

Programa

Un **programa** es una secuencia finita de **instrucciones**.

- Declaración de variables

TIPO NOMBRE;

- Asignación

VARIABLE = EXPRESIÓN;

- Condicional

if (CONDICIÓN) { PROG1 } else { PROG2 }

- Ciclo

- ***while (CONDICIÓN) { PROG1 }***
- ***For(ITERADOR) {PROG1}***

PARA CALCULAR 9!

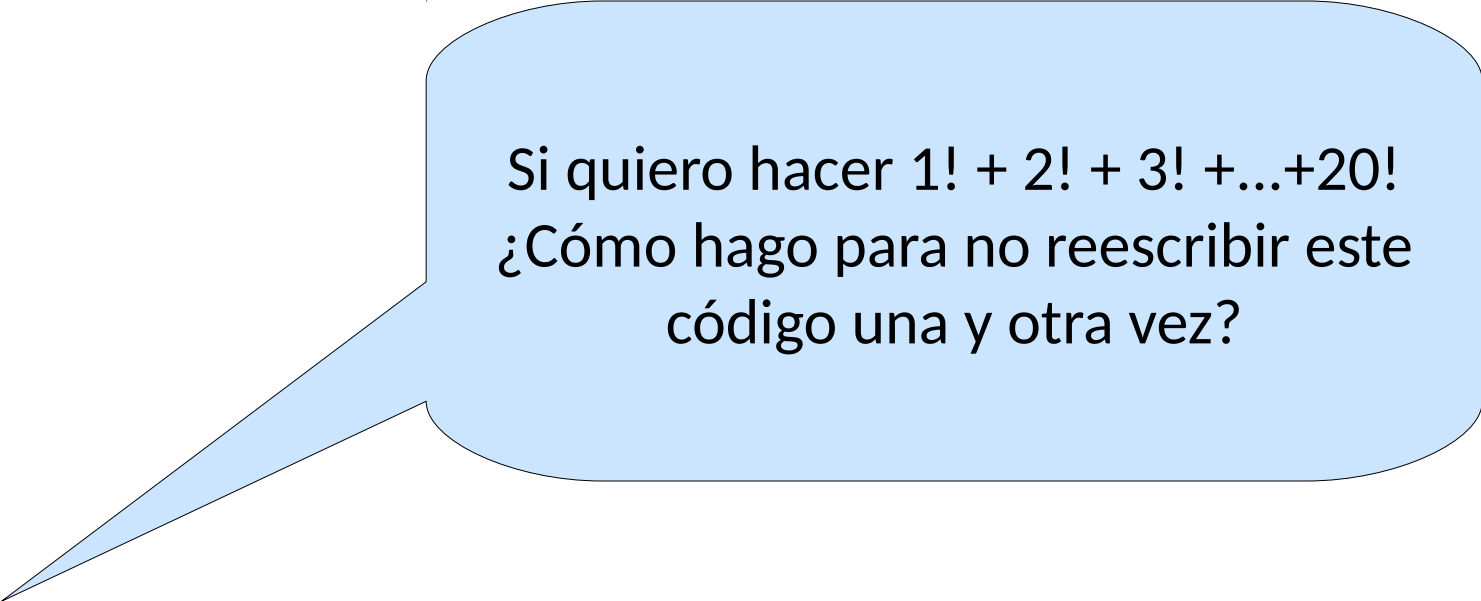
```
Y = 1;  
for i = 1:9  
    Y = Y * i;  
end
```

PARA CALCULAR 14!

```
Y = 1;  
for i = 1:14  
    Y = Y * i;  
end
```

PARA CALCULAR 10!

```
Y = 1;  
for i = 1:10  
    Y = Y * i;  
end
```



Si quiero hacer $1! + 2! + 3! + \dots + 20!$
¿Cómo hago para no reescribir este
código una y otra vez?

Vamos a usar las funciones

Resultado: **Código modular.**

- Más claro para los humanos.
- Más fácil de actualizar.

(Ej: ¿Qué pasa si ahora no quiero multiplicar todos los números del 1 al n sino todos los números pares del 1 al n ?)

Función

Una **función** es una unidad de código que **aísla** una parte de un cómputo. Es un programa dentro de un programa.

- Permite dividir un problema en **problemas más simples**.
- Permite **ordenar** conceptualmente el código para que sea más fácil de entender.
- Permite **reutilizar** soluciones a problemas pequeños en la solución de problemas mayores.

Función

Estos son los **argumentos** de la función (uno o más)

```
function y = factorial(n)
y = 1;
for i = 1 : n
    y = y * i;
end
```

Ahora que tengo definida la función **factorial**, puedo usarla en otra parte de mi código para construir nuevas expresiones.

Ejemplo:

```
X = zeros(1,10);
for i=1:length(X)
    X(i) = factorial(i);
end
```

¿Qué pasa si ahora nos fijamos en el Workspace si están las variables `i` o `n`? ¿O si las intentamos utilizar?

Cada ejecución de una función tiene su **propio espacio de memoria**, como si fuera un programa separado.

`n`, `i` son **alcanzables** dentro de `factorial`, pero no fuera.

Otro ejemplo

Aproximación de la derivada de una función

```
function y = forward(f,x,i,h)
% agarra f, h, x como columna; devuelve
% aproximacion de la derivada parcial iesima en x
% con paso h
    Id= eye(length(x));
    ei = Id(:,i);
    y = (f(x + h*ei) - f(x))/h;
```

Más de un argumento

Buena práctica: comentar el código. De esa forma, cuando lo use en un tiempo, sé lo que hace la función. Esas líneas con % no son ejecutadas.

Repaso de la clase de hoy

- Modularidad del código: funciones y procedimientos.
- Alcance (scope) de variables.