Programación: Ciclos en Wolfram Mathematica

Objetivos. Aprender a definir funciones en Wolfram Mathematica y usar los ciclos `For` y `While`.

Requisitos. Expresiones numéricas, operaciones lógicas, listas, definición de funciones.

Primer ejemplo de definición de una función

Ejemplo 1. Definimos una función de un argumento real x que calcule $x^2 - 7x + 5$:

```
f[x_] := x ^ 2 - 7 * x + 5
```

Comprobación:

```
f[2]
```
(*debe regresar* -5)

Segundo ejemplo de definición de una función

Ejemplo 2. Escribamos una función de dos argumentos p y q que calcule y regrese la lista de dos raíces de la ecuación $x^2 - 2px + q = 0$. Por ejemplo, para $p = 1$, $q = -35$ la ecuación es

$$x^2 - 2x - 35 = 0,$$

y la función tendrá que regresar la lista de dos elementos: -5 y 7.

Primera solución (breve).

```
SolveQuadrEq1[p_, q_] :=
    { p - Sqrt[p ^ 2 - q], p + Sqrt[p ^ 2 - q] }
```

Después de escribir este programa haga la siguiente comprobación:

```
SolveQuadrEq1[1, -35]
```
(*debe regresar* `{-5, 7}`)

Segunda solución (con un bloque y variables locales). Para conocer algunos elementos del lenguaje Wolfram Mathematica, escribamos otra solución del mismo problema:

```
SolveQuadrEq2[p_, q_] :=
    Module[{r, x1, x2},
        r = Sqrt[p ^ 2 - q];
        x1 = p - r; x2 = p + r;
        {x1, x2}]
```

La palabra `Module` sirve para unir las operaciones en un grupo y para declarar las variables locales `r`, `x1` y `x2`. La última expresión escrita en `Module` es el resultado que regresa la función. En nuestro caso es la lista `{x1, x2}`.

Ejemplo de uso del ciclo For

Ejemplo: ListOfCubes. Vamos a escribir una función de un argumento entero positivo n que construya y regrese la lista de los números k^3, donde k corre de 1 hasta n. Por ejemplo, para $n = 4$ la función debe regresar la lista $1, 8, 27, 64$.

Se sugiere no sólo leer las siguientes soluciones, sino probarlas, es decir, escribir y ejecutar en Wolfram Mathematica.

Primera solución. Para añadir un elemento a una lista, se usa la función AppendTo:

```
a = {4, 7, -2}
AppendTo[a, 5]
a
```

Ahora es fácil escribir la función que resuelva el problema ListOfCubes. Primero creamos una lista *vacía* `a` (es decir, una lista de longitud `0`). Luego organizamos un ciclo `For`. Denotamos por `k` al *contador del ciclo* llamado también la *variable del ciclo*. Esta variable va a tomar los valores de `1` a `n`. En cada iteración del ciclo agregamos k^3 a la lista `a`:

```
ListOfCubes1[n_] := Module[{k, a},
    a = {};
    For[k = 1, k <= n, k++, AppendTo[a, k ^ 3]];
    a]
```

Más precisamente, el último valor de la variable `k` es $n + 1$, pero con $k = n + 1$ ya no se cumple la *condición del ciclo* `k <= n`, y el *cuerpo del ciclo* `AppendTo[a, k ^ 3]` ya no se ejecuta. Después de escribir la función hay que hacer una comprobación:

```
ListOfCubes1[4]
```

Debe regresar la lista `{1,8,27,64}`.

Segunda solución. Primero creamos una lista *nula* de longitud `n` y luego la rellenamos con elementos correctos.

```
ListOfCubes2[n_] := Module[{k, a},
  a = Table[0, {n}];
  For[k = 1, k <= n, k++, a[[k]] = k ^ 3];
  a]
```

No olviden hacer la comprobación. Notemos que los elementos iniciales de la lista `a` se reemplazan dentro del ciclo por elementos nuevos, por eso en lugar de `0` podría ser otro valor inicial. En otras palabras, en vez del comando `a = Table[0, {n}]` podría ser `a = Table[777, {n}]`.

Problemas para resolver con el ciclo For

Se recomienda resolver al menos dos problemas de este grupo.

1. Problema SumList (1 % de la calificación parcial).
Escriba una función de un argumento a, donde se supone que a es una lista, que calcule y regrese la suma de los elementos de la lista a. Puede usar el siguiente esquema y cambiar ... por expresiones apropiadas:

```
SumList[a_] :=
    Module[{n, s, j},
        n = Length[a]; s = ...;
        For[j = 1, j <= ..., j++, s += ...];
        s]
```

Comprobación: `SumList[{-1,8,2,5}]` debe regresar 14.

2. Problema Fact (1 % de la calificación parcial).
Escriba una función de un argumento n (se supone que n es entero no negativo) que calcule el factorial del número n. Puede usar el siguiente esquema:

```
Fact[n_] :=
    Module[{k, result},
        result = ...;
        For[k = 1, ..., ..., ...];
        result]
```

Al escribir la función haga la comprobación. Por ejemplo, `Fact[5]` debe regresar `120`.

3. Problema Powers (1 %).
Escriba una función `Powers[a_, n_]` que construya y regrese la lista de las potencias $a^0, \ldots, a^n$. Por ejemplo, `Powers[3, 4]` debe regresar la lista `{1, 3, 9, 27, 81}`.

4. Problema ListOfFactors (2 %).
Escriba una función de un argumento n (se supone que n es entero positivo) que construya la lista de todos los divisores enteros positivos de n. Por ejemplo, para $n = 12$ la función debe regresar la lista de los números $1, 2, 3, 4, 6, 12$.

Sugerencia: use la función `Mod`. Para comprender qué hace la función `Mod` pruebe los comandos `Mod[73, 20]` y `Mod[22, 5]`, también puede consultar la Ayuda (Help).

5. Problema Fib (3 %).
Escriba una función de un argumento n (se supone que $n \geq 2$) que construya y regrese la lista de los primeros n números de Fibonacci. Se supone que $n \geq 2$. Los primeros dos números de Fibonacci son 0 y 1, y luego cada número es la suma de los dos anteriores. Por ejemplo, `Fib[7]` debe regresar la lista `{0, 1, 1, 2, 3, 5, 8}`.

Ejemplo del uso del ciclo While

Ejemplo: SqrtInt. Vamos a escribir una función de un argumento n (se supone que n es entero y positivo) que calcula el máximo número entero k que cumple la desigualdad $k^2 \leq n$. Por ejemplo, para $n = 19$ el resultado correcto será 4.

Solución.

```
SqrtInt1[n_] :=
    Module[{k = 0},
        While[(k + 1) ^ 2 <= n, k++];
        k]
```

Notemos que cuando se termina el ciclo, la condición del ciclo ya no se cumple, esto es, $(k+1)^2 > n$. Pero para el valor anterior de k esta condición todavía era válida, así que $k^2 \leq n$. Esto muestra que después de la terminación del ciclo la variable k guarda la respuesta. Comprobación:

```
{SqrtInt1[24], SqrtInt1[25], SqrtInt1[26]}
```

La misma función se puede escribir en una linea:

```
SqrtInt1[n_] := Module[{k = 0}, While[(k + 1) ^ 2 <= n, k++]; k]
```

Problemas para resolver con el ciclo While

Se recomienda resolver uno o dos problemas de este grupo.

6. Problema FirstFactor (2 %). Escriba una función de un argumento entero n (puede suponer que $n \geq 2$) que calcule el mínimo divisor de n distinto de 1. Por ejemplo, para $n = 15$ el resultado correcto es 3.

7. Problema FactTo (2 %).
Escriba una función de un argumento m (se supone que $m \geq 1$) que construya y regrese la lista de los factoriales menores o iguales a m, en el orden natural. Por ejemplo, para $m = 30$ la función debe devolver la lista de los números $1, 2, 6, 24$. Indicación: no usar la función que calcula el factorial de un número.

8. Problema Gcd (3 %).
Escriba una función de dos argumentos a y b (se supone que $a, b \geq 0$) que calcule y regrese el máximo común divisor de a y b usando el *algoritmo de Euclides*. Por ejemplo, para $a = 15$ y $b = 6$ la función debe regresar el número 3. Para $a = 0$ y $b = 7$ la función debe regresar 7.