Unidad 5 Estructura Alternativa

Introducción a la Programación (IP) Tecnicatura en Desarrollo Web

18 septiembre, 2023

1 Introducción

Las alternativas son partes de nuestra vida cotidiana, continuamente lidiamos con ellas en nuestras actividades. Por ej una alternativa como: 'si llueve voy al cine, si no llueve voy a la plaza', o una alternativa mas trascendental '¿que carrera voy a seguir?'.

Las estructuras alternativas son una estructura de control útil cuando nos encontramos con problemas que requieren ejecutar una o más instrucciones según diferentes casos. En otras palabras, las estructuras alternativas bifurcan o dirigen la ejecución de un programa hacia un grupo de sentencias u otro dependiendo de una condición. Las distintas condiciones que incluye una alternativa pueden estar explícitas en el mismo problema que debemos resolver por ej: 'dado un entero ingresado por el usuario si el valor del entero es par incrementarlo, si es impar decrementarlo', o pueden ser parte de la solución matemática o lógica de un problema. Veremos a continuación un ejemplo introductorio a las estructuras alternativas.

2 Alternativas Simples

Imaginemos que tenemos que resolver el siguiente problema:

```
Problema 1: Debemos leer un número, si el número es positivo, debemos escribir en pantalla su valor e indicar "es positivo".
```

Solución: Especificamos nuestra solución: se deberá leer un número x. Si x > 0 se escribe el mensaje "Número positivo". Diseñamos nuestra solución:

```
ALGORITMO esPositivo() RETORNA ∅
    (* Determina si un entero es positivo *)
    ENTERO num
    ESCRIBIR ("Ingrese un numero entero")
    LEER(num)
    SI ( num > 0 ) ENTONCES
            ESCRIBIR(num + "es Positivo")
    FIN SI
FIN ALGORITMO esPositivo
```

Note que:

- La instrucción SI-ENTONCES-FIN SI es una estructura alternativa.
- Entre paréntesis se escribe una condición lógica.
- Las primitivas escritas entre ENTONCES y FIN SI se denominan bloque de primitivas. Y siempre dejaremos una sangría o indentación antes de escribir las primitivas del bloque.
- esta primitiva se lee así: "Si num es mayor a cero entonces mostrar el cartel del valor de la variable concatenado con la cadena"es positivo".

Es claro que la lectura de la variable **num** en el pseudocódigo se puede escribir en PHP como:

```
echo "Ingrese un número entero";
$num = trim(fgets(STDIN));
```

Sin embargo, con las instrucciones que vimos hasta ahora no podemos tomar el tipo de decisiones que nos planteamos al final del pseudocódigo. Para resolver este problema introducimos una nueva instrucción PHP que llamaremos alternativa o condicional que tiene la siguiente forma:

```
if ( condición ) {
   < hacer algo si se da la condición >
}
```

Donde **if** es una **palabra reservada**. Las instrucciones a ejecutar en el caso que la condición sea verdadera se encierran entre paréntesis. Esas instrucciones encerradas entre paréntesis constituyen el bloque de instrucciones a ejecutar en el caso de que la condición sea verdadera.

¿De qué tipo es la condición que aparece luego de la palabra reservada *if* ? Es de tipo `LÓGICO` ó `Booleano` . Es decir la evaluación de la expresión arroja un resultado `VERDADERO` (`TRUE`) ó `FALSO` (`FALSE`). En otras palabras es una expresión booleana.

Para PHP el resultado verdadero se imprime con el valor 1 , y el valor de falso imprime con vacío. El manual de PHP http://php.net/manual/es/language.types.boolean.php (http://php.net/manual/es/language.types.boolean.php) explica el tipo de datos booleano:

- Un boolean expresa un valor que indica verdad. Puede ser TRUE (verdadero) o FALSE (falso). Para especificar un literal de tipo boolean se emplean las constantes TRUE o FALSE, siendo ambas constantes no son susceptibles a maýusculas y minúsculass.

Por ser un lenguaje dinámico PHP realiza conversiones de tipos y si el contexto requiere convertir un tipo de dato a boolean lo hára de la siguiente manera: Los siguientes valores se consideran FALSE: + el boolean FALSE mismo + el integer 0 (cero) + el float 0.0 (cero) + el valor string vaćıo, y el string ”0” + un array con cero elementos

Cualquier otro valor se considera como TRUE (incluso el valor -1 es tomado como TRUE).

2.1 Alternativa simple en PHP

```
ALGORITMO esPositivo() RETORNA ∅
    (* Determina si un entero es positivo *)
    ENTERO num
    ESCRIBIR (”Ingrese un numero entero”)
    LEER(num)
    SI ( num > 0 ) ENTONCES
       ESCRIBIR(num + ”Es Positivo”)
    FIN SI
FIN ALGORITMO esPositivo
```

```
<?php
  // Declaración de variables
  // INT $num
  echo ”Ingrese un número entero”;
  $num = trim(fgest(STDIN));
  // Mostramos por pantalla resultados
  if ($num > 0)
     echo $num . "es POSITIVO";
?>
```

En PHP, no es estrictamente obligatorio escribir llaves en cada bloque de una estructura alternativa (como **if**, **else**, **elseif**, etc.), siempre y cuando cada bloque conste de una sola instrucción . Sin embargo, se recomienda utilizar llaves en todos los bloques de control de flujo por razones de legibilidad y para evitar errores potenciales. No obstante, a medida que adquieras mas confianza con la actividad de programación utilizarás las llaves solo cuando sea necesario.

```
ALGORITMO esPositivo() RETORNA ∅
    (* Determina si un entero es positivo *)
    ENTERO num
    ESCRIBIR (”Ingrese un numero entero”)
    LEER(num)
    SI ( num > 0 ) ENTONCES
       ESCRIBIR("El numero ingresado" + num)
       ESCRIBIR("es un Numero Positivo")
    FIN SI
FIN ALGORITMO esPositivo
```

```
<?php
  // Declaración de variables
  // INT $num
  echo ”Ingrese un número entero”;
  $num = trim(fgest(STDIN));
  // Mostramos por pantalla resultados
  if ($num > 0) {
     echo "El numero ingresado" . $num;
     echo "es un número POSITIVO";
  }
?>
```

3 Repaso de Expresiones Booleanas

Recordemos que a nivel de diseño contamos con el tipo de dato LOGICO, que contiene solo dos valores: TRUE y FALSE para representar los valores de verdad verdadero y falso respectivamente. Estos valores son los únicos valores del Tipo LOGICO.

El tipo de dato LOGICO se modela en PHP con el tipo boolean. Vimos que una expresión nos permite producir o calcular un valor (resultado). Una expresión booleana o expresión lógica es una expresión que tiene como resultado TRUE o FALSE.

3.1 Expresiones de comparación

En el ejemplo que queremos resolver, la condición que queremos ver si se cumple o no es que x sea mayor que cero. Para ello, es necesario utilizar un operador de comparación para comparar valores entre sí. En particular la pregunta de si x es mayor que cero, se codifica en PHP como x > 0.

De esta forma, 5 > 3 es una expresión booleana cuyo valor es True, y 5 < 3 también es una expresión booleana, pero su valor es False. Otros operadores de comparación en pseudocódigo y en PHP son los que se detallan a continuación.

Expresión Pseudocódigo	Expresión PHP	Significado
$a <> b, a \neq b$	$a != b$	a es distinto de b
$a = b$	$a == b$	a es igual a b
$a < b$	$a < b$	a es menor que b
$a <= b$	$a <= b$	a es menor o igual que b
$a > b$	$a > b$	a es mayor que b
$a >= b$	$a > b$	a es mayor o igual que b

3.2 Operadores lógicos

De la misma manera que se puede operar entre números mediante las operaciones de suma, resta, etc., también existen tres operadores lógicos para combinar expresiones booleanas: and (y), or (o) y not (no).

El significado de estos operadores es igual al del castellano, pero vale la pena recordarlo en Tabla II:

Pseudocódigo	PHP	Significado
a AND b,$a \wedge b$	a && b	El resultado es TRUE si a es TRUE y b es TRUE de lo contrario el resultado es FALSE
a OR b,$a \vee b$	$a \| \| b$	El resultado es TRUE si a es TRUE o b es TRUE de lo contrario el resultado es FALSE
a NOT b	! a	El resultado es TRUE si a es FALSE y b es TRUE, o viceversa

Ejemplos: * a > b and a > c es verdadero si a es simultáneamente mayor que b y que c. * a > b or a > c es verdadero si a es mayor que b o a es mayor que c. * not (a > b) es verdadero si a > b es falso (o sea si a <= b es verdadero)

3.3 Alternativas Dobles

Problema: En la etapa de mantenimiento nos dicen que, en realidad, también se necesitaría un mensaje Numero no positivo cuando no se cumple la condición.

Modificamos la especificación del algoritmo y del código PHP:

```
ALGORITMO esPositivo() RETORNA ∅
    (* Determina si un entero es positivo *)
    ENTERO num
    ESCRIBIR (”Ingrese un numero entero”)
    LEER(num)
    SI ( num > 0 ) ENTONCES
        ESCRIBIR(num + ”Es Positivo”)
    SINO
        ESCRIBIR(num + ”No es Positivo”)
    FIN SI
FIN ALGORITMO esPositivo
```

```
<?php
  // INT $num
  echo ”Ingrese un número entero”;
  $num = trim(fgets(STDIN));
  // Mostramos por pantalla resultados
  if ($num > 0)
      echo $num .” es POSITIVO” ;
  else
      echo $num .” es NEGATIVO” ;
?>
```

En pseudocódigo una alternativa doble se modela de la siguiente forma:

```
ALGORITMO nombreAlgoritmo() RETORNA ∅
    (* .... *)
    SI ( expresiónBooleana ) ENTONCES
        Bloque de Sentencias a ejecutar si la expresión booleana es TRUE
    SINO
        Bloque de Sentencias a ejecutar si la expresión booleana es FALSE
    FIN SI
    ....
FIN ALGORITMO nombreAlgoritmo
```

Su significado es el siguiente: se evalúa la (expresiónBooleana), si el resultado es TRUE (verdadero) se ejecutan las acciones indicadas en “Bloque de Sentencias a ejecutar si la expresión booleana es TRUE”, y si el resultado es FALSE (falso) se ejecutan las acciones indicadas en “Bloque de Sentencias a ejecutar si la expresión booleana es FALSE”.

En PHP para implementar una alternativa doble utilizamos las palabras reservadas: *if* y *else*. Si alguno de los bloques de sentencias tiene más de una instrucción es obligatorio encerrar el bloque con llaves.

En una instrucción alternativa doble estamos implementando la operación:
(expresiónBooleana) $\oplus$ not (expresiónBooleana) donde $\oplus$ es la notación universal del operador XOR. El operador XOR representa la disyunción exclusiva. Debemos ser conscientes que el diseño de estructura alternativa que modelemos debe representar un XOR. En el pseudocódigo el símbolo $\oplus$ esta modelado por la palabra reservada SINO

4 Alternativas en la Etapa de Análisis

> Un **conjunto de datos es de estructura alternativa** si el mismo constituye un conjunto de datos en el cual se encuentran uno o varios subconjuntos. Cada subconjunto es de presencia aleatoria (condicional), con la salvedad de que **la presencia de uno excluye la de todos los demás**.

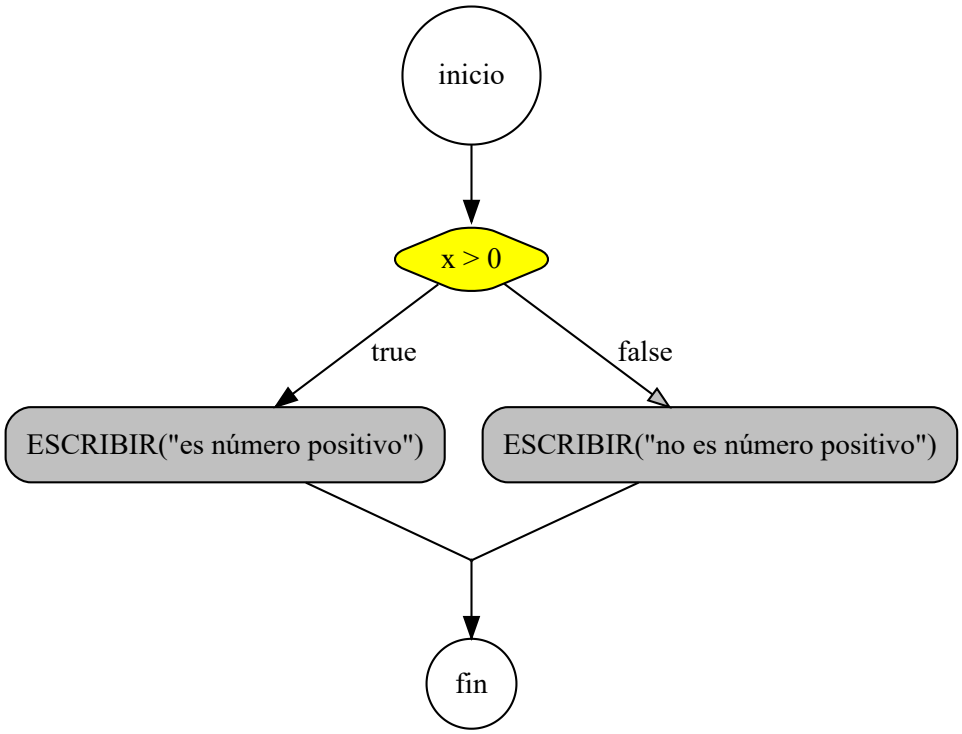
5 Alternativas en la Etapa de Diseño

Es importante darnos cuenta que en una estructura de datos secuencial todas las sentencias de un bloque son ejecutadas en secuencia. En el caso de una estructura alternativa doble el flujo de ejecución ejecuta diferentes grupos de sentencias dependiendo del valor de la expresión booleana. Es útil mostrar como se modela una estructura alternativa a partir de diagramas de flujo, porque ellos muestran explícitamente el flujo de ejecución. En este tipo de diagramas los bloques secuenciales son mostrados en rectángulos y las expresiones booleanas a partir de rombos.

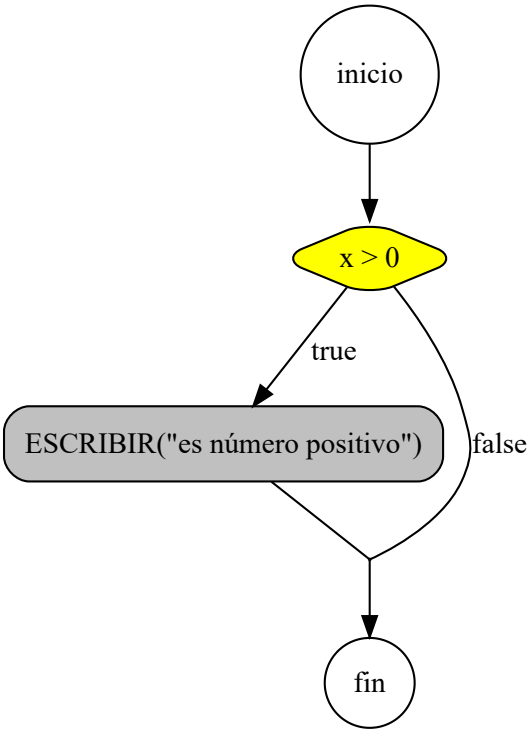
Los rombos muestran claramente las decisiones tomadas y como el flujo del programa se bifurca. Si la evaluación de la expresión booleana se evalúa a verdadero se ejecuta la secuencia de sentencias de la izquierda (graficado con un rectángulo de puntas redondeadas), si la expresión booleana se evalúa a falso se ejecuta la secuencia de sentencias de la derecha.

La expresión booleana se muestra a partir de un rombo y permite determinar la condición que seleccionará una secuencia de sentencias u otra. Los diagramas de flujo emplean notación gráfica. Sin embargo resulta engorroso leer un diagrama de flujo que modele el diseño de un algoritmo con varias estructuras de control combinadas.

Ahora observemos el flujo de datos de una alternativa doble. A continuación mostramos un diagrama de flujo para el último pseudocódigo que mostramos anteriormente:



Veamos como se distingue una alternativa simple de una alternativa doble segun el flujo de datos. Observemos primero el flujo de datos de una alternativa simple:



6 Múltiples Alternativas combinadas

La decisión de incluir una alternativa en un programa, parte de una lectura cuidadosa de la especificación. En nuestro caso la especificación nos decía: Si el número es positivo escribir un mensaje Numero positivo, de lo contrario escribir un mensaje Numero no positivo. Veamos qué se puede hacer cuando se presentan tres o más alternativas.

Problema: Si el número es positivo escribir el mensaje Numero positivo, si el número es igual a 0 escribir el mensaje Igual a 0 , y si el número es negativo escribir el mensaje ”Número negativo”. Una posibilidad es considerar que se trata de una estructura con dos casos como antes, sólo que el segundo caso es complejo (es nuevamente una alternativa):

```
ALGORITMO esPositivo() RETORNA ∅
  (* Determina si un entero es positivo,
       igual a cero ó negativo *)
  ENTERO num
  ESCRIBIR (”Ingrese un numero entero”)
  LEER(num)
  SI (num > 0) ENTONCES
      ESCRIBIR(num + ” es POSITIVO”)
  SINO
      SI (num = 0) ENTONCES
          ESCRIBIR(num + ” es CERO”)
      SINO
          ESCRIBIR(num + ”es NEGATIVO”)
      FIN SI
  FIN SI
FIN ALGORITMO esPositivo
```

```
<? php
  // int $num
  // ingreso de datos
  echo ”Ingrese un número entero”;
  $num = trim(fgets(STDIN));
  // Mostramos por pantalla resultados
  if ($num > 0)
    echo $num . ” es POSITIVO”;
  else
      if ($num == 0)
        echo $num . ” es CERO”;
      else
        echo $num . ” es NEGATIVO”;
?>
```

Esta estructura se conoce como alternativas anidadas ya que dentro de una de las ramas de la alternativa (en este caso la rama del *else*) se anida otra alternativa. Pero ésta no es la única forma de implementarlo. Existe otra construcción, equivalente a la anterior pero que no exige indentaciones cada vez mayores en el texto. Se trata de la estructura selectiva ó alternativas múltiples.

7 Alternativas Anidadas

La decisíon de incluir una alternativa en un programa, parte de una lectura cuidadosa de la especificación Veamos qué se se puede hacer cuando se presentan treso más as alternativas. Siempre que trabajemos en el mismo conjunto de datos, podremos utilizar esta estructura condicional.

Problema: Supongamos que deseamos construir un programa que en función de la hora que ingrese el usuario saludarlo en distintos momentos del día:

```
ALGORITMO saludo() RETORNA ⌀
  (* Saludo en función de la hora
     que indica el usuario *)
  ENTERO hora
  ESCRIBIR (”Ingrese la hora”)
  LEER(hora)
  SI (hora < 13) ENTONCES
      ESCRIBIR("¡buen dia!")
  OTRO-SI (hora < 20 ) ENTONCES
      ESCRIBIR("¡buenas tardes!")
  SINO
      ESCRIBIR("¡buenas noches!")
  FIN SI

FIN ALGORITMO saludo
```

```
<? php
  // INT $hora
  // ingreso de datos
  echo ”Ingrese la hora ”;
  $hora = trim(fgets(STDIN));
  // Mostramos por pantalla resultados
  if ($hora < 13)
       echo "¡buen dia!";
  elseif ($hora < 20)
      echo "¡buenas tardes!";
  else
      echo "¡buenas noches!";

?>
```

Las instrucciones **if**, **elseif** y **else** son palabras reservadas.

Se evaĺua la primera alternativa, si es verdadera se ejecuta su cuerpo. De lo contrario se eva´ua la segun daalternativa, si es verdadera se ejecuta su cuerpo, etc. Finalmente, si todas las alternativas anteriores son falsas, se ejecuta el cuerpo del **else**.

8 Estructuras Selectivas

Cuando las estructuras anidadas se refieren al tratamiento del mismo subconjunto de datos, y estos subconjuntos son de presencia aleatoria y disjuntos entre sí, es posible modelarlos empleando un mecanismo de abreviación.. Esto es posible si las estructuras anidadas utilizan el mismo conjunto de variables en sus expresiones booleanas (en el pseudocódigo notar que las expresiones booleanas de las estructuras alternativas anidadas operan sobre la variable x). Con frecuencia aparecen problemas donde necesitamos realizar diferentes acciones dependiendo de distintos valores (una cantidad finita) que asume una variable. En estos casos debemos implementar diferentes acciones de acuerdo a la evaluación de una **expresión multivaluada**.

Por ejemplo, veamos el siguiente problema: **Problema:** Se desea diseñar un módulo que dado el número de día de la semana (del 1 al 7) devuelva el nombre de ese día como una cadena de caracteres. Por ejemplo, para el día 1 que devuelva ’Lunes’, para el 2, ’Martes’, y así sucesivamente. Para modelar este tipo de problemas contamos con una estructura especial, que llamaremos estructura alternativa múltiple, de la siguiente forma:

```
ALGORITMO nombreAlgoritmo() RETORNA ⌀
    (* .... *)
    SEGUN expresionMultivaluada HACER
        valor1: sentencias si expresionMultivaluada = valor1
        valor2: sentencias si expresionMultivaluada = valor2
        ...
        valorn: sentencias si expresionMultivaluada = valorn
    FIN SEGUN
    ....
FIN ALGORITMO nombreAlgoritmo
```

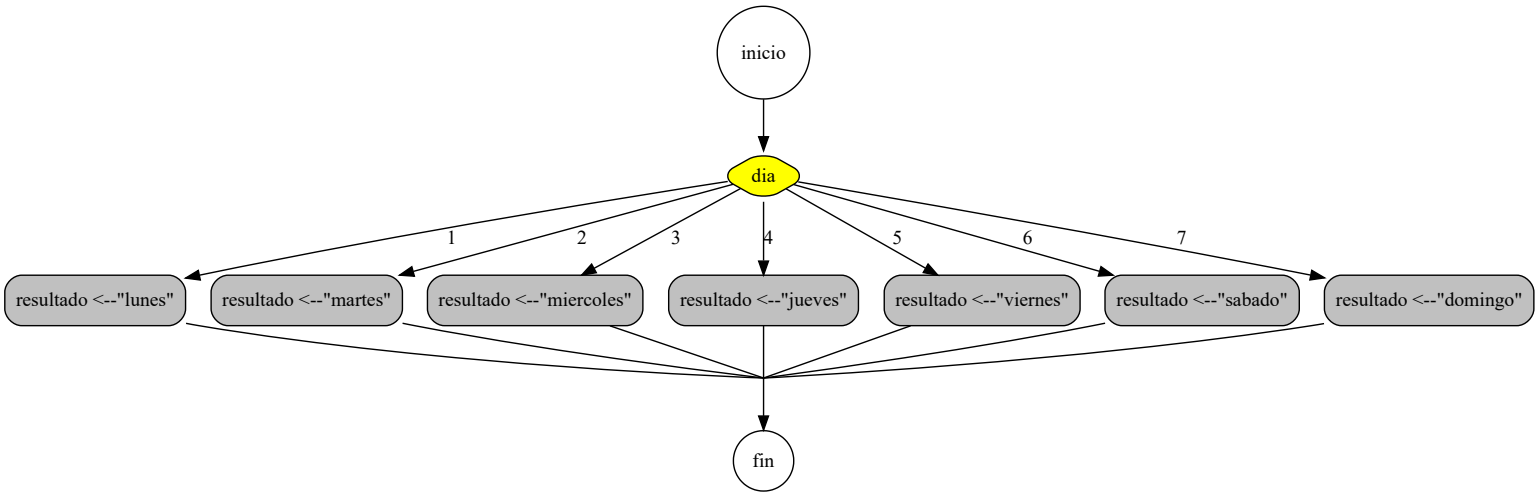
> ATENCION: En la estructura **alternativa múltiple** o tambien llamada **estructura selectiva**, la expresión no es booleana, sino de otro tipo (entero o caracter). Si la expresión fuera booleana deberiamos utilizar una estructura alternativa. En algunos lenguajes de programación el resultado de la expresión se restringe a los tipos entero y símbolo u otro tipo discreto, aunque en otros lenguajes se puede usar también texto.

Apliquemos la estructura de selectiva para resolver el problema de los días de la semana. La expresión a evaluar es el número entero día. Los valores posibles son los enteros del 1 al 7, y los valores no válidos (representado por la notación ∀o cuyo significado es ’para todo otro caso’) serán cualquier otro número. El diseño en pseudocódigo podría escribirse de la siguiente manera:

```
ALGORITMO diaSermana(ENTERO dia) RETORNA ⌀
  (* Devuelve el nombre del dia de la semana de
  acuerdo a un numero entero 1 <= dia <= 7 *)
  TEXTO resultado
  ENTERO dia
  LEER(dia)
  SEGUN dia HACER
    1: resultado ← "Lunes"
    2: resultado ← "Martes"
    3: resultado ← "Miercoles"
    4: resultado ← "Jueves"
    5: resultado ← "Viernes"
    6: resultado ← "Sabado"
    7: resultado ← "Domingo"
    ∀o: resultado ← "Error"
  FIN SEGUN
  ESCRIBIR(" corresponde a: "+ resultado)
FIN ALGORITMO diaSemana ´
```

Por lo anterior, podemos resumir que la estructura selectiva evalúa una expresión multivaluada que puede tomar valores (o grupos de valores) distintos (e_1,e_2, … ,e_n).

Luego, **según el valor que tome la expresión multivaluada, se realiza sólo una de las acciones posibles, o lo que es lo mismo, el flujo del algoritmo sigue un determinado camino entre los n posibles caminos**. En la figura a continuación puede verse el diagrama de flujo para la situación de los días de la semana. La alternativa presenta 8 posibles caminos, del 1 al 7 son caminos válidos, pero si recibe un valor distinto va a devolver un texto Error.



El lenguaje PHP tiene un constructor para implementar una estructura de este tipo. Es la sentencia **switch** que mostramos a continuación:

```php
<? php
  // Declaración de variables
  // int $dia
  // string $resultado
  echo "Ingrese un dia de semana 1-7";
  $dia = trim(fgest(STDIN));
  switch ($dia) {
      case 1:
        $resultado = "Lunes";
        break;
      case 2:
        $resultado = "Martes";
        break;
      case 3:
        $resultado = "Miercoles";
        break;
      case 4:
        $resultado = "Jueves";
        break;
      case 5:
        $resultado = "Viernes";
        break;
      case 6:
        $resultado = "Sabado";
        break;
      case 7:
        $resultado = "Domingo";
        break;
      default:
        $resultado = "Error";
        break;
  }
  echo  $dia . "corresponde a: " . $resultado;

?>
```

La forma general de *switch* es al siguiente:

```
  switch (expresiónMultivalor) {  ´
    case valor1: conjunto de sentencias; break;
    case valor2: conjunto de sentencias; break;
    case valor3: conjunto de sentencias; break;
    ....;
    case valorn: conjunto de sentencias; break;
    default: conjunto de sentencias; break;
  }´
```

La sentencia *switch* evalúa la ExpresionMultivalor y ejecuta el ConjuntoDeSentencias que aparece junto a la cláusula **case** cuyo valor corresponda con ExpresionMultivalor. Cada sentencia *case* debe ser única y el valor que evalúa debe ser del mismo tipo que el devuelto por la ExpresionMultivalor de la sentencia switch.

Las sentencias *break* que aparecen tras cada **ConjuntoDeSentencias** provocan que el control salga del switch y continúe con la siguiente instrucción al *switch*. Las sentencias break son necesarias porque sin ellas se ejecutarían secuencialmente las sentencias case siguientes. Existen ciertas situaciones en las que se desea ejecutar secuencialmente algunas o todas las sentencias case, para lo que habrá que eliminar algunos *break*.

Finalmente, se puede usar la sentencia *default* para manejar los valores que no son explícitamente contemplados por alguna de las sentencias case. Su uso es altamente recomendado.

9 Ej. Resueltos en clase

9.1 Ej. 1 TP5

Especificar un módulo esPar cuya entrada es un número y el retorno es true si el número ese par, false caso contrario. El programa principal deberá solicitar un número y utilizando el módulo esPar, informe por ejemplo: “el nro 10 es: par” o bien “el nro 51 es: impar”.

```
MODULO esPar2(ENTERO numero) RETORNA LÓGICO
  RETORNAR ((numero MOD 2) = 0)
FIN MODULO esPar2

ALGORITMO principal() RETORNA VACIO
  ENTERO num
  ESCRIBIR(“Ingrese un numero entero”)
  LEER(num)
  SI ( esPar(num) ) ENTONCES
    ESCRIBIR(“el nro ” + num + “es: “ + ” par”)
  SINO
    ESCRIBIR(“el nro ” + num + “es: “ + ”impar”)
  FIN SI
FIN ALGORITMO principal
```

```
<?php

  /** Función que verifica si un numero entero es par
   * @param INT $numero
   * @return BOOLEAN
  */
  function esPar($numero) {
     // BOOLEAN $respuesta
     $respuesta = ( ($numero % 2) == 0 );
     return $respuesta;
  }

  // principal
 // INT $num
 // BOOLEAN $rta
    echo "Ingrese un numero ";
    $num = trim(fgets(STDIN));
    $rta = esPar($num);
    if ($rta) {
        echo "el numero " . $num . " es: par";
    }
    else  {
        echo "el numero " . $num . " es: impar";
    }

    ?>


?>
```

9.2 Ej1 TP5 Otra solución

Especificar un módulo esPar cuya entrada es un número y el retorno es true si el número ese par, false caso contrario. El programa principal deberá solicitar un número y utilizando el módulo esPar, informe por ejemplo: “el nro 10 es: par” o bien “el nro 51 es: impar”.

```
(** este modulo verifica si un numero entero es par o no *)
MODULO esPar(ENTERO numero) RETORNA LÓGICO
    LOGICO respuesta
    respuesta <- ((numero MOD 2) = 0)
    RETORNAR respuesta
FIN MODULO esPar

ALGORITMO principal() RETORNA ⌀
  ENTERO num
  LOGICO rta
  ESCRIBIR(“Ingrese un numero entero”)
  LEER(num)
  rta <- esPar(num)
  SI ( rta ) ENTONCES
     ESCRIBIR(“el nro ” + num + “es: par”)
  SINO
     ESCRIBIR(“el nro ” + num + “es: impar”)
  FIN SI
FIN ALGORITMO principal
```

```
<?php

  /** Función que verifica si un numero entero es par
   * @param INT $numero
   * @return BOOLEAN
  */
  function esPar($numero) {
     return  ( ($numero % 2) == 0 );
  }

  // principal
 // INT $num
    echo "Ingrese un numero ";
    $num = trim(fgets(STDIN));
    if ( esPar($num) ) {
        echo "el numero " . $num . " es: par";
    }
    else  {
        echo "el numero " . $num . " es: impar";
    }

    ?>


?>
```

9.3 Ej.2 TP5

Especificar un módulo que a partir de dos números (num1, num2) retorne true si el primero es mayor al segundo (num1>num2), false caso contrario.

- Pseudocódigo

```
(** Este modulo verifica si el primer parametor es mayor que el segundo *)
MODULO esMayor(ENTERO numero1, numero2) RETORNA LOGICO
  LOGICO rta
  rta <- (numero1 > numero2)
  RETORNAR rta
FIN MODULO esMayor

ALGORITMO principal() RETORNA ⌀
  ENTERO num1, num2
  ESCRIBIR(“Ingrese un numero entero”)
  LEER(num1)
  ESCRIBIR(“Ingrese otro numero entero”)
  LEER(num2)
  SI ( esMayor(num1, num2) ) ENTONCES
      ESCRIBIR( num1 + “es mayor que “+ num2)
  SINO
      ESCRIBIR( num1 + “NO es mayor que “+ num2)
  FIN SI
FIN ALGORITMO principal
```

- PHP

```
<?php

  /** Función que verifica si un primer numero
   * es mayor que un segundo numero recibido como parametro
   * @param INT $numero1
   * @param INT $numero2
   * @return BOOLEAN
  */
  function esMayor($numero1, $numero2) {
    // BOOLEAN $rta
    if ($numero1 > $numero2) {
        $rta = true;
    }
    else {
        $rta = false;
    }
     return $rta;
  }
  function esMayorBis($numero1, $numero2) {
     return ($numero1 > $numero2);
  }


  // principal
  // INT $num1
  // INT $num2
    echo "Ingrese un numero entero";
    $num1 = trim(fgets(STDIN));
    echo "Ingrese otro numero entero";
    $num2 = trim(fgets(STDIN));
    if ( esMayor($num1, $num2) ) {
        echo "el numero " . $num1 . " es mayor que " . $num2;
    }
    else  {
        echo "el numero " . $num1 . " NO es mayor que " . $num2;
    }

    ?>
```

10 Parcialito 1

10.1 Ej 1

Diseñe en Pseudocódigo y traduzca PHP. Documente adecuadamente. En la resolución del ejercicio: los nombres de los parámetros actuales no deben coincidir con el nombre de los parámetros formales.

1. Desarrolle un módulo que a partir de la altura y la base calcule el área de un triángulo.
2. Desarrolle un módulo que a partir del radio calcule el área de un círculo.
3. Un PROGRAMA PRINCIPAL que interactúe con el usuario de la siguiente manera:
    a. El programa solicita al usuario que indique si desea calcular (1) el área de un triangulo o (2) el área de un círculo

b. Si el usuario elige la opción 1, el programa solicita la altura y la base y muestra la superficie del triangulo
c. Si el usuario elige la opción 2, el programa solicita el radio y muestra la superficie del círculo
d. Si el usuario elige una opción errónea, el programa le indica que la opción elegida es inválida.

- Pseudocódigo

```
(** Este modulo obtiene el area de un triangulo *)
MODULO areaTriangulo(REAL altura,base) RETORNA REAL
  REAL area
  area <- (base * altura)/2
  RETORNAR area
FIN MODULO areaTriangulo

(** Este modulo obtiene el area de un circulo*)
MODULO areaCirculo(REAL radio) RETORNA REAL
  REAL areaCirculo
  areaCirculo <- PI * potencia(radio, 2)
  RETORNAR areaCirculo
FIN MODULO areaTriangulo

ALGORITMO principal() RETORNA ∅
  ENTERO opcion
  REAL alturaT, baseT, radioC, res1, res2
  ESCRIBIR("Menú:")
  ESCRIBIR(“1. Obtener area Triángulo”)
  ESCRIBIR(“1. Obtener area Circulo”)
  LEER(opcion)
  SI ( opcion = 1) ENTONCES
      ESCRIBIR("Ingrese la altura del triangulo")
      LEER(alturaT)
      ESCRIBIR("Ingrese la baser del triangulo")
      LEER(baseT)
      res1 <- areaTriangulo(alturaT,baseT)
      ESCRIBIR("El area del triangulo es:"+ res1)
  OTROSI (ocion = 2) ENTONCES
      ESCRIBIR("Ingrese el radio del Circulo")
      LEER(radioC)
      res2 <- areaCirculo(radioC)
      ESCRIBIR("El area del circulo es:"+ res2)
  SINO
      ESCRIBIR(“La opción elegida es inválida“)
  FIN SI
FIN ALGORITMO principal
```

- PHP

```php
<?php
/** obtener el area de un triángulo
* @param FLOAT $altura
* @param FLOAT $base
* @return FLOAT */
function areaTriangulo($altura, $base) {
  // FLOAT $area
  $area = ($base * $altura) / 2;
  return $area;
}

// principal
// INT $opcion
// FLOAT $alturaT, $baseT, $radioC, $res1, $res2
  echo "Menu \n";
  echo "1: Obtener Area Triángulo \n";
  echo "2: Obtener Area Círculo \n";
  $opcion = trim(fgets(STDIN));
  if ($opcion == 1) {
      echo "Ingrese la altura";
      $alturaT = trim(fgets(STDIN));
      echo "Ingrese la base";
      $baseT = trim(fgets(STDIN));
      $res1 = areaTriangulo($alturaT, $baseT);
      echo "El area del triangulo es: " . $res1;
  }
  elseif ($opcion == 2) {
      echo "Ingrese el radio";
      $radioC = trim(fgets(STDIN));
      $res2 = areaCirculo($radioC);
      echo "El area del circulo es: " . $res2;
  }
  else {
      echo "La opcion elegida es inválida";
  }

/** obtener el area de un circulo
* @param FLOAT $radio
* @return FLOAT */
function areaCirculo($radio) {
  //FLOAT $areaCirc
  $areaCirc = M_PI * pow($radio, 2);
  return $areaCirc;
}



?>
```

10.2 Traza

Realizar una traza del ejercicio anterior suponiendo que el usuario ingresa la opción 1 con los siguientes datos; altura del triángulo 2.5 y base 1.7:

principal()

$opcion	$alturaT	$baseT	$radioC	$res1	$res2	Salida
1	2.5	1.7		2.125		Menu
						1. Obtener Area Triangulo
						2. Obtener Area Circulo
						Ingrese la altura
						Ingrese la base
						El area del triangulo es: 2.125

areaTriangulo(2.5, 1.7)

$altura	$base	$area	Valor retornado
2.5	1.7	2.125	2.125

$altura	$base	$area	Valor retornado

10.3 Ej 2

Implementar en pseudocódigo ó en PHP Declare variables y Documente correctamente. Utilice nombres significativos para funciones y variables En un Supermercado de la ciudad de Neuquén se realiza un descuento a sus clientes en el sector de cajas y se requiere un programa que establezca el valor final de la compra del cliente.

En particular, el porcentaje de descuento se determina con los siguientes criterios:

- Si el pago es con tarjeta de crédito el porcentaje de descuento es 5%.
- Si el pago es con tarjeta de débito el porcentaje de descuento es 10%
- Si el pago es en efectivo se sigue los siguientes criterios:
    - Si el importe es hasta $500, el descuento es del 8%
    - Si el importe de la compra entre $500 y $1000, el descuento es 10%
    - Si el importe de la compra es superior a $1000, el descuento es 12%
- Si el pago fuera otro medio de pago, distinto a los anteriores, el porcentaje de descuento es 0%.

Diseñe en Pseudocódigo y traduzca PHP:

a. Una función “descontarPorcentaje” que recibe por parámetro el importe total de la compra, el porcentaje a descontar y devuelve el precio que el cliente debe abonar. (Por ejemplo, si el monto de la compra es $100 y porcentaje de descuento es 10%, el módulo debe retornar $90)
b. Una función “obtenerMontoFinal” que recibe por parámetro: forma de pago (“E” = Efectivo, “D” = Tarjeta Débito, “TC” = Tarjeta Crédito), importe total de la compra. La función retorna el monto final de la compra. obs.: En el cuerpo de la función se debe determinar el porcentaje de descuento y descontar el porcentaje
c. Un Programa Principal, que solicita el valor de la compra y la forma de pago. Luego invoca a la función obtenerMontoFinal y, finalmente, muestra el resultado informando el precio final de la compra.
d. Realizar una traza al algoritmo Supermercado con los siguientes valores:

- importe total = 1000 / Forma de pago = D
- importe total = 800 / Forma Pago = E
- Pseudocódigo

```
(** Este modulo calcula el precio que el cliente debe abonar *)
MODULO descontarPorcentaje(REAL importe, porcentaje) RETORNA REAL
  REAL rta
  rta <- importe - (( importe * porcentaje) / 100)
  RETORNAR rta
FIN MODULO descontarPorcentaje

(** Este modulo obtiene el monto final en función del importe y forma de pago *)
MODULO obtenerMontoFinal(TEXTO formaPago, REAL importeTotal) RETORNA REAL
  REAL monto
  SI (formaPago = "TC") ENTONCES
     monto <- descontarPorcentaje(importeTotal, 5)
  OTROSI (formaPago = "D") ENTONCES
     monto <- descontarPorcentaje(importeTotal, 10)
  OTROSI (formaPago = "E") ENTONCES
     SI (importeTotal < 500) ENTONCES
          monto <- descontarPorcentaje(importeTotal, 8)
     OTROSI (immporteTotal >=500 AND importeTotal < 1000) ENTONCES
          monto <- descontarPorcentaje(importeTotal, 10)
     SINO
          monto <- descontarPorcentaje(importeTotal, 12)
     FIN SI
  SINO
     monto <- importeTotal
  FIN SI
  RETORNAR monto
FIN MODULO obtenerMontoFinal

ALGORITMO principal() RETORNA  ∅
  ENTERO opcion
  TEXTO fPago
  REAL precioFinal, valorCompra
  ESCRIBIR("Ingrese el valor de la compra")
  LEER(valorCompra)
  ESCRIBIR("Ingrese forma de pago (D: Debito, E: efectivo, TC: Credito): ")
  LEER(fPago)
  precioFinal <- obtenerMontoFinal(fPago, valorCompra)
  ESCRIBIR("El precio final de la compra es:"+ precioFinal)
FIN ALGORITMO principal
```