

Programa

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

Ejemplo:

Ingredientes: 15 huevos, 600 gramos de harina, 600 gramos de azucar

- 1.- Mientras no estén espumosos, batir los huevos junto con el azúcar,
- 2.- agregar la harina en forma envolvente sin batir,
- 3.- batir suavemente,
- 4.- colocar en el horno a 180 grados,
- 5.- si le clavo un cuchillo y sale húmedo, entonces ir a 4.-
- 6.- retirar del horno,
- 7.- mientras no esté frío, esperar
- 8.- desmoldar y servir

Instrucción

Una **instrucción** es una operación que:

- transforma los datos (el *estado*), o bien
- **modifica el flujo de ejecución.**

- 1.- **Mientras no estén espumosos**, batir los huevos junto con el azúcar,
- 2.- agregar la harina en forma envolvente sin batir,
- 3.- batir suavemente,
- 4.- colocar en el horno a 180 grados,
- 5.- **si le clavo un cuchillo y sale húmedo**, entonces ir a 4.-
- 6.- retirar del horno,
- 7.- **mientras no esté frío**, esperar
- 8.- desmoldar y servir

Variable

Una **variable** es un **nombre** que denota la **dirección** de una celda en la memoria, en la cual se almacena un **valor**.

En esa celda de memoria es posible:

- **leer** el valor almacenado, y
- **escribir** un valor nuevo, que reemplace al anterior.

Ejemplo en Octave:

```
x = 10;           // Asigno el valor 10 en la variable x.
```

Estado

Se denomina **estado** al valor de todas las variables de un programa en un punto de su ejecución.

**Es una “foto” de la memoria
en un momento determinado.**

Asignación

VARIABLE = EXPRESIÓN ;

Almacena el valor de la *EXPRESIÓN* en la dirección en memoria denotada por *VARIABLE*.

Ejemplos:

a = 1;

b[a] = 3;

b[b[1]] = 9* b[a] + a;

b[b[1]/2] = 123;

Condicional

if (CONDICIÓN) { PROG1 }

CONDICIÓN es una expresión que arroja resultado verdadero o falso;
PROG1 es un programa.

PROG1 se ejecuta **si y sólo si** *CONDICIÓN* arroja valor verdadero.

Ejemplo:

```
if 1 > 5  
    '1 es mayor que 5.'  
end
```

```
if 1 < 5  
    '1 es menor que 5.'  
end
```

Condicional

if (CONDICIÓN) { PROG1 } else { PROG2 }

CONDICIÓN es una expresión que arroja V o F.

PROG1 y *PROG2* son programas.

PROG1 se ejecuta **sii** *CONDICIÓN* arroja valor verdadero.

PROG2 se ejecuta **sii** *CONDICIÓN* arroja valor falso.

Ejemplo:

```
if 1 > 5
    '1 es mayor que 5.'
else
    '1 es menor que 5.'
end
```

Condicional – Otro ejemplo

¿Qué imprime por pantalla este código?

```
a = 10;  
b = [100, 1];  
  
if (b[1] / (a * 10) == b[2])  
    b[1] = b[1] - 1;  
    b[2] = b[2] * 5;  
else  
    b[1] = b[1] + 1;  
    b[2] = b[2] * 3;  
end
```

Para mayor claridad,
indentar cada bloque de código.

a

b

Ciclo

while (CONDICIÓN) { PROG1 }

CONDICIÓN es una expresión que arroja resultado verdadero o falso;
PROG1 es un programa.

PROG1 se ejecuta una y otra vez, **mientras** *CONDICIÓN* siga arrojando valor verdadero.

Ejemplo:

```
i = 1;  
while (i < 3)  
    i  
    i = i + 1;  
end
```

Ciclo – Otro ejemplo

while (CONDICIÓN) { PROG1 }

Ejemplo:

```
n = 1;  
i = 2;  
while (i <= 9)  
    n = n * i;  
    i = i + 1;  
end  
n
```

Ciclo – Otro ejemplo

while (CONDICIÓN) { PROG1 }

Ejemplo:

```
i = 0;
while (i < 3)
    if (floor(i/2) - i/2 == 0) {
        i
        'es par'
    else
        i
        'es impar'
    i = i + 1;
end
```

Ciclos anidados

¿Qué imprime por pantalla este código?

Ejemplo:

```
A = zeros(5);  
fil = 1;  
while (fil <= 5)  
    col = 1;  
    while (col <= fil)  
        A(fil,col) = 1;  
        col = col + 1;  
    end  
    fil = fil + 1;  
end
```

For

for (*ITERADOR*) { *PROG1* }

Ejemplo:

```
x = [1:10];  
for i=1:5  
    x(i)=x(i)*2;  
end  
  
x
```

For: otro ejemplo

for (*ITERADOR*) { *PROG1* }

Ejemplo:

```
x = [1:10];  
for i=1:2:10  
    x(i)=x(i)*2;  
end
```

x

Programa

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

- Declaración de variables

TIPO NOMBRE;

- Asignación

VARIABLE = EXPRESIÓN;

- Condicional

if (CONDICIÓN) { PROG1 } else { PROG2 }

- Ciclo

while (CONDICIÓN) { PROG1 }

Repaso de la clase de hoy

- Condicional: if.. ; if..else..
- Ciclo: while..; ciclos anidados. For.

Próximos temas

- Modularidad del código: funciones.