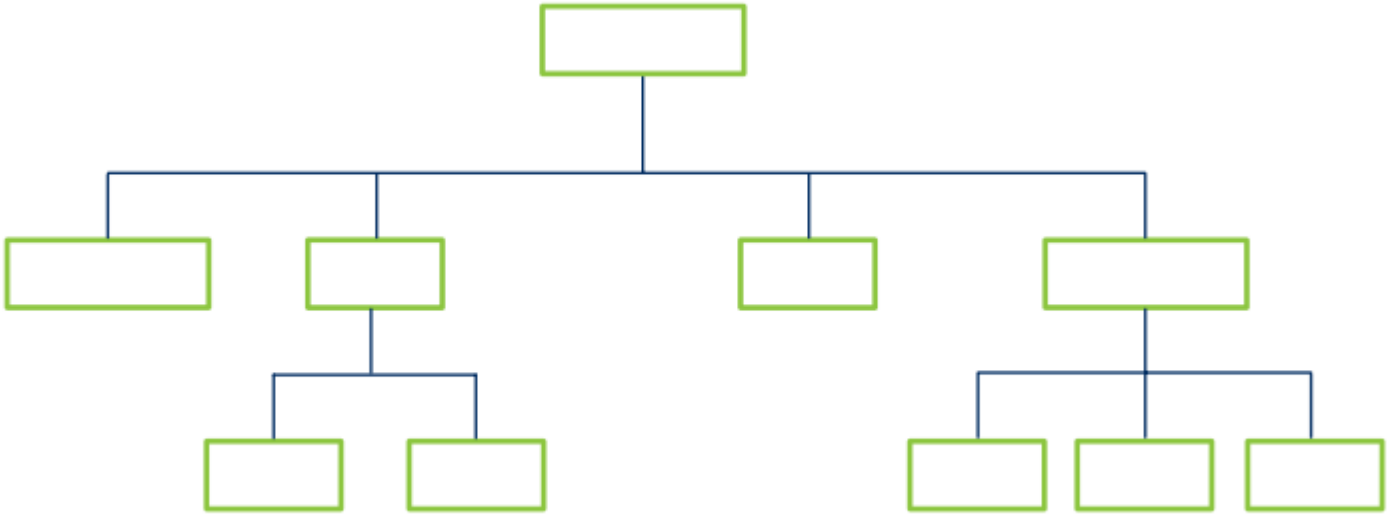
UD3: Aplicación de las estructuras de control.

IES LOS SAUCES – BENAVENTE
CFGS DESARROLLO DE APLICACIONES WEB
PROGRAMACIÓN

Programación estructurada

Paradigma de programación orientado a mejorar la claridad, calidad y tiempo de desarrollo de un programa, utilizando módulos y tres estructuras de control: secuencia, selección e iteración.



¿Qué es un módulo?

Un problema complejo debe ser dividido en varios subproblemas más simples, y estos a su vez en otros subproblemas más simples.

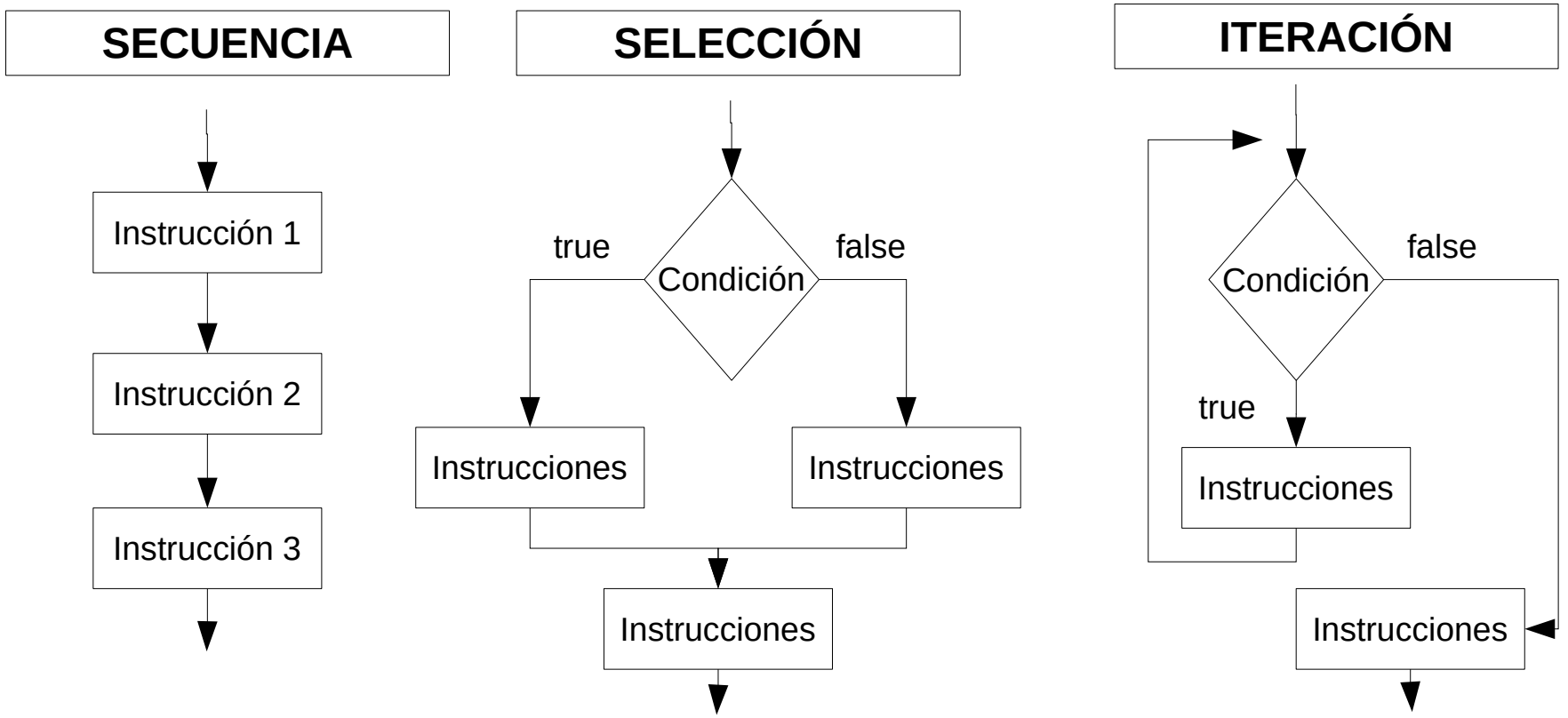
Ésta técnica se llama divide y vencerás o diseño descendente (Top-Down).

Un módulo es cada una de las partes de un programa que resuelve uno de los subproblemas en que se divide el problema complejo original.

Un módulo tiene una entrada y un salida.

Teorema del programa estructurado

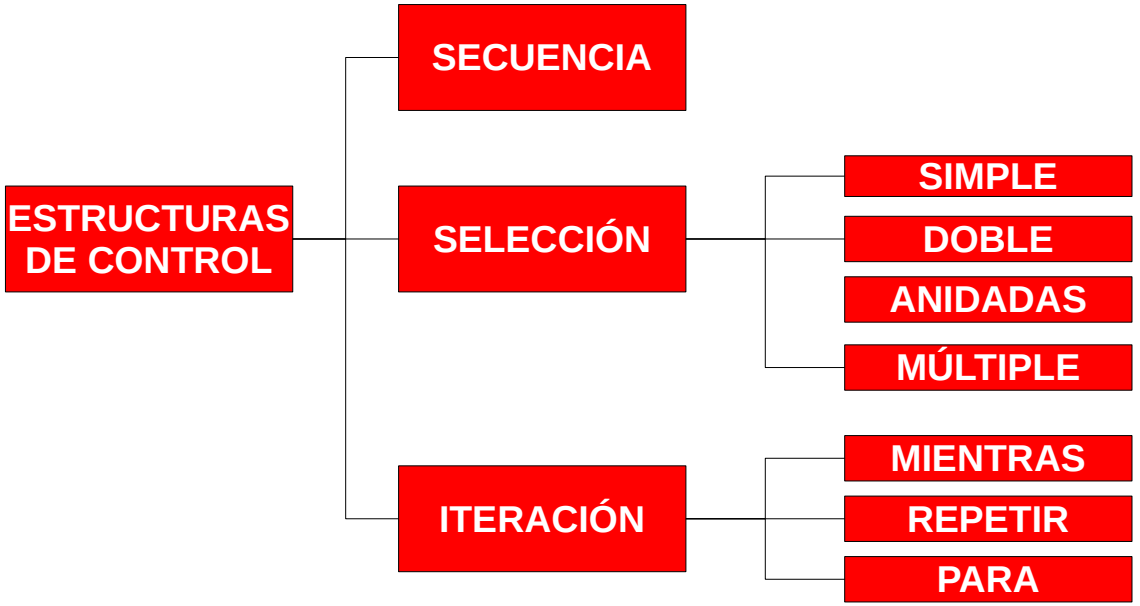
Todo programa puede escribirse utilizando únicamente tres estructuras de control: secuencia, selección e iteración.



Estructuras de control

Las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa.

Las estructuras de control de un programa sólo deben tener un punto de entrada y un punto de salida.



Estructura secuencial

La estructura secuencial es aquella en la que una instrucción sigue a otra en secuencia.

Instrucciones que podemos utilizar:

- ✔**Instrucciones declarativas**
- ✔**Instrucciones de entrada**
- ✔**Instrucciones de salida**
- ✔**Instrucciones de asignación**

Instrucciones declarativas

Se utilizan para declarar los objetos que forman parte del programa.

PSEUDOCÓDIGO

```
1  Proceso declaracion
2      Definir variable Como Entero;
3  FinProceso
```

JAVA

```
int variable;
```

Instrucciones de entrada

Se utilizan para tomar datos desde un dispositivo de entrada y llevarlos a una variable.

PSEUDOCÓDIGO

```
1  Proceso entrada
2      Leer variable;
3  FinProceso
```

JAVA

```
variable=teclado.readLine();
```

Instrucciones de salida

Se utilizan para enviar a un dispositivo de salida el resultado de evaluar una expresión.

PSEUDOCÓDIGO

```
1  Proceso salida
2      Escribir expresion;
3  FinProceso
```

JAVA

```
System.out.println(expresion);
```

Instrucciones de asignación

Se utilizan para asignar valores a las variables de un programa.

PSEUDOCÓDIGO

```
1  Proceso asignacion
2      variable<-expresion;
3  FinProceso
```

JAVA

```
variable=expresion;
```

Estructura de selección

La estructura de selección o estructura condicional o estructura alternativa es una instrucción o un conjunto de instrucciones que se pueden ejecutar o no en función del valor de una condición (expresión lógica).

Instrucciones que podemos utilizar:

- ✔**Condicional simple**
- ✔**Condicional doble**
- ✔**Condicionales anidadas**
- ✔**Condicional múltiple**

Condicional simple

Evalúa una condición (expresión lógica). Si es verdadera ejecuta una instrucción o grupo de instrucciones. Si es falsa no ejecuta nada.

PSEUDOCÓDIGO

```
1  Proceso cond_simple
2      Si condicion Entonces
3          instrucciones;
4      FinSi
5  FinProceso
```

JAVA

```
if (condicion){
    intrucciones;
}
```

Condicional simple

Pseudocódigo de un algoritmo que solicita un número entero (n) por teclado y muestra por pantalla el mensaje "POSITIVO" si dicho número es mayor que 0.

```
1   Proceso cond_simple
2       Definir n Como Entero;
3       Escribir "Introduce entero:";
4       Leer n;
5       Si (n>0) Entonces
6           Escribir "POSITIVO";
7       FinSi
8   FinProceso
```

Condicional simple

Codificación en lenguaje de programación Java de un algoritmo que solicita un número entero (n) por teclado y muestra por pantalla el mensaje "POSITIVO" si dicho número es mayor que 0.

```
import java.util.Scanner;
public class CondSimple{
    public static void main(String[] args){
        Scanner teclado=new Scanner(System.in);
        int n;

        System.out.print("Introduce entero: ");
        n=teclado.nextInt();

        if(n>0){
            System.out.println("POSITIVO");
        }
    }
}
```

Condicional doble

Evalúa una condición (expresión lógica). Si es verdadera ejecuta una instrucción o grupo de instrucciones. Si es falsa ejecuta otra instrucción o grupo de instrucciones.

PSEUDOCÓDIGO

```
1  Proceso cond_doble
2      Si (condicion) Entonces
3          Instrucciones;
4
5      Sino
6          Instrucciones;
7      FinSi
8  FinProceso
```

JAVA

```
if (condicion){
    intrucciones;
}
else{
    instrucciones;
}
```

Condicional doble

Pseudocódigo de un algoritmo que solicita un número entero (n) por teclado y muestra por pantalla uno de los siguientes mensajes: “PAR” o “IMPAR” dependiendo de si el número introducido es par o impar.

```
1   Proceso cond_doble
2       Definir n Como Entero;
3       Escribir "Introduce entero: ";
4       Leer n;
5       Si ((n % 2) = 0) Entonces
6           Escribir "PAR";
7
8       Sino
9           Escribir "IMPAR";
10      FinSi
11  FinProceso
```

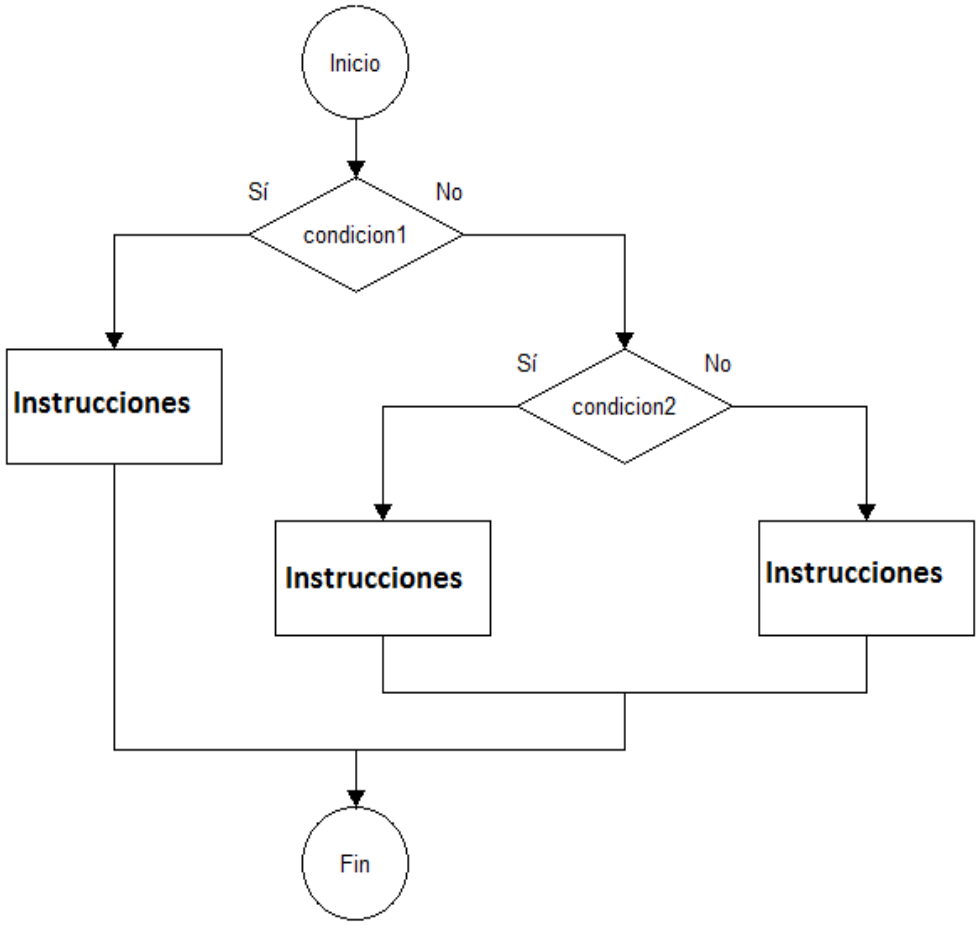
Condicional doble

Codificación en lenguaje de programación Java de un algoritmo que solicita un número entero (n) por teclado y muestra por pantalla uno de los siguientes mensajes: “PAR” o “IMPAR” dependiendo de si el número introducido es par o impar.

```
import java.util.Scanner;
public class CondDoble{
    public static void main(String[] args){
        Scanner teclado=new Scanner(System.in);
        int n;

        System.out.print("Introduce entero: ");
        n=teclado.nextInt();

        if((n%2) == 0){
            System.out.println("PAR");
        }
        else{
            System.out.println("IMPAR");

        }
    }
}
```

Condicional anidada

Un algoritmo puede evaluar varios casos colocando instrucciones condicionales dentro de otras instrucciones condicionales .



Condicional anidada

El número de condiciones a evaluar dependerá del número de alternativas distintas que necesitemos.

```
1   Proceso cond_anid
2       Si condicion1 Entonces
3   +       instrucciones;
4       Sino
5           Si condicion2 Entonces
6   +           instrucciones;
7           Sino
8   +           instrucciones;
9           FinSi
10      FinSi
11  FinProceso
```

Condicional anidada

Necesitaremos una condición menos del número de alternativas posibles.

```
if (condicion1){
      instrucciones;
}
else{
      if (condicion2){
            instrucciones;
      }
      else{
            instrucciones;
      }
}
```

Condicional anidada

Pseudocódigo de un algoritmo que solicita un número entero por teclado y muestra uno de los siguientes mensajes:

“POSITIVO”, “NEGATIVO” o “NULO”.

```
Proceso cond_anid
    Definir n Como Entero;
    Escribir "Introduce entero: ";
    Leer n;
    Si (n>0) Entonces
        Escribir "POSITIVO";
    Sino
        Si (n<0) Entonces
            Escribir "NEGATIVO";
        Sino
            Escribir "NULO";
        FinSi
    FinSi
FinProceso
```

Condicional anidada

Codificación en lenguaje de programación Java de un algoritmo que solicita un número entero por teclado y muestra uno de los siguientes mensajes:

“POSITIVO”, “NEGATIVO” o “NULO”.

```java
import java.util.Scanner;
public class CondAnidada{
    public static void main(String[] args){
        Scanner teclado=new Scanner(System.in);
        int n;

        System.out.print("Introduce entero: ");
        n=teclado.nextInt();

        if (n>0){
            System.out.println("POSITIVO");
        }
        else{
            if (n<0){
                System.out.println("NEGATIVO");
            }
            else{
                System.out.println("NULO");
            }
        }
    }
}
```

Condicional múltiple

Evalúa una expresión y permite ejecutar una instrucción o grupo de instrucciones en función del valor tomado por la expresión.

En Java la expresión deberá ser de alguno de los siguientes tipos primitivos: byte, short, char, int.

También funciona con: tipos enumerados, la clase String y las clases envoltorio: Character, Byte, Short e Integer.

Condicional múltiple

```
1  Proceso cond_mult
2      Segun expresion Hacer
3          valor_1:
4+             instrucciones;
5          valor_2:
6+             intrucciones;
7          valor_3:
8+             instrucciones;
9          De Otro Modo:
10+            instrucciones;
11     FinSegun
12 FinProceso
```

Condicional múltiple

```
switch (expresion) {
        case valor_1:
             instrucciones;
             [break;]
        case valor_2:
             instrucciones;
             [break;]
        case valor_3:
             instrucciones;
             [break;]
        [default:
             instrucciones;
        ]
}
```

Condicional múltiple

Un switch puede contener cualquier número de case, aunque no puede haber dos con el mismo valor.

La instrucción break es opcional y se usa para provocar la finalización del switch al terminar la ejecución de un case.

Si un case no contiene la instrucción break, el programa continuará con la ejecución del siguiente case independientemente de que el resultado de la expresión coincida o no con el valor indicado en el mismo.

El uso de default es opcional. Se ejecutarán las instrucciones en el supuesto de que la expresión no coincida con ningún valor de los case.

Condicional múltiple

Pseudocódigo de un algoritmo que solicita un número entero por teclado comprendido entre 1 y 7 y muestra por pantalla su correspondiente día de la semana: “lunes”, “martes”, “miércoles”, “jueves”, ”viernes”, “sábado” o “domingo”.

```
1  Proceso cond_mult
2      Definir n Como Entero;
3
4      Escribir "Introduce entero (1..7): " sin saltar;
5      Leer n;
6
7      Segun n Hacer
8          1:
9              Escribir "lunes";
10         2:
11             Escribir "martes";
12 +       ..........................
13         7:
14             Escribir "domingo";
15     FinSegun
16 FinProceso
```

Condicional múltiple

Codificación en lenguaje de programación Java de un algoritmo que solicita un número entero por teclado comprendido entre 1 y 7 y muestra por pantalla su correspondiente día de la semana: “lunes”, “martes”, “miércoles”, “jueves”, ”viernes”, “sábado” o “domingo”.

```
import java.util.Scanner;
public class CondMultiple{
    public static void main(String[] args){
        Scanner teclado=new Scanner(System.in);
        int n;

        System.out.print("Introduce entero (1..7): ");
        n=teclado.nextInt();

        switch(n){
            case 1:
                System.out.println("lunes");
                break;
            case 2:
                System.out.println("martes");
                break;
            ..........................................
            case 7:
                System.out.println("domingo");
                break;
        }
    }
}
```

Condicional múltiple

A partir de Java 12 se han introducido nuevas características al switch:

(https://openjdk.java.net/jeps/361)

- **Se pueden declarar case con múltiples valores separados por comas.**
- **Nueva sintaxis del switch en la que no es necesario usar break. En lugar de utilizar : después de cada case, utiliza ->**

  En el nuevo switch, cada bloque de instrucciones de un case debe ir entre llaves.
- **El switch se puede usar como expresión (deberá terminar con ;). Cada uno de los case devolverá un valor mediante la palabra reservada yield.**

Condicional múltiple

Codificación en lenguaje de programación Java de un algoritmo que solicita un día de la semana por teclado (“L”, “M”, “X”, “J”, ”V”, “S”, “D”) y muestra por pantalla si se trata de “LABORABLE” o “FIN DE SEMANA”

```
import java.util.Scanner;
public class CondMultiple2{
    public static void main(String[] args){
        Scanner teclado=new Scanner(System.in);
        String dia;

        System.out.print("Introduce día de la semana (L, M, X, J, V, S, D): ");
        dia=teclado.nextLine();

        switch(dia){
            case "L","M","X","J","V"-> System.out.println("LABORABLE");
            case "S", "D"           -> System.out.println("FIN DE SEMANA");
            default                 -> System.out.println("ERROR EN LA ENTRADA");
        }
    }
}
```

Condicional múltiple

Codificación en lenguaje de programación Java de un algoritmo que solicita un día de la semana por teclado (“L”, “M”, “X”, “J”, ”V”, “S”, “D”) y muestra por pantalla si se trata de “LABORABLE” o “FIN DE SEMANA”

```
import java.util.Scanner;
public class CondMultiple3{
    public static void main(String[] args){
        Scanner teclado=new Scanner(System.in);
        String dia, salida;

        System.out.print("Introduce día de la semana (L, M, X, J, V, S, D): ");
        dia=teclado.nextLine();

        salida=switch(dia){
            case "L","M","X","J","V": yield "LABORABLE";
            case "S", "D"           : yield "FIN DE SEMANA";
            default                 : yield "ERROR EN LA ENTRADA";
        };

        System.out.println(salida);
    }
}
```

Condicional múltiple

Codificación en lenguaje de programación Java de un algoritmo que solicita un día de la semana por teclado (“L”, “M”, “X”, “J”, ”V”, “S”, “D”) y muestra por pantalla si se trata de “LABORABLE” o “FIN DE SEMANA”

```
import java.util.Scanner;
public class CondMultiple4{
    public static void main(String[] args){
        Scanner teclado=new Scanner(System.in);
        String dia, salida;

        System.out.print("Introduce día de la semana (L, M, X, J, V, S, D): ");
        dia=teclado.nextLine();

        salida=switch(dia){
            case "L","M","X","J","V" -> "LABORABLE";
            case "S", "D"            -> "FIN DE SEMANA";
            default                  -> "ERROR EN LA ENTRADA";
        };

        System.out.println(salida);
    }
}
```

Estructura iterativa

La estructura iterativa o estructura repetitiva o bucle permite ejecutar una instrucción o un conjunto de instrucciones, ninguna, una o varias veces en función del valor de una condición (expresión lógica).

Estructuras iterativas:

- ✔**Mientras**
- ✔**Repetir**
- ✔**Para**

Mientras

Permite ejecutar una instrucción o grupo de instrucciones mientras se cumpla una condición.

Se utiliza cuando el grupo de instrucciones lo queremos ejecutar cero o más veces.

PSEUDOCÓDIGO

```
1  Proceso mientras
2      Mientras expresion_logica Hacer
3          instrucciones
4      FinMientras
5  FinProceso
```

JAVA

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

Mientras

1. Se evalúa la condición (expresión lógica).
2. Si la expresión es verdadera se ejecutan las instrucciones, sino el programa saldrá del bucle.
3. Ejecutadas las instrucciones volvemos al paso 1.

Las instrucciones se ejecutarán cero o más veces.

Hay que asegurarse que en algún momento la condición no se cumpla para permitir la salida del bucle, de lo contrario tendríamos un bucle infinito.

Mientras

Pseudocódigo de un algoritmo que solicite un número entero positivo por teclado. El proceso se repetirá mientras que el número introducido no sea positivo. Para finalizar mostrará el número por pantalla.

```
Proceso while
    Definir n Como Entero;

    Escribir "Introduzca entero positivo: ";
    Leer n;
    Mientras n<=0 Hacer
        Escribir "Introduzca entero positivo: ";
        Leer n;
    FinMientras

    Escribir n;

FinProceso
```

Mientras

Codificación en lenguaje de programación Java de un algoritmo que solicite un número entero positivo por teclado. El proceso se repetirá mientras que el número introducido no sea positivo. Para finalizar mostrará el número por pantalla.

```
import java.util.Scanner;
public class Mientras{
    public static void main(String[] args){
        Scanner teclado=new Scanner(System.in);
        int n;

        System.out.print("Introduzca un entero positivo: ");
        n=teclado.nextInt();
        while(n<=0){
            System.out.print("Introduzca un entero positivo: ");
            n=teclado.nextInt();
        }
        System.out.println(n);
    }
}
```

Repetir

Permite ejecutar una instrucción o grupo de instrucciones en función de una condición que se evalúa al final del bucle.

Se utiliza cuando el grupo de instrucciones lo queremos ejecutar una o más veces.

Las instrucciones se ejecutarán una o más veces.

Repetir

PSEUDOCÓDIGO

```
1 + Proceso repetir
2       Repetir
3 +         instrucciones
4       Hasta Que expresion_logica
5   FinProceso
```

1.Se ejecutan las instrucciones.

2.Se evalúa la condición.

3.Si la condición es verdadera saldremos del bucle, sino volveremos al paso 1.

Hay que asegurarse que en algún momento la condición se cumpla para permitir la salida del bucle, de lo contrario tendríamos un bucle infinito.

Repetir

JAVA

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

1.Se ejecutan las instrucciones.

2.Se evalúa la condición.

3.Si la condición es verdadera volvemos al paso 1, sino saldremos del bucle.

Hay que asegurarse que en algún momento la condición no se cumpla para permitir la salida del bucle, de lo contrario tendríamos un bucle infinito.

Repetir

Pseudocódigo de un algoritmo que solicite un número entero positivo por teclado. El proceso se repetirá hasta que el número introducido sea positivo. Para finalizar mostrará el número por pantalla.

```
Proceso RepetirHasta
    Definir n Como Entero;

    Repetir
        Escribir "Introduzca entero positivo: ";
        Leer n;
    Hasta Que n>0;

    Escribir n;

FinProceso
```

Repetir

Codificación en lenguaje de programación Java de un algoritmo que solicite un número entero positivo por teclado. El proceso se repetirá hasta que el número introducido sea positivo. Para finalizar mostrará el número por pantalla.

```
import java.util.Scanner;
public class DoWhile{
    public static void main(String[] args){
        Scanner teclado=new Scanner(System.in);
        int n;

        do{
            System.out.print("Introduzca un entero positivo: ");
            n=teclado.nextInt();
        }while(n<=0);
        System.out.println(n);
    }
}
```

Para

Permite ejecutar una instrucción o grupo de instrucciones un determinado número de veces.

Se utiliza cuando sabemos el número de veces que queremos ejecutar el grupo de instrucciones.

Para

1. Se inicializa la variable numérica al valor inicial.
2. Se comprueba si la variable numérica ha superado el valor final.
3. Si ha superado el valor final salimos del bucle, sino se ejecutarán las instrucciones.
4. Se incrementa (decrementa) la variable numérica con el valor de paso.
5. Volvemos al paso 2.

```
Proceso para_pse
    Para variable_numerica<-valor_inicial Hasta valor_final Con Paso paso Hacer
        instrucciones
    FinPara
FinProceso
```

Para

1. Se ejecuta la instrucción de inicialización.
2. Se comprueba la condición.
3. Si la condición se cumple se ejecutan las instrucciones, sino salimos del bucle.
4. Se ejecuta la instrucción de incremento (decremento)
5. Volvemos al paso 2.

```
for(inicialización;condición;incremento){
    instrucciones
}
```

Para

En el bucle for podremos omitir cualquiera de las tres expresiones pero los separadores ; deben aparecer siempre.

En la expresión de inicialización se suele declarar la variable que nos servirá de control de permanencia en el bucle.

La variable declarada muere cuando el bucle acaba.

```
for(int i=1;condición;incremento){
    instrucciones
}
```

Para

Pseudocódigo de un algoritmo que muestra por pantalla los números enteros del 1 al 10.

```
Proceso para_pse
    Definir i Como Entero;
    Para i<-1 Hasta 10 Con Paso 1 Hacer
        Escribir i;
    FinPara
FinProceso
```

Para

Codificación en lenguaje de programación Java de un algoritmo que muestra por pantalla los números enteros del 1 al 10.

```
public class Para{
    public static void main(String[] args){
        for(int i=1;i<=10;i++){
            System.out.println(i);
        }
    }
}
```

Variables que se utilizan con bucles

- ✔**Contador**: variable cuyo valor se incrementa o decrementa en una cantidad constante (normalmente 1).
- ✔**Acumulador**: variable cuyo valor se incrementa (decrementa) sucesivas veces en cantidades variables.
- ✔**Interruptor**: variable que sólo puede tomar dos valores. Habitualmente suele tratarse de una variable booleana.

Contador

- ✔**Variable cuyo valor se incrementa o decrementa en una cantidad constante (normalmente 1).**
- ✔ **Permite contabilizar el número de veces que se ha repetido una instrucción o grupo de instrucciones.**
- ✔**Permite contabilizar un suceso particular (generalmente asociado a un bucle).**
- ✔**Un contador toma un valor inicial y a continuación podrá incrementar o decrementar su valor.**

Contador

```
Proceso Contador
    Definir i,n,contadorPositivos Como Entero;

    contadorPositivos<-0;

    Para i<-1 Hasta 5 Con Paso 1 Hacer
        Escribir "Introduce entero: ";
        Leer n;
        Si (n > 0) Entonces
            contadorPositivos<-contadorPositivos+1;
        FinSi
    FinPara

    Escribir "Has introducido ",contadorPositivos," números positivos";
FinProceso
```

Contador que permite contabilizar los positivos

Contador de control de bucle

Incremento del contador cuando n es positivo

51

Acumulador

- ✔**Variable cuyo valor se incrementa (decrementa) sucesivas veces en cantidades variables.**
- ✔ **Permite obtener el total acumulado de un conjunto de cantidades.**
- ✔**Si el total acumulado registra la suma de cantidades entonces el acumulador deberá inicializarse a 0.**
- ✔**Si el total acumulado registra el producto de cantidades entonces el acumulador deberá inicializarse a 1.**

Acumulador

```
Proceso Acumulador_pse
    Definir i,n,acumulador Como Entero;

    acumulador<-0;

    Para i<-1 Hasta 5 Con Paso 1 Hacer
        Escribir "Introduce entero: ";
        Leer n;
        acumulador<-acumulador+n;
    FinPara

    Escribir "Total acumulado: ",acumulador;
FinProceso
```

Inicialización del acumulador

Incremento del acumulador

Interruptor (flag)

- ✔**Variable que sólo puede tomar dos valores. Habitualmente suele tratarse de una variable booleana.**
- ✔**No todas las variables que sólo pueden tomar dos valores son interruptores.**
- ✔ **Permite recordar en un determinado punto de un programa la ocurrencia o no de un suceso anterior, lo que nos permitirá salir de un bucle o decidir en una instrucción alternativa que acción realizar.**
- ✔**Permite que dos acciones diferentes se ejecuten alternativamente dentro de un bucle.**

Interruptor (flag)

```
Proceso Interruptor
    Definir i,n Como Entero;
    Definir hayNegativos como Logico;

    hayNegativos<-Falso;

    Para i<-1 Hasta 5 Con Paso 1 Hacer
        Escribir "Introduce entero: ";
        Leer n;
        SI (n < 0) Entonces
            hayNegativos<-Verdadero;
        FinSi
    FinPara

    si hayNegativos Entonces
        Escribir "Has introducido números negativos";
    Sino
        Escribir "NO has introducido números negativos";
    FinSi

FinProceso
```

Inicializamos el interruptor

Registramos el suceso

Comprobamos el interruptor

Bucles controlador por contador

- ✔**Los bucles controlados por contador se repiten un determinado número de veces.**
- ✔**Un contador es una variable cuyo valor se incrementa o decrementa en una cantidad constante.**
- ✔**En los bucles controlados por contador se incrementa el valor de la variable en cada vuelta del bucle.**
- ✔**Cuando el contador alcanza el límite determinado, el bucle termina.**

Bucles controlador por centinela

- ✔**Los bucles controlados por centinela se repiten siempre que se cumpla una expresión lógica llamada centinela.**
- ✔**Si la expresión lógica se trata de una variable que sólo puede tomar dos valores estaremos hablando de un interruptor (flag).**
- ✔**Un interruptor es una variable, normalmente booleana, que sólo puede tomar dos valores.**
- ✔**En los bucles controlados por centinela la variable interruptor sirve para decidir si el bucle se repite o no.**

Bucles anidados

Un bucle anidado es un bucle dentro de otro bucle.

Los bucles anidados son necesarios para la resolución de determinados problemas.

```
Proceso bucles_anidados
    Definir i,j Como Entero;
    Para i<-1 Hasta 5 Con Paso 1 Hacer
        Para j<-1 Hasta 10 Con Paso 1 Hacer
            Escribir "*" Sin Saltar;
        FinPara
        Escribir "";
    FinPara
FinProceso
```

Instrucciones de ruptura de flujo

NO LAS USAREMOS PORQUE ROMPEN EL PARADIGMA DE LA PROGRAMACIÓN ESTRUCTURADA.

break

Instrucción que se utiliza para interrumpir la secuencia lógica de un bucle o para salir de una estructura switch.

continue

Instrucción que permite detener la ejecución de una iteración de un bucle y saltar a la siguiente.