UT3.ProgramaciónBasadaEnLenguajesDeMarcasConCodigoEmbebido

Estructuras de Control

- if […] else […] elseif […]
- switch […] case (admite expresiones junto al switch(expr) y condiciones complejas en los case)

```
$a=5;
switch($a) { // selecciona aquel case que valga true
      case (5): echo "Mayor que 5"; break;
      case ($a<5): echo "Menor que 5"; break;
      case ($a>5): echo "Igual que 5"; break;
      default: echo "Imposible"; break;
}
```

Estructuras repetitivas

- while
- for
- foreach

```
foreach(exprMatriz as $valor) {
      sentencias;
}
foreach(expMatriz as $clave=>$valor) {
      sentencias;
}
```

- do … while
- break [n];
- continue [n];

Arrays

- Almacena una secuencia de valores
- Puede tener un número variable de elementos
- Los índices pueden ser no consecutivos
- Cada elemento puede tener un valor de distinto tipo
- También es posible que a su vez un elemento sea tipo array
- Un array bidimensional será un array de arrays y así sucesivamente
- Dos tipos: - escalares - asociativos

- **Creación:**
  - Array escalar

```
$paises=array("Alemania","Austria","Bélgica");
```

- **Recorrido**:

```
for($i=0;$i<count($paises);$i++) {
      echo "País[$i]: ",$paises[$i],"\n";
}
```

- Más conveniente

```
foreach($paises as $clave=>$valor) {
      echo "País[$clave]: $valor\n";
}
```

- $paises=array(1=>"Alemania","Austria",7=>"Bélgica");

En éste último hay que tener cuidado en el recorrido:

```
for($i=0;$i<count($paises);$i++) {
      echo "País[$i]: ",$paises[$i],"\n";
}
```

```
País[0]:   <warning>PHP   Notice:      Undefined   offset:   0   in
phar://eval()'d code on line 2</warning>
País[1]: Alemania
País[2]: Austria
```

- Creación dinámica

```
  for($i=1;$i<100;$i++) {
      $numeros[$i]=0;
```

```
  }
- Creación por asignación directa:
$paises=["Alemania","Austria","Bélgica"]
- Arrays bidimensionales
$a=[[1,2,3],[4,5,6]], $b=[[7,8,9],[10,11,12]];
- Arrays asociativos
$a=[1=>"Alemania","Austria",7=>"Bélgica"]
```

- **Eliminar elementos:**
  - ***unset***

- **Imprimir un array: print_r**

- **Inicializar un array vacío:**
  $a=array(); $b=[];

- **¿Qué estructura elegirías para representar la siguiente tabla?**

País	Capital	Extensión	Habitantes
Alemania	Berlín	557046	78420000
Austria	Viena	83849	7614000
Bélgica	Bruselas	30518	9932000

- **Funciones que trabajan con arrays**

  - `$meses=[1=>'Enero',2=>'Febrero',3=>'Marzo']`

  - array_keys(array $array, mixed $search_value, bool $strict= **false**): array

```
>>> array_keys($meses)
=> [
     1,
     2,
     3,
   ]

>>> array_keys($meses,"Marzo")
=> [
     3,
   ]
```

  - array_values(array $array): array

```
>>> array_values($meses)
=> [
     "Enero",
     "Febrero",
     "Marzo",
   ]
```

  - preg_grep(string $pattern, array $input, int $flags = 0): array

```
>>> preg_grep('/.*ero$/',$meses)
=> [
     1 => "Enero",
     2 => "Febrero",
   ]
```

    *** Ver Anexo I sobre expresiones regulares**

  - array_search(mixed $needle, array $haystack, bool $strict=**false**): mixed

```
>>> array_search('Marzo',$meses)
=> 3
```

```
- in_array(mixed $needle, array $haystack, bool $strict = false): bool
      >>> in_array("Febrero",$meses)
      => true

- array_pop(array &$array): mixed
      >>> array_pop($meses)
      => "Marzo"
      >>> $meses
      => [
           1 => "Enero",
           2 => "Febrero",
         ]
- array_push(array &$array, mixed $value1, mixed $... = ?): int
      >>> array_push($meses,"Marzo","Abril")
      => 4
      >>> $meses
      => [
           1 => "Enero",
           2 => "Febrero",
           3 => "Marzo",
           4 => "Abril",
         ]
- array_shift(array &$array): mixed
      >>> array_shift($meses)
      => "Enero"
      >>> $meses
      => [
           "Febrero",
           "Marzo",
           "Abril",
         ]
- array_unshift(array &$array, mixed $... = ?): int
      >>> array_unshift($meses,"Enero")
      => 4
      >>> $meses
      => [
           "Enero",
           "Febrero",
           "Marzo",
           "Abril",
         ]
- array_merge(array $array1, array $... = ?): array
      >>> $meses
      => [
           "Enero",
           "Febrero",
           "Marzo",
           "Abril",
         ]
      >>> $meses2=["Mayo","Junio","Julio","Agosto"]
      => [
           "Mayo",
           "Junio",
           "Julio",
           "Agosto",
         ]
      >>> $meses3=["Septiembre","Octubre","Noviembre","Diciembre"]
```

```
        => [
             "Septiembre",
             "Octubre",
             "Noviembre",
             "Diciembre",
           ]
        >>> array_merge($meses,$meses2,$meses3)
        => [
             "Enero",
             "Febrero",
             "Marzo",
             "Abril",
             "Mayo",
             "Junio",
             "Julio",
             "Agosto",
             "Septiembre",
             "Octubre",
             "Noviembre",
             "Diciembre",
           ]
        >>> $meses=array_merge($meses,$meses2,$meses3)
        => [
             "Enero",
             "Febrero",
             "Marzo",
             "Abril",
             "Mayo",
             "Junio",
             "Julio",
             "Agosto",
             "Septiembre",
             "Octubre",
             "Noviembre",
             "Diciembre",
           ]
- array_slice(array $array, int $offset, int $length = null, bool $preserve_keys = false): array

        >>> array_slice($meses,5,3)
        => [
             "Junio",
             "Julio",
             "Agosto",
           ]
- implode(string $glue, array $pieces): string
- implode(array $pieces): string
        >>> implode($meses)
        =>
        "EneroFebreroMarzoAbrilMayoJunioJulioAgostoSeptiembreOctubreNovi
        embreDiciembre"
        >>> implode(" ",$meses)
        => "Enero Febrero Marzo Abril Mayo Junio Julio Agosto Septiembre
        Octubre Noviembre Diciembre"
        >>> implode(" | ",$meses)
        => "Enero | Febrero | Marzo | Abril | Mayo | Junio | Julio |
        Agosto | Septiembre | Octubre | Noviembre | Diciembre"
```

- array_reverse(array $array, bool $preserve_keys = **false**): array

```
>>> array_reverse($meses)
=> [
     "Diciembre",
     "Noviembre",
     "Octubre",
     "Septiembre",
     "Agosto",
     "Julio",
     "Junio",
     "Mayo",
     "Abril",
     "Marzo",
     "Febrero",
     "Enero",
   ]
```

- range(mixed $start, mixed $end, number $step = 1): array

```
>>> range(2,7)
=> [
     2,
     3,
     4,
     5,
     6,
     7,
   ]
>>> range(10,60,5)
=> [
     10,
     15,
     20,
     25,
     30,
     35,
     40,
     45,
     50,
     55,
     60,
   ]
```

- shuffle(array &$array): bool

```
>>> $numeros=range(1,12)
=> [
     1,
     2,
     3,
     4,
     5,
     6,
     7,
     8,
     9,
     10,
     11,
     12,
   ]
>>> shuffle($numeros)
=> true
```

```
      >>> $numeros
      => [
           8,
           9,
           7,
           3,
           11,
           5,
           6,
           4,
           2,
           10,
           1,
           12,
         ]
- sort(array &$array, int $sort_flags = SORT_REGULAR): bool
      >>> sort($numeros)
      => true
      >>> $numeros
      => [
           1,
           2,
           3,
           4,
           5,
           6,
           7,
           8,
           9,
           10,
           11,
           12,
         ]
- rsort(array &$array, int $sort_flags = SORT_REGULAR): bool
      >>> rsort($numeros)
      => true
      >>> $numeros
      => [
           12,
           11,
           10,
           9,
           8,
           7,
           6,
           5,
           4,
           3,
           2,
           1,
         ]
- asort(array &$array, int $sort_flags = SORT_REGULAR): bool
      >>> $meses=[1=>'Enero',12=>'Diciembre',3=>'Marzo',4=>'Abril']
      => [
           1 => "Enero",
           12 => "Diciembre",
           3 => "Marzo",
           4 => "Abril",
```

```
    ]
>>> asort($meses)
=> true
>>> $meses
=> [
     4 => "Abril",
     12 => "Diciembre",
     1 => "Enero",
     3 => "Marzo",
    ]
```

- arsort(array &$array, int $sort_flags = SORT_REGULAR): bool

```
>>> arsort($meses)
=> true
>>> $meses
=> [
     3 => "Marzo",
     1 => "Enero",
     12 => "Diciembre",
     4 => "Abril",
    ]
```

- ksort(array &$array, int $sort_flags = SORT_REGULAR): bool

```
>>> ksort($meses)
=> true
>>> $meses
=> [
     1 => "Enero",
     3 => "Marzo",
     4 => "Abril",
     12 => "Diciembre",
    ]
```

- krsort(array &$array, int $sort_flags = SORT_REGULAR): bool

```
>>> krsort($meses)
=> true
>>> $meses
=> [
     12 => "Diciembre",
     4 => "Abril",
     3 => "Marzo",
     1 => "Enero",
    ]
```

- **Arrays predefinidos**

  - **$_GET** contiene los datos enviados con el ***method="get"***
  - **$_POST** contiene los datos enviados con el ***method="post"***
  - **$_COOKIE** contiene las ***cookies***
  - **$_REQUEST** contiene todas las variables incluidas en las anteriores
  - **$_SERVER**
    - "**HTTP_USER_AGENT**" contiene el ***User-Agent*** del cliente
    - "**REMOTE_ADDR**" dirección IP desde la cual está viendo la página el usuario.
    - "**REMOTE_PORT**" puerto empleado por la máquina del usuario para comunicarse con el servidor web.

    https://www.php.net/manual/es/reserved.variables.server.php

ANEXO I

Expresiones regulares

* PHP permite utilizar **funciones** para expresiones regulares.
* Una expresión regular permite **comparar un patrón frente a un texto**, para comprobar si el texto contiene lo especificado en el patrón.
* Ejemplos de patrones de búsqueda:
  - Patrón: **in**
    Coindicen:
    intensidad
    c**in**ta
    interior
  - Patrón: **[mp]adre**
    Coindicen:
    Mi **madre** se llama Luisa
    Tu **padre** es jardinero

* El punto

- El punto representa **cualquier carácter**. Desde la A a la Z (en minúscula y mayúscula), del 0 al 9, o algún otro símbolo.

ca.a coincide con *ca**n**a*, *ca**m**a*, *ca**s**a*, *ca**j**a*, etc…

No coincide con *ca**st**a* ni *caa*

* Principio y fin de cadena

- Si queremos indicar al patrón qué es el principio de la cadena o qué es el final, debemos hacerlo con **^ para inicio y $ para final**.

"*^olivas*" coincide con "***olivas** verdes*",

pero no con "*quiero olivas*"

* Cuantificadores

- Para indicar que cierto elemento del patrón va a repetirse un **número indeterminado de veces**, usaremos **+** (una o más veces) o ***** (cero o más veces) .

"gafas+" coincide con "*gafa***ssss**" pero no con "*gafa*"

*"clo*aca"* coincide con "*claca*", "*cloaca*", "*clooooooooaca*", etc..

* El interrogante indica que un elemento **puede que esté (una vez) o puede que no**:

  "coches?" coincide con *"coche"* y con *"coches"*

* Las llaves **{ }** definen la **cantidad de veces que va a repetirse el elemento :**

  "abc{4}" coincide con "*abcccc*"

  pero no con "*abc*" ni "*abcc*", etc…

  "abc{1,3}" coincide con "*abc*",

  "*abcc*", "*abccc*", pero no con "*abcccc*"

* Si un parámetro queda vacío, significa “un **número indeterminado**“. Por ejemplo:

  “x{5,}” la x ha de repetirse 5 veces, o más.

* **Rangos**
  - Los corchetes **[]** incluidos en un patrón permiten especificar el **rango de caracteres** válidos a comparar.

“c[ao]sa” coincide con “*casa*” y con “*cosa*”

“[a-f]” coincide con todos los caracteres alfabéticos de la “*a*” a la “*f*”

“[0-9][2-6][ANR]” coincide con “*12A*“, “*35N*“, “*84R*“, etc..

pero no con “*2**1**A*“, ni “*33**L*“, ni “*3A*“, etc…

* Dentro de los corchetes el símbolo ^ es un negador, es decir:

  “[^a-Z]” coincidirá con cualquier texto que NO tenga ningún carácter alfabético (ni minúsculas ni mayúsculas)

  “^@ ” coincide con cualquier carácter excepto “@” y “*espacio*”

* Alternancia

- Para alternar entre varias opciones usaremos el símbolo **|.** Si una de las opciones coincide el patrón será cierto.

  “aleman(ia|es)” coincide con “*aleman**ia***” y con “*aleman**es***”

  “(norte|sur|este|oeste)” coincide con cualquiera de los puntos cardinales.

* Agrupadores

- Los paréntesis nos sirven para agrupar un subconjunto.

"(abc)+" coincide con "*abc*", "*abcabc*", "*abcabcabc*", etc
"ca(sca)?da" coincide con "*cascada*" y con "*cada*"

* Escapar caracteres

- Si queremos utilizar caracteres especiales en el patrón hubiese sin que se interprete como metacarácter, tendremos que "escaparlo". Esto se hace poniendo una barra invertida justo antes:

"\." o "*"