# Algoritmos – TUPAR

Punteros y Estructuras

# ¿Cómo se maneja la memoria?

- Al definir una variable, un lenguaje reserva un lugar en memoria destinado a almacenar el valor de la misma.

# ¿Cómo se maneja la memoria?

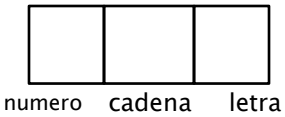
- Al definir una variable, un lenguaje reserva un lugar en memoria destinado a almacenar el valor de la misma.

```
$numero;
$cadena;
$letra;


$numero=6;
$cadena="hola";
$letra='l';


$numero=7;
$letra='a';

echo $numero . ", " . $cadena . ", " . $letra;
```

# ¿Cómo se maneja la memoria?

- Al definir una variable, un lenguaje reserva un lugar en memoria destinado a almacenar el valor de la misma.

```
$numero;
$cadena;
$letra;
```




```
$numero=6;
$cadena="hola";
$letra='l';


$numero=7;
$letra='a';

echo $numero . ", " . $cadena . ", " . $letra;
```

# ¿Cómo se maneja la memoria?

- Al definir una variable, un lenguaje reserva un lugar en memoria destinado a almacenar el valor de la misma.

```
$numero;
$cadena;
$letra;
```

```
$numero=6;
$cadena="hola";
$letra='l';
```


```
$numero=7;
$letra='a';

echo $numero . ", " . $cadena . ", " . $letra;
```

# ¿Cómo se maneja la memoria?

- Al definir una variable, un lenguaje reserva un lugar en memoria destinado a almacenar el valor de la misma.

```
$numero;
$cadena;
$letra;
```

```
$numero=6;
$cadena="hola";
$letra='l';
```


```
$numero=7;
$letra='a';
```

```
echo $numero . ", " . $cadena . ", " . $letra;
```

7, hola, a

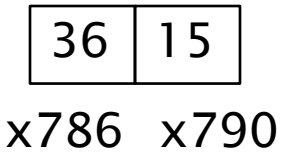
# ¿Qué son los punteros?

- Una variable guarda un VALOR del tipo de dato declarado.

- Un puntero es una variable que guarda un DIRECCION a una variable del tipo de dato que apunta.

# Más gráficos

Tengo una variable a la que le asigno el valor 36.
Tengo otra variable que le asigno el valor 15.
El lenguaje guarda los enteros en un lugar en la memoria.
Por ejemplo:



# Más gráficos

Defino un puntero.

A este, se le pueden asignar direcciones de memoria en los que se encuentran enteros.

Por ejemplo:

# Más gráficos

Defino un puntero.

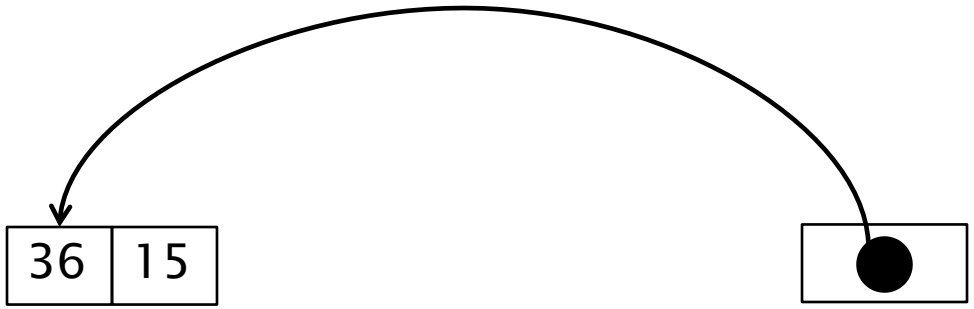
A este, se le pueden asignar direcciones de memoria en los que se encuentran enteros.

Por ejemplo:


Esto es posible si el lenguaje me permite definir punteros a enteros

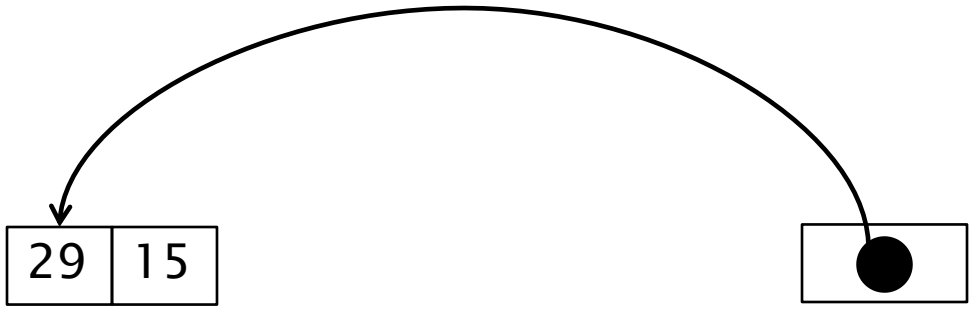
# Más gráficos

Se podría ver como que el puntero APUNTA al entero que tiene un 36



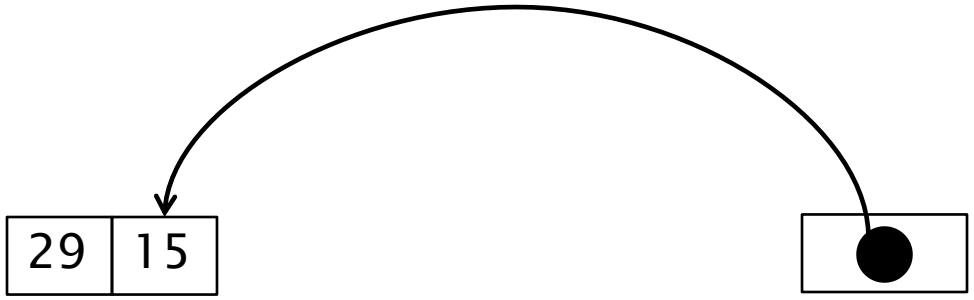
# Más gráficos

Si modifico el 36, el puntero apuntará al valor modificado.



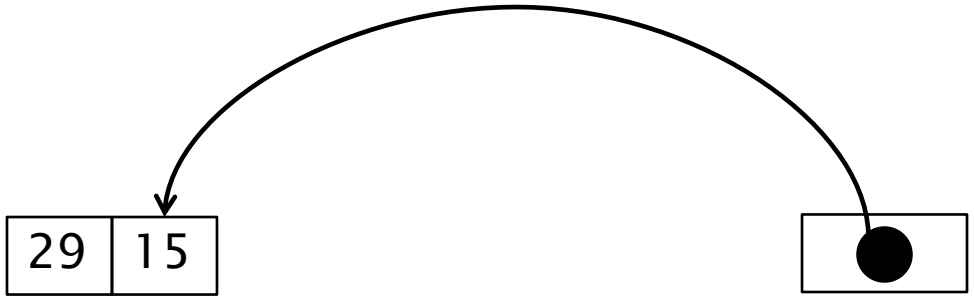
# Más gráficos

Puedo modificar el valor del puntero para que apunte a otro entero.



# Más gráficos

Lo más importante, puedo hacer que el puntero apunte a NUEVAS variables hasta ahora no definidas



# Más gráficos

Lo más importante, puedo hacer que el puntero apunte a NUEVAS variables hasta ahora no definidas

# Más gráficos

Lo más importante, puedo hacer que el puntero apunte a NUEVAS variables hasta ahora no definidas. Y darle valores a esta nueva variable desde el puntero.

# Cuestiones en php

- Tener en cuenta el tipo de cada variable. Por más que php tenga tipos dinámicos, no abusar de esto. Ser ordenado con las variables.
- Al usar punteros, estar atento ante las modificaciones.

Guarda con los pasajes por copia!

# Problemática

Se quiere realizar un programa que registre un padrón. En el mismo se tiene que guardar los datos de las personas con la siguiente información:

- Nombre
- Apellido
- DNI
- Domicilio
- Lugar dónde vota
- Mesa
- etc...

# Posible solución (1)

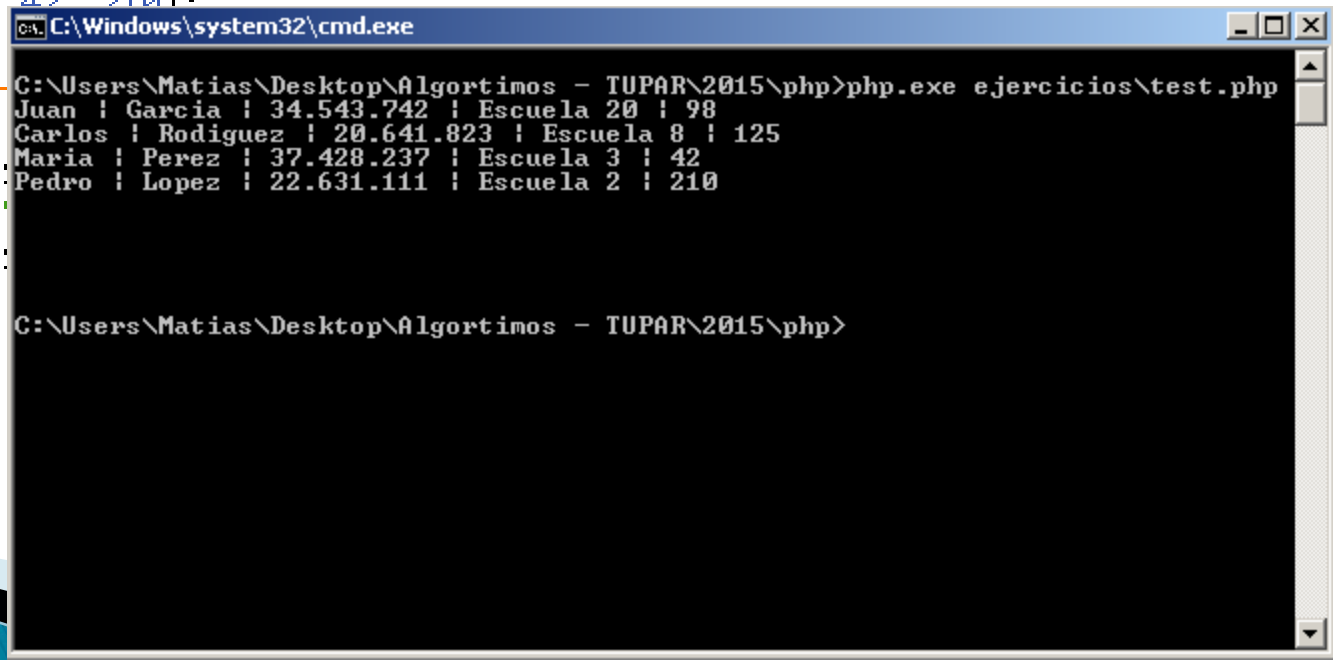
Declarar un arreglo para cada campo pedido.

```php
define("N", 4);

$nombres=array("Juan", "Carlos", "Maria", "Pedro");
$apellidos=array("Garcia", "Rodiguez", "Perez", "Lopez");
$DNI=array("34.543.742", "20.641.823", "37.428.237", "22.631.111");
$lugarVoto=array("Escuela 20", "Escuela 8", "Escuela 3", "Escuela 2");
$mesa=array(98, 125, 42, 210);

for ($i=0; $i<N; $i++){
    echo $nombres[$i] . " | ";
    echo $apellidos[$i] . " | ";
    echo $DNI[$i] . " | ";
    echo $lugarVoto[$i] . " | ";
    echo $mesa[$i] . "\n";
}
```

# Posible solución (1)

Declarar un arreglo para cada campo pedido.

```
define("N", 4);

$nombres=array("Juan", "Carlos", "Maria", "Pedro");
$apellidos=array("Garcia", "Rodiguez", "Perez", "Lopez");
$DNI=array("34.543.742", "20.641.823", "37.428.237", "22.631.111");
$lugarVoto=array("Escuela 20", "Escuela 8", "Escuela 3", "Escuela 2");
$mesa=array(98, 125, 42, 210);

for ($i=0; $i<N; $i+
    echo $nombres[$i
    echo $apellidos[
    echo $DNI[$i] . '
    echo $lugarVoto[
    echo $mesa[$i] .
}
```

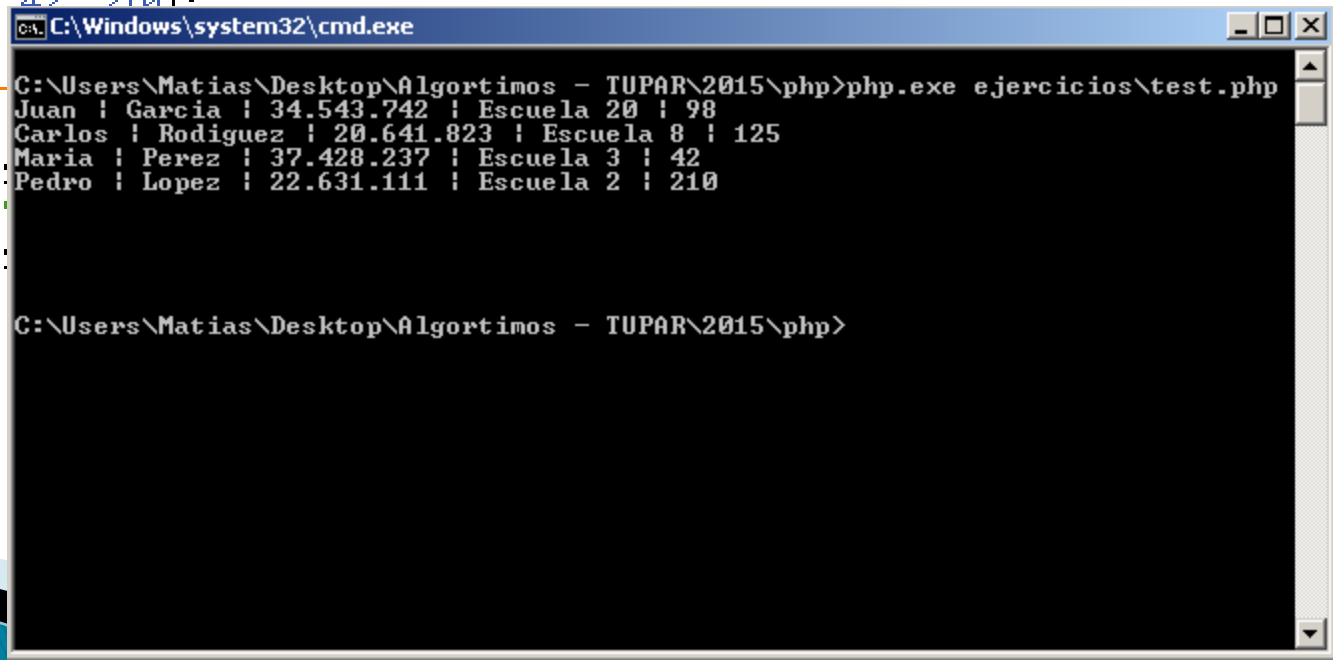


# Posible solución (1)

Dependo de las posiciones de los arreglos. Es dificil de mantener

Declarar un arreglo para cada campo pedido.

```php
define("N", 4);

$nombres=array("Juan", "Carlos", "Maria", "Pedro");
$apellidos=array("Garcia", "Rodiguez", "Perez", "Lopez");
$DNI=array("34.543.742", "20.641.823", "37.428.237", "22.631.111");
$lugarVoto=array("Escuela 20", "Escuela 8", "Escuela 3", "Escuela 2");
$mesa=array(98, 125, 42, 210);

for ($i=0; $i<N; $i+
    echo $nombres[$i
    echo $apellidos[
    echo $DNI[$i] . '
    echo $lugarVoto[
    echo $mesa[$i] .
}
```



# Posible solución (2)

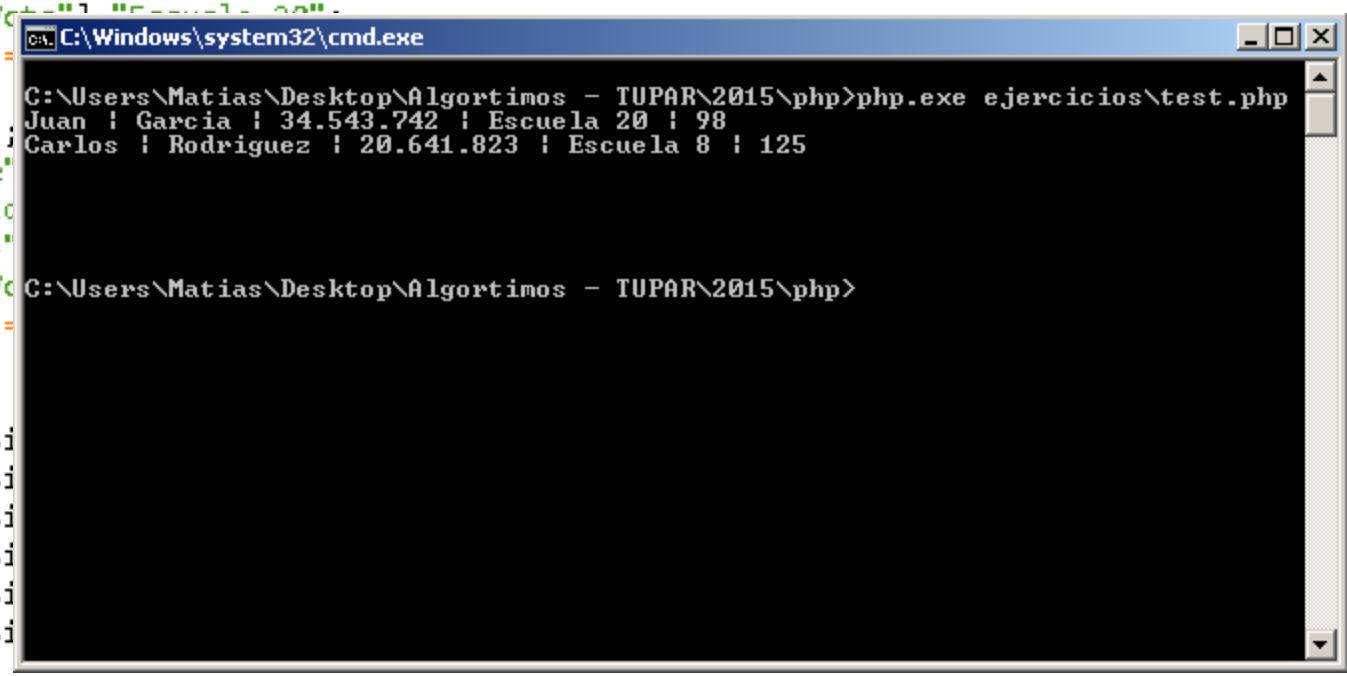
Utilizar arreglos asociativos de php.

```php
$padron=array();

$padron[0]=array();
$padron[0]["nombre"]="Juan";
$padron[0]["apellido"]="Garcia";
$padron[0]["DNI"]="34.543.742";
$padron[0]["lugarVoto"]="Escuela 20";
$padron[0]["mesa"]=98;

$padron[1]=array();
$padron[1]["nombre"]="Carlos";
$padron[1]["apellido"]="Rodriguez";
$padron[1]["DNI"]="20.641.823";
$padron[1]["lugarVoto"]="Escuela 8";
$padron[1]["mesa"]=125;


for ($i=0; $i<N; $i++){
    echo $padron[$i]["nombre"] . " | ";
    echo $padron[$i]["apellido"] . " | ";
    echo $padron[$i]["DNI"] . " | ";
    echo $padron[$i]["lugarVoto"] . " | ";
    echo $padron[$i]["mesa"] . "\n";
}
```

# Posible solución (2)

Utilizar arreglos asociativos de php.

```php
$padron=array();

$padron[0]=array();
$padron[0]["nombre"]="Juan";
$padron[0]["apellido"]="Garcia";
$padron[0]["DNI"]="34.543.742";
$padron[0]["lugarVo
$padron[0]["mesa"]=

$padron[1]=array();
$padron[1]["nombre"
$padron[1]["apellid
$padron[1]["DNI"]="
$padron[1]["lugarVo
$padron[1]["mesa"]=


for ($i=0; $i<N; $i
    echo $padron[$i
    echo $padron[$i
    echo $padron[$i
    echo $padron[$i
    echo $padron[$i
}
```

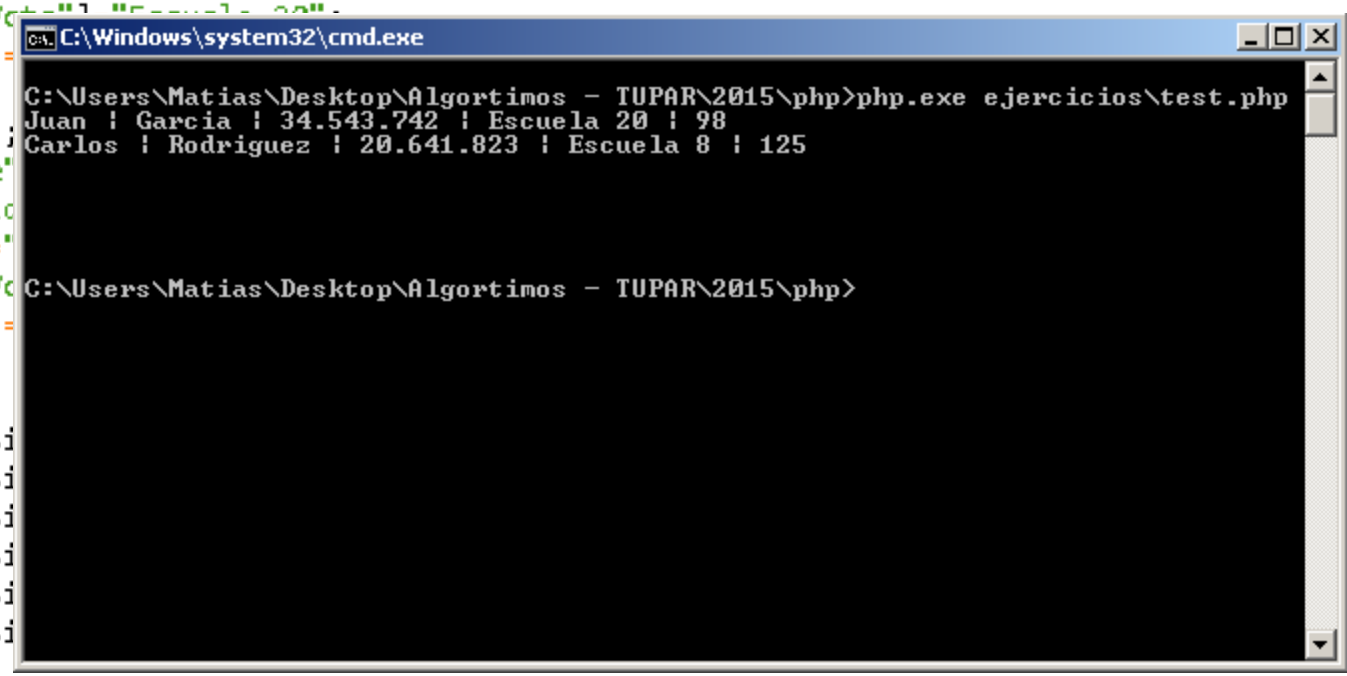


# Posible solución (2)

Hay que manejar estructuras muy grandes. Se pierde encapsulamiento de los datos

Utilizar arreglos asociativos de php.

```php
$padron=array();

$padron[0]=array();
$padron[0]["nombre"]="Juan";
$padron[0]["apellido"]="Garcia";
$padron[0]["DNI"]="34.543.742";
$padron[0]["lugarVo
$padron[0]["mesa"]=

$padron[1]=array();
$padron[1]["nombre"
$padron[1]["apellid
$padron[1]["DNI"]="
$padron[1]["lugarVo
$padron[1]["mesa"]=


for ($i=0; $i<N; $i
    echo $padron[$i
    echo $padron[$i
    echo $padron[$i
    echo $padron[$i
    echo $padron[$i
}
```



# Mejor solución

Definir una estructura que encapsule todos los atributos de una persona (un tipo nuevo de dato).

| | |
|---|---|
| Nombre: | |
| Apellido: | |
| DNI: | |
| LugarDeVoto: | |
| Mesa: | |

Crear “variables” de ese tipo de dato definido y guardarlos en un arreglo.

| | |
|---|---|
| Nombre: | Juan |
| Apellido: | Garcia |
| DNI: | 34.543.742 |
| LugarDeVoto: | Escuela 20 |
| Mesa: | 98 |

# En PHP

Para definir tipos de datos utilizamos la palabra reservada “class”.

Debo darle un nombre al tipo de dato y nombre a las variables internas.

```
class persona{
    public $nombre;
    public $apellido;
    public $DNI;
    public $lugarVoto;
    public $mesa;
}
```

# En PHP

Para definir tipos de datos utilizamos la palabra reservada “class”.

Debo darle un nombre al tipo de dato y nombre a las variables internas.

```
class persona{
    public $nombre;
    public $apellido;
    public $DNI;
    public $lugarVoto;
    public $mesa;
}
```

Utilizamos la palabra reservada “public” para poder acceder a los valores desde fuera de la estructura. Más adelante retomamos esto.

# En PHP

Para crear variables de el tipo de dato nuevo se utilizan punteros.

```
$p=new persona;
```

Para acceder a los campos internos se utiliza el operador ->, indicando “a lo que apunta”

```
$p->nombre="Juan";
$p->apellido="Garcia";
$p->DNI="34.543.742";
$p->lugarVoto="Escuela 20";
$p->mesa=98;
```

# En PHP

```php
class persona{
    public $nombre;
    public $apellido;
    public $DNI;
    public $lugarVoto;
    public $mesa;
}

$p1=new persona;
$p1->nombre="Juan";
$p1->apellido="Garcia";
$p1->DNI="34.543.742";
$p1->lugarVoto="Escuela 20";
$p1->mesa=98;

$p2=new persona;
$p2->nombre="Carlos";
$p2->apellido="Rodriguez";
$p2->DNI="20.641.823";
$p2->lugarVoto="Escuela 8";
$p2->mesa=125;

$padron=array();
$padron[0]=$p1;
$padron[1]=$p2;

for ($i=0; $i<N; $i++){
    echo $padron[$i]->nombre . " | ";
    echo $padron[$i]->apellido . " | ";
    echo $padron[$i]->DNI . " | ";
    echo $padron[$i]->lugarVoto . " | ";
    echo $padron[$i]->mesa. "\n";
}
```

# En PHP

Al estar una persona definida dentro de una estructura es más fácil agregar campos nuevos.

Es más fácil manejar estructuras internas.

Se debe tener cuidado especial al manejo de los punteros.