2.8. Comentarios finales y lista de comandos habituales

⋆ ¿Qué tipo de programas se pide en este curso?

Una duda habitual en este curso es si se deben hacer los programas de forma que el ordenador "pregunte al usuario" el input sobre el que se debe operar. La respuesta es negativa, los programas que queremos hacer en este curso son del tipo "función", que reciben un input y devuelven el correspondiente output de manera totalmente automática, es decir, sin "preguntar opciones o valores de input al usuario".

Pongamos un ejemplo, veamos una función que nos calcule el cuadrado de la distancia al origen de un punto con coordenadas (x, y). Obviamente el input son las coordenadas x e y y el output el valor x^2+y^2, de manera que la función que tenemos que programar es la transcripción al lenguaje de programación que estemos empleando (en este caso el `Maxima`) de la función matemática $f(x, y) = x^2 + y^2$. En el programa informático que implementa esta función no nos interesa la parte de cómo se introduce el input, sino sólo las operaciones que hay que hacer con ese input para producir el output. Los programas que haremos en este curso tienen esa estructura.

El objetivo de esta parte del curso es (aprender a- y comenzar a-) generar una biblioteca de funciones que nos resulten útiles en el futuro. La gran ventaja de hacer programas de esta forma (sin tener en cuenta ni el usuario ni cómo se introducen los datos) es que nos permite automatizar la operación realizada, de manera que podemos ejecutarla sobre grandes conjuntos de datos, sin que el "usuario" tenga que estar delante. Para concretar un poco más, si queremos calcular la distancia al origen al cuadrado de un único punto nos puede parecer razonable hacer un programa que "nos pregunte" los valores de x e y; pero supongamos ahora que queremos hacer esta operación sobre un millón de puntos cuyos datos están almacenados en un archivo, en ese caso está claro que esa opción es inadmisible, siendo la única opción viable programar la función $f(x, y)$ como hemos dicho más arriba y mediante otro programa dar la orden de ejecutar esta función sobre el archivo de datos que contiene los (x, y).

⋆ ¿Qué comandos o funciones básicas del Maxima necesitamos conocer?

A medida que progrese el curso iremos viendo los comandos del Maxima que necesitamos. En principio no pensamos que fuese necesario dar una lista de comandos útiles, ya que en Internet (o en la propia ayuda del Maxima) pueden encontrarse muy fácilmente. Sin embargo, la experiencia ha demostrado que sí es recomendable suministrar una mínima lista de *comandos más frecuentes*, que incluimos a continuación:

- Operaciones básicas
  - Para las operaciones básicas se emplea: `+`, `-`, `*`, `/`. Para el producto matricial se emplea `.` (las matrices que multiplicamos deben tener dimensiones compatibles, de lo contrario el Maxima no podrá realizar el producto y nos informará de un error). Para la operación "elevar a una potencia" se emplea el símbolo `^`. Es importante que el símbolo `^` aparezca realmente escrito en el código (para ello es necesario pulsar la tecla 2 veces), de lo contrario lo que hacemos es introducir un super-índice, que con frecuencia no se interpreta como un exponente. Para las funciones matemáticas habituales Maxima cuenta con las funciones habituales `sin`, `cos`, `log`, etc. En la ayuda del Maxima podremos encontrar sin ninguna dificultad cualquiera de estas funciones a medida que nos vayan haciendo falta.

- Asignaciones

  - El Maxima es distinto de la inmensa mayoría de los lenguajes de programación en lo referente a asignar valores a variables. En Maxima, para asignar un valor a una variable se emplea el símbolo "`:`". El símbolo "`=`" en Maxima se emplea para definir ecuaciones. Por ejemplo, si queremos asignar el valor 3 a la variable `x` hacemos `x:3;`. Si queremos que la variable `EQ` contenga la ecuación `LHS=RHS` hacemos `eq:LHS=RHS`.
  - Aparte de asignar valores a variables, una operación que emplearemos constantemente es **definir funciones**. Para definir funciones en Maxima se emplea el símbolo `:=`. Por ejemplo la función $f(x,y) = x^2 + y^2$ de la que hablábamos arriba, en Maxima se define como

    ```
    f(x, y) := x^2 + y^2;
    ```

    De esta forma $f(x,y)$ queda definida como una función de x e y, de manera que dados dos valores a y b, la instrucción `f(a, b);` nos devuelve el esperado valor $a^2 + b^2$.
  - El comando `block`:

    El caso más habitual en este curso es que las funciones que tendremos que programar sean considerablemente más complicadas que la anterior, de tal forma que para su correcta programación necesitaremos dar varios "pasos" antes de llegar al resultado esperado. Normalmente el resultado de cada uno de estos "pasos" intermedios de programación deberá guardarse en alguna "variable auxiliar", de modo que podamos emplearlo posteriormente. Al definir variables auxiliares en una función determinada es muy importante evitar que estas variables entren en conflicto con las variables definidas en otras funciones, o en otras partes del programa en el que estamos trabajando. La manera más sencilla de hacer esto es definir estas variables auxiliares como *variables locales* dentro de la función en la que estamos trabajando. Las variables locales de cada función sólo existen dentro de esa función, de modo que no afectan a las variables definidas fuera de la función (aunque se llamen igual). Todos los lenguajes de programación tienen implementada una manera de definir funciones con variables locales, en Maxima esto se hace empleando el comando `block`. En la ayuda del Maxima se encuentra la sintaxis de este comando, y en los problemas resueltos en este curso virtual hay muchos ejemplos de uso del comando `block`.

- Bucles

  En programación los bucles son una de las instrucciones más habituales y útiles. En Maxima para hacer un bucle hacemos, en general, lo siguiente:

  `for` índice `:` valor inicial `thru` valor final `step` incremento `do` instrucciones`;`

  Por ejemplo, el siguiente bucle escribe los primeros 50 múltiplos de 3:

  ```
  for i : 1 thru 50 step 1 do print(3*i);
  ```

  El mismo resultado se obtiene con este otro bucle

  ```
  for i : 3 thru 150 step 3 do print(i);
  ```

- Listas

  Con mucha frecuencia trabajaremos con listas de objetos (que podrán ser números, matrices, ecuaciones, vectores, cadenas de texto, gráficas, otras listas, etc, etc. etc.). La instrucción para definir una lista es:

`makelist(` elemento, índice, valor inicial, valor final, incremento `);`

Por ejemplo, esta instrucción genera una lista que contiene los números pares de 0 a 10:

```
makelist(i, i, 0, 10, 2);
```

Equivalentemente lo mismo se consigue con

```
makelist(2*i, i, 0, 5, 1);
```

⋆ ¿Cómo estar seguro de que lo estoy haciendo bien?

Pongamos que he definido la función que pedía el enunciado del problema, y he comprobado con algunos ejemplos que funciona correctamente. ¿Cómo podemos estar seguros de que la función pedida está bien programada?

Si estamos en esta situación hay dos cosas que podemos verificar:

- Los ejemplos que estoy considerando ¿cubren todos los casos posibles?
  Un motivo de error típico es que los ejemplos sobre los que operamos no son suficientemente generales, conviene revisar esto para evitar errores.
- Otro motivo de error muy habitual es que la función programada se apoya en variables globales (definidas fuera de la función), de modo que en la sesión de trabajo del programador (donde estas variables globales están definidas) la función parece operar de manera correcta, sin embargo, cuando enviamos esta función al profesor y este la ejecuta en una sesión en la que estas variables globales no están definidas, el programa ya no funciona.

  Para evitar esto una buena costumbre es la siguiente: En primer lugar se debe evitar a toda costa el uso de variables globales en la definición de ninguna función. Por último, para comprobar si la función que hemos programado funciona correctamente debemos **abrir una sesión nueva con Maxima**, y comprobarlo partiendo de cero. Si, inadvertidamente, nuestras funciones se apoyaban en variables globales, al ejecutarlas en una sesión nueva de Maxima (donde esas variables globales no están definidas) nos daremos cuenta del error.

Con lo que hemos visto hasta ahora debería ser suficiente para empezar a trabajar con el Maxima. En los problemas propuestos como ejemplo iremos viendo multitud de nuevas instrucciones útiles.