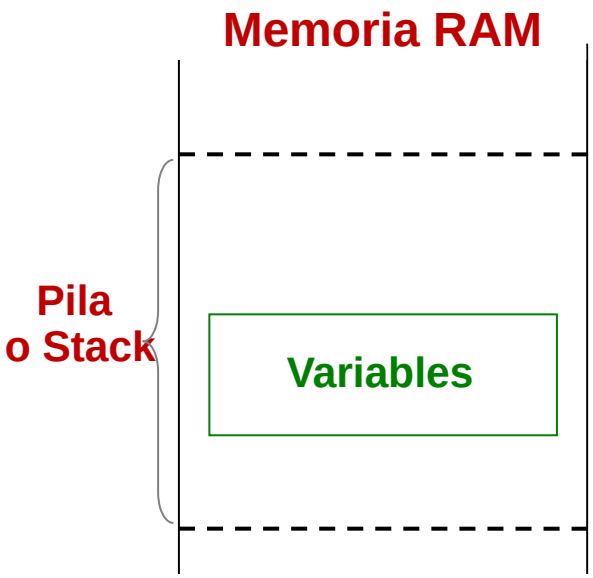
PUNTEROS

Explicación Práctica

Programación I – 2024

Facultad de Informática y Facultad de Ingeniería - UNLP

Variables estáticas



- **Se conoce el tamaño máximo que van a ocupar en memoria.**

Supongamos:

Char	*1byte*
Integer	*2 bytes*
Real	*6 bytes*
Boolean	*1 byte*
String	*cantidad de caracteres + 1*
Registro	*la suma de lo que ocupa c/ campo*
Puntero	*4 bytes*
Vector	*(dim. física) x (tam de dato almacenado en bytes)*

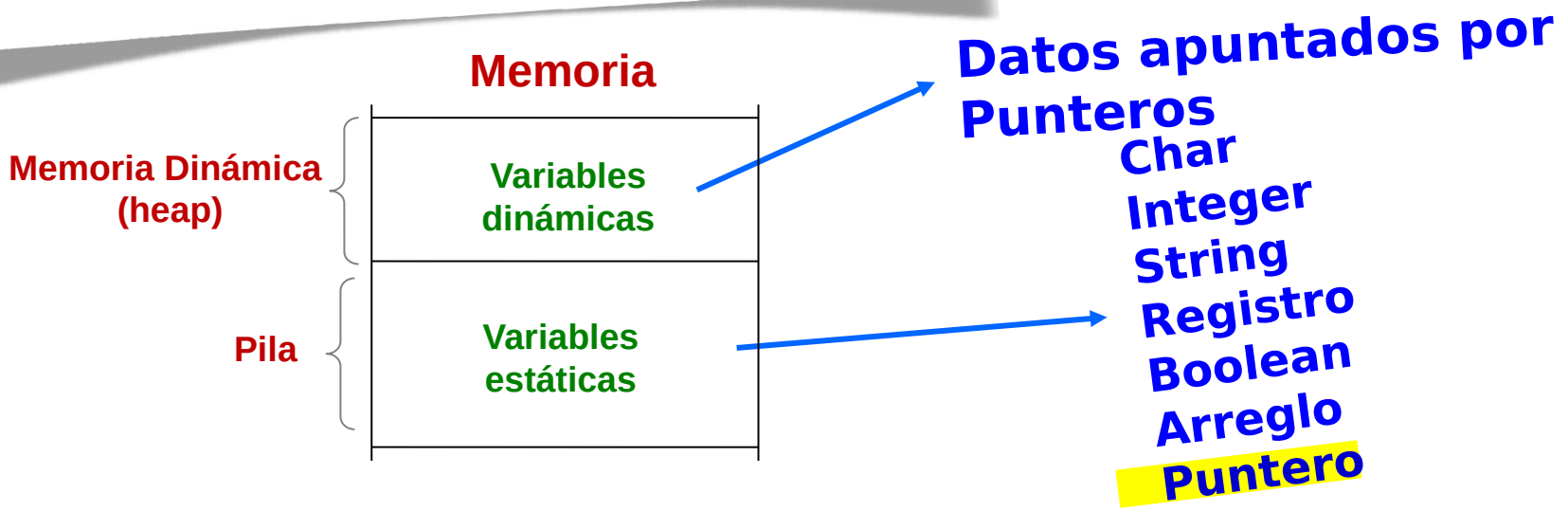
- **declara**

Punteros – Definición

Los **punteros** permiten reservar y liberar memoria en tiempo de ejecución.

Permiten crear estructuras de datos DINÁMICAS cuya cantidad de elementos varíe durante la ejecución (*No se conoce la*

Variable Puntero: almacena la dirección en memoria de otra variable *(llamada variable dinámica)*



Punteros - Uso

Declaración *(ejemplo)*

```
TYPE
  PtroEntero = ^ integer
VAR
  pun, otropun: PtroEntero
```

Valores posibles de un puntero

- NIL
- Dirección de memoria obtenida

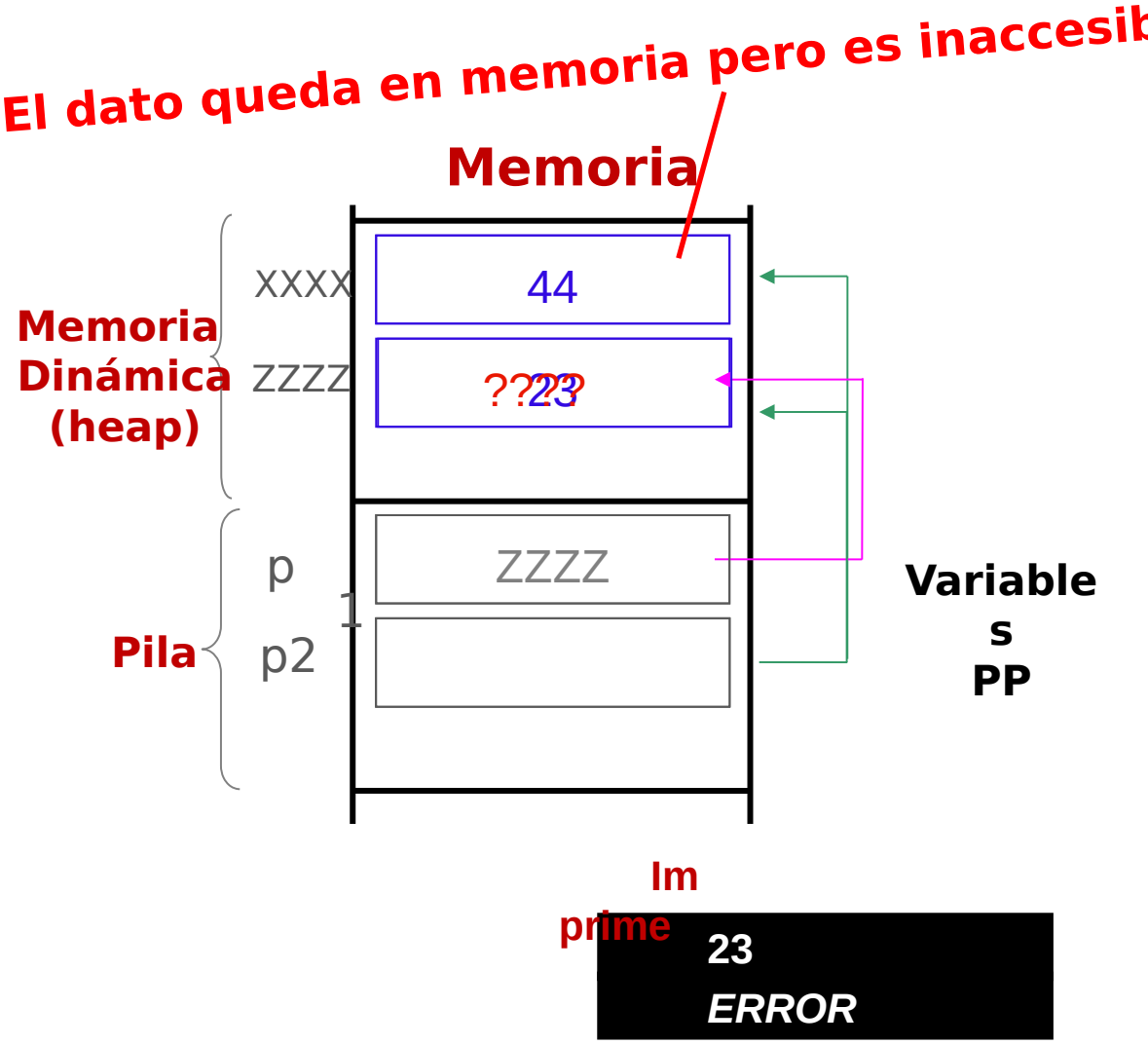
Operaciones

- **new(pun)** Adquiere la memoria necesaria para una *variable dinámica*. El valor de pun es la dirección de memoria en donde se guarda el dato del tipo apuntado.
- **dispose(pun)** Libera la memoria adquirida mediante el *new()*. El puntero queda con valor indefinido.
- **Asignación**
  - Entre punteros de igual tipo. *Ej:* `Pun:= otropun;`
  - Asignación de NIL (nada). *Ej*: `Pun:= NIL`
- **Comparación**
  - Entre punteros de igual tipo. *Ej* : `if(pun = otropun)then …`
  - Comparación con NIL. *Ej* : `if(pun <> NIL)then …`

Acceder al dato apuntado por el puntero `pun^`

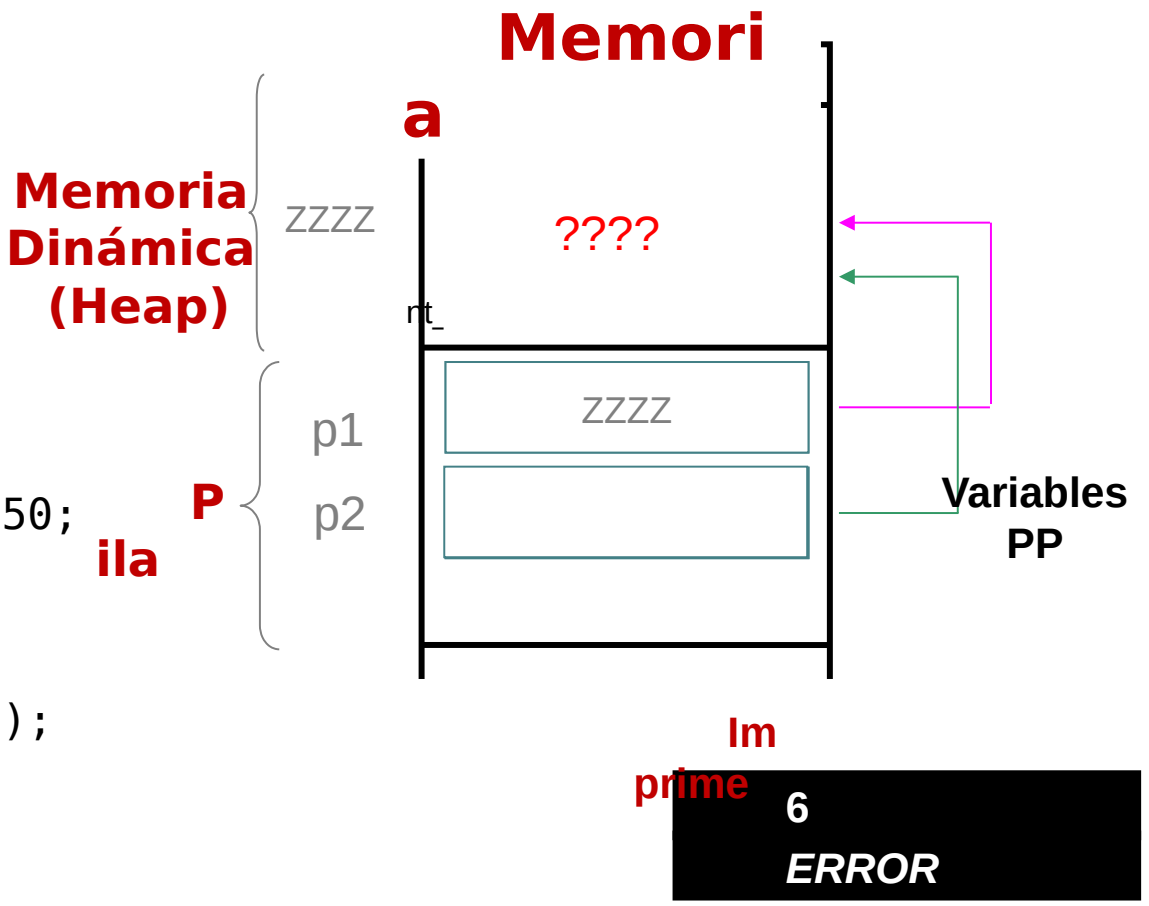
Ejemplo de uso de punteros I

```
program ejemplo;
type
  Ptro= ^integer;
var
  p1, p2: Ptro;
Begin
  new (p1);
  p1^ := 23;
  new (p2);
  p2^ := 44;
  p2 := p1;
  write (p2^);
  dispose (p2);
  write(p1^);
End.
```



Ejemplo de uso de punteros II

```
program ejemplo;
type
  casa = record
    met_cua: real;
    cant_hab: integer;
  end;
  punt_casa = ^casa;
var
  p1, p2: punt_casa;
begin
    new (p1);
    p1^.met_cua := 125.50;
    p1^.cant_hab := 5;
    p2:= p1;
    p2^.cant_hab := 6;
    write (p1^.cant_hab);
    dispose (p2);
    write(p1^.met_cua);
end
```



Punteros como parámetros

p1 recibe una copia de la dirección de **p**. Si modifico el dato apuntado por **p1**, el llamador verá el cambio.

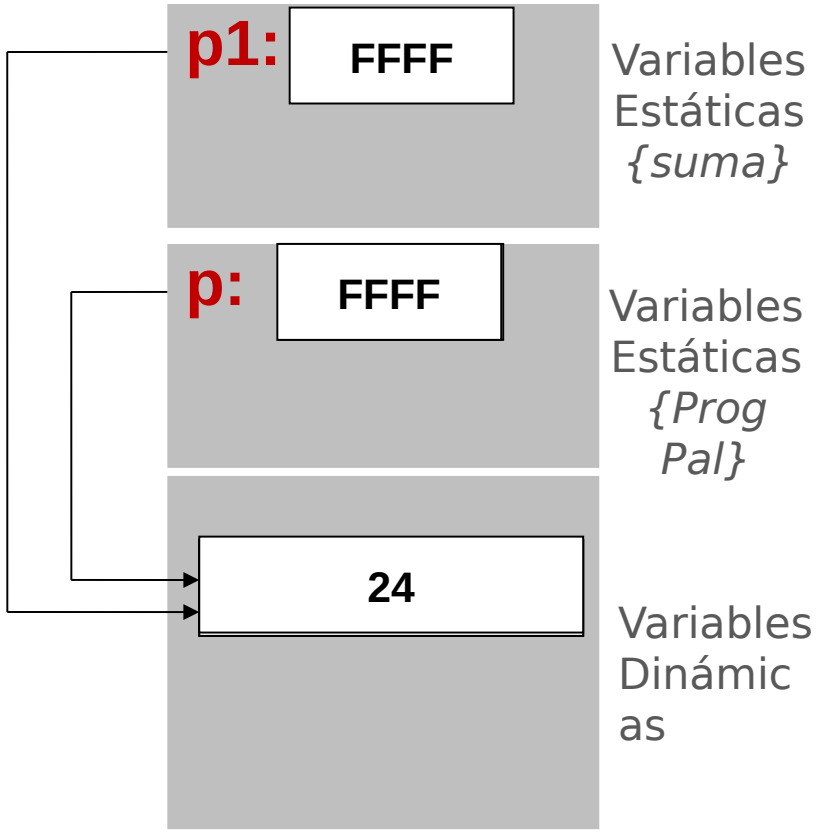
```
Program ejemplo1;
Type
  punt = ^integer;

Procedure suma (p1:punt);
Begin
  p1^ := p1^ + 1;
End;

Var
  p: punt;

Begin
  new (p);
  p^ := 23;
  suma(p);
  write(p^);
End;
```

IMPRIME 24



Punteros como parámetros

```
Program ejemplo2;
Type
  punt = ^integer;

Procedure suma (p1:punt);
Begin
  p1^  := p1^  + 1;
  new(p1);
End;

Var
  p: punt;

Begin
  new (p);
  p^ := 23;
  suma(p);
  write(p^);
End;
```

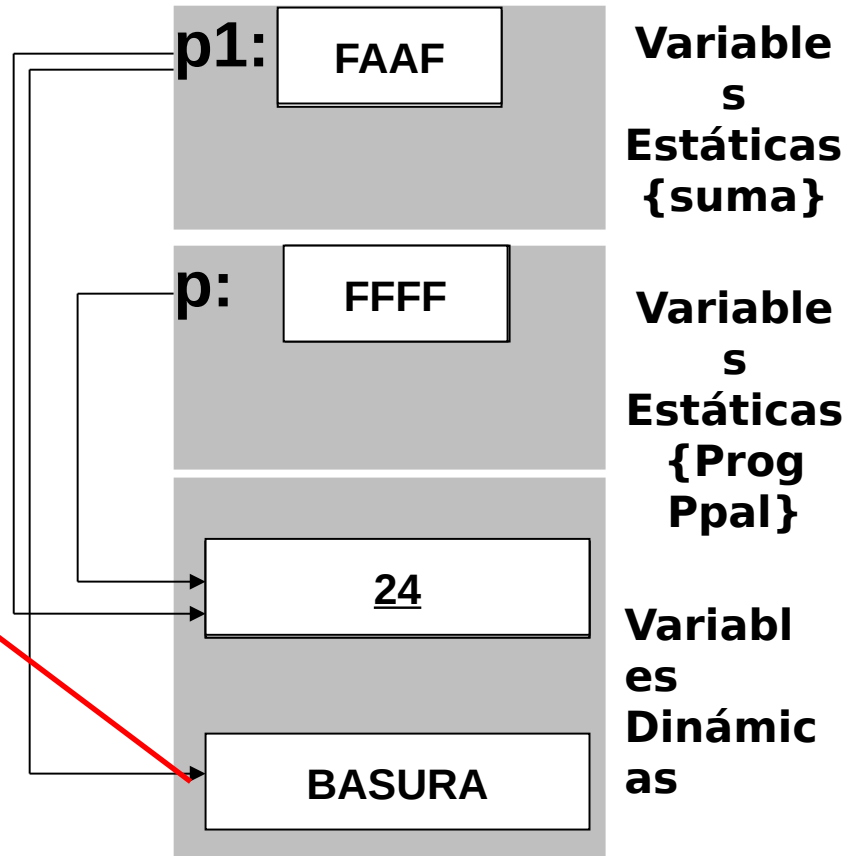
IMPRIME 24

P1 recibe una recibe una copia de la dirección de p
Si modifico la dirección P1, P en el programa ppal
Tiene la misma dirección que tenia antes de la llam

¿Qué imprime si modifico suma?

¿Qué ocurre al reemplazar new(p1) por dispose(p1)?

El dato queda en memoria pero es inaccesible



Punteros como parámetros

P1 recibe la referencia de **p**. Si modifico la dirección **P1**, estoy modificando la dirección de **P** en el programa ppal

```
Type
  punt = ^integer;

Procedure suma (VAR p1:punt);
Begin
  p1^  := p1^  + 1;
  new(p1);
  p1^:= 44;
End

Var
  p: punt;

Begin
  new (p);
  p^ := 23;
  suma(p);
  write(p^);
End;
```

IMPRIME 44

El dato queda en memoria pero es inaccesible

¿Qué imprime si modifico suma?