Programación imperativa en Maxima

Este tema es una introducción a la programación imperativa con el sistema de cálculo simbólico Maxima.

En el curso se usará wxMaxima que se puede descargar aquí.

Maxima como calculadora

```
(%i1) 2 * 5 + 6 - (100 + 3);
(%o1) - 87
(%i2) 8/2; 8/3;
(%o2) 4
      8
(%o3) -
      3
(%i4) float(8/3);
(%o4) 2.666666666666666
```

Variables y asignaciones

```
(%i1) a : 8$
(%i2) b : 2*a;
(%o2) 16
(%i3) a+b;
(%o3) 24
(%i4) a : a+1$
(%i5) a;
(%o5) 9
(%i6) [a,b] : [2,5]$
(%i7) a+2*b;
(%o7) 12
```

Bloques de instrucciones

```
(%i1) (a:2, b:5, c:3, d : a+2*b-c);
(%o1) 9
(%i2) b;
(%o2) 5
(%i3) x:3$
(%i4) block([x,y], x:2, y:5, z:x+y);
(%o4) 7
(%i5) x;
(%o5) 3
(%i6) y;
(%o6) y
(%i7) z;
(%o7) 7
```

Definición de funciones

Ejercicio 1. Definir la función `suma` tal que `suma(x,y)` es la suma de x e y. Por ejemplo,

```
(%i1) suma(2,3);
(%o1) 5
```

Solución

```
suma(x,y) := x+y$
```

Escritura y lectura

Ejercicio 2. Definir el procedimiento `suma` que lea dos números y escriba su suma. Por ejemplo,

```
(%i1) suma()$
Escribe el primer número
2$
Escribe el segundo número
3$
La suma es 5
```

Solución

```
suma() := block([a,b],
  a : read("Escribe el primer número"),
  b : read("Escribe el segundo número"),
  print("La suma es:",a+b))$
```

La estructura condicional

Condicionales simples

Ejercicio 3. Definir, usando condicionales, la función `maximo` tal que `maximo(x,y)` es el máximo de x e y. Por ejemplo,

```
(%i1) maximo(2,5);
(%o1) 5
(%i2) maximo(2,1);
(%o2) 2
```

Solución

```
maximo(x,y) :=
  if x > y
     then x
     else y$
```

Condicionales múltiples

Ejercicio 4. Definir la función `signo` tal que `signo(x)` es el signo de x. Por ejemplo,

```
(%i1) signo(5);
(%o1) 1
(%i2) signo(-7);
(%o2) - 1
(%i3) signo(0);
(%o3) 0
```

Solución

```
signo(x) := if     x > 0 then  1
            elseif x < 0 then -1
            else              0$
```

Estructuras iterativas

Bucles mientras

Ejercicio 5. Definir, con un bucle while, la función `sumaImpares` tal que `sumaImpares(n)` es la suma de los n primeros números impares. Por ejemplo,

```
(%i1) sumaImpares(3)
(%o1) 9
(%i2) sumaImpares(4)
(%o2) 16
```

Solución

```
sumaImpares(n) := block([s:0, k:0],
  while k < n do
    (s : s + 2*k + 1,
     k : k + 1),
  s)$
```

Bucle hasta

Ejercicio 6. Definir la función `mayorExponente` tal que `mayorExponente(a,n)` es el mayor k tal que aˆk divide a n. Por ejemplo,

```
(%i1) mayorExponente(2,40);
(%o1) 3
```

Solución

```
mayorExponente(a,n) := block([k:0],
  unless (mod(n,a) # 0) do
    ( r : r/a,
      k : k+1),
  k)$
```

Bucle para

Ejercicio 7. Definir, por iteración con for, la función `fact` tal que `fact(n)` es el factorial de n. Por ejemplo,

```
(%i1) fact 4
(%o1) 24
```

1ª solución

```
fact(n) := block([f:1, k],
  for k from 1 thru n do f : f*k,
  f)$
```

2ª solución

```
fact2(n) := block([f:1,k],
  for k from n thru 1 step -1 do f : f*k,
  f)$
```

Bucle para sobre listas

Ejercicio 8. Definir, por iteración, la función `suma` tal que `suma(xs)` es a suma de los números de la lista xs. Por ejemplo,

```
(%i1) suma([3,2,5])
(%o1) 10
```

Solución

```
suma(xs) := block([r:0,x],
  for x in xs do (r:x+r),
  r)$
```

Recursión

Ejercicio 9. Definir, por recursión, la función `fact` tal que `factR(n)` es el factorial de n. Por ejemplo,

```
(%i1) fact 4
(%o1) 24
```

Solución

```
fact(n) :=
  if n = 0 then 1
  else     n*fact(n-1)$
```