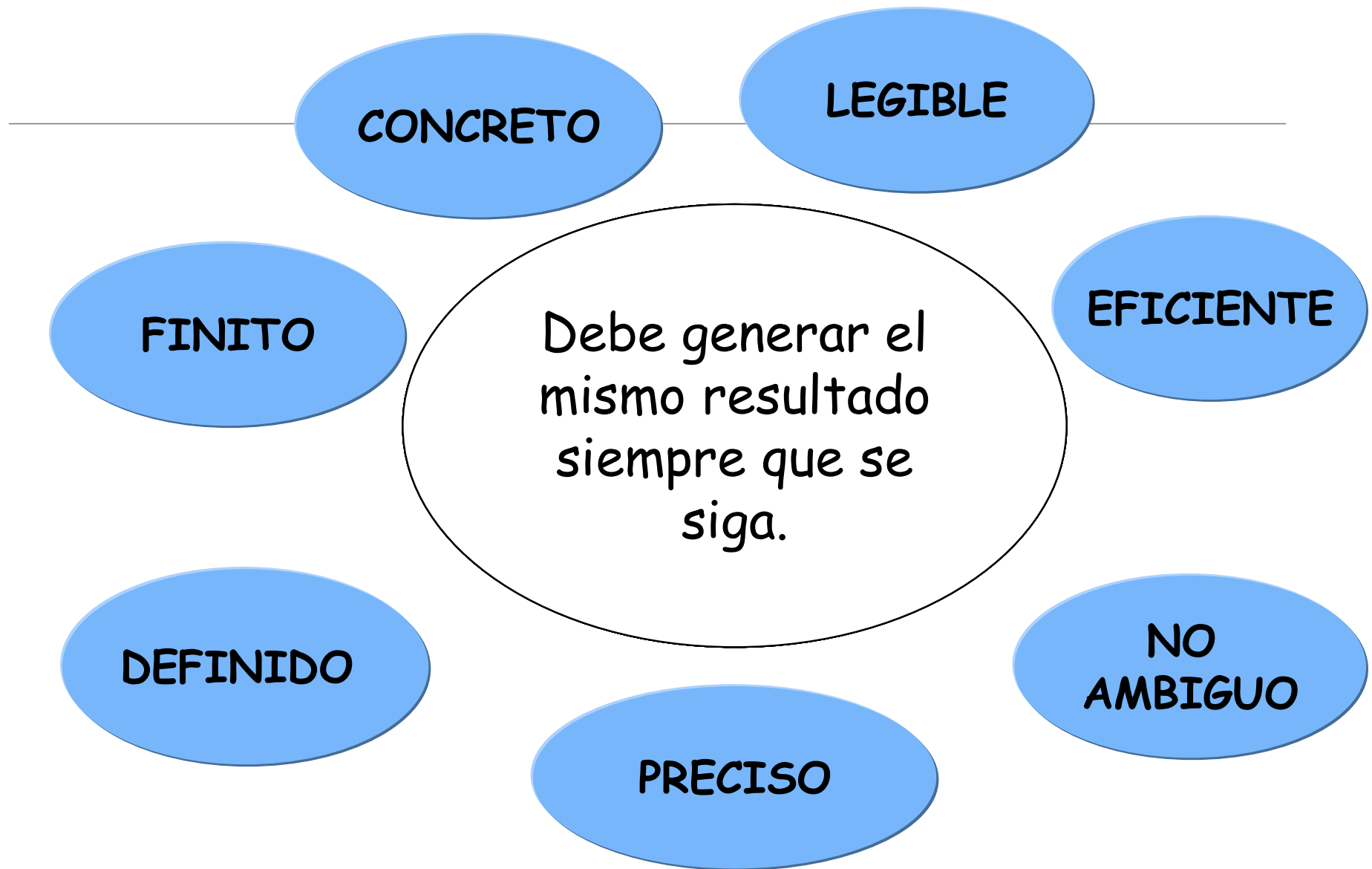

ALGORITMOS Y SENTENCIAS DE JAVA

ALGORITMO: Características



ALGORITMO: Estructura

Corresponden a los datos requeridos para realizar el algoritmo (datos de entrada) y los datos que son generados (datos de salida)

Datos

Conforma el grupo de instrucciones que realizan las operaciones con los datos.

Procesos

Determinan la organización de las instrucciones que deben ser realizadas.

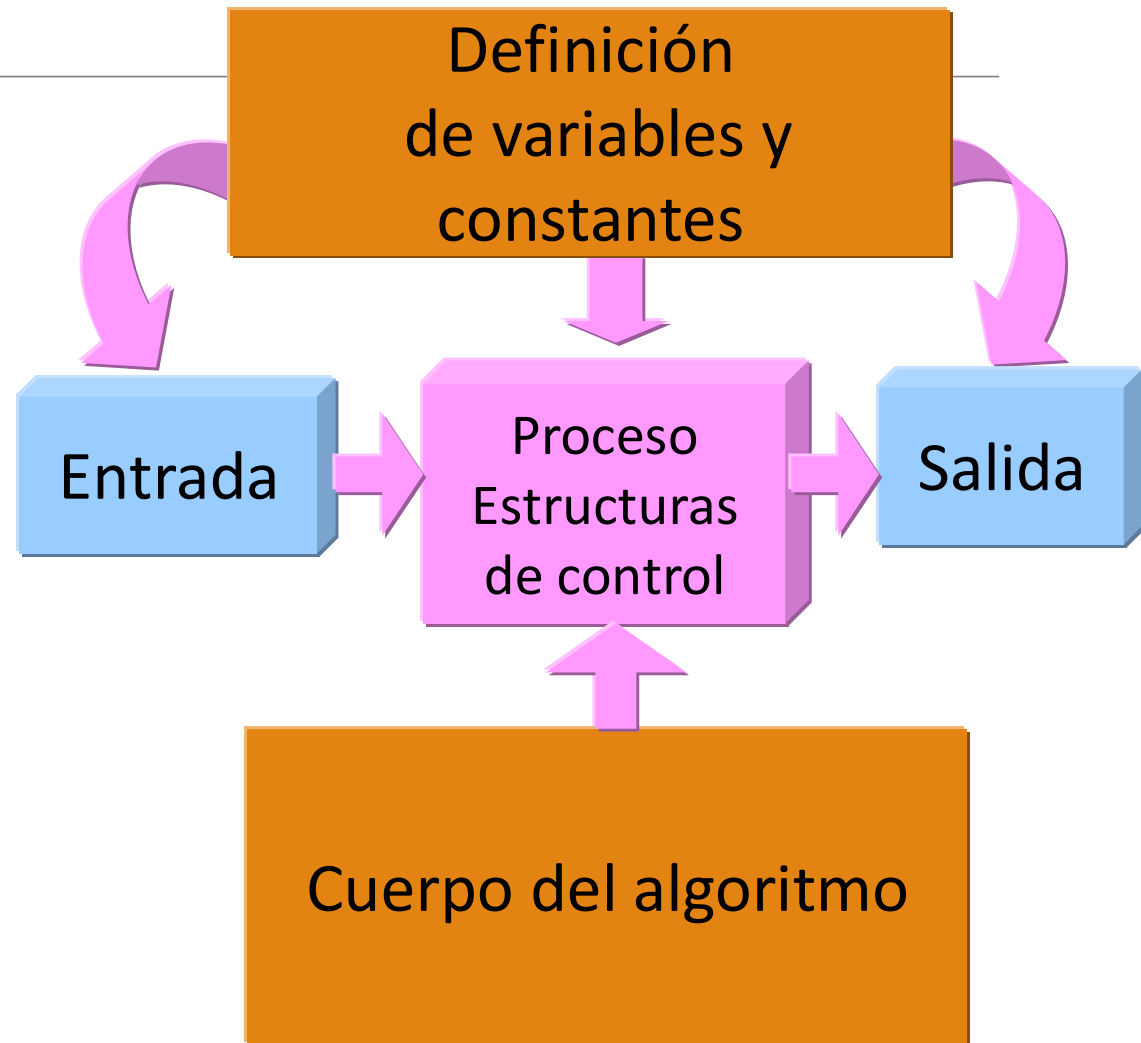
Estructuras
de Control



ALGORITMO: Elementos

Las instrucciones que se van a realizar deben estar bien estructuradas y tener un orden lógico, con el fin de evitar inconsistencias en el resultado.

Es necesario identificar que datos se necesitan ingresar, cuales sirven de forma auxiliar y cuales se van a generar.



ALGORITMO: Requisitos

Por ejemplo:

Si se requiere hallar la velocidad de un automóvil, es necesario, definir si la distancia debe ser en metros, kilómetros, etc y el tiempo estará dado en segundos u horas, ya que la velocidad puede representarse en Km/h ó mts/seg.

Debe Definirse del problema

Debe estar dentro de contexto

Debe resolver el problema

Debe evitar la ambigüedad

ALGORITMO: Técnicas de Diseño

Es una técnica de diseño descendente donde se realiza un refinamiento sucesivo, que permite darle una organización a las instrucciones, en forma de módulos o bloques.

Esta técnica permite dividir el problema en pequeñas partes, a las cuales se les da solución por separado, luego se integran las soluciones para resolver el problema principal.

Top Down

Divide y vencerás

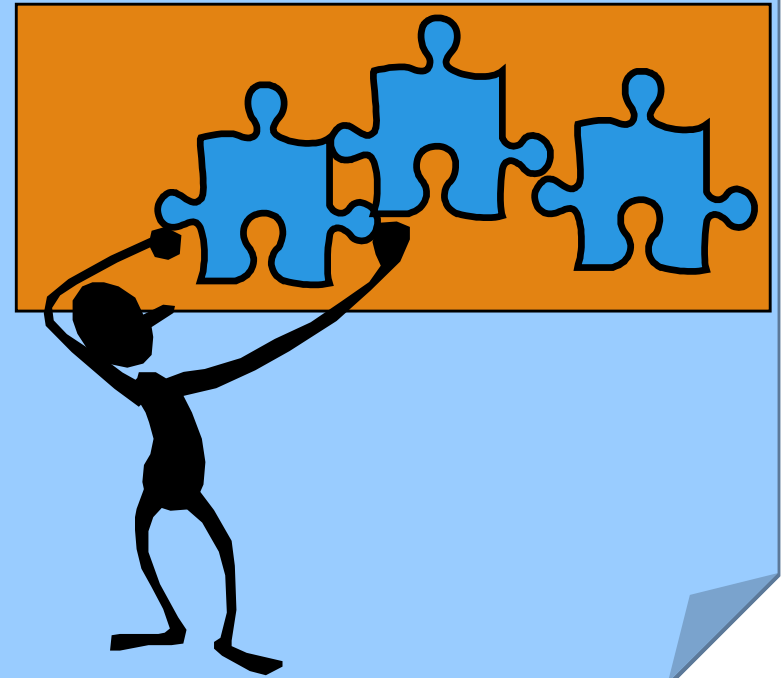
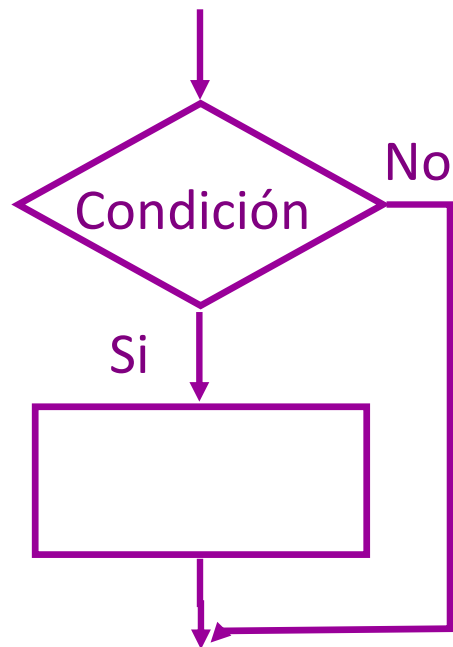


DIAGRAMA DE FLUJO: Simbología

Estructuras de Decisión (Condición)

Decisión Simple

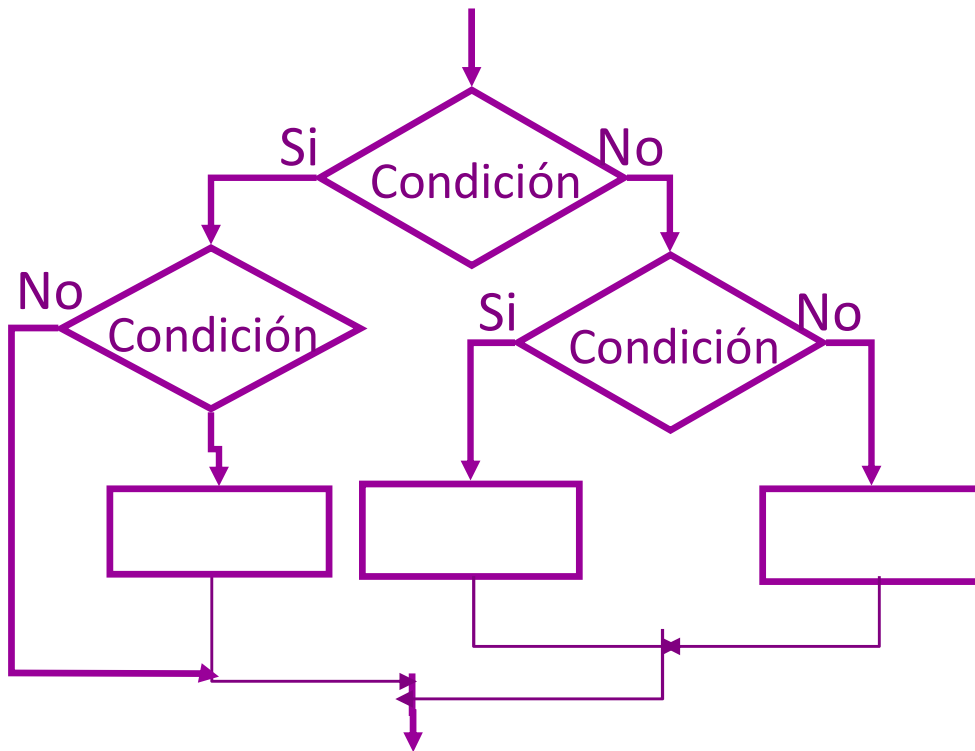


Si condición **entonces**
Instrucciones
si no
Instrucciones

DIAGRAMA DE FLUJO: Simbología

Estructuras de Decisión (Condición)

Decisión Anidada



Si condición entonces

Si condición entonces

Instrucciones

si no

Si condición entonces

Instrucciones

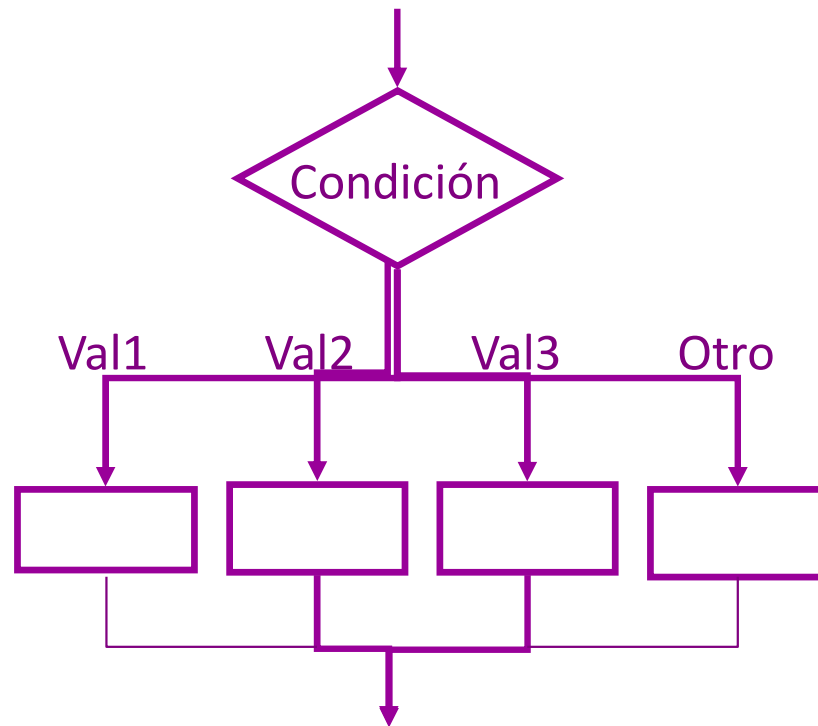
si no

Instrucciones

DIAGRAMA DE FLUJO: Simbología

Estructuras de Decisión (Condición)

Decisión Múltiple



Caso condición

Val1:

Instrucciones

Val2:

Instrucciones

Val3:

Instrucciones

Otros:

Instrucciones

Fin Caso

ALGORITMO: Fases de Diseño

Algoritmo

Definición del problema

Análisis del problema

Selección de la mejor alternativa

Diagramación

Prueba de escritorio

Estructura Condicional If

- Permite crear estructuras condicionales simples, se usa cuando se desea evaluar una comparación y realizar acción es cuando la comparación sea verdadera.

- **Sintaxis:**

if (condicion)

{ instrucciones que se ejecutan si la condición es verdadera }

else

{ instrucciones que se ejecutan si la condición es falsa }

... Estructura Condicional if

if(condición)

La (condición) tiene 3 partes:

if (variable **op.relacional** dato)

- Ejemplo: (edad > 18)

if (variable **op.relacional** variable)

- Ejemplo: (talla != peso)

Se puede unir dos o más condiciones con operadores lógicos AND, OR.

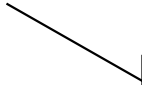
...Estructura Condicional If

Condicional simple

```
if (condicion) {  
    // bloque de sentencias  
}
```



```
if (edad > 18)  
    sumaMayor++;
```



```
if (edad > 18) {  
    sumaMayor++;  
    contador++;  
}
```

Estructuras Condicional

Estructura Condicional `if else if`

```
if (condicion1){  
    //bloque de sentencias 1  
}else if (condicion2){  
    //bloque de sentencias 2  
else{  
    //bloque de sentencias  
}
```

Estructura Condicional

Pérdida del else

```
if (temperatura > 37)
    if (tensionArterial > 12)
        contadorEnfermo++;
    else
        contadorSaludable++;
```

el else está asociado al if inmediatamente anterior

Estructura Condicional

```
if (temperatura > 37) {  
    if (tensionArterial > 12)  
        contadorEnfermo++;  
}  
else  
    contadorSaludable++;
```


CICLOS EN JAVA



WHILE

```
inicializacion;  
while (condicion) {  
    ...  
    step;  
}
```

```
int i = 0;  
while (i <= 10) {  
    System.out.println(i);  
    i++;  
}
```



WHILE

```
public static boolean esPrimo(int n) {  
    int d = 2;  
    while (d < n) {  
        if (n%d == 0){  
            return false;  
        }  
        d++;  
    }  
    return true;  
}
```



FOR

```
for (inicializacion;condición;step) {  
    ...  
}
```

```
for (int i = 0; i <= 10 ; i++) {  
    System.out.println(i) ;  
}
```



FOR

```
public static boolean esPrimo(int n) {  
    for (int d = 2; d < n ; d++) {  
        if (n%d == 0) {  
            return false;  
        }  
    }  
    return true;  
}
```



DO

```
inicializacion;  
do {  
    ...  
    step;  
} while (condición);
```

```
int i = 0;  
do {  
    System.out.println(i) ;  
    i++;  
} while (i <= 10);
```



BREAK Y CONTINUE

El **break** termina inmediatamente el ciclo o switch.

```
String s = br.readLine();  
while (s.length() > 0) {  
    System.out.println(s);  
    s = br.readLine();  
}
```

```
-----  
while (true){  
    s = br.readLine();  
    if (s.length == 0) break;  
    System.out.println(s);  
}
```



EJERCICIOS

- 1.- Elabore un programa usando do-while para calcular el promedio, el numero mayor y el menor de una lista de x números.(ingresar números hasta detener por teclado).
- 2.- Elabore un programa para mostrar la serie : $1 + 2 - 3 + 4 - 5 \dots + n$. Ingresar n por teclado.



BREAK Y CONTINUE

test:

```
if (x == 0) {  
    ciclo:  
    while (i < 9) {  
        algo: {  
            switch(control) {  
                case 0: break;  
                case 1: break algo;  
                case 2: break ciclo;  
                case 3: break test;  
                case 4: continue;  
                default: continue ciclo;  
            } instruccion1;  
        } instruccion2; i++;  
    } instruccion3;  
} instruccion4;
```



BREAK Y CONTINUE

test:

```
if (x == 0) {  
    ciclo:  
    while (i < 9) {  
        algo: {  
            switch(control) {  
                case 0: break;  
                case 1: break algo;  
                case 2: break ciclo;  
                case 3: break test;  
                case 4: continue;  
                default: continue ciclo;  
            } instruccion1;  
        } instruccion2; i++;  
    } instruccion3;  
} instruccion4;
```



BREAK Y CONTINUE

test:

```
if (x == 0) {  
    ciclo:  
    while (i < 9) {  
        algo: {  
            switch(control) {  
                case 0: break;  
                case 1: break algo;  
                case 2: break ciclo;  
                case 3: break test;  
                case 4: continue;  
                default: continue ciclo;  
            } instruccion1;  
        } instruccion2; i++;  
    } instruccion3;  
} instruccion4;
```



BREAK Y CONTINUE

test:

```
if (x == 0) {  
    ciclo:  
    while (i < 9) {  
        algo: {  
            switch(control) {  
                case 0: break;  
                case 1: break algo;  
                case 2: break ciclo;  
                case 3: break test;  
                case 4: continue;  
                default: continue ciclo;  
            } instruccion1;  
        } instruccion2; i++;  
    } instruccion3;  
} instruccion4;
```



BREAK Y CONTINUE

test:

```
if (x == 0) {  
    ciclo:  
    while (i < 9) {  
        algo: {  
            switch(control) {  
                case 0: break;  
                case 1: break algo;  
                case 2: break ciclo;  
                case 3: break test;  
                case 4: continue;  
                default: continue ciclo;  
            } instruccion1;  
        } instruccion2; i++;  
    } instruccion3;  
} instruccion4;
```



BREAK Y CONTINUE

test:

```
if (x == 0) {  
    ciclo:  
    while (i < 9) {  
        algo: {  
            switch(control) {  
                case 0: break;  
                case 1: break algo;  
                case 2: break ciclo;  
                case 3: break test;  
                case 4: continue;  
                default: continue ciclo;  
            } instruccion1;  
        } instruccion2; i++;  
    } instruccion3;  
} instruccion4;
```

