



# Conceptos de Algoritmos Datos y Programas



# CADP – Temas de la clase de hoy



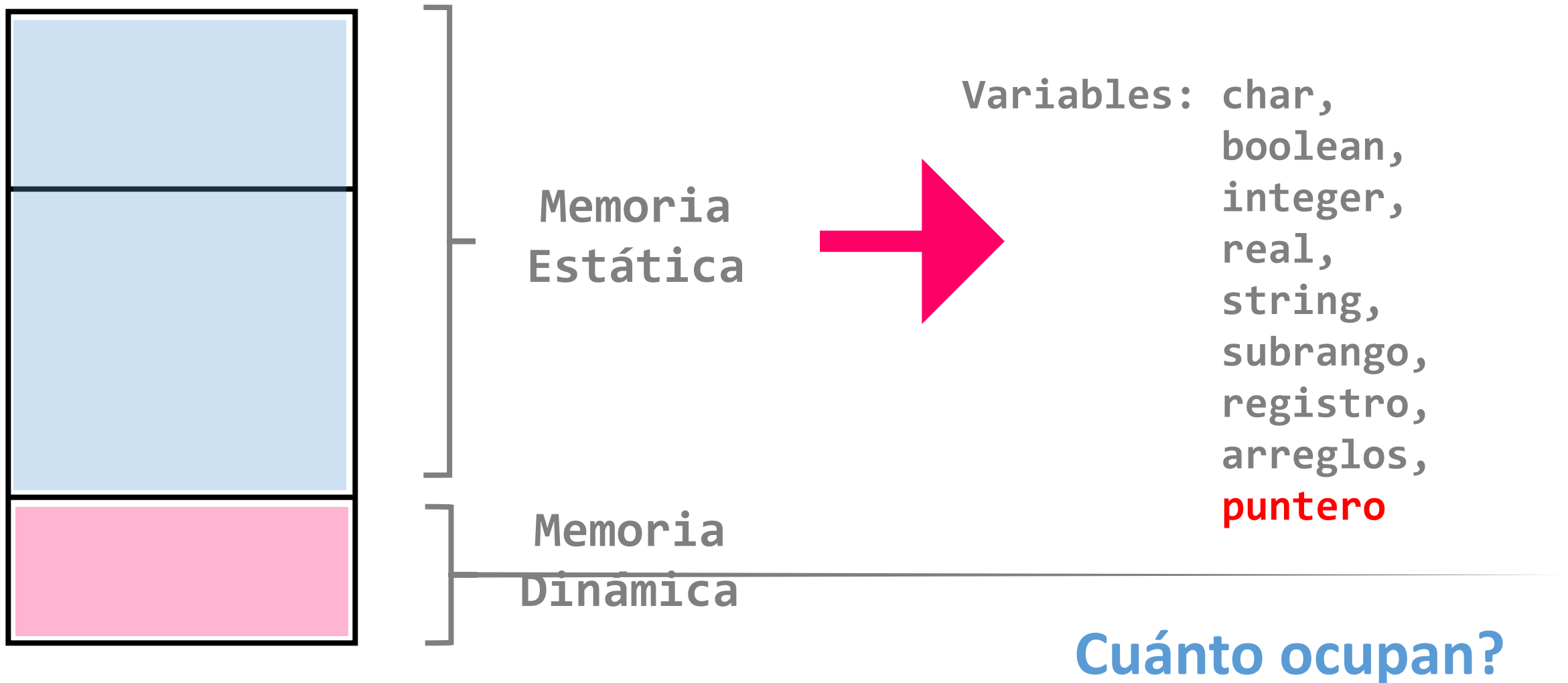
- Alocación estática y alocación dinámica
- Tipo de datos puntero

# CADP – ALOCACION ESTATICA y DINAMICA

## MEMORIA



### GRAFICAMENTE

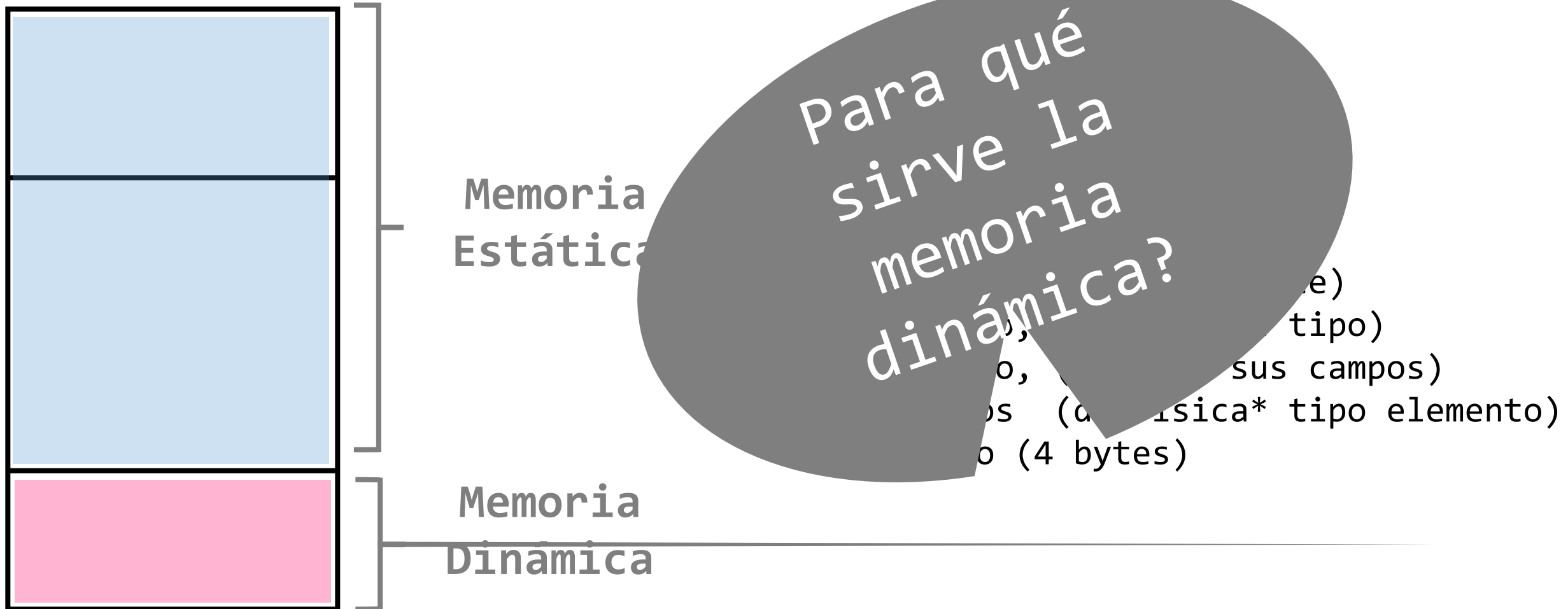


# CADP – ALOCACION ESTATICA y DINAMICA

## MEMORIA

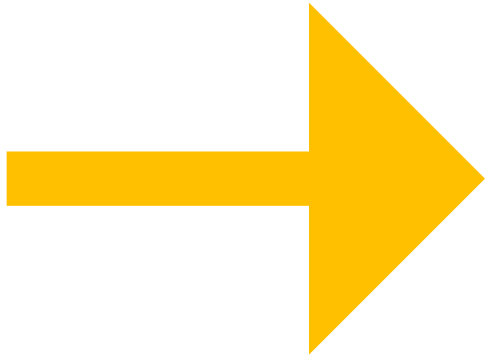


### GRAFICAMENTE





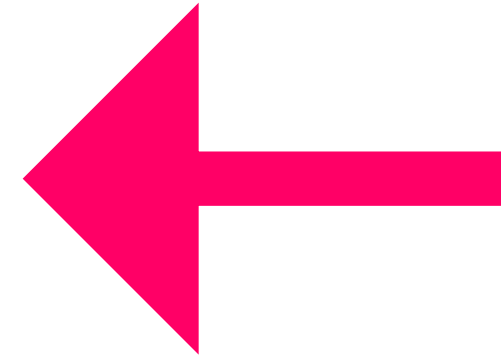
### VARIABLES ESTATICAS



No permiten modificar su tamaño en tiempo de ejecución.

Las variables y tipos reservan memoria en su declaración y se mantienen durante todo el programa. El lenguaje puede validar previo a la ejecución

### VARIABLES DINAMICAS



Permiten modificar en tiempo de ejecución la memoria utilizada.

# CADP – TIPOS DE DATOS

## PUNTERO



**SIMPLE:** aquellos que toman un único valor, en un momento determinado, de todos los permitidos para ese tipo.

**COMPUESTO:** pueden tomar varios valores a la vez que guardan alguna relación lógica entre ellos, bajo un único nombre.

### TIPO DE DATO

#### SIMPLE

#### COMPUESTO

##### DEFINIDO POR EL LENGUAJE

##### DEFINIDO POR EL PROGRAMADOR

##### DEFINIDO POR EL LENGUAJE

##### DEFINIDO POR EL PROGRAMADOR

Integer  
Real  
Char  
Boolean  
Puntero

Subrango

String

Registros  
Arreglos



## PUNTERO

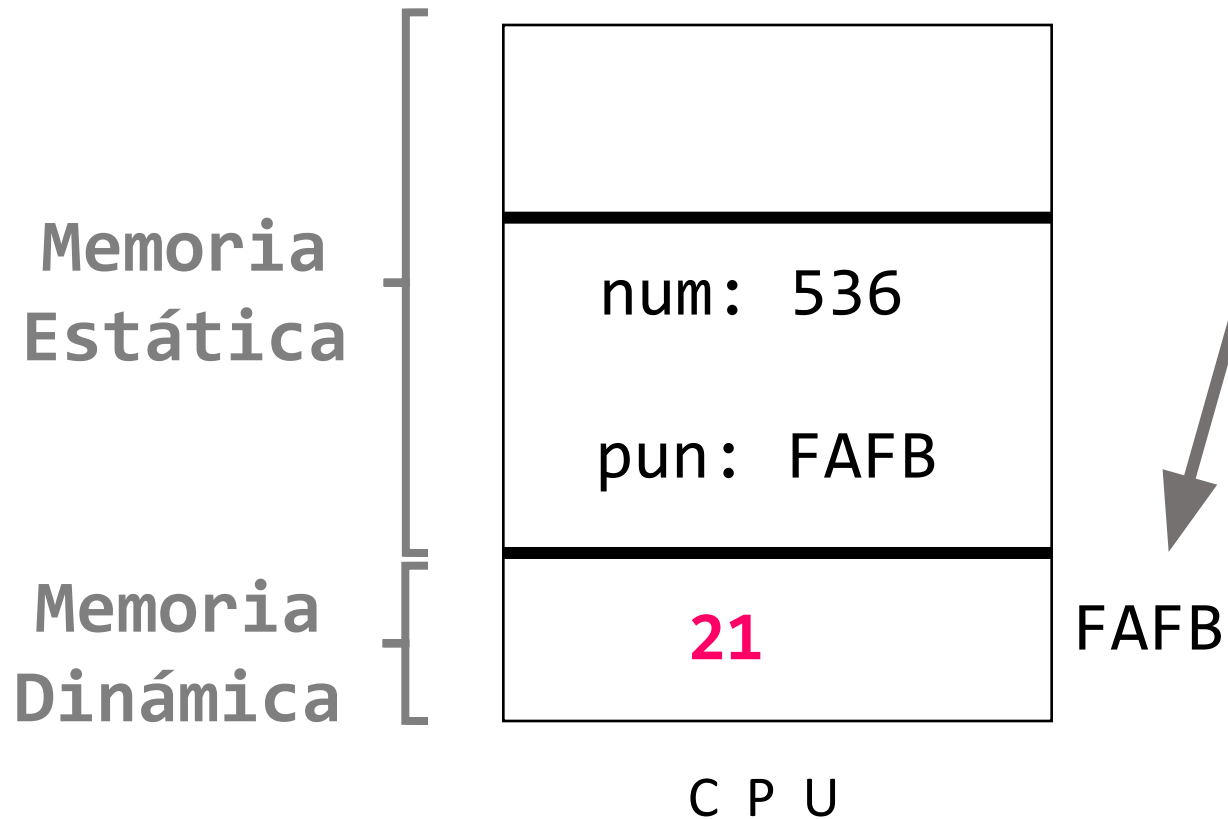
**Es un tipo de variable usada para almacenar una dirección en memoria. En esa dirección de memoria se encuentra el valor que puede ser de cualquiera de los tipos vistos (char, boolean, integer, real, string, registro, arreglo u otro puntero ).**

**Un puntero es un tipo de datos simple.**

**Gráficamente ...**



## PUNTERO



El contenido de la dirección FAFB, depende del tipo que apunte pun el cual se especifica en la declaración

Qué características tienen?





## CARACTERISTICAS

- Es un tipo de dato simple que contiene la dirección donde se encuentra almacenado el dato real.
- Pueden apuntar solamente a direcciones almacenadas en memoria dinámica (heap).
- Cada variable de tipo puntero puede apuntar a un único tipo de dato.
- Una variable de tipo puntero se indica con <sup>^</sup> y ocupa 4 bytes de memoria (stack) para su representación interna en Pascal.

**Cómo se declaran?**



## DECLARACION

type  
    puntero = ^ tipodeDatos;

Nombre  
del tipo

Var  
    pun1:puntero;

Cualquiera de los  
tipos vistos

Ejemplos...



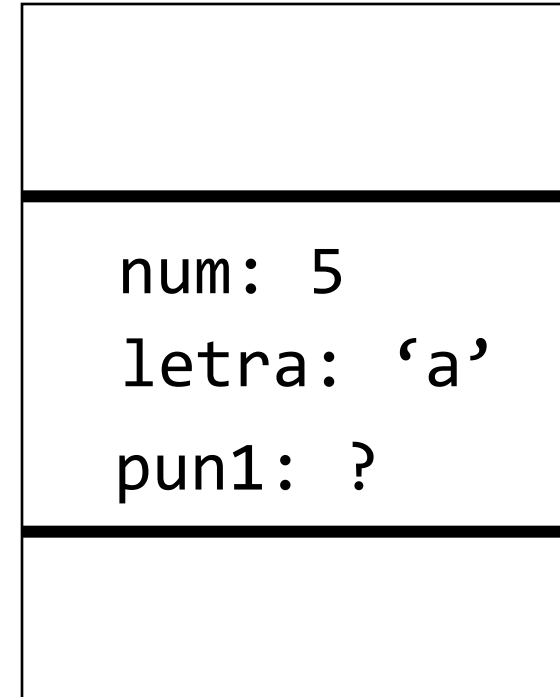
```
type
    puntero = ^ integer;
```

```
Var
    pun1:puntero;
    letra:char;
    num:integer;
```

```
Begin
```

```
...
```

```
End.
```



C P U

Otros ejemplos ....

# CADP – TIPOS DE DATOS

## PUNTERO



### Type

```
TipoCadena = array [1..10] of char;
```

```
PunCadena = ^TipoCadena;
```

```
PunReal = ^real;
```

```
PunString = ^string;
```

```
Datos = record
```

```
    nombre: string[10];
```

```
    apellido: string[10];
```

```
    altura: real;
```

```
end;
```

```
PunDatos = ^datos;
```

```
var
```

```
    pReal: PunReal;
```

```
    t: PunString;
```

```
    r: PunString;
```

```
    puntero: PunCadena;
```

```
    p,q: PunDatos;
```

```
    d:datos;
```

```
begin
```

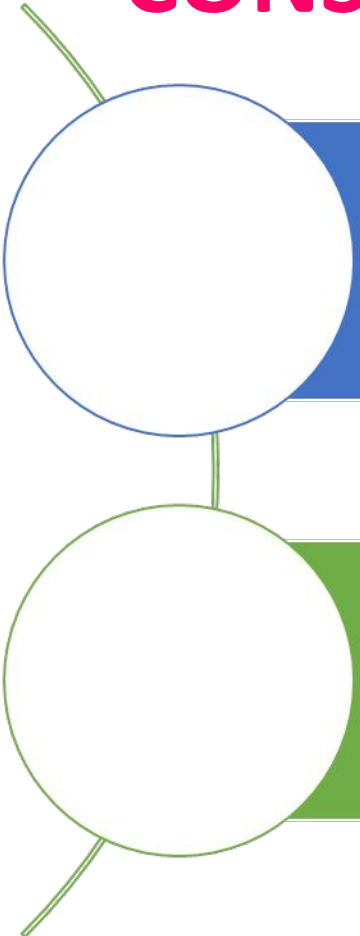
```
    ....
```

```
end.
```

**Cómo trabajamos?**



### CONSIDERACION



Una variable de tipo puntero ocupa una cantidad de memoria fija, independiente del tipo de dato al que apunta (4 bytes).

Una variable de tipo puntero puede reservar y liberar memoria durante la ejecución de un programa.

Un dato referenciado o apuntado, como los ejemplos vistos, no tienen memoria asignada, o lo que es lo mismo no existe inicialmente espacio reservado en memoria para este dato.



## OPERACIONES

Creación de una variable puntero.

Destrucción de una variable puntero.

Asignación entre variables puntero.

Asignación de un valor al contenido de una variable puntero.

Comparación de una variable puntero

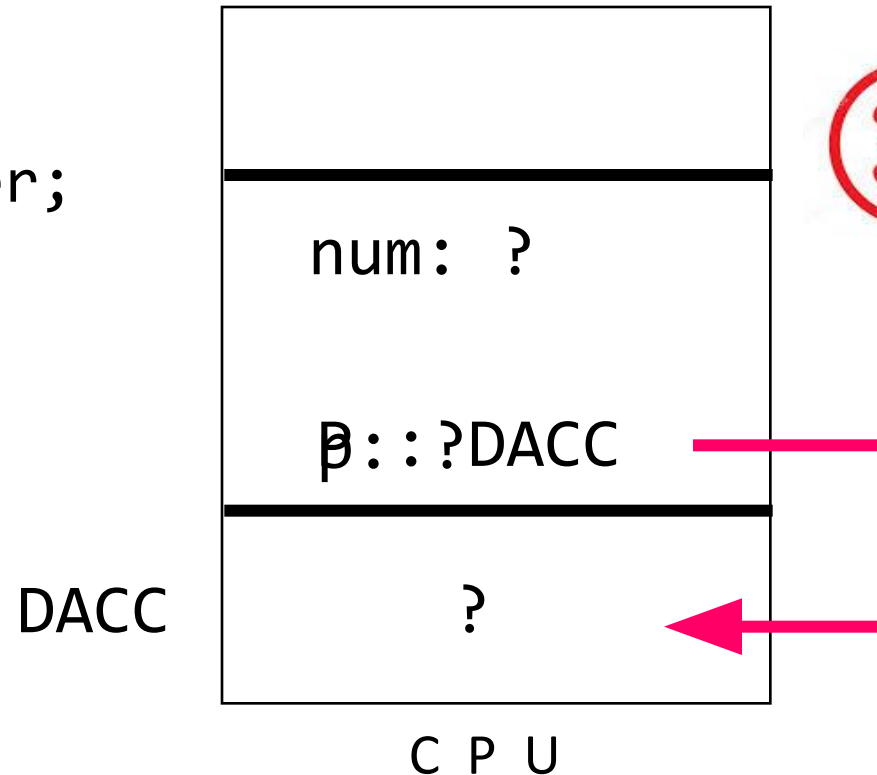




### CREACION

Implica reservar una dirección memoria dinámica libre para poder asignarle contenidos a la dirección que contiene la variable de tipo puntero. **new(variable tipo puntero)**

```
Program uno;  
Type  
  puntero = ^integer;  
Var  
  num:integer;  
  p:puntero;  
  
Begin  
  new (p);  
  ...  
End.
```



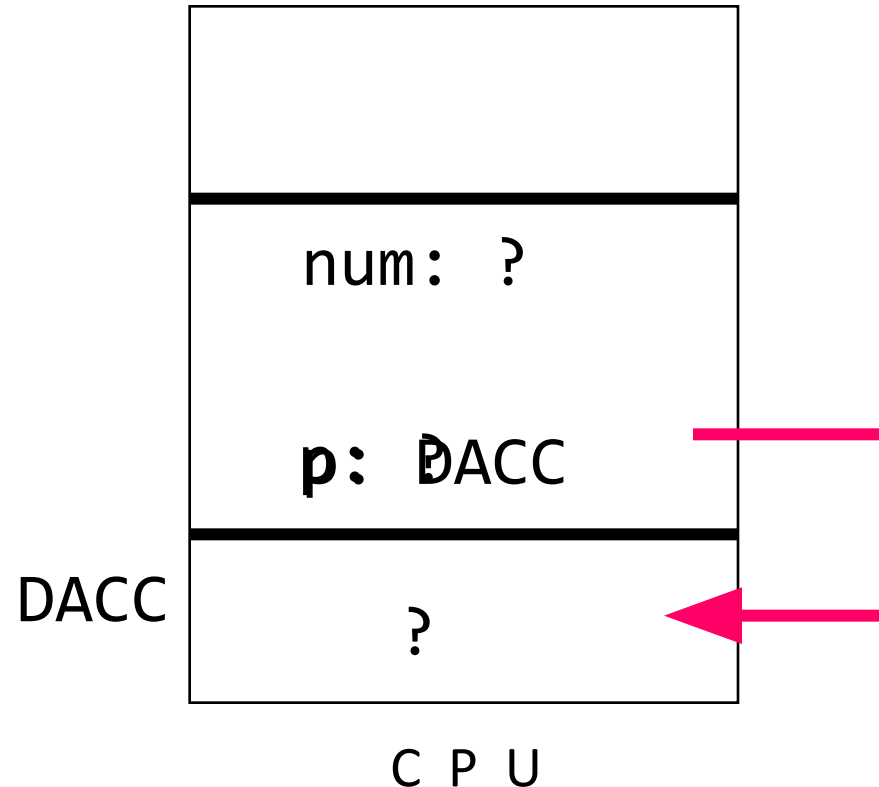
No se puede asignar a un puntero una dirección específica  
(p:= ABCD)



### ELIMINACION

Implica liberar la memoria dinámica que contenía la variable de tipo puntero. **dispose(variable tipo puntero)**

```
Program uno;  
Type  
  puntero = ^integer;  
Var  
  num:integer;  
  p:puntero;  
Begin  
  new (p);  
  dispose (p);  
End.
```



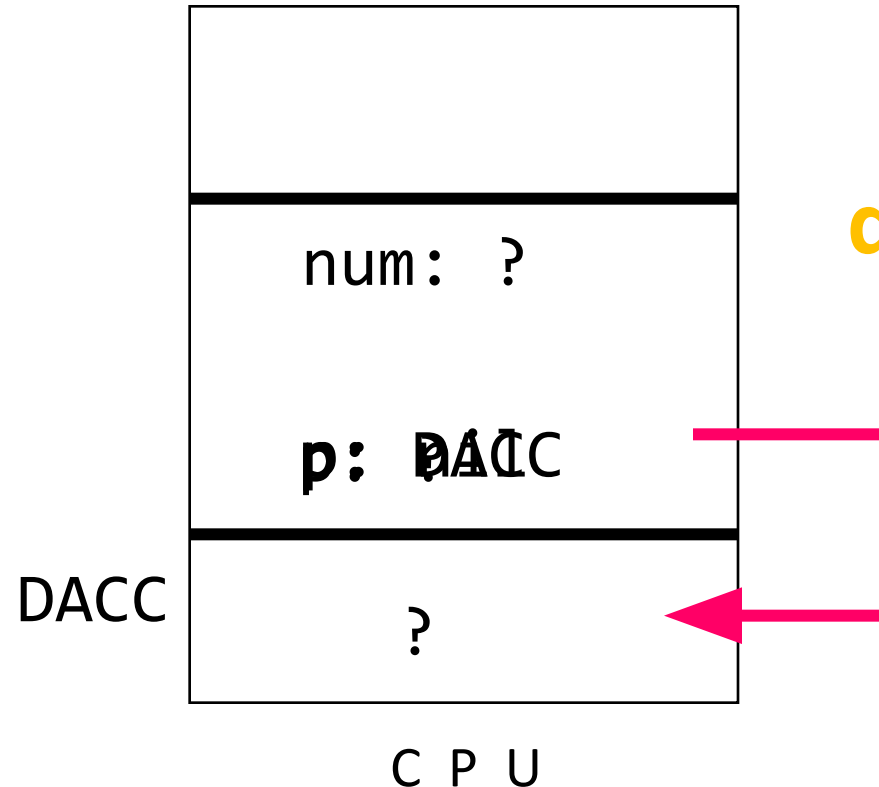




### LIBERACION

Implica cortar el enlace que existe con la memoria dinámica. La misma queda ocupada pero ya no se puede acceder. **nil**

```
Program uno;  
Type  
  puntero = ^integer;  
Var  
  num:integer;  
  p:puntero;  
Begin  
  new (p);  
  p:= nil;  
End.
```



Cuál es la  
diferencia?

# CADP – TIPOS DE PUNTERO

## DISPOSE (p)

Libera la conexión que existe entre la variable y la posición de memoria.

Libera la posición de memoria.

La memoria liberada puede utilizarse en otro momento del programa.



## DISPOSE -NIL



$p := \text{nil}$

Libera la conexión que existe entre la variable y la posición de memoria.

La memoria sigue ocupada.

La memoria no se puede referenciar ni utilizar.

Gráficamente ...?

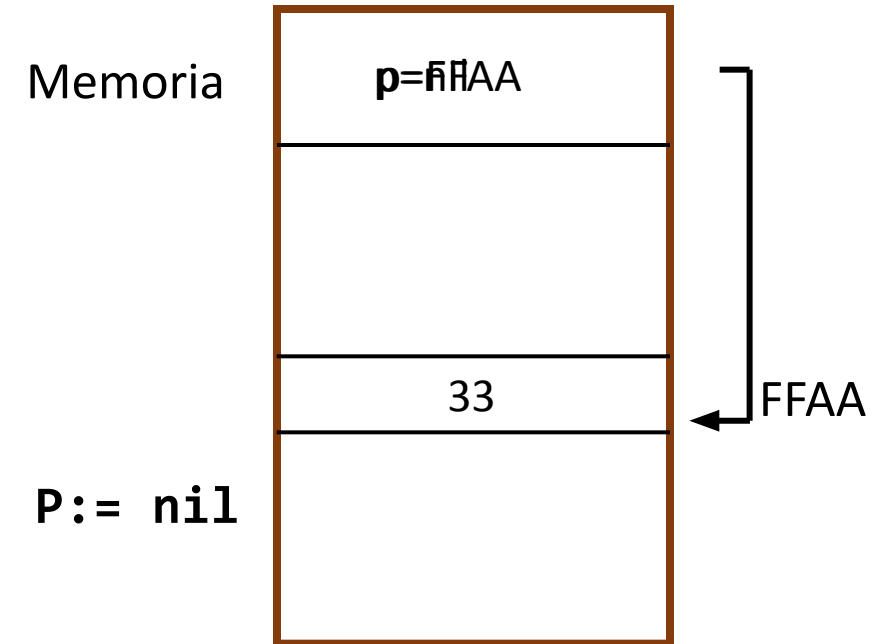
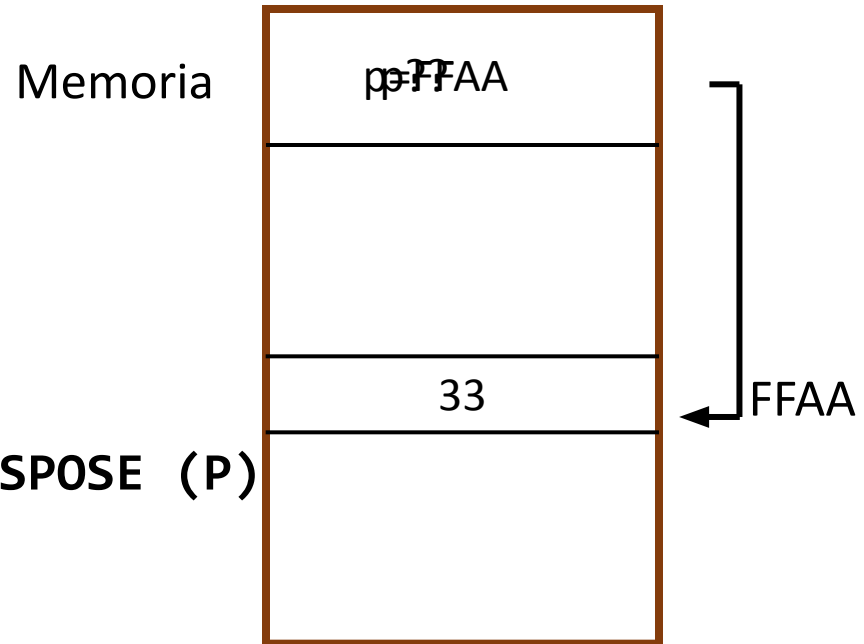
# CADP – TIPOS DE PUNTERO

## DISPOSE -NIL



DISPOSE (p)

$p := \text{nil}$



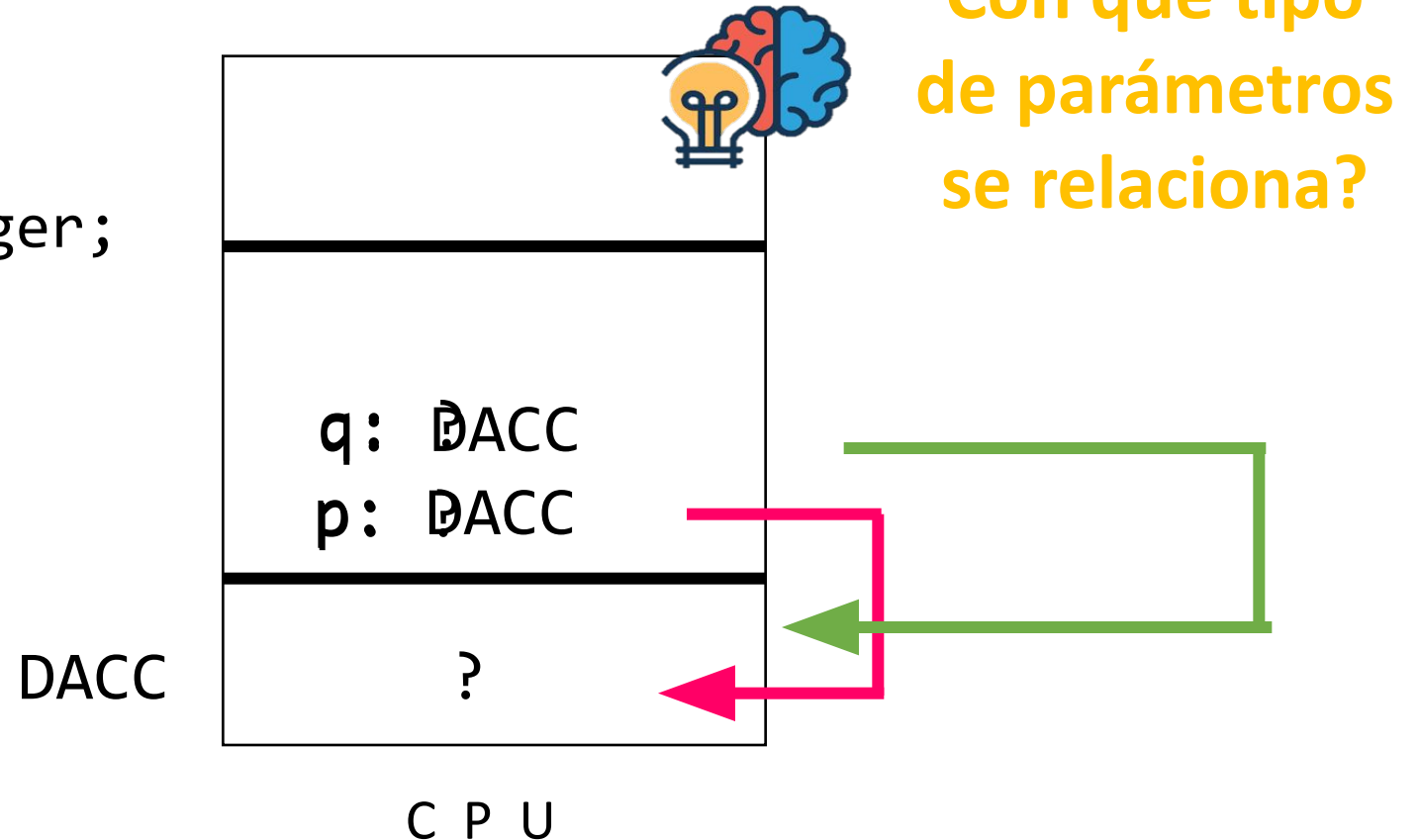
# CADP – TIPOS DE PUNTERO ASIGNACION ENTRE PUNTEROS



## ASIGNACION

Implica asignar la dirección de un puntero a otra variable puntero del mismo tipo.  $:=$

```
Program uno;  
Type  
  puntero = ^integer;  
Var  
  q:puntero;  
  p:puntero;  
Begin  
  new (p);  
  q:=p;  
End.
```



# CADP – TIPOS DE PUNTERO ASIGNACION ENTRE PUNTEROS



## ASIGNACION

Implica asignar la dirección de un puntero a otra variable puntero del mismo tipo. **:=**

```
Program uno;  
Type  
  puntero = ^integer;  
Var  
  q:puntero;  
  p:puntero;
```

**Cómo queda  
la memoria?**

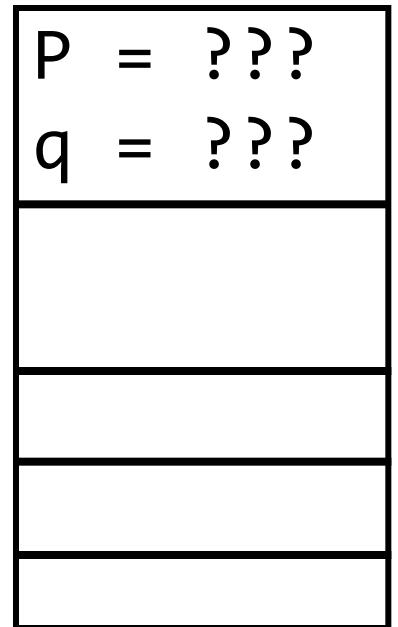
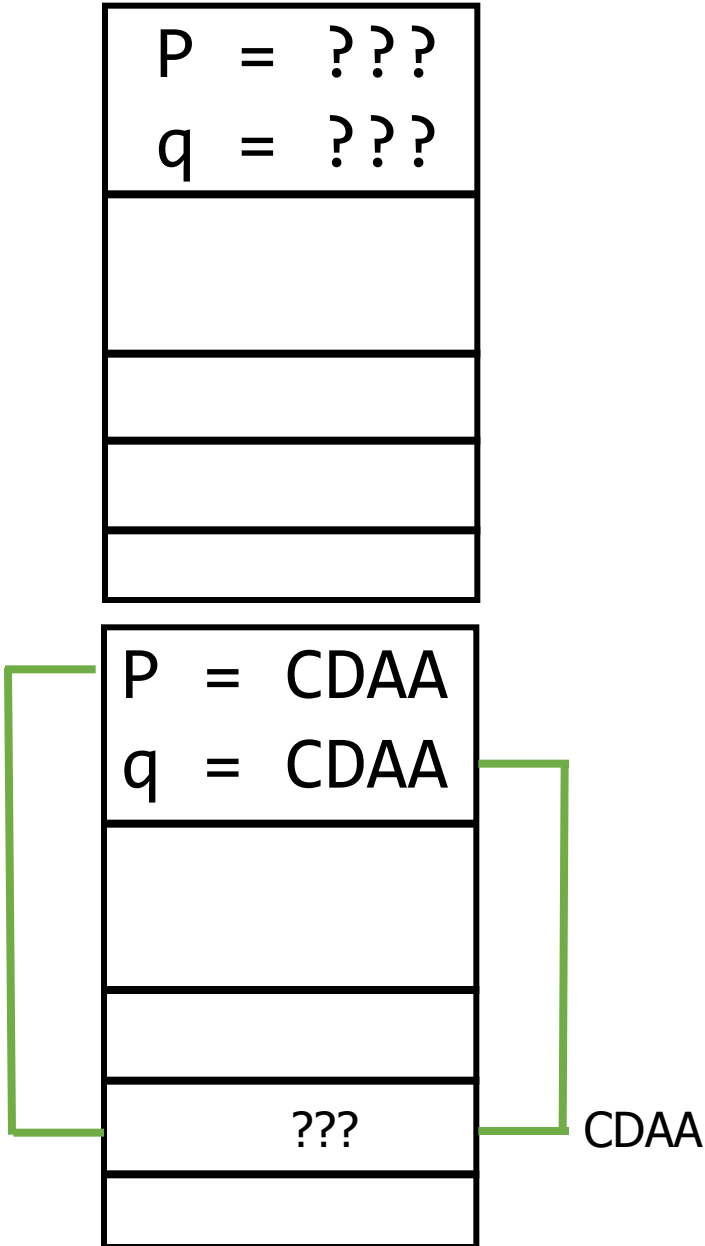
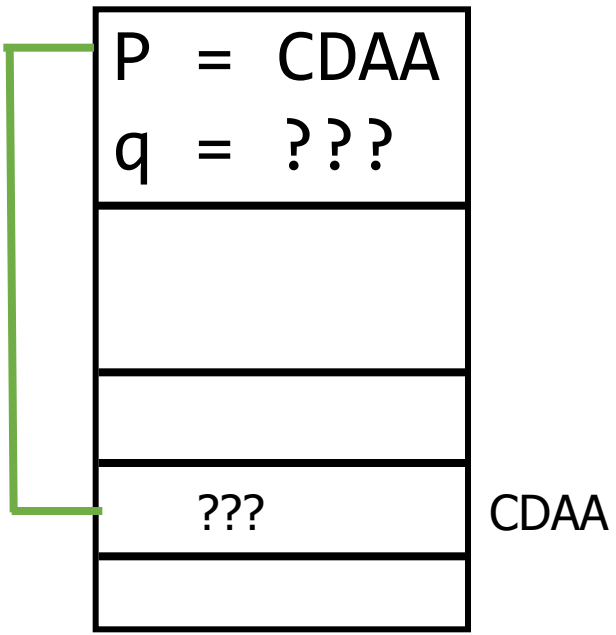
```
Begin  
  new (p);  
  q:=p;  
  dispose (p);  
End.
```

```
Begin  
  new (p);  
  q:=p;  
  p:= nil;  
End.
```

# CADP – TIPOS DE PUNTERO ASIGNACION ENTRE PUNTEROS



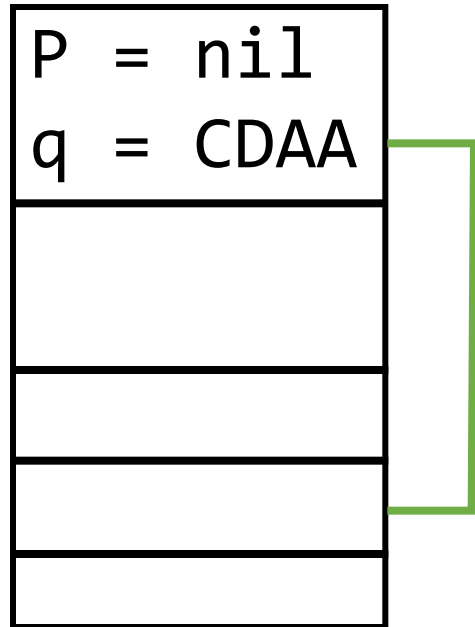
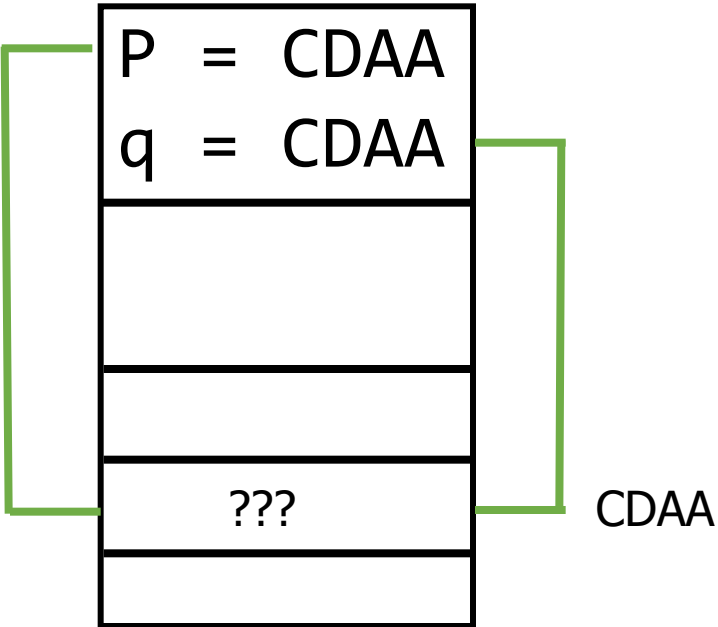
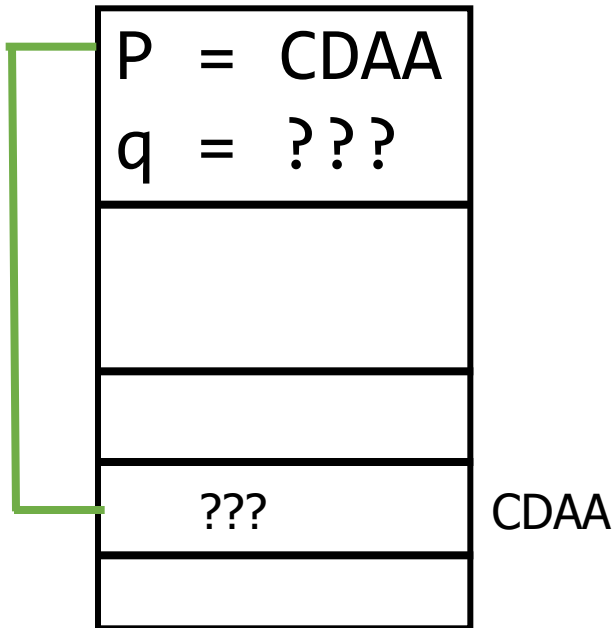
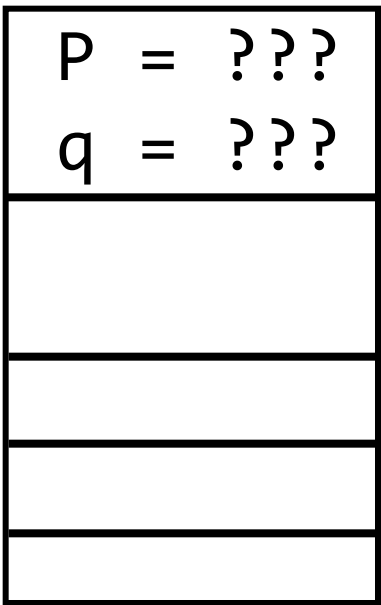
```
Var
  p,q:pun;
Begin
  new (p);
  q:=p;
  dispose(p);
End.
```



# CADP – TIPOS DE PUNTERO ASIGNACION ENTRE PUNTEROS



```
Var
  p,q:pun;
Begin
  new (p);
  q:=p;
  p:= nil;
End.
```





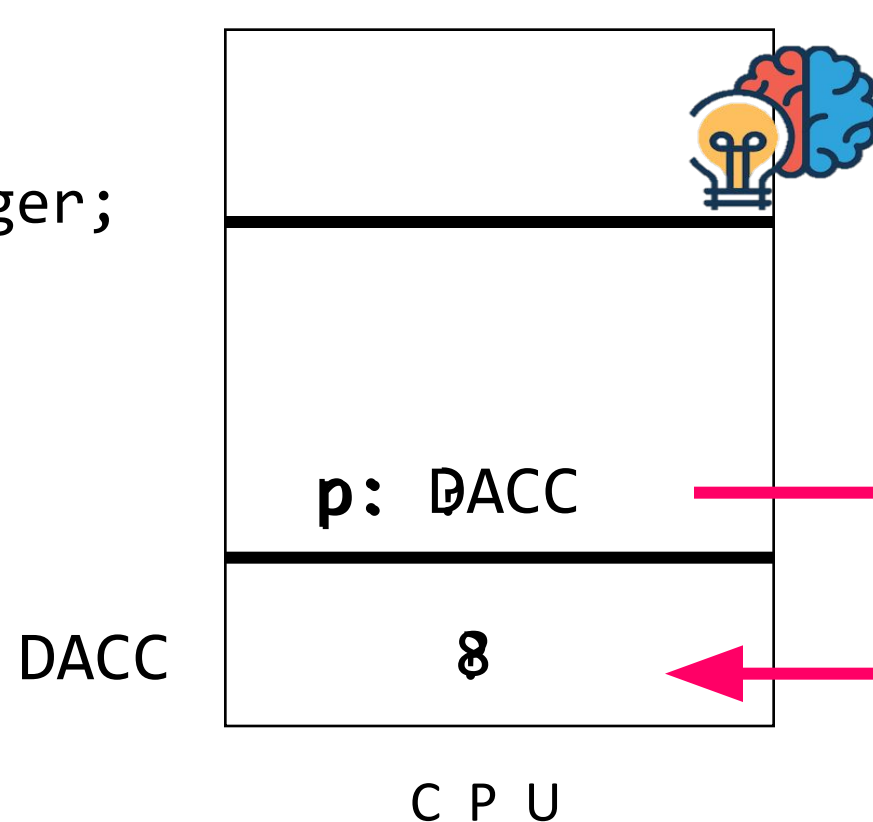
### CONTENIDO

Implica poder acceder al contenido que contiene la dirección de memoria que tiene una variable de tipo puntero. ^

```

Program uno;
Type
    puntero = ^integer;
Var
    p:puntero;

Begin
    new (p);
    p^:=8;
End.
    
```



Qué  
operaciones  
podré hacer?

Ejemplos ...



# CADP – TIPOS DE PUNTERO

## PUNTERO



**EJEMPLOS –**  
**Cómo varía la memoria?**  
**Qué imprime?**

```
Program uno;  
Type  
    puntero = ^integer;  
Var  
    p,q:puntero;  
  
Begin  
    new (p);  
    p^:= 14;  
    write (p^);  
    q:=p;  
    q^:= q^*10;  
    write (p^);  
    write(q^);  
    dispose (q);  
    write (p^);  
    write (q^);  
end.
```

```
Program dos;  
Type  
    puntero = ^integer;  
Var  
    p,q:puntero;  
    num:integer;  
  
Begin  
    num:= 63;  
    new (p);  
    new(q);  
    q^:= num - 10;  
  
    write(q^);  
    write(p^);  
end.
```

# CADP – TIPOS DE PUNTERO

## PUNTERO



**EJEMPLOS –**  
**Cómo varía la memoria?**  
**Qué imprime?**

```
Program tres;  
Type  
    puntero = ^integer;  
Var  
    p,q:puntero;
```

```
Begin  
    new (p);  
    new(q);  
    p:= q;  
    q^:=10;  
  
    write(q^);  
    write(p^);  
end.
```

```
Program cuatro;  
Type  
    puntero = ^integer;  
Var  
    p,q:puntero;
```

```
Begin  
    new (p);  
    p^:= 14;  
    write (p^);  
    q:=p;  
    q^:= q^*10;  
    write (p^);  
    write(q^);  
    q=nil;  
    write (p^);  
    write(q^);  
End.
```



## RECORDAR

- if ( $p = \text{nil}$ ) then, compara si el puntero  $p$  no tiene dirección asignada.
- if ( $p = q$ ) then, compara si los punteros  $p$  y  $q$  apuntan a la misma dirección de memoria.
- if ( $p^\wedge = q^\wedge$ ) then, compara si los punteros  $p$  y  $q$  tienen el mismo contenido.



### RECORDAR

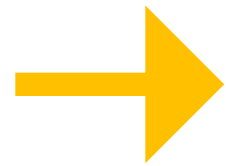
- No se puede hacer **read (p)**, si p es una variable de tipo puntero.
- No se puede hacer **write (p)**, si p es una variable de tipo puntero.
- No se puede asignar una dirección a un puntero de manera manual (**p:= ABCD**).
- No se puede comparar por mayor o menor direcciones de punteros (**p>q**)

**Cómo se calcula la ocupación de memoria?**

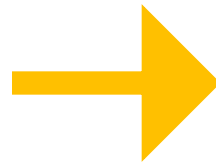


## MEMORIA DE UN PROGRAMA

En rasgos generales la memoria necesaria para la ejecución de un programa puede dividirse en dos.



**MEMORIA ESTATICA:** a modo de simplicidad consideraremos sólo las variables locales y variables globales de programa



**MEMORIA DINAMICA:** a modo de simplicidad consideraremos sólo cuando en la ejecución de un programa se reserva o libera memoria.

*Cómo se calcula  
la ocupación de  
memoria?*

# CADP – TIPOS DE DATOS

## CALCULO DE MEMORIA



### MEMORIA ESTATICA Y DINAMICA

#### Tabla de ocupación:

char, (1 byte)  
boolean, (1 byte)  
integer (4 bytes)  
real, (8 bytes)  
string, (tamaño + 1 byte)  
subrango, (depende el tipo)  
registro, (suma de sus campos)  
arreglos (dimFísica\*tipo elemento)  
puntero (4 bytes)

```
Program uno;  
Type  
    puntero = ^real;  
    puntero2 = ^char;  
    persona = record  
        nombre:string[20];  
        dni:integer;  
    end;
```

```
Var  
    p:puntero;  
    q:puntero2;  
    per: persona;  
    precio:integer;
```

```
Begin  
End.
```



MEMORIA ESTATICA: a modo de simplicidad consideraremos sólo las variables locales y variables globales de programa.

p = 4 bytes  
q = 4 bytes  
per = 21 + 4 = 25 bytes  
precio = 4 bytes

TOTAL MEMORIA ESTATICA = 37 bytes  
TOTAL MEMORIA DINAMICA = 0 bytes

# CADP – TIPOS DE DATOS

## CALCULO DE MEMORIA



### MEMORIA ESTATICA Y DINAMICA

#### Tabla de ocupación:

char, (1 byte)  
boolean, (1 byte)  
integer (4 bytes)  
real, (8 bytes)  
string, (tamaño + 1 byte)  
subrango, (depende el tipo)  
registro, (suma de sus campos)  
arreglos (dimFísica\*tipo elemento)  
puntero (4 bytes)

```
Program dos;  
Type  
    puntero = ^real;  
    puntero2 = ^char;  
    persona = record  
        nombre:string[20];  
        dni:integer;  
    end;
```

```
Var  
    p:puntero;  
    q:puntero2;  
    per:persona;  
    precio:real;  
Begin  
    new (p);  
End.
```



MEMORIA DINAMICA: a modo de simplicidad consideraremos sólo cuando en la ejecución de un programa se reserva o libera memoria.

p = se reserva espacio para el contenido de p que es real entonces 8 bytes

TOTAL MEMORIA DINAMICA = 8 bytes  
TOTAL MEMORIA ESTATICA = 37 bytes



## MEMORIA ESTATICA Y DINAMICA

### Tabla de ocupación:

char,	(1 byte)
boolean,	(1 byte)
integer	(4 bytes)
real,	(8 bytes)
string,	(tamaño + 1 byte)
subrango,	(depende el tipo)
registro,	(suma de sus campos)
arreglos	(dimFísica*tipo elemento)
puntero	(4 bytes)

```
Program tres;  
Type  
    puntero = ^real;  
    puntero2 = ^char;  
    persona = record  
        nombre:string[20];  
        dni:integer;  
    end;
```

```
Var  
    p:puntero;  
    q:puntero2;  
    per:persona;  
    precio:real;  
Begin  
    new (p);  
    new (q);  
End.
```



new (p) = se reserva espacio para el contenido de p que es real entonces 8 bytes  
new (q) = se reserva espacio para el contenido de q que es char entonces 1 byte

TOTAL MEMORIA DINAMICA = 9 bytes  
TOTAL MEMORIA ESTATICA = 37 bytes



# CADP – TIPOS DE DATOS

## CALCULO DE MEMORIA



### MEMORIA ESTATICA Y DINAMICA

#### Tabla de ocupación:

char,	(1 byte)
boolean,	(1 byte)
integer	(4 bytes)
real,	(8 bytes)
string,	(tamaño + 1 byte)
subrango,	(depende el tipo)
registro,	(suma de sus campos)
arreglos	(dimFísica*tipo elemento)
puntero	(4 bytes)

```
Program cuatro;  
Type  
    puntero = ^real;  
    persona = record  
        nombre:string[20];  
        dni:integer;  
    end;  
end;
```

```
Var  
    p,q:puntero;  
    per:persona;  
    precio:real;
```

```
Begin  
    new (p);  
    read (per.dni);  
End.
```



new (p) = se reserva espacio para el contenido de p que es real entonces 8 bytes

El read no genera mas ocupación de mas memoria estática

TOTAL MEMORIA DINAMICA = 8 bytes  
TOTAL MEMORIA ESTATICA = 37 bytes

# CADP – TIPOS DE DATOS



**Cuánta memoria estática y dinámica ocupa el siguiente programa?**

## Tabla de ocupación:

char,	(1 byte)
boolean,	(1 byte)
integer	(4 bytes)
real,	(8 bytes)
string,	(tamaño + 1 byte)
subrango,	(depende el tipo)
registro,	(suma de sus campos)
arreglos	(dimFísica*tipo elemento)
puntero	(4 bytes)

# CALCULO DE MEMORIA



Program cinco;

Type

```
puntero = ^real;  
persona = record  
  nombre:string[20];  
  dni:integer;  
end;
```

Var

```
p,q:puntero;  
precio:real;
```

Begin

```
new (p);  
new (q);  
read (precio);  
read (p^);  
dispose (q);
```

End.



**Cuánta memoria estática y dinámica ocupa el siguiente programa?**

Type

```
punreal = ^real;
```

```
pun1 = array[1..2500] of real;
```

```
pun2 = array[1..2500] of punreal;
```

Var

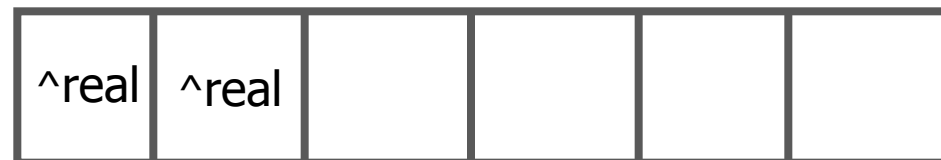
```
p1: pun1;
```



1

2500

```
pa: pun2;
```



1

2500



**Cuánta memoria estática y dinámica ocupa el siguiente programa?**

p1: pun1;



1

2500

p1 ocupa =  $2500 * (8)$

p1 ocupa = 20000 bytes

pa: pun2;



1

2500

pa ocupa =  $2500 * (4)$

pa ocupa = 10000 bytes



p1 = 20000 bytes

pa = 10000 bytes

TOTAL MEMORIA DINAMICA = 0 bytes

TOTAL MEMORIA ESTATICA = 30000 bytes



**Cuánta memoria estática y dinámica ocupa el siguiente programa?**

Type

```
punreal = ^real;
```

```
pun1 = array[1..2500] of real;
```

```
pun2 = array[1..2500] of punreal;
```

Var

```
p1: pun1;
```

```
pa: pun2;
```

Begin

```
cargar1 (p1);
```

```
cargar2 (pa);
```

End.

**Cargar p1 y pa de manera completa.**

**Después que el programa ejecuta la carga cuanto memoria estática y dinámica ocupa el programa?**

# CADP – Temas de la clase de hoy



- Tipo de datos lista

- Características

- Operaciones

# CADP – Temas de la clase de hoy

**SIMPLE:** aquellos que toman un único valor, en un momento determinado, de todos los permitidos para ese tipo.

**COMPUESTO:** pueden tomar varios valores a la vez que guardan alguna relación lógica entre ellos, bajo un único nombre.



## TIPO DE DATO

### SIMPLE

### COMPUESTO

#### DEFINIDO POR EL LENGUAJE

#### DEFINIDO POR EL PROGRAMADOR

#### DEFINIDO POR EL LENGUAJE

#### DEFINIDO POR EL PROGRAMADOR

Integer  
Real  
Char  
Boolean  
Puntero

Subrango

String

Registros  
Arreglos

Lista



Realizar un programa que lea números que representan edades y luego de realizar la lectura se quiere informar cuántos veces apareció la edad más grande. La lectura de edades finaliza cuando se lee la edad -1.

### SOLUCION ALUMNO 1:

Leo una edad, sino es -1, la almaceno en un vector y voy contando cuantos elementos voy agregando (dimensión lógica).

Después de leer las edades recorro el vector y calculo el máximo.

Luego de calcular el máximo recorro el vector y cuento cuantas veces se leyó la edad máxima.

### SOLUCION ALUMNO 2:

No sé como se resuelve pero la solución planteada por el alumno 1 está mal.

**Por qué el alumno 2  
tienen razón?**



# CADP – Tipo de Dato

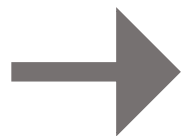
LISTA



Realizar un programa que lea números que representan edades y luego de realizar la lectura se quiere informar cuántos veces apareció la edad más grande. La lectura de edades finaliza cuando se lee la edad -1.



Disponer de alguna **ESTRUCTURA** donde almacenar los números pero que no se necesite especificar un tamaño en el momento de la declaración.

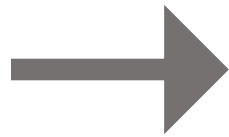


**ESTRUCTURA DINAMICA**



## LISTA

Colección de nodos, donde cada nodo contiene un elemento y en que dirección de memoria se encuentra el siguiente nodo. Cada nodo de la lista se representa con un puntero, que apunta a un dato (elemento de la lista) y a una dirección (donde se ubica el siguiente elemento de la lista). Toda lista tiene un nodo inicial.



Los **nodos** que la componen pueden no ocupar posiciones contiguas de memoria. Es decir pueden aparecer dispersos en la memoria, pero mantienen un orden lógico interno.

**Gráficamente ...**

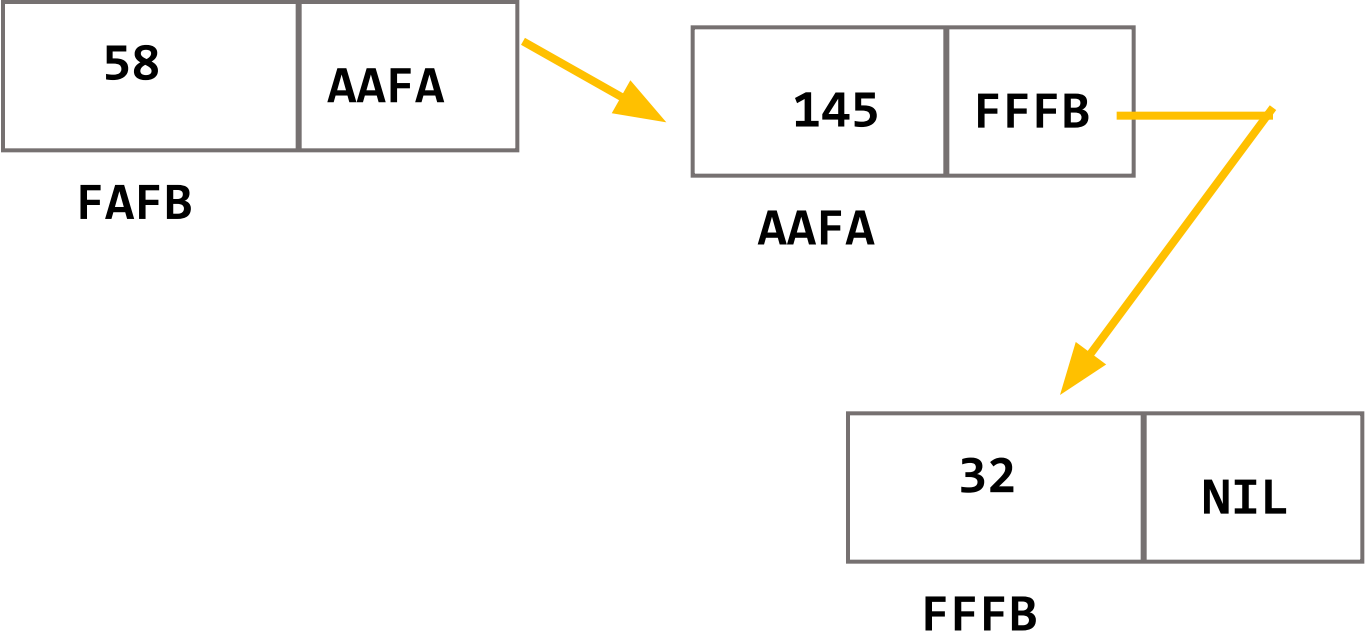
# CADP – Tipo de Dato

LISTA



LISTA

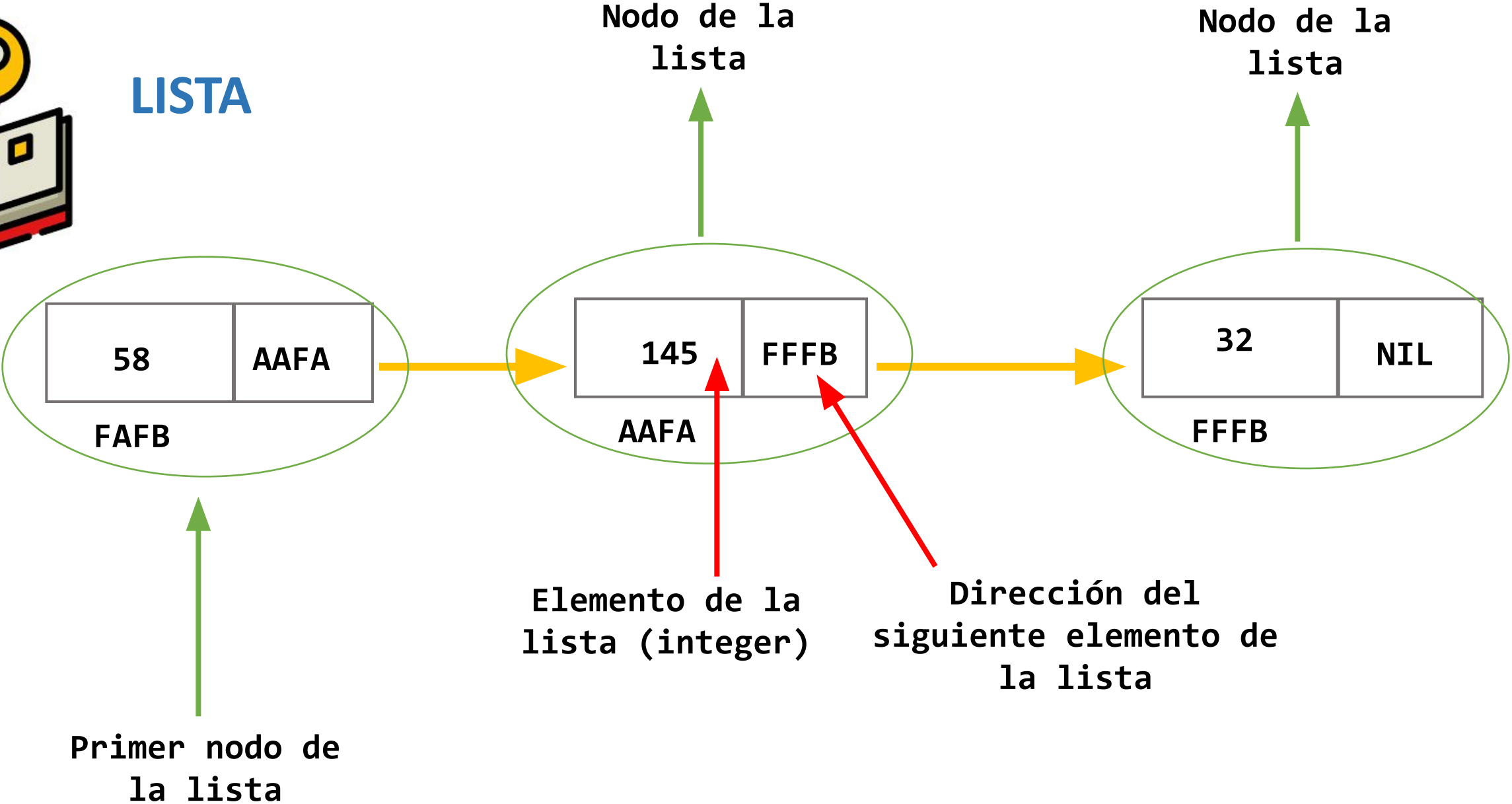
num: 536	
pun: FAFB	
145	FFFB
58	AAFA
32	nil



# CADP – Tipo de Dato



LISTA



LISTA





## CARACTERISTICAS

### Homogénea

- Los elementos son todos del mismo tipo

### Dinámica

- La cantidad de nodos puede variar durante la ejecución

### Lineal

- Cada nodo tiene un único antecesor y sucesor

### Acceso

- El acceso a cada elemento es de manera secuencial

Cómo se declara?



Cada vez que se necesite agregar un nodo se deberá reservar memoria dinámica (new) y cuando se quiera eliminar un nodo se debe liberar la memoria dinámica (dispose) .

# CADP – Tipo de Dato

LISTA



```
Program uno;  
Type nombreTipo= ^nombreElemento;  
  
    nombreElemento= Record  
        elementos : tipoElemento;  
        punteroSig: nombreTipo;  
    end;
```

*Va primero*

*Cualquiera de los tipos vistos*

*Estructura recursiva*

Var

Pri: nombreTipo; {Memoria estática reservada}

Lista de  
enteros?

Lista de  
registros?

# CADP – Tipo de Dato

LISTA



```
Program uno;
```

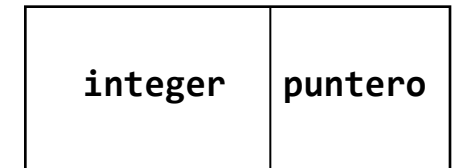
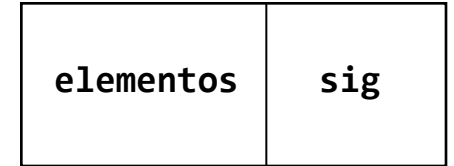
```
Type listaE= ^datosEnteros;
```

```
    datosEnteros= Record
```

```
        elementos : integer;  
        sig: listaE;  
    end;
```

```
Var
```

```
    PriEnteros: listaE; {Memoria estática reservada}
```



# CADP – Tipo de Dato

LISTA



```
Program uno;
```

```
Type
```

```
    alumno = record
```

```
        nombre:string;
```

```
        numeroA:string;
```

```
    end;
```

```
    listaA= ^datosAlumnos;
```

```
    datosAlumnos= record
```

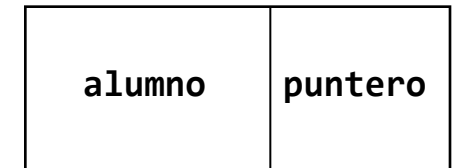
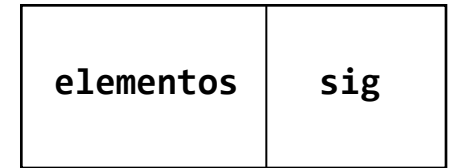
```
        elementos:alumno;
```

```
        sig: listaA;
```

```
    end;
```

```
Var
```

```
    PriAlumnos: listaA; {Memoria estática reservada}
```





# CADP – Tipo de Dato

LISTA



## OPERACIONES

Creación de una lista.

Agregar nodos al comienzo de la lista.

Recorrido de una lista.

Agregar nodos al final de la lista.

Insertar nodos en una lista ordenada

Eliminar nodos de una lista



Trabajaremos con una  
lista de enteros



## CREAR UNA LISTA

Implica marcar que la lista no tiene una dirección inicial de comienzo.

Qué valor se le asigna a un puntero para indicar que no tiene una dirección asignada?

```
Program uno;
```

```
Type listaE= ^datosEnteros;
```

```
    datosEnteros= record  
        elem:integer;  
        sig:listaE;  
    end;
```

```
Var
```

```
    pri: listaE; {Memoria estática reservada}
```

Qué representa pri en nuestro programa?

# CADP – Tipo de Dato - LISTA

CREAR



```
Program uno;
```

```
Type listaE= ^datosEnteros;
```

```
    datosEnteros= record
                        elem:integer;
                        sig:listaE;
                    end;
```

```
Var
```

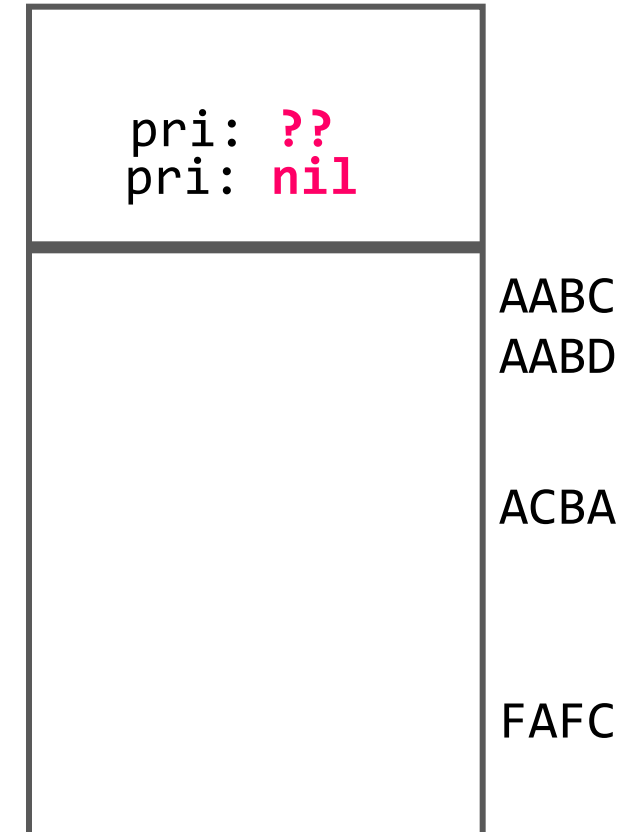
```
    pri: listaE; {Memoria estática reservada}
```

```
Begin
```

```
    pri:= nil;
```

```
End.
```

*Por qué no se hace  
new (pri)?*



*Se puede modularizar  
el crear?*

# CADP – Tipo de Dato - LISTA

CREAR



```
Program uno;
```

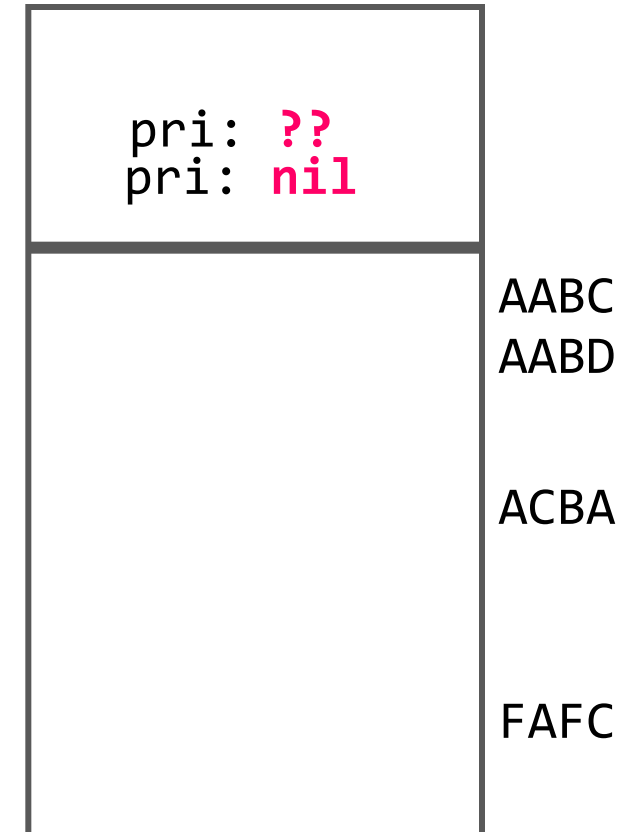
```
Type listaE= ^datosEnteros;
```

```
    datosEnteros= record
                        elem:integer;
                        sig:listaE;
                    end;
```

```
Procedure crear (var p: listaE);
begin
    p:= nil;
end;
```

```
Var
    pri: listaE;
```

```
Begin
    crear (pri);
End.
```





## RECORRER UNA LISTA

**Implica posicionarse al comienzo de la lista y a partir de allí ir “pasando” por cada elemento de la misma hasta llegar al final.**

```
Program uno;
```

```
Type listaE= ^datosEnteros;
```

```
    datosEnteros= record  
        elem:integer;  
        sig:listaE;  
    end;
```

```
Var  
    pri: listaE;
```

# CADP – Tipo de Dato - LISTA

## RECORRER UNA LISTA



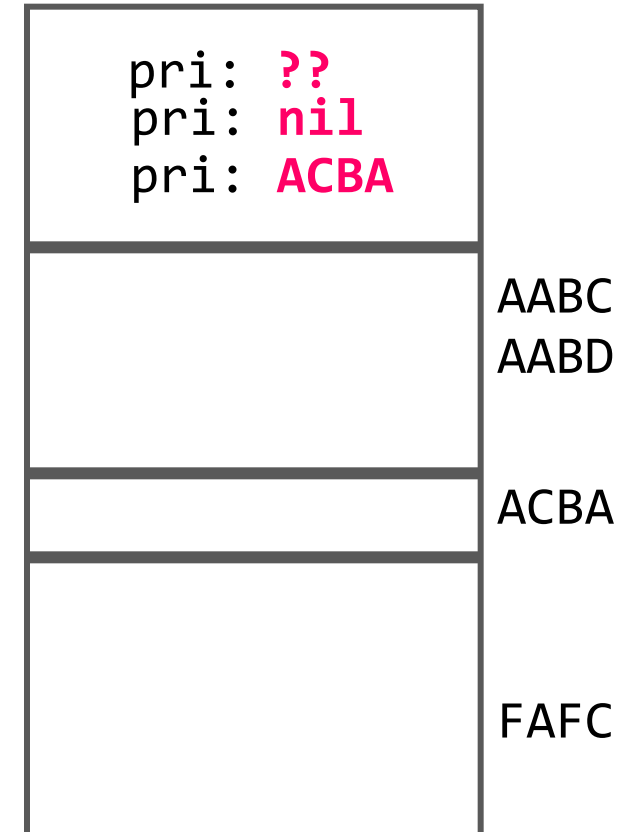
```
Program uno;
```

```
Type listaE= ^datosEnteros;
```

```
    datosEnteros= record
                    elem:integer;
                    sig:listaE;
    end;
```

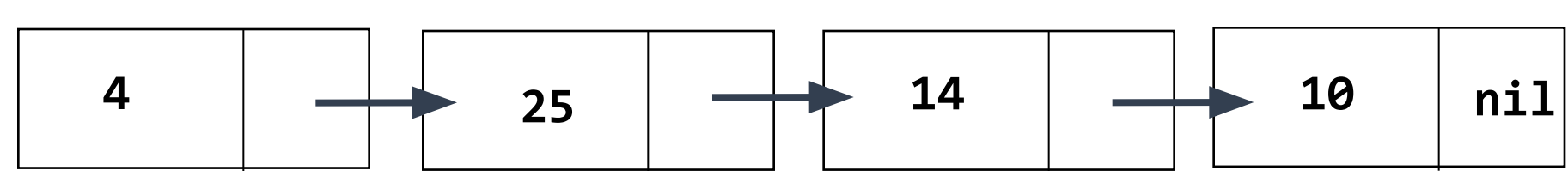
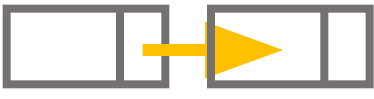
```
Var
    pri: listaE;
```

```
Begin
    crear (pri);
    cargarLista (pri); //Lo implementaremos más adelante
    recorrerLista (pri);
End.
```



# CADP – Tipo de Dato - LISTA

## RECORRER UNA LISTA



PI

aux

4  
25  
14  
10



## RECORRER UNA LISTA

Inicializo una variable **auxiliar** con la dirección del puntero inicial

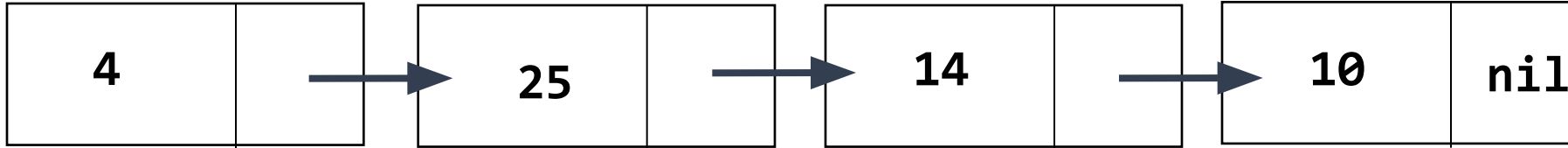
mientras (no sea el final de la lista)

proceso el elemento (por ejemplo imprimo)  
avanzo al siguiente elemento de **auxiliar**



# CADP – Tipo de Dato - LISTA

## RECORRER UNA LISTA



4  
25  
14

PI

aux

```
procedure recorrerLista (pI: listaE);  
Var  
    aux: listaE;  
  
begin  
    aux := pI;  
    while (aux^.sig <> nil) do  
        begin  
            write (aux^.elem);  
            aux := aux^.sig;  
        end;  
    end;
```

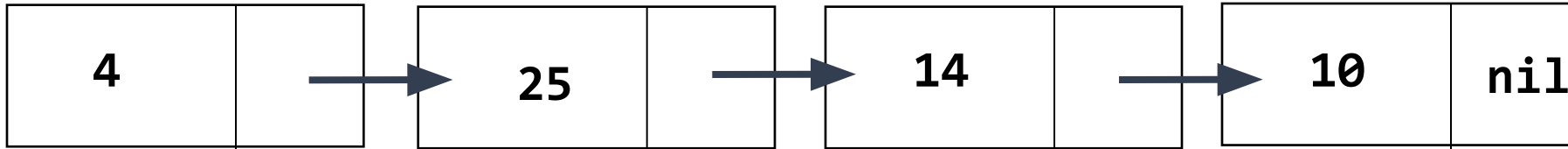
Funciona?

Funciona si la lista que  
recibo esta vacía (nil)?

Funciona si la lista tiene  
elementos?

# CADP – Tipo de Dato - LISTA

## RECORRER UNA LISTA



Qué cambios debo hacer si quiero que el procedimiento devuelva la suma de los elementos?



PI

aux

```
procedure recorrerLista (pI: listaE);
Var
  aux: listaE;
begin
  aux := pI;
  while (aux <> nil) do
  begin
    write (aux^.elem);
    aux := aux^.sig;
  end;
end;
```

Puedo no utilizar  
aux?

```
procedure recorrerLista (pI: listaE);
begin
  while (pI <> nil) do
  begin
    write (pI^.elem);
    pI := pI^.sig;
  end;
end;
```