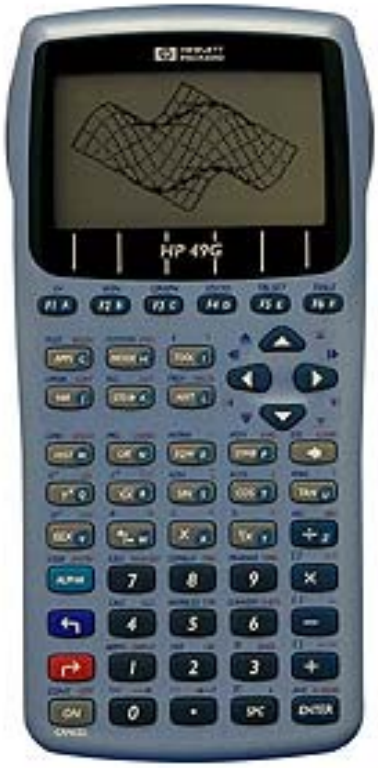
HP49G

-INSTITUTO POLITÉCNICO DE BRAGANÇA

-ESCOLA PROFISSIONAL DE TRANCOSO

Curso de programação UserRPL



DATA: 30/08/2002

UserRPl_v1.0 By:

? *Paulino de Jesus Pereira Lourenço*

? *Técnico de Mecânica/frio e climatização*

? *Estudante Eng. Mecânica*

? *INTITUTO POLIT´ÉCNICO DE BRAGANÇA*

? *Mogadouro P O R T U G A L*

HP49 FAQ:

Perguntas Frequentemente Feitas Sobre a HP49

1. Introdução e hardware

2. Utilização

3. Desempenho

4. Software

5. Miscelânea

1. Introdução e hardware

Este preliminar que HP49 FAQ é mantido por Eric Rechlin. Tudo questões e respostas são do comp.sys.hp48 newsgroup ou da literatura HP sobre o produto. Se há qualquer pergunta que acha que deveria ser incluída, por favor contacte-me e eu considerarei sua inclusão <eric@hpcalc.org>.

1.1 o que é a HP49?

As HP49G são a mais recente calculadora gráfica da Hewlett-Packard. Tem 512K de RAM e 2MB de memória flash. 1MB de memória flash são usados pelo ROM (com capacidade de se fazer upgrades) e o outro 1MB estão disponíveis ao usuário. Tem o mesmo 4 MHz Saturno como antes, mas o software é rescrito para fazer a calculadora operar mais rápida.
Embora a tela ainda é de 131x64, agora ela é preta em vez de azul resultando em um contraste muito mais alto. O teclado tem 51 teclas, mas a tecla ENTER é agora pequena e no canto de inferior direito como a maioria das outras calculadoras gráficas de outras marcas. As teclas são de borracha mas eles ainda têm um tato como da HP48. Em modo alfa, as teclas de setas, números, e tecla de retrocesso mantém seu funcionamento sem ter que desligar o modo alfa.
A caixa é azul metálica leve com uma tampa deslizante num matiz azul translúcido.
A calculadora ainda se ajusta no protetor da HP48 caso você prefira.
O infravermelho foi removido mas a porta serial (com um conector de 10 pinos no estilo da 38G) foi mantido. Os protocolos Kermit (binário e ASCII) e Xmodem (agora com 1K e 1K CRC além de 128 checksum) foram mantidos. Também pode ser conectado a um projetor LCD, um PC, a um data-logger, ou outra calculadora HP48/HP49.

1.2 por que as HP49G não são expansíveis?

A HP49G têm um total de 2.5MB de memória (512KB RAM e 2048KB flash ROM).
Porque nenhuma destas é coberta, um algoritmo de troca de banco de memória novo teve que ser criado. Além disso, a administração de memória da HP48 é MUITO lenta, especialmente com cartões de memória instalados, assim a administração de memória teve que ser rescrita e resultando em um aumento de 10 vezes na velocidade.
A arquitetura de memória nova está mais aberta e muito fácil de programar, mas isto removeu a possibilidade de uma porta de expansão. Ou era 512KB de RAM e 2048KB flash ROM ou 256KB de RAM e uma porta de expansão: alguém pensa se a decisão errada foi tomada?
Além disso, as HP49G não têm um X em seu nome de modelo (como os HP48GX e HP48SX) assim não deveria ser esperada nenhuma capacidade de expansão. A HP não anunciou nenhuma HP49GX.

1.3 por que há nenhum infravermelho?

O infravermelho foi removido na a HP49G por duas razões:
Por causa do infravermelho, a HP48 foi proibida em escolas em quase todos países europeus, na Austrália, na África, etc. Os professores não perceberam a distância limitada que o IR poderia enviar e receber e por isto tiveram medo que a calculadora seria usado para "copiar".
O mercado americano não é tão grande em comparação com o resto do mundo.
A HP realmente faria uma calculadora que não seria lucrativa ao longo do mundo?
A HP percebeu o IR era útil a algumas pessoas mas não pôde manter algo que iria afetar, e muito, as vendas.
Também havia razões técnicas para omitir o infravermelho. O CPU Saturn só tem um pino de produção de alta energia que era utilizado pelo infravermelho na HP48. Porque as 49 têm memória flash que precisa de alta energia o infravermelho tinha que ser removido. Também, havia nenhum acesso escrito ao ROM nas HP48, mas considerando que memória flash precisa disto, o sinal escrito do IR foi usado.
Basicamente era uma escolha entre ter infravermelho e vendas mais baixas ou memória flash e vendas mais altas. Parece o HP tomou a decisão certa.
Um cabo para conectar duas HP49 ou uma HP49 e uma HP48 é incluído com o calculadora. Transfere dados a 15,630 bits por segundo: aproximadamente sete vezes mais rapido que a velocidade padrão do infravermelho da HP48, ou duas vezes mais rápido quanto a velocidade máxima do infravermelho da HP48.

1.4 por que o LCD ainda é 131x64?

131x64 é a maior tela que o Yorke CPU (a implementação da HP48/49's da arquitetura do Saturn) pode suportar, de acordo com Jean-Yves Avenard.
Não se preocupe, o LCD é ainda melhor que o da HP48. Porque ele é preto em vez de azul e usa a tecnologia Crystal Clear, possui um contraste mais alto que antes. Um teste não científico independente feito por Peter Karp achou que a HP48G têm um valor de contraste de 0.24 e a HP49G tem um valor de contraste de 0.55, onde 1.0 é o contraste perfeito (o negro em branco) e 0.0 não são nenhum contraste.

1.5 qual o problema com a cor?

Algumas pessoas não gostaram da cor azul metálica da HP49G. Realmente, ela quebrou a tradição da HP de cores escuras e eternas em lugar de usar as mais recentes modas passageiras em cores para dispositivos (metálico e transparente em 1999, parece). Outros dizem que é não profissional. Alguns foram mais longe dizendo que nunca comprariam a calculadora por causa da cor. A maioria das mensagens postadas no comp.sys.hp48 parece mostrar uma certa repugnância cor.
A HP escolheu a cor para atrair aos compradores mais jovens. Talvez os estudantes de escola secundária realmente goste da cor. Eu pessoalmente, não tenho nenhum problema com a cor e de fato gosto, embora eu preferiria algo mais tradicional. Mas para mim é apenas uma calculadora, então enquanto ela trabalhar (e que trabalhe por muito tempo) eu estou contente (opinião do Eric, eu ainda não me acostumei nem com a cor nem com a forma – opinião do tradutor).

1.6 o que é memória flash?

Memória flash, ou flash ROM, é um meio de armazenamento não volátil. Flash pode ser lido tal qual uma ROM normal, praticamente com a mesma velocidade e com o mesmo consumo de energia, mas também pode ser escrita. A escrita na memória exige um consumo de alta energia e é relativamente lento, assim não é prático para memória do sistema principal.
A HP49 têm 2MB de memória flash. O primeiro megabyte é usado pelo sistema para a ROM (o software do sistema interno). Isto significa que pode ser feito o upgrade da ROM assim que bugs do sistema forem achados e corrigidos. O segundo megabyte está vazio, disponível ao usuário para armazenamento de dados. Embora algumas memórias flash estão bastante limitadas quantas vezes podem ser apagadas e rescritas, a HP garante o flash ROM da HP49G's para 1 milhão de ciclos.

1.7 qual o tamanho da HP49G?

A HP49G pesam 264 gramas. Tem 18.6 cm de comprimento por 8.9 cm de largura por 2.8 cm de espessura dentro de sua caixa protetora e 18.3 cm de comprimento por 8.7 cm de largura por 2.3 cm de espessura quando sem ela.

1.8 que acessórios estão disponíveis?

Um kit de conexão novo (parte F1897A para PC, F1898A para Mac) está disponível para uso com a HP49G com seu conector de 10-pinos, embora um kit de conexão da HP38G podera trabalhar bem. O projetor dos para os modelos 38G/48SX/GX, parte F1212A, trabalha bem através da porta serial de 10 pinos da HP49G. A impressora infravermelha porém não é suportada.
Além disso a Firmware Systems, Inc. está criando um produto chamado Portable DataLab para coletar dados com a HP49G.

2. Utilização

2.1 por que a tecla ENTER é tão minúscula e no canto?

Há vários argumentos que as pessoas pensaram de e outros que os empregados de HP têm mencionado no newsgroup:
Para ter teclas suficientes para permitir as teclas de números, ENTER, retrocesso, e as setas e tudo mais pudessem ser utilizadas enquanto a HP estivesse em modo de alfa, mais teclas tiveram que ser somadas. Duas teclas menores poderiam ser postas em lugar de uma tecla ENTER grande o que resulta em uma tecla ENTER menor ganhando uma tecla.

Uma vez que a tecla ENTER é menor, não há nenhuma real razão para ficar no meio do teclado. Por que não coloca-lá no canto inferior direito?
TI e Casio puseram a tecla ENTER lá, assim por que não a HP?

2.2 por que tecla / está compartilhada com a tecla Z?

Novamente para permitir que todas as teclas de números, setas, retrocesso, e ENTER possam ser acessíveis em modo alfa, um compromisso teve que ser feito. Neste caso, a tecla / se torna Z em modo alfa.
A tecla / ainda pode ser digitada em modo alfa com a tecla shift direita, assim necessitasse de apenas mais um toque de teclado para aciona-lá. Isto não é nada comparado a quantidade de digitação economizada por não ter que desactivar o modo alfa para usar as teclas de seta.

2.3 por que não existem três teclas postas à esquerda das setas para resolver o problema do ENTER e do /?

É verdade que três teclas se ajustariam lá - porém, não seria uma boa ideia. Apesar do fato que a HP48G têm menos teclas que as calculadoras gráficas da série TI (da Texas), parece que estas tem mais por causa de seu layout. A HP49G na verdade têm mais teclas do que a TI mas parece ter menos por causa de seu layout.
Algumas pessoas têm medo de comprar uma HP48 porque o desenho de seu teclado parece complexo. A HP49 têm um desenho de teclado que parece menos complexo o que a faz parecer mais amigável para compradores potenciais.

2.4 as funções de matemática tem que estar no menu CHOOSE

Não, você tem o melhore de ambos os mundos mais uma vez. Usuários novos não gostam de usar menus de softkey porque a limitação de 5 letras faz com que fique difícil de ver a função de cada uma, assim eles preferem o menu CHOOSE com mais letras mostradas. Os usuários avançados preferem o menu de softkey porque eles são mais rápidos e precisa-se de menos toques no teclado.
A HP49 tem como padrão o menu ESCOLHER mas permite que o usuário troque por um menu de softkey se desejar. Simplesmente ajuste o flag -117 e você regressam ao menu por softkeys.
Este novo menu CHOOSE é mais rápido que o menu da HP48, permite a utilização de teclas de atalho, tem uma barra de rolagem, e mostra até oito linhas.

2.5 o que é a tecla CAT?

A tecla CAT expõe um menu CHOOSE que lista TODOS os comandos da HP49 incluindo os comandos de bibliotecas instaladas. É dinâmico contudo muito rápido.

2.6 o que é a tecla TOOL?

A tecla TOOL expõe um menu sensível ao contexto de ferramentas relacionadas com a aplicação corrente. Por exemplo, no Matrix Writer, isto pode trazer um menu de comandos úteis de matriz. No Text Editor, isto pode trazer um menu com o comando Find/Replace e outros comandos que editam o texto.

2.7 quão segura é a memória flash?

Porque as memórias flash não tem um interruptor físico de protecção contra escrita como em cartões RAM, algumas pessoas se preocuparam com a segurança de dados nestas memórias.
Uma limpeza de memória (ON-A-F) não apagará os dados armazenados na memória flash. Afinal de contas, se isto fosse possível, o software de sistema poderia ser apagado a qualquer instante. Isso não é uma coisa boa...
É IMPOSSÍVEL apagar os dados em uma memória flash por acidente, de acordo com Jean-Yves Avenard. A HP testou enviando 25,000 volts pelo porta serial e a memória flash sobreviveu.

2.8 por que o padrão está no modo algébrico?

Quando alguém testa uma calculadora pela primeira vez em uma loja, ele/ela espera ser capaz de a usar imediatamente. A maioria das decisões de compra são feitos nos primeiros três minutos. Por ter a calculadora como padrão o modo algébrico, o comprador potencial saberá como usa-lá, uma vez que a maioria das pessoas não está familiarizado com RPN.
O usuário pode trocar facilmente para o modo RPN, bastando, apertar algumas teclas. É um ajuste de flag, assim o modo RPN permanecerá ligado, até mesmo depois de desligar a calculadora ou após uma “partida a quente”.
Todas as funções de RPN familiares permanecem (SWAP na seta direita, pilha interativa seta para cima, DUP na tecla ENTER, e DROP no retrocesso).

2.9 teclas de borracha? O que a HP está pensando?

Embora a HP49G tenha teclas de borracha, eles não são iguais as teclas de borracha tradicionais.
Eles têm um senso tátil semelhante ao teclado da HP48. Estes são teclas de qualidade, que devem durar tanto quanto qualquer outro teclado de HP.
O teclado novo também elimina o “balanço” das teclas um problema com o teclado da HP48. A HP48 trabalhou este problema reduzindo a velocidade de acesso ao teclado que reduz a velocidade do software por mais de três vezes. O teclado da HP49 não possue este “balanço” assim pode esquadrinhar o teclado em menor tempo e resultar em softwares mais rapidos.

2.10 Eeek. Quem quereria uma proteção rígida?

A tampa rigida protege as teclas e o display. Desliza para trás sobre a calculadora enquanto esta em uso, assim não há nenhum problema com precisar achar um lugar para pôr uma tampa separado, como com a capa da HP48.

Para esses que preferem uma capa no estilo HP48, a HP49 se ajusta na capa da HP48.
Se alguém quer mais proteção, até mesmo a HP49 em sua capa rigida se ajusta dentro da antiga capa, dando proteção acolchoada contra quedas, arranhões quando em uma mochila.

3. Desempenho

3.1 quão rápido é o plotador 3D?

Apesar de ter uma CPU de 4 MHz, a HP49G podem fazer plotagens tridimensionais mais rapidamente que QUALQUER calculadora atual, inclusive a TI-89.
Todos os gráficos 3D podem ser girados nos eixos X, Y, e Z em TEMPO REAL, mais rápido que a TI-89. Ela desenha uma matriz 14 por 14 por volta de 6 quadros por segundo

3.2 eu ainda não vejo como uma CPU de 4 MHz pode fazer uma calculadora tão rápida

A HP contratou alguns dos melhores programadores de HP48do mundo para projetar a HP49G.
Jean-Yves Avenard, Gerald Squelard, e Cyrille de Brebisson, todos do time do Meta Kernel, estão lá. Mika Heiskanen, autor de Jazz, ALG48, BZ, QPI, e muito mais, também está trabalhando nisto. Bernard Parisse, autor do Erable, também tem contribuído.
A rotina de administração de memória nova é consideravelmente mais rápida, e memória coberta não é mais uma preocupação. O GUI, incluindo janelas de informação e caixas CHOOSE, são de 10 a 100 vezes mais rapidos.Plotagem 3D é muito mais rápida. Manipulação simbólica é mais rápida do que qualquer outra calculadora no mercado.

3.3 por que o CPU é um Saturn de 4 MHz, inalterado, das HP48G/GX?

1 O CPU ainda é um Saturno por causa do custo enorme e de tempo relacionado para usar um CPU novo. O sistema operacional teria que ser completamente rescrito a partir de rascunhos, desperdiçando os esforços dos últimos 15 anos. Levou mais de 200 engenheiros/anos terminar a HP48 somente.
A TI levou cinco anos para criar a TI-92 (de acordo com o gerente de projeto da TI-92), com seu processador 68000, um novo CPU para TI usar em uma calculadora naquele

tempo. Usando um CPU novo demorariam outros três ou cinco anos para ter uma calculadora nova

2 O CPU ainda está em 4 MHz por razões técnicas e práticas. Uma companhia alemã roda o Saturno a 6 MHz através de overclocking, mas isto gasta as baterias duas vezes mais depressa.
Também, o chip Saturno era avaliada para até 2 MHz quando era usado na série HP48S.A HP já tinha executado o processo de overclock para 4 MHz para a série HP48G, assim fazer novamente um overclock poderia estar além dos modos especificadoes. Porque o HP quer um produto de qualidade, o chip não sera novamente overclocked.
Aumentar a tolerância do CPU para faze-lo rodar a uma velocidade mais alta iria tomar, estimados, 19 meses, de acordo com Preston Brown que impulsionou o CPU, de 2 a 4 MHz quando a série G foi projetada. O tempo adicional seria necessário para redesenhar e requalificar a mistura entre os modos analógico e IC digital.

Uma explicação mais longa, mais técnica de Dave Arnett que projetou os o hardware da série: HP48G:

“eu vi os dados de qualificação no Yorke IC. Eu sei que eles têm bastante margem de velocidade ao fluxo de processo nominal e a temperatura ambiente. Mas eu também sabia que, aos cantos do CMOS a janela de processamento e a temperaturas extremas, as partes margeavam em 4 MHz. Eu me lembro como nós seriamente consideramos fazer uma divisão de por lotes e ordenar as partes rápidas das partes lentas, e introduzir a 48G como uma máquina de 2MHz. Eu imagino que a maioria das maquinas da série HP48G poderiam ser overclocked contanto que eles não tenham corrido pela bateria de teste ambiental completo da HP naquele modo. E eu tinha um dilema ético primordial se a administração me apertasse para lançar um produto com esses requerimentos”

Porque o software foi rescrito para fazer a calculadora operar 10 a 100 vezes mais rapidamente, uma nova CPU não é tão necessária quanto alguns podem pensar.

3.4 exatamente quão rápida está a calculadora?

Embora o CAS da HP49G está baseado no ALG48, é o resultado de muitos meses de trabalho melhorando-o. Porque tudo é feito com endereços absolutos, o código antigo roda mais rapido também.
100! * 100! que leva aproximadamente 16 segundos para ser calculado com precisão completa em uma HP48 com ALG48, leva aproximadamente 2.5 segundos em uma HP49. Isso é mais rápido que uma TI-89 ou uma TI-92! Integração numérica não é notoriamente mais rápida que na 48 mas integração simbólica é muito mais rápida.
Uma vez que as 49G estejam disponíveis em lojas, porém, nós poderemos fazer nosso próprio teste de desempenho e esta seção será ampliada.

4. Software

4.1 que ferramentas de programação estão incluídas?

A HP49 têm software de desenvolvimento embutido para User RPL, System RPL, e Linguagem assembler. Um disassembler também está incluído! Além, de uma nova linguagem chamada HP Basic que está incluída. Esta é mais ou menos uma versão algébrica do User RPL. Uma amostra do código é mostrado abaixo:

```
FOR(i,1,100)
    DISP(i,1);
    IF  I5  ==  50  THEN  DISP("Hello  World",2)  ELSE  DISP("I'm
off",2) END
STEP(1)
```

As ferramentas de desenvolvimento linguagem assembler e o System RPL não são oficialmente suportadas pela HP, mas eles estão disponíveis em ROM para esses que as querem. Todos as
rotinas de gráficos embutidas apoiam interiormente a escala de cinza.

4.2 ela pode rodar programas da HP48?

A HP49G não é bináriamente compatível com a HP48 porque pontos de entrada foram movido. Porém, é compatível no nível de código de fonte assim os programas podem simplesmente ser recompilados.
Programas em User RPL serão automaticamente recompilados quando envioados a partir de uma HP48 ou de um computador no modo Kermit ASCII.

4.3 quão poderoso é o CAS?

O CAS, ou sistema de álgebra de computador, está baseado no ALG48 e no Erable mas é o resultado de muitos meses de melhorias.
O CAS tem manipulação simbólica dinâmica e resoução. Rotinas para a resolução de cálculo e álgebra linear rotinas "passo-a-passo" para mostrar como a resposta é obtida, e todos os passos são mostrados em "caixas de texto" para uma fácil compreensão.
Características de álgebra incluem fatoração, expansão, substituição, sistemas de equações lineares e álgebra linear. Matrizes simbólicas são suportadas, com redução de fila Gaussian-Jordan, eigenvalues e eigenvectors, diagonalização e
decomposição.
Características de cálculo incluem limites, derivação, integração, equações diferenciais e polinômios de Taylor.
São incluídas características de estatísticas avançadas, como listas, variável única, freqüências, regressão, e somatoria estatísticas. Amostra de descrição de dados pode ser usada para fazer e medir inferencias sobre a população de interesse com teste de hipótese e intervalos de confiança.

4.4 o que aconteceu com a biblioteca de equações?

A HP49G já não incluem uma biblioteca de equações.

Porém, retém a biblioteca de 40 constantes, e conversões de unidade são apoiadas entre 127 unidades embutidas.

5. Miscelânea

5.1 quanto valerá e quando eu posso comprar uma?

A HP não anunciou o preço ou data de lançamento ainda.

5.2 até lá haverá uma HP49GX?

A HP não disse nada sobre uma HP49GX.

5.3 por que a HP49G não têm [inserir condições aqui]?

Entender esta resposta, você precisará primeiro um pouco de resumo da história das calculadoras HP, graças a Jake Schwartz. De 1970 a 1993, todo desenho de calculadora residia em Corvallis, Oregon, no E.U.A.. Em 1993, ao redor do época do lançamento da HP48G/GX, as operações de desenho de calculadoras foi mudada para Cingapura.
Depois que foram movidas para Cingapura, a única calculadora lançada foi a HP38G, o resultado de uma combinação de esforços de Cingapura e algumas pessoas, da HP Corvallis.
A 38G, apontada para estudantes de escola secundária, foi em grande parte um fracasso e a série HP48 permanecia como a calculadora de escolha. Depois de não lançar NENHUMA calculadora nova depois da 38G, parecia que a HP Cingapura não ia fazer qualquer coisa. Assim, em 1997 na HP Handheld Confererence em Londres, foi anunciado um novo grupo de desenvolvimento de calculadoras, a Operação de Calculadoras Australiana (ACO), que foi formada na Austrália em primeiro de novembro, 1997 projetar as futuras calculadoras.
Eu tenho que admitir que a HP49G é parcialmente minha falta. Também em 1997, eu postei na comp.sys.hp48 uma lista de três calculadoras novas eu queria que o HP criasse. A primeiro calculadora que eu sugeri era uma calculadora científica barata que eu chamara de HP23, semelhante as TI-34 mas preferencialmente com RPN. Isto se tornou verdade com a HP6S.
A segunda calculadora que eu sugeri foi a HP48F/FX (embora eu mudasse depois para o N/NX porque F soa tão semelhante a S). Teria 512K de RAM e inclui o Erable, ALG48, e o Meta Kernel e tudo se tornou realidade com a 49. Apesar de minha sugestão de aumentar a velocidade do clock e ficar com um esquema de cor mais tradicional não se tornou realidade.
A terceira calculadora que eu sugeri foi a HP58, com uma arquitetura de CPU nova, um resolução de tela mais alta, 2MB flash ROM, um GUI escrito em assemblere uma interface de linha de comando, e muita RAM. Algo disto se tornou realidade com a 49 e alguma coisa ainda esta para chegar.
Porque o ACO da HP não recebeu nenhum apoio das operações de calculadora prévias, eles estavam completamente por conta própria e teve que provar a eles mesmos com trazer capital. O primeiro produto, anunciado em março de 1998, foi a HP48G+. O

custo da HP48+ foi virtualmente nada para criar e trouxe o dinheiro inicial para continuar os projetos. A HP6S, uma calculadora científica barata anunciada em abril de 1999l, era o segundo produto e a primeira calculadora verdadeiramente nova. A HP49G foi o terceiro produto, baseado, livremente nas HP48.
A HP49G não é um produto de topo de linha por várias razões. Primeiro são as razões listadas acima: O ACO de HP é relativamente novo a calculadoras e não quis arriscar tudo em uma calculadora cara, e não tiveram dinheiro suficiente para investir em um projeto grande. Segundo, a HP precisa impulsionar seu mercado educacional. O único modo para fazer isso é lançar uma calculadora disponível para competir diretamente contra a TI-89.
Além, uma calculadora de topo de linha levaria entre três e cinco anos para projetar. Considerando a HP49G foi anunciado só um ano e um meio após a formação do ACO houve uma espera relativamente pequena para a 49, que é basicamente uma HP48redesenhada. Eu penso que a maioria das pessoas concordará que é melhor vir a HP49G em pouco tempo do que esperar dois ou mais anos para um novo, calculadora de topo de linha.
Em resumo: não seria impráticavel para a HP sair com uma calculadora faz-tudo topo de linha neste momento, assim alguns compromissos tiveram que ser feitos.

5.4 são a HP49G permitidos em testes unificados?

Uso da HP49G é permitido em testes do College Board de , inclusive exames AP, o SAT, e o PSAT/NMSQT. Não é conhecido contudo se a HP49G sera permitida no ACT, mas provavelmente será proibida por causa de sua capacidade de matemática simbólica

5.5 o que vem com a HP49G?

O seguinte é incluído na caixa com a HP49G:
Calculadora HP49G com tampa deslizante (F1633A)
Três baterias AAA
Cabo de ligação de unidade-para-unidade com adaptador para conectar a outra HP49G ou uma HP48
Guia do usuário
Guia de Bolso de Referência rápido
Cartão de inscrição

1-Introdução a Programação

Este curso esta orientado a todos aqueles que pretendem programar em UserRPl, programar na Hp49 não é muito complicado, apenas tens de saber um pouco de lógica, e de diagramas de fluxo, e aprender os comandos de programação existentes.

A HP49G pode ser programada em três linguagens diferentes UserRPL, SistemRPL e Assembly aqui vamos falar somente da linguagem de alto nível que é o UserRPL, linguagem pela qual o utilizador tem acesso a partir do menu **PRG**(program), estes comandos são aproximadamente 160, esta é a linguagem mais elementar e a mais fácil, esta linguagem e similar a outras como Visual Vasic, Fortran etc..

Por ser de alto nível esta linguagem precisa de mais memória e muito mais tempo para executar, visto que tem de realizar umas provas para ver se existe algum erro no código de forma a preveni-lo, o que não sucede com as linguagens de baixo nível Assembly e SysRPL que caso exista erro ocorre uma falha de memória no sistema , e pode causar danos no processador da calculadora.

O problema que existe de incompatibilidade entre a hp48gx e a HP49g tem a ver com os endereçamentos de memória, que na Hp49g são diferentes da Hp48, mas este problema só afecta as linguagens de baixo nível (Assembly e SysRPL) e programas de alto nível (UserRPl), que estão comprimidos codificados ou em librarias. Esta incompatibilidade é devida aos programas que codificam e comprimem estão escritos em baixo nível, a solução para os programas de baixo nível (UserRPL) é descodifica-los descomprimilos ou passar de librarias a variáveis antes de transferir para a HP49G, assim pode corrigir ou ajustar alguns detalhes. Para converter os programas em Assembly ou SystemRPL é muito mais complicado, consiste em mudar os endereçamentos para a HP49G, e não esquecer também os comandos de memória, isto mediante umas tabelas.

Este curso foi criado a pensar na HP49G mas aplica-se perfeitamente a HP48 excepto alguns comandos que são exclusivo da HP49G.

2- O que um programa:

Um programa para a calculadora é tudo aquilo que esta dentro de « ». Um exemplo de um programa pode ser: « 60 9 +», o qual coloca os dois números na pilha e depois soma-os
Para guardar pode utilizar o comando **sto** pode ver mais explicitamente no exemplo a seguir:

3: « 60 9 + »

2: ‘nome_do_programa’
1: **sto**

para o executar pode por nome da programa na pilha e enter ou **eval.**

3- Variáveis:

Dentro dos programa é muito comum chamar objectos, programas ou variáveis, que podem ser estáticas, locais ou globais, esta parte é muito importante na programação, e deve-se ter bastante atenção na maneira como guardamos e utilizamos as variáveis.

3.1- Variáveis locais:

Estas variáveis são armazenadas na memorial temporal durante a execução de um programa, esta forma e bastante recomendada visto que utiliza menos memória e o programa executa-se mais rápido.

Exemplo:

*Neste programa as variáveis são **a,b** e **c** e encontram-se no nível* 1 2 e 3 da pilha.

```
« → a b c
 « a b* (a b + c +)/ »
»
```

Sintaxes:

1- primeiro devem encontrar-se as variáveis nos níveis antes do símbolo(→).
2- Símbolo.
3- Nome das variáveis.
4- Subprograma que utiliza as variáveis («....»).
5- ‘variável’ **STO** (este comando armazena o valor com o nome “variável”).

Atenção:

Estas variáveis só estão disponíveis até ao fim de execução de um Subprograma , uma vez terminado as variáveis deixam de existir.

Exemplo:

```
« → a              Armazenamento da variável local
      « a 2 /      Realiza a operação a+1
      »            Fim do subprograma
      a 1 –        Esta operação falha, não existe variável a
»
```

3.2- Variáveis globais:

Estas variáveis são as que se armazenam na memória do menu **VAR** e podem ser utilizadas depois do programa terminar.

Deve-se ter cuidado ao usar estas variáveis, visto que pode haver programas ou librarias com o mesmo nome da variável, e pode causar erros...

Exemplo:

*Este programa guarda o valor de 0 numa variável cujo nome é **a.***

```
« 0 ‘a’ STO   @Armazena o valor 0 em a
 a 1 +        @Realiza a operação a+1
‘a’ STO       @Guarda o novo valor
»
```

Sintaxes:

1- primeiro deve-se encontrar o valor no nível antes do nome da variável
2- Nome da variável com o símbolo (‘)
3- Comando **STO.**
4- ‘variável’ **STO** (este comando permite mudar o valor da variável)

Para armazenar mais do que uma variável, uma forma fácil é por os valores numa lista, fazer o mesmo com os nomes e guardar.

Exemplo:

```
« { 9 20 8 1 } { a b c d } STO »
```

4-MENU PRG (program)

O menu **PRG**, conhecido pelo menu de programação é composto por:

4.1- STACK
4.2– MEM
4.3- BRCH (Directório especial)
4.4- TEST
4.5- TYPE
4.6- LIST
4.7- GROB
4.8- PICT
4.9- CHARS
4.10- MODES
4.11- IN
4.12- OUT
4.13- TIME
4.14- ERROR
4.15- RUN

Cada um destes submenus vão ser abordados como vamos ver seguidamente.

4.1- comandos do subdirectório STACK

- Este menu tem os comandos que manipulam os dados na pilha da HP.

4.1.1 comando DUP

Este comando duplica o primeiro nivel do **STACK**

4.1.2- comando SWAP

Este comando troca o primeiro nível com o segundo.

4.1.3- comando DROP

Apaga o primeiro nível do stack

4.1.4- comando OVER

Copia o nivel 2 para o nivel 1

4.1.5- comando ROT

Roda em painel rolante os três primeiros níveis de baixo para cima.

4.1.6- comando UNROT

Roda em sentido contrario do **ROT**

4.1.7- comando ROLL

4 ROLL

Este comando tem de inserir o nível que quer trocar pelo primeiro, neste caso troquei o 4 , (4 ROLL).

4.1.8- comando ROLLD

3 ROLLD

Este comando passa o nível 1 para o nível indicado neste caso passamos o 1 para o 3 (3 ROLLD).

4.1.9- comando PICK

3 PICK

Este comando permite Copiar o nível desejado para o nível 1 neste caso copiamos o nível 3 (3 PICK).

4.1.10- comando UNPICK

5 UNPICK

Este comando é similar ao anterior cota o nível 1 e cola no nível desejado, neste caso o nível 5 (5 UNPICK).

4.1.11- comando PICK3

Este comando Copia para o nivel 1 o comando que se encontra no nivel 3

4.1.12- comando DEPTH

Este comando dá-nos a quantidade de níveis que temos em na pilha, neste caso são três.

4.1.13- comando DUP2

Este comando duplica os objectos que se encontrem nos dois primeiros níveis.

4.1.14- comando DUPN

1 DUPN

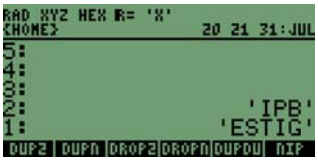
Este comando é similar ao anterior mas temos de fornecer o numero de níveis que queremos duplicar.

4.1.15- comando DROP2

Este comando apaga os dois primeiros níveis do stack

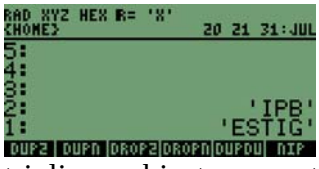
4.1.16- comando DROPN

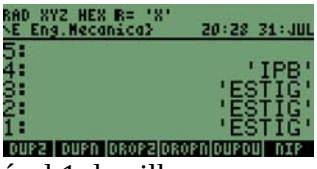
3 DROPN



Este comando permite apagar um determinado numero de níveis na pilha neste caso apagamos 3 (3 DROPN).

4.1.17- comando DUPDU





Este comando triplica o objecto que esta no nível 1 da pilha.

4.1.18- comando NIP

Este comando elimina o objecto que se encontra no nível 2.

4.1.19- comando NDUPN

3 NDUPN

Este comando duplica n vezes o primeiro nivel.

4.2- comandos do subdirectório MEM

- Estes comandos trabalham mais directamente com a memória da maquina.

4.2.1- comando MEM

Este comando damos a quantidade em bits livre que existe na porta 0, aprox. 240 kb.

4.2.2- comando BYTES

Determina o tamanho e Checksun de qualquer objecto, neste caso é o tamanho do numero 246518.5 numero antes de aplicar o comando.

4.2.3- comando NEWOBJ

Cria um novo objecto

4.2.4- comando ARCHIVE

Cria um backup de tudo o que se encontra na Home

4.2.5- comando RESTORE

Restaura a copia backup na Home

4.2.6- comandos da secção DIR

PURGE: Apaga um directório ou uma variável
RCL: Recupera o objecto de uma variável
STO: Guarda uma variável ou directório

CRDIR: Cria um novo directório
PGDIR: Apaga um directório
PATH: Devolve a lista aos dados actuais
VARS: devolve numa lista as variáveis do menu actual
TVARS: Devolve numa lista as variáveis de um certo tipo
ORDER: Ordena as variáveis segundo a ordem que se encontram na lista

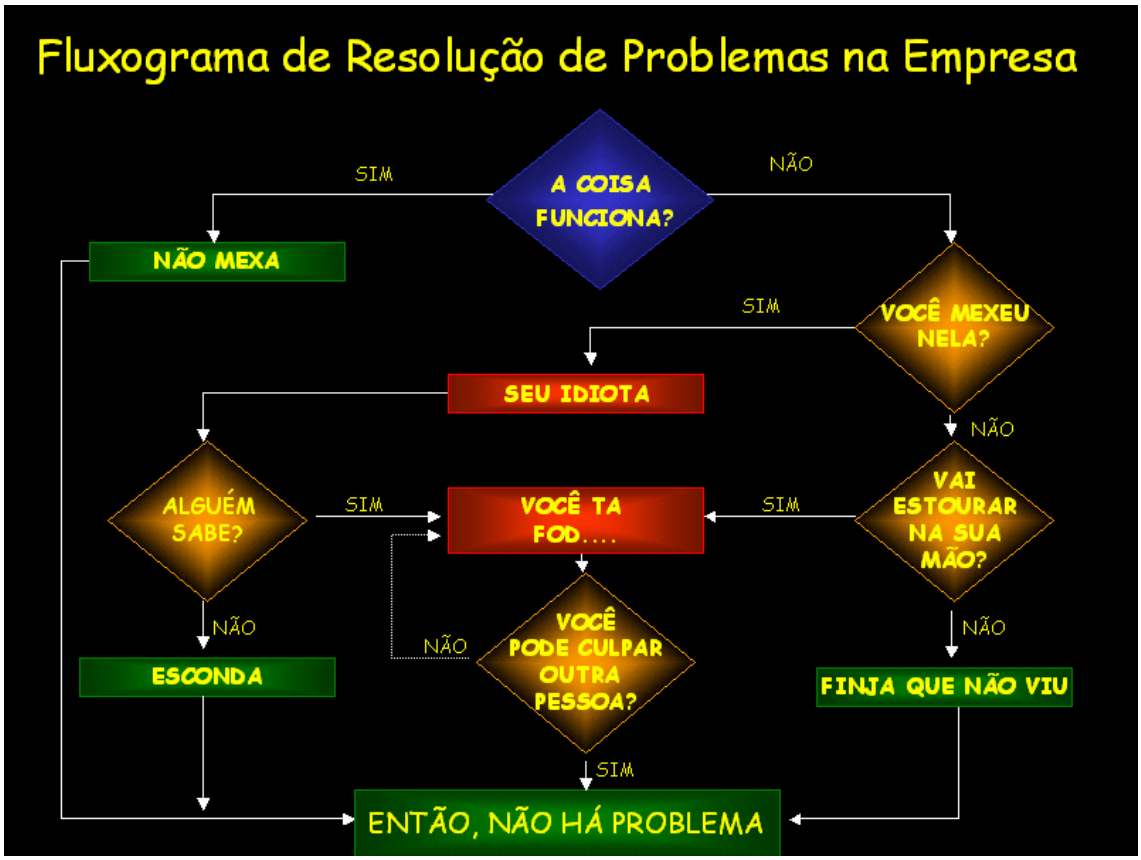
4.2.7- comandos do subdirectório ARITH

STO+: Soma um valor ao valor da variável
STO-: subtrai um valor ao valor da variável
STO*: multiplica um valor pelo valor da variável
STO/: divide a variável por um dado valor
SINV: Restaura o inverso do valor anterior
SNEG: Restaura o valor conjugado de seno
SCONJ: Reúne um objecto a outra variável conjuga
INCR: Restaura o valor anterior incrementado 1 e devolve o valor actual
DECR: Restaura o valor anterior do encremento e devolve o valor actual

4.3- BRCH

ESTRUTURAS CONDICIONAIS E TESTES:

Uma das características das linguagens de programação moderna e ser estruturada, conseguem decidir e responder a uma estrutura sequencial, de acordo com um dado programa. Para escrever um programa numa linguagem primeiro, devemos escrever um algoritmo, que resolvam situações mais complexas, podemos usar fluxogramas para tornar o algoritmo mais fácil de entender, veja o seguinte exemplo, que visa resolver os problemas de uma empresa:



ESTRUTURAS CONDICIONAIS:

As estruturas condicionais permitem a calculadora tomar uma decisão baseada em um resultado de um ou mais testes. As estruturas condicionais são:
- IF ... THEN ... END.
- IF ... THEN ... ELSE ... END.
- CASE ... END.

4.3.1- A Estrutura IF ... THEN ... END

IF ... THEN ... END executa uma sequência de comandos (cláusula verdadeira) se a condição for verdadeira
(cláusula de teste). Sua sintaxe é a seguinte:

IF *cláusula de teste* THEN *cláusula verdadeira* END

A cláusula de teste (condição de teste) pode ser uma sequência de comandos (por exemplo: A B >) ou um algébrico (por exemplo: 'A>B'). No caso de ser um algébrico a expressão é avaliada automaticamente para um número, não necessitando do EVAL ou de NUM.

Exemplo 1:

Os dois programas seguintes fazem um teste do valor no nível 1. Se o valor é positivo transforma-se em negativo. O primeiro programa usa uma sequência de comandos como cláusula de teste:

```
<< DUP IF 0 > THEN NEG END >>
```

O valor na pilha deve ser duplicado porque o comando > extrai dois argumentos da pilha (a cópia do valor é feita
pelo DUP)

A seguinte versão usa um algébrico como cláusula de teste:

```
<< x << IF 'x>0' THEN x NEG END >> >>
```

Exemplo 2:
Este programa multiplica dois números se ambos são diferentes de zero:
Programa: Comentários:

```
<<
→ x y       @Cria variáveis locais x e y que contém os dois números da
             pilha.
<<
IF           @Começa a estrutura de teste condicional.

'x=0'        @Verifica se o número é diferente de zero, deixando o
             resultado do teste na pilha.

'y=0'       @Verifica se o número é diferente de zero,
             deixando o resultado do teste na pilha.

AND         @Verifica se os dois testes são verdadeiros.

THEN        @Executa somente se o teste retornou verdadeiro.

x y *       @Se AND devolver verdade, multiplica os dois números.

END        @Termina a estrutura condicional.
>>
>>
```

O seguinte programa tem o mesmo efeito que o anterior:

```
<< DUP2 IF AND THEN * ELSE DROP2 END >>
```

Exemplo 3:
Este programa pergunta se os dois números da pilha são iguais, em caso afirmativo gera uma mensagem caso contrario não se realiza nenhuma acção.

```
<< IF == THEN "elementos iguais"
   END
>>
```

Como funciona a estrutura IF...THEN...END:

IF começa a cláusula do teste, deixando o resultado do teste na pilha. THEN extrai o resultado do teste da pilha. Se o valor é diferente de zero, executa-se a cláusula verdadeira.
Caso contrário, a execução do programa continua após o comando END.

4.3.2- A Estrutura IF...THEN...ELSE...END

IF...THEN...ELSE...END executa uma sequência de comandos (cláusula verdadeira) se o teste é verdadeiro e
outra (cláusula falsa) se o teste é falso. Sua sintaxe é a seguinte:

IF *cláusula de teste* THEN *cláusula verdadeira* ELSE *cláusula falsa* END

Se a cláusula de teste for um algébrico (por exemplo: 'A>B'), a expressão é avaliada automaticamente para um
número, não necessitando do EVAL ou de NUM.

Exemplo 1:
O seguinte programa toma um valor da pilha e calcula sin(x)/x. Porém em x=0 a divisão estará errada,
por isso o programa devolve o valor limíte 1 neste caso:

```
<< → x << IF 'x=0' THEN x SIN x / ELSE 1 END >>
```

Exemplo 2:
Este programa, como o exemplo 2 para IF...THEN...END, multiplica dois números se ambos são diferentes
de zero. Porém o programa devolve a cadeia "ZERO" se algum dos dois é 0.

Programa:

```
<<
→ n1 n2     @Armazena nas variáveis locais os níveis 1
              e 2 da pilha.
<<
IF          @Começa a estrutura de teste condicional.
'n1=0       @AND Verifica se ambos os números são diferentes
n2=0'       @de zero, deixando o resultado do teste na pilha.
THEN        @Executa somente se o teste retornou verdadeiro.
n1 n2 *     @Se AND devolver verdade, multiplica os dois
            números.
ELSE        @Executa somente se o teste retornou falso.
"ZERO"      @Devolve a cadeia "ZERO"
END         @Termina a estrutura condicional.
>>
>>
```

O seguinte programa tem o mesmo efeito:

```
<< DUP2 AND IF THEN * ELSE DROP2 "ZERO" END >>
```

Programa 2:
Verifica se um numero é positivo ou negativo, e manda a mensagem correspondente.

```
<< →A
      <<IF A 0 >
            THEN "positivo"
            ELSE "negativo"
            END
      <<
<<
```

Como funciona a estrutura IF...THEN...ELSE...END:

IF começa a cláusula do teste, deixando o resultado do teste na pilha. THEN extrai o resultado do teste da pilha. Se o valor é diferente de zero, executa-se a cláusula verdadeira. Caso contrário, se executa a cláusula falsa. Depois da execução da cláusula apropriada, a execução do programa continua após o comando END.

4.3.3- A Estrutura CASE...END

A estrutura CASE...END permite executar uma série de *casos* (testes). O primeiro teste que tem um resultado verdadeiro causa a execução da correspondente cláusula verdadeira, finalizando a estrutura CASE...END.
Opcionalmente, pode-se incluir após o último teste uma cláusula de default que se executará se todos os testes forem falsos.

A estrutura CASE...END tem a seguinte sintaxe:

CASE
***cláusula de teste1* THEN *cláusula verdadeira1* END**
***cláusula de teste2* THEN *cláusula verdadeira2* END**
. . .
***cláusula de teste* ..n THEN *cláusula verdadeir*a ..n END**
cláusula default *(opcional)*
END

Exemplo 1:
O seguinte programa armazena o argumento do nível 1 em uma variável chamada STR se o argumento for uma cadeia; numa variável chamada LIST se o argumento for uma lista e em PRG se o argumento for um programa.

Programa:

```
<<
→ y              @Armazena o argumento na variáveis local y.
<<
CASE             @Começa a estrutura do case.
y TYPE 2 SAME    @Case 1: se o argumento é do tipo cadeia,
THEN y 'STR'     @armazena-o na variável 'STR'
```

```
STO END


y TYPE 5 SAME    @Case 2: se o argumento é do tipo lista,
THEN y 'LIST'    @armazene-o na variável 'LIST'
STO END
y TYPE 8 SAME    @Case 3: se o argumento é do tipo programa,
THEN y 'PRG'     @armazene-o na variável 'PRG'
STO END
END              @Termina a estrutura case.
>>
>>
```

Programa 2:

O case só executa a primeira condição verdadeira, se não encontrar nenhuma executara a condição por defeito.

Exemplo com IF:

```
<< → N
 IF N 0 == THEN "Zero"
 ELSE
      IF N 0 > THEN "Positivo"
      ELSE "Negativo"
      END
 END
>>
```

Exemplo com CASE:

```
<< → S
  << CASE                        @caso
      N 0 == THEN "zero" END     @N=0 então "zero"
      N 0 > THEN "positivo" END @N>0 então "positivo"
      "é negativo"               @condição por defeito
      END
  >>
>>
```

Como funciona a estrutura CASE...END:

Ao executar CASE, se calcula a cláusula teste1. Se o teste é verdadeiro, se executa a cláusula do test1, e a execução salta para END. Se a cláusula teste é falsa, a execução passa para a próxima cláusula teste. A execução na estrutura CASE continua até que todas as cláusulas teste tenham sido avaliadas como falsas. Opcionalmente, pode-se incluir uma cláusula de default, que é executada caso todas as outras cláusulas tenham sido avaliadas como falsas.

4.3.4- O Comando IFT (if-Then-End)

O comando IFT toma dois argumentos: o resultado de um teste no nível 2 e um objecto no nível 1 (a "cláusula verdadeira"). O objecto do nível 1 é executado se o resultado do teste é verdadeiro.

Exemplo: O programa abaixo extrai um número da pilha e retorna a cadeia "POSITIVO" se o número for
positivo:

```
<< 0 > "POSITIVO" IFT >>
```

A função IFTE

A função IFTE toma três argumentos: o resultado de um teste no nível 3, e os objetos dos níveis 2 e 1. O objeto do nível 2 (correspondente a cláusula verdadeira) é executado se o resultado do teste é verdadeiro. Caso contrário, o objeto do nível 1 (a cláusula falsa) é executado.

Exemplo1:

O programa abaixo extrai um número da pilha e retorna a cadeia "POSITIVO" se o número for positivo e "NEGATIVO" caso o número seja negativo:

```
<< 0 > "POSITIVO" "NEGATIVO" IFTE >>
```

Exemplo2: Podemos utilizar a função IFTE dentro de um algébrico: O programa abaixo calcula sin(x)/x se x é diferente de zero. Se x é zero, o programa devolve 1:

```
<< → x 'IFTE( x=0 , S N(x)/x , 1 )' >>
```

ESTRUTURAS ITERATIVAS:

As estruturas iterativas executam uma parte de um programa repetidamente. Existem dois tipos básicos de laços:

- *Para um laço definido,* o programa especifica previamente quantas vezes será executada a cláusula do laço.
- *Em um laço indefinido,* o programa utiliza um teste para determinar se deve executar novamente a cláusula do laço.

Estruturas Iterativas Definidas

Fazem parte das estruturas interactivas definidas as seguintes variações:

- START...NEXT e START...STEP.
- FOR...NEXT e FOR...STEP.

4.3.5- A Estrutura START...NEXT

START...NEXT executa uma parte do programa um determinado número de vezes. Sua sintaxe é:

início fim **START** ***cláusula do laço*** **NEXT**

Exemplo: O programa seguinte cria uma lista que contém dez cópias da cadeia "ABC":

```
<< 1 10 START "ABC" NEXT 10 LIST >>
```

Como funciona START...NEXT:

START toma dois números da pilha (início e fim) e os armazena como valores inicial e final para o contador do laço. Depois, executa a *cláusula do laç*o. NEXT incrementa o contado em 1 e verifica se este valor é menor ou igual ao fim. Se é, executa novamente a *cláusula do laço.*

```
<<
1 #vezes START acção
NEXT
>>
```

4.3.6- A Estrutura START...STEP

START...STEP funciona exactamente da mesma forma que o START...NEXT, excepto que permite especificar um incremento diferente de 1. Sua sintaxe é:

***início fim* START *cláusula do laço incremento* STEP**

Exemplo: O programa seguinte toma um número x da pilha e calcula o quadrado deste número x/3 vezes:

```
<< DUP x << x 1 START x SQ -3 STEP >>
```

Como funciona START...STEP:

START toma dois números da pilha (início e fim) e os armazena como valores inicial e final para o contador do laço. Depois, executa a *cláusula do laç*o. STEP toma o *incremento da pilha* e incrementa o contador com este valor. O incremento pode ser positivo ou negativo. Se é positivo, executa novamente a *cláusula do laço* quanto o contador é menor ou igual ao fim. Se o incremento é negativo, executa o laço quando o contador é maior ou igual ao fim.

4.3.7- A Estrutura FOR...NEXT

Um laço FOR...NEXT executa uma parte de um programa um número especificado de vezes, utilizando uma variável local como contador das iterações. Pode-se utilizar esta variável dentro do laço. Sua sintaxe é:

***início fim* FOR *contador cláusula do laço* NEXT**

Exemplo1: O programa seguinte coloca na pilha os quadrados dos número inteiros de 1 a 10:

```
<< 1 10 FOR j j SQ NEXT >>
```

Exemplo2: O programa seguinte calcula o fatorial de um número da pilha:

```
<< 1 1 ROT FOR j j * NEXT >>
```

Como funciona FOR...NEXT:

FOR toma dois números da pilha (início e fim) e os armazena como valores inicial e final para o contador de iterações, depois cria uma variável local *contador* como contador de iterações.
Depois, se executa a cláusula do laço. NEXT incrementa o *contador* em 1 e verifica se este valor é menor ou igual ao *f*im. Se é, executa novamente a *cláusula do laç*o. Ao sair do laço o contador é apagado, ou seja ele só existe dentro da cláusula do laço.

4.3.8- A Estrutura FOR...STEP

FOR...STEP funciona exactamente da mesma forma que FOR...NEXT, excepto que permite especificar um incremento diferente de 1 ao contador. Sua sintaxe é:

***início fim* FOR *contador cláusula do laço incremento* STEP**

Exemplo: O programa seguinte calcula os quadrados dos inteiros ímpares de 1 a 15:

```
<< 1 21 FOR j j SQ 2 STEP >>
```

Como funciona FOR...STEP:

FOR toma dois números da pilha (início e fim) e os armazena como valores inicial e final para o contador de iterações, depois cria uma variável local *contador* como contador de iterações.
Depois, executa a cláusula do laço. STEP toma o *incremento* da pilha e incrementa o *contador* com este valor, e verifica se o valor do *contador* é menor ou igual ao *f*im. Se é, executa novamente a *cláusula do laç*o. Ao sair do laço o contador é apagado, ou seja ele só existe dentro da cláusula do laço.

Estruturas Iterativas Indefinidas

Fazem parte das estruturas interactivas definidas as seguintes variações:
- DO...UNTIL...END.
- WHILE...REPEAT...END.
A Estrutura DO...UNTIL...END

4.3.9- DO...UNTIL...END:

Executa repetidamente um laço enquanto a cláusula de teste retornar um valor falso. Como se executa primeiramente a cláusula do laço e depois a cláusula do teste, executa-se ao menos uma vez o laço. Sua sintaxe é a seguinte:

DO *cláusula do laço* UNTIL *cláusula de teste* END

Exemplo:

O programa seguinte calcula n + 2n + 3n + ... para um valor de n. O programa para quando a soma exceder 10000, e devolve a soma e o coeficiente de n:

Programa:

```
<<
DUP 1 → n s c @Duplica n e armazena o valor em n e s;
            inicializa o contador c com 1.
<<
DO @Começa a estrutura do. 'c' INCR Incrementa o contador         em  1
e devolve para a pilha o novo valor de c. n * 's' STO+ Calcula c*n, e
soma o produto a s.
UNTIL @Começa a estrutura de teste. s 10000 > Repete o laço       até
que s>10000.
END @Termina a cláusula de teste. s c Coloca na pilha s e         c.
>>
>>
```

Como funciona DO...UNTIL...END:

DO começa a cláusula do laço. UNTIL finaliza a cláusula do laço e começa a cláusula de teste. A cláusula de teste deixa o resultado do teste na pilha. END extrai o resultado deste teste da pilha. Se o valor é zero, executa novamente a cláusula do laço; caso contrário, a execução do programa continua após o END.

```
<< DO processo      @FAZER processo
    UNTIL condição  @ATÉ condição
END                 @Fim
>>
```

4.3.10- A Estrutura WHILE...REPEAT...END:

WHILE...REPEAT...END avalia repetidamente um teste e executa a cláusula do laço se o teste é verdadeiro.
Como a cláusula do teste ocorre antes da cláusula do laço, nunca se executa um laço sem antes verificar se a cláusula do teste é verdadeira. Sua sintaxe é a seguinte:

WHILE *cláusula de teste* REPEAT *cláusula do laço* END

Exemplo: O programa seguinte realiza uma divisão por dois sobre o número que está na pilha repedidamente sempre que o resultado da divisão seja divisível por um número par.

```
<< WHILE DUP 2 MOD 0 == REPEAT 2 / DUP END DROP >>
```

Como funciona WHILE...REPEAT...END:

WHILE executa-se a cláusula do teste e devolve o resultado do teste para a pilha. REPEAT toma os valores da pilha. Se o valor é diferente de zero, continua a execução do laço, caso contrário, a execução do programa continua após o END.

```
<< WHILE condição @ENQUANTO condição
REPEAT processo   @REPETIR processo
END               @Fim
>>
```

4.4- operadores relacionais, secção TEST:

Os operadores relacionais são aqueles que comparam, testam argumentos, devolvendo 1 (True = Verdadeiro) ou um 0 (False = Falso) de acordo com o que é perguntado. Estas funções estão descritas na seguinte tabela:

Função de comparação	Descrição
SAME	Testa se dois objectos são iguais
==	Testa se dois objectos são iguais
≠	Testa se dois objectos são diferentes
<	Testa se o obj2 é maior que o obj1
>	Testa se o obj2 é maior que o obj1
≥	Testa se o obj2 é maior ou igual ao obj1
≤	Testa se o obj2 é menor ou igual ao obj2

4.4.1- Operadores de lógica:

Provas lógicas são aquelas que permitem dar a conhecer a condição entre duas relações.

4.4.2- AND

Esta função permite dar a conhecer se se cumprem simultaneamente duas condições, e a resposta vem representada na tabela verdade:

Condição #1	Condição #2	Resposta
0	0	0
1	0	0
0	1	0
1	1	1

4.4.3- OR

Com este operador vasta que umas das condições seja verdadeira para que seja verdade, ver tabela verdade:

Condição #1	Condição #2	Resposta
0	0	0
1	0	1
0	1	1
1	1	1

4.4.4- XOR

Este operador toma o valor verdadeiro se as duas condições forem diferentes, veja tabela verdade:

Condição #1	Condição #2	Resposta
0	0	0
1	0	1
0	1	1
1	1	0

4.4.5- NOT

Este operador serve para inverter o estado de uma condição; a tabela verdade é a seguinte:

condição	Condição negada
0	1
1	0

4.4.5- Tabela verdade das três operações lógicas:

Prova			**AND**	**OR**	**XOR**
1	1		1	1	0
0	0		0	0	0
1	0		0	1	1

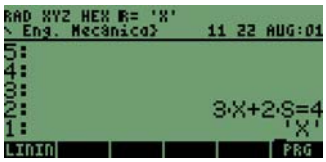
AND: *Este comando equivale ao símbolo (^)*
OR: *Este comando equivale ao símbolo (v)*

4.4.6- FLAGS indicadores de sistema

Aqui vamos falar da utilidade dos flags e dos comandos que permitem alterar ou testar os flags, estes indicadores de sistema permitem ligar ou desligar funções na calculadora, estes são por isso bastante úteis em programação:

SF: Activa o indicador de sistema seleccionado
CF: Desactiva o indicador de sistema seleccionado
FS?: Verifica se um dado indicador esta activo
FC?: Verifica se um dado indicador esta desactivado
FS?C: Verifica se um indicador esta activado, e desactiva-o
FC?C: Verifica se um indicador esta desactivado, e activa-o

LININ:



Este comando verifica se uma equação é linear ou não, esta requer a equação e a variável da equação; neste caso é linear (1-True).

4.5- comandos da sessão TYPE

Estes comandos ajudam a combinar diferentes tipos de objectos.
Antes de entrar com estes comandos deve conhecer bem os diferentes tipos de objectos que existem. A seguinte tabela descreve alguns deles.

Tipo	Descrição	exemplo
0	Número real	-6.023E23
1	Número complexo	(2,5)
2	Texto (String)	“Gripar o motor”
3	Arrays (vector ou matriz)	[1 3], [[2 3][6 4]]
4	Arrays com números complexos	[(2,3)(1,2)]
5	Listas	{“ESTIG” 9547}
6	Nome	‘x’

7	Variável em uso	'i'« FOR i 1+ NEXT»
8	Programa	« X 3 / »
9	Algébrico (equação)	'X+3'
10	Numero Binario	#05B15h
11	Grob (gráficos)	Graphic 131x64
12	Objecto etiquetado	Datos:5354
13	Unidades	25_Km
14	Nome da XLIB	XLIB 543 8
15	Directório	DIR A5B2....END
16	Libraria	Library 1500: FIS2
17	Backup object	Backup MYDIR
18	Função interna	SIN
19	comando interno	CLEAR
20	Internal binary integer	<128d>
21	Numero real estendido	1.23E2
22	Numero complexo estendido	Complexo longo
24	Caractère	®
25	Code Object	Code
26	Dados de libraria	Library Data
28	Números inteiros	25
30	Fontes	Ft8_25: Wilancha

4.5.1- TYPE:

TYPE

Este comando dá-nos o numero do tipo de objecto que esta no nível 1.

4.5.2- VTYPE

Este comando dá-nos o numero do tipo de objecto que se encontra numa determinada variável, neste caso a variável 'PPAR' é uma lista.

4.5.3- →**ARRY** (Arrays)

Este comando permite-nos a construção de arrays (vectores ou matrizes).
Vectores- elementos do vector e posição.
Matrizes- são elementos ordenados em filas e colunas numa lista (#fila #coluna).

4.5.4- →LIST (listas)

2 →LIST

Este comando serve para construir listas, isto é , agrupar elementos de qualquer tipo de objectos, mas para isso é necessário o numero de elementos que deseja agrupar, neste caso da figura agrupamos só 2.

4.5.5- →STR

Este comando converte qualquer elemento em String.

4.5.6- →TAG (tagged object)

Este comando é um etiquetador de valores, primeiro deve ter os valores e depois o nome da etiqueta e accionar o comando.

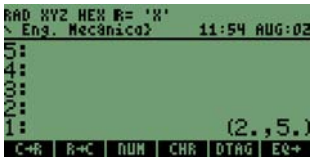
4.5.7- →UNIT (Unidade)

Este comando permite um numero a unidade que de deseja

4.5.8- →OBJ

Este comando faz o inverso dos últimos 5 comandos vistos.

4.5.9- C→R



Este comando decompõe um numero complexo na sua parte imaginaria e real. (funciona para sistemas dimensionais)

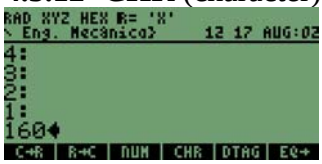
4.5.10- R→C

Este comando faz o contrario do anterior, tendo o valor real e imaginário passa para complexo. (pode aplicar-se em sistemas dimensionais)

4.5.11- NUM

Este comando devolve o numero do primeiro caracter de uma strig, no exemplo acima o primeiro caracter é um E o numero do caracter E é 69. (*ver guia de bolso pag*. 18)

4.5.12- **CHR** (character)

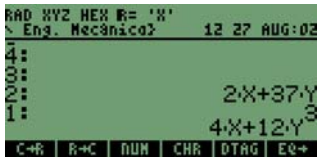


Este comando é o inverso do anterior a partir do numero do caracter dá-nos o caracter.

4.5.13- DTAG

Este comando faz o inverso de →TAG, ou seja devolve o valor de um objecto etiquetado.

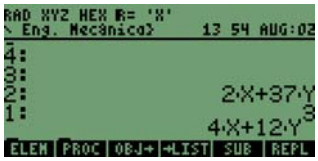
4.5.14- EQ→



Este comando decompõe uma equação a partir do sinal de igual.

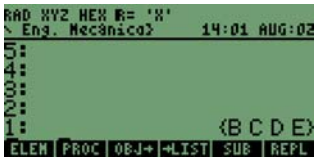
4.6- LIST

Estes comandos tem a finalidade de manipular e processar listas.



4.6.1- SUB

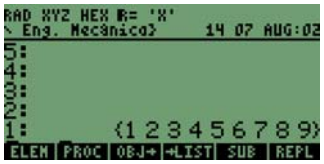
2 5 SUB



Este comando requer um valor inicial e outro final para seleccionar os elementos que se desejam obter de uma lista.

4.6.2- REPL

3 ESTIG REPL



Este comando permite substituir elementos dentro de uma lista, este requer o valor da posição e o novo ou novos elementos.

4.6.3- ELEM (Elementos)

4.6.3.1- GET

2 GET

Este comando permite obter um determinado elementos uma lista, este requer a posição.

4.6.3.2- GETI

2 GETI

Este comando é semelhante ao anterior, mas este permite obter a lista original, e o numero da posição seguinte e o elemento, requer a posição do elemento.

4.6.3.3- PUT

3 ABC PUT

Este comando trocar um elemento da lista por um novo, este precisa da posição e novo elemento.

4.6.3.4- PUTI

3 MAR PUTI

Este comando é similar ao anterior, a diferença é que este dá-nos a posição do elemento seguinte.

4.6.3.5- SIZE

Este elemento dá-nos o numero de elementos que contem uma lista.

4.6.3.6- POS

Este comando indica a posição de um elemento na lista.

4.6.3.7- HEAD

Este comando devolve o primeiro elemento que esta na lista

4.6.3.8- TAIL

Este comando devolve todos os elementos menos o primeiro.

4.6.4 – PROC

4.6.4.1- DOLIST

Este comando executa um programa ou uma função a partir de listas.

Sintaxe:

1- **Listas** a utilizar
2- **Quantide** de listas a utilizar
3- **Programa ou função** que se executara com os valores das listas.
4- DOLIST

(1)- Não se surpreenda os comandos seguintes vão ser explicados posteriormente

Exemplo:

```
« “Insira valores de A” { “{}” {1 2}V} IMPUT OBJ→
   “Insira valores de B” { “{}” {1 2}V} IMPUT OBJ→
2 « 2 ^ SWAP 2 ^ + √ » DOLIST
»
```

4.6.4.2- DOSUBS

Este comando aplica um procedimento sequencial com os elementos de uma lista.

Sintaxe:

1- **Lista** onde se encontram os valores a utilizar
2- **Numero** de elementos a utilizar em cada procedimento
3- **Programa** o função que se efectua com esta sequência
4- DOSUBS

Exemplo:
Neste exemplo desejamos somar três números e eleva-los ao cubo.

```
« “INSIRA A LISTA” { “{}” {1 2}V} INPUT OBJ→
3 « + + 3 ^ »
DOSUBS »
```

Sintaxe 2:

1- **Lista** onde se encontram os elementos
2- **Programa ou função** a efectuar-se em cada elemento
3- DOSUBS

Exemplo:
Neste exemplo vão-se efectuar as seguintes funções Ln(Abs(x*cos(x))) *para n valores de X.*

```
« “Insira valores” { “{}” {1 2}V} INPUT OBJ→
« → X « X X COS * ABS LN »  »
DOSUBS »
```

4.6.4.3- NSUB

Este comando devolve a posição em que se esta efectuando o processo dentro do comando DOSUBS.

4.6.4.4- ENDSUB

Este comando devolve a quantidade de vezes que efectuou o programa por cada valor, e funciona dentro do comando DOSUBS.

4.6.4.5- STREAM

Este comando executa uma função com cada elemento que exita dentro de uma lista.

Sintaxe:

1- **Lista** onde se encontram os valores a utilizar por STREAM
2- **Programa ou função** a executar com os valores
3- **STREAM**

Exemplo:

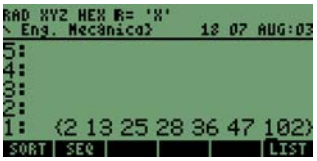
Neste exemplo vão-se efectuar as seguintes funções Ln(Abs(x*cos(x))) *para n valores de X.*

```
« “Insira valores” { “{}” {1 2}V} INPUT OBJ→
« → X « X X COS * ABS LN »  »
STREAM »
```

4.6.4.6- REVLIST

Este comando inverte a posição dos elementos de uma lista.

4.6.4.7- SORT



Este comando ordena os elementos de uma lista numericamente ou alfabeticamente.

4.6.4.8- SEQ

Este comando troca numa variável de uma função ou objecto valores dados um inicial e outro final, variando num incremento, devolve uma lista de valores obtidos.

Sintaxe:

1- **Função ou objecto** que se utiliza para trocar uma determinada variável.
2- **Variável** da função onde se deseja trocar os valores.
3- **Valor inicial** a trocar na função.
4- **Valor final** a trocar na função.
5- **Incremento** dado para cada repetição.
6- SEQ

Exemplo:

Neste exemplo deseja-se substituir só os números pares de uma função, introduzindo a função valor inicial e valor final.

```
« “Insira a função” {“’’” {1 2} ALG} INPUT OBJ→
  “Insira a variável” { “” {1 1} ALG} IMPUT OBJ→
  “Insira valor inicial e final” {“:Vi: :Vf:” {1 5}V} INPUT OBJ→ DTAG
SWAP DTAG DUPDUP 2 / IP 2 * = « 1 + » IFT SWAP 2 SEQ »
```

4.7- GROB

Estes comandos servem para substituir, sobrepor, capturar, criar, animar imagens.

4.7.1- →GROB

Converte um texto a gráfico, requer o tamanho do texto.

Sintaxe:

1- **Objecto** ou texto que se deseja converter a grob
2- **Tamanho** ao qual se quer converter o objecto a gráfico 1 ou 2.
3- GROB

4.7.2- BLANK (espaço em branco)

Cria uma GROB em branco, requer as dimensões.

Sintaxe:

1- **Largura** da grob em numero inteiro binário
2- **Altura** da grob em numero inteiro binário
3- BLANK

4.7.3- GOR

Sobrepõe uma grob sobre outra.

4.7.4- XGOR

Este comando é similar ao anterior, mas este inverte a imagem antes de sobrepo-la.

Sintaxe:

1- **Grob** donde se vai sobrepor uma imagem.
2- **Localização**, pode ser um vector em números inteiros binários.
3- **Grob** que se deseja sobrepor.
4- GOR ou **XGOR**

4.7.5- SUB

Este comando extrai uma porção de uma grob, requer o valor inicial e final.

Sintaxe:

1- **Grob** ou **PICT** em caso de se desejar utilizar a imagem que se encontra em modo gráfico.
2- **Valor inici**al, ponto da origem, vector em números binários.
3- **Valor final**, ponto do final, vector em números binários.
4- SUB

4.7.6- REPL

Substitui ou sobrepõe uma porção de Grob em outro grob ou Pict.

Sintaxe:

1- **Grob** ou **PICT** em caso de se desejar utilizar a imagem que se encontra em modo gráfico.
2- **Ponto** onde se deseja substituir uma grob, vector em números binários.
3- **Grob** que se deseja substituir
4- **REPL**

4.7.7- →LCD

Mostra a grob dado na tela grafica (lcd) em modo texto.

4.7.8- LCD→

Converte a tela de modo texto a grob.

4.7.9- SIZE

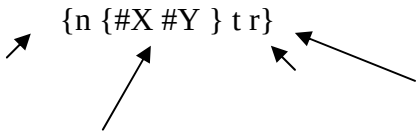
Devolve o tamanho da grob em números binários.

4.7.10- ANIMATE

Comando de animação requer varias grobs para visualizar sequencialmente.

Sintaxe:

1- **Grobs** a utilizar na sequência de animação
2- **Numero** de grobs que se utiliza para a animação ou uma lista da seguinte forma:



numero de grobs

Tempo de cada imagem em segundos

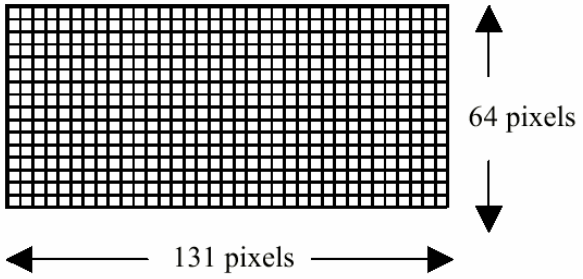
Ciclos de repetição

Posição em números binários

3- ANIMATE

4.8- PICT (picture)

Nesta secção encontra-se todos os comandos de desenho, que são utilizados manualmente para desenhar qualquer coisa. Com estes comandos pode desenhar pode-se desenhar gráficos por meio de programas da mesma forma que se faz manualmente, antes de continuar deve-se conhecer a tela gráfica da HP49G.



Para marcar um ponto na tela existem duas maneiras:

- Números binários:

Antes de começar a calculadora deve encontrar-se na base 10, para trocar o modo binário para a base 10 (DEC) faz-se o seguinte:

Left shift + SYMB → BASE → DEC

Para marcar um ponto na tela é bastante fácil visto que o nosso ponto de origem, encontra-se na parte superior esquerda da tela. Deste ponto podemos marcar qualquer ponto na tela tendo em conta que este (origem) é o ponto (0,0), e os limites da tela serão (130,63).

A HP49G lê este as coordenadas da seguinte forma:

{ #nº de fila #nº de coluna}

Deve ter-se em conta que antes do numero da fila ou da coluna deve meter-se sempre o símbolo # para que seja reconhecido como numero binário e automaticamente a HP49G depois de convertido põe a letra **d** para indicar que esta na base decimal.

- Sistema de coordenadas:

Neste caso a origem encontra-se no centro da tela, esta forma não é muito aconselhável, por eu primeiro é preciso conhecer a escala em que se encontram as coordenadas, este é alterado pelas funções de PLOT, mediante as funções postas para serem desenhadas, esta se ajusta automaticamente, consoante as funções. A forma de introduzir este tipo de coordenadas é:

(valor de X, valor de Y)

4.8.1- PICT

Este comando é parecido com uma variável onde se guarda tudo o que se vê em modo gráfico. Para poder obter ou sacar o gráfico de esta variável pressiona-se PICT no nível 1 da pilha e vê-se PICT, logo o comando RCL (left shift + K).

4.8.2- PDIM

Cria uma pintura em branco com dimensões personalizadas e restaura-o na tela do modo gráfico.

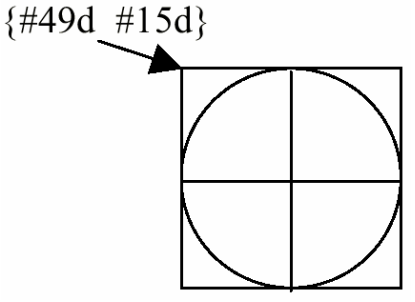
Sintaxe:

1- **Numero binário** valor do comprimento da PICT
2- **Número binário** valor da altura da PICT
3- PDIM

Os comandos seguintes precisam das coordenadas dos pontos a utilizar, estes podem estar em forma binaria ou em coordenadas, conhecendo a escala. Para poder e ver os gráficos realizados por meio destes comandos deverá ir ao modo gráfico pressionando o cursor esquerdo das teclas.

LINE:- desenha uma linha recta, requer dois pontos
TLINE:- desenha uma linha, requer dois pontos
BOX:- desenha um rectângulo, requer dois pontos
ARC:- desenha um arco, requer centro, raio, e os ângulos que determinam o arco. A calculadora deve estar no modo DEG, visto que é este modo que nos interessa para os ângulos.

Exemplo:
No seguinte exemplo desejamos criar um gráfico mediante um programa:



Temos uma circunferência inscrita num quadrado e duas rectas perpendiculares que passam pelo centro da circunferência, o raio da circunferência são 15 pixels, conhecendo o raio e um ponto do gráfico pode desenhar os outros objectos sem necessidade de mais dados.

Ha 5 maneiras diferentes de realizar o exercício depende do modo em que quiser trabalhar os pontos ou coordenadas:

MODO **DEC:** (decimal)
« ERASE { # 0d # 0d }PVIEW
{ # 49d # 15d }{ # 79d # 45d } BOX
{# 64d # 30d } # 21d 0360 ARC

```
 { # 64d # 15d} { # 64d # 45d }TLINE
{ # 49d # 30d }{ # 79d # 30d } LINE
{ } PVIEW
»
```

MODO **HEX:** (hexadecimal)

```
« ERASE { # 0h # 0h } PVIEW
{ # 31h # Fh }{ # 4Fh # 2Dh } BOX
{# 40h # 1Eh } # 15h 0360 ARC
 { # 40h # Fh} { # 40h # 2Dh }TLINE
 { # 31h # 1Eh }{ # 4Fh # 1Eh } LINE
{ } PVIEW
»
```

MODO **OCT:** (octogonal)

```
« ERASE { # 0o # 0o }PVIEW
{ # 61o # 17o }{ # 117o # 55o } BOX
{ # 100o # 36o }# 25o 0 360 ARC
 {# 100o # 17o } {# 100o # 55o } TLINE
{ # 61o # 36o } {# 117o # 36o } LINE
{} PVIEW
»
```

MODO **BIN:** (binario)

```
« ERASE { # 0b # 0b }PVIEW
 { # 110001b# 1111b } {# 1001111b # 101101b} BOX
 { # 1000000b# 11110b } # 10101b 0 360 ARC
{ # 1000000b# 1111b } {# 1000000b # 101101b} TLINE
 { # 110001b # 11110b } {# 1001111b # 11110b } LINE
{ } PVIEW
»
```

A outra maneira de fazer isto é por coordenadas vai depender da escala, primeiro tem de ver a escala e adaptar a escala em vigor, atenção que o tamanho da imagem muda com o comando **PLOT.**

PIXON:- Activa um determinado pixel da tela requer um ponto.
PIXOF:- Desactiva um determinado pixel, requer um ponto
PIX?:- Comprova se um pict esta activado.
PX→C:- Converte as coordenadas pixel (binário) a usuário (sistema de coordenadas).
C→PX:- Converte as coordenadas do usuário a pixel.

Exemplo:

Com este programa vamos utilizar a segunda forma de coordenadas que são (X,Y), o comando **R→C** converte dois números reais a coordenadas, vamos utilizar o comando RAND (RANDOM "aleatório").

```
«
ERASE { #0 #0 } PVIEW 0 20
FOR i RAND i + RAND i + → X Y

«
X Y R→C X NEG Y R→C X Y NEG R→C
X NEG Y NEG R→C Y X R→C Y NEG X R→C Y X NEG
 R→C Y NEG X NEG R→C
»
PIXON PIXON PIXON PIXON PIXON PIXON PIXON PIXON 0.1 STEP
»
```

4.9- CHARS

Estes comandos servem para modificar caracteres dentro de uma (strig) a partir de um programa.

4.9.1- SUB

5 11 SUB

Este comando devolve uma parte da string, temos que dar a posição valores que são inicial e final.

4.9.2- REPL

3 "123" RPL

substitui uma string de caracteres em outra mediante o valor da posição.

4.9.3- POS

"D" POS

Este comando indica a posição do caracter que se encontra na string, neste caso a letra D esta na posição 5.

4.9.3- SIZE

Este comando dá-nos a quantidade de caracteres que tem uma string.

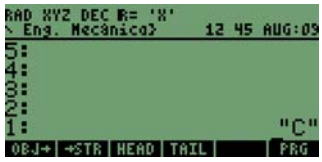
4.9.11- NUM

Este comando devolve o numero do primeiro caracter de uma strig, no exemplo acima o primeiro caracter é um E o numero do caracter E é 69. (*ver guia de bolso pag*. 18)

4.9.12- CHR (character)

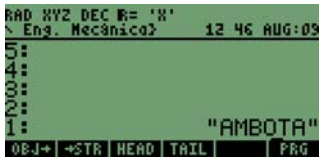
Este comando é o inverso do anterior a partir do numero do caracter dá-nos o caracter.

4.9.4- HEAD



Este comando devolve o primeiro caracter de uma strig.

4.9.5 TAIL



Este comando retira-nos o primeiro caracter de uma strig.

4.10- MODES

Estes comandos servem para fazer trocas no sistema desde um programa permitindo trocar formato numérico(FMT), medida de angulo(ANGLE), indicadores de sistema(FLAG), teclas de usuário(KEIS)e menus de acesso.

4.10.1 - FTM

STD:- Troca para o formato standard.
FIX:- Troca para o formato Fixed, requer o numero antes, de casas 0 < numero FIX < 11.
SCI:- Troca para o formato scientific, requer o numero antes, de casas 0 < numero SCI < 11.
ENG:- Troca para o formato de Engenharia, requer o numero antes, de casas 0 < numero ENG < 11.

4.10.2- ANGLE

DEG:- Troca para o formato DEG
RAD:- Troca para o formato RAD
GRAD:- Troca para o formato GRAD
RECT:- Troca para o formato Rectangular
CYLIN:- Troca para o formato Cilíndrico
SPHERE:- Troca para o formato Esférico

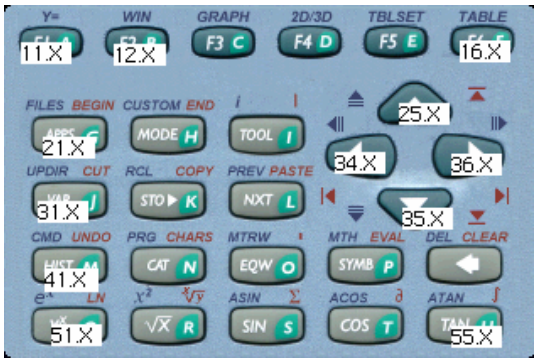
4.10.3- FLAG (indicadores)

STOF:- Restaura a configuração dos indicadores de sistema.
RCLF:- Devolve a configuração dos indicadores ao sistema.

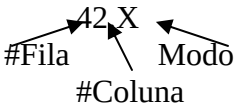
4.10.4- KEYS

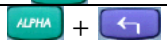
ASN:- Restaura uma tecla de usuário.
STOKEYS:- Restaura a configuração de teclas do usuário.
RCLKEYS:- Devolve a configuração de teclas do usuário.
DELKEYS:- apaga uma tecla do usuário.

Um programa em execução pode tomar uma decisão de acordo com a tecla que é primida, isto é possível porque o programa pode reconhecer qualquer tecla, já que cada botão do teclado esta identificado. Para reconhecer um botão assume-se que o teclado da calculadora é similar a uma matriz, isto é, esta numerado segundo uma certa ordem e tem a seguinte forma:



Formato Key:



Tecla que acciona a função	Tecla da HP49G	Valor
Nenhuma	---	0 ou 1
Right shift		2
Left shift		3
Alpha		4
Alpha + Left shift		5
Alpha + Right shift		6

Se tivermos como identificação de tecla o 42.2 significa que carregamos o Left shift e depois a tecla N e temos deste modo acesso a menu PRG (programa).
Os comandos de usados para reconhecimento de teclado na HP49G são "**KEY**", "**KEYEVAL**" e "**WAIT**".

KEY:- Este comando para identificar qualquer tecla no momento de execução de um programa. A forma como reconhece uma tecla é feita de uma maneira instantânea, a sua função é decidir, o que se executa devolve um numero de identificação da tecla que esta carregar nesse instante, se não esta a carregar em nenhuma devolve o valor 0.

Exemplo:

Com este comando devolve o numero de tecla que se em que se carrega:

```
«
      DO KEY DUP
      UNTIL 0 ≠
      END
»
```

Este program executa a instrução KEY para isso vasta carregar numa tecla, enquanto o valor for 0 o ciclo esta sempre a ser repetido.

KEYEVAL:- Este comando é um comando que acciona a tecla directamente, para isso vasta introduzir o código de tecla seguido de KEYEVAL.

Exemplo:

42.2 KEYEVAL

WAIT:- Este comando é parecido ao KEY mas com a diferença de que este comando precisa de especificar o tempo, só depois de passar tempo é que este comando actua.

Exemplo:

```
«
      2 WAIT
»
```

A vantagem deste comando sobre KEY é que pode utiliza-lo para esperar que carregue numa tecla num determinado período de tempo sim usar nenhuma estrutura interactiva como usamos no KEY, este comando funciona rapidamente se fizermos o seguinte exemplo.

Exemplo 2:

```
«
      0 WAIT
»
```

Uma possível desvantagem da utilização deste comando é quando se prime a tecla ON, com este comando será devolvido 0 em vez do código de tecla da posição correspondente.

4.10.5- MENU

Estes comandos em programação são bastante importantes porque nos permitem trabalhar com os menus da calculadora como vamos ver seguidamente.

MENU:- Devolve o menu chamado.
CST:- Menu personalizado de acesso directo.
TMENU:- Menu temporário.
RCLMENU:- Devolve o numero do menu.

O quadro seguinte tem os códigos dos menus da calculadora bastante úteis em programação:
NUMERO DOS MENUS:

0 LAST MENU	63 PRG NXT MODES
1 CUSTOM	64 PRG NXT MODES FMT
2 VAR	65 PRG NXT MODES ANGLE
3 MTH	66 PRG NXT MODES FLAG
4 MTH VECTR	67 PRG NXT MODES KEYS
5 MTH MATRX	68 PRG NXT MODES MENU
6 MTH MATRX MAKE	69 PRG NXT MODES MISC

7 MTH MATRX NORM	70 PRG MEM
8 MTH MATRX FACTR	71 PRG MEM DIR
9 MTH MATRX COL	72 PRG MEM ARITH
10 MTH MATRX ROW	73 PRG/TOOL STACK
11 MTH LIST	* 74 old menu-based SOLVE
12 MTH HYP	* 75 old menu-based SOLVE ROOT
13 MTH NXT PROB	* 76 old menu-based SOLVE DIFFEQ
14 MTH REAL	* 77 old menu-based SOLVE POLY
15 [MTH] BASE	* 78 old menu-based SOLVE SYS
16 [MTH] BASE NXT LOGIC	* 79 old menu-based SOLVE TVM
17 [MTH] BASE NXT BIT	* 80 old menu-based TVM solver
18 [MTH] BASE NXT BYTE	* 81 old menu-based PLOT
19 MTH NXT FFT	* 82 old menu-based PLOT PTYPE
20 MTH NXT CMPLX	* 83 old menu-based PLOT PPAR
21 MTH NXT CONST	* 84 old menu-based PLOT 3D
22 PRG	* 85 old menu-based PLOT 3D PTYPE
23 PRG BRCH	* 86 old menu-based PLOT 3D VPAR
24 PRG BRCH IF	* 87 old menu-based PLOT STAT
25 PRG BRCH CASE	* 88 old menu-based PLOT STAT PTYPE
26 PRG BRCH START	* 89 old menu-based PLOT STAT SigmaPAR
27 PRG BRCH FOR	* 90 old menu-based PLOT STAT SigmaPAR MODL
28 EDIT	* 91 old menu-based PLOT STAT DATA
29 PRG BRCH DO	* 92 old menu-based PLOT FLAG
* 30 old menu-based solver	* 93 old SYMBOLIC menu
31 PRG BRCH WHILE	94 PRG NXT NXT TIME
32 PRG TEST	95 PRG NXT NXT TIME ALRM
33 PRG TYPE	* 96 old menu-based STAT
34 PRG LIST	* 97 old menu-based STAT DATA
35 PRG LIST ELEM	* 98 old menu-based STAT SigmaPAR
36 PRG LIST PROC	* 99 old menu-based STAT SigmaPAR MODL
37 PRG NXT GROB	*100 old menu-based STAT 1VAR
38 PRG NXT PICT	*101 old menu-based STAT PLOT
39 PRG NXT IN	*102 old menu-based STAT FIT
40 PRG NXT OUT	*103 old menu-based STAT SUMS
41 PRG NXT NXT RUN	*104 old menu-based I/O
42 [CONVERT] UNITS	*105 old menu-based I/O SRVR
43 [CONVERT] UNITS LENG	*106 old menu-based I/O IOPAR
44 [CONVERT] UNITS AREA	*107 old menu-based I/O PRINT
45 [CONVERT] UNITS VOL	*108 old menu-based I/O PRINT PRTPAR
46 [CONVERT] UNITS TIME	*109 old menu-based I/O SERIAL
47 [CONVERT] UNITS SPEED	*110 LIBRARY commands (PVARS, LIBS, DETACH, ATTACH, PINIT)
48 [CONVERT] UNITS NXT MASS	*111 same result as LIBS
49 [CONVERT] UNITS NXT FORCE	*112 same result as LIBS
50 [CONVERT] UNITS NXT ENRG	*113 old EQLIB menu
51 [CONVERT] UNITS NXT POWR	*114 old EQLIB EQLIB menu (empty)
52 [CONVERT] UNITS NXT PRESS	*115 old EQLIB COLIB menu
53 [CONVERT] UNITS NXT TEMP	
54 [CONVERT] UNITS NXT NXT ELEC	
55 [CONVERT] UNITS NXT NXT	

ANGL 56 [CONVERT] UNITS NXT NXT LIGHT 57 [CONVERT] UNITS NXT NXT RAD 58 [CONVERT] UNITS NXT NXT VISC 59 [CONVERT] UNITS TOOLS 60 PRG NXT NXT ERROR IFERR 61 PRG NXT NXT ERROR 62 PRG NXT CHARS	*116 old EQLIB MES menu (empty) *117 old EQLIB UTILS menu Atenção: Os menus que apresentam "*" antes só são acessíveis pelo comando **MENU.**

NOVOS MENUS DA HP49G:

*118 abandoned UNITS TOOLS (same as menu #59) 119 APPS CAS 120 S.SLV 121 EXP&LN 122 TRIG 123 CALC 124 ALG 125 ARITH 126 ARITH POLY 127 ARITH INTEG 128 ARITH MODUL 129 MATRICES 130 CMPLX 131 CONVERT *132 menu-based NUM.SLV *133 menu-based TVM (SOLVR fails if flag -117 is clear) 134 SYMB ARITH *135 abandoned SYMB CONV *136 abandoned SYMB DIFF *137 abandoned SYMB MATRX *138 abandoned SYMB MOD 139 SYMB TRIG *140 abandoned SYMB TRIGC *141 abandoned SYMB UNARY *142 abandoned SYMB BASIC 143 SYMB	*144 bad PRG menu: BRCH is malformed *145 malformed PRG BRCH menu 146 MATRICES CREAT *147 abandoned subset of MATRICES OPER 148 MATRICES FACT *149 ? abandoned MATRICES COL ? *150 ? abandoned MATRICES ROW ? 151 SYMB ALG 152 SYMB CALC 153 SYMB GRAPH 154 SYMB SOLVE 155 SYMB NXT EXPLN 156 MATRICES OPER 157 MATRICES QUADF 158 MATRICES LIN-S 159 MATRICES EIGEN 160 MATRICES NXT VECT 161 TRIG HYP 162 CALC DERIV 163 CALC LIMIT 164 CALC DIFF 165 MATRICES CREAT COL 166 MATRICES CREAT ROW *167 abandoned TIME (same as menu #94) 168-255 (não exixtem)

NUMERO DE MENUS QUE CORRESPONDEM AS BIBLIOTECAS IMBUTIDAS:

256 Hacker's toolkit

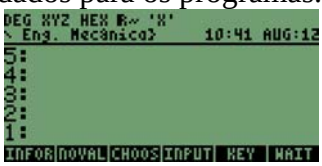
788 CAS version 4.19990717 (new, improved Erable library)
1792 IF/THEN-type structure commands (same as in 48S/G)
2050 Library 2 (48S command set + 15 new ones; see below)
2057 Statistical test functions
2219 Library 171 (48G command set + 3 new ones; see below)
2269 MetaKernel goodies
2270 ADDTOREAL
2289 Stat stuff

NOVOS COMADOS LIB:

107 DRAW3DMATRIX
108 ->KEYTIME
109 KEYTIME->

4.11- IN (Comandos de entrada)

Nesta secção vamos ver todos os comandos os comandos de introdução de dados para os programas.



4.11.1- INFORM

Este comando permite desenhar formulários para a introdução de dados como vamos ver mais a frente, no nível 2 devolve os valores armazenados numa lista, e no nível 1 um valor que pode ser 1 se o formulário for desenhado correctamente ou 0 caso se cancele o formulário, manualmente por erro.

Sintaxe:

1- **Nome** que leva o formulário, este nome deve estar em strig
2- **Label**, são os elementos que existem {{“nome” “ajuda” TIPO DE OBJECTO(S)}{}...n}
3- **Numero de colunas**, #colunas,{#colunas}, {#colunas #espaço}
4- **Valores Reset** estes são os valores que restauram quando se pressiona a opção reset do menu do formulário {VALORES RESET}
5- **Valores iniciais** estes são valores de iniciais {VALORES INICIAIS}
6- **INFORM**

Exemplo:

Este programa realiza um formulário que permite a entrada de uma função, e calcular o valor de X e calcula-lo.

```
« “Introdução de dados”
{{“F(x):” “Introduza a função”9}
{“X:” “Valor para X” 0}}
{1-1}{}{} INFORM DROP OBJ→
DROP X SWAP = SUBST →NUM
“F(x)” →TAG →STR MSGBOX
»
```

4.11.2- NOVAL

Este comando é um campo vazio que é utilizado junto ao comando inform.

4.11.3- CHOOSE:

O comando CHOOSE é utilizado quando necessitamos que o usuário faça uma escolha dentro de um programa. O comando gera um menu de barras, contendo as opções, ou objectos desejados.

Este comando cria um menu de escolha no centro do display que permite a escolha usando o cursor(setas para cima e para baixo), neste comando se seleccionar alguma opção devolve o numero 1 senão devolve 0.

```
« "Fisica by IPB" {
{ " PARALELO" PARALELO }
{" ESTRLA→TRIANGULO " ESTRLA→TRIANGULO }
{" TRIANGULO→ESTRELA " TRIANGULO→ESTRELA} { "
Ajuda" Ajuda }
{ " SAIR" « 2 MENU » } } 1.
CHOOSE DROP EVAL
  »
```

A barra deve ser deslocada até o objecto desejado e em seguida pressionada a tecla ENTER ou OK para
efectuarmos a escolha, ou CANCEL para cancelarmos. Se a função foi cancelada teremos como resposta o valor
0 no stack. Se a função foi confirmada temos o valor 1 como resposta, e na segunda linha do stack, teremos a
opção (objecto) seleccionada.

Interpretação do código:

PARALELO, ESTRLA→TRIANGULO, TRIANGULO→ESTRELA e Ajuda são programas que são chamas a ser executados, estes programas foram feitos a parte, assim é muito mais fácil detectar os erros, este programa foi criado num directório e convertido para libraria, mas temos um pormenor em sair, é isso mesmo além de podermos chamar **programas e funções** também podemos por programas a executar dentro dessa estrutura.

4.11.4- INPUT

Este comando é utilizado para introduzir dados a partir do teclado de uma forma sequencial. É aconselhável utiliza-lo quando se esta trabalhando com poucas variáveis. A síntese que caracteriza este comando para uma variável é a seguinte:

```
«
 “Titulo:” “:Nome_variavel:” INPUT
»
```

NOTA:
É importante que se coloque o nome da variável entre “::” para que seja reconhecida pelo INPUT.

Se quer trabalhar com mais variável a síntese é a seguinte:

```
«
“Titulo”
{“:Nome_variavel_1: :Nome_variavel_2:
:Nome_variavel_3:” {1 0}} INPUT
»
```

Para colocar varias variáveis coloca-se numa lista o nome das mesmas, o ultimo elemento da lista “{1 0}” especifica a fila.

Sintaxe:

1- **Mensagem** que deve estar em strig
2- **Formato** {“valor inicial” {#Fila #Coluna}Modo(s)} o “valor inicial”.
3- INPUT

Exemplo:

Este programa permite receber dados mediante o comando XRECV, escrevendo o nome em que se armazena o arquivo recebido.

```
« “nome do objecto a
 receber por XRECV ” {“”{1 2}α}
INPUT OBJ→ XRECV »
```

4.11.5- KEY

Retorna o resultado de um teste para verificar se alguma tecla está sendo pressionada, em caso afirmativo retorna também a posição da tecla pressionada (linha, coluna).

Exemplo:

```
«
 WHILE KEY NOT REPEAT
“Nenhuma tecla pressionada” 1 DISP END
“A tecla pressionada foi:” SWAP + 1 DISP
3 FREEZE
»
```

4.11.6- WAIT

Este comando interrompe um determinado tempo para isso vasta que pressione uma tecla. (1)

(1)- este comando já foi visado anteriormente.

4.11.7- PROMPT

Este comando é responsável por interromper o programa e mostrar uma mensagem na parte superior do display.

```
  «
{ { "CARTESIANA" RECT } { "POLAR"
CYLIN } { "S.LIN" « IFERR RREF SCROLL
      THEN
"IPB:
  Insira a matriz" DOERR END » }
 { } { "HELP" « "                        "
{ 1. 2 3. 4. 5 6. 7.} DISP
"www.alunos.ipb.pt/~em9547"
MSGBOX » } { "Menu" Fis2} } TMENU
“    FIS2 by: IPB     ” PROMPT
  »
```

4.12- OUT (Comandos de Saída)

Estes comandos servem para visualizar resultados e mensagem, com som de um programa.

4.12.1- PVIEW

Este comando troca para a tela de modo grafico.

Sintaxe:

1- **Tipo** de troca: {}=troca permanente; {#0 #0}=visualiza por um momento.
2- **PVIEW**

4.12.2- TEXT

Troca a tela para modo de texto.

4.12.3- **CLLCD** (clear LCD)

Limpa a tela de modo texto.

4.12.4- **DISP**

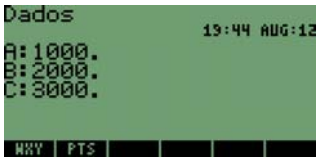
Mostra um objecto numa linha do LCD(2).

Sintaxe:

1- **Objecto** que se visualiza no LCD.
2- **Numero** da linha que se deseja visualizar no LCD (1-7)
3- **DISP**

Exemplo:

```
« "C:" SWAP + SWAP
"B:" SWAP + ROT
"A:" SWAP + UNROT
SWAP CLLCD 5 DISP 4 DISP 3 DISP
"Dados" 1 DISP 0 WAIT DROP
»
```



(2)- Liquid Cristal Display

4.12.5- **FREEZE** (congelar)

Este comando permite congelar determinada área do LCD.

Comando	Área congelada
1 FREEZE	Área do status
2 FREEZE	Área da pilha
3 FREEZE	Área do status + pilha
4 FREEZE	Área do menu

```
« "ABC
DEF
GHI" CLLCD 1 DISP 3 FREEZE
»
```

Aqui temos o funcionamento do comando FREEZE como o LCD tem 7 displays, podemos agrupa-los com o comando FREEZE aqui o nosso 1 DISP esta congelado e abrange a área do status + pilha (3=2+1).

4.12.6- **MSGBOX**

Este comando permite criar caixas de dialogo na calculadora. Estas caixas só tem como objectivo informar algumas mensagens, estas são fechadas primindo a tecla OK.

Para criar uma caixa de mensagem só tem que usar a seguinte síntese:

```
« “Texto” MSGBOX »
```

Exemplo:

```
«
CLLCD “Este comando verifica a capacidade do comando MSGBOX ” MSGBOX
»
```

Este comando dividirá o texto em 5 linha e 15 caracteres cada linha.

```
« "                    "
{ 1. 2 3. 4. 5 6. 7.} DISP
"www.alunos.ipb.pt/~em9547"
MSGBOX »
```

Ora bem apenas com estes dois comandos pode fazer coisas bonitas, como criar “fundos”..., neste caso usei o caracter , mas pode usar qualquer caracter da calculadora, como @,#,& entre muitos outros.

4.12.7- BEEP

Este comando emite um beep com uma frequência e uma duração especificados.

Sintaxe:

1- **Frequência** em Hertz
2- **Tempo** de duração em segundos
3- BEEP

Exemplo:

```
« “som de erro” MSGBOX
1500 0.088 BEEP »

« 550 2 BEEP    @emite um beep de 550 Hz durante 2 segundos
  6500 .5 BEEP   @emite um beep de 6500 Hz durante .5 seg.
»

Podem também ser usadas listas veja o seguinte exemplo:
« { 100 257 600 4000} .58 BEEP »
```

ou

```
« { 100 257 600 4000 } {.2 3 2 .5} BEEP »
```

Atenção:

Deve ter-se em conta que o flag 57 esta activo para que se possa escutar o som.

Agora vou fornecer umas tabelas para que possa compilar as suas próprias musicas para a calculadora.

NOTA	FREQUENCIA EM Hz	PROGRAMA	NOME DO PROGRAMA
LÁ	440	<< 440 T >>	A
SÍ	493,9	<< 493,9 T >>	B
DÓ	523,2	<< 523,2 T >>	C
RÉ	587	<< 587 T >>	D
MÍ	659,2	<< 659,2 T >>	E
FÁ	698,4	<< 698,4 T >>	F
SOL	783,8	<< 783,8 T >>	G
Programa para Executar a Nota		<< SWAP BEEP >>	T

A função SWAP, que troca o valor da linha 1: pelo da linha 2: e vice-versa, do programa T, foi necessária pois da maneira executada as posições de frequência e tempo estão invertidas.

Para minha decepção acabei descobrindo que existiam outras notas intermediárias, os sustenidos e bemóis. Por exemplo, o meio tom intermediário entre o DÓ e o RÉ chama-se DÓ sustenido, ou RÉ bemol. Apesar dos dois nomes se trata da mesma nota, ou seja, fisicamente é a mesma frequência. Isto me levou a criar então mais alguns programas simples:

NOTA	FREQUÊNCIA EM Hz	PROGRAMA	NOME DO PROGRAMA
LÁ#	466,2	<< 466,2 T >>	AS
DÓ#	554	<< 554 T >>	CS
RÉ#	622	<< 622 T >>	DS
FÁ#	739,8	<< 739,8 T >>	FS
SOL#	830,4	<< 830,4 T >>	GS

Existe SÍ e MÍ sustenido e bemol, no entanto pelo fato destas notas já serem meio tons, MÍ sustenido tem a mesma frequência da nota FÁ.

Pronto, já tenho todas as notas ! Infelizmente ainda não, descobri ainda que existe algo chamado oitavas. Oitava quer dizer que a mesma nota pode estar em uma escala superior ou inferior (oitava acima e abaixo). Na prática uma nota uma oitava acima significa o dobro da frequência, e a metade da frequência se uma oitava abaixo. Por exemplo o som de frequência 220 Hz e o som de 880 Hz são uma oitava abaixo e uma acima da nota LÁ 440 Hz. Façamos então mais alguns programas para criarmos uma oitava acima da já criada:

NOTA	FREQUÊNCIA EM Hz	PROGRAMA	NOME DO PROGRAMA
LÁ	880	<< 880 T >>	A2
LÁ#	932,4	<< 932,4 T >>	AS2
SÍ	987,8	<< 987,8 T >>	B2
DÓ	1046,2	<< 1046,2 T >>	C2
DÓ#	1108	<< 1046,2 T >>	CS2
RÉ	1174	<< 1046,2 T >>	D2
RÉ#	1244	<< 1046,2 T >>	DS2
MÍ	1318,4	<< 1318,4 T >>	E2
FÁ	1396,8	<< 1396,8 T >>	F2
FÁ#	1479,6	<< 1479,6 T >>	FS2
SOL	1567,6	<< 1567,6 T >>	G2
SOL#	1660,8	<< 1660,8 T >>	GS2

Conforme a música que você deseja inserir você perceberá a necessidade de mais oitavas e também de uma pausa. Inseri até hoje 50 músicas em minha HP, e para isto precisei inserir apenas 4 oitavas. Para criar uma pausa usei o programa:

`<< WAIT >>` com o nome P

A função WAIT também funciona com um argumento de tempo em segundos, de modo que 3 WAIT faz com que a calculadora aguarde 3 segundos antes de executar a próxima tarefa do programa.

Agora sim PRONTO ! Vamos experimentar o programa criando uma melodia simples que todos já ouviram, para tanto a música (sequência de notas) deve ser colocada dentro de outro programa (TESTE) que será executado:

```
<< ,3 C ,3 D ,3 E ,3 F ,2 F ,2 F ,2 P ,3 C ,3 D ,3 C ,3 D ,2 D ,2 D >>
TESTE
```

De início o tempo das notas (,3 e ,2 no caso) é algo meio de ouvido, você experimenta um valor e vê se o andamento está OK.

Para fazer uma música maior, será necessário copiá-la de algum local, e não tem outro jeito a não ser de uma partitura. Lá fui eu aprender a ler partituras para tocar HP. Uma partitura, aquelas 5 linhas com clave de sol cheias de "bolinhas" e "risquinhos", contém tudo o que é preciso para fazer a música, ou seja, as notas e seus respectivos tempos. As notas são dadas pela posição da "bolinha" nas linhas, e os tempos são dados pelo tipo das "bolinhas". No entanto prefiro não entrar neste assunto por não dominá-lo o suficiente para escrever sobre. Se você já sabe ler partituras com certeza já pegou o espírito da coisa.

Como, usando RPL, a HP só pode emitir uma nota a cada vez, não é possível tocar acordes (que são conjuntos de no mínimo 3 notas). Aconselho então, ao tirar músicas, usar partituras de instrumentos melódicos, como flauta doce.

Experimente agora a seguinte sequência de notas:

```
<< ,2 C ,2 D ,4 E ,4 G ,4 G ,4 E ,8 F ,4 P ,2 C ,2 D ,4 E ,4 G ,4 G ,4
F ,8 E >>
```

4.13- TIME (tempo)

Estes comandos servem para manipular o tempo, data hora na calculadora.

4.13.1- DATE

Este comando devolve a data actual no formato MM.DDAAAA

4.13.2- →DATE

Restaura uma nova data

4.13.3- TIME

Devolve a hora actual no formato HH.MMSS

4.13.4- →TIME

Este comando restaura uma nova hora.

4.13.5- TICKS

Devolve a hora do sistema em ticks.

4.13.6- DATE+

Calcula a data, a x dias da data

4.13.7- DDAYS

Calcula o numero de dias que há entre duas datas.

4.13.7- →HMS

Converte uma hora em HMS a decimais

4.13.8- HMS→

Converte uma hora em decimais para o formato HMS.

4.13.9- HMS+

Soma duas horas em formato HMS.

4.13.10- HMS-

Subtrai dias horas em formato HMS

4.13.11- TSTR

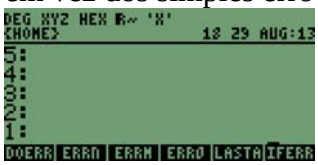
cria uma string com a data e hora

4.13.12- CLKADJ

Ajusta o relógio ao sistema x/8192 seg.

4.14- ERROR

Este comando permite avisar o utilizador que ocorreu um erro e como o evitar em vez dos simples erro da calculadora, este evita que a execução do programa termine.



4.14.1- DOERR

Este comando permite criar uma mensagem de erro na parte central do LCD, quando ocorre um erro. Também este comando chama um erro especifico mediante o seu numero hexadecimal.

Exemplo:

```
«
 "ERRO:
ACESSO NEGADO" DOERR
»
```

Com este erro criamos uma mensagem de alerta, neste caso se estiver a proibir alguém de entrar numa pasta mete um código e com esta mensagem finaliza.

4.14.2- ERRN

Devolve o numero do ultimo erro ocasionado em modo hexadecimal.

4.14.3- ERRM

Devolve a mensagem do ultimo erro em string.

Exemplo:

Para fazer este Exemplo vasta propositar um erro para obter a mensagem.
ERRN: → #201h
ERRM: → "Too Few Arguments"

4.14.4- ERRO

Elimina da memória o ultimo erro gerado.

4.14.5- LASTARG

Devolve no nível 1 o ultimo comando (semelhante ao comando manual UNDO).

4.14.5- IFERR

Este comando verifica se existe algum erro durante a execução de um programa, numa determinada secção.

Sintaxe:
IFERR:- Verifica se existe erro.
THEN:- Então (acção_1).
ELSE:- De outra maneira(acção_2).
END:- Fim (finaliza o comando IFERR).

Exemplo:

```
«                                   "pressione uma tecla" 1 DISP
```

```
IFERR DO 0 WAIT
      UNTIL 54.6 ==
      END
THEN DROP P15 END
“Programa Finalizado”
DOERR
»
```

A parte seleccionada é a parte que requer o comando IFERR verifica se ocorre algum erro durante a execução. O único erro que pode ocorrem sem pressionar a tecla ON(Cancel), assim pode executar-se as instruções em caso de erro. **P15** nome do programa onde deve estar guardado.

4.15- RUN

Execução de um Programa Passo-a-Passo

É mais fácil compreender como funciona um programa se o rodarmos passo-a-passo, observando o efeito
de cada comando. Este procedimento geralmente é utilizado para corrigir erros de programação dentro de um
programa e também para ajudar a entender como programas escritos por outras pessoas funcionam, por isso este
procedimento é chamado de DEBUG.
As operações para executar o DEBUG estão contidas no menu PRG RUN.

4.15.1- Comandos do DEBUG

DBUG: toma como argumento o nome, ou o programa a ser executado passo-a-passo, começando a execução do
programa e depois suspendendo como se fosse executado o comando HALT.
SST: executa o próximo comando do programa suspenso.
SST̄: igual a SST, porém executa subrotinas do programa principal passo-a-passo também.
NEXT: mostra no display o próximo comando a ser executado.
HALT: suspende a execução de um programa na posição do comando HALT.
CONT: retorna a execução de um programa suspenso.
KILL: cancela a execução e o processamento passo-a-passo de todos os programas suspensos.
Exemplo de execução passo-a-passo:
1. Coloque o programa ou o nome do programa no nível 1.
2. Pressione [PRG] [NEXT] RUN DBUG.
3. opcional: pressione NEXT para mostrar o próximo comando.
4. Pressione SST para executar cada passo do programa.
obs:
- para abandonar a execução do programa pressione KILL
- para continuar a executar até o fim do programa normalmente pressione CONT.

Se for necessário executar passo-a-passo um programa a partir de um determinado ponto no seu interior,

basta colocarmos um comando de parada: HALT. Executamos o programa normalmente, e ele irá parar quando
chegar no comando HALT, passando o controle ao DEBUG. A partir daí, podemos executar passo-a-passo o
programa seguindo os itens 2, 3 e 4 descritos acima.

Avançadas:

COMUNICAÇÃO SERIAL:

Antes de iniciarmos a transferência de dados é preciso ajustar os parâmetros de comunicação serial da calculadora:

PORT: selecciona o modo de comunicação (*Wire para HP-PC* ou *Infrared para HP-H*P).

TYPE: seleciona o protocolo de comunicação *(Kermit* ou *XMode*m).

FMT: seleciona o formato dos dados *(ASCII* ou *Binar*y).

XLAT: seleciona o tipo de tradução de caracteres usado no formato ASCII.

CHK: seleciona o metodo de verificação de erros (checksum) usado na transferência.

BAUD: seleciona a velocidade de transferência de dados (1200 2400 4800 ou 9600).

PARITY: seleciona o tipo de método para gerar a paridade durante a comunicação.

OVRW: habilita ou não a gravação (overwrite) de variáveis já existentes.

COMUNICAÇÃO HP-HP

Podemos transferir qualquer tipo de objecto entre duas calculadoras HP através da comunicação por infravermelho. Para a estabelecer a transferência de programas entre duas calculadoras seguimos os seguintes
passos:
1- Entre no menu de comunicação I/O (pressione [ ] [I/O])
2- Seleccione [Transfer...]
3- Configure os parâmetros do mesmo modo nas duas calculadoras: (imprescindível: port=IR) (recomendado: type=Kermit fmt=Bin Xlat=None chk=1)
4- Pressione [NEXT] [OK]
5- Entre novamente no menu de comunicação I/O (pressione [][I/O])
6- Seleccione [Send to HP 48...] na calculadora que irá mandar os programas.

7- Pressione [CHOOS] e seleccione os programas que serão enviados pressionando a tecla [_chk], caso o programa
esteja em outro directório pressione novamente [CHOOS] e seleccione o novo directório.
8- Antes de mandar o programa pressionando [SEND] é necessário entrar em [Get from HP 48] na calculadora que irá receber.
9- Finalmente podemos mandar o programa pressionando [SEND]

COMUNICAÇÃO HP-PC

A transferência entre a calculadora e o computador é praticamente igual ao processo descrito acima, com algumas modificações:
1- Entre no menu de comunicação I/O (pressione [ ] [I/O])
2- Seleccione [Transfer...]
3- Configure os parâmetros do mesmo modo nas duas calculadoras:
(imprescindível port=Wire)
(recomendado type=Kermit fmt=Bin Xlat=None chk=1)
(recomendado baud=9600 parity=None)
4- Pressione [NEXT] [OK]
5- Entre novamente no menu de comunicação I/O (pressione [] [I/O])
6- Seleccione [Send to HP 48...] no caso da calculadora ir mandar os programas.
7- Pressione [CHOOS] e seleccione os programas que serão enviados pressionando a tecla [_chk], caso o programa esteja em outro directório pressione novamente [CHOOS] e seleccione o novo directório.
8- Antes de mandar o programa pressionando [SEND] é necessário preparar o computador para receber.
9- Finalmente podemos mandar o programa pressionando [SEND]
10- No caso em que a calculadora deve receber o programa, deve-se entrar em [Get from HP 48] e através do computador mandar o programa.

KERMIT

Para utilizar a transferência de programas do computador para a calculadora é necessário conhecer os
seguintes comandos do Kermit:
help ou h - mostra o help do programa.
ta porta1 - roda um programa de inicialização automática da porta 1
ta porta2 - roda um programa de inicialização automática da porta 2
set por 1 - utiliza a porta serial 1 para a transferência.
set bau 9600 - configura a velocidade de transmissão.
send [programa] - manda o [programa] para a calculadora.
recv - prepara o computador para receber um programa.
* Para configurar a serial: ta porta1
* Para mandar: send [nome do programa]
* Para receber: recv

COMANDOS DE TRANSFÊRENCIA DE DADOS:

Para escrevermos programas que realizem uma comunicação serial (com o protocolo Kermit ou Xmode) precisamos conhecer os seguintes comandos:

SEND: transfere os objectos para serial cujos nomes estão no stack em forma de lista.

RECV: recebe dados via comunicação serial.

SERVER: entra no modo de servidor.

KGET: recebe um objecto cujo nome é enviado em forma de string.

FINISH: envia uma mensagem de término de transferência de dados ou do modo server.

RECN: recebe um objecto mudando o seu nome para outro que esta indicado no stack.

PKT: envia um comando via serial em forma de string.

KERRM: obtém o código do último erro ocorrido durante a transmissão de dados.

COMANDOS DE COMUNICAÇÃO SERIAL:

Algumas vezes precisamos receber informações de um circuito electrónico ou de algum outro tipo de dispositivo que não permite a comunicação bidireccional, ou seja a calculadora ou só transmite dados ou só recebe dados. Por isso devemos conhecer os seguinte comandos de comunicação serial que nos fornecem um controle maior sobre a serial da calculadora.
OPENIO: "abre" a porta serial, ou seja habilita a recepção de dados e a sua armazenagem num buffer de no máximo 255 caracteres.

CLOSEIO: "fecha" a porta serial , ou seja desabilita a recepção de dados limpando o buffer.

XMIT: envia para a serial da calculadora uma string. (transmissão de dados)

SRECV: obtém uma string de um tamanho especificado do buffer. (recepção de dados)

STIME: especifica o tempo de intervalo entre as tentativas de transferência de dados.

SBRK: envia para a serial um sinal de Break (parada)

BUFLEN: obtém a quantidade de caracteres recebidos e guardados no buffer de comunicação serial e um numero que indica se ocorreu algum tipo de erro durante a recepção dos dados. (0 se ocorreu e 1 se nada ocorreu de errado).

Curiosidades:

Interpretando números seriais HP
O número que aparece atrás das calculadoras HP na forma YYWWC##### pode ser interpretado da seguinte maneira:
YY número de anos após 1960.
WW Semana em que a calculadora foi feita (não é exato)
C Código do país:A =América(USA),B =Brasil,J =Japão,S =Singapura.
Numero serial no período da semana e ano dado

Interpretando números seriais II
Desde 1996, alguns números seriais tomaram a forma CCYWW##### onde CC Código do país: US = USA, SG = Singapura, ID = Indonésia. Y Último digito do ano em que a calculadora foi feita WW Semana em que a calculadora foi feita ##### Numero serial no período da semana e ano dado.

Para a HP49G executar uma determinada função toda vez que a mesma for resetada é só armazenar o programa desejado no arquivo 'STARTUP'

Velocidade HP48 x HP49

Tem se discutido muito sobre a velocidade da HP49G, que muitas vezes se apresenta mais lentamente do que a série 48, isso se deve principalmente aos programas de álgebra adicionados a essa calculadora e ao diferente tratamento dado pela calculadora a números reais (mostrados com adição de ponto ao seu final ex: 3.) ou reais inteiros (sem ponto ao seu final ex:3).

Existem alguns flags que controlam esse modo de cálculo algébrico e tratamento de reais, alguns que pode setar para melhorar um pouco a velocidade são:
-3 SF → função numérica
-79 SF → stack standard (Não sei se realmente vai afectar a velocidade)

-100 CF → passo a passo desligado
-105 SF → modo aproximado ligado**.**

Para calcular o produto escalar entre dois vectores é só coloca-los na pilha operacional e executar DOT. Para calculo de produto vectorial CROSS.

Idioma das mensagem de erro da HP49G

Você pode mudar o idioma das mensagens de sua HP49G para o francês ou espanhol. Para isso é só digitar 1. →LANGUAGE ou 2. →LANGUAGE. Será que algum dia sai uma ROM com tradução para o Português?

Realmente a HP48 é lacrada. Ela é lacrada com 10 rebites de plástico que são prensados depois que ela é toda montada. Assim, para abri-la, você precisa quebrar de algum modo esses rebites. Um método é cortá-los ao meio em pontos escondidos da HP, para que a HP não pareça ser danificada. O ponto fraco desse método é que você irá danificar a estrutura da HP e ela parecerá 'mole' depois do upgrade. O outro método consiste em retirar aquela capa fina de metal que encobre o teclado para cortar a 'cabeça' dos rebites. Esse método é ruim porque a capa de metal não sai sem alguns amassados, mas em compensação a estrutura ficará semi-preservada.

Como compilar programas em sysrpl na 49G

Se você já programa ou pretende começar a programar em sysrpl na 49G saiba que essa máquina já possui em sua ROM (versões 1.10 ou superior) um compilador de sysrpl e ML. Caso sua ROM seja alguma anterior aconselho atualiza-la!
Para compilar você só vai precisar transferir para sua HP o arquivo extable encontrado para download na hpcalc e ao iniciar sua máquina activar as bibliotecas 256, 257 e 258 digitando: { 256. 257. 258. } ATTACH.
Após escrever seu programa em sysrpl em uma string no formato mostrado ao lado é só digitar ASM ou acessar esse comando através de 256. TMENU. Para converter um programa sysrpl em seu código fonte use →S2.
Não se esqueça de sempre adicionar o @ ao final de seus
códigos fonte.

```
"
::
ZEROZERO
$ "TESTE"
$>grob
XYGROBDISP
SetDa1Temp
;
@
"
```

Programa de conversão User-RPL sysRPL

Vou dar um único motivo para que vocês entendam por que ainda não existe um programa de conversão de código User para sysrpl.

Os comandos e funções em sysrpl, diferentemente do USER (na maioria dos casos) são diferentes para cada tipo de objecto, ou seja, se em USER você usa + para somar dois reais, dois complexos, duas strings, dois gráficos, duas listas etc... em sysrpl para somar dois reais você usa %+, para dois complexos C%+, para duas strings &$ e assim por diante. Por esse motivo, para um programa

realmente converter um programa de USER para sys ele deveria verificar o tipo de objecto que esta antes de uma operação e de acordo com esse tipo de objecto trocar o comando/função USER pela equivalente em sysrpl. Dessa maneira um programa que convertesse esses programas deveria conseguir fazer as seguintes alterações, por exemplo:

Programa em USER-RPL	Programa em sys-RPL

`<<`
`2 5 +`
`"123" 1 DISP`
`>>`
	`::`
	`%2`
	`%5`
	`%+`
	`$ "123"`
	`DISPROW1`
	`;`

Perceberam a diferença? Se os programas de conversão, encontrados para download pela Internet, convertesse dessa forma seria super vantajoso já que o tamanho e velocidade de execução cairia. Já convertendo apenas o comando para o seu código sysrpl, a velocidade não se altera significativamente e o tamanho tende a aumentar. A vantagem é que o seu programa fica protegido, e por experiência própria é mais difícil ler e entender um programa convertido esse tipo de programa do que um programa feito directamente em USER ou sysrpl!

· Na HP49G você pode recortar, copiar e colar objectos como no PC. Para isso, durante a edição de objectos, coloque o cursor na posição inicial, clique em [->] BEGIN, mova-se até o final da selecção desejada e clique em [->] END.
Agora para copiar, recortar ou colar use [->] COPY, [->] CUT e [->] PASTE respectivamente.

· Armazenando um programa na variável STARTUP na HP49G, esse programa será executado toda vez que a HP for resetada (warmstart) Armazenando um binário inteiro em TOFF você pode especificar o tempo que sua HP irá desligar se não for operada. Para especificar o tempo, lembre-se que cada segundo é equivalente a #8192d.

· Para aceder ao DIR escondido (onde estão especificações de alarme e USER keys) você deve executar #15777h SYSEVAL EVAL na HP48G séries e #272FEh SYSEVAL EVAL na HP49G.

· Para iniciar a HP49G sem executar o STARTUP e sem iniciar as bibliotecas instaladas, ao iniciar a calculadora mantenha a tecla [<-] (antigo DROP) accionada.

A HP possui dois comandos que podem ser muito úteis para um programa simples de produtória. Os comandos são PLIST (função achada em [MTH] |LIST|).
Esse comando faz a produtória de valores de uma lista.
O outro comando interessante é o SEQ que cria uma lista com valores de acordo com uma função, valor inicial, final e incremento dados... para explicar melhor veja o exemplo:
'X^2' 'X' 0 10 1 SEQ. A HP criará uma lista com os valores {0 1 49162536496481100},ou seja ,uma lista com os valores de X^2variandode 0 a 10 com incrementos de 1.
Mesclando as duas funções e fazendo um programinha simples para multiplicatória você pode fazer:
<< 'X' 3 ROLLD 1 SEQ (PI)LIST >>
Para usar o programa você deve entrar na pilha operacional a função a ser feita a multiplicatória, os valores iniciais e finais respectivamente, e rodar o programa Se quiser variar os incrementos ou variável esteja a vontade para usar e abusar dessas funções...

Fim do RPN?
Primeiro foi a HP38 com sistema operacional somente algébrico agora essas outras. Se for assim logo estaremos sem RPN.:-(Parece que a RPN (Revese Polish Notation) esta fadado a sair de linha, :-(mas é porque a maioria dos usuários não sabe RPN. A coisa se complica ainda mais num mundo com excesso de informação. Vejam que a maioria das pessoas que conheço também não sabe nem mandar uma foto anexada por e-mail. É muita informação e a cada dia mais crescem as necessidades de se aprender novas coisas. Infelizmente o RPN(que gosto muito e é muito bom principalmente para trabalhar com listas) não é fácil de ser usado, precisando ser APRENDIDO pelo usuário. Talvez por isso a HP prefere o Algébrico, que é mais fácil de ser aprendido por estar mais próximo da linguagem que usamos em nossos cadernos e livros. A HP optou por Algébrico, suponho, para satisfazer a maioria dos usuários.

Interpretação de "%%HP: T(2)A(R)F(,)"

O cabeçalho que aparece quando se abre um programa da HP transferido em ASCII para o PC "%%HP: T(2)A(R)F(,)" pode ser interpretado as seguinte forma:
T(2)-modo de tradução 2 (translate do HP-Comm)
A(R)- angulo em radianos

F(,)- Vírgula como separador decimal

Conversão de unidades

Aproveitando a deixa do Desastre no Rio Iguaçu venho relembrar as unidades da HP48/49.
Vazaram da refinaria da Petrobrás 4000000 (4 milhões de litros) de petróleo cru.

Na HP48 ou na HP49 em modo RPN podemos fazer:
4000000 [->][UNITS] |VOL| e pressionar |L| o que vai aparecer: 4000000_l

Ai pressiona-se [<-] |unidade Desejada e se obtém o resultado

Ai estão alguns dados do Desastre:

4000000_l	4000000000_ml
4000000000_cm^3	4000_m^3
4000_st (stereos ---> 1_stereo=1_m^3)	5231,80247726_yd^3 (jardas cubicas)
141258.666886_ft^3 (pés cúbicos)	244094976.379_in^3 (polegadas cubicas)
105668.20943_gal (galões)	25159.2430817_bbl (barris)

Multa estimada em 5.980.487.719,89 Escudos, Aprox. 29830546.98 Euros

Jogos escondidos na HP49

Existem 2 jogos de tetris na hp49.

1º Entre em 2D/3D ([<-] + [F4]) e no campo EQ escreva "HpMad" seguido de [ENTER]. O jogo começa.

2º Válido para roms 1.19-1 e superiores:
- verifica se o modo passo-a-passo (__Step/Step) está desactivado ([MODE] |CAS|)
- Entre em [EQW] e escreva MINEISBETTER
- Seleccione o texto e pressione [F6] (|SIMP|)

Os gráficos do 2 º jogo são melhores.

Sobre o Autor:

Olá o meu nome é Paulino Lourenço, nasci em Algosinho concelho de Mogadouro sou Técnico de Mecânica/frio e climatização, formado pela Escola Profissional de Trancoso, neste momento frequento a Licenciatura em Eng. Mecânica no Instituto Politécnico de Bragança, e espero que este curso lhe possa ser útil.

Eu queria dedicar este curso aos meus pais, se não fossem eles, eu não existia, a Escola Profissional de Trancoso e ao Instituto Politécnico de Bragança, na verdade a escola e a nossa segunda casa.... “pelo que dizem”..

Possíveis contactos:

Em9547@iol.pt
www.alunos.ipb.pt/~em9547
GSM: 964729661

Qualquer duvida ou esclarecimento contacte-me... um abraço

“Apesar dos humanos serem um ser muito inteligente, somos um ser mortal ... a preservação do planeta esta nas nossas mãos ... nestas condições em poucas centenas de anos estamos extintos.”

Paulino Lourenço 2002