Programação em Maple

O maple, embora tenha poucos comandos de programação, é um pacote completo, no qual se pode fazer praticamente tudo em termos de programas de computador. O manual do maple Programming Guide, tem mais de 600 páginas, o que demonstra, que há bastante por estudar.

A propósito, aqui vem uma sugestão de alguém que tem bastantes quilômetros de caminhada na estrada da programação de computadores: (exatamente 38 anos até 2012) É muito difícil, quase impossível programar sem uma boa referência: nenhuma memória humana consegue guardar os milhares de detalhes que a maldita máquina exige. Então vai o conselho: comprem um pendrive específico para o MAPLE. Coloquem nele todos os livros, cartelas, resumos e principalmente o programa MAPLE. Guardem nele todos os programas que vierem a fazer, pois um programa já escrito (e funcionando) sempre é uma excelente referência. E, se fizerem isso, não esqueçam de fazer um backup por semana do pendrive.

Uma diferença importante aqui é o surgimento do comando `print`, cujo formato é `print(v1, v2, ...);` e cuja função é mostrar os valores das variáveis citadas. No maple interativo este comando não é necessário, basta escrever o nome da variável, mas dentro de um programa não dá para fazer assim, logo ele é necessário.

Escrita do programa

Há várias maneiras de fazer isso, mas só vai-se descrever a mais funcional delas:

Usar um editor ASCII convencional (notepad, SC1, Notepad+, entre outros) para criar um arquivo de texto contendo o programa. Dentro do Maple basta emitir o comando `read`, cujo formato é `read "nome-do-arquivo";`. Nessa hora o conteúdo do arquivo é lido e tudo se dá como se tudo o que estiver lá fosse digitado no Maple. (esta facilidade funciona no V e no 15).

Embora o nome do arquivo possa ser qualquer, vamos combinar aqui que ele sempre vai ter a extensão MPL, isto vai ajudar a recuperar programas maple (onde está aquele programa ?????).

Em outras palavras, para escrever o programa P1, o arquivo que o contém sempre vai se chamar P1.MPL.

Tome cuidado, se usar o notepad do windows, que ele tem a mania iritante de incluir a extensão TXT.

Comando Condicional

Trata-se do comando IF, cujo formato é

```
if <condição> then
   comando1
   comando2
   ...
[ else
   comando k
   ... ]
fi;
```

Ele serve para determinar caminhos que dependem de alguma condição. No caso, se <condição> é verdadeira os comandos que seguem o then são executados. Se a condição for falsa e a cláusula else existir (ela é opcional), os comandos que a seguem é que serão executados. Tudo se encerra com a cláusula end if, devidamente seguida por um ponto-e-vírgula.

Relembre que quando um caminho é seguido, o oposto (se existir) é convenientemente saltado, sem ser executado. Veja-se um exemplo:

```
if soma > 250 then
   print('limite ultrapassado')
end if;
```

Outro exemplo:

```
if (x>=0) then
   r1:=sqrt(x);
else
   print('raizes negativas, saindo');
fi;
```

Embora o comando todo possa ser escrito em uma única linha, isso NÃO É UMA BOA IDÉIA. O capricho ao escrever código é que distingue o programador responsável. Lembre do velho ditado válido entre programadores: "Quando eu escrevo um programa, só eu e Deus sabemos o que aquilo faz. Uma semana mais tarde, só Deus..."

Outra possibilidade é um comando completo formado por um if, vários elif, um else e um end if. Este comando é irrelevante, já que pode perfeitamente ser feito com vários if e elses.

Comandos de repetição

Existem dois comandos de repetição. São eles o `for` e o `while`. Ambos resolvem todas as necessidades em termos de desvio e repetição. O comando `for` tem o seguinte formato

```
for <variável> from <v1> to <v2> do
   comando 1,  comando 2,
   ...
od;
```

A variável citada no comando, é inicializada com o valor 1. A seguir, a lista de comandos é executada. A variável é incrementada em 1 e tem o seu valor testado com o valor 2. Se a variável for menor ou igual, a lista de comandos é reexecutada. Quando a variável ultrapassar o valor2, o comando for se encerra.

Se o incremento for diferente de um, a primeira linha do comando passa a ser:

```
for <variável> from <v1> to <v2>
   by <incremento> do ...
```

Assim como nos demais comandos, nada impede que um for seja construído dentro do outro. Daí cada um deles controla sua variável e tudo funciona legal.

Às vezes, a repetição não deve ser feita um número determinado de vezes, e sim deve depender da verificação de uma dada condição. Nesses casos, o for não pode ser usado e sim o comando while, cujo formato é

```
while <condição> do
   comando1    comando2
   ...
od;
```

A <condição> é avaliada. Se ela for verdadeira, a sequencia de comandos é executada. Quando o end do é encontrado, ocorre um desvio para o while, quando a condição é reavaliada, e assim por diante.

Dentro de ambos os laços podem ser incluídos os comandos `break` e `next`. O comando `break` encerra prematuramente o laço, independente de outras condições. Quando break é executado, o próximo comando é o seguinte ao end do.

Quando `next` é executado dentro de um `for`, a variável é incrementada e a condição testada. Quando dentro de um `while`, apenas a condição é reavaliada.

Procedimentos

É o que em outras linguagens recebe o nome de programa. O formato de um procedimento (programa) em maple é

```
<nome-procedimento> := proc()
   comando1    comando2
   ...
end;
```

Para chamar (executar) o programa acima, basta fornecer o comando nome-procedimento();

É comum, o programa receber variáveis externas, que serão conhecidas apenas em tempo de execução. Tais variáveis são citadas dentro do parênteses e são conhecidas pelo programa através de seu valor (passagem por valor). Também é comum que o procedimento precise devolver um determinado valor. Usa-se o comando opcional return <valor>. Se este comando não for citado o procedimento devolve a última coisa que for calculada. Note que esta definição de procedimento é muito parecida com a definição de uma função.

Se a variável de chamada deve ser de um determinado tipo (sob pena de erro), seu nome pode ser citado como (<variavel>::<tipo>). Veja no exemplo:

```
P := proc(x::posint)
   if isprime(x) then
      RETURN true;
   else
      RETURN false;
   fi;
end;
```

Todas as variáveis citadas em um procedimento são locais, a menos que a declaração global <variável> seja emitida dentro do procedimento.

Deve-se ressaltar que um procedimento pode receber outro procedimento como parâmetro em sua chamada. A quantidade de parâmetros passados para um procedimento pode ser obtida através da variável nargs. A sequencia de parâmetros passados para o procedimento é guardada na variável args, que é naturalmente indexada.

Maple possui o comando printf, muito similar ao seu homônimo em C, e que permite fazer impressão formatada.

☞ Para você fazer

Entre no Maple, até ter uma folha de trabalho em branco para começar a sua tarefa. A seguir

1. Localize e ambiente-se com um bom editor de textos ASCII
2. Organize e prepare uma boa estrutura de diretórios (possivelmente usando o seu pendrive) para guardar e recuperar depois os programas maple
3. Escreva, implemente e teste o programa Hello World abaixo

   ```
   #programa alo.mle - Hello World
   restart:
   alo:=proc()
      print("alo mundo"):
   end:

   alo():
   ```

4. Desenvolva junto com o professor o procedimento para cálculo automático do CPF e do CNPJ. Implemente ambos, e teste-os com os seguintes valores:

   ```
   CPF ([1,2,3,4,5,6,7,8,9]);
     [1,2,3,4,5,6,7,8,9,0,9]
   > CNPJ ([1,2,3,4,5,6,7,0,1,1,1,1,0,0]);
     [1,2,3,4,5,6,7,6,1,1,1,1,9,4]
   ```

5. Nos exercícios a seguir, escreva o programa, implemente-o, teste-o e depois que ele estiver funcionando, transcreva aqui o programa coeto:
6. Escreva um programa MAPLE que receba 3 números e devolva a média entre os 2 maiores números. Teste com 33 44 11 e deve dar a resposta 38.5.
7. Escreva um programa MAPLE que receba uma lista de 10 números e devolva o maior número da lista. Teste com [1,5,6,22,11,13,77,7,8,9] e deve dar a resposta 77.
8. Escreva um programa MAPLE que receba um número n e imprima o n-ésimo número primo. (2,3,5,7,11,13,...). Teste com 100 e deve dar 541.

Importante Responda no verso desta folha, ou em uma folha independente, GRAMPEADA a esta.

Obrigado.

- 1 - /