1:5; b=-3:0.1:3; z=y'*(-b**2+2); plot3d(y,b,z);
```

Maiores informações são encontradas no menu ajuda ao usuário (buscar pelo comando plot), ou, digitar na linha de comando help(plot).

Tabela 3.3 Argumentos que especificam o tipo de linha.

Especificador	Tipo de linha
-	Linha sólida (*default*)
--	Linha tracejada
:	Linha pontilhada
-.	Linha tracejada-pontilhada

Tabela 3.4 Argumentos que especificam a cor de uma linha.

Especificador	Cor
r	Vermelho
g	Verde
b	Azul
c	Ciano
m	Magenta
y	Amarelo
k	Preto
w	Branco

Tabela 3.5 Argumentos que especificam os tipos de marcadores para os pontos a serem desenhados nos gráficos.

Especificador	Tipo de marcador
+	Sinal de +
o	Círculo
*	Asterisco
.	Ponto
x	Sinal de multiplicação
‘square’ ou ‘s’	Quadrado
‘diamond’ ou ‘d’	Diamante
^	Triângulo voltado para baixo
v	Triângulo voltado para cima
>	Triangulo voltado para direita
<	Triângulo voltado para esquerda
‘pentagram’	Estrela com cinco pontas
‘none’	Nenhum marcador

3.9 Elaborando programas: scripts e funções

No programa Scilab podemos criar arquivos contendo comandos que serão executados posteriormente dentro do seu ambiente. Podemos criar dois tipos diferentes de arquivos. Um deles é chamado de script; o outro recebe o nome de função. O script é um arquivo com a extensão sce e que contém uma seqüência de comandos. Quando chamado a execução no ambiente do Scilab os comandos são processados (ou interpretados). Veremos primeiramente como programar em um script. O arquivo deverá, preferencialmente, ser criado empregando o editor do Scilab, o Scipad.

O primeiro passo será abrirmos o editor Scipad (Figura 3.16). Ele pode ser aberto digitando na linha de comando Scipad ou clicando sobre o respectivo ícone (ver Figura 3.2).

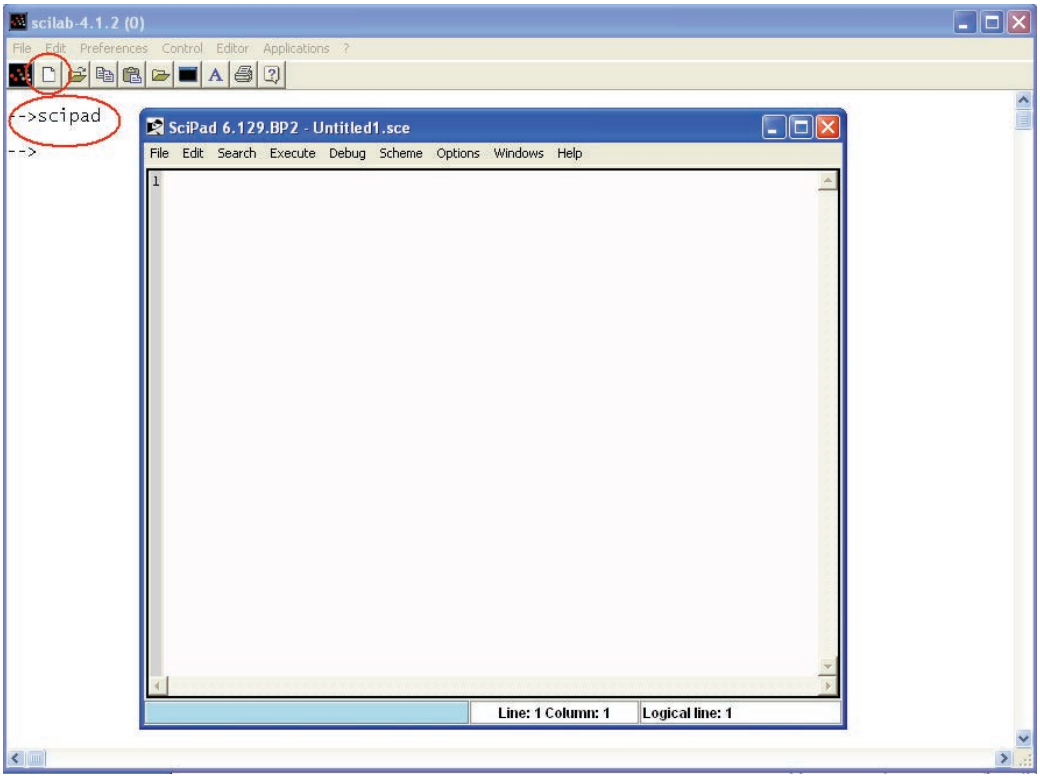


Figura 3.16 O editor Scipad pode ser aberto de duas formas: digitando-se scipad na linha de comando ou clicando sobre o ícone destacado.

Um ponto muito importante é o local (diretório) onde os arquivos criados pelo editor Scipad serão gravados. Vale lembrar que o Scilab só executará os arquivos (do tipo script ou função) localizados no diretório de trabalho. Por exemplo, para sabermos para qual diretório o Scilab está apontando (local onde irá procurar os arquivos) utilize o comando **pwd**. Por padrão (default) ao iniciarmos

o Scipad ele irá salvar os arquivos no atual diretório de trabalho do Scilab. Para esclarecer tomemos como referência a Figura 3.17.

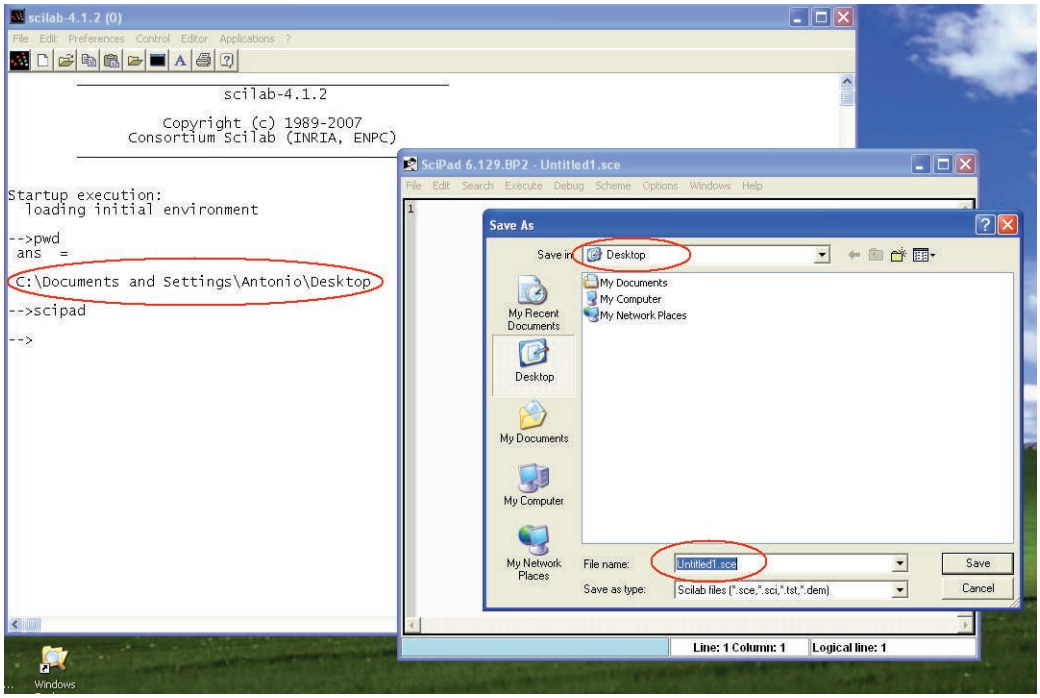


Figura 3.17 Procedimento de inicialização do Scilab, identificação do diretório de trabalho (comando **pwd**), inicialização do editor Scipad e local onde os arquivos serão gravados.

Nesta figura vemos a indicação do diretório de trabalho do Scilab obtida com o comando pwd. Em seguida, foi aberto o editor Scipad (digitando na linha de comando scipad). No editor Scipad, clique na opção File (canto superior esquerdo) e, em seguida, escolha a opção (“Save as”, 6ª linha). Podemos verificar que o editor Scipad irá salvar o arquivo (“Untitled.sce”) no mesmo diretório de trabalho do Scilab (“Desktop”).

O usuário poderá escolher o diretório onde deseja salvar seus arquivos, contudo precisa lembrar que para o Scilab executar estes arquivos precisa apontar para o diretório escolhido.

Criando scripts

Retomemos o primeiro fluxograma elaborado na unidade 2. Neste exemplo elaboramos algoritmo para calcular a área de um triângulo (a) empregando a fórmula de Herão (ou de Heron) a partir dos comprimentos dos três lados do triângulo (a, b, c).

Fórmula para cálculo da área:

$$\text{área} = \sqrt{s \cdot (s-a) \cdot (s-b) \cdot (s-c)}, \text{ em que } s = \frac{(a+b+c)}{2}.$$

Abra o editor Scipad digitando na linha de comando do Scilab: Scipad(). Em seguida, digite o programa. Atribua um nome ao programa salvando-o em algum diretório do computador. A Figura 3.18 apresenta a implementação do algoritmo na forma de um script no editor Scipad. Para executar este programa no Scilab é preciso executar o seguinte comando no prompt do Scilab:

```
–> exec("Alg-Heron.sce")        [enter]
```

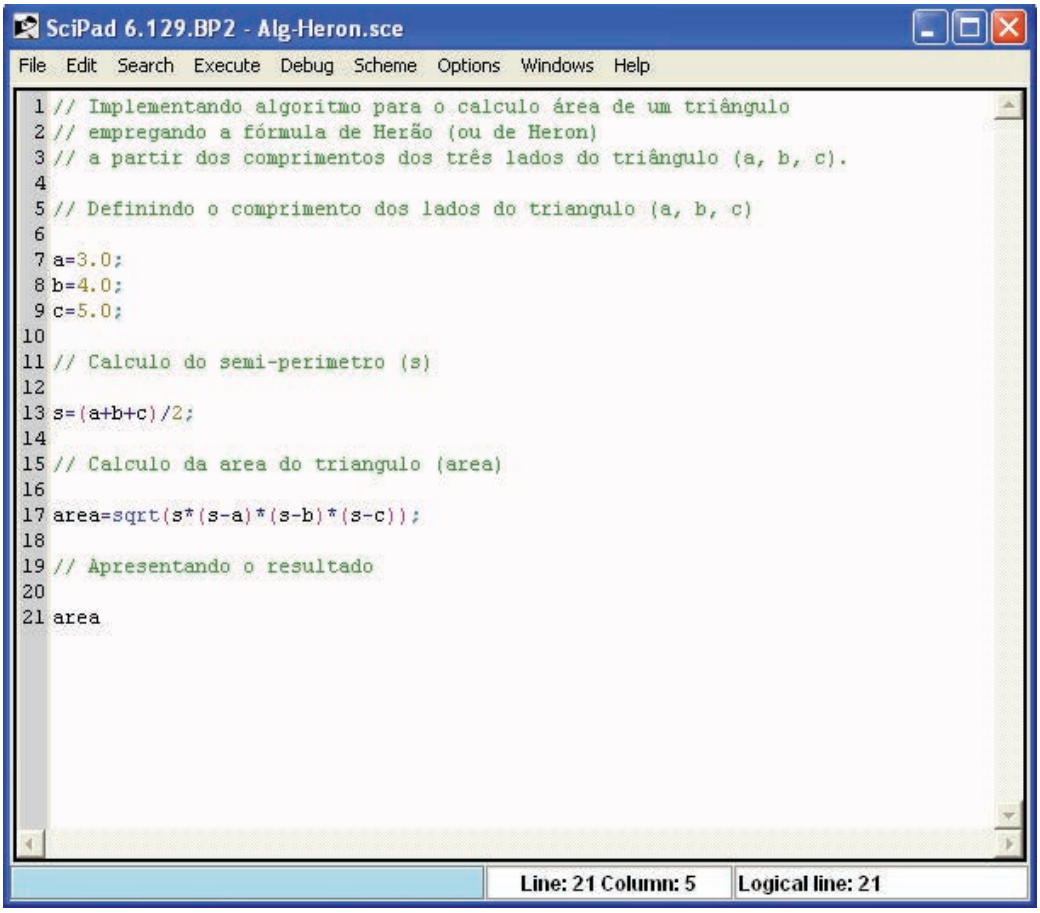


Figura 3.18 Implementação em forma de programa do tipo script do algoritmo para cálculo da área de um triângulo a partir do comprimento dos lados.

Alternativamente você pode executar o programa clicando em File > Exec ..., e escolher selecionar o programa "Alg-Heron.sce".

As próximas figuras (3.19 a 3.20) apresentam a implementação dos exemplos 2 e 3 da unidade 2.

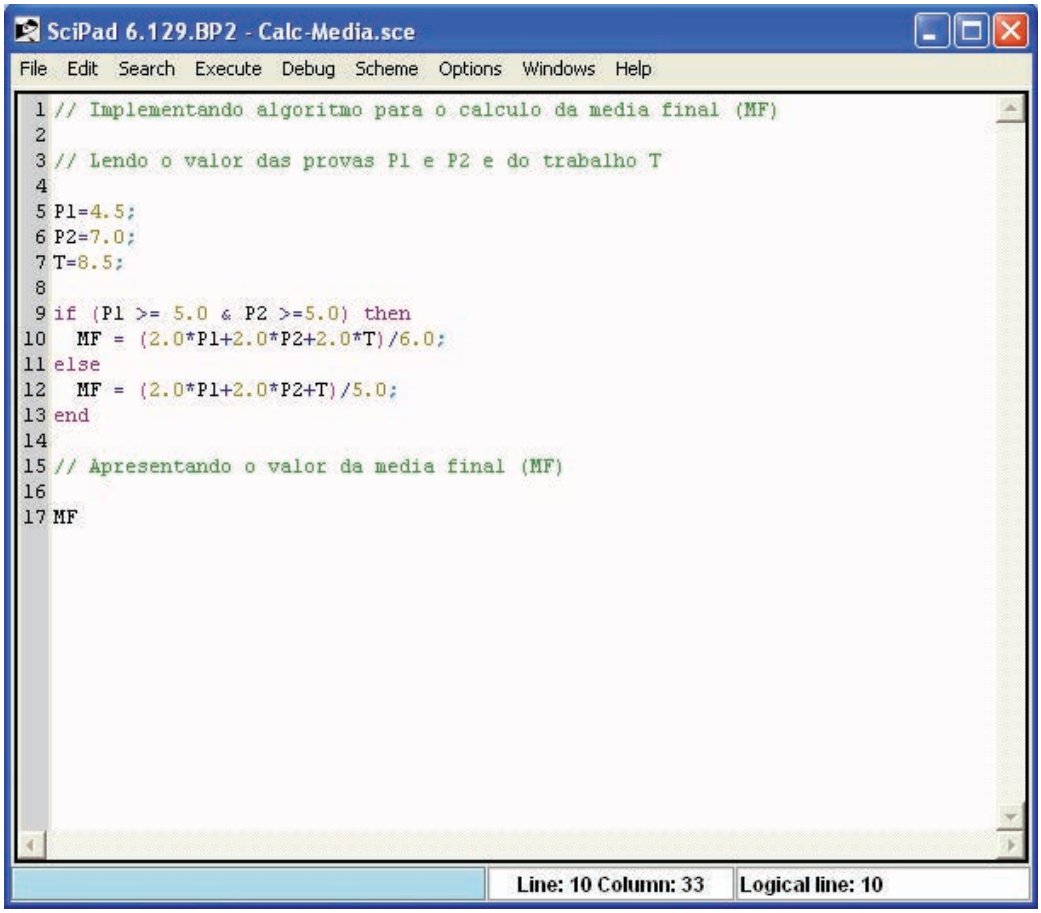


Figura 3.19 Implementação em forma de programa do tipo script do algoritmo para cálculo da média final (ver exemplo 2, unidade 2).

Digite os textos contendo a programação no editor do Scilab. Salve os dois programas atribuindo nomes a eles. Em seguida execute-os com o seguinte comando:

Para o programa do calculo da média:

```
–> exec(“Calc-Media.sce”)          [enter]
```

Para o programa de determinação das raízes da equação de segundo grau:

```
–> exec(“Segundo-grau.sce”)        [enter]
```

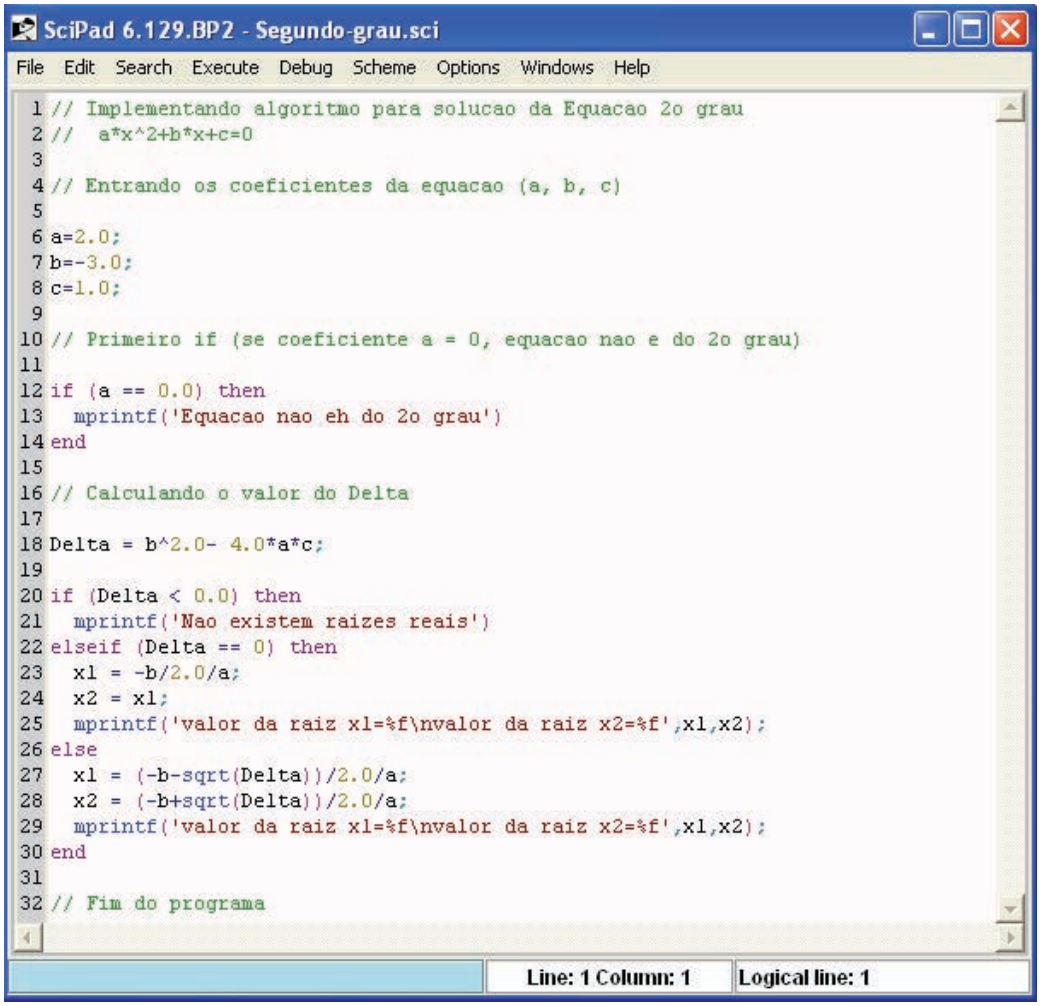


Figura 3.20 Implementação em forma de programa do tipo script do algoritmo para cálculo das raízes da equação do segundo grau (ver exemplo 3, unidade 2).

Criando programas do tipo função

Uma função é um arquivo com a extensão sci, com entradas e saídas bem definidas e uma seqüência de comandos. Quando chamada a execução no ambiente do Scilab os comandos são processados (ou interpretados). O arquivo deverá, preferencialmente, ser criado empregando o editor do programa Scilab: o Scipad.

Uma função obedece a uma estrutura da forma:

```
function [y1, y2, y3, ..., yn] = nome_da_funcao(x1, x2, x3,
...,xn)
  instrucao_1
  instrucao_2
  ...
  Instrucao_n
endfunction
```

Em uma função as variáveis declaradas são variáveis locais, ou seja, não valem no ambiente do Scilab. Em um script as variáveis empregadas são variáveis globais, ou seja, mantêm seu valor no ambiente do Scilab.

Uma função após ser implementada pode ser chamada a qualquer momento a partir da linha de comando do Scilab, de um script ou mesmo por outra função.

A Figura 3.21 apresenta a implementação do algoritmo na forma de uma função no editor Scipad. Para executar este programa no Scilab é preciso em primeiro lugar declarar a função criada e em seguida executá-la. Os seguintes comandos são digitados no prompt do Scilab, para versões anteriores à versão 5.3:

```
–> exec Alg-Heron.sci   [enter]     (1º comando)
–> [area] = Heron(4,3,2)   [enter]      (2º comando)
```

Fica claro neste tipo de programa as entradas: valores a, b e c, e a saída: valor da área calculada pela função.

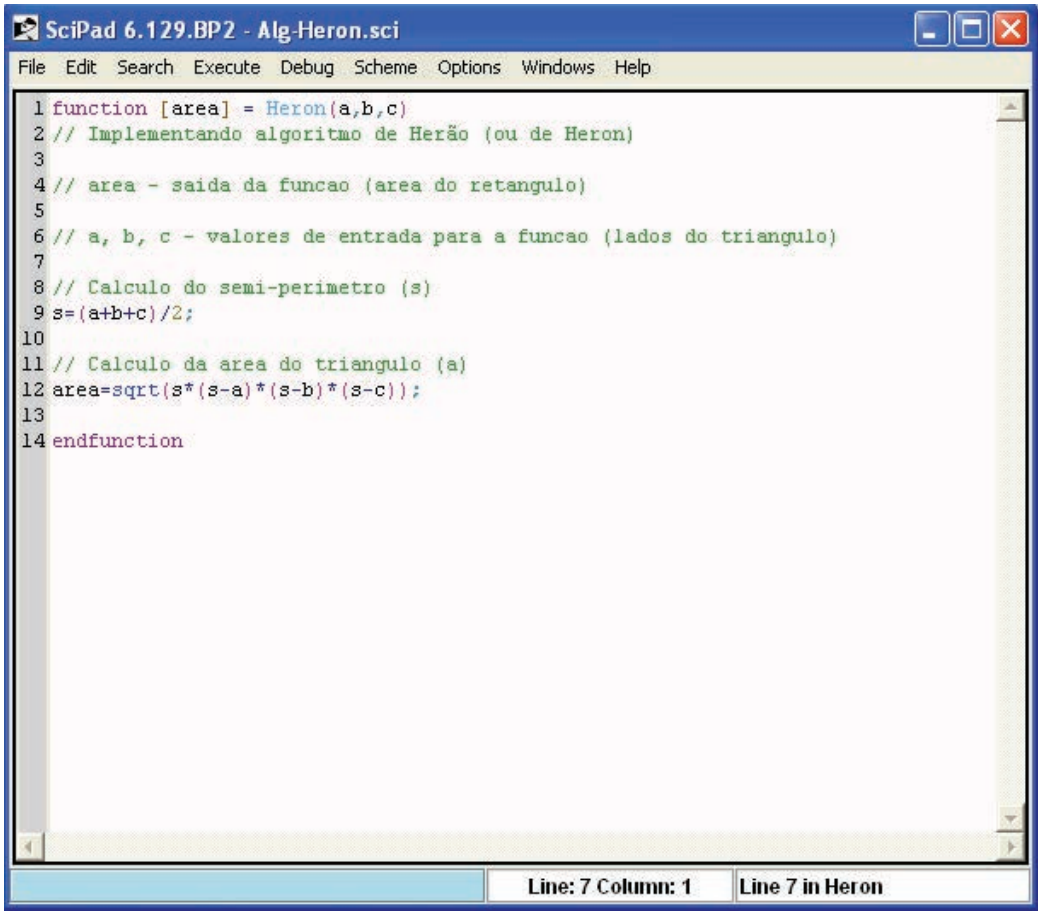


Figura 3.21 Implementação em forma de programa do tipo função do algoritmo para o cálculo da área de um triângulo a partir do comprimento dos lados.

As próximas figuras (3.22 a 3.23) apresentam a implementação dos exemplos 2 e 3 da unidade 2 na forma de programas do tipo função.

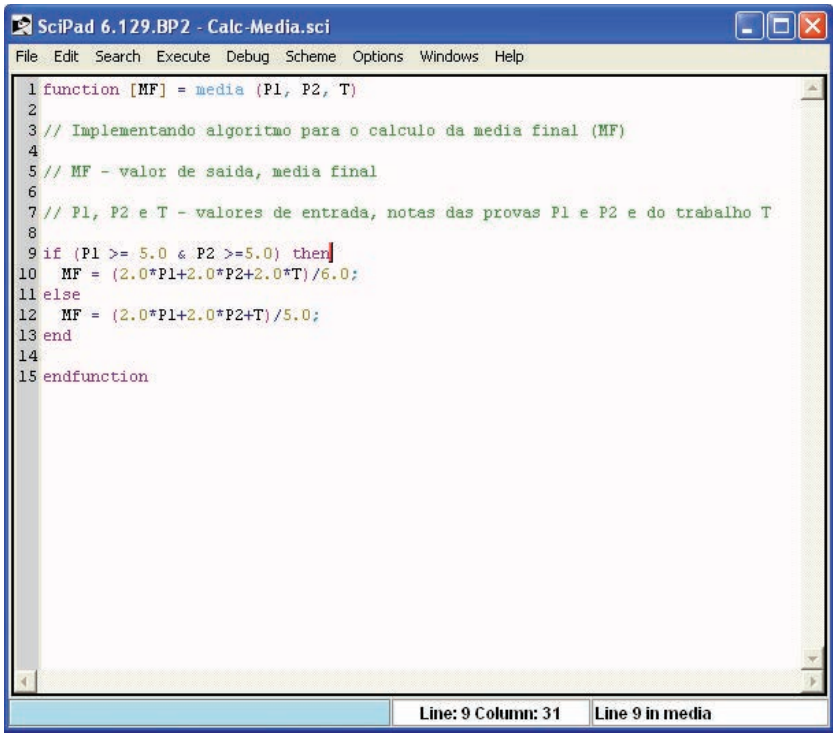


Figura 3.22 Implementação em forma de programa do tipo função do algoritmo para cálculo da média final (ver exemplo 2, unidade 2).

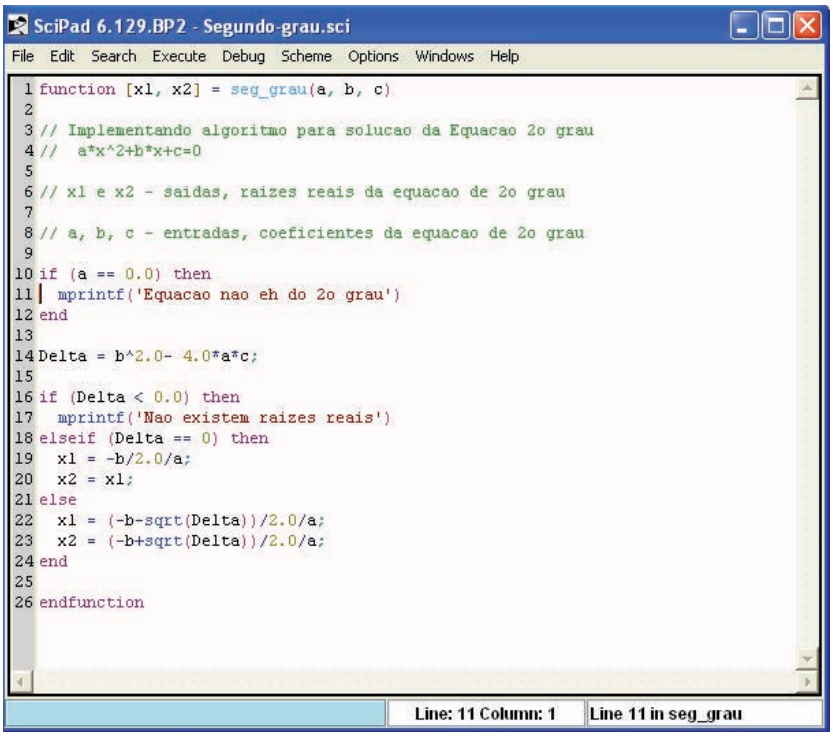


Figura 3.23 Implementação em forma de programa do tipo função do algoritmo para o cálculo das raízes da equação do segundo grau (ver exemplo 3, unidade 2).

3.10 Considerações finais

Vimos nesta unidade como resolver problemas simples com auxílio de uma ferramenta computacional: o aplicativo Scilab. Com a utilização de um número pequeno de comandos e funções pré-definidas é possível elaborar programas para resolver, de forma automática, problemas.

3.11 Referências bibliográficas

CARO, A. A. SEPÚLVEDA, C. V. Fundamentos de Scilab y Aplicaciones. Versão 0.1, 2004. Disponível em: <http://www.scilab.org/publications/index_publications.php? page=freebooks >. Acesso em: 13 de agosto de 2008.

PIRES, P. S. M. *Introdução ao Scilab*. Versão 3.0, 2004. Disponível em: <http://www.dca.ufrn.br/~pmotta>. Acesso em: 15 de junho de 2008.

Manual de Ajuda do programa Scilab v. 4.1.2.

UNIDADE 4

Planilhas eletrônicas

4.1 Primeiras palavras

Caro aluno(a): Na unidade anterior utilizamos o programa Scilab como ferramenta computacional para programação de algoritmos. Nesta unidade faremos a apresentação de mais um aplicativo que também pode ser utilizado com esta finalidade: a planilha eletrônica chamada Calc do pacote de programas para escritório BrOffice.org. Optou-se por trabalhar com este programa pelo mesmo motivo já mencionado: trata-se de programa distribuído livremente, e também pelo fato do programa Calc apresentar similaridades com a planilha Excel que faz parte do pacote computacional Office comercializado pela Microsoft.

No aplicativo BrOffice.org Calc é possível elaborar vários tipos de gráficos, trabalhar com matrizes, resolver sistemas de equações lineares e não lineares. O programa possui várias funções (matemáticas, estatísticas, etc) pré-definidas. O conhecimento básico deste programa fornecerá os subsídios necessários para resolução de problemas típicos de um curso de engenharia. O maior conhecimento desta ferramenta, permitirá que o usuário possa utilizá-lo na solução de problemas mais complexos. O programa BrOffice.org Calc dispõe de um sistema de ajuda ao usuário (Ajuda) bastante prático (em português).

4.2 Problematizando o tema

Nesta unidade será apresentado o programa de planilha eletrônica BrOffice.org Calc. O objetivo será iniciar os primeiros passos nesta importante ferramenta, fornecendo subsídios para sua utilização na solução de problemas em outras disciplinas.

Serão resolvidos neste programa os exemplos apresentados na unidade 2.

4.3 Introdução às planilhas eletrônicas

As planilhas na forma de papel são conhecidas há muito tempo. Porém, com o advento dos computadores e a sua rápida evolução, as planilhas saíram do papel e lápis e passaram para a forma digital, na forma de programas de computador.

A primeira planilha eletrônica comercial surgiu no final da década de setenta e tinha o nome de Visicalc. Foi elaborada por Dan Bricklin. É um tipo de programa de computador que utiliza tabelas para realização de cálculos ou apresentação de dados. Cada tabela da planilha é formada por uma grade composta de um determinado número de linhas e colunas. O nome eletrônica se deve à sua implementação por meio de programas de computador. O surgimento deste

programa contribuiu de forma significativa para que os computadores passassem a ser encarados como ferramentas de negócios.

Hoje existem no mercado vários programas baseados em planilhas sendo o mais conhecido o Excel, integrante do pacote para escritório Office da empresa Microsoft.

Contudo, nesta disciplina optou-se por utilizar o programa Calc do pacote para escritório BrOffice.org. Uma das vantagens deste pacote em relação ao similar comercial (Office) deve se ao fato de ser uma versão distribuída gratuitamente. O download (cópia a partir de um endereço na internet) deste aplicativo em sua versão 2.4, incluindo o programa Calc, é realizado a partir do seguinte endereço eletrônico: <http://www.broffice.org/download>.

Arquivos contendo a documentação destes programas são encontrados no seguinte endereço eletrônico:

<http://www.broffice.org/docs>.

4.4 O ambiente de trabalho do programa BrOffice.org Calc

Para utilizarmos o aplicativo Calc é preciso, em primeiro lugar, instalar o pacote BrOffice.org. Uma vez instalado no computador, este aplicativo pode ser iniciado a partir do seguinte caminho (padrão da instalação do pacote BrOffice.org):

INICIAR > TODOS OS PROGRAMAS > Br.Office.org 2.4 > Br.Office.org Calc

Após a execução deste comando o programa Calc tem início, sendo exibida a tela ilustrada na Figura 4.1.

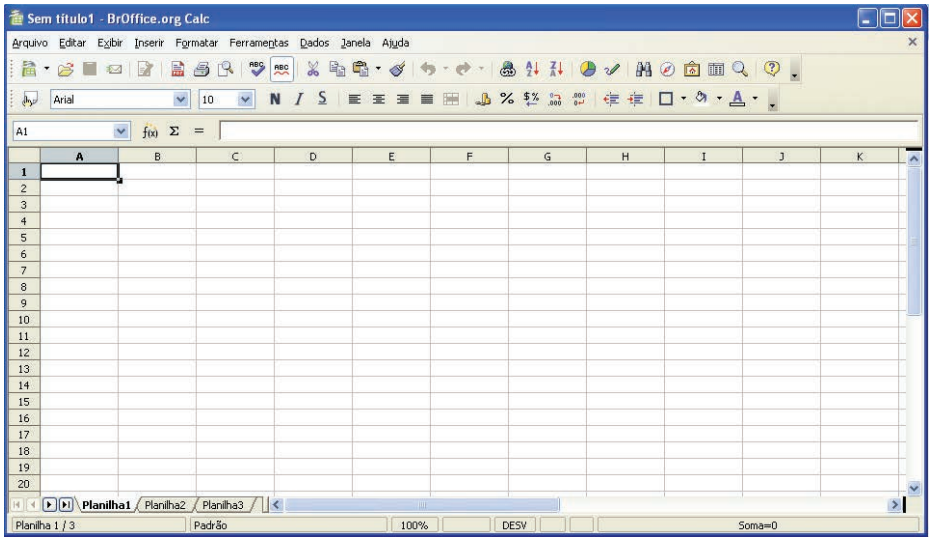


Figura 4.1 Tela inicial do programa BrOffice.org Calc.

A planilha é toda a área quadriculada, sendo que cada quadro recebe o nome de célula. A posição de cada célula na planilha é referenciada por seu índice (ou rótulo). Os índices de linhas são numéricos e estendem-se de “1” a “65536”. Os índices de coluna são alfabéticos e começam com a letra “A” seguindo até a combinação “IV”. Para irmos até a última linha ou até a última coluna basta apertar a tecla <Ctrl> juntamente com as setas para direita (→) ou para baixo (↓), respectivamente. Para retornarmos ao ponto de partida utilizamos as teclas (←) ou (↑), mantendo-se a tecla <Ctrl> pressionada.

Na Figura 4.2 destacamos alguns dos componentes que compõe a área de trabalho do programa BrOffice.org Calc.

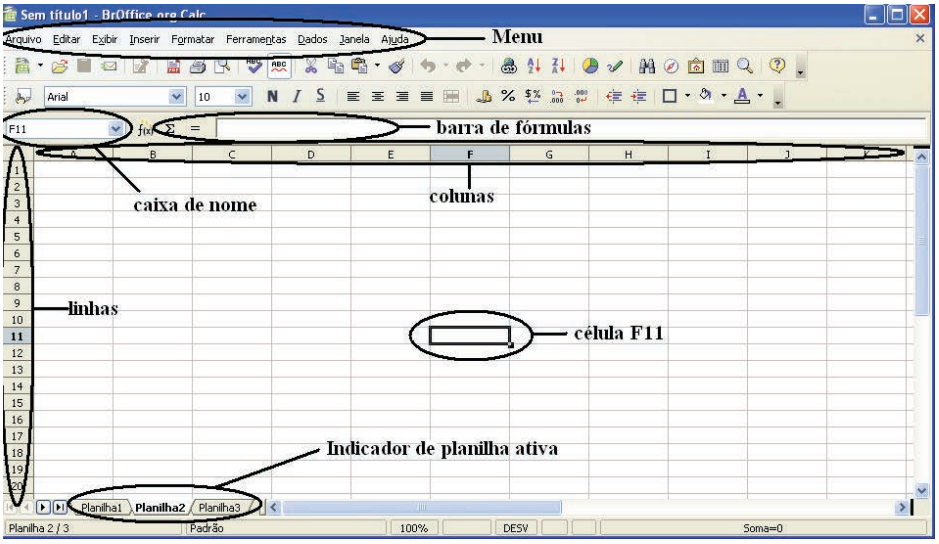


Figura 4.2 Alguns componentes da área de trabalho do programa BrOffice.org Calc.

- Barra de Menu (ou Menu): apresenta os nomes dos menus para acesso às listas de comandos e funções;
- Caixa de nome: apresenta o nome da célula ativa (no exemplo a célula F11);
- Barra de Fórmulas: apresenta as informações de uma determinada célula, fórmulas, etc;
- Célula: é a unidade de preenchimento de valores;
- Indicador de planilha ativa: indica a planilha que está em uso.

Quando se abre um arquivo novo no BrOffice.org Calc aparecem sempre três planilhas (padrão). É possível trabalhar com mais de três em um mesmo arquivo.

Como primeiro exercício, vá até a Barra de Menu na opção <Ajuda>, clique sobre esta opção e, em seguida na opção <O que é isto?>. Observe que o cursor adquiriu um sinal de interrogação. Agora, passe o cursor lentamente sobre os vários objetos (ícones) da área de trabalho. Em cada um deles surge uma caixa de texto amarela contendo uma explicação (Figura 4.3). Aconselhamos o leitor a realizar esta atividade com bastante calma percorrendo toda a área de trabalho identificando cada componente e sua função.

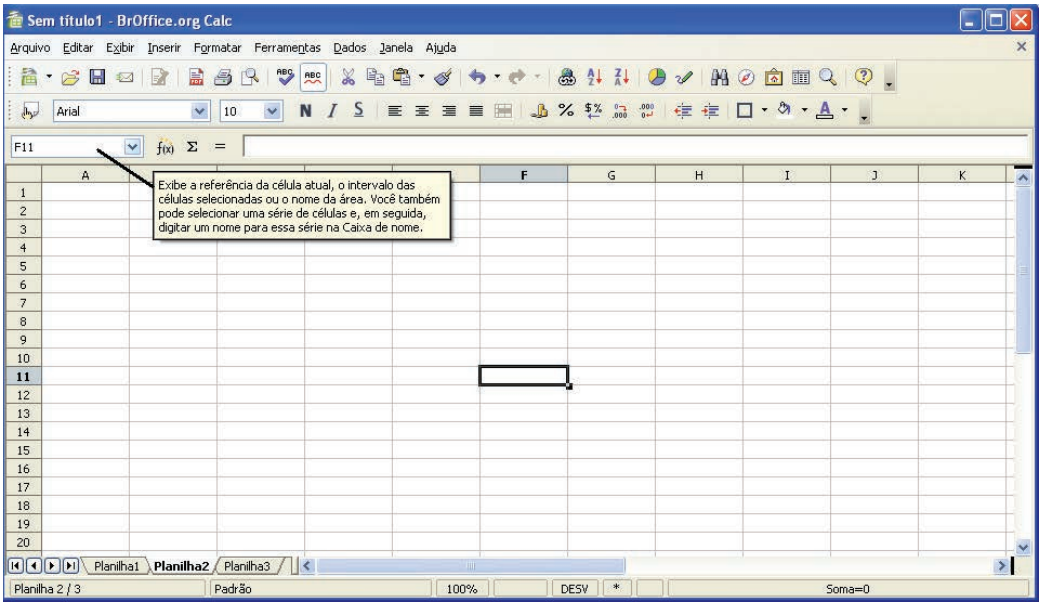


Figura 4.3 Descobrindo a área de trabalho a opção <O que é isto?> ativa. O cursor do mouse encontra-se posicionado sobre a caixa de nome.

O usuário pode a qualquer momento solicitar a ajuda do aplicativo BrOffice.org Calc clicando na Barra de Menu sobre <Ajuda> e em seguida <? Ajuda do

BrOffice.org>, ou alternativamente clicando na tecla de atalho <F1>. A tela ilustrada na Figura 4.4 é apresentada ao usuário.

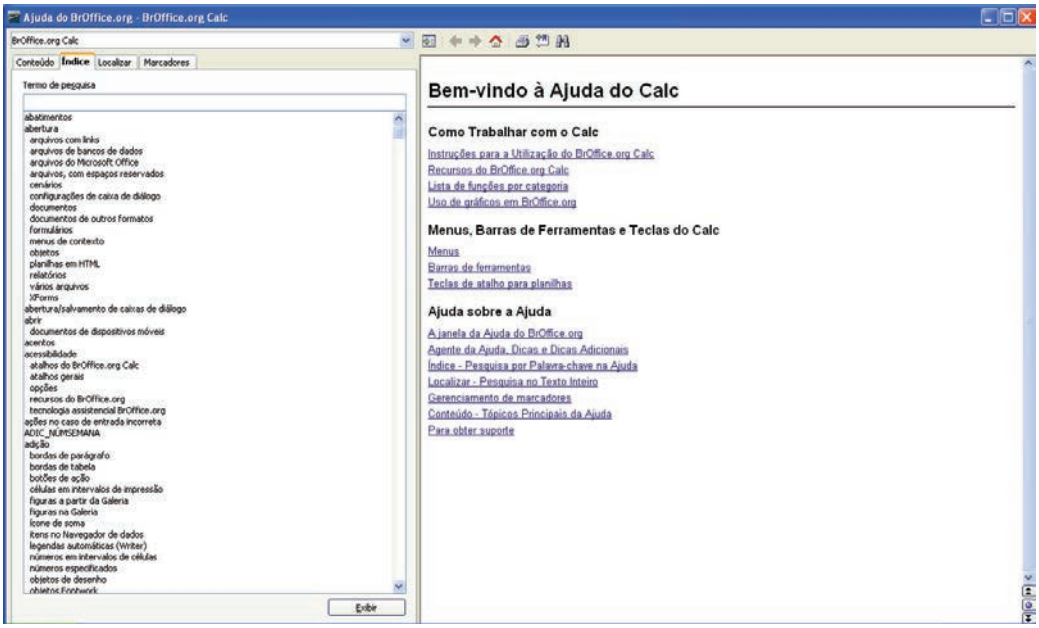


Figura 4.4 Tela de ajuda do BrOffice.org Calc.

O programa BrOffice.org Calc é uma poderosa ferramenta. Nesta unidade iremos nos limitar a apresentar as funções necessárias para implementar os algoritmos desenvolvidos na unidade 2.

4.5 Primeiros passos

No programa BrOffice.org Calc as células podem assumir diferentes categorias (formatos). Para ilustrar melhor, recomendamos que o usuário selecione uma célula qualquer e, em seguida, clique com o botão direito do mouse e escolha a opção <Formatar Células...>. Na Figura 4.5 a célula C10 foi selecionada para formatação.

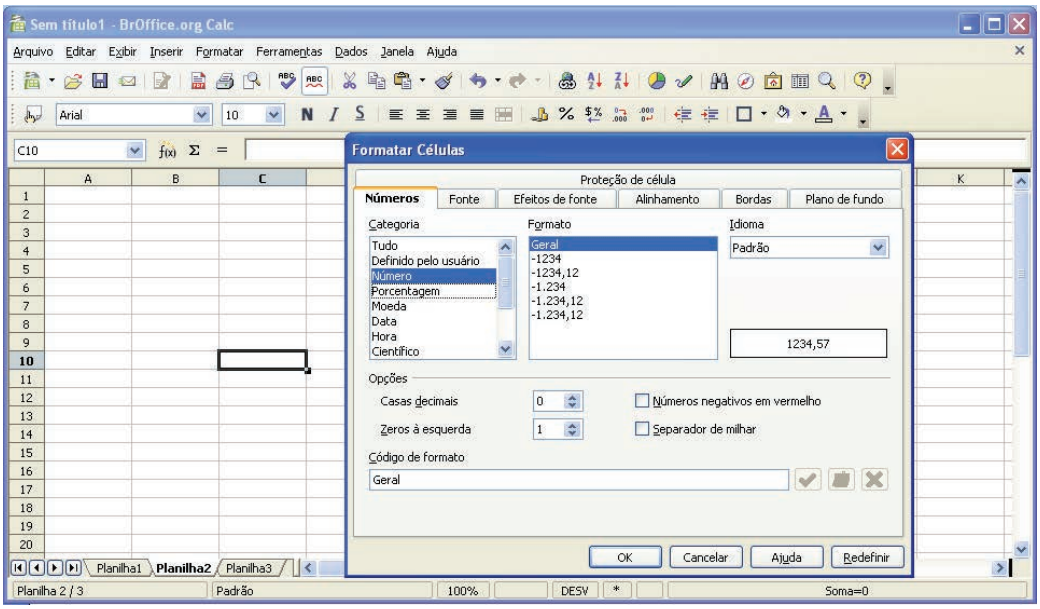


Figura 4.5 Definindo a categoria (ou formato) de uma célula no aplicativo BrOffice.org Calc.

Por padrão (default) as células são todas definidas na categoria número. O usuário pode, nesta janela, escolher o formato de representação do número, a quantidade de casas decimais, de zeros à esquerda, se deseja que números negativos sejam mostrados em vermelho, etc (ver as informações contidas na caixa “Formatar Células”).

Dependendo da configuração do sistema operacional instalado no computador, o BrOffice.org Calc irá utilizar o ponto ou a vírgula como separador do dígito decimal.

4.5.1 Inserindo comentários

Quando foi apresentado o programa Scilab, comentou-se que é sempre conveniente utilizar comentários nos programas, ou seja, documentá-lo para posterior consulta ou para facilitar sua interpretação por outra pessoa que for utilizá-lo.

Em uma planilha um comentário pode ser inserido na forma de texto em qualquer posição. Basta formatar a célula desejada com a categoria texto. Qualquer célula também pode ser formatada com relação à fonte (tipo; tamanho; cor; forma: negrito, sublinhado, itálico; alinhamento: à esquerda, centralizada, à direita), à cor do plano de fundo, por exemplo. Na elaboração de planilhas é sempre conveniente inserir textos explicativos com o objetivo de facilitar sua utilização.

4.5.2 Funções matemáticas

O programa BrOffice.org Calc possui várias funções matemáticas que podem ser facilmente utilizadas.

Para acessar este conjunto de funções, basta selecionar na Barra de Menu o item <Inserir> escolhendo a opção <Inserir Função...>, (Figura 4.6). Pode-se também clicar diretamente sobre o ícone inserir funções .

Além das funções matemáticas o aplicativo disponibiliza outras categorias cada qual contendo várias funções (veja, por exemplo, a categoria estatística e suas funções).

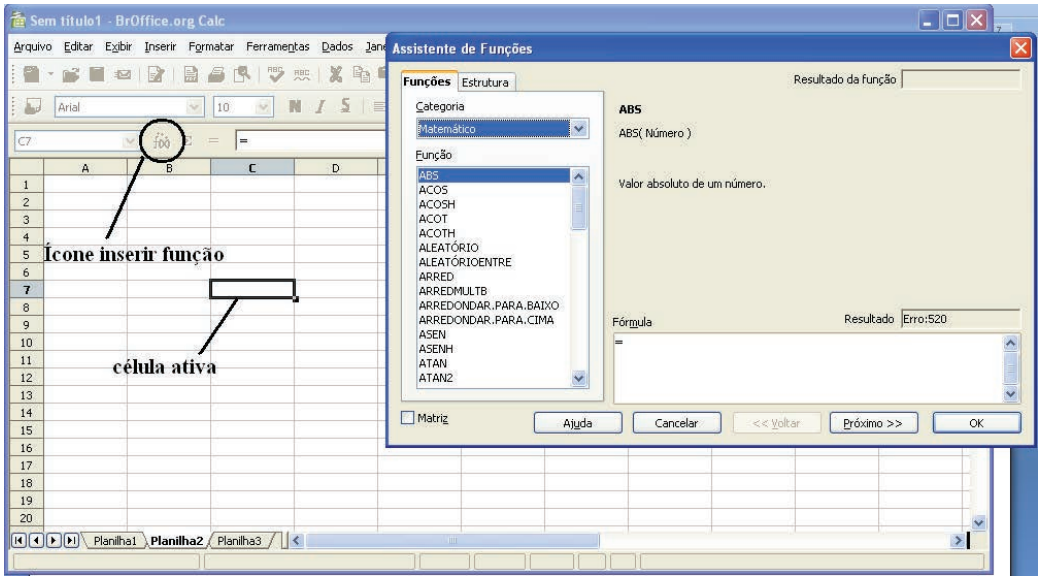


Figura 4.6 Inserindo funções matemáticas. (1) Clicar sobre o ícone inserir funções ( ) para abrir o "Assistente de Funções". No menu categoria escolheu-se a opção <Matemático>. No menu função (logo abaixo) encontram-se listadas todas as funções existentes para esta categoria. Neste exemplo encontra-se selecionada a função valor absoluto de um número (ABS(Numero)).

Experimente solicitar a ajuda do programa pressionando a tecla F1. Digite no campo <Termo de pesquisa> a palavra: "funções" e clique no termo logo abaixo "Assistente de funções". Na página à direita, clique no termo "Lista de Categorias e Funções", e na próxima página clique no termo "Matemático". Abrir-se-á outra página com a descrição e sintaxe de todas as funções matemáticas do programa BrOffice.org Calc. Listam-se na Tabela 4.1 algumas destas funções.

Tabela 4.1 Exemplo de algumas funções matemáticas do programa BrOffice.org Calc.

Função	Sintaxe
Exponencial de um número	EXP(numero)
Valor da constante Pi	PI()
Fatorial de um número	FATORIAL(numero)
Raiz quadrada positiva de um número	RAIZ(numero)
Seno do ângulo especificado (em radianos)	SEN(numero)
Coseno do ângulo especificado (em radianos)	COS(numero)
Tangente de um ângulo (em radianos)	TAN(numero)
Logaritmo natural de um número com base na constante “e”	LN(numero)
Logaritmo de um número com base 10	LOG10(numero)

Assim como no programa Scilab, é possível obter o valor da constante matemática π (pi) e do número de Euler por meio do uso de funções. Vejamos, para obter o valor numérico da constante pi digite em alguma célula da planilha: “=PI()”, em seguida pressione a tecla <enter> (↵). Para obter o número de Euler digite em outra célula: “EXP(1)”.

4.5.3 Funções lógicas

As funções lógicas são importantes para a implementação de decisão ou seleção. Como foi visto na unidade 3, a sua função é testar condições e desviar o fluxo da programação.

No programa BrOffice.org Calc a principal função lógica é a **SE**. Esta função especifica um teste lógico a ser efetuado. Sua sintaxe é:

```
= SE(Teste; Valor_se_verdadeiro; De outra forma_valor)
```

em que:

- Teste: expressão ou valor que pode ser VERDADEIRO ou FALSO;
- Valor_se_verdadeiro: valor retornado se o teste lógico for VERDADEIRO;

- De outra forma_valor: valor retornado se o teste lógico for FALSO.
- Outras duas são a função E e a função OU. Vejamos sua sintaxe:
- Função E: retornará o valor VERDADEIRO se todos os argumentos forem VERDADEIROS. Se um dos elementos for FALSO, esta função retornará o valor FALSO. Sua sintaxe é:

```
= E(Valor lógico 1; Valor lógico 2 ...Valor lógico 30)
```

em que:

- Valor lógico 1; Valor lógico 2 ...Valor lógico 30: são as condições que deverão ser verificadas. Todas as condições podem ser VERDADEIRAS ou FALSAS. Se um intervalo for inserido como um parâmetro, a função utilizará o valor do intervalo contido na coluna ou na linha atual. O resultado será VERDADEIRO se o valor lógico de todas as células do intervalo de células for VERDADEIRO.
- Função OU: retornará o valor VERDADEIRO se pelo menos um argumento for VERDADEIRO. Sua sintaxe é:

```
= OU(Valor lógico 1; Valor lógico 2 ...Valor lógico 30)
```

em que:

- Valor lógico 1; Valor lógico 2 ...Valor lógico 30: são as condições que deverão ser verificadas. Todas as condições podem ser VERDADEIRAS ou FALSAS. Se um intervalo for inserido como um parâmetro, a função utilizará o valor do intervalo contido na coluna ou na linha atual.

Para informações adicionais consulte a Ajuda do BrOffice.org Cal.

4.5.4 Operações básicas

É possível realizar operações matemáticas básicas com os elementos numéricos das células. A soma, a subtração, a multiplicação e a divisão são representadas pelos respectivos símbolos: +, -, *, /.

A exponenciação de um número é realizado com o símbolo ^.

No BrOffice.org Calc, para realizar estas operações devemos digitar na célula destino a operação desejada. Por exemplo: suponha que nas células A1 e B1 tenhamos os valores 3 e 5. Para realizar a soma destes dois valores devemos escolher uma célula destino, por exemplo, a célula C1, e digitar a operação nesta célula: "=A1+B1" (somente o texto entre as aspas). Em seguida, digite <enter> (↵). A Figura 4.7 ilustra estas operações básicas.

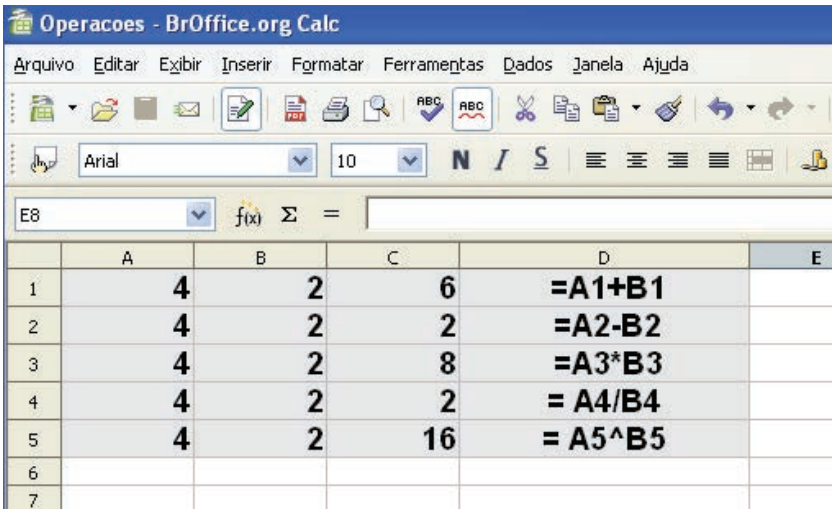


Figura 4.7 Operações de soma, subtração, multiplicação, divisão e exponenciação realizadas na planilha BrOffice.org Calc.

A hierarquia na realização dos operadores algébricos segue a mesma ordem apresentada (ver Figura 3.8, unidade 3).

Existem também os operadores de comparação (ou lógicos). Estes operadores não servem para fazer contas. Eles apenas retornam os valores Verdadeiro ou Falso. A Figura 4.8 apresenta a sintaxe dos operadores lógicos utilizados pelo programa Calc.

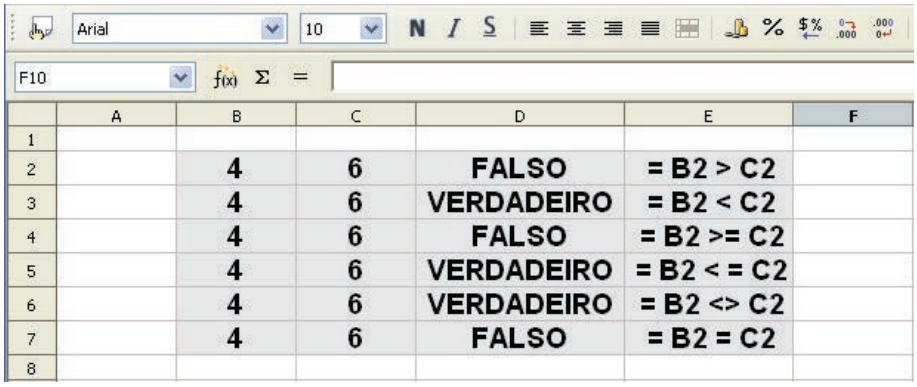


Figura 4.8 Operadores lógicos do aplicativo BrOffice.org Calc.

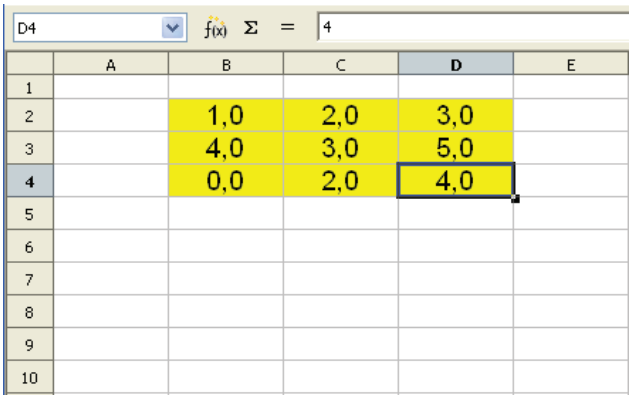
4.6 Trabalhando com vetores e matrizes

O programa BrOffice.org Calc possui um conjunto de funções, dentre elas a biblioteca de funções matriciais. É possível realizar várias operações empregando matrizes. Vejamos alguns exemplos.

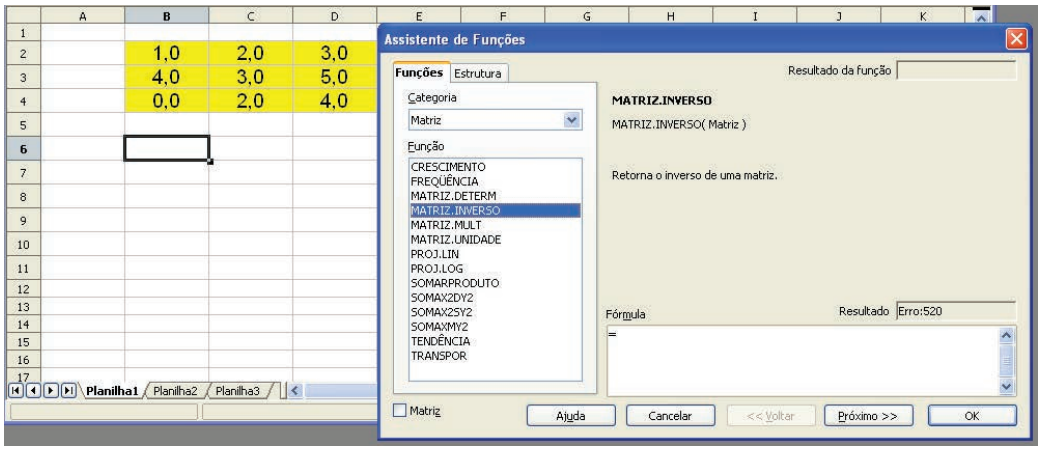
Suponhamos a seguinte matriz, contendo três linhas e três colunas:

$$A = \begin{bmatrix} 1.0 & 2.0 & 3.0 \\ 4.0 & 3.0 & 5.0 \\ 0.0 & 2.0 & 4.0 \end{bmatrix}_{3x3}$$

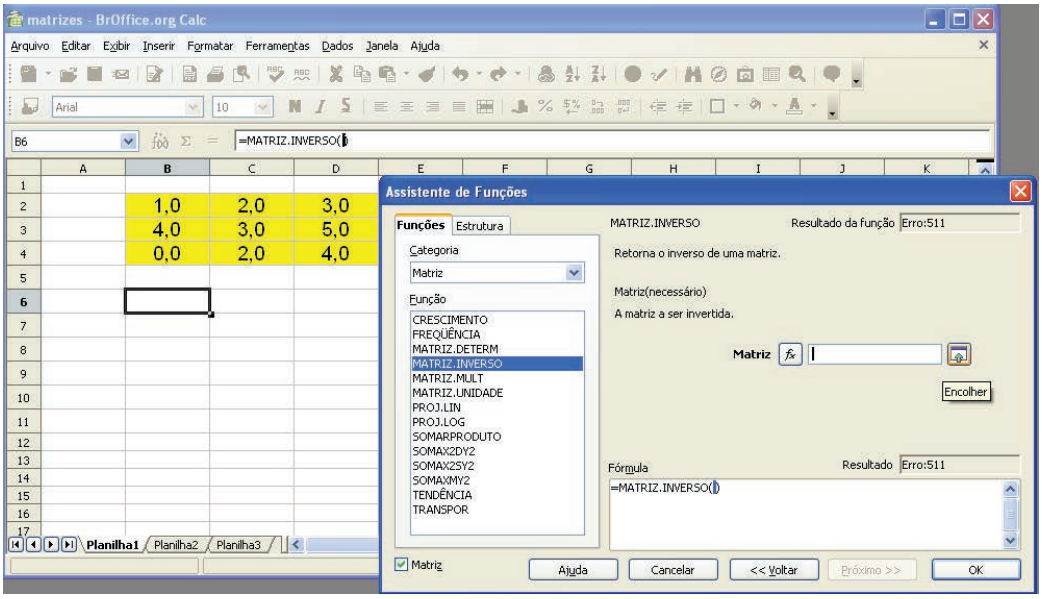
Para calcularmos a inversa da matriz A ,o primeiro passo consiste em digitarmos os valores dos elementos desta matriz nas células da planilha. Em seguida, clicamos no botão de função,  , fazendo com que o assistente de funções seja aberto. Escolhemos na caixa de texto “Categoria”: Matriz. Dentre as opções disponíveis, selecionamos a função MATRIZ.INVERSO. Em seguida, clicamos no botão <Próximo>, localizado na parte inferior do assistente de funções. O próximo passo consiste em selecionar as células que compõe a matriz, clicando no botão <Seleção>, Figura 4.9. Selecionamos as células da matriz e clicamos novamente no mesmo botão. Basta agora clicar no botão <OK>, e a matriz inversa será exibida na planilha. A Figura 4.9 ilustra os passos seguidos para o cálculo da matriz inversa.



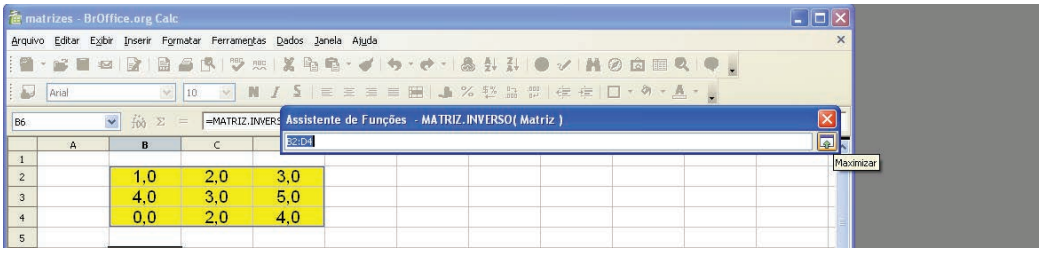
(a)


Assistente de Funções
Funções
Estrutura
Categoria
Matriz
Função
CRESCIMENTO
FREQÜÊNCIA
MATRIZ.DETERM
MATRIZ.INVERSO
MATRIZ.MULT
MATRIZ.UNIDADE
PROJ.LIN
PROJ.LOG
SOMARPRODUTO
SOMAX2DY2
SOMAX2SY2
SOMAXMY2
TENDÊNCIA
TRANSPOR
MATRIZ.INVERSO
MATRIZ.INVERSO(Matriz)
Retorna o inverso de uma matriz.
Resultado da função
Fórmula
Resultado
Erro:520
Matriz
Ajuda
Cancelar
<< Voltar
Próximo >>
OK
Planilha1
Planilha2
Planilha3

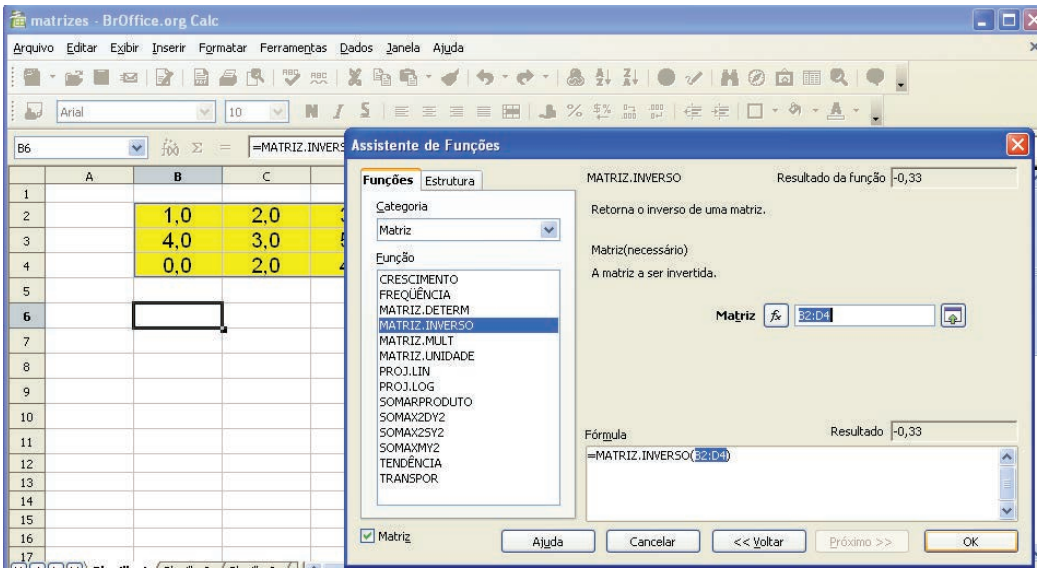
(b)


matrizes - BrOffice.org Calc
Arquivo Editar Exibir Inserir Formatar Ferramentas Dados Janela Ajuda
B6
=MATRIZ.INVERSO(
Assistente de Funções
Funções
Estrutura
Categoria
Matriz
Função
CRESCIMENTO
FREQÜÊNCIA
MATRIZ.DETERM
MATRIZ.INVERSO
MATRIZ.MULT
MATRIZ.UNIDADE
PROJ.LIN
PROJ.LOG
SOMARPRODUTO
SOMAX2DY2
SOMAX2SY2
SOMAXMY2
TENDÊNCIA
TRANSPOR
MATRIZ.INVERSO
Resultado da função
Erro:511
Retorna o inverso de uma matriz.
Matriz(necessário)
A matriz a ser invertida.
Matriz
Encolher
Fórmula
Resultado
Erro:511
=MATRIZ.INVERSO()
Matriz
Ajuda
Cancelar
<< Voltar
Próximo >>
OK
Planilha1
Planilha2
Planilha3

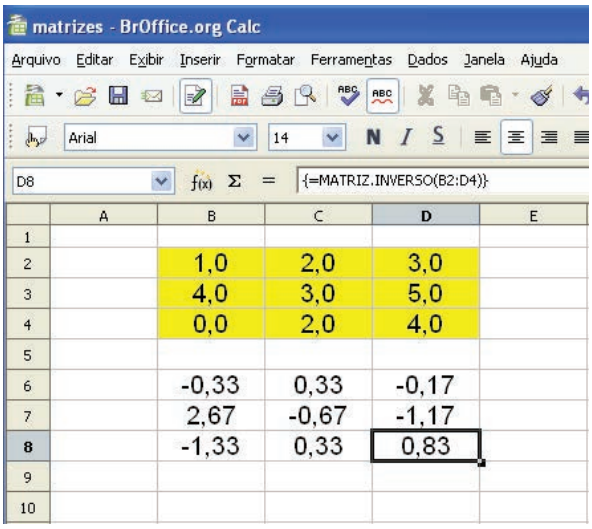
(c)


matrizes - BrOffice.org Calc
Arquivo Editar Exibir Inserir Formatar Ferramentas Dados Janela Ajuda
Assistente de Funções - MATRIZ.INVERSO(Matriz)
Maximizar

(d)



(e)



(f)

Figura 4.9 Passos na realização do cálculo da matriz inversa: (A) Digitando nas células de B2 a D4 os elementos da matriz; (B) Chamando o assistente de funções pelo ícone $f(x)$; (C) Selecionando a categoria Função e dentre elas a função MATRIZ.INVERSO; (D) Selecionando a matriz a ser invertida; (E) Retornando ao assistente de funções e clicando no botão <Próximo>; (F) Resultado da matriz inversa.

Há quatorze funções disponíveis na categoria Matriz, dentre elas: MATRIZ. DETERMINANTE, MATRIZ.INVERSO, MATRIZ.MULTIPLICAÇÃO, MATRIZ. UNIDADE e MATRIZ.TRANSPOSTA.

Exemplo: Resolver um sistema de equações lineares.

Obter a solução do sistema de equações lineares:

$$2x + y - 2z = 10$$
$$3x + 2y + 2z = 1$$
$$5x + 4y + 3z = 4$$

Colocado na forma matricial, tem-se:

$$A = \begin{pmatrix} 2 & 1 & -2 \\ 3 & 2 & 2 \\ 5 & 4 & 3 \end{pmatrix} \qquad x = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \qquad b = \begin{pmatrix} 10 \\ 1 \\ 4 \end{pmatrix}$$

O primeiro passo será inserir a matriz na planilha, células B3 a D5 (representação na planilha B3:D5). Utilizaremos agora a função MATRIZ.DETERMINANTE, para verificar o valor do determinante da matriz **A** (se det(**A**)=0), o sistema de equações não terá solução):

célula B8: "=MATRIZ.DETERM(B3:D5)"

Na célula A11 é realizado um teste condicional para verificar se o sistema tem (det(**A**) 0) ou não tem solução (det(**A**)=0).

célula A11: "=SE(B8<>0;'Sistema tem solução';'Sistema não tem solução')"

Em seguida, iremos calcular com auxílio da função MATRIZ. INVERSO a inversa da matriz **A** ($\mathbf{A}^{-1}$).

células I11:K13: "=MATRIZ.INVERSO(B3:D5)"

A solução será obtida realizando a multiplicação da matriz inversa calculada, $\mathbf{A}^{-1}$, com o vetor **b**. Para isto, a função MATRIZ.MULTIPLICAÇÃO é empregada.

células B15:B17: "=MATRIZ.MULT(I11:K13;G3:G5)"

A Figura 4.10 ilustra a planilha elaborada.

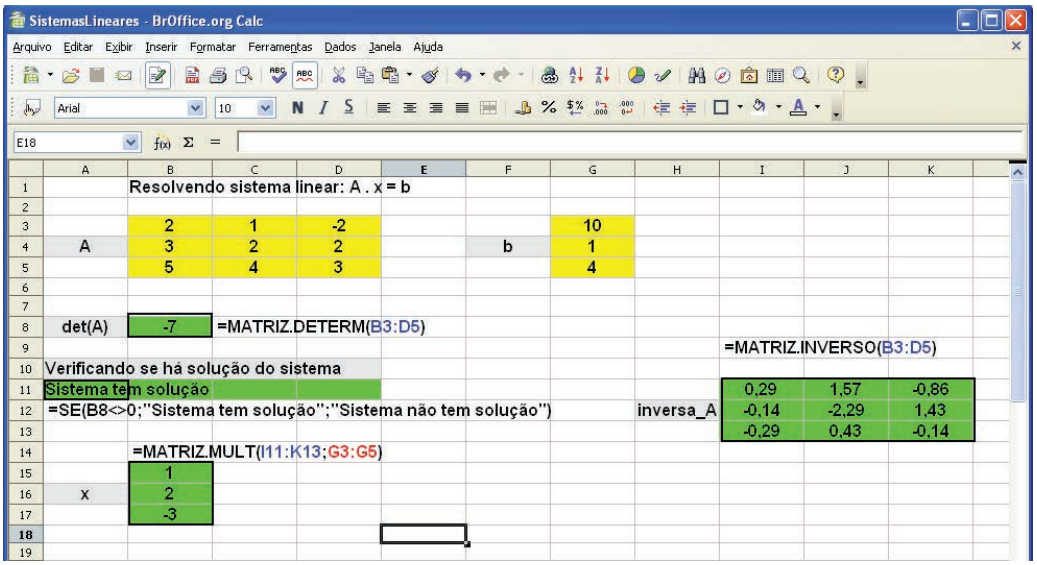


Figura 4.10 Planilha para solução do sistema linear **A**.**x** = **b**.

4.7 Construindo gráficos

A visualização das informações na forma gráfica é facilmente implementada neste aplicativo. O BrOffice.org Calc possui uma capacidade gráfica bem completa, permitindo a visualização dos dados em nove tipos de gráficos. São eles: coluna, barra, pizza, área, linha, XY (dispersão), rede, ações e coluna e linha.

Para construir o gráfico da função $f(x) = x^2 - 4 \cdot x + 3$ no intervalo $-3 \leq x \leq 6$ precisamos construir o vetor **x** na coluna A e o vetor y na coluna B (intervalo de 0,5).

Na célula A1 coloque o texto “x” e na célula B1 “f(x)” (legendas). Para criar o vetor coluna coloque na célula o A2 o valor “–3”. Na célula A3 escreva “=A2+0,5”. Em seguida, com auxílio do mouse, puxe a célula A3 pela alça até a célula A20. Acabamos de criar o vetor x = [–3; –2,5; –2,0; ...; 5,5; 6; 0].

Passemos agora para o vetor **y**. Coloque na célula B2 a equação “=A2^2-4*A2+3”. Clique na tecla <enter>. Com auxílio do mouse, puxe a célula B2 pela alça até a célula B20. Está criado o vetor **y**, com os valores da função f(x).

Para iniciar o assistente de gráfico, selecione na Barra de Menu o item <Inserir> escolhendo a opção <Gráfico...>. Pode-se também clicar diretamente sobre o ícone inserir gráfico, . A Figura 4.11 ilustra os passos iniciais na elaboração do gráfico.

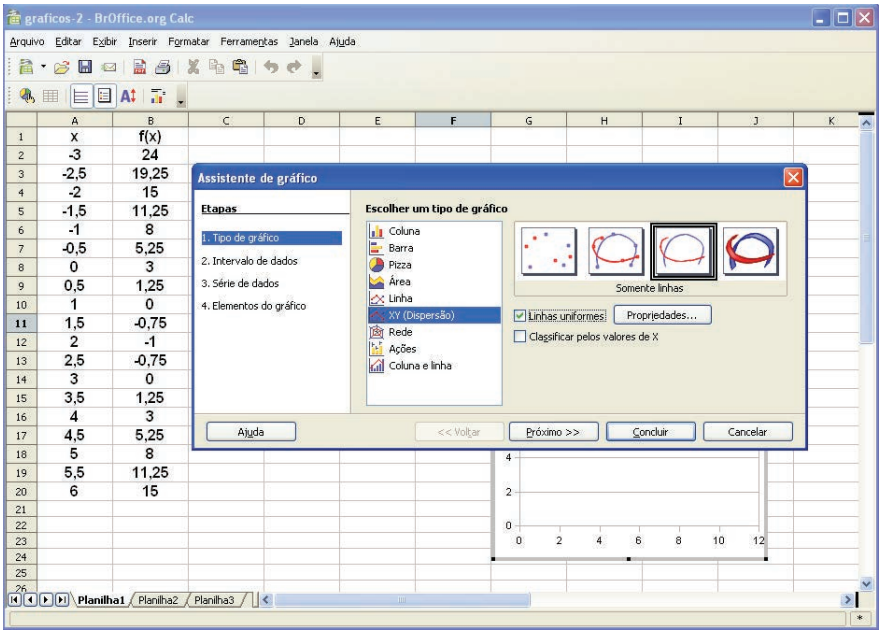


Figura 4.11 Criando o gráfico da função $f(x) = x^2 - 4 \cdot x + 3$ no BrOffice.org Calc com auxílio do assistente de gráfico.

Com o assistente de gráfico aberto, selecione como tipo de gráfico XY(Dispersão), etapa 1. Clique na opção “Linhas uniformes” e em seguida no botão < Próximo> para avançar até a etapa 3 (Série de dados). Adicione uma série de dados clicando no botão <Adicionar>. É necessário agora fornecer os valores de X e os valores de Y. Isto é feito clicando-se no botão e selecionando as células A2 a A20. Faça o mesmo para selecionar as células B2 a B20 como vetor Y (Figura 4.12).

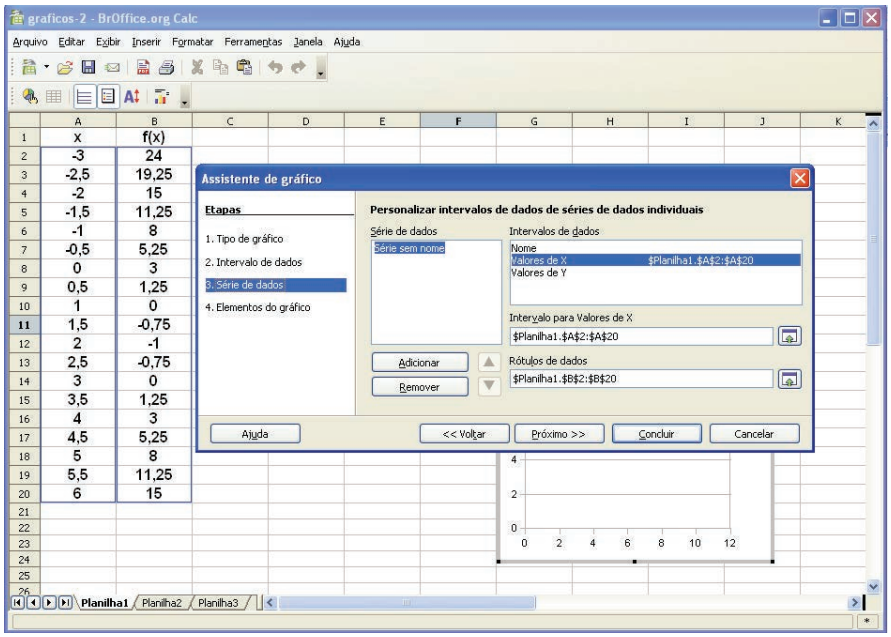


Figura 4.12 Definindo os valores do vetor X e do vetor Y.

Clique no botão <Próximo> para seguir para a quarta etapa: Elementos do gráfico. Insira um título para o gráfico (“Gráfico de f(x)”), para o eixo X (“x”) e para o eixo Y (“y”). Desative as opções “Exibir legenda” e “Exibir grades” nos eixos X e Y (Figura 4.13).

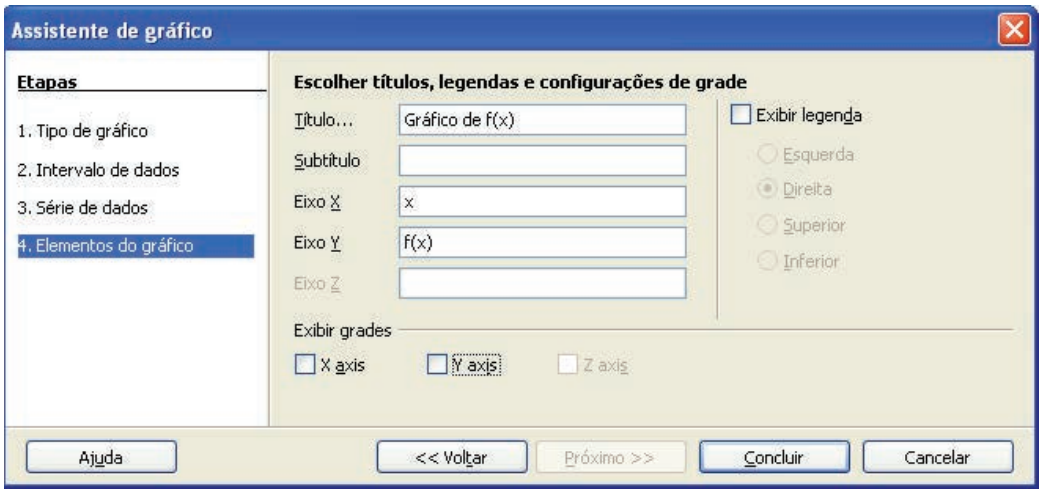


Figura 4.13 Definindo os elementos do gráfico.

Clique no botão <Concluir> para o gráfico ser exibido (Figura 4.14).

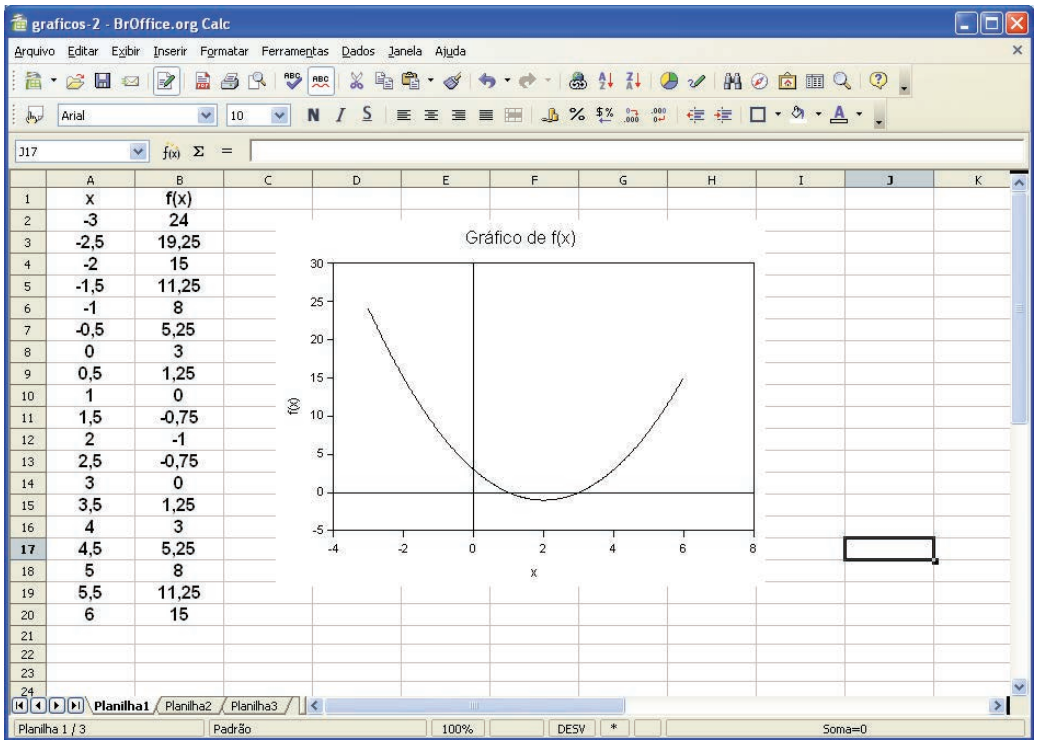


Figura 4.14 Gráfico da função $f(x) = x^2 - 4 \cdot x + 3$ ilustrando com um x os pontos x, f(x) no intervalo $-3 \leq x \leq 6$.

Experimente buscar informações sobre a formatação dos gráficos por meio do comando Ajuda do BrOffice.org Calc.

Outros tipos de gráfico são facilmente construídos com auxílio do “Assistente de Gráfico”.

4.8 “Travando” uma célula na planilha

Muitas vezes quando se pretende implementar a solução de um problema utilizando a planilha de cálculo torna-se necessário empregar algum valor fixo (constante) no cálculo.

Vejamos um exemplo:

Suponhamos que uma pessoa faça uma aplicação de uma determinada quantia financeira (M_0) por um período de 12 meses (n) a uma taxa de juros (i) fixa de 0.6% ao mês. Para calcular o montante final a ser resgatado ao final da aplicação (após 12 meses), o seguinte cálculo é realizado, mês a mês:

$M_1 = M_0 \cdot (1+i)$ (valor ao final do primeiro mês)

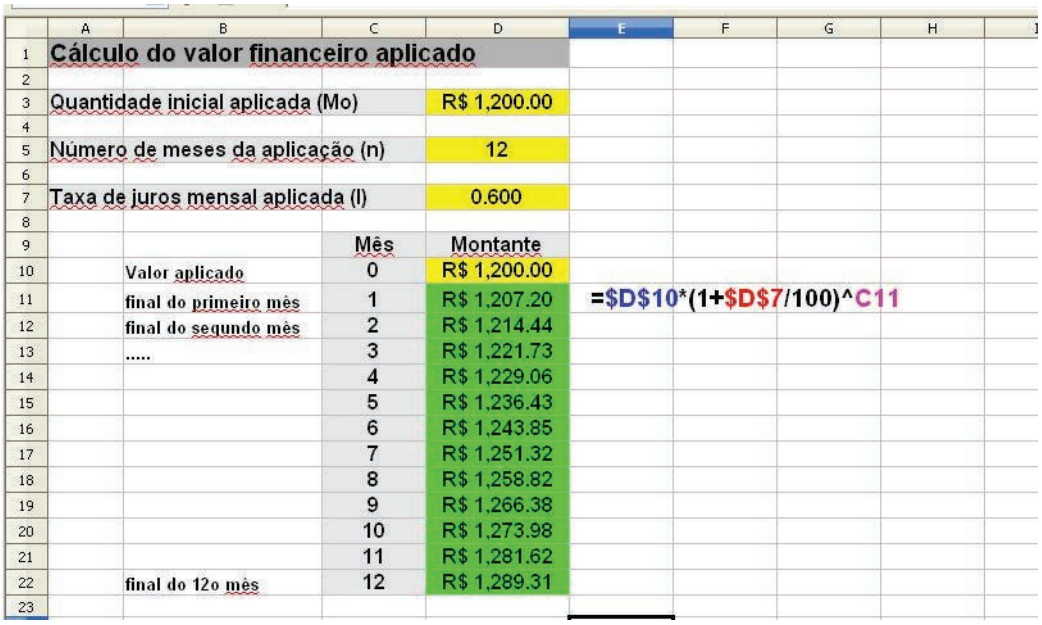
$M_2 = M_1 \cdot (1+i) = M_0 \cdot (1+i) \cdot (1+i) = M_0 \cdot (1+i)^2$ (valor ao final do segundo mês)

$M_3 = M_2 \cdot (1+i) = M_0 \cdot (1+i)^2 \cdot (1+i) = M_0 \cdot (1+i)^3$ (valor ao final do terceiro mês)

…

$M_{12} = M_{11} \cdot (1+i) = M_0 \cdot (1+i)^{11} \cdot (1+i) = M_0 \cdot (1+i)^{12}$ (valor ao final do décimo segundo mês)

A Figura 4.15 apresenta este cálculo implementado na planilha Calc para obter o valor da aplicação mês a mês até o momento do resgate da aplicação.



	A	B	C	D	E	F	G	H
1	Cálculo do valor financeiro aplicado							
2								
3	Quantidade inicial aplicada (Mo)			R$ 1,200.00				
4								
5	Número de meses da aplicação (n)			12				
6								
7	Taxa de juros mensal aplicada (I)			0.600				
8								
9			Mês	Montante				
10		Valor aplicado	0	R$ 1,200.00				
11		final do primeiro mês	1	R$ 1,207.20	=$D$10*(1+$D$7/100)^C11			
12		final do segundo mês	2	R$ 1,214.44				
13		.....	3	R$ 1,221.73				
14			4	R$ 1,229.06				
15			5	R$ 1,236.43				
16			6	R$ 1,243.85				
17			7	R$ 1,251.32				
18			8	R$ 1,258.82				
19			9	R$ 1,266.38				
20			10	R$ 1,273.98				
21			11	R$ 1,281.62				
22		final do 12o mês	12	R$ 1,289.31				
23								

Figura 4.15 Exemplo de um cálculo financeiro implementado na planilha Calc.

Para elaborar esta planilha, os seguintes passos foram seguidos:

1. Formatação das células D3 e D10 a D22 na categoria moeda, formato português (moeda real);
2. Formatação da célula D7 para categoria número com a opção de 3 casas decimais;
3. Fórmula inserida na célula D11 = $D$10*(1+$D$7/100)^C11, correspondente à equação:
4. Puxar a célula D11 pelo *cursor de alça* até a célula D22, obtendo os valores da aplicação financeira mês a mês até o décimo segundo mês.

Observe que foi utilizado o símbolo $ (antes e depois do D) para as células D7 (D) e D10 ($D$10). O uso do símbolo $ faz com que o valor desta célula seja “travado” quando se replica o cálculo para as demais células da coluna (células de D11 a D22).

4.9 Utilizando a ferramenta “Atingir Meta”

A planilha de cálculo Calc dispõe de uma ferramenta chamada de “Atingir Meta”. Vejamos um exemplo de sua aplicação.

Exemplo: Dada uma função f(x), contínua e diferenciável, e que possua raiz(es) no em um intervalo [x1... x2].

Veremos como encontrar o valor desta(s) raiz(es) neste intervalo.

Seja a função f(x) dada por:

$f(x) = x^2 - 4$

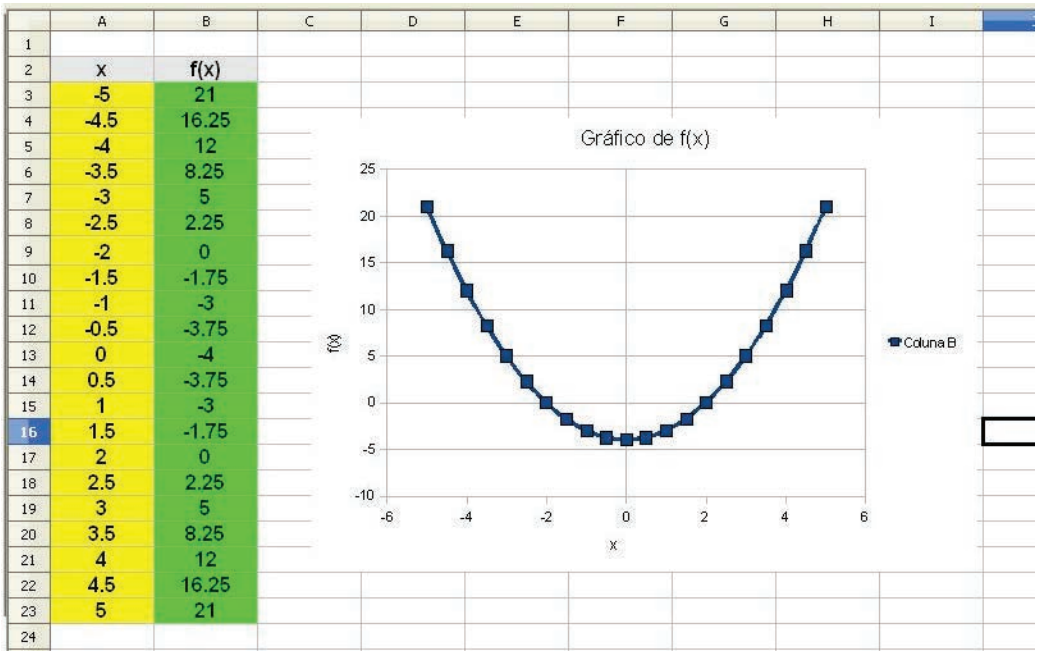
Encontrar a(s) raiz(es) desta função no intervalo [–5.0 ; 5.0].

Solução: O primeiro passo consiste em criarmos um gráfico para a função f(x), aproveitando as facilidades da planilha de cálculo (Figura 4.16).

O próximo passo consiste em utilizar a ferramenta “Atingir Meta” para encontrar as raízes desta função no intervalo [–5.0 ; 5.0]; Neste caso temos duas raízes, –2.0 e 2.0 facilmente visualizadas com auxílio do gráfico na Figura 4.16.

Passos para utilização da ferramenta “Atingir Metas”:

1. Digite, por exemplo, o valor 1.0 na célula B4;
2. Na célula C4, digite a função, ou seja, “=B4^2-4.0”;
3. Vá até o menu **Ferramentas** e escolha a opção **“Atingir Meta”.** Uma caixa de diálogo abre-se (Figura 4.17). Preencha os campos conforme a Figura 4.17 e clique em OK. Outra caixa de diálogo surge (Figura 4.18), mostrando o valor da raiz. Clicando em OK o valor da raiz é inserido na célula B4 (Figura 4.19).



	x	f(x)
3	-5	21
4	-4.5	16.25
5	-4	12
6	-3.5	8.25
7	-3	5
8	-2.5	2.25
9	-2	0
10	-1.5	-1.75
11	-1	-3
12	-0.5	-3.75
13	0	-4
14	0.5	-3.75
15	1	-3
16	1.5	-1.75
17	2	0
18	2.5	2.25
19	3	5
20	3.5	8.25
21	4	12
22	4.5	16.25
23	5	21

Figura 4.16 Gráfico da função .

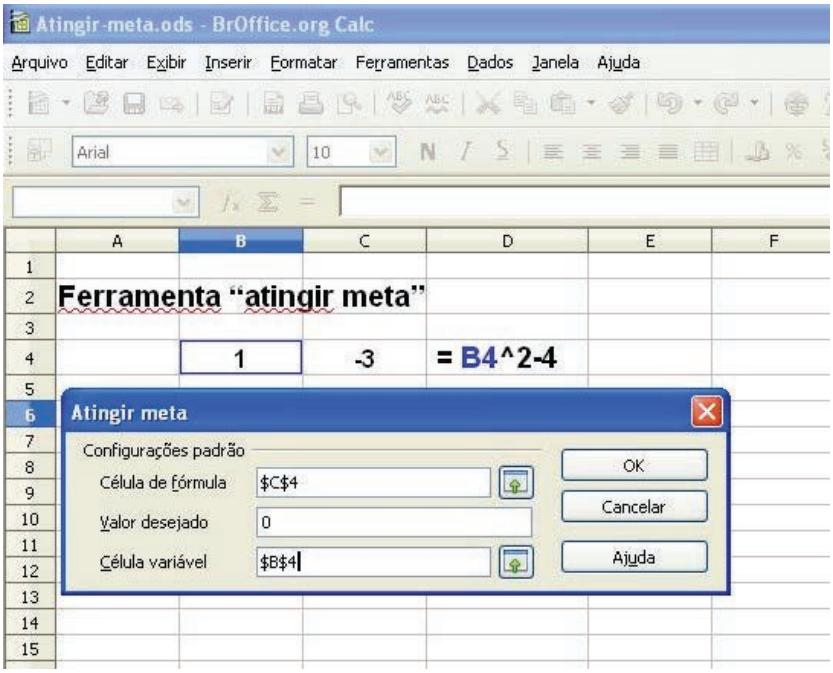


Figura 4.17 Caixa de diálogo da ferramenta "Atingir meta".

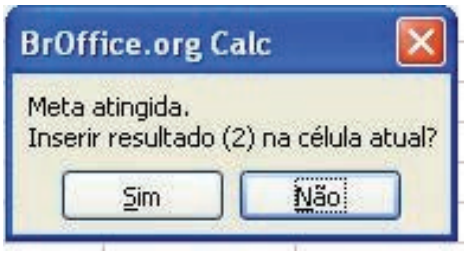


Figura 4.18 Caixa de diálogo final da ferramenta "Atingir Meta".

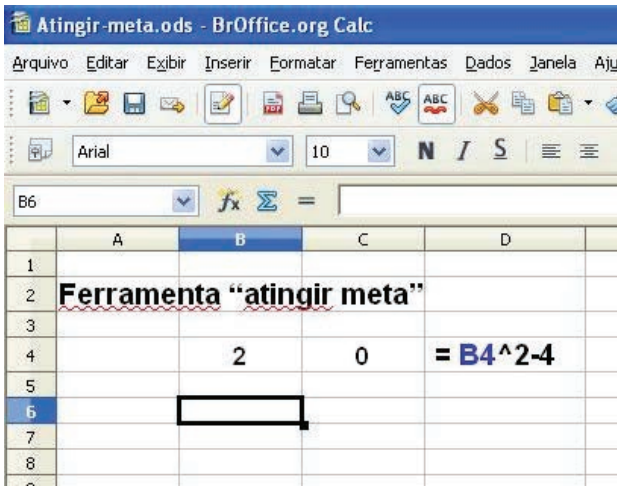


Figura 4.19 Resultado final do cálculo da raiz de f(x) com a ferramenta "Atingir Meta".

E como fazer para obter o valor da outra raiz (–2.0) ?

É simples. Digite na célula B4 um valor numérico próximo desta raiz, por exemplo –3.0. Siga os passos:

Vá até o menu **Ferramentas** e escolha a opção **"Atingir Meta".** Na caixa de diálogo preencha os campos conforme a Figura 4.17 e clique em OK. O caixa de diálogo surgirá mostrando o valor da outra raiz. Clicando em OK este valor é inserido na célula B4.

Esta é apenas uma aplicação da ferramenta "Atingir Metas". Ela é uma ferramenta bastante útil para funções de uma variável.

Com um pouco de criatividade, pode-se ampliar sua aplicação.

4.10 Resolvendo problemas

Nesta unidade iremos mostrar como resolver, com auxílio da planilha eletrônica, os problemas apresentados na unidade 2. Os algoritmos desenvolvidos na unidade 2 serão também implementados no BrOffice.org Calc.

Primeiro exemplo: Cálculo da área de um triângulo (area) a partir do comprimento dos seus três lados (a, b, c) empregando a fórmula de Herão (ou Herón).

1. Ao iniciar o aplicativo BrOffice.org Calc uma planilha em branco é apresentada. Digite na célula B2 o texto: "Cálculo da área (area) de um triângulo empregando a fórmula de Herão (ou Heron)" e na célula B3: "a partir do comprimento dos três lados do triângulo (a, b, c)".
2. Nas células B5, B6, B7, B9 e B11 digite, respectivamente, "a", "b", "c", "s" e "area".
3. Formate todas estas células com a opção texto e, se desejar, escolha uma cor para o fundo da célula (no exemplo será cinza). Você já aprendeu a fazer isto!
4. Utilizaremos agora a seguinte convenção: células com fundo amarelo representam células que o usuário pode alterar valores; células com fundo verde representam células que mostram resultados de um cálculo (seus valores são alterados em função dos valores digitados nas células amarelas).
5. Altere a cor das células C5, C6 e C7 para amarelo e digite os valores "3,0", "4,0" e "5,0", respectivamente (os valores dos lados do triângulo).
6. Altere a cor das células C9 e C11 para verde e digite as seguintes equações:

7. Célula C9 (cálculo do semi-perímetro): “=(C5+C6+C7)/2” <enter> (↵)
8. Célula C11 (cálculo da área): “=RAIZ(C9*(C9-C5)*(C9-C6)*(C9-C7))” <enter> (↵)
9. Finalizamos a implementação do algoritmo de Herão. Experimente alterar os valores dos lados do triângulo (a, b, c). Veja o que acontece. A Figura 4.20 apresenta esta planilha.

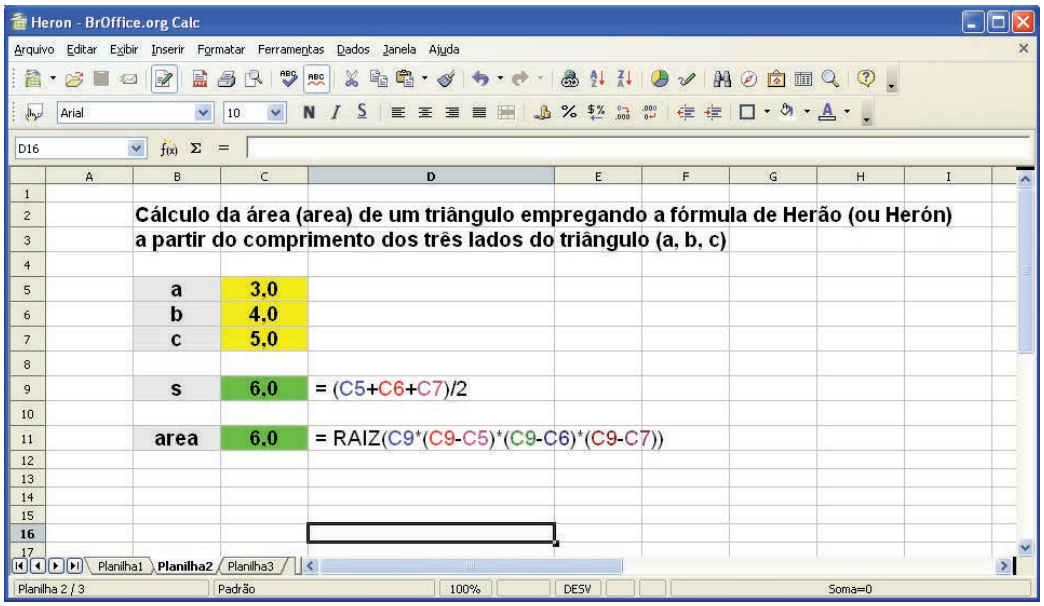


Figura 4.20 Planilha para o cálculo da área de um triângulo (area) a partir do comprimento dos seus três lados (a, b, c) empregando a fórmula de Herão (ou Herón) no programa BrOffice.org Calc.

Um ponto importante a observar é a dinâmica da planilha. Alterando qualquer um dos valores dos lados do triângulo o valor do semi-perímetro (s) e da área (area) são imediatamente recalculados.

Segundo exemplo: Cálculo da média final (MF) de um aluno em uma disciplina em função do valor das notas obtidas nas avaliações P1 e P2 e de um trabalho (T). Relembrando: Nesta disciplina os alunos realizaram duas provas (P1 e P2) e um trabalho (T). O critério de avaliação estabelecido pelo professor para o cálculo da média final (MF) foi:

Se P1 ≥ 5,0 e P2 ≥ 5,0 a média final é calculada por:

$$MF = \frac{(2 \cdot P1 + 2 \cdot P2 + 2 \cdot T)}{6}$$

Caso contrário:

$$MF = \frac{(2 \cdot P1 + 2 \cdot P2 + T)}{5}$$

Vejamos como implementar este problema na planilha eletrônica.

1. Na planilha do BrOffice.org Calc, digite nas células B2 e B3, respectivamente, os textos “Implementando algoritmo para o cálculo da média final (MF)” e “em função das notas obtidas nas avaliações P1 e P2 e do trabalho T”.
2. Nas células B5, B6, B7 e B9 digite, respectivamente, “P1”, “P2”, “T” e “MF”.
3. Formate todas estas células com a opção texto e, se desejar, escolha uma cor para o fundo da célula (no exemplo será cinza).
4. Altere a cor das células C5, C6 e C7 para amarelo e digite os valores “4,5”, “7,0” e “8,5”, respectivamente (os valores das notas P1, P2 e T).
5. Altere a cor da célula C9 para verde e digite as seguintes equações:
6. Célula C9 (cálculo da média final):

```
“SE(E(C6>=5,0;C7>=5,0);(2*C6+2*C7+2*C8)/6;(2*C6+2*C7+C8)/5)”
<enter>
```

O resultado é apresentado na Figura 4.21.

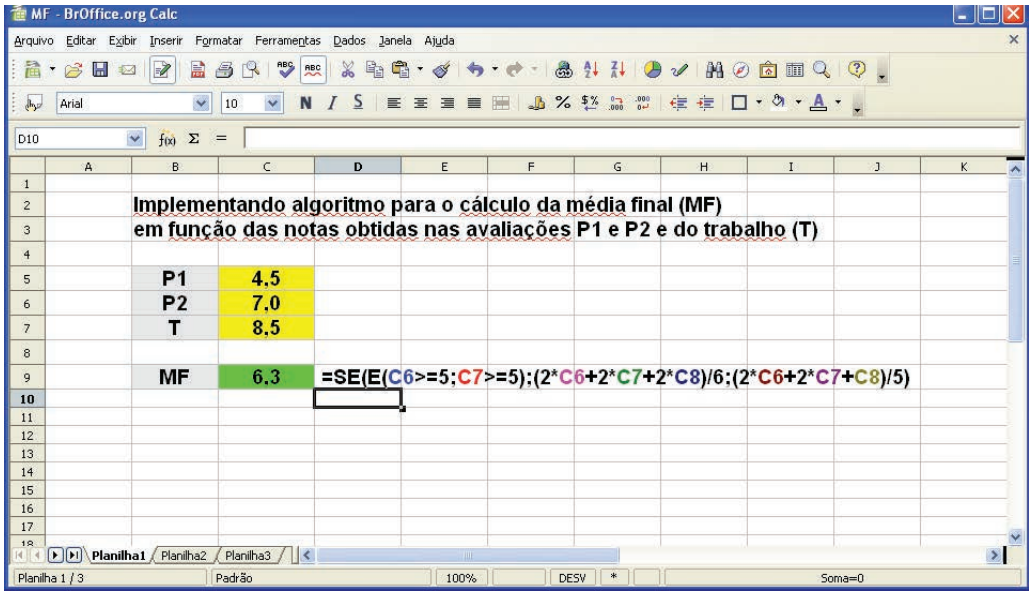


Figura 4.21 Planilha para o cálculo da média final (MF) de um aluno em uma disciplina em função do valor das notas obtidas nas provas P1 e P2 e da nota do trabalho (T) no programa BrOffice.org Calc.

Neste exemplo utilizamos os operadores condicionais **SE** e **E** para implementar a estrutura de seleção do algoritmo.

Terceiro exemplo: Resolução do algoritmo para calcular as raízes (x1 e x2) de uma equação do segundo grau () a partir dos valores das constantes a, b e c fornecidas pelo usuário. O algoritmo deve utilizar a fórmula de Báskara.

1. Na planilha do BrOffice.org Calc, digite na célula B2 o texto “Cálculos das raízes reais de uma equação do segundo grau: a.x^2+b.x+c=0”
2. Nas células B4, B5, B6, B8, B10, B12 e B13 digite, respectivamente, “a”, “b”, “c”, “Delta”, “x1” e “x2”.
3. Formate todas estas células com a opção texto e, se desejar, escolha uma cor para o fundo da célula (no exemplo será cinza).
4. Altere a cor das células C4, C5 e C6 para amarelo e digite os valores “1,0”, “–3,0” e “1,0”, respectivamente (os coeficientes da equação a, b e c).
5. Altere a cor da célula C8, C10, C12 e C13 para verde e digite as seguintes equações:

   Célula C8 (Verificando se a equação é do segundo grau):

```
“=SE(C4=0;”Não é equação do 2o grau”;”Equação é do segundo grau”)”  <enter> (↵)
```

   Célula C10 (Calculando o determinante):

```
“=SE(C4=0;...;C5^2-4*C4*C6)” <enter> (↵)
```

   Célula C12 (Calculando a raiz x1):

```
“=SE(E(C4<>0;C10>=0);(-C5+RAIZ(C10))/2/C4;‘’)” <enter> (↵)
```

   Célula C13 (Calculando a raiz x2):

```
“=SE(E(C4<>0;C10>=0);(C5+RAIZ(C10))/2/C4;...)” <enter> (↵)
```

O resultado é apresentado na Figura 4.22.

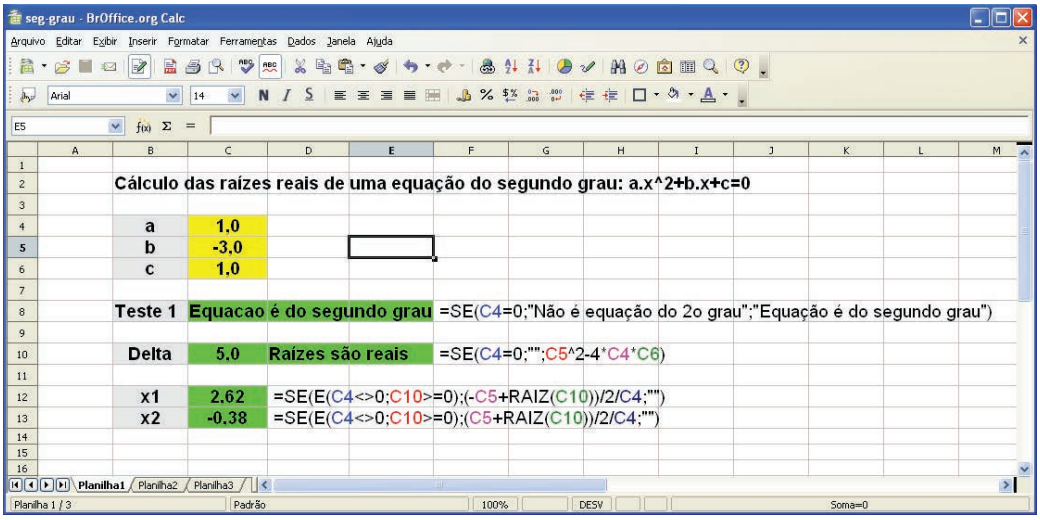


Figura 4.22 Planilha para o cálculo das raízes (x1 e x2) de uma equação do segundo grau no programa BrOffice.org Calc.

Experimente alterar os valores dos coeficientes da equação (a, b, e c) e veja os resultados que são exibidos na planilha.

4.11 Considerações finais

Nesta unidade buscou-se apresentar um conhecimento básico para possibilitar o uso da planilha eletrônica BrOffice.org Calc na solução de problemas normalmente encontrados em disciplinas que contemplem o uso do computador como ferramenta de suporte.

Vimos que foi possível resolver os problemas apresentados na unidade 2 com auxílio da planilha eletrônica.

4.12 Referências bibliográficas

Manual do BrOffice.org Calc.

Moura, L. F. *Excel para Engenharia*: formas simples para resolver problemas complexos. São Carlos: EdUFSCar, 2007, v. 1.

SOBRE O AUTOR

Antonio José Gonçalves Cruz

Engenheiro Químico (1993), Mestre (1996) e Doutor (2000) em Engenharia Química pela Universidade Federal de São Carlos. Professor das disciplinas “Balanços de Massa e Energia”, “Análise e Simulação de Processos Químicos”, “Desenvolvimento de Processos Químicos 1 e 2” no curso de graduação em Engenharia Química da UFSCar. Na pós-graduação ministra as disciplinas de “Fundamentos de Processos Químicos” e “Análise Numérica em Engenharia Química”.

Este livro foi impresso em junho de 2013 pelo Departamento de Produção Gráfica – UFSCar.