1 Aritmética

Cuando una instrucción de MAPLE es muy larga y no cabe en una sóla línea, se puede continuar en la línea siguiente pulsando MAYÚSCULAS-ENTRAR. Si sólo se utiliza ENTRAR, MAPLE interpreta que debe evaluar la línea, y se encuentra que falta el signo `;`, dando un error. En MAPLE, el signo `%` es una abreviatura para el último cálculo realizado. Análogamente, `%%` y `%%%`.

1.1 Aritmética entera El tipo de datos más sencillo que maneja MAPLE es el de enteros, con las operaciones habituales: `+`, `-`, `*`, `^`, `!`:

```
> 20!+1;
```

$$2432902008176640001$$

Cociente y resto de la división euclídea: `irem()`, `iquo()`.

```
> irem(13,4), iquo(13,4);
```

$$1, 3$$

1.2 Aritmética racional Los números racionales se manejan como cociente de dos enteros (con denominador positivo), siempre en forma irreducible.

1.3 Aritmética decimal MAPLE también puede hacer cálculos con números decimales: aunque, si los datos de entrada son simbólicos, elegirá la respuesta exacta. Se pueden aproximar decimalmente mediante la función `evalf()`:

```
> 1+1/2;
```

$$\frac{3}{2}$$

```
> evalf(%);
```

$$1.500000000$$

Los números decimales son 'contagiosos'. Si una expresión contiene un número decimal, MAPLE realiza todos los cálculos con números decimales.

1.4 Aritmética compleja También se pueden manejar números complejos. El signo que MAPLE interpreta como la unidad imaginaria i es `I`:

```
> (1-I)/(3-2*I);
```

$$\frac{5}{13} - \frac{1}{13}\,I$$

1.5 Aritmética modular MAPLE también puede realizar cálculos en aritmética modular; con la notación habitual:

```
> 125+634 mod 7;
```

$$3$$

Se debe sustituir el signo `^` por el signo `&^`.

1.6 Aritmética con números especiales MAPLE también puede trabajar simbólicamente con cantidades irracionales. Naturalmente, estas cantidades no se pueden expresar como un número con inifinitas cifras decimales, algunos ejemplos son: `sqrt(2)`, `Pi`, `exp(1)`, etc.

1.7 Aritmética booleana Hay otro tipo de constantes en MAPLE: `true` y `false`, las constantes booleanas. Podemos forzar una evaluación booleana con la función `evalb()`

2 Asignación

En MAPLE, el operador que asigna un valor a un nombre es `:=`. Es importante no confundir el operador de asignación `:=` con el signo de ecuación `=`. La operación contraria de la asignación `:=` es 'variable', por ejemplo `x:='x';`.

3 Cálculos polinómicos básicos

MAPLE considera en general que un producto indicado es más simple que una suma de téminos, no aplica la ley distributiva por defecto. Para aplicar la ley distributiva, se usa `expand()`. La instrucción simétrica de `expand()` es `factor()`, que también sirve se pueden sacar factores comunes de una expresión racional:

```
> expr_racional:=(x^3-y^3)/(x^4-y^4);
```

$$expr\_racional := \frac{x^3 - y^3}{x^4 - y^4}$$

```
> factor(expr_racional);
```

$$\frac{y^2 + y\,x + x^2}{(y + x)\,(x^2 + y^2)}$$

Véase `?simplify()`.

4 Sucesiones, listas, conjuntos

4.1 Sucesiones Una sucesión es una estructura consistente en un número finito de expresiones separadas por comas. Por ejemplo:

```
> s2:=seq(x^i,i=2..4),y;
```

$$s2 := x^2, x^3, x^4, y$$

4.2 Listas Uuna lista es una estructura de datos que se define escribiendo sus elementos entre corchetes, separados por comas (puede no tener ningún elemento).

```
> l1:=[a,0,[1,2],sin(x)/(x^3+exp(x))];
```

$$l1 := [a, 0, [1, 2], \frac{\sin(x)}{x^3 + e^x}]$$

Y se manipulan igual que las sucesiones:

```
> l1[1], op(1,l1);
```

$$a, a$$

```
> l1[1..2];
```

$$[a, 0]$$

```
> nops(l1);
```

$$4$$

```
> l3:=[op(l1),t];
```

$$l3 := [a, 0, [1, 2], \frac{\sin(x)}{x^3 + e^x}, t]$$

4.3 Conjuntos Un conjunto no tiene elementos repetidos, y el orden en que aparecen no tiene por qué ser respetado.

```
> X:={1,2,3,t,u};
```

$$X := \{1, 2, 3, t, u\}$$

```
> Y:={1,2,v,w,5,6,7};
```

$$Y := \{1, 2, 5, 6, 7, w, v\}$$

```
> X union Y;
```

$$\{1, 2, 3, 5, 6, 7, t, u, w, v\}$$

Véanse las páginas de ayuda de `?union`, `?intersect`, `?member`.

5 Manipulación de expresiones

Las funciones `op()`, `nops()`, y `subsop()` son básicas a la hora de partir expresiones, seleccionar partes, borrar o cambiar elementos, etc. La función `nops()` devuelve el número de partes (operandos) que tiene una *expresión* cualquiera de MAPLE. La función `op()` da la *sucesión* de elementos que componen una expresión. La función `subsop()` cambia partes (operandos) de una expresión. Es importante no confundir las funciones `subsop()` y `subs()`. Véase la ayuda para más detalles.

```
> op([1,2,3,4]);
```

$$1, 2, 3, 4$$

```
> subsop(1=z,x^2), subsop(1=NULL,[1,2,3]);
```

$$z^2, [2, 3]$$

6 Tablas y matrices

6.1 Tablas Operamos con un ejemplo:

```
> recta:=table([punto=[0,0,1], vector=[2,2,1]]);
```

$$recta := \mathrm{table}([punto = [0, 0, 1], vector = [2, 2, 1]])$$

La sintaxis general de table es la siguiente:
`table([`⟨*ecuaciones de valores iniciales*⟩`])`

Debido a las especiales reglas de evaluación de las tablas y las matrices, cuando se quiere examinar una tabla se debe usar la función `op()` (o `print()`):

```
> op(recta);
```

$$\text{table}([\mathit{punto} = [0,\ 0,\ 1],\ \mathit{vector} = [2,\ 2,\ 1]])$$

Cuando se quiere acceder a un elemento concreto de la tabla, se usa una sintaxis semejante a las listas:

```
> recta[vector];
```

$$[2,\ 2,\ 1]$$

6.2 Matrices Las formas más habituales de usar las matrices son:

```
> A:=array(1..3,1..3,[[u,v,w],[1,2,3],[4,5,6]]);
```

$$A := \begin{bmatrix} u & v & w \\ 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

```
> C:=array([[1],[2],[3]]);
```

$$C := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

```
> evalm(A &*C);
```

$$\begin{bmatrix} u + 2\,v + 3\,w \\ 14 \\ 32 \end{bmatrix}$$

El signo de multiplicación de matrices es `&*`, y que para efectuar el cálculo es necesaria la función `evalm()`.

7 paquetes

El modo de activar una función determinada se encuentra en la ayuda de la función. Una referencia de las funciones conocidas originalmente y que no necesitan activarse, se puede encontrar

Por ejemplo, la función determinante, no es una función conocida originalmente, sino que se encuentra en la librería de Algebra Lineal `linalg`:

```
> with(linalg):
```

Ahora podemos usar `det()`:

```
> det(A);
```

$$-3\,u + 6\,v - 3\,w$$

La lista completa de los paquetes de Maple se encuentra en la página de ayuda `?index,packages`.

8 Programación

Usaremos, en general, procedimientos. Los procedimientos en Maple se definen como

`proc(`⟨*argumentos*⟩`)`
⟨*variables locales y/o globales*⟩
⟨*instrucciones*⟩
`end proc;`

Veamos un ejemplo

```
> f:=proc(x)
>    if type(x,even) then
>       return 1;
>    else
>       return -1;
>    end if;
> end proc:
> f(2), f(3);
```

$$1,\ -1$$

Las funciones también son útiles a veces:

```
> map(x->x^2,[1,2,3,4]);
```

$$[1,\ 4,\ 9,\ 16]$$

8.1 `if...then...else...end if` La construcción completa es:

`if` ⟨*condición*$_1$⟩ `then`
⟨*acciones*$_1$⟩
`elif` ⟨*condición*$_2$⟩ `then`
⟨*acciones*$_2$⟩
⋮
`else`
⟨*acciones*$_n$⟩
`end if;`

Lo único obligado de esta construcción es la primera y la última línea. Puede no haber `elif` o `else`, o puede haber varios `elif`, pero sólo un `else`.

8.2 `while...do...end do` El bucle `while` tiene una sintaxis del tipo:

`while` ⟨*condición*⟩ `do`
⟨*acciones*⟩
`end do;`

El significado de este bloque es que Maple repita las acciones hasta que la condición sea verdadera. En particular, puede ocurrir que las acciones no se ejecuten nunca.

8.3 `for...do...end do` El bloque `for` tiene la sintaxis siguiente:

`for` ⟨*índice*⟩ `from` ⟨*inicial*⟩ `by` ⟨*diferencia*⟩ `to` ⟨*final*⟩ `do`
⟨*acciones*⟩
`end do;`

La parte `from` *inicial*, si se omite, tomará valor 1. Lo mismo ocurre con la parte `by` *diferencia*. El significado es que se repiten las *acciones* para cada valor del *índice*, que es un nombre de variable, entre *inicial* y *final* con incrementos de *diferencia*. Una forma especial y muy útil es la siguiente:

`for` ⟨*índice*⟩ `in` ⟨*expresión*⟩ `do`
⟨*acciones*⟩
`end do;`

Instrucciones básicas de Maple.

Departamento de Álgebra

Este tríptico contiene unas brevísimas instrucciones de Maple. No son una visión exhaustiva, ni un manual de referencia. Este tríptico acompaña las sesiones introductorias de Maple y son, más bien, una lista con funciones que se usan muy frecuentemente.