

**Problema 1**  
Escribir un programa que muestre la siguiente tabla:

Nº raíz  
2    Nº  
3    Nº  
...  
30  Nº

- Algoritmo**
1. inicializar variable n con valor 2
  2. repetir hasta que  $n > 30$ :
    - escribir n y  $\sqrt{n}$
    - sumar 1 a n

- alternativamente**
- 1.inicializar variable n con valor 2
  2. repetir mientras  $n \leq 30$ :
    - escribir n y  $\sqrt{n}$
    - sumar 1 a n

**Programa**

```
U.println("Nº raíz");

//inicializar variable n con valor 2
int n = 2;

//repetir mientras n<=30
while( n <= 30 )
{
    //escribir n y √n
    U.println(n + " " + Math.sqrt(n));

    //sumar 1 a n
    n = n + 1;
}
```

**Instrucción while**

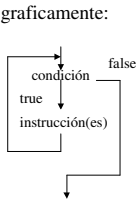
**Sintaxis**

```
while( condición )
    instrucción;
```

**o, si se necesita repetir varias instrucciones**

```
while(condición)
{
    instrucciones;
}
```

**Semántica**  
Mientras condición se cumpla (sea true)  
ejecutar instrucción(es)



Ejecución del programa

	instrucción	result		instrucción	result
1º	int n=2;	n=2	...		
2º	while(n<=30)	true	85º	n=n+1;	n=30
3º	println(...)	2 1.4	86º	while(n<=30)	true
4º	n=n+1;	n=3	87º	println(...)	30 5.4
5º	while(n<=30)	true	88º	n=n+1;	n=31
6º	println(...)	3 1.7	89º	while(n<=30)	false
7º	n=n+1;	n=4	90º	fin	

## clase4: instrucción while

**¿Qué sucede si la condición es falsa la primera vez?**

Ejemplo:

```
int maximo=U.readInt("n° máximo?");
int n = 2;
while( n <= maximo ){
    U.println(n + " " + Math.sqrt(n));
    n = n + 1;
}
```

Si maximo<2, las instrucciones no se repiten nunca.

**Nota.** Esta propiedad permite diseñar ciclos que eventualmente no se repiten nunca.

**¿Qué sucede si la condición es siempre verdadera?**

Ejemplo:

```
U.println("N° raíz");
int n = 2;
while( n <= 30 )
{
    U.println(n + " " + Math.sqrt(n));
}
```

**Las instrucciones se repiten indefinidamente ("loop")**  
error: no se incrementa variable n

**Corolario.** El programador debe preocuparse de incluir instrucciones que influyan en la condición, de modo que eventualmente sea falsa.

**Problema.** Escribir un programa que calcule el promedio de una cantidad indeterminada de números reales siguiendo el diálogo:

```
número ? 4.5
cuenta=1 promedio=4.5

número ? 5.5
cuenta=2 promedio=5.0
. . .
número ? 0
```

**Nota.** El número 0 se usa como indicador de fin de datos

**Solución 1: usando patrón leer/while/procesar/leer**

```
//acumulador(sumatoria) y contador de números
double suma=0; int n=0;

//obtener primer número
double numero=U.readDouble("número ? ");

//repetir hasta fin de datos
while( numero != 0 )
{
    //procesar numero
    suma=suma+numero; n=n+1;
    U.println("cuenta="+n+" promedio="+suma/n);

    //obtener siguiente número
    numero=U.readDouble("número ? ");
}
```

**Solución 2: usando patrón while(true)-break**

```
double suma=0; int n=0;

//repetir indefinidamente
while( true ) //condición siempre verdadera
{
    //obtener número
    double numero=U.readDouble("número ? ");

    //quebrar repetición al detectar fin
    if(numero==0) break;//pasa a inst sgte a while

    //procesar número
    suma=suma+numero;
    n=n+1;
    U.println("cuenta="+n+" promedio="+suma/n);
}
```

**Explicaciones**

**suma = suma + numero; //variable=expresión;**

1° evaluar expresión suma + numero  
2° guardar resultado en variable suma (reemplazando valor previo)

Por ejemplo, la 1ª vez suma=0.0 y numero=4.5, por lo tanto, suma+numero es 4.5 y se guarda como el nuevo valor de suma. La 2ª vez, suma=4.5 y numero=5.5 entonces suma+numero es 10.0 que reemplaza el valor de suma

**double numero=U.readDouble("número?");**

- La variable numero es local (interna) al bloque {...} de la inst while
- se crea ("nace") al comienzo de cada iteración
- desaparece ("muere") al terminar de ejecutarse las instrucciones del bloque.

## clase4: instrucción while

**Ejercicio.** Escribir los sgtes métodos:  
**//factorial(x): entrega 1\*2\*...\*x**  
//ejs: factorial(3)=1\*2\*3=6, factorial(0)=1  
static public int **factorial**(int x){//x>=0  
...  
}  
**//Mostrar factorial de cada n° de una lista**  
**//Al final escribir el mayor factorial.**  
static public void **main**(String[]args){  
...  
}  
**Diálogo (ejemplo)**  
n°? 4  
4!=24  
n°? 12  
12!=479001600  
...  
n°? -1 (fin de los datos)  
mayor=12!=479001600

```
static public int factorial(int x)
{
    int producto=1, i=1;

    while( i <= x )
    {
        producto = producto * i;
        i = i + 1;
    }
    return producto;
}
```

**Solución 2:**  $x! = x * (x-1) * \dots * 1$  ( $0!=1$ )

```
static public int factorial(int x)
{
    int producto=1;

    while(x>0)
    {
        producto=producto*x;
        x=x-1;
    }
    return producto;
}
```

Nota.

- x termina con el valor 0
- argumento en la llamada no se modifica
- ejemplo: factorial(n) no modifica n

```
static public void main(String[]args)
throws IOException
{
    int mayor=0; //para mayor n°

    int n=U.readInt("n°?");
    while(n != -1) //o while(x >= 0)
    {
        U.println(n+"!="+factorial(n));

        if(n>mayor) mayor=n;

        n=U.readInt("n°?");
    }
    U.println("mayor="+mayor+"!="+factorial(mayor));
}
```

### Solución 2

```
int mayor=0; //para mayor n°

while(true)
{
    int n=U.readInt("n°?");
    if(n<0) break; //o if(n== -1)break;

    U.println(n+"!="+factorial(n));

    if(n>mayor) mayor=n;
}
U.println("mayor="+mayor+"!="+factorial(mayor));
```

### Solución 3

```
int mayorNro=0, mayorFact=1; //para mayor

while(true)
{
    int n=U.readInt("n°?");
    if(n<0) break;

    int fact=factorial(n);
    U.println(n+"!="+fact);

    if(n>mayorNro){
        mayorNro=n;
        mayorFact=fact;
    }
}
U.println("mayor="+mayorNro+"!="+mayorFact);
```