0.4

Conceptos básicos de programación

Variables y Funciones

```
Clear["Global`*"]
```

Observe y ejecute las siguientes asignaciones de datos a variables:

```
a=Cos[Pi]

a=3 Sin[Pi/2]-2;

a

a=7*3+5

a=Log[2]

a=4-b

a=(3<1)

a="Esto es un texto"

a=Plot[Sin[x],{x,-Pi,Pi}];
```

Observe la diferencia entre las dos formas de asignación de datos a variables en los dos siguientes bloques:

```
x=2;
a=x;
x=3;
a

x=2;
a:=x;
x=3;
a
```

A continuación se muestra cómo se define una función con Mathematica mediante algunos ejemplos:

```
f[x_]:=x^3

f[2]

f[x_]:=Cos[x]

f[Pi]
```

Para que la definición sea correcta, el símbolo x debe ser una variable y por tanto no debe tener asignado valor alguno. Obsérvense la salidas del siguiente ejemplo:

```
x=3

f[x_]:=x^2
```

```
f[x]
```

```
f'[x]
```

y ahora las de éste otro:

```
Clear[x]
```

```
f[x]
```

```
f'[x]
```

¿cuál es la diferencia?.

Análogamente se definen y evaluan funciones de varias varibles como se muestra a continuación:

```
f[x_,y_]:=x^2+y^2
```

```
f[2,3]
```

Expresiones lógicas. Órdenes condicionales.

Las siguientes son expresiones lógicas construidas con los conectivos y operadores lógicos que Mathematica manipula. Los conectivos lógicos son:

==	igualdad
! =	desigualdad
<	menor que
>	mayor que
<=	menor o igual que
>=	mayor o igual que

y los operadores lógicos son:

&&	o And[,]	conjunción " y " ,
||	u Or[,]	disyunción " o " ,
Xor[,]		disyunción exclusiva ,
!	o Not[ ]	negación.

```
a=3
```

```
a==5
```

```
a!=5
```

```
a<5
```

```
a<=5
```

```
a>5
```

```
a>=5
```

```
b=7
```

```
a<=5 && b>=8
```

```
a<=5 || b>=8
```

```
Xor[a<=5,b>=5]
```

```
Not[a<=5]
```

Observemos ahora como actúan las órdenes condicionales

If[condición, proceso1, proceso2]

y

Which[condición1, proceso1, condición2, proceso2, ...]

Para ello, ejecute los siguientes bloques de órdenes

```
a=2;
b=3;
If[(a<3) && (b==4), Print[a];Print[b],
                    Print["Falso"]]

If[(a<=4) || (b>=5), Print["a=",a];Print["b=",b],
                     Print["Falso"]]
```

Compruebe que la orden If actúa del siguiente modo: si la condición es cierta se realiza el proceso1 y, en caso contrario, se realiza el proceso2. Asimismo, ejecute y observe este otro grupo de órdenes:

```
f[x_]:=Which[0<=x<=1, x^2, 1<=x<=2, 2-x^2]

f[0.5]

f[1.5]

f[3]
```

En este último caso no se ha generado salida ya que f[x] no está definida para x=3. Pero la siguiente función está definida en todo R.

```
f[x_]:=Which[0<=x<=1, x^2, 1<=x<=2, 2-x^2, True, 0]

f[3]
```

Bucles

Las órdenes más importantes que definen bucles, o procesos repetitivos, son For, Do y While, cuyas estructuras son del tipo:

For[contador=inicio, condición, paso, proceso]

Do[proceso, {contador, incio, fin, paso}]

While[condición, proceso]

Como ejemplo en el que se muestre la diferencia de estructuras nos planteamos el problema de escribir los 9 primeros números naturales y sus cubos me- diante cada una de estas sentencias.

```
For[i=1,i<=9,i++,Print[i,"   ",i^3]]

Do[Print[i,"    ",i^3],{i,1,9}]

i=1;
While[i<=9, Print[i,"    ",i^3];i=i+1]
```

En el siguiente siguiente ejemplo observe cómo actuán las variables i, llamada contador, y s, llamada acumulador. Se trata del cálculo de la suma de los 20 primeros números naturales pares, del 2 al 40.

```
s=0;
Do[s=s+2*i, {i,1,20}]
s
```

Iteradores Sum y Product

La anterior suma se podría haber calculado directamente utilizando la orden

Sum[expresión, {contador, inicio, fin, paso}]

que en nuestro caso se traduce en:

```
Sum[2*i,{i,20}]
```

Si en lugar de la suma se deseara el producto se utilizaría la sentencia

Product[expresión, {contador, inicio, fin, paso}]

es decir, el producto de los 20 primeros números naturales pares es:

```
Product[2*i,{i,20}]
```

Análogamente, la suma y el producto de los cuadrados de los múltiplos de 3 entre 6 y 27 es:

```
Sum[i,{i,6,27,2}]
```

```
Product[i,{i,6,17,2}]
```

Listas

Una lista es un conjunto de datos cualesquiera. Los siguientes ejemplos asignan listas a variables, efectúan operaciones con listas o evalúan funciones en listas.

```
a={1,2,3}
```

```
a^2
```

```
b={-1,0,1}
```

```
a+b
```

```
Sin[a]
```

```
%//N
```

```
E^b
```

La orden

Length[lista]

calcula el número de componentes de la lista.

```
Length[a]
```

Una lista puede tener como componentes otras listas. Por ejemplo:

```
a={{1,2,3},{-1,-2,-3},{2,3,1}}

a^2
```

En el caso en el que todas las sublistas de una lista sean de la misma longitud, se puede considerar la lista como una tabla en la que cada fila está formada por las componentes de cada sublista y puede visualizarse en esta forma con la orden

TableForm[a]

```
TableForm[a]
```

La orden Table genera una lista de elementos. Observe su estructura en los siguientes ejemplos:

```
Table[i,{i,1,3}]

Table[i*j, {i,-1,2},{j,2}]

Table[i+j,{i,1,3},{j,2,6,2}]
```

La orden

Dimensions[lista]

genera una lista con las dimensiones de la lista dada y las sublistas que la componen, en el caso de que existan.

```
a={-1,3,5};
Dimensions[a]

a={{1,2},{3,5},{-1,1}};
Dimensions[a]
```

Un elemento de una lista se indica con el nombre de la lista y la posición que ocupa indicada entre dobles corchetes. Por ejemplo el segundo elemento de la lista a se escribe:

```
a[[2]]

b[[1]]
```

Si los elementos de una lista son sublistas, la expresión

a[[i,j]]

indica la componente j-ésima de la sublista i-ésima.

```
a[[2,2]]

a[[3,1]]
```

Por último, si se desea añadir una nueva componente a una lista se escribirá la orden

AppendTo[lista, elemento]

que añade el elemento cmo última componente a la lista. Observe esta orden en los siguientes ejemplos:

```
m=Table[i*j,{j,3},{i,1,7,2}]

v=Table[i^3,{i,-2,8,3}]

AppendTo[m,v]

AppendTo[v,1000]
```

Ejercicios

1.- Calcule, usando la orden Do, la suma de los números comprendidos entre 1890 y ds que sean múltiplos de 5. Resolver el mismo problema usando la orden Sum.

2.- Genere, y guarde en la variable pol, una lista de 10 polinomios tales que el que ocupa la posición i-ésima sea x^i+d1 x+d2. Efectúe su suma y su producto con las órdenes Sum y Product. Vuelva a calcular la suma y el producto utilizando acumuladores y observe que se obtiene el mismo resultado (escriba el producto en la base canónica del espacio correspondiente, no expresado como producto de factores).

3.- Sea v un vector de n componentes. Se define la longitud de v como la norma euclídea del mismo, es decir, la raíz cuadrada de la suma de los cuadrados de sus componentes. Escriba el vector cuyas componentes son los dígitos de su D.N.I. y calcular su longitud.

4.- Use las órdenes Do e If para programar el siguiente problema: se desea calcular el producto de las componentes del vector del problema anterior que no sean nulas. Compruebe que se obtiene el mismo resultado cuando en el vector v se suprimen las componentes nulas y se calcula el producto de las restantes con la orden Product. Aplíquelo todo de nuevo al vector v1={1,0,1,0,1,0,dm,0}.