PROGRAMANDO A $HP-50g$

(Uma Introdução ao Fascinante Universo da Programação)

$$a_{ij} = (-1)^{\left\lfloor \frac{i-1}{2^{j-1}} \right\rfloor}$$

$$a_{ij} = (-1)^{\mu_{2^{j-1}}^{i-1}}$$

$$a_{ij} = (-1)^{\binom{i-1}{2^{j-1}}}$$

```
≪ → L r
   ≪ { } 'LC' STO L SIZE 'N' STO
      1 2 N ^ FOR I
            1 N FOR J
                  IF '(-1)^FLOOR((I-1)/2^(J-1))==1'
                  THEN J
                  END
               NEXT DEPTH DUP 'C' STO
                  IF 'C==r'
                  THEN ROW→ 'V' STO 1 r
                        FOR K L 'V(K)' EVAL
                              GET
                        NEXT r
                     →LIST 1 →LIST LC + 'LC' STO
                  ELSE CLEAR
                  END
            NEXT LC
   ≫
≫
```

Gentil Lopes da Silva

www.dmat.ufrr.br/gentil

Programando a $HP - 50g$

Engº Gentil Lopes da Silva

24 de maio de 2009

Prefácio

A motivação para escrever o presente trabalho foi dupla. Em 1996 eu me encontrava na UFSC quando fui solicitado, pelos alunos da física e engenharia, a ministrar um curso de programação da HP; quando, na ocasião, tive a oportunidade de escrever uma *apostila* "Programando a $HP-48$" para conduzir o curso. Em 2009 encontro-me na UFRR ministrando a disciplina Cálculo Numérico, na qual decidi adotar a $HP-50g$. Este trabalho foi escrito tomando por base "a velha apostila" e o `Guia do Usuário-HP`.

No que diz respeito à eficiência da maioria dos alunos em utilizar os recursos disponíveis na calculadora − pelo que tenho observado − é a mesma de um proprietário de uma possante `ferrari` que, no entanto, se desloca (movimenta) em um `monociclo`.

Creio que o conteúdo deste livro é o essencial para quem se inicia em programação (aliás este curso se dirige a principiantes). É importante salientar que iremos aprender a programar não somente a $HP-50g$, como também, em decorrência, outras linguagens de computação, como por exemplo, PASCAL, MATLAB, C^{++}, etc[†]. Estou salientando este fato devido ao pré-conceito generalizado existente a respeito da programação de calculadoras comparado à de microcomputadores − pré-conceito este que só se explica em função da ignorância de quem desconhece que essencialmente não há diferença entre programar uma calculadora e um microcomputador (nas linguagens supramencionadas, por exemplo). Inclusive devido à portabilidade (pode-se carregá-la no bolso, ou na bolsa, como queiram) é que eu penso que os pais deveriam dar a seus filhos, juntamente com o vídeo-game, uma calculadora programável, pois desta forma estão lhes abrindo as portas de uma alternativa profissional (e ainda por que a programação desenvolve a inteligência, logicamente falando).

No prefácio de um dos meus livros ([4]) em 1999 escrevi:

"*Há algum tempo tenho o sonho de ver estudantes, de todos os níveis, com calculadoras programáveis em sala de aula e programando os problemas em tempo real.*"

Este livro se constitui num auxílio para que este sonho se torne realidade.

Informamos que o presente livro serve igualmente aos que possuem as calculadoras anteriores (por exemplo, $HP-48$ e $HP-49$), já que a programação (sintaxe) é a mesma. O que muda é apenas o modo de se acessar os comandos (instruções), mas isto de fato não se constitui em empecilho uma vez que todos os comandos podem ser digitados diretamente, caracter a caracter.

Para finalizar enfatizo que seremos gratos a críticas e sugestões no sentido de melhoria de futuras versões do presente trabalho.

www.dmat.ufrr.br/gentil
gentil.silva@gmail.com

Este opúsculo foi escrito com o processador de texto LaTeX 2_ε.

Por mais este trabalho concluido louvo ao bom Deus.

Gentil Lopes da Silva

`Boa Vista-RR`, 24 de `maio de` 2009.

[†]Já que a lógica (estrutura) de programação é a mesma.

Sumário

TESOUROS NO CÉU

– Não ajunteis tesouros na terra, onde a traça e a ferrugem tudo consomem, e onde os ladrões minam e roubam. Mas ajuntai tesouros no céu, onde nem a traça nem a ferrugem consomem, e onde os ladrões não minam nem roubam. (Mt. 6 : 19 – 20)

– Exegese: Daqui podemos inferir que tudo o que se deteriora com o tempo, ou que é passível de furto, não pode ser tesouro no céu. Ao contrário, o que é atemporal e à prova de furtos, tem chances de ser um tesouro no céu.

Como por exemplo, cada uma das pérolas a seguir:

$$a_{nm} = (-1)^{\left\lfloor \frac{n-1}{2^{m-1}} \right\rfloor} \qquad a_{nm} = (-1)^{\binom{n-1}{2^{m-1}}} \qquad a_{nm} = (-1)^{\mu_{2^{m-1}}^{n-1}}$$

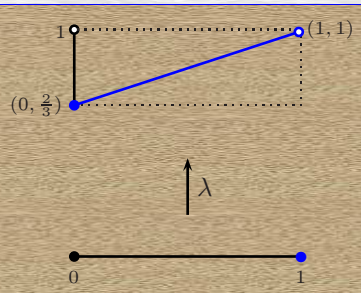
$$1^m + 2^m + 3^m + \cdots + n^m = \sum_{j=0}^{m} \binom{n}{j+1} a_{(m-j)}$$

$$a_{(m-j)} = \sum_{k=0}^{j} (-1)^k \binom{j}{k} (1-k+j)^m$$

$$(x,\, y,\, z) \equiv (X,\, Y) = (\, y - x \cdot \operatorname{sen}\theta,\; z - x \cdot \cos\theta)$$

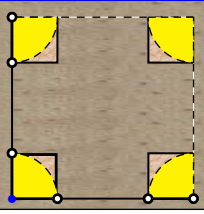
$$a_{nm} = \prod_{j=0}^{m} a_{1(m-j)}^{\binom{n-1}{j}} \qquad a_{nm} = \sum_{j=0}^{m} \binom{n-1}{j} a_{1(m-j)} \qquad \begin{cases} i = \lfloor \frac{n-1}{N} \rfloor + 1 \\ j = n - N\lfloor \frac{n-1}{N} \rfloor \end{cases}$$

$$0,999\ldots = \frac{9}{10} + \frac{9}{100} + \frac{9}{1000} + \cdots = 0 \qquad n = f(i,\, j) = N(i-1) + j$$



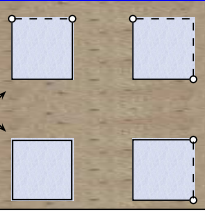
$$\Rightarrow \qquad \lambda(t) = \begin{cases} \frac{2}{3} + \frac{1}{3}t, & 0 \le t < 1; \\ 0, & t = 1. \end{cases}$$

$\frac{n}{2^{j-1}} \in \mathbb{N} \iff \binom{n-1}{2^{j-1}}$ e $\binom{n}{2^{j-1}}$ têm paridades distintas.



Topologia Quântica

O Milagre!:conexo por caminhos



Gentil/fev–2009

Capítulo 1

Introdução à Programação

"Mas a atividade mais feliz e mais bem-aventurada é aquela que produz. Ler é delicioso, mas ler é uma atividade passiva, enquanto criar coisas dignas de serem lidas é ainda mais precioso." (`Ludwig Feuerbach`)

1.1 Introdução

Programar a calculadora significa introduzir em sua memória (RAM− Random Access Memory − memória de acesso aleatório) uma série de instruções e comandos para que ela os execute seqüêncialmente, cumprindo alguma tarefa específica. Por exemlo, resolver uma equação quadrática, multiplicar ou dividir polinômios, imprimir textos, elaborar um gráfico, tocar música, etc.

Para tanto é necessário que as instruções e os comandos sejam digitados no padrão sintático da linguagem da calculadora e dispostos seqüêncialmente na ordem em que devem ser executados. A fim de que a execução seja perfeita e apresente os resultados objetivados com precisão, não basta atender estes requisitos. É preciso que o programa não contenha erros de lógica, cuja detecção não é feita pela calculadora, que está preparada para apontar somente erros de sintaxe.

Os recursos de programação postos à nossa disposição pela calculadora $HP-50g$ são excepcionalmente valiosos e variados e a melhor forma de conhecê-los, entender sua finalidade e alcance e fixá-los em nossa memória é através da prática.

Embora mencionada como uma calculadora por causa de seu formato compacto similar aos dispositivos de cálculo manuais típicos, a $HP-50g$ deve ser vista como um computador programável/gráfico.

Programação estruturada

A $HP-50g$ encoraja o uso de programação estruturada. Cada programa possui apenas um ponto de entrada: seu início. Também possui apenas um ponto de saída: seu final. Não existem rótulos dentro de um programa para desvios e, portanto, não existem comandos **GOTO** (ir para).

Já neste momento vamos apresentar duas importantes tabelas que serão extensamente utilizadas em nosso curso. Sugerimos que todos os exemplos sejam executados para serem devidamente assimilados.

1.1.1 Funções para números reais (← MTH REAL)

A tabela a seguir mostra os comandos, sua descrição e exemplos.

Comando/Descrição	Exemplos	
	Antes	Depois
ABS. Valor absoluto	1 : −12	1 : 12
CEIL. Menor inteiro maior ou igual ao argumento.	1 : −3.5	1 : −3
	1 : 3.5	1 : 4
FLOOR. Maior inteiro menor ou igual ao argumento.	1 : 6.9	1 : 6
	1 : −6.9	1 : −7
FP. Parte fracionária do argumento.	1 : 5.234	1 : .234
	1 : −5.234	1 : −.234
IP. Parte inteira do argumento.	1 : 5.234	1 : 5
	1 : −5.234	1 : −5
MANT. Mantissa do argumento.	1 : $1.23E12$	1 : 1.23
MAX. O maior valor entre dois números.	2 : 5	
	1 : −6	1 : 5
MIN. O menor valor entre dois números.	2 : 5	
	1 : −6	1 : −6
MOD. Resto da divisão entre dois números.	2 : 6	
	1 : 4	1 : 2
RND. Arredonda o número de acordo com o valor do argumento: $n = 0$ até 11.	2 : 1.2345678	
	1 : 5	1 : 1.23457
SIGN. Retorna +1 para argumentos positivos, −1 para argumentos negativos e 0 para argumentos nulos.	1 : −2.7	1 : −1
TRNC. Trunca o número de acordo com o valor do argumento: $n = 0$ até 11.	2 : 1.2345678	
	1 : 5	1 : 1.23456

• Nota: Caso você não consiga acessar os comandos acima através das teclas ← MTH REAL, então o sinalizador 117 da sua calculadora deve ser ativado.

– Para ativar o sinalizador 117 (dizemos: escolher os menus SOFT) pressione: MODE H FLAGS e com o auxílio da tecla ▽ "desça" até 117 (marcar: ✓).

• Nota: Após digitar ← MTH REAL pressione, se necessário, a tecla NXT L para mover para a próxima página (do menu); a combinação ← PREV move para a página anterior (do menu).

Para mais informações sobre `sinalizadores` ver subseção 1.1.3 na pág. 8.

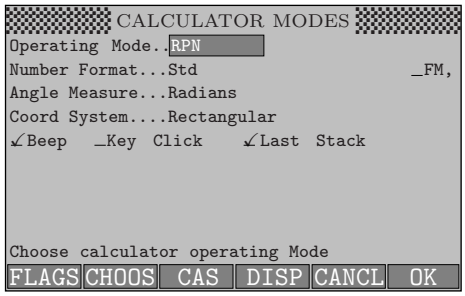
1.1.2 Menu de comandos da pilha (↰ PRG STACK)

A tabela a seguir mostra os comandos, sua descrição e exemplos.

Comando/Descrição	Exemplos			
	Antes		Depois	
DUP. duplica o objeto do nível 1.	2 :		2 :	5
	1 :	5	1 :	5
SWAP. permuta os objetos dos níveis 1 e 2.	2 :	9	2 :	10
	1 :	10	1 :	9
DROP. deleta o objeto do níveil 1.	2 :	7	2 :	
	1 :	8	1 :	7
OVER. retorna uma cópia do objeto do níve 2 para o nível 1.	4 :		4 :	
	3 :		3 :	'AB'
	2 :	'AB'	2 :	234
	1 :	234	1 :	'AB'
ROT. rotaciona os três primeiros objetos da pilha.	3 :	12	3 :	34
	2 :	34	2 :	56
	1 :	56	1 :	12
ROLL. move o objeto do nível n+1 para o nível 1. (n está no nível 1).	4 :	444		
	3 :	333	3 :	333
	2 :	222	2 :	222
	1 :	3	1 :	444
PICK. retorna uma cópia do objeto do nível n+1 para o nível 1 (n está no nível 1)	4 :	123	4 :	123
	3 :	456	3 :	456
	2 :	789	2 :	789
	1 :	3	1 :	123
DEPTH. retorna o número de objetos na pilha.	3 :		3 :	16
	2 :	16	2 :	−5
	1 :	−5	1 :	2
DROP2. remove os objetos dos níveis 1 e 2.	3 :	12	3 :	
	2 :	10	2 :	
	1 :	8	1 :	12
DROPN. remove os primeiros n+1 objetos da pilha (n está no nível 1).	4 :	123	4 :	
	3 :	456	3 :	
	2 :	789	2 :	
	1 :	2	1 :	123
DUPN. duplica n objetos da pilha, começando no nível 2 (n está no nível 1).	5 :		5 :	333
	4 :	333	4 :	222
	3 :	222	3 :	111
	2 :	111	2 :	222
	1 :	2	1 :	111

1.1.3 Sinalizadores (flags)

Um sinalizador é um valor Booleano que pode ser ativado ou desativado (verdadeiro ou falso), que especifica uma dada configuração da calculadora ou uma opção em um programa. Os sinalizadores na calculadora são identificados por números. Existem 256 sinalizadores, numerados de -128 a 128. Sinalizadores positivos são chamados de sinalizadores de usuários e estão disponíveis para programação pelo usuário. Os sinalizadores representados pelos números negativos são chamados de sinalizadores de sistema e afetam a forma que a calculadora opera. Para ver a configuração atual do sinalizador de sistema pressione a tecla MODE H :



e depois a tecla virtual FLAGS . Você obterá um visor denominado

SYSTEM FLAGS listando os números dos sinalizadores e as configurações.

Um sinalizador pode ser considerado ativado se você ver a marca de seleção (✓) na frente do número do sinalizador. Caso contrário, o sinalizador não está ativado. Para alterar o status de um sinalizador de sistema pressione a tecla virtual ✓CHK enquanto o sinalizador que você deseja alterar é ressaltado. Você pode usar as teclas de setas para cima e para baixo (⊙ ⊙) para se deslocar ao redor da lista de sinalizadores do sistema.

Exemplo de configuração de sinalizador: Por exemplo, o sinalizador 02 quando ativo (✓) disponibiliza uma constante na sua forma numérica e não como símbolo. Por exemplo, ative este sinalizador e, após, coloque π na pilha. Depois desative-o e coloque, novamente, π na pilha.

Como mais um exemplo: Ative a trava automática do Modo Alfa para somente uma digitação de ALPHA (ao invés de duas). Para fazer isto, ative o sinalizador 60.

Ainda: Se você quer `vírgula` (ao invés de `ponto`) em decimais tais como .333333333333 ative o sinalizador 51, para obter: ,333333333333.

* * *

– Uma observação oportuna é a de que assim como não existe uma única maneira de se resolver determinado problema na física ou na matemática, por exemplo, também não existe um único modo de fazer determinado *programa* no computador (ou na calculadora); provavelmente cada programador o faça de modo distinto. Um critério bastante geral para se avaliar um bom programa é a simplicidade do mesmo, embora nesta altura dos acontecimentos (isto é, para o iniciando na programação) o importante é que o programa seja feito e funcione de acordo com o esperado.

1.2 Programação de fórmulas (equações)

Para a $HP-50\,g$ um programa é um objeto delimitado pelos símbolos $\ll\ \gg$, isto é, todos os programas devem ser digitados entre estes símbolos*.

Variáveis

Variáveis são como arquivos em um disco rígido de computador. Uma variável pode armazenar um objeto (valores numéricos, expressões algébricas, listas, vetores, matrizes, programas, etc).

As variáveis são reconhecidas pelos seus nomes, que podem ser qualquer combinação de caracteres alfabéticos ou numéricos, iniciando com uma letra.

Aguns caracteres não alfabéticos, tais como a seta ($\rightarrow$) podem ser usados em um nome de variável, se combinados com um caractere alfabético. Assim, '$\rightarrow$ A' é um nome válido de variável, mas '$\rightarrow$ ' não é. Exemplos válidos de nomes de variáveis são: 'A', 'B', 'a', 'b', 'α', 'β', 'A1', 'AB12', '$\rightarrow$ A12', 'Vel', 'Z0', 'Z1', etc.

Uma variável não pode ter o mesmo nome de uma função da calculadora.

Você não pode ter uma variável SIN por exemplo, já que existe um comando SIN na calculadora. Os nomes reservados das variáveis da calculadora são os seguintes: ALRMDAT, CST, EQ, EXPR, IERR, IOPAR, MAXR, MINR, PICT, PPAR, PRTPAR, VPAR, ZPAR, der_, e, i, n1,n2, ..., s1, s2, ..., DAT, PAR, π, ∞.

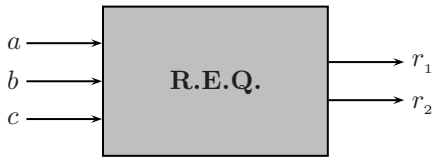
Nota: Letras maiúsculas e minúsculas não são equivalentes.

Estrutura de variável local

O comando† $\rightarrow$ define nomes de variáveis locais (isto é, variáveis que somente são válidas dentro do programa em que foram definidas) – ao contrário das variáveis globais, que são definidas pelo comando **STO**.

Antes de se iniciar a programação de determinado problema é importante que se tenha bem claro em mente quais são os dados de entrada (no programa) e quais são os dados de saída; por exemplo:

1º) Resolver a equação quadrática $ax^2 + bx + c = 0$. Temos:



Onde:

– Dados de entrada: a, b e c.

– Dados de saída: r_1 e r_2 (são as raízes).

– **R.E.Q.**: Variável que irá armazenar o programa (e que será referenciada sempre que o programa for executado).

Obs: o nome **R.E.Q.** é apenas um exemplo, o nome poderia ser um outro, a seu critério.

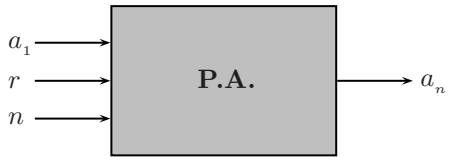
*O qual encontra-se acima da tecla [+], em vermelho. É acessado assim [↱] [≪ ≫]

†Este comando (de atribuição) encontra-se acima da tecla [0], em vermelho.

2º) Calcular o $n-$ésimo termo de uma progressão aritmética:

$$a_n = a_1 + (n-1)r$$

Temos:



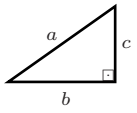
Onde:

– Dados de entrada: O primeiro termo a_1; a razão da *P.A.* r e a posição n do termo que queremos encontrar.

– Dados de saída: O $n-$ésimo termo a_n.

– **P.A.**: Variável que irá armazenar o programa (e que será referenciada sempre que o programa for executado).

3º) Cálculo da hipotenusa de um triângulo retângulo: $a = \sqrt{b^2 + c^2}$. Temos:



Onde:

– Dados de entrada: Os catetos b e c.

– Dado de saída: A hipotenusa a.

– **HIP.**: Variável que irá armazenar o programa.

Nos três programas anteriores devemos fornecer os dados de entrada e o programa calcula e fornece os dados de saída.

Uma estrutura de variável local possui uma das seguintes organizações dentro de um programa:

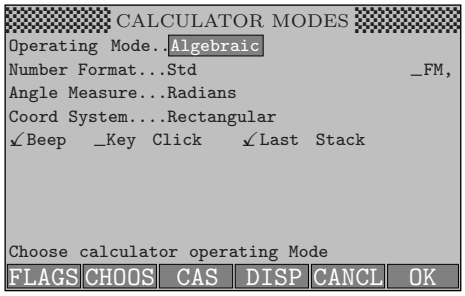
$$\ll \rightarrow \ \text{nome}_1 \quad \text{nome}_2 \ldots \text{nome}_n \ \text{'objeto algébrico'} \gg \tag{1.1}$$

Ou,

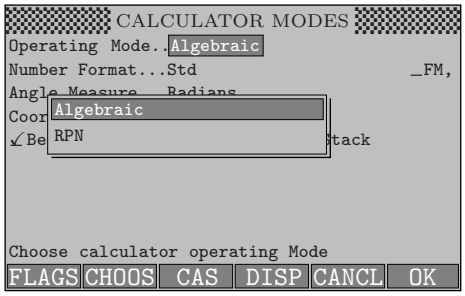
$$\ll \rightarrow \ \text{nome}_1 \quad \text{nome}_2 \ldots \text{nome}_n \quad \ll \text{programa} \gg \gg \tag{1.2}$$

O comando (de atribuição) $\rightarrow$ remove n objetos da pilha e os armazena nas variáveis locais, cujos nomes são listados em seguida.

Importante: A calculadora opera (trabalha) em dois modos: `pilha` (RPN) e `algébrico` (ALG). Neste livro optaremos pelo modo pilha. Caso sua calculadora esteja no modo algébrico – isto estará indicado na primeira linha do visor à direita, com o distinvo ALG – para colocá-la no modo pilha pressione a tecla MODE H para ir ao visor,



Agora pressione a tecla virtual CHOOS (escolher) para ir à seguinte tela:



Agora basta selecionar o modo RPN: ▽ e pressionar OK OK (isto é, 2×).

Exemplo 1: O seguinte programa toma dois números da pilha e retorna um resultado numérico (o valor absoluto da diferença entre ambos, isto é: $|a-b|$).

$$\ll \rightarrow \text{ a b } \text{ 'ABS(a-b)' } \gg \tag{1.3}$$

Este programa é da forma (1.1). Antes de programarmos o exemplo acima, observamos que você pode acessar os comandos, como ABS por exemplo, pelas teclas do menu* ou digitá-los (letra a letra).

Pois bem, vamos retomar o programa anterior. observe que em,

$$\text{'ABS(a-b)'}$$

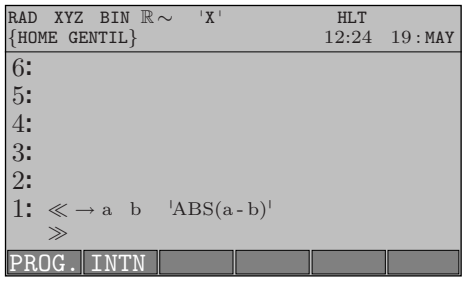
temos aspas simples (chamadas doravante de plics). Estas aspas simples (plics) encontram-se na tecla ' O .

Digite o programa dado em (1.3). Observe que entre a seta e a primeira variável (no caso a) existe um espaço (mas não se preocupe pois a calculadora o coloca automaticamente; isto é, após digitar a seta – veja nota de rodapé † na pág. 9.)

*Ver Tabela na pág. 6.

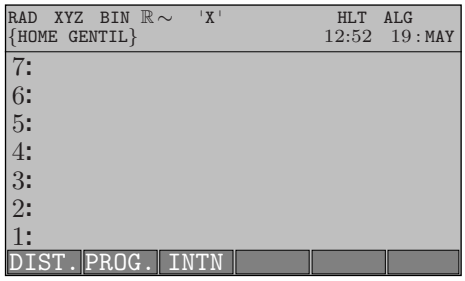
1.2.1 Para armazenar ou nomear um programa

1º) Entre com o programa, já digitado, na pilha. Isto é, após digitá-lo pressione ENTER . O programa irá para o nível 1 da pilha, assim:



2º) Agora precisamos armazenar este programa em uma variável (nomeá-lo), para isto pressione plics, ' O , e escolha um nome para o programa. Por exemplo, 'DIST.' . Pressione ENTER e, em seguida, pressione a tecla STO▶ K .

Você acabou de armazenar (STO) o programa na variável que você escolheu (isto é, no nome dado ao programa). Observe que o programa (e o seu nome) desaparecem da pilha mas o nome (do programa) aparece no menu de variáveis, assim:



Se na sua calculadora o programa não aparece no primeiro menu, como acima, basta você pressionar a VAR J .

Nota: O procedimento para armazenar qualquer programa é sempre o mesmo, isto é, siga os passos 1º) e 2º) acima.

1.2.2 Para executar um programa

Coloque os dados na pilha (no caso do programa DIST., os números que serão armazenados nas variáveis a e b, por exemplo 3 e 5), pressione VAR J (caso necessário) e depois a tecla do menu que contém o nome dado ao programa.

Como funciona o programa anterior: Se o leitor colocar na pilha, por exemplo, os seguintes dados de entrada: 3 e 5; ao executar o programa o comando de atribuição (→) armazenará estes valores nas variáveis a e b, respectivamente. Em seguida a expressão algébrica será avaliada (calculada) para aqueles valores que, a estas alturas, já estarão armazenadas nas variáveis que constam na expressão (entre plics).

Modos de programação

Em nosso curso utilizaremos dois modos de programação, os dados nas referências (1.1) e (1.2) (pág. 10). O primeiro é mais prático para a programação de fórmulas matemáticas, enquanto o segundo para programas mais elaborados, como teremos oportunidade de exemplificar.

Exemplo 2: Por exemplo, o programa a seguir,

```
≪ → b c ' √(b^2+c^2)' ≫
```

(ver pág. 10) calcula a hipotenusa de um triângulo retângulo.

Exemplo 3: Este mesmo programa no modo (1.2) fica assim:

```
≪ → b c
    ≪ ' √(b^2+c^2)' EVAL
    ≫
≫
```

O comando EVAL (acessado na tecla EVAL N) avalia uma expressão algébrica (isto é, uma expressão entre plics) que o precede. Sugerimos ao leitor digitar este programa e, em seguida, seguir os passos descritos nas subseções 1.2.1 e 1.2.2.

1.3 Para visualizar ou editar um programa

Para visualizar − ou editar um programa − coloque seu nome entre plics e em seguida na pilha. Pressionando (se necessário) as teclas:

VAR J ' O ENTER

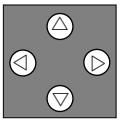
Nota: Caso o programa que você deseja editar não se encontre em um dos menus exibidos na tela pressione a tecla NXT L até encontrá-lo.

Com o nome do programa na pilha pressione, em seguida, as duas teclas:

← ▽

Pronto! Agora é só editar (ou visualizar).

Nota: Utilizando os quatro botões prateados, abaixo,



você posiciona o cursor em qualquer posição do programa.

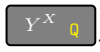
Exemplo 4: Como mais um exemplo de programação de fórmulas vamos resolver uma equação quadrática $ax^2 + bx + c = 0$, segundo a conhecida fórmula,

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2 \cdot a}$$

O programa fica assim (ver pág. 9):

```
≪ → a b c
   ≪  '(−b + √(b^2 − 4 ∗ a ∗ c))/(2 ∗ a)'  EVAL
      '(−b − √(b^2 − 4 ∗ a ∗ c))/(2 ∗ a)'  EVAL
   ≫
≫
```

Notas: − O aspirante a programador deve ficar atento para a inserção correta dos parêntesis, sob pena de redundar em erro de lógica, o qual a calculadora não detecta (ela não adivinha o que você gostaria de calcular).

− O "chapeuzinho" em b^2 (isto é, b^2) é acessado na tecla  .

− Se, ao digitar um programa, "a tela (visor) se tornar pequena(o)" use as teclas  para iniciar em uma nova linha.

Sugerimos ao leitor carregar e executar este programa.

1.4 Executando um programa passo-a-passo (DBUG)

A calculadora nos oferece um recurso muito importante, principalmente para quem se inicia em programação, que é a possibilidade de se executar um programa passo-a-passo. Este recurso é poderoso e deve, amiúde, ser utilizado, por duas razões:

1ª) Permite que se compreenda o funcionamento (e a lógica) de um programa;

2ª) Facilita a correção de eventuais erros.

Este recurso nós o chamamos de DBUG. É fácil entender como um programa trabalha (funciona) se você executá-lo passo-a-passo, observando o efeito de cada etapa (comando). Isto facilita "depurar" seus próprios programas ou entender programas escritos por outros.

Utilizando o DBUG

1. Coloque todos os dados requeridos pelo programa na pilha, e nos níveis apropriados*;

2. Coloque o nome do programa no nível 1 da pilha. Isto é, abra plics e pressione o nome do programa o qual é acessado − como já dissemos − pressionando-se a tecla VAR J e depois a tecla do menu que contém o nome dado ao programa.

Pressione a tecla ENTER;

3. Pressione a seguinte sequência de teclas:

← PRG NXT L NXT L RUN DBUG

Neste momento o nome do programa desaparece da pilha. Agora basta ir pressionando, sucessivamente, o menu SST .

• Para continuar com a execução normal pressione o comando CONT. Este comando é acessado pressionando-se as teclas: ← ON.

Sugerimos ao leitor executar passo-a-passo o programa para o cálculo das raízes de uma equação quadrática.

Para avançar passo-a-passo no meio do programa

1. Insira o comando HALT na linha de programa a partir da qual você deseja iniciar o avanço manual (DBUG);

2. Execute o programa normalmente. Ele pára quando o comando HALT é executado, o anúncio HALT é exibido (na tela).

3. Tome qualquer ação:

• Para ver o próximo passo do programa exibido na área de mensagens e então executá-lo, pressione o menu SST .

• Para ver, mas não executar o próximo (ou os próximos dois passos), pressione NXT L.

• Para continuar com a execução normal, pressione CONT (← ON.).

Quando você desejar que o programa seja executado normalmente (novamente) remova o comando HALT do programa.

1.5 Exibindo variáveis de entrada e saída

A instrução PROMPT faz com que a calculadora nos mostre (exiba) a(s) variável(eis) de entrada num programa. Esta instrução é acessada com a seguinte sequência de teclas:

← PRG NXT L IN NXT L PROMP

A instrução →TAG rotula um objeto com um nome ou texto descritivo.

Esta instrução é acessada com a seguinte sequência de teclas:

*No caso do programa da hipotenusa, a ordem em que os catetos são fornecidos não altera o resultado final. Já no caso do programa para o cálculo da raízes de uma equação quadrática, a ordem − na qual os dados entram na pilha − é decisivo.

[↰] PRG TYPE →TAG

Nota importante: Lembramos que todos os comandos (instruções) podem ser digitados "na mão". Se o leitor decidir digitar a instrução →TAG alertamos que não deve haver espaço entre a seta e TAG. Esta setinha encontra-se acima da tecla 0, em vermelho.

Exemplo 5: Para exemplificar o uso destas instruções vamos acrescentá-las ao programa das raízes de uma equação quadrática, assim:

```
≪ ''Entre com a, b e c'' PROMPT → a b c
   ≪ '(−b + √(b^2 − 4 ∗ a ∗ c))/(2 ∗ a)' EVAL
        ''R1'' →TAG
        '(−b − √(b^2 − 4 ∗ a ∗ c))/(2 ∗ a)' EVAL
        ''R2'' →TAG
   ≫
≫
```

Nota: Acesse as aspas duplas assim: [↱] [''''].

Ao encontrar PROMPT o processamento do programa é interrompido e a calculadora fica esperando os dados de entrada; coloque-os na pilha e, após, pressione CONT (continue*).

. Excluindo uma variável

Para excluir (deletar) uma variável proceda assim: coloque, entre plics, o nome da variável, pressione ENTER para colocá-lo na pilha, após pressione: [TOOL I] PURGE. Para excluir várias variáveis ao mesmo tempo coloque-as (isto é, seus nomes) entre chaves ({ ... ... }) e proceda como antes.

*Lembramos que este comando é acessado com [↰] ON.

Capítulo 2

Listas e Matrizes

> "O gênio, porque sabe encontrar relações novas entre as coisas, revela-nos novas harmonias e nos aproximam do pensamento de Deus." ($E = m \cdot c^2$) (`Pietro Ubaldi`)

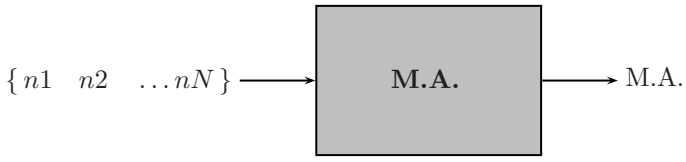
Com o fito de aumentar ainda mais nosso poder (potência) de programação é que incluimos este capítulo com dois importantes recursos para programação: listas e matrizes.

2.1 Listas

Uma lista é constituida de objetos (números, letras, gráfico, etc.) entre chaves e separados por um espaço (tecla SPC). Uma lista é o que, em matemática, comumente conhecemos por *conjunto*. Exemplo de lista:

$$\{\,1 \quad 5 \quad \mathrm{a} \quad \{\,\mathrm{b} \quad \mathrm{c}\,\}\,\}$$

Este é um recurso muito importante para manipulação de objetos. Para fornecer um exemplo de utilização de listas vamos elaborar, oportunamente (pág. 28), um programa para calcular a *média aritmética* de N números, fornecidos em uma lista, tipo:



- Criando listas

Para introduzir uma lista a partir do teclado:

1. Use ↰ + para indicar o início e o fim de uma lista.
2. Entre com os elementos da lista, separados por SPC.

Para montar uma série de elementos numa lista:

1. Entre com os elementos na pilha.

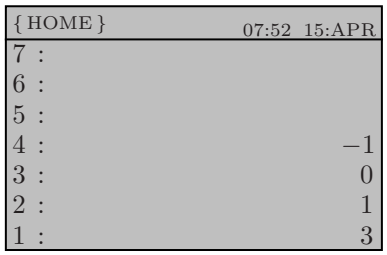
2. Digite o número de elementos no nível 1 da pilha.
3. Use ↰ PRG LIST →LIST para converter os elementos da pilha em uma lista.

Exemplo: Crie uma lista com os elementos −1, 0 e 1, usando →LIST.

Passo 1: Entre com os elementos e o número de elementos na pilha.

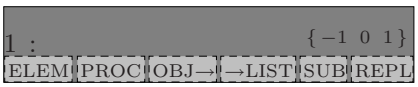
1 +/- W SPC 0 SPC 1 SPC 3 ENTER

Após este procedimento a pilha deve apresentar-se assim:



Passo 2: Converta a pilha para uma lista.

↰ PRG LIST →LIST



Importante: Para realizar o processo inverso do anterior; digo, para `desmontar` uma lista (ou ainda, para retirarmos os elementos de uma lista) basta digitarmos

↰ PRG LIST OBJ→

Nota: A lista deve está no nível 1 (tal como na tela anterior). Observe, ademais, que este comando nos devolve, no nível 1, o número de elementos na lista.

• Manipulação de Listas

As funções a seguir oferecem maneiras de manipular os elementos de uma lista. Sugerimos ao leitor fazer uma simulação em cada ítem para que os respectivos comandos fiquem perfeitamente compreendidos.

1) ↰ MTH LIST SORT coloca os elementos de uma lista em ordem ascendente. A lista deve estar no nível 1.

2) ↰ MTH LIST REVLI reverte a ordem dos elementos de uma lista. A lista deve estar no nível 1.

3) + adiciona itens ao início ou no final de uma lista ou concatena duas listas. Para acrescentar um elemento ao início da lista, entre com o ítem, a lista e pressione +. Para adicionar um ítem no final da lista, entre com a lista, depois com o ítem e pressione +.

4) ↰ PRG LIST ELEM NXT L HEAD substitui a lista do nível 1 pelo seu primeiro ítem isolado.

5) ↰ PRG LIST ELEM NXT L TAIL substitui a lista do nível 1 por todos

seus elementos, com exceção do primeiro.

6) ← PRG LIST ELEM GET substitui a lista do nível 2 e um número de posição (índice) do nível pelo elemento da lista naquela posição indicada.

7) ← PRG LIST ELEM GETI é similar a GET, porém ela também incrementa o número de posições (índice). O novo índice é colocado no nível 2. A lista original estará no nível 3.

8) ← PRG LIST ELEM PUT toma um objeto do nível 1 e substitui um objeto existente dentro da lista. Você deve fornecer a posição do objeto no nível 2 e a lista (ou matriz) no nível 3. A lista resultante estará no nível 1.

9) ← PRG LIST ELEM PUTI é similar a PUT, porém ela também incrementa o índice. O novo índice é colocado no nível 1 e a nova lista no nível 2.

10) ← PRG LIST ELEM SIZE substitui a lista do nível 1 pelo número de elementos que ela possui.

11) ← PRG LIST ELEM POS substitui a lista do nível 2 e um elemento daquela lista (nível 1) por um índice contendo a primeira ocorrência daquele elemento na lista. Se o elemento não for encontrado, 0 será retornado.

2.2 Matrizes

A $HP-50\,g$ possui grandes recursos para entrada e manipulação de matrizes. Muitas das operações descritas aqui também se aplicam a vetores (que são matrizes com apenas uma linha ou uma coluna). O nome *matrizes* também inclui objetos vetoriais.

- Criando e Montando Matrizes

Você pode introduzir uma matriz de duas maneiras:

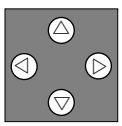
– **O Editor de Matrizes (Matrix writer)**. Um método mais intuitivo para entrar, visualizar e editar elementos de matrizes.

– **Linha de comando**. O método básico de entrada de objetos.

Para entrar com uma matriz usando o Editor de Matrizes:

1. Pressione ← MTRW para acessar a tela e menu do Editor de Matrizes.
2. Para cada elemento na primeira linha execute uma das opcções abaixo:
   . Digite um número real ou complexo e pressione ENTER.
   . Calcule o elemento usando a linha de comando e pressione ENTER. Para calcular o elemento, digite os argumentos separados com SPC e pressione a função desejada para o cálculo.
3. Pressione ▽ para marcar o final da primeira linha (a qual especifica o número de colunas para a matriz).

Nota: Manipulando os quatro botões prateados, abaixo,



você pode colocar o cursor em qualquer posição da matriz.

4. Para cada elemento do restante da matriz, digite (ou compute) seu valor e pressione ENTER. Ou, se você desejar, introduza números em mais de uma célula de uma só vez, digitando-os na linha de comando e usando SPC para separá-los. Pressione ENTER ao final para introduzi-los.

5. Após a entrada de todos os elementos na matriz, pressione ENTER para colocá-la na pilha operacional.

Para entrar com uma matriz utilizando a linha de comando:

1. Pressione ↰ [ ] duas vezes para entrar com os delimitadores para a matriz e para a primeira linha.

2. Digite os elementos da primeira linha. Pressione SPC para separar os elementos.

3. Pressione ▷ para avançar o cursor para além do "fecha colchetes".

4. Pressione ↰ [ ] para preencher uma nova linha (se necessário).

5. Repita os passos 3. e 4., se necessário. Ao término pressione ENTER para colocar a matriz na pilha.

Para montar uma matriz por linhas a partir de uma série de vetores:

1. Introduza cada vetor na pilha, na mesma ordem que você deseja que apareçam na matriz.

Entre com o vetor da linha 1 antes, depois a linha 2, e assim por diante.

2. Entre com o número de linhas para a matriz desejada.

3. Pressione ↰ MTH MATRX COL COL→ para montar os vetores numa matriz.

Para criar uma matriz preenchida com uma constante dada:

1. Entre com um dos seguintes ítens na pilha:

. Uma lista contendo as dimensões da matriz desejada na forma: {linha coluna}.

. Qualquer matriz, cujos elementos você não se importa que sejam alterados.

2. Introduza a constante com a qual você deseja preencher a matriz.

3. Pressione ↰ MTH MATRX MAKE CON. Isto retorna uma matriz com as dimensões especificadas, preenchida com a constante fornecida.

Para criar uma matriz identidade:

1. Entre com um dos seguintes ítens na pilha:

. Um número real que representa o número de linhas e colunas de uma matriz identidade (quadrada).

. Qualquer matriz quadrada, cujo conteúdo você não se importa que sejam alterados.

2. Pressione ↰ MTH MATRX MAKE IDN. Esta operaçã retorna uma matriz identidade com as dimensões dadas.

Desmontando matrizes

A $HP-50g$ monta e desmonta os elementos de uma matriz de duas dimensões, seguindo a ordem de preenchimento de linhas (da esquerda para a direita, e de cima para baixo). Começando com o elemento corrente (frequentemente o elemento da linha 1 e coluna 1), a ordem de preenchimento de linhas assume que o "próximo" elemento será o próximo naquela linha. Caso não existam mais

elementos na linha, então o próximo será o primeiro da linha seguinte. Esta convenção de preenchimento é semelhante a um processador de textos, onde uma linha é preenchida e, assim que cheia, o preenchimento prossegue no início da próxima.

Para desmembrar uma matriz em seus elementos

1. Entre com a matriz na pilha.

2. Pressione ↰ PRG TYPE OBJ→. A matriz será desmontada na ordem de preenchimento por linhas, colocando cada elemento em um nível separado. O nível 1 contém uma lista com as dimensões originais da matriz.

Para montar uma matriz a partir de uma sequência de elementos

1. Entre com com os elementos na pilha na ordem de preenchimento por linhas.

2. Entre com uma lista contendo as dimensões da matriz desejada na forma: { linhas colunas }.

3. Pressione ↰ PRG TYPE →ARRY. Para montar a matriz.

Para desmembrar uma matriz em vetores-linha

1. Entre com a matriz na pilha.

2. Pressione ↰ MTH MATRX ROW →ROW. A matriz é desmembrada em vetores (primeira linha até a última). O nível 1 da pilha contém um número real representando o número de linhas da matriz original.

Para desmembrar uma matriz em vetores-coluna

1. Entre com a matriz na pilha.

2. Pressione ↰ MTH MATRX COL →COL. A matriz é desmembrada em vetores (primeira coluna até a última). O nível 1 da pilha contém um número real representando o número de colunas da matriz original.

Inserindo linhas e colunas

Para inserir uma ou mais linhas novas numa matriz:

1. Entre com a matriz alvo (aquela que você deseja modificar) na pilha operacional.

2. Entre com o vetor, matriz ou elemento (quando o objeto alvo é um vetor) que você deseja inserir. Uma matriz inserida deve possuir o mesmo número de linhas e colunas que a matriz alvo.

3. Entre com o número da linha sobre a qual você eseja inserir a nova matriz. Todos os elementos localizados naquela linha da matriz alvo serão movidos para baixo, acomodando a inserção. Números de linhas começam de 1 e não de 0.

4. Pressione ↰ MTH MATRX ROW ROW+ para inserir as novas linhas.

Para inserir uma ou mais colunas novas numa matriz:

1. Entre com a matriz alvo (aquela que você deseja modificar) na pilha operacional.

2. Entre com o vetor, matriz ou elemento (quando o objeto alvo é um vetor) que você deseja inserir. Uma matriz inserida deve possuir o mesmo número de linhas e colunas que a matriz alvo.

3. Entre com o número da coluna sobre a qual você eseja inserir a nova matriz. Todos os elementos localizados naquela coluna da matriz alvo serão movidos

para a direita, acomodando a inserção. Números de colunas começam de 1 e não de 0.

4. Pressione ← MTH MATRX COL COL+ para inserir as novas colunas.

Extraindo Linhas e Colunas

Para extrair uma linha específica de uma matriz:

1. Entre com a matriz na pilha operacional.

2. Entre com o número da linha (ou a posição do elemento, caso a matriz alvo seja um vetor) que você deseja extrair.

4. Pressione ← MTH MATRX ROW ROW-. O vetor-linha (ou elemento) extraído é colocado no nível 1 e a matriz com a linha removida é colocada no nível 2. **Para extrair uma coluna específica de uma matriz:**

1. Entre com a matriz na pilha operacional.

2. Entre com o número da coluna (ou a posição do elemento, caso a matriz alvo seja um vetor) que você deseja extrair.

4. Pressione ← MTH MATRX COL COL- . O vetor-coluna (ou elemento) extraído é colocado no nível 1 e a matriz com a coluna removida é colocada no nível 2.

Invertendo Linhas e Colunas

Para inverter a posição de duas linhas numa matriz:

1. Entre com a matriz na pilha. Se a matriz for um vetor, ele será considerado um vetor-coluna.

2. Entre com os dois números das linhas que serão trocadas.

4. Pressione ← MTH MATRX ROW NXT L RSWP. A matriz modificada será deixada no nível 1.

Para inverter a posição de duas colunas numa matriz:

1. Entre com a matriz na pilha. Se a matriz for um vetor, ele será considerado um vetor-linha.

2. Entre com os dois números das colunas que serão trocadas.

4. Pressione ← MTH MATRX COL CSWP. A matriz modificada será deixada no nível 1.

Extraindo e Substituindo Elementos em Matrizes

Para extrair um elemento de uma posição específica:

1. Entre com a matriz na pilha.

2. Entre com um dos seguintes objetos:

   . Uma lista contendo a linha e a coluna do elemento que você deseja extrair: {linha, coluna}.

   . A posição do elemento que você deseja extrair (contando da esquerda para a direita, e de cima para baixo).

3. Pressione ← MTH MATRX MAKE NXT L GET para extrair o elemento específico.

Para substituir um elemento de uma posição específica:

1. Entre com a matriz na pilha.
2. Entre com um dos seguintes objetos:

   . Uma lista contendo a linha e a coluna do elemento que você deseja extrair: {linha coluna}.

   . A posição do elemento que você deseja extrair (contando da esquerda para a direita, e de cima para baixo).
3. Entre com o elemento substituto .
4. Pressione ← MTH MATRX MAKE NXT PUT para substituir o elemento específico na localização escolhida.

Obtendo as dimensões de uma matriz: Para obter as dimensões (número de linhas e colunas) de uma matriz inicialmente coloque-a na pilha, após pressione ← MTH MATRX MAKE SIZE

Transformando Matrizes

Para transpor uma matriz:

1. Entre com a matriz na pilha.
2. Pressione ← MTH MATRX MAKE TRN. Para transpor a matriz.

Para inverter uma matriz quadrada:

1. Entre com a matriz quadrada na pilha.
2. Pressione 1/X Y

Para alterar as dimensões de uma matriz

1. Entre com a matriz na pilha.
2. Entre com uma lista contendo as novas dimensões da matriz na forma: {linha coluna}.
3. Pressione ← MTH MATRX MAKE RDM para redimensioná-la. Elementos da matriz original são recolocados na matriz redimensionada, da esquerda para a direita, da 1ª linha atá a útima. Caso haja menos elementos na nova matriz, os elementos em excesso da original serão descartados. Caso haja mais elementos na nova matriz, as posições faltantes serão preenchidas com zeros (ou $(0, 0)$ se a matriz for complexa).

Usando Matrizes e Seus Elementos em Expressões Algébricas

Você pode realizar cálculos com elementos de matrizes usando sintaxe algébrica. A matriz deve ser representada por um nome na expressão simbólica ou equação.

Para utilizar um elemento da matriz numa expressão algébrica

1. Certifique-se de que a matriz está armazenada numa variável.
2. Crie a expressão algébrica e, no ponto onde o elemento de matriz será usado, digite o nome da matriz e pressione ← ( ).
3. Entre com os índices para o elemento:

   . Para um vetor, digite um índice (número da posição do elemento).

   . Para uma matriz, entre os dois índices separados → , (número da linha e da coluna para o elemento).

Vejamos um exemplo deste importante recurso. Coloque na pilha a seguinte matriz,

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Para isto veja, por exemplo: **Para entrar com uma matriz utilizando a linha de comando**: na pág. 20.

Após, armazene-a em uma variável. Por exemplo, para armazenar esta matriz na variável JOSE, proceda assim: digite JOSE e pressione ENTER para colocar este nome no nível 1 da pilha, em seguida pressione STO▶ K para armazenar a matriz neste nome. Este é o nome da matriz (com o qual ela será referenciada).

Para se certificar disto basta pressionar VAR J JOSE, a matriz deverá retornar à pilha.

Digite a seguinte expressão: '$\mathrm{JOSE}(1,1)+\mathrm{JOSE}(1,2)$'

Agora pressione ENTER para colocar este objeto algébrico na pilha. Em seguida mande avaliá-lo pressionando EVAL N. O resultado deverá ser 3, precisamente a soma dos dois primeiros elementos da primeira linha da matriz.

Vamos "sofisticar" um pouco mais nosso exemplo. O seguinte objeto algébrico*

$$'\textstyle\sum(k=1,3,\mathrm{JOSE}(1,k))'$$

o qual é equivalenta a,

$$\sum_{k=1}^{3} \mathrm{JOSE}(1,k) = \mathrm{JOSE}(1,1) + \mathrm{JOSE}(1,2) + \mathrm{JOSE}(1,3)$$

quando avaliado (EVAL N) nos fornece o resultado 6, que é a soma dos elementos da primeira linha da matriz JOSE.

Agora vamos "abusar" da boa vontade da HP. O seguinte objeto algébrico

$$'\textstyle\sum(J=1,3,\sum(I=1,2,\mathrm{JOSE}(I,J)))'$$

o qual é equivalente a,

$$\begin{aligned}\sum_{J=1}^{3}\sum_{I=1}^{2} \mathrm{JOSE}(I,J) = {} & \mathrm{JOSE}(1,1) + \mathrm{JOSE}(2,1) \\ & + \mathrm{JOSE}(1,2) + \mathrm{JOSE}(2,2) \\ & + \mathrm{JOSE}(1,3) + \mathrm{JOSE}(2,3)\end{aligned}$$

quando avaliado nos fornece o resultado 21, que é a soma − por colunas − de todos os elementos da matriz JOSE.

*Este somatório, entre plics, pode ser acessado assim: ' O → SIN S.

Capítulo 3

Estruturas de Programação

"... que o meu pensamento quis aproxima-se dos problemas do espírito pela via de uma diversa experimentação de caráter abstrato, especulativo, resultante das conclusões de processos lógicos da mais moderna físico-matemática."(`Pietro Ubaldi`)

Introdução

Uma `estrutura de programação` permite a um programa tomar uma decisão sobre como ele deve ser executado, dependendo das condições dadas ou dos valores de argumentos em particular. Um uso cuidadoso e inteligente destas estruturas tornam possível a criação de programas com extraordinária flexibilidade.

Diriamos que a programação propriamente dita começa aqui com estruturas de programação, pois o que fizemos anteriormente foi a programação de fórmulas apenas. Estas estruturas que iremos estudar são comuns a várias linguagens de programação, como por exemplo, PASCAL, FORTRAN, C^{++}, MATLAB, etc. Quero dizer: você entendendo-as neste contexto, também estará apto a executá-las em qualquer outra linguagem em que estas se façam presentes; daí a importância de entendê-las nesta aqui, isto é, na HP.

As estruturas de programação

As estruturas que iremos estudar são as seguintes:

- Estruturas cíclicas : $\begin{cases} FOR - NEXT \\ FOR - STEP \\ WHILE - REPEAT - END \end{cases}$

- Estruturas condicionais : $\begin{cases} IF - THEN - END \\ IF - THEN - ELSE - END \end{cases}$

3.1 Estruturas cíclicas

3.1.1 FOR - NEXT

Entre com esta estrutura num programa pressionando a seguinte sequência de teclas:

A sintaxe desta estrutura é assim:

```
≪ ... início fim FOR contador
                    cláusula-cíclica
                NEXT ...
≫
```

Esta estrutura executa uma porção do programa por um número definido de vezes usando o conteúdo de uma variável local como contador, a qual pode ser usada dentro do `loop` para cálculos ou outros propósitos. A cláusula cíclica é executada pelo ao menos uma vez.

FOR extrai dois números da pilha: início e fim e cria uma variável local (contador) para controle de incremento do ciclo. Depois disto a cláusula-cíclica é executada, sendo que a variável contador pode aparecer dentro da cláusula. NEXT incrementa o contador em 1 (uma unidade) e testa se ele é menor ou igual ao valor fim. Se for este o caso, a cláusula-cíclica é repetida com o novo valor do contador. Caso contrário, a execução prossegue após NEXT. Quando o ciclo (`loop`) terminar o contador é apagado.

Vamos dar alguns exemplos de programas que utilizam esta estrutura, comentar alguns aspectos destes exemplos e colocar alguns problemas que devem ser programados com esta estrutura.

Exemplo 6: (Somatório). Vamos construir um programa para calcular a soma dos N primeiros números Naturais. Isto é, estamos querendo o valor de:

$$\sum_{i=1}^{N} i = 1 + 2 + 3 + \cdots + N \tag{3.1}$$

Devemos fornecer ao programa o valor de N (até onde queremos que o mesmo some) e este deve nos devolver o valor da soma. O programa fica assim:

```
≪ → N
   ≪ 0 'S' STO
        1 N FOR I
                S I + 'S' STO
            NEXT S ''SOMA'' →TAG
   ≫
≫
```

Neste programa fornecemos o valor de N, que será armazenado na variável local N e a execução começa inicializando uma variável (global por causa de STO) com zero. A variável S irá armazenar (acumular) o valor da soma. O

contador (ou variável de controle) é o I, que irá variar de 1 até N, o que concorda com a variação de i no somatório em (3.1).

Para que o leitor entenda o funcionamento (lógica) do programa sugerimos que o mesmo seja executado com o auxílio do `DBUG`. (ver 1.4, pág. 15).

Existe uma fórmula para o somatório dado em (3.1), assim:

$$\sum_{i=1}^{N} i = 1 + 2 + 3 + \cdots + N = \frac{N \cdot (N+1)}{2}$$

a qual pode fácilmente ser programada, assim:

$$\ll \rightarrow \ N \quad {}'N * (N+1)/2' \ \gg$$

Exemplo 7: (Fatorial). O fatorial de um número natural n é definido assim:

$$n! = n(n-1)(n-2)\cdots 1$$

Por exemplo, $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$. O programa seguinte calcula o fatorial de um número natural N.

```
≪ → N
   ≪ 1 'F' STO
        1 N FOR I
              F I * 'F' STO
            NEXT F ''FAT.'' →TAG
   ≫
≫
```

Sugestão: execute-o no `DBUG`. *Nota*: Podemos inicializar um somatório sempre com 0 e um produtório sempre com 1.

Exemplo 8: (Soma dos termos de uma P.A.). No estudo das *progressões aritméticas* existem duas fórmulas dadas por,

$$a_n = a_1 + (n-1) \cdot r, \qquad S_n = n \cdot a_1 + \frac{n \cdot (n-1)}{2} \cdot r$$

para o cálculo do n-ésimo termo e para a soma dos n primeiros termos, respectivamente.

Embora a segunda fórmula possa ser programada diretamente (e mais fácilmente), faremos um programa, para o cálculo da soma dos termos de uma P.A., utilizando a primeira fórmula – como mais um exemplo de utilização do FOR-NEXT. O programa fica assim:

```
≪ → a1 r N
   ≪ 0 'S' STO
        1 N FOR n
                'a1 + (n − 1) ∗ r ' EVAL
                 S + 'S ' STO
               NEXT S ''SOMA'' →TAG
   ≫
≫
```

Observe que utilizamos − para o cálculo de a_n − o *modo algébrico* após o FOR, poderiamos ter utilizado o *modo pilha*.

Exemplo 9: (média aritmética). Como mais uma ilustração do FOR-NEXT vamos construir, utilizando o recurso de listas, um programa para o cálculo da *Média aritmética* de n números, fornecidos em uma lista. Veja diagrama de bloco à pág. 17. O programa é dado a seguir:

```
≪ → L
   ≪ L OBJ→ 'N' STO
        1 N 1 − FOR I
                   +
               NEXT N / ''M.A.'' →TAG
   ≫
≫
```

Sugestão: Para entender a lógica do programa execute-o no `DBUG`. Entrando com a seguinte lista, por exemplo,

$$\{\,1\quad 5\quad 8\quad 2\,\}$$

o programa deve retornar: M.A.: 4.

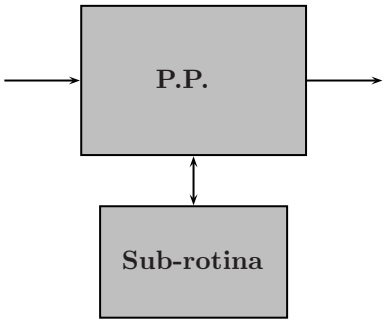
Observe que a variável de controle I não tem utilidade dentro do laço, atua como contador de ciclos apenas.

Exercício: Faça um programa para o cálculo da média aritmética utilizando *vetores* ao invés de lista.

. Sub-rotinas

Um recurso (ou técnica) bastante utilizado em programação é o que se chama de sub-rotina, que consiste na possibilidade de um programa ser acessado por um outro.

Uma situação em que é recomendável o uso de sub-rotina é quando temos um conjunto de instruções que é utilizado em diversas partes de um programa e para que não seja reescrito diversas vezes, é colocado em um programa à parte onde o primeiro programa (podemos chamá-lo de *principal*) acessa o segundo (sub-rotina). Podemos resumir a idéia no seguinte diagrama:



Os dados requeridos pelo programa sub-rotina são passados pelo programa principal.

Para avançar passo a passo quando o próximo passo é uma sub-rotina

▪ Para executar a sub-rotina em uma única etapa, pressione SST (ver DBUG, pág. 15).

▪ Para executar a sub-rotina passo a passo, pressione SST 1.

Exemplo 10: (Combinações). Como ilustração da utilização de sub-rotinas vamos fazer um programa para o cálculo do número de combinações de m objetos tomados n a n, segundo a fórmula:

$$\binom{m}{n} = \frac{m!}{n!\,(m-n)!} \tag{3.2}$$

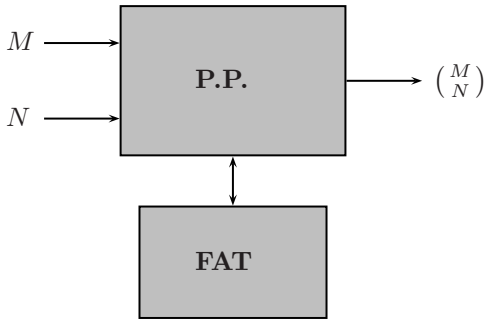
Observe que devemos calcular três fatoriais e para isto vamos usar o programa feito na pág. 27 como sub-rotina. Para os nossos propósitos aquele programa fica:

```
≪ → N
   ≪  1  'F' STO
        1  N  FOR  I
                F  I  *  'F'  STO
              NEXT  F
   ≫
≫
```

Vamos armazenar este programa com o nome de FAT. O programa principal (para o cálculo de (3.2)) pode ser:

```
≪ → M  N
   ≪      M  FAT
          N  FAT
     M  N  -  FAT
        *  /  ''COMB(M,N)''  →TAG
   ≫
≫
```

Observe que o programa principal chama a sub-rotina (armazenada em FAT) três vezes. O programa principal coloca o dado (inicialmente M) na pilha e chama a sub-rotina, esta calcula o fatorial e deixa o resultado na pilha. Execute-o no `DBUG`. Em blocos, temos:

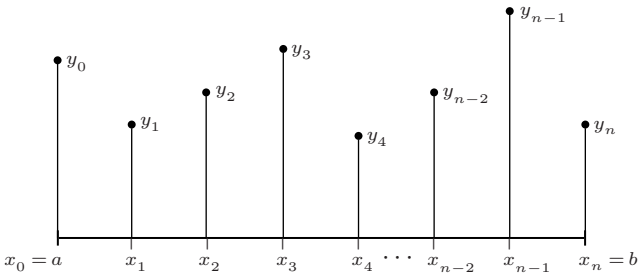


. Primeira Regra de Simpson

Exemplo 11: No cálculo numérico de *integrais*,

$$I = \int_a^b f(x)\,dx$$

efetuamos uma *partição* do intervalo $[\,a,\,b\,]$ em n subintervalos e "amostramos" a função f no pontos desta partição, assim:



Onde:

$$y_0 = f(x_0),\;\; y_1 = f(x_1),\;\; y_2 = f(x_2),\;\ldots,\; y_{n-1} = f(x_{n-1}),\;\; y_n = f(x_n)$$

A primeira regra (fórmula) de Simpson estabelece que,

$$I = \frac{h}{3}\,(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + \cdots + 2y_{n-2} + 4y_{n-1} + y_n) \qquad (3.3)$$

com erro dado por,

$$\varepsilon = -\frac{(b-a)^5}{180n^4}\,f^{(4)}(\xi),\;\; a \le \xi \le b.$$

Onde: $h = (b-a)/n$ e n é o número de subintervalos (nos quais dividimos o intervalo $[\,a,\,b\,]$), devendo ser um múltiplo de 2.

De momento, nos interessa a sequência (a_I) dos coeficientes que comparecem na equação (3.3), veja:

$$\begin{array}{lccccccccc} I: & 0 & 1 & 2 & 3 & 4 & \ldots & n-2 & n-1 & n \\ a_I: & 1 & 4 & 2 & 4 & 2 & \ldots & 2 & 4 & 1 \end{array} \tag{3.4}$$

Uma fórmula para o termo geral desta sequência é:

$$a_I = \begin{cases} 1, & \text{se } I = 0 \text{ ou } I = n; \\ 3-(-1)^I, & \text{se } 1 \leq I \leq n-1 \end{cases}$$

O seguinte programa recebe n e nos devolve um vetor contendo os termos da sequência (a_I):

```
≪ → n
   ≪ 1 1 n 1 -
        FOR I
             '3-(-1)^I' EVAL
        NEXT 1 n 1 + →ARRY
   ≫
≫
```

Nota: Armazene este programa na variável CPRS, pois será referenciado − com este nome − por um outro programa (isto é, será uma subrotina). Não esquecer que n deve ser um múltiplo de 2.

Para implementar (programar) a fórmula (3.3) necessitaremos de uma nova estrutura cíclica:

3.1.2 FOR - STEP

Entre com esta estrutura num programa pressionando a seguinte sequência de teclas:

A sintaxe desta estrutura é assim:

```
≪ ... início fim FOR contador
                     cláusula-cíclica
          incremento STEP ...
≫
```

Esta estrutura executa uma porção do programa por um número definido de vezes usando o conteúdo de uma variável local como contador, da mesma forma que em FOR-NEXT. A diferença é que você pode especificar um incremento diferente de 1.

FOR extrai dois números da pilha: início e fim e cria uma variável local (contador) para controle de incremento do ciclo. Depois disto a cláusula-cíclica

é executada, sendo que a variável contador pode aparecer dentro da cláusula. STEP aumenta o contador com o valor do incremento e testa se ele é menor ou igual ao valor fim. Se for este o caso, a cláusula-cíclica é repetida com o novo valor do contador (se o valor do incremento é negativo, o ciclo é repetido enquanto o contador for maior ou igual ao valor fim). Se o incremento de STEP for um objeto algébrico ou nome (variável), ele é automaticamente convertido para um número. Ele também pode ser positivo ou negativo. Se for positivo, o ciclo é executado novamente enquanto o contador for menor ou igual a fim. Caso contrário, a execução prossegue após STEP.

Implementação da 1ª regra de Simpson

Para programar a fórmula (3.3) vamos tomar como exemplo o cálculo da seguinte integral,

$$\int_a^b \frac{1}{1+x^2}\,dx \tag{3.5}$$

Antes vamos armazenar $f(x)$ numa variável, isto pode ser feito assim:

$$\ll \rightarrow\ x\quad \text{'}1/(1+x\,\hat{}\,2)\text{'}\ \gg$$

Armazene este programa na variável FUNC (função). Pois bem, a fórmula (3.3) está implementada pelo seguinte programa:

```
≪ → a b n
   ≪ '(b-a)/n' EVAL 'h' STO
                a b FOR x
                      x FUNC
                   h STEP n 1 + →ARRY
                   n CPRS DOT h 3 / *
   ≫
≫
```

A instrução x FUNC (dentro do FOR - STEP) coloca x na pilha e calcula o valor da função (previamente armazenada na variável FUNC).

Ao sair do laço existem $n+1$ valores de $y = f(x)$ na pilha, os quais são colocados em um vetor, assim:

$$[\,y_0 \quad y_1 \quad y_2 \quad y_3 \quad y_4 \quad \cdots \quad y_{n-2} \quad y_{n-1} \quad y_n\,]$$

Em seguida, o programa coloca n na pilha e chama a subrotina CPRS (programa dado na pág. 31) para gerar o seguinte vetor (coeficientes da fórmula (3.3)):

$$[\,1 \quad 4 \quad 2 \quad 4 \quad 2 \quad \ldots \quad 2 \quad 4 \quad 1\,]$$

Dados dois vetores $\mathbf{x} = [\,a_1 \quad a_2 \quad \ldots \quad a_n\,]$ e $\mathbf{y} = [\,b_1 \quad b_2 \quad \ldots \quad b_n\,]$ a instrução DOT realiza o *produto interno* entre estes vetores, assim:

$$\mathbf{x}\cdot\mathbf{y} = a_1\cdot b_1 + a_2\cdot b_2 + \cdots + a_n\cdot b_n$$

Portanto, após o DOT no nosso programa, obtemos a soma:

$$1\cdot y_0 + 4\cdot y_1 + 2\cdot y_2 + 4\cdot y_3 + 2\cdot y_4 + \cdots + 2\cdot y_{n-2} + 4\cdot y_{n-1} + 1\cdot y_n$$

Para usar o programa para uma outra função basta editar a variável FUNC.

Para $a = 0$, $b = 1$ e $n = 8$, o valor da integral (3.5) é: $0,785398$.

Exercícios

Nota: Os exercícios seguintes devem ser implementados com FOR-NEXT.

1) Faça um programa para sair com os N primeiros termos de uma P.A. (progressão aritmática), de dois modos distintos:
(a) em uma lista (b) em um vetor.

2) Faça um programa para calcular o produto dos N primeiros termos de uma P.A.

3) Faça um programa para calcular a soma dos N primeiros termos de uma P.G.

4) Faça um programa para calcular o seguinte somatório: $\sum_{i=1}^{n} i^2$.

5) Seja a sequência,

$$1, \quad 1, \quad -1, \quad -1, \quad 1, \quad 1, \quad -1, \quad -1, \quad 1, \quad 1, \quad -1, \quad -1, \quad \ldots$$

cuja fórmula do termo geral e do produto são,

$$a_n = (-1)^{\frac{(n-1)(n-2)}{2}} \quad \text{e} \quad P_n = (-1)^{\frac{n(n-1)(n-2)}{6}}$$

Faça um programa para sair com um vetor contendo os N primeiros termos desta sequência e mais ainda o produto destes N primeiros termos.

6) Faça um programa para calcular o somatório: $\sum_{i=1}^{n} i^m$; onde m e n são valores arbitrários que devem ser fornecidos ao programa.

Sugestão: Veja fórmula (**??**), pág. **??**.

7) Faça um programa para calcular para calcular a soma,

$$1 \cdot 2 + 2 \cdot 3 + 3 \cdot 4 + \cdots + n \cdot (n+1)$$

8) Faça um programa para calcular o *produto interno canônico* de dois vetors, assim:

$$[\, a_1 \quad a_2 \quad \ldots \quad a_n \,] \cdot [\, b_1 \quad b_2 \quad \ldots \quad b_n \,] = a_1 \cdot b_1 + a_2 \cdot b_2 + \cdots + a_n \cdot b_n$$

9) Faça um programa para sair com um vetor contendo os N primeiros termos de uma P.A., usando a fórmula de recorrência (definição) de uma P.A.:

$$\begin{cases} a_1 & = a \\ a_n & = a_{n-1} + r, \quad n \geq 2. \end{cases}$$

10) Faça um programa para sair com um vetor contendo os N primeiros termos de uma P.G., usando a fórmula de recorrência (definição) de uma P.G.:

$$\begin{cases} a_1 & = a \\ a_n & = a_{n-1} \cdot q, \quad n \geq 2. \end{cases}$$

11) Faça um programa para sair (em lista ou vetor) com os N primeiros termos das seguintes sequências :

(i) $a_1 = 4; \quad a_n = (-1)^n \cdot a_{n-1}$.

(ii) $a_1 = -2; \quad a_n = (a_{n-1})^2$.

(iii) (U.F.CE -81) Os termos da sucessão a_1, a_2, ..., a_n, estão relacionados pela fórmula $a_{n+1} = 1 + 2 \cdot a_n$, onde $n = 1, 2, 3, \ldots$. Se $a_1 = 0$, então a_6 é:

a) 25 b) 27 c) 29 d) 31

12) (PUC-SP - 81) Na sequência $(a_1, a_2, \ldots)$ tem-se:

$$a_1 = 1 \text{ e } a_{n+1} = \frac{2 + a_n^2}{2a_n}$$

Qual dos números abaixo está mais próximo de a_3?

a) 1 b) 2 c) $\sqrt{2}$ d) $\sqrt{3}$ e) $\sqrt{5}$

Nota: Lembramos que nestes exercícios queremos que você faça um pouco mais do que o exercício pede. Queremos que você faça um programa para listar os n primeiros termos das sequências dadas.

13) (U.F.BA. - 81) sejam as sequências

$$\begin{cases} a_1 = 4 \\ a_{n+1} = 1 + \frac{2}{a_n} \end{cases} \quad \text{e} \quad \begin{cases} b_1 = 5 \\ b_{n+1} = \frac{1}{1+b_n} \end{cases}$$

Se $\mathbf{P} = a_n \cdot b_n$, tem-se:

a) $P < 0$ b) $0 \leq P < 1$ c) $1 \leq P < 2$ d) $2 \leq P < 3$ e) $P \geq 3$

14) (U.F.PR. - 80) Seja f uma função tal que $f(1) = 2$ e $f(x + 1) = f(x) - 1$, para todo valor real de x. Então $f(100)$ é igual a:

a) -99 b) -97 c) 96 d) 98 e) 100

Nota: Aqui o leitor deverá fazer um programa para fornecer $f(N)$ onde $N > 1$ é um um número natural.

15) (PUC - SP - 85) Na sequência $(a_0, a_1, a_2, \ldots)$ onde $a_0 = 1$ e $a_{n+1} = a_n + n$, para todo $n \in \mathbb{N}$, a soma dos 7 primeiros termos é

a) 41 b) 42 c) 43 d) 63 e) 64

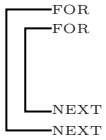
Nota: O leitor deverá sempre resolver os problemas de forma generalizada. Por exemplo, aqui faça um programa para calculara soma dos N primeiros termos da sequência .

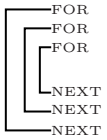
16) (PUC - SP - 70) Sendo $f\colon \mathbb{R} \to \mathbb{R}$ definida por $f(x) = 2x + 3$, então $f(1) + f(2) + f(3) + \cdots + f(25)$ é igual a:

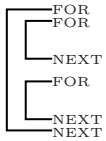
a) 725 b) 753 c) 653 d) 1375 e) 400

3.1.3 FOR - NEXT's concatenados

O que chamamos de concatenação de FOR - NEXT's é o mesmo que encaixe (ou aninhamento) de FOR - NEXT's que, dependendo do programa, pode tomar diversas configurações. Por exemplo, assim:

a)


b)


c)


A concatenação é, amiúdo, útil para se trabalhar com matrizes. Vejamos os seguintes exemplos:

Exemplo 12: (U.E.LONDRINA - 84) Dada a matriz $A = (\,a_{_{mn}}\,)_{_{2\times 2}}$ onde $a_{_{mn}} = 2^{n-m}$, a soma de todos os elementos que compõe a matriz A^2 é igual a:

a) $81/4$ b) 10 c) 9 d) $25/4$ e) -6

Motivados pelo desafio acima vamos fazer um programa para construir uma matriz (quadrada de ordem N) e que, em particular ($N = 2$) tenhamos a matriz do problema anterior.

Lembramos que o FOR - NEXT é acessado com a seguinte sequência de teclas:

O programa procurado fica assim:

```
≪ → N
   ≪ 1 N FOR m
        1 N FOR n
                 '2 ^(n − m)' EVAL
              NEXT N ROW→
           NEXT N ROW→
   ≫
≫
```

Observe que temos uma concatenação tipo **a)**. O primeiro FOR (ou ainda, o primeiro laço) fixa a linha e o segundo varia as colunas, de modo que a matriz vai sendo construida linha a linha e de cima para baixo. Para maiores detalhes execute-o no `DBUG`.

Para obter a matriz A^2, após a execução do programa, basta pressionar ENTER e multiplicar ×. Por exemplo, fornecendo 2 ao programa, este nos devolve a matriz:

$$\begin{bmatrix} 1 & 2 \\ 0.5 & 1 \end{bmatrix}$$

cujo quadrado é,

$$\begin{bmatrix} 2 & 4 \\ 1 & 2 \end{bmatrix}$$

Exemplo 13: (CESCEM - 71) Define-se a distância entre duas matrizes $\mathbf{A} = (a_{ij})$ e $\mathbf{B} = (b_{ij})$ quadradas e de mesma ordem n pela fórmula:

$$d(\mathbf{A}, \mathbf{B}) = \max |a_{ij} - b_{ij}|, \qquad i, j = 1, 2, 3, \ldots, n.$$

Assim a distância entre as matrizes $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ e $\begin{bmatrix} 5 & 7 \\ 6 & 8 \end{bmatrix}$ é:

a) -5 b) -3 c) 0 d) 3 e) 5

Nota: Na fórmula da distância max significa máximo (maior).

Antes de implementar o desafio acima em um programa, vejamos como fica esta distância para duas matrizes quadradas de ordem 2, isto é,

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \text{ e } \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

Primeiro, construimos a matriz diferença:

$$\mathbf{A} - \mathbf{B} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} - \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11} - b_{11} & a_{12} - b_{12} \\ a_{21} - b_{21} & a_{22} - b_{22} \end{bmatrix}$$

Agora temos que:

$$d(\mathbf{A}, \mathbf{B}) = \max \left\{ \begin{matrix} |a_{11} - b_{11}| & |a_{12} - b_{12}| \\ |a_{21} - b_{21}| & |a_{22} - b_{22}| \end{matrix} \right\}$$

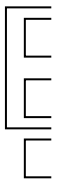
Para as matrizes dadas no desafio temos:

$$d(\mathbf{A}, \mathbf{B}) = \max \left\{ \begin{matrix} |1-5| & |2-7| \\ |3-6| & |4-8| \end{matrix} \right\} \Rightarrow d(\mathbf{A}, \mathbf{B}) = \max \left\{ \begin{matrix} 4 & 5 \\ 3 & 4 \end{matrix} \right\} = 5$$

O programa para calcular a distância entre duas matrizes quadradas de ordem n, fica assim:

```
≪ → A B
   ≪ A SIZE OBJ→ DROP DROP 'N' STO
        1 N FOR I
            1 N FOR J
                  'ABS(A(I,J)-B(I,J))' EVAL
               NEXT
          1 N 1 - FOR k
                  MAX
               NEXT
            NEXT
          1 N 1 - FOR k
                  MAX
               NEXT ''d(A,B)'' →TAG
   ≫
≫
```

Observe que temos uma concatenação de FOR's com a seguinte configuração:



– Onde: O laço maior fixa as linhas; o primeiro laço interno ao maior varia as colunas e calcula o valor absoluto da diferença dos elementos de mesma posição; o segundo laço interno encontra o maior valor de cada linha (de $\mathbf{A}-\mathbf{B}$), e o último laço encontra o maior valor dentre os maiores valores obtidos nas linhas (de $\mathbf{A}-\mathbf{B}$). Para maiores detalhes, execute-o no `DBUG`.

Nota: Veja a função MAX na tabela de funções para números reais (pág. 6).

Exercícios

Nota: Os exercícios seguintes devem ser implementados com FOR-NEXT.

17) (U. MACK. - 73) Sendo $\mathbf{A} = (a_{ij})$ uma matrix quadrada de ordem 2 e $a_{ij} = j - i^2$, o determinante da matriz $\mathbf{A}$ é:

a) 0 b) 1 c) 2 d) 3 e) 4

Nota: Novamente enfatizo que você deve implementar os exercícios da forma mais geral possível. Por exemplo, neste, entre com N (dimensão da matriz quadrada) e o programa deve sair com o determinante (DET) da matriz.

18) A seguinte fórmula (matriz) é de importância decisiva para encontrarmos combinações. Teremos oportunidade de usá-la futuramente (Apêndice):

$$a_{ij} = (-1)^{\left\lfloor \frac{i-1}{2^{j-1}} \right\rfloor}$$

Nota: $\lfloor x \rfloor$ é o maior inteiro que não supera x. Na HP você acessa esta função com o comando FLOOR (ver tabela dos reais, pág. 6).

Elabore um programa onde se entra com N e o mesmo nos devolva uma matriz de ordem $2^N \times N$.

Para dedução desta fórmula veja [4].

19) Uma outra distância entre matrizes $\mathbf{A} = (a_{ij})$ e $\mathbf{B} = (b_{ij})$ é dada pela igualdade

$$d(\mathbf{A}, \mathbf{B}) = \left[\sum_{i=1}^{m} \sum_{j=1}^{n} (a_{ij} - b_{ij})^2 \right]^{1/2}$$

onde $\mathbf{A}$ e $\mathbf{B}$ são matrizes retangulares de ordem $m \times n$. Esta é conhecida como distância euclidiana. Programe a fórmula acima.

3.1.4 WHILE - REPEAT - END

Esta é uma outra estrutura cíclica bastante utilizada. Entre com esta estrutura num programa pressionando a seguinte sequência de teclas:

A sintaxe desta estrutura é assim:

```
≪ ...
      WHILE   cláusula-de-teste
      REPEAT
            cláusula-cíclia
      END ...
≫
```

Esta estrutura executa um ciclo repetidamente, enquanto o resultado de um teste seja 1 (Verdadeiro). Ela pode ser interpretada como:

```
≪ ...
      ENQUANTO   isto ocorrer
      REPITA
            esta instrução
      FIM ...
≫
```

Como a cláusula-de-teste é executada antes da cláusula-cíclica, o ciclo jamais será executado se o teste já for falso desde o princípio. WHILE inicia a cláusula-de-teste, a qual deixa no nível 1 da pilha o resultado do teste. REPEAT

remove da pilha o resultado do teste e o avalia: se for verdadeiro ($\neq 0$), a cláusula-cíclica é executada novamente; caso contrário a execução prossegue com a instrução seguinte a END.

Se o argumento do REPEAT for um objeto algébrico ou nome (de uma variável), ele é automaticamente convertido para número.

Vamos ilustrar o uso desta estrutura através de alguns exemplos.

Exemplo 14: (UNESP - 84) Seja $S_n = \frac{1}{2^1} + \frac{1}{2^2} + \cdots + \frac{1}{2^n}$, n um número natural diferente de zero. O menor número n tal que $S_n > 0,99$ é:

a) 5 b) 6 c) 7 d) 8 e) 9

A idéia aqui é variarmos n (a partir de 1) e irmos somando os termos $\frac{1}{2^n}$ até que o resultado da soma seja maior que $0,99$. Vejamos alguns computos:

n	S_n
1	$\frac{1}{2} = 0,5$
2	$\frac{1}{2} + \frac{1}{2^2} = 0,75$
3	$\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} = 0,875$
...	.....

Vamos fazer melhor: o programa vai receber como entrada um número L que, em particular, pode ser $L = 0,99$. Então:

```
≪ → L
   ≪ 1 'n' STO 0 'S' STO
      WHILE 'S ≤ L'
      REPEAT '1/2^n' EVAL
                S + 'S' STO
                'n' INCR DROP
      END n 1 −
   ≫
≫
```

Para melhor compreensão de como o programa funciona, execute-o no `DBUG`.

Exemplo 15: (F.C.M. STA. CASA - 82) Para que a soma dos termos da sequência $(-81, -77, -73, \ldots)$ seja um número positivo, deve-se considerar no mínimo:

a) 35 termos b) 39 termos c) 41 termos d) 42 termos e) 43 termos

Solução: A sequência dada é uma P.A. de primeiro termo $a_1 = -81$ e $r = -77 - (-81) = 4$. Logo, $a_n = a_1 + (n-1)r = -81 + (n-1) \cdot 4$. Podemos também usar a fórmula da soma dos termos de uma P.A., assim:

$$\begin{aligned} S_n &= na_1 + \frac{n(n-1)}{2} r \\ &= n \cdot (-81) + \frac{n(n-1)}{2} 4 = -81n + 2n(n-1) \end{aligned} \tag{3.6}$$

Então, vamos calcular esta soma, incrementando n, até que a mesma seja positiva. Temos,

```
≪  1   ' n '   STO
   WHILE   ' 2n(n − 1) − 81n ≤ 0 '
   REPEAT   ' n '   INCR   DROP
   END   n
≫
```

Rodando o programa este nos devolve $n = 42$. Podemos confirmar em (3.6):

$$n = 41 \;\Rightarrow\; S_{41} = -81 \cdot 41 + 2 \cdot 41 \cdot (41 - 1) = -41$$

$$n = 42 \;\Rightarrow\; S_{42} = -81 \cdot 42 + 2 \cdot 42 \cdot (42 - 1) = 42$$

Exercícios

20) Determine n tal que $\sum_{i=1}^{n} 2^i = 4088$.

21) Resolva o problema anterior utilizando a fórmula para a soma dos termos de uma P.G.: $S_n = \frac{a_1 \cdot q^n - a_1}{q-1}$.

22) Resolva a questão da UNESP (pág. 39) utilizando a fórmula para a soma dos termos de uma P.G.

23) Elabore um programa onde entramos com o primeiro termo e a razão de uma P.A., e ainda com um número L (maior ou igual ao primeiro termo), e o programa saia com n, a quantidade máxima de termos que podemos somar para que a soma não ulrapasse L.

24) Qual é o primeiro termo positivo da P.A. $(-81, -77, -73, \ldots)$?

25) Encontre o valor de k de modo que:

$$\sum_{j=1}^{k} \sum_{i=1}^{30} (-2i + 3j) = 300$$

26) Quantos termos devem ser somados na sequência ,

$$2 \quad -1 \qquad 2 \quad -1 \qquad 2 \quad -1 \qquad 2 \quad -1 \qquad 2 \quad -1 \quad \ldots$$

a partir do primeiro termo, para que a soma seja 15?

Dado: $S_n = [\, 2n - 3 \cdot (-1)^n + 3 \,] \cdot \frac{1}{4}$.

27) Quantos termos devem ser somados na sequência ,

$$-1 \qquad 3 \quad -1 \qquad 3 \quad -1 \qquad 3 \quad -1 \qquad 3 \quad -1 \qquad 3 \quad \ldots$$

a partir do primeiro termo, para que a soma seja 13?

Dado: $S_n = (-1)^n + n - 1$.

28) Diz-se que o número real L é limite da sequência (x_n) de números reais, e escreve-se $L = \lim x_n$ ou $L = \lim_{n\to\infty} x_n$, quando para cada número real $\varepsilon > 0$, dado arbitrariamente, for possível obtermos um inteiro $n_0 \in \mathbb{N}$ tal que $|x_n - L| < \varepsilon$ sempre que $n > n_0$. Em linguagem simbólica:

$$\lim_{n\to\infty} x_n = L \Leftrightarrow \forall \varepsilon > 0, \exists\, n_0 \in \mathbb{N};\ n > n_0 \ \Rightarrow\ |x_n - L| < \varepsilon$$

Exemplos:

i) A sequência dada por $1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{n}, \ldots$ tem 0 como limite;

ii) A sequência dada por $x_n = \sqrt[n]{n}$ tem 1 como limite.

Faça um programa no qual entramos com $\varepsilon > 0$ e o mesmo saia com o n_0 da definição.

Obs: a fórmula do termo geral da sequência e o seu limite devem estar embutidos no programa.

29) Faça um programa que saia com os N primeiros números primos em uma lista.

3.2 Estruturas Condicionais

3.2.1 IF - THEN - END

Entre com esta estrutura num programa pressionando a seguinte sequência de teclas:

A sintaxe desta estrutura é assim:

```
≪ ... IF
        cláusula-de-teste
      THEN
        cláusula-verdadeira
      END ...
≫
```

Esta estrutura executa uma sequência de comandos somente se o teste resultar verdadeiro. A cláusula de teste pode ser uma sequência de comandos − da pilha − (por exemplo $A\ B\ <$) ou uma expressão algébrica (por exemplo $'A < B'$), sendo neste último caso desnecessário executar →NUM ou EVAL.

A palavra IF inicia a cláusula-de-teste, a qual deixa o resultado do teste (0 ou 1) na pilha. THEN remove este resultado. Se o valor é diferente de zero, a cláusula verdadeira é executada. Caso contrário, a execução do programa prossegue com a instrução seguinte a END.

Exemplo 16: (O menor).

```
≪ → A B
   ≪ IF
        A B <
      THEN A
      END
   ≫
≫
```

```
≪ → A B
   ≪ IF
        'A < B'
      THEN A
      END
   ≫
≫
```

Os dois programas acima executam a mesma tarefa, no da esquerda a cláusula-de-teste é dada via operações de pilha, no da direita via expressão algébrica.

O programa recebe dois números, se o primeiro é menor então é "impresso".

Exemplo 17: (Nos diz se um número é par).

Iremos fazer um programa que nos diz se um dado inteiro é par. Faremos este programa de dois modos distintos para ilustrar duas funções para números reais que são FP e MOD (ver tabela, pág. 6)

i) Usando FP

```
≪ → N
   ≪ IF 'FP(N/2) == 0'
      THEN ''O número é par.''
      END
   ≫
≫
```

Nota: O "duplo igual" : == é de comparação. Você pode digitá-lo diretamente do teclado da calculadora, digitando duas vezes =.

ii) Usando MOD

```
≪ → N
   ≪ IF
        N 2 MOD 0 ==
      THEN ''O número é par.''
      END
   ≫
≫
```

Neste programa N e 2 são colocados na pilha, MOD coloca na pilha o resto da divisão de N por 2, o qual é comparado com 0. Execute-o no `DBUG`.

Vamos nos deter mais na próxima estrutura condicional, por ser a mais utilizada.

3.2.2 IF - THEN - ELSE - END

Entre com esta estrutura num programa pressionando a seguinte sequência de teclas:

A sintaxe desta estrutura funciona assim:

```
≪ ... IF
        cláusula-de-teste
      THEN
        cláusula-verdadeira
      ELSE
        cláusula-falsa
      END ...
≫
```

Esta estrutura executa uma sequência de comandos se o teste resultar verdadeiro e outra, caso seja falso. Se a cláusula-de-teste for um objeto algébrico, ela é automáticamente convertida para um valor numérico.

A palavra IF inicia a cláusula-de-teste, a qual deixa o resultado (0 ou 1) na pilha. THEN remove este resultado. Se o valor é diferente de 0, a cláusula-verdadeira é executada; caso contrário, a cláusula-falsa é executada. Após ter executado a cláusula apropriada, o programa prossegue com a instrução seguinte à END.

Exemplo 18: (Nos diz se um número é par ou ímpar).

O programa seguinte nos diz se um número é par ou ímpar.

```
≪ → N
   ≪ IF   'FP(N/2) == 0'
      THEN  ''O número é par.''
      ELSE  ''O número é impar.''
      END
   ≫
≫
```

Exemplo 19: . Fazer um programa para sair com os N primeiros termos da sequência

$$a_n = \begin{cases} \frac{n}{2}, & \text{se } n \text{ é par;} \\ \frac{n+1}{2}, & \text{se } n \text{ é ímpar.} \end{cases}$$

Então,

```
≪ → N
   ≪ 1 N
         FOR n
            IF 'FP(n/2) == 0'
            THEN n 2 /
            ELSE n 1 + 2 /
            END
         NEXT N ROW→
   ≫
≫
```

Interprete o programa acima e execute-o no `DBUG`.

Exemplo 20: (U.E.CE. - 80) Considere a sequência de números reais definida por

$$a_n = \begin{cases} a_{n-1}, & \text{se } n \text{ é par;} \\ \frac{n+1}{2}, & \text{se } n \text{ é ímpar.} \end{cases}$$

Então o produto dos seis primeiros termos é igual a:

a) 48 b) 30 c) 36 d) 42

Vamos fazer um programa para sair com os N primeiros termos da sequência acima. Observe que quando n é par o $n$ésimo termo é igual ao termo anterior (a_{n-1}), motivo pelo qual usaremos um vetor para guardar os termos da sequência . Então:

```
≪ → N
   ≪ [0] {N} RDM 'A' STO
     1 N FOR n
            IF 'FP(n/2) ≠ 0'
            THEN '(n+1)/2' EVAL 'A(n)' STO
            ELSE 'A(n-1)' EVAL 'A(n)' STO
            END
         NEXT A
   ≫
≫
```

Interprete o programa acima e execute-o no `DBUG` para dirimir quaisquer dúvidas.

Uma fórmula alternativa para a sequência dada é: $a_{n} = [\,2n+1-(-1)^{n}\,]/4$.

Exemplo 21: (PUC- SP - 76) Se **A** é uma matriz 3 por 2 definida pela lei

$$a_{ij} = \begin{cases} 1, & \text{se } i = j; \\ i^2, & \text{se } i \neq j. \end{cases}$$

Então **A** se escreve:

a) $\begin{bmatrix} 1 & 4 & 9 \\ 1 & 1 & 9 \end{bmatrix}$ b) $\begin{bmatrix} 1 & 1 \\ 4 & 1 \\ 9 & 9 \end{bmatrix}$ c) $\begin{bmatrix} 1 & 1 \\ 1 & 4 \\ 9 & 9 \end{bmatrix}$ d) $\begin{bmatrix} 1 & 1 & 9 \\ 1 & 4 & 9 \end{bmatrix}$ e) $\begin{bmatrix} 1 & 1 \\ 4 & 1 \\ 6 & 6 \end{bmatrix}$

Vamos resolver este problema para uma matriz de dimensão genérica $M \times N$. Assim:

```
≪ → M N
   ≪ 1 M FOR I
          1 N FOR J
                IF 'I == J'
                THEN 1
                ELSE I 2 ∧
                END
             NEXT N ROW→
          NEXT M ROW→
   ≫
≫
```

Exemplo 22: (Com)prove que uma matriz **A** cujos elementos estão dados pela relação,

$$a_{ij} = \begin{cases} (-1)^{j-1}\binom{j-1}{i-1}, & \text{se } i < j; \\ (-1)^{i-1}, & \text{se } i = j; \\ 0, & \text{se } i > j. \end{cases}$$

satisfaz a relação $\mathbf{A}^2 = \mathbf{I}$.

Vamos fazer um programa para comprovar esta relação. Antes, observamos que $\binom{j-1}{i-1}$ significa uma combinação e que, na HP, encontra-se disponível em

[←] [MTH] [NXT L] [PROB] [COMB] . O programa fica assim:

```
≪ → L r
   ≪ 1 N FOR I
        1 N FOR J
                IF 'I < J'
                THEN
                    '(−1) ^ (J − 1) * COMB(J-1, I-1)' EVAL
                ELSE
                    IF 'I == J'
                    THEN '(−1) ^ (I − 1)' EVAL
                    ELSE 0
                    END
                END
            NEXT N ROW→
        NEXT N ROW→
   ≫
≫
```

O programa sai com a matriz $\mathbf{A}$. Para comprovar que $\mathbf{A}^2 = \mathbf{I}$ ($\mathbf{I}$ é matriz identidade) basta dar ENTER e $\times$.

Exemplo 23: (Limite da função quadrática)

Definição (Limite): Sejam $D \subset \mathbb{R}$ um conjunto de números reais, $f\colon D \to \mathbb{R}$ uma função real e $a \in D'$ um *ponto de acumulação* de D. Diz-se que o número real L é *limite* de $f(x)$ quando x tende a a, e escreve-se $\lim\limits_{x \to a} = L$, quando, para todo $\varepsilon > 0$ dado arbitrariamente, pode-se obter $\delta > 0$ tal que se tem $|f(x) - L| < \varepsilon$ sempre que $x \in X$ e $0 < |x - a| < \delta$.

Em [7] provamos o seguinte,

Teorema 1. *Se* $f(x) = ax^2 + bx + c$ *então* $\lim\limits_{x \to d} f(x) = f(d)$, *onde* $f(d) = a \cdot d^2 + b \cdot d + c$.

No referido opúsculo deduzimos a seguinte fórmula,

$$\delta(\varepsilon) = \begin{cases} \min\left\{1,\ \frac{\varepsilon}{a\,(2d + \frac{a}{|a|}) + b}\right\}, & \text{se} \quad a\,(2d-1) + b \geq 0; \\ \min\left\{1,\ \frac{-\varepsilon}{a\,(2d - \frac{a}{|a|}) + b}\right\}, & \text{se} \quad a\,(2d-1) + b < 0. \end{cases} \tag{3.7}$$

que nos dá o δ em função de ε para o limite em questão.

Exemplos (pág. 63-LEITHOLD): Prove que o limite é o número indicado,

35. $\lim\limits_{x \to 1} x^2 = 1$. Neste caso, temos: $a = 1$, $b = 0$, $d = 1$. Então, sendo

$$a\,(2d-1) + b = 1 \cdot (2 \cdot 1 - 1) + 0 = 1 > 0, \quad \text{temos}$$

$$\delta(\varepsilon) = \min\left\{1,\ \frac{\varepsilon}{a\,(2d + \frac{a}{|a|}) + b}\right\}$$

$$= \min\left\{1,\ \frac{\varepsilon}{1\cdot(\,2\cdot 1 + \frac{1}{|1|}\,) + 0}\right\} = \min\left\{1,\ \frac{\varepsilon}{3}\right\}$$

36. $\lim\limits_{x\to -3} x^2 = 9$. Neste caso, temos: $a = 1$, $b = 0$, $d = -3$. Então, sendo

$$a\,(2d-1) + b = 1\cdot(2\cdot(-3) - 1) + 0 = -7 < 0, \quad \text{temos}$$

$$\delta(\varepsilon) = \min\left\{1,\ \frac{-\varepsilon}{a\,(2d - \frac{a}{|a|}) + b}\right\}$$

$$= \min\left\{1,\ \frac{-\varepsilon}{1\cdot(\,2\cdot(-3) - \frac{1}{|1|}\,) + 0}\right\} = \min\left\{1,\ \frac{\varepsilon}{7}\right\}$$

15. $\lim\limits_{x\to 0}(x^2 + 3x - 4) = -4$; $\varepsilon = 0,\,03$. Neste caso, temos: $a = 1$, $b = 3$, $d = 0$. Então, sendo

$$a\,(2d-1) + b = 1\cdot(2\cdot 0 - 1) + 3 = 2 > 0, \quad \text{temos}$$

$$\delta(\varepsilon) = \min\left\{1,\ \frac{\varepsilon}{a\,(2d + \frac{a}{|a|}) + b}\right\}$$

$$= \min\left\{1,\ \frac{0,\,03}{1\cdot(\,2\cdot 0 + \frac{1}{|1|}\,) + 3}\right\} = \min\left\{1;\ 0,\,0075\right\} = 0,\,0075$$

O programa a seguir implementa a fórmula (3.7):

```
≪ → a b d ε
   ≪ IF 'a*(2*d-1)+b≥0'
        THEN 1
             'ε/(a*(2*d+a/ABS(a))+b)' EVAL
             MIN ''δ(ε)'' →TAG
        ELSE 1
             '-ε/(a*(2*d-a/ABS(a))+b)' EVAL
             MIN ''δ(ε)'' →TAG
        END
   ≫
≫
```

Nota: Caracteres como δ e ε são acessados pressionando-se as teclas  CHARS.

Para acessar outros caracteres, além dos que são mostrados de imediato, pressione ▽ (botão prateado). Coloque (com auxílio dos botões prateados) o cursor sobre o símbolo desejado aí é só pressionar a tecla virtual ECHO 1.

. Segunda Regra de Simpson

Exemplo 24: A segunda regra de Simpson, para o cálculo de integrais, consta da seguinte fórmula:

$$I = \frac{3h}{8}(y_0 + 3y_1 + 3y_2 + 2y_3 + 3y_4 + 3y_5 + 2y_6 + \cdots + 3y_{n-2} + 3y_{n-1} + y_n) \quad (3.8)$$

Onde: n deve ser um múltiplo de 3 e $h = (b-a)/n$. O erro é dado por,

$$\varepsilon = -\frac{(b-a)^5}{80n^4} f^{(4)}(\xi), \quad a \leq \xi \leq b.$$

Antes de implementar esta fórmula vamos fazer um programa para gerar a sequência dos coeficientes:

$$\begin{array}{lccccccccccc} I: & 0 & 1 & 2 & 3 & 4 & 5 & 6 & \ldots & n-2 & n-1 & n \\ a_{_I}: & 1 & 3 & 3 & 2 & 3 & 3 & 2 & \ldots & 3 & 3 & 1 \end{array} \quad (3.9)$$

Esta sequência é gerada com o seguinte programa:

```
≪ → n
   ≪ 1   1 n 1 −
          FOR I
              IF   'FP(I/3) == 0'
              THEN 2
              ELSE 3
              END
          NEXT 1 n 1 + →ARRY
   ≫
≫
```

Nota: Armazene este programa na variável CSRS, pois será referenciado − com este nome − por um outro programa (será uma subrotina). Não esquecer que n deve ser um múltiplo de 3.

− Implementando a Segunda Regra de Simpson. Como exemplo, vamos fazer um programa para calcular a integral,

$$\int_a^b \ln(x^3 + \sqrt{(}\mathbf{e}^x + 1))\, dx$$

Antes vamos armazenar $f(x)$ numa variável, isto pode ser feito assim:

```
≪ → x   'LN(x^3 + √(EXP(x) + 1))'  ≫
```

Armazenamos este programa em FUNC. O programa muda muito pouco em relação ao da primeira regra de Simpson (pág. 32), assim:

```
≪ → a b n
   ≪ '(b-a)/n' EVAL 'h' STO
                a b FOR x
                      x FUNC
                   h STEP n 1 + →ARRY
                   n CSRS DOT 3 h * 8 / *
   ≫
≫
```

Para $a = 1$, $b = 4$ e $n = 9$, resulta: $I = 8,5620$.

. Integração Dupla

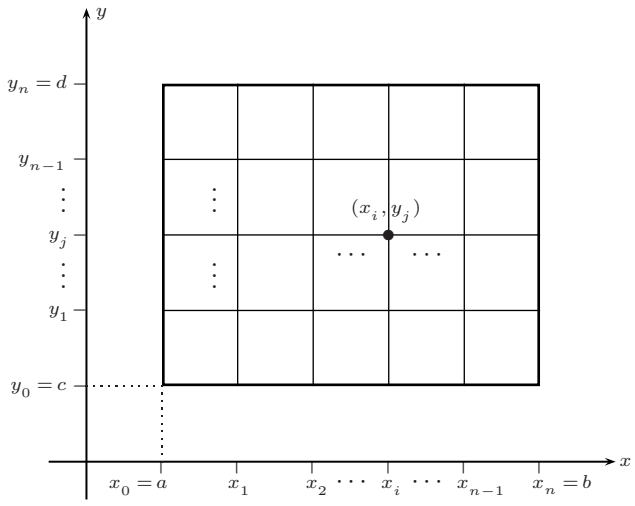
Seja o problema de calcular a integral dupla a seguir:

$$I = \int_D \int f(x,\, y)\, dx\, dy$$

onde D é o retângulo delimitado por:

$$a \leq x \leq b$$
$$c \leq y \leq d$$

Veremos que as regras apresentadas anteriormente podem ser usadas aqui. Inicialmente fazemos uma partição do retângulo anterior em nx subintervalos em $[\,a,\, b\,]$ e em ny subintervalos em $[\,c,\, d\,]$, assim:



Na verdade o n que comparece no eixo x é nx e o n que comparece no eixo y é ny (não são necessáriamente iguais). Pois bem, temos

$$I = \int_D \int f(x,\, y)\, dx\, dy = \int_a^b dx \int_c^d f(x,\, y)\, dy$$

Chamando $\int_c^d f(x,\, y)\, dy$ de $G(x)$, isto é,

$$G(x) = \int_c^d f(x,\, y)\, dy \tag{3.10}$$

podemos escrever:

$$I = \int_a^b G(x)\, dx$$

Para resolver esta integral simples aplicaremos a segunda regra de Simpson (equação (3.8), pág. 48), assim:

$$I = \frac{3h}{8}\big(\, G(x_{_0}) + 3G(x_{_1}) + 3G(x_{_2}) + 2G(x_{_3}) + 3G(x_{_4}) + 3G(x_{_5}) + 2G(x_{_6}) + \cdots$$

$$+ 3G(x_{_{n-2}}) + 3G(x_{_{n-1}}) + G(x_{_n})\,\big) \tag{3.11}$$

Da equação (3.10), temos:

$$G(x_{_i}) = \int_c^d f(x_{_i},\, y)\, dy; \qquad (i = 0,\, 1,\, 2, \ldots, nx) \tag{3.12}$$

Observe que, para cada i fixo, devemos calcular a integral da função (de uma única variável) $f(x_{_i},\, y)$ no intervalo $[\, c,\, d\,]$ (veja partição do retângulo). Para resolver esta integral em y utilizaremos a primeira regra de Simpson (equação (3.3), pág. 30), por exemplo:

$$G(x_{_i}) = \frac{h}{3}\,\big(\, f(x_{_i},\, y_{_0}) + 4f(x_{_i},\, y_{_1}) + 2f(x_{_i},\, y_{_2}) + 4f(x_{_i},\, y_{_3}) + 2f(x_{_i},\, y_{_4}) + \cdots$$

$$+ 2f(x_{_i},\, y_{_{n-2}}) + 4f(x_{_i},\, y_{_{n-1}}) + f(x_{_i},\, y_{_n})\,\big)$$

Observe que temos $nx + 1$ desta equações (veja equação (3.12)). Ademais, observe que como estamos particionando o intervalo $[\, c,\, d\,]$ em ny subintervalos a rigor temos,

$$G(x_{_i}) = \frac{h}{3}\,\big(\, f(x_{_i},\, y_{_0}) + 4f(x_{_i},\, y_{_1}) + 2f(x_{_i},\, y_{_2}) + 4f(x_{_i},\, y_{_3}) + 2f(x_{_i},\, y_{_4}) + \cdots$$

$$+ 2f(x_{_i},\, y_{_{ny-2}}) + 4f(x_{_i},\, y_{_{ny-1}}) + f(x_{_i},\, y_{_{ny}})\,\big) \quad (i = 0,\, 1,\, 2, \ldots, nx)$$

Pois bem, substituindo estas $nx + 1$ equações na equação (3.11), obtemos:

$$
\begin{aligned}
I=\tfrac{3hx}{8}\Big[\ & \tfrac{hy}{3}\Big(F(x_0,y_0)+4F(x_0,y_1)+2F(x_0,y_2)+4F(x_0,y_3)+\cdots+4F(x_0,y_{n-1})+F(x_0,y_n)\Big)\\
&+3\cdot\tfrac{hy}{3}\Big(F(x_1,y_0)+4F(x_1,y_1)+2F(x_1,y_2)+4F(x_1,y_3)+\cdots+4F(x_1,y_{n-1})+F(x_1,y_n)\Big)\\
&+3\cdot\tfrac{hy}{3}\Big(F(x_2,y_0)+4F(x_2,y_1)+2F(x_2,y_2)+4F(x_2,y_3)+\cdots+4F(x_2,y_{n-1})+F(x_2,y_n)\Big)\\
&+2\cdot\tfrac{hy}{3}\Big(F(x_3,y_0)+4F(x_3,y_1)+2F(x_3,y_2)+4F(x_3,y_3)+\cdots+4F(x_3,y_{n-1})+F(x_3,y_n)\Big)\\
&+3\cdot\tfrac{hy}{3}\Big(F(x_4,y_0)+4F(x_4,y_1)+2F(x_4,y_2)+4F(x_4,y_3)+\cdots+4F(x_4,y_{n-1})+F(x_4,y_n)\Big)\\
&+3\cdot\tfrac{hy}{3}\Big(F(x_5,y_0)+4F(x_5,y_1)+2F(x_5,y_2)+4F(x_5,y_3)+\cdots+4F(x_5,y_{n-1})+F(x_5,y_n)\Big)\\
&+2\cdot\tfrac{hy}{3}\Big(F(x_6,y_0)+4F(x_6,y_1)+2F(x_6,y_2)+4F(x_6,y_3)+\cdots+4F(x_6,y_{n-1})+F(x_6,y_n)\Big)\\
&-------------------------------\\
&+3\cdot\tfrac{hy}{3}\Big(F(x_{n-2},y_0)+4F(x_{n-2},y_1)+2F(x_{n-2},y_2)+4F(x_{n-2},y_3)+\cdots+F(x_{n-2},y_n)\Big)\\
&+3\cdot\tfrac{hy}{3}\Big(F(x_{n-1},y_0)+4F(x_{n-1},y_1)+2F(x_{n-1},y_2)+4F(x_{n-1},y_3)+\cdots+F(x_{n-1},y_n)\Big)\\
&+\tfrac{hy}{3}\Big(F(x_n,y_0)+4F(x_n,y_1)+2F(x_n,y_2)+4F(x_n,y_3)+\cdots+4F(x_n,y_{n-1})+F(x_n,y_n)\Big)\Big]
\end{aligned}
$$

Arrumando a casa obtemos:

$$
\begin{aligned}
I=\tfrac{3hx}{8}\cdot\tfrac{hy}{3}\Big[\ & 1\cdot\Big(F(x_0,y_0)+4F(x_0,y_1)+2F(x_0,y_2)+4F(x_0,y_3)+\cdots+4F(x_0,y_{n-1})+F(x_0,y_n)\Big)\\
&+3\cdot\Big(F(x_1,y_0)+4F(x_1,y_1)+2F(x_1,y_2)+4F(x_1,y_3)+\cdots+4F(x_1,y_{n-1})+F(x_1,y_n)\Big)\\
&+3\cdot\Big(F(x_2,y_0)+4F(x_2,y_1)+2F(x_2,y_2)+4F(x_2,y_3)+\cdots+4F(x_2,y_{n-1})+F(x_2,y_n)\Big)\\
&+2\cdot\Big(F(x_3,y_0)+4F(x_3,y_1)+2F(x_3,y_2)+4F(x_3,y_3)+\cdots+4F(x_3,y_{n-1})+F(x_3,y_n)\Big)\\
&+3\cdot\Big(F(x_4,y_0)+4F(x_4,y_1)+2F(x_4,y_2)+4F(x_4,y_3)+\cdots+4F(x_4,y_{n-1})+F(x_4,y_n)\Big)\\
&+3\cdot\Big(F(x_5,y_0)+4F(x_5,y_1)+2F(x_5,y_2)+4F(x_5,y_3)+\cdots+4F(x_5,y_{n-1})+F(x_5,y_n)\Big)\\
&+2\cdot\Big(F(x_6,y_0)+4F(x_6,y_1)+2F(x_6,y_2)+4F(x_6,y_3)+\cdots+4F(x_6,y_{n-1})+F(x_6,y_n)\Big)\\
&-------------------------------\\
&+3\cdot\Big(F(x_{n-2},y_0)+4F(x_{n-2},y_1)+2F(x_{n-2},y_2)+4F(x_{n-2},y_3)+\cdots+F(x_{n-2},y_n)\Big)\\
&+3\cdot\Big(F(x_{n-1},y_0)+4F(x_{n-1},y_1)+2F(x_{n-1},y_2)+4F(x_{n-1},y_3)+\cdots+F(x_{n-1},y_n)\Big)\\
&+1\cdot\Big(F(x_n,y_0)+4F(x_n,y_1)+2F(x_n,y_2)+4F(x_n,y_3)+\cdots+4F(x_n,y_{n-1})+F(x_n,y_n)\Big)\Big]
\end{aligned}
$$

No “quadro” acima podemos oberver, pelos índices de y_j, que temos $ny+1$ parcelas em cada linha e, pelos índices de x_i, observamos que temos $nx+1$ linhas.

Para programar esta equação vamos, antes, fazer um programa que calcule uma matriz com os valores da função em todos os vértices da malha de partição, digo,

$$
\big(F(x_i,\,y_j)\big);\qquad (i=0,\,1,\,2,\ldots,nx),\ \ (j=0,\,1,\,2,\ldots,ny) \tag{3.13}
$$

Ou ainda,

$$\begin{bmatrix} F(x_0, y_0) & F(x_0, y_1) & F(x_0, y_2) & \dots & F(x_0, y_{ny}) \\ F(x_1, y_0) & F(x_1, y_1) & F(x_1, y_2) & \dots & F(x_1, y_{ny}) \\ \dots & \dots & \dots & \dots & \dots \\ F(x_{nx}, y_0) & F(x_{nx}, y_1) & F(x_{nx}, y_2) & \dots & F(x_{nx}, y_{ny}) \end{bmatrix}$$

Vamos tomar como exemplo o cálculo da seguinte integral:

$$I = \int_0^{\pi/2} \int_0^{0,4} (y^2 + y) \cos x \, dy \, dx, \qquad \begin{cases} 0 \leq y \leq 0,4 \\ 0 \leq x \leq \pi/2 \end{cases}$$

Inicialmente necessitamos de um programa que calcule $F(x, y) = (y^2 + y) \cos x$, isto pode ser feito assim:

$$\ll \rightarrow \; x \quad y \quad \text{'}(y\,\hat{}\,2 + y) * \cos(x)\text{'} \; \gg$$

Armazene este programa na variável FUNC (função). Pois bem, o programa para o cálculo da matriz acima é dado a seguir:

```
≪ → a b c d nx ny
  ≪ '(b − a)/nx' EVAL 10 TRNC 'hx' STO
     '(d − c)/ny' EVAL 10 TRNC 'hy' STO
        c d FOR y
           a b FOR x
                    x y FUNC 12 RND
              hx STEP nx 1 + →ARRY
           hy STEP ny 1 + ROW→
  ≫
≫
```

Nota: Armazene este programa na variável FDV, pois será referenciado − com este nome − por um outro programa (será uma subrotina).

Comentários:

a) Este programa calcula a matriz $\left(F(x_i, y_j) \right)$ para: $a \leq x \leq b$, $c \leq y \leq d$. nx e ny é o número de subintervalos nos intervalos $[a, b]$ e $[c, d]$, respectivamente.

b) Na segunda linha (10 TRNC) `truncamos` o passo hx pois, caso contrário, dá problema no FOR-STEP (para incrementos envolvendo π).

Por exemplo, para a HP, $\frac{\pi}{2} + \frac{\pi}{4} \neq \frac{3\pi}{4}$; na verdade, $\frac{\pi}{2} + \frac{\pi}{4} > \frac{3\pi}{4}$. Confirme isto na sua calculadora.

c) Dentro do FOR x (12 RND) arredondamos o resultado da função; isto se deve a que, por exemplo, $\cos \frac{\pi}{2} = -5,1034 \times 10^{-12}$ e não 0 como deveria ser. Com este arredondamento na matriz sai 0 mesmo.

Por exemplo, entrando no programa anterior com os seguintes dados:

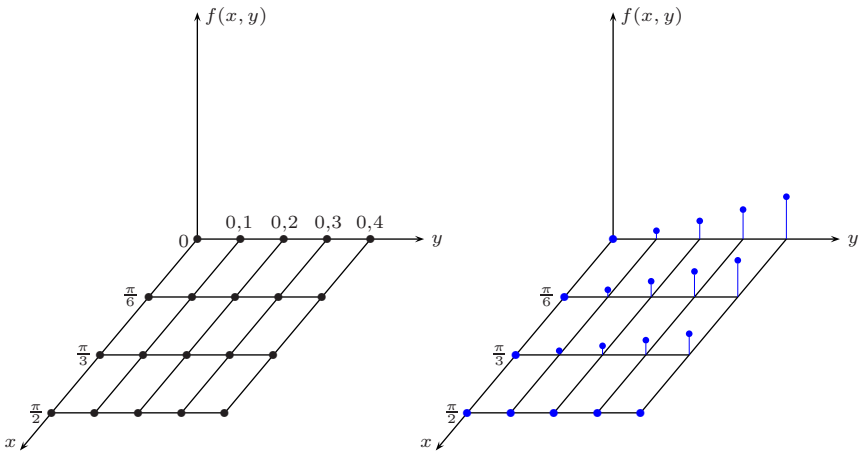
$$\rightarrow \quad 0 \quad \pi/2 \quad 0 \quad 0,4 \quad 3 \quad 4$$

recebemos na saida a seguinte matriz:

$$\begin{bmatrix} 0.0000 & 0.1100 & 0.2400 & 0.3900 & 0.5600 \\ 0.0000 & 0.0953 & 0.2078 & 0.3377 & 0.4850 \\ 0.0000 & 0.0550 & 0.1200 & 0.1950 & 0.2800 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix}$$

com 4 casas decimais.

– Na figura a seguir plotamos (esquerda) a malha da partição e, na figura da direita, os valores $f(x_i, y_j)$ de acordo com a matriz anterior, assim:



Nota: Na página 71 mostramos como plotar o gráfico da superfície $f(x, y)$.

Finalmente, o programa que nos interessa é dado a seguir:

```
≪ → a b c d nx ny
   ≪ nx 1 + 1 →LIST 0 CON 'VL' STO
      a b c d nx ny FDV 'MF' STO
      nx CSRS 'Vx' STO
      ny CPRS 'Vy' STO
      1 nx 1 + FOR I
                  MF I ROW− SWAP DROP
                  Vy DOT VL I ROT PUT
                  'VL' STO
               NEXT VL Vx DOT
               3 hx * 8 / hy 3 / * *
   ≫
≫
```

Comentários:

a) Os dados de entrada para este programa é o retângulo $[\,a,\,b\,]\times[\,c,\,d\,]$ e o número de subretângulos, $nx \times ny$, da partição.

b) A segunda linha do programa cria um vetor, de comprimento $nx+1$, com a constante 0:

$$[\,0 \quad 0 \quad 0 \quad 0 \quad \ldots \quad 0\,] \quad \text{(comprimento deste vetor} = nx+1)$$

Este vetor é guardado na variável VL. Observe que $nx+1$ é igual ao número de linhas no "quadro" da pág. 51;

c) A terceira linha do programa disponibiliza os dados da subrotina FDV que calcula a matriz $\big(\,F(x_{_i},\,y_{_j})\,\big)$ e guarda-a na variável MF;

d) A quarta linha do programa disponibiliza os dados da subrotina CSRS (pág. 48) que calcula os coeficientes da segunda regra de Simpson e os guarda no vetor Vx;

e) A quinta linha do programa disponibiliza os dados da subrotina CPRS (pág. 31)que calcula os coeficientes da primeira regra de Simpson e os guarda no vetor Vy;

f) Agora o FOR-NEXT vai extrair cada uma das linhas da matriz $\big(\,F(x_{_i},\,y_{_j})\,\big)$ e fazer o `produto interno`, de cada linha, com o vetor dos coeficientes Vy, conforme "quadro" à pág. 51; observe que o resultado de cada um destes produtos internos vai sendo guardado no vetor VL que estava inicialmente "vazio". Ao sair do laço o vetor VL guarda, em sua posição i, a soma da linha i do "quadro" (apenas as somas entre os dois parêntesis maiores);

g) Agora calculamos o produto interno entre os vetores VL e Vx e, para finalizar, multiplicamos o resultado por $\frac{3hx}{8}\cdot\frac{hy}{3}$.

Por exemplo, entrando no programa anterior com os seguintes dados:

$$\rightarrow \quad 0 \quad \pi/2 \quad 0 \quad 0,4 \quad 3 \quad 4$$

recebemos de volta: 0.1014, como valor para a Integral dupla.

3.3 Operadores lógicos relacionais

Um outro recurso − não menos importante − que a programação nos oferece são os operadores lógicos booleanos: OR, AND, XOR, NOT, etc., acessados com a seguinte sequência de teclas:

Estes operadores estão definidos pelas respectivas tabelas-verdade, digo:

p	q	p OR q
V	V	V
V	F	V
F	V	V
F	F	F

p	q	p AND q
V	V	V
V	F	F
F	V	F
F	F	F

p	q	p XOR q
V	V	F
V	F	V
F	V	V
F	F	F

p	$\bar{p}$
V	F
F	V

Ou ainda, na forma em que a HP entende e opera:

p	q	p OR q
1	1	1
1	0	1
0	1	1
0	0	0

p	q	p AND q
1	1	1
1	0	0
0	1	0
0	0	0

p	q	p XOR q
1	1	0
1	0	1
0	1	1
0	0	0

p	$\bar{p}$
1	0
0	1

Vejamos um exemplo de aplicação destes operadores:

Exemplo 25: (CESGRANRIO - 76) Seja H o conjunto $\{\, n \in \mathbb{N} \colon 2 \leq n \leq 40 \,\}$, n múltiplo de 2, n não múltiplo de 3. O número de elementos de H é:

a) 12 b) 14 c) 7 d) 13 e) 6

Observe, na definição do conjunto em questão: n é múltiplo de 2 **e** n não é múltiplo de 3. Aqui temos uma tarefa para o operador lógico **AND**.

Vamos fazer melhor, vamos fazer um programa que recebe dois números naturais M e N $(N < M)$ e devolve todos os múltiplos de 2, e não de 3, entre N e M; assim:

```
≪ → N M
   ≪ N M FOR n
                IF 'FP(n/2) == 0 AND FP(n/3) ≠ 0 '
                THEN n
                END
             NEXT DEPTH →LIST
   ≫
≫
```

Nota: O comando DEPTH retorna o número de objetos na pilha (veja tabela de comandos da pilha, pág. 7).

O programa seguinte sai com os múltiplos de 2 **ou** 3 entre N e M ($N < M$).

```
≪ → N M
   ≪ N M FOR n
            IF 'FP(n/2) == 0 OR FP(n/3) == 0 '
            THEN n
            END
         NEXT DEPTH →LIST
   ≫
≫
```

Exercícios

30) (F. SANTANA - 83) Dadas as matrizes $\mathbf{A} = (\,a_{ij}\,)_2$, tal que

$$a_{ij} = \begin{cases} 1+i, & \text{se } i = j; \\ 0, & \text{se } i \neq j. \end{cases}$$

e $\mathbf{B} = (\,B_{ij}\,)_2$, tal que $b_{ij} = 2i - 3j$, então $\mathbf{A} + \mathbf{B}$ é igual a:

a) $\begin{bmatrix} -1 & 4 \\ -1 & -2 \end{bmatrix}$ b) $\begin{bmatrix} 1 & -4 \\ -1 & -2 \end{bmatrix}$ c) $\begin{bmatrix} -1 & 4 \\ 1 & 2 \end{bmatrix}$ d) $\begin{bmatrix} 1 & -4 \\ 1 & 2 \end{bmatrix}$ e) $\begin{bmatrix} 1 & 4 \\ 1 & 2 \end{bmatrix}$

31) Fazer um programa para sair com a matriz identidade de ordem N:

$$a_{ij} = \begin{cases} 1, & \text{se } i = j; \\ 0, & \text{se } i \neq j. \end{cases}$$

32) (U. MACK. - 80) Dada a matriz $\mathbf{A} = (\,a_{ij}\,)_2$ tal que

$$a_{ij} = \begin{cases} \operatorname{sen} \frac{\pi}{2} j, & \text{se } i = j; \\ \cos \pi j, & \text{se } i \neq j. \end{cases}$$

Então $\mathbf{A}^2$ é a matriz:

a) $\begin{bmatrix} -1 & -1 \\ 1 & 0 \end{bmatrix}$ b) $\begin{bmatrix} 0 & -1 \\ 1 & 1 \end{bmatrix}$ c) $\begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix}$ d) $\begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix}$ e) $\begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix}$

33) (UFRS - 83) $\mathbf{A} = (\,a_{ij}\,)$ é uma matriz de ordem 2×2 com $a_{ij} = 2^{-i}$ se $i = j$ e $a_{ij} = 0$ se $i \neq j$. A inversa de $\mathbf{A}$ é:

a) $\begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{4} \end{bmatrix}$ b) $\begin{bmatrix} -\frac{1}{2} & 0 \\ 0 & -\frac{1}{4} \end{bmatrix}$ c) $\begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}$ d) $\begin{bmatrix} -2 & 0 \\ 0 & -4 \end{bmatrix}$ e) $\begin{bmatrix} 2 & 0 \\ 0 & 2^{\frac{1}{2}} \end{bmatrix}$

34) (FCM STA CASA - 81) Seja a matriz $\mathbf{A} = (\, a_{ij}\,)$, de ordem 3, tal que

$$a_{ij} = \begin{cases} 1 & \text{se } i < j; \\ k & \text{se } i = j \text{ e } k \in \mathbb{R}; \\ -1 & \text{se } i > j. \end{cases}$$

Se o determinante de $\mathbf{A}$ é igual a zero, então k pertence ao conjunto:

a) $k \in \{\, -3,\, 1,\, 3\,\}$ b) $k \in \{\, -2,\, 1,\, 2\,\}$ c) $k \in \{\, 0,\, 1/3,\, 1/2\,\}$ d) $k \in \{\, -\sqrt{3},\, \sqrt{3}\,\}$
e) $k \in \{\, -1/3,\, 1/3\,\}$

35) (U. MACK - 80) Seja a função $f : \mathbb{R} \to \mathbb{R}$ definida por

$$f(x) = \begin{cases} |x| + 3 & \text{se } |x| \leq 2; \\ |x - 3| & \text{se } |x| > 2 \end{cases}$$

O valor de $f(f(f(\ldots f(0)\ldots)))$,

a) é 0 b) pode ser 1 c) é 3 d) pode ser 3 e) é impossível de calcular.

Neste exercício faça um programa onde entra-se com N e o mesmo sai com uma lista com as N composições de f em 0. Exemplo,

$$\text{Se } N = 1 \text{ então } f(f(0))$$

$$\text{Se } N = 2 \text{ então } f(f(f(0))), \text{ etc.}$$

36) (PUC CAMP. - 80) Considerando $\mathbb{N} = \{\, 0,\, 1,\, 2,\, 3, \ldots\}$ e, ainda,

$$A = \{\, x \in \mathbb{N} / \; \frac{24}{x} = n,\; x \in \mathbb{N}\,\}$$

$$B = \{\, x \in \mathbb{N} / \; 3x + 4 < 2x + 9\,\}$$

podemos afirmar que,

a) $A \cup B$ tem 8 elementos b) $A \cup B = A$ c) $A \cap B = A$
d) $A \cap B$ possui 4 elementos b) n.d.a.

37) (ITA - 66) Quantos números inteiros existem, de 1000 a 10000, não divisíveis nem por cinco nem por sete?

38) (UFRN - 84) O número de múltiplos de sete entre 50 e 150 é:

a) 9 b) 12 c) 14 d) 16 e) 23

39) (CESCEA - 75) Quantos números ímpares há entre 14 e 192?

a) 88 b) 89 c) 87 d) 86 e) 90

40) (FGV - 81) A soma dos números naturais não superiores a 1000, não divisíveis por 7, é:

a) 429429 b) 500500 c) 500500/7 d) 999999/7 e) n.d.a.

41) (CESGRANRIO - 84) A soma dos números naturais menores que 100 e que divididos por 5 deixam resto 2 é:

a) 996 b) 976 c) 990 d) 991 e) 998

Capítulo 4

Outros Tópicos de Interesse

4.1 Cálculo de Combinações

> "A obtenção de um resultado novo em pesquisa é, para o cientista, uma fonte de intenso prazer, ligado intimamente ao instinto de criação e eternidade, pois, independentemente da importância da contribuição no contexto da ciência, ou de sua utilização, representa algo acrescentado ao conhecimento humano que marca sua existência na terra." (`Pierre Curie`)

A conhecida fórmula da análise combinatória $\binom{n}{r} = \frac{n\,!}{(n-r)!\,r!}$ nos fornece o número de combinações dos n elementos de um conjunto, tomados r a r. Mas esta fórmula não nos fornece as tais combinações. O nosso objetivo neste apêndice é apresentar uma fórmula que tem precisamente esta finalidade.

Problema: Dado o conjunto $A = \{a_1, a_2, a_3, a_4\}$, obter todas as combinações possíveis de seus elementos.

Solução: O raciocínio que será desenvolvido a seguir se estende a um conjunto com um número arbitrário de elementos.

Todas as combinações podem ser obtidas da seguinte matriz:

a_1	a_2	a_3	a_4	$\{a_i\}$
1	1	1	1	$\{a_1, a_2, a_3, a_4\}$
−1	1	1	1	$\{a_2, a_3, a_4\}$
1	−1	1	1	$\{a_1, a_3, a_4\}$
−1	−1	1	1	$\{a_3, a_4\}$
1	1	−1	1	$\{a_1, a_2, a_4\}$
−1	1	−1	1	$\{a_2, a_4\}$
1	−1	−1	1	$\{a_1, a_4\}$
−1	−1	−1	1	$\{a_4\}$
1	1	1	−1	$\{a_1, a_2, a_3\}$
−1	1	1	−1	$\{a_2, a_3\}$
1	−1	1	−1	$\{a_1, a_3\}$
−1	−1	1	−1	$\{a_3\}$
1	1	−1	−1	$\{a_1, a_2\}$
−1	1	−1	−1	$\{a_2\}$
1	−1	−1	−1	$\{a_1\}$
−1	−1	−1	−1	$\emptyset$

Onde convencionamos que 1 significa que o elemento entra na combinação e que -1 significa que o elemento não entra na combinação.

O leitor pode inferir fácilmente a lei de construção desta matriz. Só observamos que o número de colunas é igual ao número n de elementos do conjunto, o número de linhas é igual ao de subconjuntos 2^n.

Esta matriz, apropriadamente modificada, comparece em Lógica e Eletrônica Digital. Na referência [5] deduzimos (e demonstramos) a seguinte fórmula,

$$a_{ij} = (-1)^{\left\lfloor \frac{i-1}{2^{j-1}} \right\rfloor} \qquad (\texttt{MC})$$

que gera a matriz de combinações acima.

Na pág. 62 provamos que, de fato, esta matriz calcula combinações.

Nota: $\lfloor x \rfloor$ é o maior inteiro que não supera x. Na HP você acessa esta função com o comando FLOOR (ver tabela dos reais, pág. 6).

– Já não conto mais o número de fórmulas que deduzi (demonstrei) na matemática, confesso que, pela fórmula acima, tenho um carinho todo especial*.

O programa a seguir sai com as combinações de N objetos, tomados r a r. Devemos entrar com uma lista contendo os objetos e r (nesta ordem).

```
≪ → L r
   ≪ { } 'LC' STO L SIZE 'N' STO
      1 2 N ^ FOR I
            1 N FOR J
                  IF '(-1)^FLOOR((I-1)/2^(J-1)) == 1'
                  THEN J
                  END
               NEXT DEPTH DUP 'C' STO
                  IF 'C==r'
                  THEN ROW→ 'V' STO 1 r
                       FOR K L 'V(K)' EVAL
                           GET
                       NEXT r
                         →LIST 1 →LIST LC + 'LC' STO
                  ELSE CLEAR
                  END
            NEXT LC
   ≫
≫
```

Por exemplo, para a seguinte entrada,

```
{ a b c d } 3
```

o programa nos devolve,

```
{ { a b c } { a b d } { a c d } { b c d } }
```

Para a seguinte entrada: `{ a ∑ c π -1 } 3`, o programa nos devolve,

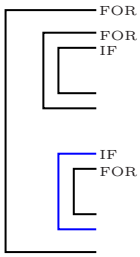
*Precisamente pelos detalhes técnicos envolvidos em sua dedução e demonstração.

{ { a ∑ c } { a ∑ π } { a c π } { ∑ c π } { a ∑ -1 }
{ a c -1 } { ∑ c -1 } { a π -1 } { ∑ π -1 } { c π -1 } }

Nota: Para calcular o número de combinações, $\binom{n}{r}$, na *HP*, pressione as teclas: ← MTH NXT PROB COMB

Descrição do programa:

Inicialmente observe que temos uma concatenação de estruturas com a seguinte configuração:



1. Inicialmente (segunda linha do programa) armazenamos uma lista vazia na variável LC (esta lista guardará as combinações "úteis");

2. Depois colocamos a lista (com os objetos) na pilha e pedimos seu comprimento (dimensão) o qual é armazenado na variável N;

3. Em seguida fixamos a variação do primeiro FOR: de 1 até 2^N; este FOR irá gerar os índices das linhas da *matriz de combinações*;

4. Após, fixamos a variação do segundo FOR: de 1 até N; este FOR irá gerar os índices das colunas da *matriz de combinações*;

5. O primeiro IF *testa* se o elemento a_{ij} da matriz é igual a 1; se for este o caso colocamos o índice da coluna deste elemento na pilha;

– Resumindo até aqui: o primeiro FOR fixa uma linha (da matriz); o segundo FOR pesquisa todas as colunas desta linha procurando os elementos iguais a 1 e guarda os índices (posições) destes elementos.

6. Ao sair do segundo laço (FOR) perguntamos (via comando DEPTH) quantos elementos (índices) existem na pilha;*

7. Em seguida, duplicamos esta informação e guardamos uma cópia na variável C;

8. Após, temos o segundo IF que testa se o número de elementos iguais a 1, na linha pesquisada, é igual a r; se for este o caso guardamos (com ROW→) os índices em um vetor V e o armazenamos na variável V; o FOR a seguir tem a incubência de retirar, da lista original, todos os elementos com índices no vetor V (ou seja, uma combinação válida; digo, com r elementos); ao sair do FOR colocamos os elementos da pilha em uma lista e depois novamente entre chaves (uma lista dentro da outra); em seguida colocamos na pilha a lista LC (inicialmente vazia) e colocamos (+) a lista com a combinação válida dentro da lista LC e armazenamos a lista "atualizada" na variável LC (nela própria).

*Em outras palavras: estamos contabilizando o saldo de nossa "pescaria".

Em seguida o último NEXT incrementa a linha (isto é, faz o processamento ir para uma nova linha da matriz) e tudo se repete como anteriormente.

Se não for este o caso (isto é, se o número de elementos iguais a 1, na linha pesquisada, *não é* igual a r), simplesmente limpamos (CLEAR) a pilha (esta linha não nos interessa);

9. Ao término o programa disponibiliza, na pilha, a lista com todas as combinações úteis.

Prova de que a matriz (MC) calcula combinações

Agora envidaremos esforços para provar que a matriz (MC) (pág. 60) presta-se ao cálculo de combinações.

Para a consecução do nosso intento iremos precisar de alguns resultados:

Lema 1. *Seja $A = \{a_1, a_2, \ldots, a_n\}$ um conjunto com n elementos, e seja $A' = A \cup \{a_{n+1}\}$. Então o número de subconjuntos de A' é o dobro do número de subconjuntos de A; e mais: seus subconjuntos são precisamente os mesmos de A juntamente com cada um destes unido com $\{a_{n+1}\}$.*

Prova: Temos pela transitividade da inclusão que todo subconjuto B de A o é de A' (isto é, $B \subset A, A \subset A' \Rightarrow B \subset A'$). Ainda: dado $B \subset A \Rightarrow B \cup \{a_{n+1}\} \subset A'$. Agora vamos mostrar que todo subconjunto de A' é da forma acima, isto é:

"Dado $D \subset A'$ então $D \subset A$ ou $D = B \cup \{a_{n+1}\}$, para algum $B \subset A$".

De fato, se $D = \emptyset$ é óbvio. Suponha que $\emptyset \neq D \subset A'$ e $D \not\subset A$ e $D \neq B \cup \{a_{n+1}\}, \forall B \subset A$. Então existe $x \in D$ tal que $x \in A'$ e $x \notin A$; logo só pode ser $x = a_{n+1}$. Sendo $D \neq B \cup \{a_{n+1}\}$ $(\forall B \subset A)$, temos duas possibilidades:

(i) existe $y \in D$ tal que $y \notin B \cup \{a_{n+1}\}, \forall B \subset A$. Absurdo, tome $B = A$.

(ii) $\forall B \subset A$ existe $z \in B \cup \{a_{n+1}\}$ tal que $z \notin D$. Absurdo, tome $B = \emptyset$.

■

Usando demonstração por indução sobre n, decorre trivialmente do lema anterior o seguinte:

Corolário 1. *Dado um conjunto com n elementos, o número de seus subconjuntos é 2^n.*

Prova: $n = 1$: $A = \{a_1\} \Rightarrow \mathcal{P}(A) = \{\emptyset, \{a_1\}\}$.

Admitamos a validade da proposição para $n = p$. Isto é, se $A = \{a_1, a_1, \ldots, a_p\} \Rightarrow \#\mathcal{P}(A) = 2^p$. Mostremos que a proposição ainda é verdadeira para $n = p + 1$. Isto é, se $A = \{a_1, a_1, \ldots, a_p, a_{p+1}\}$ implica que $\#\mathcal{P}(A) = 2^{p+1}$.
Mas isto é imediato pelo lema anterior. ■

Vamos agora apresentar uma importante propriedade da matriz MC:

Lema 2 (Propriedade do DNA)**.** *Seja $n \geq 2$ um natural arbitrariamente fixado e $j = 1, \ldots, n-1$. Sob estas condições é válida a seguinte identidade:*

$$a_{ij} = a_{(i+2^{n-1})j}$$

Prova:

$$a_{_{(i+2^{n-1})j}} = (-1)^{\left\lfloor \frac{(i+2^{n-1})-1}{2^{j-1}} \right\rfloor}$$

$$= (-1)^{\left\lfloor \frac{i-1}{2^{j-1}} + \frac{2^{n-1}}{2^{j-1}} \right\rfloor}$$

Sendo $2^{n-1}/2^{j-1} = 2^{n-j}$; e tendo em conta que o maior valor assumido por j − dentro das nossas hipóteses − é $n-1$; o menor valor neste expoente será $n-(n-1)=1$, do que podemos inferir tratar-se sempre de uma potência de dois. Portanto,

$$a_{_{(i+2^{n-1})j}} = (-1)^{\left\lfloor \frac{i-1}{2^{j-1}} \right\rfloor + \frac{2^{n-1}}{2^{j-1}}} = a_{ij}$$

■

Vamos concretizar a propriedade anterior: Observe

i:	1	2	···	2^{n-1}
$i+2^{n-1}$:	$2^{n-1}+1$	$2^{n-1}+2$	···	$2\cdot 2^{n-1}$

o que significa que a identidade $a_{ij} = a_{_{(i+2^{n-1})j}}$ nos assegura que haverá uma cópia da metade superior para a metade inferior da matriz binária (isto só até a coluna $n-1$, bem entendido).

Observe os exemplos abaixo, para $n=2$, $n=3$ e $n=4$:

1	1
-1	1
1	-1
-1	-1

1	1	1
-1	1	1
1	-1	1
-1	-1	1
1	1	-1
-1	1	-1
1	-1	-1
-1	-1	-1

1	1	1	1
-1	1	1	1
1	-1	1	1
-1	-1	1	1
1	1	-1	1
-1	1	-1	1
1	-1	-1	1
-1	-1	-1	1
1	1	1	-1
-1	1	1	-1
1	-1	1	-1
-1	-1	1	-1
1	1	-1	-1
-1	1	-1	-1
1	-1	-1	-1
-1	-1	-1	-1

Observe que se tomarmos $j=n$ na demonstração anterior obtemos $a_{ij} = -a_{_{(i+2^{n-1})j}}$; o que explica a última coluna nas matrizes acima.

Na verdade a identidade em questão nos conta mais que esta interpretação; mas, para o propósito que temos em mente, isto já é suficiente.

Teorema 2 (Gentil/1997)**.** *Dado um conjunto* $A = \{a_1, a_2, \ldots, a_n\}$ *com* n *elementos, a matriz abaixo*

$$a_{ij} = (-1)^{\left\lfloor \frac{i-1}{2^{j-1}} \right\rfloor}$$

nos fornece todos os seus subconjuntos, para $i = 1, 2, \ldots, 2^n$ *e* $j = 1, 2, \ldots, n$*; de acordo com a convenção feita anteriormente.*

Prova: Indução sobre o número n de elementos de A.

(i) $n = 1$ ($A = \{a_1\}$) $\Rightarrow$ $j = 1$ e $i = 1, 2$.

a_1	$\{a_i\}$
1	$\{a_1\}$
-1	$\emptyset$

$\Rightarrow \quad \mathcal{P}(A) = \{\emptyset, \{a_1\}\}$.

(ii) Suponhamos a validade da fórmula para $n = p$ elementos.
Por hipótese a matriz (a_{ij}) nos fornece os 2^p subconjuntos de $A = \{a_1, a_2, \ldots, a_p\}$.

(iii) Mostremos que a fórmula é válida para $n = p + 1$ elementos. Isto é, que a fórmula nos fornece todos os 2^{p+1} subconjuntos de $A' = \{a_1, a_2, \ldots, a_p, a_{p+1}\}$.

De fato, tendo em conta os dois lemas anteriores, é suficiente mostrar que

$$a_{i(p+1)} = \begin{cases} 1, & \text{se} \quad i = 1, 2, \ldots, 2^p; \\ -1, & \text{se} \quad i = 2^p + 1, \ldots, 2^{p+1}. \end{cases}$$

temos,

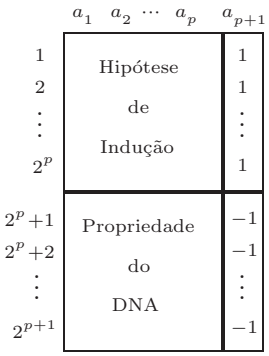
$$i = 1, 2, \ldots, 2^p \Leftrightarrow 0 \leq i - 1 < 2^p \Leftrightarrow 0 \leq \frac{i-1}{2^p} < 1 \Leftrightarrow \left\lfloor \frac{i-1}{2^p} \right\rfloor = 0$$
$$\Rightarrow a_{i(p+1)} = 1.$$

Por outro lado,

$$i = 2^p + 1, \ldots, 2^{p+1} \Leftrightarrow 2^p \leq i - 1 < 2^{p+1} \Leftrightarrow 1 \leq \frac{i-1}{2^p} < 2$$
$$\Leftrightarrow \left\lfloor \frac{i-1}{2^p} \right\rfloor = 1 \Rightarrow a_{i(p+1)} = -1.$$

■

Para melhor entendimento da demonstração anterior veja a figura ao lado:



Desenvolvimento Binário

Uma outra utilidade para a matriz de combinações é no desenvolvimento binário de um inteiro positivo n. Com efeito, se trocarmos -1 por 0, assim:

$$a_{ij} = \begin{cases} 1, & \text{se } \left\lfloor \frac{i-1}{2^{j-1}} \right\rfloor \text{ é par;} \\ 0, & \text{se } \left\lfloor \frac{i-1}{2^{j-1}} \right\rfloor \text{ é ímpar.} \end{cases} \tag{4.1}$$

Obtemos,

2^0	2^1	2^2	2^3	
1	1	1	1	$15=\mathbf{1}\cdot 2^0+\mathbf{1}\cdot 2^1+\mathbf{1}\cdot 2^2+\mathbf{1}\cdot 2^3$
0	1	1	1	$14=\mathbf{0}\cdot 2^0+\mathbf{1}\cdot 2^1+\mathbf{1}\cdot 2^2+\mathbf{1}\cdot 2^3$
1	0	1	1	$13=\mathbf{1}\cdot 2^0+\mathbf{0}\cdot 2^1+\mathbf{1}\cdot 2^2+\mathbf{1}\cdot 2^3$
0	0	1	1	$12=\mathbf{0}\cdot 2^0+\mathbf{0}\cdot 2^1+\mathbf{1}\cdot 2^2+\mathbf{1}\cdot 2^3$
1	1	0	1	$11=\mathbf{1}\cdot 2^0+\mathbf{1}\cdot 2^1+\mathbf{0}\cdot 2^2+\mathbf{1}\cdot 2^3$
0	1	0	1	$10=\mathbf{0}\cdot 2^0+\mathbf{1}\cdot 2^1+\mathbf{0}\cdot 2^2+\mathbf{1}\cdot 2^3$
1	0	0	1	$9=\mathbf{1}\cdot 2^0+\mathbf{0}\cdot 2^1+\mathbf{0}\cdot 2^2+\mathbf{1}\cdot 2^3$
0	0	0	1	$8=\mathbf{0}\cdot 2^0+\mathbf{0}\cdot 2^1+\mathbf{0}\cdot 2^2+\mathbf{1}\cdot 2^3$
1	1	1	0	$7=\mathbf{1}\cdot 2^0+\mathbf{1}\cdot 2^1+\mathbf{1}\cdot 2^2+\mathbf{0}\cdot 2^3$
0	1	1	0	$6=\mathbf{0}\cdot 2^0+\mathbf{1}\cdot 2^1+\mathbf{1}\cdot 2^2+\mathbf{0}\cdot 2^3$
1	0	1	0	$5=\mathbf{1}\cdot 2^0+\mathbf{0}\cdot 2^1+\mathbf{1}\cdot 2^2+\mathbf{0}\cdot 2^3$
0	0	1	0	$4=\mathbf{0}\cdot 2^0+\mathbf{0}\cdot 2^1+\mathbf{1}\cdot 2^2+\mathbf{0}\cdot 2^3$
1	1	0	0	$3=\mathbf{1}\cdot 2^0+\mathbf{1}\cdot 2^1+\mathbf{0}\cdot 2^2+\mathbf{0}\cdot 2^3$
0	1	0	0	$2=\mathbf{0}\cdot 2^0+\mathbf{1}\cdot 2^1+\mathbf{0}\cdot 2^2+\mathbf{0}\cdot 2^3$
1	0	0	0	$1=\mathbf{1}\cdot 2^0+\mathbf{0}\cdot 2^1+\mathbf{0}\cdot 2^2+\mathbf{0}\cdot 2^3$
0	0	0	0	$0=\mathbf{0}\cdot 2^0+\mathbf{0}\cdot 2^1+\mathbf{0}\cdot 2^2+\mathbf{0}\cdot 2^3$

Na matriz `MC` a contagem de linhas e colunas inicia-se em 1, na matriz acima (matriz binária ou `MB`) a contagem de linhas e colunas deve iniciar-se em* 0, o que significa que devemos fazer uma translação nos índices da matriz (4.1); tendo em conta esta observação e escrevendo a equação anterior em função do natural n, a ser desenvolvido, resulta:

*Podemos contar as linhas debaixo para cima iniciando com zero, não há problemas.

$$a_{nj} = \begin{cases} 1, & \text{se } \left\lfloor \frac{n}{2^j} \right\rfloor \text{ é par;} \\ 0, & \text{se } \left\lfloor \frac{n}{2^j} \right\rfloor \text{ é ímpar.} \end{cases} \tag{4.2}$$

Figura 4.1: Desenvolvimento binário de n

Esta equação nos fornece o j-ésimo bit do desenvolvimento binário de n.

Para programar esta equação precisamos da variação de j, e isto se consegue assim:

$$\frac{n}{2^j} \geq 1 \quad \Rightarrow \quad 2^j \leq n \quad \Rightarrow \quad j \leq \texttt{log}_2^n$$

Então, fazemos: $j = 0,\ 1,\ 2, \ldots, \left\lfloor \texttt{log}_2^n \right\rfloor$.

O programa a seguir recebe um número natural n e devolve n em base 2.

```
≪ → n
   ≪ 0    'FLOOR(LOG(n)/LOG(2))'    EVAL
          FOR j
              IF   'FP(FLOOR(n/2^j)/2)==0'
              THEN 1
              ELSE 0
              END
          NEXT DEPTH ROW→
   ≫
≫
```

Um Desafio!!!

Mostre que as três matrizes a seguir,

$$a_{nm} = (-1)^{\left\lfloor \frac{n-1}{2^{m-1}} \right\rfloor} \qquad a_{nm} = (-1)^{\binom{n-1}{2^{m-1}}} \qquad a_{nm} = (-1)^{\mu_{2^{m-1}}^{n-1}}$$

são iguais. Isto é, nos fornecem o mesmo resultado para naturais m e n arbitrariamente fixados.

No expoente da matriz do meio temos uma *combinação*. Por exemplo,

$$n = 5,\ m = 2 \quad \Rightarrow \quad \binom{n-1}{2^{m-1}} = \binom{5-1}{2^{2-1}} = \binom{4}{2} = 6$$

Para o símbolo no expoente da matriz da direita estamos fazendo uso da seguinte convenção:

$\mu_{2^{m-1}}^{n-1}$ é o bit (digito) de posição $m-1$ no desenvolvimento binário de $n-1$.

Estamos contando as posições da esquerda para a direita, iniciando em zero.

Por exemplo,

$$n = 5,\ m = 2 \quad \Rightarrow \quad (n-1)_2 = 4_2 = 0\,0\,1\,0 \quad \Rightarrow \quad \mu_{2^{2-1}}^{5-1} = \mu_{2^1}^4 = 0$$

Tabelas-verdade da lógica e eletrônica digital

A seguinte fórmula (matriz):

$$a_{ij} = \begin{cases} 1, & \text{se } \left\lfloor \frac{i-1}{2^{N-j}} \right\rfloor \text{ é par;} \\ 0, & \text{se } \left\lfloor \frac{i-1}{2^{N-j}} \right\rfloor \text{ é ímpar.} \end{cases} \tag{4.3}$$

Figura 4.2: Tabela-verdade da lógica

gera a tabela-verdade que comparece na Lógica. Nesta fórmula, $\lfloor x \rfloor$ = maior inteiro que não supera x (função piso) e N é o número de variáveis lógicas (ou o número de colunas da tabela-verdade). Para gerar a *tabela* fazemos $i = 1, 2, \ldots, 2^N$ e $j = 1, 2, \ldots, N$. Para $N = 3$ variáveis, por exemplo, geramos a seguinte tabela:

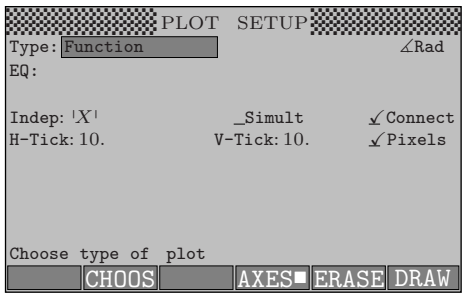
p	q	r
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0

Para obter a fórmula da tabela-verdade da eletrônica digital basta permutarmos 0 e 1 na equação (4.3). Estas fórmulas serão úteis para automatizar, através de *softwares*, a análise de sentenças lógicas. O programa dado a seguir gera a tabela-verdade da lógica, segundo a equação (4.3).

```
≪ → N
   ≪ 1 2 N ∧
      FOR I 1 N
         FOR J
           'FLOOR((I − 1)/2 ∧ (N − J))'      EVAL
            IF 2 MOD 0 ==
            THEN 1
            ELSE 0
            END
         NEXT N ROW→
      NEXT 2 N ∧ ROW→
   ≫
≫
```

4.2 Traçando gráficos

Aqui faremos apenas um breve resumo sobre plotagem de gráficos. Para acessar o ambiente de plotagem de gráficos na calculadora, usamos a sequência de teclas [↰] [2D/3D] ([F4 D]):



Se você estiver usando o modo RPN estas duas teclas devem ser pressionadas simultaneamente. Ou ainda: mantenha a tecla [↰] pressionada e, em seguida, pressione a tecla [2D/3D] ([F4 D]).

Configurações:

• A marca em _Simult significa que se você tem uma ou mais plotagens no mesmo gráfico, eles serão plotados simultaneamente ao produzir o gráfico.

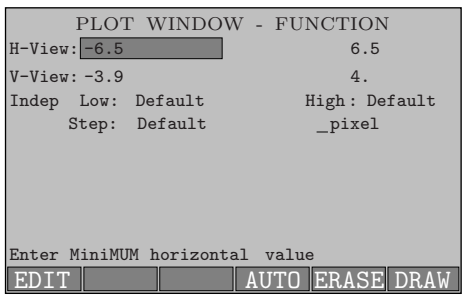
• A marca em _Connect significa que a curva será uma curva contínua em vez de um conjunto de pontos individuais.

• Uma marca em _Pixels significa que as marcas indicadas pelo H-Tick e V-Tick será separadas por estes diversos pixels.

• O valor padrão para ambos por H-Tick e V-Tick é 10.

A título de ilustração vamos plotar o gráfico da função $y = \sin x$. Ao entrar no ambiente gráfico (como dito acima), em Type (tipo de plotagem) devemos ter Function, como no visor acima, (os tipos de plotagem estão disponíveis na tecla virtual [CHOOS]). Inicialmente certifique-se que sua calculadora esteja operando no modo angular radiano (veja o campo superior à direita do visor); após "desça" com o botão ▽ para EQ ('EQ' significa uma varivel tipo equação; isto é, a variável que vai armazenar a função a ser plotada), abra plics e digite sin(x), em seguida pressione [ENTER].

Agora vamos *redimensionar a janela de plotagem*, isto é, vamos escolher, por exemplo, plotar esta função no intervalo $[0, 2\pi]$. Para isto pressione, simultaneamente, as teclas [↰] [WIN] ([F2 B]):

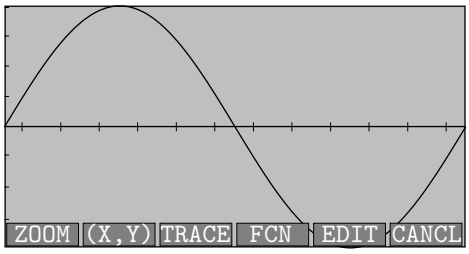


Para o valor mínimo de x digite 0 e dê ENTER. Para colocar 2π no extremo direito de H-View, pressione NXT em seguida a tecla virtual CALC. Delete o valor anterior (isto é, 6.5) coloque na pilha os valores 2 e π multiplique-os e pressione OK.

Para preencher os dois argumentos da faixa vertical, V-View, você tem duas opções: digita diretamente os valores, por exemplo,

$$\texttt{H-View}: -1.0 \qquad 1.0$$

ou digita a tecla virtual AUTO (automático). Finalmente, pressione as teclas virtuais: ERASE DRAW, para obter um gráfico similar ao seguinte:

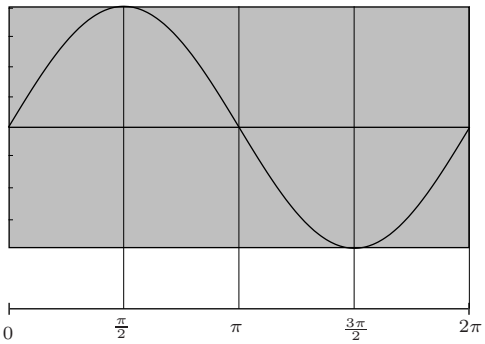


Nota: ERASE apaga algum eventual gráfico que tenha sido traçado anteriormente; enquanto DRAW traça o novo gráfico.

Para adicionar rótulos no gráfico pressione EDIT NXT LABEL.

Pressione MENU para remover as etiquetas do menu e obter uma visão total do gráfico. Pressione NXT para recuperar o primeiro menu gráfico atual.

Adendo: Algo que sempre me incomodou* em gráficos tais como o acima foi o "espaçamento" nas marcas dos eixos x e y. No caso deste gráfico eu gostaria que as marcas coincidissem com os pontos de máximo e mínimo da função, tipo:



Encontrei uma solução para este "problema", a qual desejo compartilhar com quem interessar possa. Inicialmente vejamos como redimensionar a tela de plotagem:

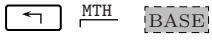
PDIM

A função PDIM toma como entrada um dos dois pares ordenados ($xmin$, $ymin$) ($xmax$, $ymax$) ou dois números inteiros binários $\#h$ e $\#v$. O efeito de PDIM é

*Desde os meus primeiros tempos de HP

redimensionar a variável gráfica PICT*. Quando o argumento for ($xmin$, $ymin$)-($xmax$, $ymax$), estes valores tornam-se a faixa das coordenadas definida pelo usuário no PPAR (variável que armazena parâmetros para plotagem). Quando o argumento for $\#h$ e $\#v$, as faixas das coordenadas definidas pelo usuário no PPAR se mantém inalteradas, porém o tamanho dos gráficos são alterados para $\#h \times \#v$ pixels.

Por exemplo, para redimensionar a tela gráfica para 131×80 pixels execute a seguinte sequência de teclas:



Em seguida,

131 R→ B 80 R→ B

Agora digite PDIM e pressione ENTER . Então, agora que já fixamos as dimensões de nossa tela de plotagem, lembramos que na janela PLOT SETUP, o valor `H-Tick:`10. significa que *entre* duas marcas consecutivas temos 10 pixels. Então, vamos aumentar o número de pixels entre marcas consecutivas, resolvendo para tanto a seguinte regra de três:

$$\begin{aligned} 2\pi &\longleftrightarrow 131 \\ \frac{\pi}{2} &\longleftrightarrow x \quad \Rightarrow \quad x = \frac{131}{4} = 32.75 \end{aligned}$$

Portanto, basta fazer: `H-Tick:`32.75

De igual modo podemos controlar as marcas no eixo y.

Plotando funções definidas por mais de uma sentença

Vamos plotar, por exemplo, o gráfico da função definida por:

$$f(x) = \begin{cases} -x, & \text{se } x \leq -1; \\ 1, & \text{se } -1 < x \leq 1; \\ x^2, & \text{se } x > 1. \end{cases}$$

Inicialmente convertemos a função em um programa, assim:

```
≪
    IF  'x ≤ -1'
    THEN  '-x'
    ELSE
        IF  'x > -1  AND  x ≤ -1'
        THEN  1
        ELSE  'x ^ 2'
        END
    END
≫
```

Armazene este programa na variável EQ: 'EQ'. Agora entre no ambiente gráfico, observe que o programa a ser plotado já se encontra nesta janela. Proceda como anteriormente; se necessário redimensione a janela de plotagem.

*variável chamada PICT armazena o conteúdo atual da janela dos gráficos.

Plotagem de Superfícies: Plotagens 'Fast 3D':

Plotagens `Fast 3D` são usadas para visualizar superfícies tridimensionais representadas por equações da forma $z = f(x, y)$. Por exemplo, se quiser visualizar $z = f(x, y) = x^2 + y^2$ podemos usar o seguinte:

• Pressione [←] [2D/3D] simultaneamente se estiver no modo RPN para acessar a janela PLOT SETUP.

• Altere TYPE para `Fast3D`. Isto pode ser feito com o auxílio das teclas [CHOOS] e ▽.

• Pressione ▽ e digite '$x \wedge 2 + y \wedge 2$' [OK] .

• Certifique-se de que 'X' seja selecionado como `Indep`: e 'Y' como `Depnd`:.

• Pressione [NXT] [OK] para retornar ao visor normal da calculadora.

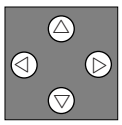
• Pressione [←] [WIN] simultaneamente se estiver no modo RPN para acessar a janela `PLOT WINDOW`.

• Mantenha as faixas da janela de plotagem padrão para ler:

X-Left:-1, XRight: 1, Y-Near:-1, Y-Far: 1, Z-Low: -1, Z-High: 1, Step Indep: 10, Depnd: 8

Nota: A etapa `Indep`: e `Depnd`: os valores representam o número de linhas de grade usadas na plotagem. Quanto maior estes números mais lento o gráfico é produzido, embora o tempo utilizado para a geração de gráfico seja relativamente rápido. No presente manteremos os valores padrões de 10 e 8 para data de etapa.

• Pressione [ERASE] [DRAW] para desenhar a superfície tridimensional. O resultado é uma imagem aramada da superfície com o sistema de coordenada de referência mostrado no canto esquerdo inferior do visor. Ao usar as teclas com seta (botões prateados),



você pode alterar a orientação da superfície. A orientação do sistema de coordenada de referência será alterada de acordo. Tente alterar a orientação da superfície sozinho.

Ao terminar, pressione [EXIT] .

• Pressione [CANCL] para retornar ao ambiente `PLOT WINDOW`.

• Altere para ler: `Step Indep`: 20 `Depnd`: 16

• Pressione [ERASE] [DRAW] para ver a plotagem de superfície.

Vamos agora plotar o gráfico da superfície dada por $F(x, y) = (y^2 + y)\cos x$ (pág. 52). Então:

• Pressione [←] [2D/3D] simultaneamente se estiver no modo RPN para acessar a janela PLOT SETUP.

• Altere, se necessário, TYPE para `Fast3D`.

• Pressione ▽ e digite '$(y \wedge 2 + y) * \cos(x)$' [OK] .

• Certifique-se de que 'X' seja selecionado como `Indep`: e 'Y' como `Depnd`:.

• Pressione NXT L OK para retornar ao visor normal da calculadora.

• Pressione ← WIN simultaneamente se estiver no modo RPN para acessar a janela `PLOT WINDOW`.

• Mantenha as faixas da janela de plotagem padrão para ler:

X-Left:-1, XRight: 1, Y-Near:-1, Y-Far: 1, Z-Low: -1, Z-High: 1, Step Indep: 10, Depnd: 8

Vamos plotar esta função no seguinte domínio:

$$\begin{cases} 0 \leq x \leq \pi/2 \\ 0 \leq y \leq 0,4 \end{cases}$$

Faça: X-Left:0, XRight: $\pi/2$. Para colocar $\pi/2$ em XRight pressione NXT L em seguida a tecla virtual CALC. Delete o valor anterior, coloque na pilha os valores π e 2 divida e pressione OK. Quanto à variável Y, coloque: Y-Near:0, Y-Far: 0.4. Quanto à variável Z, faça: Z-Low: -0.25, Z-High: 0.8.

Observe que na matriz da pág. 53 (ver também gráfico na pág. seguinte) o maior valor de $Z = F(x_i, y_j) = 0.5600$, daí a razão desta escolha para Z-High.

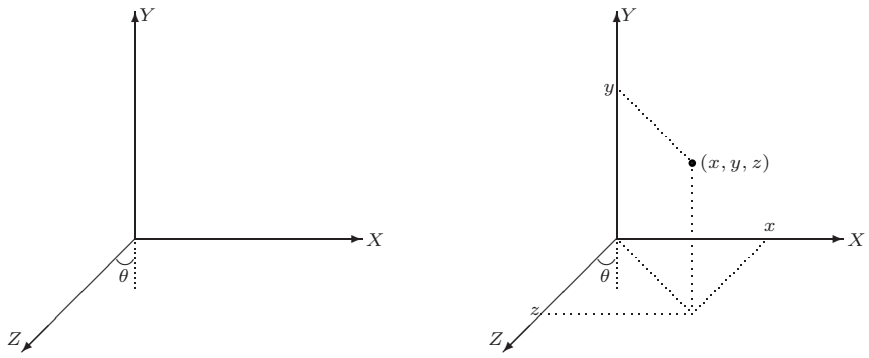
Finalmente, pressione as teclas virtuais: ERASE DRAW.

4.2.1 Plotando pontos no espaço $\mathbb{R}^3$

Em várias situações necessitamos de plotar um ponto no espaço $\mathbb{R}^3$, desenvolveremos a seguir um algoritmo com esta finalidade e mostraremos algumas aplicações do mesmo; o referido algoritmo será útil também na construção de novos comando LaTeX(macros).

Dedução do algoritmo

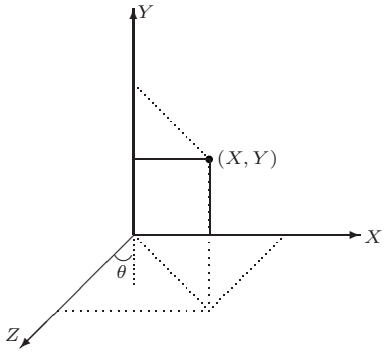
Pois bem, temos o seguinte desafio a resolver: Queremos plotar no espaço tridimensional o ponto de coordenadas (x, y, z), mas só dispomos de uma superfície bidimensional (a tela do computador ou uma folha de papel, por exemplo), como proceder? Começando com o gráfico à esquerda

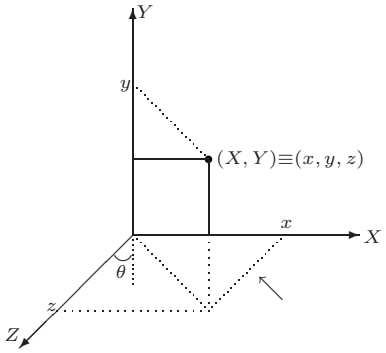


plotamos em seguida − gráfico à direita − o ponto (x, y, z) no plano do papel.

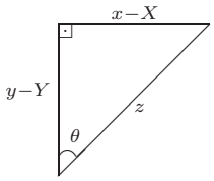
Usaremos do seguinte artifício para "iludir" o computador: ao invés de plotar

o ponto de coordenadas $(x,\, y,\, z)$ - como seria do nosso dever - vamos plotar o ponto de coordenadas $(X,\, Y)$ mostrado na figura seguinte à esquerda





O nosso interesse estará centrado na figura da direita. Desta figura destacamos o seguinte triângulo (ver seta):



$$\operatorname{sen}\theta = \tfrac{x-X}{z} \;\Rightarrow\; X = x - z\cdot\operatorname{sen}\theta$$

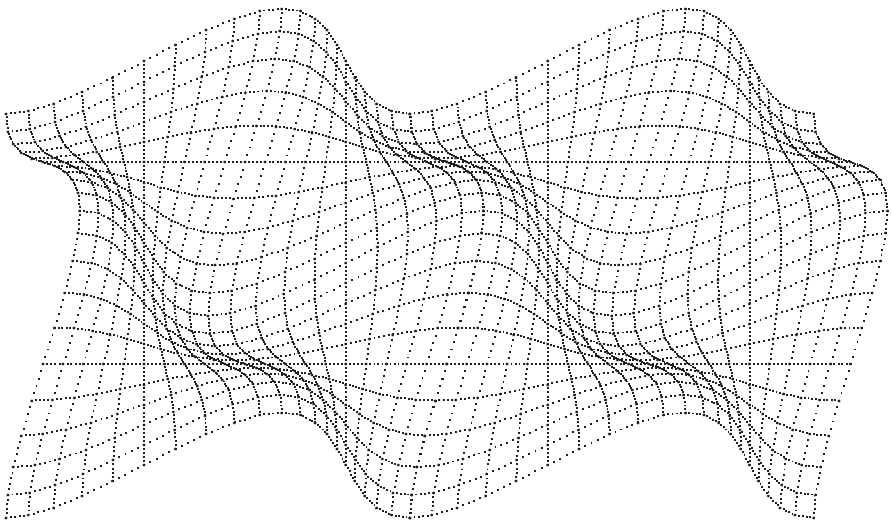
$$\cos\theta = \tfrac{y-Y}{z} \;\Rightarrow\; Y = y - z\cdot\cos\theta$$

Então, o "menor algoritmo do mundo" para o traçado de superfícies, é:

$$\boxed{(x,\, y,\, z) \equiv (X,\, Y) = (\, x - z\cdot\operatorname{sen}\theta,\, y - z\cdot\cos\theta\,)}$$

Aplicações do algoritmo

Na figura seguinte temos o gráfico da superfície dada por $z(x,\, y) = \cos x\cdot\cos y$ no domínio $[0,\, 4\pi]\times[0,\, 2\pi]$ (isto é, $0 \le x \le 4\pi$, $0 \le y \le 2\pi$), com $\theta = 35^{\circ}$:

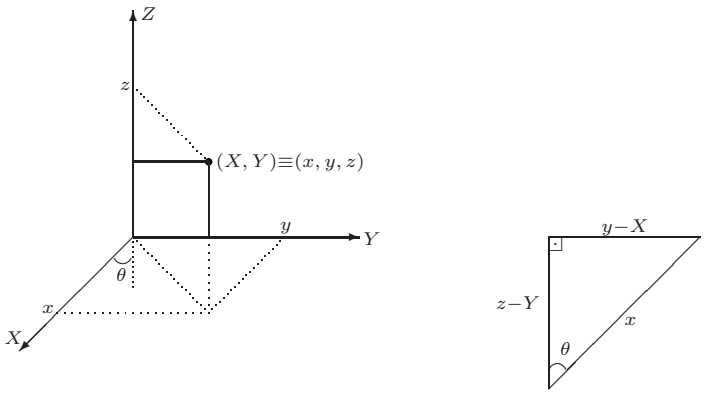


Esta superfície foi plotada no ambiente *picture* do LaTeX. Programamos nosso algoritmo (fórmula) em HP para nos fornecer as coordenadas dos pontos.

"...O matemático, como o pintor ou o poeta, é um desenhista. Se os seus desenhos são mais duradouros que os deles, é porque são feitos com idéias." (G.H. Hardy)

Uma nova configuração dos eixos

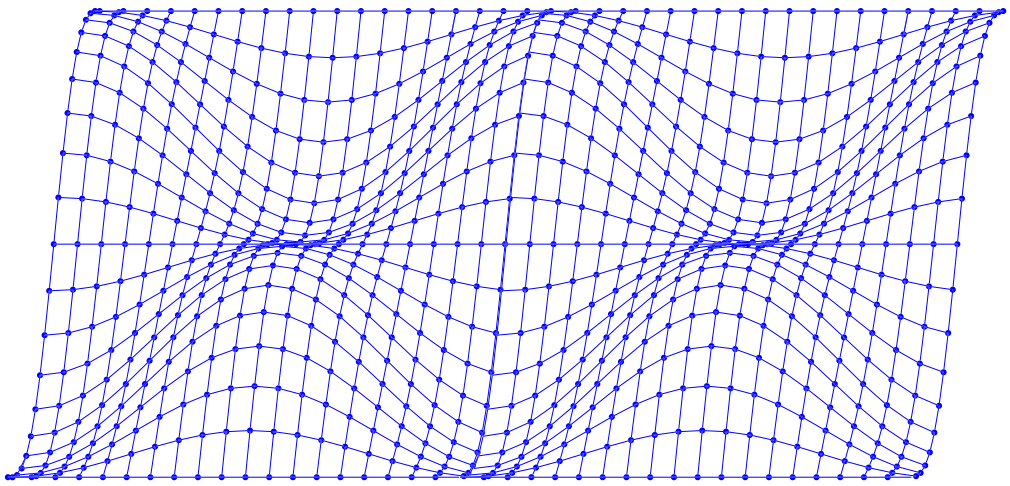
Para a seguinte configuração dos eixos coordenados,



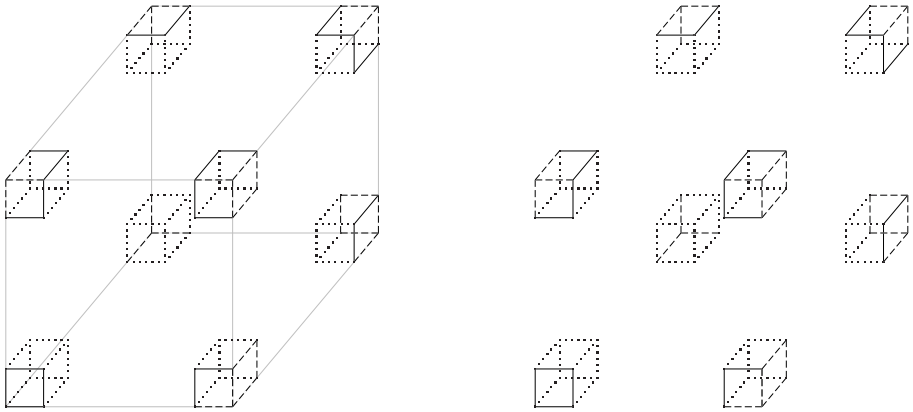
obtemos o algoritmo,

$$\boxed{(x,\, y,\, z) \equiv (X,\, Y) = (\, y - x \cdot \operatorname{sen}\theta,\, z - x \cdot \cos\theta\,)}$$

Na figura seguinte temos o gráfico da superfície dada por $z(x,\, y) = \operatorname{sen} x \cdot \cos y$, com $\theta = 35^{\circ}$ e no domínio $0 \leq x \leq 2\pi$, $0 \leq y \leq 4\pi$, para esta configuração.



Certa feita nos deparamos com a necessidade de esboçar a seguinte figura:

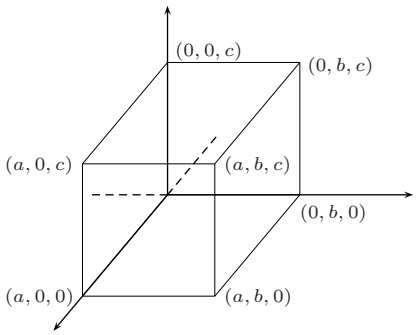


Para traçar esta figura (no ambiente picture do látex) programamos o algoritmo anterior em uma `HP` para nos fornecer os dados.*

Como mais um exemplo, o gráfico da pág. 53 foi plotado com auxílio deste algoritmo.

Paralelepípedos

Agora vamos construir um comando para desenhar um *paralelepípedo* como o da figura a seguir:



Inicialmente vamos calcular as coordenadas do algoritmo para cada um dos vértices da figura, assim:

$$\begin{cases} X = y - x \operatorname{sen} \theta \\ Y = z - x \cos \theta \end{cases}$$

*Para ver aplicações deste algoritmo no LaTeX veja nosso artigo [9].

então,

$$(a,\,0,\,0):\begin{cases}X=0-a\cdot\,\mathrm{sen}\,\theta\\ Y=0-a\cdot\cos\theta\end{cases}\qquad (a,\,0,\,c):\begin{cases}X=0-a\cdot\,\mathrm{sen}\,\theta\\ Y=c-a\cdot\cos\theta\end{cases}$$

$$(a,\,b,\,0):\begin{cases}X=b-a\cdot\,\mathrm{sen}\,\theta\\ Y=0-a\cdot\cos\theta\end{cases}\qquad (a,\,b,\,c):\begin{cases}X=b-a\cdot\,\mathrm{sen}\,\theta\\ Y=c-a\cdot\cos\theta\end{cases}$$

$$(0,\,b,\,0):\begin{cases}X=b-0\cdot\,\mathrm{sen}\,\theta\\ Y=0-0\cdot\cos\theta\end{cases}\qquad (0,\,b,\,c):\begin{cases}X=b-0\cdot\,\mathrm{sen}\,\theta\\ Y=c-0\cdot\cos\theta\end{cases}$$

$$(0,\,0,\,0):\begin{cases}X=0-0\cdot\,\mathrm{sen}\,\theta\\ Y=0-0\cdot\cos\theta\end{cases}\qquad (0,\,0,\,c):\begin{cases}X=0-0\cdot\,\mathrm{sen}\,\theta\\ Y=c-0\cdot\cos\theta\end{cases}$$

à esquerda temos a codificação dos vértices inferiores (do "chão") e à direita a codificação dos vértices superiores. Inicialmente faremos uma subrotina para gerar estes oito vértices, assim:

```
≪ → a b c
  ≪ RAD '40*π/180' →NUM 'θ' STO
    '0-a*sin(θ)' EVAL
    '0-a*cos(θ)' EVAL R→C 'V1' STO
    'b-a*sin(θ)' EVAL
    '0-a*cos(θ)' EVAL R→C 'V2' STO
    'b-0*sin(θ)' EVAL
    '0-0*cos(θ)' EVAL R→C 'V3' STO
    '0-0*sin(θ)' EVAL
    '0-0*cos(θ)' EVAL R→C 'V4' STO
    '0-a*sin(θ)' EVAL
    'c-a*cos(θ)' EVAL R→C 'V5' STO
    'b-a*sin(θ)' EVAL
    'c-a*cos(θ)' EVAL R→C 'V6' STO
    'b-0*sin(θ)' EVAL
    'c-0*cos(θ)' EVAL R→C 'V7' STO
    '0-0*sin(θ)' EVAL
    'c-0*cos(θ)' EVAL R→C 'V8' STO
  ≫
≫
```

Observe, na segunda linha, que estamos fixando θ, em radianos, equivalente a 40^{o}. Os zeros, acima, foram digitados apenas por uma questão didática e estética.

O comando $R\to C$, acessado com ← MTH NXT L CMPLX $R\to C$, converte um par de números reais em um par ordenado. Armazene esta subrotina com o nome de `VPLD`.

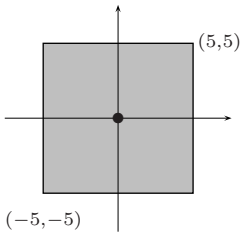
O programa que gera o paralelepípedo é como a seguir:

```
≪ → a b c
   ≪  a b c VPLD ERASE
      -5 5 YRNG
      '−5 ∗ 131/80' EVAL '5 ∗ 131/80' EVAL XRNG
        V1 V2 LINE V2 V3 LINE
        V3 V4 LINE V4 V1 LINE
        V5 V6 LINE V6 V7 LINE
        V7 V8 LINE V8 V5 LINE
        V1 V5 LINE V2 V6 LINE
        V3 V7 LINE V4 V8 LINE
        { } PVIEW
   ≫
≫
```

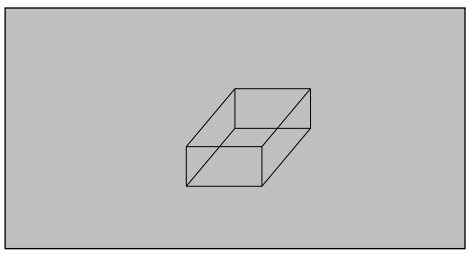
Na primeira linha o programa coloca na pilha as arestas do paralelepípedo e chama a subrotina VPLD que calcula os seus vértices. Na segunda e terceira linhas escolhemos o range para os eixos y e x, na janela `PLOT WINDOW`. A princípio escolhemos:

$$H - View: \quad -5 \quad 5$$
$$V - View: \quad -5 \quad 5$$

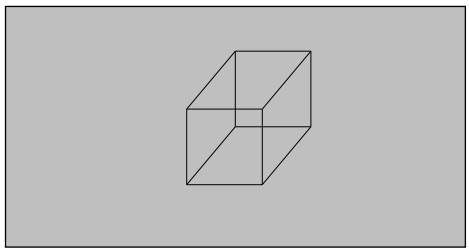
Ou seja, a seguinte janela de plotagem:



Nota: Observe que o vértice (0, 0, 0) do nosso paralelepípedo estará no centro deste quadrado. Pois bem, com este range ao plotarmos um cubo de arestas $a = b = c = 3$, obtemos a seguinte figura (esquerda):



esta "distorção" se deve a que, nossa tela de plotagem, tem a dimensão 131×80 pixels, por esta razão ao fazermos, no range do eixo x, a correção que comparece no programa, ao voltar a traçar o cubo $a = b = c = 3$, obtemos a figura:

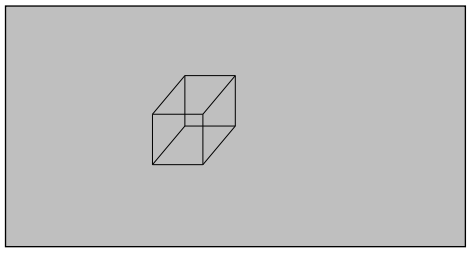


mais condizente com a "realidade". Quanto à última instrução do programa vejamos a sua função:

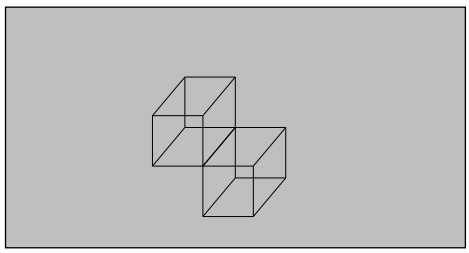
PVIEW

Este comando toma como entrada as coordenadas de um ponto como coordenadas do usuário (x, y) ou pixels $\{\#n, \#m\}$ e coloca o conteúdo de PICT com o canto esquerdo superior no local do ponto especificado. Você pode usar também uma lista vazia como argumento quando a imagem for **centrada** no visor.

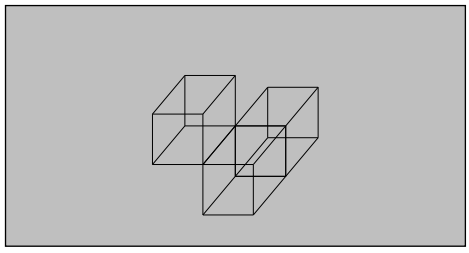
Na figura da pág. 75, que deu origem ao nosso programa, observamos que as "arestas" do paralelepípedo na verdade são *coordenadas* por isto podem assumir valores negativos como, por exemplo $a = 2$, $b = -2$ e $c = 2$, obtendo:



Retirando o ERASE do programa principal e executando o programa para $a = 2$, $b = 2$ e $c = -2$, "superpomos" este novo paralelepípedo ao anterior, assim:



Para $a = -2$, $b = 2$ e $c = -2$, obtemos:

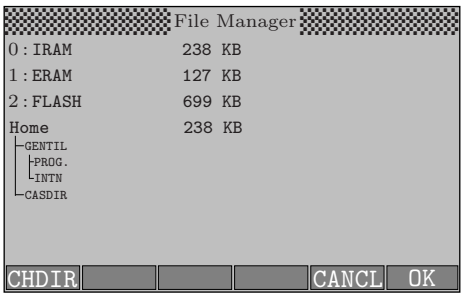


4.3 HOME: Variáveis e Diretórios

Organizar dados na calculadora

Você pode organizar dados na sua calculadora armazenando as variáveis em uma árvore de diretório. Para compreender a memória da calculadora, observe primeiro o diretório de arquivo.

Pressione a combinação de teclas [←] FILES (a função FILE ativa o navegador de arquivo na memória da calculadora) para obter o visor do gerenciador de arquivo da calculadora:



Atenção: Este é o visor da configuração atual da minha calculadora.

Vamos abrir um parentesis aqui para explicar a configuração de memória da $HP-50g$ e depois retornamos para os diretórios.

Estrutura da memória

A calculadora contém um total de 2.5 MB de memória do qual 1 MB é usada para armazenar o sistema operacional (memória do sistema) e 1.5 MB é usado para a operação da calculadora e armazenagem de dados (memória do usuário). Os usuários não têm acesso ao componente da memória do sistema.

O visor acima indica a existência de três portas de memória (ou partições da memória). As portas da memória disponíveis são:

- Porta 0, chamada IRAM
- Porta 1, chamada IRAM
- Porta 2, chamada FLASH

A seção HOME da memória funciona como um disco num computador pessoal. Cada objeto com nome em HOME é semelhante a um arquivo num disco de computador. O `diretório principal` (ou raiz) para a $HP-50g$ é chamado HOME.

A porta 0 e o diretório HOME compartilham a mesma área da memória, então quanto mais dados armazenados no diretório HOME, por exemplo, menos memória estará disponível para a armazenagem na Porta 0. O tamanho total da memória para a área da memória do diretório Porta 0/HOME é 241 KB.

Porta 1 (ERAM) pode conter até 128 KB de dados. Porta 1, juntamente com a Porta 0 e o diretrio HOME, constitui o segmento RAM da calculadora (memória de acesso aleatório) da memória da calculadora.

A Porta 2 pertence ao segmento Flash ROM da calculadora (memória apenas de leitura). A Porta 2 pode armazenar até 1085 KB de dados.

A quarta linha e as linhas subsequentes no visor acima mostram a árvore do diretório da calculadora. O diretório superior, como já ressaltamos, é o diretório Home e tem pré-definido em seu interior um sub-diretório chamado CASDIR. O visor File Manager possui três funõeses associadas às teclas do menu virtual:

CHDIR: Seleciona o diretório (CHange DIRectory).
CANCL: Cancela a ação.
OK: Aprova a seleção.

O subdiretório CASDIR contém um número de variáveis necessárias para a operação adequada do CAS (sistema algébrico do computador).

Por exemplo, para obsevar os arquivos (variáveis) dentro do (sub)diretório CASDIR, desça com a seta para baixo, ▽ , até destacá-lo; agora pressione a tecla virtual OK . Vamos aproveitar para explicar a função dos menus desta nova tela*:

Funções para manipular variáveis (arquivos)

Este visor inclui 20 comandos associados às teclas do menu soft que podem ser usados para criar, editar e manipular variáveis. As primeiras seis funções são as seguintes:

EDIT : Para editar uma variável ressaltada.
COPY: Para copiar uma varivel ressaltada.
MOVE: Para mover uma variável ressaltada.
RCL : Para retornar o conteúdo de uma variável ressaltada (coloca no nível 1 da pilha).
EVAL: Para avaliar (executar) uma varivel ressaltada.
TREE: Para ver a árvore do diretório onde a variável está contida.

Se você pressionar a tecla NXT L, o próximo conjunto de funções fica disponível:

PURGE: Para excluir ou apagar uma variável.
RENAM: Para renomear uma variável (Arquivo, "pasta", etc.).
NEW : Para criar uma nova variável (ou "Pasta").
ORDER: Para ordenar um conjunto de variáveis no diretório.
SEND : Para enviar uma variável para outra calculadora ou computador.
RECV : Para receber uma variável de uma outra calculadora ou computador.

Se você pressionar a tecla NXT L, o terceiro conjunto de funções fica disponível:

HALT : Para retornar para a pilha temporariamente.
VIEW : Para ver o conteúdo de uma variável.
EDITB: Para editar o conteúdo de uma variável binária (similar a EDIT)
HEADE: Para mostrar o diretório contendo a variável no cabeçalho.
LIST : Fornece uma lista de nomes e descrição de variáveis.
SORT : Seleciona as variáveis de acordo com um critério de seleção.

Se você pressionar a tecla NXT L, o último conjunto de funções fica disponível:

XSEND: Para enviar a variável com o protocolo XModem.
CHDIR: Para alterar o diretório.

Criando diretórios, movendo, copiando e renomeando Arquivos

Como não tenho a menor idéia da configuração da árvore de diretórios da sua calculadora, para ilustrar a utilização de algumas das funções descritas anteriormente vamos criar, inicialmente, dois (sub)diretórios dentro do diretório HOME. Pois bem, para começar do zero retorne, se necessário, à tela principal de sua

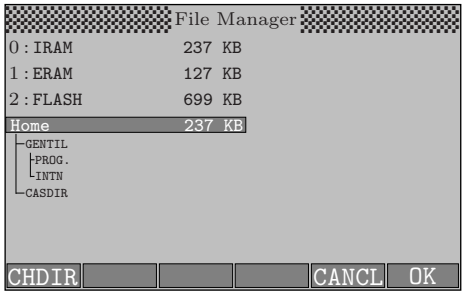
*Tenha paciência, daqui a pouco estaremos ensinando a criar diretórios, copiar arquivos, renomear arquivos, etc.

calculadora (para a "pilha"). Então:

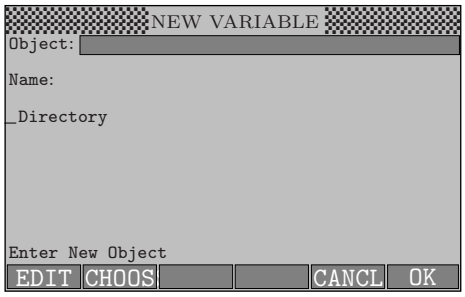
1º) Se você está na pilha observe na parte superior do visor (segunda linha) que existe um par de chaves: { HOME ... } isto indica qual o (sub)diretório atual da sua calculadora. Vamos unificar a minha e a sua calculadora no diretório raiz (HOME) para isto pressione [←] UPDIR possivelmente mais de uma vez até que tenhamos entre chaves apenas o diretório HOME: { HOME }.

2º) Vamos entrar na árvore de diretórios: pressione: [←] FILES ;

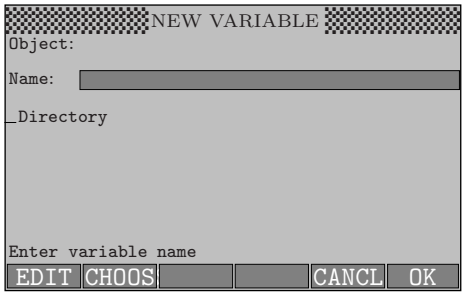
3º) na sua calculadora deverá aparecer um visor **similar** ao seguinte:



Agora pressione a tecla virtual OK para adentrarmos neste diretório. Digite NXT L NEW . Isto produzirá a seguinte forma de entrada:



O campo de entrada Object , o primeiro campo de entrada no formulário, é ressaltado por padrão. Este campo de entrada pode manter o conteúdo de uma nova variável que está sendo criada. Dado que não temos neste ponto nenhum conteúdo para o novo subdiretório, simplesmente pulamos este campo de entrada pressionando a tecla com a seta para baixo, ▽ , uma vez. O campo de entrada Name é agora ressaltado.

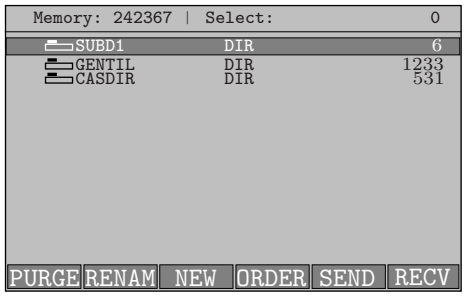


Neste local inserimos o nome do novo subdiretório (ou variável, de acordo

com o caso), conforme a seguir: ALPHA ALPHA SUBD1 ENTER

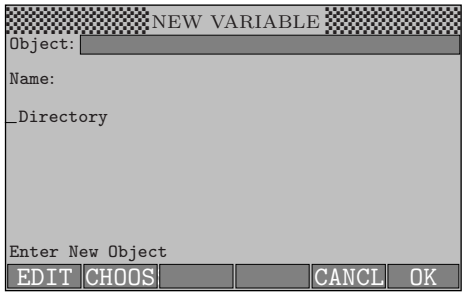
SUBD1 (subdiretório1) foi o nome que escolhemos para o nosso primeiro subdiretório.

Após a última operação acima o cursor se move para o campo de seleção _Directory. Pressione a tecla tecla virtual ✓CHK para especificar que você está criando um diretório e pressione OK para sair do formulário de entrada. As variáveis relacionadas para o diretório HOME serão mostradas no visor conforme a seguir:



Isto na minha calculadora, na sua certamente deve diferir. Mas o que importa é que o subdiretório que acabamos de criar comparece em destaque no visor acima.

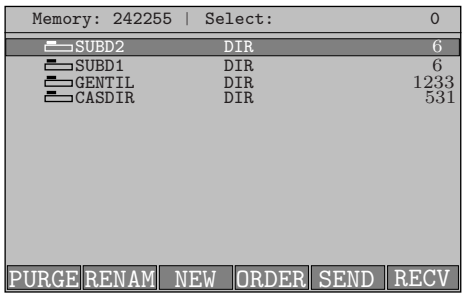
Para o propósito que temos em mente, vamos criar ainda um outro subdiretório dentro do diretório HOME. Para isto pressione a tecla virtual NEW ; estamos, novamente, na tela:



proceda como antes e crie o novo subdiretório SUBD2. Isto é, pressione a tecla com a seta para baixo, ▽, uma vez. O campo de entrada Name é agora ressaltado.

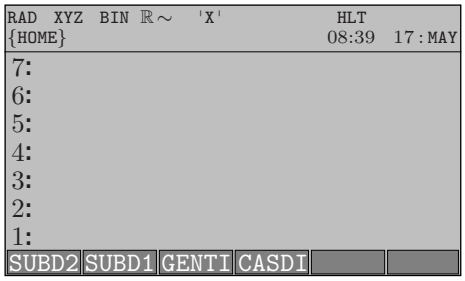
Neste local insira o nome do novo subdiretório, conforme a seguir: ALPHA ALPHA SUBD2 ENTER.

Após a última operação acima o cursor se move para o campo de seleção _Directory. Pressione a tecla tecla virtual ✓CHK e pressione OK para sair do formulário de entrada. As variáveis relacionadas para o diretório HOME serão novamente mostradas no visor, no caso da minha calculadora assim:



O nosso objetivo (itinerário) agora será o seguinte: criaremos uma variável (arquivo) para guardar no subdiretório SUBD1 e, depois, *copiaremos* este arquivo no subdiretório SUBD2 e, em seguida, vamos renomeá-lo.

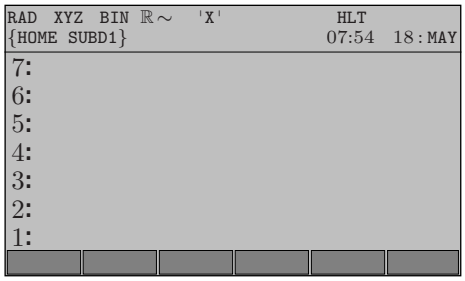
Pois bem, agora pressione ON para retornar à pilha. No menu ("embaixo") é mostrado o conteúdo do diretório HOME; no caso da minha calculadora, assim:



A variável que iremos guardar no subdiretório SUBD1 é o nosso primeiro programa (pág. 11), o qual repetimos aqui,

$$\ll \rightarrow \text{a b } \text{'ABS(a-b)'} \gg \tag{4.4}$$

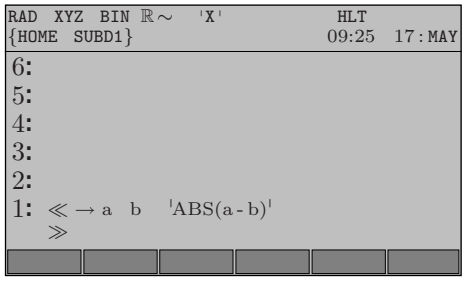
Antes de digitar este programa selecione, na tela acima, o subdiretório no qual vamos armazená-lo; isto é, pressione a tecla virtual SUBD1. O visor de sua calculadora deverá mostrar-se similar ao seguinte:



Observe, na parte superior do visor (segunda linha) que estamos no subdiretório SUBD1 o qual, por sua vez, encontra-se dentro do diretório HOME. Observe, ademais, que os menus embaixo encontram-se todos vazios, isto se deve a que não temos nenhuma variável (arquivo) dentro deste subdiretório.

Pois bem, pedimos ao leitor que digite o programa (4.4) e o coloque (ENTER) na

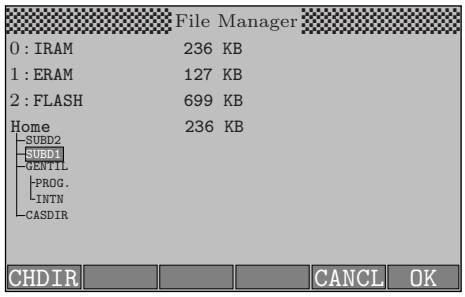
pilha. Após esta operação, no caso da minha calculadora, o visor apresenta-se assim:



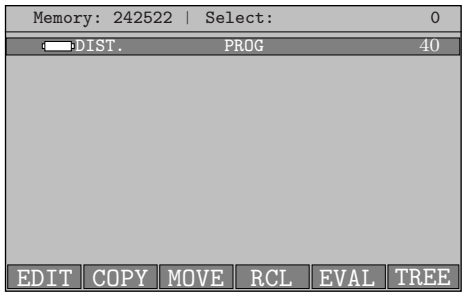
Pois bem, vamos armazenar este programa em uma variável, digamos: 'DIST. '

Agora pressione ENTER para pôr o nome na pilha e, em seguida, STO K para armazenar o programa com este nome. Observe que o programa (digo, seu nome) aparece no primeiro menu, embaixo. Nunca é demais ressaltar: Isto significa que esta variável encontra-se no subdiretório SUBD1, o qual encontra-se dentro do diretório HOME.

Então, dando continuidade ao nosso escopo inicial, vamos `copiar` agora este arquivo (variável na HP) para o sudiretório SUBD2. Para isto pressione ← FILES para adentrarmos na árvore de diretórios:



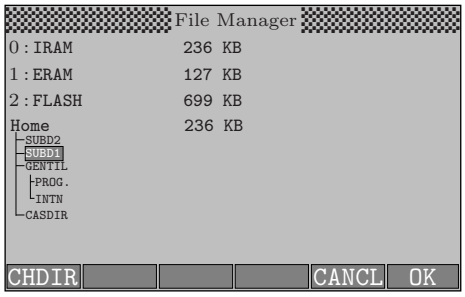
Observe que o subdiretório SUBD1 (diretório corrente) já aparece em destaque. Pressione a tecla virtual OK para adentrarmos nesta pasta; digo, neste subdiretório:



O visor indica que existe uma nova variável dentro do subdiretório SUBD1.

A faixa em destaque neste visor nos diz que a variável DIST. é do tipo programa e que ocupa 40 bytes de memória. Agora pressione COPY para obter uma cópia deste arquivo. A calculadora responderá com um visor denominado PICK DESTINATION:

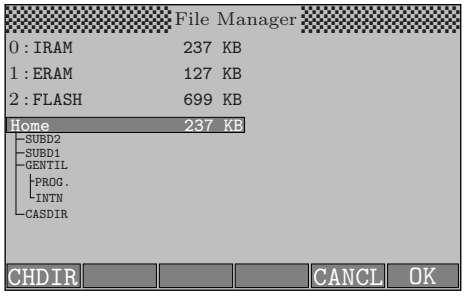
Use a tecla com a seta para cima △ para selecionar o subdiretório SUBD2 e pressione OK . Após isto o visor volta para a tela anterior, isto é, para o subdiretório SUBD1. Para confirmar pressione a tecla virtual TREE entrando na árvore de diretórios:



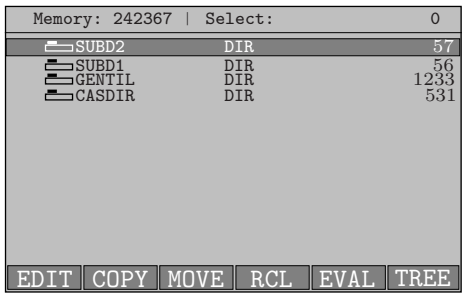
Vamos agora renomear a variável que acabamos de transferir para o SUBD2. Para isto use a tecla com a seta para cima △ para selecionar o subdiretório SUBD2 e pressione OK . Agora pressione NXT L RENAM. Na nova tela que se apresenta digite um novo nome para a variável DIST. e, após dê ENTER. Pressione ON para retornar à pilha.

Excluir subdiretórios

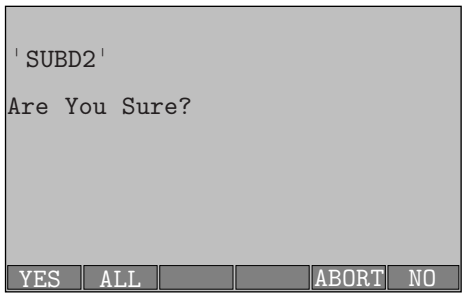
Vamos agora excluir os dois subdiretórios criados anteriormente, para tanto: pressione a tecla ↰ FILES para ativar o menu FILES. Selecione (destaque), com o auxílio da tecla △, o diretório HOME (que contém os subdiretório que desejamos excluir), assim:



e pressione OK para adentrarmos neste diretório. Um visor similar ao seguinte será apresentado:



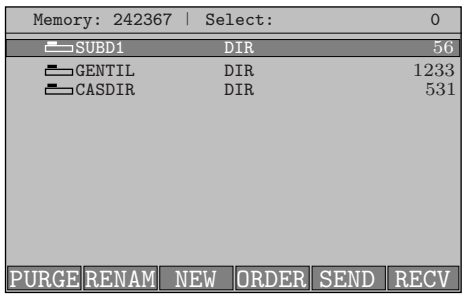
Como o subdiretório SUBD2 já está assinalado pressione NXT L PURGE. O seguinte visor será apresentado:



O segmento 'SUBD2' neste formulário é o nome do subdiretório que está sendo excluído. As teclas do menu virtual fornecem as seguintes opções*:

YES : Continue a excluir o subdiretório (ou variável).
ALL : Continue a excluir todos os subdiretórios (ou variáveis).
ABORT: Não exclua o subdiretório (ou variável) da lista.
NO : Não exclua o subdiretório (ou variável).

Pressione YES ; o visor agora se apresenta assim:



sem o subdiretório SUBD2. Com o mesmo procedimento anterior excluimos o subdiretório SUBD1.

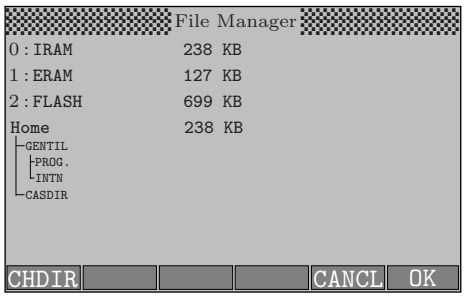
Após excluir o subdiretório SUBD1 pressione a tecla ← UPDIR para retornar à (nova) arvore de diretórios (ao diretório HOME) agora sem os subdiretórios excluídos.

*Sure: certo, seguro, firme.

. Transmitindo Dados Entre Duas $HP-50g$

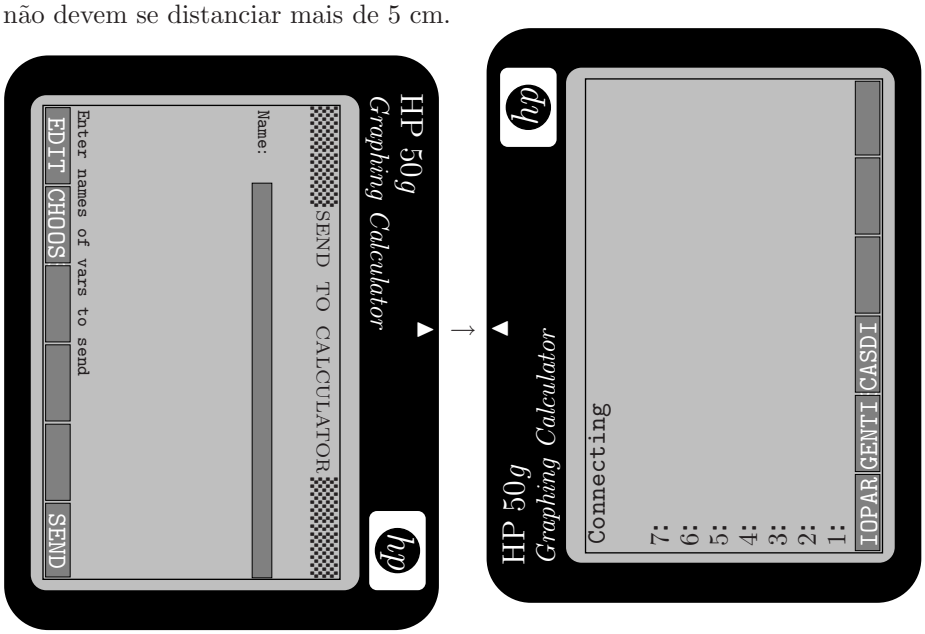
Para transferir objetos de uma $HP-50g$ para outra:

Se o objeto (um programa, por exemplo) que você deseja transferir não encontra-se no diretório corrente* então você deve *selecionar* o diretório onde este objeto encontra-se; ou seja, pressione [←] FILES para chegar na árvore de diretórios; isto é, a um visor similar ao seguinte:



Em seguida com a tecla ▲ ou ▼ selecione o diretório no qual o objeto, a ser transferido, encontra-se. Após selecionar o diretório pressione CHDIR para que este seja o diretório corrente. Talvez você queira também selecionar, na calculadora que irá receber os dados, o diretório que irá recebê-los. Então,

1. Alinhe as portas infravermelhas através das marcas △. As calculadoras não devem se distanciar mais de 5 cm.

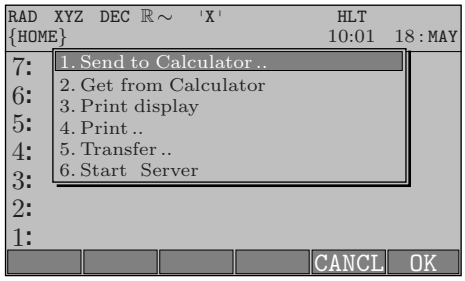


Na figura acima a calculadora da esquerda irá transmitir e a da direita irá receber os dados.

*Para se certificar disto pressione a tecla VAR J e pesquise nos menus "embaixo".

2. **A máquina que enviará a informação**.

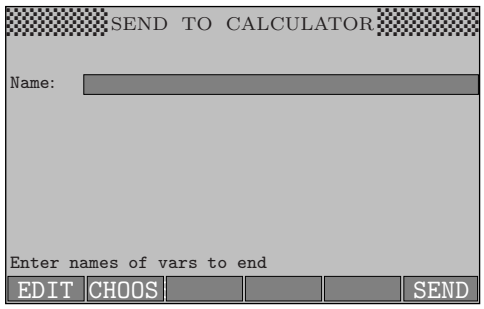
2.1. Pressione a sequência de teclas: APPS G ▽ OK para chegar na seguinte tela:



Vamos explicar a função desta seis aplicações:

1. Send to Calculator..	.....	Envia os dados para outra calculadora (ou a um computador com porta infravermelha).
2. Get from Calculator	.....	Recebe dados de outra calculadora (ou de um computador com porta infravermelha).
3. Print display	.....	Envia o visor para impressão.
4. Print..	.....	Imprime o objeto selecionado da calculadora.
5. Transfer..	.....	Transfere os dados para outro dispositivo.
6. Start Server	.....	Calculadora definida como um servidor para a comunicao com os computadores.

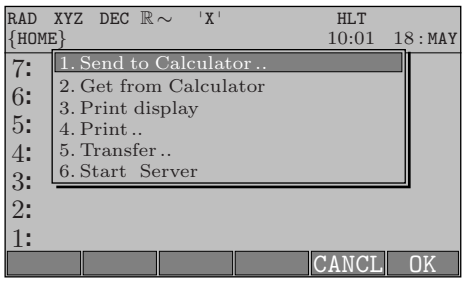
2.2. Pois bem, pressionando OK na tela acima o visor da calculadora que irá transmitir estará no seguinte modo:



Pressionando CHOOS a calculadora entrará no diretório corrente para que você possa selecionar o objeto a ser transmitido. Faça isto e pressione OK , a calculadora retornará com o visor acima já com o objeto pronto para ser enviado, o que pode ser feito pressionando-se SEND , mas antes você deve prepará a calculadora receptora.

3. **A máquina que receberá a informação**.

3.1. Pressione a sequência de teclas: APPS G ▽ OK para chegar na seguinte tela:



3.2. Agora pressione ▽ OK para receber a informação.

. Quadratura Gaussiana

Inrodução (motivação): Com o intuito de apresentar outros recursos disponibilizados pela calculadora, resolvemos implementar mais uma das técnicas de cálculo numérico de integrais, a `quadratura gaussiana`.

Confesso que à medida que fui pesquisando − no manual da $HP-50g$ − os recursos de que necessitava para a referida implementação fui simplesmente tomado de um sentimento misto de *pasmo (assombro)* e *contentamento** com os recursos de computação algébrica (simbólica) desta calculadora.

Apresentaremos a seguir a técnica da quadratura gaussiana mas, antes, fazemos a observação de que, mesmo que o leitor não esteja interessado nesta técnica, creio que êle só terá a ganhar em acompanhar nossa exposição pois nela estaremos patenteando alguns recursos de Cálculo e Aritmética que, creio, deve ser do interesse da maioria dos usuários da $HP-50g$.

Uma outra observação que faço é que, desde o início da composição deste livro, sempre estive atento para não torná-lo extenso (um "outro manual") assim é que não estaremos explorando aqui todos os recursos sobre cálculo e aritimética mas tão somente aqueles que estão em um "entorno" da solução do nosso problema.

Uma outra observação é a de que os referidos recursos podem ser trabalhados, na $HP-50g$, tanto no *modo algébrico* quanto no *modo de pilha* (RPN), nos manteremos fiéis à nossa opção inicial de trabalhar apenas no modo RPN.

Quadratura Gaussina

Para calcular a integral $\int_a^b f(x)\,dx$ pela técnica da quadratura gaussiana devemos fazer uma mudança de variável, assim:

$$\int_a^b f(x)\,dx = \int_{-1}^{1} F(t)\,dt,$$

onde:

$$x = \frac{1}{2}\cdot(b-a)\,t + \frac{1}{2}\cdot(b+a) \quad \therefore \quad dx = \frac{1}{2}\cdot(b-a)\,dt$$

e,

$$F(t) = \frac{1}{2}\cdot(b-a)\cdot f\Big(\frac{1}{2}\cdot(b-a)\,t + \frac{1}{2}\cdot(b+a)\Big)$$

*Ao perceber que meu problema poderia ser resolvido de um modo "`simples e estético`".

Gauss nos diz que:

$$\int_{-1}^{1} F(t)\,dt = \sum_{i=0}^{n-1} w_{_i}\,F(t_{_i}) = w_{_0}\,F(t_{_0}) + w_{_1}\,F(t_{_1}) + \cdots + w_{_{n-1}}\,F(t_{_{n-1}})$$

onde:

$n-$ é o número de pontos que escolhemos no intervalo $]\,-1,\,1\,[$;

$w_{_i}-$ são os coeficientes (pesos);

$t_{_i}-$ são as raízes.

O erro da estimativa acima é dado por:

$$E_{_n} = \frac{2^{2n+1}\,(n!)^4}{(2n+1)\cdot\big((2n)!\big)^3}\cdot F^{(2n)}(\xi), \qquad -1 < \xi < 1.$$

Fixado arbitrariamente um natural n, $t_{_i}$ é a $i-$ésima raiz do polinômio de Legendre de ordem n: $P_{_n}(t)$. Estes polinômios estão disponíveis na $HP-50g$ como veremos logo mais.

Os coeficientes (pesos) $w_{_i}$, são obtidos pela expressão:

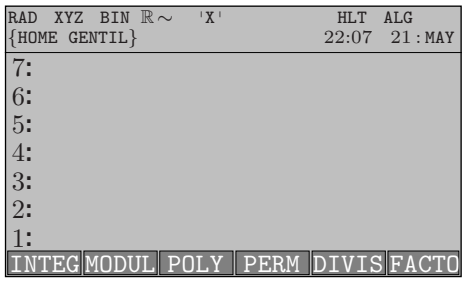
$$w_{_i} = \frac{2}{(1-t_{_i}^2)\cdot[\,P'_{_n}(t_{_i})\,]^2} \tag{4.5}$$

onde $P'_{_n}(t_{_i})$ é o valor da derivada de $P_{_n}(t)$ avaliada no ponto $t_{_i}$.

Bem, posto o nosso problema vejamos agora alguns recursos que a calculadora nos fornece para resolvê-lo:

O Menu ARITHMETIC

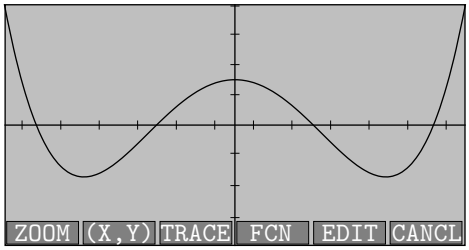
O menu ARITHMETIC contém um número de submenus para as aplicações específicas na teoria numérica (inteiros, polinômios, etc.), como também um número de funções que se aplicam às operações aritméticas. O menu ARITHMETIC é ativado através da combinação de teclas [↰] [ARITH]. Com o sinalizador do sistema 117 configurado assim: ✓ 117 `Soft MENU`, ao pressionar [↰] [ARITH], o resultado é o seguinte:



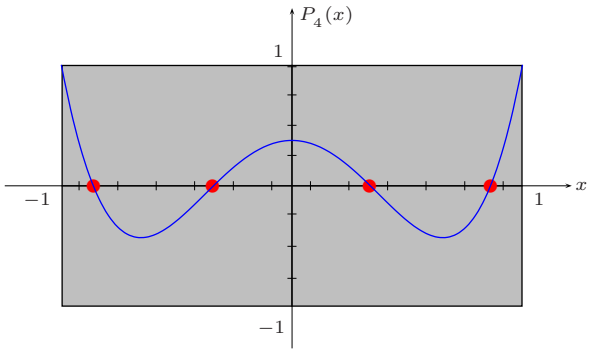
Ao pressionar: [POLY] [NXT L] [NXT L] temos acesso aos polinômios de Legendre. Para obter o polinômio de ordem 4, por exemplo, basta colocar 4 na pilha e pressionar a tecla virtual [LEGEN]. Podemos fácilmente plotar os polinômios de Legendre de ordem n. Por exemplo, vamos plotar o polinômio de ordem 4. Antes redimensione a janela de plotagem ([↰] [WIN]) para:

```
H-View: -1      1
V-View: -1      1
```

Armazene o seguinte programa: ≪ 4 LEGENDRE ≫ na variável `EQ`, agora basta ir ao ambiente de plotagem (veja na pág. 68). Você deverá ver algo como:



Retirando os menus e pondo em destaque as raizes, temos:



Vamos nos concentrar na equação (4.5), então necessitaremos dos zeros do polinômio de Legendre e também de sua derivada. Por exemplo, coloque $P_3(x)$ na pilha*,

$$\frac{5x^3 - 3x}{2}$$

Para obter os zeros deste polinômio inicialmente coloque a variável X na pilha e depois digite ZEROS; a calculadora nos dará os zeros em uma lista:

$$\left\{ 0 \quad -\frac{\sqrt{15}}{5} \quad \frac{\sqrt{15}}{5} \right\}$$

*Para isto digite, se necessário, a sequência de teclas:

[↰] [ARITH] [POLY] [NXT L] [NXT L] 3 [LEGEN]

Coloque novamente $P_3(x)$ na pilha para calcularmos sua derivada. Agora, pressione: ↰ CALC 'DERIV' 'DERVX' para obter,

$$\frac{(5x^2-1)\cdot 3}{2}$$

Iremos necessitar de uma importante função das listas:

A função MAP

A função MAP usa como argumentos uma lista de números e uma função $f(X)$ ou um programa de formulário $\ll \rightarrow \cdots \gg$, e produz uma lista consistindo da aplicação daquela função ou programa para a lista de números. Por exemplo, coloque na pilha:

$$\begin{aligned} 2: &\quad \{0\ \ 1\ \ 2\} \\ 1: &\quad 1-X^2 \end{aligned}$$

ao digitar MAP e dá ENTER a calculadora nos devolve:

$$\left\{\, 1-0^2\ \ 1-1^2\ \ 1-2^2 \,\right\}$$

Digite: XNUM ENTER, para obter,

$$\{\, 1\ \ 0\ \ -3\,\}$$

Como mais um exemplo, coloque na pilha:

$$\begin{aligned} 2: &\quad \{1\ \ 2\ \ 3\ \ 4\ \ 5\} \\ 1: &\quad \ll \rightarrow\ \ X\ \ \text{'}X\ \text{\^{}}2\text{'}\ \gg \end{aligned}$$

ao digitar MAP e dá ENTER a calculadora nos devolve:

$$\{1\ \ 4\ \ 9\ \ 16\ \ 25\}$$

Programando a fórmula de quadratura gaussiana

Creio que já dispomos das principais ferramentas para implementar a quadratura gaussiana. Como exemplo, vamos resolver a integral:

$$\int_a^b \frac{\log x + x^2}{(x+3)^2}\,dx$$

Antes vamos armazenar $f(x)$ numa variável, isto pode ser feito assim:

$$\ll \rightarrow\ x\ \ \text{'}(\,\mathrm{LOG}(x) + x\ \text{\^{}}2\,)/(x+3)x\ \text{\^{}}2\ \text{'}\ \gg$$

Armazene este programa na variável FUNC (função).

Agora faremos uma subrotina que recebe n (a ordem do polinômio de Legendre) e calcula as raizes deste polinômio e, ademais, os respectivos pesos, segundo a fórmula:

$$w_i = \frac{2}{(1-t_i^2)\cdot[\,P_n'(t_i)\,]^2}$$

Então,

```
≪ → n
   ≪ n LEGENDRE DUP DERVX SWAP
      'X' ZEROS DUP 'ti' STO
      SWAP MAP XNUM 2 ^ ti '1− X^2'
      MAP XNUM * 2 SWAP / 'Ai' STO
   ≫
≫
```

Armazene (STO) esta subrotina como: 'tiAi' . Se o leitor quiser ter acesso à lista com as raizes t_i e à lista A_i, com os respectivos pesos pressione, se necessário, a tecla VAR J e procure, estas listas, no menu.

Finalmente, o programa principal fica assim:

```
≪ → n a b
   ≪ n tiAi ti
      '(b − a)/2 ∗ FUN((b − a)/2 ∗ X + (b + a)/2)'
       MAP XNUM Ai ∗ ΣLIST
   ≫
≫
```

A instrução $\sum$LIST, acessada com a seguinte sequência de teclas: ← MTH LIST ∑LIST, calcula a soma de todos os elementos de uma lista. Se necessário, execute-o no DBUG. Para o cálculo da integral de nosso exemplo, entrando com: 2 4 8, (isto é, $a = 2$, $b = 4$ e $n = 8$) obtemos $I = 0.521284$.

Em ← FILES podemos ter acesso ao tamanho (em bytes) de cada um dos nossos programas para a quadratura gaussiana, temos:

FUNC	→ 58
tiAi	→ 150
QGAUSS	→ 157

de sorte que os três programas somam 365 bytes.

GENTIL LOPES DA SILVA

Gentil Lopes da Silva (1960 – ?) nasceu em Boa Vista-RR. *Ayahuasqueiro* e pai de quatro filhos: Agnus, Aline, Ananda e Aarão.

Afora escrever artigos matemáticos, me resta um único hobby: bailar na força da ayahuasca. Por sinal sou produtor (feitor) deste sacramento (enteógeno).

Até 1979/1 (ano de conclusão do 2º Grau/atual ensino médio) o autor, não sendo exceção à regra, possuia aversão pela Matemática; tendo sido reprovado em dois anos escolares (6ª série e básico – aqui aquelas famosas expressões algébricas: "de dentro para fora: primeiro parentesis, depois colchetes e, por último, chaves" que tortura!...). Em 1979, após ter deixado incompleto um curso por correspondência em eletrônica (Instituto Universal Brasileiro), partiu, com "toda" esta bagagem, para Belém-Pa com o intuito de cursar Engª Elétrica. Após um semestre (79/2) de estudos ininterruptos (o autor morava em uma república de estudantes e sobrevivia com uma bolsa da SEC do seu estado..."sem dinheiro no banco, sem parentes importantes e vindo do interior...") o autor foi reprovado no vestibular de jan./80, sendo que nas provas de Física e Matemática (60 questões cada uma) fez 7 e 8 questões, respectivamente (com "chute e tudo"). Acontece que o autor sempre acreditou no velho adágio popular que diz:

"Água molhe, pedra dura, tanto bate ate que fura"

Uma lição ficou do malogro no vestibular: Precisava rever toda a álgebra elementar (aproximadamente a matemática de 5ª a 8ª séries). Cruzou com um livro "Álgebra elementar" (de Barnett Rich) – Coleção Schaum – estudou este livro com afinco por três meses ininterruptos, após o que, não mais sentiu dificuldades quanto à Matemática e Física secundárias. Em função disto, sempre que solicitado, sugere a seus alunos que adquiram um forte fundamento de: frações, potênciação, radiciação, polinômios,... e por aí vai.

Nota: Hoje, chego à conclusão de que a parte "algoritmica" (mecânica) – ou ainda: cálculos, contas – embora essencial (pré-requisito) não chega a ser matemática. Matemática é lógica, é criatividade. Estou tentando contribuir no sentido de se entender (justificar) por que embora um indivíduo não seja um "exímio calculista" mesmo assim pode ter êxito na matemática (lógica).

⊙ ⊙ ⊙

O autor é graduado em Engenharia Elétrica/Eletrônica pela Universidade Federal do Pará e Mestre em Matemática pela Universidade Federal de Santa Catarina.

Ensinou nas seguintes instituições:

(*i*) Universidade Federal de Roraima;

(*ii*) Centro de Educação Tecnológica do Paraná (CEFET-Pr);

(*iii*) Universidade Federal de Santa Catarina;

(*iv*) Faculdades Integradas do Planalto Central (FIPLAC- Brasília-D.F.);

(*v*) Universidade Federal de Roraima (Novamente/atual).

Também trabalhou como engenheiro de telecomunicações do Sistema Telebrás.

www.dmat.ufrr.br/gentil
gentil.silva@gmail.com

"... da matemática que é eterna, porque suas melhores manifestações podem, como as melhores manifestações da literatura, continuar causando uma intensa satisfação a milhares de pessoas, milhares de anos depois." (G.H. Hardy)

"A obtenção de um resultado novo em pesquisa é, para o cientista, uma fonte de intenso prazer, ligado intimamente ao instinto de criação e eternidade, pois, independentemente da importância da contribuição no contexto da ciência, ou de sua utilização, representa algo acrescentado ao conhecimento humano que marca sua existência na terra" (Pierre Curie (Físico))

"Um exame superficial da matemática pode dar uma impressão de que ela é o resultado de esforços individuais separados de muitos cientistas espalhados por continentes e épocas diversas. No entanto, a lógica interna de seu desenvolvimento nos lembra muito mais o trabalho de um único intelecto, desenvolvendo o seu pensamento sistemático e consistentemente, usando a variedade das individualidades humanas somente como um meio. Assemelha-se a uma orquestra executando uma sinfonia composta por alguém. Um tema passa de um instrumento a outro, e quando chegou a hora de um dos participantes abandonar o tema, ele é substituído por outro, que o executa com precisão irrepreensível..." (I.R. Shafarevich)

"Em suas fases primitivas o homem não podia adorar senão a um Deus feito à sua imagem e semelhança, porque não sabia conceber algo melhor. Atualmente, o Deus cósmico, que a ciência nos deixa entrever, já não cabe dentro das velhas concepções religiosas. As nossas idéias evoluem intimamente relacionadas ao progresso da nossa capacidade de concepção. A religião de amanhã se unirá à ciência e deverá se basear em postulados racionalmente demonstrados, se quiser ser aceita." (P. Ubaldi/A Descida dos Ideais)

"Tudo isso, que à primeira vista parece excesso de irrazão, na verdade é o efeito da finura e da extensão do espírito humano e o método para encontrar verdades até então desconhecidas." (Voltaire)

"Nenhuma produção de ordem superior, nenhuma invenção jamais procedeu do homem, mas emanou de uma fonte ultraterrena. Portanto, o homem deveria considerá-la um dom inspirado do Alto e aceitá-la com gratidão e veneração. Nestas circunstâncias, o homem é somente o instrumento de uma Potência Superior, semelhante a um vaso julgado digno de receber um conteúdo divino." (Goethe)

"É uma experiência como nenhuma outra que eu possa descrever, a melhor coisa que pode acontecer a um cientista, compreender que alguma coisa que ocorreu em sua mente corresponde exatamente a alguma coisa que aconteceu na natureza. É surpreendente, todas as vezes que ocorre. Ficamos espantados com o fato de que um construto de nossa própria mente possa realmente materializar-se no mundo real que existe lá fora. Um grande choque, e uma alegria muito grande" (Leo Kadanoff).

"... mas é natural também que, com o desenvolvimento da inteligência, se prefira lutar contra inimigos mais importantes tais como a animalidade de cada um a superar, o ignoto a conquistar, o mistério a revelar, e que o amor não seja só para a mulher gerar, mas para o super-ser que encarna, com o ideal, um tipo superior de vida." (Pietro Ubaldi/A Descida dos Ideais)

Amaringo-Ayahuasca

Referências Bibliográficas

[1] HEWLETT PACKARD. *Guia do Usuário da $HP-50g$* .

[2] Dieguez, José Paulo P. *Métodos numéricos computacionais para a engenharia*. Rio de Janeiro: Âmbito Cultural, 1994.

[3] Barroso, Leônidas Conceição, et. all. *Cálculo Numérico (com aplicações)*. São Paulo: Editora Harbra, 1987.

[4] Silva, Gentil Lopes. *Novas Seqüências Aritméticas e Geométricas*. Brasília - DF: THESAURUS EDITORA, 2000.

[5] Silva, Gentil Lopes. *Seqüências Aritméticas e Geométricas Multidimensionais* (www.dmat.ufrr.br/gentil), 2008.

[6] Silva, Gentil Lopes. *Progressões Aritméticas e Geométricas de ordem m* (www.dmat.ufrr.br/gentil), 2006.

[7] Silva, Gentil Lopes. *Limite da função quadrática* (www.dmat.ufrr.br/gentil), 2008. (Menu Alunos).

[8] Silva, Gentil Lopes. *Topologia Quântica* (www.dmat.ufrr.br/gentil), 2008.

[9] Silva, Gentil Lopes. *Traçados $3-D$(Um auxílio para o traçado de figuras no LaTeX)* (www.dmat.ufrr.br/gentil), 2007.